DATABASE MANAGEMENT SYSTEMS
MODULE-1

Contents:

Introduction DBMS

Data: The facts that can be recorded and which have implicit meaning known as 'data'.
Data can be represented in the form of text, number, image, video.
Example: If You Consider Student -----
1. sname.
2. sno.
3. smarks.

Meta Data: Data about data is called Meta data.

Database: It is a collection of interrelated data which can be stored in the form of tables.
A database can be of any size and varying complexity.
A database may be generated and manipulated manually or it may be computerized.
Example: If you consider Student database consists the fields as sname, sno, and smarks

Sname	Sno	Smarks

Operations perform on database are

Insertion: To add the new information.

Update: To modify the existing information.

Deletion: To remove the unwanted information.

Retrieve: To view or retrieving stored information.

Sorting: Arrange the information in ascending or descending order.

Database Management System (DBMS): It is a collection of programs that enables user to create and maintain a database. In other words, it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

Components of DBMS: The DBMS has five important components in it which plays a major role in it and they are as follows:

Hardware: The hardware is nothing but the actual computer system which is used for keeping and accessing database and DBMS hardware has the secondary storage devices like the hard disks, database machines. These secondary storage devices are designed specifically to support the database.

Software: Software is the actual DBMS and between the actual stored data and the users of the system there is a presence of a layer of software called DBMS. It can control the access and can maintain the consistency of the information.

Data: The most vital component of the DBMS environment is none other than the data from the user point of view. It acts as a machine between the machine components and the user components, database should contain all the data needed by the organization i.e. user.

Users: user can access the data on demand by using the applications and interfaces provided by the database management system.

Procedures: Procedures refer to the instructions and rules that govern the design of database along with that we require the following procedure to run the system:

1.Log onto DBMS
2.Use a particular DBMS facility
3.Start and stop the DBMS
4.Make backup copies of database
5.Handle hardware or software failures

DBMS USERS: This differentiation is made according to the interaction of users to the database. Database system is made to store information and provide an environment for retrieving information. There are four types of database users in DBMS.

Application Programmers: Application programmers are the one who writes application programs that uses the database. These application programs are written in programming languages like COBOL or PL (Programming Language 1), Java and fourth generation language. These programs meet the user requirement and made according to user requirements. Retrieving information, creating new information and changing existing information is done by these application programs.

End Users: End users are those who access the database from the terminal end. They use the developed applications and they don’t have any knowledge about the design and working of database. These are the second class of users and their main motto is just to get their task done. There are basically two types of end users that are discussed below.

1. **Casual User:** These users have great knowledge of query language. Casual users access data by entering different queries from the terminal end. They do not write programs but they can interact with the system by writing queries.
2. **Naive User:** Any user who does not have any knowledge about database can be in this category. Their task is to just use the developed application and get the desired results. For example: Clerical staff in any bank is a naïve user. They don't have any DBMS knowledge but they still use the database and perform their given task.

DBA (Database Administrator): DBA can be a single person or it can be a group of person. Database Administrator is responsible for everything that is related to database. i.e. The DBA is responsible for authorizing access to the database by grant and revoke permissions to the users, for coordinating and monitoring its use, managing backups and repairing damage due to hardware and/or software failures and for acquiring hardware and software resources as needed. In case of small organization, the role of DBA is performed by a single person and in case of large organizations there is a group of DBA's who share responsibilities. He makes the policies, strategies and provides technical supports.

System Analyst: System analyst is responsible for the design, structure and properties of database. All the requirements of the end users are handled by system analyst. Feasibility, economic and technical aspects of DBMS is the main concern of system analyst.

Examples of Popular Database Management Systems (DBMS):

Ms-Access, MySQL Database, Oracle Database, Microsoft SQL Server etc.

25.08.23

File: A file is a sequence of records stored in binary format, Relative data and information is stored collectively in file formats.

File Oriented Approach: The earliest business computer systems were used to process business records and produce information. They were generally faster and more accurate than equivalent manual systems. These systems stored groups of records in separate files, and so they were called **file processing systems.** In a typical file processing system, each department has its own files, designed specifically for those applications. The department itself working with the data processing staff, sets policies or standards for the format and maintenance of its files.

Disadvantages of File Processing System:

1. **Duplicate Data**: Data is stored more than once in different files, that means duplicate data may occur in all these files. Since all the files are independent on each other so it is very difficult to overcome this error and if anyone finds this error then it will take time and effort to solve this issue.

For Example: A student is having record in college library and in Examination department. Then his name, roll number, fathers name and class will be same in both the departments. Also these departments are not dependent on each other. So it creates lots of duplicates value about that student and when he needs any change for his name or class then he has to go to both the departments to make these changes happen otherwise it will create problem for him.

2. **Inconsistency:** In file processing system, various copies of same data may contain different values. Data is not consistent in this system; it means if a data item needs to be changed then all the files containing that data need to be modified. It may create a risk of out dated values of data.

   For Example: If you change student name in library then his name should be changed in all the departments related to the student.

3. **Access Anomalies:** Accessing anomalies means that it is not easy to access data in a desired or efficient way. It makes supervision of department very difficult. If a user wants information in a specific manner, then he requires creating a program for it.

   For Example: Let's say, if admin of the college wants any student information like his name, fathers name, roll number, marks and class then program for it is written but if he wants records of whose students whose numbers are more than 80 percent then he requires to create a different program for it.

4. **Poor Data Integrity:** A collection of data is integrated if it meets certain consistency constraints. A programmer always puts these constraints in the programs by adding some codes. In File Processing System, poor data integrity often arises and it becomes very difficult to add new constraints at that time.

   For Example: The maximum marks of the student can never be more than 100.

5. **Poor Data Security:** Poor data security is the most threatening problem in File Processing System. There is very less security in File Processing System as anyone can easily modify and change the data stored in the files. All the users must have some restriction of accessing data up to a level.

