DATABASE MANAGEMENT SYSTEMS
(22A0583)
LABORATORY MANUAL

B.TECH

(IIYEAR–ISEM)

(2023-24)

PREPARED BY
Dr. A.V.H SAI PRASAD
Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

MALLAREDDY COLLEGE OF ENGINEERING &TECHNOLOGY
(Autonomous Institution–UGC, Govt.ofIndia)
Recognizedunder2(f)and12(B) of UGCACT1956
(AffiliatedtoJNTUH,Hyderabad,ApprovedbyAICTE-AccreditedbyNBA&NAAC–'A'Grade-ISO9001:2015Certified)
Maisammaguda,Dhulapally(PostVia.Hakimpet),Secunderabad–500100,TelanganaState,India

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision

- To achieve high quality in technical education that provides the skills and attitude to adapt to the global needs of the Information Technology sector, through academic and research excellence.

Mission

- To equip the students with the cognizance for problem solving and to improve the teaching learning pedagogy by using innovative techniques.
- To strengthen the knowledge base of the faculty and students with motivation towards possession of effective academic skills and relevant research experience.
- To promote the necessary moral and ethical values among the engineers, for the betterment of the society.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1 : PROFESSIONALISM & CITIZENSHIP

To create and sustain a community of learning in which students acquire knowledge and learn to apply it professionally with due consideration for ethical, ecological and economic issues.

PEO2: TECHNICAL ACCOMPLISHMENTS

To provide knowledge based services to satisfy the needs of society and the industry by providing hands on experience in various technologies in core field

PEO3: INVENTION, INNOVATION AND CREATIVITY

To make the students to design, experiment, analyze, interpret in the core field with the help of other multi-disciplinary concepts wherever applicable.

PEO4: PROFESSIONAL DEVELOPMENT

To educate the students to disseminate research findings with good soft skills and become a successful entrepreneur.

PEO5: HUMAN RESOURCE DEVELOPMENT

To graduate the students in building national capabilities in technology, education and research.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B. Tech Information Technology, the graduates will have the following Program Specific Outcomes:

1. **Fundamentals and critical knowledge of the Computer System:-** Able to Understand the working principles of the computer System and its components , Apply the knowledge to build, asses, and analyze the software and hardware aspects of it .
2. **The comprehensive and Applicative knowledge of Software Development:** Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.
3. **Applications of Computing Domain & Research:** Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify the research gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design / development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.
12. **Life- long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY

Maisammaguda, Dhulapally Post, Via Hakimpet, Secunderabad – 500100

DEPARTMENT OF INFORMATION TECHNOLOGY

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting 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 to the 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 lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) 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 lab observation 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 **Principal**

INDEX

S.No	Week.No	List of Experiments	Page No
1	WEEK-1	Concept design with E-R Model: Apply cardinalities for each relationship, identify strongentities and weak entities for relationships like generalization, aggregation, specialization	19
2	WEEK-2	Relation Model: Represent attributes as columns intables and different types of attributes.	31
3	WEEK-3	Normalization-Types of Normal Forms	39
4	WEEK-4	Practicing DDL commands(Create, Alter, Drop)	43
5	WEEK-5	Practicing DML Commands(Select, Insert, Update, delete)	49
6	WEEK-6	A) Querying(Using Any, All, Union, Intersect, Join, Constraints etc) B) Nested, Correlated Subqueries	53
7	WEEK-7	Queries Using Aggregate Functions, Group By and Having Clause, Creation and Dropping of Views	63
8	WEEK-8	Triggers(Creation of Insert Trigger, Delete Trigger and Update Trigger)	69
9	WEEK-9	Procedures(Using IN, OUT and INOUT Parameters)	77
10	WEEK-10	Usage of Cursors	83
11	WEEK-11	Installation of MySQL/MongoDB and Practicing DDL commands	87
12	WEEK-12	Practicing DCL(Grant, Revoke) and TCL Commands(Commit, Rollback and Savepoint)	100
13	-	CASESTUDY 1	109
14	-	CASESTUDY 1	110

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 inter section 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 attractivequery language
- Retrieval of information stored in more than one table
- An RDBMS product has to satisfy at least Seven of the 12 rules
  of E.F Codd to be accepted as a full- fledged RDBMS.

Relational Database Management System

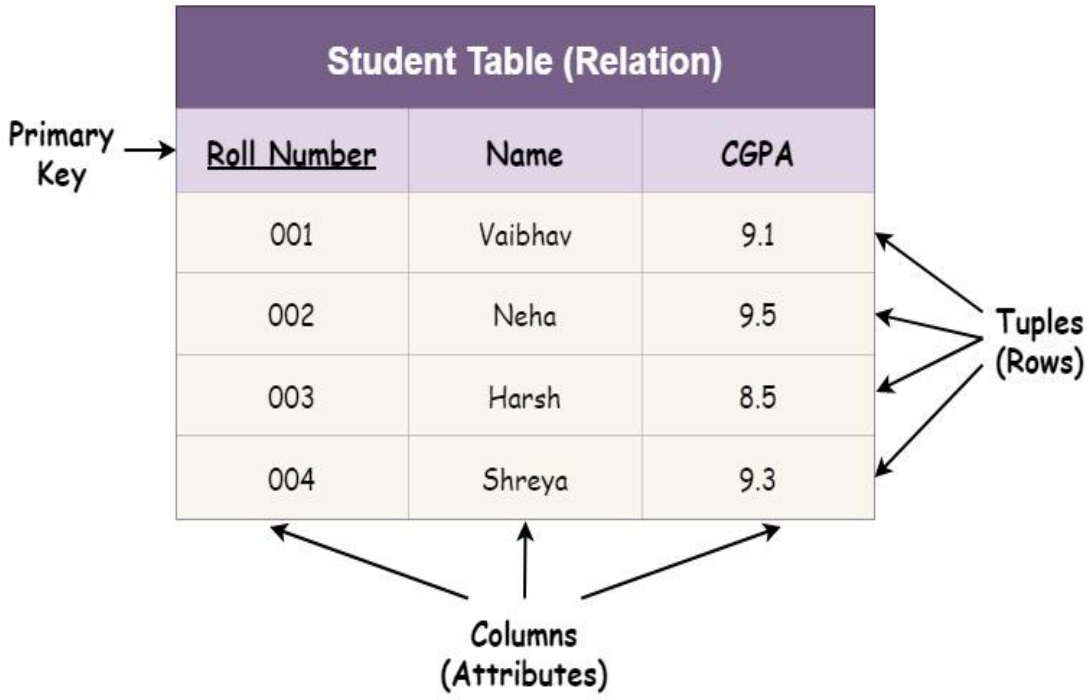
RDBMS is acronym for Relation Database Management System. Dr. E. F. Codd first introduced the Relational Database Model in 1970. The Relational model allows data to be represented in a simple row- column. Each data field is considered as a column and each record is considered as a row. Relational Database 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.

Importance of Relational Model?

The relational model for database management is an approach to logically represent and manage the data stored in a database. In this model, the data is organized into a collection of two-dimensional inter-related tables, also known as relations. Each relation is a collection of columns and rows, where the column represents the attributes of an entity and the rows (or tuples) represents the records.

Simple example of Relational model is as follows:

Relational Model in DBMS



As we can notice from the above relation:

- Any given row of the relation indicates a student i.e., the row of the table describes a real-world entity.

- The columns of the table indicate the attributes related to the entity. In this case, the roll number, CGPA, and the name of the student

Students Table

Student	ID*
John Smith	084
Jane Bloggs	100
John Smith	182
Mark Antony	219

Activities Table

ID*	Activity1	Cost1	Activity2	Cost2
084	Tennis	$36	Swimming	$17
100	Squash	$40	Swimming	$17
182	Tennis	$36		
219	Swimming	$15	Golf	$47

Here, both tables are based on student's details. Common field in both tables is ID.So we can say both tables are related with each other through Student ID column.

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 themost 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)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 storage representation or access methods.

6. The Logical Data Independence Rule

The changes that are made should not affect the user's ability to work with the data. 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

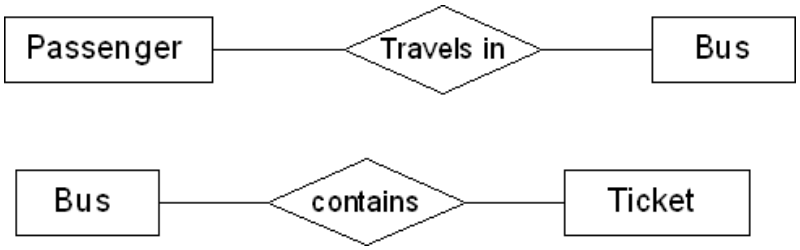
AIM: Concept Design with E-R Model

ER-Model:

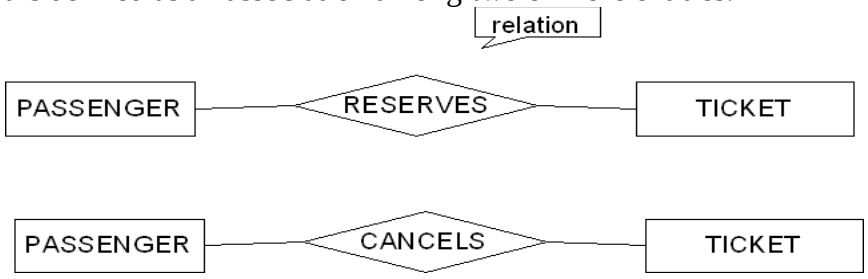
Relate the entities appropriately. Apply cardinalities for each relationship. Identify strong entities and weak entities (if any). Indicate the type of relationship (total/partial). Tryto incorporate generalization, aggregation, specialization etc wherever required.

The Following are the entities:

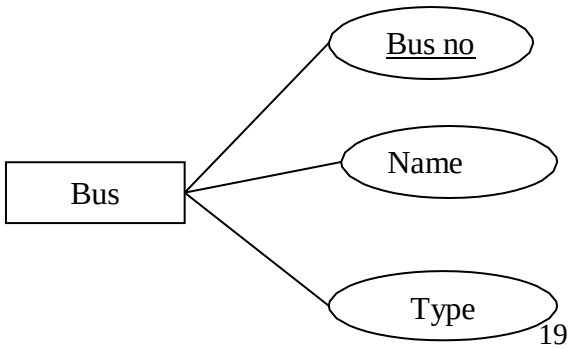
1. **Bus**
2. **Reservation**
3. **Ticket**
4. **Passenger**
5. **Cancellation**



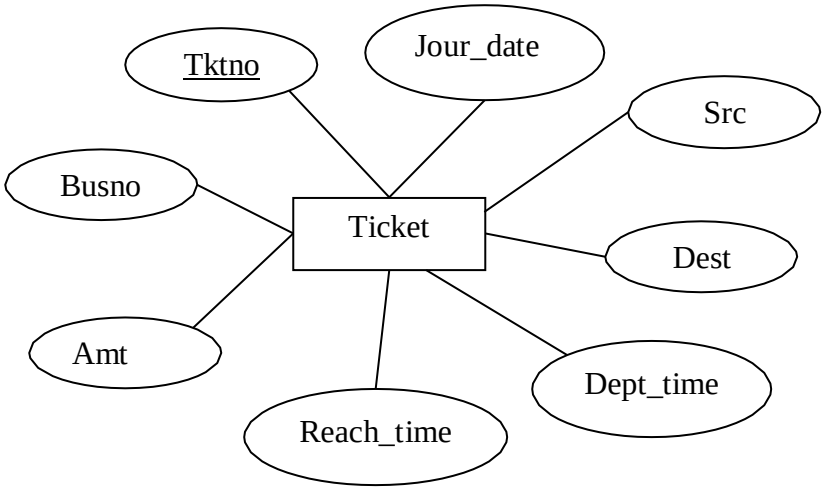
Relationship: - it is defined as an association among two or more entities.



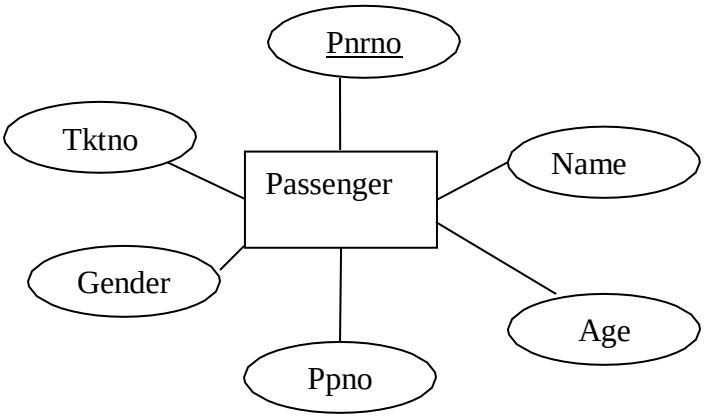
Entity diagram for BUS



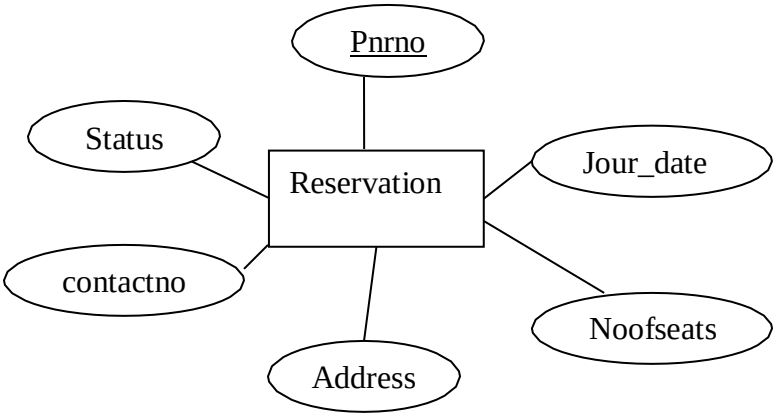
Entity diagram for *Ticket*



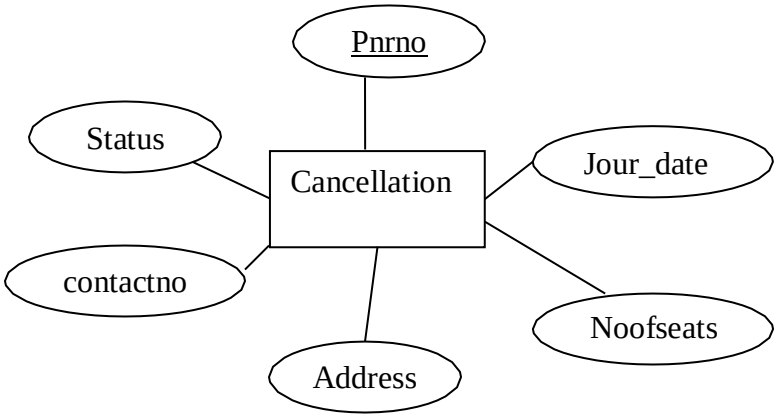
Entity diagram for *Passenger*



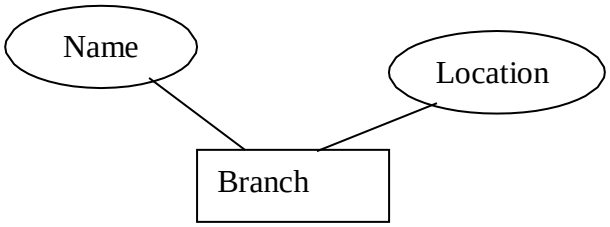
Entity diagram for *Reservation*



Entity diagram for *Cancellation*



Entity diagram for *Branch*



The cardinality ratio: - specifies the number of entities to which another entity can beassociated via a relationship set.

