Department of Computer Science & Engineering

DBMS
UNIT-1

1. Overview of Database Management System:

Database:

The database is a collection of interrelated data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

For example: The college Database organizes the data about the admin, staff, students and faculty etc.

Using the database, you can easily retrieve, insert, and delete the information.

Database Management System:

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 provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.

It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.
DBMS allows users the following tasks:

Data Definition: It is used for creation, modification, and removal of definition that defines the organization of data in the database.

Data Updation: It is used for the insertion, modification, and deletion of the actual data in the database.

Data Retrieval: It is used to retrieve the data from the database which can be used by applications for various purposes.

User Administration: It is used for registering and monitoring users, maintaining data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

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 contains ACID properties which maintain data in a healthy state in case of failure.
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.

Advantages of DBMS over File Processing System:

Controls database redundancy: It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in 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 failures and restores the data if required.

multiple user interface: It provides different types of user interfaces 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 software.

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.

DBMS - Architecture:

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

1. **1-Tier architecture:**

   In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end- users. Database designers and programmers normally prefer to use single-tier architecture.

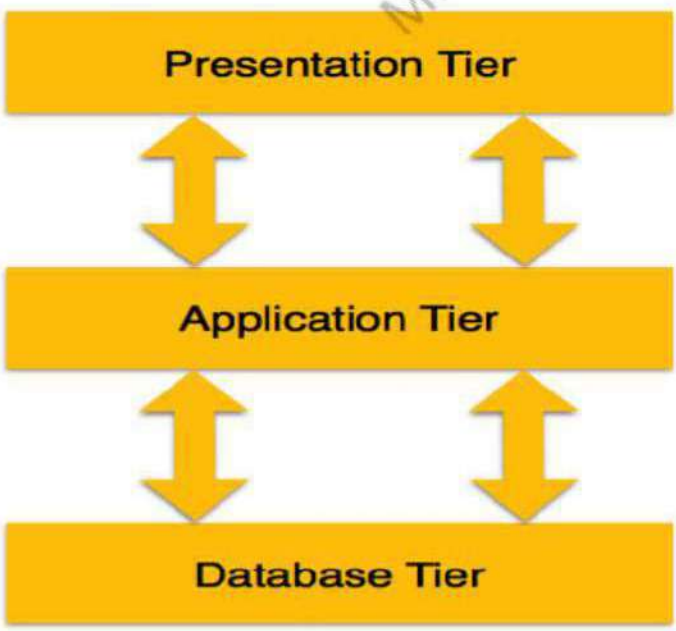
2. **2-Tier architecture:**

   If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed.

   Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3. **3-Tier architecture:**

   A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS -

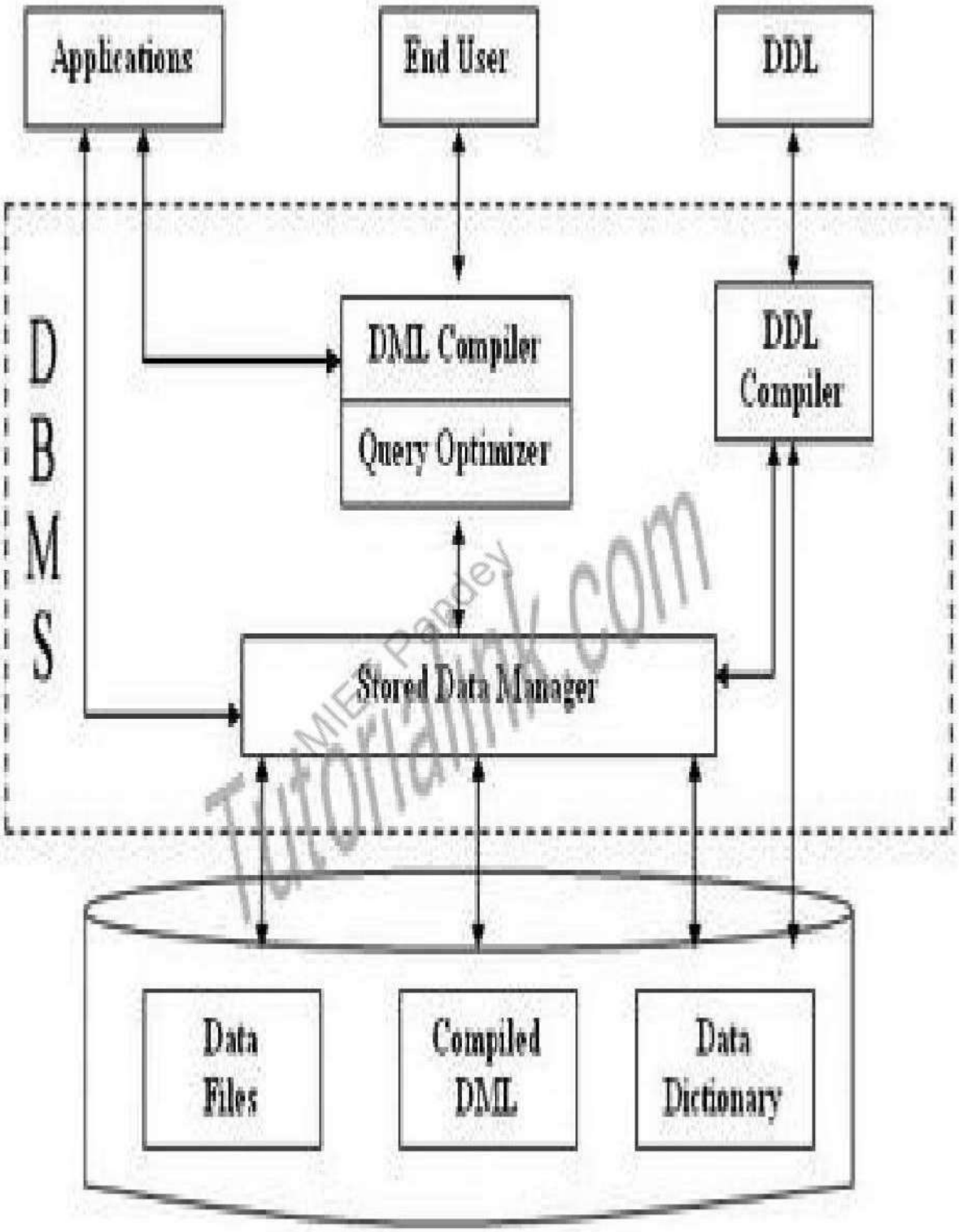


- **Database (Data) Tier** – At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** – At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End- users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end- user and the database.
- **User (Presentation) Tier** – End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

2. DBMS Component Modules and structure:

Structure of DBMS:

- DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database.
- The components of DBMS perform these requested operations on the database and provide necessary data to the users.



The various components of DBMS are described below:

Components of a DBMS

- The components of DBMS can be divided into two parts:

Function and Services of DBMS:

1. **DDL Compiler:**
   - Data Description Language compiler processes schema definitions specified in the DDL.
   - It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.
2. **DML Compiler and Query optimizer:**
   - The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access.
   - The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.
3. **Data Manager:**
   - The Data Manager is the central software component of the DBMS also knows as Database Control System.
   - The Main Functions Of Data Manager Are:
     1. Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.
     2. Controls DBMS information access that is stored on disk.

3. It also controls handling buffers in main memory.
4. It also enforces constraints to maintain consistency and integrity of the data.
5. It also synchronizes the simultaneous operations performed by the concurrent users.
6. It also controls the backup and recovery operations.

4. **Data Dictionary**:
   - Data Dictionary, which stores metadata about the database, in particular the schema of the database.
   - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.
   - Detailed information on physical database design such as storage structure, access paths, files and record sizes.
   - Usage statistics such as frequency of query and transactions.
   - Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as an important part of the DBMS
5. **Data Files:**
   - Which store the database itself?
6. **Compiled DML:**
   - The DML complier converts the high-level Queries into low level file access commands known as compiled DML.
7. **End Users:**
   - The second class of users then is end user, who interacts with system from online workstation or terminals.
   - Use the interface provided as an integral part of the database system software.
   - User can request, in form of query, to access database either directly by using particular language, such as SQL, or by using some pre-developed application interface.
   - Such request is sent to query evaluation engine via DML pre-compiler and DML compiler
   - The query evaluation engine accepts the query and analyses it.
   - It finds the suitable way to execute the compiled SQL statements of the query.
   - Finally, the compiled SQL statements are executed to perform the specified operation

8. Query Processor Units:

Interprets DDL statements into a set of tables containing metadata.
Translates DML statements into low level instructions that the query evaluation engine understands.
Converts DML statements embedded in an application program into procedure calls int he hosts language.
Executes low level instructions generated by DML compiler.

- DDL Interpreter
- DML Compiler
- Embedded DML Pre-compiler
- Query evaluation Engine
- Storage Manager Units

Checks the authority of users to access data.
Checks for the satisfaction of the integrity constraints. Preserves atomicity and controls concurrency.
Manages allocation of space on disk.