   For Example: If a student can access his data in the college library then he can easily change books issued date. Also he can change his fine detains to zero.

Advantages and Disadvantages of Database Management System:

We must evaluate whether there is any gain in using a DBMS over a situation where we do not use it. Let us summarize the advantages.

1. **Reduction of Redundancy:** This is perhaps the most significant advantage of using DBMS. Redundancy is the problem of storing the same data item in more one place. Redundancy creates several problems like requiring extra storage space, entering same data more than once during data insertion, and deleting data from more than one place during deletion. Anomalies may occur in the database if insertion, deletion etc. are not done properly.

2. **Sharing of Data**: In a paper-based record keeping, data cannot be shared among many users. But in computerized DBMS, many users can share the same database if they are connected via a network.

3. **Data Integrity:** We can maintain data integrity by specifying integrity constrains, which are rules and restrictions about what kind of data may be entered or manipulated within the database. This increases the reliability of the database as it can be guaranteed that no wrong data can exist within the database at any point of time.

4. **Data Security:** We can restrict certain people from accessing the database or allow them to see certain portion of the database while blocking sensitive information. This is not possible very easily in a paper-based record keeping.

5. **Data Consistency:** DBMS controls data redundancy which in turn controls data consistency. Data consistency means if you want to update data in any files then all the files should not be updated again. As in DBMS, data is stored in a single database so data becomes more consistent in comparison to file processing system. Also updated values are available to all the users immediately.

6. **Efficient data access:** A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

7. **Data administration:** When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and fine-tuning the storage of the data to make retrieval efficient.

8. **Concurrent access and Crash recovery:** A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.

Disadvantages of DBMS:

1. **Cost:** DBMS requires high initial investment for hardware, software and trained staff. A significant investment based upon size and functionality of organization if required. Also organization has to pay concurrent annual maintenance cost.

2. **Complexity:** A DBMS fulfil lots of requirement and it solves many problems related to database. But all these functionality has made DBMS an extremely complex software. Developer, designer, DBA and End user of database must have complete skills if they want to use it properly. If they don't understand this complex system, then it may cause loss of data or database failure.

3. **Database Failure:** As we know that in DBMS, all the files are stored in single database so chances of database failure become more. Any accidental failure of component may cause loss of valuable data. This is really a big question mark for big firms.

4. **Extra Cost of Hardware**: A DBMS requires disk storage for the data and sometimes you need to purchase extra space to store your data. Also sometimes you need to a dedicated machine for better performance of database. These machines and storage space increase extra costs of hardware.

5. **Size:** As DBMS becomes big software due to its functionalities so it requires lots of space and memory to run its application efficiently. It gains bigger size as data is fed in it.

Applications and Uses of Database Management System (DBMS)

Railway Reservation System: Database is required to keep record of ticket booking, train's departure and arrival status. Also if trains get late then people get to know it through database update.

Library Management System: There are thousands of books in the library so it is very difficult to keep record of all the books in a copy or register. So DBMS used to maintain all the information relate to book issue dates, name of the book, author and availability of the book.

Banking: We make thousands of transactions through banks daily and we can do this without going to the bank. So how banking has become so easy that by sitting at home we can send or get money through banks. That is all possible just because of DBMS that manages all the bank transactions.

Universities and Colleges: Examinations are done online today and universities and colleges maintain all these records through DBMS. Student's registrations details, results, courses and grades all the information are stored in database.

Online shopping: Online shopping has become a big trend of these days. No one wants to go to shops and waste his time. Everyone wants to shop from home. So all these products are added and sold only with the help of DBMS. Purchase information, invoice bills and payment, all of these are done with the help of DBMS.

Social Media Sites: We all are on social media websites to share our views and connect with our friends. Daily millions of users signed up for these social media accounts like Facebook, twitter, Pinterest and Google plus. But how all the information of users is stored and how we become able to connect to other people, yes this all because DBMS.

Military: Military keeps records of millions of soldiers and it has millions of files that should be keep secured and safe. As DBMS provides a big security assurance to the military information so it is widely used in militaries. One can easily search for all the information about anyone within seconds with the help of DBMS.

Manufacturing: Manufacturing companies make products and sales them on the daily basis. To keep records of all the details about the products like quantity, bills, purchase, supply chain management, DBMS is used.

Airline Reservation System: Same as railway reservation system, airline also needs DBMS to keep records of flights arrival, departure and delay status.

Human Resource Management: Big firms have many workers working under them. Human resource management department keeps records of each employee's salary, tax and work through DBMS.

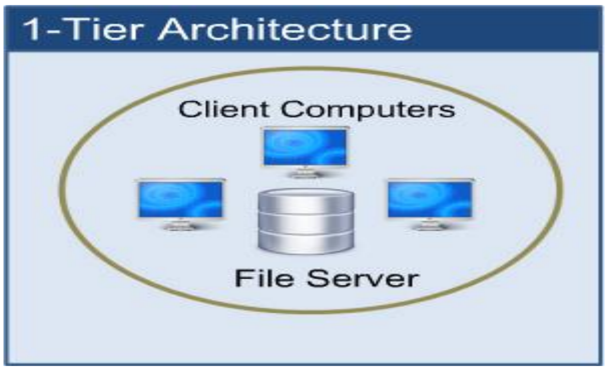
Credit Card Transactions: For purchase of credit cards and all the other transactions are made possible only by DBMS. A credit card holder knows the importance of their information that all are secured through DBMS.

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.

