CHAPTER - 5

RDBMS CONCEPTS AND MYSQL

Introduction

The key to organizational success is effective decision making which requires timely, relevant and accurate information. Hence information plays a critical role in today's competitive environment. Database Management Software (DBMS) simplifies the task of managing the data and extracting useful information out of it. In this chapter, we shall learn about the basic concepts of databases.

BASICS OF DATABASE SYSTEM

Database

Database : Database is a collection of inter-related data i.e it is composed of a collection of files that are linked in such a way that information from one of the files may be combined with information from other files so that user may receive the exact information needed. For example, school database organizes the data about students, teachers, and admin staff etc. which helps in efficient retrieval, insertion and deletion of data from it.

Basic building block of Database is Data. The input to database is known as data and output as information.

Data and Information

Most people believe that the terms "data" and "information" are interchangeable and mean the same thing. However, there is a distinct difference between the two words .

Data can be any character, text, word, number. Data comes from a Latin word, datum, which means "To give something". **Data is a raw and unorganized fact that required to be processed to make it meaningfuli.e Data is the collection of raw facts and figures**. We collect data from different resources. After collection, data is entered into computer for processing.

However, **information is data formatted in a manner that allows it to be utilized by human beings in some significant wayi.e processed data is called information**. When raw facts and figures are processed and arranged in some proper order then they become information. Information has proper meanings. Information is useful in decision-making. Actually we process data to convert it into information.

For Example: 1 ,XYZ,15 etc.

In the above example the three data items have no meaning. But if we organize these items in following way, then they collectively represent meaningful information.

Roll_No	Name	Age
1	XYZ	15

Information is created from data

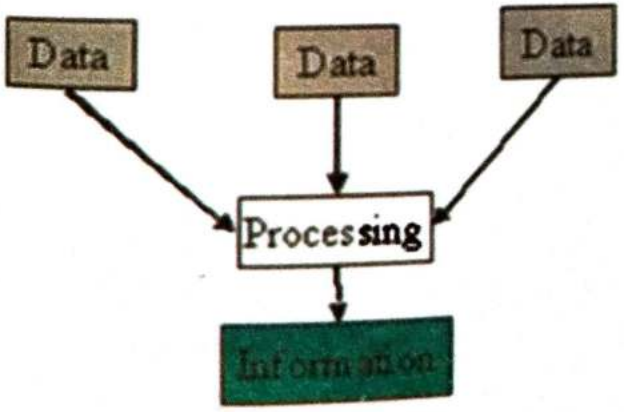
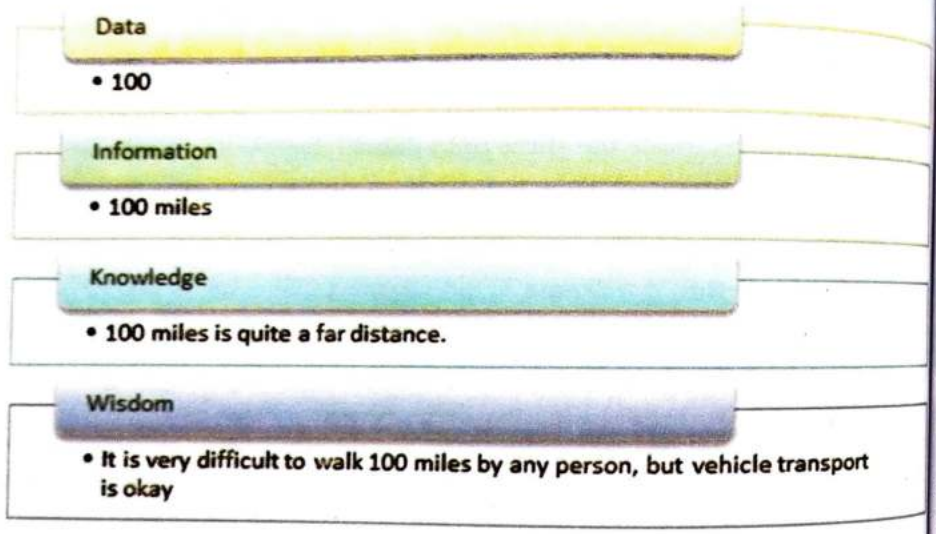


Figure. 1

The Figure 1 depicts the data processing cycle, where data is fed to processor for it to be processed into required output in the form of information.

DIKW (Data Information Knowledge Wisdom)

DIKW is the model used for discussion of data, information, knowledge, wisdom and their interrelationships. It represents structural or functional relationships between data, information, knowledge, and wisdom.



DATABASE ELEMENTS

- **Tables**

A database table is composed of records and fields that hold data. Tables are also called datasheets. Each table in a database holds data about a different, but related, subject. Below is example of database table Student :

		Student		
ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18

- **Records**

Data is stored in records. A record is composed of fields and contains all the data about one particular person, company, or item in a database. Records appear as rows in the database table. A record for ROLL_NO 3 is highlighted in table Student.

		Student		
ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	XXXXXXXXXX	19
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	19

- **Fields**

A field is part of a record and contains a single piece of data for the subject of the record. In the database table illustrated in , each record contains five fields: ROLL_NO,NAME,ADDRESS, PHONE,AGE.Fields appear as columns in a database table. Data from the NAME field for four records is highlighted in the table Student below :

		Student		
ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	XXXXXXXXXX	19
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	19

- The collection of Information stored in database at a particular instance of time is called **instance** of database.
- The overall design of the database is called database **schema.**

DBMS (DATABAE MANAGEMENT SYSTEM)

The main purpose of database management system is to manage the data. Consider a school that keeps the data of students, teachers, courses, books etc. To manage this data we need to store this data somewhere, where we can add new data, delete unused data, update outdated data, retrieve data, to perform these operations on data we need a database management system that allows us to store the data in such a way so that all these operations can be performed on the data efficiently.

DBMS stands for **Data**base **M**anagement **S**ystem. A Data Base Management System is a system software for easy, efficient and reliable data processing and management. It can be used for:

- Creation of a database.
- Retrieval of information from the database.
- Updating the database.
- Managing a database.

We can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can **define DBMSlike this : DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.** For Example, MySQL, Oracle etc. are popular commercial DBMS used in different applications.

DBMS allows users the following tasks :

- **Data Definition:** It helps in creation, modification and removal of definitions that define the organization of data in database.
- **Data Updation:** It helps in insertion, modification and deletion of the actual data in the database.

- **Data Retrieval :** It helps in retrieval of data from the database which can be used by applications for various purposes.
- **User Administration :** It helps in registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control and recovering information corrupted by unexpected failure.

DBMS Applications

Applications where we use Database Management Systems are :

- **Telecom** : There is a database to keeps track of the information regarding calls made, network usage, customer details etc. Without the database systems it is hard to maintain that huge amount of data that keeps updating every millisecond.
- **Industry** : Where it is a manufacturing unit, warehouse or distribution centre, each one needs a database to keep the records of ins and outs. For example distribution centre should keep a track of the product units that supplied into the centre as well as the products that got delivered out from the distribution centre on each day; this is where DBMS comes into picture.
- **Banking System** : For storing customer info, tracking day to day credit and debit transactions, generating bank statements etc. All this work has been done with the help of Database management systems.
- **Sales** : To store customer information, production information and invoice details.
- **Airlines** : To travel though airlines, we make early reservations, this reservation information along with flight schedule is stored in database.
- **Education sector** : Database systems are frequently used in schools and colleges to store and retrieve the data regarding student details, staff details, course details, exam details, payroll data, attendance details, fees details etc.
- **Online shopping** : You must be aware of the online shopping

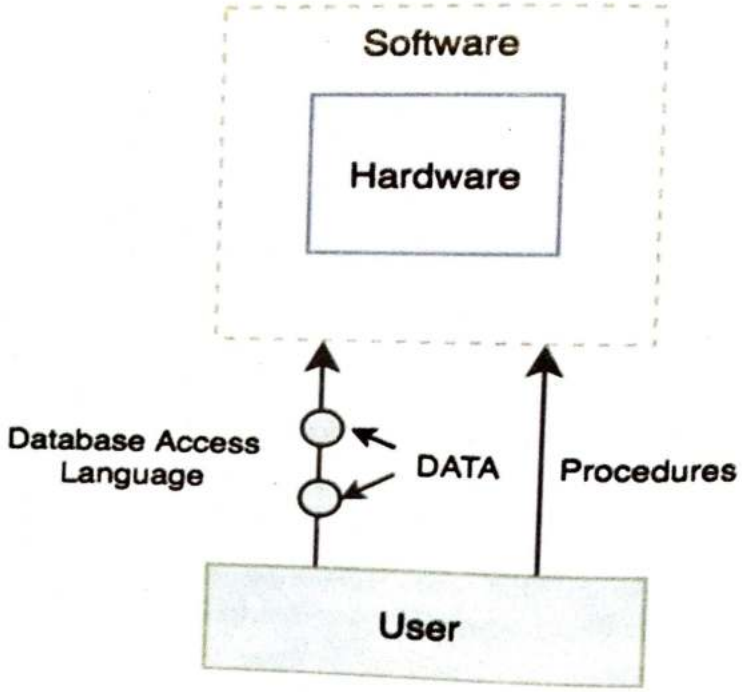
websites such as Amazon, Flipkart etc. These sites store the product information, your addresses and preferences, credit details and provide you the relevant list of products based on your query. All this involves a Database management system.

COMPONENTS OF DATABASE SYSTEM

A database system is composed of following components :

- Data
- Hardware
- Software.
- Database Access Language
- Users

which coordinate with each other to form an effective database system.



1. **Data** - Data is that resource, for which DBMS is designed. The motive behind the creation of DBMS is to store and utilize data. In a typical Database, the user saved Data is present and **meta data** is stored.

   Metadata is data about the data. This is information stored by the DBMS to better understand the data stored in it.

   For example : When you store yourName in a database, the DBMS will store when the name was stored in the database, what is the size of the name, is it stored as related data to some other data, or is it independent, all this information is metadata.

2. **Hardware-** When we say Hardware, we mean computer, hard disks, I/O channels for data, and any other physical component involved before any data is successfully stored into the memory.When we run MySQL on our personal computer, then our computer's Hard Disk, our Keyboard using which we type in all the commands, our computer's RAM, ROM all become a part of the DBMS hardware.

3. **Software**- This is the main component, as this is the program which controls everything. The DBMS software is more like a wrapper around the physical database, which provides us with an easy-to-use interface to store, access and update data. The DBMS software is capable of understanding the Database Access Language and interpret it into actual database commands to execute them on the database.

4. **Data base access language**- Database Access Language is a simple language designed to write commands to access, insert, update and delete data stored in any database. A user can write commands in the Database Access Language and submit it to the DBMS for execution, which is then translated and executed by the DBMS. User can create new databases, tables, insert data, fetch stored data, update data and delete the data using the access language.

