MRCET CAMPUS

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(AUTONOMOUS INSTITUTION – UGC, GOVT. OF INDIA)

B.TECH (R-22 Regulation)
(II YEAR – I SEM)
(2023-24)

DATABASE MANAGEMENT SYSTEMS

(R22A0584)

LAB MANUAL

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

(Autonomous Institution – UGC, Govt. of India)

Recognized under 2(f) and 12(B) of UGC ACT 1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE-Accredited by NBA & NAAC – 'A' Grade - ISO 9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad–500100, Telangana State, India

Department of Computer Science and Engineering

EMERGING TECHNOLOGIES

DATABASE MANAGEMENT SYSTEM LAB

(R22A0584)

LAB MANUAL

Prepared by

CH NAVEEN KUMAR, Asst.Prof.

On

12.09.2023

Department of Computer Science and Engineering

EMERGING TECHNOLOGIES

Vision

- "To be at the forefront of Emerging Technologies and to evolve as a Centre of Excellence in Research, Learning and Consultancy to foster the students into globally competent professionals useful to the Society."

Mission

The department of CSE (Emerging Technologies) is committed to:

- To offer highest Professional and Academic Standards in terms of Personal growth and satisfaction.
- Make the society as the hub of emerging technologies and thereby capture opportunities in new age technologies.
- To create a benchmark in the areas of Research, Education and Public Outreach.
- To provide students a platform where independent learning and scientific study are encouraged with emphasis on latest engineering techniques.

QUALITY POLICY

- To pursue continual improvement of teaching learning process of Undergraduate and Post Graduate programs in Engineering & Management vigorously.
- To provide state of art infrastructure and expertise to impart the quality education and research environment to students for a complete learning experience.
- Developing students with a disciplined and integrated personality.
- To offer quality relevant and cost-effective programs to produce engineers as per requirements of the industry need.

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY
Maisammaguda, Dhulapally Post, Via Hakimpet, Secunderabad – 500100

DEPARTMENT OF EMERGING TECHNOLOGIES

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to thestarting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared tothe lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the labsession.
b. Laboratory Record updated up to the last session experiments and other utensils (ifany) needed in the lab.
c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the labobservation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems,which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system /seat is kept properly.

Head of the Department **Lab Incharge**

INDEX

INTRODUCTION

Types of database models:

Hierarchical Model

This model is like a hierarchical tree structure, used to construct a hierarchy of records in the form of nodes and branches. The data elements present in the structure have Parent-Child relationship. Closely related information in the parent-child structure is stored together as a logical unit. A parent unit may have many child units, but a child is restricted to have only one parent.

The drawbacks of this model are:

- The hierarchical structure is not flexible to represent all the relationship proportions, which occur in the real world.
- It cannot demonstrate the overall data model for the enterprise because of the non-availability of actual data at the time of designing the data model.
- It cannot represent the Many-to-Many relationship.

Network Model

- It supports the One-To-One and One-To-Many types only. The basic objects in this model are Data Items, Data Aggregates, Records and Sets.
- It is an improvement on the Hierarchical Model. Here multiple parent-child relationships are used. Rapid and easy access to data is possible in this model due to multiple access paths to the data elements.

Relational Model

- Does not maintain physical connection between relations Data is organized in terms of rows and columns in a table
- The position of a row and/or column in a table is of no importance The intersection of a row and column must give a single value

Features of an RDBMS

- The ability to create multiple relations and enter data into them An attractive query language

Retrieval of information stored in more than one table

- An RDBMS product has to satisfy at least Seven of the 12 rules of Codd to beaccepted as a full- fledged RDBMS.

Relational Database Management System

RDBMS is acronym for Relation Database Management System. Dr. E. F. Codd first introduced the Relational Database Model in 1970. The Relational model allows data to be represented in a simple row- column. Each data field is considered as a column and each record is considered as a row. Relational Database is more or less similar to Database Management System. In relational model there is relation between their data elements. Data is stored in tables. Tables have columns, rows and names. Tables can be related to each other if each has a column with a common type of information. The most famous RDBMS packages are Oracle, Sybase and Informix.

Simple example of Relational model is as follows:

Student Details Table

Roll_no	Sname	S_Address
1	Rahul	Satelite
2	Sachin	Ambawadi
3	Saurav	Naranpura

Student Marksheet Table

Rollno	Sub1	Sub2	Sub3
1	78	89	94
2	54	65	77
3	23	78	46

Here, both tables arebased on students details. Common field in both tables is Rollno.

So wecan say both tables are related with each other through Rollno column.

Degree of Relationship

One to One (1:1)
One to Many or Many to One (1:M / M: 1)
Many to Many (M: M)

The Degree of Relationship indicates the link between two entities for a specified occurrence of each. **One to One Relationship: (1:1) Student Has Roll No.**

One student has only one Rollno. For one occurrence of the first entity, there can be, at the most one related occurrence of the second entity, and vice-versa.

One to Many or Many to One Relationship: (1:M/M: 1)

1 :M

Course Contains Students

As per the Institutions Norm, One student can enroll in one course at a time however, in one course, there can be more than one student.

For one occurrence of the first entity there can exist many related occurrences of the second entity and for every occurrence of the second entity there exists only one associated occurrence of the first.

Many to Many Relationship: (M:M)

Students Appears Tests

The major disadvantage of the relational model is that a clear-cut interface cannot be determined. Reusability of a structure is not possible. The Relational Database now accepted model on which major database system are built.

Oracle has introduced added functionality to this by incorporated object-oriented capabilities. Now it is known is as Object Relational Database Management System (ORDBMS). Object- oriented concept is added in Oracle8.

Some basic rules have to be followed for a DBMS to be relational. They are known as Codd's rules, designed in such a way that when the database is ready for use it encapsulates the relational theory to its full potential. These twelve rules are as follows.

E. F. Codd Rules

1. The Information Rule

All information must be store in table as data values.3

2. The Rule of Guaranteed Access

Every item in a table must be logically addressable with the help of a table name.

3. The View Updating Rule

All views that are theoretically updatable are also updatable by the system.

4. The Insert and Update Rule

This rule indicates that all the data manipulation commands must be operational on sets of rows having a relation rather than on a single row.

5. The Physical Independence Rule

Application programs must remain unimpaired when any changes are made in storagerepresentation or access methods.

6. The Logical Data Independence Rule

The changes that are made should not affect the user's ability to work with thedata.The change can be splitting table into many more tables.

7. The Integrity Independence Rule

The integrity constraints should store in the system catalog or in the database.

8. The Distribution Rule

The system must be access or manipulate the data that is distributed in other systems.

9. The Non-subversion Rule

If a RDBMS supports a lower level language then it should not bypass any integrity constraints defined in the higher level.

What is MYSQL

MySQL is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company.

MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

It allows us to implement database operations on tables, rows, columns, and indexes.

It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.

It provides the Referential Integrity between rows or columns of various tables.

It allows us to updates the table indexes automatically.

It uses many SQL queries and combines useful information from multiple tables for the end-users

MySQL is named after the daughter of co-founder Michael Widenius whose name is "My".

To communicate with Oracle, mysql supports the following categories of commands:

1. **Data Definition Language**

Create, Alter, Drop and Truncate

2. **Data Manipulation Language**

Insert, Update, Delete and Select

3. **Transaction Control Language**

Commit, Rollback and Save point

4. **Data Control Language**

Grant and Revoke

MySQL uses many different data types broken into three categories −

- Numeric
- Date and Time
- String Types.

Numeric Data Types

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you.

The following list shows the common numeric data types and their descriptions −

INT − A normal-sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.

TINYINT − A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 4 digits.

SMALLINT − A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits.

MEDIUMINT − A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify a width of up to 9 digits.

BIGINT − A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits.

- **FLOAT(M,D)** − A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.
- **DOUBLE(M,D)** − A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 places for a DOUBLE. REAL is a synonym for DOUBLE.
- **DECIMAL(M,D)** − An unpacked floating-point number that cannot be unsigned. In the unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for DECIMAL.

Date and Time Types

The MySQL date and time datatypes are as follows −

- **DATE** − A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31. For example, December 30th, 1973 would be stored as 1973-12-30.
- **DATETIME** − A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 1973-12-30 15:30:00.
- **TIMESTAMP** − A timestamp between midnight, January 1st, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000 (YYYYMMDDHHMMSS).
- **TIME** − Stores the time in a HH:MM:SS format.
- **YEAR(M)** − Stores a year in a 2-digit or a 4-digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be between 1970 to 2069 (70 to 69). If the length is specified as 4, then YEAR can be 1901 to 2155. The default length is 4.

String Types

Although the numeric and date types are fun, most data you'll store will be in a string format. This list describes the common string datatypes in MySQL.

- **CHAR(M)** − A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- **VARCHAR(M)** − A variable-length string between 1 and 255 characters in length. For example,VARCHAR(25). You must define a length when creating a VARCHAR field.

ROADWAY TRAVELS

Roadway Travels: "Roadway Travels" is in business since 1997 with several buses connecting different places in India. Its main office is located in Hyderabad. The company wants to computerize its operations in the following areas:

- Reservations and Ticketing
- Cancellations

- **Reservations &Cancellation:**

Reservations are directly handled by booking office. Reservations can be made 30 days in advance and tickets issued to passenger. One Passenger/person can book many tickets (to his/her family).

- Cancellations are also directly handed at the booking office.

In the process of computerization of Roadway Travels you have to design and develop a Database which consists the data of Buses, Passengers, Tickets, and Reservation and cancellation details. You should also develop query's using SQL to retrieve the data from the database.

WEEK-1

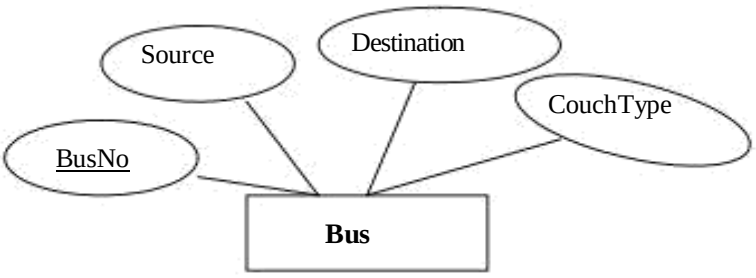
AIM: Analyze the problem and come with the entities in it. Identify what Data has to be persisted in the databases.

The Following are the entities:

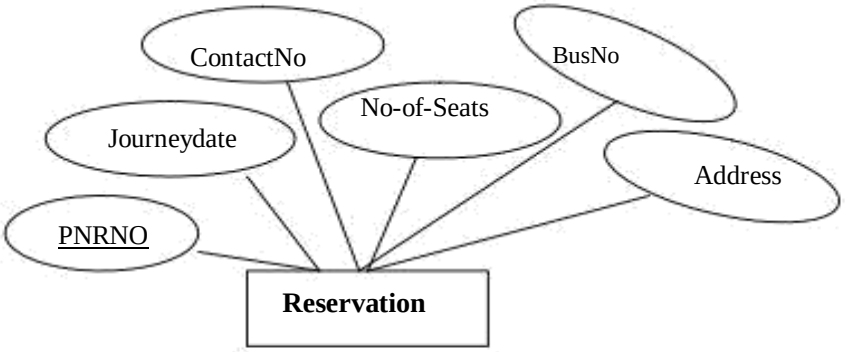
1. **Bus**
2. **Reservation**
3. **Ticket**
4. **Passenger**
5. **Cancellation**

TheattributesintheEntities:

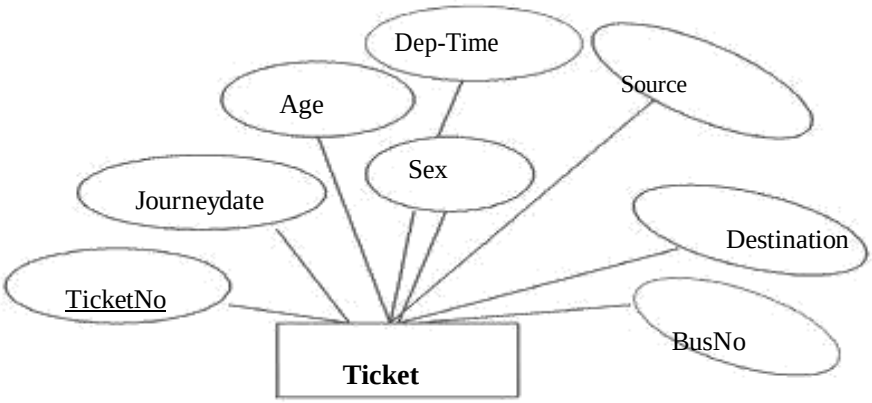
Bus:(Entity)



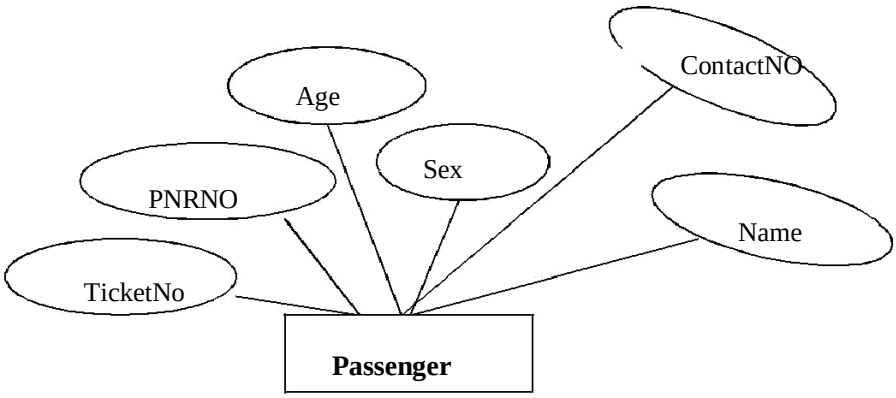
Reservation(Entity)



