Higher Secondary Course

XII

# Computer Science

Part II

Government of Kerala

DEPARTMENT OF EDUCATION

State Council of Educational Research and Training (SCERT); Kerala

2015

## THE NATIONAL ANTHEM

Jana-gana-mana adhinayaka, jaya he
Bharatha-bhagya-vidhata.
Punjab-Sindh-Gujarat-Maratha
Dravida-Utkala-Banga
Vindhya-Himachala-Yamuna-Ganga
Uchchala-Jaladhi-taranga
Tava subha name jage,
Tava subha asisa mage,
Gahe tava jaya gatha.
Jana-gana-mangala-dayaka jaya he
Bharatha-bhagya-vidhata.
Jaya he, jaya he, jaya he,
Jaya jaya jaya, jaya he!

## PLEDGE

India is my country. All Indians are my brothers and sisters.

I love my country, and I am proud of its rich and varied heritage. I shall always strive to be worthy of it.

I shall give respect to my parents, teachers and all elders and treat everyone with courtesy.

I pledge my devotion to my country and my people. In their well-being and prosperity alone lies my happiness.

*Prepared by*.
State Council of Educational Research and Training (SCERT)
Poojappura, Thiruvananthapuram 695012, Kerala
*Website:* www.scertkerala.gov.in *e-mail* : scertkerala@gmail.com
Phone : 0471–2341883, Fax: 0471–2341869
Typesetting and Layout: SCERT

Dear students,

Computer Science, a subject belonging to the discipline of Science and of utmost contemporary relevance, needs continuous updating. The Higher Secondary Computer Science syllabus has been revised with a view to bringing out its real spirit and dimension. The constant and remarkable developments in the field of computing as well as the endless opportunities of research in the field of Computer Science and Technology have been included.

In Class XI, we started with the history of computing followed by har dware and software components, computer network and Internet. The major part of the textbook as well as the syllabus established a strong foundation to construct and enhance the problem solving and programming skills of the learner.

The syllabus of Class XII gives thrust on the means of handling complex data involved in problem solving and the development of web applications. The textbook, designed in accordance with the syllabus, begins with some advanced features of C++ programming language like structures and pointers. Web technology is introduced with the theoretical background and proceeds with the design of simple web pages followed by a brief idea about both client-side and server-side scripting. The concept of database and facilities of information retrieval are included. An exclusive section about the advances in computing opens the door to the latest developments in the world of computing discipline. Considering the increase in the use of Internet, an awar eness about Cyber laws is also presented to safeguard against Cyber crimes.

I hope this book will meet all the requirements for stepping to levels of higher education in Computer Science and pave your way to the peak of success.

Wish you all success.

**Dr S. Raveendran Nair**
Director
SCERT, Kerala

# Textbook Development Team

**Joy John**
HSST, St. Joseph's HSS
Thiruvananthapuram.

**Prasanth P. M.**
HSST, St. Joseph's Boys' HSS,
Kozhikode.

**Vinod V.**
HSST, NSS HSS,
Prakkulam, Kallam.

**Mustafa Shamsul Haq K. K.**
HSST, GHSS Kuthuparamba,
Kannur

**A. S. Ismael**
HSST, PJMS GHSS, Kandassankadavu,
Thrissur.

**Rajamohan C.**
HSST, Nava Mukunda HSS, Thirunavaya,
Malappuram.

**Aboobacker P.**
HSST, Govt. GHSS, Chalappuram,
Kozhikode

**Najeeb P. P.**
HSST, Himayathul Islam HSS,
Kozhikode

**Dr Lajish V. L.**
Assistant Professor, Dept. of Computer Science, University of Calicut

**Dr Madhu S. Nair**
Assistant Professor, Dept. of Computer Science, University of Kerala

**Madhu V. T.**
Director, Computer Centre, University of Calicut

**Dr Binu P. Chacko**
Associate Professor, Dept. of Computer Science, Prajyoti Niketan College, Pudukad, Thrissur

**Dr Sushil Kumar R.**
Associate Professor, Dept. of English, D.B. College, Sasthamcotta

**Dr Deepa L. C.**
Assistant Professor, Dept. of English, Govt. Women's College, Trivandrum

**Vinayakumaran Nair N.**
Assistant Commandant, Hi-Tech Cell, Police Head Quarters, Trivandrum

**Dr Kabeer V.**
Asst. Prof & Head, Dept. of Computer Science, Farook College, Kozhikode

**Sudheer Y.**

**Vineeth V.**

**Dr Meena S.**
Research Officer, SCERT

# Contents

**8. Database Management System ........................................241**

8.1 Concept of database
8.2 Components of the DBMS environment
8.3 Data abstraction and data independence
8.4 Users of database
8.5 Relational data model
8.6 Terminologies in RDBMS
8.7 Relational algebra

**9. Structured Query Language ............................................. 267**

9.1 Structured Query Language
9.2 Working on MySQL
9.3 SQL commands
9.4 Creating Tables
9.5 Inserting data into tables
9.6 Retrieving information from tables
9.7 Modifying data in tables
9.8 Changing the structure of a table
9.9 Deleting rows from a table
9.10 Removing table from a database
9.11 Nested queries
9.12 Concept of views

**10. Server Side Scripting Using PHP ........................................313**

10.1 Overview of PHP
10.2 Fundamentals of PHP
10.3 Control structures in PHP
10.4 Arrays in PHP
10.5 Functions in PHP
10.6 Three tier architecture in PHP
10.7 PHP forms
10.8 Connecting PHP to database (MySQL)

**11. Advances in Computing ........................................359**

11.1 Distributed computing paradigms
11.2 Distributed computing
11.3 Artificial Intelligence
11.4 Computational Intelligence

**12. ICT and Society ........................................385**

12.1 ICT services
12.2 Information security

## Icons used in this textbook

[ - ] et us do

Know your progress

6] «a ion box

E Let us practice

Let us conclude

# 8 Database Management System

**FIR**

*After the completion of this chapter, the learner*

- recognises the need for files.
- identifies the major limitations of the conventional file management system.
- lists and explains the different advantages of the Database Management System.
- lists the various components of DBMS and explains their purpose.
- recognises the types of users and their roles in the DBMS environment.
- explains the levels of data abstraction and data independence in DBMS.
- explains relational data model by citing examples.
- uses different terminologies in RDBMS, appropriately.
- applies and evaluates various operations in Relational algebra.

This is an era of information. The survival of organisations in this competitive world largely depends on the need for information obtained with high accuracy and speed. We know that information is obtained through the processing of data. For this, a huge amount of data is to be collected, stored and processed to generate information. Every organisation that we can think of like schools, banks, business organisations, etc. are in need of information. Can you imagine how this huge quantity of data was handled traditionally? In earlier days, this was handled manually with data recorded in books termed as 'book keeping'. It is obvious that large storage space is a hazard and processing of data is laborious. With the advent of computers the data could be stored effectively, but the possibility of duplication, inconsistency, invalidity, etc. remained. This chapter provides an effective mechanism to overcome these limitations. The concept of Database Management System (DBMS) is introduced in this chapter as an effective record keeping system (earlier called book keeping). Various operations are also discussed to retrieve the required and relevant information from the database.

## 8.1 Concept of database

Consider the Single Window System (SWS) of the Department of Higher Secondary Education, maintaining a large collection (say 5,00,000 applications or 19 GB) of data concerning students, courses, schools and grades each year for Class XI admission process. This data is accessed simultaneously by several schools and students. Questions about the allotment of students and schools must be answered quickly, changes made to the data by different schools must be applied consistently, and access to certain parts of the data (e.g., Grades or WGPA) must be restricted.

We can manage the data by storing it in the conventional file management system. But, this approach has many drawbacks:

- We must keep more copies of the same data for different applications. This storage leads to duplication of data.
- There is no mechanism to protect the data from inconsistent changes made by different users accessing the database simultaneously.
- If the data are not properly organised, retrieval of information will be difficult, and time consuming, and there may be chances of inaccuracy in the information..
- There is no way to ensure that data is restored to a consistent state if the system crashes while changes are being made.
- Operating systems provide only a password mechanism for security. This is not sufficiently flexible to enforce security policies in data.
- There is no standardisation on data.

### 8.1.1 Need of database

The drawbacks mentioned above can be overcome by using database. In situations, where huge amount of data is to be mantained and manipulated, conventional file system will not be sufficient. In such a situation we use database. ***Database*** is an organized collection of inter-related data stored together with minimum redundancy, in a manner that makes them accessible for multiple applications. Now we can try to manage the data in SWS by storing it in a database. The software Database Management System (DBMS) is essentially a set of programs which facilitates storage, retrieval and management of database. The primary goal of DBMS is to provide an environment that is both convenient and efficient to use in retrieving and storing database.

### 8.1.2 Advantages of DBMS

Database systems are designed to manage large amounts of data. DBMS involves both the database structures definition for the storage of data and the provision of mechanisms for the manipulation of data. In addition, the database system must ensure the safety of the data stored, against unauthorized access or system failure. If data is to be shared among several users, the system must avoid possible anomalous results. The database management system has a number of advantages over the traditional file management system. They are listed below.

- **Controlling data redundancy**: In file management systems, data may be placed in many files. The storing of the same data in multiple locations (may be in the same file or different files) or duplication of data is known as data redundancy. Redundancy leads to higher cost in storage and data access. Database systems do not maintain redundant data, instead all the data is kept at one place in a centralized manner. All the applications or users that require data refer to the centrally maintained database. Sometimes there can be technical or business reasons for maintaining several copies of the same data. However, redundancy should be carefully controlled in any case.
- **Data consistency**: Data redundancy may lead to data inconsistency; that is, the various copies of the same data show different values in different files. Assume that your class teacher and Principal maintain separate copies of the address list of all students admitted in your class. During periodic address change a few students report to the Principal and a few students report to the changes to the class teacher. After a certain period of time both the address lists become irrelevant and inconsistent since total corrections are not updated in both. By controlling data redundancy, data consistency is obtained. If a data item appears only once, any update to its value has to be performed only once and the updated value is immediately available to all users.
- **Efficient data access**: A DBMS utilises a variety of techniques to store and retrieve data efficiently.
- **Data integrity**: Data integrity refers to the overall completeness, accuracy and consistency of data in the database. This can be indicated by an absence of any alteration in data between two updates of a data record. Data integrity is imposed within a database at its design stage through the use of standard rules and procedures. Data integrity can be maintained through the use of error checking and validation routines.
- **Data security**: The information inside a database is valuable to any company or organization. Therefore it must be kept secure and private. Data security refers to the protection of data against accidental or intentional disclosure or unauthorized destruction or modification by unauthorized persons. The various programs and users may share data in common. But access to specific information can be limited to selected users by setting the access rights. Through

the use of passwords, the information in a database is made available only to authorised persons.

- **Sharing of data**: The data stored in the database can be shared among several users or programs even simultaneously and each may use it for different purposes.
- **Enforcement of standards**: With central control of the database, a Database Administrator (OBA) defines and enforces the necessary standards. Standards can be defined for data formats to facilitate exchange of data between systems. Applicable standards might include naming conventions, display formats, report structures, terminology, documentation standards, update procedures, access rules and so on. This facilitates communication and cooperation among various departments, projects and users within the organization.
- **Crash recovery**: When a system crashes, all or a portion of the data can become unusable. DBMS provides some mechanism to recover data from the crashes. Thus the DBMS protects data from the effects of system failures.

**Know your progress**

1. Storing of the same data in different places is called ______ .
2. Address of a particular student is stored in two ways in a school record; this situation is known s ________.
3. Unauthorised accessing of data can be prevented from using _______.
4. Sharing of data can reduce data redundancy. State whether true or false?
5. Data redundancy will reduce data consistency. State whether true or false?

## 8.2 Components of the DBMS environment

DBMS have several components, each performing very significant tasks in its environment. The components are

- Hardware
- Software
- Data
- Users
- Procedures

**Hardware:** The hardware is the actual computer system used for storage and retrieval of the database. This includes computers (PCs, workstations, servers and supercomputers), storage devices (hard disks, magnetic tapes), network devices (hubs, switches, routers) and other supporting devices for keeping and retrieval of data.

**Software:** The software part consists of the actual DBMS, application programs and utilities. DBMS acts as a bridge between the user and the database. In other words, DBMS is the software that interacts with the users, application programs, and databases. All requests from users for access to the database are handled by the DBMS. Database management system software consists of several software components that handle various tasks such as data definition, data manipulation, data security, data integrity, data recovery and performance optimization. One general function provided by the DBMS is thus the shielding of database from complex hardware-level detail. The DBMS controls the access and helps to maintain the consistency of the data.

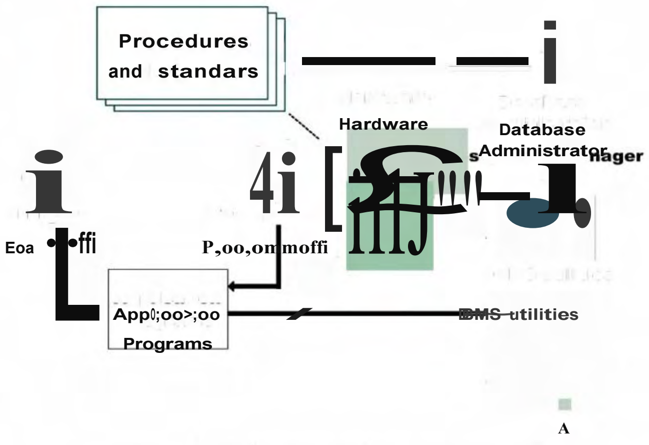


*Fig. 8.1: Database system environment*

Application programs are most commonly used to access data found within the database to generate reports, tabulations, and other information to facilitate decision making. Utilities are the software tools used to help manage the database system. For example, all major DBMS provide graphical user interfaces (GUIs) to help create database structures, control database access, and monitor database operations.

**Data:** It is the most important component of DBMS environment from the end users point of view. The database contains operational data and the meta-data (data about data). The database should contain all the data needed by the organization. The major feature of databases is that the actual data and the programs that uses the data are separated from each other. For effective storage and retrieval of information, data is organized as fields, records and files.

Assume a box containing a collection of cards which stores the Admission Number, Name, Batch, Result, Marks of students in a class. Each card will have the same format but the data written on them is different as in Figure 8.2.

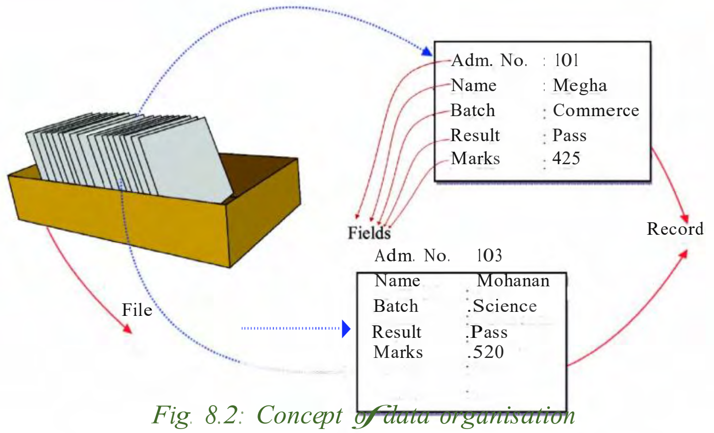


*Fig. 8.2: Concept of data organisation*

***Fields:*** A field is the smallest unit of stored data. Each field consists of data of a specific type. In Figure 8.2 Adm. No., Name, Batch, Result, and Marks are the fields.

***Record:*** A record is a collection of related fields. In the Figure 8.2, each card in the box contains the related fields of a record. For example, the fields 103, Mohanan, Science, Pass, and 520 constitute a record.

***File:*** A file is a collection of all occurrences of same type of records. The box shown in Figure 8.2 may be considered as a file.

**Users:** There are a number of users who can access data on demand using application programs. The users of a database system can be classified depending on the mode of their interactions with DBMS. The different categories of users are Database Administrator (DBA), Application Programmers, Sophisticated users and Naive Users.

**Procedures:** Procedures refer to the instructions and rules that govern the design and use of the database. The users of the system and the person that manages the database require documented procedures on how to use or run the system. These may consist of instruction on how to:

i. log onto the DBMS.

ii. use a particular DBMS facility or application program.

iii. start and stop the DBMS.

iv. make backup copies of the database or handle hardware or software failures

v. reorganise the database across multiple disks, improve performance, or archive data to secondary storage.

**Know your progress**

1. Data about data is called _ _ _ _ .
2. List the name of major components of a database system?
3. Categorize the following components of DBMS environment. Hard disk, Switch, DBA, Payroll system, End user, railway reservation system.

## 8.3 Data abstraction and data independence

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database system users are not computer trained, developers hide the complexity from users through several levels of abstraction. The data in a DBMS is described at three levels of abstraction, as illustrated in Figure 8.3. The database description consists of a structure at each of these three levels of abstraction: the physical level, conceptual level and view level.