Database architecture uses programming languages to design a particular type of software for businesses or organizations. Database architecture focuses on the design, development, implementation and maintenance of computer programs that store and organize information for businesses, agencies and institutions. A database architect develops and implements software to meet the needs of users.
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. The tiers are classified as follows:
1-tier architecture
2-tier architecture
3-tier architecture

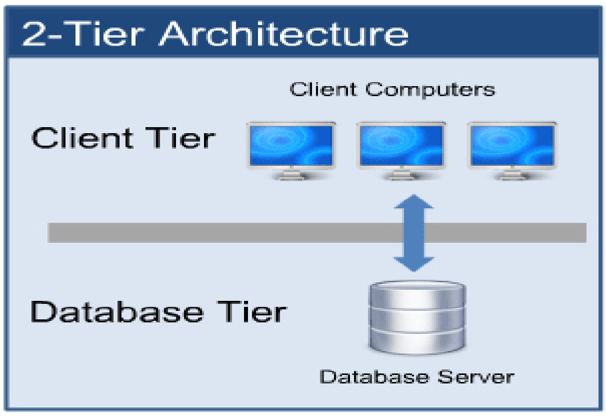
1-tier architecture:
One-tier architecture involves putting all of the required components for a software application or technology on a single server or platform.



1-tier architecture

Basically, a one-tier architecture keeps all of the elements of an application, including the interface, Middleware and back-end data, in one place. Developers see these types of systems as the simplest and most direct way.
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.

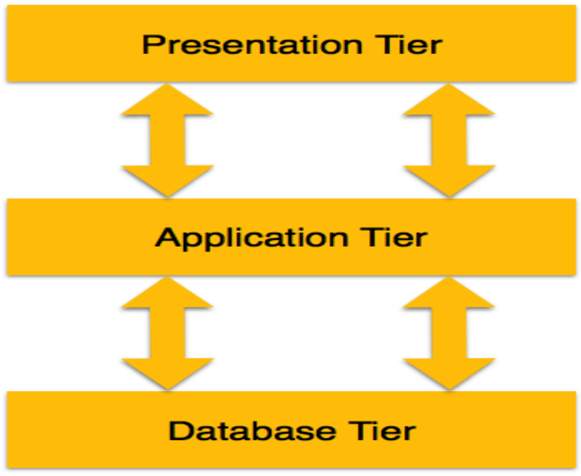
2-tier architecture:
The two-tier is based on Client Server architecture. The two-tier architecture is like client server application. The direct communication takes place between client and server. There is no intermediate between client and server.



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- 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.

Database Schema and Instance: The Description or design of a database is called the schema, which is specified during database design and is not expected to change frequently.
Most of the data models have certain conventions for displaying schemas in diagrams.
A displayed schema is called **schema diagram**.

Sub Schema:
It can be defined as the subset or sub-level of schema that has the same properties as the schema. In simple words it is just a effective plan or the schema for the view.

The schema diagram displays the structure of each record type but not the actual instance of records.