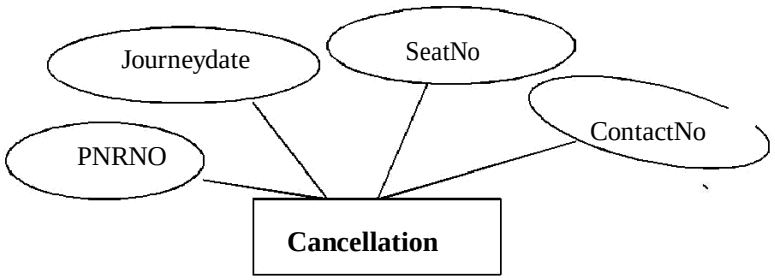
Ticket:(Entity)



Passenger:

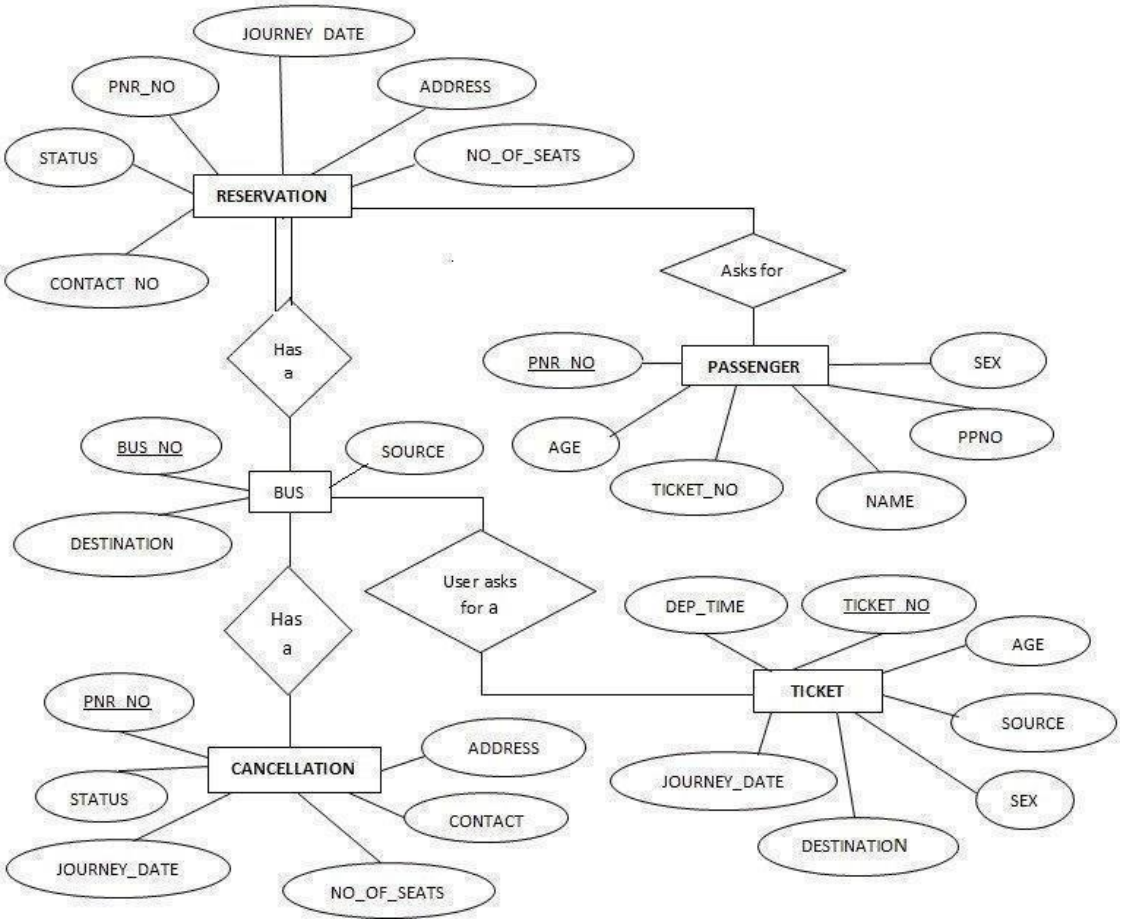


Cancellation(Entity)



WEEK- 2

Concept design with E-R Model:



Signature of the Faculty

WEEK- 3

Relational Model

Represent all entities and all relationships in a tabular fashion

The following are tabular representation of the above entities and relationships

BUS:

COLUMNNAME	DATATYPE	CONSTRAINT
BusNo	varchar2(10)	**PrimaryKey**
Source	varchar2(20)	
Destination	varchar2(20)	
CoachType	varchar2(20)	

Reservation:

COLUMNNAME	DATATYPE	CONSTRAINT
PNRNo	number(9)	**PrimaryKey**
Journeydate	Date	
No-of-seats	integer(8)	
Address	varchar2(50)	
ContactNo	Number(9)	Should be equal to 10 Numbers and not allow Other than numeric
BusNo	varchar2(10)	**Foreign key**
Seatno	Number	

Ticket:

COLUMNNAME	DATATYPE	CONSTRAINT
Ticket_No	number(9)	**PrimaryKey**
Journeydate	Date	
Age	int(4)	
Sex	Char(10)	
Source	varchar2(10)	
Destination	varchar2(10)	
Dep-time	varchar2(10)	
BusNo	Number2(10)	

Passenger

COLUMNNAME	DATATYPE	CONSTRAINT
PNRNo	Number(9)	**PrimaryKey**
TicketNo	Number(9)	Foreignkey
Name	varchar2(15)	
Age	integer(4)	
Sex	char(10)	(Male/Female)
Contactno	Number(9)	Should be equal to 10numbers And not allow other than numeric

Cancellation:

COLUMNNAME	DATATYPE	CONSTRAINT
PNRNo	Number(9)	Foriegn-key
Journey-date	Date	
Seatno	Integer(9)	
Contact_No	Number(9)	Should be equal to 10numbers And not allow other than numeric

WEEK- 4

Normalization

Database normalization is a technique for designing relational database tables to minimize duplication of information and, in so doing, to safeguard the database against certain types oflogical or structural problems, namely data anomalies.

For example, when multiple instances of a given piece of information occur in a table, the possibility exists that these instances will not be kept consistent when the data within the table is updated, leading to a loss of data integrity.

A table that is sufficiently normalized is less vulnerable to problems of this kind, because its structure reflects the basic assumptions for when multiple instances of the same information should be Normalization is a process of converting a relation to be standard form by decomposition a larger relation into smaller efficient relation that depicts a good database design.

• 1NF: A Relation scheme is said to be in 1NF if the attribute values in the relation are atomic.i.e., Mutli –valued attributes are not permitted.

• 2NF: A Relation scheme is said to be in 2NF,iff and every Non-key attribute is fully functionally dependent on

primary Key.

• 3NF: A Relation scheme is said to be in 3NF,iff and does not have transitivity dependencies. A Relation is said to be 3NF if every determinant is a key for each & every functional dependency.

• BCNF: A Relation scheme is said to be BCNF if the following statements are true for eg FD P->Q in set F of FDs that holds for each FD. P->Q in set F of FD's that holds over R. Here P is the subset of attributes of R & Q is a single attribute of R represented by a single instance only.

Normalized tables are:-

mysql> create table Bus(BusNo varchar(20) primary key,Source varchar(20),Destination varchar(20));

mysql>Create table passenger(PPN varchar(15) Primary key,Name varchar(20),Age integer,Sex char,Address varchar(20));

mysql> Create table PassengerTicket(PPN varchar(15) Primary key,TicketNo integer);

mysql> Create table Reservation(PNRNO integer Primary key, JourneyDate DateTime,NoofSeats int,Address varchar(20),Contact No Integer);

mysql> create table Cancellation(PNRNO Integer primary key,JourneyDate DateTime,NoofSeats Integer,Address varchar(20), ContactNo Integer, foreign key(PNRNO) references Reservation(PNRNO));

mysql> Create table Ticket(TicketNo Integer Primary key,JourneyDate DateTime, Age Int(4),Sex char(2),Source varchar(20),Destination varchar(20),DeptTime varchar(2));

VIVA QUESTIONS

1. Explain the need of normalization?
2. What is functional dependency?
3. Explain difference between third normal form and boyce codd normal form?
4. What is PJNF?
5. What is transitivity dependency?

Signature of the Faculty

WEEK-5

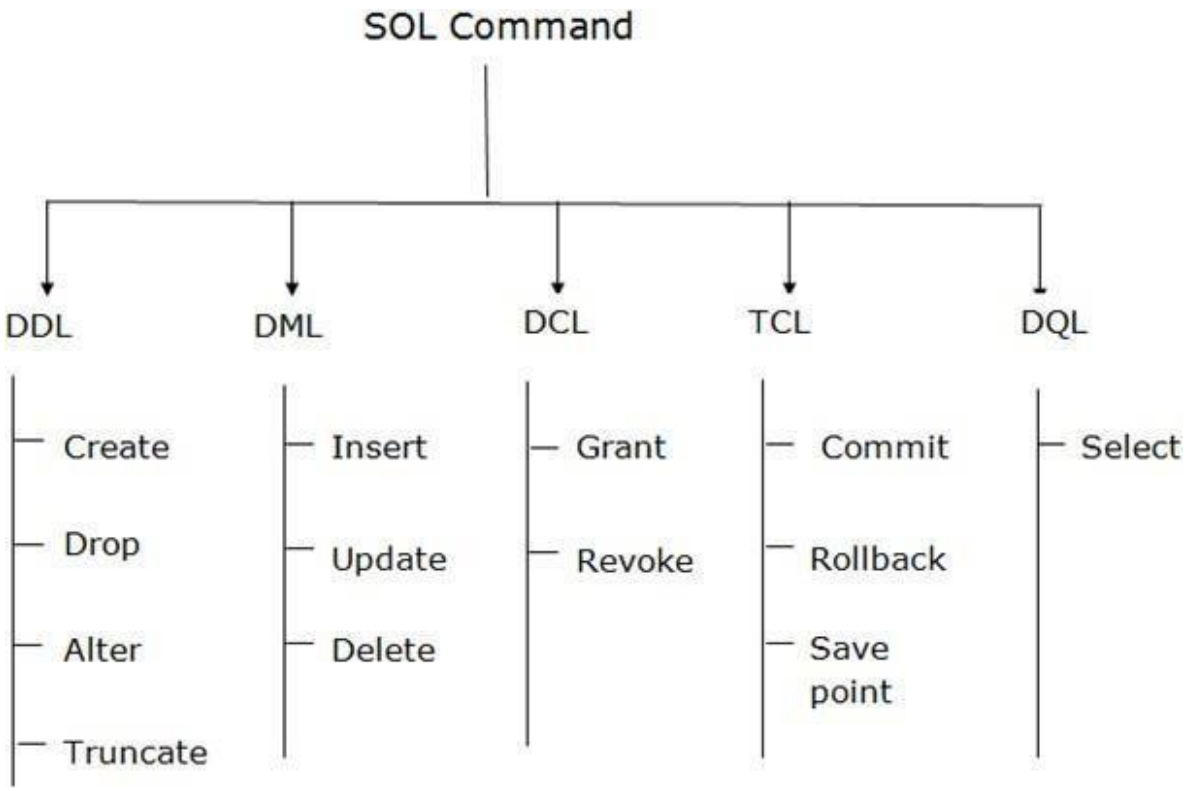
AIM: Practicing DDL Commands.

PRACTICING DDL COMMANDS

Data Definition Language

The data definition language is used to create an object, alter the structure of an object and also drop already created object. The Data Definition Languages used for table definition can be classified into following:

- Create table command
- Alter table command
- Truncate table command
- Drop table command



1. CREATION OF TABLES:

CREATE TABLE:

Table is a primary object of database, used to store data in form of rows and columns. It is created using following command:

Syntax: CREATE TABLE tablename (column_name data_ type constraints, …)

CREATING TABLES:

Example:

mysql> create table Bus (Bus_No varchar(5), source varchar(20), destination varchar(20), daysperweek int);
Table Created.

Above definition will create simple table. Still there are more additional option related with create table for the object-relation feature we will discuss it afterwards.

Desc command:

Describe command is external command of Oracle. The describe command is used to view thestructure of table as follows.

Desc <table name>
mysql> desc Bus**;**

```
Field   Type    Null    Key     Default Extra
Bus_No  varchar(5)      YES             NULL
source  varchar(20)     YES             NULL
destination     varchar(20)     YES             NULL
```

Creating Passenger table:

Mysql>create table passenger(pnrno integer,ticketno integer,name varchar(20),age int,sex char,ppno integer);

Mysql>desc passenger;

```
Field   Type    Null    Key     Default Extra
pnrno   int     YES     MUL     NULL
ticketno        int     YES             NULL
name    varchar(20)     YES             NULL
age     int     YES             NULL
sex     char(1) YES             NULL
ppno    int     YES             NULL
```

Reservation Table:

mysql > create table Reservation (PNR_NO integer(9), No_of_seats integer(8), Address varchar(50), Contact_No Bigint(12), Status varchar(10));

desc Reservation;

```
Field   Type    Null    Key     Default Extra
PNR_NO  int     YES             NULL
No_of_seats     int     YES             NULL
Address varchar(50)     YES             NULL
Contact_No      bigint  YES             NULL
status  varchar(10)     YES             NULL
```

Cancellation Table:

mysql > create table Cancellation (PNR_NO integer (9), No_of_seats integer (8), Address varchar (50), Contact_No integer (12), Status char (3));
Table created.