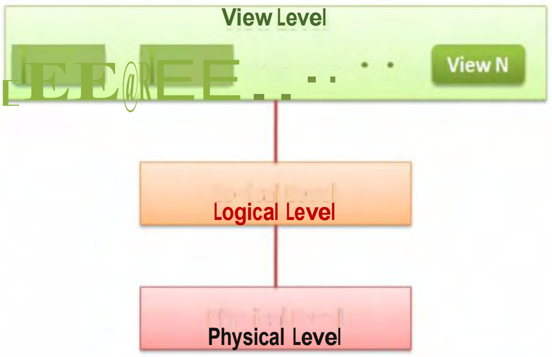


*Fig. 8.3: Levels of abstraction*

### a. Physical level

The lowest level of abstraction describes how data is actually stored on secondary storage devices such as disks and tapes. The physical level describes complex low• level data structures in detail. We must decide what file organisations are to be used to store the relations and create auxiliary data structures, called indexes, to speed up data retrieval operations.

A sample physical schema for the SWS database follows:

- Store all relations as unsorted files of records. (A file in a DBMS is either a collection of records or a collection of data)
- Create indexes on the first column of the files Student, School, and Course.

### b. Logical level

The next-higher level of abstraction describes what data is stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction. Logical level is also referred as conceptual level.

### c. View level

View level is the highest level of database abstraction and is the closest to the users. It is concerned with the way in which individual users view the data. It describes only a part of the entire database. Most of the users of the database are not concerned with all the information that is contained in the database. Instead they need only a part of the database that is relevant to them. This simplifies their interaction with the system. The system may provide many views for the same database. Figure 8.4 shows the three levels of data abstraction for a STUDENT file with fields AdmNo, Name, Batch, Result, Marks.

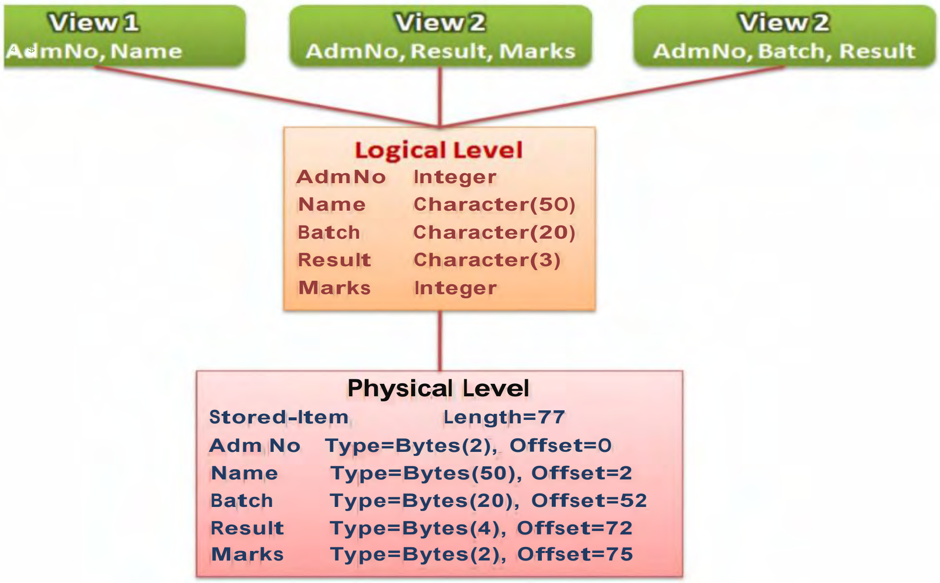


*Fig. 8.4: Example for levels of abstraction*

## 8.3.1 Data independence

Since a database may be viewed through three levels of abstraction, any change in the database structure at a particular level may affect the schema of other levels. The frequent changes made on database should not lead to the redesigning and re•implementation of the database. The ability to modify the schema definition (data structure definition) in one level without affecting the schema definition at the next higher level is called data independence. There are two levels of data independence, physical data independence and logical data independence.

### a. Physical data independence

Physical data independence refers to the ability to modify the schema followed at the physical level without affecting the schema followed at the conceptual level. That is, the application programs remain the same even though the schema at physical level gets modified.

### b. Logical data independence

Logical data independence refers to the ability to modify a conceptual schema without causing any changes in the schema followed at view (external) level. The logical data independence ensures that the application programs remain the same.

It is more difficult to achieve logical data independence than physical data independence because the application programs are heavily dependent on the logical structure of the database.

## 8.4 Users of database

Depending on the degrees of expertise or the mode of the interactions with DBMS the users of a database system can be classified into the following groups:

- Database Administrator (DBA)
- Application Programmers
- Sophisticated Users
- Naive Users

### 8.4.1 Database Administrator

The person who is responsible for the control of the centralized and shared database is the Database Administrator (DBA). The DBA is responsible for many critical tasks such as,

**Design of the conceptual and physical schemas:** The DBA is responsible for interacting with the users of the system to understand what data is to be stored in the DBMS and how it is likely to be used. Based on this knowledge, the DBA must design the conceptual schema and the physical schema.

**Security and authorization:** The DBA is responsible for ensuring authorized access of data. For example, in a school, teachers allow students to find out course details, results of the student and the details as to who teaches a particular subject. At the same time students shall not be permitted to see teachers' salaries or grades of other students. The DBA can enforce this policy by giving permission to students to read only the course view.

**Data availability and recovery from failures:** The DBA must take steps to restore the data to a consistent state when the system fails to complete a transaction or in case of a system crash. The DBMS provides software support for these functions, but the DBA is responsible for implementing procedures to back up the data periodically and maintain logs (special files for storing all activities in a database such as insertion, deletion, updation etc.) of system activity (to facilitate recovery from a crash).

### 8.4.2 Application programmers

Application programmers are computer professionals who interact with the DBMS through application programs. Application programs are programs written in any host language (for example Visual Basic, C, C ++,Java, etc.) and interact with the DBMS through Data Manipulation Language (DML). Application programs should ideally access data through the external schema.

### 8.4.3 Sophisticated users

Sophisticated users include engineers, scientists, business analysts, and others who are thoroughly familiar with the facilities of the DBMS. They interact with the systems through their own queries (a request to a database) to meet their complex requirements.

### 8.4.4 Naive users

Naive users interact with the system by invoking one of the application programs that were written previously. They are not concerned with or even aware of the details of the DBMS. Naive users deal only with the higher level of abstraction. People accessing data over the web, clerical staff in an office, billing clerk in a supermarket or hotels, bank clerk, etc. are examples of some naive users.

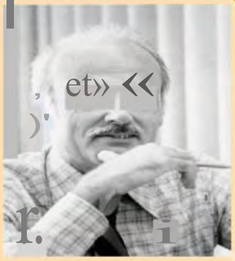
**Know your progress**

1. The person who interacts with the database through query language is called __________
2. The billing clerk in a Supermarket is a __________ user.
3. Who provides data security in a database?
4. Who changes the structure of a database?
5. __________ interacts with the database through the prewritten application program.

## 8.5 Relational data model

The relational data model represents database as a collection of tables called relations, each of which is assigned a unique name. In relational model, both data and the relationships among them are represented in tabular form. This representation enables even beginners to understand the concepts of a database easily.

**Edgar Frank Codd (19 August 1923 - 18 April 2003) was an English computer scientist who invented the** relational model **for** database management. He was born on the Isle of Portland in England. He served as a pilot in the Royal Air Force during the Second World War. In 1948, he joined IBM. He received the Turing Award in 1981. He died of heart failure at his home in Williams Island, Florida, at the age of 79 on 18 April 2003.



Today, a vast majority of database products are based on the relational model and they are known as Relational DataBase Management System (RDBMS). The major advantages of the relational model over the other data models are its simple data representation and the ease with which even complex queries can be expressed. The popular RDBMS are Oracle, Microsoft SQL Server, MySQL, DB2, Informix and Ingress. Most commercial relational database systems offer a query language that includes Structured Query Language (SQL), Query-by-Example (QBE) or Datalog. We shall study the widely used query language SQL in the Chapter 9.

## 8.6 Terminologies in RDBMS

Before discussing the operations on relational databases, let us be familiar with some terminologies associated with RDBMS.

### a. Entity

An entity is a person or a thing in the real world that is distinguishable from others. For example, each student is an entity, and each school can be considered as another entity.

### b. Relation

Relation is a collection of data elements organized in terms of rows and columns. A relation is also called Table. A sample relation named STUDENT is shown in Table 8.1.

STUDENT relation

| AdmNo | Roll | Name | Batch | Marks | Result |
|---|---|---|---|---|---|
| 101 | 24 | Sachin | Science | 480 | EHS |
| 102 | 14 | Rahul | Commerce | 410 | EHS |
| 103 | 4 | Fathima | Humanities | 200 | NHS |
| 104 | 12 | Mahesh | Commerce | 180 | NHS |
| 105 | 24 | Nelson | Humanities | 385 | EHS |
| 106 | 8 | Joseph | Commerce | 350 | EHS |
| 107 | 24 | Shaji | Humanities | 205 | NHS |
| 108 | 2 | Bincy | Science | 300 | EHS |

*Table. 8.1: A sample relation*

### c. Tuple

The rows (records) of a relation are generally referred to as tuples. A row consists of a complete set of values used to represent a particular entity. In Table 8.1, each row in the STUDENT relation represents the result of a particular student.

### d. Attribute

The columns of a relation are called attributes. AdmNo, Roll, Name, Batch, Marks and Result are attributes of the STUDENT relation. The values of each attribute are taken from the range of possible values called domain.

### e. Degree

The number of attributes in a relation determines the degree of a relation. The relation STUDENT has six columns or attributes and therefore the degree of the STUDENT relation is 6.

### f. Cardinality

The number of rows or tuples in a relation is called cardinality of the relation. The relation STUDENT has eight tuples and hence the cardinality of the STUDENT relation is 8.

### g. Domain

A domain is a pool of values from which actual values appearing in a given column are drawn. For example, the domain of the column Batch in the relation STUDENT shown in Table 8.1 is the set of values {Science, Humanities, Commerce}. That is, any one of the values from this set only can appear in the column Batch. Similarly, the set {EHS, NHS} is the domain of the column Results.

## h. Schema

The description or structure of a database is called the database schema, which is specified during database design. In the relational model, the schema for a relation specifies its name, the name of each column, and the type of each column. As an example, student information in a School database may be stored in a relation with the following structure:

```
STUDENT          (Admno  : integer,
                  Roll   : integer,
                  Name   : character(50),
                  Batch  : character(20),
                  Marks  : decimal,
                  Result : character(4))
```

## i. Instance

An instance of a relation is a set of tuples in which each tuple has the same number of fields as the relational schema. The preceding schema says that each row in the STUDENT relation has six columns, with column names and types as indicated. An example for instance of the STUDENT relation is shown in Table 8.1.

**Know your progress**

1. Organization of data in terms of rows and columns is called
2. ____________ in a table gives the complete data of a particular ~~entity.~~
3. Number of rows in a relation is called ______
4. Number ot ____________ In a relation is called degree of the relation. ____________
5. In the relational model, data is organised as __ ________ .

## 8.6.1 Keys

A relation is defined as a set of tuples. All the tuples in a relation must be distinct. That is, no two tuples can have the same combination of values for all their attributes. Therefore there should be a way to identify a tuple in a relation. The concept of a key allows us to make such distinctions. A key is an attribute or a collection of attributes in a relation that uniquely distinguishes each tuple from other tuples in a given relation. If a key consists of more than one attribute then it is called a composite key. In the extreme, the entire tuple is the key since each tuple in the relation is guaranteed to be unique. However, we are interested in smaller keys, if they exist, for a number of practical reasons.

## a. Candidate key

A candidate key is the minimal set of attributes that uniquely identifies a row in a relation. In the STUDENT relation of Table 8.1, AdmNo can uniquely identify a row. Therefore it can be considered as a candidate key. There may be more than one candidate key in a relation. Also, a candidate key need not be just one single attribute. It can be a composite key. For example, a combination of Roll, Batch and Marks can also be used to identify a particular student. Therefore, Roll + Batch + Year can be considered as another candidate key of STUDENT relation.

## b. Primary key

A primary key is one of the candidate keys chosen to be the unique identifier for that table by the database designer. A primary key is a set of one or more attributes that can uniquely identify tuples within the relation. As it uniquely identifies each entity, it cannot contain null value and duplicate value.

Candidate keys are considered as candidates for primary key position. From the candidate keys the one with the least number of attributes may be selected as primary key. In our example (STUDENT relation), the attribute AdmNo can be used as the primary key. That is, no two students in the STUDENT relation can have the same AdmN o. In Table 8.1, we can see unique values in the column Name, but in real case scenario, more than one student can have the same name.

## c. Alternate key

A candidate key that is not the primary key is called an alternate key. In the case of two or more candidate keys, only one of them can serve as the primary key. The rest of them are alternate keys. In our example the combination of Roll+ Batch + Year is the alternate key since AdmNo is taken as the primary key.

## d. Foreign key

A key in a table can be called foreign key if it is a primary key in another table. Since a foreign key can be used to link two or more tables it is also called a reference key. Suppose we have used Batch code instead of Batch name as shown in Table 8.2 and we have a relation BATCH as shown in Table 8.3. It is clear that BatchCode is the primary key in relation Batch, but it is used in STUDENT table as a non-key attribute. So, BatchCode is referred to as a foreign key with respect to STUDENT relation.

STUDENT relation

| AdmNo | Roll | Name | BatchCode | Marks | Result |
|---|---|---|---|---|---|
| 101 | 24 | Sachin | S2 | 480 | EHS |
| 102 | 14 | Rahul | C2 | 410 | EHS |
| 103 | 4 | Fathima | H2 | 200 | NHS |
| 104 | 12 | Mahesh | C2 | 180 | NHS |
| 105 | 24 | Nelson | H2 | 385 | EHS |
| 106 | 8 | Joseph | C2 | 350 | EHS |
| 107 | 24 | Shaji | H2 | 205 | NHS |
| 108 | 2 | Bincy | S2 | 300 | EHS |

*Table 8.2: Modified STUDENT relation*

BATCH relation

| BatchCode | BatchName | Strength |
|---|---|---|
| s1 | Science | 150 |
| S2 | Science | 150 |
| C1 | Commerce | 100 |
| C2 | Commerce | 100 |
| HI | Humanities | 100 |
| H2 | Humanities | 100 |

*Table 8.3: Instance of Batch relation*

**Know your progress**

1. The minimal set of attributes that uniquely identifies a row in a relation is ________
2. In a table Employee Emp_code, Pan_no are the candidate keys. If Emp_code is taken as the primary key then Pan_no is ________ key.
3. How many primary keys are possible in a relation?
4. If a key consists of more than one attributes then it is called

## 8.7 Relational algebra

We have discussed the features of relational model, which provides facilities for craeting a database. Once the database is designed and data is stored, the required information is to be retrieved. A variety of operations are provided by RDBMS. The collection of operations that is used to manipulate the entire relations of a database is known as relational algebra. These operations are performed with the

help of a special language associated with the relational model, called query language. We will learn this language in the next chapter to perform these operations. The operations involved in relational algebra take one or two relations as input and produces a new relation as the result The fundamental operations in relational algebra are SELECT, PROJECT, UNION, INTERSECTION, SET DIFFERENCE, CARTESIAN PRODUCT, etc. The SELECT and PROJECT operations are unary operations because they operate on one relation. The remaining operations are binary operations as they operate on pairs of relations.

### 8.7.1 SELECT operation

SELECT operation is used to select rows from a relation that satisfies a given predicate. The predicate is a user defined condition to select rows of user's choice. This operation is denoted using lower case letter sigma (σ). 'The general format of select is as follows:

$$\sigma_{condition} \text{ (Relation)}$$

The result of SELECT operation is another relation containing all the rows satisfying the given predicate (or conditions). The relational algebra uses various comparison operators < (less than), <= (less than or equal to), > (greater than), >= (greater than or equal to), = (equal to) and < > (not equal to) to set up simple conditions, and logical operators ∨ (OR), ∧ (AND) and ! (NOT) to construct composite conditions.

To illustrate the SELECT operation, consider the relation STUDENT given in Table 8.1. The following examples show how SELECT operations are expressed in relational algebra and what output they produce.

**Example 8.1:** To select all the students who are eligible for higher studies.

$$\sigma_{Result="EHS"} \text{ (STUDENT)}$$

The output of this operation is a relation as shown in Table 8.4.

| AdmNo | Roll | Name | Batch | Marks | Result |
|---|---|---|---|---|---|
| 101 | 24 | Sachin | Science | 480 | EHS |
| 102 | 14 | Rahul | Commerce | 410 | EHS |
| 105 | 24 | Nelson | Humanities | 385 | EHS |
| 106 | 8 | Joseph | Commerce | 350 | EHS |
| 108 | 2 | Bincy | Science | 300 | EHS |

*Table 8.4: Output of Example 8.1*

**Example 8.2:** To select all the students in the Commerce batch who are failed.

σ Result="NHS" ∧ Batch="Commerce" (STUDENT)

The result of this operation is a relation as shown in Table 8.5

| AdmNo | Roll | Name | Batch | Marks | Result |
|---|---|---|---|---|---|
| 104 | 12 | Mahesh | Commerce | 180 | NHS |

*Table 8.5: Output of Example 8.2*

**Example 8.3:** To select all the students in the batch Science or Commerce.

σ Batch="Science" ∨ Batch="Commerce" (STUDENT)

Table 8.6 shows the output this operation.

| AdmNo | Roll | Name | Batch | Marks | Result |
|---|---|---|---|---|---|
| 101 | 24 | Sachin | Science | 480 | EHS |
| 102 | 14 | Rahul | Commerce | 410 | EHS |
| 104 | 12 | Mahesh | Commerce | 180 | NHS |
| 106 | 8 | Joseph | Commerce | 350 | EHS |
| 108 | 2 | Bincy | Science | 300 | EHS |

*Table 8.6: Output of Example 8.3*

## 8.7.2 PROJECT operation

The PROJECT operation selects certain attributes from the table and forms a new relation. If the user is interested in selecting the values of a few attributes, rather than all the attributes of the relation, then use PROJECT operation. It is denoted by lower case letter $\pi$. The general format of project operation is as follows:

$\pi$ A1, A2, ...., An (Relation)

Here A1, A2, ...., An refer to the various attributes that would make up the relation specified.