Fetches data from disk storage to memory for being used.

a. Authorization Manager

b. Integrity Manager

c. Transaction Manager

d. File manager

e. Buffer Manager

4. Three Levels Database Architecture:

Following are the three levels of database architecture,

1. Physical Level
2. Conceptual Level
3. External Level

three levels database architecture

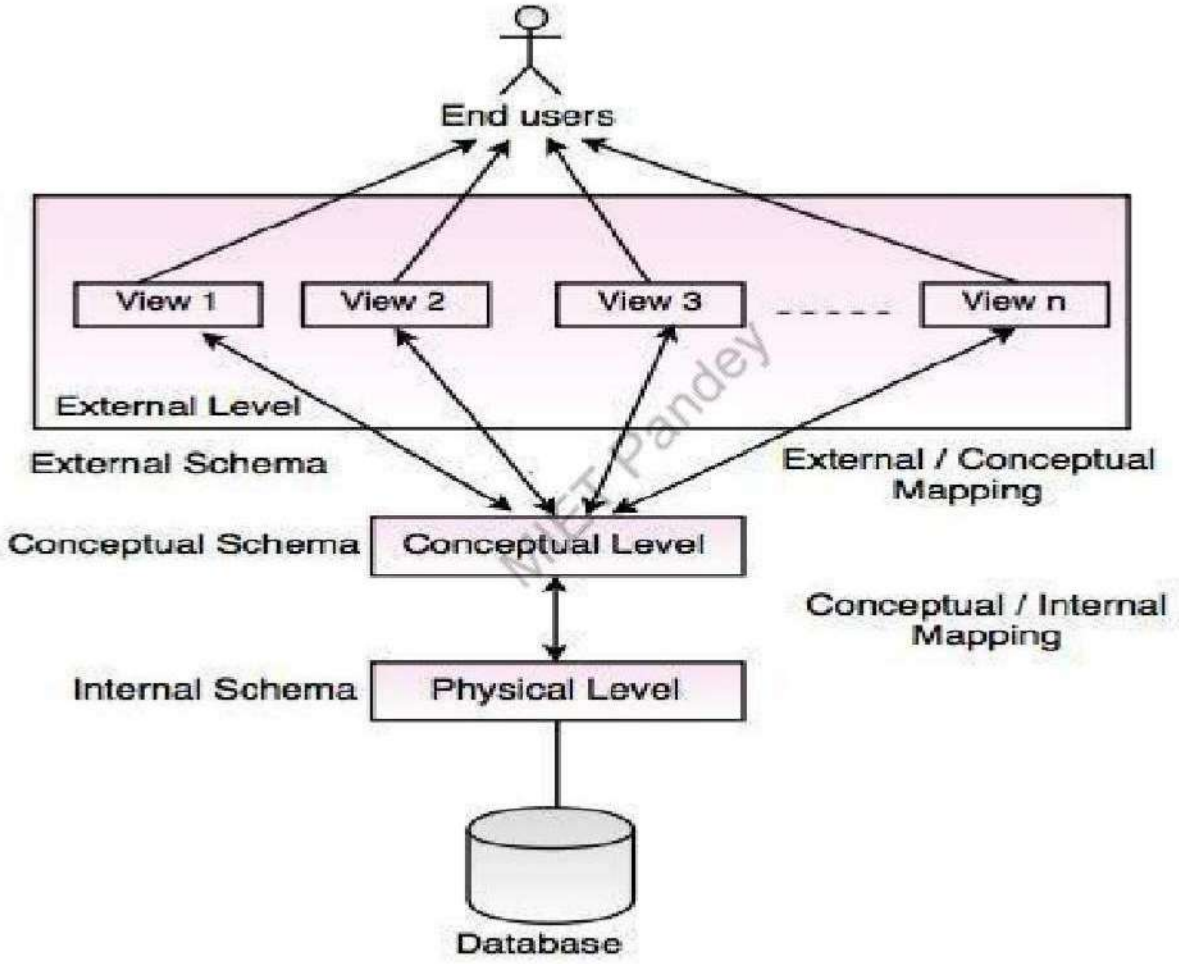


Fig. Three Level Architechture of DBMS

In the above diagram,
It shows the architecture of DBMS.
Mapping is the process of transforming request response between various database levels of architecture.
Mapping is not good for small database, because it takes more time.
In External / Conceptual mapping, DBMS transforms a request on an external schema against the conceptual schema.
In Conceptual / Internal mapping, it is necessary to transform the request from the conceptual to internal levels.

1. Physical Level:

Physical level describes the physical storage structure of data in database.
It is also known as Internal Level.

This level is very close to physical storage of data.
At lowest level, it is stored in the form of bits with the physical addresses on the secondary storage device.
At highest level, it can be viewed in the form of files.

The internal schema defines the various stored data types. It uses a physical data model.

2. Conceptual Level:

Conceptual level describes the structure of the whole database for a group of users.
It is also called as the data model.
Conceptual schema is a representation of the entire content of the database.
These schemas contain all the information to build relevant external records.
It hides the internal details of physical storage.

3. External Level:

External level is related to the data which is viewed by individual end users.
This level includes a no. of user views or external schemas. This level is closest to the user.
External view describes the segment of the database that is required for a particular user group and hides the rest of the database from that user group.

5. Schema and Instance:

- The data which is stored in the database at a particular moment of time is called an instance of the database.
- The overall design of a database is called schema.
- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints.

Example

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

6.Data Independence:

One of the biggest advantages of database is data independence. It means we can change the conceptual schema at one level without affecting the data at other level. It means we can change the structure of a database without affecting the data required by users and

program. This feature was not available in file-oriented approach. There are two types of data independence and they are:

1. Physical data independence
2. Logical data independence

Data Independence The ability to modify schema definition in on level without affecting schema definition in the next higher level is called data independence. There are two levels of data independence:

1. **Physical data independence** is the ability to modify the physical schema without causing application programs to be rewritten. Modifications at the physical level are occasionally necessary to improve performance. It means we change the physical storage/level without affecting the conceptual or external view of the data. The new changes are absorbed by mapping techniques.

2. **Logical data independence** in the ability to modify the logical schema without causing application program to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered (for example, when money-market accounts are added to banking system).

Logical Data independence means if we add some new columns or remove some columns from table then the user view and programs should not change. It is called the logical independence. For example: consider two users A & B. Both are selecting the empno and ename. If user B add a new column salary in his view/table then it will not affect the external view user; user A, but internal view of database has been changed for both users A & B. Now user A can also print the salary.

7. Data Models:

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

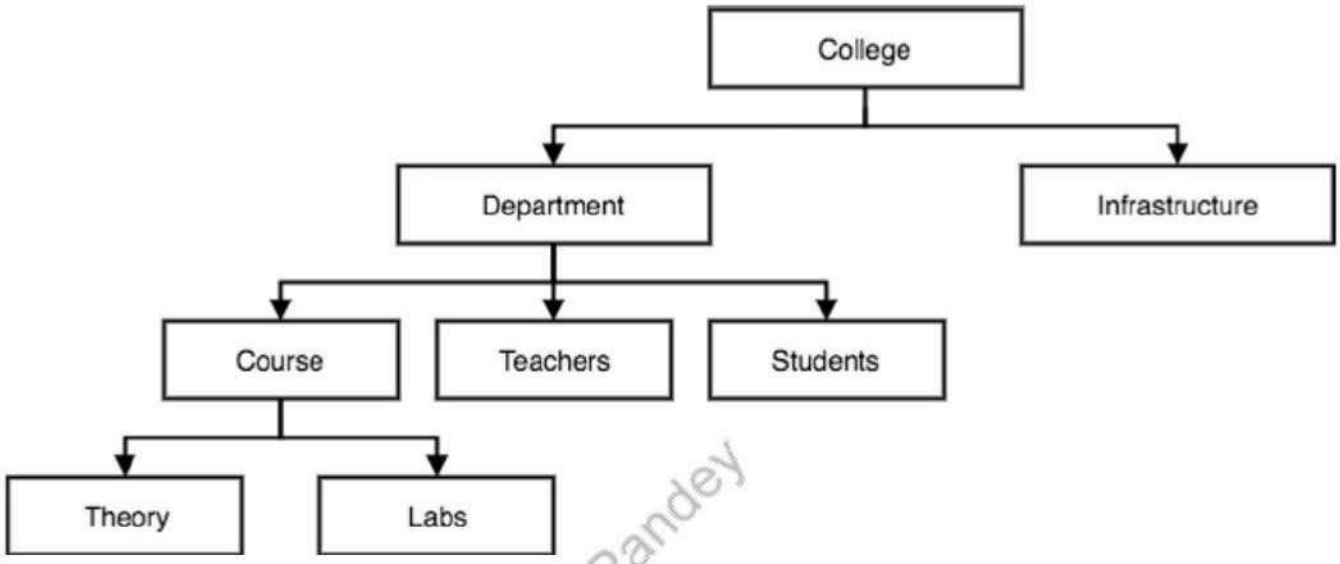
Hierarchical Model

This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

In this model, a child node will only have a single parent node.

This model efficiently describes many real-world relationships like index of a book, recipes etc.

In hierarchical model, data is organized into tree-like structure with one one-to-many relationship between two different types of data, for example, one department can have many courses, many professors and of-course many students.

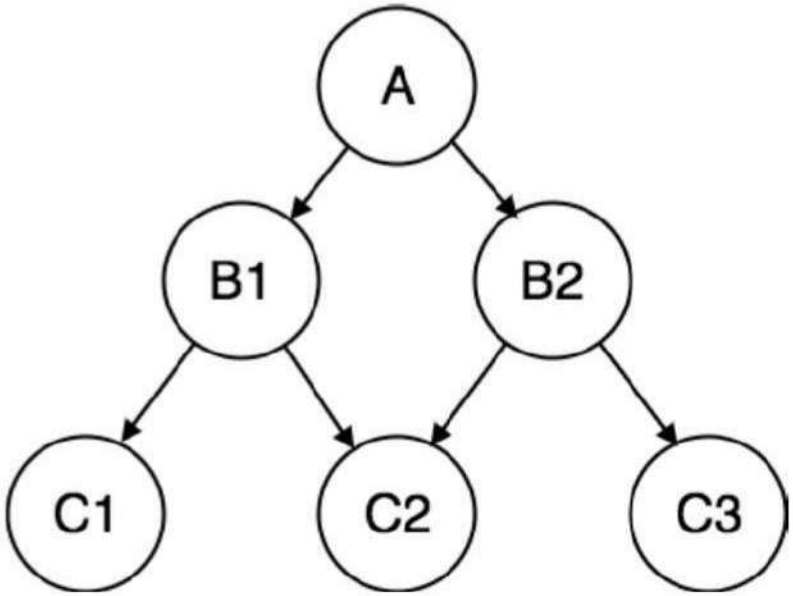


Network Model

This is an extension of the Hierarchical model. In this model data is organized more like a graph, and are allowed to have more than one parent node.