SQL> desc Cancellation;

```
Field   Type    Null    Key     Default Extra
PNR_NO  int     YES             NULL
No_of_seats     int     YES             NULL
Address varchar(50)     YES             NULL
Contact_No      int     YES             NULL
Status  char(3) YES             NULL
```

Ticket Table:

mysql >create table Ticket(Ticket_No integer(9) primary key, age int, sex char(4) Not null, source varchar(2), destination varchar(20), dep_time varchar(4));
Table created;

mysql > desc Ticket;

```
Field   Type    Null    Key     Default Extra
Ticket_No       int     NO      PRI     NULL
age     int     YES             NULL
sex     char(4) NO              NULL
source  varchar(2)      YES             NULL
destination     varchar(20)     YES             NULL
dep_time        varchar(4)      YES             NULL
```

ALTER TABLE :

To ADD a column:

SYNTAX: ALTER TABLE <TABLE NAME>ADD (<NEW COLUMN NAME><DATA TYPE>(<SIZE>), <NEW COLUMNNAME><DATA TYPE>(<SIZE>) .....);

Example:

mysql > alter table Reservation add column fare integer;

mysql > desc Reservation;

Field	Type	Null	Key	Default	Extra
PNR_NO	int	YES		NULL	
No_of_seats	int	YES		NULL	
Address	varchar(50)	YES		NULL	
Contact_No	int	YES		NULL	
Status	char(3)	YES		NULL	
fare	int	YES		NULL	

To DROP a column:

SYNTAX: ALTER TABLE <TABLE NAME>DROP COLUMN <COLUMN NAME>;

Example:

mysql >alter table Reservation drop column fare ;

mysql > desc Reservation;

Field	Type	Null	Key	Default	Extra
PNR_NO	int	YES		NULL	
No_of_seats	int	YES		NULL	
Address	varchar(50)	YES		NULL	
Contact_No	int	YES		NULL	
Status	char(3)	YES		NULL	

To MODIFY a column:

SYNTAX: ALTER TABLE <TABLE NAME>MODIFY COLUMN <COLUMN NAME> <NEW DATATYPE>(<NEW SIZE>);

Example:

mysql >alter table Reservation modify column status varchar(10);

mysql >desc Reservation;

```
Field   Type    Null    Key     Default Extra
PNR_NO  int     YES             NULL
No_of_seats     int     YES             NULL
Address varchar(50)     YES             NULL
Contact_No      int     YES             NULL
status  varchar(10)     YES             NULL
```

TO ADD FOREIGN KEY TO THE EXISTING TABLE

mysql > ALTER TABLE passenger ADD FOREIGN KEY (pnrno) REFERENCES Reservation (PNR_NO);
Table altered.
Desc passenger;

```
Field   Type    Null    Key     Default Extra
pnrno   int     YES     MUL     NULL
ticketno        int     YES             NULL
name    varchar(20)     YES             NULL
age     int     YES             NULL
sex     char(1) YES             NULL
ppno    int     YES             NULL
```

mysql > ALTER TABLE Cancellation ADD FOREIGN KEY (PNR_NO) REFERENCES Reservation (PNR_NO);

Table altered.

```
Field   Type    Null    Key     Default Extra
PNR_NO  int     YES     MUL     NULL
No_of_seats     int     YES             NULL
Address varchar(50)     YES             NULL
Contact_No      int     YES             NULL
Status  char(3) YES             NULL
```

TRUNCATE TABLE:

If there is no further use of records stored in a table and the structure is required then only data can be deleted using truncate command. Truncate command will delete all the records permanently of specified table as follows.

Truncate table <table name>
EXAMPLE: try your own Query

RENAME A TABLE

Rename command is used to give new names for existing tables.

RENAME table old tablename TO new tablename;

Example:
MYSQL>RENAME table passenger TO Passenger;

VIVA QUESTIONS

1. Define data and information.
2. Define Data base management system.
3. What is SQL?
4. What is the syntax for creating a table?
5. List the components of SQL.
6. Define DDL? What are the DDL commands?
7. List out the uses of alter command.
8. What is Syntax for truncate a table?
9. What is the use drop table command?

Signature of the Faculty

WEEK- 5

AIM: Applying DML commands on Road Way Travels Tables.

DML COMMANDS

1. INSERTING DATA INTO TABLE

Insert command is used to insert rows into the table.

SYNTAX:
INSERT INTO tablename values (columnname1, columnname2,….columnname n)

INSERTION of Data can also be done by the following Syntax:

SYNTAX

INSERT IN TO tablename (columnname1, columnname2,….columnname n)
VALUES(Value1,Value2,..Value n);

Inserting values into "Bus" table:

mysql > insert into Bus values('w1234','hyderabad', 'tirupathi',4);

mysql >insert into Bus values ('p2345','hyderabad', 'Banglore',3);

mysql >insert into Bus values ('9w01','hyderabad','Kolkata',4);

Inserting values into " RESERVATION" table:

mysql >insert into Reservation values(1,2,'masabtank',9009897812,'confirm');
mysql>insert into Reservation values(1,2,'masabtank',9009897812,'confirm');

Inserting values into "PASSENGER" table:

mysql >insert into Passenger values (1, 1,'SACHIN', 12,'m', 1234);
mysql >insert into Passenger values (2, 2,'rahul', 34,'m', 3456);
mysql >insert into Passenger values(3,3,'swetha',24,'f',8734);
mysql >insert into Passenger values(5,5,'Arun',24,'m',7387);
mysql >insert into Passenger values(6,6,'Aruna',25,'f',7389);
mysql >insert into Passenger values(4,3,'rohith',24,'m',734);

UPDATE

This SQL command is used to modify the values in an existing table.

mysql >**UPDATE** tablename **SET** column1= expression1, column2= expression 2,...

WHERE somecolumn=somevalue;

An expression consists of either a constant (new value), an arithmetic or string operation or an SQL query. Note that the new value to assign to <column> mustmatching data type.

An update statement used without a where clause results in changing respective attributes of all tuples in the specified table.

EXAMPLE:

mysql >update Passenger set age='43' where pnrno='2';
TEST OUTPUT:

DELETE

In order to delete rows from a table we use this command

mysql >**DELETE** FROM tablename WHERE condition;

EX: delete from Passenger where pnrno='3';
1 row deleted.

TEST OUTPUT:
SQL> select * from Passenger;

TO RETRIEVE / DISPLAY DATA FROM TABLES:

a. Select command is used to select values or data from table

SYNTAX

SELECT * FROM TABLENAME;

Example:

SQL> select * from Passenger;

TEST OUTPUT:

b. Elimination of duplicates from the select statement

SQL> **SELECT DISTINCT** columnname 1, columnname 2,…. columnname n FROM tablename;

EXAMPLE QUERY:

TEST OUTPUT:

c. The retrieving of specific columns from a table

Mysql > SELECT columnname 1, columnname 2,…. columnname n FROM tablename;

Mysql>select name,age,sex from Passenger;

TEST OUTPUT:

Example1:

Display Data From BUS Table

Example2: Display Data From Reservation Table

VIVA QUESTIONS

1. What are the DML commands?
2. How the data or values to be entered into a table?
3. What is the use of DELETE command?
4. How the data or values to be updated on a table?
5. List out the uses of SELECT command?
6. How the data or values are retrieved from a table?
7. Define DML? What are the DML commands?

Signature of the Faculty

WEEK -6

AIM : QUERIES USING ANY, ALL, IN, INTERSECT, UNION

UNION

Union is used to combine the results of two queries into a single result set of all matching rows. Both the queries must result in the same number of columns and compatible data types in order to unite. All duplicate records are removed automatically unless UNION ALL is used.

INTERSECT

It is used to take the result of two queries and returns the only those rows which are common in both result sets. It removes duplicate records from the final result set.

EXCEPT

It is used to take the distinct records of two one query and returns the only those rows which do not appear in the second result set.

EXAMPLES:

Let us create tables for sailors, Reserves and Boats

```
CREATE TABLE sailors ( sid integer, sname varchar(20),rating integer,age integer);

insert into sailors values(22,'dustin',7,45);
insert into sailors values(29,'brutus',1,33);
insert into sailors values(31,'lubber',79,55);
insert into sailors values(32,'andy',8,25);
insert into sailors values(58,'rusty',10,35);
insert into sailors values(58,'buplb',10,35);
insert into sailors values(58,'buplerb',10,35);

CREATE TABLE boats( bid integer, bname varchar(20),color varchar(20));
insert into boats values(101,'interlake','blue');
insert into boats values(102,'interlake','red');
insert into boats values(103,'clipper','green');
insert into boats values(104,'marine','red');

CREATE TABLE reserves( sid integer, bid integer, day date);
insert into reserves values(22,101,'2004-01-01');
insert into reserves values(22,102,'2004-01-01');
insert into reserves values(22,103,'2004-02-01');
insert into reserves values(22,105,'2004-02-01');
insert into reserves values(31,103,'2005-05-05');
insert into reserves values(32,104,'2005-04-07');
```

QUERIES

1. **Find all sailor id's of sailors who have a rating of at least 8 or reserved boat 103.**

```
mysql >(SELECT sid  FROM sailors WHERE rating>=8)
UNION
(SELECT sid  FROM reserves WHERE bid=103);
```

TEST OUTPUT:

2. **Find all sailor id's of sailors who have a rating of at least 8 and reserved boat 103.**

```
mysql >( (SELECT sid  FROM sailors WHERE rating>=8)
intersect
(SELECT sid  FROM reserves WHERE bid=103);
```

TEST OUTPUT:

3. **Find the names of sailors who have reserved boat number 103.**

```
mysql >(select s.sname from sailors s  where s.sid in (select r.sid from reserves r where r.bid=103);
```

TEST OUTPUT:

4. **Find the names of sailors who have never reserved boat number 103.**

```
mysql >(select s.sname from sailors s where s.sid not in (select r.sid from reserves r where r.bid=103);
```

TEST OUTPUT:

5. **Find sailors whose rating is better than some sailor called Horatio**

```
mysql >(select s.sid from sailors s where s.rating > any(select s2.rating from sailors s2 where
s2.sname='Horatio');
```

TEST OUTPUT:

6. Find the sailors with the highest rating

mysql >(select s.sid from sailors s where s.rating >= all (select s2.rating from sailors s2);

TEST OUTPUT:

QUERIES ON ROADWAY TRAVELS DATABASE

1. Display unique PNR_no of all Passengers.. .

Mysql>select distinct(pnrno) from Passenger;

TEST OUTPUT:

2. Display all the names of male passengers

Mysql >select Name from Passenger where Sex='m';

TEST OUTPUT:

3. Display Ticket numbers and names of all Passengers.

Mysql>select ticketno,Name from Passenger;

TEST OUTPUT:

4. Find the ticket numbers of the passengers whose name start with 'r' and ends with 'h'.

Mysql>select ticketno from Passenger where Name like'r%h';

TEST OUTPUT:

5. Find the names of passengers whose age is between 30 and 45.

Mysql>select Name from Passenger where age between 30 and 45;

TEST OUTPUT:

6. Display all the passengers names beginning with 'A'.

Mysql>select Name from Passenger where Name like 'A%';

TEST OUTPUT:

7. Display the sorted list of passengers names

Mysql>select name from Passenger order by

Name;TEST OUTPUT:

VIVA QUESTIONS

1) Explain the flow of execution for a Nested query

2) Differentiate between flow of execution in Nested Query and Correlated Query?

3) What happens if we eliminate HAVING clause in a query which is having both GROUP BY and HAVING clauses.

4) What are the different types of Nested Queries?

5) What are the different types of Correlated Queries?

Signature of the Faculty

Signature of the Faculty

WEEK-7

Querying Using Aggregate functions (COUNT, SUM, AVERAGE using GROUPBY and HAVING) Creation and dropping of Views.

Aggregate operators: In addition to simply retrieving data, we often want to perform some computation or summarization. SQL allows the use of arithmetic expressions.

1. COUNT:

SYNTAX:

Select count ([<distinct>/<ALL]<expr>)

2. SUM:

SYNTAX:

Select SUM ([<distinct>/<ALL]<column name>)

3. AVG:

SYNTAX:

Select AVG ([<distinct>/<ALL]<column name>)

4. MINIMUM(MIN):

SYNTAX:

Select MIN ([<distinct>/<ALL]<expr>)

5. MAXIMUM(MAX):

SYNTAX:

Select MAX ([<distinct>/<ALL]<expr>)

GROUP BY and HAVING Clause

SYNTAX

Select [DISTINCT] select list FROM from list WHERE qualification
Group by Groupinglist having group-qualification

1. **Write a Query to display the Information present in the Reservation and cancellation tables.**

```
mysql>select * from Reservation
        union
      select * from Cancellation;
```

TEST OUTPUT:

2. **Display the number of days in a week on which the 9W01 bus is available.**

```
mysql> select daysperweek from Bus where Bus_No='9w01';
```

TEST OUTPUT:

3. **Find number of tickets booked for each PNR_no using GROUP BY CLAUSE**

```
mysql>select count(No_of_seats),PNR_NO from Reservation group by PNR_NO;
```

TEST OUTPUT:

4. **Find the number of tickets booked by a passenger where the number of seats is greater than 1**

```
mysql>select count(No_of_seats) from Reservation group by PNR_NO having
sum(No_of_seats)>1;
```

TEST OUTPUT:

5. **Find the distinct PNR numbers that are present.**

mysql>select distinct(PNR_NO) from Reservation;

TEST OUTPUT:

6. **Find the total number of cancelled seats.**

mysql > select sum(No_of_seats) AS Cancelled_seats from Cancellation;

TEST OUTPUT:

VIEWS

After a table is created and populated with data, it may become necessary to prevent all users from accessing all columns of a table, for data security reasons. This would mean creating several tables having the appropriate number of columns and assigning specific users to each table as required. This will achieve the security requirements but will rise to a great deal of redundant data being resident in tables, in the database. To reduce redundant data to the minimum possible, oracle allows the creation ofan object called a view.