**Example 8.4:** Select Name, Result and Marks attributes in STUDENT relation.

$\pi$ Name, Marks, Result (STUDENT)

The output of this operation is given in Table 8.7.

It is possible to combine the SELECT and PROJECT operations into a single statement. The illustration of this is shown in Examples 8.5 and 8.6.

| Name | Marks | Result |
|---|---|---|
| Sachin | 480 | EHS |
| Rahul | 410 | EHS |
| Fathima | 200 | NHS |
| Mahesh | 180 | NHS |
| Nelson | 385 | EHS |
| Joseph | 350 | EHS |
| Shaji | 205 | NHS |
| Bincy | 300 | EHS |

*Table 8.7: Output relation of Example 8.4*

**Example 8.5:** To select admission number and name of students who are Eligible for Higher Studies.

mo_(G%int. (STUDENT))

The resultant relation of this operation is given in Table 8.8. Compare it with Table 8.4 for the verfication of correctness of the result.

| AdmNo | Name |
|---|---|
| 101 | Sachin |
| 102 | Rahul |
| 105 | Nelson |
| 106 | Joseph |
| 108 | Bincy |

*Table 8.8: Output relation of Example 8.5*

**Example 8.6:** To select name and marks of those students in the Humanities batch who are Not eligible for Higher Studies.

ame, Marks•(O res a-ns A\ » atc a" - umanities - (STUDENT))

The result of this nested operation is shown in Table 8.9.

| Name | Marks |
|---|---|
| Fathima | 200 |
| Shaji | 205 |

*Table 8.9: Output relation of Example 8.6*

## 8.7.3 UNION operation

UNION operation is a binary operation and it returns a relation containing all tuples appearing in either or both of the two specified relations. It is denoted by U . The two relations must be union-compatible, and the schema of the result is defined to be identical to the schema of the first relation. If two relations are union-compatible, then they have the same number of attributes, and corresponding attributes, taken in order from left to right, have the same domain. Note that attribute names are not used in defining union-compatibility.

Consider two relations ARTS and SPORTS given in Tables 8.10 and 8.11 containing the details of students who participate in the arts festival and the sports meet of a school, respectively. Both the relations ARTS and SPORTS consist of AdmNo, Name and BatchCode as fields. It is clear that that these two relations are union

**ARTS relation**

| AdmNo | Name | BatchCode |
|---|---|---|
| 101 | Sachin | S2 |
| 103 | Fathima | H2 |
| 106 | Joseph | C2 |
| 110 | Nikitha | S1 |
| 132 | Vivek | C1 |
| 154 | Nevin | C1 |

**SPORTS relation**

| AdmNo | Name | BatchCode |
|---|---|---|
| 102 | Rahul | C2 |
| 103 | Fathima | H2 |
| 105 | Nelson | H2 |
| 106 | Joseph | C2 |
| 108 | Bincy | S2 |
| 132 | Vivek | C1 |

| | | |
|---|---|---|
| 164 | Rachana | S1 |

*Table 8.10: instance of ARTS relation*

*Table 8.11: instance of SPORTS relation*

$\pi$ a

M

compatible. That is, the two relations have the same number of attributes and the type of the corresponding attributes are also the same.

Relation ARTS U SPORTS returns the details of the students participated in arts or sports or both. That is, the expression ARTS U SPORTS returns a table as shown in Table 8.12. This table consists of the records belonging to the tables ARTS or SPORTS or both, eliminating the duplication. In Table 8.12, we can see that records of students with admission numbers 103, 106, and 132 appear only once in the relation.

| AdmNo | Name | BatchCode |
|---|---|---|
| 101 | Sachin | S2 |
| 103 | Fathima | H2 |
| 106 | Joseph | C2 |
| 110 | Nikitha | s1 |
| 132 | Vivek | C1 |
| 154 | Nevin | C1 |
| 102 | Rahul | C2 |
| 105 | Nelson | H2 |
| 108 | Bincy | S2 |
| 164 | Rachana | s1 |

*Table 8.12: Relation of ARTS U SPORTS*

## 8.7.4 INTERSECTION operation

INTERSECTION operation is also a binary operation and it returns a relation containing the tuples appearing in both of the two specified relations. It is denoted by ∩. The two relations must be union-compatible, and the schema of the result is defined to be identical to the schema of the first relation.

If we apply INTERSECT operation on the relations in Tables 8.10 and 8.11, the expression ARTS ∩ SPORTS returns the details of the students participated in both arts and sports. That is ARTS ∩ SPORTS returns a table consisting of rows common to ARTS and SPORTS as shown in Table 8.13.

| AdmNo | Name | BatchCode |
|---|---|---|
| 103 | Fathima | H2 |
| 106 | Joseph | C2 |
| 132 | Vivek | C1 |

*Table 8.13: Relation of ARTS ∩ SPORTS*

## 8.7.5 SET DIFFERENCE operation

SET DIFFERENCE operation is also a binary operation and it returns a relation containing the tuples appearing in the first relation but not in the second relation. It is denoted by – (minus). The two relations must be union-compatible, and the schema of the result is defined to be identical to the schema of the first relation.

The result of SET DIFFERENCE operation ARTS – SPORTS on Tables 8.10 and 8.11 returns the details of the students participated in arts but not in sports. That is, the resultant table will contain the rows appearing in relation ARTS but not in relation SPORTS as given in Table 8.14.

Relation SPORTS – ARTS returns the details of the students participated in sports but not in arts as shown in Table 8.15.

| AdmNo | Name | BatchCode |
|---|---|---|
| 101 | Sachin | S2 |
| 110 | Nikitha | S1 |
| 154 | Nevin | C1 |

*Table. 8.14: Relation of ARTS - SPORTS*

Union and Intersection operations are commutative, that is the order of relation is not important. For example, the result of ARTS U SPORTS and SPORTS U ARTS are same. Also the result of ARTSQ SPORTS and SPORTS⊂ARTS are the same. But Set Difference operation is not commutative, that is the order of relation is important. For example, the result of ARTS – SPORTS and SPORTS – ARTS are not same (refer to Tables 8.14 and 8.15).

| AdmNo | Name | BatchCode |
|---|---|---|
| 101 | Rahul | C2 |
| 105 | Nelson | H2 |
| 108 | Bincy | S2 |
| 164 | Rachana | S1 |

*Table. 8.15: Relation of SPORTS - ARTS*

## 8.7.6 CARTESIAN PRODUCT operation

CARTESIAN PRODUCT returns a relation consisting of all possible combinations of tuples from two relations. It is a binary operation on relations, which has a degree (number of attributes) equal to the sum of the degrees of the two relations operated upon. The cardinality (number of tuples) of the new relation is the product of the number of tuples of the two relations operated upon. CARTESIAN PRODUCT is denoted by x (cross). It is also called CROSS PRODUCT. All the tuples of the first relation are concatenated with tuples of the second relation to form tuples of the new relation.

Let us consider a relation TEACHER as shown in Table 8.16, which contains the details of teachers in the school. We can use this relation to perform cartesian production operation with STUDENT relation in Table 8.2. 'The output of the operation STUDENT x TEACHER is shown in Table 8.17. This table shows that each record of STUDENT relation is concatenated with the rows in TEACHER relation.

**TEACHER relation**

| TeacherId | Name | Dept |
|---|---|---|
| 1001 | Viswesaran | English |
| 1002 | Meenakshi | Computer |

*Table 8.16: Instance of TEACHER relation*

| Adm No | Roll | Name | Batch Code | Marks | Result | TeacherId | Name | Dept |
|---|---|---|---|---|---|---|---|---|
| 101 | 24 | Sachin | S2 | 480 | EHS | 1001 | Viswesaran | English |
| 101 | 24 | Sachin | S2 | 480 | EHS | 1002 | Meenakshi | Computer |
| 102 | 14 | Rahul | C2 | 410 | EHS | 1001 | Viswesaran | English |
| 102 | 14 | Rahul | C2 | 410 | EHS | 1002 | Meenakshi | Computer |
| 103 | 4 | Fathima | H2 | 200 | NHS | 1001 | Viswesaran | English |
| 103 | 4 | Fathima | H2 | 200 | NHS | 1002 | Meenakshi | Computer |
| 104 | 12 | Mahesh | C2 | 180 | NHS | 1001 | Viswesaran | English |
| 104 | 12 | Mahesh | C2 | 180 | NHS | 1002 | Meenakshi | Computer |
| 105 | 24 | Nelson | H2 | 385 | EHS | 1001 | Viswesaran | English |
| 105 | 24 | Nelson | H2 | 385 | EHS | 1002 | Meenakshi | Computer |
| 106 | 8 | Joseph | C2 | 350 | EHS | 1001 | Viswesaran | English |
| 106 | 8 | Joseph | C2 | 350 | EHS | 1002 | Meenakshi | Computer |
| 107 | 24 | Shaji | H2 | 205 | NHS | 1001 | Viswesaran | English |
| 107 | 24 | Shaji | H2 | 205 | NHS | 1002 | Meenakshi | Computer |
| 108 | 2 | Bincy | S2 | 300 | EHS | 1001 | Viswesaran | English |
| 108 | 2 | Bincy | S2 | 300 | EHS | 1002 | Meenakshi | Computer |

*Table 8.17: Result of STUDENT X TEACHER operation*

**A Database model defines the logical design of data. The model describes the relationships between different parts of the data. The different models used in database design are Hierarchical Model, Network Model, Relational Model and Object-Oriented Model. Hierarchical structures were widely used in the early mainframe database management systems, such as the Information Management System (IMS) by IBM. Popular DBMS product in Network model were Cincom Systems' Total and Cullinet's IDMS.**

## us tonclude

We have discussed the basic concepts of DBMS and its components. The advantages of database over traditional file system have been detailed. A brief idea about various terminologies associated with database is presented in the context of relational data model. Once the data is organised systematically in database, the operations provided by relational algebra to generate required information are experienced with the help of sample relations. A good understanding about the concepts introduced by this chapter is essential to learn the next chapter effectively. In that chapter we will discuss how database is created and information is retrieved using a query language.

## Let us assess

1. Who is responsible for managing and controlling the activities associated with the database?

   a. Database administrator  b. Programmer

   c. Naive user  d. End user

2. In the relational model, cardinality is the

   a. number of tuples  b. number of attributes

   c. number of tables  d. number of constraints

3. Cartesian product in relational algebra is

   a. a Unary operator  b. a Binary operator

   c. a Ternary operator  d.notdefined

4. Abstraction of the database can be viewed as

   a. two levels  b. four levels

   c. three levels  d. one level

5. In a relational model, relations are termed as

   a. tuples  b. attributes

   c. tables  d. rows

6. In the abstraction of a database system the external level is the

   a. physical level  b. logical level

   c. conceptual level  d. view level

7. Related fields in a database are grouped to form a

   a. data file  b. data record

   c. menu  d.bank

8. A relational database developer refers to a record as

   a. criteria  b. relation

   c. tuple  d. attribute

9. An advantage of the database management approach is

   a. data is dependent on programs

   b. data redundancy increases

c. data is integrated and can be accessed by multiple programs

d. none of the above

10. Dataindependencemeans
    a. data is defined separately and not included in programs
    b. programs are not dependent on the physical attributes of data
    c. programs are not dependent on the logical attributes of data
    d. both (b) and (c)
11. Key to represent relationship between tables is called
    a. primary key  b. candidate Key
    c. foreign Key  d. alternate Key
12. Which of the following operations is used if we are interested only in certain columns of a table?
    a. PROJECTION  b. SELECTION
    c. UNION  d. SELECT
13. Which of the following operations need the participating relations to be union compatible?
    a. UNION  b. INTERSECTION
    c. SET DIFFERENCE  d. All of the above
14. Which database level is closest to the users?
    a. External  b. Internal
    c. Physical  d. Conceptual
15. The result of the UNION operation between R1 and R2 is a relation that includes
    a. all the tuples of R1
    b. all the tuples of R2
    c. all the tuples of R1 and R2
    d. all the tuples of R1 and R2 which have common columns
16. A file manipulation command that extracts some of the records from a file is called
    a.SELECT  b.PROJECT
    c.JOIN  d. PRODUCT
17. An instance of relational schema R (A, B, C) has distinct values of A including NULL values. Which one of the following is true?
    a. A is a candidate key  b. A is not a candidate key
    c. A is a primary Key  d. Both (a) and (c)

18. How many distinct tuples are there in a relation instance with cardinality 22?

    a. 22 b. 11

    c. 1 d. None

19. A set of possible data values is called

    a. Attribute b. Degree

    c. Tuple d. Domain

20. Why should you choose a database system instead of simply storing data in conventional files?

21. Explain the different levels of data abstraction in DBMS?

22. How are schema layers related to the concepts of logical and physical data independence?

23. Consider the instance of the EMPLOYEE relation shown in the following table. Identify the attributes, degree, cardinality and domain of Name and Emp_code.

| Emp_Code | Name | Department | Designation | Salary |
|---|---|---|---|---|
| 1000 | Sudheesh | Purchase | Manager | 25000 |
| 1001 | Dhanya | Sales | Manager | 25000 |
| 1002 | Fathima | Marketing | Clerk | 12000 |
| 1003 | Shajan | Sales | Clerk | 13000 |

24. Identify primary key, candidate keys and alternate keys in the instance of EMPLOYEE relation in Question 23.

25. Consider the instance of the STUDENT relation shown in the following table. Assume Reg_no as the primary key.

    a. Identify the candidate keys and alternate keys in the STUDENT relation.

    b. How are the primary key and the candidate key related?

| Reg_no | Name | Batch | Result | Marks |
|---|---|---|---|---|
| 101 | Sachin | Science | Pass | 480 |
| 103 | Fathima | Humanities | Fail | 200 |
| 106 | Joseph | Commerce | Pass | 350 |
| 108 | Bincy | Science | Pass | 300 |

26. What is a database? Describe the advantages and disadvantages of using DBMS.
27. What is data independence? Explain the difference between physical and logical data independence.
28. Enforcement of standard is an essential feature of DBMS. How are these standards applicable in a database?
29. Cardinality of a table Tl is 10 and of table T2 is 8 and the two relations are union compatible. If the cardinality of result Tl UT2is 13, then what is the cardinality of Tl T2? Justify your answer.
30. Cardinality of a table Tl is 10 and of table T2 is 8 and the two relations are union compatible
    a. What will be the maximum possible cardinality of Tl U T2?
    b. What will be the minimum possible cardinality of Tl ∩ T2?
31. Consider the relations, City (city_name, state) and Hotel (name, address, city_name). Answer the following queries in relational algebra
    a. Find the names and address of hotels in Kochi.
    b. List the details of cities in Kerala state.
    c. List the names of the hotels in Thrissur.
    d. Find the names of different hotels.
    e. Find the names of hotels in Kozhikode or Munnar.
32. Using the instance of the EMPLOYEE relation shown in question 23, write the result of the following relational algebra expressions.
    a. peace-sa· (EMPLOYEE).
    b. (J" salary>20000 ∧ Department="Sales" (EMPLOYEE).
    c. O5 – 2000ov Dearest=s.. (EMPLOYEE).
    d. %0.a. (EMPLOYEE).
    e. π %«cs – o.%oar=w% – (EMPLOYEE)).

    £ π %e.0erament( o%stator="ea"shy > 200O(EMPLOYEE)).
33. Consider the instance of the BORROWER and DEPOSITOR relations shown in following figure which stores the details of customers in a Bank. Answer the following queries in relational algebra.
    a. Display the details of the customers who are either a depositor or a borrower.
    b. Display the name of customers who are both a depositor and a borrower.

c. Display the details of the customers who are depositors but not borrowers.

d. Display the name and amount of customer who is a borrower but not depositor.

**BORROWER**

| Acc_No | Name | Amount |
|---|---|---|
| AC123 | Albin | 50000 |
| AC103 | Rasheeda | 25000 |
| AC106 | Vishnu | 25000 |
| AC108 | Aiswarya | 30000 |

**DEPOSITOR**

| Acc_No | Name | Amount |
|---|---|---|
| AC123 | Albin | 500 |
| AC105 | Shabana | 25000 |
| AC116 | Vishnu | 125000 |
| AC108 | Aiswarya | 3000 |

34. Consider the instance of the CUSTOMER and BRANCH relations shown in the following table. Write the Cartesian Product of the two relations.

