

UNIT-3: Concepts of Database:

3.1 Database characteristics:

3.1.1 Data Independence (Logical and Physical)

3.1.2 Components of Database (User, Application, DBMS, Database)

3.1.3 Database Architecture (1-tier, 2-tier, and 3-tier)

3.1.3.1 Comparison, advantages and disadvantages.

3.2 Database Models (Hierarchical, Network, E/R, Relational)

3.2.1 E/R model: Entity, Relationship, Attribute

3.2.2 E/R Diagram: One to one, one to many, many to one, many to many

3.2.3 Strong entity, weak entity

3.2.4 Key attribute, derived attribute, Multi-valued attribute

3.3 Types of keys:

3.3.1 Super key, candidate key, Primary key, Composite key, Foreign key, Unique key.

3.4. Normalization

3.4.1. Why normalization (Insertion, Updating, Deletion anomalies)

3.4.2. Normalization Rules

3.4.2.1. Concepts of Dependency, Transitive Dependency

3.4.2.2. Armstrong Axioms

3.4.2.3. 1st Normal Form, 2nd Normal Form, 3rd Normal Form, B.C.N.F.

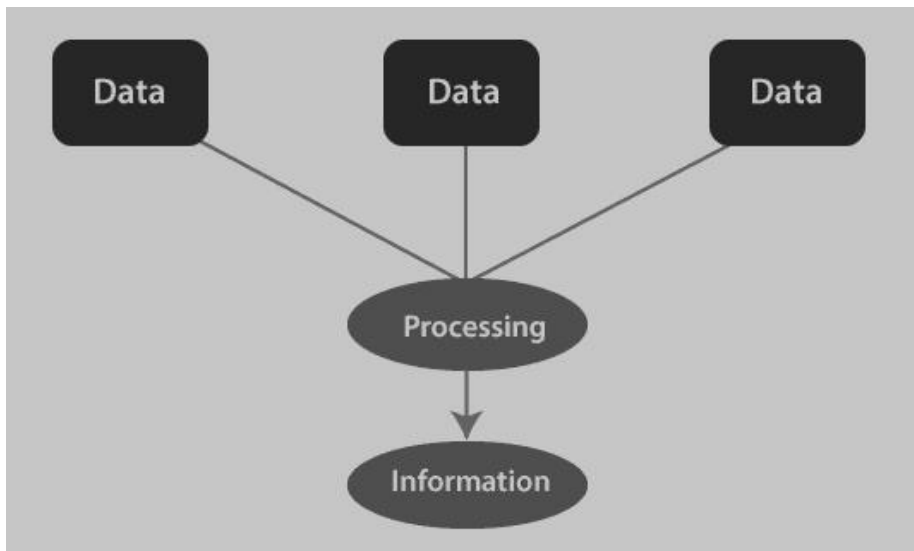
3.1 Database characteristics:

Meaning of Data, Information and Database

Data: Data is the raw fact. **Example:** Product name, Name of student.

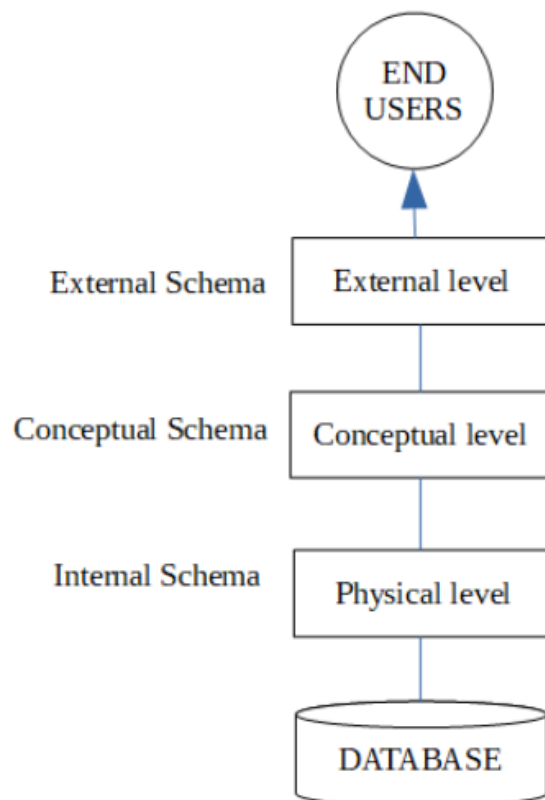
Information: It is a processed form of data. **Example:** Report card of student.

Database: Database is an organized collection of structured information, or data, typically stored electronically in a computer system. **Example:** sales transactions, customer data.



Levels of DBMS

The three levels present in this architecture are Physical level, Conceptual level and External level.



The details of these levels are as follows –

Physical Level

This is the lowest level in the three level architecture. It is also known as the internal level. The physical level describes how data is actually stored in the database.

Conceptual Level

The conceptual level is at a higher level than the physical level. It is also known as the logical level. It describes how the database appears to the users conceptually and the relationships between various data tables.

External Level

This is the highest level in the three level architecture and closest to the user. It is also known as the view level.

3.1.1 Data Independence (Logical and Physical)

The ability to modify schema definition in one level without affecting a schema definition in the next higher level is called data independence. There are two levels of data independence.

1. Physical Data Independence: -

Physical data independence is the ability to modify the physical schema without affecting logical schema modification at the physical level are occasionally necessary to improve performance.

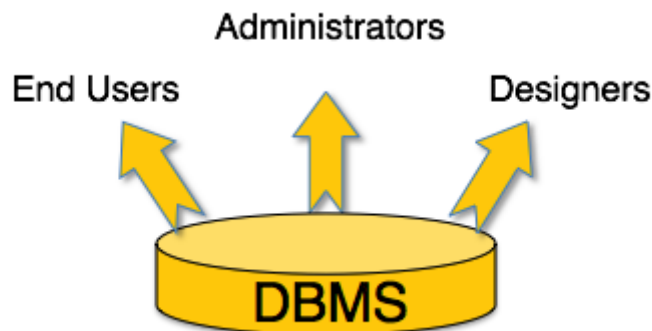
2. Logical Data Independence: -

Logical data independence is the ability to modify the logical schema without causing application program to be rewritten. Modifications at the logical levels are necessary whenever the logical structure of the database is altered.

3.1.2 Components of Database (User, Application, DBMS, Database)

➤ User

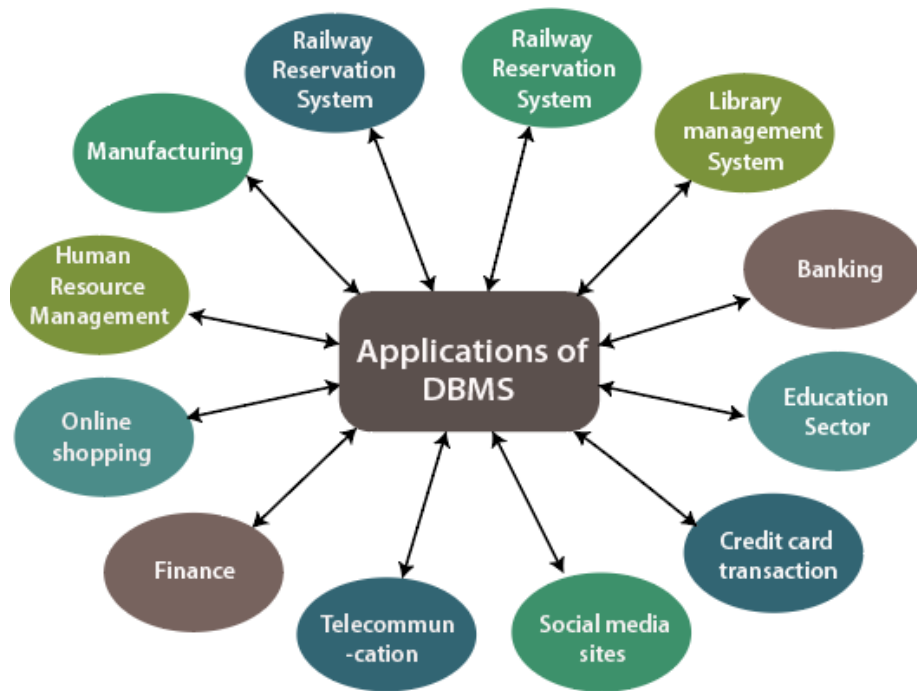
A typical DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be categorized as follows –



- **Administrators** – Administrators maintain the DBMS and are responsible for administering the database. They create access profiles for users and apply limitations to maintain isolation and force security. Administrators also look after DBMS resources like system license, required tools, and other software and hardware related maintenance.
- **Designers** – Designers are the group of people who actually work on the designing part of the database. They keep a close watch on what data should be kept and in what format.
- **End Users** – End users are those who actually collect the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

➤ Applications

- There are various fields where a database management system is used. Following are some applications which make use of the database management system:



1. Railway Reservation System: In the railway reservation system, the database is required to store the record or data of ticket bookings, status about train's arrival and departure. Also if trains get late, people get to know it through database update.