In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast. This database model was used to map many-to-many data relationships.

This was the most widely used database model, before Relational Model was introduced.



Entity-relationship Model

In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

Different entities are related using relationships.

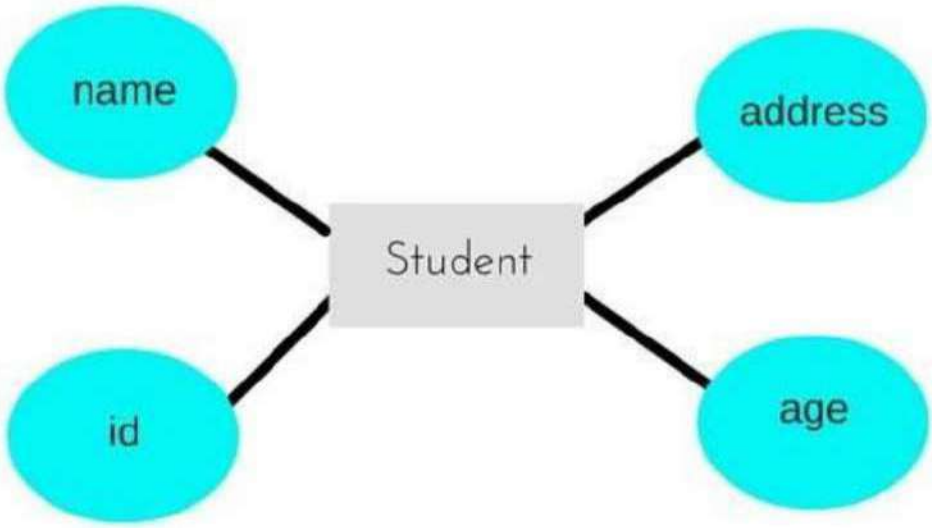
E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

This model is good to design a database, which can then be turned into tables in relational model(explained below).

Let's take an example, If we have to design a School Database,

then **Student** will be an **entity** with **attributes** name, age, address etc.

As **Address** is generally complex, it can be another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.



Relational Model

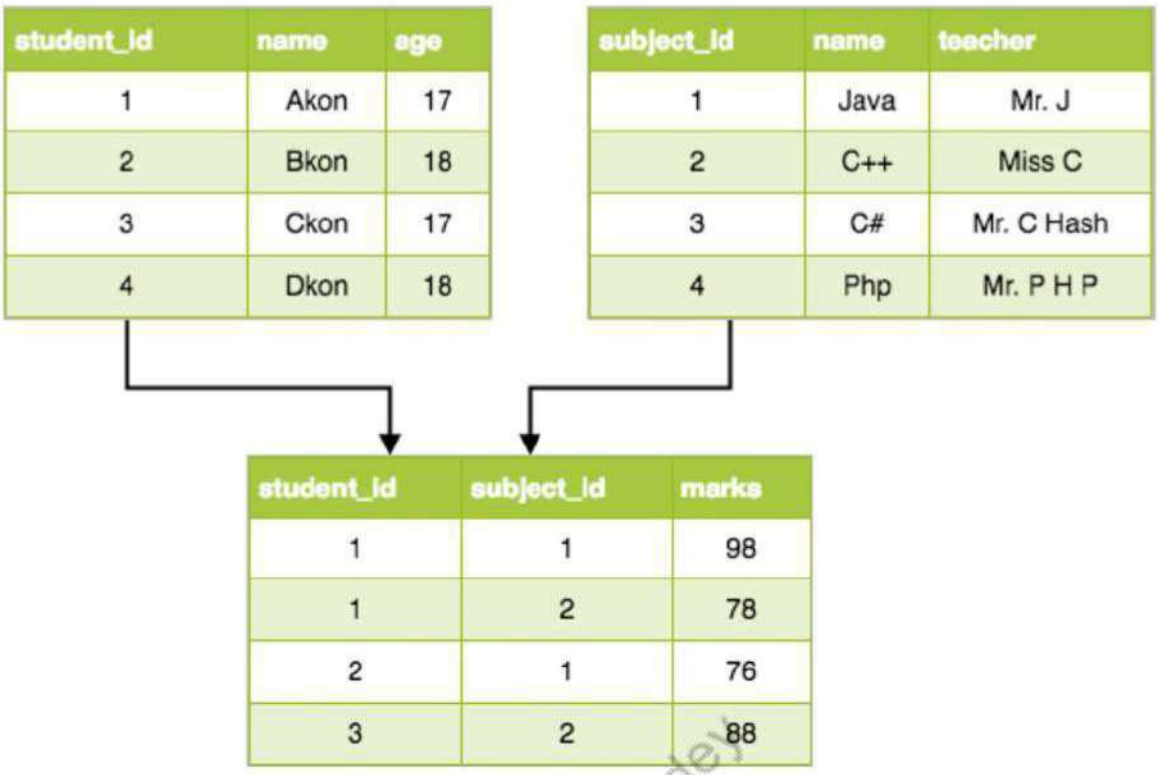
In this model, data is organized in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.

The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.

Hence, tables are also known as **relations** in relational model.

In the coming tutorials we will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.



student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

8. All Database languages:

Data Definition Language (DDL):

statements are used to classify the database structure or schema. It is a type of language that allows the DBA or user to depict and name those entities, attributes, and relationships that are required for the application along with any associated integrity and security constraints. Here are the 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 records 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.

Data Manipulation Language:

A language that offers a set of operations to support the fundamental data manipulation operations on the data held in the database. Data Manipulation Language (DML) statements are used to manage data within schema objects. Here are the 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.

• DELETE - It deletes all records from a table, the space for the records remain.

• MERGE - UPSERT operation (insert or update).

• CALL - It calls a PL/SQL or Java subprogram.

• EXPLAIN PLAN - It explains access path to data.

• LOCK TABLE - It controls concurrency.

Data Control Language:

There are another two forms of database sub-languages. The Data Control Language (DCL) is used to control privilege in Database. To perform any operation in the database, such as for creating tables, sequences or views we need privileges. 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. DCL is used to define two commands. These are:

• Grant - It gives user access privileges to a database.

• Revoke - It takes back permissions from the user.

TCL Commands

Transaction Control Language (TCL) Commands are:

Commit − It is used to save the transactions in the database.

Rollback − It is used to restore the database to that state which was last committed.

Begin − It is used at the beginning of a transaction.

Savepoint − The changes done till savpoint will be unchanged and all the transactions after savepoint will be rolled back.

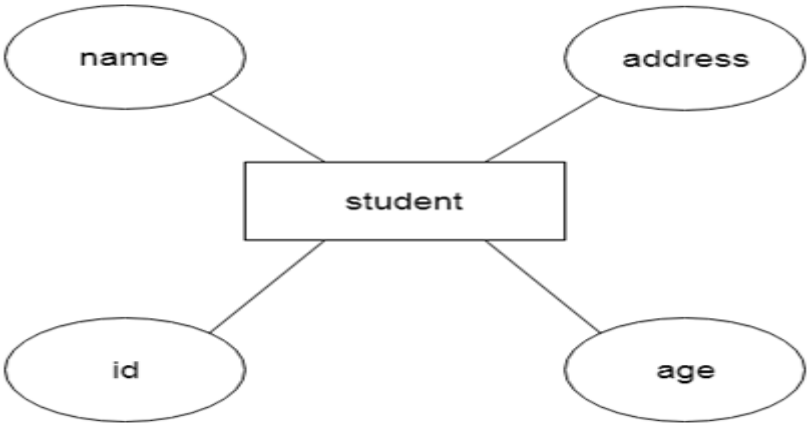
ER (Entity Relationship) Diagram in DBMS

ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.

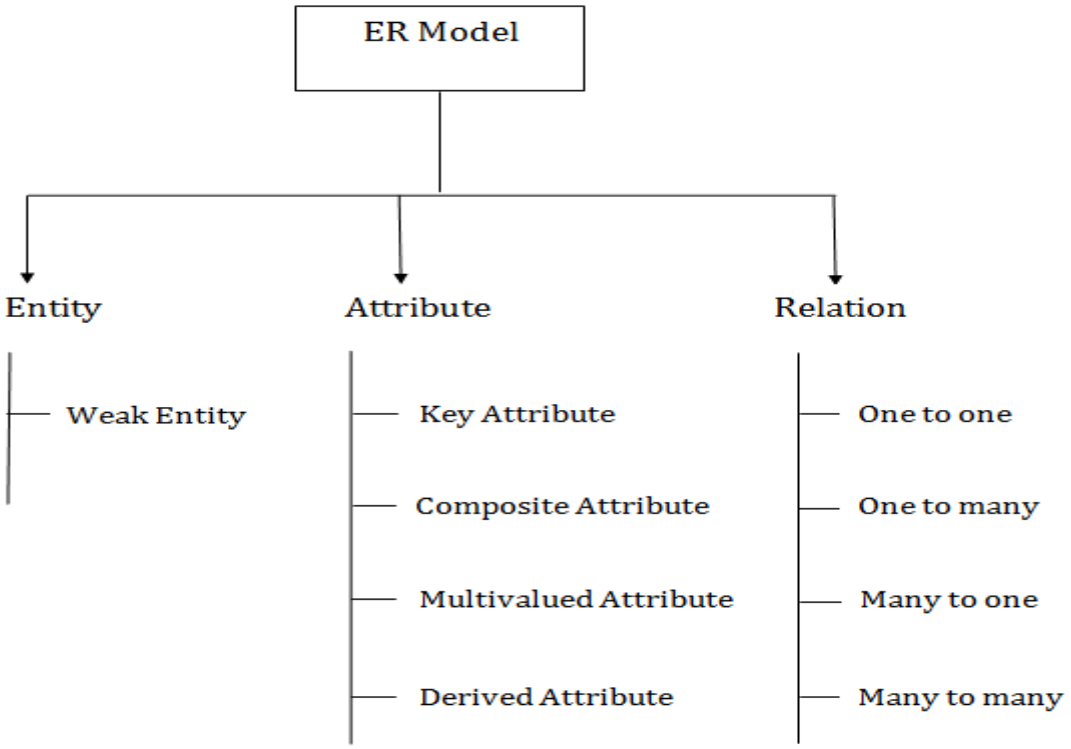
It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