Example schema diagrams are

STUDENT

Name	SID	Class	Department Id

COURSE

Course Id	Course Name	Department Id

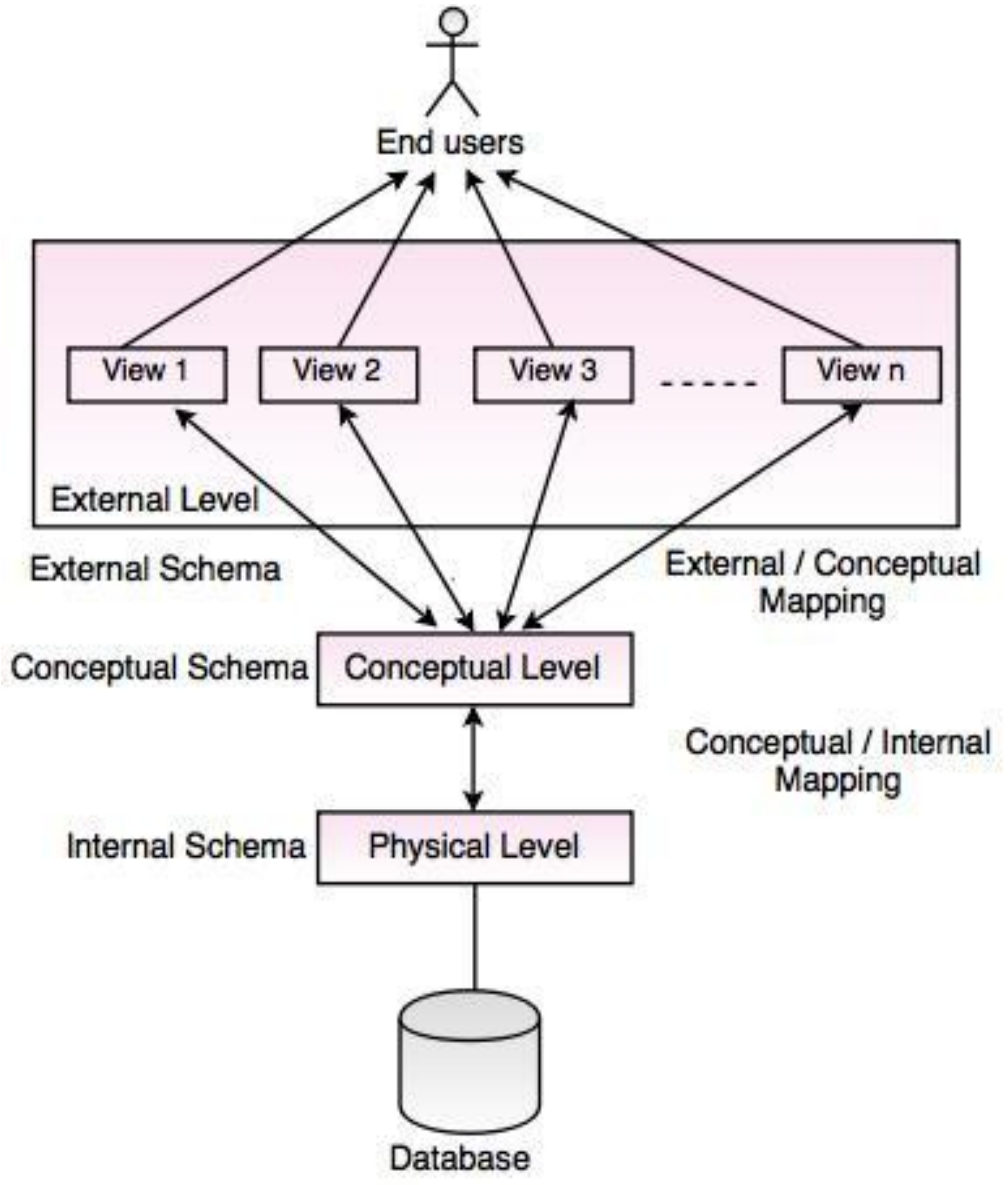
DEPARTMENT

Department Id	Department Name

The Three-Schema Architecture:

The goal of three-schema architecture is to separate the user applications and the physical database.

1. **Physical level/Internal Schema**: This is the lowest level of data abstraction. It describes how data is actually stored in database and it describes the complete details of data storage and access paths for the database. You can get the complex data structure details at this level.
2. **Logical level/Conceptual Schema**: Which describes the structure of the whole database for a community of users. This schema hides the details of physical storage structure and concentrate on describing entities, data types, relationships, user operations, and constraints. The implementation conceptual schema is often based on a conceptual schema design in a high-level data model.
3. **View level/External Schema**: This level describes the user interaction with database system. It hides the rest of database from that user group.



<u>Instance:</u> The data stored in a database at a particular moment of time is called instance of database or **database state**. Database schema defines the variable declaration in tables that belongs to a particular database, the value of that variables at a moment of the time is called instance of that database.

Database Languages: Database languages are used for read, update and store data in a database. There are several such languages that can be used for this purpose; one of them is SQL (Structured Query Language).

Types of DBMS Languages:

1. **Data Definition Language (DDL)**
2. **Data Manipulation Language (DML)**
3. **Data Control Language (DCL)**
4. **Transaction Control Language (TCL)**

Data Definition Language (DDL): DDL is used for used to define database patterns or structures.

- CREATE – used to create objects in database
- ALTER – alter the pattern of database
- DROP – helps in detecting objects
- TRUNCATE – erase all records from table
- COMMENT – adding of comments to data dictionary
- RENAME – useful in renaming an object

CREATE DATABASE Statement:

Syntax:

We use CREATE DATABASE statement in order to create a database. This is how it is used:

```
CREATE DATABASE DBName;
```

Here DBName can be any string that would represent the database name.

Example – The below statement would create a database named **employee**

```
SQL> CREATE DATABASE Employee;
```

CREATE TABLE Statement: CREATE TABLE statement is used for creating tables in a database. Tables are organized in rows and columns. Where columns are the attributes and rows are known as records.

Syntax:

```
CREATE TABLE tableName
(
columnName_1 data_type,
columnName_2 data_type,
columnName_3 data_type,
....
PRIMARY KEY (column_Name(s))
);
```

ALTER Statement: **ALTER s**tatement enhance the object of database. In structured query language it modifies the properties of database object.

Syntax: ALTER type of object name of object

DROP Statement: DROP statement destroys or deletes database or table. In structured query language, it also deletes an object from relational database management system.

Syntax: DROP type of object name of object

RENAME Statement: **RENAME** statement is used to rename a database.

RENAME TABLE old name of table to new name of table

Data Manipulation Language (DML): DML is used for accessing and manipulating data in a database.

- SELECT – useful in holding data from a database
- INSERT – helps in inserting data in to a table
- UPDATE – used in updating the data
- DELETE – do the function of deleting the records
- MERGE – this do the UPSERT operation i.e. insert or update operation
- CALL – this calls a structured query language or a java subprogram
- EXPLAIN PLAN – has the parameter of explaining data
- LOCK TABLE – this ha the function of controlling concurrency

SELECT Statement: Select query is used for fetching data from table(s). We have flexibility to fetch few columns, few rows or entire table using SELECT Query.

Syntax: SELECT * FROM table_name;

INSERT Statement: The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax-1:

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
```

Syntax-2:

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

UPDATE Statement: Update Query is used for updating existing rows(records) in a table

Syntax:

```
UPDATE TableName
SET column_name1 = value, column_name2 = value....
WHERE condition;
```

DELETE Statement: Delete Query is used for deleting the existing rows(records) from table. Generally, DELETE query is used along with WHERE clause to delete the certain number of rows that fulfils the specified condition. However, DELETE query can be used without WHERE clause too, in that case the query would delete all the rows of specified table.

Syntax: To Delete a particular set of rows:

```
DELETE FROM TableName
WHERE condition;
```

To Delete all the rows of a table:

```
DELETE FROM TableName;
```

Data Control Language (DCL): DCL is used for granting and revoking user access on a database

- GRANT - gives user's access privileges to database
- REVOKE - withdraw access privileges given with the GRANT command

Transaction Control Language (TCL): Transaction Control Language has commands which are used to manage the transactions or the conduct of a database. They manage the changes made by data manipulation language statements and also group up the statements in o logical management.

- COMMIT – use to save work
- SAVE POINT – helps in identifying a point in the transaction, can be rolled back to the identified point
- ROLL BACK – has the feature of restoring the database to the genuine point, since from the last COMMIT
- SET TRANSACTION – have parameter of changing settings like isolation level and roll back point

Database Constraints: Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity** (the overall completeness, assurance of the accuracy and consistency of data**)** during an update/delete/insert into a table.

Types of Constraints:

- NOT NULL
- UNIQUE
- DEFAULT

- CHECK
- Key Constraints – PRIMARY KEY, FOREIGN KEY
- Domain constraints

<u>NOT NULL:</u> NOT NULL constraint makes sure that a column does not hold NULL value. When we don't provide value for a particular column while inserting a record into a table, by default it takes NULL value. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.

Example: Here I am creating a table "STUDENTS". I have specified NOT NULL constraint for columns ROLL_NO, STU_NAME and STU_AGE which means you must provide the value for these three fields while inserting/updating records in this table. It enforces these column(s) not to accept null values.