2. Library Management System: There are lots of books in the library so; it is tough to store the record of all the books in a register or copy. So, the database management system (DBMS) is used to maintain all the information related to the name of the book, issue date, availability of the book, and its author.

3. Banking: Database management system is used to store the transaction information of the customer in the database.

4. Education Sector: Presently, examinations are conducted online by many colleges and universities. They manage all examination data through the database management system (DBMS). In spite of that student's registration details, grades, courses, fee, attendance, results, etc. all the information is stored in the database.

5. Credit card transactions: Database Management system is used for purchasing on credit cards and generation of monthly statements.

6. Social Media Sites: We all use of social media websites to connect with friends and to share our views with the world. Daily, millions of people sign up for these social media accounts like Facebook, Twitter, and Google plus.

7. Telecommunications: Without DBMS any telecommunication company can't think. The Database management system is necessary for these companies to store the call details and monthly post-paid bills in the database.

8. Finance: The database management system is used for storing information about sales, holding and purchases of financial instruments such as stocks and bonds in a database.

9. Online Shopping: These days, online shopping has become a big trend. No one wants to visit the shop and waste their time. Everyone wants to shop through online shopping websites (such as Amazon, Flipkart, snapdeal) from home. So all the products are sold and added only with the help of the database management system (DBMS). Invoice bills, payments, purchase information all of these are done with the help of DBMS.

10. Human Resource Management: Big firms or companies have many workers or employees working under them. They store information about employee's salary, tax, and work with the help of database management system (DBMS).

11. Manufacturing: Manufacturing companies make different types of products and sale them on a daily basis. In order to keep the information about their products like bills, purchase of the product, quantity, supply chain management, database management system (DBMS) is used.

12. Airline Reservation System: This system is the same as the railway reservation system. This system also uses a database management system to store the records of flights departure, arrival, and delay status.

➤ DBMS

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data.

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

There are the following components of DBMS:

1. Software
2. Hardware
3. Procedures
4. Data
5. Users

Software

- The main component of a Database management system is the software. It is the set of programs which is used to manage the database and to control the overall computerized database.
- The DBMS software provides an easy-to-use interface to store, retrieve, and update data in the database.

Hardware

- This component of DBMS consists of a set of physical electronic devices such as computers, I/O channels, storage devices, etc that create an interface between computers and the users.
- This DBMS component is used for keeping and storing the data in the database.

Procedures

- Procedures refer to general rules and instructions that help to design the database and to use a database management system.
- Procedures are used to setup and install a new database management system (DBMS), to login and logout of DBMS software, to manage DBMS or application programs, to take backup of the database, and to change the structure of the database, etc.

Data

- It is the most important component of the database management system.
- The main task of DBMS is to process the data. Here, databases are defined, constructed, and then data is stored, retrieved, and updated to and from the databases.

➤ Database

Databases often store information about people, such as customers or users. For example, social media platforms use databases to store user information, such as names, email addresses and user behavior. The data is used to recommend content to users and improve the user experience.

Purpose of Database

Databases are used for storing, maintaining and accessing any sort of data.

Advantages using DBMS

- Database minimizes data redundancy to a great extent.
- The database can control the inconsistency of data to a large extent.
- Sharing of data is also possible using the database.
- The use of Databases can ensure data security.
- Integrity can be managed using the database.

3.1.3 Database Architecture (1-tier, 2-tier, 3-tier)

3.1.3.1 Comparison, Advantages and Disadvantages

A Database Management system is not always directly available for users and applications to access and store data in it. A Database Management system can be **centralised** (all the data stored at one location), **decentralised** (multiple copies of database at different locations) or **hierarchical**, depending upon its architecture.

Database Architecture is logically of two types:

1. 1-tier DBMS architecture
2. 2-tier DBMS architecture
3. 3-tier DBMS architecture

1-tier DBMS architecture

1-tier DBMS architecture also exist, this is when the database is directly available to the user for using it to store data. Generally such a setup is used for local application development, where programmers communicate directly with the database for quick response.

Advantages

1. Easy to implement and optimize performance.
2. Do not have compatibility or Context switching issues.
3. The cost of deployment is less e.g. - development and management cost.

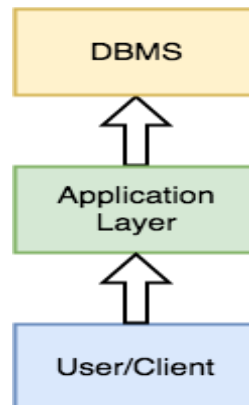
Disadvantages

1. Do not support remote/ distributed access for data resources.
2. The cost of the central mainframe is high.

2-tier DBMS architecture

2-tier DBMS architecture includes an **Application layer** between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.

An application interface known as **ODBC**(Open Database Connectivity) provides an API that allows client side program to call the DBMS. Most DBMS vendors provide ODBC drivers for their DBMS.



Such architecture provides the DBMS extra security as it is not exposed to the End User directly. Also, security can be improved by adding security and authentication checks in the Application layer too.

Advantages:

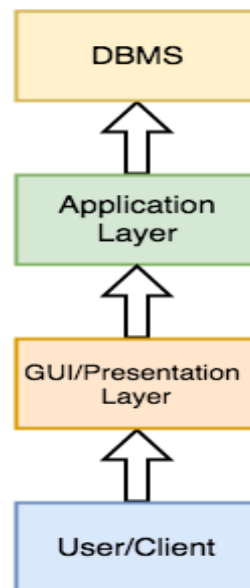
1. Easy to maintain and modification is bit easy.
2. Communication is faster.

Disadvantages:

1. In two tier architecture application performance will be degrade upon increasing the users.
2. Cost-ineffective

3-tier DBMS architecture

3-tier DBMS architecture is the most commonly used architecture for web applications.



It is an extension of the 2-tier architecture. In the 2-tier architecture, we have an application layer which can be accessed programmatically to perform various operations on the DBMS. The application generally understands the Database Access Language and processes end users requests to the DBMS.

In 3-tier architecture, an additional Presentation or GUI Layer is added, which provides a graphical user interface for the End user to interact with the DBMS.

For the end user, the GUI layer is the Database System, and the end user has no idea about the application layer and the DBMS system.

If you have used **MySQL**, then you must have seen **PHPMyAdmin**, it is the best example of a 3-tier DBMS architecture.

Advantages

1. High performance, lightweight persistent objects
2. High degree of flexibility in deployment platform and configuration
3. Better Re-use

4. Improved Security – Client is not direct access to database.
5. In three tier architecture application performance is good.