Component of ER Diagram



1. Entity:

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

STRONG ENTITY :- An entity which is independent is strong entity

e.g :- Bank Loan , Person , Bank Account , Customer

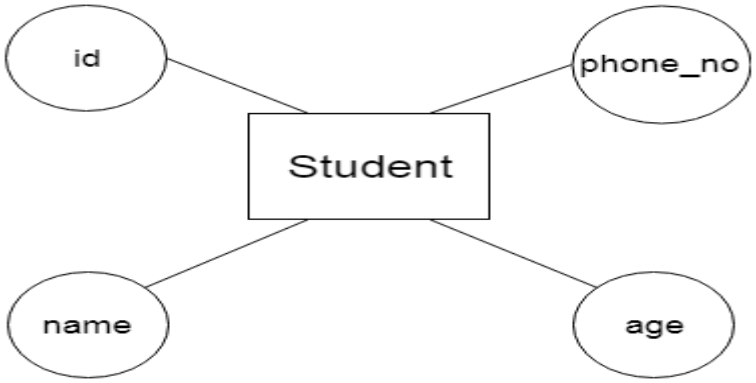
a. Weak Entity

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.

Attribute

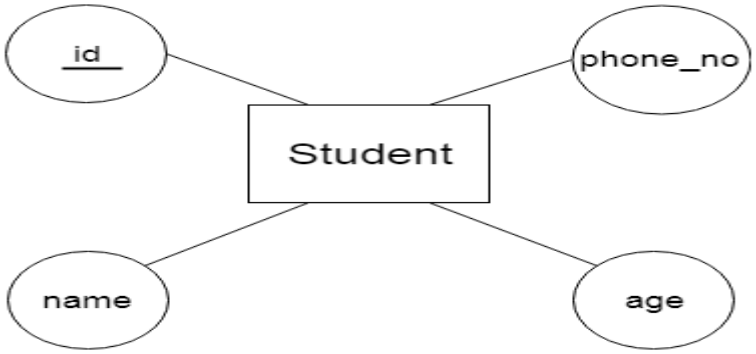
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



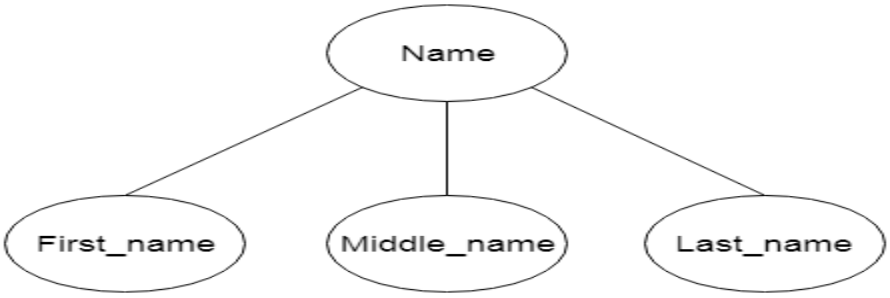
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

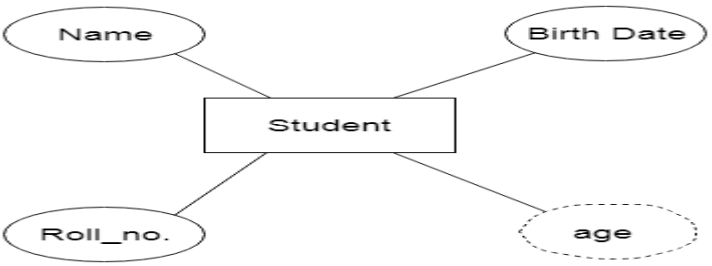
An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.

d. 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

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

Types of relationship are as follows:

a. **One-to-One Relationship**

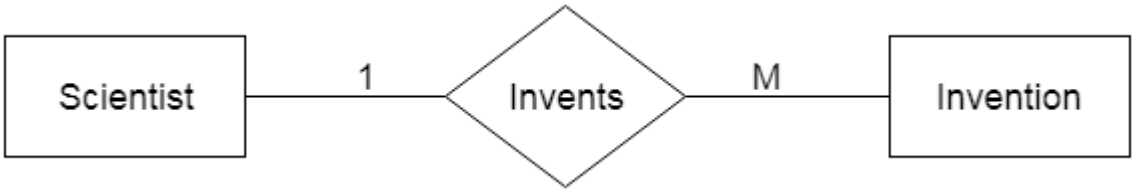
When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

For example, A female can marry to one male, and a male can marry to one female.

b. **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.

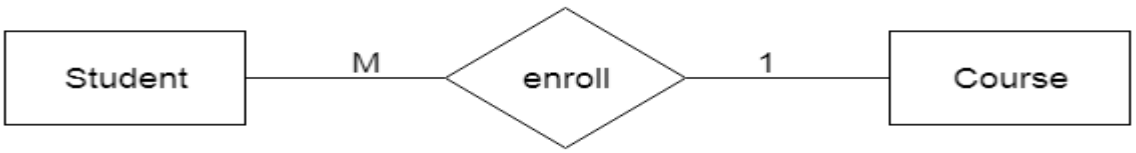
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



c. **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 then it is known as a many-to-one relationship.

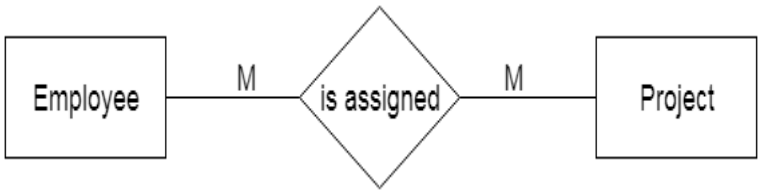
For example, Student enrolls for only one course, but a course can have many students.



d. **Many-to-many relationship**

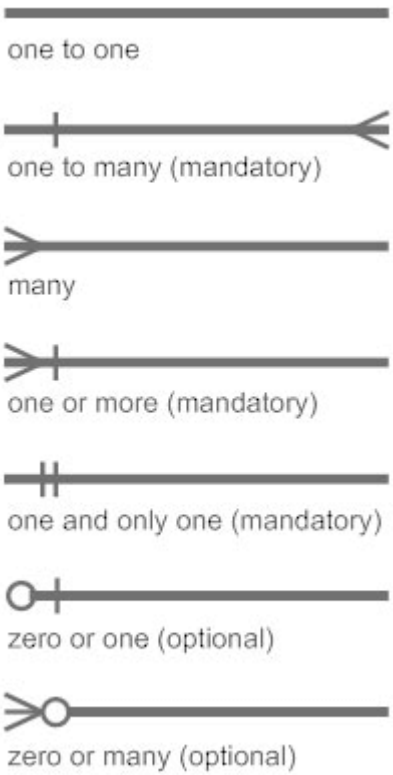
When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.



Notation of ER diagram

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



Cardinality in DBMS (Mapping Constraints)

DBMS stands for Database Management System, which is a tool, or a software used to do various operations on a Database like the Creation of the Database, Deletion of the Database, or Updating the current Database. To simplify processing and data querying, the most popular types of Databases currently in use typically model their data as rows and columns in a set of tables. The data may then be handled, updated, regulated, and structured with ease. For writing and querying data, most Databases employ Structured Query Language (SQL).

Cardinality

Cardinality means how the entities are arranged to each other or what is the relationship structure between entities in a relationship set. In a Database Management System, Cardinality represents a number that denotes how many times an entity is participating with another entity in a relationship set. The Cardinality of DBMS is a very important attribute in representing the structure of a Database. In a table, the number of rows or tuples represents the Cardinality.

Cardinality Ratio

Cardinality ratio is also called Cardinality Mapping, which represents the mapping of one entity set to another entity set in a relationship set. We generally take the example of a binary relationship set where two entities are mapped to each other.

Cardinality is very important in the Database of various businesses. For example, if we want to track the purchase history of each customer then we can use the one-to-many cardinality to find the data of a specific customer. The Cardinality model can be used in Databases by Database Managers for a variety of purposes, but corporations often use it to evaluate customer or inventory data.

There are four types of Cardinality Mapping in Database Management Systems:

One to one

Many to one

One to many

Many to many

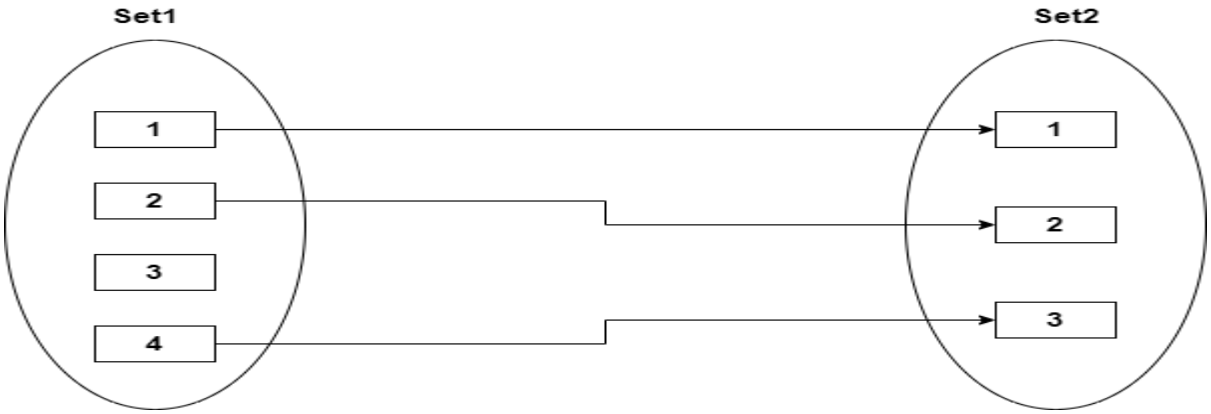
One to One

One to one cardinality is represented by a 1:1 symbol. In this, there is at most one relationship from one entity to another entity. There are a lot of examples of one-to-one cardinality in real life databases.

