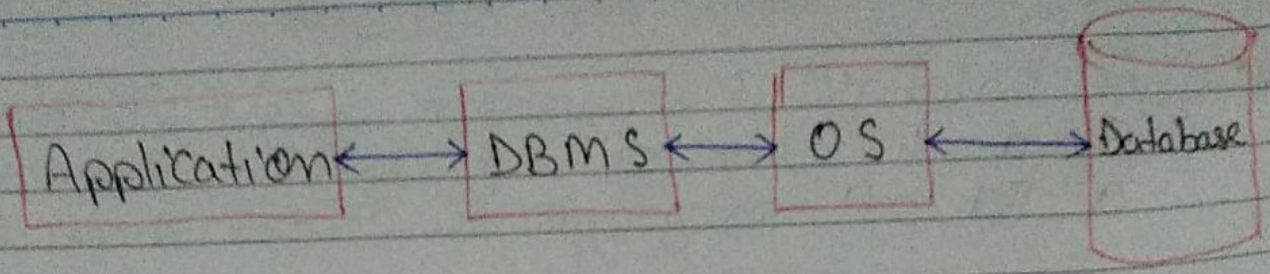
Database Tutorials - 01

Introduction of DBMS

- A Database Management System (DBMS) is a Collection of interrelated data and a set of programs to access those data.
- DBMS (Database management System) is used to organize the data in the form of a table, Schema, view and report etc.
- The Primary goal of a DBMS is to provide a way to store and retrieve database information that is both Convenient and efficient.
- Database management System is the Combination of two words:-

  Database + Management System = DBMS
- A Database is a Collection of related information Stored, so that it is available to many users for different purpose.
- Database management Syetem is a Collection of programs that enables Users to Create and maintain the database.



- DBMS can also be define as an interface between the Application program and the operating System to Access and manipulate that database.

- Database Management System is a Software which is used to manage the database. for Example: MYSQL, Oracle, etc are a very popular Commercial database which is used in Different applications.

DBMS Tutorials - 02

Characteristics and Applications of DBMS

Characteristics of DBMS

- It Uses a digital repository established on a server to store and manage the information.
- It Can provide a clear and logical view of the process that manipulates data.
- DBMS Contains automatic backup and recovery procedures.
- It Contain ACID properties which maintain data in a healthy state in case of failure.

Atomicity Consistency Isolation Durability

- It Can reduce the Complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It Can view the database from different viewpoints according to the requirements of the user.

Applications of DBMS

1. Banking ⇒ For maintaining Customer information, accounts, loans and banking transactions.
2. Universities ⇒ For maintaining Student records, Course registration and grades.
3. Railway Reservation ⇒ for checking the availability of reservation in Different trains, tickets, etc.
4. Airlines ⇒ for reservation and schedule information.
5. TeleCommunication ⇒ for keeping records of Calls made, generating monthly bills etc.
6. finance ⇒ for Storing information about holidays, Sales and purchase of financial instruments.
7. Sales ⇒ For Customer, product and purchase information.

Database (DBMS) Tutorial

Advantages and Disadvantages of DBMS

Advantages of DBMS ⇒

- **Control 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 the database.
- **Data Sharing ⇒** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily maintenance ⇒** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time ⇒** It reduces development time and maintenance need.
- **Backup ⇒** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failure and restores the data if required.
- **Multiple user interface ⇒** It provides

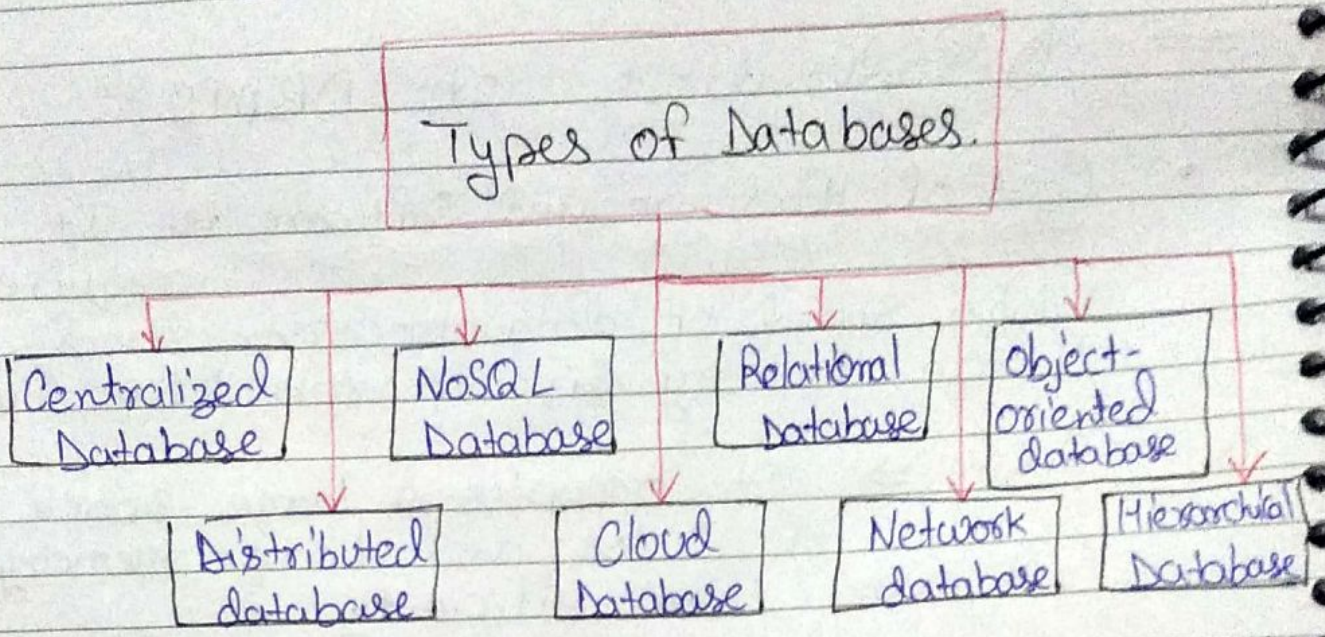
Different types of user interface like graphical user interfaces, application program interfaces.

Disadvantages of DBMS

- Cost of Hardware and Software ⇒ It requires a high speed of data processor and Large memory size to run DBMS s/w.
- Size ⇒ It occupies a large space of disks and Large memory to run them efficiently.
- Complexity ⇒ Database system Creates additional Complexity and requirements.
- Higher impact of failure ⇒ failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database Corruption then the data may be lost forever.

Types of Databases

There are various types of databases used for storing different varieties of data.



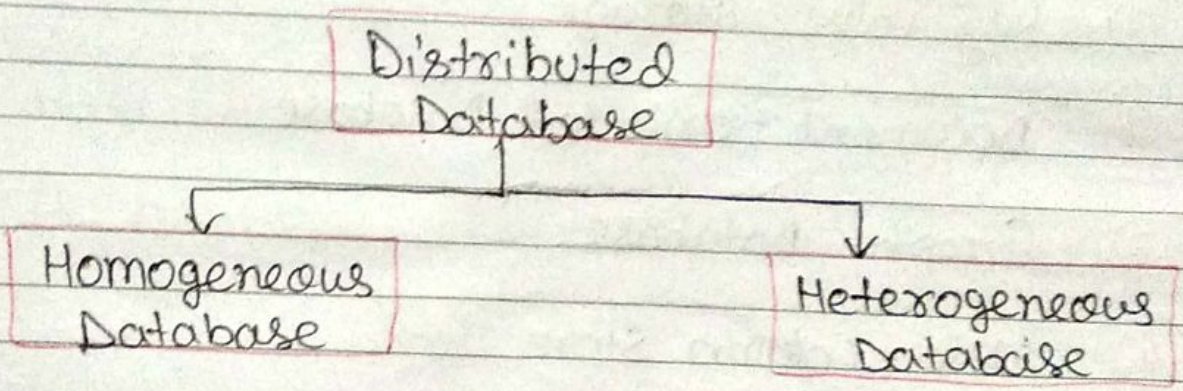
1) Centralized Database ⇒ • It is the type of Database that stores data at a Centralized database system.

- It Comforts the users to access the stored data from different location through several applications.
- these Applications Contain the authentication process to let users access data securely.
- An Example of a Centralized database Can be Central Library that carries a Central database of each library in a College/University.

2) Distributed Database ⇒ • In Distributed Database, data is distributed among different database System of an organization.

- These database system are connected via communication links. Such links help the End-users to access the data easily.

It is divided into two subpart



3) Relational Database ⇒ • It Stores data in the form of rows (Tuple) and Columns (attributes), and together forms a table (relation).

- A relational database uses SQL for storing, manipulating, as well as maintaining the data.
- Each table in the database carries a key that makes the data unique from others.
- Example of Relational databases are MYSQL, Microsoft SQL server, Oracle, etc.

4) NoSQL Database ⇒ • Non-SQL / Not Only SQL is a type of database that is used for storing a wide range of data sets.

- It is not a relational database as it stores data not only in tabular form but in several different ways.
- It also divided into four sub part.

1) Key-Value Storage
2) Document-oriented Database
3) Graph Database
4) Wide-Column Store.

5) Cloud Database ⇒ • It is a type of database where data is stored in a virtual environment and executes over the cloud computing platform.

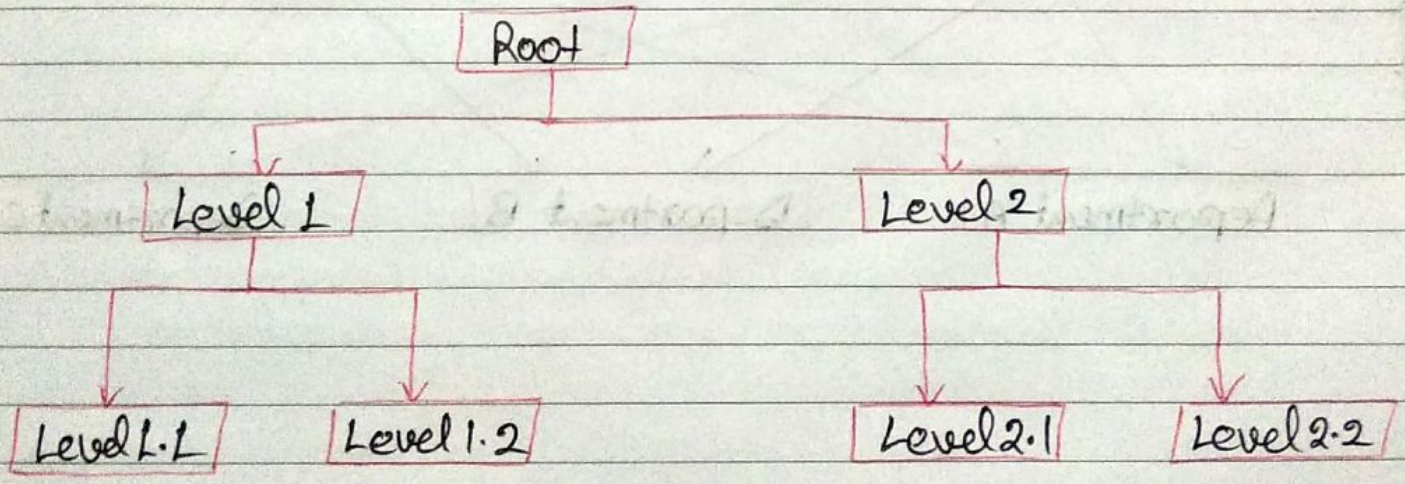
- It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc) for accessing the database.
- Some example of cloud database such as Amazon Web Services (AWS), Microsoft Azure, Google Cloud SQL etc.

6) Object-Oriented Database ⇒ • The type of database that uses the object-based data model approach for storing data in the database system.

- The data is represented and stored as objects which are similar to the objects used in the object-oriented programming language.

7) Hierarchical Databases ⇒ • It is a type of Database that stores data in the form of parent-children relationship node.

- It organizes data in tree-like structure.

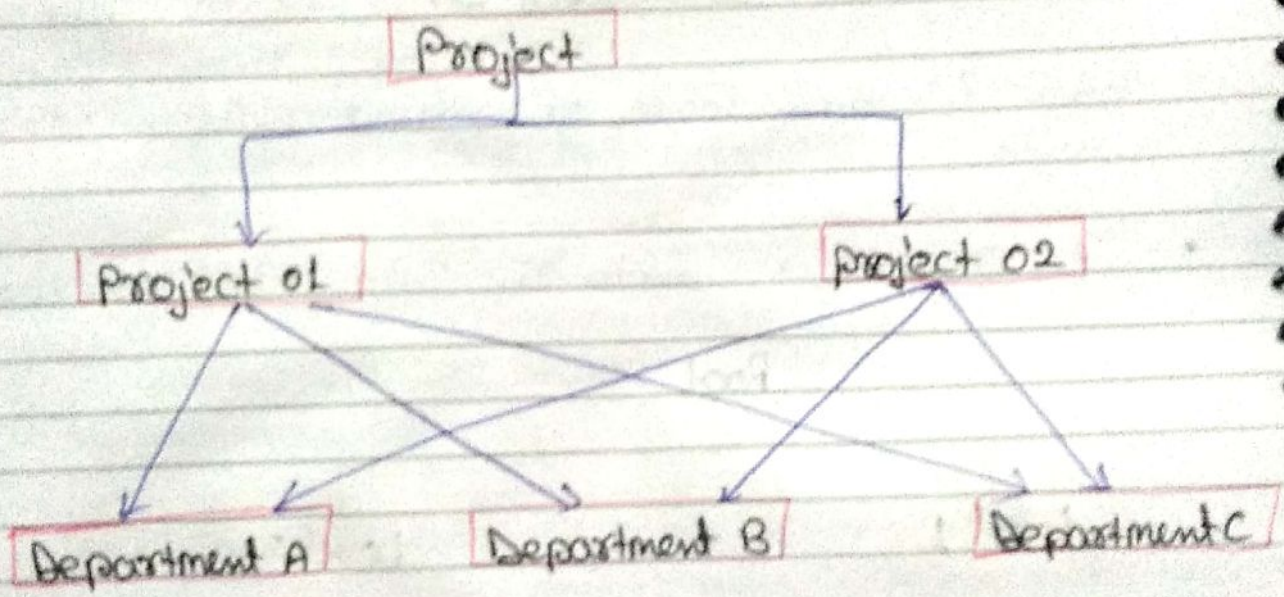


- Data get stored in the form of records that are connected via links.
- Each child record in the tree will contain only one parent. On the other hand, each parent record can have multiple child records.

Camlin

8) Network Database ⇒ It is the database that typically follows the network data model

- In this representation of data it's in the form of nodes connected via links between them.
- It allows each record to have multiple children and parent nodes to form a generalized graph structure.



What is RDBMS

- RDBMS Stands for Relational Database management System
- RDBMS is a database management System that is based on the relational model as Introduced by Dr. E. F. Codd.
- RDBMS Stores data in the form of related tables.
- An Important feature of relational Systems is that a Single database Can be Spread across several tables.
- All modern database management System like SQL, MS SQL server, IBM DB2, ORACLE, MY-SQL and microsoft Access are based on RDBMS.
- A relational database is the most Commonly used database. It Contains several tables, and Each table has its primary key.
- Due to a Collection of an organized set of tables, data Can be accessed easily in RDBMS.
- Everything in a relational database is Stored in the form of relations.

primary key

Columns or field or Attributes

EMP_ID	EName	Post	Salary
E1	Rahul	Clerk	20000
E2	Kamal	Peom	80000
E3	Kailash	faculty	120000
E4	Kamal	manager	8000

Row or Record or Tuple

Domain

Data Value

Degree (No. of Columns) = 4

Cardinality (No. of Rows) = 4

- table / Relation ⇒ • Everything in a relational database is stored in the form of relations.
- The RDBMS database uses tables to store data.
- A table is a collection of related data entries and contains rows and columns to store data.

Properties of Relational tables ⇒

- Value are atomic
- Column value are of the same kind
- Each row is Unique.
- Each Column has a unique name
- The sequence of rows is insignificance.
- The sequence of Columns is insignificance.

- Row or record ⇒ • A row of a table is also called a record or tuple.
  - Row Contains the specific information of each Entry in the table.
  - It is a horizontal Entity in the table.

Properties of a row ⇒
- No two tuples are identical to each other in all their Entries.
- All tuples of the relation have the same format and the same number of Entries.
- The order of the tuple is irrelevant. They are identified by their Content, not by their position.

- Column / attributes / fields ⇒ • A Column is a vertical Entity in the table which Contains all information associated with a specific field in a table.

Properties of an Attributes ⇒
- Every attribute of a relation must have a name.
- Null values are permitted for the attributes.
- Default values Can be specified for an attribute automatically inserted if no other value is specified for an attribute.
- Attribute that uniquely identify each tuple of a relation are the primary key.

- Data item/cells ⇒ • The smallest unit of data in the table is the individual data item.
  - It is stored at the intersection of tuples and attributes.
- Degree ⇒ • the total number of attributes that comprise a relation is known as the degree of the table.
- Cardinality ⇒ • The total number of tuples at any one time in a relation is known as the table's Cardinality.
  - The relation whose Cardinality is zero is called an Empty table.
- Domain ⇒ • The domain refers to the possible values each attribute can contain.
  - It can be specified using standard data types such as integers, floating number etc.

Dr. E.F Codd's Rules for RDBMS

- Dr. E.F. Codd is an IBM researcher who first developed the relational data model in 1970.
- In 1985, Dr. Codd published a list of 12 rules that define an ideal relational database and has provided a guideline for the design of all relational database S/m.

Rule 1: The Information Rule

This rule simply requires that all data should be presented in table form. This is the basis of relational model.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value) and attribute-name (column value).

Rule 3: Systematic Treatment of Null value

The Null values in a database must be given a systematic and uniform treatment. This is very important Rule because a NULL can be interpreted as one the following - data is missing, data is not known or data is not Applicable.

Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as data dictionary, which can be

accessed by authorized users.
- Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule
A database can only be accessed using a language having linear syntax that support data definition, data manipulation and transaction management operations.
All commercial relational databases use forms of SQL as their supported language.

Rule 6: View Updating Rule
Data can be presented in different logical combinations called views.
Each view should support the same full range of data manipulation that has direct access to a table available.

Rule 7: High Level Insert, Update and Delete
A database must support high-level insertion, updation, and deletion. This must not be limited to a single row, that is, It must also support union, intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence
The data stored in a database must be independent of the applications that access the database.

Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications

Rule 9: Logical Data Independence
The logical data in a database must be independent of its user's view (Application). Any change in Logical data must not affect the applications use it.

Rule 10: Integrity Independence
The database Language (like SQL) should support Constraints on user input that maintain database integrity.
No Component of a primary key can have a null value.
If a foreign key is defined in one table, any value in it must exist as a primary key in another table.

Rule 11: Distribution Independence
The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database System.

Rule 12: Non-Subversion Rule

There should be no way to modify the database structure other than through the multiple row database language (SQL) most databases today support administrative tools that allows some direct manipulation of the data structure.

Difference Between DBMS and RDBMS

DBMS	RDBMS
1) In DBMS relationship between two tables or files are maintained Programmatically.	1) In RDBMS, relationship between two tables and files can be specified at the time of table creation.
2) DBMS applications store data as files.	2) RDBMS applications store data in tabular form.
3) Normalization is not present in DBMS	3) Normalization is present in RDBMS.
4) DBMS does not support Client/server architecture	4) Most of the RDBMS support Client/server architecture
5) DBMS does not apply any security with regards to data manipulation	5) RDBMS defines the integrity constraint for the purpose of ACID (Automicity, Consistency, isolation and Durability).
6) DBMS uses file system to store data, so there will be no relation between the tables.	6) In RDBMS, data values are stored in the form of tables, so a relationship between these data value will be stored in the form of a table as well.

DBMS	RDBMS
7) DBMS does not Support distributed database	7) RDBMS Support distributed database.
8) DBMS is ment to be for small organization and deal with small data. it support single user.	8) RDBMS is designed to handled Large amount of data. it Supports multiple users.
9) DBMS may satisfy less than 7 to 8 rules of Dr. E.F. Codd.	9) RDBMS Support all 12 rules of Dr. E.F. Codd.
10) Example of DBMS are file system, xml etc.	10) Example of RDBMS are mysql, postgre, oracle etc.

Difference Between DBMS and File Processing System

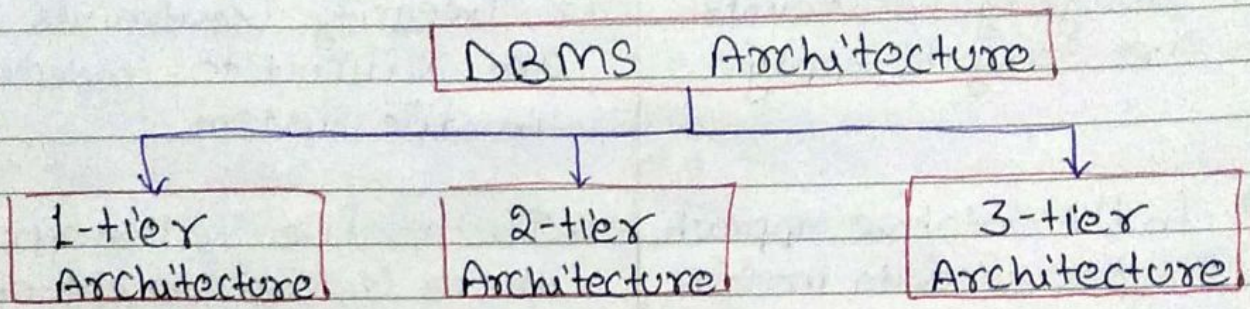
DBMS	File Processing System
01⇒ DBMS is a Collection of data. In DBMS, the user is not required to write the procedures.	01⇒ The file system is a collection of data. In this system, the user has to write the procedures for managing the database.
02⇒ In DBMS Due to Centralized approach, data sharing is easy.	02⇒ In file System Data is Distributed in many files, and it may be of different format, so it's not easy to share data.
03⇒ DBMS gives an abstract view of data that hides the details.	03⇒ The file system provides the detail of the data representation and storage of data.
04⇒ In DBMS Data redundancy problem is not found.	04⇒ In file System redundancy problem is found.
05⇒ Data Inconsistency does not Exist	05⇒ Data Inconsistency Exist.
06⇒ DBMS database Structure is Complex to design	06⇒ The file system approach has a simple structure.

DBMS	File Processing System
07⇒ Accessing database is easier	07⇒ Accessing database is comparatively difficult.
08⇒ In DBMS, Data Independence exists, and it can be of two types: • Logical Data Independence • Physical Data Independence	08⇒ In the file system approach, there exists no Data Independence.
09⇒ Integrity Constraints are Easy to Apply.	09⇒ Integrity Constraints are difficult to implement in file system.
10⇒ In the database approach, 3 types of data models exist: Hierarchical data Model Network data Model Relational data Model	10⇒ In the file system approach, there is no concept of data model exists.

Application Architecture of DBMS

- The DBMS design depends upon its architecture.
- The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- DBMS architecture depends upon how users are connected to the database to get their request done.

TYPES OF DBMS ARCHITECTURE



Database architecture can be seen as a Single tier or multitier. But Logically, database Architecture is of two types like 2-tier Architecture and 3-tier Architecture.