Disadvantages

1. Increase Complexity

3.2 Database Models (Hierarchical, Network, E/R, Relational)

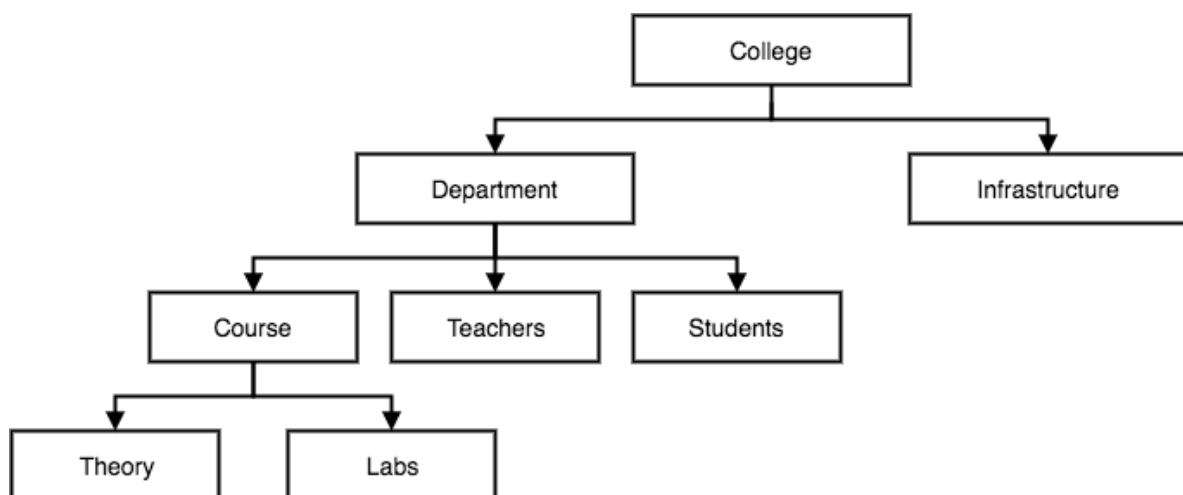
➤ **Hierarchical Model**

This database model organises 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 organised into tree-like structure with one -to-many relationship between two different types of data, **For example**, one department can have many courses, many professors and many students.

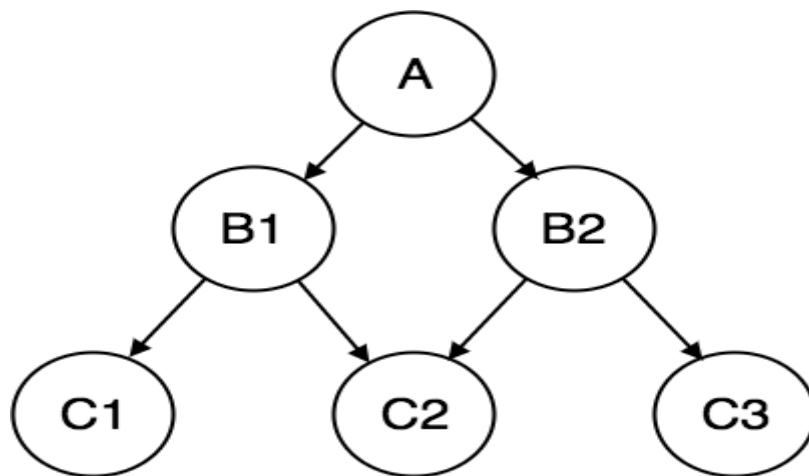


➤ **Network Model**

This is an extension of the Hierarchical model. In this model data is organised 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 (E/R) 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.

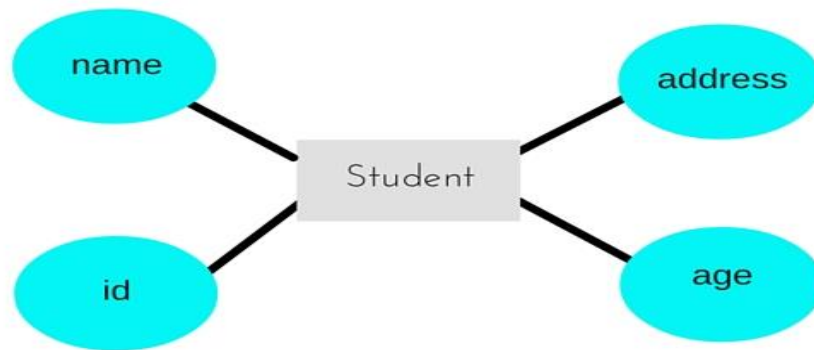
E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

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, pin code, city etc., and there will be a relationship between them.

Relationships can also be of different types.



➤ Relational Model

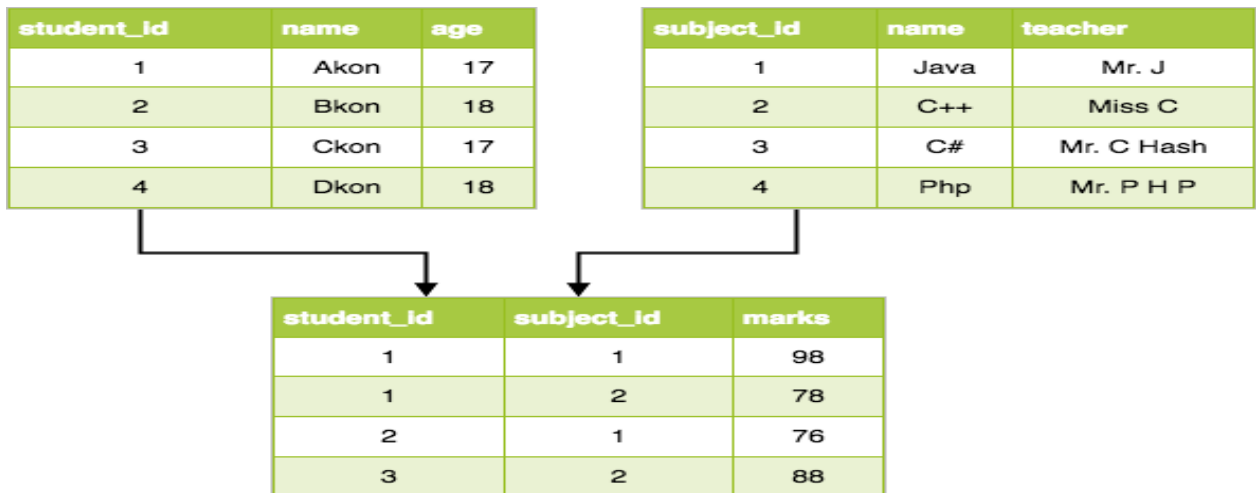
In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, in fact, 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.

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.



3.2.1 E/R model: Entity, Relationship, Attribute

➤ ENTITY

A real-world thing either living or non-living that is easily recognizable and nonrecognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.

Examples of entities:

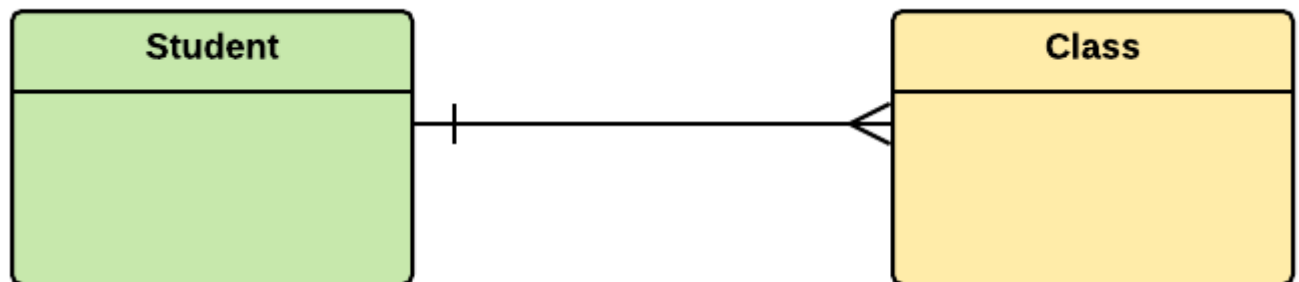
- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

Notation of an Entity

Entity set:

Student

An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values. Entities are represented by their properties, which also called attributes. All attributes have their separate values. For example, a student entity may have a name, age, class, as attributes.