5. **Users-** Users are those persons who need the information from the database to carry out their primary business responsibilities i.e. Personnel, Staff, Clerical, Managers, Executives etc. On the basis of the job and requirements made by them they are provided access to the database totally or partially.

The various types of users which can access the database are:-

- Database Administrators (DBA)
- Database Designers
- End Users
- Application Programmers

RDBMS (Relational Database Management System)

RDBMS

RDBMS stands for "Relational Database Management System. An RDBMS is a DBMS designed specifically for relational databases. Relational Database Management System (RDBMS) is an advanced version of a DBMS system. It came into existence during 1970's. RDBMS system also allows the organization to access data more efficiently then DBMS.

Relational Database - A relational database refers to a database that stores data in a structured format, using rows and columns. This makes it easy to locate and access specific values within the database. It is "relational" because the values within each table are related to each other. Tables may also be related to other tables. The relational structure makes it possible to run queries across multiple tables at once.

A relational database management system (RDBMS) is a program that allows you to create, update, and administer a relational database. Most relational database management systems use the SQL language to access the database. An RDBMS is a type of DBMS with a row-based table structure that connects related data elements and includes functions that maintain the security, accuracy, integrity and consistency of the data.

RDBMS is a software system which is used to store only data which need to be stored in the form of tables. In this kind of system, data is managed and stored in rows and columns which is known as tuples and attributes. RDBMS is a powerful data management system and is widely used across the world. Example of RDBMS is MySQL, Oracle, SQL Server, etc.

Traditional File System

Traditional File Processing Systems : It is totally computer based system where all the**information is stored in different computer files.** Also traditional files system stores data in a manner that all the departments of an organization have their own set of files that **creates data redundancy**.

But this system is good only for small organizations having small number of items. In this traditional file system, each file is independent of other file and data in the different file can be integrated only by writing an individual program for each application.

A file system is a method for sorting and organizing computer files and the data they contain to make it easy to find and access them.

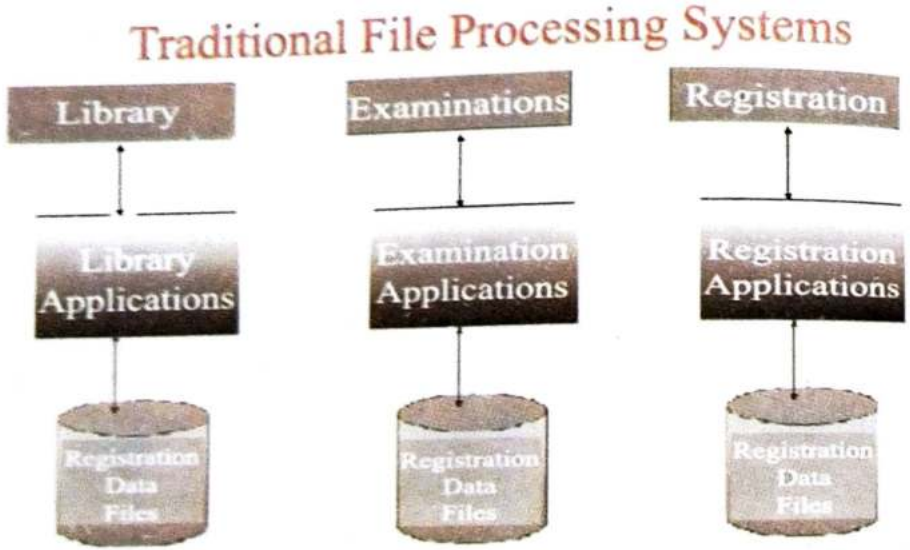
To illustrate Traditional File Processing Systems definition, lets us take an example of school where student record for examination is stored in other file and his library record is stored in different file that creates many duplicate values like roll Number, Name and Father Name.

Characteristics OF Traditional File Processing System

- It stores data of an organization in group of files.
- Files carryingdata are independent on each other.
- COBOL, C, C++ programming languages were used to design th files.
- Each file contains data for some specific area or department lik library, student fees, and student examinations.
- It is less flexible and has many limitations.

- It is very difficult to maintain file processing system.
- Any change in one file affects all the files that creates burden on the programmer.
- File in Traditional File Processing Systems are called flat files.

Overall. **Traditional File Processing Systems** is good in many cases in compare to manual non computer based system but still it has many limitation which are discussedbelow .



Traditional File System

Problems in Traditional File oriented approach

- **Data redundancy** : Data redundancy refers to the duplication of data, lets say we are managing the data of a school where a student is enrolled for two courses, the same student details in such case will be stored twice, which will take more storage than needed. Data redundancy often leads to higher storage costs and poor access time.
- **Data inconsistency :** Data redundancy leads to data inconsistency, let's take the same example that we have taken above, a student is

enrolled for two courses and we have student address stored twice, now let's say student requests to change his address, if the address is changed at one place and not on all the records then this can lead to data inconsistency.

- **Data Isolation :** Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
- **Dependency on application programs :** Changing files would lead to change in application programs.
- **Atomicity issues :** In database system atomicity is one the ACID (Atomicity, Consistency, Isolation, Durability) transaction properties. Atomicity of a transaction refers to "All or nothing", which means either all the operations in a transaction executes or none. For example: Let's say A transfers 100 rupeesto B's account. This transaction consists multiple operations such as debit 100 rupees from A's account, credit 100 rupees to B's account. Like any other device, a computer system can fail let's say it fails after first operation then in that case A's account would have been debited by 100 rupees but the amount was not credited to B's account, in such case the rollback of operation should occur to maintain the atomicity of transaction. It is **difficult to achieve atomicity in file processing systems.**
- **Difficult Data Access :** A user should know the exact location of file to access data, so the process is very cumbersome and tedious. If user wants to search student hostel allotment number of a student from 10000 unsorted students' records, how difficult it can be.
- **Unauthorized Access :** File System may lead to unauthorized access to data. If a student gets access to file having his marks, he can change it in unauthorized way.
- **No Concurrent Access :** The access of same data by multiple users at same time is known as concurrency. File system does not allow concurrency as data can be accessed by only one user at a time.

- **No Backup and Recovery :** File system does not incorporate any backup and recovery of data if a file is lost or corrupted.
- **Data Security :** Data should be secured from unauthorized access, for example a student in a school should not be able to see the payroll details of the teachers, such kind of security constraints are difficult to apply in file processing systems.
- **Limited Data Sharing** : There islimited data sharing possibilities with the traditional file system. Each application has its own private files and users have little choice to share the data outside their own applications. Complex programs required to be written to obtain data from several incompatible files.

Difference between DBMS and Traditional File System

- DBMS is very expensive but, the traditional file system is cheap.
- DBMS is good for the large system but, the traditional file system is good for a small system having a small number of items.
- DBMS required lots of effort for designing but, the traditional file system is very low design efforts.
- DBMS is highly secured but, the traditional file system is not secure.
- DBMS is data sharable but, the traditional file system is isolated data sharable.
- DBMS is flexible but, the traditional file system has a lack of flexibility and has many limitations.
- DBMS has no integrity issue but, the traditional file system has an integrity problem.
- DBMS has a complex backup system but, the traditional file system has a simple backup system.

Advantages and Disadvantages of DBMS

ADVANTAGES :

There are several advantages of Relational Database management system. Few of them are as follows :

- **Minimal Redundancy :** Unlike traditional file-system storage, Data redundancy in DBMS is very less or not present. Data redundancy occurs when the same data are stored unnecessarily at different places. Data redundancy is reduced or eliminated in DBMS because all data are stored at a centralized location rather than being created by individual users and for each application. No data duplication saves storage and improves access time.
- **Data Inconsistency can be avoided :** In traditional file system storage, the changes made by one user in one application doesn't update the changes in other application, given both have the same set of details. While this is not the case with DBMS systems as there is a single repository of data that is defined once and is accessed by many users, and data are consistent.
- **Searching capability :** Searching and retrieving of data is very easy in DBMS systems. The need to write separate programs for each of the search is eliminated as in the case with a traditional file-based approach. In DBMS, we can write small queries to search for multiple information at a time from the data from database servers.
- **Data Sharing :** Data Sharing is the primary advantage of Database management systems. DBMS system allows users and applications to share Data with multiple applications and users. Data are stored in one or more servers in the network and that there is some software locking mechanism that prevents the same set of data from being changed by two people at the same time. While the file system doesn't have this capability.
- **Data Security :** DBMS systems provide a strong framework to protect data privacy and security. DBMS ensures that only authorized users have access to data and there is a mechanism to define access privileges in DBMS.
- **Flexibility** : Database systems are more flexible than file processing systems.

- **Data Integration** : Data integration is a process of combining the data residing at different locations and present the user with a unified view of data. DBMS systems allow Data Integration with much feasibility.
- **Easy access to data :** A database system manages data in such a way so that the data is easily accessible with fast response times.
- **Concurrent access to data :** Data can be accessed concurrently by different users at same time in DBMS.
- **Data Backup and Recovery :** This is another advantage of DBMS as it provides a strong framework for Data backup, users are not required to back up their data periodically and manually, it is automatically taken care by DBMS. Moreover, in case of a server crash, DBMS restores the Database to its previous condition.
- **Improved decision making :** Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are based.

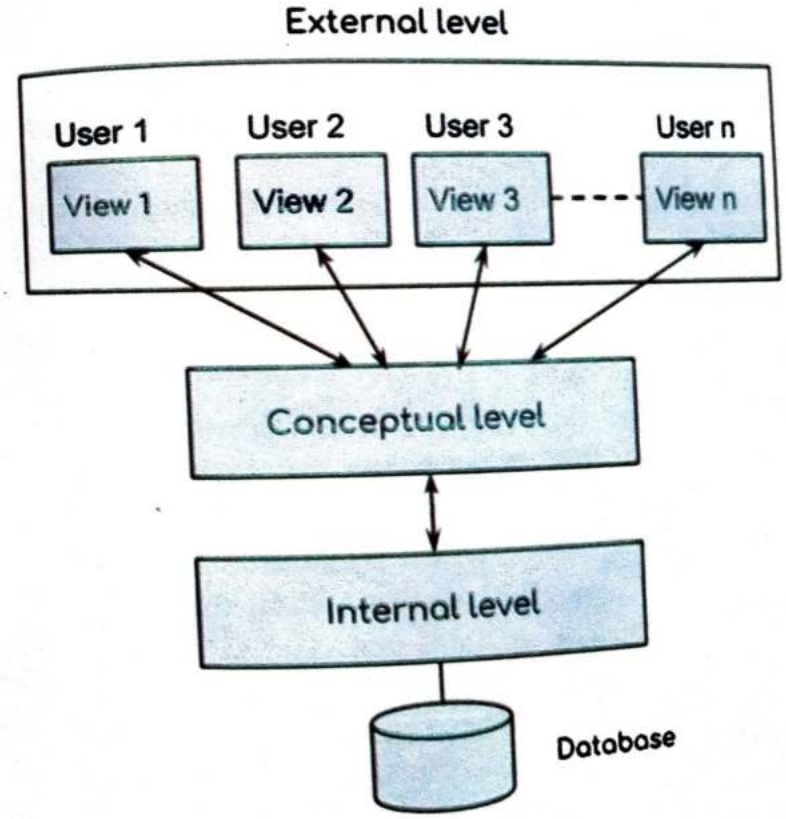
DISADVANTAGES OF DBMS :

- DBMS requires high initial investment for hardware, software and trained staff.
- In order to convert our data into a Database Management System we need to spend a lot which adds on to the cost of the Database Management System.
- In order to work with a Database Management System a team of technical staff is required who understand DBMS.
- A DBMS requires disk storage for the data and sometimes you need to purchase extra space to store your data.
- Database systems are complex to understand
- DBMS does not give a good performance as its speed is very slow.
- Backup and recovery are more difficult. This is because of increased complexity and because databases are often processed by several users concurrently.