For example, one student can have only one student id, and one student id can belong to only one student. So, the relationship mapping between student and student id will be one to one cardinality mapping.

Another example is the relationship between the director of the school and the school because one school can have a maximum of one director, and one director can belong to only one school.

Note: it is not necessary that there would be a mapping for all entities in an entity set in one-to-one cardinality. Some entities cannot participate in the mapping.

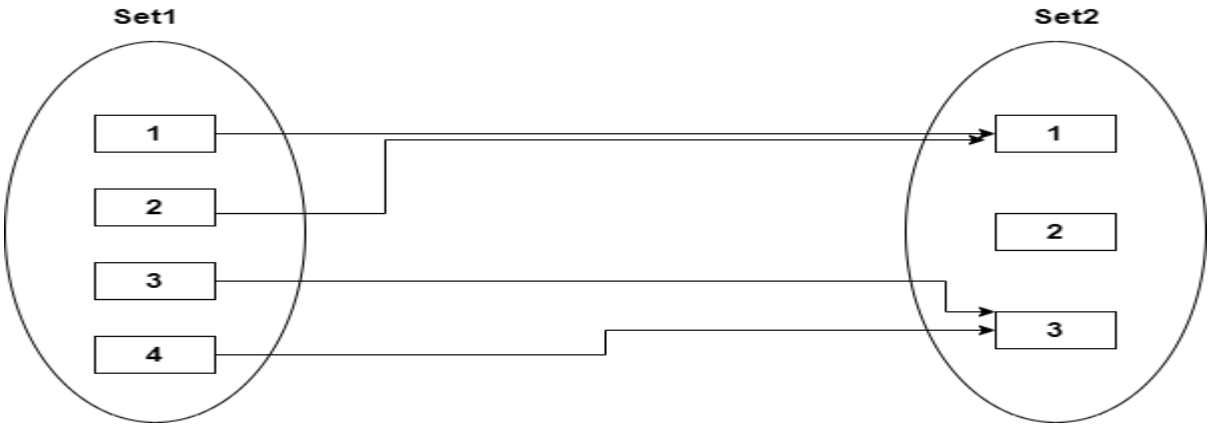


Many to One Cardinality:

In many to one cardinality mapping, from set 1, there can be multiple sets that can make relationships with a single entity of set 2. Or we can also describe it as from set 2, and one entity can make a relationship with more than one entity of set 1.

One to one Cardinality is the subset of Many to one Cardinality. It can be represented by M:1.

For example, there are multiple patients in a hospital who are served by a single doctor, so the relationship between patients and doctors can be represented by Many to one Cardinality.

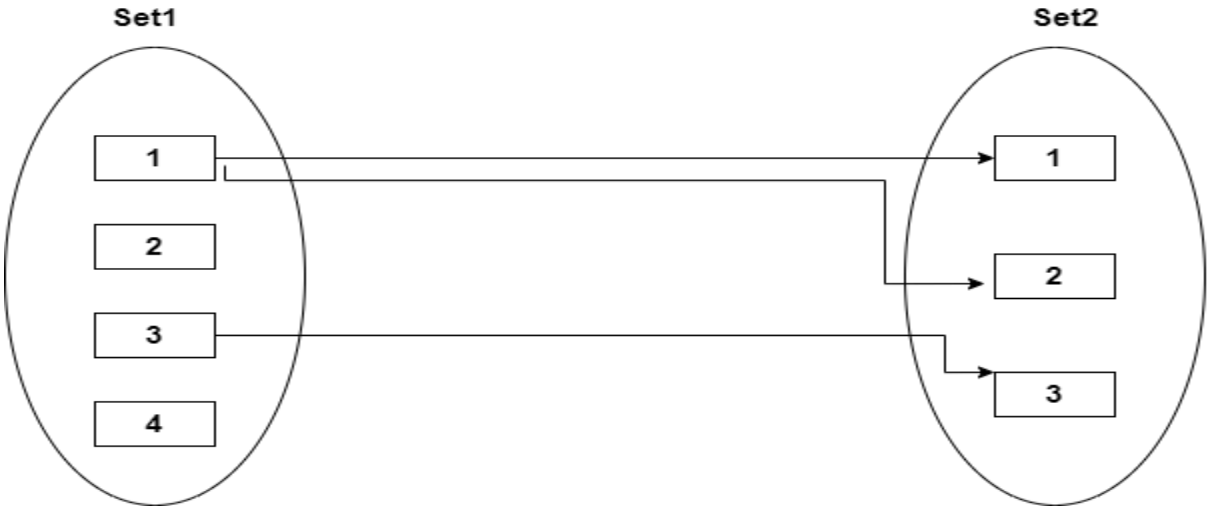


One to Many Cardinalities:

In One-to-many cardinality mapping, from set 1, there can be a maximum single set that can make relationships with a single or more than one entity of set 2. Or we can also describe it as from set 2, more than one entity can make a relationship with only one entity of set 1.

One to one cardinality is the subset of One-to-many Cardinality. It can be represented by 1: M.

For Example, in a hospital, there can be various compounders, so the relationship between the hospital and compounders can be mapped through One-to-many Cardinality.



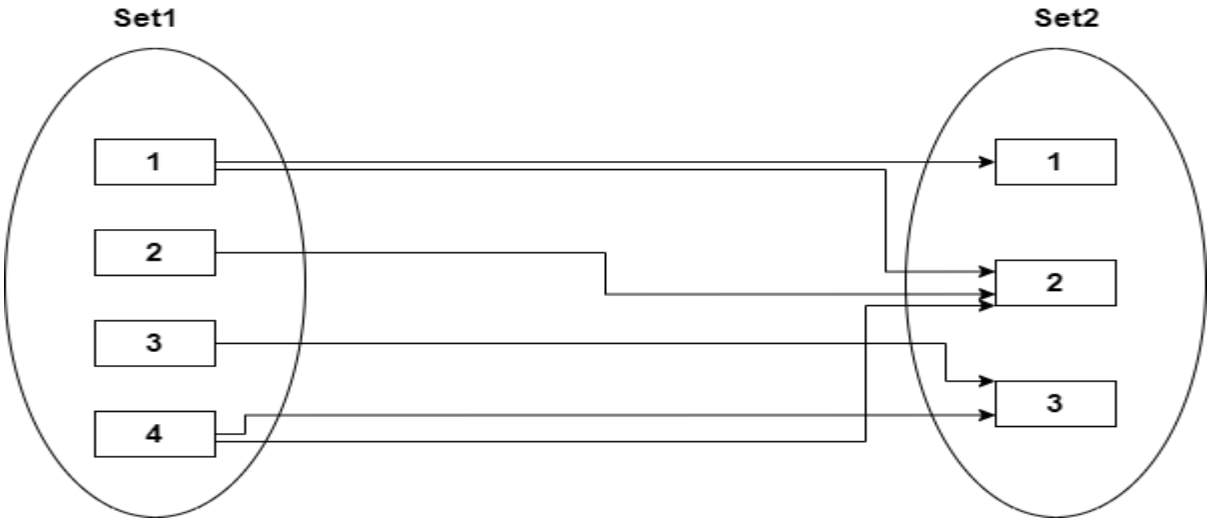
Many to Many Cardinalities:

In many, many cardinalities mapping, there can be one or more than one entity that can associate with one or more than one entity of set 2. In the same way from the end of set 2, one or more than one entity can make a relation with one or more than one entity of set 1.

It is represented by M: N or N: M.

One to one cardinality, One to many cardinalities, and Many to one cardinality is the subset of the many to many cardinalities.

For Example, in a college, multiple students can work on a single project, and a single student can also work on multiple projects. So, the relationship between the project and the student can be represented by many to many cardinalities.



Appropriate Mapping Cardinality

- Evidently, the real-world context in which the relation set is modeled determines the Appropriate Mapping Cardinality for a specific relation set.
- We can combine relational tables with many involved tables if the Cardinality is one-to-many or many-to-one.

- One entity can be combined with a relation table if it has a one-to-one relationship and total participation, and two entities can be combined with their relation to form a single table if both of them have total participation.
- We cannot mix any two tables if the Cardinality is many-to-many

The Enhanced ER Model

As the complexity of data increased in the late 1980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.

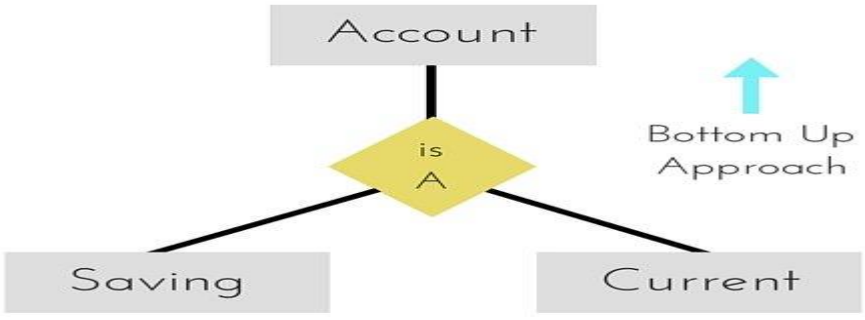
Hence, as part of the Enhanced ER Model, along with other improvements, three new concepts were added to the existing ER Model, they were:

- **Generalization**
- **Specialization**
- **Aggregation**

Generalization

Generalization is a bottom-up approach in which two lower level entities combine to form a higher level entity. In generalization, the higher level entity can also combine with other lower level entities to make further higher level entity.