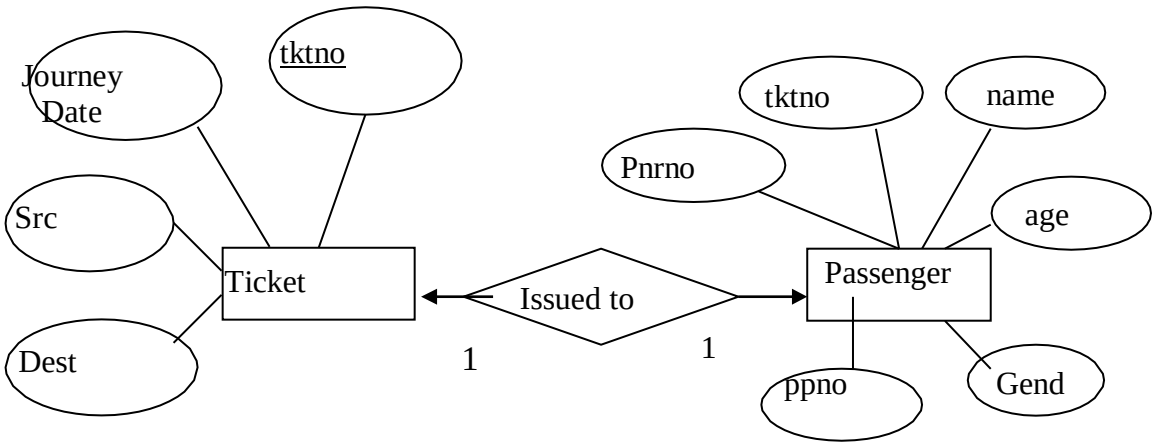
For a binary relationship set R between entity sets A & B, the mapping cardinality must be one of the following:

The cardinality ratio: - specifies the number of entities to which another entity can beassociated via a relationship set.

For a binary relationship set R between entity sets A & B, the mapping cardinality mustbe one of the following:

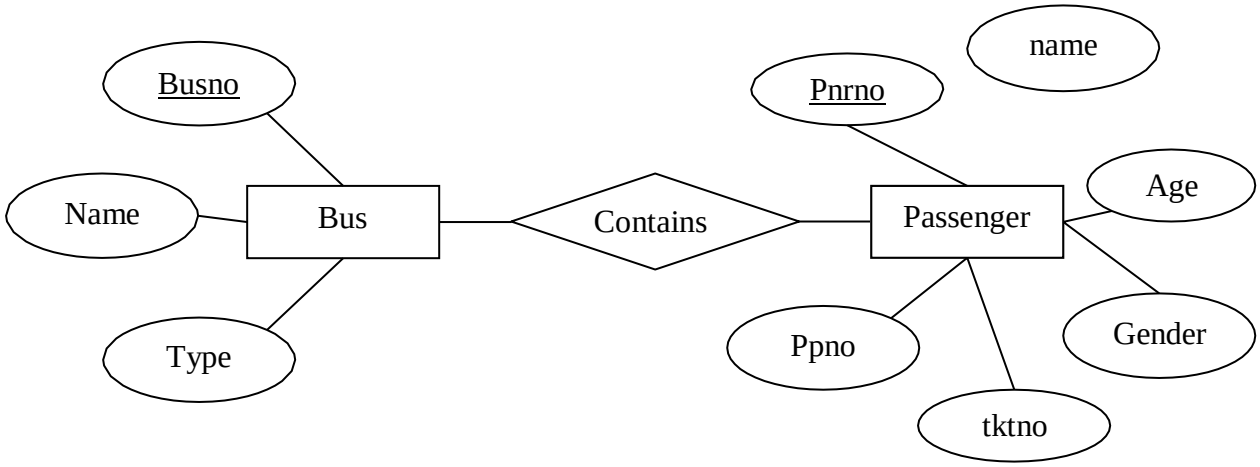
1. **One-to-One:** An entity in A is associated with at most one entity in B and vice versa.

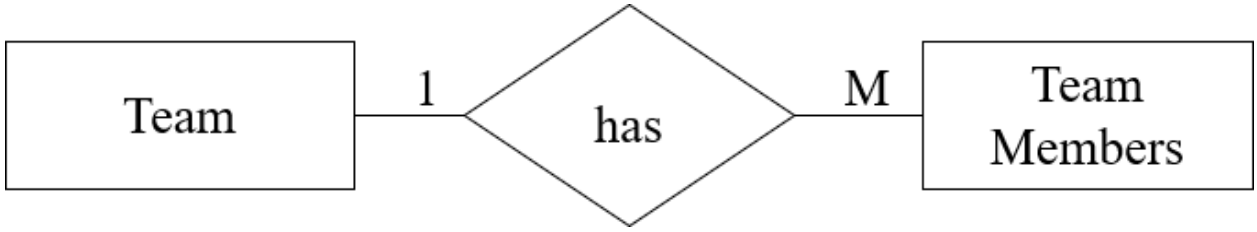
   Ex: “Issued to” relation between ticket and passenger entities.



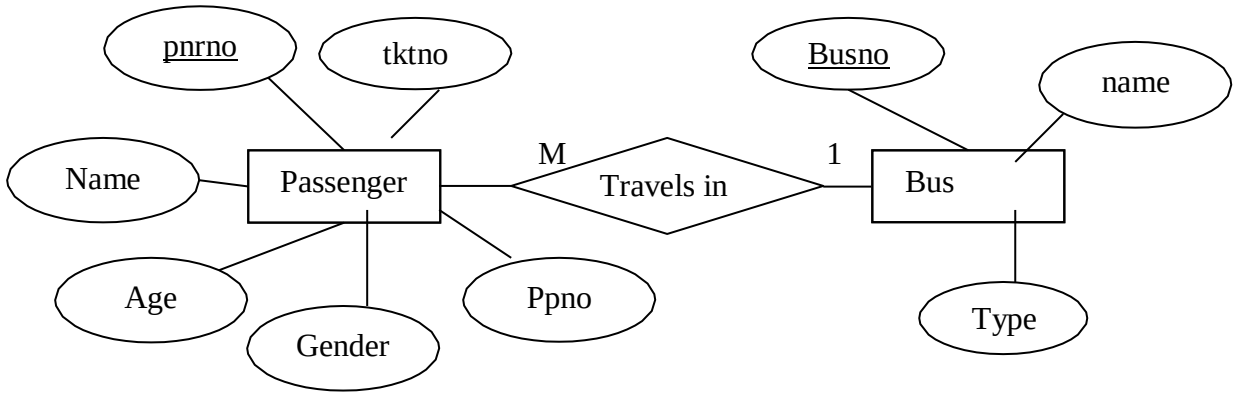
2. **One-to-many:** An entity in A is associated with any no. of entities in B. An entity in B isat most associated with at most one entity in A.

Ex: “contains” relation between bus and passenger entities.

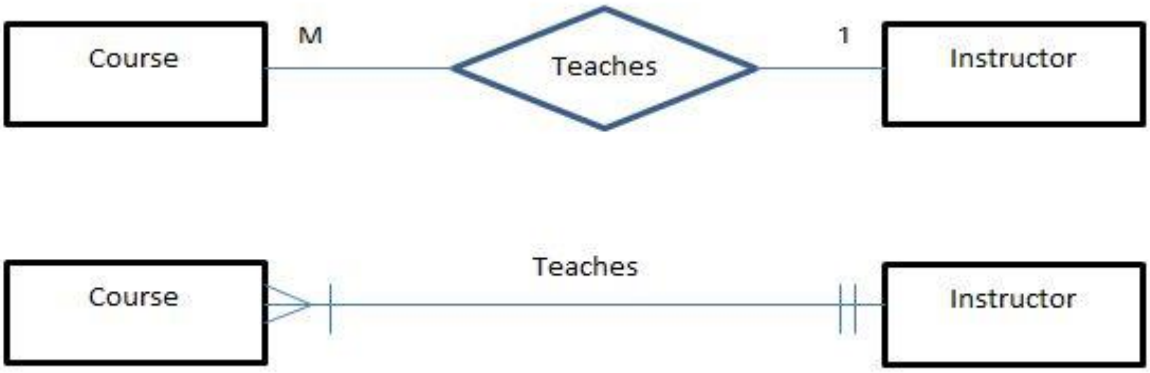




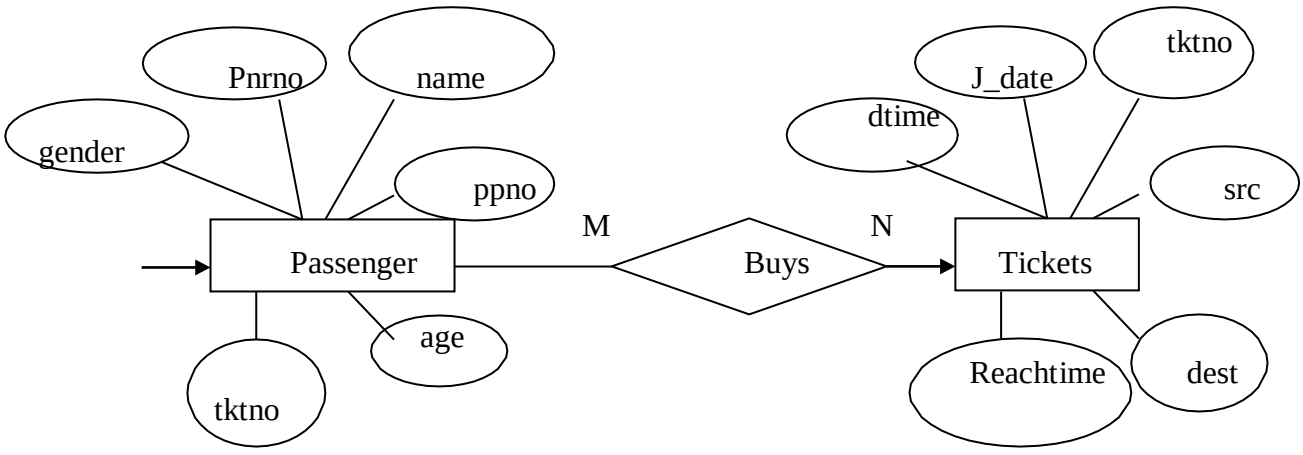
3. **Many-to-One:** An entity in A is associated with at most one entity in B. However, an entity in B can be associated with any no. of entities in A.

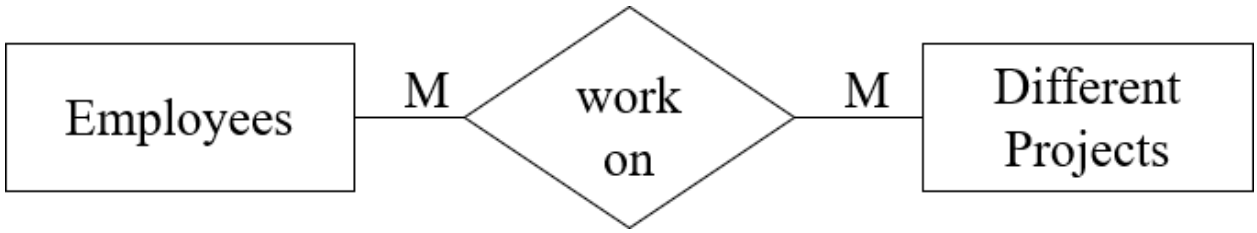


Ex: "Travels in" relation between passenger and bus entities.



4.Many-to-many: An entity in A is associated with an no. of entities in B and an entity inB can be associated with any no. of entities in A.