Three level architecture of DBMS

DBMS is collection of inter- related data and set of procedures to access that data. Database systems are made-up of complex data structures. The main purpose of DBMS is to provide users with an abstract view of the data .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.** There are mainly three levels of data abstraction :

1. **Internal or Physical level :** Actual physical storage structure and access paths.
2. **Conceptual or Logical Level :** Structure and constraints for the entire database
3. **External or View level :** Describes various user views



Internal Level

It is physical representation of the database on the computer. This level is also known as physical level. This level describes how the data is actually stored in the storage devices.

Physical level is also responsible for allocating space to the data. This is the lowest level in the three level architecture. The physical level also discusses compression and encryption techniques.

Conceptual Level

It is also called **logical level**. The whole design of the database such as relationship among data, schema of data etc. are described in this level. Database constraints and security are also implemented in this level of architecture. This level is maintained by DBA (database administrator). The conceptual level does not care for how the data in the database is actually stored.

External Level

It is also called **view level**. The reason this level is called "view" is because several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level mapping.

The user doesn't need to know the database schema details such as data structure, table definition etc. user is only concerned about data which is what returned back to the view level after it has been fetched from database (present at the internal level). External level is the "**top level**" of the three level DBMS architecture. This is the highest level in the three level architecture and closest to the user.

The external level only shows the relevant database content to the users in the form of views and hides the rest of the data. So different users can see the database as a different view as per their individual requirements. An external schema describes the part of the database which specific user is interested in. It hides the unrelated details of the database from the user. There may be "n" number of external views for each database.

An external view is just the content of the database as it is seen by some specific particular user. For example, a user from the sales department will see only sales related data.

For example : Let's say we are storing student information in a student 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.

ROLES OF DBA (DATABASE ADMINISTRATOR)

Typically there are three types of users for a DBMS. They are :

1. **The Application Programmer :** These users write application programs to interact with the database. Application programs can be written in some programming language such a COBOL, PL/I, C++, JAVA or some higher level fourth generation language. Such programs access the database by issuing the appropriate request, typically a SQL statement to DBMS.
2. **The End User :** End users are the users, who use the applications developed. End users need not know about the working, database design, the access mechanism etc. They just use the system to get their task done. End users are of two types:

   a) Direct users b) Indirect users

   a) **Direct users :** Direct users are the users who see the computer, database system directly, by following instructions provided in the user interface. They interact using the application programs already developed, for getting the desired result. E.g. People at railway reservation counters, who directly interact with database.

b) Indirect users : Indirect users are those users, who desire benefit from the work of DBMS indirectly. They use the outputs generated by the programs, for decision making or any other purpose. They are just concerned with the output and are not bothered about the programming part.

3. **System Analyst :** System Analyst determines the requirement of end users, especially naïve and parametric end users and develops specifications for transactions that meet these requirements. System Analyst plays a major role in database design, its properties; the structure prepares the system requirement statement, which involves the feasibility aspect, economic aspect, technical aspect etc. of the system.

The Database Administrator (DBA)

Database Administrator (DBA) is the person which makes the strategic and policy decisions regarding the data of the enterprise, and who provide the necessary technical support for implementing these decisions. Therefore, DBA is responsible for overall control of the system at a technical level. In database environment, the primary resource is the database itself and the secondary resource is the DBMS and related software administering these resources is the responsibility of the Database Administrator (DBA).

DBA responsibilities include designing, implementing, and maintaining the database system; establishing policies and procedures pertaining to the management, security, maintenance, and use of the database management system; and training employees in database management and use.

The role of the DBA is very important and is defined by the following functions :

1. **Schema Definition :**
   - The DBA define the logical Schema of the database. A Schema refers to the overall logical structure of the database.
   - According to this schema, database will be developed to store required data for an organization.

2. **Storage Structure and Access Method Definition :**
   - The DBA decides how the data is to be represented in the stored database.
3. **Assisting Application Programmers :**
   - The DBA provides assistance to application programmers to develop application programs.
4. **Physical Organization Modification :**
   - The DBA modifies the physical organization of the database to reflect the changing needs of the organization or to improve performance.
5. **Approving Data Access :**
   - The DBA determines which user needs access to which part of the database.
   - According to this, various types of authorizations are granted to different users.
6. **Liaising with Users :** The DBA needs to interact continuously with the users to understand the data in the system and its use. The DBA figures out which client needs access to which part of the database
7. **Monitoring Performance :** The DBA monitors performance of the system. The DBA ensures that better performance is maintained by making changes in physical or logical schema if required.
8. **Backup and Recovery :** Database should not be lost or damaged. The DBA ensures this periodically backing up the database on magnetic tapes or remote servers. In case of failure, such as virus attack database is recovered from this backup.

DATA DICTIONARY

A data dictionary contains metadata i.e data about the database. It is a dictionary about the data that we store in the database. It contains all the information about the data objects. Data Dictionary is like storing

all up-to-date information about the objects like tables, columns, index, constraints, functions etc.

The data dictionary is very important as it contains information such as what is in the database, who is allowed to access it, where is the database physically stored etc. The users of the database normally don't interact with the data dictionary, it is only handled by the database administrators.

The data dictionary in general contains information about the following :

- Names of all the database tables and their schemas.
- Details about all the tables in the database, such as their owners, their security constraints, when they were created etc.
- Physical information about the tables such as where they are stored and how.
- Table constraints such as primary key attributes, foreign key information etc.
- Information about the views of the database.
- It also contains the physical information of the table like about their storage, about their alteration, etc.

Why do we need all these information ?

It makes us easily identify access and understand the factors about the object. One can imagine data dictionary as storing information about house like house name, address, how many live in the house, who is the eldest/youngest person, responsibilities of each member in the house etc.The data dictionary contains the bookkeeping information about the database so that it can manage the data. It does not contain the information of the actual data of the database. Without the presence of a data dictionary, a database management system cannot access the data from the database. Database administrators handle the data dictionary, and users don't interact with it.

In the case of a table, data dictionary provides information about :

- Its name
- Security information like who is the owner of the table, when was it created, and when it was last accessed.
- Physical information like where is the data stored for this table
- Structural information like its attribute names and its data types, constraints and indexes.

The table below is an example of a typical data dictionary entry :

Primary Key	Field Name	Field Type	Field Length
Yes	Roll_No	Number	6
No	Name	Text	30
No	Address	Text	30

There are two types of data dictionary – Active and Passive.

Active Data Dictionary

Any changes to the database object structure via DDLs will have to be reflected in the data dictionary. But updating the data dictionary tables for the changes are responsibility of database in which the data dictionary exists. If the data dictionary is created in the same database, then the DBMS software will automatically update the data dictionary. Hence there will not be any mismatch between the actual structure and the data dictionary details. Such data dictionary is called active data dictionary

Passive Data Dictionary

In some of the databases, data dictionary is created separately from the current database as entirely new database to store only data dictionary information. Sometimes it is stored as xml, excels or in any other file format. In such case, an effort is required to keep data dictionary in sync with the database objects. This kind of data dictionary

is called passive data dictionary. In this case, there is a chance of mismatch with the database objects and the data dictionary. This kind of Data Dictionary has to be handled with utmost care.

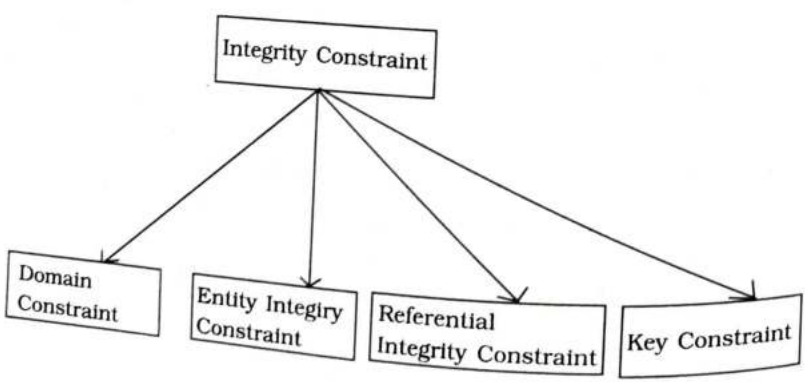
Integrity Constraints

Database integrity refers to the validity and consistency of stored data. Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate. Constraints may apply to each attribute or they may apply to relationships between tables.

Integrity constraints are used to ensure accuracy and consistency of data in a relational database. Data integrity is handled in a relational database through the concept of referential integrity.

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.
- For example, A blood group must be 'A' or 'B' or 'AB' or 'O' only (can not any other values else).

Types of Integrity Constraint



1. **Domain constraints :** Domain integrity means the definition of a valid set of values for an attribute. It ensures that only a valid range of value is allowed to be stored in a column.
   - Domain constraints can be defined as the definition of a valid set of values for an attribute.
   - The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example :

Marks table

ROLL_NO	NAME	MARKS	AGE
1	RAM	90	18
2	RAMESH	50	18
3	SUJIT	80	20
4	SURESH	95	18
5	HARSH	85	A

Here A is not allowed in AGE attribute, because it is an integer attribute.

2. **Entity integrity constraints**
   - The entity integrity constraint states that primary key value can't be null.
   - This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
   - A table can contain a null value other than the primary key field.

Example :

Marks table			
ROLL_NO	**NAME**	**MARKS**	**AGE**
1	RAM	90	18
2	RAMESH	50	18
	SUJIT	80	20
4	SURESH	95	18
5	HARSH	85	19

Here ROLL_NO attribute is not allowed as Primary key because primary key can't contain NULL value.

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be available in Table 2.

Example :

Foreign Key of table student

Student			
ROLL_NO	**NAME**	**AGE**	**SUBJECT_ID**
1	RAM	18	S001
2	RAMESH	18	S002
3	SUJIT	20	S003
4	SURESH	18	S004

Subject	
SUBJECT_ID	**SUBJECT_NAME**
S001	COMPUTER SCIENCE
S002	ENGLISH
S004	HINDI

Here SUBJECT_ID is not allowed as Foreign Key of table Student because SUBJECT_ID S003 is not defined as Primary key of table Subject.