**CUSTOMER**

| Acc_No | Name | Branch_ID | Amount |
|---|---|---|---|
| AC123 | Albin | B1001 | 50000 |
| AC103 | Rasheeda | B1001 | 25000 |
| AC106 | Vishnu | B1001 | 25000 |
| AC108 | Aiswarya | B1077 | 30000 |

**BRANCH**

| Branch_ID | Name |
|---|---|
| B1001 | Kochi |
| B1002 | Guruvayur |
| B1077 | Idukki |

# 9 Structured Query Language

**EIRE**

*After the completion of this chapter, the learner*

- recognises the importance and features of Structured Query Language.
- explains the components of SQL.
- distinguishes the features of DDL, DML and DCL commands.
- identifies the characteristics of MySQL.
- lists different data types and their features.
- explains the effect of different constraints in SQL.
- performs operations using DDL commands like CREATE, ALTER, DROP.
- uses DML commands like SELECT, INSERT, UPDATE, DELETE for data manipulation.
- identifies various clauses associated with SQL commands and their purpose.
- uses operators for setting different conditions.
- lists different aggregate functions and explains their usage.
- constructs nested queries for information retrieval.

In the last chapter, we discussed the Relational Database Management System (RDBMS). We know that relational database is a set of related data stored in tables called relations. We also have a basic idea about relational algebra, which deals with various operations performed on relations. Now, we need more clarity on these operations which include creating a table, inserting data into a table, manipulating the data stored in a table and deleting data from a table, modifying the structure of a table, removing a table, etc. on a relational database. This chapter introduces a language called Structured Query Language (SQL) for these operations. Most of the relational database management systems like MySQL, Oracle, Sybase, Informix, Postgres, SQL Server and MS Access use SQL as standard database language. We use one of the most popular open source RDBMS, like MySQL, to implement Structured Query Language.

## 9.1 Structured Query Language

Structured Query Language (SQL) is a language designed for managing data in relational database management system (RDBMS). SQL provides an easy and efficient way to interact with relational databases. There are numerous versions of SQL. The original version was developed in the 1970's by Donald D. Chamberlin and Raymond F. Boyce at IBM's San Jose Research Laboratory (now the Almanden Research Centre). This language was originally called Structured English Query Language (Sequel) and later its name was changed to SQL. In 1986, American National Standard Institute (ANSI) published an SQL standard.

As we know a relational database system is a structured collection of tables (relations) and the data is stored in these tables. Tables are uniquely identified by their names and are comprised of columns and rows. A column (field) in a table represents a particular type of information. In a table, each row represents a collection of related data. We know that rows in a table are known as tuples (or records) and columns are known as attributes.

Examine the following table, named "Student" (refer to Table 9.1) and answer the questions given below for recollecting the basic terminologies related to database.

| Adm_no | Name | Gender | Date_Birth | Income | Course |
|---|---|---|---|---|---|
| 1001 | Alok | M | 2/10/1998 | 24000 | Science |
| 1002 | Nike | M | 26/11/1998 | 35000 | Science |
| 1003 | Bharath | M | 1/1/1999 | 45000 | Commerce |
| 1004 | Virat | M | 5/12/1998 | 22000 | Science |
| 1005 | Meera | F | 15/8/1998 |  | Science |
| 1006 | Divakar | M | 21/2/1998 |  | Humanities |

*Table 9.1: Student table*

i. The cardinality of the table is ________ .

ii. The degree of the table is ________ .

iii. List out the different tuples in the table.

iv. List out the different attributes in the table.

v. What are the values in the domain of the attribute 'Course'?

SQL is a powerful tool for implementing RDBMS. It provides facilities to create a table, insert data into a table, retrieve information from a table, modify data in the table, delete the existing data from a table, modify the structure of a table, remove a table from a database, etc.

### 9.1.1 Features of SQL

Structured Query Language is an ANSI/ISO standard language for writing database queries. A query is a request to a database. It can perform all the relational operations mentioned earlier. SQL is effective in framing queries because of the following features:

- SQL is a relational database language, not a programming language like C, C++.
- Itis simple, flexible and powerful.
- It provides commands to create and modify tables, insert data into tables, manipulate data in the tables etc.
- It gives guidelines to major popular RDBMS like Oracle, SQL Server, MySQL, MS Access, Sybase, Informix and Postgres to perform database operations.
- SQL is a non-procedural language since it describes what data to retrieve, delete, or insert, rather than how to perform the operation.
- As part of ensuring data security, SQL provides facility to add or remove different types of access permissions to users on databases or tables.
- It provides the concept of views (This concept will be discussed later in this chapter).

### 9.1.2 Components of SQL

SQL has three components, namely Data Definition Language (DDL), Data Manipulation language (DML) and Data Control Language (DCL). Let us discuss these components and their roles in developing RDBMS.

#### Data Definition Language

Consider Table 9.1 (Student table). How can we create such a table? Is it possible to add a new column to this table? How can we remove a table from the database? The Data Definition Language (DDL) will give solutions to all these questions.

DDL is a component of SQL that provides commands to deal with the schema (structure) definition of the RDBMS. The ***DDL commands*** are used to create, modify and remove the database objects such as tables, views and keys. The common DDL commands are CREATE, ALTER, and DROP.

#### Data Manipulation Language

In Table 9.1, we can see several tuples (or rows or records). How are these tuples inserted into the table? Suppose the monthly family income of a particular student is to be modified. Is it possible? How can we delete the record of a student from the table? The Data Manipulation Language (DML) provides commands for these types of manipulations.

DML is a component of SQL that enhances efficient user interaction with the database system by providing a set of commands. ***DML*** permits users to insert data into tables, retrieve existing data, delete data from tables and modify the stored data. The common DML commands are SELECT, INSERT, UPDATE and DELETE.

### Data Control Language

Data Control Language (DCL) is used to control access to the database, which is very essential to a database system with respect to security concerns. ***DCL*** includes commands that control a database, including administering privileges and committing data. The commands GRANT and REVOKE are used as a part of DCL.

| | | |
|---|---|---|
| **GRANT** | : | Allows access privileges to the users to the database. |
| **REVOKE** | : | Withdraws user's access privileges given by using GRANT command. |

### Know your progress

1. SQL stands for ____________.
2. Which are the three components of SQL?
3. SQL can be used to:
   a. create database structures only. b. query database data only.
   c. modify database data only. d. All of these can be done by SQL.
4. SQL is:
   a. a programming language. b. an operating system.
   c. a data sublanguage. d. a DBMS.
5. Which of the following is not an RDBMS package?
   a. ORACLE b. SQL SERVER c. MySQL d. HTML

## 9.2 Working on MySQL

The American National Standards Institute (ANSI) in 1986, and the International Organization for Standardization (ISO) in 1987, standardised SQL. Since 1986, the SQL standard has been evolving to include a larger set of features. The standard has been revised several times and several versions exist. SQL:2011 is the seventh revision of the ISO and ANSI standard for the SQL database query language. It was formally adopted in December 2011. Despite the existence of such standards, the different database software packages provide their own versions of the standard

ANSI SQL. Therefore, most SQL codes are not completely portable among different database software without adjustments. In this chapter, SQL will be discussed using open source database software MySQL.

MySQL is a free, fast, easy-to-use RDBMS, used for many applications. It is developed, marketed, and supported by MySQL AB, which is a Swedish company. MySQL is becoming very popular for many reasons:

- MySQL is released under an open-source license. So it is customizable.
- It provides high security to the database.
- It is portable as it works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works rapidly and effectively even with large volume of data.
- It is highly compatible with PHP, one of the popular languages for web development.

MySQL was developed by Michael "Monty" Widenius and David Axmark in 1995. It was originally owned by a Swedish company called MySQL AB, and was bought over by Sun Microsystems in 2008. Sun Microsystems was acquired by Oracle in 2010.

MySQL is often deployed in a Linux-Apache-MySQL-PHP (LAMP), Windows-Apache-MySQL-PHP (WAMP), or Mac-Apache-MySQL-PHP (MAMP) environment. All components in LAMP are free and open-source, inclusive of the Operating System. The official site for MySQL is www.mysql.com. The reference for MySQL is the "MySQL Reference Manual", available at http://dev.mysql.com/doc

## 9.2.1 Opening MySQL

We can work on MySQL by giving commands at the `mysql>` prompt. In Ubuntu Linux, we have to open the Terminal window using the following command sequence to get this prompt:

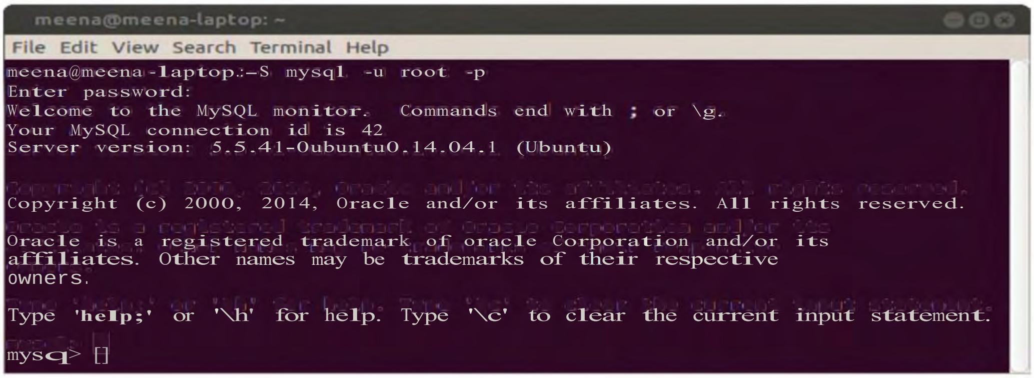
Applications ➔ Accessories -> Terminal

In the Terminal window we give the following command to start MySQL:

```
mysql -u root -p
```

MySQL in Windows OS can be opened by proceeding as follows:

Start ➔ Programs -» MySQL -» MySQL Server (version number) -» MySQL command line client



```
meena@meena-laptop:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 42
Server version: 5.5.41-0ubuntu0.14.04.1 (Ubuntu)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> []
```

*Fig 9.1: MySQL prompt at Ubuntu Linux Terminal window*

When we open MySQL, it may ask for the password for verification. (Here we should use the same password that we entered during the installation process). After the password verification, we will get the prompt of MySQL as shown **in** Figure 9.1.

**SQL is not case sensitive. That is, commands can be given in the upper or the lower case or even in a mix. But hereafter, we will use some styles to distinguish SQL commands and keywords from other texts. Commands and keywords will be specified in the upper case letters, whereas user-defined words such as table name, column name etc. will be in the lower case. Commands and outputs (or responses) can be stored in a text file after using the command `tee`. For example, `tee E:\outputs.txt` will create a file outputs.txt in E: drive to store whatever appears in the screen after the execution of this command. In this chapter the outputs stored in this file will be presented as figures.**

The prompt gives us the message that MySQL is ready to accept any query from the user. Now we can input our queries at this prompt.

To exit from MySQL, give the command **QUIT** or **EXIT** at the prompt as:

```
mysql> EXIT;
```

## 9.2.2 Creating a database

We need to create a database before we work on the data. The database is the container in which we store the tables. To create a database in MySQL, we use the **CREATE DATABASE** command. The syntax is as follows:

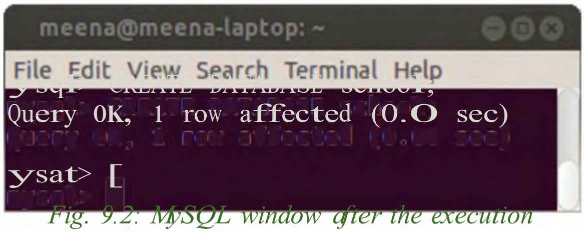
```
CREATE DATABASE <database_name>;
```

While creating a database, the following points are to be remembered:

- The `<database_name>` in the syntax indicates the name of the database that we want to create. It is recommended that the database name should be as meaningful and descriptive as possible.

- The `<database_name>` should be unique. We cannot have two databases with the same name in a MySQL database server.

Let us start our database operations with the creation of a new database called "**school**". Figure 9.2 shows the screen shot after the execution of the command. We can see the MySQL command prompt and the command in the first line. The second line is the message returned by MySQL as a response to the command being executed. From the message, itis clear that a new database with the name *school* has been created successfully. *(Note that here onwards we will avoid such screen shots as figures, instead the command requiredfor the specfied operation will be presented in a separatefont.)*



```
ysql> CREATE DATABASE school;
Query 0K, 1 row affected (0.O sec)

ysat> [
```

*Fig. 9.2: MySQL window after the execution of a command*

## 9.2.3 Opening database

To perform operations on a database, we have to open it explicitly. When we open a database, it becomes the active database in the MySQL server. MySQL gives a command **USE** to open a database. The syntax is:

```
USE <database_name>;
```

Let us open the data base school using the command as follows:

```
USE school;
```

The response of this command after the execution is given below:

```
Database changed
```

Now the database named *school* is the active database in our system. That means, the different DDL, DML and DCL commands we execute hereafter will be related to the database *school* We can check the existence of a database. The SHOW DATABASE command is used to check whether a database exists or not. It will list the entire databases in our system. The syntax is:

```
SHOW DATABASES;
```

The output of this command is shown in Figure 9.3.

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| school             |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

*Fig. 9.3: Output of SHOW DATABSES command*

## 9.2.4 Data types in SQL

*Data type* defines the type of value that may be entered in the column of a table. Data types ensure the correctness of the data, if we use them in a meaningful way. Care should be taken to assign correct data types for columns during the designing of the database. For example, if the numeric value 2 is designated as a text data type, such as a string, then it cannot be used in a mathematical operation; whereas the same number stored in an integer column can be used mathematically. So, let us understand the concept of SQL data types like, the type of data they represent, range of values supported by each of them, etc. Data types differ in different versions of SQL. Here, we look at the different data types available in MySQL.

MySQL data types are classified into three. They are numeric data type, string (text) data type, and date and time data type. All numerical values like 7, 100.234, -456,0, etc. can be represented by any of the numeric data types. The data "Aleena" (name of a student), "Kerala" (name of a state), 'F' (specification of a gender), etc. are string type by nature. Data like '01-01-2020', '23:34:3' can be represented by Date and Time data types.

## a. Numeric Data types

Numeric data type values can be used like any normal number. They can be added, subtracted, multiplied and divided. The most commonly used numeric data types in MySQL are **INT or INTEGER** and **DEC or DECIMAL**.

### (i) INT or INTEGER

As we know, integers are whole numbers without a fractional part. They can be positive, zero or negative. An integer value can be represented in MySQL by INT or INTEGER data type. The data items like 69, 0,-112 belong to INT data type.

### (ii) DEC or DECIMAL

Numbers with fractional parts can be represented by DEC or DECIMAL data type. The standard form of this type is DECIMAL (**size**,D) or DEC (**size**, D). The parameter **size** indicates the total number of digits the value contains including decimal part. The parameter **D** represents the number of digits after the decimal point. For example, the type specification DEC ( 5, 2) or DECIMAL ( 5, 2) denotes that 5 is the precision and 2 is the scale. The column with this specification is able to store any value having a maximum of five digits, out of which two are after the decimal point. That is, the range of values will be from -999.99 to 999.99.

Table 9.2 shows an overview of numeric data types in MySQL. Remember that the values are version dependent.

| Data types | Usage | Signed | Unsigned | Storage in Bytes |
|---|---|---|---|---|
| TINY INT | Very small integer values | -128to127 | 0 to 255 | 1 |
| SMALL INT | A small integer | -32768 to 32767 | 0 to 65535 | 2 |
| MEDIUM INT | A medium-sized integer value | -8388608 to 8388607 | 0 to 16777215 | 3 |
| INT | Normal sized | 2147483648to | | |
| | integer value | 2147483647 | 0 to 4294967295 | 4 |
| BIG INT | Large integer value | Value up to 19 digits | Value up to 264 | 8 |
| FLOAT (M, D) | Floating point numbers | Decimal precision can go to 24 places | | 4 |
| DOUBLE (M, D) | A double precision floating-point number | Decimal precision can go to 53 places | | 8 |
| DECIMAL (M, D) | Store exact preclslon values | A Decimal type can store a Maximum of 65 Digits, with 30 digits after decimal point. | | 8 |

*Table 9.2: Numeric data types of MySQL and their characteristics*

## b. String (Text) data types

String is a group of characters. The most commonly used string data types in MySQL are **CHARACTER or CHAR** and **VARCHAR**.

### (i) CHAR or CHARACTER

Character includes letters, digits, special symbols etc. The CHAR is a fixed length character data type. The syntax of this data type is CHAR (x), where x is the maximum number of characters that constitutes the data. The value of x can be between 0 and 255. CHAR is mainly used when the data in a column are of the same fixed length and small in size. For example, if we want to store data like 'M' for male and 'F' for female, in the column *Gender* of a table, it is better to declare that column as of type CHAR. It always uses the specified amount of space even though the data

need not require that much. If the number of characters in the data is less than the declared size of the column, the remaining character positions in the string will be filled with white spaces (spacebar character). But when we retrieve this value from the table, all trailing spaces are removed. Note that if the size of a column of type CHAR is 1, it is not necessary to mention the size, because the default size of CHAR type is 1.

### (ii) VARCHAR(size)

VARCHAR represents variable length strings. It is similar to CHAR, but the space allocated for the data depends only on the actual size of the string, not on the declared size of the column. For example, if we want to store data in the column *Name* of a table, it is better to declare that column as of type VARCHAR, because the data in the column may contain different number of characters. The length of the string can vary from 0 to 65535 characters (MySQL version dependent). The VARCHAR type saves memory space since VARCHAR type did not append spaces with the values when they are stored. The data like name of people, addresses etc. are examples of this data type.