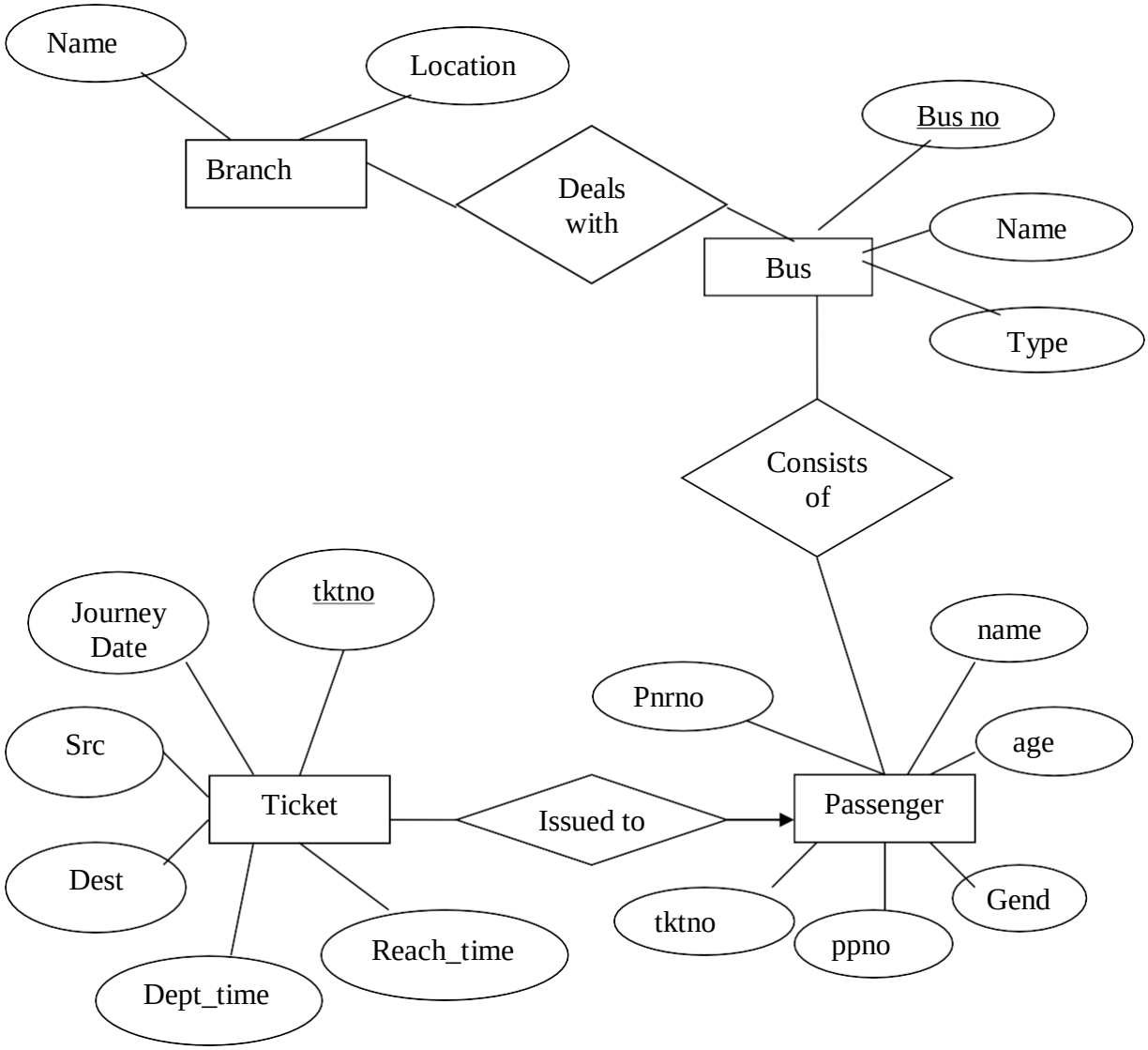




Types of entities :-

Weak and strong entity: - an entity set may not have sufficient attributes to form a primary key. Such an entity set is termed a weak entity set. An entity set that has primary key is termed a strong entity set.

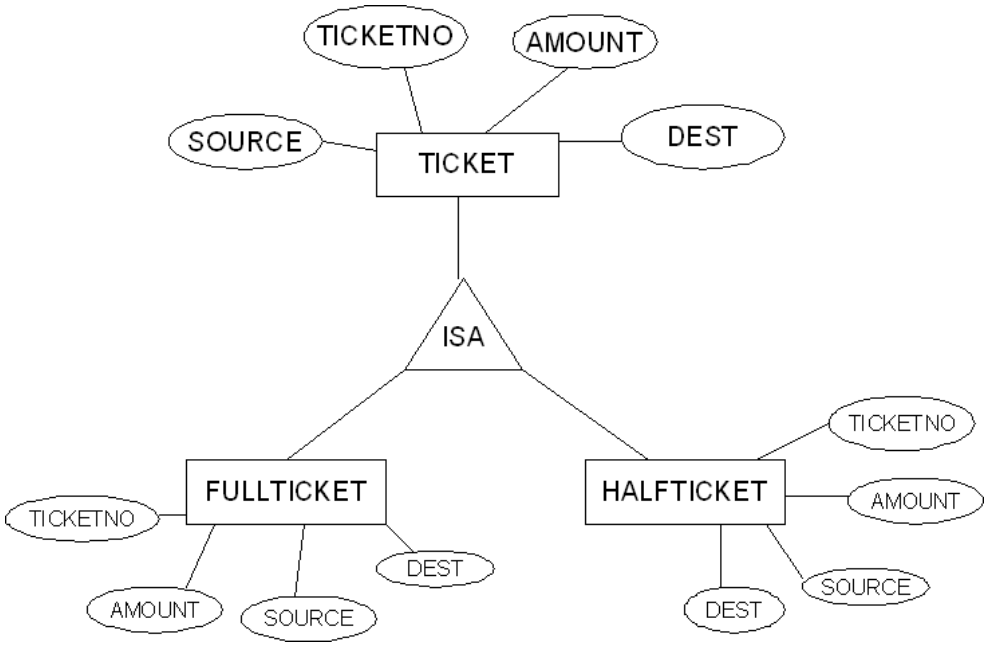
Entity Relationship diagram consisting of Bus, ticket, Passenger and Branch entities:



1. **Generalization**: It consists of identifying some common characteristics of a collection of entity set and creating new entity set that contains entities possessing these common characteristics.

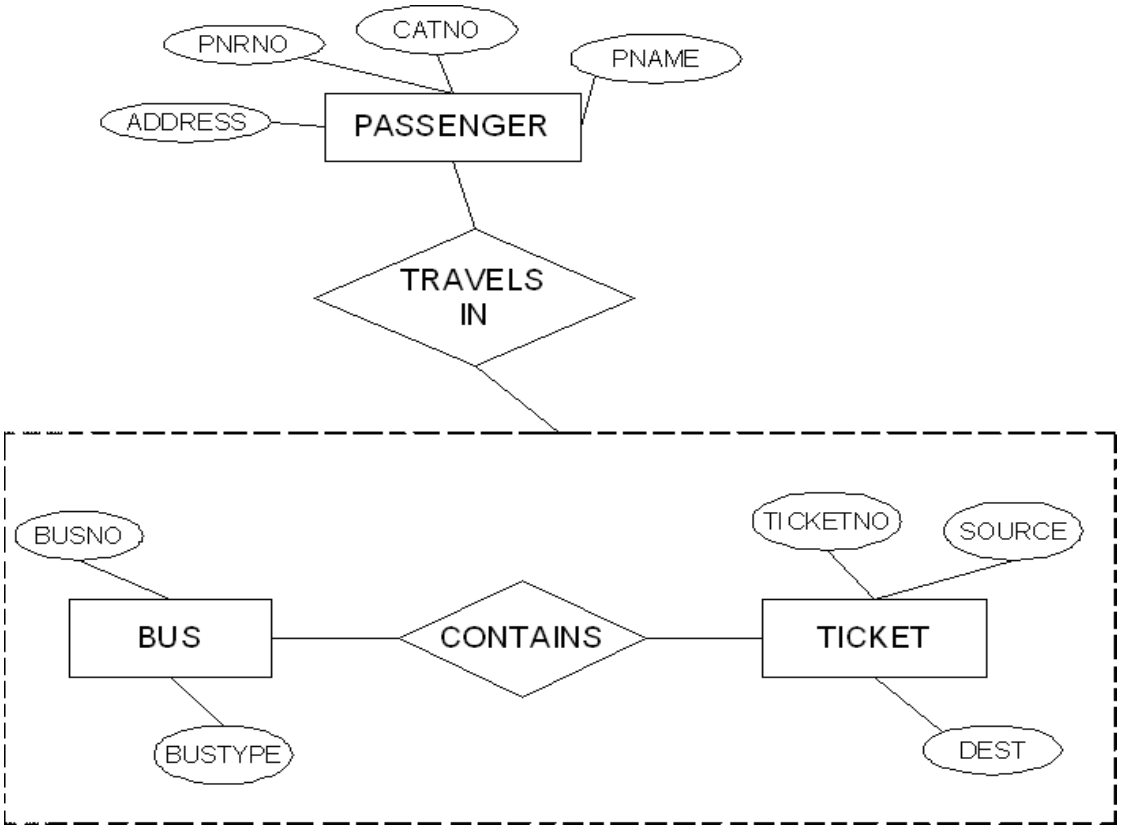
   It is defined by using 'ISA' (Is a) relationship.

Ex:



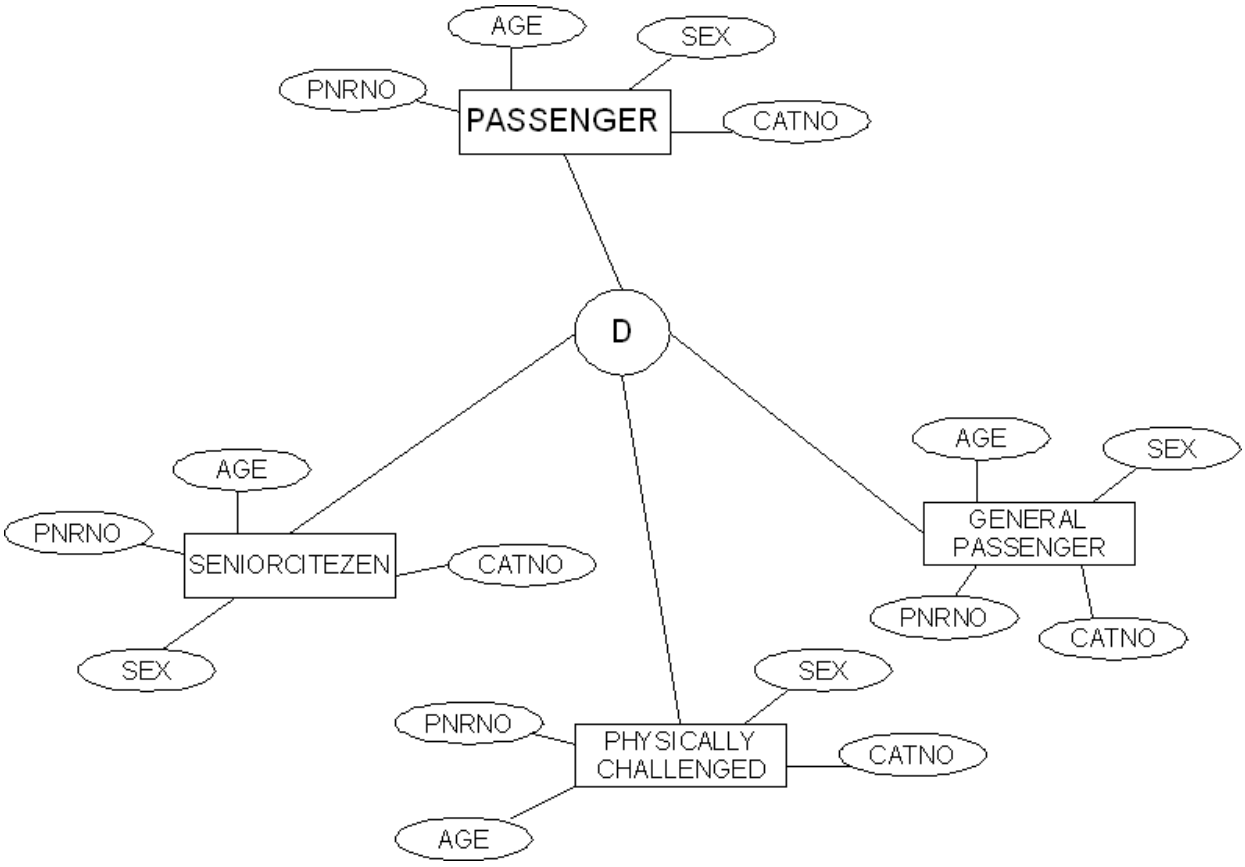
2. **Aggregation:** It allows us to indicate that a relationship set participates in another relationship set.

Ex:

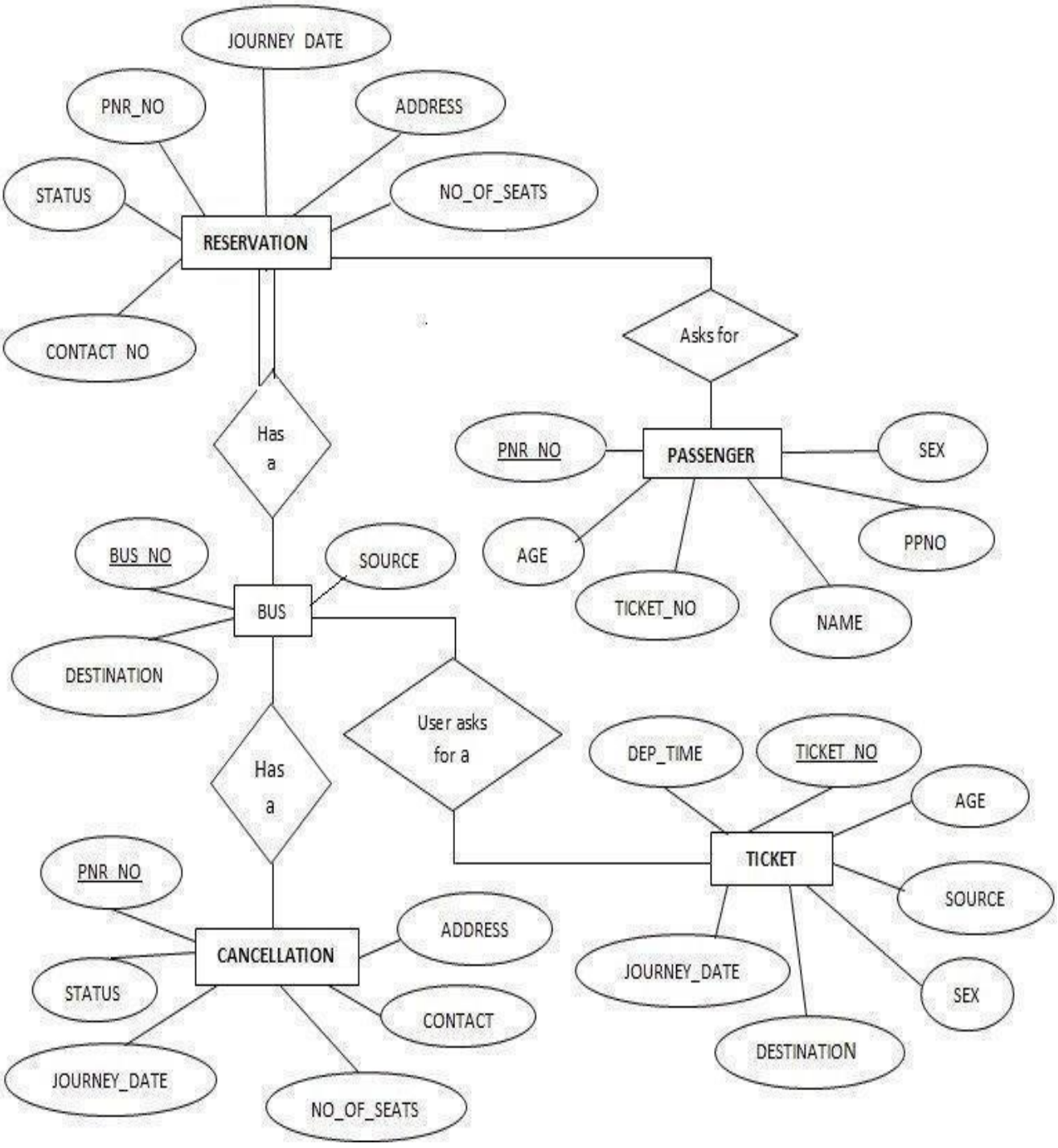


3. **Specialization**: It is the process of identifying subsets of an entity set (the super set) that share some distinguishing characteristics. This entity type is called the super class of the specialization.