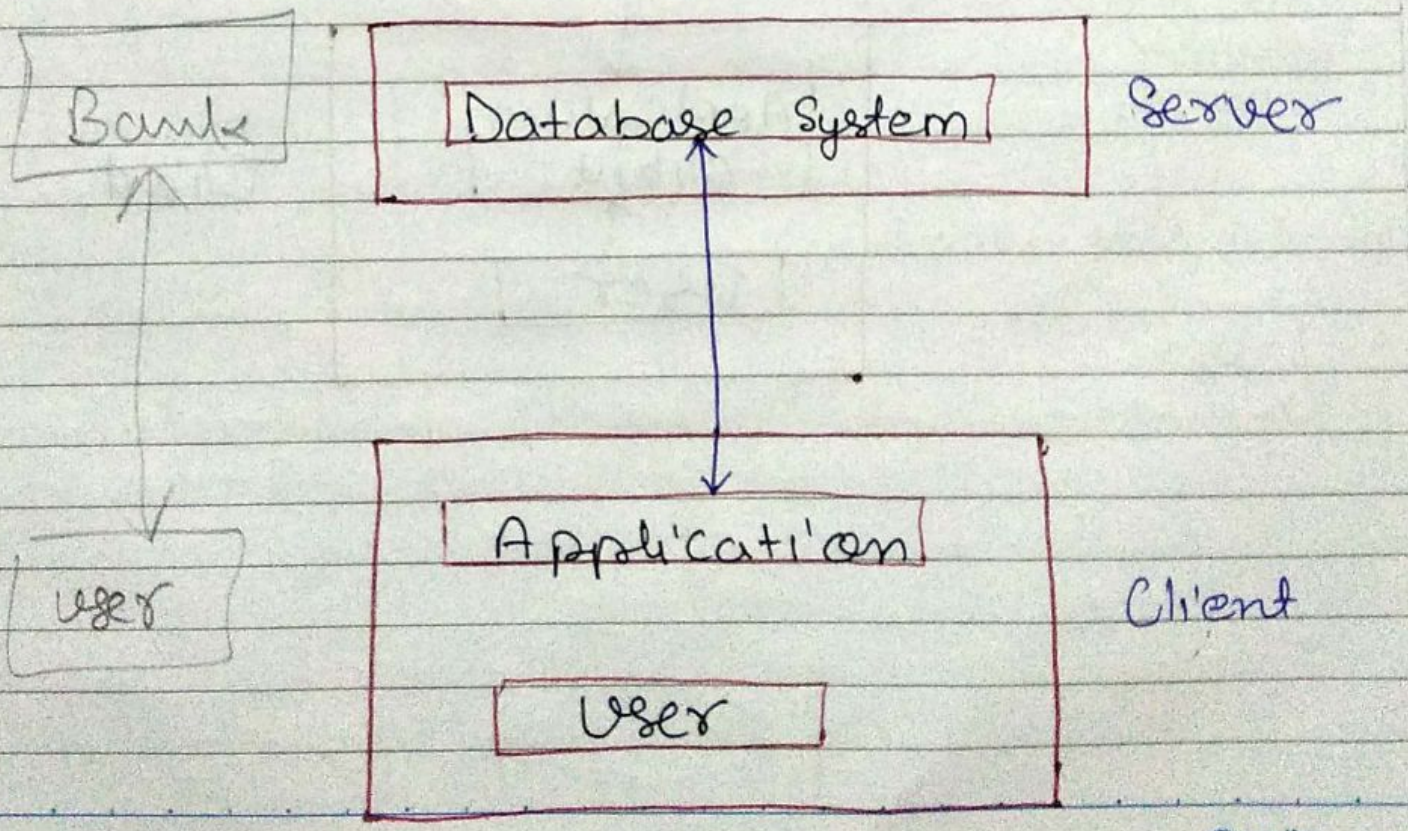
1-Tier Architecture ⇒

- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.

- The 1-tier architecture is used for development of the local Application, where programmers can directly communicate with the database for the quick response.

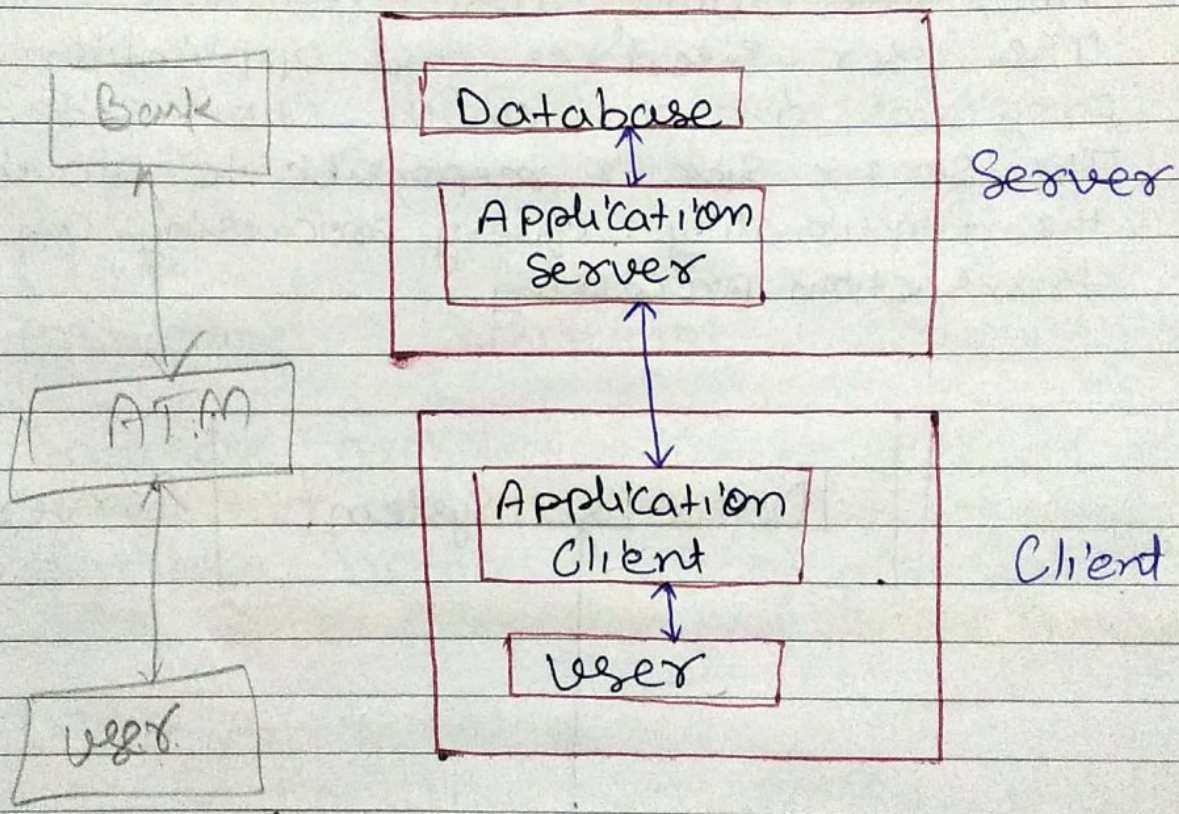
2-Tier Architecture ⇒

- The 2-Tier architecture is same as basic Client-Server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like ODBC, JDBC are used.
- The user Interfaces and application programs are run on the Client-side.
- The server side is responsible to provide the functionality : query processing and transaction processing.



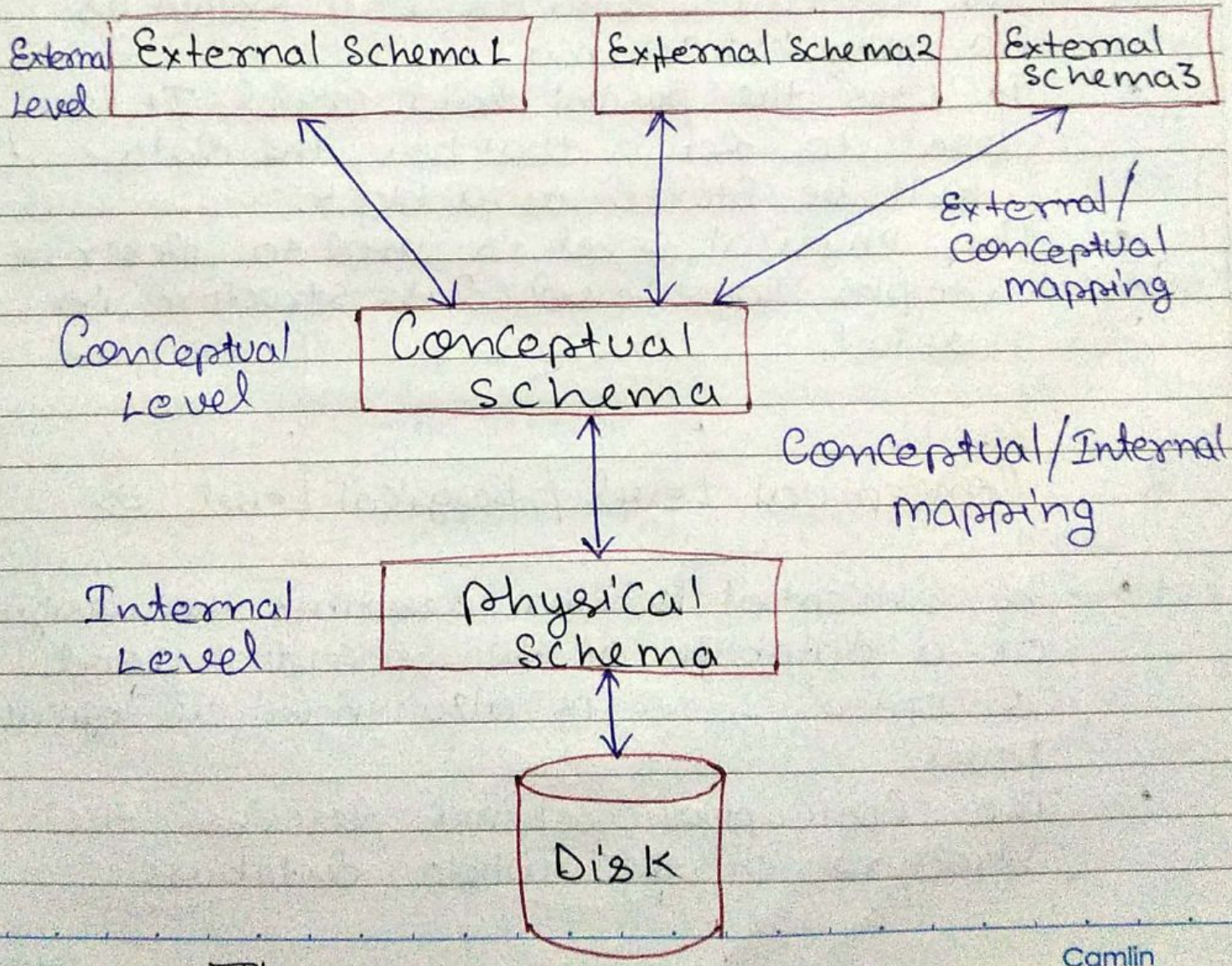
3-tier Architecture ⇒

- The 3-tier architecture Contains another Layer between the Client and Server.
- In this architecture, Client can't directly Communicate with the Server.
- The application on the Client-end interacts with an application Server with further Communicates with the database System.
- The 3-tier architecture is used in Case of Large web application.



Three Schema Architecture of DBMS

- The overall design of the database is called the database schema.
- The tree Schema Architecture is also called ANSI/ SPARC (American National Standards Institute, Standards Planning and Requirements Committee) architecture or three-Level architecture.
- The tree schema architecture is also used to separate the user applications and physical database.



Three-Schema Architecture.

(10) → 5

1 Internal Level / Internal View ⇒

- The internal Level has an internal Schema which describes the physical Storage Structure of the database.

Internal View

STORED_EMPLOYEE record Length 60
Empno : 4 decimal offset 0 unique
Ename : String Length 15 offset 4
Salary : 8, 2 decimal offset 19
Deptno : 4 decimal offset 27
Post : String Length 15 offset 31

- The internal schema is also known as a physical schema
- It uses the physical data model. It is used to define that how the data will be Stored in a block.
- The Physical Level is used to describe Complex Low-Level data structure in detail.

2 Conceptual Level / Logical Level ⇒

- The Conceptual Schema describes the design of a database at the Conceptual Level. Conceptual Level is also known as logical Level.
- The Conceptual Schema describes the Structure of the whole database

Camlin

Create table Emp (Empno integer(4), ...);

Employee
Empno : integer(4) key
Ename : String(15)
Salary : String(8)
Deptno : integer(4)
Post : String(15)

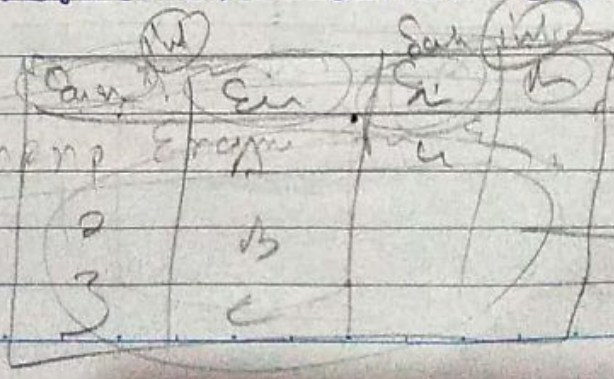
- In the Conceptual Level, internal details such as an implementation of the data structure are hidden.
- Programmer and database administrator work at this level.

3 External Level / View Level ⇒

- At the External Level, a database contains several schemas that sometimes called as Subschema. The Subschema is used to describe the different view of the database.

Empno	Ename	Salary	Deptno	Post

- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The View Schema describes the End user interaction with database Systems.



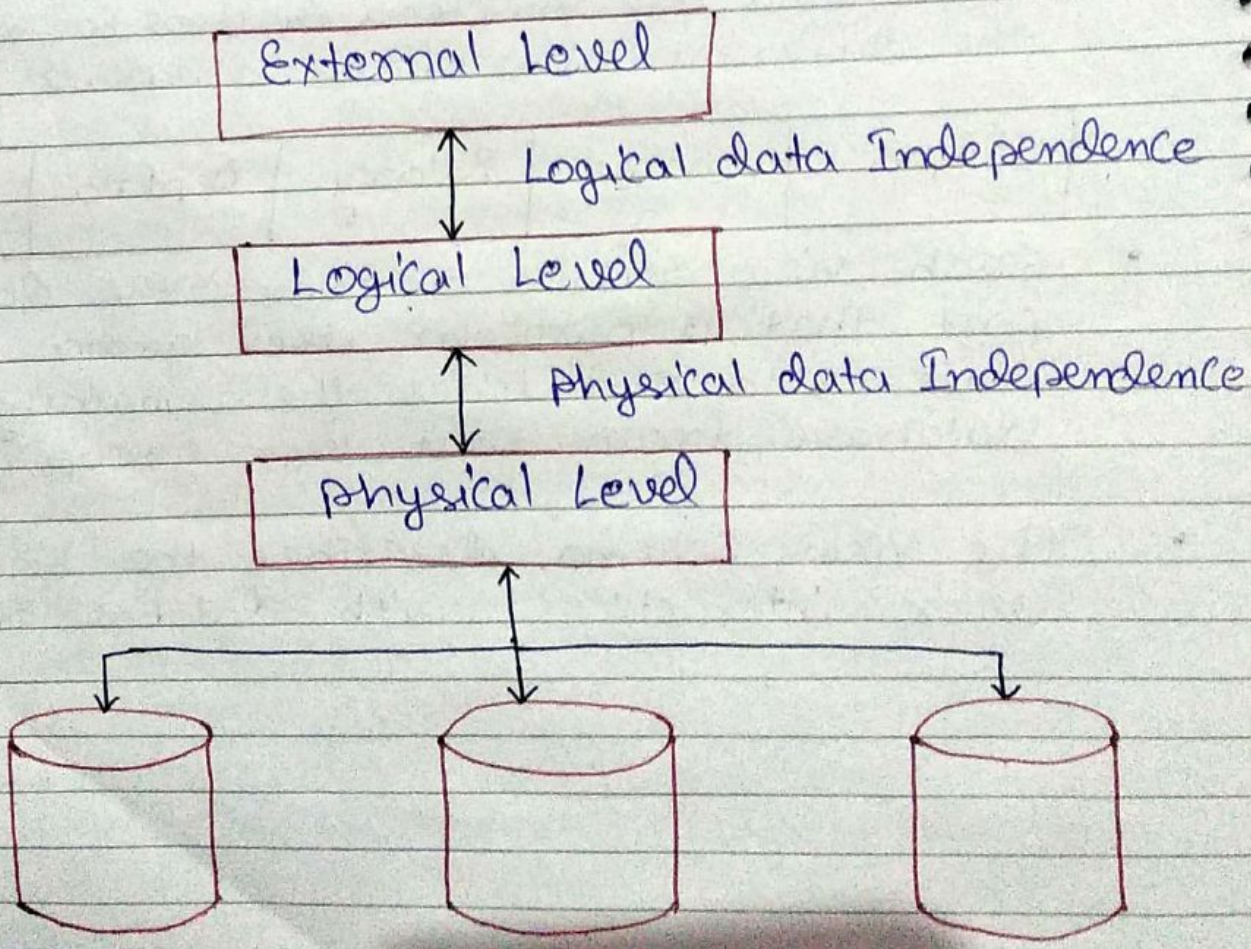
Data Independence

- The ability to modify a schema definition in one level without affecting a schema definition in the next higher level is called Data Independence.

- Data independence is one of the main advantages of DBMS.

Data independence is two types

- Physical Data Independence
- Logical Data Independence.

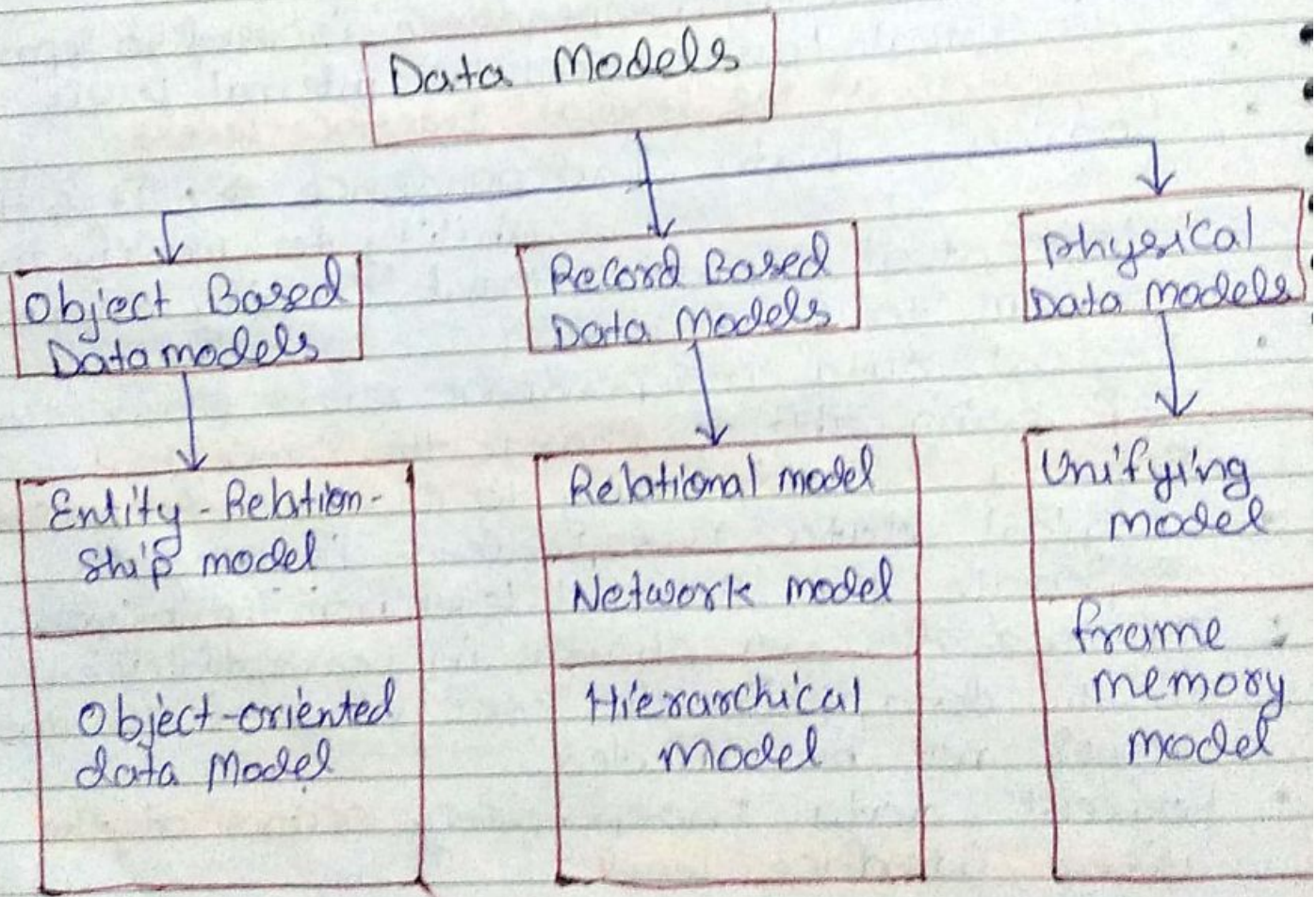


- Physical Data Independence ⇒ • Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
- If we do any changes in the storage size of the database system server, then the conceptual structure of the database will not be affected.
- It is the ability to modify the physical schema without causing application programs to be rewritten.
- physical data Independence is used to separate conceptual levels from the internal levels.
- It occurs at the logical interface levels.
- Logical Data Independence ⇒ • It is the ability to modify the conceptual schema without causing application program to be rewritten.
- Logical data Independence refers characteristic of being able to change the conceptual schema without having to change the external schema.
- Logical data Independence is used to separate the external level from the conceptual.
- If we do any change in conceptual view of the data, then the user view of the data would not be affected.
- Logical data Independency occurs at the user interface level.