## C. Date and Time data types

MySQL has data types for storing dates and times. The data type used to store date type value is **DATE** and to store time value is **TIME**.

### (i) DATE

The DATE data type is used to store dates. MySQL represents date values in YYYY•MM-DD format. The supported range is from 1000-01-01 to 9999-12-31. Dates are displayed in MySQL in one format, but we can use various date formats in our SQL statements. The YYYY-MM-DD is the standard format. But we can use any punctuation character between the date parts. For example, '2011-01-24', '2011/01/25', '20110126' are valid date combinations in MySQL. We can insert date type values into a column of DATE data type in any of the above formats. Although MySQL tries to interpret values in several formats, date parts must always be given in year-month-day order (for example, '98-09-04').

### (ii) TIME

The TIME data type is used to specify a column to store time values in MySQL. It shows values in the standard HH:MM:SS format. The TIME data type can be used to store a specific point in time (like 10 hours 05 minutes 25 seconds) as well as an interval of time between two points in time (like the time between now and the weekend) that may sometimes be larger than 23 hours. When manually entering a time into MySQL it is highly recommended that you use the exact format HH:MM:SS.

Let us do

Now look at Table 9.3 and fill the **Data type** column with suitable MySQL data type for

| Value | Data type |
|---|---|
| 325.678 | |
| 'A' | |
| 'Computer' | |
| '2016-01-01' | |
| 450 | |
| 22:32:45 | |
| 456787 | |

*Table 9.3: Data type of values*

**Know your progress**

1. ~~SQL stands for~~ ______ .
• ______ command is used to make a database active.
3. How can we see the names of databases in the system?
4. What is the difference between CHAR and VARCHAR data types?
5. Which is the format for storing date type data in MySQL?
6. Can we store the number 234 in a column declared with CHAR ( 5) data type?

## 9.3 SQL commands

SQL provides commands to perform different operations on database. As we mentioned earlier, the commands are classified as DDL commands, DML commands and DCL commands. Here, we will discuss the most commonly used DDL commands and DML commands. DDL commands are used to perform operations associated with the structure of database. The operations include creation of tables, modification in the structure of tables and removal of tables. DML commands are associated with the operations on the content of tables. These operations include insertion of records, retrieval of records, modification or updation of records and deletion of records.

These commands will be introduced in such a way that we can create a table for organizing data of a particular entity, retrieve required information and remove those we do not want to keep further.

## 9.4 Creating tables

Tables are the central and the most important objects in any relational database. The primary purpose of any database is to hold data that are stored in tables. Now

let us again consider the table *student* given in Table 9.1. How can we create a table with a set of columns? The DDL command **CREATE TABLE** is used to define a table by specifying the name of the table and giving the column definitions consisting of name of the column, data type and size, and constraints if any, etc. Remember that each table must have at least one column. The syntax of CREATE TABLE command is:

```
CREATE TABLE <table_name>
(<column_name> <data_type> [<constraint>]
[, <column_name> <data_type> [<constraint>,]
  ..............................
  ..............................   ) ;
```

Here, the <table_name> represents the name of the table that we want to create; <column_name> represents the name of a column in the table; <data_type> represents the type of data in a column of the table; and <constraint> specifies the rules that we can set on the values of a column. All the columns are defined within a pair of parentheses, and are separated by commas. We can define all the columns even in a single line.

## 9.4.1 Rules for naming tables and columns

While naming the tables and columns, certain points are to be remembered and they are listed below.

- The name may contain letters (A- Z, a-z), digits (0-9), under score(_) and dollar ($) symbol.
- The name must contain at least one character. (Names with only digits are invalid).
- The name must not contain white spaces, special symbols.
- The name must not be an SQL keyword .
- The name should not duplicate with the names of other tables in the same data base and with other columns in the same table.

In some MySQL versions, the table name can be a quoted identifier. The identifier quote character is the backtick(" "). If we use table name as quoted, then we can include any special symbols in the name of the table.

Now, let us create a table *student* to store the details of a group of higher secondary students in a school. The fields of the table and their descriptions are given in Table 9.4.

| Sl. No. | Attributes | Description |
|---|---|---|
| 1 | Admission number | Integer value |
| 2 | Name | String of 20 characters long |
| 3 | Gender | A single character |
| 4 | Date of birth | Date type |
| 5 | Course | String of 15 characters long |
| 6 | Family income | Integer value |

*Table 9.4: Attributes of student table*

The SQL statement or query to create a table in MySQL to incorporate the details shown in Table 9.4 is given in Query 9.1.

**Query9.1**

```
CREATE TABLE student
(adm_no INT,
name VARCHAR(20),
gender CHAR,
dob DATE,
course VARCHAR(15),
f_income INT);
```

Here, the CREATE TABLE statement creates a table named student with five columns adm_no, name, gender, dob, course and f_income. The columns adm_no can contain integer values, name can contain strings up to a maximum of 20 characters, gender can store a single character indicating whether a student is a boy or a girl, dob can store date of birth, course can contain the group in which he/she is studying and f_income can hold the monthly income of the family. The columns name and course will spare only the actual space required by the respective string data.

After creating a table, when we insert records in that table, some kind of restriction may need to be enforced in certain columns. Restrictions may be in the form of denying empty values in some columns and refusing duplicate values in some other columns. But it should be addressed during the table creation itself MySQL provides some keywords called constraints, for this purpose.

## 9.4.2 Constraints

***Constraints*** are the rules enforced on data that are entered into the column of a table. When we create a table, we can apply constraints on the values that can be entered into its fields. If this is specified in the column definition, SQL will not accept any values that violate the criteria concerned. This ensures the accuracy and reliability of the data in the database. The constraints ensure database integrity and hence they are often called data base integrity constraints. Constraints could be column level or table level.

## a. Column Constraints

Column constraints are applied only to individual columns. They are written immediately after the data type of the column. The following are column constraints:

### i. NOT NULL

This constraint specifies that a column can never have NULL values. NULL is a keyword in SQL that represents an empty value. It is important to remember that NULL does not equate to a blank or a zero; it is something else entirely. Though a blank is equal to another blank and a zero is equal to another zero, a NULL is never equal to anything, not even another NULL. Two NULL values cannot be added, subtracted or compared.

### ii. AUTO_INCREMENT

MySQL uses the AUTO_INCREMENT keyword to perform an auto-increment feature. If no value is specified for the column with AUTO_INCREMENT constraint, then MySQL will assign serial numbers automatically and insert the newly assigned value in the corresponding column of the new record. By default, the starting value for AUTO_INCREMENT is 1, and it will be incremented by 1 for each new record. This special behavior also occurs if we explicitly assign the value NULL to the column. The AUTO_INCREMENT feature makes it easy to assign a unique ID to each new row, because MySQL generates the values for us. The auto increment column must be defined as the primary key of the table. Only one AUTO_INCREMENT column per table is allowed.

### iii. UNIQUE

It ensures that no two rows have the same value in the column specified with this constraint.

### iv. PRIMARY KEY

This constraint declares a column as the primary key of the table. This constraint is similar to UNIQUE constraint except that it can be applied only to one column or a combination of columns. The primary keys cannot contain NULL values. In other words, it can be considered as a combination of UNIQUE and NOT NULL constraints. A PRIMARY KEY constraint is used to enforce a rule that a column should contain only unique, non-NULL data.

### v. DEFAULT

Using this constraint, a default value can be set for a column, in case the user does not provide a value for that column of a record.

Let us apply some of these constraints in the student table and modify Query 9.1 as given in Query 9.2.

Query9.2

```
CREATE TABLE student
(adm_no INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(20) NOT NULL,
gender CHAR DEFAULT 'M',
dob DATE,
course VARCHAR(15)
f_income INT);
```

In Query 9.2, the constraints PRIMARY KEY and AUTO_INCREMENT are applied to the column adm_no. So, this column will not allow duplicate values during data entry. If we donot specify a value for this column, MySQL will generate a data automatically. The constraint NOT NULL applied to the column name does not allow to leave the column with an ul l value. That is, data is a must in this column. Similarly, if we do not give a value to the column gender, 'M' will be stored as the default value.

## b. Table constraints

Table constraints are similar to column constraints; the main difference is that table constraints can be used not only on individual columns, but also on a group of columns. When a constraint is to be applied on a group of columns of a table, it is called table constraint. The table constraint appears at the end of the table definition. For example, Query 9.3 creates a table named ***stock***. The constraint UNIQUE is applied to the combination of icode and iname.

Query9.3

```
CREATE TABLE stock
(icode CHAR(2) PRIMARY KEY AUTO INCREMENT,
iname VARCHAR(30) NOT NULL,
dt_purchase DATE,
rate DECIMAL (10, 2),
qty INT,
UNIQUE (icode, iname));
```

In Query 9.3, the constraint UNIQUE is applied to the combination of values of columns icode and iname. It enforces a situation that no two rows can have the same values for the columns icode and iname when taken together.

### 9.4.3 Viewing the structure of a table

We have created two tables, *student* and *stock*. How do we know the structure of the table after its creation? The **DESCRIBE** command is used to display the structure definitions of a table. The syntax is:

```
DESCRIBE  <table_name>;
          OR
DESC  <table_name>;
```

The structure of the table student can be viewed using the command:

```
DESC  student;
```

Figure 9.4 shows the output of this command.

```
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| adm_no   | int(11)     | NO   | PRI | NULL    | auto_increment |
| name     | varchar(20) | NO   |     | NULL    |                |
| gender   | char(1)     | YES  |     | M       |                |
| dob      | date        | YES  |     | NULL    |                |
| course   | varchar(15) | YES  |     | NULL    |                |
| f_income | int(11)     | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
6 rows in set (0.02 sec)
```

*Fig. 9.4: Structure of student table*

Note that in Query 9.2, we did not mention the size for the columns adm_no and f_income while creating the table. But MySQL takes the default size 11 as the size of these two columns. The column adm_no is declared with PRIMARY KEY and AUTO_INCREMENT. So this column will not allow duplicate values and null values. If we do not specify a value for this column, a new value will be generated by adding 1 to the value of the respective column of the previous record. In the absence of a value for this column for the first record, MySQL gives 1 as the value. We can also see that a default value 'M' is set for the column gender. Figure 9.4 illustrates these aspects. There is a command SHOW TABLES, which shows the tables created in the current database as given in Figure 9.5.

```
+-------------------+
| Tables_in_school  |
+-------------------+
| student           |
+-------------------+
1 row in set (0.00 sec)
```

*Fig. 9.5: Output of SHOW TABLES*

Write the structure of the table ***stock***, referring to Figure 9.4 and Query 9.3. Verify your answer in the lab with the command, `DESC stock;`

**Know your progress**

1. Which of the following commands is used to display the structure of a table?
   a. LIST  b. SHOW  c. DESCRIBE  d. STRUCT
2. Write the syntax of CREATE TABLE command.
3. Name the different column constraints.
4. What is the difference between primary key constraint and unique constraint?
5. What are the features of AUTO_INCREMENT constraint?
6. Write down the rules for naming a table.
7. How many columns in a table can be specified as primary key of the table?

## 9.5 Inserting data into tables

We have created a database and its tables. Now we need to put some records into these tables. We have the details of six students in Table 9.1. Let us discuss how these records can be inserted into the table.

The DML command **INSERT INTO** is used to insert tuples into tables. The syntax is:

```
INSERT INTO <table_name> [<column1>,<column2>, ... ,<columnN>]
VALUES(<value1>,<value2>, ... ,<valueN>);
```

Here <table_name> is the name of the table into which the tuples are to be inserted; <column1>, <column2>, ...., <columnN> indicate the name of columns in the table into which values are to be inserted; <value1>,<value2>, ...., <valueN> are the values that are inserted into the columns specified in the <column_list>.

For example, let us insert a new record into the table *student* with data *1001, 'Alok', 'M', 1998/10/2, 'Science', 24000* into the columns adm_no, name, gender, dob, course and f_income, respectively. Query 9.4 makes it possible.

Query9.4

```
INSERT INTO student
VALUES (1001, 'Alok', 'M', '1998/10/2', 'Science',
        24000);
```

The response of the system will be:

```
Query OK, 1 row affected (0.05 sec).
```

The INSERT statement adds a new row to the table *student* giving a value for every column in the table. While inserting a row, if we provide values for all the columns of the table, we need not specify the name of column(s) in the query. But we need to make sure that the order of the values is in accordance with the order of the columns in the table. Now let us insert another row with some modifications as shown in Query9.5.

**Query9.5**

```
INSERT INTO student (name, dob, course, f_income)
VALUES  ('Nike', '1998/11/26', 'Science', 35000);
```

The response for this statement will be:

```
Query OK, 1 row affected (0.01 sec)
```

In Query 9.5, admission number and gender are not provided. Being a record after the one with admission number 1001, the admission number of this student will be 1002. The gender will be set with the default value 'M'.

While inserting data into tables, the following points are to be taken care of:

- While adding a new row, we should ensure the data type of the value and the column matches.
- We follow the integrity constraints, if any, defined for the table.
- CHAR or VARCHAR type data should be enclosed in single quotes or double quotes.
- Column values for DATE type columns are to be provided within single quotes. The string will internally be converted into DATE data type.
- Null values are specified as NULL (or null) without quotes.
- If no data is available for all columns, then the column list must be included, following the table name.

MySQL allows inserting several rows into a table with a single INSERT command by specifying multiple value lists. The general format is as follows:

```
INSERT INTO <table-name> VALUES( ... ), ( ... ), ... ;
```

Let us insert two more records given in Table 9.1 using Query 9.6.

**Query9.6**

```
INSERT INTO student (name, dob, course, f_income)
VALUES('Bharath', '1999/01/01', 'Commerce' ,45000),
           ('Virat', '1998/12/05', 'Science' ,22000);
```

The response of the system will be:

```
Query OK, 2 rows affected (0.02 sec)
Records: 2 Duplicates: 0  Warnings: 0
```

We can observe that the two records are given within separate pairs of parentheses. The response indicates that the two records are inserted successfully.

Suppose we do not have the monthly income data of a student. How can we insert the record? Query 9.7 illustrates the solution.

**Query9.7**

```
INSERT INTO student(name, dob, gender, course)
VALUES ('Meera','1998/08/15', 'F', 'Science');
```

As a response to this query, the value for `adm_no` will be generated by the system, but the value for `f_income` column will be kept as `NULL`. The order of columns is also changed in this query. The absence of values in a row can be managed in another way also as shown in Query 9.8.

**Query9.8**

```
INSERT INTO student(name, dob, gender, course,
                                     f_income)
VALUES('Divakar', '1998/02/21', 'Science',NULL);
```

Note that, in the `VALUES` clause, `NULL` is given for `f_income`.

Let us insert some more records in table *student*. Write queries to store the details of students shown in Table 9.5.

| adm_no | name | gender | doh | course | f_income |
|---|---|---|---|---|---|
| 1025 | Kaushi | M | 1998/10/2 | Commerce | 17000 |
| 1026 | Niveditha | F | 1999/03/04 | Humanities | 52000 |
| 1027 | Sreekumar | M | 1998/06/06 | Science | |
| 1057 | Chaithanya | F | 1999/06/03 | Science | ◯ |

*Table 9.5: More records for Student table*

## Know your progress

1. Which of the following is used to add a row into a table?
   a. ADD    b. CREATE    c. INSERT    d. MAKE
2. Which statement is used to insert new data into a table?
   a. ADD RECORD    b. INSERT RECORD
   c. INSERT INTO    d. INSERT ROW
3. Write the essential keywords used along with `INSERT` command.

Let us do

Write SQL statements to insert some records in the table stock created using Query 9.3. While giving values to the columns, utilise the facility of AUTO_INCREMENT and UNIQUE constraints.

## 9.6 Retrieving information from tables

We have created a database school and a table student, and then inserted ten records into it. Now let us learn how information is retrieved from the data stored in tables. It is a kind of data manipulation operation and SQL provides the command **SELECT** for this purpose. It is used to retrieve information from specified columns in a table. The SELECT command has several forms of its own. The simplest form of SELECT command is:

```
SELECT <column_name> [,<column_name>, <column_name>, ... ]
FROM <table_name>;
```

Here <column_name> indicates the column from which data is retrieved and <table_name> denotes the name of the table from which the information is retrieved. The name of the table is given with the keyword **FROM**, which is an essential clause with SELECT command. The SELECT command will display the data in columns, in the order in which they appear along with the SELECT command.

Now let us illustrate the execution of SELECT command through various queries. On executing Query 9.9 we get the name and course of students in the table *student* as shown in Figure 9.6.

Query9.9

```
SELECT name, course
FROM student;
```

```
+------------+-------------+
| name       | course      |
+------------+-------------+
| Alok       | Science     |
| Nike       | Science     |
| Bharath    | Commerce    |
| Virat      | Science     |
| Meera      | Science     |
| Divakar    | Science     |
| Kaushi     | Commerce    |
| Niveditha  | Humanities  |
| Sreekumar  | Science     |
| Chaithanya | Science     |
+------------+-------------+
10 rows in set (0.00 sec)
```

*Fig. 9.6: Output of Query 9.9*

If we want to display the entire column values of a table, we need not give a complete list of columns of the relation. Instead, an asterisk (*) symbol can be used to substitute the complete list of columns as shown in Query 9.10. The output is shown in Figure 9.7.