```
CREATE TABLE STUDENTS(
       ROLL_NO    INT             NOT NULL,
       STU_NAME VARCHAR (35)      NOT NULL,
       STU_AGE   INT              NOT NULL,
       STU_ADDRESS  VARCHAR (235) ,
       PRIMARY KEY (ROLL_NO)
);
```

Specify the NULL constraint for already existing table:

In the above section we learnt how to specify the NULL constraint while creating a table. However, we can specify this constraint on a already present table also. For this we need to use ALTER TABLE statement.

```
ALTER TABLE STUDENTS
   MODIFY STU_ADDRESS VARCHAR (235) NOT NULL;
```

After this STU_ADDRESS column will not accept any null values.

<u>UNIQUE:</u> UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a Unique constraint, it means that particular column cannot have duplicate values in a table.

Example: Here we are setting up the UNIQUE Constraint for two columns: STU_NAME & STU_ADDRESS. which means these two columns cannot have duplicate values.

Note: STU_NAME column has two constraints (NOT NULL and UNIQUE both) setup.

```
CREATE TABLE STUDENTS(
       ROLL_NO    INT             NOT NULL,
       STU_NAME VARCHAR (35)      NOT NULL UNIQUE,
       STU_AGE   INT              NOT NULL,
       STU_ADDRESS  VARCHAR (35) UNIQUE,
       PRIMARY KEY (ROLL_NO)
);
```

Example to Set UNIQUE Constraint on already created table

```
ALTER TABLE STUDENTS
ADD UNIQUE (STU_AGE);
```

DEFAULT: The DEFAULT constraint provides a default value to a column when there is no value provided while inserting a record into a table.

Example: Here we are creating a table "STUDENTS", we have a requirement to set the exam fees to 10000 if fees is not specified while inserting a record (row) into the STUDENTS table. We can do so by using DEFAULT constraint. As you can see we have set the default value of EXAM_FEE column to 10000 using DEFAULT constraint.

```
CREATE TABLE STUDENTS(
       ROLL_NO    INT             NOT NULL,
       STU_NAME VARCHAR (35)    NOT NULL,
       STU_AGE  INT             NOT NULL,
       EXAM_FEE INT             DEFAULT 10000,
       STU_ADDRESS  VARCHAR (35) ,
       PRIMARY KEY (ROLL_NO)
);
```

Specify DEFAULT Constraint on already created table

Syntax:

```
ALTER TABLE <table_name>
  MODIFY <column_name> <column_data_type> DEFAULT <default_value>;
```

Example:

```
ALTER TABLE STUDENTS
  MODIFY EXAM_FEE INT DEFAULT 10000;
```

CHECK: This constraint is used for specifying range of values for a particular column of a table. When this constraint is being set on a column, it ensures that the specified column must have the value falling in the specified range.

```
CREATE TABLE STUDENT(
ROLL_NO   INT  NOT NULL CHECK(ROLL_NO >1000) ,
STU_NAME VARCHAR (35)  NOT NULL,
STU_AGE INT  NOT NULL,
EXAM_FEE INT DEFAULT 10000,
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO)
);
```

In the above example we have set the check constraint on ROLL_NO column of STUDENT table. Now, the ROLL_NO field must have the value greater than 1000.

Key Constraints:

1. **PRIMARY KEY:** A primary key is a column or set of columns in a table that uniquely identifies tuples (rows) in that table. It must have unique values and cannot contain nulls.

   In the below example the ROLL_NO field is marked as primary key, that means the ROLL_NO field cannot have duplicate and null values.

```
CREATE TABLE STUDENT(
ROLL_NO   INT  NOT NULL,
STU_NAME VARCHAR (35)  NOT NULL UNIQUE,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (35) UNIQUE,
PRIMARY KEY (ROLL_NO)
);
```

2. **FOREIGN KEY:** Foreign keys are the columns of a table that points to the primary key of another table. They act as a cross-reference between tables.

   Example: In the below example the Stu_Id column in Course_enrollment table is a foreign key as it points to the primary key of the Student table.

Course_Id	Stu_Id
C01	101
C02	102
C03	101
C05	102
C06	103
C07	102

Student table:

Stu_Id	Stu_Name	Stu_Age
101	Chaitanya	22
102	Arya	26
103	Bran	25
104	Jon	21

Note: Practically, the foreign key has nothing to do with the primary key tag of another table, if it points to a unique column (not necessarily a primary key) of another table then too, it would be a foreign key.

Domain Constraints: Domain constraints are **user defined data type** and we can define them like this:

Domain Constraint = data type + Constraints (NOT NULL / UNIQUE / PRIMARY KEY / FOREIGN KEY / CHECK / DEFAULT)

Example: For example I want to create a table “student_info” with “stu_id” field having value greater than 100, I can create a domain and table like this:

```
create domain id_value int
constraint id_test
check(value > 100);


create table student_info (
stu_id id_value PRIMARY KEY,
stu_name varchar(30),
stu_age int
);
```

Mapping Constraints:

Mapping constraints can be explained in terms of mapping cardinality.

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 represents the Cardinality.

Mapping Cardinality:

One to One: An entity of entity-set A can be associated with at most one entity of entity-set B and an entity in entity-set B can be associated with at most one entity of entity-set A. **1:1**

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.

One to Many: An entity of entity-set A can be associated with any number of entities of entity-set B and an entity in entity-set B can be associated with at most one entity of entity-set A **M:1**.

Many to One: An entity of entity-set A can be associated with at most one entity of entity-set B and an entity in entity-set B can be associated with any number of entities of entity-set A.

Many to Many: An entity of entity-set A can be associated with any number of entities of entity-set B and an entity in entity-set B can be associated with any number of entities of entity-set A.

We can have these constraints in place while creating tables in database.

Example:

```
CREATE TABLE Customer (
customer_id int PRIMARY KEY NOT NULL,
first_name varchar(20),
last_name varchar(20)
);

CREATE TABLE Order (
order_id int PRIMARY KEY NOT NULL,
customer_id int,
order_details varchar(50),
constraint fk_Customers foreign key (customer_id)
       references dbo.Customer
);
```

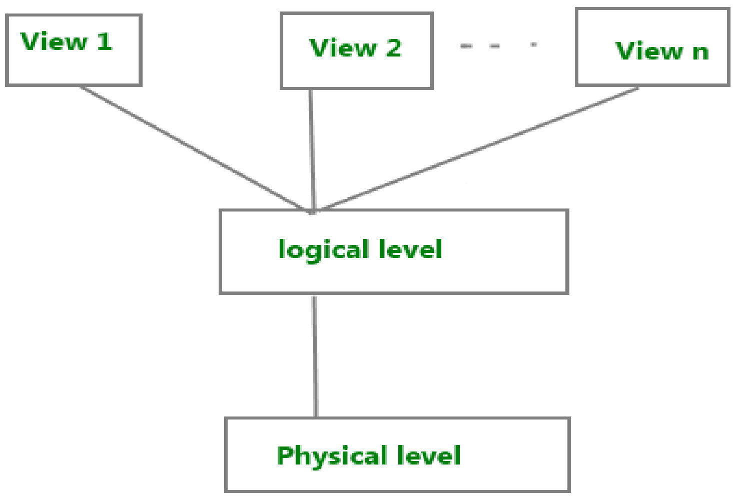
Assuming, that a customer orders more than once, the above relation represents **one to many** relations.

Data Abstraction: Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.

Physical level: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.

Logical level: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

View level: Highest level of data abstraction. This level describes the user interaction with database system.



Three Levels of data abstraction

Example: Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.

At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

<u>Data Independence:</u> Data independence is ability to modify a schema definition in one level without affecting a schema definition in the next higher level. A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.

Logical Data Independence:

Logical data independence is ability to modify the conceptual schema without requiring any change in application programs.

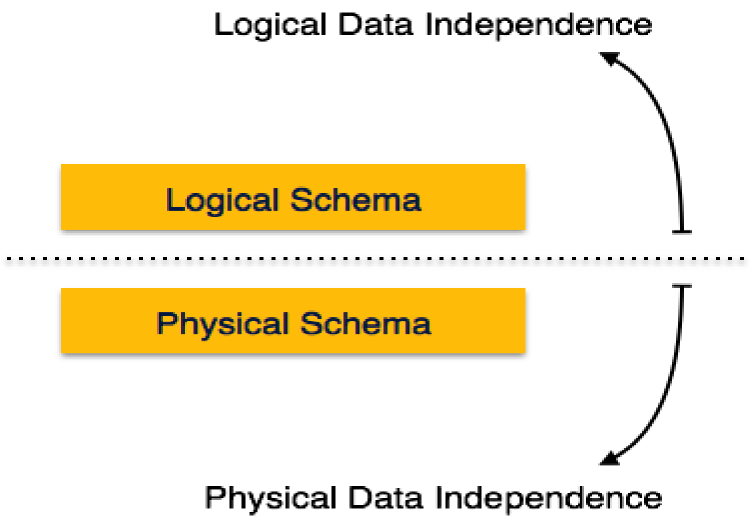
Modification at the logical levels are necessary whenever the logical structures of the database is altered.

Comparatively it is difficult to achieve logical data independence.

Application programs are heavily dependent on logical structures of the data they access.so any change in logical structure also requires programs to change.

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.



Physical Data Independence:

Physical Data Independence is the ability to modify the physical schema without requiring any change in application programs.

Modifications at the internal levels are occasionally necessary to improve performance. All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

Physical data independence separates conceptual levels from the internal levels.

This allows to provide a logical description of the database without the need to specify physical structures.