Ex:

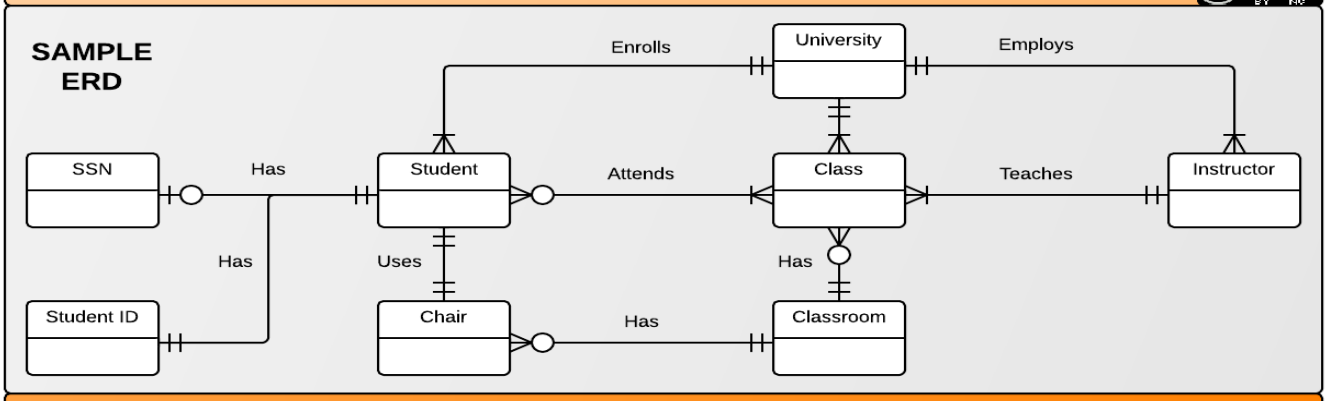


Concept design with E-R Model:

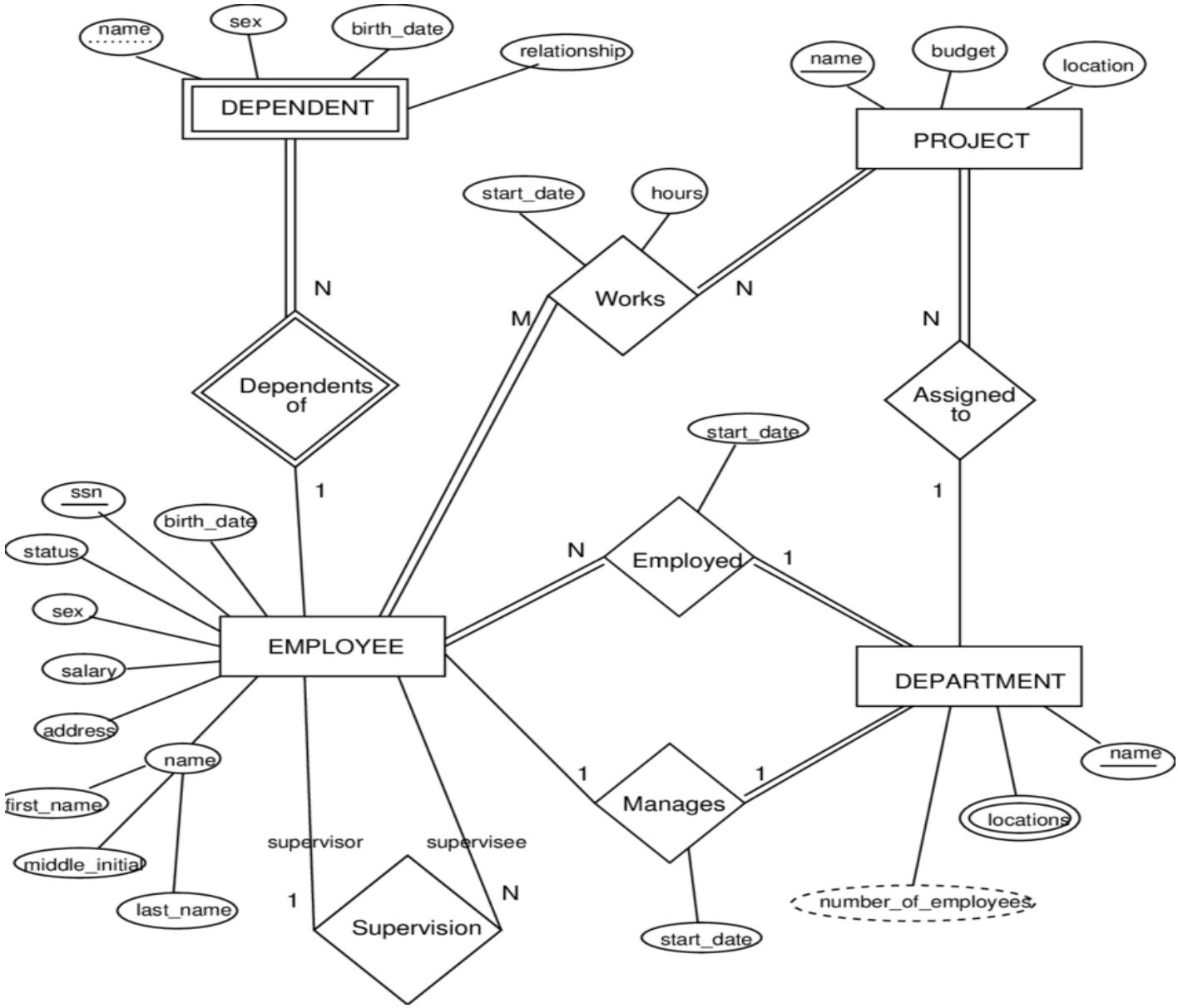


ERD "Crow's Foot" Relationship Symbols [Quick Reference]

Created by Vivek M. Chawla | @VivekMChawla | April 7, 2013



Notation	Meaning	Example
	Relationship	Student — Enrolls — University
	One	Student — Has — Student ID Number
	Many	Student — Attends — Class
	One and ONLY One	Student — Uses — Chair
	Zero or One	Student — Has — Social Security Number
	One or Many	Instructor — Teaches — Class
	Zero or Many	Classroom — Has — Chair



Viva Questions

1. What is SQL?

2. what is entity and relationship?

3. What is DBMS?

4. What is a Database system?

5. Advantages of DBMS?

6.How many types of database languages are available?

WEEK: 2
Relational Model

Represent all entities and all relationships ina tabular fashion

The following are tabular representation of the above entities and relationships

1.BUS:

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

2.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	

3.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)	

4.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

5.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

Mysql>create table Bus(BusNo varchar(10),source varchar(20),Destinationvarchar(20),coachType varchar(10),primary key(BusNo));

Mysql>desc Bus;

```
mysql> use cse;
Database changed
mysql> create table Bus(BusNo varchar(10),source varchar(20),Destination varchar(20),coachType varchar(10),primary key(BusNo));
Query OK, 0 rows affected (0.06 sec)

mysql> desc Bus;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| BusNo       | varchar(10) | NO   | PRI |         |       |
| source      | varchar(20) | YES  |     | NULL    |       |
| Destination | varchar(20) | YES  |     | NULL    |       |
| coachType   | varchar(10) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql>
```

Ticket:

Ticket(TicketNo: string, DOJ: date, Address:string,ContactNo: string, BusNo:String ,SeatNo:Integer,Source: String, Destination: String)

ColumnName	Datatype	Constraints	Type of Attributes
TicketNo	Varchar(20)	Primary Key	Single-valued
DOJ	Date		Single-valued
Address	Varchar(20)		Composite
ContactNo	Integer		Multi-valued
BusNo	Varchar(10)	Foreign Key	Single-valued
SeatNo	Integer		Simple
Source	Varchar(10)		Simple
Destination	Varchar(10)		Simple

Mysql>create table Ticket(TicketNo varchar(20),DOJ date,Address varchar(20),ContactNo varchar(15)BusNo varchar(10),SeatNo int,Source varchar(10),primary key(TicketNo,BusNo),foreign key(BusNo) references Bus(BusNo));

Mysql>desc Ticket;

```
mysql> create table Ticket (TicketNo varchar(20),DOJ date,Address varchar(20),ContactNo varchar(15),BusNo varc
har(10),seatNo int,Source varchar(10),Destination varchar(10),primary key(TicketNo,BusNo),foreign key(BusNo)re
ferences Bus(BusNo));
Query OK, 0 rows affected (0.05 sec)

mysql> desc Ticket;
+-------------+------------+------+-----+---------+-------+
| Field       | Type       | Null | Key | Default | Extra |
+-------------+------------+------+-----+---------+-------+
| TicketNo    | varchar(20) | NO   | PRI |         |       |
| DOJ         | date       | YES  |     | NULL    |       |
| Address     | varchar(20) | YES  |     | NULL    |       |
| ContactNo   | varchar(15) | YES  |     | NULL    |       |
| BusNo       | varchar(10) | NO   | PRI |         |       |
| seatNo      | int(11)    | YES  |     | NULL    |       |
| Source      | varchar(10) | YES  |     | NULL    |       |
| Destination | varchar(10) | YES  |     | NULL    |       |
+-------------+------------+------+-----+---------+-------+
8 rows in set (0.00 sec)

mysql>
```

Passenger:

Passenger(PassportID:String,TicketNo:string,Name:String, ContactNo:string,Age: integer, Sex: character, Address: String);

ColumnName	Datatype	Constraints	Type of Attributes
PassportID	Varchar(15)	Primary Key	Single-valued
TicketNo	Varchar(20)	Foreign Key	Single-valued
Name	Varchar(20)		Composite
ContactNo	Varchar(20)		Multi-valued
Age	Integer		Single-valued
Sex	character		Simple
Address	Varchar(20)		Composite

Mysql> Create table passenger(passportID varchar(15) ,TicketNo varchar(15),Name varchar(15),ContactNo varchar(20),Age integer, sex char(2),address varchar(20), primary key(passportID,TicketNo),foreign key(TicketNo) references Ticket(TicketNo));

Mysql> desc passenger;

```
mysql> use cse;
Database changed
mysql> create table passenger(passportid varchar(10),ticketno varchar(15),name varchar(15),contactno varchar(1
5),age integer,sex char(2),address varchar(20),primary key(passportid,ticketno),foreign key(ticketno) referenc
es ticket(ticketno));
Query OK, 0 rows affected (0.08 sec)

mysql> desc passenger;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportid | varchar(10) | NO   | PRI |         |       |
| ticketno   | varchar(15) | NO   | PRI |         |       |
| name       | varchar(15) | YES  |     | NULL    |       |
| contactno  | varchar(15) | YES  |     | NULL    |       |
| age        | int(11)     | YES  |     | NULL    |       |
| sex        | char(2)     | YES  |     | NULL    |       |
| address    | varchar(20) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
7 rows in set (0.03 sec)
```

Reservation:

Reservation(PNRNo: String, DOJ: Date, NoofSeats: integer , Address: String ,ContactNo:String, , BusNo: String,SeatNo:Integer)

ColumnName	Datatype	Constraints	Type of Attributes
PNRNo	**Varchar(20)**	**Primary Key**	**Single-valued**
DOJ	**date**		**Single-valued**
No_of_Seats	**Integer**		**Simple**
Address	**Varchar(20)**		**Composite**
ContactNo	**Varchar(10)**		**Multi-valued**
BusNo	**Varchar(10)**	**Foreign Key**	**Single-valued**
SeatNo	**Integer**		**Simple**

Mysql> Create table Resevation(PNRNo varchar(20),DOJ date,NoofSeates integer,Address varchar(20),ContactNo varchar(20),BusNo varchar(20),SeatNo integer, primary key(PNRNo,BusNo),foreign key(BusNo) references Bus(BusNo));

Mysql> desc reservation;

```
mysql> create table reservation(PNRNo varchar(20),DOJ date,NofSeats integer,Address varchar(20),ContactNo varc
har(20),BusNo varchar(20),SeatNo integer,primary key(PNRNo,BusNo),foreign key(BusNo) references Bus(BusNo));
Query OK, 0 rows affected (0.05 sec)

mysql> desc Reservation;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| PNRNo     | varchar(20) | NO   | PRI |         |       |
| DOJ       | date        | YES  |     | NULL    |       |
| NofSeats  | int(11)     | YES  |     | NULL    |       |
| Address   | varchar(20) | YES  |     | NULL    |       |
| ContactNo | varchar(20) | YES  |     | NULL    |       |
| BusNo     | varchar(20) | NO   | PRI |         |       |
| SeatNo    | int(11)     | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

Cancellation:

Cancellation(PNRNo: String,DOJ: Date, SeatNo: integer,ContactNo: String,Status:String)

ColumnName	Datatype	Constraints	Type of Attributes
PNRNo	**Varchar(10)**	**Primary Key**	**Single-valued**
DOJ	**date**		**Single-valued**
SeatNo	**Integer**		**Simple**
ContactNo	**Varchar(15)**		**Multi-valued**
Status	**Varchar(10)**		**Simple**

Mysql> create table cancellation(PNRNo varchar(10),DOJ date,SeatNo integer, ContactNo varchar(15),Status varchar(10), primary key(PNRNo), foreign key(PNRNo) references reservation(PNRNo));

Mysql> desc cancellation;

```
mysql> create table cancellation(PNRNo varchar(10),DOJ date,SeatNo integer,ContactNo varchar(15),Status varcha
r(10),primary key(PNRNo),foreign key(PNRNo) references Reservation(PNRNo));
Query OK, 0 rows affected (0.05 sec)

mysql> desc cancellation;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| PNRNo     | varchar(10) | NO   | PRI |         |       |
| DOJ       | date        | YES  |     | NULL    |       |
| SeatNo    | int(11)     | YES  |     | NULL    |       |
| ContactNo | varchar(15) | YES  |     | NULL    |       |
| Status    | varchar(10) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Conclusion: The Student is able draw the tabular representation of the relations of Roadway travels.