M.M.I

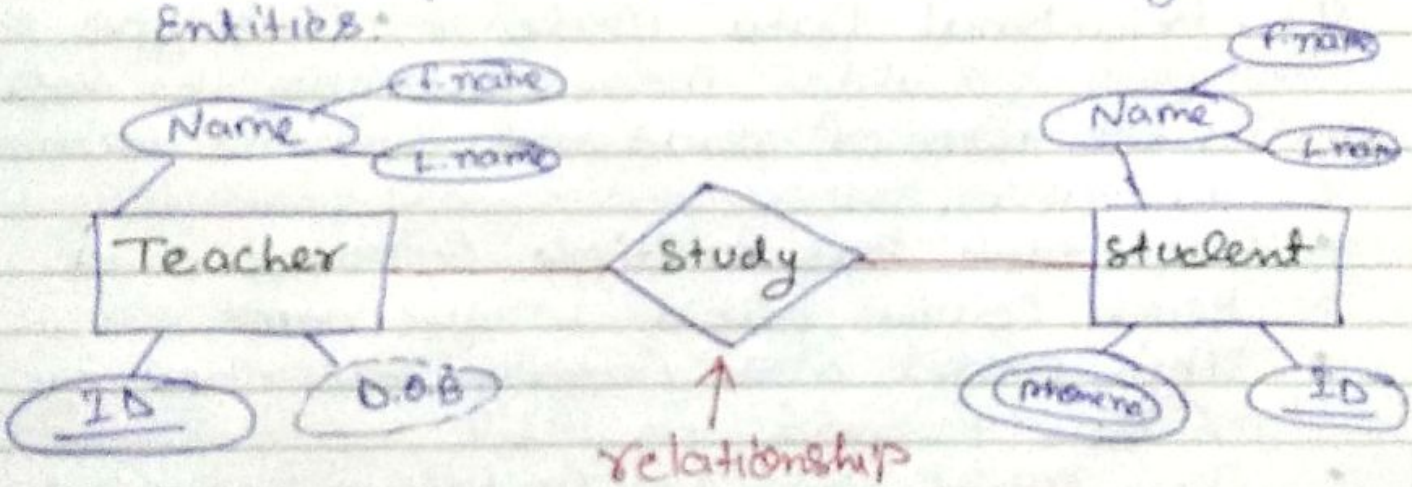
Data Models

- Data Model is the modeling of the data description, data semantics, and Consistency Constraints of the data.
- Data model provides the Conceptual tools for describing the design of a database at each level of data abstraction.
- A Data model can also be define as the Collection of high-level data description Constructs that hide many low level Storage Details.
- There are mainly three types of Data model:



1) object Based data model ⇒ • It is used to describe the data at the Logical and view Level.
- Object Based data model provide flexible structuring and structuring capabilities and allow to specify data constraints.
- There are mainly two types of object Based data model.

a) Entity Relationship model ⇒ An ER model is the Logical representation of data as objects and relationship among them.
- These objects are known as Entities and relationship is an association among these Entities:



b) Object-oriented Data model ⇒ • In an object-oriented model, information or data is displayed as an object and these objects store the value in the instance variable.
- In this model, object-oriented programming images are used.

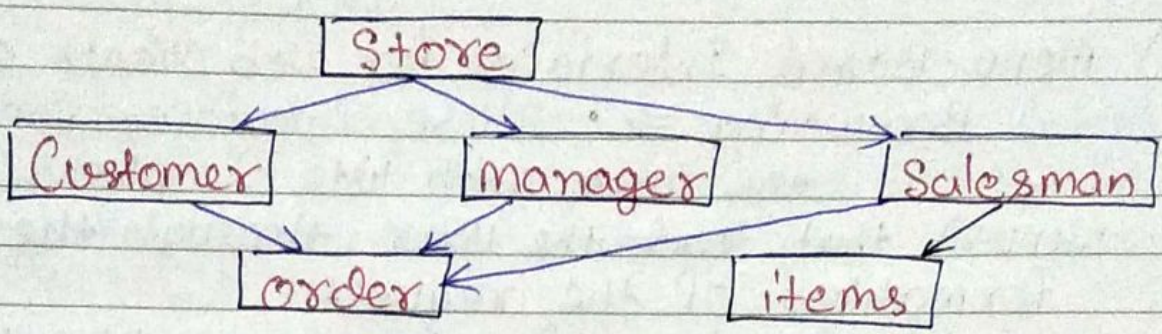
- This model works with object-oriented programming Language like - python, Java, VB.net and Perl etc. It was Constructed in the 1980s.

2) Record Based Data models ⇒ • It is used to describe data at Logical and view Level.
- This data model is used to Specify the overall Logical structure and to Specify the higher Level structure and provide higher Level description.
- There are three types of Record Based Data model

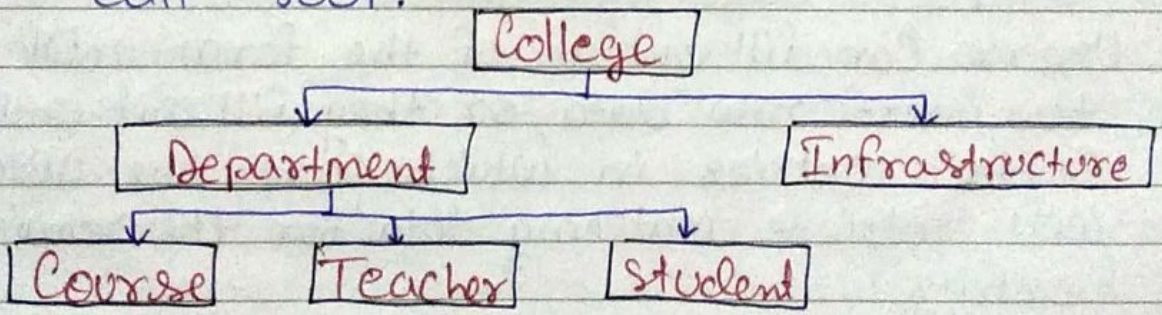
a) Relational Data Model ⇒ • This type of model designs the data in the form of rows and columns within a table.
- Each table has multiple Columns and Each Column has a unique name.
- This model was initially described by Edgar F Codd in 1969.
- This model uses the Certain mathematical operations from relational Algebra and relational Calculus on the relation such as union, join etc

Roll no.	Name	address
01	Kailash	Haridwar
02	Kamal	Dehradun
03	Karan	Rishikesh
04	Ram	Delhi

b) Network data Model ⇒ • In network data model, data is organized into graph. And it can have more than one parent node.
• It permits the modeling of many to many relationships in data.



c) Hierarchical Data model ⇒ • The Hierachical Data model organizes data in a tree structure.
• In this model, Each Entity has only one parent and many abstract Children. There is only one Entity in this model that we Call root.



3) Physical Data Model ⇒ • This data model is used to describe the data at low Level.

DBMS Interfaces

- A Database management System (DBMS) interface is a user interface that allows for the ability to input queries to a database without using the query Language itself. There are following types of Interface provide by a DBMS:-

1) Menu-Based Interfaces for Web Clients or Browsing ⇒ • These interfaces present the user with lists of options (called menus) that lead the user through the formation of the request.
- Basic advantage of using menus is that they removes the tension of remembering specific commands and syntax of any query language.

2) form-Based Interfaces ⇒ • A form-based interfaces displays a form to each user.
- Users can fill out all of the form entries to insert new data, or they fill out only certain entries, in which case the DBMS will retrieve matching data for the remaining entries.
- Many DBMS form specification Language, special Languages that help programmers specify such forms.

3) Graphical-User Interfaces ⇒ • A graphical user interface (GUI) typically displays a schema to the user in diagrammatic form.

Camlin

- The user can specify a query by manipulating the diagram.
- In many cases, GUIs utilize both menus and forms.
- Most GUIs use a pointing device such as mouse, to pick a certain part of the displayed schema diagram.

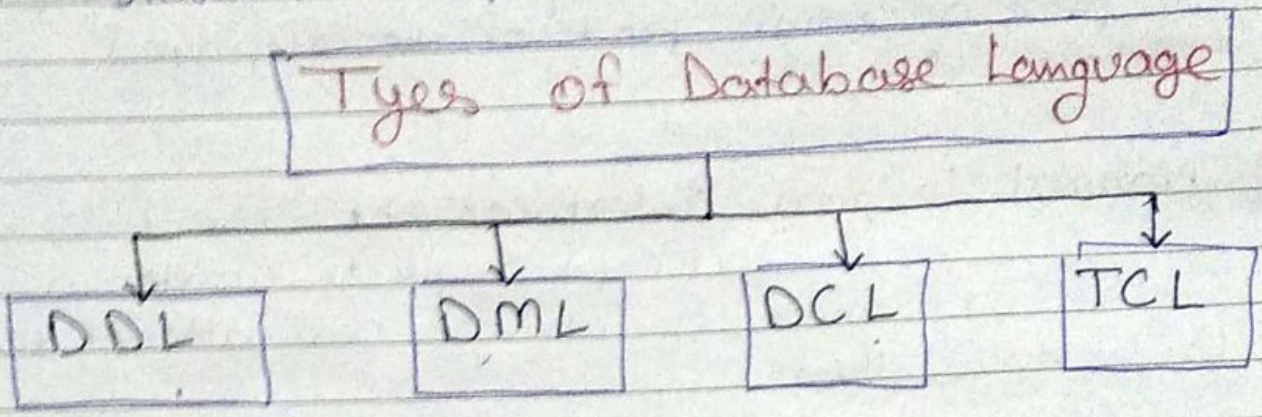
4) Natural Language Interfaces ⇒ • These interfaces accept requests written in English or some other language and attempt to understand them.
- A natural language interface usually has its own schema which is similar to the database conceptual schema.

5) Interface for Parametric Users ⇒ • Parametric users such as bank tellers, often have a small set of operation that they must perform repeatedly System analyst and programmers design and implement a special interface for a known class of naive users.

6) Interface for the DBA ⇒ • Most database System contain privileged commands that can be used only by the DBA's staff.
- These includes commands for creating accounts, setting system parameters etc.

DBMS Languages

- A DBMS has appropriate Languages and interfaces to Express database queries and updates.
- Database Language can be used to read, store and update the data in the database.



1⇒ DDL (Data Definition Language) ⇒

- DDL used to define database Structure or pattern.
- It is a set of SQL Commands used to Create, modify and delete database Structure but not data.
- It is used to Create Schema, tables, indexes, Constraints etc. in the database.
- These Commands are normally not used by a general user, who should be accessing the database via an application.
- They are normally used by the DBA to a limited Extent, a database designer or application are immediate developer.
- DDL updates a Special set of tables Called the data dictionary or data directory.

Lists of tasks that come under DDL:

- CREATE ⇒ Used to Create objects in the database.
- ALTER ⇒ Used to alters the structure of the database.
- DROP ⇒ Used to delete objects from the database.
- TRUNCATE ⇒ Used to remove all record from a table, including all spaces allocated for the records are removed.
- COMMENT ⇒ Used to add comments to the data dictionary.
- RENAME ⇒ Used to rename an object.

2⇒ DML (Data Manipulation Language) ⇒

- It is a set of SQL Commands used to select, modify and delete data in database not Database structure.
- It is used for accessing and manipulating data in a database. It handles user requests.
- DML Statements are used to manage data within schema objects

Lists of tasks that come under DML:

- SELECT ⇒ It retrieves data from a database
- INSERT ⇒ It inserts data into a table.
- UPDATE ⇒ It updates Existing data within a table

DQL → Data query Langue → Select (40)

- DELETE ⇒ It deletes all records from a table.
- MERGE ⇒ It performs UPSERT operation such as insert or update operation.
- CALL ⇒ It is used to call a structured query Language or a Java subprogram.
- LOCK TABLE ⇒ It Control Concurrency.

3) DCL (Data Control Language) ⇒ • The Data Control Language is used to control privilege in Databases.

- It is the Component of SQL Statement that Control access to data and to the database.
- To perform any operation in the database, such as for creating table, sequences or view we need privileges.

Lists of tasks that Come under DCL:
Privileges are of two types:

- SYSTEM ⇒ Creating a session, table etc. are all types of System privilege.
- OBJECT ⇒ any Command or query to work on tables comes under object privilege.

List of tasks that Come under DCL:

- Grant ⇒ It gives user access privileges to a database.
- Revoke ⇒ It take back permissions from the user.

Camlin

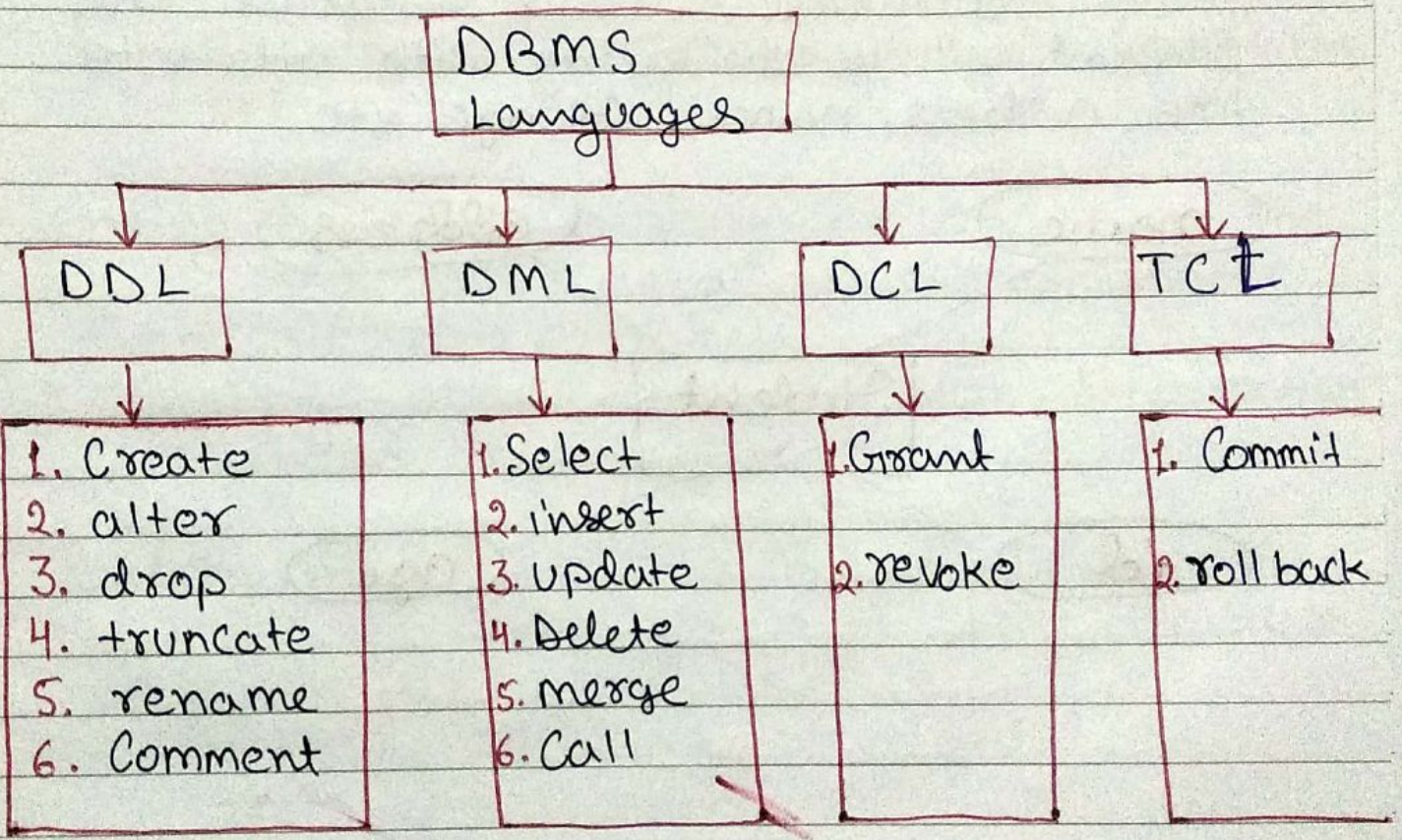
4) TCL (Transaction Control Language) ⇒ • TCL is used to run the changes made by the DML Statement.

• TCL can be grouped into a Logical transaction.

Lists of tasks that come under TCL:

• Commit ⇒ It is used to save the transaction on the database.

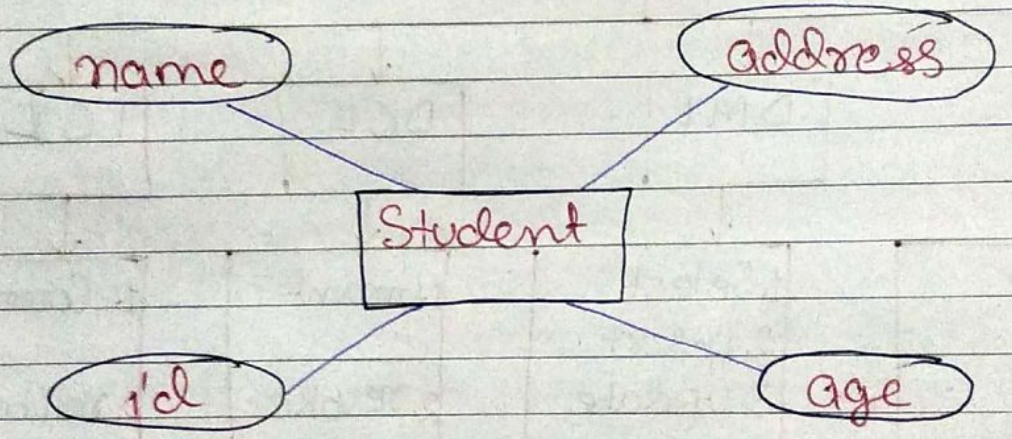
• Rollback ⇒ It is used to restore the database to original since the last commit.



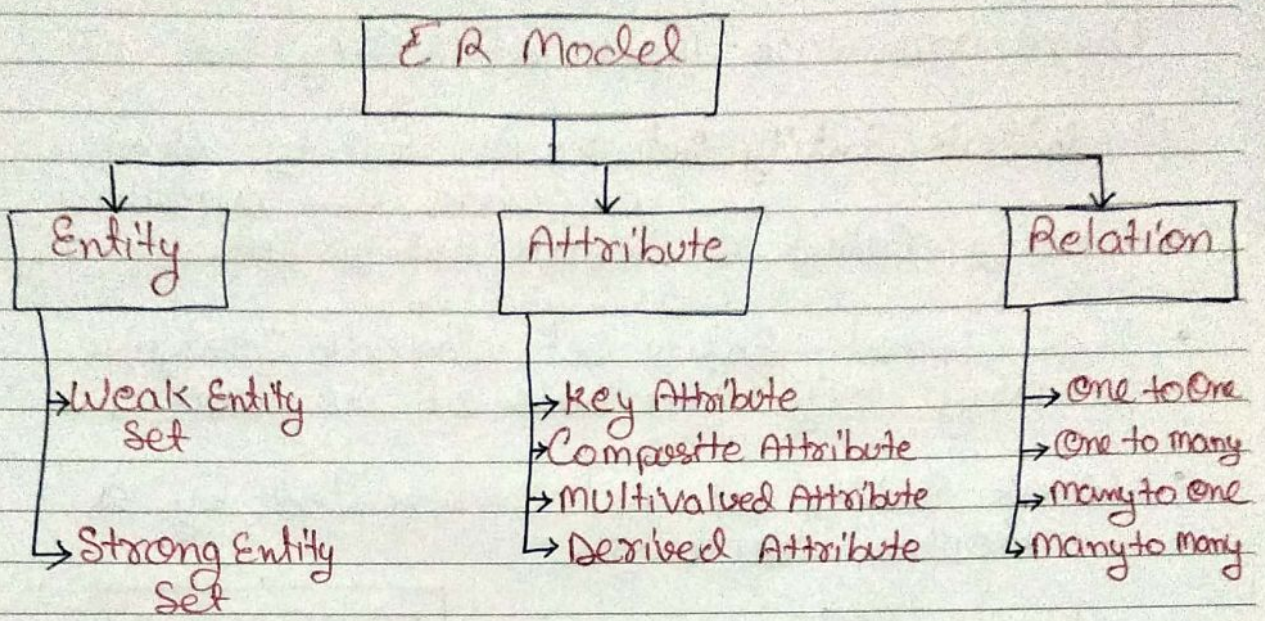
E-R Model Concepts

- The Entity-relationship (E-R) model is a high-level data model.
- It is based on a perception of a real world that consists of a collections of basic objects, called entities and of relationships among these objects.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

for Example ⇒ Suppose we design a School database. In this database, the Student will be an entity with attribute like address, name, id, age etc.



Component of ER Diagram

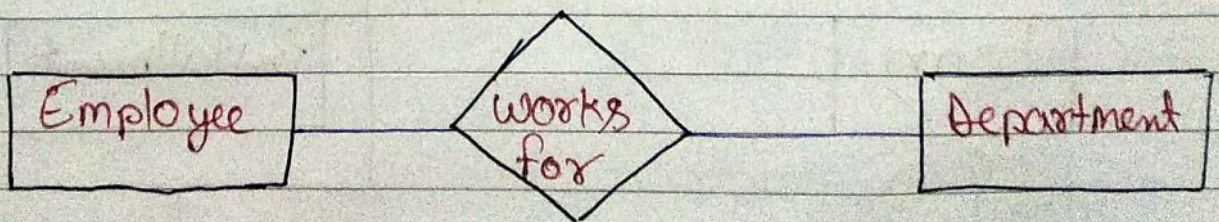


Entity ⇒ • It is a thing or object in the real world that is distinguishable from all other object.

• Anything about which we store information is called an Entity.

Entity Set ⇒ • It is a set of Entities of the same type that share the same properties or attributes.

• An Entity Set can be represented as rectangles.

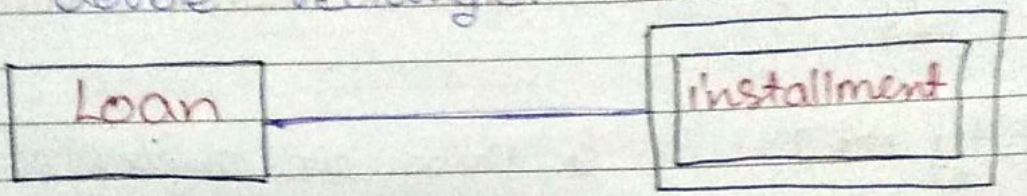


Types of Entity Set

There are two types of Entity Set

1) Weak Entity Set ⇒ • An Entity that depends on another Entity Called a Weak Entity Set.

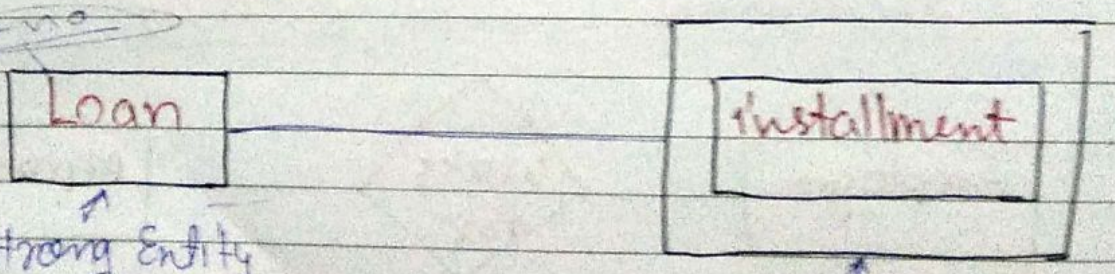
- The weak Entity Set doesn't Contain any key attribute of its own.
- Weak Entity Set is represented by a double rectangle.



Weak Entity Set

2) Strong Entity Set ⇒ • A Strong Entity Set is an Entity Set that Contains Sufficient attributes to uniquely identify all its Entities.

- Primary key Exists for a strong Entity Set.
- Single rectangle is used to representing a Strong Entity Set.