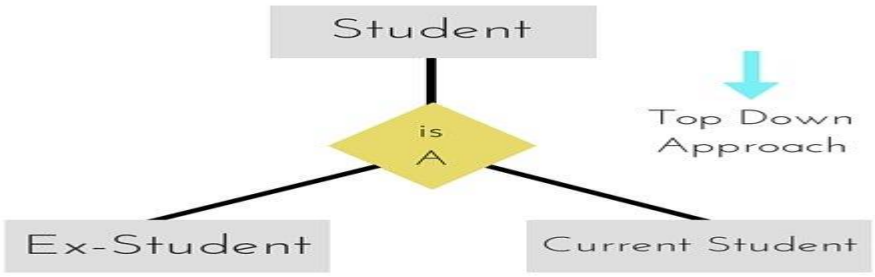
It's more like Superclass and Subclass system, but the only difference is the approach, which is bottom-up. Hence, entities are combined to form a more generalised entity, in other words, sub-classes are combined to form a super-class.



For example, saving and Current account type's entities can be generalised and an entity with name Account can be created, which covers both.

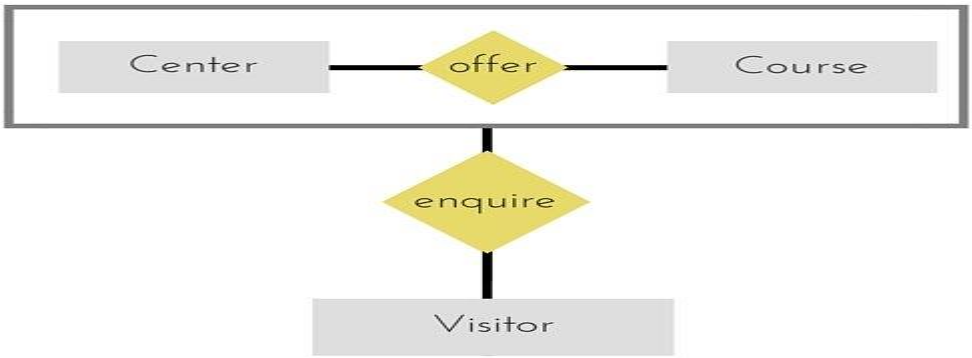
Specialization

Specialization is opposite to Generalization. It is a top-down approach in which one higher level entity can be broken down into two lower level entities. In specialization, a higher level entity may not have any lower-level entity sets, it's possible.



Aggregation

Aggregation is a process when relation between two entities is treated as a single entity.



In the diagram above, the relationship between Centre and Course together, is acting as an Entity, which is in relationship with another entity Visitor. Now in real world, if a Visitor or a Student visits a Coaching Centre, he/she will never enquire about the centre only or just about the course, rather he/she will ask enquire about both.

Keys

Super Key – A super key is a group of single or multiple keys which identifies rows in a table.

Primary Key – is a column or group of columns in a table that uniquely identify every row in that table.

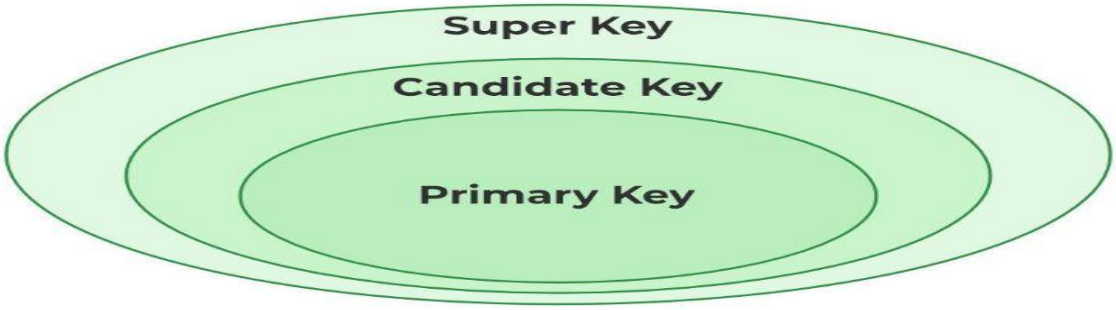
Candidate Key – is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes.

Alternate Key – is a column or group of columns in a table that uniquely identify every row in that table.

Foreign Key – is a column that creates a relationship between two tables. The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity.

Compound Key – has two or more attributes that allow you to uniquely recognize a specific record. It is possible that each column may not be unique by itself within the database.

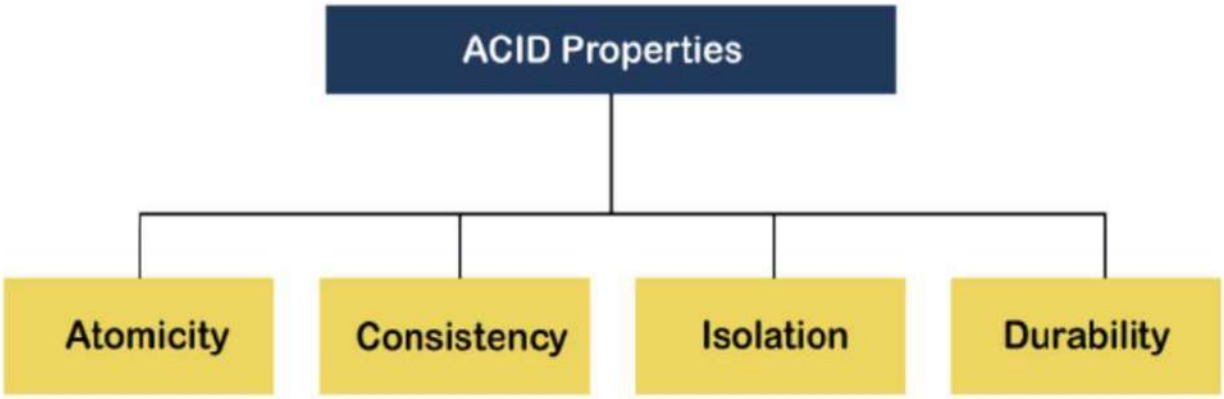
Composite Key – is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individual uniqueness is not guaranteed.



Relation between Primary Key, Candidate Key, and Super Key

ACID Properties

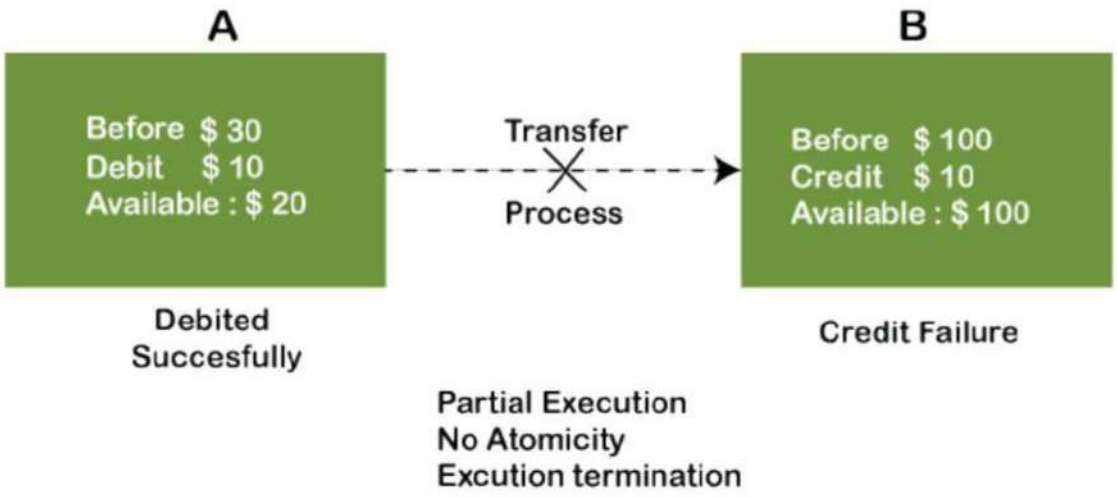
The expansion of the term ACID defines for:



1) Atomicity

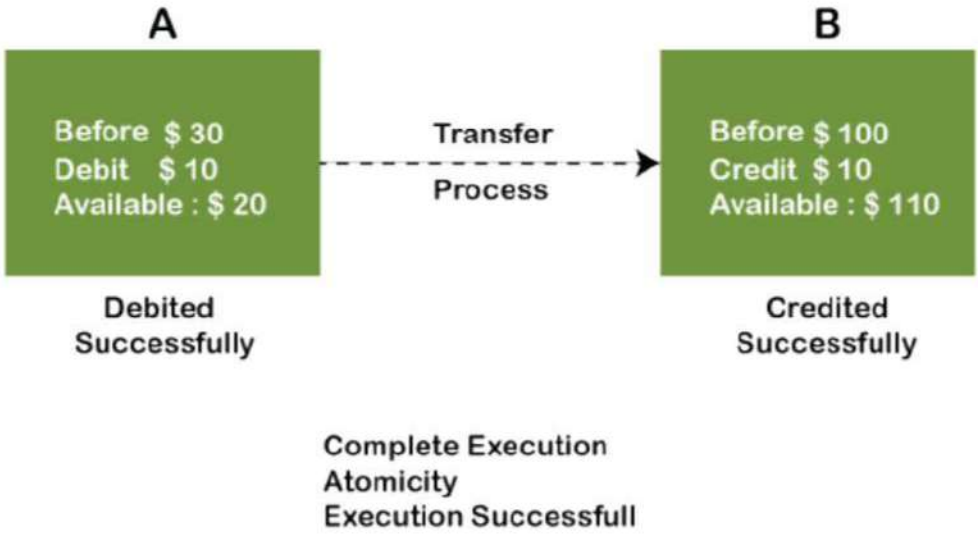
The term atomicity defines that the data remains atomic. It means if any operation is performed on the data, either it should be performed or executed completely or should not be executed at all. It further means that the operation should not break in between or execute partially. In the case of executing operations on the transaction, the operation should be completely executed and not partially.