Viva Questions:

1. What is the difference between SUM and COUNT ?

2. What is VIEW ? and What will you get when you use VIEW

3. What is difference between TRUNCATE and DELETE?

SAMPLE DATA IN TABLES:

1.Bus

Busno	name	type
100	Xyz	a/c
101	Pop	Non a/c
102	Xxx	a/c

2.Reservation:

Pnrno	Jour_date	Noofseats	Address	Contactno	Status
1001	20-07-10	4	Hyd	9492500000	Yes
1002	21-07-10	5	Sec	9492511111	Yes
1003	05-08-10	10	hyd	9949022222	No

3.Ticket:

Tktno	Jour_date	Src	Dest	amt	busno	Dept_time	Reach_time
10001	20-07-10	Hyd	Delhi	800	100	06:00	22:00
10002	21-07-10	Hyd	Chennai	700	101	08:00	23:00
10003	05-08-10	Delhi	Hyd	800	102	06:00	22:00

4.Passenger:

Pnrno	name	tktno	age	Gender	ppno
1001	Alekhya	10001	25	F	ff11112
1002	Krupani	10002	27	F	ff22332
1003	Prathima	10003	28	F	F234444
1004	Prem	10004	30	M	Ff202020

5.Cancellation:

Pnrno	Joudate	Noofseats	Address	Contact_no	Status
1001	20-07-10	4	Hyd	9492500000	Yes
1002	21-07-10	5	Sec	9492511111	Yes

WEEK: 3

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 of logicalor 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, if 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, foreignkey(PNRNO) references Reservation(PNRNO));

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

The normalization forms are:

1. **First Normal Form**: 1NF requires that the values in each column of a table are atomic. By atomic we mean that there are no sets of values within a column.
2. **Second Normal Form**: where the 1NF deals with atomicity of data, the 2NF deals with relationships between composite key columns and non-key columns. To achieve 2NF the tables should be in 1NF. The 2NF any non-key columns must depend on the entire primary key. In case of a composite primary key, this means that non-key column can't depend on only part of the composite key.
3. **Third Normal Form**: Any transitive dependencies have been removed.
4. **Boyce/Codd normal Form**: Any remaining anomalies that result from functional dependencies have been removed.
5. **Fourth Normal Form**: Any multi valued dependencies have been removed.
6. **Fifth Normal Form**: Any remaining anomalies have been removed.

Applying Normalization to our Entities
Consider Passenger Entity

- A Passenger may consist of two phone numbers, but atomic values should be there. so, we normalize the relation as follows:

Passenger:

Pnrno	pname	age	Gender	ticketno	address	phno
2001	Alekhya	25	F	1111	H.no:101	9999900000 9999911111
2002	Krupani	26	F	2222	H.no:102	9999912345
2003	pratima	28	F	3333	H.no:103	9000000000

Pnrno	pname	age	Gender	ticketno	address	phno
2001	Alekhya	25	F	1111	H.no:101	9999900000
2001	Alekhya	25	F	1111	H.no:101	9999911111
2002	Krupani	26	F	2222	H.no:102	9999912345
2003	pratima	28	F	3333	H.no:103	9000000000

The above relation is now in 1NF and the relation is 2NF as there are no partial functional dependencies and the relation is also in 3NF as there are no transitive dependencies.

Normalization of Bus entity:

Bus:

Busno	serviceno	source	destination	bustype	Noofseats
1001	3300	Hyd	Delhi	A/c	20
1002	4400	Hyd	Chennai	A/c	28
1003	5500	Hyd	Bglore	Non a/c	30

In this relation the values in each column are atomic so it is already in 1NF. In the Bus entity **Busno+serviceno** is the primary key.

There exists following partial dependencies.

Busno -------> Bustype,Noofseats

Serviceno -------> Source,Dest

So the relation will be in 2NF as follows

Busno	serviceno
1001	3300
1002	4400
1003	5500

Busno	bustype	Noofseats
1001	A/c	20
1002	A/c	28
1003	Non a/c	30

serviceno	source	destination
3300	Hyd	Delhi
4400	Hyd	Chennai
5500	Hyd	Bglore

The above relation is 2NF. And all columns directly depend on primary key. So there is no transitive dependency and the relation is 3NF.

Normalization of Ticket entity:

Ticketno	Joudate	Source	Destination	Amount	Catcard
1111	2010-10-08	Hyd	Delhi	1200	No
2222	2010-10-08	Hyd	Chennai	1000	Yes
3333	2010-08-08	Hyd	Bglore	800	Yes

In this relation the values in each column are atomic so it is already in 1NF.

In the above relation there are no partial functional dependencies so the relation is in 2NF. The ticket entity might face the following transitive dependency

Ticketno ---------- > catcard

Catcard ---------- >amount

So the relation is not in 3NF.

Ticketno	Joudate	Source	Destination	Catcard
1111	2010-10-08	Hyd	Delhi	No
2222	2010-10-08	Hyd	Chennai	Yes
3333	2010-08-08	Hyd	Bglore	Yes

Put the catcard and amount attributes in a separate table. Then the relation should be in 3NF.

Catcard	Amount
No	1200
Yes	1000
Yes	800

The above relation is 3NF as we have eliminated the transitive dependencies.

The above relation is 3NF as we have eliminated the transitive dependencies.

Finally all the tables are normalized and free from data redundancy, partial functional dependencies and transitive dependencies.

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?

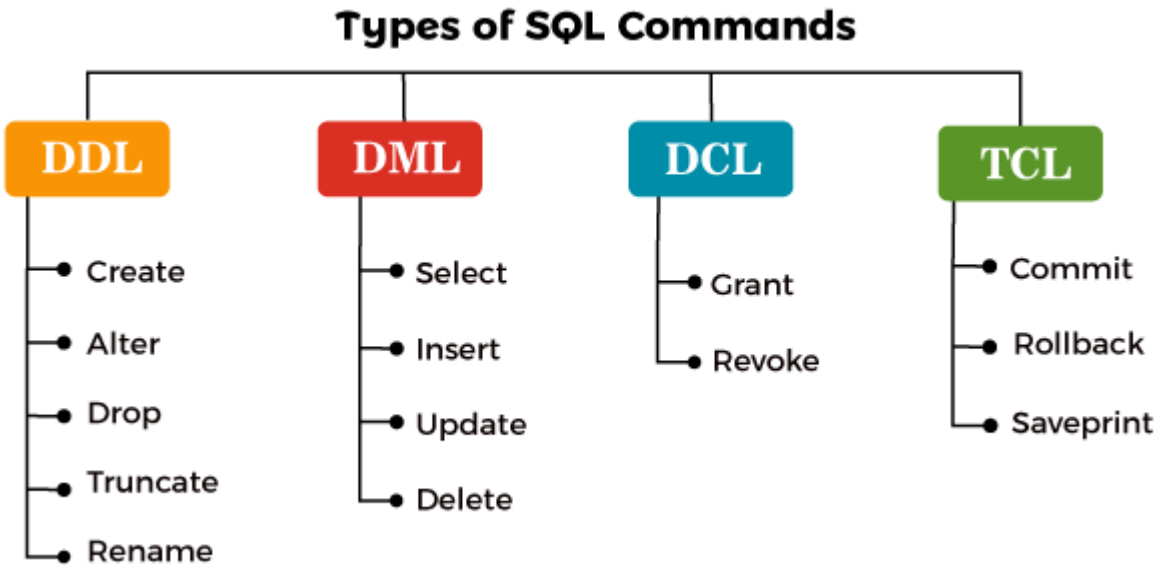
WEEK-4

Aim: PRACTICING DDL COMMANDS on Road Way travels Tables.

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
- Rename 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?

WEEK- 5

AIM: Practicing DML Commands on Road Way Travels

Tables.

DML COMMANDS

1. INSERTING DATA IN TO TABLE(INSERT)

Insert command is used to insert rows into the table.

SYNTAX:
INSERT INTO table_name values (column_name1, column_name2,….column__name n)

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

SYNTAX

INSERT IN TO table_name (column_name1, column_name2,….column_name 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);
```

2. UPDATE Date Into Table(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:

3.DELETE date from table(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(SELECT)

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

SYNTAX

SELECT * FROM TABLENAME;

Example:

SQL> select * from Passenger;

TESTOUTPUT:

b. Elimination of duplicates from the select statement

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

EXAMPLE QUERY:

MySQL>select distinct age from Passenger;

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?

WEEK -6

AIM : A)QUERYING USING ANY, ALL, IN, UNION,INTERSECT, JOIN,CONSTRAINTS etc.)

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.

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:

AIM: B) Nested, Correlated Subqueries

MySQL> select * from reserves;

SID	BID	DAY
22	101	10-OCT-98
22	102	10-OCT-98
22	103	08-OCT-98
22	104	07-OCT-98
31	102	10-NOV-98
31	103	06-NOV-98
31	104	12-NOV-98
64	101	05-SEP-98
64	102	08-SEP-98
74	103	08-SEP-98

10 rows selected.

MySQL> select * from sailors;

SID	SNAME	RATING	AGE
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35
64	Horataio	7	35
71	Zorba	10	16
74	Horataio	9	35
85	Art	3	25.5
95	Bob	3	63.5

10 rows selected.

MySQL> select * from boats;

BID	BNAME	COLOR
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

1. If boat Number is 103.Then find the name of sailors?(Using Joins)

```
select s.sname
from sailors s,reserves r
where s.sid=r.sid and r.bid=103;

Output:

SNAME
--------------------
Dustin
Lubber
Horataio
```

2. Find the names of sailors who have never reserved boat number 103.(using Joins)

The SQL IN condition (sometimes called the IN operator) allows you to easily check whether any value in a value list **Matches an expression. It is used in a SELECT, INSERT, UPDATE,** or DELETE statement to help reduce the need for multiple OR conditions.

The SQL NOT IN condition (sometimes called the IN operator) allows you to easily check whether any value in a value list **NOT Matches** an expression. It is used in a SELECT, INSERT, UPDATE or DELETE statement to help reduce the need for multiple OR conditions.

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

Output:
SNAME
--------------------
Zorba
Art
Horataio
Rusty
```

3. Find the name of Sailors who Reserved Red boats?(Using Joins)

```
select sname
from sailors s,boats b,reserves r
where s.sid=r.sid and b.bid=r.bid and b.color='red';

Output:
SNAME
--------------------
Dustin
Dustin
Lubber
Lubber
Horataio
```

4.What is the color of boat reverse by Lubber?(Using Joins)

```
select b.color
from boats b,sailors s,reserves r
where s.sid=r.sid and b.bid=r.bid and s.sname='Lubber';

COLOR
--------------------
red
green
red
```

5.Find the sids of sailors with age over 20 who have not reserved a red boat.(Using Joins)

```
select s.sid,s.sname
from sailors s,boats b,reserves r
where s.sid=r.sid and b.bid=r.bid and s.age>20 and b.color!='red';


 SID SNAME
---------- --------------------
        22 Dustin
        22 Dustin
        31 Lubber
```

64 Horataio
74 orataio

Nested Queries

Sailors(sid, sname, rating, age)

Reserve(sid, bid, day)

Boats(bid, bname, color)

The names of sailors that reserved boat 103

SELECT S.sname FROM Sailors S WHERE S.sid IN (SELECT R.sid FROM Reserve R WHERE R.bid = 103);

Correlated Subqueries

The name of correlated subqueries means that a subquery is correlated with the outer query. The correlation comes from the fact that the subquery uses information from the outer query and the subquery executes once for every row in the outer query.