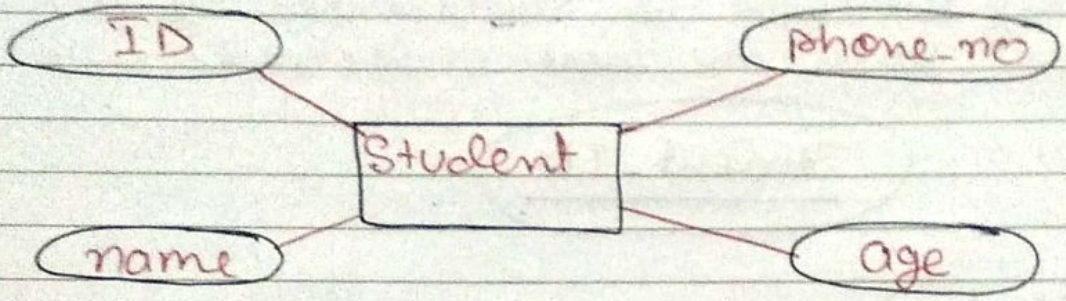
Strong Entity Set

Weak Entity Set

Attributes in ER model

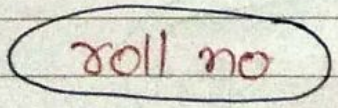
- The attribute is used to describe the property of an Entity.
- An Entity set may Contain any number of attribute
- Attributes are represented in an elliptical Shape.

For Example ⇒ ID, age, Contact number, name etc Can be attributes of a Student.

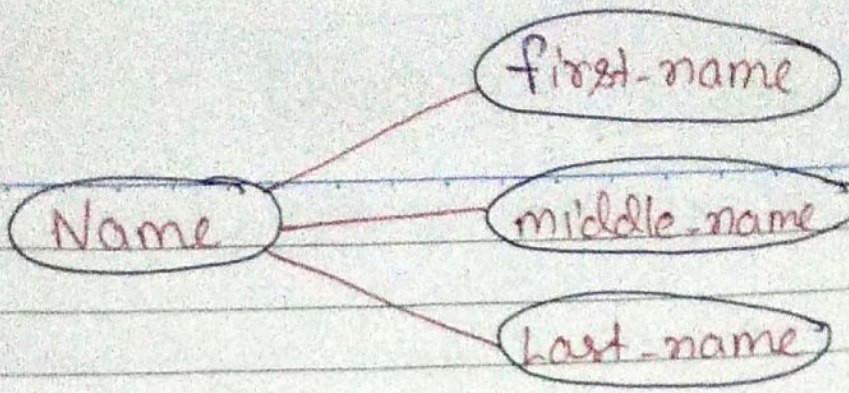


Types of attribute

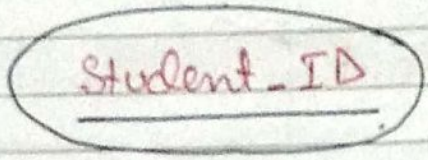
a) Simple attribute ⇒ • An attribute that Cannot be further Subdivided into Components is a Simple attribute.
• It is represented by ellipse
Ex ⇒ The roll number of a student, the id number of an Employee



b) Composite attribute ⇒ • An attribute that Can be Split into Components is a Composite attribute.
• The Composite attribute is represented by an ellipse, and those Ellipse are connected with an ellipse.

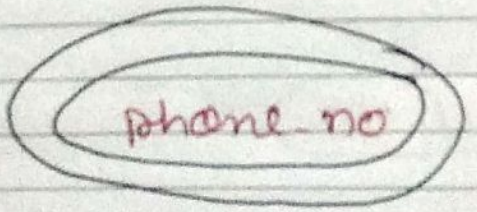


c) Key Attribute ⇒ • The key attribute is used to represent the main Characteristics of an Entity.
- It represent a primary key.
- The key attribute is represented by an ellipse with the next underlined.



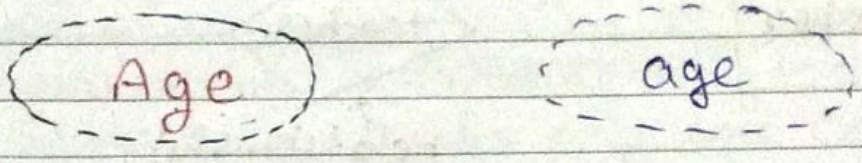
d) multivalued Attribute ⇒ • An attribute Can have more than one value. These attributes are known as a multivalued attribute.
- The double Ellipse is used to represent multivalued attribute.

For Example ⇒ a student Can have more than one phone number.



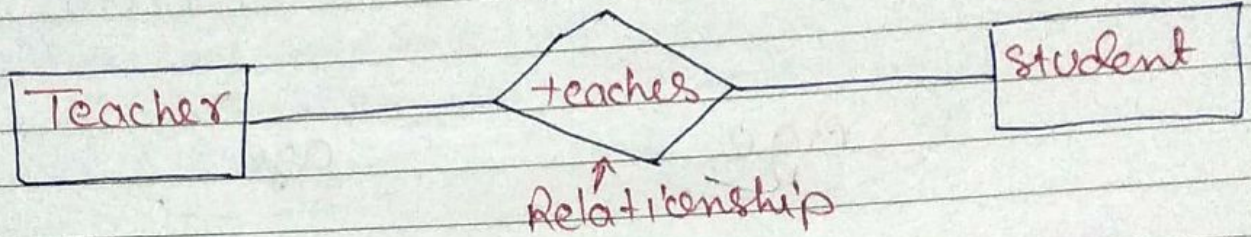
e) Derived Attribute ⇒ • An attribute that can be derived from other attribute is known as a derived attribute.
• It can be represented by a dashed Ellipse.

for Example ⇒ A person's age changes over time and can be derived from another attribute like date of birth.



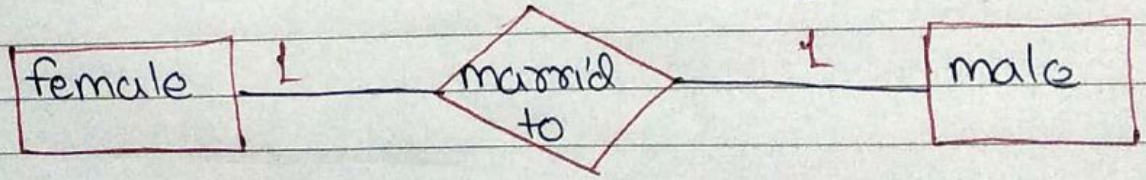
Relationship / Mapping Constraints

- A relationship is used to describe the relation between entities.
- Diamond or rhombus Box is used to represent the relationship.
- Mapping Cardinalities, or Cardinality ratios, express the no. of Entities to which another entity can be associated via a relationship set.



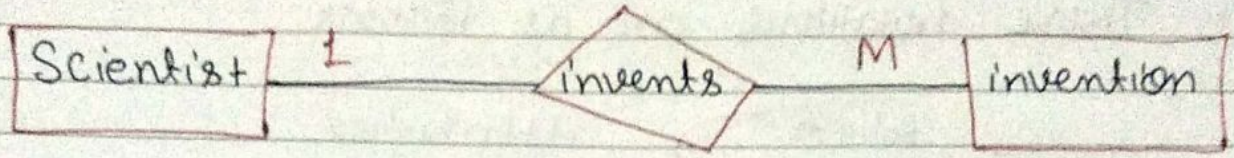
There are four types of mapping Constraints or relationship.

1) One to One Relationship ⇒ When only one instance of an entity is associated with the relationship, then it's known as one to one relationship.
For Example ⇒ A female can marry to one male, and a male can marry to one female.

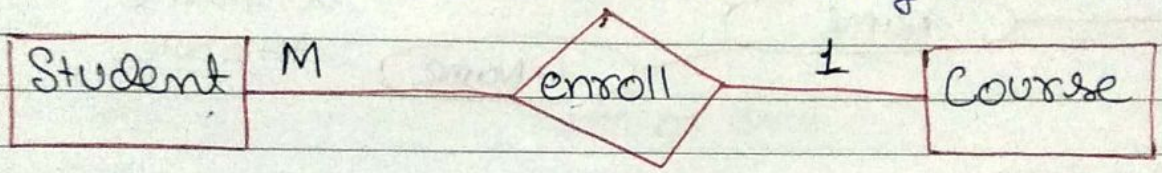


2) One to many Relationship ⇒ When only one instance of the Entity on the left and more than one instance of an Entity on the right associates with the relationship then this is known as a one-to many relationship.

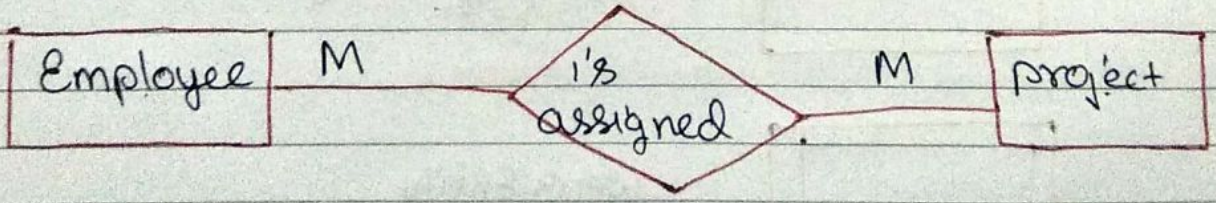
for Example ⇒ Scientist can invent many inventions, but the invention is done by the only Specific Scientist.



3) Many to one Relationship ⇒ When more than one instance of the Entity on the Left and only one instance of an entity on the right associates with the relationship is known as many to one Relation.
For Example ⇒ Student enrolls for only one Course, but a Course can have many students.

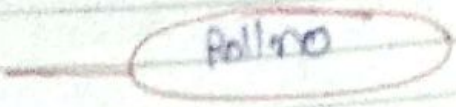


4) Many to many Relationship ⇒ When more than one instance of the Entity on the Left, and more than one instance of an Entity on the right associates with the relationship then it is known as a many to many relationship.
For Example ⇒ Employee can assign by many projects and project can have many Employee.

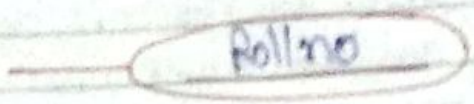


Notation of E-R Diagram

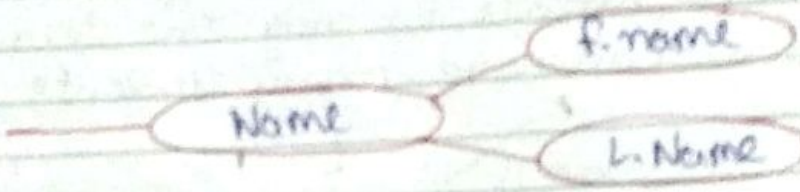
- Database can be represented using the notations. In ER Diagram, many notations are used to express the cardinality.
- These notations are as follows.



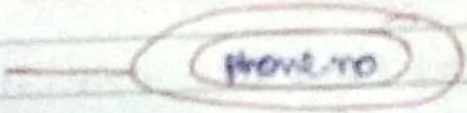
Attributes



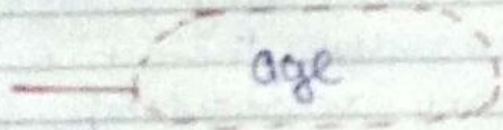
Key Attributes



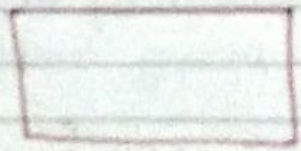
Composite Attributes



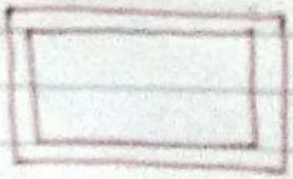
Multivalue attributes



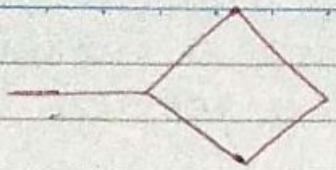
Derived Attributes



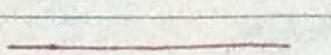
Strong Entity Set



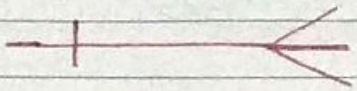
- Weak Entity Set



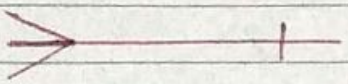
Relationship



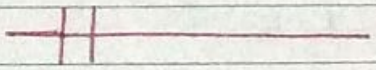
Links (one to one)



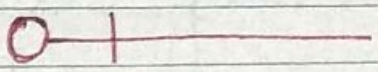
one to many



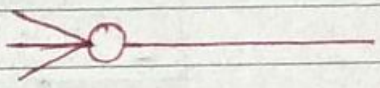
many to one



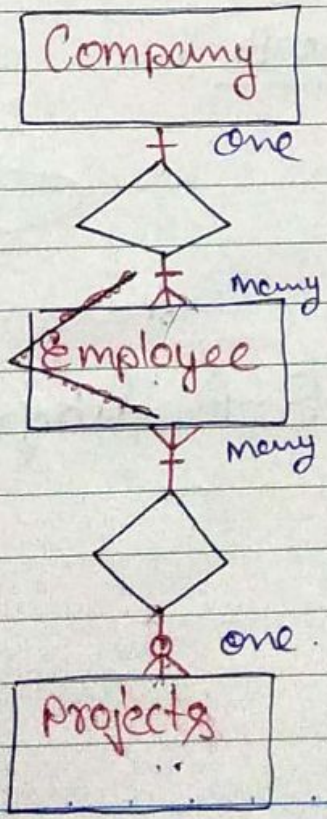
one and only one



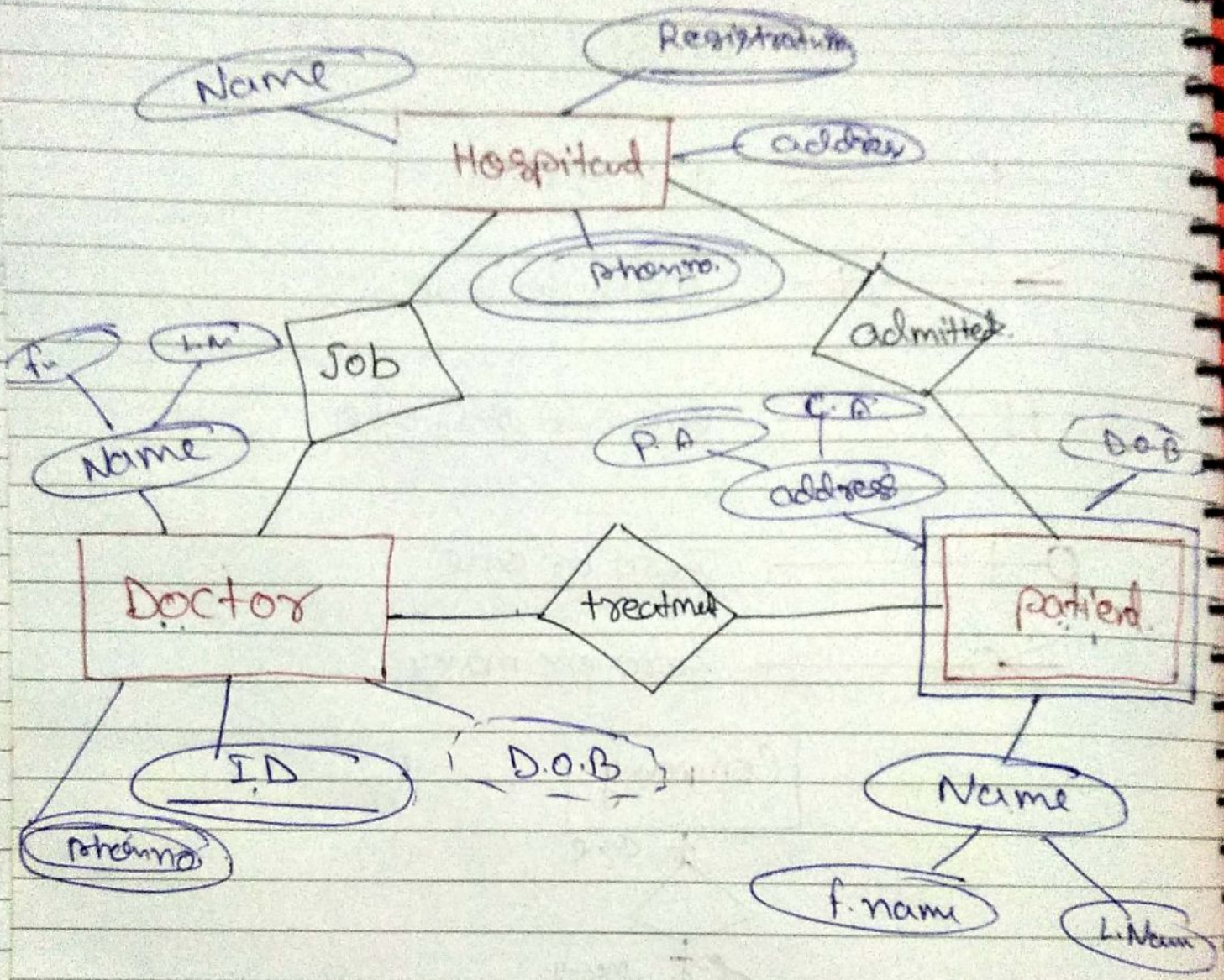
Zero or one



Zero or many



Q⇒ Construct an E-R Diagram for a hospital with a set of patients and a set of medical doctors.

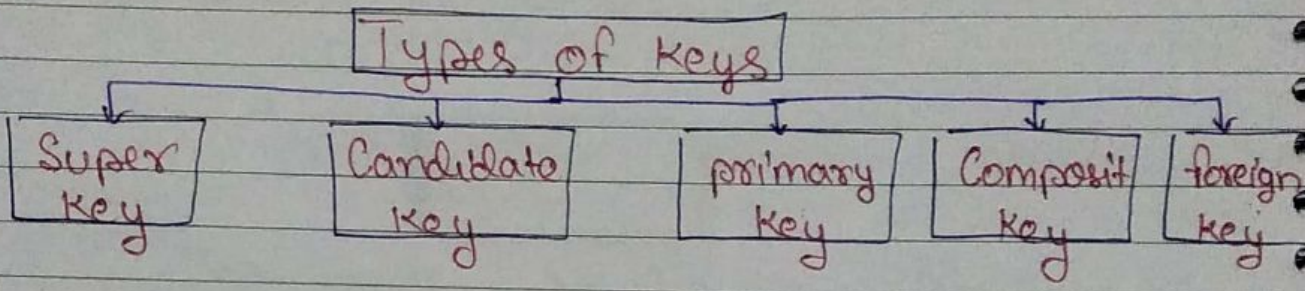


Q- Create a E-R diagram of our College.

M.M.I Keys in DBMS

- A key is a value which can always be used to uniquely identify an object instance.
- Key is used to uniquely identify any record or row of data from the table.
- It is also used to establish and identify relationships between tables.

For Example ⇒ ID is used as a key in the Student table because it is unique for each student. In the Person table, passport number, License-number are keys since they are unique for each person.



1) **Super key** ⇒ A super key is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the Entity set.

for Example ⇒ In Student table with attribute (S_Rollno., S_Name, S_Branch, S_Year).

Super key ⇒ S1 → S_Rollno., S_Name
S2 → S_Rollno. S_Branch
S3 → S_Rollno, S_Year
S4 → S_Rollno S_Name S_Branch
S5 → S_Rollno S_Branch S_Year

2) **Candidate key** ⇒ The minimal set of attributes that can uniquely identify a

tuple is known as a Candidate key.
- Candidate key can be define as the minimum no. of super key that identifies the record uniquely.
- It must contain unique values.
- Every table must have at least a single Candidate key.

for Example ⇒ In Student table with attribute (S_Rollno., S-Name, S_Branch, S_Year).
Candidate key ⇒ C1 ⇒ S_Rollno.
C2 ⇒ S-Rollno, S-name

3) **Primary key** ⇒ • Primary key can be define as the minimum no. of Candidate key that is Chosen by the database designer as the principal means of identifying entities within an Entity set.
- It is a unique key.
- It can identify only one tuple (a record) at a time
- It has no duplicate values, it has unique values.
- It cannot be NULL
- Primary keys are not necessarily to be a single Column, more than one the Column can also be a primary key for a table.

for Example ⇒ In Student table with attribute (S_Rollno, S-Name, S-Branch, S-Year)

Primary key ⇒ P1 ⇒ S_Rollno

4) **Composite key** ⇒ • Whenever a primary key Consists of more than one

attribute, it is known as a Composite key.
for Example ⇒ In student table with attribute (S_Rollno, S_ID, S_Name, S_Branch)

Composit key ⇒ S_Rollno, S_ID

5) **Foreign key** ⇒ • A foreign key is a Column whose value are the same as the primary key of another table.
- It Combines two or more relations (table) at a time.
- They act as a Cross reference between the tables.
- foreign key are the Column of the table used to point to the primary key of another table.

Student 1

P.K Roll no.	Name	ID (f.k)
1	Kailash	123
2	Kamal	234
3	Karan	456

foreign key

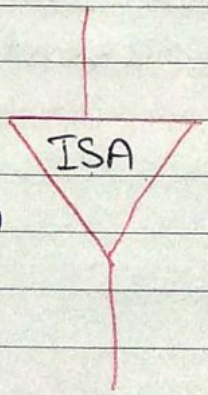
Student 2

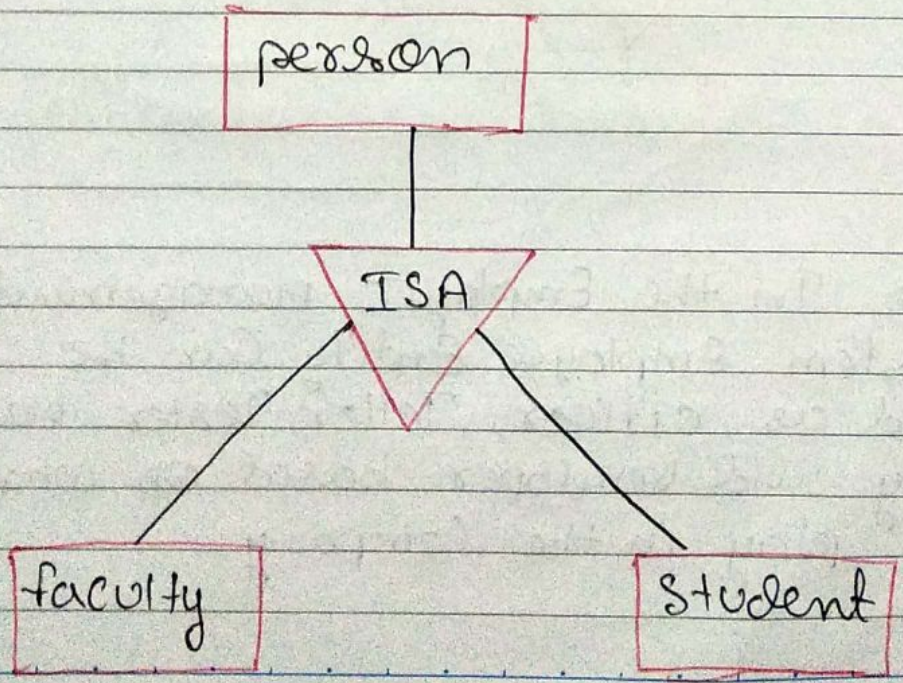
P.K ID	Branch	Address
234	IT	Dehradun
456	CSE	Rishikesh
678	EC	Haridwar

DBMS Generalization

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level Entity if they have some attributes in common.
- Generalization is a relationship that exists between a high-level Entity set and one or more lower-level Entity set.
- Generalization is more like Subclass and Superclass System, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized Entity suchas subclass are combined to make a Superclass.
- Generalization is represented by a triangle Component Labeled ISA. The label ISA stands for "is a" and represents.