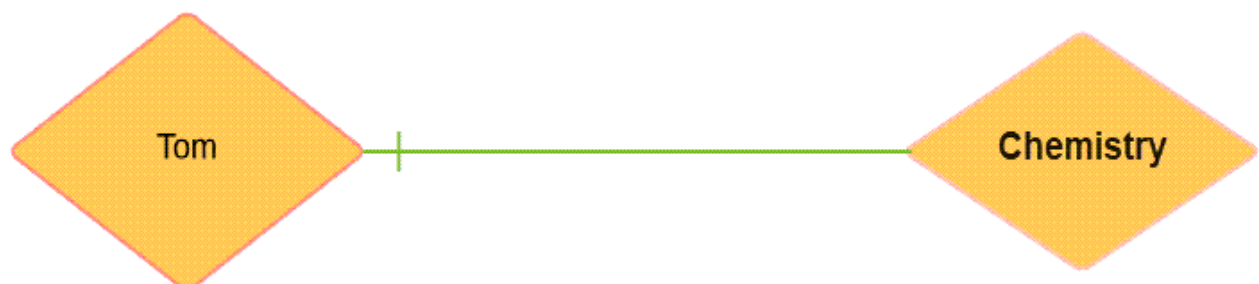
**Example of Entities:**

A university may have some departments. All these departments employ various lecturers and offer several programs.

Some courses make up each program. Students register in a particular program and enroll in various courses. A lecturer from the specific department takes each course, and each lecturer teaches a various group of students.

Relationship

Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.



Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

For example:

- You are attending this lecture

- I am giving the lecture

We can classify relationships according to relationship-types:

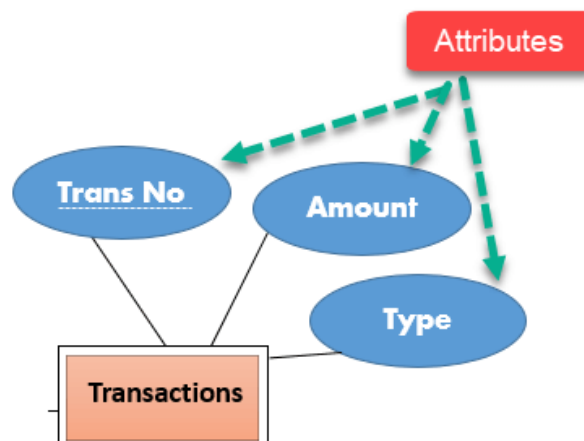
- A student attends a lecture
- A lecturer is giving a lecture.

➤ **Attributes**

It is a single-valued property of either an entity-type or a relationship-type.

For example, a lecture might have attributes: time, date, duration, place, etc.

An attribute is represented by an Ellipse



3.2.2 E/R Diagram: One to one, one to many, many to one, many to many

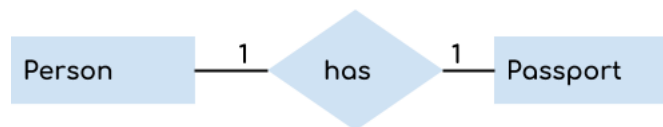
A relationship is represented by diamond shape in ER diagram; it shows the relationship among entities.

There are four types of relationships:

1. One to One
2. One to Many
3. Many to One
4. Many to Many

1. One to One Relationship

When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship. For example, a person has only one passport and a passport is given to one person.



2. One to Many Relationship

When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship. For example – a customer can place many orders but an order cannot be placed by many customers.



3. Many to One Relationship

When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship. For example – many students can study in a single college but a student cannot study in many colleges at the same time.



4. Many to Many Relationship

When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship. For example, a can be assigned to many projects and a project can be assigned to many students.



3.2.3 Strong entity, weak entity

➤ Strong Entity

The strong entity has a primary key. Weak entities are dependent on strong entity. Its existence is not dependent on any other entity.

Strong Entity is represented by a single rectangle –

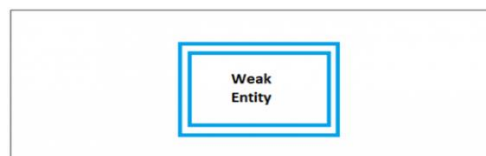


Continuing our previous example, **Professor** is a strong entity here, and the primary key is Professor **ID**.

➤ Weak Entity

The weak entity in DBMS do not have a primary key and are dependent on the parent entity. It mainly depends on other entities.

Weak Entity is represented by double rectangle –



Continuing our previous example, **Professor** is a strong entity, and the primary key is **Professor_ID**. However, another entity is **Professor_Dependents**, which is our Weak Entity.

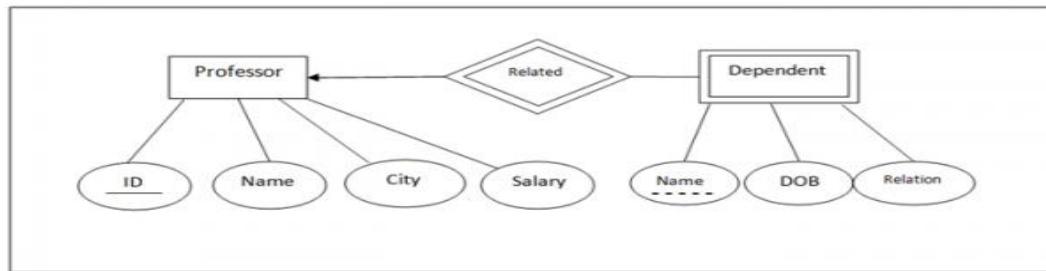
<Professor_Dependents>

Name	DOB	Relation
------	-----	----------

This is a weak entity since its existence is dependent on another entity **Professor**, which we saw above. A Professor has Dependents.

Example of Strong and Weak Entity

The example of strong and weak entity can be understood by the below figure.



The Strong Entity is **Professor**, whereas **Dependent** is a Weak Entity.

ID is the primary key (represented with a line) and Name in **Dependent** entity is called **Partial Key** (represented with a dotted line).

Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity	The member of a weak entity

set is called as dominant entity set.	set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entities set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

3.2.4 Key attribute, derived attribute, Multi-valued attribute

➤ **Key Attribute –**

The attribute which **uniquely identifies each entity** in the entity set is called key attribute. For example, Roll_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.



➤ **Derived Attribute**

An attribute which can be **derived from other attributes** of the entity type is known as derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, derived attribute is represented by dashed oval.



➤ **Multivalued Attribute**

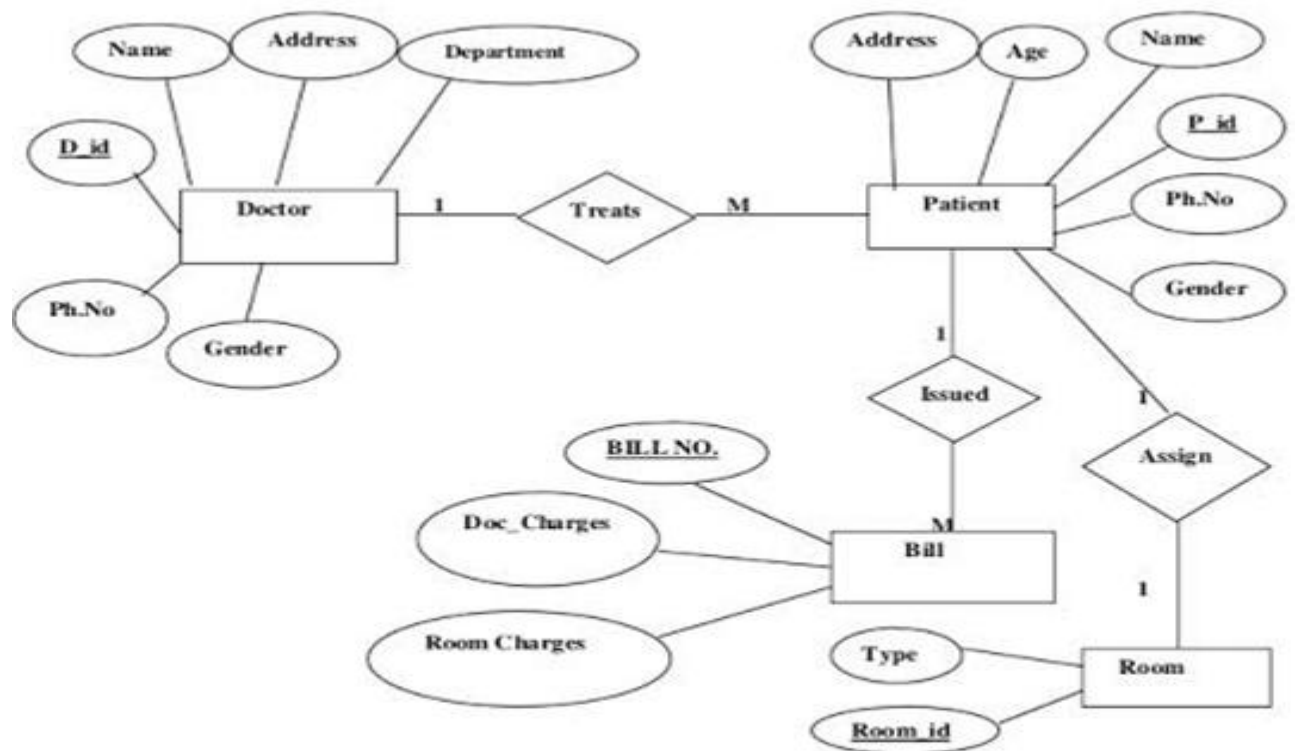
An attribute consisting **more than one value** for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, multivalued attribute is represented by double oval.