A view is a virtual table or logical representation of another table or combination of tables. A view consists of rows and columns just like a table. The difference between a view and a table is that views are definitions built on top of other tables (or views), and do not hold data themselves. If data is changing in the underlying table, the same change is reflected in the view. A view can be built on top of a single table or multiple tables. It can also be built on top of another view. A view derives its data from the tables on which it is based. These tables are called base tables. Base tables might in turn be actual tables or might be views themselves. All operations performed on a view actually affect the base table of the view. We can use views in almost the same way as tables. Also can query, update, insert into and delete from views, just as in standard tables

AIM : Implement Views:

Syntax: Create View <View_Name> As Select statement;
Example:
SQL>Create View EmpView As Select * from Emp_master where job='clerk';

View created.

Syntax: Select columnname,columnname from <View_Name>;

Example:
SQL>Select Empno,Ename,Salary from EmpView where salary in (10000,20000);

TEST OUTPUT:

UPDATABLE VIEWS:
Syntax for creating an Updatable View:

Create View Emp_vw As
Select Empno,Ename,Deptno from Employee;

View created.

SQL>Insert into Emp_vw values(1126,'Brijesh',20);

SQL>Update Emp_vw set Deptno=30 where

Empno=1125;

1 row updated.

SQL>Delete from Emp_vw where

Empno=1122;

TEST OUTPUT:

mysql >Update EmpDept_Vw set salary=4300 where Empno=1125;

TEST OUTPUT:

mysql >Delete From EmpDept_Vw where Empno=1123;

TEST OUTPUT

DESTROYING A VIEW:

Syntax: Drop View <View_Name>;

Example:

mysql >Drop View Emp_Vw;

TEST OUTPUT:

VIVA QUESTIONS:

1. Define view.
2. What is the need of a view?
3. List out the advantages of views.
4. What is the syntax for creating a view?
5. How can you insert data into a view?
6. How can you update data into from a view?
7. What is the syntax for deleting a view?
8. What is the syntax for renaming the columns of a view

Signature of the Faculty

Signature of the Faculty

WEEK-8

Triggers

In MySQL, a trigger is a set of SQL statements that is invoked automatically when a change is made tothe data on the associated table. A trigger can be defined to be invoked either before or after the data ischanged by INSERT, UPDATE or DELETE statement.

- BEFORE INSERT – activated before data is inserted into the table.
- AFTER INSERT – activated after data is inserted into the table.
- BEFORE UPDATE – activated before data in the table is updated.
- AFTER UPDAT – activated after data in the table is updated.
- BEFORE DELETE – activated before data is removed from the table.
- AFTER DELETE – activated after data is removed from the table.

A database trigger is procedural code that is automatically executed in response to certain events on a particular table or view in a database. The trigger is mostly used for maintaining the integrity of the information on the database.

The events that fire a trigger include the following:
1)DML statements that modify data in a table (INSERT , UPDATE , or DELETE
)2)DDL statements.
3) System events such as startup, shutdown, and error messages.
4) User events such as logon and logoff. Note: Oracle Forms can define, store, and run triggersofa different sort.

To View list of triggers;
Show triggers;

To remove a trigger for Database
drop trigger trigger_name;

ex: drop trigger ins_sal;

Types of Triggers:-

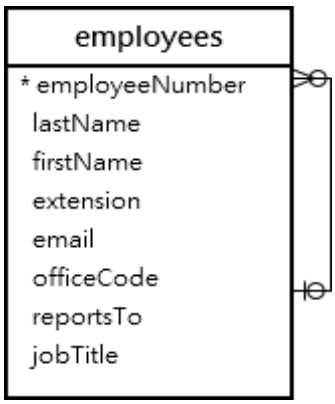
1. **Row Triggers** :-A row trigger is fired each time the table is affected by the triggering statement. For example, if an UPDATE statement updates multiple rows of a table, a row trigger is fired once for each row affected by the UPDATE statement. If a triggering statement affects no rows, a row trigger is not executed at all.

Row triggers are useful if the code in the trigger action depends on data provided by the triggering statement or rows that are affected. For example, Figure 15 - 3 illustrates a row trigger thatuses the values of each row affected by the triggering statement.

2. **Statement Triggers** :A statement trigger is fired once on behalf of the triggering statement, regardless of the number of rows in the table that the triggering statement affects (even if no rows are affected). For example, if a DELETE statement deletes several rows from a table, a statement-level DELETE trigger is fired only once, regardless of how many rows are deleted from the table.

Statement triggers are useful if the code in the trigger action does not depend on the data provided by the triggering statement or the rows affected. For example, if a trigger makes a complex security check on the current time or user, or if a trigger generates a single audit record based on the type of triggering statement, a statement trigger is used.

When defining a trigger, specify the trigger timing. That is, specify whether the trigger action isto be executed before or after the triggering statement. BEFORE and AFTER apply to both statement and row triggers



Example:

```
CREATE TRIGGER trigger_name       trigger_time      trigger_event
ON table_name
FOREACH ROW
BEGIN
END;
```

trigger_time=before/after
trigger_event=insert/delete/update

Example:

```
CREATE TRIGGER sal_sum after insert ON emp
FOR EACH ROW SET @sal = @sal + NEW.sal;
```

FIRING A TRIGGER:

creating a trigger in MySQL to log the changes of the employeestable.

The CREATE TRIGGER statement creates a new trigger. Here is the basic syntax of the CREATE TRIGGER statement:

```
CREATE TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE| DELETE }
ON table_name FOR EACH ROW
trigger_body;
```

First, create a new table named employees_audit to keep the changes to the employees table:

```
CREATE TABLE employees_audit (
id INT AUTO_INCREMENT PRIMARY KEY,
employeeNumber INT NOT NULL,
lastname VARCHAR(50) NOT NULL,
changedat DATETIME DEFAULT NULL,
action VARCHAR(50) DEFAULT NULL
);
```

create a BEFORE UPDATE trigger that is invoked before a change is made to the employees table.

```
CREATE TRIGGER before_employee_update
   BEFORE UPDATE ON employees
   FOR EACH ROW
INSERT INTO employees_audit
SET action = 'update',
    employeeNumber = OLD.employeeNumber,
    lastname = OLD.lastname,
    changedat = NOW();
```

Inside the body of the trigger, we used the OLD keyword to access values of the columns employeeNumber and lastname of the row affected by the trigger.

show all triggers in the current database by using the SHOW TRIGGERS statement:

```
mysql> show triggers;
```

mysql> **update a row in the employees table:**

```
UPDATE employeesSET
   lastName = 'Phan'WHERE
employeeNumber = 1056;
```

query the employees_audit table to check if the trigger was fired by the UPDATE statement:

```
SELECT * FROM employees_audit;
```

OUTPUT:

MySQL AFTER INSERTtrigger

create a new table called members:

```
DROP TABLE IF EXISTS members;

CREATE TABLE members ( id INT
AUTO_INCREMENT,
name VARCHAR(100) NOT NULL, email
VARCHAR(255),
birthDate DATE, PRIMARY KEY (id)
);
```

create another table called reminders that stores reminder messages to members.

```
DROP TABLE IF EXISTS reminders;

CREATE TABLE reminders ( id INT
AUTO_INCREMENT,
memberId INT,
message VARCHAR(255) NOT NULL,
PRIMARY KEY (id , memberId)
);
```

VIVA QUESTIONS:

1. Define database triggers.
2. List out the uses of database triggers.
3. What are the pars of triggers and it uses?
4. List out the types of trigger.
5. What is the use of row trigger?
6. What is the use of statement trigger?
7. What do you meant by trigger time?
8. Compare before trigger and after trigger.
9. What is the syntax for DROP a trigger?
10. List out the some situations to apply before and after triggers.

Signature of the Faculty

WEEK-9

Procedures: Creation of stored procedures, Execution of procedure and modification of procedures.

PROCEDURES

Procedure (often called a stored procedure) is a collection of pre-compiled SQL statements stored inside the database. It is a subroutine or a subprogram in the regular computing language. A procedure always contains a name, parameter lists, and SQL statements. We can invoke the procedures by using triggers, other procedures and applications such as Java, Python, PHP, etc.

It was first introduced in MySQL version 5. Presently, it can be supported by almost all relational database systems.

Creating a procedure:

The following syntax is used for creating a stored procedure in MySQL. It can return one or more value through parameters or sometimes may not return at all. By default, a procedure is associated with our current database. But we can also create it into another database from the current database by specifying the name as **database_name.procedure_name**.

```
DELIMITER &&
CREATE PROCEDURE procedure_name [[IN | OUT | INOUT] parameter_name datatype [, parameter datatype]) ]
BEGIN
   Declaration_section
   Executable_section
END &&
DELIMITER ;
```

Parameter Explanations

The procedure syntax has the following parameters:

Parameter Name	Descriptions
procedure_name	It represents the name of the stored procedure.
parameter	It represents the number of parameters. It can be one or more than one.
Declaration_section	It represents the declarations of all variables.

Executable_section	It represents the code for the function execution.

MySQL procedure parameter has one of three modes:

IN parameter

It is the default mode. It takes a parameter as input, such as an attribute. When we define it, the calling program has to pass an argument to the stored procedure. This parameter's value is always protected.

OUT parameters

It is used to pass a parameter as output. Its value can be changed inside the stored procedure, and the changed (new) value is passed back to the calling program. It is noted that a procedure cannot access the OUT parameter's initial value when it starts.

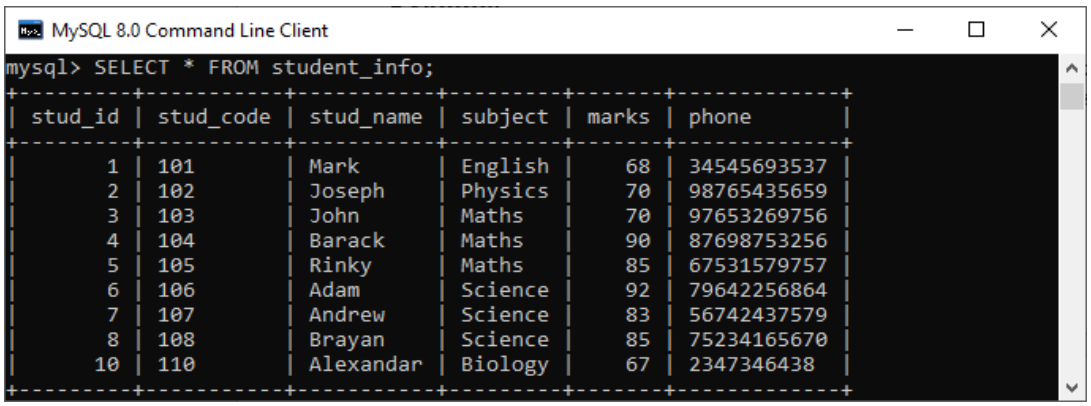
INOUT parameters

It is a combination of IN and OUT parameters. It means the calling program can pass the argument, and the procedure can modify the INOUT parameter, and then passes the new value back to the calling program.

How to call a stored procedure?

CALL procedure_name (parameter(s))

Suppose this database has a table named **student_info** that contains the following data:



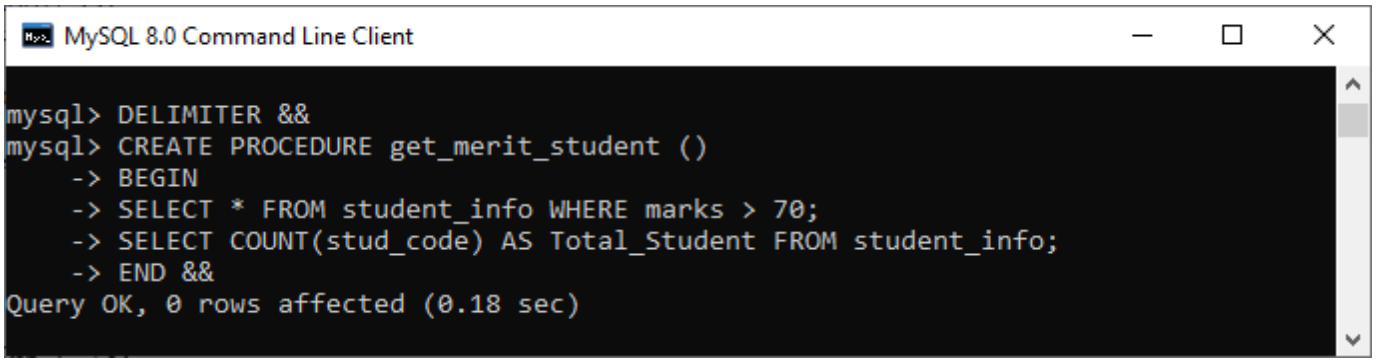
Procedure without Parameter

Suppose we want to display all records of this table whose marks are greater than 70 and count all the