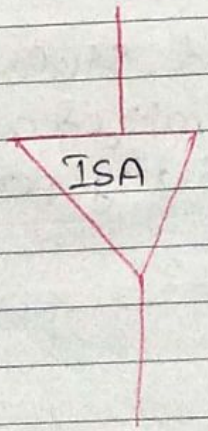
for Example ⇒ faculty and student Entities can be generalized and Create a higher level Entity person



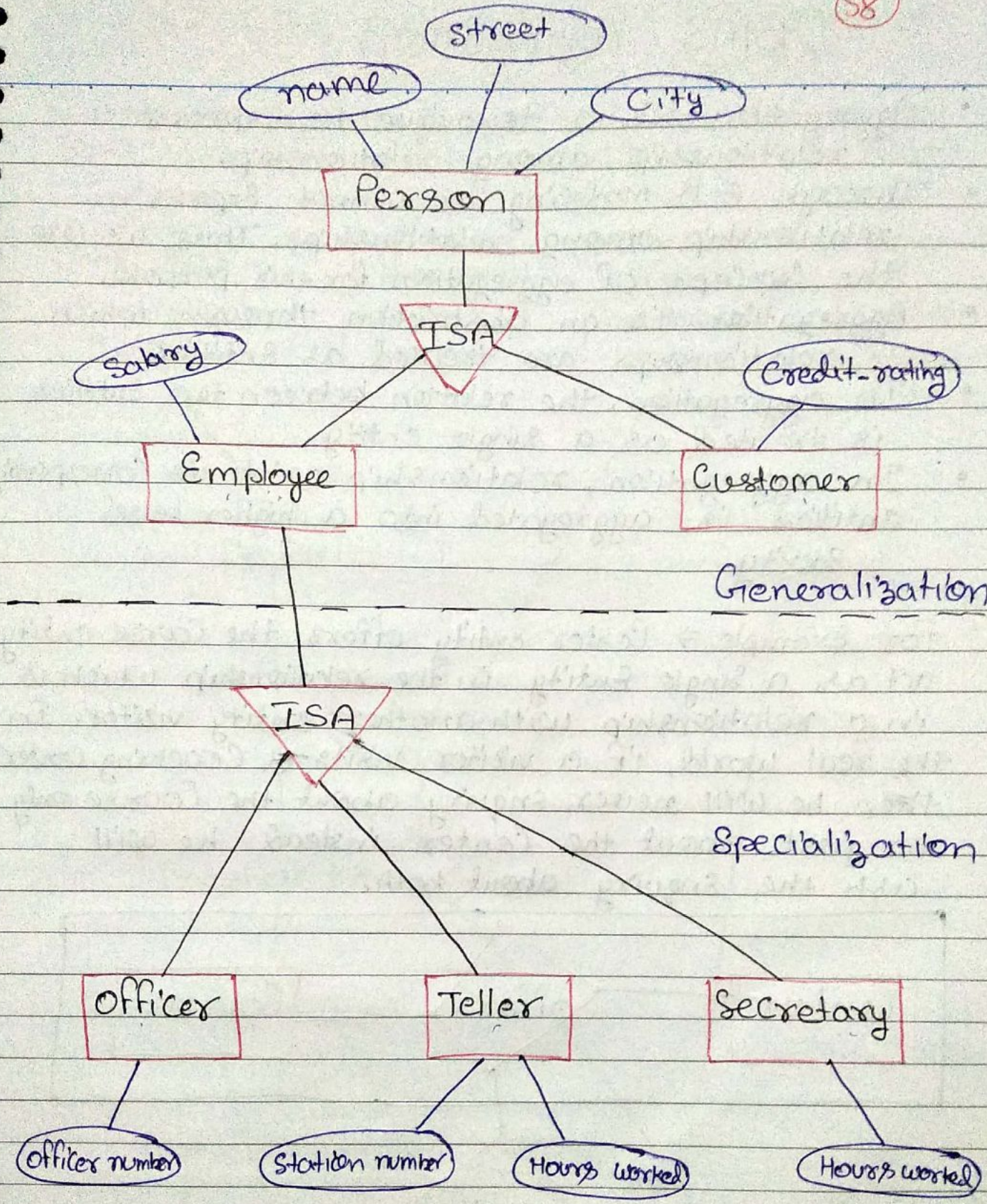


DBMS Specialization

- Specialization is a top-down approach, and It is opposite to Generalization. In Specialization, one higher level Entity can be broken down into two lower level entities.
- Specialization is used to identify the subset of an Entity set that shares some distinguishing characteristics.
- The Specialization Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.
- Specialization is represented by a traingle component labeled ISA. The label ISA stands for "is a" and represents.



for Example ⇒ In the Employee management System, Employee Entity can be Specialized as officer, Teller, Tester, Developer Secretary and Developer based on what role they play in the Company.

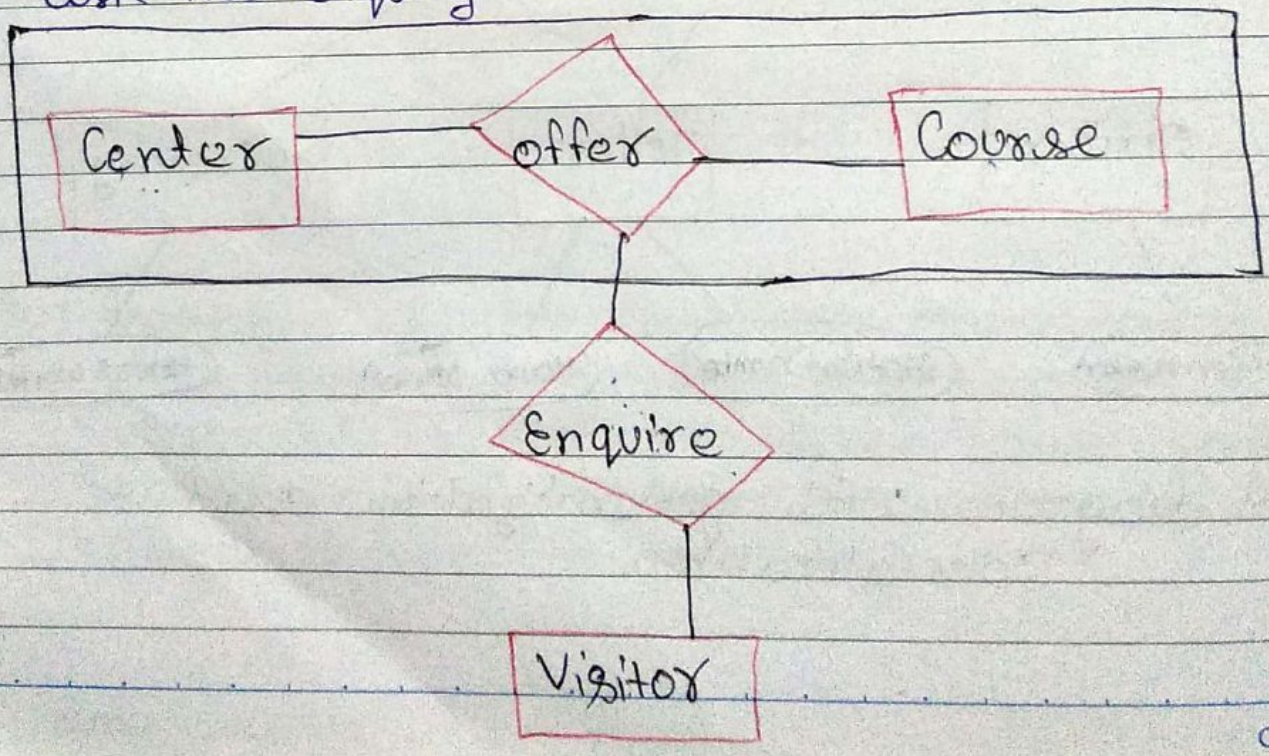


ER diagram with generalization and Specialization.

DBMS Aggregation

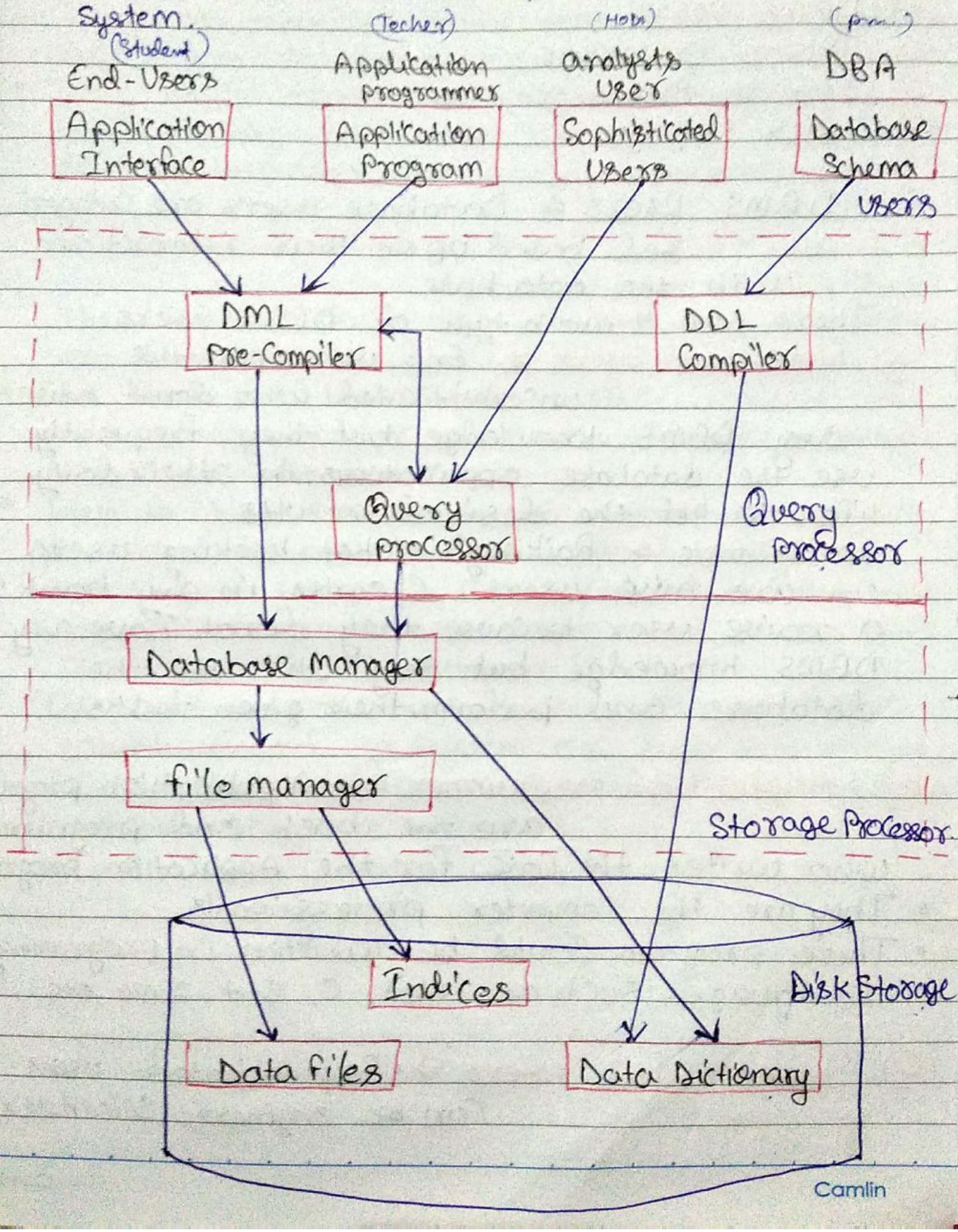
- Aggregation is a technique to express relationship among relationship.
- Through E-R modeling we cannot express relationship among relationships. Thus, we use the concept of aggregation for this purpose.
- Aggregation is an abstraction through which relationships are treated as Entities.
- In aggregation, the relation between two Entities is treated as a single entity.
- In aggregation, relationship with its corresponding entities is aggregated into a higher level Entity.

For Example ⇒ Center Entity offers the Course entity act as a single Entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a Coaching Center then he will never enquiry about the Course only or just about the Center instead he will ask the Enquiry about both.



M.I DBMS Architecture/Structure/Component

A database System is partitional into modules that deal with each of the responsibilities of the overall System.



DBMS architecture divided into four parts.

1) DBMS Users
2) Query Processor
3) Storage Processor
4) Disk Storage.

1) DBMS Users ⇒ Database users are categorized based upon their interaction with the data base.
There are 4 main type of DBMS users.

a) Naive/End users ⇒ End users are the unsophisticated who don't have any DBMS knowledge but they frequently use the database applications in their daily life to get the desired results.
for Example ⇒ Railway's ticket booking users are naive users. Clearks in any bank is a naive user because they donot have any DBMS knowledge but they still use the database and perform their given tasks.

b) Application programmer ⇒ • Application program are the back end programmers who writes the code for the Application program.
• They are the computer professionals.
• These program could be written in programming Languages such as .Net, C, C++, Java etc.

c) Sophisticated Users ⇒ • Sophisticated users can be enginers, Scientists,

business analyst, who are familiar with the database.

- They can develop their own database application according to their requirement.
- They don't write the program code but they interact the data base by writing SQL queries directly through the query processor.

d) Database Administrator (DBA) ⇒ • DBA is a person/team who defines the schema and also controls the 3 Levels of database.

- The DBA will then create a new account id and password for the user if he/she need to access the database.
- DBA is also responsible for providing security to the data base and he allows only the authorized users to access/modify the database.
- DBA monitors the recovery and backup and provide technical support.
- The DBA has a DBA account in the DBMS which called a System or Superuser account.
- DBA repairs damage caused due to hardware and/or software failures.

2) Query Processor ⇒ • It interprets the requests (queries) received from end user via an application program into instructions.
• It also executes the user request which is received from the DML Compiler.
Query Processor Contains the following Component

a) DDL Compiler ⇒ The DDL Statements are sent to DDL Compiler, which Converts these statements to a set of tables. These tables Contains the metadata Concerning the database and are in the form that Can be used by other Components of the DBMS.

b) DML Pre-Compiler and Query Processor ⇒ The DML pre-compiler Converts the DML Statements embedded in an application program to normal procedure Calls in the host Language.
The Query processor Component include:

i) DDL Interpreter ⇒ It processes the DDL Statements into a set of table Containing meta data (data about data).

ii) DML Compiler ⇒ It proceses the DML Statements into Low Level instruction (machine Language), So that they Can be executed.

iii) Query Evaluation Engine ⇒ Which Executes Low-Level instructions generated by the DML Compiler.

3) Storage Manager/ Processor ⇒ • Storage manager is a program that provides an interface between the data stored in the database and the queries received.

- It is also known as database Control System.
- It maintains the Consistency and integrity of the database by applying the Constraints and Executes the DCL statements.
- It is responsible for updating, storing, deleting and retrieving data in the database.

It Contains the following Components:-

a) Autorization Manager ⇒ It tests for the satisfaction of integrity Constraints and checks the authority of users to access data.

b) Transaction Manager ⇒ Which ensures that the database remains in a Consistent (correct) state despite System failures, and that Concurrent transaction execution proceed without conflicting.

c) file manager ⇒ Which manages the allocation of space on disk storage and the data Structures used to represent information Stored on disk.

d) Buffer Manager ⇒ It is responsible for Cache memory and the transfer of data between the secondary Storage and main memory.

4) Disk Storage : It Contains the following Components.

a) Data files ⇒ It Stores the data.

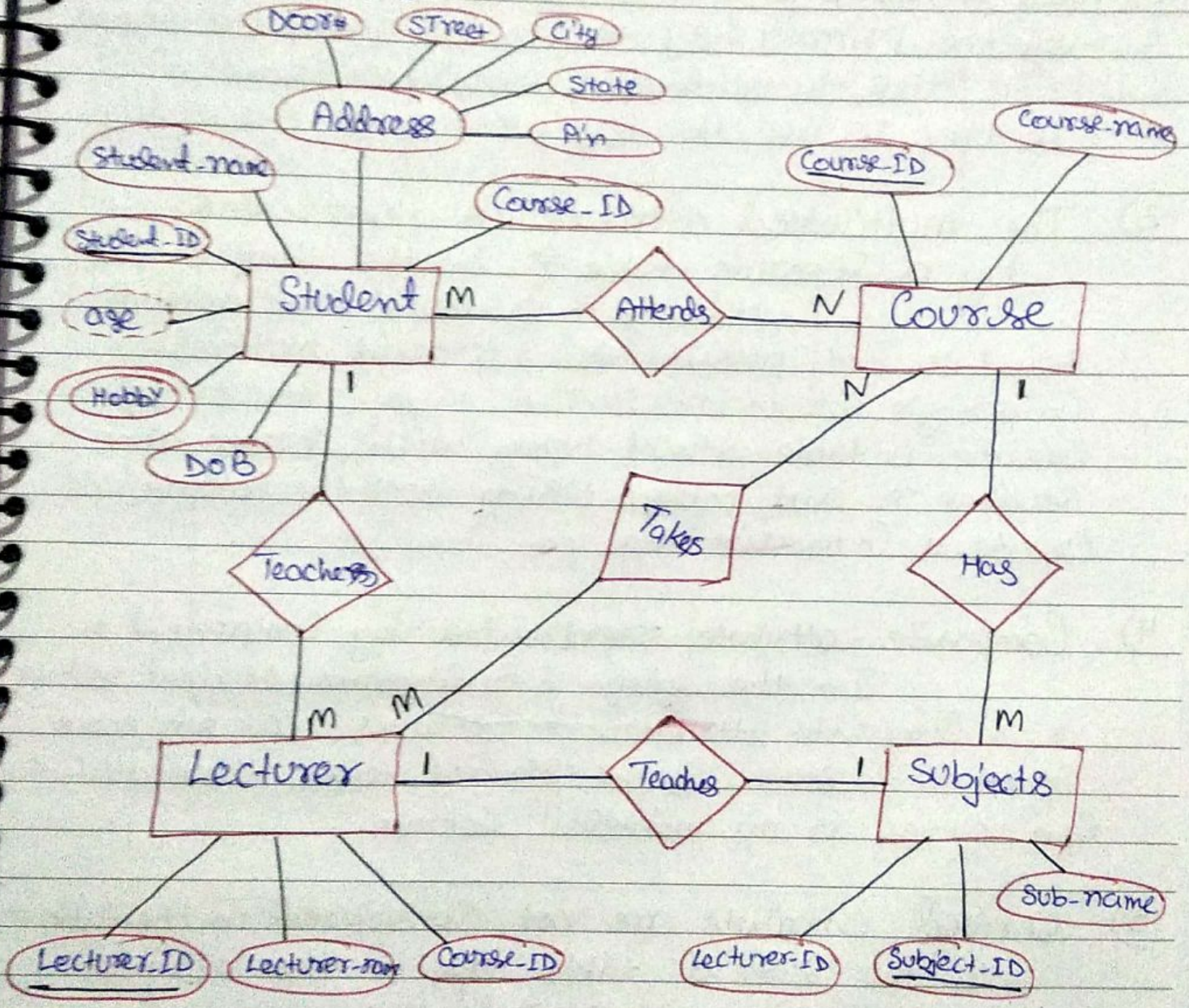
b) Data Dictionary ⇒ • It Contains the information about the Structure of any database object.
• It is the responsitory of information that governs the metadata.

c) Indices ⇒ . It provide fast access to data items that hold particular values.

Reduction of ER Diagram to Table

- The database can be represented using the notations and these notation can be reduced to a collection of tables.
- In the database, every Entity Set or relationship Set can be represented in tabular form.

The E-R diagram is given blow to convert in table.



There are Some points for Converting the ER diagram to the table:

1) Entity type become a table ⇒ In the given ER diagram, Lecturer, Student, Subject and Course form individual tables.

2) Key Attribute of the Entity type represented by the primary key ⇒ In the given ER diagram Course-ID, Student-ID, Subject-ID and Lecturer-ID are the key attribute of the entity.

3) The multivalued attribute is represented by a separate table. ⇒ In the Student table, a hobby is a multivalued attribute. So it is not possible to represent multivalues in a single column of Student table. Hence we Create a table student_hobby with Column name Student-ID and hobby. Using both the Column, we Create a Composite key.

4) Composite attribute represented by Component ⇒ In the given E-R diagram, Student address is a Composite attribute. It Contains City, pin, Door#, Street and State. In the Student table, these attributes Can merge as an individual Column.

5) Derived attribute are not Considered in the table ⇒ In the Student table, age is the derived attribute. It Can be calculated at any point of time by calculating the difference between Current data and Date of Birth.

Camlin

Using these rules, you can Convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:

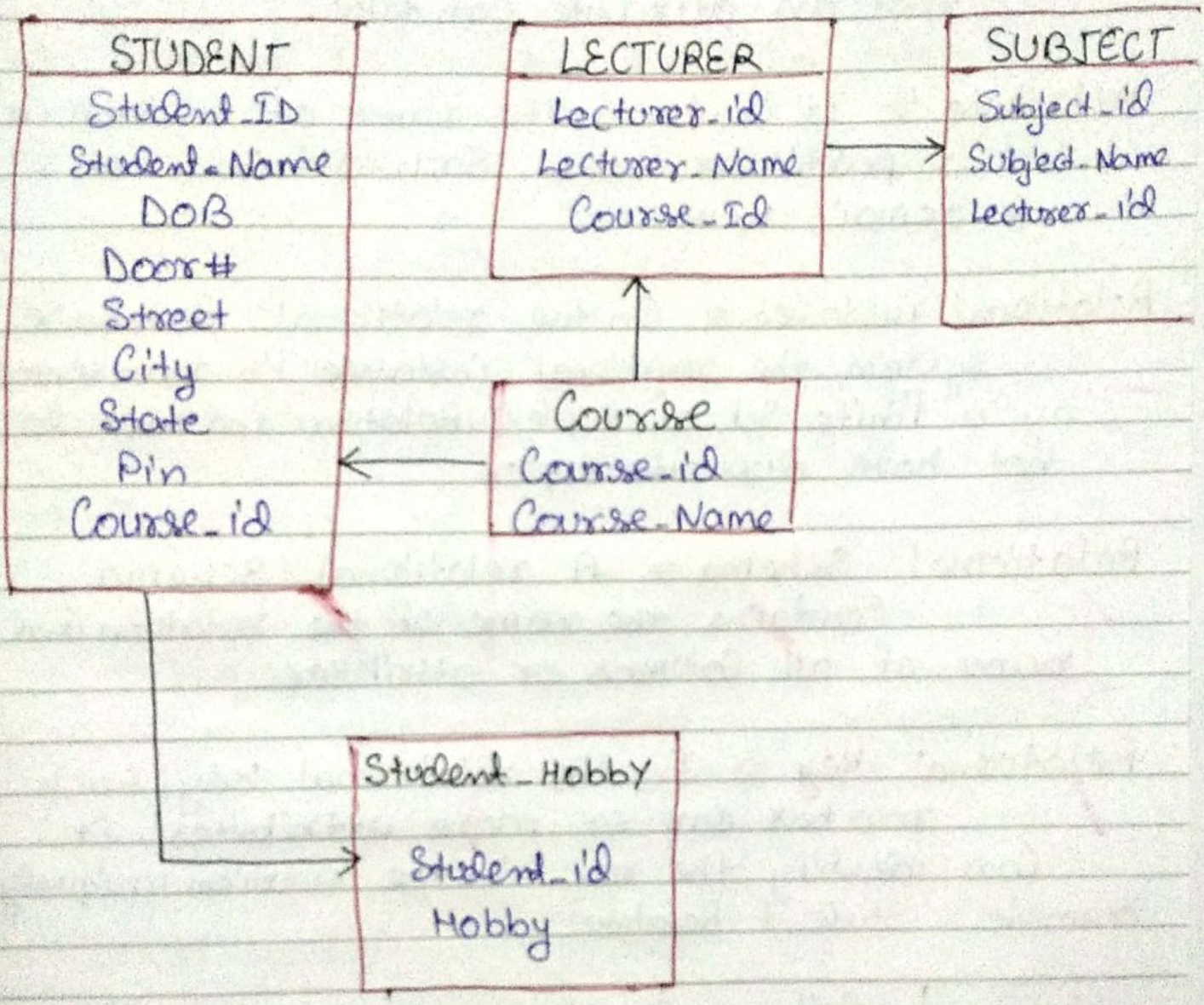


Table Structure

Relational Model Concept

- Relational model can represent as a table with columns and rows.
- Each row is known as a tuple.
- Each table of the column has a name or attribute.