SELECT S.sname FROM Sailors S WHERE EXISTS (SELECT * FROM Reserve R WHERE R.bid = 103);

**The nested query in this query is a correlated subquery.

Test that a relation satisfies some condition

… WHERE EXISTS (SELECT …) -TRUE if subquery result is not empty

SELECT S.sname

WHERE **EXISTS** (SELECT *

FROM Sailors S

FROM Reserves R

WHERE R.bid=103 AND S.sid=R.sid)

*Subquery is **CORRELATED** with parent query

Constraints:

```
CREATE TABLE Sailors ( sid int,
sname varchar(32),
rating int,
age double,
CONSTRAINT PK_Sailors PRIMARY KEY (sid) );
insert into Sailors (sid,sname,rating,age) values(22,'Dustin',7,45);
insert into Sailors (sid,sname,rating,age) values(29,'Brutus',1,33);
insert into Sailors (sid,sname,rating,age) values(31,'Lubber',8,55.5);
insert into Sailors (sid,sname,rating,age) values(32,'Andy',8,25.5);
insert into Sailors (sid,sname,rating,age) values(58,'Rusty',10,35);
insert into Sailors (sid,sname,rating,age) values(64,'Horatio',7,35);
insert into Sailors (sid,sname,rating,age) values(71,'Zorba',10,16);
insert into Sailors (sid,sname,rating,age) values(74,'Horatio',9,40);
insert into Sailors (sid,sname,rating,age) values(85,'Art',3,25.5);
insert into Sailors (sid,sname,rating,age) values(95,'Bob',3,63.5);

select * from Sailors;

CREATE TABLE Boats ( bid int,
bname varchar(32),
color varchar(32),
CONSTRAINT PK_Boats PRIMARY KEY (bid) );

insert into Boats (bid,bname,color) values (101,'Interlake','blue');
insert into Boats (bid,bname,color) values (102,'Interlake','red');
insert into Boats (bid,bname,color) values (103,'Clipper','green');
insert into Boats (bid,bname,color) values (104,'Marine','red');

select * from Boats;
```

```
CREATE TABLE Reserves ( sid int,
bid int,
day date,
CONSTRAINT PK_Reserves PRIMARY KEY (sid, bid, day),
FOREIGN KEY (sid) REFERENCES Sailors(sid),
FOREIGN KEY (bid) REFERENCES Boats(bid) );

insert into Reserves (sid,bid,day) values (22,101,'1998-10-10');
insert into Reserves (sid,bid,day) values (22,102,'1998-10-10');
insert into Reserves (sid,bid,day) values (22,103,'1998-10-8');
insert into Reserves (sid,bid,day) values (22,104,'1998-10-7');
insert into Reserves (sid,bid,day) values (31,102,'1998-11-10');
insert into Reserves (sid,bid,day) values (31,103,'1998-11-6');
insert into Reserves (sid,bid,day) values (31,104,'1998-11-12');
insert into Reserves (sid,bid,day) values (64,101,'1998-9-5');
insert into Reserves (sid,bid,day) values (64,102,'1998-9-8');
insert into Reserves (sid,bid,day) values (74,103,'1998-9-8');

select * from Reserves;
```

Ex:1

Find the name and age of the oldest sailor.

```
SELECT S.sname,S.age
FROM Sailors S
WHERE S.age >= ALL ( SELECT age
FROM Sailors);
```

Ex:2

For each rating, find the average age of sailors at that level of rating

```
SELECT S.rating, AVG(S.age) AS average FROM Sailors S
GROUP BY S.rating;
```

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?

WEEK 7

Aim: Queries Using Aggregate Functions (Sum, Avg, Max, Min and Count) ,Group By and Having Clause, Creation and Dropping of Views

There are various aggregate functions available in MySQL. Some of the most commonly used aggregate functions are summarised in the below table:

Aggregate Function	Descriptions
count()	ows, including rows with NULL values in a group.
sum()	It returns the total summed values (Non-NULL) in a set.
avg()	It returns the average value of an expression.
min()	It returns the minimum (lowest) value in a set.
max()	It returns the maximum (highest) value in a set.

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 PNR_NO, total Number of tickets booked by a passenger where the number of seats is greater than 1**

mysql> select PNR_NO,sum(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:

7. **Find out maximum age of a Passenger?**

   Mysql> select max(age) from passenger;

   (or)

   Mysql> select max(age) as max_age from passenger;

8. **Find out minimum age of a Passenger?**

   Mysql> select min(age) from passenger;

   (or)

   Mysql> select min(age) as min_age from passenger;

   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 Emp_View As Select * from Emp_master where job='clerk';

View created.

Syntax: Select column_name,column_name 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. List out the criteria for updatable views.
9. What is the syntax for renaming the columns of a view

WEEK 8

AIM: Triggers(Creation of Insert Trigger, Delete Trigger and Update Trigger)

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.

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.

Row-Level Trigger: It is a trigger, which is activated for each row by a triggering statement such as insert, update, or delete. For example, if a table has inserted, updated, or deleted multiple rows, the row trigger is fired automatically for each row affected by the insert, update, or delete statement.

Statement-Level Trigger: It is a trigger, which is fired once for each event that occurs on a table regardless of how many rows are inserted, updated, or deleted.

Types of Triggers in MySQL?

We can define the maximum six types of actions or events in the form of triggers:

1. **Before Insert:** It is activated before the insertion of data into the table.
2. **After Insert:** It is activated after the insertion of data into the table.
3. **Before Update:** It is activated before the update of data in the table.
4. **After Update:** It is activated after the update of the data in the table.
5. **Before Delete:** It is activated before the data is removed from the table.
6. **After Delete:** It is activated after the deletion of data from the table.

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 triggersofadifferent sort.

To View list of triggers;

Show triggers;

To remove a trigger for Database

drop trigger trigger_name;

ex: drop trigger

ins_sal;

.

When defining a trigger, specify the trigger timing. That is, specify whether the trigger action is to 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_eventON table_name
FOR EACH
ROW
BEGIN
Executable
Statements;
END;

- **Table_name** is the name of the table. Actually, a trigger is always associated with a specific table. Without a table, a trigger would not exist hence we have to specify the table name after the 'ON' keyword.

- **Trigger_time** is the time of trigger activation and it can be BEFORE or AFTER. We must have to specify the activation time while defining a trigger. We must use BEFORE if we want to process action prior to the change made on the table and AFTER if we want to process action post to the change made on the table.
- **Trigger_event** can be INSERT, UPDATE, or DELETE. This event causes the trigger to be invoked. A trigger only can be invoked by one event. To define a trigger that is invoked by multiple events, we have to define multiple triggers, one for each event.
- **BEGIN…END** is the block in which we will define the logic for the trigger.

Trigger Syntax and Examples

Here is a simple example that associates a trigger with a table, to activate for `INSERT` operations. The trigger acts as an accumulator, summing the values inserted into one of the columns of the table.

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.01 sec)
```

The `CREATE TRIGGER` statement creates a trigger named `ins_sum` that is associated with the `account` table. It also includes clauses that specify the trigger action time, the triggering event, and what to do when the trigger activates:

- The keyword `BEFORE` indicates the trigger action time. In this case, the trigger activates before each row inserted into the table. The other permitted keyword here is `AFTER`.
- The keyword `INSERT` indicates the trigger event; that is, the type of operation that activates the trigger. **In the example, `INSERT` operations cause trigger activation**. You can also create triggers for `DELETE` and `UPDATE` operations.

- The statement following `FOR EACH ROW` defines the trigger body; that is, the statement to execute each time the trigger activates, which occurs once for each row affected by the triggering event. In the example, the trigger body is a simple `SET` that accumulates into a user variable the values inserted into the `amount` column. The statement refers to the column as `NEW.amount` which means "the value of the `amount` column to be inserted into the new row."

  To use the trigger, set the accumulator variable to zero, execute an `INSERT` statement, and then see what value the variable has afterward:

  ```
  mysql> SET @sum = 0;
  mysql> INSERT INTO account VALUES(1,14.98),(2,1937.50),(3,-100.00);
  // cause trigger activation
  mysql> SELECT @sum AS 'Total amount inserted';
  +----------------------+
  | Total amount inserted |
  +----------------------+
  |              1852.48 |
  +----------------------+
  ```

  In this case, the value of `@sum` after the `INSERT` statement has executed is `14.98 + 1937.50 - 100`, or `1852.48`.

  To destroy the trigger, use a `DROP TRIGGER` statement. You must specify the schema name if the trigger is not in the default schema:

  ```
  mysql> DROP TRIGGER ins_sum;
  ```

Update Trigger:

`UPDATE` trigger that checks the new value to be used for updating each row, and modifies the value to be within the range from 0 to 100. This must be a `BEFORE` trigger because the value must be checked before it is used to update the row:

```
mysql> delimiter //
mysql> CREATE TRIGGER upd_check BEFORE UPDATE ON account
    FOR EACH ROW
    BEGIN
      IF NEW.amount < 0 THEN
        SET NEW.amount = 0;
      ELSEIF NEW.amount > 100 THEN
        SET NEW.amount = 100;
      END IF;
    END;//
mysql> delimiter ;
```

// for trigger activation or firing a trigger update the table as below.

Mysql>update account set amount =500 where acct_num=2; (Here amount is greater than 100)

Output:

Acct_num	amount
2	0

Note: As per the code written inside the update trigger,when the amount exceeds more than 500, the corresponding amount becomes zero.

Triggers On Multiple Tables :

EX-1 :

```
mysql>create table employees(empid int primary key,
first_name varchar(30),
last_name varchar(30),
hire_date date);
```

```
mysql>create table employees_audit(id int auto_increment primary key,
empno int,
lastname varchar(30),
updated_name varchar(30),
changedate datetime,
action varchar(30));

mysql> insert into employees values(1,'Ajay','Varma','2000-6-15');
mysql> insert into employees values(2,'Praveen','Kumar','2000-7-10');



mysql> select * from employees;
mysql>create trigger be_up  before update on employees for each row
    insert into employees_audit
   set action='Update',
   empno=old.empid,
  lastname=old.last_name,
  updated_name=new.last_name,
  changedate=now( );
```

Testing the above trigger **be_up : (Firing a Trigger)**

```
mysql> update employees set last_name='Sharma' where empid=2;
mysql> select * from employees;
mysql> select * from employees_audit;
```

id	empno	lastname	updated_name	changedate	action
1	2	Kumar	Sharma	current date&time	Update

EX-2 :

```
mysql> create table members(id int,
```

```
    name  varchar(40),
     email  varchar(40),
    birthdate date);

mysql> create table reminders(id int auto_increment,
        memberid  int,
       message varchar(40),
       primary key(id));


mysql> delimiter //
       create trigger af_in after insert on members for each row
       begin
      if new.birthdate is null then
insert into reminders(memberid,message) values(new.id,concat('Hi ',new.name, ' Pl.
Update your  date of birth.'));
 end if;
 end //
mysql> Query OK
mysql> delimiter ;
```

Testing above trigger **af_in : (Firing a Trigger)**

```
Insert into members values(1 ,'Kalyan', 'kalyan@gmail' ,null),
                          (2, 'Arjun', 'arjun@gmail', '2000-4-15');


mysql> Query OK
mysql>select * from members;
```

mysql>Select * from reminders;

Id	memberid	message
1	1	Hi Kalyan Pl. Update your date of birth.

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.

WEEK 9

Aim: Procedures(Using IN, OUT and INOUT Parameters)

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 MySQL statements.

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.

Syntax:

```
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:

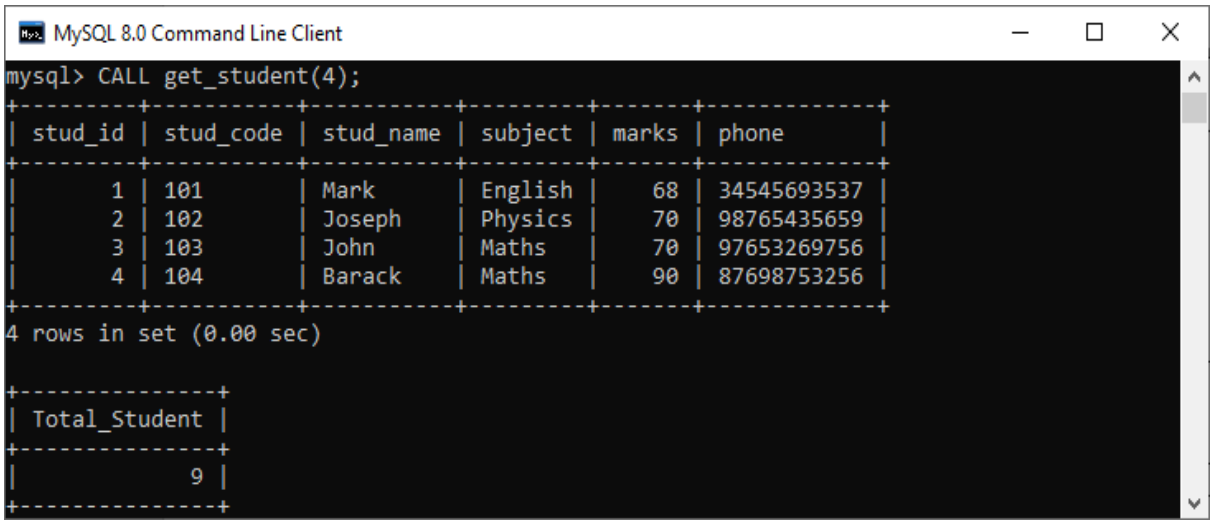
```
mysql> CALL get_merit_student();
```

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);

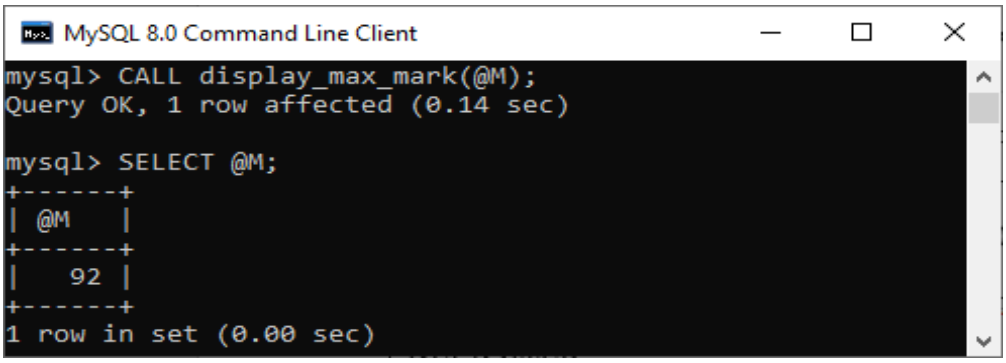


Procedures with OUT Parameter

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;