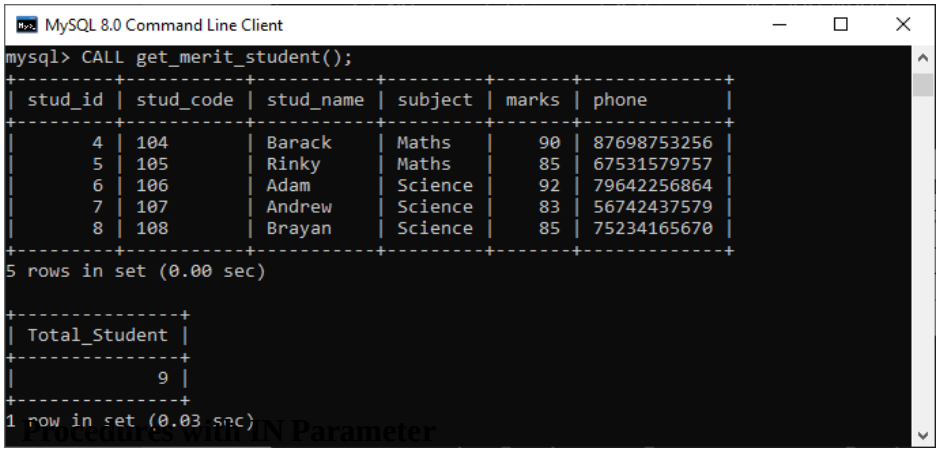
table rows. The following code creates a procedure named get_merit_students**:**

```
DELIMITER &&
CREATE PROCEDURE get_merit_student ()
BEGIN
   SELECT * FROM student_info WHERE marks > 70;
   SELECT COUNT(stud_code) AS Total_Student FROM student_info;
END &&
DELIMITER ;
```

If this code executed successfully, we would get the below output:



mysql> CALL get_merit_student();

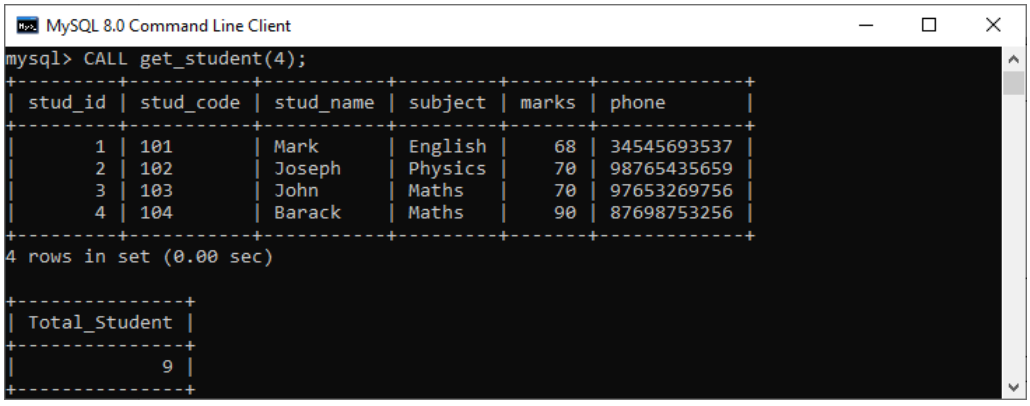


Procedures with IN Parameter

In this procedure, we have used the IN parameter as 'var1' of integer type to accept a number from users. Its body part fetches the records from the table using a SELECT statement
and returns only those rows that will be supplied by the user. It also returns the total number of rows of the specified table. See the procedure code**:**

```
DELIMITER &&
CREATE PROCEDURE get_student (IN var1 INT)
BEGIN
   SELECT * FROM student_info LIMIT var1;
   SELECT COUNT(stud_code) AS Total_Student FROM student_info;
END &&
DELIMITER ;
```

mysql> CALL get_student(4);

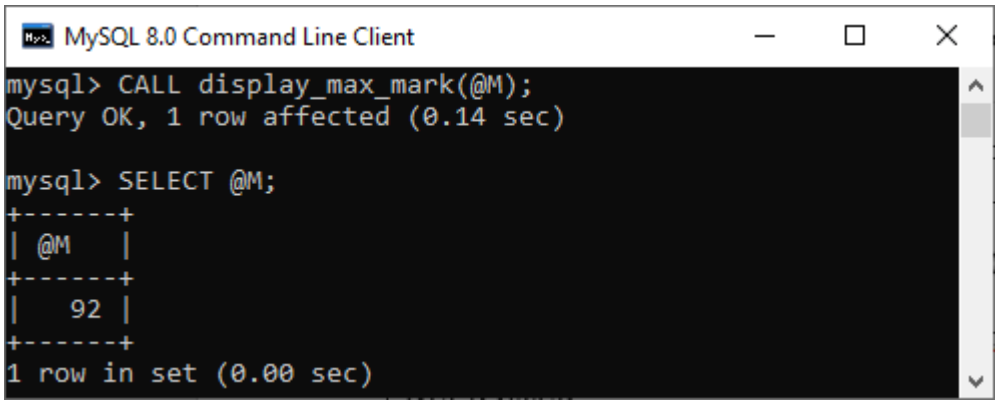


Procedures with OUT Parameter:

In this procedure, we have used the OUT parameter as the '**highestmark**' of integer type. Its body part fetches the maximum marks from the table using a **MAX() function.** See the procedure code

```
DELIMITER &&
CREATE PROCEDURE display_max_mark (OUT highestmark INT)
BEGIN
   SELECT MAX(marks) INTO highestmark FROM student_info;
END &&
DELIMITER ;
```

mysql> CALL display_max_mark(@M);
mysql> SELECT @M;

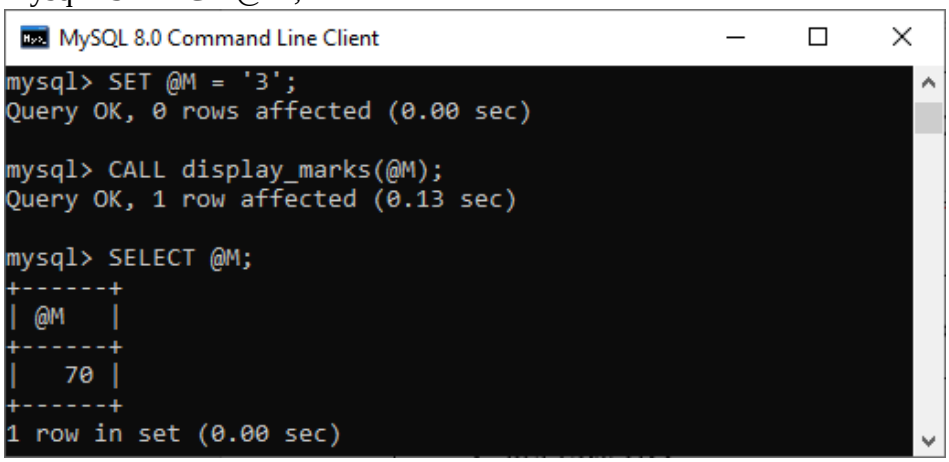


Procedures with INOUT Parameter:

In this procedure, we have used the INOUT parameter as 'var1' of integer type. Its body part first fetches the marks from the table with the specified id and then stores it into the same variable var1. The var1 first acts as the IN parameter and then OUT parameter. Therefore, we can call it the INOUT parameter mode. See the procedure code:

```
DELIMITER &&
CREATE PROCEDURE display_marks (INOUT  var1 INT)
BEGIN
   SELECT marks INTO var1 FROM student_info WHERE stud_id = var1;
END &&
DELIMITER ;
```

mysql> SET @M = '3';
mysql> CALL display_marks(@M);
mysql> SELECT @M;



CREATING PROCEDURE ON FOR ROADWAY TRAVELS

```
CREATE PROCEDURE  myProc() BEGIN
SELECT COUNT (Tickets) FROM Ticket WHERE age>=40;
End;
Procedures created
```

1) WRITEA PROGRAM TO PRINT HELLOWORLD

```
BEGIN
DBMS_OUTPUT.PUT_LINE('HELLOWORLD');
END;
```

OUTPUT:-

2) WRITE A PROGRAM TO PRINT EVEN NUMBERS FROM 1 TO 100

```
DECLARE
NNUMBER(3):=0;
BEGIN
WHILE N<=100
LOOP
N:=N+2;
DBMS_OUTPUT.PUT_LINE(N);
ENDLOOP;
END;
```

OUTPUT:-

VIVA QUESTIONS:

1. What is Stored Procedure?
2. What is difference between Function and Stored Procedure?
3. What are the various types of parameters in procedures?

Signature of the Faculty

WEEK-10

Usage of Cursors

Cursors allow us to process data row-by-row, which can be useful when we need to perform complex calculations or transformations on the data. Cursors allow us to iterate over a result set multiple times, which can be useful when we need to perform multiple operations on the same data.

A cursor in SQL is a database object stored in temp memory and used to work with datasets. You can use cursors to manipulate data in a database, one row at a time. A cursor uses a SQL SELECT statement to fetch a rowset from a database and then can read and manipulate one row at a time.

There are two types of cursors in SQL:

1. Implicit Cursor
2. Explicit Cursor

Implicit Cursor

The system generates and uses these types of cursors to manipulate a DML query (INSERT, UPDATE, and DELETE). In addition, a system also generates an implicit cursor when a SELECT command selects a single row.

Explicit Cursor

This type of cursor is generated by the user using a SELECT command. An explicit cursor contains more than one row, but only one row can be processed at a time. An explicit cursor moves one by one over the records. An explicit cursor uses a pointer that holds the record of a row. After fetching a row, the cursor pointer moves to the next row.

Steps for working with Cursors:

1. Declare <Cursor_Name> for select_statement
2. Open <Cursor_Name> initializes the result set for operation
3. Fetch <Cursor_Name> into variable list to retrieve the next row pointed by the cursor and move the cursor to the next row in the result set
4. Close <Cursor_Name> to deactivate the cursor and release any memory associated with it

Create a Cursor to get the Emp Name and Salary from Employee table?

```
delimeter $$
1. create procedure proc_emp
2. begin
3. declare v_ename varchar(100);
4. declare v_salary int;
5. declare v_finished integer default 0;
6. declare c1 cursor for select ename,sal from employee;
7. declare continue handler for NOT FOUND set v_finished=1;
8. open c1;
9. get_emp: LOOP
10. fetch c1 into v_ename,v_salary;
11. if v_finished=1 then
12. leave get_emp;
13. end if;
14. select concat(v_name,concat('-',v_salary));
15. END LOOP get_emp;
16. close c1;
end $$
```

EXECUTION CURSOR:-

```
Call  proc_emp();
$$
Or
Delimeter;
```

1. **To write a cursor to display the list of male and female passengers?**
2. **To write a cursor to display list of passengers from passenger table?**

VIVA QUESTIONS:

1. what is cursor?
2. Types of cursors?

Signature of the Faculty

WEEK-11

Aim: Installation of MySQL / MongoDB and practicing DDL commands.

1. Steps for installing MySQL

Step1

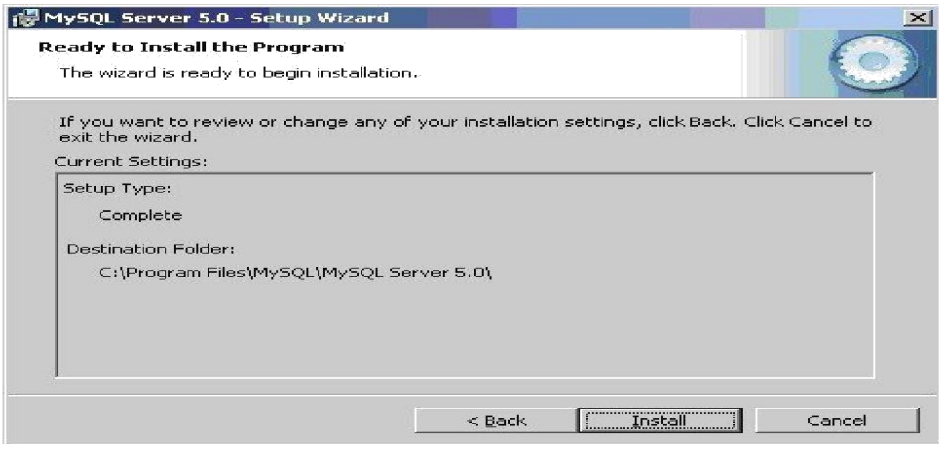
Make sure you already downloaded the **MySQL essential 5.0.45 win32.msi file**. Double click on the .msi file.