Domain ⇒ It Contains a set of atomic values that an attribute can take.

Attribute ⇒ It Contains the name of a column in a particular table. Each attribute is a domain value.

Relational instance ⇒ In the relational database system, the relational instance is represented by a finite set of tuples, Relation instance do not have duplicate tuples.

Relational Schema ⇒ A relational schema Contains the name of the relation and name of all columns or attributes.

Relational key ⇒ In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

Example: Student Relation

Name	Roll_no	Phone_No	address
Ram	123	34567	Delhi
Kamal	234	25678	Dehradun
Kailash	345	12345	Haridwar

- In the given table, Name, Rollno, Phone-No are the attributes.
- The instance of schema Student has 3 Tuples

Properties of Relations

⇒ Name of the relation is distinct from all other relations.

⇒ Each relation cell contains exactly one atomic (single) value

⇒ Each attribute contains a distinct Name

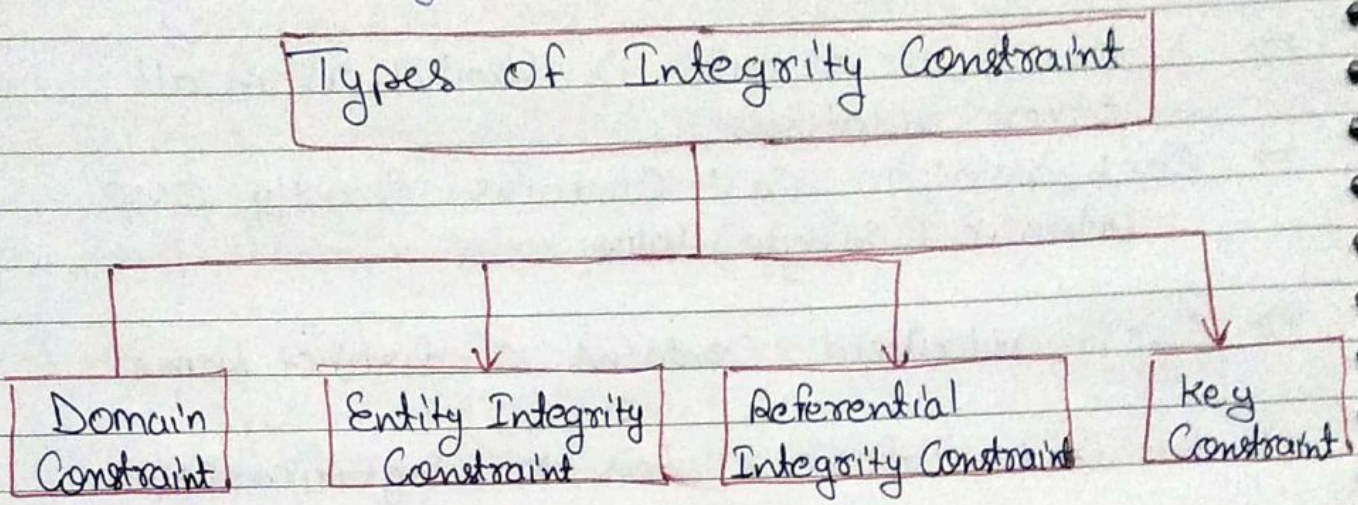
⇒ Attribute domain has no significance

⇒ Tuple has no duplicate value.

⇒ Order of tuple can have a different sequence.

11.1 Integrity Constraints

- Integrity Constraints are a set of rules. It is used to maintain the quality of information.
- Integrity Constraints ensure that the data insertion, updating and other processes have to be performed in such a way that data integrity is not affected.



1) Domain Constraint ⇒ Domain Constraints can be defined as the definition of a valid set of values for an attribute. The datatype of domain includes string, character, integer, time, date etc. The value of the attribute must be available in the corresponding domain.

Ex ⇒

ID	Name	AGE
101	Aman	20
102	Arun	A
103	1	30
104	Karan	35

Not allowed because age is an integer attributes.

Not allowed because Name is an string attribute

2) Entity integrity Constraints ⇒ • The entity integrity Constraint states that primary key value can't be null.

- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example ⇒

P.K

EMP-ID	EMP_Name	Salary
123	Kamal	30000
345	Karan	40000
567	Arun	30000
NULL	Ram	50000

Not Allowed as primary key can't contain a NULL value.

3) Referential Integrity Constraints ⇒ • A referential integrity Constraint is specified between two tables.

- In the Referential integrity Constraints, if a foreign key in Table 1 refers to the Primary key on Table 2, then every value of the foreign key in Table 1 must be null or be available in table 2.

P.K (Table 1) foreign key

E_No	Name	Age	D_No
1	Kamal	20	11
2	Aman	30	24
3	Arun	30	18
4	Ram	25	13

→ Not Allowed as D-No 18 is not Defined as a primary key of table 2 and table 1 D-No is a foreign key defined

Relationships

(Table 2)

P.K

D_No	D_Location
11	Delhi
24	Uttarakhand
13	Mumbai

4) Key Constraints ⇒ • Keys are the Entity set that is used to identify an Entity within its Entity Set uniquely.

- An Entity Set can have multiple keys, but out of which one key will be primary key.
- A primary key can contain a Unique value in the relational table.

P.K

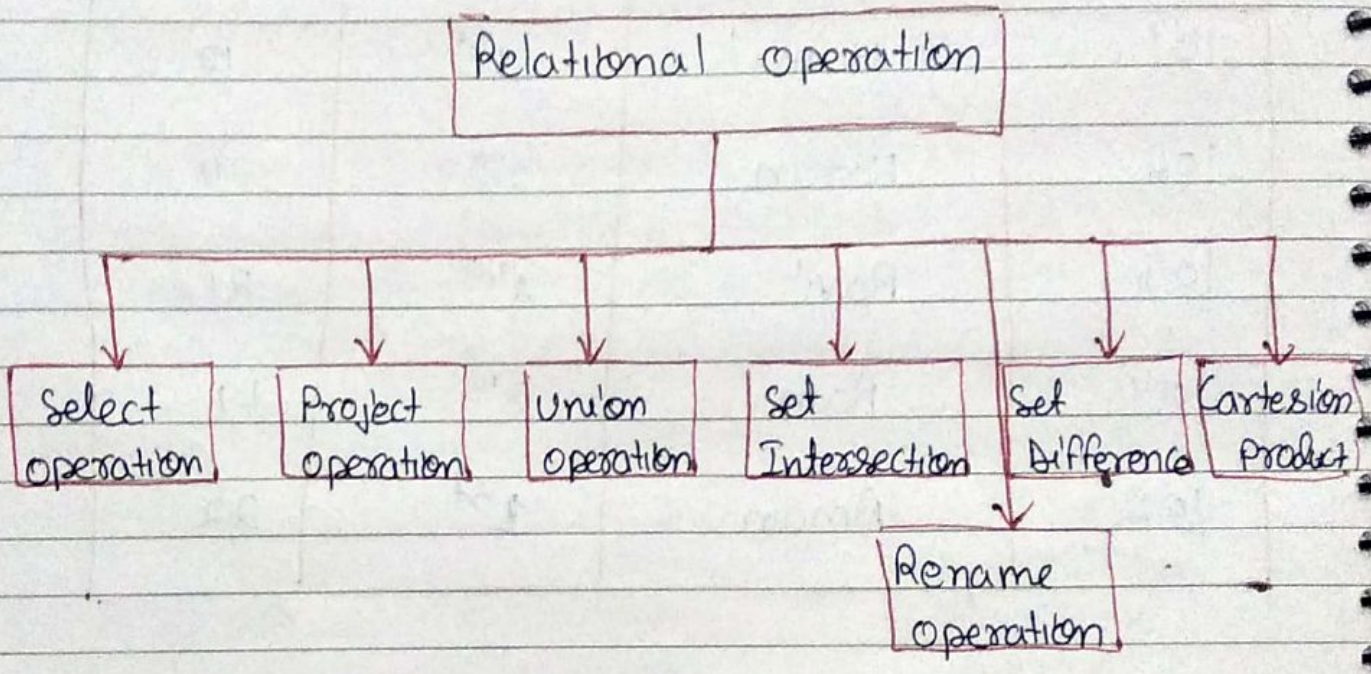
ID	Name	Sem	Age
101	Kamal	1st	17
102	Karan	2nd	24
103	Ravi	2nd	21
104	Ram	3rd	19
102	Aman	1st	22

Not allowed. Because all value of primary key must be Unique

Relational Algebra

- Relational algebra is a procedural query language.
- It gives a step by step process to obtain the result of the query.
- Relational algebra mainly provides theoretical foundation for relational databases and SQL
- It uses operators to perform queries.

Types of Relational operation.



1) Select operation ⇒ (σ) • The Select operation is used to select a subset of the tuples from a relation that satisfies a select condition.
   - σ (Sigma) is used to denote Select operator
   - The Select operator is univory such that it is applied to a single relation.

The general format of select operation is

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

Where σ is used for selection prediction

R is used for Relation

Select condition is the relational operators like $=, \neq, \geq, <, \leq$.

Ex ⇒ Student

Name	Rollno.	address
Aman	02	Dehradun
Karan	04	Rishikesh
Arun	08	Delhi
Ankit	13	Bombay

query 1 ⇒ Give all information of student having Rollno is 04.

Solution ⇒ $\sigma_{\text{Rollno}=04}(\text{student})$

query 2 ⇒ find all information of student having Name is Arun and address is delhi

Solution ⇒

$\sigma_{\text{Name}=\text{"Arun"} \text{ and address}=\text{"Delhi"}}(\text{student})$

or

$\sigma_{(\text{Name}=\text{"Arun"}) \vee (\text{address}=\text{"Delhi"})}(\text{student})$

2) Project Operation ⇒ (Π) ⇒ • Project operation selects certain Columns from the table and discard the other columns.

- This operation shows the list of those attributes that we wish to appear in the result.
- Rest of the attributes are Eliminated from the table.

The general format for Project operation is

$$\Pi_{<\text{attribute List}>}(R)$$

where Π is the Symbol used to represent the project operation and attribute list is the list of attributes from the attributes of Relation R.

Ex ⇒ Student

Name	Rollno.	address
Aman	02	Dehradun
Karan	04	Rishikesh
Arun	08	Delhi
Ankit	13	Bombay

query 01 ⇒ find Student Name in the Student table.

Solution ⇒ $\Pi_{\text{Name}}(\text{student})$

query 02 ⇒ find student Name and address List.

Solution ⇒ $\Pi_{Name, address}(student)$

3) UNION operation ⇒ (∪) ⇒ It performs binary union between two given Relations and is define as.

$$R \cup S$$

Where R and S are either database relations or relation result set (temporary relation).

Union operation to be valid, the following conditions must hold -

- R and S must have the same number of attributes.
- Attribute domains must be Compatible.
- Duplicate tuples are automatically Eliminated.

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

4) Set Intersection (∩) ⇒ It performs binary Intersection between two given Relations and is define as:-

$$R \cap S$$

Where R and S are either database relations or relation result set.

Ex ⇒ $\prod_{name}(Student1) \cap \prod_{name}(Student2)$

5) Set Difference (−) ⇒ The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

$$R - S$$

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

Ex ⇒ $\prod_{name}(Student1) - \prod_{name}(Student2)$

6) Cartesian Product (X) ⇒ Combines information of two different relations into One.

$$R \times S$$

Where R and S are relations and

Their output will be define as -

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

Ex ⇒

$$\sigma_{Name = 'kamal'} (Student1 \times Student2)$$

7) Rename Operation (ρ) ⇒ Rename operation is used to rename the output of a relation. rename operation is denoted with Small Greek Letter rho (ρ)

$$\rho_x(E)$$

Where the result of Expression E is saved with name of x.

Ex ⇒ Query to rename the table Student to Employee and its attributes name, address, phoneno.

$$\rho_{Employee(name, address, phoneno)} (student)$$

Normalization

Functional Dependency

- The functional Dependency is a relationship that Exists between two attributes.
- It typically Exists between the primary key and non-key attributes within a table.

$$X \longrightarrow Y$$

The left side of FD is known as a determinard, The right side of the production is known as a dependent.

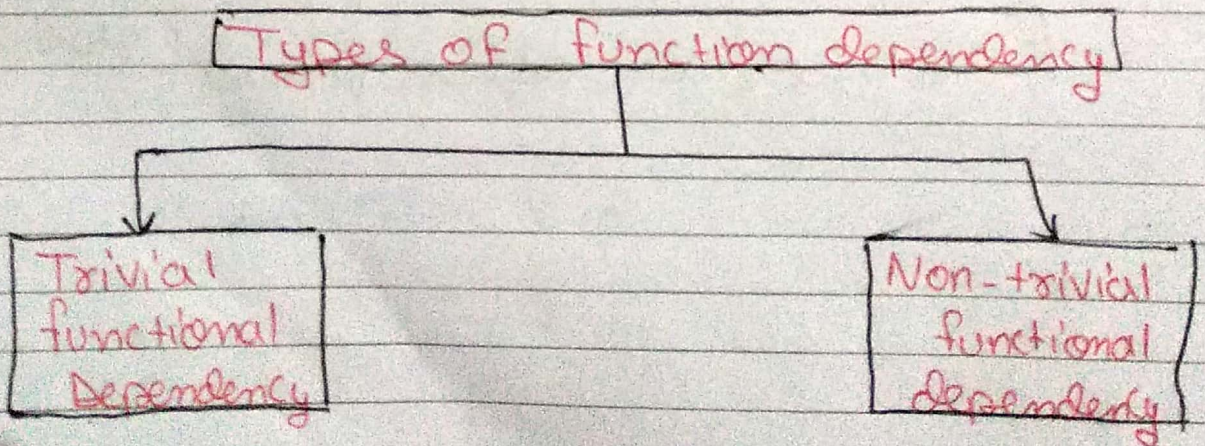
for Example ⇒ Assume we have an Employee table with attributes: Emp-Id, Emp-Name, Emp-Address

Here Emp-Id attribute can uniquely identify the Emp-Name attribute of Employee table because if we known the Emp-Id, we can tell that Employee name associated with it.

functional dependency can be written as:

Emp-Id ⟶ Emp-Name

We can say that Emp-Name is functionally dependent on Emp-Id.



Trivial functional dependency ⇒

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

Ex ⇒ Consider a table with two columns Employee_id and Employee_Name.
{Employee_id, Employee_Name} → Employee_id is a trivial functional dependency as Employee_id is a subset of {Employee_id, Employee_Name}
Also, Employee_id → Employee_id and Employee_Name → Employee_Name are trivial dependencies too.

Non-Trivial functional Dependency ⇒

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

Ex ⇒ Consider a table with three Column Emp_id, Emp_Name and Emp_age
Emp_id → Emp_Name is a Non trivial functional dependency because Emp_Name is not a subset of Emp_id.
Similarly
Emp_id, Emp_Name → Emp_Age is non-trivial because Emp_age is not a subset of Emp_id, Emp_Name

Closure of a Set of functional Dependencies

Inference Rule (IR)

- The inference rule is a type of assertion. It can apply a set of FD (functional dependency) to derive other FD.
- Using the inference Rule, we can derive additional functional dependency from the initial set.

1) Reflexive Rule (IR_1) ⇒ In the Reflexive Rule, if Y is a subset of X, then X determines Y.

if $Y \subseteq X$ then $X \rightarrow Y$

Example ⇒

$X = \{a, b, c, d, e\}$
$Y = \{a, b, c\}$

2) Augmentation Rule (IR_2) ⇒ The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

if $X \rightarrow Y$ then $XZ \rightarrow YZ$

Ex ⇒

for R(ABCD)

if $A \rightarrow B$ then

$AC \rightarrow BC$

3) Transitive Rule (IR_3) ⇒ In the transitive Rule, if x determines Y and Y determine Z, then X must also determine Z.

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

4) Union Rule (IR_4) ⇒ Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

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

$X \longrightarrow YZ$

5) Decomposition Rule (IR_5) ⇒ Decomposition Rule is also known as project rule. It is the reverse of union rule.
This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

if $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

6) Pseudo transitive Rule (IR_6) ⇒ In Pseudo transitive Rule, if X determine Y and YZ determine W, then XZ determines W.

if $X \rightarrow Y$ and $YZ \rightarrow W$ then
$XZ \rightarrow W$.

Q. Find the Candidate key of Relation R(A, B, C, D, E, F) with functional dependency

$A \to C$
$C \to D$
$D \to B$
$E \to F$

(Set of attributes whose Closure contains all attributes of given relation)

Sol:

S.K → $ABCDEF^+ = \{A, B, C, D, E, F\}$

$A \to C$

S.K → $ABDEF^+ = \{A, B, C, D, E, F\}$

$A \to C \quad C \to D$
$A \to D$ { transitive dependency }

S.K → $ABEF^+ = \{A, B, C, D, E, F\}$

$A \to C \quad C \to D \quad D \to B$
$A \to B$ transitive dependency

S.K → $AEF^+ = \{A, B, C, D, E, F\}$

$E \to F$

$\boxed{AE^+} = \{A, B, C, D, E, F\}$

Candidate key

Prime attribute = (A, E) if no prime attribute available to Right hand side of any function dependency so it has only one Candidate key (AE^+)

Ex ⇒ find the possible candidate key of the Relation R(A, B, C, D) with functional dependency A → B, B → C, C → A

Soln ⇒ S.K $ABCD^+$ ⟶ {A, B, C, D}

S.K ACD^+ ⟶ {A, B, C, D}

S.K. AD^+ ⟶ {A, B, C, D}

AD is Candidate key

AD Prime attribute A, D, Prime attribute A available in Right Side of function dependency C → A so another candidate key CD

again C is available in Righ Side of functional dependency B → C so another Candidate key

BD

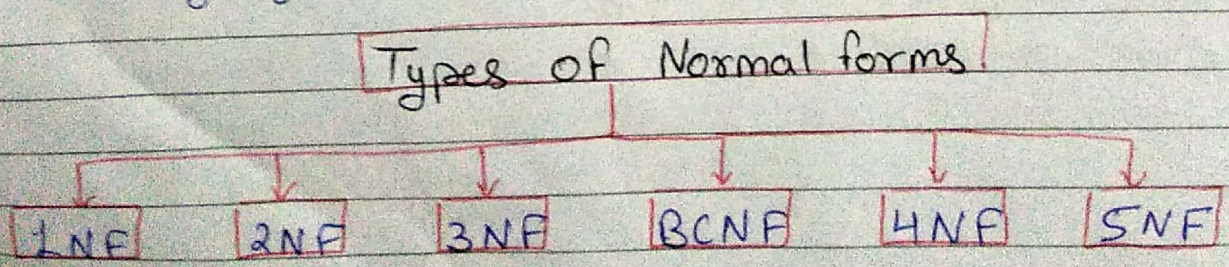
So Candidatkey = AD, CD, BD.

Normalization

- Normalization is the process of organizing the data in the database.
- Normalization is used to minimize the redundancy from a relation or set of relations
- It is used to eliminate undesirable characteristics like insertion, Update and Deletion Anomalies
- Normalization divides the larger table into smaller and links them using relationships.
- The normal form is used to reduce redundancy from the database table.

When we normalize the database, we have four goals:

1) Arranging data into logical groupings such that each group describes a small part of the whole.
2) minimizing the amount of duplicate data, called redundancy, stored in a database.
3) organizing the data such that, when you modify it, you make the changes only in one place.
4) Building a database in which you can access and manipulate the data quickly and efficiently without compromising the integrity of the data in storage.



1NF (first Normal form) ⇒ • A relation is in 1NF if it contains an atomic values.
• It Eliminate Repeating Groups.

2NF (second Normal form) ⇒ • A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key.
• It Eliminate Partial functional dependency.

3NF (Third Normal form) ⇒ • A Relation will be in 3NF if it is in 2NF and no transitive dependency Exists.
• It Eliminate Transitive dependency.

BCNF (Boyce Codd's normal form) ⇒ A Stronger Definition of 3NF is known as Boyce Codd's normal form.
• It is also called 3.5 NF

4NF (fourth Normal form) ⇒ • A relation will be in 4NF if it is in Boyce Codd's Normal form and has no multi-valued dependency
• Eliminate multi-Values Dependency.

5NF (fifth Normal form) ⇒ • A relation is in 5NF if It is in 4NF and does not Contain any
• join dependency, joining should be Lossless.
• Eliminate Join Dependency.

Advantages & Disadvantages of Normalization

Advantages of Normalization

- Normalization helps to minimize data redundancy.
- Greater overall database organization.
- Data consistency within the database.
- Much more flexible database design.
- Fewer indexes per table ensure faster maintenance tasks (index rebuilds).
- Relizes the option of joining only the tables that are needed.
- A better handle on database security.
- Increase storage efficiency.
- Speed up data access.

Disadvantages of Normalization

- You cannot start building the database before knowing what the user needs.
- Requires much CPU, memory and I/O process thus normalized data gives reduced database performance
- Requires more joins to get the desired result. A poorly-written query can bring the database down.
- It is very time-consuming and difficult to normalize relations of a higher degree.
- Careless decomposition may lead to a bad database design, leading to serious problems.

First Normal form (1NF)

- A relation will be 1NF if it contains an atomic value.
- It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attribute.
- First Normal form disallow the multi-valued attribute, Composite attribute and their combinations.

Ex ⇒ Relation Student is not in 1NF because of multi-valued attribute ~~Std~~ Phone.

Student table:

Stud_id	Stud_name	Stud_Phone	Stud_Branch
11	Kamal	7276854823, 3214172829	IT
12	Karan	8762547890	CS
13	Ravi	32456278, 34563578	ES
14	Ram	8176345630	ME
15	Komal	3456873200	IT

The Decomposition of the Student table into 1NF as:

Stud_id	Stud_name	Stud_Phone	Stud_Branch
11	Kamal	7276854823	IT
11	Kamal	3214172829	IT
12	Karan	8762547890	CS
13	Ravi	32456278	ES
13	Ravi	34563578	ES
14	Ram	8176345630	ME
15	Komal	3456873200	IT

Second Normal form (2NF)

- In the 2NF, relation must be in 1NF.
- No attributes of the table should be functionally dependent on only one part of a Concatenated primary key.
- In the second normal form, all non-key attributes are fully functional dependent on the primary key.