Comparatively, it is easy to achieve physical data independence.

<u>**Mapping:**</u> Process of transforming request and results between three level is called mapping.

There are the two types of mappings:

1. Conceptual/Internal Mapping
2. External/Conceptual Mapping

Conceptual/Internal Mapping:

The conceptual/internal mapping defines the correspondence between the conceptual view and the store database.

It specifies how conceptual record and fields are represented at the internal level.

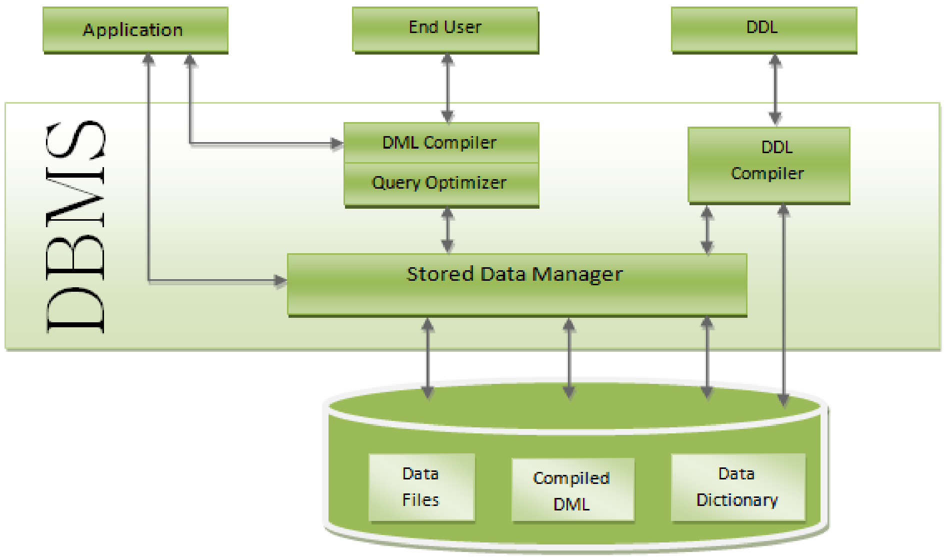
It relates conceptual schema with internal schema.

External/Conceptual Mapping:

 The external/conceptual mapping defines the correspondence between a particular external view and conceptual view.

 It relates each external schema with conceptual schema.

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.



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.

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.

Data Manager: The Data Manager is the central software component of the DBMS also knows as Database Control System.

Main Functions of Data Manager are

• 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.

• Controls DBMS information access that is stored on disk.

• It also controls handling buffers in main memory.

• It also enforces constraints to maintain consistency and integrity of the data.

• It also synchronizes the simultaneous operations performed by the concurrent users.

• It also controls the backup and recovery operations.

Data Dictionary: Data Dictionary is a repository of description of data in the database.

It contains information about

• Data - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.

• Relationships between database transactions and data items

Constraints on data i.e. range of values permitted.

• Detailed information on physical database design such as storage structure, access paths, files and record sizes.

• Access Authorization - is the Description of database users their responsibilities and their access rights.

• 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.

Data Files: It contains the data portion of the database.