Step2

This is MySQL Server 5.0 setup wizard. The setup wizard will install MySQL Server 5.0 release 5.0.45 on your computer. To continue, click **next.**

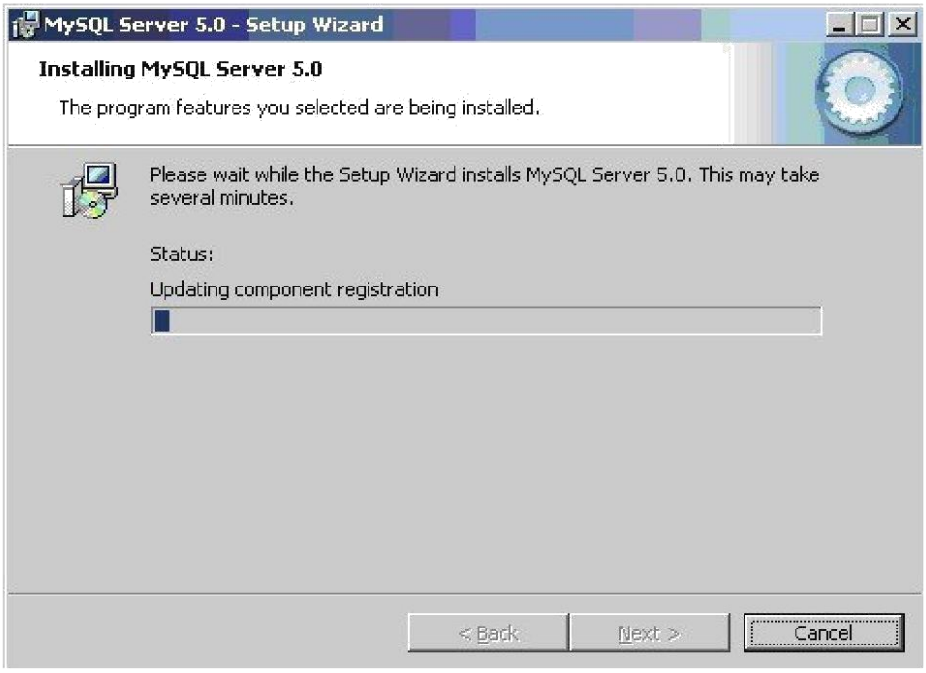
Step 3

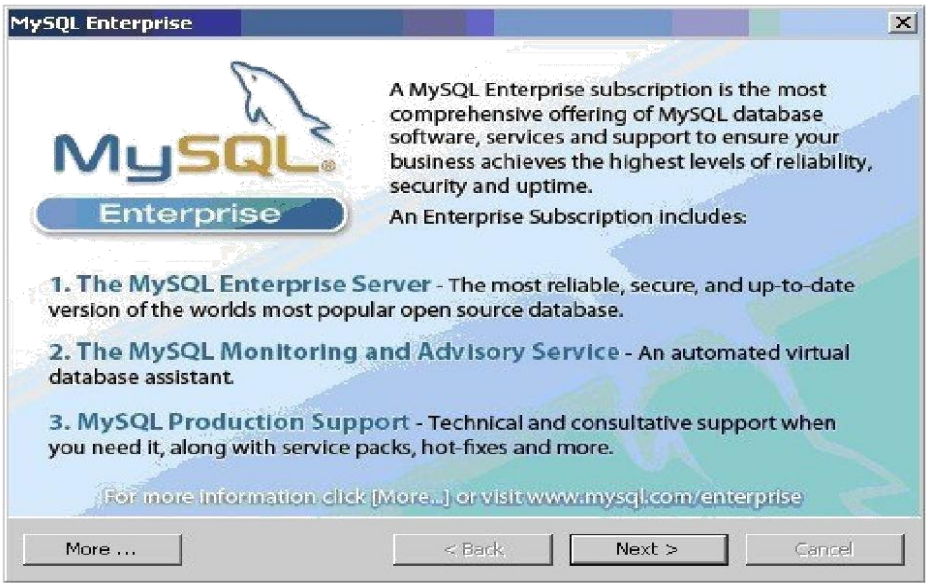
Choose the setup type that best suits your needs. For common program features select ***Typical*** and it's recommended for general use. To continue, click **next**.

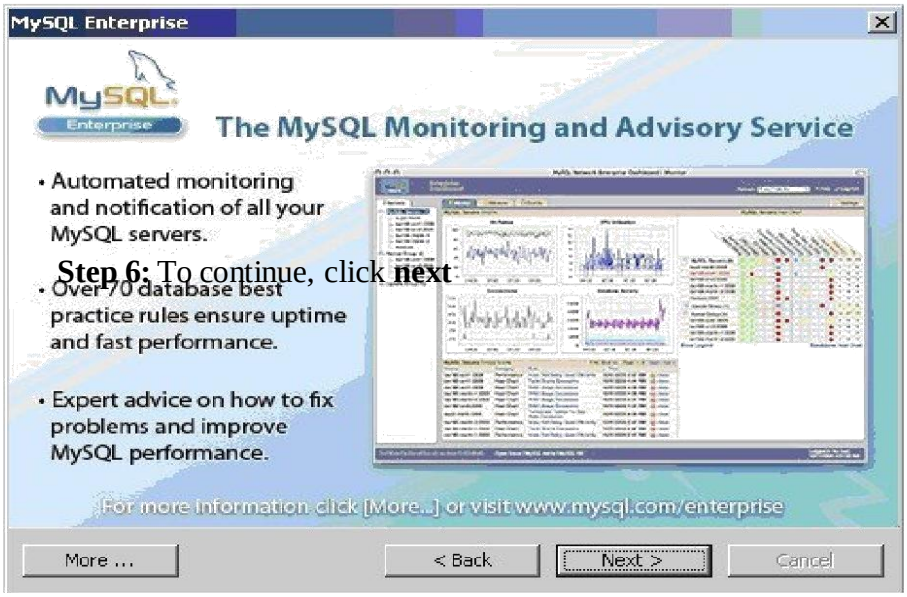


Step 4

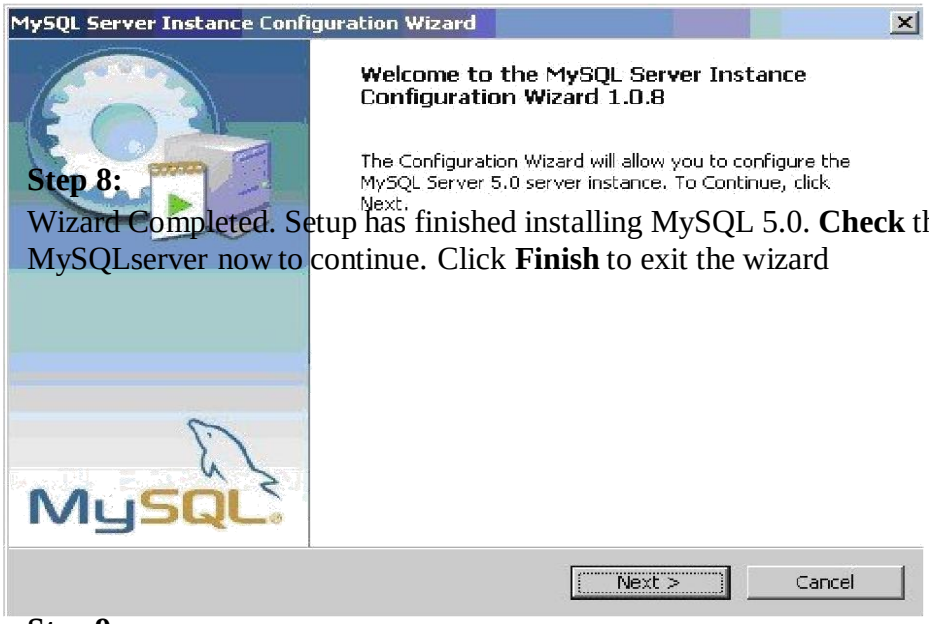
This wizard is ready to begin installation. Destination folder will be in **C:\ProgramFiles\MySQL\MySQLServer5.0**.Tocontinue,click**next**.


MySQL Server 5.0 - Setup Wizard
Installing MySQL Server 5.0
The program features you selected are being installed.
Please wait while the Setup Wizard installs MySQL Server 5.0. This may take several minutes.
Status:
Updating component registration
< Back
Next >
Cancel


MySQL Enterprise
MySQL
Enterprise
A MySQL Enterprise subscription is the most comprehensive offering of MySQL database software, services and support to ensure your business achieves the highest levels of reliability, security and uptime.
An Enterprise Subscription includes:
1. The MySQL Enterprise Server - The most reliable, secure, and up-to-date version of the worlds most popular open source database.
2. The MySQL Monitoring and Advisory Service - An automated virtual database assistant.
3. MySQL Production Support - Technical and consultative support when you need it, along with service packs, hot-fixes and more.
For more information click [More...] or visit www.mysql.com/enterprise
More ...
< Back
Next >
Cancel



Step 6: To continue, click **next**

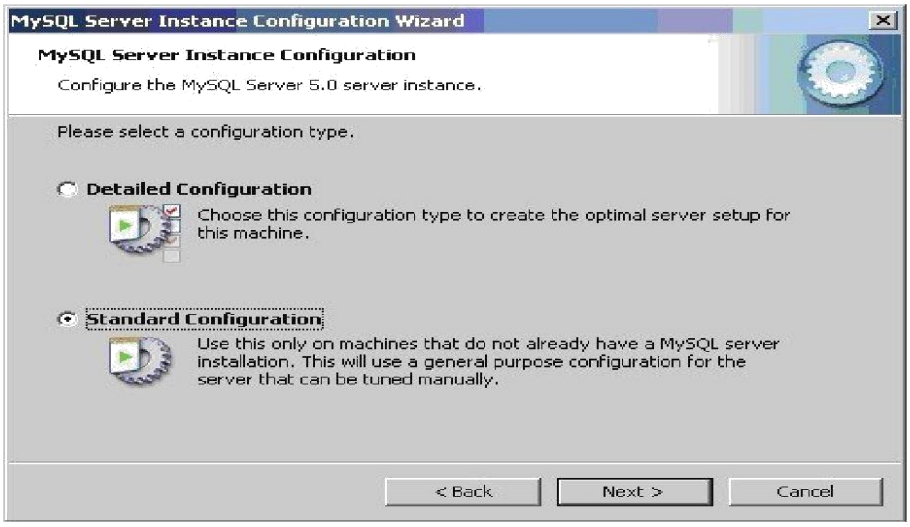


Step 8:

Wizard Completed. Setup has finished installing MySQL 5.0. **Check** the configure the MySQLserver now to continue. Click **Finish** to exit the wizard

Step 9:

The configuration wizard will allow you to configure the MySQL Server 5.0 server instance. To continue,click **next**

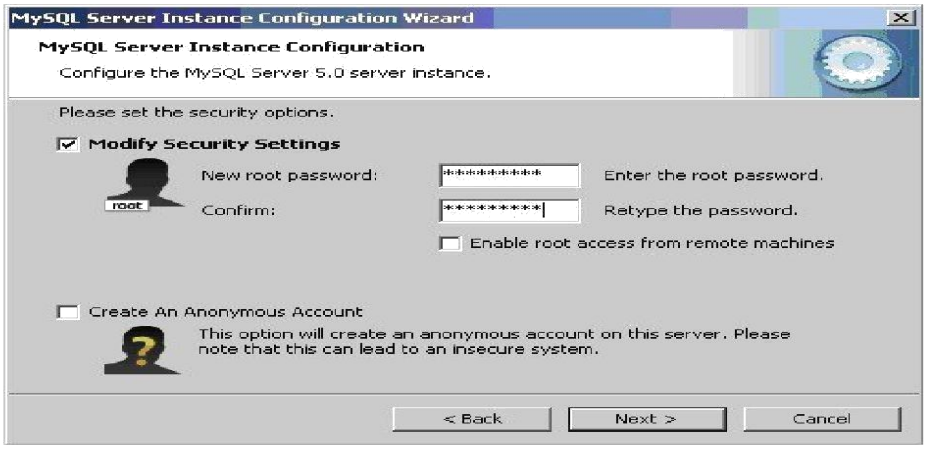


Step 10 :

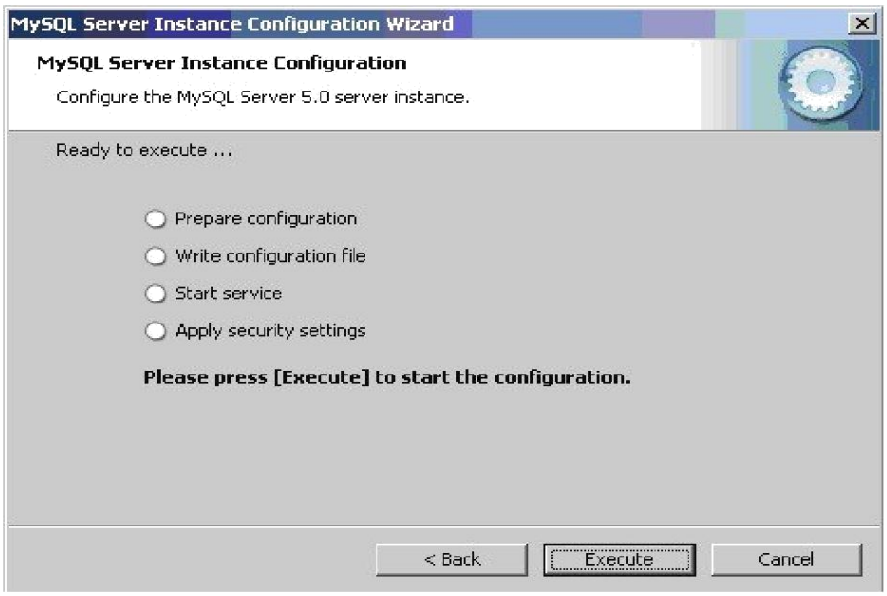
Select a standard configuration and this will use a general purpose configuration for the serverthat can be tuned manually. To continue, click next

Step:11

Check on the **install as windows service** and **include bin directory in windows path**. Tocontinue,click**next**.

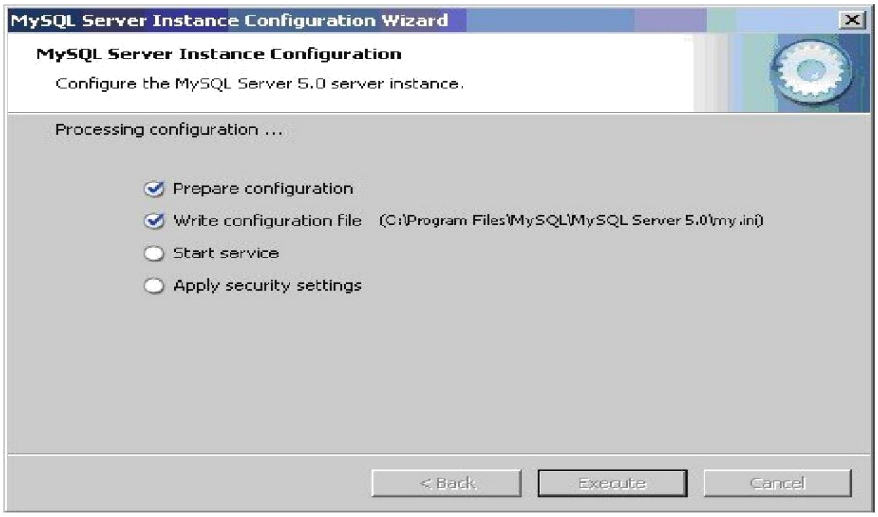


Step12: Please set the security options by entering the root password and confirm retype thepassword.continue, click next.