Example: If Remo has account A having $30 in his account from which he wishes to send $10 to Sheero's account, which is B. In account B, a sum of $ 100 is already present. When $10 will be transferred to account B, the sum will become $110. Now, there will be two operations that will take place. One is the amount of $10 that Remo wants to transfer will be debited from his account A, and the same amount will get credited to account B, i.e., into Sheero's account. Now, what happens - the first operation of debit executes successfully, but the credit operation, however, fails. Thus, in Remo's account A, the value becomes $20, and to that of Sheero's account, it remains $100 as it was previously present.



In the above diagram, it can be seen that after crediting $10, the amount is still $100 in account B. So, it is not an atomic transaction.

The below image shows that both debit and credit operations are done successfully. Thus the transaction is atomic.

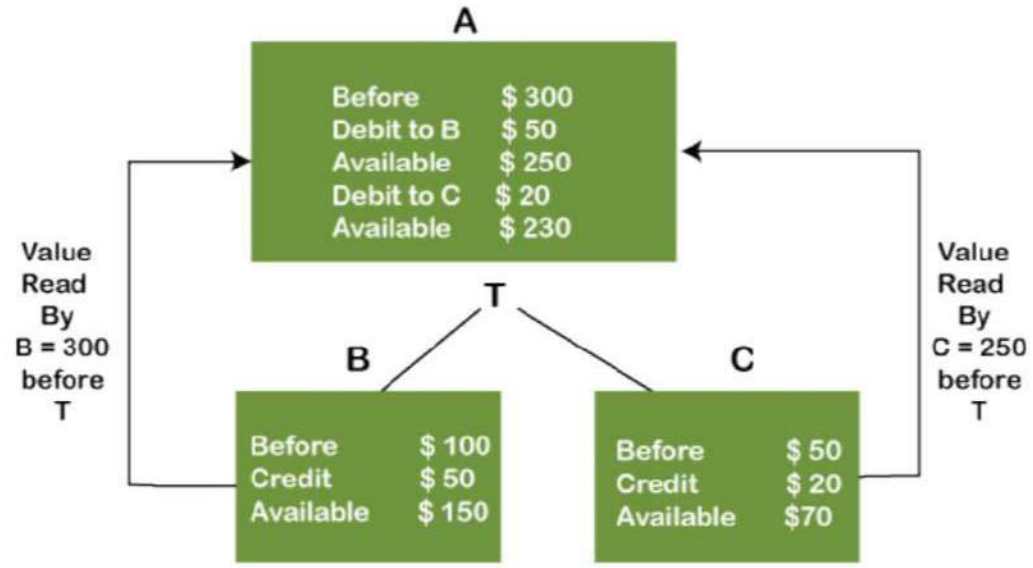


Thus, when the amount loses atomicity, then in the bank systems, this becomes a huge issue, and so the atomicity is the main focus in the bank systems.

2) Consistency

The word **consistency** means that the value should remain preserved always. In DBMS, the integrity of the data should be maintained, which means if a change in the database is made, it should remain preserved always. In the case of transactions, the integrity of the data is very essential so that the database remains consistent before and after the transaction. The data should always be correct.

Example:



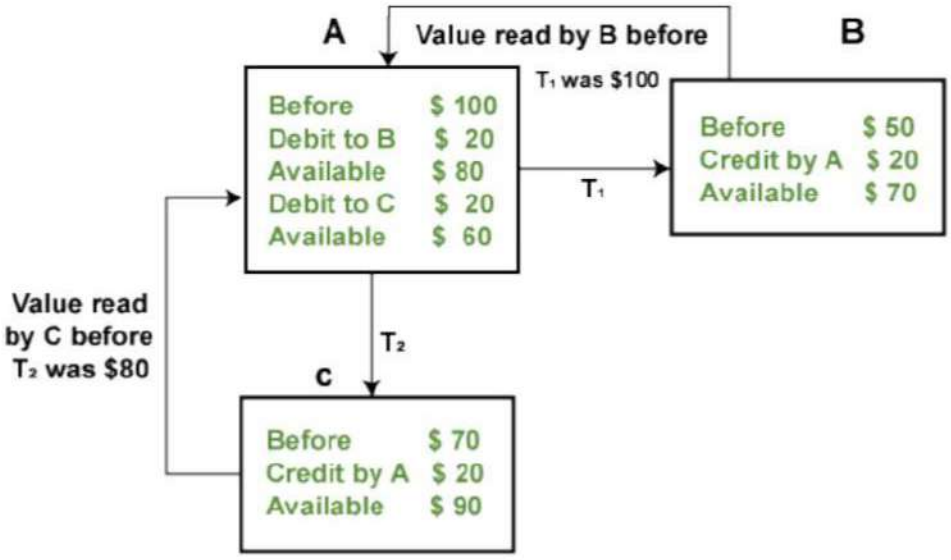
Data Consistent

In the above figure, there are three accounts, A, B, and C, where A is making a transaction T one by one to both B & C. There are two operations that take place, i.e., Debit and Credit. Account A firstly debits $50 to account B, and the amount in account A is read $300 by B before the transaction. After the successful transaction T, the available amount in B becomes $150. Now, A debits $20 to account C, and that time, the value read by C is $250 (that is correct as a debit of $50 has been successfully done to B). The debit and credit operation from account A to C has been done successfully. We can see that the transaction is done successfully, and the value is also read correctly. Thus, the data is consistent. In case the value read by B and C is $300, which means that data is inconsistent because when the debit operation executes, it will not be consistent.

3) Isolation

The term 'isolation' means separation. In DBMS, Isolation is the property of a database where no data should affect the other one and may occur concurrently. In short, the operation on one database should begin when the operation on the first database gets complete. It means if two operations are being performed on two different databases, they may not affect the value of one another. In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained. Any changes that occur in any particular transaction will not be seen by other transactions until the change is not committed in the memory.

Example: If two operations are concurrently running on two different accounts, then the value of both accounts should not get affected. The value should remain persistent. As you can see in the below diagram, account A is making T1 and T2 transactions to account B and C, but both are executing independently without affecting each other. It is known as Isolation.