4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example :

Student			
ROLL_NO	**NAME**	**AGE**	**SUBJECT_ID**
1	RAM	18	S001
2	RAMESH	18	S002
3	SUJIT	20	S003
2	SURESH	18	S004

Here attribute ROLL_NO is not allowed as Primary Key because all rows must be unique.

SQL Datatypes

Datatypes are rules that define what data may be stored in a column and how that data is actually stored. Data types are used for several reasons :

1. Data types enables us to restrict the type of data that can be stored in a column. For example, a numeric data type column only accepts numeric values.

2. Data type specifies storage efficiently and instructs show it stores data internally in memory.

3. Data types allow for more efficient storage. Properly defining the fields in a table is important to the overall optimization of your database. You should use only the type and size of field you really need to use. For example, do not define a field 10 characters wide, if you know you are only going to use 2 characters.

4. Data type allow for alternate sorting orders.

In SQL there are three main data types :

- Numeric
- Date and time
- String type

1. Numeric Data Type :

Numeric data types are normally used to store numeric data like age, marks, salary, price etc. Numeric datatypes are following :

I. int – used for number without decimal.

II. Decimal(m,d) – used for floating/real numbers.

Here m denotes the total length of number and d is number of decimal digits.

2. Date and Time Data Type :

I. date –used to store date in YYYY-MM-DD format.

II. time –used to store time in HH:MM:SS format.

3. String Data type :

String data types are normally used to store names, addresses, descriptions or any value that contains letters and numbers including binary data, like image or audio files. String datatypes are following :

I. char (n) – used to store a fixed length string.
Here, n denotes maximum number of characters.

II. varchar(n) –used to store a variable length string.
Here, n denotes maximum number of characters.

Difference between char and varchar datatypes :

Sr. No	Char Datatype	Varchar Datatype
1.	It specifies a fixed length character String.	It specifies a variable length character string.
2.	When a column is given datatype as CHAR(n), then MySQL ensures that all values stored in that column have this length i.e. n bytes. If a value is shorter than this length n then blanks are added, but the size of value remains n bytes.	When a column is given data type as VARCHAR(n),then the maximum size a value in this column can have is n bytes. Each value that is stored in this column store exactly as you specify it i.e. no blanks are added if the length is shorter than maximum length n.

SQL Statements

Introduction to SQL

SQL stands for Structured Query Language, which is a standardized language for interacting with RDBMS (Relational Database Management System).Structure Query Language(SQL) is a database query language used for storing and managing data in Relational DBMS. SQL was the first commercial language introduced for E.F Codd's **Relational** model of database. Today almost all RDBMS (MySql, Oracle, Infomix, Sybase, MS Access) use **SQL** as the standard database query language. SQL is used to perform all types of data operations in RDBMS.

Characteristics of SQL

- SQL stands for Structured Query Language. It is used for storing and managing data in relational database management system (RDBMS).
- All the RDBMS like MySQL, Informix, Oracle, MS Access and SQL Server use SQL as their standard database language.

- SQL allows users to query the database in a number of ways, using English-like statements.
- SQL is used to perform C.R.U.D (Create, Retrieve, Update& Delete) operations on relational databases.
- SQL can also perform administrative tasks on database such as database security, backup, user management etc.
- We can create databases and tables inside database using SQL.
- Structure query language is not case sensitive. Generally, keywords of SQL are written in uppercase.
- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text line.
- Using the SQL statements, you can perform most of the actions in a database.
- SQL depends on tuple relational calculus and relational algebra.

SQL Commands :

SQL commands are instructions, coded into SQL statements, which are used to communicate with the database to perform specific tasks, work, functions and queries with data.

SQL commands can be used not only for searching the database but also to perform various other functions like, for example, you can *create tables*, add data to tables, or modify data, drop the table, set permissions for users. SQL commands are grouped into four major categories depending on their functionality:

- **Data Definition Language (DDL)**- DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema .These SQL commands are used for creating, modifying, and dropping the structure of database objects.

Examples of DDL commands :

- **CREATE** – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
- **DROP** – is used to delete objects from the database.

- **ALTER**-is used to alter the structure of the database.
- **TRUNCATE**–is used to remove all records from a table, including all spaces allocated for the records are removed.
- **COMMENT** –is used to add comments to the data dictionary.
- **RENAME** –is used to rename an object existing in the database

- **Data Manipulation Language (DML)** - The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language.These SQL commands are used for storing, retrieving, modifying, and deleting data.

Examples of DML :

- **SELECT** – is used to retrieve data from the database.
- **INSERT** – is used to insert data into a table.
- **UPDATE** – is used to update existing data within a table.
- **DELETE** – is used to delete records from a database table.

- **Transaction Control Language (TCL)** - These SQL commands are used for managing changes affecting the data. These commands are COMMIT, ROLLBACK, and SAVEPOINT. TCL commands deals with the transaction within the database.

Examples of TCL commands :

- **COMMIT**– commits a transaction.
- **ROLLBACK**– rollbacks a transaction in case of any error occurs.
- **SAVEPOINT**–sets a savepoint within a transaction.
- **SET TRANSACTION**–specify characteristics for the transaction

- **Data Control Language (DCL)** - These SQL commands are used for providing security to database objects. These commands are GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

Examples of DCL commands :

- **GRANT**-gives user's access privileges to database.
- **REVOKE**-withdraw user's access privileges given by using the GRANT command.

DDL STATEMENTS

CREATE,ALTER,DROP,RENAME, TRUNCATE are commands of DDL language which helps in defining objects.In SQL database is container of tables .The actual data is stored in tables and each table that is created is stored in a database.

- **CREATE**

There are two CREATE statements available in SQL :

1. CREATE DATABASE
2. CREATE TABLE

CREATE DATABASE

A **Database** is defined as a structured set of data. So, in SQL the very first step to store the data in a well structured manner is to create a database. The **CREATE DATABASE** statement is used to create a new database in SQL.

Syntax :

CREATE DATABASE database_name;

database_name: name of the database.

Example:

This query will create a new database in SQL and name the database as *my_database*.

CREATE DATABASE my_database;

- You can **USE statement** to use the database and make it the current database.

For example:

USE my_database;

CREATE TABLE

We have learned above about creating databases. Now to store the data we need a table to do that. The CREATE TABLE statement is used to create a table in SQL. We know that a table comprises of rows and columns. So while creating tables we have to provide all the information

about the names of the columns, type of data to be stored in columns, size of the data etc. Let us now dive into details on how to use CREATE TABLE statement to create tables in SQL.

Syntax :

```
CREATE TABLE table_name
(
column1 data_type(size),
column2 data_type(size),
column3 data_type(size),
....
);
```

table_name: name of the table.

column1 name of the first column.

data_type: Type of data we want to store in the particular column. For example,**int** for integer data.

size: Size of the data we can store in a particular column. For example if fora column we specify the data_type as int and size as 10 then this column can store an integer number of maximum 10 digits.

Example :

This query will create a table named Students :

```
CREATE TABLE Student
(
ROLL_NO int(3),
NAME varchar(30),
ADDRESS varchar(30),
PHONE_NOint(12),
AGE int(2),
);
```

This query will create a table named Student. The ROLL_NO,PHONE_NO and AGE fields are of type int and can store

integer number of size 3 .12,2 respectively. Three columns NAME and ADDRESS are of type varchar and can store characters and the size 30 specifies that these fields can hold maximum of 30 characters.

- **DROP**

DROP is used to delete a whole database or just a table.The SQL DROP command is used to remove an object from the database. If you drop a table, all the rows in the table is deleted and the table structure is removed from the database. Once a table is dropped we cannot get it back, so be careful while using DROP command. When a table is dropped all the references to the table will not be valid.

Syntax :

DROP TABLE table_name;

table_name : Name of table to be deleted.

DROP DATABASE database_name;

database_name: Name of the database to be deleted.

Example :

To drop the table Student table, the query would be like

DROP TABLE Student ;

- **TRUNCATE**

TRUNCATE statement is a Data Definition Language (DDL) operation that is used to mark the extents of a table for deallocation (empty for reuse). The SQL TRUNCATE command is used to delete all the rows from the table and free the space containing the table.

Syntax :

TRUNCATE TABLE table_name;

table_name: Name of the table to be truncated.

Example :

To delete all the rows from Student table, the query would be like,

TRUNCATE TABLE Student;

- **INSERT**

Once a table is created most natural thing to do is load this table with data to be manipulated later. This data can be inserted by using INSERT command.

The **INSERT INTO** statement of SQL is used to insert a new row in a table. There are two ways of using INSERT INTO statement for inserting rows:

1. **Only values :** First method is to specify only the value of data to be inserted without the column names.

Syntax :

INSERT INTO table_name

VALUES (value1, value2, value3,...valueN);

table_name: name of the table.

value1, value2,..: value of first column, second column,... for the new record

While inserting a row, if you are adding value for all the columns of the table you need not specify the column(s) name in the sql query. But you need to make sure the order of the values is in the same order as the columns in the table. When adding a new row, you should ensure the datatype of the value and the column matches.

2. **Column names and values both :** In the second method we will specify both the columns which we want to fill and their corresponding values as shown below:

INSERT INTO table_name

[(col1, col2, col3,...colN)]

VALUES (value1, value2, value3,...valueN);

table_name: name of the table.

col1: name of first column, second column .....

value1, value2, value3 : value of first column, second column,... for the new record.

NOTE : When adding a row, only the characters or date values should be enclosed with single quotes. New row can be inserted using INSERT command only one row is inserted at a time.

Example :

INSERT INTO Student

VALUES (1, 'RAM', 'DHARAMSHALA',XXXXXXXXXX, 18);

If you want to insert only roll number ,name and age to the student table, the query would be like,

INSERT INTO Student (ROLL_NO, NAME, AGE)VALUES (1, 'RAM', 18) :

- **SELECT**

Select is the most commonly used statement in SQL. **The SELECT Statement in SQL is used to retrieve or fetch data from a database.** We can fetch either the entire table or according to some specified rules. The data returned is stored in a result table. This result table is also called result-set.

With the SELECT clause of a SELECT command statement, we specify the columns that we want to be displayed in the query result and, optionally, which column headings we prefer to see above the result table.

Syntax :

SELECT column1,column2 FROM table_name;

column1, column2: names of the fields of the table

table_name: from where we want to fetch

This query will return all the rows in the table with fields column1 , column 2.

To fetch the entire table or all the fields in the table:

SELECT * FROM table_name;

Sample table :

Student

ROLL_ NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	PRATIK	MANDI	XXXXXXXXXX	20

Example Queries

- Query to fetch the fields ROLL_NO, NAME, AGE from the table Student:

SELECT ROLL_NO, NAME, AGE FROM Student ;

Output :

ROLL_NO	NAME	AGE
1	RAM	18
2	RAMESH	18
3	SUJIT	20
4	SURESH	18
5	HARSH	19
6	PRATIK	20

- To fetch all the fields from the table Student:

SELECT * FROM Student;

Output :

ROLL_ NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	PRATIK	MANDI	XXXXXXXXXX	20

WHERE Clause

WHERE clause is used with SELECT statement to specify the condition based on which rows will be extracted from the table. WHERE keyword is used for fetching filtered data in a result set.