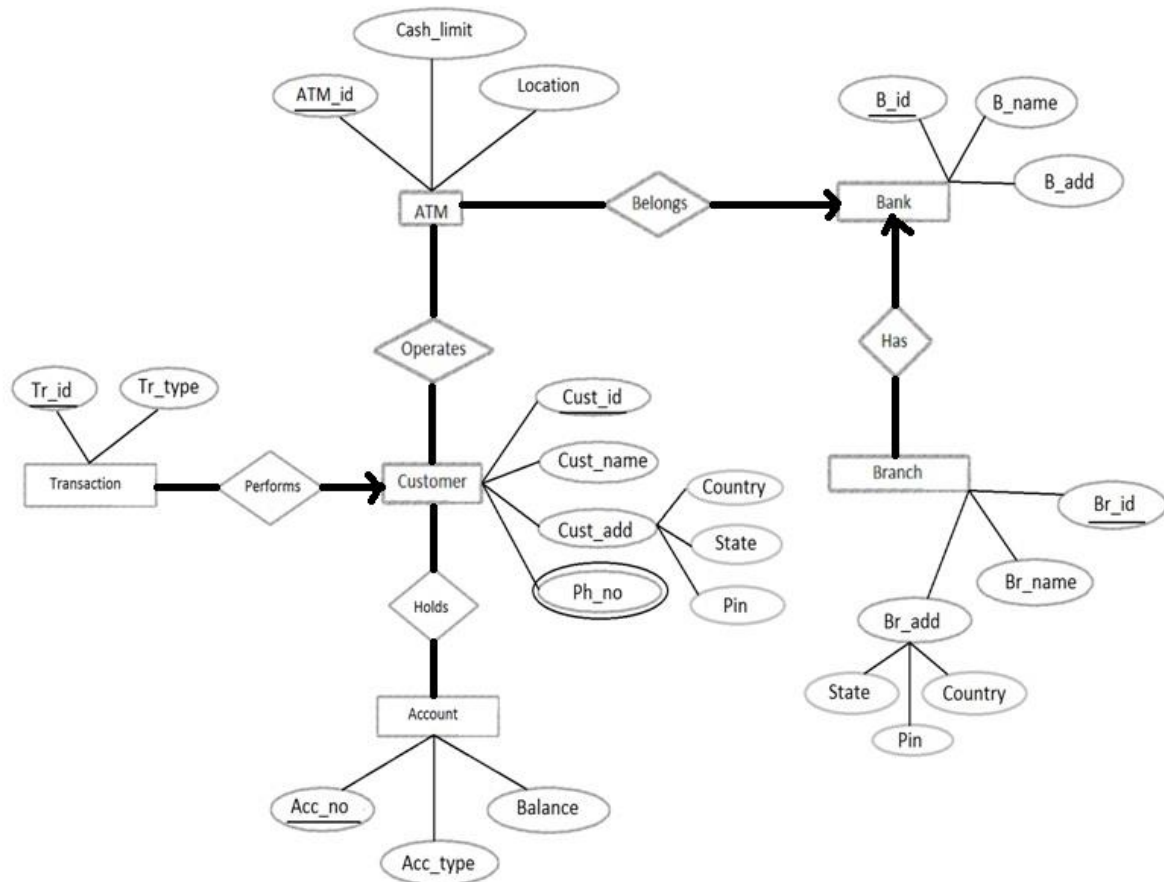
Examples of ER Diagram:-

1) Hospital

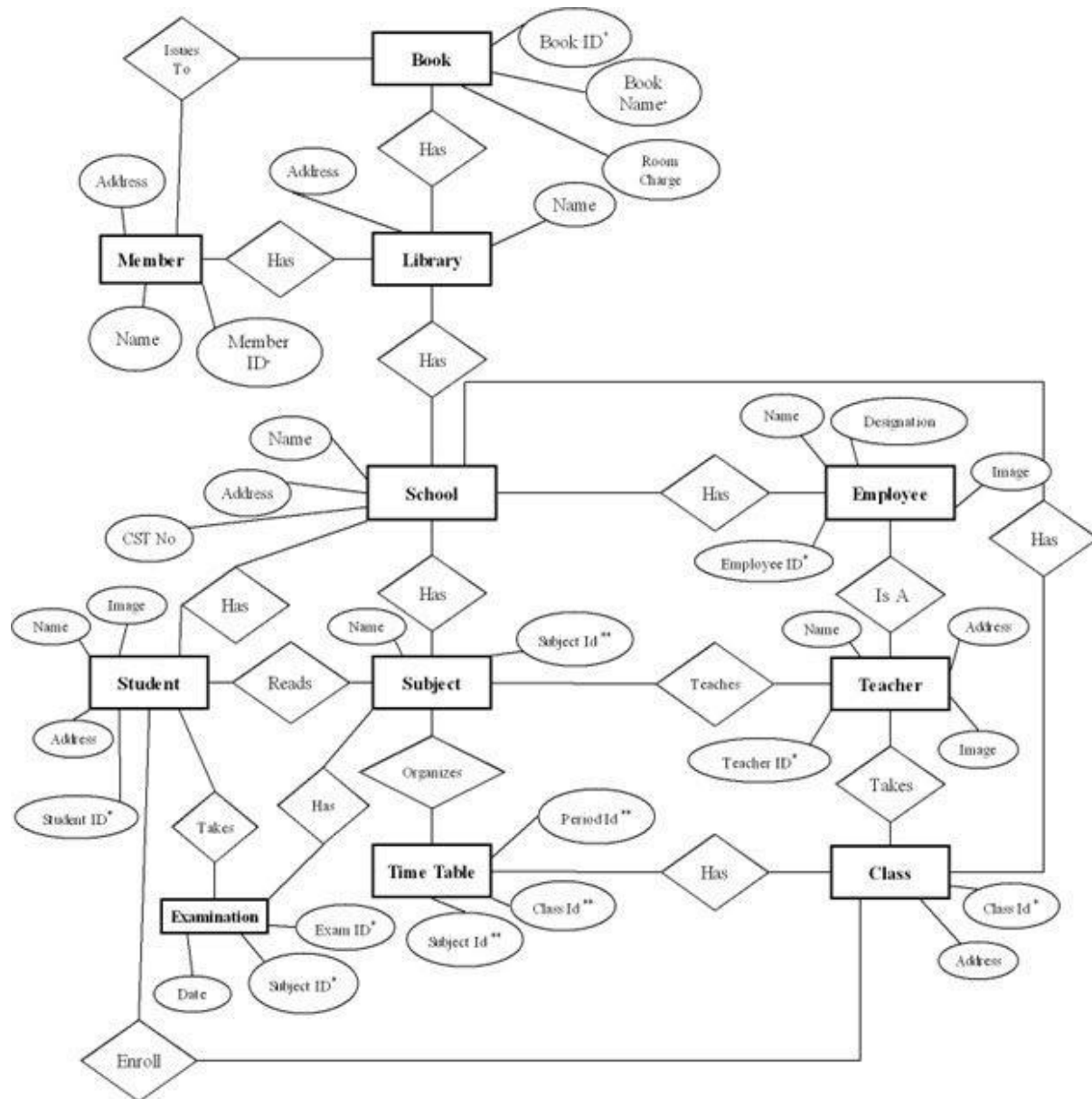
Management



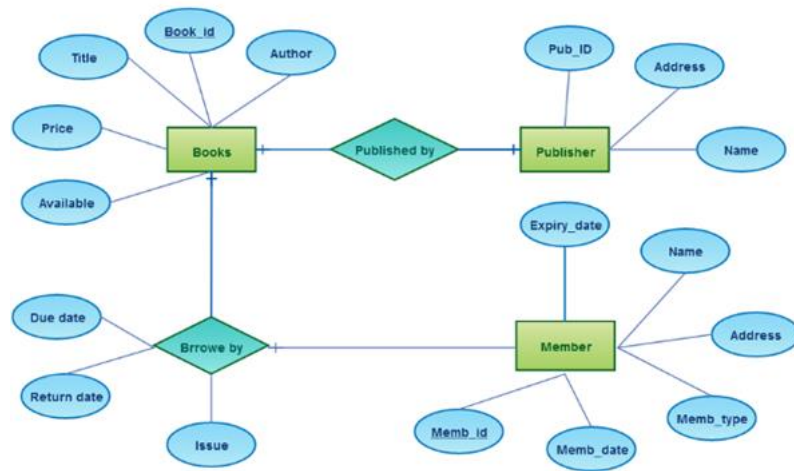
2) Banking management system



3) School/College management system



4) Library management System



3.3 Types of keys:

3.3.1 Super key, candidate key, Primary key, Composite key, Foreign key, Unique key.

➤ Super key

A superkey is a group of single or multiple keys which identifies rows in a table. A Super key may have additional attributes that are not needed for unique identification.

Example:

EmpSSN	EmpNum	Empname
9812345098	AB05	Shown
9876512345	AB06	Roslyn
199937890	AB07	James

In the above-given example, EmpSSN and EmpNum name are superkeys.

➤ Candidate Key

CANDIDATE KEY is a set of attributes that uniquely identify tuples in a table. Candidate Key is a super key with no repeated attributes. The Primary key should be selected from the candidate keys. Every table must have at least a single candidate key. A table can have multiple candidate keys but only a single primary key.

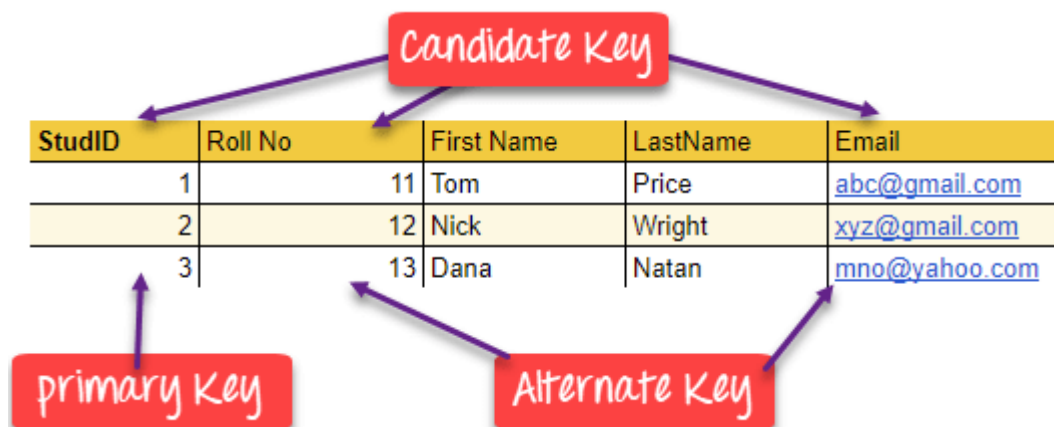
Properties of Candidate key:

- It must contain unique values
- Candidate key may have multiple attributes
- Must not contain null values
- It should contain minimum fields to ensure uniqueness

- Uniquely identify each record in a table

Example: In the given table Stud ID, Roll No, and email are candidate keys which help us to uniquely identify the student record in the table.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	abc@gmail.com
2	12	Nick	Wright	xyz@gmail.com
3	13	Dana	Natan	mno@yahoo.com



➤ Primary Key

PRIMARY KEY is a column or group of columns in a table that uniquely identify every row in that table. The Primary Key can't be a duplicate meaning the same value can't appear more than once in the table. A table cannot have more than one primary key.

Rules for defining Primary key:

- Two rows can't have the same primary key value

- It must for every row to have a primary key value.
- The primary key field cannot be null.
- The value in a primary key column can never be modified or updated if any foreign key refers to that primary key.

Example:

In the following example, StudID is a Primary Key.

StudID	Roll No	First Name	LastName	Email
1	11	Tom	Price	<u>abc@gmail.com</u>
2	12	Nick	Wright	<u>xyz@gmail.com</u>
3	13	Dana	Natan	<u>mno@yahoo.com</u>

➤ Foreign key

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. It acts as a cross-reference between two tables as it references the primary key of another table.

Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

In this example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.

In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