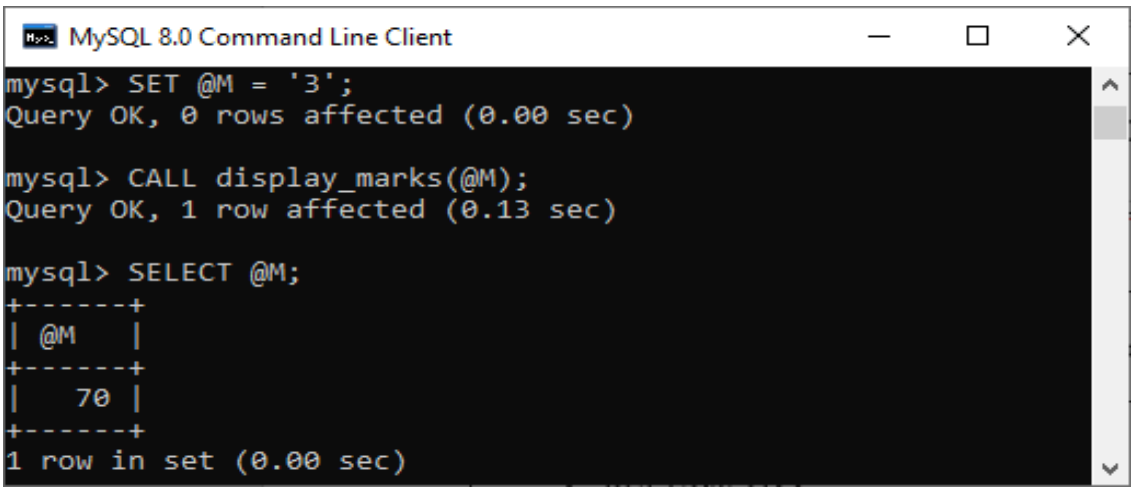


Procedures with INOUT Parameter:

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;
```



Difference between Triggers and Procedures:

BASIS FOR COMPARISON	TRIGGERS	PROCEDURES
Running	It can execute automatically based on the events.	It can be invoked explicitly by the user.
Parameter	We can not pass parameters to triggers.	We can pass parameters to procedures.
Return	Trigger never return value on execution.	Procedure may return value/s on execution.

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?

WEEK-10

Aim: Usage of Cursors.

Introduction to MySQL cursor

To handle a result set inside a stored procedure, you use a cursor. A cursor allows you to iterate a set of rows returned by a query and process each row individually.

Working with MySQL cursor

1. Declare Cursor

A cursor is a select statement, defined in the declaration section in MySQL.

Syntax

DECLARE cursor_name CURSOR FOR
Select statement;

2. Open Cursor

After declaring the cursor the next step is to open the cursor using open statement.

Syntax

Open cursor_name;

3. Fetch Cursor

After declaring and opening the cursor, the next step is to fetch the cursor. It is used to fetch the row or the column.

Syntax

FETCH [NEXT [FROM]] cursor_name INTO variable_list;

4. Close Cursor

The final step is to close the cursor.

Syntax

Close cursor_name;

It is a good practice to always close a cursor when it is no longer used.

When working with MySQL cursor, you must also declare a NOT FOUND handler to handle the situation when the cursor could not find any row.

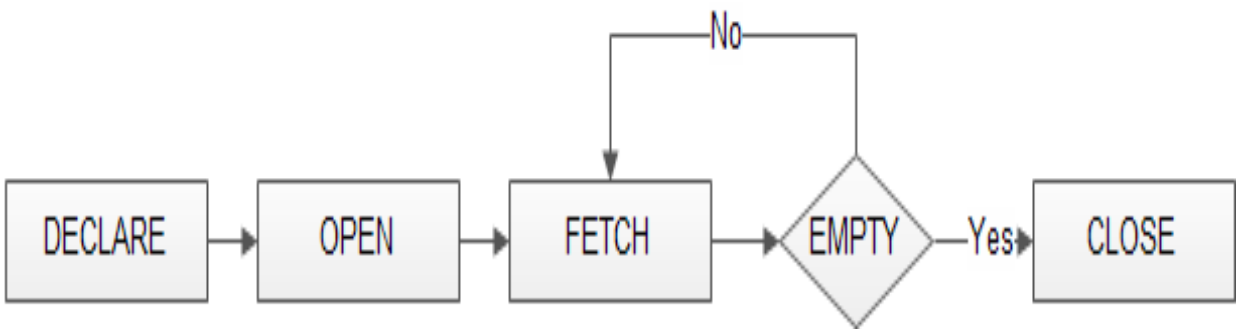
Because each time you call the FETCH statement, the cursor attempts to read the next row in the result set. When the cursor reaches the end of the result set, it will not be able to get the data, and a condition is raised. The handler is used to handle this condition.

To declare a NOT FOUND handler, you use the following syntax:

Declare continur handler for not found set finished=1;

The finished is a variable to indicate that the cursor has reached the end of the result set. Notice that the handler declaration must appear after variable and cursor declaration inside the stored procedures.

The following diagram illustrates how MySQL cursor works.



Example for the cursor:

Step 1: Open the database and table.

```
MySQL 8.0 Command Line Client
mysql> use test1;
Database changed
mysql> select *from table1;
+------+-----------+-------+
| id   | name      | class |
+------+-----------+-------+
|    1 | Shristee  | MCA   |
|    2 | Ajay      | BCA   |
|    3 | Shweta    | MCA   |
|    4 | Dolly     | BCA   |
|    5 | Heena     | MCA   |
|    6 | Kiran     | BCA   |
|    7 | Sonal     | MCA   |
|    8 | Dimple    | BCA   |
|    9 | Shyam     | MCA   |
|   10 | Mohit     | BCA   |
+------+-----------+-------+
10 rows in set (1.24 sec)
```

Step 2: Now create the cursor.

```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE list_name (INOUT name_list varchar(4000))
    -> BEGIN
    -> DECLARE is_done INTEGER DEFAULT 0;
    -> DECLARE s_name varchar(100) DEFAULT "";
    -> DECLARE stud_cursor CURSOR FOR
    -> SELECT name FROM table1;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET is_done = 1;
    -> OPEN stud_cursor;
    -> get_list: LOOP
    -> FETCH stud_cursor INTO s_name;
    -> IF is_done = 1 THEN
    -> LEAVE get_list;
    -> END IF;
    -> SET name_list = CONCAT(s_name, ";",name_list);
    -> END LOOP get_list;
    -> CLOSE stud_cursor;
    -> END$$
Query OK, 0 rows affected (0.24 sec)
```

Step 3: Now call the cursor.

Query:

```
mysql> SET @name_list ="";
Query OK, 0 rows affected (0.00 sec)