- It is used to fetch data according to a particular criteria.
- WHERE keyword can also be used to filter data by matching patterns.

Syntax :

SELECT column1,column2

FROM table_name

WHERE column_name operator value;

column1 , column2: fields in the table

table_name: name of table

column_name: name of field used for filtering the data

operator: operation to be considered for filtering

value: exact value or pattern to get related data in result

List of operators that can be used with where clause :

Operator	Description
>	Greater Than
>=	Greater than or Equal to
<	Less Than
<=	Less than or Equal to
=	Equal to
<>	Not Equal to
BETWEEN	In an inclusive Range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

Example Queries

- To fetch record of students from table Student with age equal to 20

SELECT * FROM Student WHERE AGE=20;

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
6	PRATIK	MANDI	XXXXXXXXXX	20

- To fetch Name and Address of students from table Student with ROLL_NO greater than 3

SELECT ROLL_NO,NAME,ADDRESS FROM Student WHERE ROLL_NO > 3;

Output :

ROLL_NO	NAME	ADDRESS
4	SURESH	DELHI
5	HARSH	SOLAN
6	PRATIK	MANDI

BETWEEN operator

The BETWEEN operator is used to specify a range of values. The range you specify contains a lower and upper range .It is used to fetch filtered data in a given range inclusive of two values.

Syntax :

SELECT column1,column2 FROM table_name WHERE column_name BETWEEN value1 AND value2;

BETWEEN : operator name

value1 AND value2 : exact value from value1 to value2 to get related data in result set.

Example Queries

- To fetch records of students from Student table where ROLL_NO is between 1 and 3 (inclusive)

SELECT * FROM Student WHERE ROLL_NO BETWEEN 1 AND 3;

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20

- To fetch NAME,ADDRESS of students where Age is between 20 and 30 (inclusive)

SELECT NAME,ADDRESS FROM Student WHERE Age BETWEEN 20 AND 30 ;

Output :

NAME	ADDRESS
SUJIT	DHARAMSHALA
PRATIK	MANDI

LIKE operator

In certain cases you may not always know the exact value to search for. You can select rows that match a character pattern by using LIKE operator which is used to fetch filtered data by searching for a particular pattern in WHERE clause. LIKE operator is used with CHAR datatype.

Syntax :

SELECT column1, column2 FROM table_name WHERE column_name LIKE pattern;

LIKE : operator name

pattern : exact value extracted from the pattern to get related data in result set.

Note : The character(s) in pattern are case sensitive.

Example Queries

- To fetch records of students from Student table where NAME starts with letter S.

SELECT * FROM Student WHERE NAME LIKE 'S%';

The '%'(wildcard) signifies the later characters here which can be of any length and value.

Output:

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18

- To fetch records of students from Student table where NAME contains the patter 'AM'.

SELECT * FROM Student WHERE NAME LIKE '%AM%';

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18

IN operator

It is used to fetch filtered data same as fetched by '=' operator just the difference is that here you can specify multiple values for which we can get the result set.

Syntax :

SELECT column1,column2FROM table_name WHERE column_nameIN (value1,value2,..);

IN : operator name

value1,value2,..: exact value matching the values given and get related data in result set.

Example Queries

- To fetch NAME and ADDRESS of students from table Student where Age is 18 or 20.

SELECT NAME, ADDRESS FROM Student WHERE Age IN (18,20);

Output :

NAME	ADDRESS
RAM	DELHI
RAMESH	SHIMLA
SUJIT	DHARAMSHALA
SURESH	DELHI
PRATIK	MANDI

- To fetch records of students from table Student where ROLL_NO is 1 or 4.

SELECT * FROM Student WHERE ROLL_NO IN (1,4);

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAMESH	DELHI	XXXXXXXXXX	18
4	SURESH	DELHI	XXXXXXXXXX	18

AND ,OR and NOT operators

In SQL, the AND & OR operators are used for filtering the data and getting precise result based on conditions.

- The AND and OR operators are used with the WHERE clause.
- These two operators are called conjunctive operators.

AND Operator : This operators displays only those records where both the conditions condition1 and condition2 evaluates to True.

OR Operator : This operators displays the records where either one of the conditions condition1 and condition2 evaluates to True. That is,either condition1 is True or condition2 is True.

- **AND Operator**

Syntax :

SELECT * FROM table_name WHERE condition1 AND condition2 and ...conditionN ;

table_name: name of the table

Sample Queries :

- To fetch all the records from Student table where Age is 18 and Address is DELHI.

SELECT * FROM Student WHERE AGE = 18 AND ADDRESS = 'DELHI';

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
4	SURESH	DELHI	XXXXXXXXXX	18

- To fetch all the records from Student table where NAME is Ram and Age is 18.

SELECT * FROM Student WHERE AGE = 18 AND NAME = 'RAM';

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	Age
1	RAM	DELHI	XXXXXXXXXX	18

- **OR Operator**

Syntax :

SELECT * FROM table_name WHERE condition1 OR condition2 OR... conditionN ;

table_name : name of the table

condition1,2,..N : first condition, second condition and so on

Sample Queries :

To fetch all the records from Student table where name is RAM or name is SUJIT.

SELECT * FROM Student WHERE NAME = 'RAM' OR NAME = 'SUJIT';

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20

- To fetch all the records from Student table where name is RAM or age is 20.

SELECT * FROM Student WHERE NAME = 'RAM' OR AGE = 20;

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
6	PRATIK	MANDI	XXXXXXXXXX	20

- **NOT Operator**

NOT operator returns false if following condition is true.

Syntax :

SELECT * FROM table_name WHERE condition1 OR condition2 OR... conditionN;

table_name: name of the table

condition1,2,..N : first condition, second condition and so on

Example Queries

- To fetch all the records from Student table where Age is not equal to 20.

SELECT * FROM Student WHERE AGE<>20;

Output :

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	GURGAON	XXXXXXXXXX	18
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19

ORDER BY

The ORDER BY statement in SQL is used to sort the fetched data in either ascending or descending according to one or more columns.

- By default ORDER BY sorts the data in ascending order.
- We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.
- **Sort according to one column:** To sort in ascending or descending order we can use the keywords ASC or DESC respectively.

Syntax :

SELECT * FROM table_name ORDER BY column_name ASC|DESC

table_name : name of the table.

column_name : name of the column according to which the data is needed to be arranged.

ASC : to sort the data in ascending order.

DESC : to sort the data in descending order.

Sort according to multiple columns : To sort according to multiple columns, separate the names of columns by (,) operator.

Syntax :

SELECT * FROM table_name ORDER BY column1 ASC|DESC , column2 ASC|DESC

Sample table Student :

Student				
ROLL _NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	PRATIK	MANDI	XXXXXXXXXX	20

Example Queries :

- **Sort according to single column**: In this example we will fetch all data from the table **Student** and sort the result in descending order according to the column **ROLL_NO**.

SELECT * FROM Student ORDER BY ROLL_NO DESC;

Output :

ROLL _NO	NAME	ADDRESS	PHONE_NO	AGE
6	PRATIK	MANDI	XXXXXXXXXX	20
5	HARSH	SOLAN	XXXXXXXXXX	19
4	SURESH	DELHI	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
2	RAMESH	SHIMLA	XXXXXXXXXX	18
1	RAM	DELHI	XXXXXXXXXX	18

To sort in ascending order we have to use ASC in place of DESC.

- **ALTER (ADD, DROP, MODIFY)**

Tables can be altered in one of the three ways: by adding a column to existing table ,by changing a column's definition or by dropping a column.

ALTER TABLE command is used to add, delete/drop or modify columns in the existing table. It is also used to add and drop various constraints on the existing table.

ALTER TABLE – ADD

ADD is used to add columns into the existing table. Sometimes we may require to add additional information, in that case we do not require to create the whole database again, **ADD** comes to our rescue.

Syntax :

ALTER TABLE table_name
ADD (Columnname_1 datatype,
Columnname_2 datatype,
...
Columnname_n datatype);

table_name – Name of the table

columnname- Name of the new column.

datatype - Datatype of the new column

ALTER TABLE – DROP

You can drop a column from a table by using the ALTER TABLE statement DROP COLUMN clause is used to delete the unwanted columns from the table.

Syntax :

ALTER TABLE table_name

DROP COLUMN column_name;

table_name – Name of the table

column_name- Name of the column that you want to delete.

ALTER TABLE-MODIFY

You can modify a column definition by using the ALTER TABLE statement with the **MODIFY**clause. Which is used to modify the existing columns in a table. Column modification can include changes to a column's datatype, size and default value.Multiple columns can also be modified at once.

Syntax :

ALTER TABLE table_name

MODIFY column_name column_type;

Sample table Subject.

Subject	
ROLL_NO	**NAME**
1	RAM
2	RAMESH
3	SUJIT
4	SURESH
5	HARSH
6	PRATIK

QUERY :

- To ADD 2 columns AGE and SUBJECT to table Subject.

  ALTER TABLE Subject ADD (AGE int(3),SUBJECT varchar(40));

OUTPUT :

ROLL_NO	NAME	AGE	SUBJECT
1	RAM		
2	RAMESH		
3	SUJIT		
4	SURESH		
5	HARSH		
6	PRATIK		

- **MODIFY column SUBJECT in table Subject**

  ALTER TABLE Subject MODIFY SUBJECTvarchar(20);

After running the above query maximum size of SUBJECT Column is reduced to 20 from 40.

- **DROP column SUBJECT in table Subject**

  ALTER TABLE Subject DROP COLUMN SUBJECT;

OUTPUT :

ROLL_NO	NAME	AGE
1	RAM	
2	RAMESH	
3	SUJIT	
4	SURESH	
5	HARSH	
6	PRATIK	

- **UPDATE**

The UPDATE statement in SQL is used to update the data of an existing table in database. We can update single columns as well as multiple columns using UPDATE statement as per our requirement.

Syntax

UPDATE table_name SET column1 = value1, column2 = value2,...

WHERE condition;

table_name: name of the table

column1: name of first , second, third column....

value1: new value for first, second, third column....

condition: condition to select the rows for which the values of columns needs to be updated.

NOTE : In the above query the **SET** statement is used to set new values to the particular column and the **WHERE** clause is used to select the rows for which the columns are needed to be updated. If we have not used the WHERE clause then the columns in **all** the rows will be updated. So the WHERE clause is used to choose the particular rows.

Sample table:

Student				
ROLL _NO	**NAME**	**ADDRESS**	**PHONE_NO**	**AGE**
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	PRATIK	MANDI	XXXXXXXXXX	20

Example Queries

- **Updating single column**: Update the column NAME in Student table and set the value to 'DEEPAK' in all the rows where Age is 20.