This concept is also known as Referential Integrity.

➤ Composite key

COMPOSITE KEY is a combination of two or more columns that uniquely identify rows in a table. The combination of columns guarantees uniqueness, though individually uniqueness is not guaranteed. Hence, they are combined to uniquely identify records in a table.

The difference between compound and the composite key is that any part of the compound key can be a foreign key, but the composite key may or maybe not a part of the foreign key.

➤ **Unique Key**

A unique key is a group of one or more than one fields or columns of a table which uniquely identify database record.

A unique key is the same as a primary key, but it can accept one null value for a table column. It also cannot contain identical values. Unique constraints are referenced by the foreign key of other tables.

Stud ID can have a unique constraint where entries in Stud ID column can be unique because each student of a university must have a unique ID number. In case, if student is changing the university, in that case, he or she would not have any stud ID. The entry may have a null value as only one null is allowed in the unique key constraint.

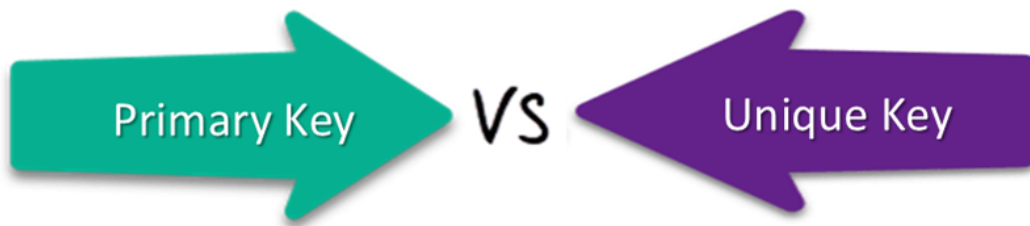
Table: Student

Stud ID	Roll No	First Name	Last Name	Email
---------	---------	------------	-----------	-------

↑
Unique key

- The purpose of a unique key is to make sure that information in the column for each table record is unique.
- When you allow the user to enter the null value.
- Unique key is used because it creates a non-clustered index by default.
- Unique key can be used when you have to keep null values in column.
- When one or more than one field/columns of a table that uniquely identify a record in a database table.

Difference between Primary key and unique key



Here are the important differences between primary key and unique key:

Primary Key	Unique Key
There can be one primary key in a table	There can be multiple unique keys in the table
It does not allow null columns.	It allows null columns.
Default Index is clustered	Default Index is no-clustered
The purpose of the primary key is to enforce entity integrity.	The purpose of unique key is to enforce unique data.
Primary key can be created using syntax: <pre>CREATE TABLE Employee (ID int PRIMARY KEY, Name varchar(255), City varchar(150))</pre>	Unique key can be created using syntax: <pre>CREATE TABLE Employee (ID int UNIQUE, Name varchar(255) NOT NULL, City varchar(150))</pre>

It is SQL constraint which allows you to uniquely identify each record or row in the database table.

It is SQL constraint that does not allow the same value to be assigned to two isolated records in a database table.

In the primary key, duplicate keys are not allowed.