Query9.10

```
SELECT * FROM student;
```

```
+--------+------------+--------+------------+------------+----------+
| adm_no | name       | gender | deb        | course     | f_income |
+--------+------------+--------+------------+------------+----------+
|   1001 | Alok       | M      | 1998-10-02 | Science    |    24000 |
|   1002 | Nike       | M      | 1998-11-26 | Science    |    35000 |
|   1003 | Bharath    | M      | 1999-01-01 | Commerce   |    45000 |
|   1004 | Virat      | M      | 1998-12-05 | Science    |    22000 |
|   1005 | Meera      | F      | 1998-08-15 | Science    |     NULL |
|   1006 | Divakar    | M      | 1998-02-21 | Science    |     NULL |
|   1025 | Kaushi     | M      | 1998-10-02 | Commerce   |    17000 |
|   1026 | Niveditha  | F      | 1999-03-04 | Humanities |    52000 |
|   1027 | Sreekumar  | M      | 1998-06-06 | Science    |    15000 |
|   1057 | Chaithanya | F      | 1999-06-03 | Science    |     NULL |
+--------+------------+--------+------------+------------+----------+
10 rows in set (0.00 sec)
```

*Fig. 9.7: Entire content of student table*

We can see NULL values in column f_income of some rows in Figure 9.7. It is due to the missing of values in the insertion of those rows (refer Queries 9.7, 9.8 and Table 9.1, and 9.5).

## 9.6.1 Eliminating duplicate values in columns using DISTINCT

Suppose we want to know the names of different courses in the table *student*. If we construct a query **SELECT course FROM student;** it will display the column course with all the ten values as shown in Figure 9.7. Data will be selected from all rows of the relation to display, even if the data appearing in the result get duplicated. This duplication can be eliminated using the keyword **DISTINCT** as given in Query 9.11. Observe the output given in Figure 9.8.

Query 9.11

```
SELECT DISTINCT course
FROM student;
```

In the output, there are no duplicate values. If the column used with DISTINCT keyword contains more than one NULL value, only one will be shown in the result.

If we give the keyword ALL in the place of DISTINCT, then the result will contain all the duplicate values in the column. That is, the output will be the same as that in the case when we neither specify DISTINCT nor ALL.

```
+------------+
| course     |
+------------+
| Science    |
| Commerce   |
| Humanities |
+------------+
3 rows in set (0.25 sec)
```

*Fig. 9.8: Use of DISTINCT*

## 9.6.2 Selecting specific rows using WHERE clause

In certain situations, we need to display only a subset of a table. For example, we may require the details of female students only, or the details of students whose family monthly income is below Rs. 25000/-. In these cases, there is a selection in

the retrieval of records. Obviously the selection is based on some condition(s). SQL enables us to impose some selection criteria for the retrieval of records with **WHERE** clause of SELECT command. The syntax of SELECT command with WHERE clause is:

```
SELECT  <column_name>[,<column_name>,<column_name>,  ... ]
FROM  <table_name>
WHERE  <condition>;
```

When a WHERE clause is present, the SELECT command goes through the entire table one row at a time and examines whether the row satisfies the condition(s) or not. If a row satisfies the condition, that row is displayed in the output. The conditions are expressed with the help of relational operators and logical operators. MySQL provides a variety of such operators and they are listed in Table 9.6.

| Operator | | Meaning/Result |
|---|---|---|
| [illegible] | = | Equal to |
| | <> or != | Not equal to |
| | > | Greater than |
| | < | Less than |
| | >= | Greater than or equal to |
| | <= | Less than or equal to |
| [illegible] | NOT | True when condition is false |
| | AND | True if both the conditions are true |
| 0 | OR | True if either of the conditions is true |

*Table 9.6: Operators used for setting conditions*

Let us write an SQL statement to display the details of female students in the table. Query 9.12 is the required statement which contains a WHERE clause and = operator for setting the condition and Figure 9.9 shows the output.

```
SELECT * FROM student
WHERE gender='F';
```

```
+--------+-------------+--------+------------+-------------+----------+
| adm_no | name        | gender | dob        | course      | f_income |
+--------+-------------+--------+------------+-------------+----------+
   1005    Meera         F        1998-08-15   Science          NULL
   1026  | Niveditha   | F      | 1999-03-04 | Humanities  |   52000 |
   1057  | Chaithanya  | F      | 1999-06-03 | Science     |    NULL |
+--------+-------------+--------+------------+-------------+----------+
3 rows in set (0.06 sec)
```

*Fig. 9.9: List of female students - the output of Query 9.12*

Suppose we want to see the name, course and monthly family income of only the Science group students whose income is below Rs. 25000. Here we have to constitute two conditions - one for checking the income and the other for checking the course. When these two conditions are matched while going through the records in the table, the values of their name, course and f_income columns will be retrieved for display. Query 9.13 is the required query and Figure 9.10 shows the output. Note that **AND** operator is used to combine the two conditions.

**Query9.13**

```
SELECT name,  course,  f_income  FROM  student
WHERE  course='Science'  AND  f_income<25000;
```

```
+-----------+---------+----------+
| name      | course  | f_income |
+-----------+---------+----------+
| Alok      | Science |    24000 |
| Virat     | Science |    22000 |
| Sreekumar | Science |    15000 |
+-----------+---------+----------+
3 rows in set (0.00 sec)
```

*Fig. 9.10: Use of AND operator*

Write a query to display the name, course and monthly income of students studying in courses other than *Science* group (i.e., the students of *Commerce* and *Humanities* groups).

While writing the above query, you might have used **OR** operator for setting the condition. The same information can be obtained by the use of **NOT** operator. Query 9.14 illustrates this and Figure 9.11 shows the corresponding output.

**Query9.14**

```
SELECT name,  course,  f_income  FROM  student
WHERE  NOT  course=' Science';
```

Identifysomerequirementswithrespect [illegible] by setting different conditions, and construct queries using appropriate operators.

```
+-----------+------------+----------+
| [illegible]                       |
+-----------+------------+----------+
| Bharath   | Commerce   |    45000 |
| [illegible]                       |
| Kaushi    | Commerce   |    17000 |
| Niveditha | Humanities |    52000 |
+-----------+------------+----------+
3 rows in set (0.00 sec)
```

*Fig. 9.11: Use of NOT operator*

SQL has a set of special operators for setting conditions. These include **BETWEEN ... AND**, **IN**, **LIKE** and **IS**. Let us discuss how these operators help us to constitute conditions.

## a. Condition based on a range of values

A range of values can be given as condition. The SQL operator **BETWEEN ... AND** is used to specify the range. Suppose we need a list of students whose monthly income falls in the range of Rs. 25000/- to Rs. 45000/-. We know that this can be obtained by the statement given in Query 9.15. Figure 9.12 shows the output.

**Query9.15**

```
SELECT name,  f income  FROM  student
WHERE  f  income>=25000  AND  f  income<=45000;
```

The same output can be obtained by using the statement given in Query 9.16. In this statement the operator BETWEEN ... AND is used to construct the condition. The output includes both the lower and upper values given in the range.

Query9.16

```
SELECT  name,  f_income  FROM  student
WHERE  f_income  BETWEEN  25000   AND  45000;
```

From Query 9.16 and Figure 9.12, we can conclude that the BETWEEN ... AND operator allows specifying a range of values which belong to either numeric or date and time type data.

```
+----------+----------+
| name     | f_income |
+----------+----------+
| Nike     |    35000 |
| Bharath  |    45000 |
+----------+----------+
2 rows in set (0.00 sec)
```

*Fig. 9.12: Use of BETWEEN...AND operator*

## b. Conditions based on a list of values

While setting conditions for the retrieval of records, a list of values can be provided. The values may be of any data type, but it should match with those of the column used in the condition. In such a case, the operator IN is used with the list. Suppose we want to retrieve the details of students in *Commerce* and *Humanities* groups. The statement in Query 9.17 can generate the details as shown in Figure 9.13.

```
SELECT*  FROM  student
WHERE  course='Commerce'  OR  course='Humanities';
```

```
+--------+-----------+--------+------------+------------+----------+
| adm_no | name      | gender | dob        | course     | f_income |
+--------+-----------+--------+------------+------------+----------+
|   1003 | Bharath   | M      | 1999-01-01 | Commerce   |    45000 |
|   1025 | Kaushi    | M      | 1998-10-02 | Commerce   |    17000 |
|   1026 | Niveditha | F      | 1999-03-04 | Humanities |    52000 |
+--------+-----------+--------+------------+------------+----------+
3 rows in set (0.00 sec)
```

*Fig. 9.13: Output of Query 9.17*

The same output can be obtained by using IN operator in the condition as shown in Query 9.18.

```
SELECT*  FROM  student
WHERE  course  IN ('Commerce',  'Humanities') ;
```

As we see in Query 9.18, the IN operator checks whether the value in the specified column (here it is course) of a record matches any of the values in the given list. When matches are found while going through the records, they will be displayed. Since there are only three courses in the table, the result shown in Figure 9.13 can also be obtained using Query 9.19.

```
SELECT*  FROM  student
WHERE  course  NOT  IN ('Science') ;
```

### c. Conditions based on pattern matching

There may be the need of retrieving data based on some pattern matching. That is, string data having some similarities in characters may be the criterion for the retrieval of records. SQL provides a pattern matching operator **LIKE** for this purpose. Patterns are specified using two special wildcard characters% and_ (underscore), where% (percentage) matches a substring of characters and_ (underscore) matches a single character. Patterns are case sensitive; i.e.; uppercase characters do not match lower case characters. Consider the following examples:

- "Ab%" matches any string beginning with "Ab".
- "%cat%" matches any string containing "cat" as substring. For example, "education", "indication", "catering" etc.
- "____" matches any string of exactly four characters without any space in between them.
- "__%" matches any string of at least 3 characters.

Query 9.20 illustrates the use of LIKE operator. It gives a list of names in the table that end with the substring 'ar'. Figure 9.14 shows the output of this query.

Query 9.20

```
SELECT name FROM student
WHERE name LIKE '%ar';
```

```
+-----------+
| name      |
+-----------+
| Divakar   |
| Sreekumar |
+-----------+
2 rows in set (0.00 sec)
```

*Fig. 9.14: Use of pattern matching*

The following query will display only 'Divakar' since there are two _(underscore) characters for pattern matching.

```
SELECT name FROM student
WHERE name LIKE 'Div__ar';
```

### d. Conditions based on NULL value search

We have seen that there may be NULL values in some fields of a record. We can retrieve such records with the help of IS operator. The condition will be true when the specified column contains a NULL value in the records. In the *student* table, the f_income column contains NULL value in three records (refer to Figure 9.7). Query 9.21 retrieves these records and Figure 9.15 shows the output.

```
+------------+---------+
| name       | course  |
+------------+---------+
| Meera      | Science |
| Divakar    | Science |
| Chaithanya | Science |
+------------+---------+
3 rows in set (0.00 sec)
```

*Fig. 9.15: NULL value checking*

Query 9.21

```
SELECT name, course FROM student
WHERE f_income IS NULL;
```

On the other hand, if we want to retrieve the records containing non-null values in the f_income column, the following statement can fulfill the task:

```
SELECT name, course FROM student
WHERE f_income IS NOT NULL;
```

Let us do

Identify some requirements with respect to the table ***stock*** which need use of the special SQL operators we discussed, and construct queries to solve the problems.

**Know your progress**

1. Name the keyword used with `SELECT` command to avoid duplication in the values of a column.
2. Which is the essential clause for `SELECT` query?
3. Which of the following operators is used to check for `NULL` value in a column?

   a. `IN`   b. `LIKE`   c. `IS`   d. `NOT`
4. The operator used for checking pattern matching is ________.
5. What is wrong with the following statement?

   `SELECT * FROM emp WHERE grade= NULL;`
6. The command that extracts records from a table is _ _ _ _ _.

## 9.6.3 Sorting results using ORDER BY clause

As we can see in the results of the `SELECT` queries discussed so far, the records are always obtained in the order in which they appear in the table. Is it possible to display them in a specific order - ascending or descending, of some column values? Yes. The result of a query can be sorted in the ascending or descending order by making use of **ORDER BY** clause. The order is to be specified by using the keyword **ASC** (for ascending) or **DESC** (for descending) along with the column name that is used with `ORDER BY` clause. By default, the display will be in the ascending order. Remember that only the results that appear on the screen are sorted. The order of records in the table will be kept unaltered.

Query 9.22 will display the details of students in the alphabetical order of their names. Figure 9.16 shows the output.

Query9.22

```
SELECT * FROM student ORDER BY name;
```

```
+---------+-------------+--------+------------+------------+----------+
| adm_no  | name        | gender | dob        | course     | f_income |
+---------+-------------+--------+------------+------------+----------+
|    1001 | Alok        | M      | 1998-10-02 | Science    |    24000 |
|    1003 | Bharath     | M      | 1999-01-01 | Commerce   |    45000 |
|    1057 | Chaithanya  | F      | 1999-06-03 | Science    |     NULL |
|    1006 | Divakar     | M      | 1998-02-21 | Science    |     NULL |
|    1025 | Kaushi      | M      | 1998-10-02 | Commerce   |    17000 |
|    1005 | Meera       | F      | 1998-08-15 | Science    |     NULL |
|    1002 | Nike        | M      | 1998-11-26 | Science    |    35000 |
|    1026 | Niveditha   | F      | 1999-03-04 | Humanities |    52000 |
|    1027 | Sreekumar   | M      | 1998-06-06 | Science    |    15000 |
|    1004 | Virat       | M      | 1998-12-05 | Science    |    22000 |
+---------+-------------+--------+------------+------------+----------+
10 rows in set (0.00 sec)
```

*Fig. 9.16: Details of students according to the alphabetical order of their names*

Suppose we want to see the details according to their monthly family income. If we have to obtain the result from the highest income to the lowest, Query 9.23 can serve the purpose. Figure 9.17 shows the output.

**Query9.23**
```
SELECT * FROM student
ORDER BY f_income DESC;
```

```
+--------+-------------+--------+------------+------------+----------+
| adm_no | name        | gender | dob        | course     | f_income |
+--------+-------------+--------+------------+------------+----------+
|   1026 | Niveditha   | F      | 1999-03-04 | Humanities |    52000 |
|   1003 | Bharath     | M      | 1999-01-01 | Commerce   |    45000 |
|   1002 | Nike        | M      | 1998-11-26 | Science    |    35000 |
|   1001 | Alok        | M      | 1998-10-02 | Science    |    24000 |
|   1004 | Virat       | M      | 1998-12-05 | Science    |    22000 |
|   1025 | Kaushi      | M      | 1998-10-02 | Commerce   |    17000 |
|   1027 | Sreekumar   | M      | 1998-06-06 | Science    |    15000 |
|   1006 | Divakar     | M      | 1998-02-21 | Science    |     NULL |
|   1005 | Meera       | F      | 1998-08-15 | Science    |     NULL |
|   1057 | Chaithanya  | F      | 1999-06-03 | Science    |     NULL |
+--------+-------------+--------+------------+------------+----------+
10 rows in set (0.00 sec)
```

*Fig. 9.17: Age-based listing of students (younger to older)*

As shown in Figure 9.17, if the column used with `ORDER BY` clause contains `NULL` values, they will be listed last.

Multiple sorting can be performed after selection with the help of `ORDER BY` clause. For example, if we need a course-based listing of students in the alphabetical order of their names, a statement as given in Query 9.24 can be used. Figure 9.18 shows its output.

**Query9.24**
```
SELECT name, course FROM student
ORDER BY course, name;
```

Look at the output shown in Figure 9.18. First the values in the column course is arranged alphabetically and then the names are arranged under each course.

```
+-------------+-------------+
| name        | course      |
+-------------+-------------+
| Bharath     | Commerce    |
| Kaushi      | Commerce    |
| Niveditha   | Humanities  |
| Alok        | Science     |
| Chaithanya  | Science     |
| Divakar     | Science     |
| Meera       | Science     |
| Nike        | Science     |
| Sreekumar   | Science     |
| Virat       | Science     |
+-------------+-------------+
10 rows in set (0.00 sec)
```

*Fig. 9.18: Multiple sorting*

Earlier we have used the WHERE clause for condition based retrieval of records. The retrieved records can be displayed in a particular order using ORDER BY clause.

Suppose we want to display the name and family income of *Science* group students according to the descending order of income. The two clauses can be combined to generate this output as shown in Query 9.25. The output is shown in Figure 9.19.

```
SELECT name, course, f_income FROM student
WHERE course='Science'
ORDER BY f_income DESC;
```

As shown in Figure 9.19, the records with NULL values in the sorted column will appear at last as they appear in the table. Another important aspect is that, if an ORDER BY clause is used in a SELECT statement, it should appear only after the WHERE clause, if any. Because, the records should be selected first, then only they are given for reordering.

```
+------------+-----------+-----------+
| name       | course    | f_income  |
+------------+-----------+-----------+
| Nike       | Science   |   35000   |
| Alok       | Science   |   24000   |
| Virat      | Science   |   22000   |
| Sreekumar  | Science   |   15000   |
 Meera                     Science
 NULL Divakar              Science
 NULL  Chaithanya          Science
 NULL
+------------+-----------+-----------+
7 rows in set (0.00 sec)
```

*Fig. 9.19: Condition based selection and sorting*

## Know your progress

1. What is the meaning of ORDER BY clause?
2. Which keyword is used for sorting the data in descending order inMySQL?

   a. DESC    b.ASC    c. SORT    d. MODIFY
3. In which order are the records displayed in the absence of ORDER BY clause.
4. In SQL, what is the default sort order of the ORDER BY clause?