  UPDATE Student SET NAME = 'DEEPAK' WHERE Age = 20;

Output :

This query will update two rows(third row and sixth row) and the table **Student** will now look like,

ROLL _NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	DEEPAK	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	DEEPAK	MANDI	XXXXXXXXXX	20

Updating Multiple Columns : Update the columns NAME to 'PRATIK' and ADDRESS to 'MANDI' where ROLL_NO is 1.

UPDATE Student SET NAME = 'PRATIK', ADDRESS = 'MANDI' WHERE ROLL_NO = 1;

Output :

The above query will update two columns in the first row and the table **Student** will now look like,

ROLL _NO	NAME	ADDRESS	PHONE_NO	AGE
1	PRATIK	MANDI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	PRATIK	MANDI	XXXXXXXXXX	20

Note : For updating multiple columns we have used comma(,) to separate the names and values of two columns.

Distinct Clause

The distinct keyword is used in conjunction with SELECT keyword. It is helpful when there is need of avoiding the duplicate values present in any specific columns/table. When youn use DISTINCT keyword only the unique values are fetched.

Syntax :

SELECT DISTINCT column1,column2 FROM table_name;

column1 , column2 : names of the fields of the table

table_name: from where we want to fetch

This query will return all the unique combination of rows in the table with fields

column1 , column2.

NOTE: If distinct keyword is used with multiple columns, the distinct combination is displayed in the result set.

Example Queries

- To fetch unique names from the NAME field of Student table

SELECT DISTINCT NAME FROM Student;

Output :

NAME
RAM
RAMESH
SUJIT
SURESH
HARSH
PRATIK

- To fetch unique combination of rows from the whole table

SELECT DISTINCT ADDRESS FROM Student;

Output:

ADDRESS
DELHI
SHIMLA
DHARAMSHALA
SOLAN
MANDI

Arithmetic Operators

We can use various Arithmetic Operators on the data stored in the tables.

Operators	Descriptions	Examples
+	It is used to add containing values of both operands	a+b will give 150
-	It subtracts right hand operand from left hand operand	a-b will give -50
*	It multiply both operand's values	a*b will give 5000
/	It divides left hand operand by right hand operand	b/a will give 2
%	It divides left hand operand by right hand operand and returns remainder.	b%a will give 0

Sample Table Student Fee

Student Fee		
ROLL_NO	**MONTH-YEAR**	**FEE**
1	01-2019	1000
2	01-2019	1000
3	01-2019	1000
4	01-2019	1000
5	01-2019	1000
6	01-2019	1000

Addition (+) :

It is used to perform **addition operation** on the data items, items include either single column or multiple columns.

Syntax :

SELECT Column name operator

Example Query

SELECT ROLL_NO,MONTH-YEAR, FEE, FEE + 200

FROM Student Fee;

Output:

ROLL_NO	MONTH-YEAR	FEE	FEE+200
1	01-2019	1000	1200
2	01-2019	1000	1200
3	01-2019	1000	1200
4	01-2019	1000	1200
5	01-2019	1000	1200
6	01-2019	1000	1200

Defining column Alias

Aliases are the temporary names given to table or column for the purpose of a particular SQL query. It is used when name of column or table is used other than their original names, but the modified name is only temporary.

- Aliases are created to make table or column names more readable.
- The renaming is just a temporary change and table name does not change in the original database.
- Aliases are useful when table or column names are big or not very readable.
- These are preferred when there are more than one table involved in a query.

Syntax :

- For column alias :

SELECT column as alias_name FROM table_name;

column: fields in the table

alias_name: temporary alias name to be used in replacement of original column name

table_name: name of table

Example Query

- fetch ROLL_NO , NAME from Student table using S_ CODE , S_NAME as alias name respectively.

SELECT ROLL_NO As S_CODE, NAME AS S_NAME FROM Student;

Output :

S_CODE	S_NAME
1	RAM
2	RAMESH
3	SUJIT
4	SURESH
5	HARSH
6	PRATIK

Transactions

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: **success** or **failure**.

Incomplete steps result in the failure of the transaction. A database transaction, by definition, must be atomic, consistent, isolated and durable. These are popularly known as ACID (Atomicity, Consistency, Isolation, and Durability) properties.

How to implement Transactions using SQL ?

Following commands are used to control transactions. It is important to note that these statements cannot be used while creating tables and are only used with the DML Commands such as – INSERT, UPDATE and DELETE.

- **SET TRANSACTION :** Places a name on a transaction.

Syntax :

SET TRANSACTION [READ WRITE | READ ONLY];

- **COMMIT :**

COMMIT is the SQL command that is used for storing changes performed by a transaction. When a COMMIT command is issued it saves all the changes since last COMMIT or ROLLBACK.

Syntax :

COMMIT;

Example: Sample table 1

Student				
ROLL_NO	**NAME**	**ADDRESS**	**PHONE_NO**	**AGE**
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	PRATIK	MANDI	XXXXXXXXXX	20

Following is an example which would delete those records from the table which have age = 20 and then COMMIT the changes in the database.

Example Queries:

DELETE FROM Student WHERE AGE = 20;

COMMIT;

Output :

Thus, two rows from the table would be deleted and the SELECT statement would look like,

OUTPUT

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19

- **ROLLBACK :**

ROLLBACK is the SQL command that is used for reverting changes performed by a transaction. When a ROLLBACK command is issued it reverts all the changes since last COMMIT or ROLLBACK.

Syntax :

ROLLBACK;

Example Query :

From the above example Sample table Student,

Delete those records from the table which have age = 20 and then ROLLBACK the changes in the database.

DELETE FROM Student WHERE AGE = 20;

ROLLBACK;

Output:

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	PRATIK	MANDI	XXXXXXXXXX	20

- **SAVEPOINT :** A SAVEPOINT is a point in a transaction in which you can roll the transaction back to a certain point without rolling back the entire transaction.

Syntax :

SAVEPOINT SAVEPOINT_NAME;

This command is used only in the creation of SAVEPOINT among all the transactions.

In general ROLLBACK is used to undo a group of transactions.

Syntax for rolling back to Savepoint command:

ROLLBACK TO SAVEPOINT_NAME;

You can ROLLBACK to any SAVEPOINT at any time to return the appropriate data to its original state.

Example query:

From the above example **Sample table Student**,

Delete those records from the table which have age = 20 and then ROLLBACK the changes in the database by keeping Savepoints.

Example Queries:

SAVEPOINT SP1;

//Savepoint created.

DELETE FROM Student WHERE AGE = 20;

//deleted

SAVEPOINT SP2;

//Savepoint created.

Here SP1 is first SAVEPOINT created before deletion. In this example one deletion have taken place.

After deletion again SAVEPOINT SP2 is created.

Output:

Roll_No	Name	Adress	Phone	Age
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
4	SURESH	DELHI	XXXXXXXXXX	18

Deletion have been taken place, let us assume that you have changed your mind and decided to ROLLBACK to the SAVEPOINT that you identified as SP1 which is before deletion.

deletion is undone by this statement .

ROLLBACK TO SP1;

//Rollback completed.

Output:

ROLL_NO	NAME	ADDRESS	PHONE_NO	AGE
1	RAM	DELHI	XXXXXXXXXX	18
2	RAMESH	SHIMLA	XXXXXXXXXX	18
3	SUJIT	DHARAMSHALA	XXXXXXXXXX	20
4	SURESH	DELHI	XXXXXXXXXX	18
5	HARSH	SOLAN	XXXXXXXXXX	19
6	PRATIK	MANDI	XXXXXXXXXX	20

Notice that first deletion took place even though you rolled back to SP1 which is first SAVEPOINT.

- **RELEASE SAVEPOINT :-** This command is used to remove a SAVEPOINT that you have created.

Syntax :

RELEASE SAVEPOINT SAVEPOINT_NAME

Once a SAVEPOINT has been released, you can no longer use the ROLLBACK command to undo transactions performed since the last SAVEPOINT.

Creating User Accounts in SQL

The DBA creates the user by executing the CREATE USER statement. The user does not have any privileges at this point. The DBA can then grant number of privileges to that user. These privileges determine what the user can do at database level.

Syntax :

CREATE USER user

IDENTIFIED BY password;

User : is the name of user to be created.

Password : specifies that the user must log in with password

Example :

CREATE USER DEEPAK

IDENTIFIED BY 'DHARAMSHALA';

SQL GRANT REVOKE Commands

DCL commands are used to enforce database security in a multiple user database environment. Two types of DCL commands are GRANT and REVOKE. Only Database Administrator's or owner's of the database object can provide/remove privileges on a database object.

- **SQL GRANT Command**

SQL GRANT is a command used to provide access or privileges on the database objects to the users.

Syntax

GRANT privilege_name

ON object_name

TO {user_name |PUBLIC |role_name};

privilege_name is the access right or privilege granted to the user. Some of the access rights are ALL, EXECUTE, and SELECT.

object_name is the name of an database object like TABLE, VIEW, STORED PROC and SEQUENCE.

user_name is the name of the user to whom an access right is being granted.

Example :

Give the user DEEPAK all data manipulation permission on the table Student.

GRANT ALL

ON Student

TO DEEPAK;

- **REVOKE Statement**

The Revoke statement is used to revoke some or all of the privileges which have been granted to a user in the past.

Syntax :

```
REVOKE    privilege_name
ON        object_name
FROM      {user_name |PUBLIC |role_name};
```

object : It is the name of the database object from which permissions are being revoked. In the case of revoking privileges from a table, this would be the table name.

user : It is the name of the user from whom the privileges are being revoked.

Example. Revoke the permission on Student table to user DEEPAK.

```
REVOKE    ALL
ON        Student
FROM      DEEPAK;
```

DELETING USER ACCOUNTS

To delete a user account (along with associated rights and privileges), you can use DROP USER statement.

Syntax :

```
DROP USER user_name
```

Example: delete the user DEEPAK

```
DROP USER DEEPAK;
```

Constraints

Constraints are the rules enforced on the data columns of a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints could be either on a column level or a table level. The column level constraints are applied only to one column, whereas the table level constraints are applied to the whole table.

Following are some of the most commonly used constraints available in MYSQL.

- **NOT NULL Constraint** "Ensures that a column cannot have NULL value.
- **DEFAULT Constraint** "Provides a default value for a column when none is specified.
- **UNIQUE Constraint** "Ensures that all values in a column are different.
- **PRIMARY Key** "Uniquely identifies each row/record in a database table.
- **FOREIGN Key** "Uniquely identifies a row/record in any of the given database table.
- **CHECK Constraint** "The CHECK constraint ensures that all the values in a column satisfies certain conditions.

How to specify constraints ?

We can specify constraints at the time of creating the table using CREATE TABLE statement. We can also specify the constraints after creating a table using ALTER TABLE statement.