In a unique key, if one or more key parts are null, then duplicate keys are allowed.

Normalization

4.1 Why Normalization

- Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

Anomalies in DBMS

- There are three types of anomalies that occur when the database is not normalized.
- These are – Insertion, update and deletion anomaly.

Let's take an example to understand this.

Example:

Suppose a manufacturing company stores the employee details in a table named employee that has four attributes:

- EMP_ID for storing employee's id,
- EMP_NAME for storing employee's name,
- EMP_ADDRESS for storing employee's address
- and EMP_DEPT for storing the department details in which the employee works.

At some point of time the table looks like this:

EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_DEPT
101	RICHA	DELHI	D001
101	RICHA	DELHI	D002
123	MEGHA	AGRA	D890
166	GANESH	CHENNAI	D900
166	GANESH	CHENNAI	D004

The above table is not normalized. We will see the problems that we face when a table is not normalized.

Update anomaly:

- An update anomaly is a data inconsistency that results from data redundancy and a partial update.
- In the above table we have two rows for employee Richa as she belongs to two departments of the company.
- If we want to update the address of Richa then we have to update the same in two rows or the data will become inconsistent.
- If somehow, the correct address gets updated in one department but not in other then as per the database, Richa would be having two different addresses, which is not correct and would lead to inconsistent data.

Insert anomaly:

- An **insertion anomaly** is the inability to add data to the database due to absence of other data.
- Suppose a new employee joins the company, who is under training and currently not assigned to any department then we would not be able to insert the data into the table if emp_dept field doesn't allow nulls.

Delete anomaly:

- A **deletion anomaly** is the unintended loss of data due to deletion of other data.
- Suppose, if at a point of time the company closes the department D890 then deleting the rows that are having emp_dept as D890 would also delete the information of

employee Megha since she is assigned only to this department.

To overcome these anomalies we need to normalize the data.

4.2 Normalization Rules

4.2.1 Dependency

Dependencies in DBMS is a relation between two or more attributes. It has the following types in DBMS

- Functional Dependency
- Fully-Functional Dependency
- Transitive Dependency
- Multivalued Dependency
- Partial Dependency

Prime Attribute and Non Prime Attribute:-

- Attributes that form a candidate key of a relation, i.e. attributes of candidate key, are called prime attributes. And rest of the attributes of the relation are non prime.
- **e.g.** for the relation {rollno, name, city, phoneno}
- {rollno} is a candidate key and hence rollno is a prime attribute and name, city, phoneno are non prime attributes.

Functional Dependency

If the information stored in a table can uniquely determine another information in the same table, then it is called Functional Dependency.

- Consider it as an association between two attributes of the same relation.
- If P functionally determines Q, then $P \rightarrow Q$

Example

EmpID	EmpName	EmpAge
E01	Amit	28
E02	Rohit	31

In the above table, EmpName is functionally dependent on EmpID because EmpName can take only one value for the given value of EmpID.

$\text{EmpID} \rightarrow \text{EmpName}$

The same is displayed below –



Fully-functionally Dependency

- An attribute is fully functional dependent on another attribute, if it is Functionally Dependent on that attribute and not on any of its proper subset.
- **For example**, an attribute Q is fully functional dependent on another attribute P, if it is Functionally Dependent on P and not on any of the proper subset of P.

Example

Table :ProjectCost

ProjectID	ProjectCost
001	1000
002	5000

Table :EmployeeProject

EMPID	PROJECTID	DAYS (SPENT ON THE PROJECT)
E099	001	320
E056	002	190

The above relations states:

- EMPID, PROJECTID, PROJECTCOST -> DAYS
- However, it is not fully functional dependent.
- Whereas the subset {EMPID, PROJECTID} can easily determine the {DAYS} spent on the project by the employee.
- This summarizes and gives our fully functional dependency –
- {EMPID, PROJECTID} -> (DAYS)

Partial Dependency

- Partial Dependency occurs when a nonprime attribute is functionally dependent on part of a candidate key.

Example

Table :StudentProject

StudentID	ProjectNo	StudentName	ProjectName
S01	199	Kalpesh	Geo Location
S02	120	Smita	Cluster Exploration

In the above table, we have partial dependency; let us see how —

- The prime key attributes are STUDENTID and PROJECTNO.
- As stated, the non-prime attributes i.e. STUDENTNAME and PROJECTNAME should be functionally dependent on part of a candidate key, to be Partial Dependent.
- The STUDENTNAME can be determined by STUDENTID that makes the relation Partial Dependent.
- The ProjectName can be determined by PROJECTID, which that the relation Partial Dependent.

Transitive Dependency

- By its nature, a transitive dependency requires three or more attributes (or database columns) that have a functional dependency between them, meaning that

Column A in a table relies on Column B through an intermediate Column C.

Example

Table: - AUTHORS

AUTHOR_ID	AUTHOR	BOOK	AUTHOR_CITY
AUTH_001	BHUSAN TRIVEDI	NETWORKING	AHEMDABAD
AUTH_001	BHUSAN TRIVEDI	ARTIFICIAL INTELLIGENCE	AHEMDABAD
AUTH_002	PRASHANT DOLIA	DATABASE CONCEPTS	SURAT

In the AUTHORS example above:

BOOK → AUTHOR:

- Here, the Book attribute determines the Author attribute.
- If you know the book name, you can learn the author's name.
- However, Author does not determine Book, because an author can write multiple books.
- For example, just because we know the author's name Bhusan Trivedi, we still don't know the book name.

AUTHOR → AUTHOR_CITY:

- Likewise, the Author attribute determines the Author_City, but not the other way around; just

because we know the City does not mean we can determine the author.

But this table introduces a transitive dependency:

BOOK $\rightarrow$ AUTHOR_CITY:

- If we know the book name, we can determine the City via the Author column.

Multi-valued dependency

- When existence of one or more rows in a table implies one or more other rows in the same table, then the Multi-valued dependencies occur.
- If a table has attributes P, Q and R, then Q and R are multi-valued facts of P.
- It is represented by double arrow $\rightarrow\rightarrow$
- For our example:
P $\rightarrow\rightarrow$ Q
P $\rightarrow\rightarrow$ R
- In the above case, Multivalued Dependency exists only if Q and R are independent attributes.

Example

Table: STUDENT

STUDENTNAME	COURSEDISCIPLINE	ACTIVITIES
AMIT	MATHEMATICS	SINGING
AMIT	MATHEMATICS	DANCING
YUVRAJ	COMPUTERS	CRICKET
AKASH	LITERATURE	DANCING
AKASH	LITERATURE	CRICKET

AKASH	LITERATURE	SINGING
-------	------------	---------

- In the above table, we can see Students AMIT and AKASH have interest in more than one activity.
- This is multivalued dependency because COURSEDISCIPLINE of a student are independent of Activities, but are dependent on the student.
- Therefore, multivalued dependency –
- STUDENTNAME $\twoheadrightarrow$ COURSEDISCIPLINE
STUDENTNAME $\twoheadrightarrow$ ACTIVITIES

4.2.2 Inference Rule (IR):

- The Armstrong's axioms are the basic inference rule.
- Armstrong's axioms are used to conclude functional dependencies on a relational database.
- The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
- Using the inference rule, we can derive additional functional dependency from the initial set.
- The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR1)

- In the reflexive rule, if Y is a subset of X, then X determines Y.
- If $X \supseteq Y$ then $X \rightarrow Y$

- **Example:**

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

$$Y = \{a, b, c\}$$

2. Augmentation Rule (IR2)

- 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 $XZ \rightarrow YZ$
- **Example:**

For R(ABCD), if $A \rightarrow B$ then $AC \rightarrow BC$

3. Transitive Rule (IR3)

- 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 (IR4)

- 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 $X \rightarrow YZ$
- **Proof:**

$$1. \quad X \rightarrow Y \quad (\text{given})$$

$$2. \quad X \rightarrow Z \quad (\text{given})$$

$$3. \quad X \rightarrow XY \text{ (using IR2 on 1 by augmentation with X. Where } XX = X)$$

$$4. \quad XY \rightarrow YZ \text{ (using IR2 on 2 by augmentation with X)}$$

Y)