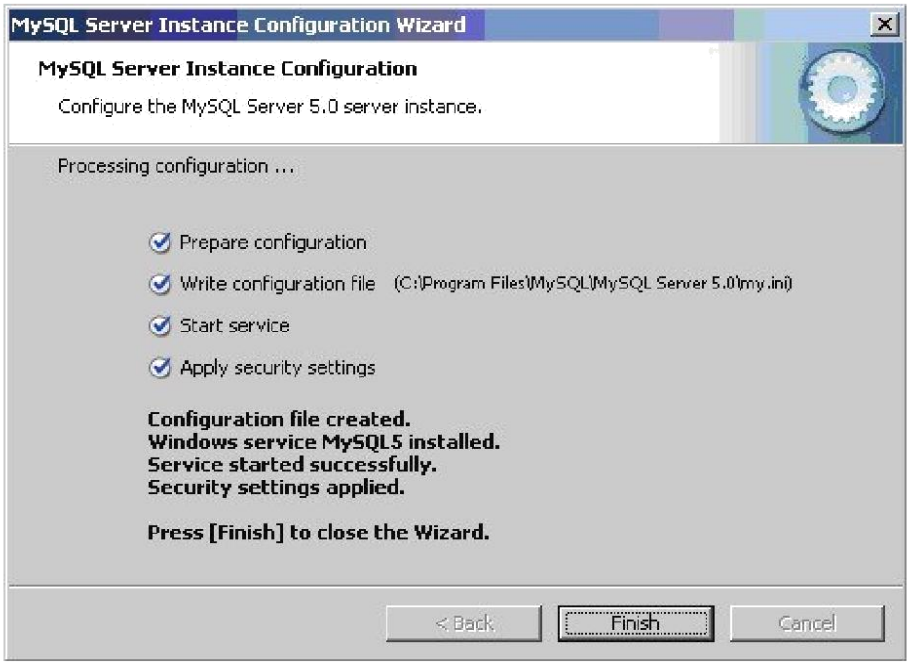


Step13

Ready to execute? Clicks **execute** to continue.



Step14
Processing configuration in progress.



Step15

Configuration file created. Windows service MySQL5 installed. Press **finish** to close the wizard.

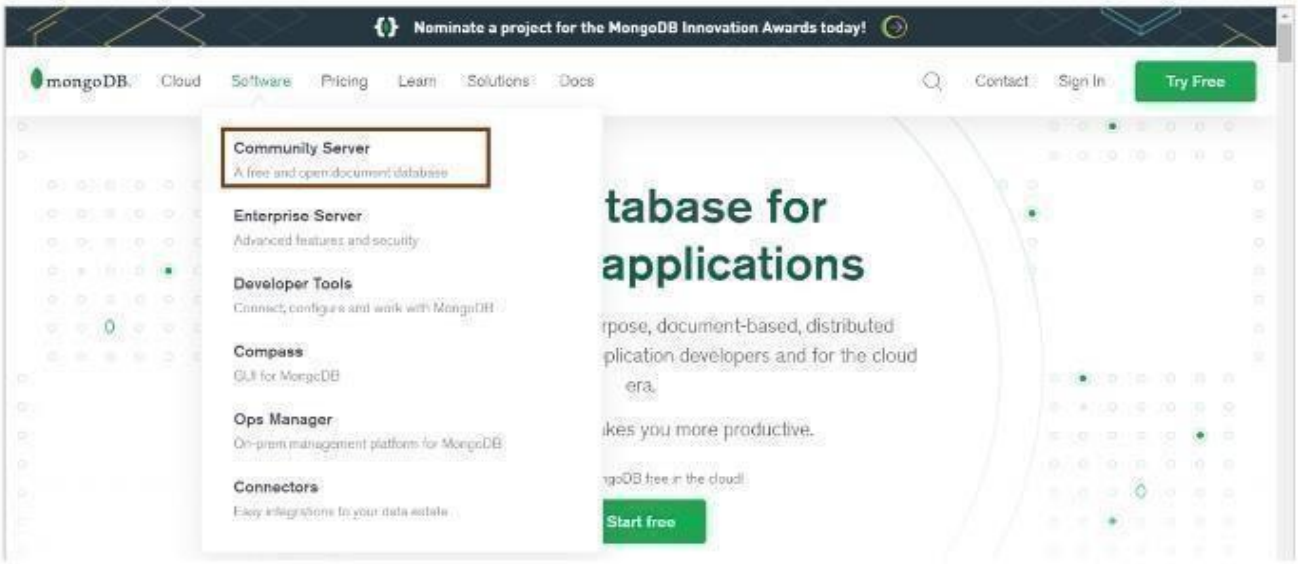
Installation of MangoDB

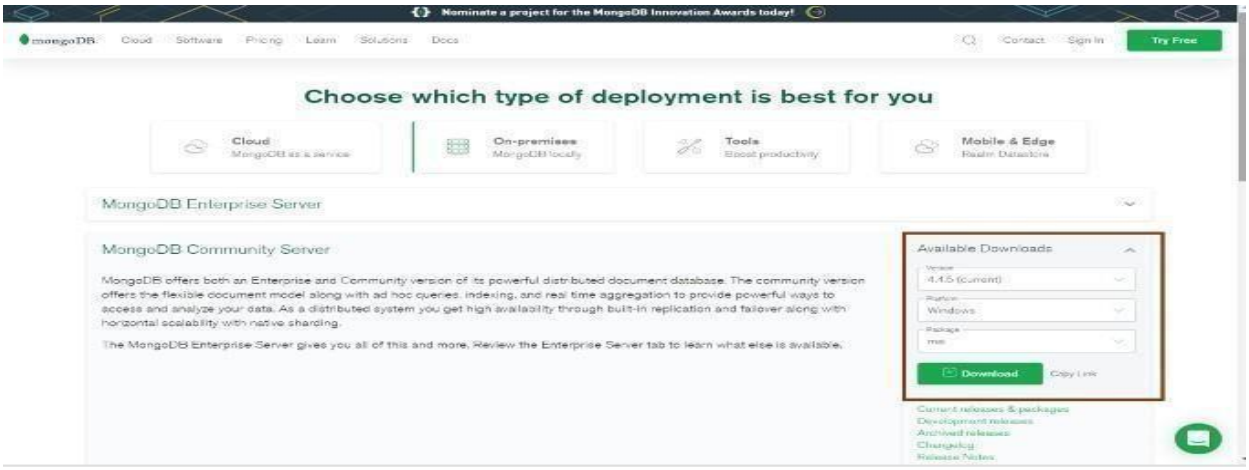
Navigate to the download site:

Navigate to the official MongoDB website https://www.mongodb.com/

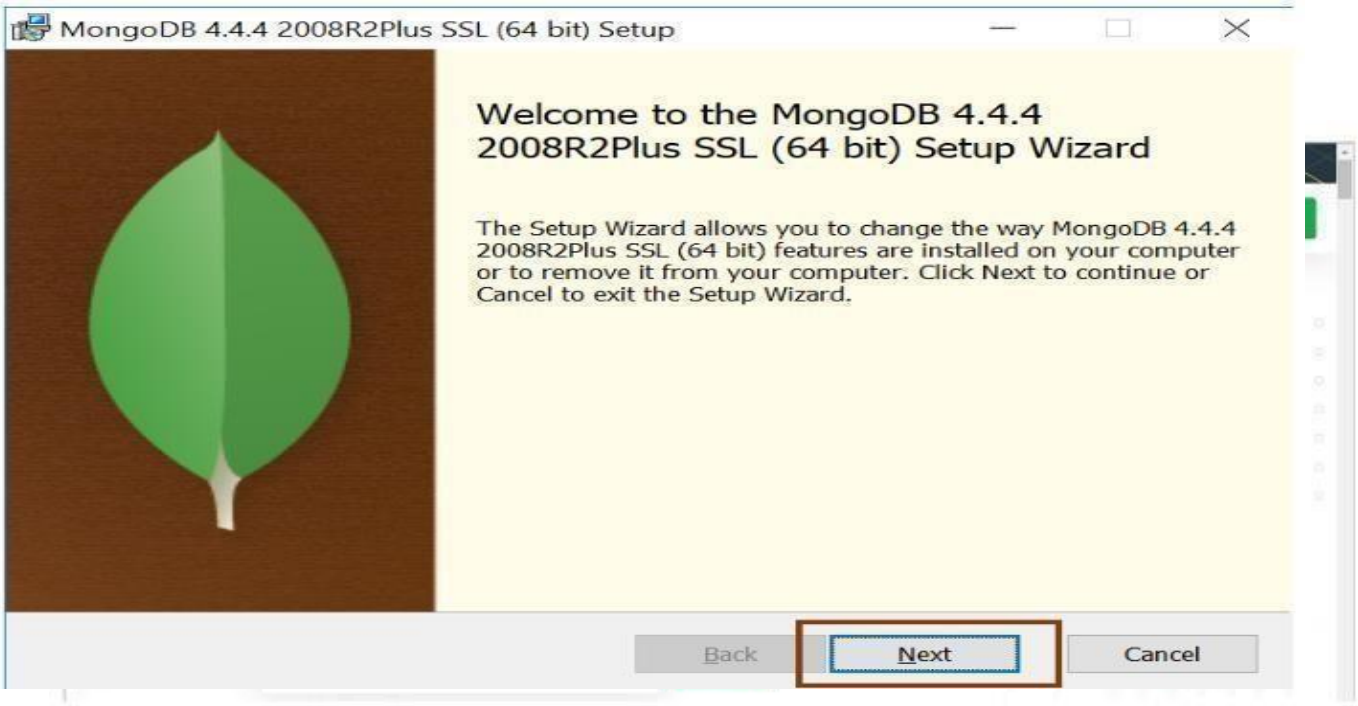
Cross-check the Specifications and Download MongoDB

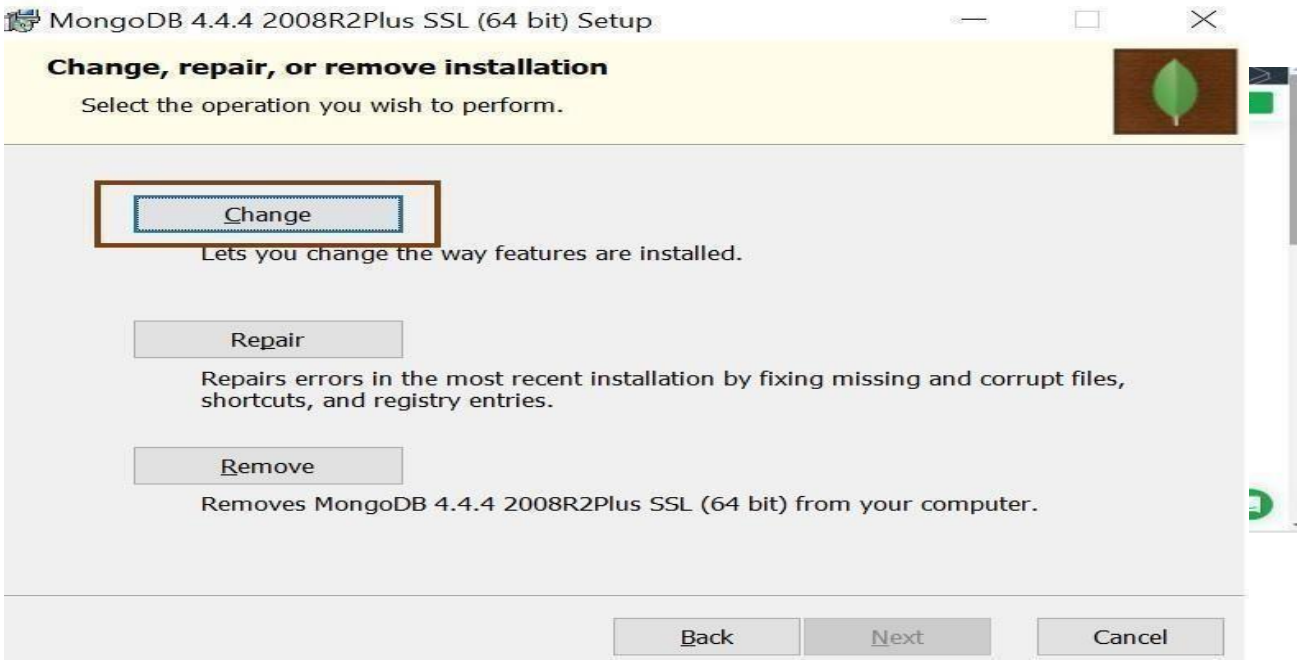
Under the Software section, click on the Community server version.





At the time of writing, the latest version is 4.4.5. Ensure that the platform is Windows, and the packageis MSI. Go ahead and click on download.