Isolation - Independent execution of T_1 & T_2 by A

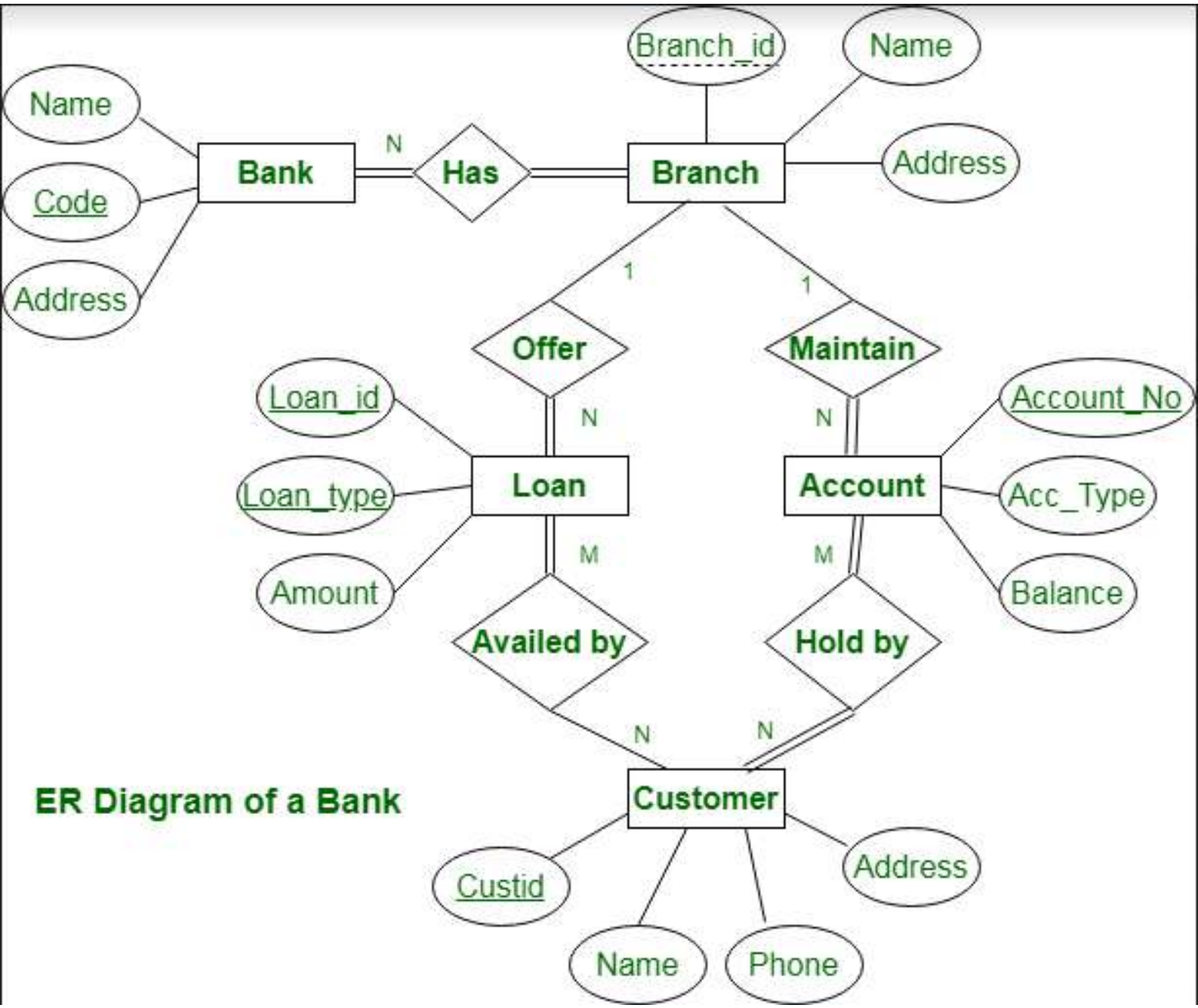
4) Durability

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

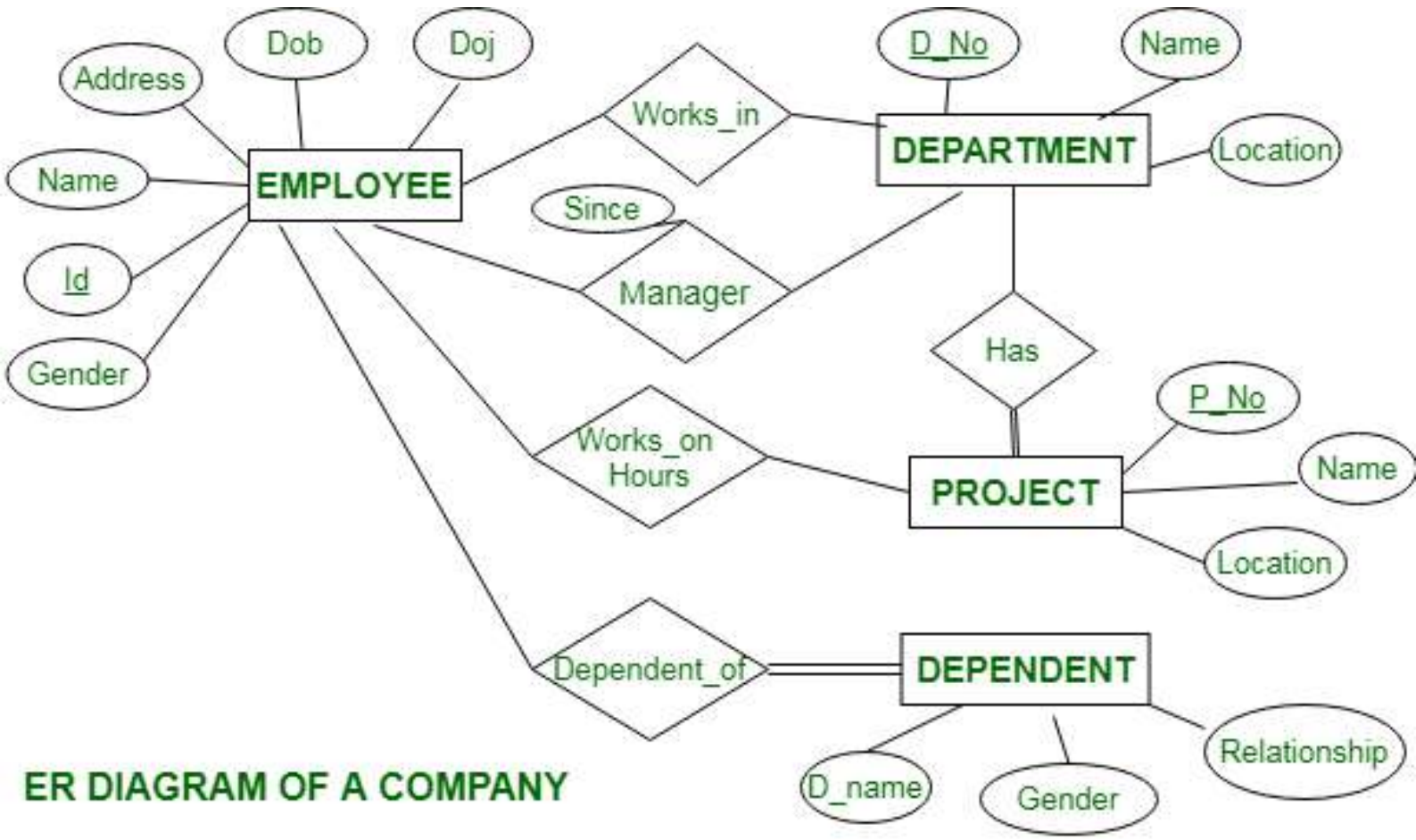
responsibility of the recovery manager for ensuring the durability of the database. For committing the values, the COMMIT command must be used every time we make changes.

Difference between hierarchical model and network model

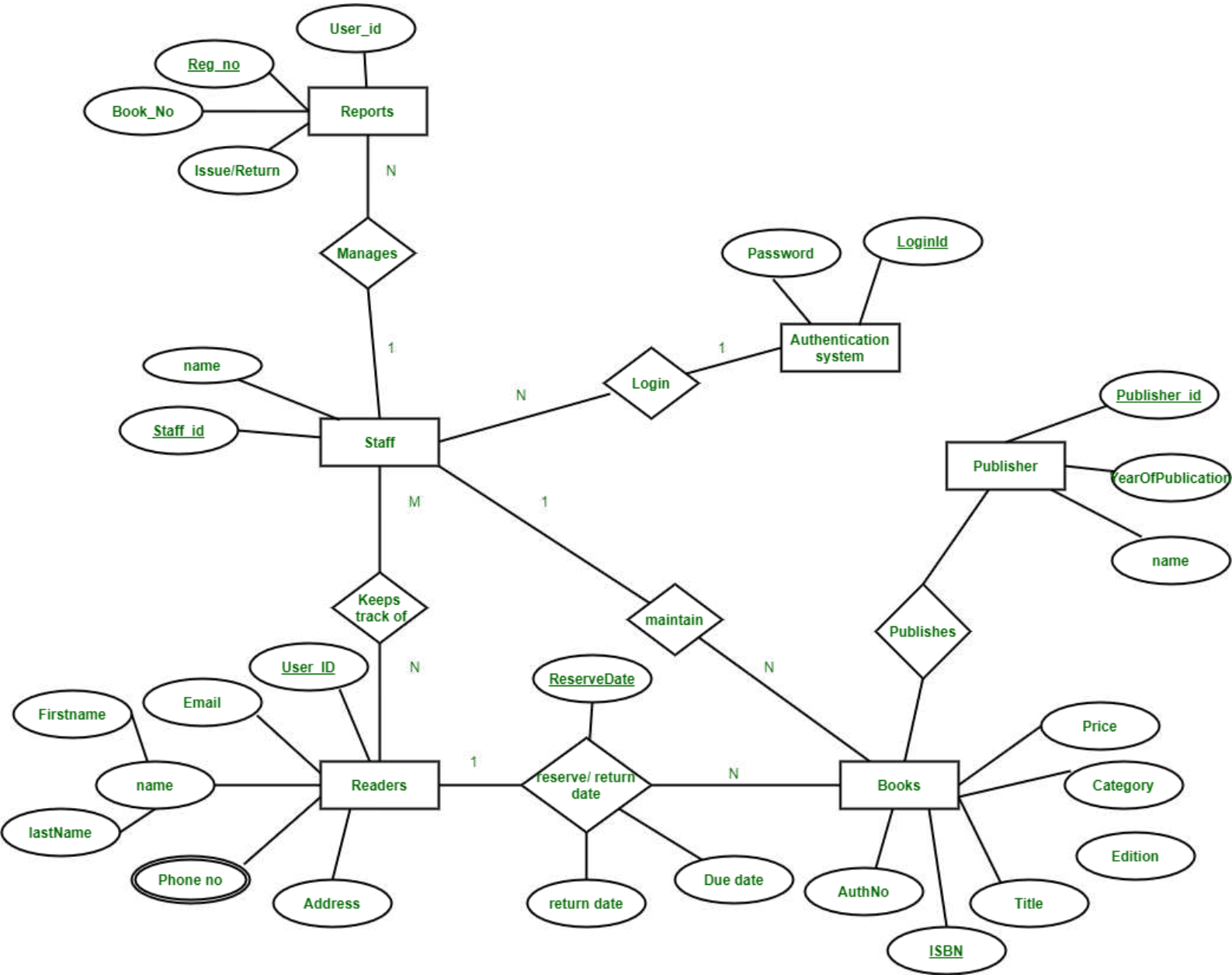
Sr. No.	Key	Hierarchical Data Model	Network Data Model
1	Basic	Relationship between records is of the parent child type	Relationship between records is expressed in the form of pointers or links.
2	Data Inconsistency	It can have data inconsistency during the updation and deletion of the data	No Data inconsistency
3	Traversing	Traversing of data is complex	Data traversing is easy because node can be accessed from parent to child or child to parent
4	Relationship	It does not support many to many relationships	It support many to many relationships
5	Structure	Its create tree like structure	It support graph like structure



ER Diagram of a Bank



ER DIAGRAM OF A COMPANY


User_id
Reg_no
Book_No
Reports
Issue/Return
N
Manages
1
Password
LoginId
Authentication system
1
Login
N
name
Staff_id
Staff
M
1
Keeps track of
maintain
Publisher_id
Publisher
YearOfPublication
name
Publishes
N
N
User_ID
ReserveDate
Firstname
Email
name
lastName
Readers
1
reserve/ return date
N
Books
Price
Category
Edition
Phone no
Address
return date
Due date
AuthNo
Title
ISBN