Ex ⇒ 2NF is based on the Concept of full functional Dependency $X \longrightarrow Y$ is a fully function Dependency (FFD) if removal of any attribute (A) from X means that the dependency does not hold any more.

In the Book-order table such as

order No.	title	Qty	Unit Price
1	Computer Network	1	250
1	Java	1	275
1	DBMS	2	295
2	multimedia	1	300
2	Data Structure	1	190
3	DBMS	1	295
3	multimedia	2	300
3	Computer Network	5	250

It is not in 2NF because it hold Partical function Dependency and fully function dependency.

title ⟶ Unit Price

P.F.D.

and

OrderNo, Title → Qty } F.F.D.

OrderNo $\xrightarrow{\times}$ Qty

Title $\xrightarrow{\times}$ Qty

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

Order_master table

Order No.	Title	Qty
1	Computer Network	1
1	Java	1
1	DBMS	2
2	Multimedia	1
2	Data Structure	1
3	Multimedia	2
3	Computer Network	5

Book_Master table

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

Third Normal form (3NF)

- A relation is in 3NF, if it is in 2NF and no non-prime attribute functionally dependent on other non-prime attributes.
- A relation is in 3NF, if it is in 2NF and no attributes of the table should be transitively functionally dependent on the primary key.
- 3NF is used to reduce the data duplication. It is also used to achieve the data integrity.
- If there is no transitive dependency for non-prime attributes, then the relation must be in third normal form.

A relation is in third normal form if it holds atleast one of the following conditions for every non-trivial function dependency $X \rightarrow Y$.

1. X is Super key
2. Y is a prime attribute such as each element of Y is part of some Candidate key.

Ex ⇒ Employee_Department_Location table

EmpNo	E_Name	Sal	Depart_Name	Depart_Location
1001	Vishal	7500	Accounts	102
1002	Amit	5000	Sales	104
1003	Anuj	10000	Accounts	102
1004	Vikas	4500	Sales	104
1005	Sumit	6500	Store	106

Table Not in 3NF because it hold the transitive dependency.

EmpNo → Dept_Name → Dept_Location

EmpNO → Dept_Location

To make it in 3NF we decompose and remove the transitive dependency. so we convert the given table in 3NF decompose two sub table Such as :-

(Table1) Employee_Department

EmpNO	E_Name	Sal	Dep_Name
1001	Vishal	7500	ACCount
1002	Amit	5000	Sales
1003	Anuj	10000	ACCounts
1004	Vikas	4500	Sales
1005	Sumit	6500	Store

Table 2 Department_Location

Dept_Name	Dept_Location
ACCounts	102
Sales	104
Store	106

Table converted in 3NF.

Boyce Codd Normal Form (BCNF)

- BCNF is the advance version of 3NF. It is stricter than 3NF.
- A table is in BCNF if every functional dependency $X \rightarrow Y$, X is the super key of the table.
- For BCNF, the table should be in 3NF, and for every FD, LHS is Super key.

Example:- Let's assume there is a company where employees work in more than one department.

Employee table

Emp_id	Emp_Country	Emp_Dept	Dept_type	Emp_Dept_No
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

In the above table Functional dependencies are as follows:-

Emp_id $\rightarrow$ Emp_Country

Emp_Dept $\rightarrow$ {Dept_type, Emp_Dept_No}

Candidate key: {~~Emp_type~~, Emp_id, Emp_Dept}

the table is not in BCNF because neither Emp_dept nor Emp_id alone are keys.

To convert the given table into BCNF, we decompose it into three tables.

Emp_Country table:

Emp_id	Emp_Country
264	India
364	UK

Emp_Dept table:

Emp_Dept	Dept_type	Emp_Dept_No
Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

Emp_Dept_mapping table:

Emp_id	Emp_Dept
D394	283
D394	300
D283	232
D283	549

functional dependencies:

Emp_id → Emp_Country

Emp_Dept → {Dept_type, Emp_Dept_No}

Candidate keys:

for the first table: Emp_id

for the second table: Emp_Dept

for the third table: {Emp_id, Emp_Dept}

Now, this is in BCNF because left side part of both the functional dependencies is a key.

Fourth Normal form (4NF)

- A relation will be in 4NF if it is in Boyce codd normal form (BCNF) and has no multi-valued dependency.
- MVD (multi-value dependency) occurs when two or more independent multi-valued facts about the same attribute occurs within the same relation.
- MVD is denoted by

$$X \rightarrow\rightarrow Y$$

It will be readers "there is a multi-valued dependency of Y" or "X multi-determines Y".

Example ⇒ Faculty

faculty	Subject	Committee
Kailash	DBMS	Placement
Kailash	Java	Placement
Kailash	C	Placement
Kailash	DBMS	Scholarship
Kailash	Java	Scholarship
Kailash	C	Scholarship

The given faculty table is in 3NF, but the Subject and Committee are two independent Entity. Hence there is no relationship between Subject and Committee.

In the faculty relation, a faculty with faculty name Kailash Contains three Subject DBMS, Java and C, and two Committee placement

and Scholarship. So there is a multi-valued dependency on faculty name, which leads to unnecessary repetition of data.

Kailash → → Placement
Kailash → → Scholarship

So to make the above table into 4NF, we can decompose it into two tables:

Table 1 faculty_Course

faculty	Subject
Kailash	DBMS
Kailash	Java
Kailash	C

Table 2 faculty - Committee

faculty	Committee
Kailash	Placement
Kailash	Scholarship

Fifth Normal form (5NF)

- The 5NF (fifth Normal form) is also known as project-join Normal form.
- A relations is in fifth Normal form (5NF), if it is in 4NF, and won't have Lossless decomposition into Smaller tables.
- 5NF is satisfied when all the tables are broken into a many tables as possible in order to avoid redundancy After that you combined these all tables if it is Equal to original table then 5NF.
- You can also Consider that a relation is in 5NF, if the Candidate key implies every join dependency in it.

Example ⇒ faculty (original table)

faculty	Subject	Committee
Kailash	DBMS	Placement
Kailash	Java	placement
Kailash	C	placement
Kailash	DBMS	Scholarship
Kailash	Java	Scholarship
Kailash	C	Scholarship

the given table is not in 4NF and 5NF first we convert it in 4NF with Converting it two Sub table.

Table1 faculty - Subject

faculty	Subject.
Kailash	DBMS
Kailash	Java
Kailash	C

Table 2 faculty - Committee

faculty	Committee
Kailash	Placement
Kailash	Scholarship

To Convert it in SNF, we join both table1 and table2 if it give the result same as original table (faculty) then it's in SNF otherwise not in SNF.

Table1 + Table2

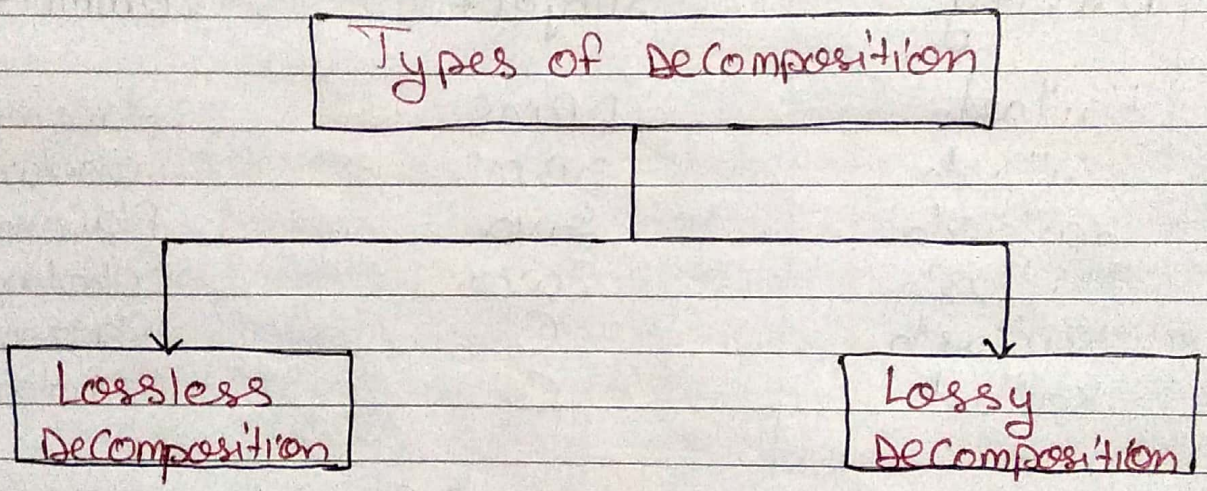
faculty	Subject	Committee
Kailash	DBMS	Placement
Kailash	DBMS	Scholarship
Kailash	Java	Placement
Kailash	Java	Scholarship
Kailash	C	Placement
Kailash	C	Scholarship

It's Equal to original tal

So it is in SNF

Relational Decomposition

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies and redundancy.



Lossless Decomposition ⇒

- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless.
- The lossless decomposition guarantees that the join of relations will result in the same relation as it was decomposed.

- The relation is said to be lossless decomposition if natural joins of all the decomposition give the original relation.

Ex ⇒ Emp_Info

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_N
E001	Kamal	29	Haridwar	Dpt 1	Operation
E002	Karan	32	Dehradun	Dpt 2	HR
E003	Ravi	22	Delhi	Dpt 3	finance

Decompose the above table into two tables:

1) EmpDetails

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Kamal	29	Haridwar
E002	Karan	32	Dehradun
E003	Ravi	22	Delhi

2) Dept Details

Dept_ID	Emp_ID	Dept_Name
Dpt 1	E001	Operation
Dpt 2	E002	HR
Dpt 3	E003	finance

Now, Natural Join is applied on the above two tables :
The result will be

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Kamal	29	Haridwar	Dpt1	Operations
E002	Karan	32	Dehradun	Dpt2	HR
E003	Ravi	22	Delhi	Dpt3	finance

Therefore, the above relation had lossless decomposition there is no loss of information if we join all Decompose tables.

Lossy Decomposition ⇒ When a relation is decomposed into two or more relational schemas, the loss of information unavoidable when the original relation is retrieved.
Let us see an example -
Ex ⇒

EmpInfo

Emp_ID	Emp_Name	Emp_Age	Emp_Location	Dept_ID	Dept_Name
E001	Kamal	29	Haridwar	Dpt1	Operations
E002	Karan	32	Dehradun	Dpt2	HR
E003	Ravi	22	Delhi	Dpt3	finance

Decompse the table into two tables -

~~EmpInfo~~ EmpDetails>

Emp_ID	Emp_Name	Emp_Age	Emp_Location
E001	Kamal	29	Haridwaar
E002	Karan	32	Dehradun
E003	Ravi	22	Delhi

<Dept Details>

Dept_ID	Dept_Name
Dpt1	Operations
Dpt2	HR
Dpt 3	Finance

Now, you won't be able to join the above tables, since Emp_ID is not part of DeptDetails relation.

Therefore, the above relation has Lossy Decomposition.

Q> Consider the Schema S = (V, W, X, Y, Z) Suppose the following F.D hold:

$Z \rightarrow V$

$W \rightarrow Y$

$XY \rightarrow Z$

$V \rightarrow WX$

State whether the following decomposition Scheme S is lossless join decomposition or Loss decomposition.

i) $S_1 = (V, W, X)$
$S_2 = (V, Y, Z)$

ii) $S_1 = (V, W, X)$
$S_2 = (X\ Y\ Z)$

Ans=> R_1 & R_2 are two relation i's lossless decomposition if

$R_1 \cap R_2 \longrightarrow R_1$

$R_1 \cap R_2 \longrightarrow R_2$

i) $S_1 = (V, W, X)$

$S_2 = (V, Y, Z)$

$S_1 \cap S_2 = V \longrightarrow S_1$

$V \rightarrow S_2$

$V \rightarrow V$
$V \rightarrow WX$ }

$V \rightarrow VWX$

$V \rightarrow S_1$

So the decomposition

$S_1 = (V, W, X)$

$S_2 = V, Y, Z)$ it's lossless.

ii) $S_1 = (V, W, X)$

$S_2 = (X, Y, Z)$

$S_1 \cap S_2 \rightarrow X \rightarrow S_1 \rightarrow VWX$

$X \rightarrow S_2 \rightarrow XYZ$

$X \rightarrow X$

$X \not\rightarrow W$

$X \not\rightarrow Y$

$X \not\rightarrow Z$

Lossy decomposition because

X not determine V, WX or XYZ

$X \not\rightarrow VWX$

$X \not\rightarrow XYZ$ Lossy decomposition

TRANSACTION SYSTEM

- Collection of operations that form a single logical unit of work are called transaction.
- A trasaction is a unit of program execution that accesses and possibly updates various data items.
- Trasaction is define as a logical unit of database processing that includes one or more database access operations.

Transaction access data using two operations:

1) Read(X) :⇒ Read operation is used to read the value of Account x from the database and stores it in a buffer in main memory.

2) Write(X) :⇒ Write operation is used to write the value back to the database from the buffer.

Let's take an Example to debit trasaction from an account which Consists of following operations: X=1000 Y=1000

1) R(X);
2) X = X-500;
3) W(X)

T_1	T_2
read(X)	read(Y)
X = X-500	Y = Y+500
write(X)	write(Y)
(X = 500)	Y = 1500

Let's assume the value of X before Starting of trasaction is 4000.

- The first operation read X's value from database and Stores it in a buffer.
- The Second operation will decrease the value of X by 500. So buffer will Contain 3500.
- The third operation will write the buffer's value to the database. So X's final value will be 3500.

But it may be possible that because of the failure of hardware, Software or power, etc. that trasaction may fail before finished all the operations in the set.

for Example:⇒ If the given trasaction, the debit trasaction fails after Executing operation 2 then X's value will remain 400 in the database which is not acceptable by the bank.

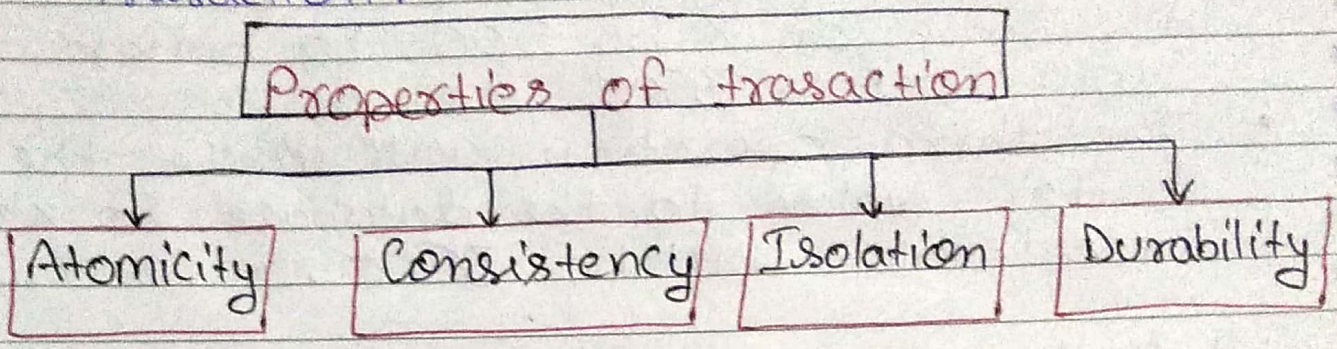
To Solve this problem, we have two important operations:

1) Commit:⇒ It is used to Save the work done permanently.

2) Rollback,⇒ It is used to undo the work done.

ACID properties of Transaction

Transactions have four basic properties, which are called ACID properties.
These are used to maintain Consistency in a database, before and after the trasaction.



1) Atomicity ⇒ • It States that all operations of the transaction take place at once if not, the transaction is aborted.
• There is no midway such as 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 the following operations:-

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

Example:⇒ Let's assume that following transaction T Consisting of T_1 and T_2. A Consists of Rs 600 and B Consists of Rs 300. Transfer Rs 100 from Account A to Account B

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

After Completion of the transaction A Consists of Rs 500 and B Consists of Rs 400.

If the transaction T fails after the Completion of transaction T_1 but before Completion T2, then the amount will be deducted from A but not added to B. This shows the inconsistent database state. In order to ensure Correctness of database State, the transaction must be executed in entirety.

2) Consistency ⇒
- The integrity Constraints are maintained so that the database is Consistent before and after the transaction.
- The Execution of a transaction will Leave a database in either its prior stable State or a new stable State.
- The Consistency property of database States that every transaction sees a Consistent database instance.
- The transaction is used to transform the database from one Consistent State to another Consistent State.

for Example ⇒ Let's assume that following transaction T consisting of T1 and T2. A Consists of Rs 600 and B Consists of Rs 300. Transfer Rs 100 from Account A to Account B.

T1	T2
Read (A)	Read (B)
A = A - 100	B = B + 100
Write (A)	Write (B)

The total amount must be maintained before or after the transaction.

Total before T occurs = 600 + 300 = 900
Total after T occurs = 500 + 400 = 900

Therefore, the database is consistent. In the case when T1 is completed but T2 fails, then inconsistency will occur.

3) Isolation ⇒ • 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.
• 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 the transaction T1 ends.

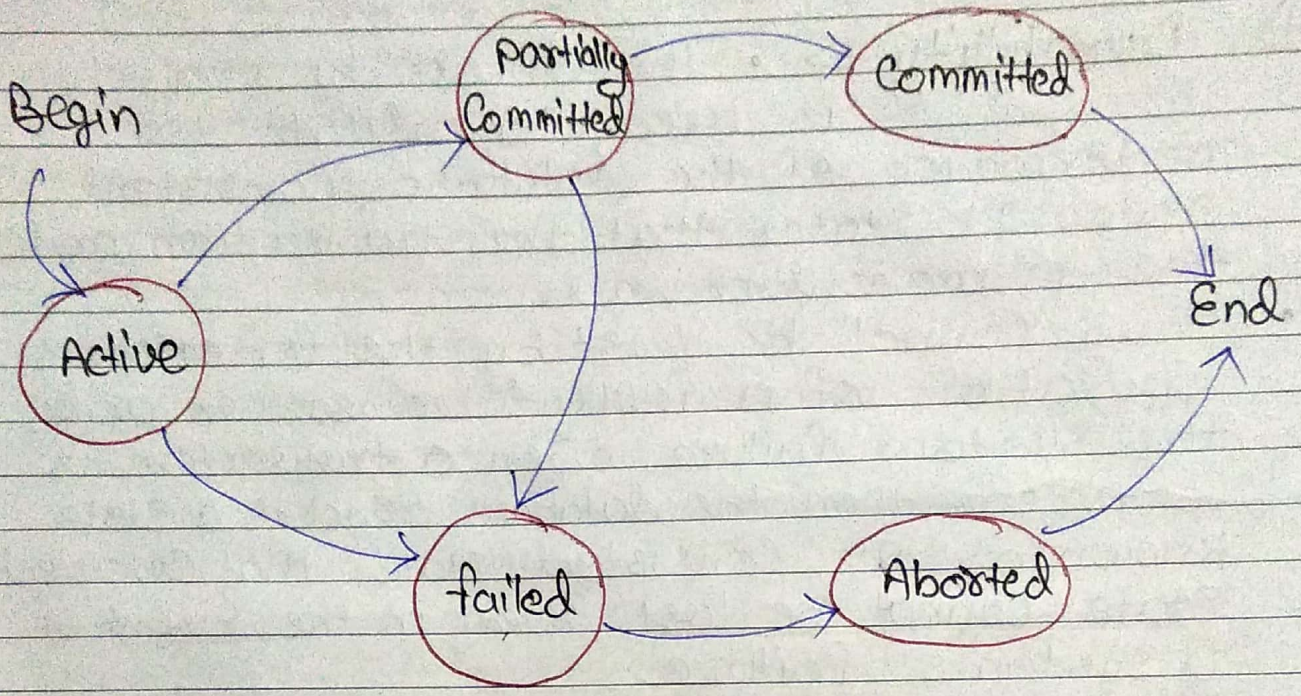
Camlin

- The Concurrency Control Subsystem of the DBMS enfored the isolation property.

4) Durability ⇒ • The durability property is used to indicate the performance of the database's Consistent State. It States that the transaction made the permanent changes.
- They cannot be Lost by the erroneous operation of a faulty transaction or by the system failure. when a transaction is Completed, then the database reaches a State known as the Consistent State. that Consistent State Cannot be lost, even in the Event of a System's failure.
- The recovery Subsystem of the DBMS has the responsibility of Durability property

States of Transaction

In a database, the transaction can be in one of the following states:-



1) Active State ⇒ • It is the first state of every transaction. In this state, the transaction is being executed.
• for Example ⇒ Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

2) Partially Committed ⇒ • In the partially Committed state, a transaction executes its final operation, but the data is still not saved to the database.
• In the total mark Calculation Example, a final display of the total marks step is executed in this state.

3) Committed ⇒ • A transaction is said to be in a committed state if it executes all its operations successfully.
• In this state, all the effects are now permanently saved on the database System.

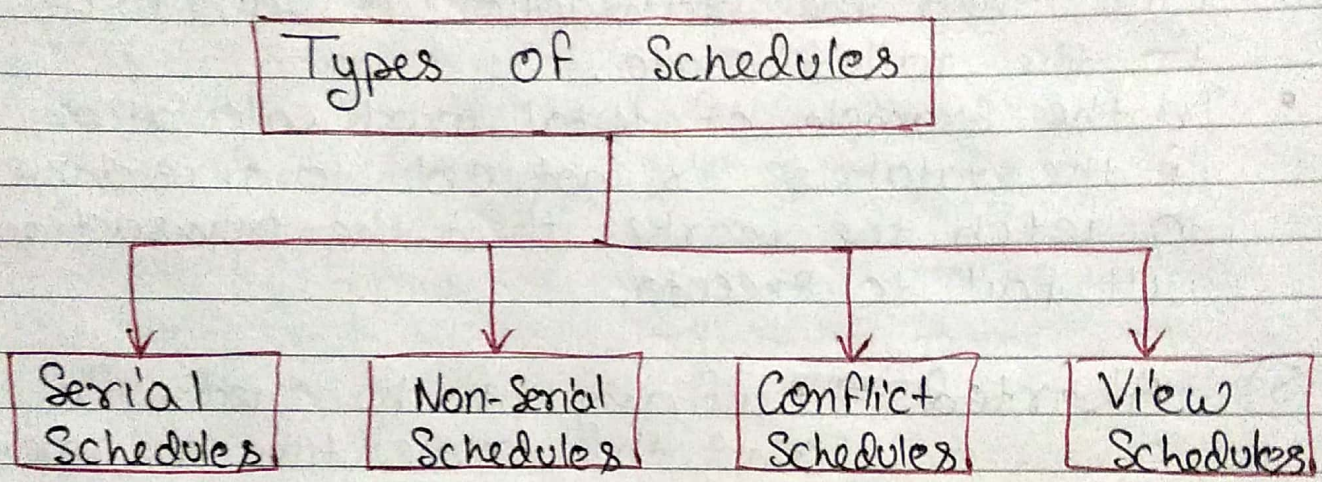
4) failed state ⇒ • If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
• In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

5) Aborted ⇒ • if any of the checks fail and the transaction has reached a failed state then the database recovery System will make sure that the database is in its previous consistent state. If not then it will abort or rollback the transaction to bring the database into a consistent state.
• After aborting the transaction, the database recovery module will select one of the two operations:-
1) Re-start the transaction
2) Kill the transaction

Serializability of Schedules

Schedule ⇒ When several transaction are executing concurrently then the order of execution of various instruction is known as a schedule.