Syntax :

Below is the syntax to create constraints using CREATE TABLE statement at the time of creating the table.

```
CREATE TABLE table_name
(
column1 data_type(size) constraint_name,
column2 data_type(size) constraint_name,
column3 data_type(size) constraint_name,
....
);
```

table_name: Name of the table to be created.

data_type: Type of data that can be stored in the field.

constraint_name: Name of the constraint. For example- NOT NULL, UNIQUE, PRIMARY KEY etc.

1. **NOT NULL :** By default, a column can hold NULL values. If you do not want a column to have a NULL value, then you need to define such a constraint on this column specifying that NULL is now not allowed for that column. This constraint is placed immediately after the data type of a column.

Note : A NULL is not the same as no data, rather, it represents unknown data.

Example :

For example, the following SQL query creates a new table called Student1 and adds five columns, three of which, are Roll_No,Name and Age. In this we specify not to accept NULLs "

```
CREATE TABLE Student1(
Roll_No  INT (6)          NOT NULL,
  Name      VARCHAR (20)     NOT NULL,
Address  CHAR(25),
  Age       INT (4)          NOT NULL,
  Fee       DECIMAL(18,2)
  );
```

2. **DEFAULT :** The DEFAULT constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value.

Example

For example, the following SQL creates a new table called Student1 and adds five columns. Here, the Fee column is set to 1000.00 by default, so in case the INSERT INTO statement does not provide a value for this column, then by default this column would be set to 1000.00.

```
CREATE TABLE Student1(
Roll_No  INT  (6)          NOT NULL,
  Name      VARCHAR (20)     NOT NULL,
Address  CHAR(25),
  Age       INT  (6)         NOT NULL,
  Fee       DECIMAL(18,2)  DEFAULT 1000.00
);
```


3. **Unique Key Constraint :** The UNIQUE Constraint prevents two records from having identical values in a column. It ensures values entered into a column are unique. This can apply only to columns that have also been declared as NOT NULL. In the student1 table, for example, you might want to prevent two or more students from having an identical age.

Example

For example, the following SQL query creates a new table called Student1 and adds five columns. Here, the Age column is set to UNIQUE, so that you cannot have two records with the same age.

```
CREATE TABLE Student1(
Roll_No  INT  (6)          NOT NULL,
  Name      VARHAR (20)     NOT NULL,
Address  CHAR(25),
  Age       INT (6)          NOT NULL UNIQUE,
  Fee       DECIMAL(18,2)
);
```

You can also define a group of columns as unique with a UNIQUE constraint.

4. **Primary Key :** A primary key is a field in a table which uniquely identifies each record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.A table can have only one primary key.If a table has a primary key defined on any field, then you cannot have two records having the same value of that field.

Note : Primary Key constraint is same as UNIQUE constraint ,except that only one PRIMARY KEY can be defined in table.

Here is the syntax to define the Roll_No attribute as a primary key in a Student1 table.

```
CREATE TABLE Student1(
    Roll_No  INT (6)  NOT NULL PRIMARY KEY,
    Name     VARCHAR (20) NOT NULL,
    Address CHAR(25),
    Age      INT (6) NOT NULL UNIQUE,
    Fee      DECIMAL(18,2)
    );
```

5. **FOREIGN KEY** : A foreign key is a key used to link two tables together.A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table. A foreign key constraint also known as a Referential Integrity Constraint, specifies that the value of the foreign key corresponds to actual values of primary key in the other table. This is sometimes also called as a referencing key. A Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table.

The relationship between two tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

Consider the structure of the following two tables.

Student1 table:

```
CREATE TABLE Student1(
    Roll_No  INT (6) NOT NULL,
    Name     VARCHAR (20) NOT NULL,
    Address CHAR(25),
    Age      INT (6) NOT NULL,
    Fee      DECIMAL(18,2)
    PRIMARY KEY(Roll_No)
    );
```


Subject table :

```
CREATE TABLE Subject (
    Subject_id  INT (6)  NOT NULL,
    Subject_ Name VARCHAR (20),
    Roll_No INT(4),
    PRIMARY  KEY(Subject_id),
    FOREIGN KEY (Roll_No)  REFERENCES Student1(Roll_No)
    );
```

6. **CHECK CONSTRAINT :** The CHECK Constraint enables a condition to check the value being entered into a record. If the condition evaluates to false, the record violates the constraint and isn't entered the table.

Example

For example, the following program creates a new table called Student1 and adds five columns. Here, we add a CHECK with Age column, so that you cannot have any Student who is below 18 years.

```
CREATE TABLE Student1(
    Roll_No  INT (6)  NOT NULL PRIMARY KEY,
    Name     VARCHAR (20)     NOT NULL,
    Address  CHAR(25),
    Age      INT (6) NOT NULL CHECK (AGE >=18),
    Fee      DECIMAL(18,2)
    );
```

- **ADDING CONSTRAINT TO EXISTING TABLE**

You can add a constraints for existing tables by using a ALTER TABLE statement with ADD clause