Identify some requirements with respect to the table ***stock*** which need the sorting of records based on different conditions, and construct queries to solve the problems.

### 9.6.4 Aggregate functions

MySQL provides a number of built-in functions that can be applied to all rows in a table or to a subset of the table specified by WHERE clause. These functions are called aggregate functions because they operate on aggregate of tuples (records). The result of an aggregate function is a single value.

Commonly used aggregate functions are given in Table 9.7.

| **Function** | **Return value** |
|---|---|
| SUM ( ) | Total of the values in the column specified as argument. |
| AVG() | Average of the values in the column specified as argument. |
| MIN () | Smallest value in the column specified as argument. |
| MAX ( ) | Largest of the values in the column specified as argument. |
| COUNT ( ) | Number of non NULL values in the column specified as argument. |

*Table 9.7: Some built-in functions of MySQL*

Let us discuss these functions to generate useful information from the table. Query 9.26 gives the highest, lowest and average family monthly incomes of the students. Figure 9.20 shows the result.

Query9.26

```
SELECT MAX(f_income), MIN(f_income), AVG(f_income)
FROM  student;
```

While calculating these values, the NULL values in the argument column are not considered.

```
+---------------+---------------+---------------+
| MAX(f_income) | MIN(f_income) | AVG(f_income) |
+---------------+---------------+---------------+
|         52000 |         15000 |    30000.0000 |
+---------------+---------------+---------------+
1 row in set (0.03 sec)
```

*Fig. 9.20: Use of aggregate functions*

These functions can be applied to a subset of the table formed by some selection crierion as shown in Query 9.27. It gives the number of students in the *Science* group. Figure 9.21 shows the output.

```
SELECT  COUNT (*) ,  COUNT (f_income)
FROM  student
WHERE  course='Science';
```

If we refer to Figure 9.19, we can see that the correct answer is 7. Then, why is there a mismatch between the two column values in Figure 9.21? The first column is COUNT () and it gives the number of records having *Science* as the value in column course. Note that the* (asterisk) symbol stands for the collection of all the columns in the table. So, if there is at least one field in a record, that record will be taken into consideration for COUNT ( *) . But COUNT ( f_income) in Query 9.27 counts only the non-NULL values in column f income of the records which contains *Science* in column course. This is why the second column in Figure 9.21 contains only **4**. Refer to Figure 9.17 and identify the rows that conatins NULL in column f_income.

```
+----------+------------------+
| COUNT(*) | COUNT (£_income) |
+----------+------------------+
|        7 |                4 |
+----------+------------------+
1 row in set (0.03 sec)
```

*Fig. 9.21: Use of COUNT() function*

## 9.6.5 Grouping of records using GROUP BY clause

Sometimes, we need to divide the tuples in a table into different groups based on the values in a column. The rows of a table can be grouped together based on a common value using the **GROUP BY** clause. The attribute (column) specified in the **GROUP BY** clause is used to form groups. Tuples with the same value in the attribute specified in the GROUP BY clause are placed together in one group. Thus different groups are formed based on distinct values in the column. So this process can be considered as categorisation of records.

Using Query 9.27, number of students in *Science* group is obtained. Suppose we want to know the number of students in each group along with average family monthly income. Query 9.28 can be used. The output is shown in Figure 9.22.

```
SELECT course, COUNT ( *) , AVG ( f income)
FROM student
GROUP BY course;
```

Here different groups are formed under different values (*'Commerce'*, *'Humanities'* and *'Science'*) in column course and the functions COUNT ( *) and AVG(f_income) are applied to each group.

```
+------------+----------+-------------+
| course     | COUNT()  | AVG(f_income) |
+------------+----------+-------------+
| Commerce   |        2 |  31000.0000 |
| Humanities |        1 |  52000.0000 |
| Science    |        7 |  24000.0000 |
+------------+----------+-------------+
3 rows in set (0.00 sec)
```

*Fig. 9.22: Grouping of rows using GROUP BY clause*

## 9.6.6 Applying conditions to form groups using HAVING clause

We have already learnt about the WHERE clause, which places conditions on individual rows. We can apply conditions to form groups with the help of **HAVING** clause.

This clause is used along with GROUP BY clause. The condition in HAVING clause is applied to form groups of records, not individual rows.

Query 9.29 applies a condition for forming groups. Here the groups will be formed only if the condition provided with HAVING clause is satisfied.

Query9.29

```
SELECT course, COUNT (*) FROM student
GROUP BY course
HAVING COUNT(*) > 3;
```

```
+----------+----------+
| course   | COUNT(*) |
+----------+----------+
| science  |        7 |
+----------+----------+
1 row in set (0.00 sec)
```

*Fig. 9.23: Condition based grouping*

If we refer to Figures 9.22 and 9.23, we can find that only *Science* group has more than 3 students. Groups are not formed based on values '*Commerce*' and '*Humanities*', since the number of records with each of these values are 2 and 1, respectively. That is why details of these groups are not shown by Query 9.29.

**Know your progress**

1. List the aggregate functions of SQL.
2. How do COUNT (*) and COUNT (column_name) differ?
3. What is the difference between WHERE clause and HAVING clause?
4. The usage SUM (*) or MAX (*) is invalid. Why?
5. What will be the result of the following query?
   `SELECT COUNT (DISTINCT course) FROM student;`

Let us do

Identify some queries with respect to the table ***stock*** which requires the utilisation of aggregate functions and ORDER BY clause. Write SQL statements to answer these queries.

## 9.7 Modifying data in tables

In certain situations, we need to change the values in the columns of a table. For example, following a revision in salary or wages, the family monthly income of a student may be changed. Similarly, the missing values (NULL) in monthly income of some students may need to be replaced by valid data at a later stage. This kind of changes can be done using the DML command **UPDATE.** It changes the values in one or more columns of specified rows. These changes may be affected in all or only in selected rows of the table. The new data for the column within these rows is given using the keyword **SET,** which is the essential clause of UPDATE command. The new data can be a constant, an expression or data from other tables.

4 u ¨

The syntax of UPDATE command is:

```
UPDATE   <table_name>
SET <column_name>= <value> [,<column_name> = <value>, ... ]
[WHERE  <condition>] ;
```

Suppose the family monthly income of the student *Kaushi* is to be changed to Rs. 27000/-. Query 9.30 can perform this modification.

**Query9.30**

```
UPDATE   student
SET  f–income=27000
WHERE  name=' Kaushi';
```

After the execution of this query, the response of MySQL will be as follows:

```
Query  OK,  1  row  affected   (0.08  sec)
Rows  matched:  1   Changed:  1   Warnings:  0
```

Let us see the change in the record using the query,

```
SELECT  *  FROM  student  WHERE  name='Kaushi';
```

The output of this query is shown in Figure 9.24. Compare this figure with Figure 9.1 7 and see the change.

```
+--------+--------+--------+------------+----------+----------+
| adm_no | name   | gender | dob        | course   | f_income |
+--------+--------+--------+------------+----------+----------+
|   1025 | Kaushi | M      | 1998-10-02 | Commerce |    27000 |
+--------+--------+--------+------------+----------+----------+
1 row in set (0.00 sec)
```

*Fig. 9.24: Details of Kaushi with modified data in f_income column*

We can use expressions to set the column values in records. Query 9.31 illustrates this concept.

**Query9.31**

```
UPDATE   student
SET  f_income=f_income+1000
WHERE  f_income<25000;
```

The output of this query will be as follows:

```
Query  OK,  3  rows  affected   (0.06  sec)
Rows  matched:  3   Changed:  3   Warnings:  0
```

The output reveals that three records satisfy the given condition and hence the values in f_income column of these records are incremented by 1000 (refer to Figures 9.17, 9.25).

The columns with NULL values can also be modified with UPDATE command. Suppose we want to store Rs. 20000/- as the monthly income in the respective field of the records in which the column contains NULL value at present. Query 9.32 makes it possible.

**Query9.32**

```
UPDATE student
SET f_income=20000
WHERE f_income IS NULL;
```

The changes made by Queries 9.31 and 9.32 can be observed in Figure 9.25, which is obtained by the query: `SELECT * FROM student;`

```
+--------+------------+--------+------------+------------+----------+
| adrn_no | name       | gender | dob        | course     | f_income |
+--------+------------+--------+------------+------------+----------+
|   1001 | Alok       | M      | 1998-10-02 | Science    |    25000 |
|   1002 | Nike       | M      | 1998-11-26 | Science    |    35000 |
|   1003 | Bharath    | M      | 1999-01-01 | Commerce   |    45000 |
|   1004 | Virat      | M      | 1998-12-05 | Science    |    23000 |
|   1005 | Meera      | F      | 1998-08-15 | Science    |    20000 |
|   1006 | Divakar    | M      | 1998-02-21 | Science    |    20000 |
|   1025 | Kaushi     | M      | 1998-10-02 | Commerce   |    27000 |
|   1026 | Niveditha  | F      | 1999-03-04 | Humanities |    52000 |
|   1027 | Sreekumar  | M      | 1998-06-06 | Science    |    16000 |
|   1057 | Chaithanya | F      | 1999-06-03 | Science    |    20000 |
+--------+------------+--------+------------+------------+----------+
10 rows in set (0.00 sec)
```

*Fig. 9.25: Modifications in column f_income*

**A** Let us do

With respect to the table ***stock,*** write SQL statements to make some modifications in some column values.

## 9.8 Changing the structure of a table

In some situations, we may need to modify the structure of tables. It is an operation related to the schema and hence SQL provides a DDL command **ALTER TABLE** to modify the structure of a table. The alteration will be in the form of adding or dropping column(s), changing the data type and size of existing column(s), renaming a table, etc. Let us see how these changes are incorporated.

### 9.8.1 Adding a new column

One or more columns can be added at any position in an existing table. The syntax for adding a new column to a table is:

```
ALTER TABLE <table_name>
ADD <column_name> <data_type>[<size>] [<constraint>]
 [FIRST | AFTER <column_name>];
```

Here the <table_name> indicates the name of the table which is to be altered; **ADD** is the keyword to add the new column; <column_name> <data_type> [<size>] indicates the description of the new column. **FIRST** | **AFTER** indicates the position of the newly added column.

If we want to add a column at the first position, use the clause FIRST. If we want to add a column at a specified position, use the clause AFTER <column_name>. Note that if we do not specify the position of the new column, then by default, it will be added as the last column of the table. Remember that the newly added column contains only NULL values.

Let us add two columns **gr_mks** and **reg_no** in the table ***student*** to hold the grace marks awarded to the students and their register number for the Higher Secondary examinations, respectively. Query 9.33 can make these changes in the structure of the table.

Query9.33
```
ALTER TABLE student
ADD gr_mks INTEGER AFTER dob,
ADD reg_no INTEGER;
```

The response of MySQL for this query will be as follows:

```
Query OK, 10 rows affected (0.25 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

### 9.8.2 Changing the definition of a column

We can modify the characteristics of a column like data type, size and/or constraints by using the clause **MODIFY** with ALTER TABLE command. The syntax to modify the definition of a column is:

```
ALTER TABLE <table_name>
MODIFY <column_name> <data_type>[<size>] [<constraint>];
```

Let us modify the newly added column reg_no by providing a constraint **UNIQUE** to ensure that no two students have the same register number. Query 9.34 makes this change.

Query9.34
```
ALTER TABLE student
MODIFY reg_no INTEGER UNIQUE;
```

The response of MySQL will be the same as that we obtained for Query 9.30. We can observe the changes made by the Queries 9.33 and 9.34 using the command: DESC student;. The output is shown in Figure 9.26.

```
+-----------+---------------+------+-----+---------+----------------+
| Field     | Type          | Null | Key | Default | Extra          |
+-----------+---------------+------+-----+---------+----------------+
| adm_no    | int(11)       | NO   | PRI | NULL    | auto_increment |
| name      | varchar(20)   | NO   |     | NULL    |                |
| gender    | char(1)       | YES  |     | M       |                |
| dob       | date          | YES  |     | NULL    |                |
| gr_mks    | int (11)      | YES  |     | NULL    |                |
|           |               |      |     |         |                |
|  course   | varchar(15)   | YES  |     | NULL    |                |
|  f income | int(11)       | YES  |     | NULL    |                |
   reg_no     int(11)         YES    UNI   NULL
+-----------+---------------+------+-----+---------+----------------+
8 rows in set (0.00 sec)
```

*Fig. 9.26: Structure of table student after alteration*

We can see from Figure 9.26 that column `gr_mks` is added after `dob` and `reg_no` as the last column. Also note that column `Key` for `reg no` contains **UNIQUE** constraint. Now, if we see the contents in the new columns using the command `SELECT * FROM student;`, we can see `NULL` values as shown in Figure 9.27.

```
+--------+--------------+--------+------------+--------+------------+----------+--------+
| adm no | name         | gender | dob        | gr_mks | course     | f_income | reg_no |
+--------+--------------+--------+------------+--------+------------+----------+--------+
|   1001 | Alok         | M      | 1998-10-02 |   NULL | Science    |    25000 |   NULL |
|   1002 | Nike         | M      | 1998-11-26 |   NULL | Science    |    35000 |   NULL |
|   1003 | Bharath      | M      | 1999-01-01 |   NULL | Commerce   |    45000 |   NULL |
|   1004 | Virat        | M      | 1998-12-05 |   NULL | Science    |    23000 |   NULL |
|   1005 | Meera        | F      | 1998-08-15 |   NULL | Science    |    20000 |   NULL |
|   1006 | Divakar      | M      | 1998-02-21 |   NULL | Science    |    20000 |   NULL |
|   1025 | Kaushi       | M      | 1998-10-02 |   NULL | Commerce   |    27000 |   NULL |
|   1026 | Niveditha    | F      | 1999-03-04 |   NULL | Humanities |    52000 |   NULL |
|   1027 | Sreekumar    | M      | 1998-06-06 |   NULL | Science    |    16000 |   NULL |
    1057   Chaithanya     F        1999-06-03     NULL   Science        20000     NULL
+--------+--------------+--------+------------+--------+------------+----------+--------+
10 rows in set (0.00 sec)
```

*Fig. 9.27: Contents in the new columns after modifying the structure of table student*

Write SQL statements to fill in data in the new columns with proper values and display the changes in the records.

Make some structural modifications in the table ***stock*** and store values according to the changes.

### 9.8.3 Removing column from a table

If we want to remove an existing column from a table, we can use **DROP** clause along with ALTER TABLE command. The syntax for removing a column from a table is:

```
ALTER TABLE <table-name>
DROP <column-name>;
```

For example, if we want to remove the column `gr_mks` from the table *student*, Query 9.35 is sufficient.

Query9.35

```
ALTER TABLE student
DROP gr_mks;
```

The response of MySQL will be the same as we received for the earlier ALTER TABLE statements. The change can be observed using the command: DESC student;

### 9.8.4 Renaming a table

We can rename a table in the database by using the clause **RENAME TO** along with ALTER TABLE command. We can rename a table even though it contains tuples. The syntax for renaming a table is:

```
ALTER TABLE <table_name>
RENAME TO <new_table_name>;
```

For example, if we want to rename the table *student* as ***student2015***, Query 9.36 can help us.

Query9.36

```
ALTER TABLE student
RENAME TO student2015;
```

The response of this query will be as follows:

```
Query OK, 0 rows affected (0.06 sec)
```

```
+------------------+
| Tables_in_school |
+------------------+
| student2015      |
+------------------+
1 row in set (0.00 sec)
```

*Fig. 9.28: Renaming of table*

The change can be viewed using the query: SHOW TABLES; and the output will be as shown in Figure 9.28.

## 9.9 Deleting rows from a table

Sometimes we need to remove one or more records from the table. The DML command **DELETE** is used to remove individual or a set of rows from a table. The rows which are to be deleted are selected by using the WHERE clause. If the WHERE clause is not used, all the rows in the table will be deleted. The DELETE command removes only records and not the individual field values. The syntax of the DELETE command is:

```
DELETE FROM <table_name> [WHERE <condition>];
```

For example, if the record of *Sreekumar* with admission number *1027* is to be deleted from the table, Query 9.37 can be used.