5. $X \rightarrow YZ$ (using IR3 on 3 and 4)

5. Decomposition Rule (IR5)

- 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 $X \rightarrow Z$
- **Proof:**

1. $X \rightarrow YZ$ (given)
2. $YZ \rightarrow Y$ (using IR1 Rule)
3. $X \rightarrow Y$ (using IR3 on 1 and 2)

6. Pseudo transitive Rule (IR6)

- In Pseudo transitive Rule, if X determines Y and YZ determines W , then XZ determines W .
- If $X \rightarrow Y$ and $YZ \rightarrow W$ then $XZ \rightarrow W$
- **Proof:**

1. $X \rightarrow Y$ (given)
2. $WY \rightarrow Z$ (given)
3. $WX \rightarrow WY$ (using IR2 on 1 by augmenting with W)
4. $WX \rightarrow Z$ (using IR3 on 3 and 2)

4.2.3 Normalization

Here are the most commonly used normal forms:

- First normal form(1NF)
- Second normal form(2NF)

- Third normal form(3NF)
- Boyce & Codd normal form (BCNF)

First normal form (1NF)

- As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

Example: Suppose a company wants to store the names and contact details of its employees. It creates a table that looks like this:

EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_MOBILE
101	HERSHAL	NEW DELHI	8912312390
102	JIGAR	KANPUR	8812121212 9900012222
103	ROMA	CHENNAI	7778881212
104	LEENA	BANGALORE	9990000123 8123450987

- Two employees (JIGER & LEENA) are having two mobile numbers so the company stored them in the same field as you can see in the table above.
- This table is not in 1NF as the rule says “each attribute of a table must have atomic (single) values”, the EMP_MOBILE values for employees JIGER & LEENA violates that rule.

To make the table complies with 1NF we should have the data like this:

EMP_ID	EMP_NAME	EMP_ADDRESS	EMP_MOBILE
101	HERSHAL	NEW DELHI	8912312390
102	JIGER	KANPUR	8812121212
102	JIGER	KANPUR	9900012222
103	ROMA	CHENNAI	7778881212
104	LEENA	BANGALORE	9990000123
104	LEENA	BANGALORE	8123450987

Second normal form (2NF)

- A table is said to be in 2NF if both the following conditions hold:
- Table is in 1NF (First normal form)
- No non-prime attribute is dependent on the proper subset of any candidate key of table.
- An attribute that is not part of any candidate key is known as non-prime attribute.

Example: Suppose a school wants to store the data of teachers and the subjects they teach. They create a table that looks like this: Since a teacher can teach more than one subjects, the table can have multiple rows for a same teacher.

TEACHER_ID	SUBJECT	TEACHER_AGE
111	MATHS	38
111	PHYSICS	38
222	BIOLOGY	38
333	PHYSICS	40
333	CHEMISTRY	40

- Candidate Keys: {TEACHER_ID, SUBJECT}
- Non prime attribute: TEACHER_AGE

- The table is in 1 NF because each attribute has atomic values.
- However, it is not in 2NF because non prime attribute TEACHER_AGE is dependent on TEACHER_ID alone which is a proper subset of candidate key.
- This violates the rule for 2NF as the rule says “no non-prime attribute is dependent on the proper subset of any candidate key of the table”.

To make the table complies with 2NF we can break it in two tables like this:

TABLE : TEACHER_DETAILS

TEACHER_ID	TEACHER_AGE
111	38
222	38
333	40

TABLE : TEACHER_SUBJECT

TEACHER_ID	SUBJECT
111	MATHS
111	PHYSICS
222	BIOLOGY
333	PHYSICS
333	CHEMISTRY

Now the tables comply with Second normal form (2NF).

Third Normal form (3NF)

- A table design is said to be in 3NF if both the following conditions hold:
- Table must be in 2NF

- Transitive functional dependency of non-prime attribute on any super key should be removed.
- An attribute that is not part of any candidate key is known as non-prime attribute.
- In other words 3NF can be explained like this: A table is in 3NF if it is in 2NF and for each functional dependency $X \rightarrow Y$ at least one of the following conditions hold:
 - X is a super key of table
 - Y is a prime attribute of table

Example: Suppose a company wants to store the complete address of each employee, they create a table named EMPLOYEE_DETAILS that looks like this:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY	EMP_DISTRICT
1001	JOHN	282005	UP	AGRA	DAYAL BAGH
1002	AJEET	222008	TN	CHENNAI	M-CITY
1006	LORA	282007	TN	CHENNAI	URRAPAKKAM
1101	LILLY	292008	UK	PAURI	BHAGWAN
1201	STEVE	222999	MP	GWALIOR	RATAN

- Super keys: {EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}...so on
- Candidate Keys: {EMP_ID}

- Non-prime attributes: all attributes except emp_id are non-prime as they are not part of any candidate keys.
- Here, EMP_STATE, EMP_CITY & EMP_DISTRICT dependent on EMP_ZIP. And, EMP_ZIP is dependent on EMP_ID that makes non-prime attributes (EMP_STATE, EMP_CITY & EMP_DISTRICT) transitively dependent on super key (EMP_ID). This violates the rule of 3NF.

To make this table complies with 3NF we have to break the table into two tables to remove the transitive dependency:

Table :EMPLOYEE TABLE

EMP_ID	EMP_NAME	EMP_ZIP
1001	JOHN	282005
1002	AJEET	222008
1006	LORA	282007
1101	LILLY	292008
1201	STEVE	222999

Table :EMPLOYEE_ZIP

EMP_ZIP	EMP_STATE	EMP_CITY	EMP_DISTRICT
282005	UP	AGRA	DAYAL BAGH
222008	TN	CHENNAI	M-CITY
282007	TN	CHENNAI	URRAPAKKAM
292008	UK	PAURI	BHAGWAN
222999	MP	GWALIOR	RATAN

Boyce Codd normal form (BCNF)

- It is an advance version of 3NF that's why it is also referred as 3.5NF.
- BCNF is stricter than 3NF. A table complies with BCNF if it is in 3NF and for every functional dependency $X \rightarrow Y$, X should be the super key of the table.

Example: Suppose there is a company wherein employees work in more than one department. They store the data like this:

EMP_ID	EMP_NATIONALITY	EMP_DEPT	DEPT_TYPE	DEPT_NO_OF_EMP
1001	AUSTRIAN	PRODUCTION AND PLANNING	D001	200
1001	AUSTRIAN	STORES	D001	250
1002	AMERICAN	DESIGN AND TECHNICAL SUPPORT	D134	100
1002	AMERICAN	PURCHASING DEPARTMENT	D134	600

- Functional dependencies in the table above:
 $EMP_ID \rightarrow EMP_NATIONALITY$
 $EMP_DEPT \rightarrow \{DEPT_TYPE, DEPT_NO_OF_EMP\}$
- Candidate key: $\{EMP_ID, EMP_DEPT\}$
- The table is not in BCNF as neither EMP_ID nor EMP_DEPT alone are keys.

To make the table comply with BCNF we can break the table in three tables like this:

Table :EMP_NATIONALITY

EMP_ID	EMP_NATIONALITY
1001	AUSTRIAN
1002	AMERICAN

Table :EMP_DEPT TABLE

EMP_DEPT	DEPT_TYPE	DEPT_NO_OF_EMP
PRODUCTION AND PLANNING	D001	200
STORES	D001	250
DESIGN AND TECHNICAL SUPPORT	D134	100
PURCHASING DEPARTMENT	D134	600

Table :EMP_DEPT_MAPPING

EMP_ID	EMP_DEPT
1001	PRODUCTION AND PLANNING
1001	STORES
1002	DESIGN AND TECHNICAL SUPPORT
1002	PURCHASING DEPARTMENT

- Functional dependencies:
EMP_ID -> EMP_NATIONALITY
EMP_DEPT -> {DEPT_TYPE, DEPT_NO_OF_EMP}
- Candidate keys:
For first table: EMP_ID
For second table: EMP_DEPT
For third table: {EMP_ID, EMP_DEPT}
- This is now in BCNF as in both the functional dependencies left side part is a key.