The concept of serializability of schedules is used to identify which schedules are connected when transaction executions have interleaving of their operations in the schedules.



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

T_1	T_2
read_item(X);	
X = X - N;	
write_item(X);	
read_item(Y);	

Y = Y+N;
write_item(Y);
	read_item(X);
	X = X+m;
	write_item(X);

Schedule A

The Schedules A is called serial scheduler. entire transaction are perform in serial order T_1 and then T_2 or T_2 and Then T_1.

2) Non-serial Schedules ⇒ • if interleaving of operations is allowed, then there will be non-serial schedule.

• A Schedules are called non-serial Schedule if the operations of each transaction are executed non-consecutively with interleaved operations from the other transaction.

T_1	T_2
read_item(X); X = X+N;	
	read_item(X); X = X+m;
write_item(X); read_item(Y);	
	write_item(X);
Y = Y+N; write_item(Y);	

3) Conflict Schedules ⇒ • A Schedules is called Conflict Serializablity if after Swapping of non-Conflicting operations, it can transform into a Serial Schedule.

• The schedule will be a Conflict Serializable if it is Conflict Equivalent to a Serial Schedule.

Conflicting Operations

The two operations become Conflicting if all conditions satisfy:

1. Both belong to Separate transactions.
2. They have the same data item
3. They contain at least one write operation.

Example ⇒ 1) Swapping is possible only if S1 and S2 are logically Equal

T_1	T_2
Read(A)	
	Read(A)

Schedule S1

Swapped →

T_1	T_2
~~Read(A)~~	Read(A)
Read(A)	

Schedule S2

Here S1 = S2 that means it is non-Conflict.

2)

T	T_2
Read(A)	
	Write(A)

Swapped ⟹

T_1	T_2
	Write(A)
Read(A)	

Here $S_1 \neq S_2$ that means it is Conflict.

4) View Serializability / Schedule ⇒ • A Schedule will View Serializable if it is View Equivalent to a Serial Schedule.
- if a schedule is Conflict Serializable, then it will be View Serializable.
- The View Serializable which does not Conflict Serializable Contains blind writes

Two Schedules S1 and S2 are Said to be View Equivalent if they Satisfy the following Conditions:-

1) Initial Read ⇒ An initial read of both Schedules must be the same. Suppose two Schedule S1 and S2. In Schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A

T_1	T_2
read(A)	
	write(A)

Schedule S1

T_1	T_2
	write(A)
Read(A)	

Schedule S2

Above two Schedules are View Equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

2) Update Read ⇒ In Schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated

by T_j.

T_1	T_2	T_3
Write(A)		
	Write(A)	
		Read(A)

Schedule S1

T_1	T_2	T_3
	Write(A)	
Write(A)		
		Read(A)

Schedule S2

Above two Schedules are not view Equal because, in S1, T3 is reading A updated by T2 and in S2, T3 is reading A updated by T1.

3) final write ⇒ A final write must be the same between both the Schedules. In Schedule S1, if a transaction T1 update A at last then in S2, final writes operations should also be done by T1.

T_1	T_2	T_3
Write(A)		
	Read(A)	
		Write(A)

T_1	T_2	T_3
	Read(A)	
Write(A)		
		Write(A)

Above two schedules is view Equal because final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

TRANSACTION SYSTEM

- Collection of operations that form a single logical unit of work are called transaction.
- A trasaction is a unit of program execution that accesses and possibly updates various data items.
- Trasaction is define as a logical unit of database processing that includes one or more database access operations.

Transaction access data using two operations:

1) Read(X) :⇒ Read operation is used to read the value of Account x from the database and stores it in a buffer in main memory.

2) Write(X) :⇒ Write operation is used to write the value back to the database from the buffer.

Let's take an example to debit trasaction from an account which consists of following operations: X=1000 Y=1000

		T_1	T_2
1)	R(X) ;	read(X)	read(Y)
2)	X = X-500;	X = X-500	Y = Y+500
3)	W(X)	write(X)	write(Y)
		(X = 500)	Y = 1500

Let's assume the value of X before starting of trasaction is 4000.

- The first operation read X's value from database and Stores it in a buffer.
- The second operation will decrease the value of X by 500. So buffer will Contain 3500.
- The third operation will write the buffer's value to the database. So X's final value will be 3500.

But it may be possible that because of the failure of hardware, Software or power, etc. that trasaction may fail before finished all the opercations in the set.

for Example:⇒ If the given trasaction, the debit trasaction fails after Executing operation 2 then X's value will remain 400 in the database which is not acceptable by the bank.

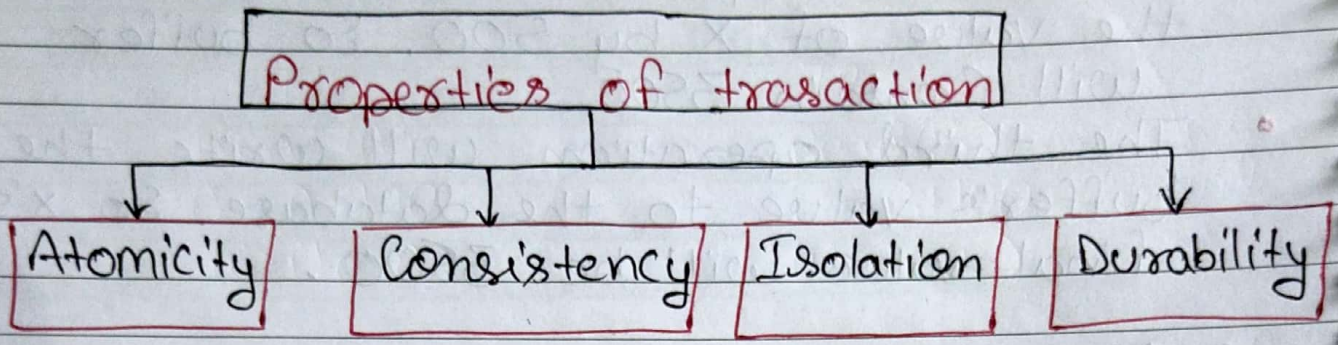
To Solve this problem, we have two important operations:

1) Commit:⇒ It is used to Save the work done permanently.

2) Rollback,⇒ It is used to undo the work done.

ACID properties of Transaction

Transactions have four basic properties, which are called ACID properties. These are used to maintain Consistency in a database, before and after the trasaction.



1) Atomicity ⇒ • It States that all operations of the transaction take place at once if not, the transaction is aborted.
• There is no midway such as 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 the following operations:-

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

Example:⇒ Let's assume that following transaction T Consisting of T_1 and T_2. A Consists of Rs 600 and B Consists of Rs 300. Transfer Rs 100 from Account A to Account B

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

After Completion of the transaction A Consists of Rs 500 and B Consists of Rs 400.

If the transaction T fails after the Completion of transaction T_1 but before Completion T2, then the amount will be deducted from A but not added to B. This shows the inconsistent database state. In order to ensure Correctness of database state, the transaction must be executed in entirety.

2) Consistency ⇒
- The integrity Constraints are maintained so that the database is Consistent before and after the transaction.
- The Execution of a transaction will Leave a database in either its prior stable state or a new stable state.
- The Consistency property of database States that every transaction sees a Consistent database instance.
- The transaction is used to transform the database from one Consistent State to another Consistent State.

for Example ⇒ Let's assume that following transaction T Consisting of T1 and T2. A Consists of Rs 600 and B Consists of Rs 300. Transfer Rs 100 from Account A to Account B.

T1	T2
Read (A)	Read (B)
A = A - 100	B = B + 100
Write (A)	Write (B)

The total amount must be maintained before or after the transaction

Total before T occurs = 600 + 300 = 900
Total after T occurs = 500 + 400 = 900

Therefore, the database is Consistent. In the Case when T1 is Completed but T2 fails, then inconsistency will occur.

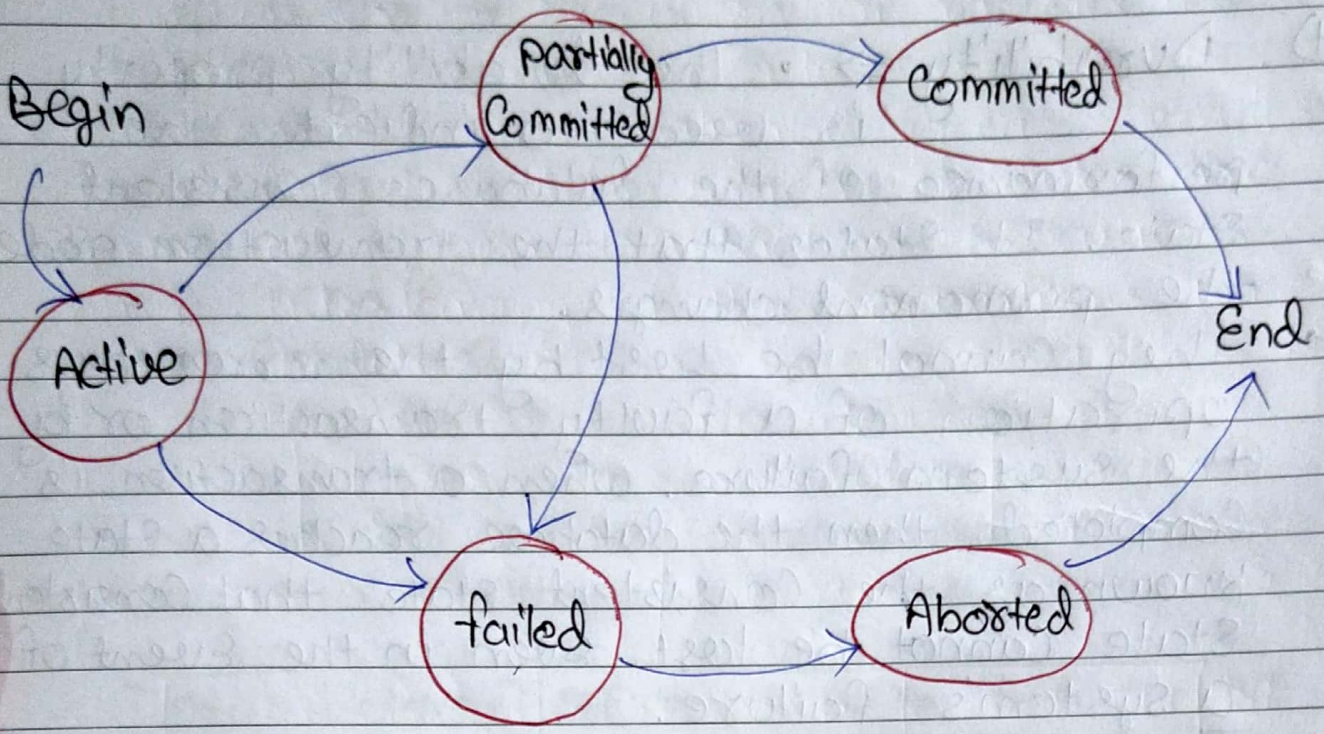
3) Isolation ⇒ • It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.
• In isolation, if the transaction T1 is being Executed and using the data item x, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.

- The Concurrency Control Subsystem of the DBMS enfored the isolation property.

4) Durability ⇒ • The durability property is used to indicate the performance of the database's Consistent State. It States that the transaction made the permanent changes.
- They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. when a transaction is Completed, then the database reaches a state known as the Consistent state. that Consistent State cannot be lost, even in the Event of a system's failure.
- The recovery subsystem of the DBMS has the resposibility of Durability property.

States of Transaction

In a database, the transaction can be in one of the following states:-



1) Active State ⇒ • It is the first state of every transaction. In this state, the transaction is being executed.
• for Example ⇒ Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

2) Partially Committed ⇒ • In the partially Committed state, a transaction executes its final operation, but the data is still not saved to the database.
• In the total mark Calculation Example, a final display of the total marks step is executed in this state.

3) Committed ⇒ • A transaction is said to be in a committed state if it executes all its operations successfully.
• In this state, all the effects are now permanently saved on the database System.

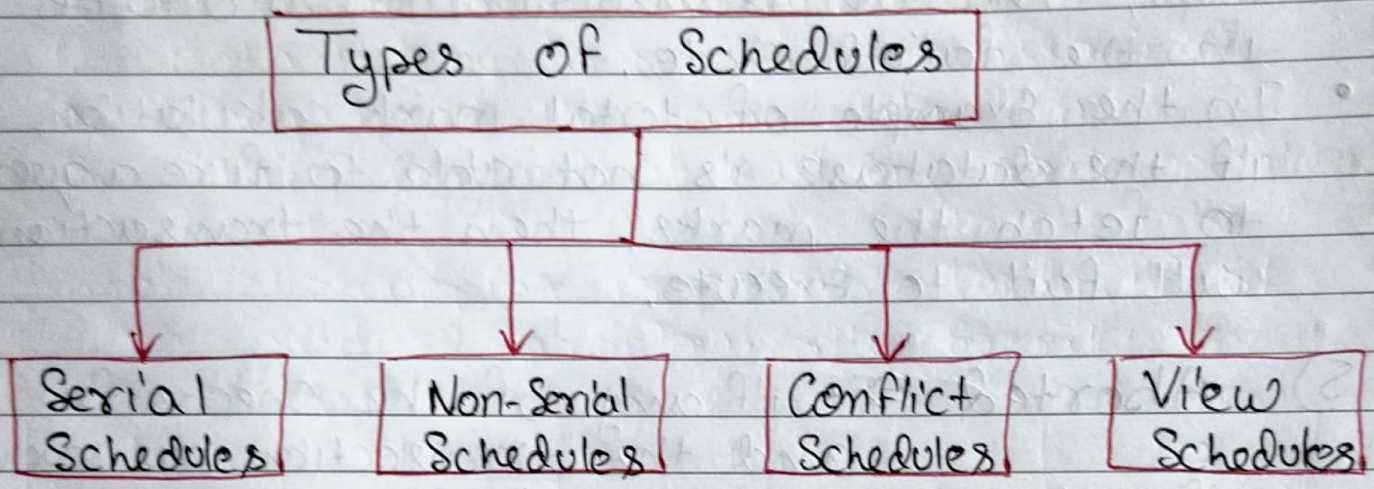
4) failed state ⇒ • If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
• In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

5) Aborted ⇒ • if any of the checks fail and the transaction has reached a failed state then the database recovery System will make sure that the database is in its previous Consistent state. If not then it will abort or rollback the transaction to bring the database into a Consistent state.
• After aborting the transaction, the database recovery module will select one of the two operations:-
1) Re-Start the transaction
2) Kill the transaction

Serializability of Schedules

Schedule ⇒ When several transaction are executing concurrently then the order of execution of various instruction is known as a schedule.

The concept of serializability of schedules is used to identify which schedules are connected when transaction executions have interleaving of their operations in the schedules.



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

T_1	T_2
read_item(X);	
X = X - N;	
write_item(X);	
read_item(Y);	

	read_item(X);
Y = Y+N;	X = X+m;
write_item(Y);	write_item(X);

Schedule A

The Schedules A is called serial scheduler. entire transaction are perform in serial order T_1 and then T_2 or T_2 and Then T_1.

2) Non-serial Schedules ⇒ • if interleaving of operations is allowed, then there will be non-serial schedule.

• A Schedules are called non-serial Schedule if the operations of each transaction are Executed non-Consecutively with interleaved operations from the other transaction.

T_1	T_2
read_item(X);	
X = X+N;	
	read_item(X);
	X = X+m;
write_item(X);	
read_item(Y);	
	write_item(X);
Y = Y+N;	
write_item(Y);	

3) Conflict Schedules ⇒ • A Schedules is called Conflict Serializablity if after swapping of non-conflicting operations, it can transform into a serial schedule.

• The schedule will be a conflict serializable if it is conflict equivalent to a serial schedule.

Conflicting Operations

The two operations become conflicting if all conditions satisfy:

1. Both belong to separate transactions.
2. They have the same data item
3. They contain at least one write operation.

Example ⇒ 1) Swapping is possible only if S1 and S2 are logically equal

T_1	T_2
Read(A)	
	Read(A)

Schedule S1

Swapped →

T_1	T_2
~~Read(A)~~	Read(A)
Read(A)	

Schedule S2

Here S1 = S2 that means it is non-conflict.

2)

T	T_2
Read(A)	
	Write(A)

Swapped ⟹

T_1	T_2
	Write(A)
Read(A)	

Here $S_1 \neq S_2$ that means it is conflict.

4) View Serializability / Schedule ⇒ • A Schedule will view Serializable if it is view Equivalent to a serial Schedule
- if a schedule is Conflict Serializable, then it will be view Serializable.
- The view Serializable which does not Conflict Serializable Contains blind writes.

Two Schedules S1 and S2 are said to be view Equivalent if they Satisfy the following Conditions:-

1) Initial Read ⇒ An initial read of both Schedules must be the same. Suppose two Schedule S1 and S2. In Schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A

T_1	T_2
read(A)	
	write(A)

Schedule S1

T_1	T_2
	write(A)
Read(A)	

Schedule S2

Above two Schedules are view Equivalent because Initial read operation in S1 is done by T1 and in S2 it is also done by T1.

2) Update Read ⇒ In Schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated

by T_3.

T_1	T_2	T_3
Write(A)		
	Write(A)	
		Read(A)

Schedule S1

T_1	T_2	T_3
	Write(A)	
Write(A)		
		Read(A)

Schedule S2

Above two Schedules are not view Equal because, in S1, T3 is reading A updated by T2 and in S2, T3 is reading A updated by T1.

3) final write ⇒ A final write must be the same between both the Schedules. In schedule S1, if a transaction T1 update A at last then in S2, final writes operations should also be done by T1.

T_1	T_2	T_3
Write(A)		
	Read(A)	
		write(A)

T_1	T_2	T_3
	Read(A)	
Write(A)		
		Write(A)

Above two schedules is view Equal because final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.

Testing of Serializability

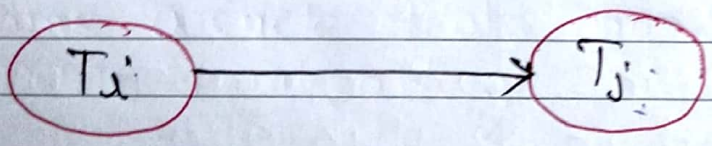
Serialization Graph is used to test the Serializability of a schedule.
for testing of serializability the simple and effecient method is to Construct a directed graph, Called a precedence graph from S

$$G = (V, E)$$

Where V Consists a Set of vertices and E Consists a set of edges. The set of vertices is used to Contain all the transactions participating in the Schedule.
The set of edges is used to Contain all edges $T_i \rightarrow T_j$ for which one of the three conditions holds:

1: Create a node $T_i \rightarrow T_j$ if T_i executes write (Q) before T_j executes read (Q).
2: Create a node $T_i \rightarrow T_j$ if T_i executes read (Q) before T_j Executes write (Q).
3: Create a node $T_i \rightarrow T_j$ if T_i Executes write (Q) before T_j execute write (Q).

Precedence graph for Schedule S



- A precedence graph Contains a single edge $T_i \rightarrow T_j$, then all the instructions of T_i are executed before the first instruction of T_j is executed.

- A precedence graph for Schedule S Contain a Cycle, then S is non-serializable. If the precedence graph has no Cycle, the S is known as serializable.

for Example ⇒

T_1	T_2	T_3
Read(A)		
	Read(B)	
$A := f_1(A)$		
		Read(C)
	$B := f_2(B)$	
	write(B)	
		$C := f_3(C)$
		write(C)
write(A)		
		Read(B)
	Read(A)	
	$A := f_4(A)$	
Read(C)		
	write(A)	
$C := f_5(C)$		
write(C)		
		$B := f_6(B)$
		write(B)

Schedule SL

Explanation:

Read(A): In T_1, no subsequent writes to A, so no new edges.
Read(B): In T_2, no subsequent writes to B, so no new edges.
Read(C): In T_3, no subsequent write to C, so no new edges.

Write(B): B is Subsequently read by T_3, so add edge $T_2 \rightarrow T_3$.

Write(C): C is Subsequently read by T_1, so add edge $T_3 \rightarrow T_1$.

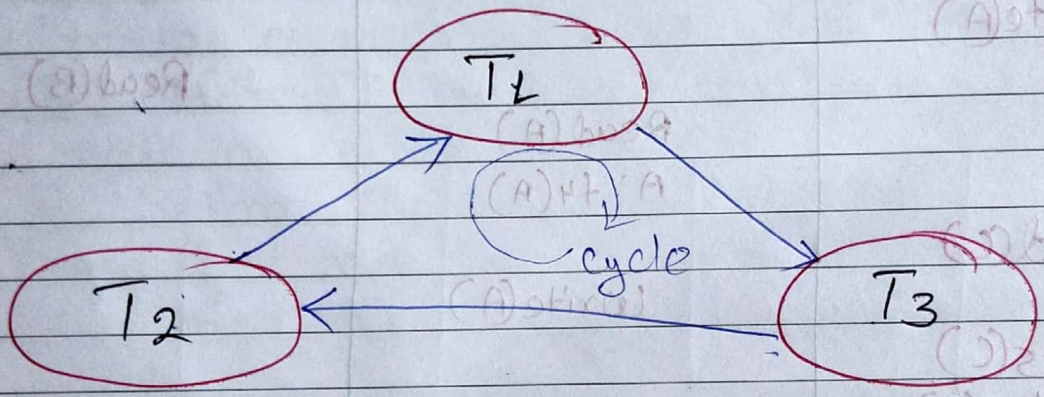
Write(A): A is Subsequently read by T_2, so add edge $T_1 \rightarrow T_2$

Write(A): In T_2, no Subsequent reads to A, so no new edges.

Write(C): In T_1, no Subsequent read to C, so no new edges.

Write(B): In T_3, no subsequent read to B, so no new edges.

Precedence graph for Schedule S1:



The precedence graph for Schedule S1 Contains a cycle that's why Schedule S1 is non-Serializable.

Deadlock in DBMS

- A deadlock is a condition where two or more transactions are waiting indefinitely for one another to give up locks.
- A System is said to be in deadlock state when in a set of transactions, every transaction is waiting for another transaction to finish its operations.
- A deadlock is a state of statement that may result when two or more transactions are each waiting for locks held by the other to be released.

for Example ⇒ In the Student table, transaction T1 holds a lock on some rows and needs to update some rows in the grade table. Simultaneously, transaction T2 hold locks on some rows in the grade table and needs to update the rows in the Student table held by transaction T1.

Now the main problem arises. Transaction T1 is waiting for T2 to release its locks and Similarly, transaction T2 is waiting for T1 to release its lock: All activities come to a halt state and remain at a standstill.