Compiled DML: The DML complier converts the high level Queries into low level file access commands known as compiled DML.

End Users: End Users are the people who interact with the database through applications or utilities.

Data Models in Database: Data models define how the logical structure of a database is modelled and defines how data will be stored, accessed, and updates in database system.

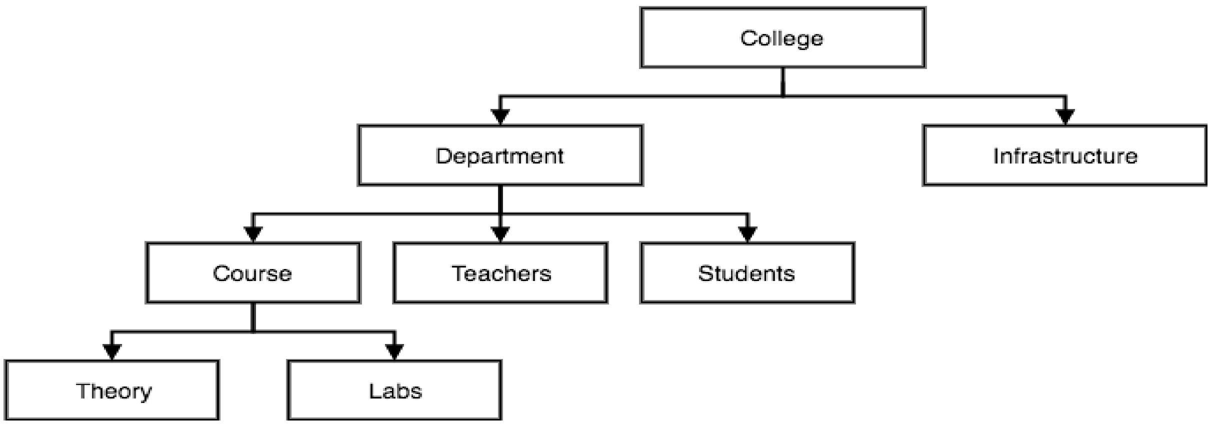
Data Model can be defined as an integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization.

The purpose of a data model is to represent data and to make the data understandable.

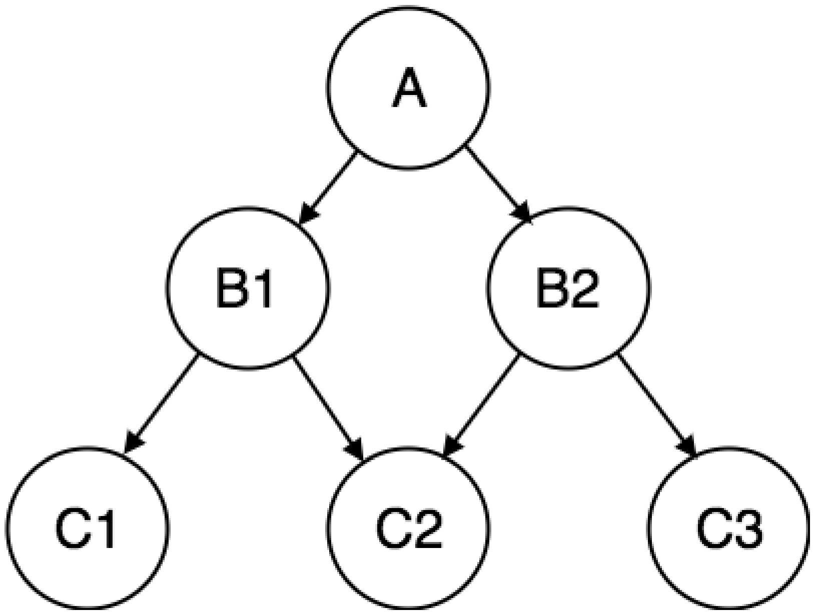
We have different types of data models they are widely used data models.

1. Hierarchal Model
2. Network Model
3. Entity-Relationship Model
4. Relational Model

1. **Hierarchal Model:** In **hierarchical model**, data is organized into a tree like structure with each record is having one parent record and many children. The main drawback of this model is that, it can have only one to many relationships between nodes.
   Example:



2. **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.



3. **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.

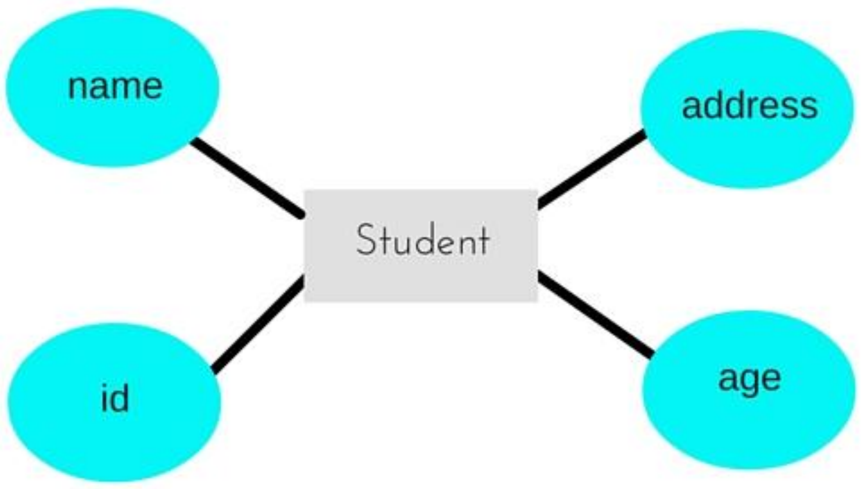
   Entity: Entity is real world object that is described in database. Entity can be represented in rectangle shape.

   Attribute: Attribute can be defined as characteristics of entity. This can be represented in ecllipse shape.

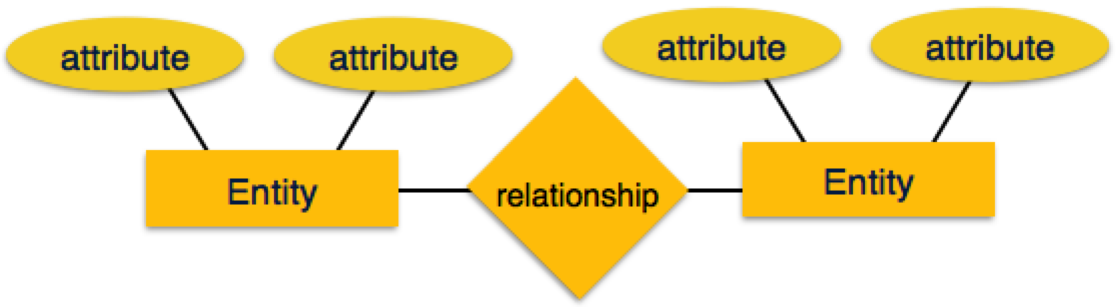
   Relationship − The logical association among entities is called ***relationship***.

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

   Example-1: 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.



Example-2: The following is the basic Entity-Relationship diagram



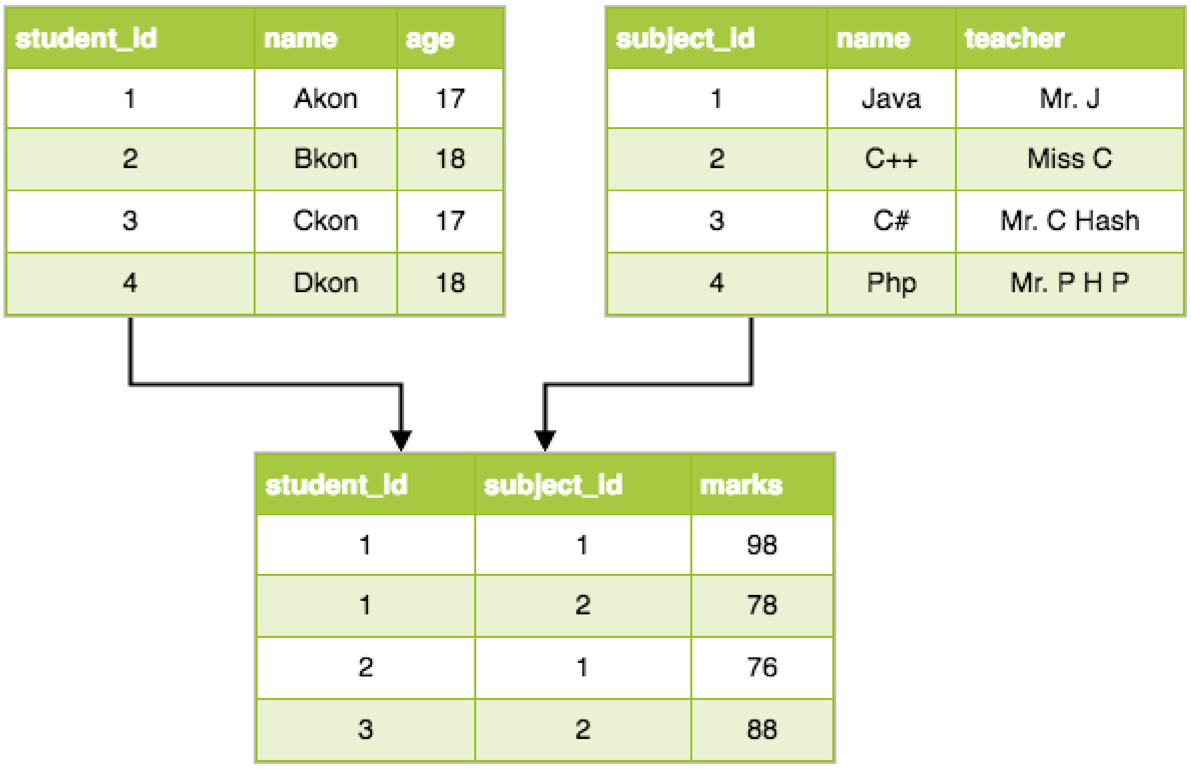
4. **Relation Model:** In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.

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

   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 this model, each row in a relation contains a unique value.

   Example:



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