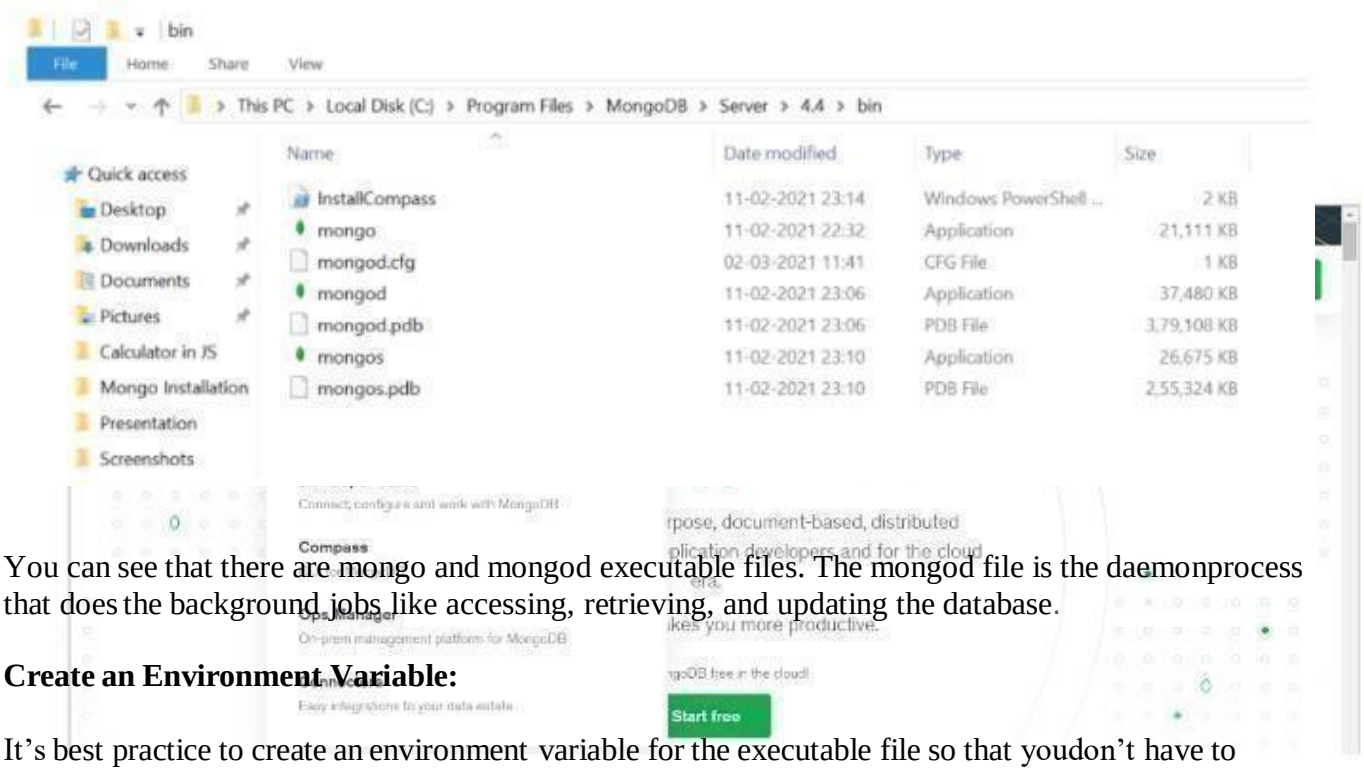




Mongo DB Installation:

You can find the downloaded file in the downloads directory. You can follow the stepsmentioned there and install the software.

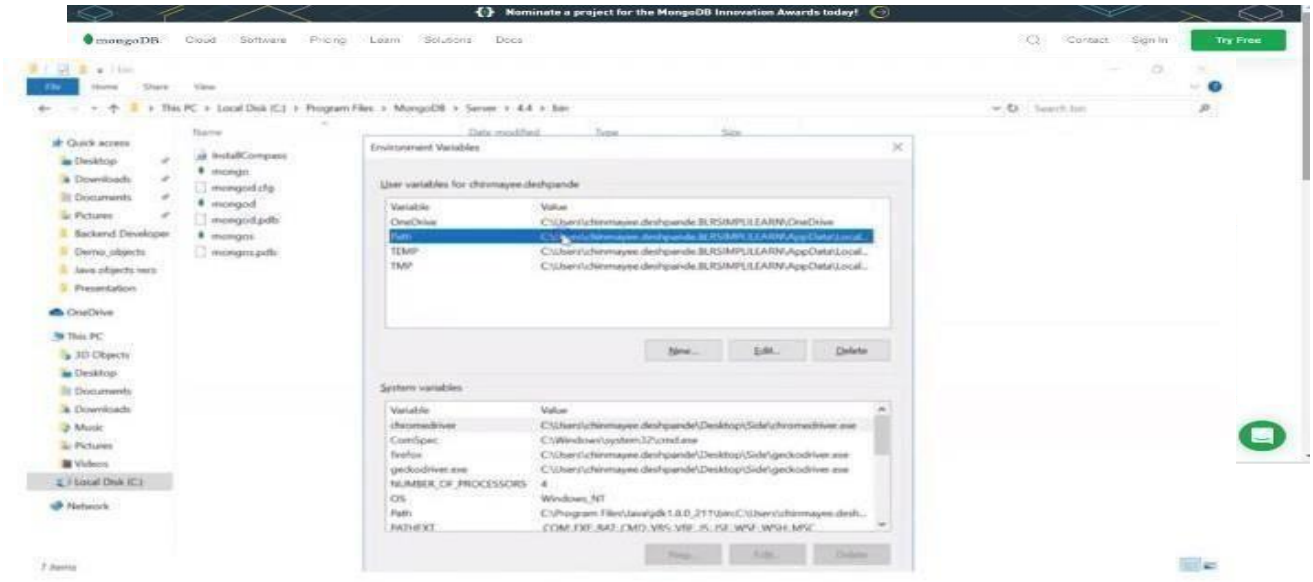
On completing the installation successfully, you will find the software package in your Cdrive. C:\Program Files\MongoDB\Server\4.4\bin.

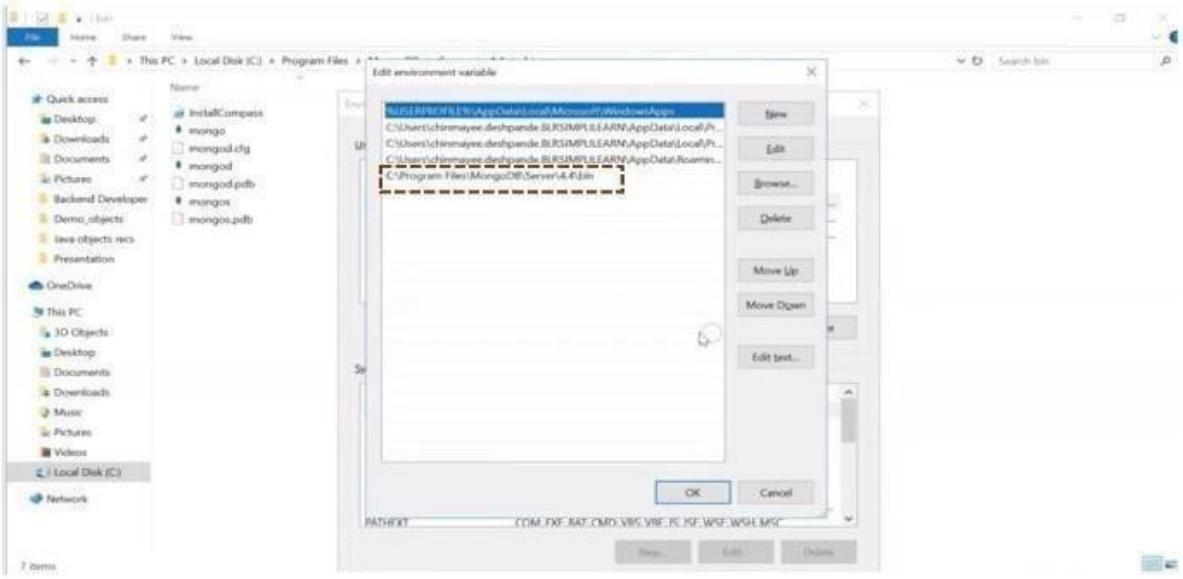


You can see that there are mongo and mongod executable files. The mongod file is the daemonprocess that does the background jobs like accessing, retrieving, and updating the database.

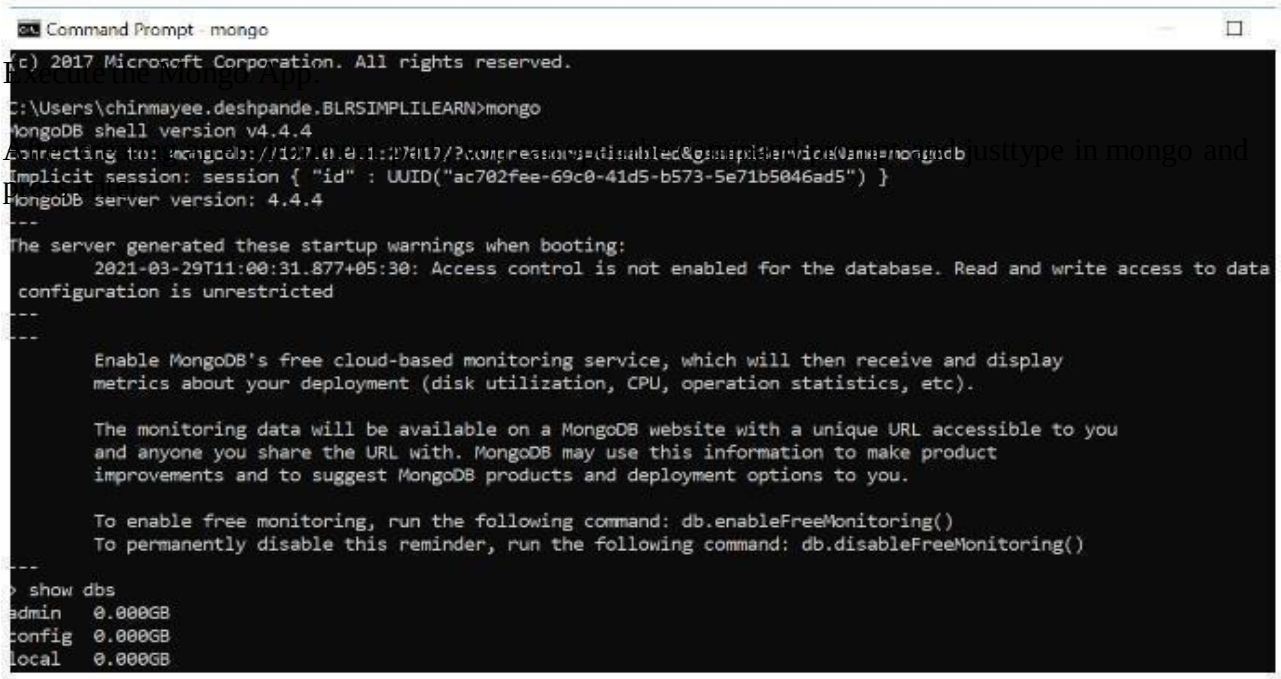
Create an Environment Variable:

It's best practice to create an environment variable for the executable file so that youdon't have to change the directory structure every time you want to execute the file.



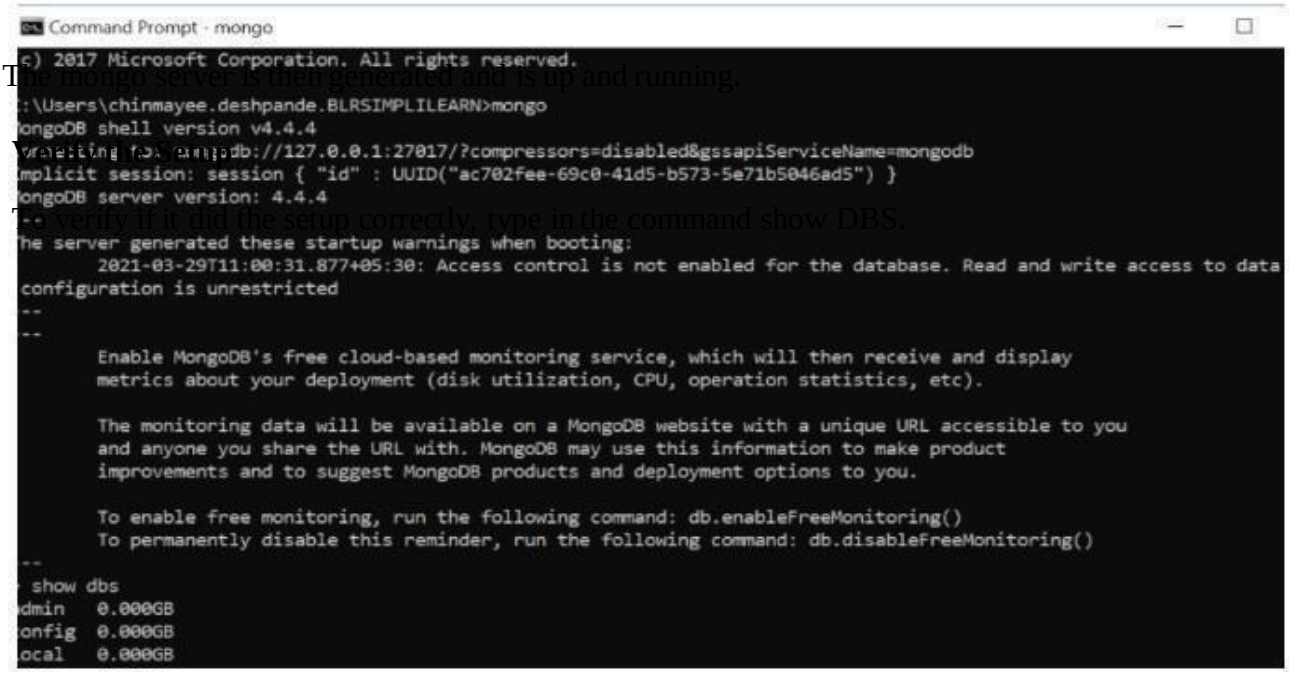


Execute the Mongo App.



Just type in mongo and

The mongo server is then generated and is up and running.



To verify if it did the setup correctly, type in the command show DBS.

With this we have successfully installed MongoDB on your Windows system.