```
DELETE FROM student2015 WHERE adm_no=1027;
```

The output of this query will be: **Query OK, 1 row affected (0.08 sec)**

Note that the name of the table is specified as *student2015*, because we have renamed the table *student* as *student2015*. The change in the table can be observed in Figure 9.29, which is obtained by the query: SELECT * FROM student2015;

```
+--------+----------------+--------+-------------+-------------+----------+--------+
| adm_no | name           | gender | dab         | course      | f_income | reg_no |
+--------+----------------+--------+-------------+-------------+----------+--------+
|   1001 | Alok           | M      | 1998-10-02  | Science     |    25000 |  NULL  |
|   1002 | Nike           | M      | 1998-11-26  | Science     |    35000 |  NULL  |
|   1003 | Bharath        | M      | 1999-01-01  | Commerce    |    45000 |  NULL  |
|   1004 | Virat          | M      | 1998-12-05  | Science     |    23000 |  NULL  |
|   1005 | Meera          | F      | 1998-08-15  | Science     |    20000 |  NULL  |
|   1006 | Divakar        | M      | 1998-02-21  | Science     |    20000 |  NULL  |
|   1025 | Kaushi         | M      | 1998-10-02  | Commerce    |    27000 |  NULL  |
|   1026 | Niveditha      | F      | 1999-03-04  | Humanities  |    52000 |  NULL  |
|   1057 | Chaithanya     | F      | 1999-06-03  | Science     |    20000 |  NULL  |
+--------+----------------+--------+-------------+-------------+----------+--------+
9 rows in set (0.02 sec)
```

*Fig. 9.29: Contents in the renamed table after deleting a record*

Figure 9.29 shows only seven columns. This is because of the removal of column `gr_mks` as a result of Query 9.35.

The deletion of values in a column is not possible. The effect may be brought out by filling the column with `NULL` values using `UPDATE` command as follows:

```
UPDATE <table name>
SET <column name>= NULL
[WHERE <condition>];
```

## 9.10 Removing table from a database

If we do not need a table, it can be removed from the database using **DROP TABLE** command. This DDL command removes a table from the database permanently even though the table contains tuples. We have to be very careful while deleting any existing table because once data is lost, it cannot be recovered later. The syntax to remove a table is:

```
DROP TABLE <table_name>;
```

For example, if we want to remove the table *student2015* from the database *school*, Query 9.38 can be used.

**Query 9.38**
```
DROP TABLE student2015;
```

### Know your progress

1. The command used to edit the structure of a table is ________.
2. Restructuring of a column affects the values in it. State whether true or false.
3. How will you remove a column from a table?
4. Name the command used to remove a row from a table.
5. What happens when we use `DELETE FROM` command without a `WHERE` clause?

## 9.11 Nested queries

We are all familiar with the concept of nesting as we discussed it in nested if, nested loop, etc. Now, we will discuss nested queries. We know that nested means one inside another. Here the result of one query is dynamically substituted in the condition of another. A MySQL inner query is also called sub query, while the query that contains the sub query is called an outer query. SQL first evaluates the inner query (sub query) within the WHERE clause and the result of inner query is then substituted in the condition of the outer query. While using relational operators, ensure that the sub query (inner query) returns a single row output.

Suppose we want to display the name and course of the students who have the highest family monthly income. Query 9.39 can perform this task and Figure 9.30 shows the output.

Query9.39

```
SELECT  name,  course  FROM  student2015
WHERE  f_income= ( SELECT  MAX (f_income)
                   FROM  student2015);
```

In this example, the sub query (here query inside the pair of parentheses) is evaluated first and the highest value in f_income column (here 52000) is returned. This value is then compared with the value in f_income attribute of the records in the table and when a match is found, it is retrieved by the outer query.

```
+-----------+------------+
| name      | course     |
+-----------+------------+
| Niveditha | Humanities |
+-----------+------------+
1 row in set (0.03 sec)
```

Fig. 9.30: Result of nested query

## 9.12 Concept of views

MySQL supports the concept of views, which is a feature of RDBMS. A *view* is a virtual table that does not really exist in the database, but is derived from one or more tables. The table(s) from which the tuples are collected to constitute a view is called base table(s). These tuples are not physically stored anywhere, rather the definition of the view is stored in the database. Views are just like windows through which we can view the desired information that is actually stored in a base table. The contents of a view are taken from other tables depending upon a query condition. A view can be created with the DDL command CREATE VIEW. The syntax is:

```
CREATE  VIEW  <view_name>
AS  SELECT  <column_name1>  [, <column_name2], ... ]
FROM  <table_name>
[WHERE  <condition>] ;
```

Let us create a view that contains the details of only those students who were born before January 1, 1999. Query 9.40 creates this view named ***student1998***.

Query9.40
```
CREATE  VIEW  student1998
AS  SELECT*  FROM  student2015
WHERE  dob  <  '1999-1-1';
```

The output of this query will be: **Query OK, 0 rows affected (0. 31 sec)**

Figure 9.31 shows the structure of the view and Figure 9.32 shows the contents of the view. These can be obtained by the commands `DESC student1998;` and `SELECT * FROM student1998;` respectively.

```
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| adm no    | int (11)    | NO   |     | 0       |       |
| name      | varchar(20) | NO   |     | NULL    |       |
| gender    | char(1)     | YES  |     | M       |       |
| dob       |     date    |      |     |   YES   |       |
| NULL course     varchar(15)   YES|     | NULL    |       |
| f income  |  int(11)    |  YES |     | NULL    |       |
| reg_no    | int(11)     | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

*Fig. 9.31: Structure of the view student1998*

```
+--------+---------+--------+------------+------------+----------+--------+
| adm_no | name    | gender | dob        | course     | f_income | reg_no |
+--------+---------+--------+------------+------------+----------+--------+
|   1001 | Alok    | M      | 1998-10-02 | Science    |    25000 |  NULL  |
|   1002 | Nike    | M      | 1998-11-26 | Science    |    35000 |  NULL  |
|   1004 | Virat   | M      | 1998-12-05 | Science    |    23000 |  NULL  |
|   1005 | Meera   | F      | 1998-08-15 | Science    |    20000 |  NULL  |
|   1006 | Divakar | M      | 1998-02-21 | Science    |    20000 |  NULL  |
|   1025 | Kaushi  | M      | 1998-10-02 | Commerce   |    27000 |  NULL  |
+--------+---------+--------+------------+------------+----------+--------+
6 rows in set (0.00 sec)
```

*Fig. 9.32: Contents in the view student1998*

Note that in Figure9.31, the column Key does not contain the constraints of adm_no and reg_no, and Figure 9.32 shows the details of students born in the year 1998 only.

We can use all the DML commands with the view in quite a similar fashion as in the case of tables. Figure 9.32is a result of such a query. Remember that we are accessing the base table though the associated views. So, take care when operations like `update` and `delete` are performed on views as they actually reflect in the base tables. Query 9.41 illustrates this concept.

Query9.41
```
UPDATE  student1998
SET  reg_no=2201020
WHERE  adm_no=1001;
```

Let us check the data in the table '***student2015***' after executing this query. Figure 9.33 shows the output of the query: `SELECT * FROM student2015;`

```
+--------+------------+--------+------------+------------+----------+---------+
| adm_no | name       | gender | dob        | course     | f_income | reg_no  |
+--------+------------+--------+------------+------------+----------+---------+
|   1001 | Alok       | M      | 1998-10-02 | Science    |    25000 | 2201020 |
|   1002 | Nike       | M      | 1998-11-26 | Science    |    35000 |    NULL |
|   1003 | Bharath    | M      | 1999-01-01 | Commerce   |    45000 |    NULL |
|   1004 | Virat      | M      | 1998-12-05 | Science    |    23000 |    NULL |
|   1005 | Meera      | F      | 1998-08-15 | Science    |    20000 |    NULL |
|   1006 | Divakar    | M      | 1998-02-21 | Science    |    20000 |    NULL |
|   1025 | Kaushi     | M      | 1998-10-02 | Commerce   |    27000 |    NULL |
|   1026 | Niveditha  | F      | 1999-03-04 | Humanities |    52000 |    NULL |
|        |            |        |            |            |          |         |
|   1057 | Chaithanya | F      | 1999-06-03 | Science    |    20000 |    NULL |
+--------+------------+--------+------------+------------+----------+---------+
9 rows in set (0.00 sec)
```

*Fig. 9.33: Effect of updation in the base table through the view*

The advantages of views lie in the fact that without sparing extra storage space, we can use the same table as different tables (but virtual). Another advantage is that the views implement sharing along with privacy. It also helps to reduce the complexity of conditions with `WHERE` clause while retrieving, updating or deleting records from tables.

If a view is no longer needed, it can be removed from the database with **DROP VIEW** command. But it will not affect the base table(s). The syntax is:

```
    DROP VIEW <view name>;
```

For example, if we want to delete the view *student1998* from the database, Query 9.42 can be used.

```
                DROP VIEW student1998;
```

## Know your progress

1. What is mean by nested query?
2. What is view in SQL?
3. Creation of a view needs a `SELECT` command. State whether true or false.
4. Removal of a view deletes a table from the database. State whether true or false.
5. Can we update a table using its view?

**Let us do**

We have discussed all the DDL and DML commands of SQL, which are implemented through MySQL, with examples and their outputs. Let us summarise them in Table 9.8. The first two rows are filled in and the remaining rows are left to you.

| SQL | Command | Essential Keyword(s) | Purpose of the command |
|---|---|---|---|
| DDL | CREATE TABLE | | To create tables |
| | ALTER TABLE | ADD/MODIFY | To change the structure of tables |
| | DROP TABLE | | |
| | CREATE VIEW | | |
| | DROP VIEW | | |
| | INSERT | | |
| DML | SELECT | | |
| | UPDATE | | |
| | DELETE | | |

*Table 9.8: Summary of SQL command*

## Let us conclude

Structured Query Language is used to operate on relational database. MySQL is a popular RDBMS package by which we can implement SQL commands. We have used various DDL commands to perform operations related with the structure of database. Constraints have been presented to ensure data validity and integrity in database. We have also discussed different DML commands to perform operations associated with the data contained in tables of a database. These operations include insertion, retrieval, modification and deletion of data in tables. We have also familiarized ourselves with nested queries and the concept of views. A good understanding of this chapter is essential for your higher studies in the field of computers.

## Let us practice

1. The structure of a table is given to store the details of marks scored by students in an examination.

| Data | Type | Description |
|---|---|---|
| Register number | Numeric | A unique and essential data to identify a student |
| Name | String | A maximum of 30 characters |
| Course | String | It can be Science, Commerce or Humanities |
| Marks of six subjects | Numeric each | Six separate columns are required |

Write SQL statements for the creation of the table and the following requirements:

a. Insert data into the fields (at least 10 records).

b. Display the details of all students.

c. List the details of Science group students.

d. Count the number of students in each course.

e. Add a new column named Total to store the total marks.

f. Fill the column Total with the sum of the six marks of each student.

g. Display the highest total in each group.

h. Find the highest, lowest and average score in Subject 6 in Commerce group.

i. Display the names in the alphabetical order in each course.

j. Display the name of the student with the highest total.

2. The structure of a table is given to store the details of items in a computer shop.

| Data | Type | Description |
|---|---|---|
| Item number | Numeric | A unique and essential data to identify an item |
| Item name | String | A maximum of 30 characters |
| Date of purchase | Date | Duplication is allowed |
| Unit price | Fractional Number | Price of a single item |
| Quantity | Numeric | Number of items |
| Manufacturer | String | Name of supplier (can duplicate) |

Write SQL statements for the creation of the table and the following requirements:

a. Insert data into the fields (at least 10 records).

b. Display the details of all items in the table.

c. Display the names of items and total price of each.

d. List the items manufactured by a company *(specify the name)* avialable in the table.

e. Find the number of items from each manufacturer.

f. Display the details of items with the highest price.

g. List the names of items whose price is more than the average price of all the items.

h. Display the names of items purchased after 1-1-2015.

i. Get the details of items manufactured by two or three companies *(specfy the names)* avialable in the table.

j. Display the details of items from a company *(specfy the name)* with a stock of more than 20 pieces.

3. The structure of a table is given to store the details of higher secondary school teachers.

| Data | Type | Description |
|---|---|---|
| Teacher ID | Numeric | A unique and essential data to identify a teacher |
| Name | String | A maximum of 30 characters |
| Gender | Character | Male or Female |
| Date of joining | Date | Duplication is allowed |
| Department | String | Science, Commerce, Humanities or Language |
| Basic pay | Numeric | Basic salary of a teacher |

Write SQL statements for the creation of the table and the following requirements:

a. Insert data into the fields (at least 10 records).

b. Display the details of all female teachers in the table.

c. List the details of male teachers in the Science department.

d. Display the names and basic pay of teachers in the Language department whose basic pay is Rs. 21000 /- or more.

e. Display the names and 71% of basic pay of the teachers.

f. Find the number of teachers in each department.

g. Display the details of teachers whose basic pay is less than the average basic pay.

h. List the male teachers who joined before 1-1-2010.

i. Increment the basic pay of all teachers by Rs. 1000/-.

j. Delete the details of teachers from the Language department.

4. The structure of a table is given to store the details of customers in a bank.

| Data | Type | Description |
|---|---|---|
| Account number | Numeric | A unique and essential data to identify a customer |
| Name | String | A maximum of 30 characters |
| Gender | Character | Male or Female |
| Date of joining | Date | Duplication is allowed |
| Type of account | String | SB or Current |
| Balance amount | Numeric | Can be a fractional number |

Write SQL statements for the creation of the table and the following requirements:

a. Insert data into the fields (at least 10 records).

b. Display the details of customers having SB account.

c. Display the names of customers with a balance amount greater than Rs. 5000/-.

d. Display the details of female customers with a balance amount greater than Rs. 10000/-.

e. Count the number of male and female customers.

f. Display the names of customers with the highest balance amount.

g. Display the names of customers whose names end with 'kumar'.

h. Update the balance amount of a particular customer with a deposit amount of Rs. 2000/-.

i. Display the details of customers with a tax deduction of 2% of the balance amount for those who have Rs.2,00,000/- in their account.

j. Delete the details of customers with current account.

## Let us assess

1. The command to remove rows from a table 'CUSTOMER' is:

   a. REMOVE FROM CUSTOMER  b. DROP TABLE CUSTOMER

   c. DELETE FROM CUSTOMER  d. UPDATE CUSTOMER

2. If values for some columns are unknown, how is a row inserted?

3. Distinguish between CHAR and VARCHAR data types of SQL.

4. What is the difference between PRIMARY KEY and UNIQUE constraints?

5. What do you mean by NULL value in SQL?

6. Which of the following is the correct order of clauses for the SELECT statements?

   a. SELECT, FROM, WHERE, ORDER BY

   b. SELECT, FROM, ORDER BY, WHERE

   c. SELECT, WHERE, FROM, ORDER BY

   d. SELECT, WHERE, ORDER BY, FROM

7. The SQL operator _________ ls used with pattern matching.

8. Read the following strings:

   (i) 'Sree Kumar' (ii) 'Kumaran' (iii) 'Kumar Shanu' (iv) 'Sreekumar'

   Choose the correct option that matches with the pattern '%Kumar', when used with LIKE operator in a SELECT statement.

   a. Strings (i) and (ii) only

   b. Strings (i), (iii) and (iv) only

   c. Strings (i) and (iii) only

   d. All the strings

9. List any five built-in functions ofSQL and the value returned by each.

10. Distinguish between WHERE clause and HAVING clause.

11. Write any four DML commands in SQL.

12. Write the essential clause required for each of the following SQL command.

    a. INSERT INTO b. SELECT c. UPDATE

13. Consider the given table Customer and write the output ofthe following SQL querles:

| AccNo | Name | Branch | Amount |
|---|---|---|---|
| 1001 | Kumar | Calicut | 10000 |
| 1002 | Salim | Trivandrum | 20000 |
| 1003 | Fida | Kottayam | 18000 |
| 1004 | John | Kannur | 30000 |
| 1005 | Raju | Thrissur | 5000 |

a. `SELECT * FROM customer WHERE Amount>25000;`

b. `SELECT Name FROM customer`
   `WHERE Branch IN ('Calicut, 'Kannur');`

c. `SELECT COUNT(*) FROM customer WHERE Amount< 20000;`

d. `SELECT Name FROM customer WHERE Name LIKE "%m%";`

e. `SELECT* FROM customer ORDER BY Amount DESC;`

14. Distinguish between COUNT(*) and COUNT (column-name).

15. Consider the given table ITEMS.

| Item Code | Name | Category | Unit Price | Sales Price |
|---|---|---|---|---|
| 0001 | Pencil Pen | Stationery | 5.00 | 8.00 |
| 0002 | NoteBook | Stationery | 8.00 | 10.00 |
| 0003 | Chappal | Stationery | 10.00 | 20.00 |
| 0004 | Apple | Footwear | 50.00 | 70.00 |
| 0005 | Orange | Fruits | 60.00 | 90.00 |
| 0006 | Pen | Fruits | 40.00 | 60.00 |
| 0007 |  | Stationery | 10.00 | 12.00 |
|  |  |  |  |  |

a. Suggest a suitable primary key for the above table. Give justification.

b. Write SQL statements for the following:

i. To list all stationery items.

ii. To list itemcode, name and profit of all items.

iii. To count the number of items in each category.

iv. To list all stationery items in the descending order of their unit price.

v. To find the item with the highest selling price.

vi. To create a view that contains the details of all stationery items.

16. What are the different modifications that can be made on the structure of a table? Which is the SQL command required for this? Specify the clauses needed for each type of modification.

17. A table is created in SQL with 10 records. Which SQL command is used to change the values in a column of specified rows? Write the format.

18. Name the keyword used with SELECT command to avoid duplication of values in a column.

19. Distinguish between DISTINCT and UNIQUE in SQL.

20. Pick the odd one out and give reason:

a. CREATE b. SELECT c. UPDATE d. INSERT