Syntax

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint] type (column);
```

In the Syntax

table_name : is name of the table.

Constraint: name of the constraint

type : is the constraint type

Column : is the name of column affected by constraint.

CONCEPT OF KEYS

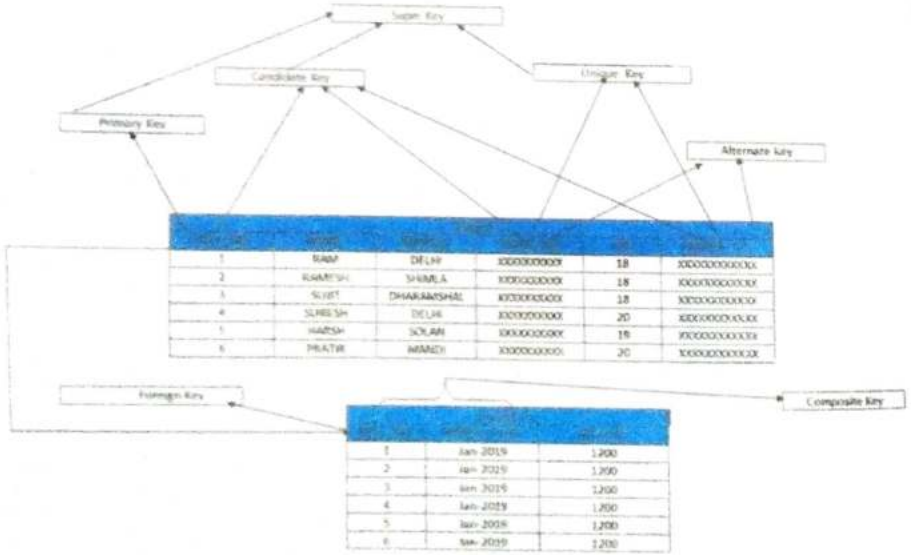
What is Key ?

Keys are fields in a table which participate in below activities in RDBMS systems :

1. To create relationships between two tables.
2. To maintain uniqueness in a table.
3. To keep consistent and valid data in database.
4. Might help in fast data retrieval by facilitating indexes on column(s).

SQL Server supports various types of keys, which are listed below :

1. Candidate Key
2. Primary Key
3. Unique Key
4. Alternate Key
5. Composite Key
6. Super Key
7. Foreign Key



1. **Candidate Key :** Candidate key is a key of a table which can be selected as a primary key of the table. A table can have multiple candidate keys, out of which one can be selected as a primary key.

Example : In above sample table **Student**, ,ROLL_NO,PHONE_NO, and AADHAR_NO are candidate keys.

2. **Primary Key :** Primary key is a candidate key of the table selected to identify each record uniquely in table. Primary key does not allow null value in the column and keeps unique values throughout the column. In above example, ROLL_NO is a primary key of Student table. A table can have only one primary key and primary key can be defined in SQL Server using below SQL statements :
   1. CRETE TABLE statement (at the time of table creation) – In this case, system defines the name of primary key.
   2. ALTER TABLE statement (using a primary key constraint) – User defines the name of the primary key

Example : ROLL_NO is a primary key of Student table.

3. **Unique Key :** Unique key is similar to primary key and does not allow duplicate values in the column.The difference between primary key and Unique key is that ,it allows one null value in the column.

Example : PHONE_NO and AADHAR_NO are Unique keys of Student table.

4. **Alternate Key :** Alternate key is a candidate key, currently not selected as primary key of the table.

Example : PHONE_NO and AADHAR_NO are alternate keys of student table.

5. **Composite Key :** Composite key (also known as compound key or concatenated key) is a group of two or more columns that identifies each row of a table uniquely. Individual column of composite key might not able to uniquely identify the record. It can be a primary key or candidate key also.

Example : In Student Fee table, ROLL_NO and MONTH-YEAR are combined together to identify each row uniquely in Student Fee table.

6. **Super Key :** Super key is a set of columns on which all columns of the table are functionally dependent. It is a set of columns that uniquely identifies each row in a table. Super key may hold some additional columns which are not strictly required to uniquely identify each row. Primary key and candidate keys are minimal super keys or you can say subset of super keys.

In above example, In Student table, column ROLL_NO is sufficient to uniquely identify any row of the table, so that any set of column from Student table which contains ROLL_NO is a super key for Student Table.

For **example:** {ROLL_NO}, {ROLL_NO, NAME}, {ROLL_NO, NAME, ADDRESS} etc.

7. **Foreign Key :** In a relationship between two tables, a primary key of one table is referred as a foreign key in another table. Foreign key can have duplicate values in it and can also keep null values if column is defined to accept nulls.

Example : ROLL_NO (primary key of Student table) is a foreign key in Student Fee table.

Difference between Primary and Candidate Key :

S.NO	Primary Key	Candidate Key
1.	Primary key is a minimal super key. So there is one and only one primary key in a relation.	While in a relation there can be more than one candidate key.
2.	Any attribute of Primary key can not contain NULL value.	While in Candidate key any attribute can contain NULL value.
3.	Primary key can be optional to specify any relation.	But without candidate key there can't be specified any relation.
4.	Primary key specifies the important attribute for the relation.	Candidate specifies the key which can qualify for primary key.
5.	It is confirmed that a primary key is a candidate key.	But it is not confirmed that a candidate key can be a primary key.

Functions

For doing operations on data SQL has many built-in functions, they are categorized in two main categories and further sub-categorized in different seven functions under each category. The categories are:

1. **Aggregate functions :** These functions are used to do operations from the values of the column and a single value is returned.
   1. AVG()
   2. COUNT()
   3. FIRST()
   4. LAST()
   5. MAX()
   6. MIN()
   7. SUM()
2. **Scalar functions :** These functions are based on user input, these too returns single value.
   1. UCASE()
   2. LCASE()
   3. MID()
   4. LEN()
   5. ROUND()
   6. NOW()

Marks table

ROLL_NO	NAME	MARKS	AGE
1	RAM	90	18
2	RAMESH	50	18
3	SUJIT	80	20
4	SURESH	95	18
5	HARSH	85	19

Aggregate Functions

- **AVG()**: It returns average value after calculating from values in a numeric column.

Syntax :

SELECT AVG(column_name) FROM table_name;

Queries :

1. **Computing average marks of students from Marks table:**

SELECT AVG(MARKS) AS AvgMarks FROM Marks;

Output:

AvgMarks
80

2. **Computing average age of students from Marks table:**

SELECT AVG(AGE) AS AvgAge FROM Marks;

Output:

AvgAge
18.6

- **COUNT():** It is used to count the number of rows returned in a SELECT statement.

Syntax:

SELECT COUNT(column_name) FROM table_name;

Queries :

1. **Computing total number of students.**

SELECT COUNT(*) AS NumStudents FROM Marks;

Output :

NumStudents
5

2. **Computing number of students with unique/distinct age.**

SELECT COUNT(DISTINCT AGE) AS NumStudents FROM Marks;

Output :

NumStudents
3

- **FIRST():** The FIRST() function returns the first value of the selected column.

Syntax :

SELECT FIRST(column_name) FROM table_name;

Queries :

1. **Fetching marks of first student from the Marks table.**

SELECT FIRST(MARKS) AS MarksFirst FROM Marks;

Output :

MarksFirst
90

2. **Fetching age of first student from the Marks table.**

SELECT FIRST(AGE) AS AgeFirst FROM Marks;

Output :

AgeFirst
18

- **LAST():** The LAST() function returns the last value of the selected column.

Syntax :

SELECT LAST(column_name) FROM table_name;

Queries :

1. **Fetching marks of last student from the Student table.**

SELECT LAST(MARKS) AS MarksLast FROM Student;

Output :

MarksLast
82

2. **Fetching age of last student from the Marks table.**

SELECT FIRST(AGE) AS AgeLast FROM Marks;

Output :

AgeLast
19

- **MAX()** : The MAX() function returns the maximum value of the selected column.

Syntax :

SELECT MAX(column_name) FROM table_name;

Queries :

1. **Fetching maximum marks among students from the Marks table.**

SELECT MAX(MARKS) AS MaxMarks FROM Marks;

Output :

MaxMarks

95

2. **Fetching max age among students from the Marks table.**

SELECT MAX(AGE) AS MaxAge FROM Marks;

Output :

MaxAge

20

- **MIN():** The MIN() function returns the minimum value of the selected column.

Syntax:

SELECT MIN(column_name) FROM table_name;

Queries :

1. **Fetching minimum marks among students from the Marks table.**

SELECT MIN(MARKS) AS MinMarks FROM Marks;

Output :

MinMarks

50

2. **Fetching minimum age among students from the Marks table.**

SELECT MIN(AGE) AS MinAge FROM Marks;

Output :

MinAge

18

- **SUM():** The SUM() function returns the sum of all the values of the selected column.

Syntax :

SELECT SUM(column_name) FROM table_name;

Queries :

1. **Fetching summation of total marks among students from the Marks table.**

SELECT SUM(MARKS) AS TotalMarks FROM Marks;

Output :

TotalMarks

400

2. **Fetching summation of total age among students from the Marks table.**

SELECT SUM(AGE) AS TotalAge FROM Marks;

Output :

TotalAge

93

Scalar Functions

- **UCASE()**: It converts the value of a field to uppercase.

Syntax :

SELECT UCASE(column_name) FROM table_name;

Queries :

1. **Converting names of students from the table Marks to uppercase.**

SELECT UCASE(NAME) FROM Marks;

Output:

NAME
RAM
RAMESH
SUJIT
SURESH
HARSH

- **LCASE()**: It converts the value of a field to lowercase.

Syntax :

SELECT LCASE(column_name) FROM table_name;

Queries :

1. **Converting names of students from the table Marks to lowercase.**

SELECT LCASE(NAME) FROM Marks;

Output :

NAME
ram
ramesh
sujit
suresh
harsh

- **MID():** The MID() function extracts texts from the text field.

Syntax :

SELECT MID(column_name,start,length) AS some_name FROM table_name ; specifying length is optional here, and start signifies start position (starting from 1)

Queries:

1. **Fetching first four characters of names of students from the Marks table.**

SELECT MID(NAME,1,3) FROM Marks;

Output :

NAME
RAM
RAM
SUJ
SUR
HAR

- **LEN():** The LEN() function returns the length of the value in a text field.

Syntax:

SELECT LENGTH(column_name) FROM table_name;

Queries :

1. **Fetching length of names of students from Marks table.**

SELECT LENGTH(NAME) FROM Marks;

Output :

NAME
3
6
5
6
5

- **ROUND() :** The ROUND() function is used to round a numeric field to the number of decimals specified.

NOTE : Many database systems have adopted the IEEE 754 standard for arithmetic operations, which says that when any numeric .5 is rounded it results to the nearest even integer i.e. 5.5 and 6.5 both gets rounded off to 6.

Syntax :

SELECT ROUND(column_name,decimals) FROM table_name;

decimals- number of decimals to be fetched.

Queries :

1. **Fetching maximum marks among students from the Marks table.**

SELECT ROUND(MARKS,0) FROM Marks;

Output :

MARKS
90
50
80
95
85

- **NOW():** The NOW() function returns the current system date and time.

Syntax :

SELECT NOW() FROM table_name;

Queries :

1. **Fetching current system time.**

SELECT NAME, NOW() AS DateTime FROM Marks;

Output :

NAME	DateTime
RAM	19/02/2019 1:30:11 PM
RAMESH	19/02/2019 1:30:11 PM
SUJIT	19/02/2019 1:30:11 PM
SURESH	19/02/2019 1:30:11 PM
HARSH	19/02/2019 1:30:11 PM

SUMMARY

- **Data** is a raw and unorganized fact that required to be processed to make it meaningful.
- **Information** is the processed data on which decision and actions are based.
- **Database** is a collection of inter-related data. It is composed of a collection of files that are linked in such a way that information from one of the files may be combined with information from other files
- **ACID** - The acronym standing for the properties maintained by standard database management systems, standing for Atomicity, Consistency, Isolation, and Durability.
- A database is a collection of **tables.**
- A database **table** is composed of records and fields that hold data
- Each table contains rows of data, which are also called **records**..
- **Record** is composed of fields and contains all the data about one particular person, company, or item in a database.
- A **column** is the smallest unit of storage in a relational database.
- A row (record, tuple) is a collection of column values.
- A **field** is part of a record and contains a single piece of data for the subject of the record
- Collection of records is called a **file.**
- Process of hiding irrelevant details from user is called data abstraction.
- **Data redundancy** means duplication of data.
- **Data Type** The basic kind of data that can be stored in a column.
- **DBMS** stands for **Database Management system**. It is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.
- The collection of Information stored in database at a particular instance of time is called **instance** of database.

- Process of hiding irrelevant details from user is called **data abstraction**.
- The overall design of the database is called database **schema.**
- A **data dictionary** contains metadata i.e data about the database. It is a dictionary about the data that we store in the database.
- **Database integrity** refers to the validity and consistency of stored data.
- **SQL stands for Structured Query Language**, which is a standardized language for interacting with RDBMS (Relational Database Management System).
- **SQL commands** are instructions, coded into SQL statements, which are used to communicate with the database to perform specific tasks, work, functions and queries with data.
- A **query**is a type of command that retrieves data from the server.
- A **view**is an alternative way to present a table (or tables).
- When you issue a query to a database, you get back a **result set.**
- **Key** - A column or columns on which an index is constructed to allow rapid and/or sorted access to a table's row.
- **Candidate key** is a key of a table which can be selected as a primary key of the table. A table can have multiple candidate keys, out of which one can be selected as a primary key.
- **Primary key** is a candidate key of the table selected to identify each record uniquely in table.
- **Alternate key** is a candidate key, currently not selected as primary key of the table.
- **Composite key** (also known as compound key or concatenated key) is a group of two or more columns that identifies each row of a table uniquely.
- In a relationship between two tables, a primary key of one table is referred as a **foreign key** in another table
- **Super key** is a set of columns on which all columns of the table are functionally dependent

- A **transaction**is a collection of database operations that are treated as a unit.
- A **commit**marks the successful end of a transaction.
- A **rollback** marks the unsuccessful end of a transaction.
- **Constraints** are the rules enforced on the data columns of a table

TRUE/FALSE

Q.1 State whether the following statements are True or False :

1. Data redundancy means repetition or duplication of data.
2. SQL statements are case sensitive.
3. SELECT is Data Definition Statement.
4. Collections of fields is called a record.
5. Primary key does not allow null value in the column.
6. Null value is same as a zero or blank space.
7. COMMIT is a Transaction control statement.
8. DISTINCT clause is used to avoid the duplicate values present in any specific columns.
9. DROP TABLE command eliminates a table from database.
10. The ORDER BY clause is used to sort the rows.

Solution :

1. True	**2.** False	**3.** False	**4.** True	**5.** True
6. False	**7.** True	**8.** True	**9.** True	**10.** True.

EXERCISE

1. What is the difference between Data and Information ?
2. What is Database ?
3. What is RDBMS ? What are advantages and Disadvantages of RDBMS ?
4. What are limitations of Traditional File System ?
5. What is the difference between DBMS and Traditional File Oriented System.

6. Discuss three tier structure of DBMS.
7. Who is DBA and what are the different roles and responsibilities of DBA ?
8. What do you mean by SQL ? What are the characteristics of SQL ?
9. What are different data types in SQL ? Explain.
10. Differentiate between char and varchar data types.
11. What do you mean by constraints ? Explain.
12. What is Domain integrity constraint ?
13. What is referential integrity constraint ?
14. What do you mean by DML and DDL ? Explain
15. What do you mean by DCL and TCL ? Explain.
16. What do you mean by table ? How can you create a table in SQL ?
17. What is the function of SELECT command ? Explain.
18. How can you insert rows in table ?
19. What is the use of DISTINCT clause in SELECT statement ?
20. What is the purpose of WHERE clause with SELECT statement ?
21. What is the function of ORDER BY clause ? Explain.
22. Discuss the use of AND, OR and NOT operators in SQL.
23. Why BETWEEN operator is used ?
24. How can you remove a table in database ?
25. How can you create users in SQL ?
26. How can you delete user accounts in SQL ?

—End—

CHAPTER - 6

PL/SQL

6.1 Introduction :

PL/SQL is an extension of Structured Query Language (SQL) that is used in Oracle. Unlike SQL, PL/SQL allows the programmer to write code in a procedural format. Full form of PL/SQL is "Procedural Language extensions to SQL".PL/SQL is superset of SQL.

It combines the data manipulation power of SQL with the processing power of procedural language to create super powerful SQL queries.

Architecture of PL/SQL

The PL/SQL architecture mainly consists of following three components:

1. PL/SQL block
2. PL/SQL Engine
3. Database Server

Def : PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements .All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

Features of PL/SQL :

1. PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.
2. PL/SQL can execute a number of queries in one block using single command.