mysql> CALL list_name(@name_list);
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT @name_list;
+-------------------------------------------------------------------+
| @name_list                                                        |
+-------------------------------------------------------------------+
| Mohit;Shyam;Dimple;Sonal;Kiran;Heena;Dolly;Shweta;Ajay;Shristee; |
+-------------------------------------------------------------------+
1 row in set (0.00 sec)
```

1) **Implicit cursors:** are automatically created when select statements are executed.
2) **Explicit cursors:** needs to be defined explicitly by the user by providing a name. They are capable of fetching a single row at a time. Explicit cursors can fetch multiple rows

Viva Questions:

What is a cursor?

What are the steps involves in using cursor?

What is Read-Only cursor?

What is difference between implicit cursor and explicit curcor?

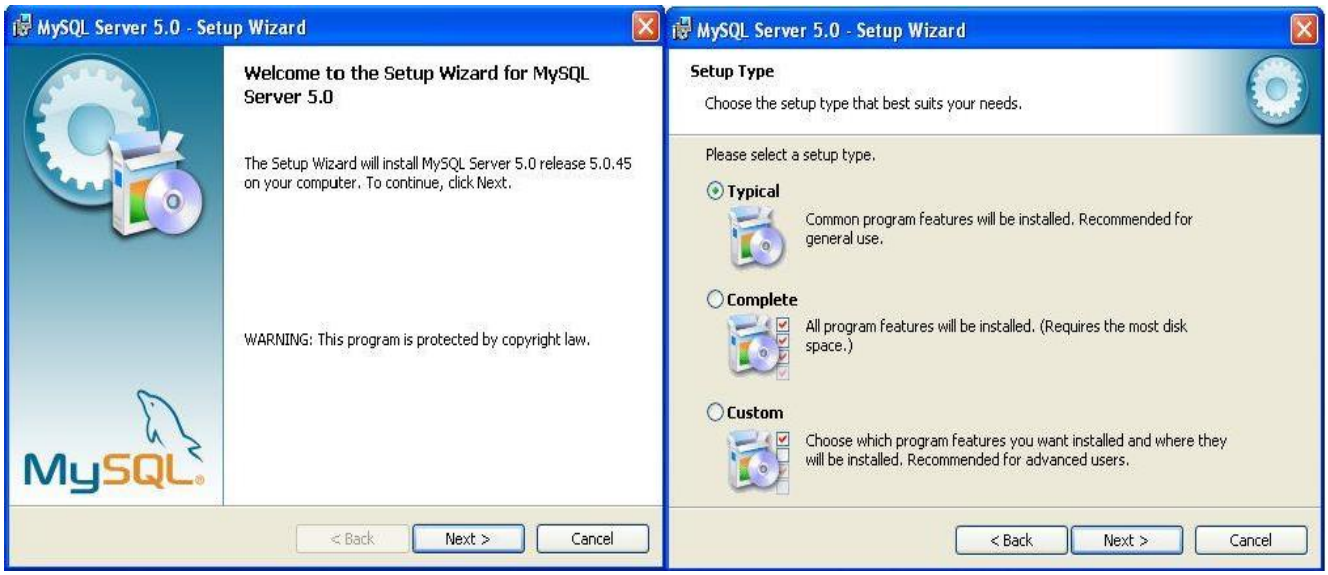
WEEK-11

AIM: Installation of MySQL and practicing DDL commands.

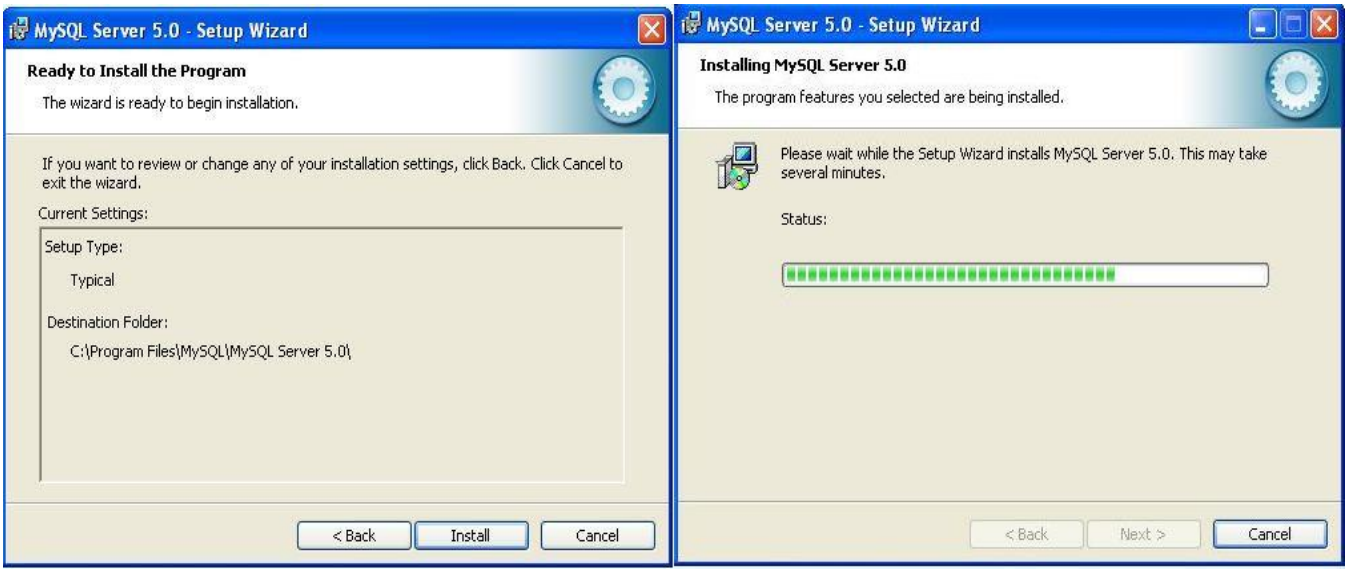
1. Steps for installing MySQL

Step: 1 download mysql essential from the website www.mysql.com/downloads and save the .exe file.

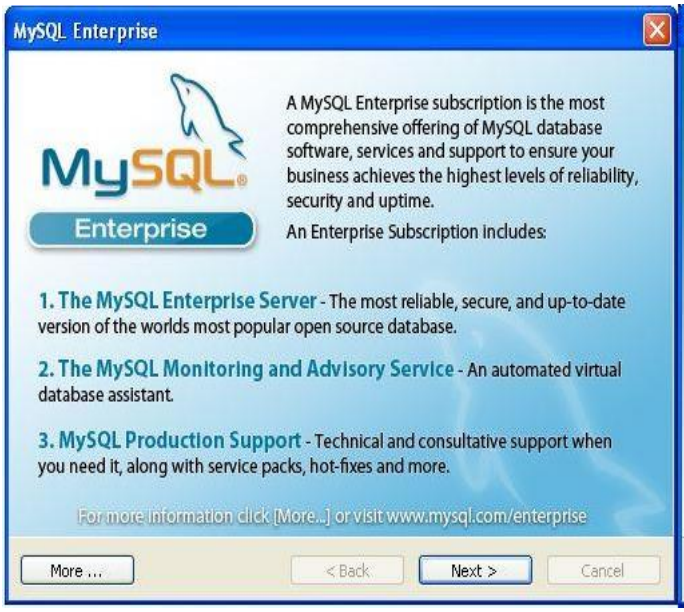
Steps: 2&3double click on the mysql.exe file to start installation.

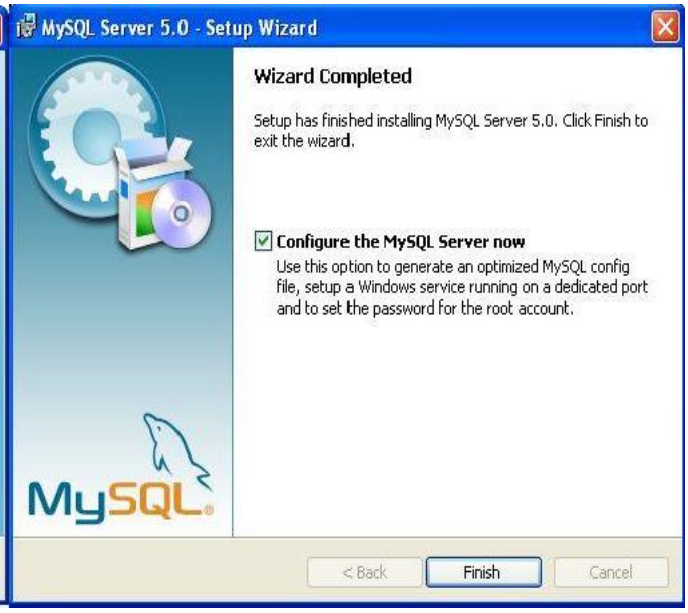


Steps: 4&5

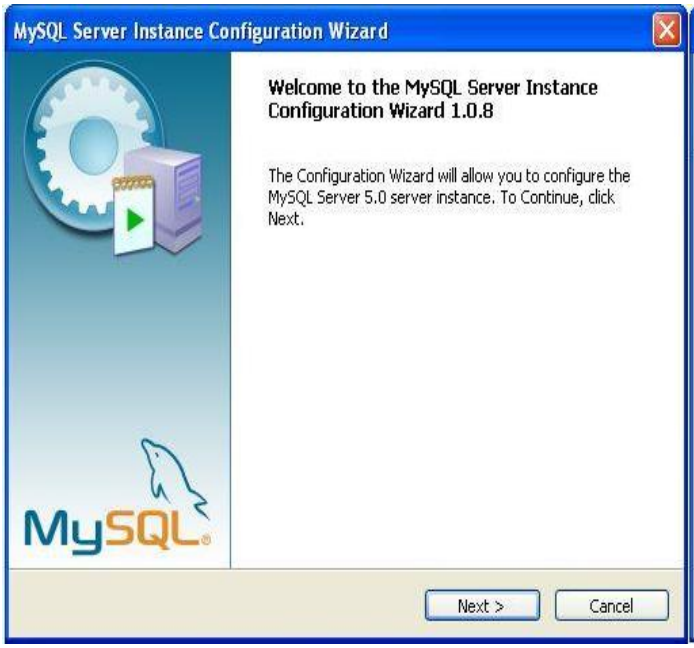


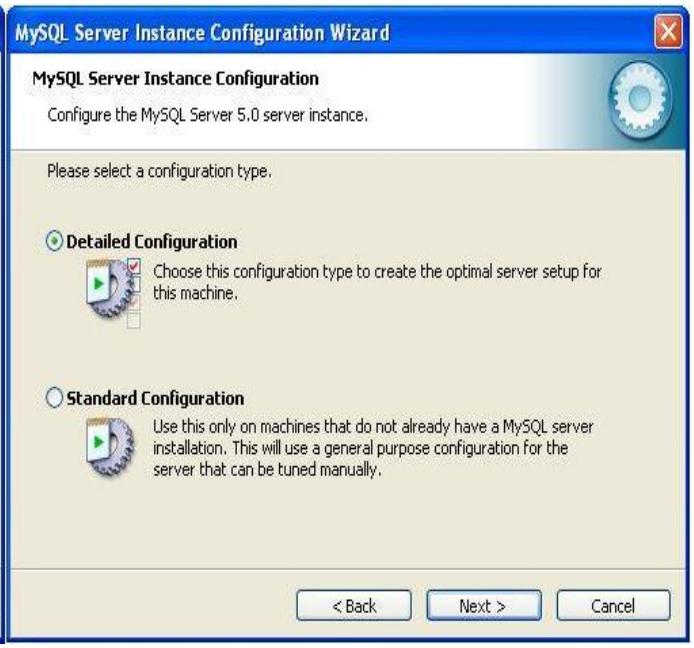
Steps: 6&7



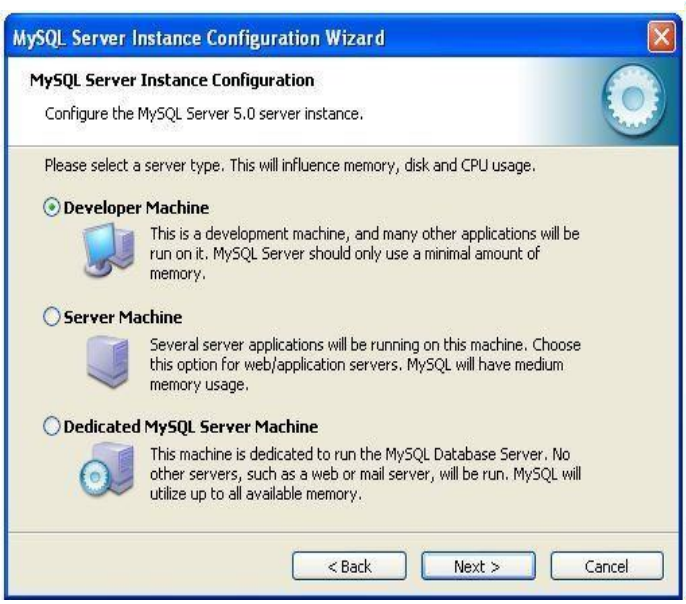


Steps: 8&9

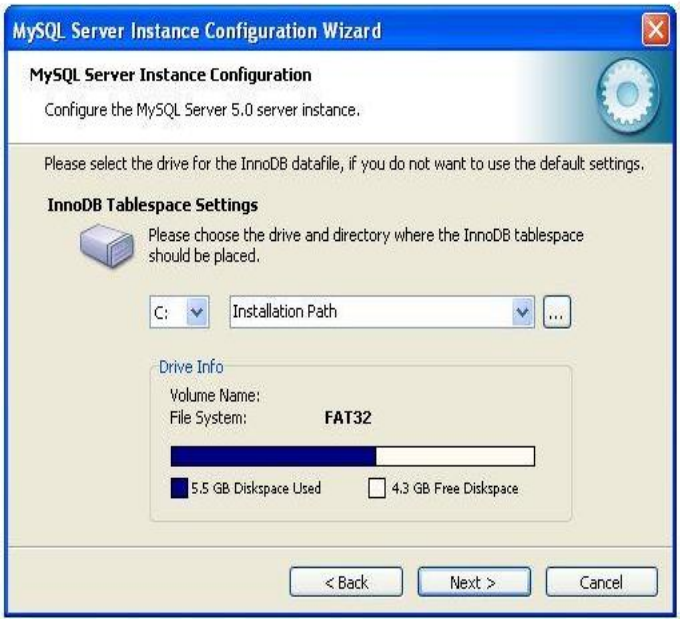


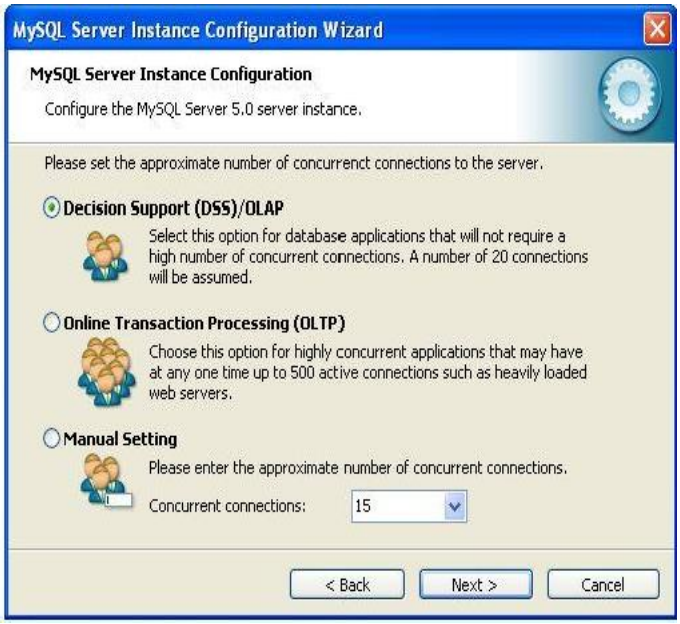


Steps: 10&11

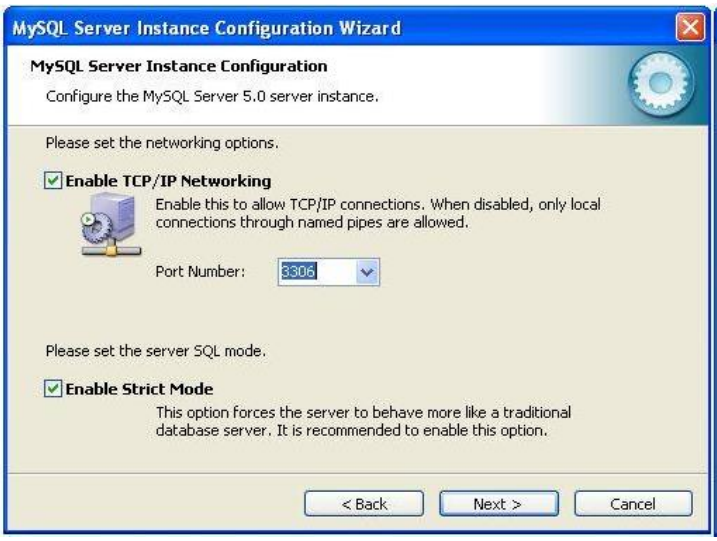


Step: 12&13

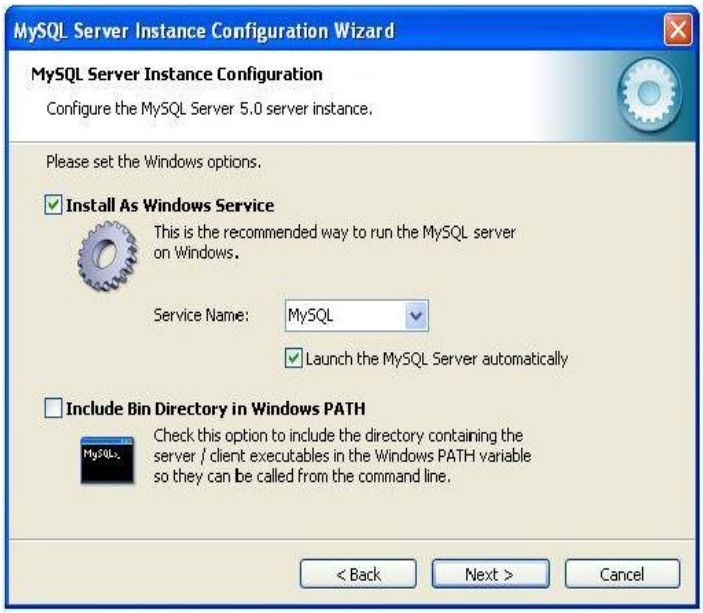




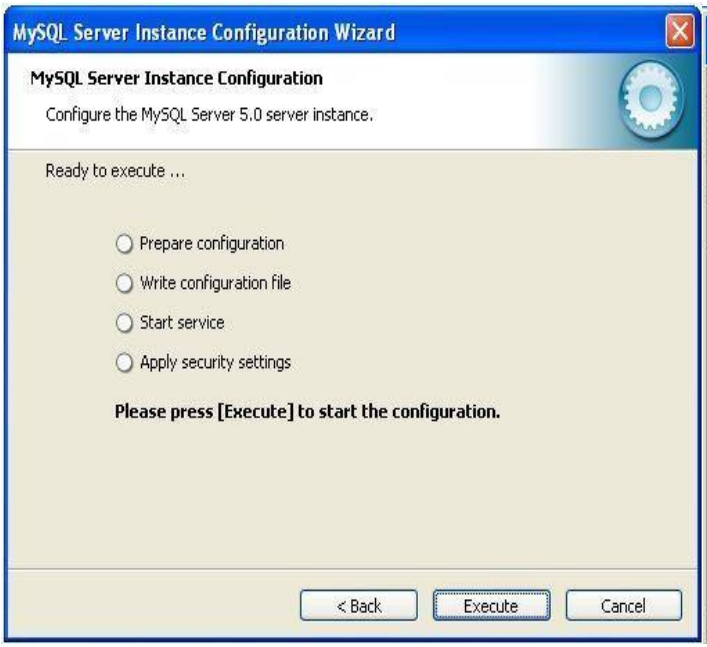
Steps: 14&15

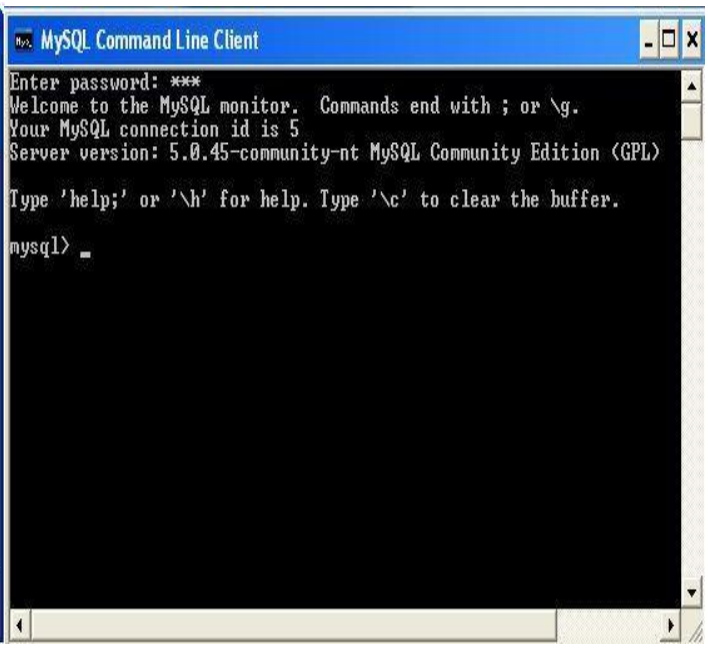


Steps: 16&17



Steps: 18&19


MySQL Server Instance Configuration Wizard
MySQL Server Instance Configuration
Configure the MySQL Server 5.0 server instance.
Ready to execute ...
Prepare configuration
Write configuration file
Start service
Apply security settings
Please press [Execute] to start the configuration.
< Back
Execute
Cancel


MySQL Command Line Client
Enter password: ***
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.0.45-community-nt MySQL Community Edition (GPL)
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>

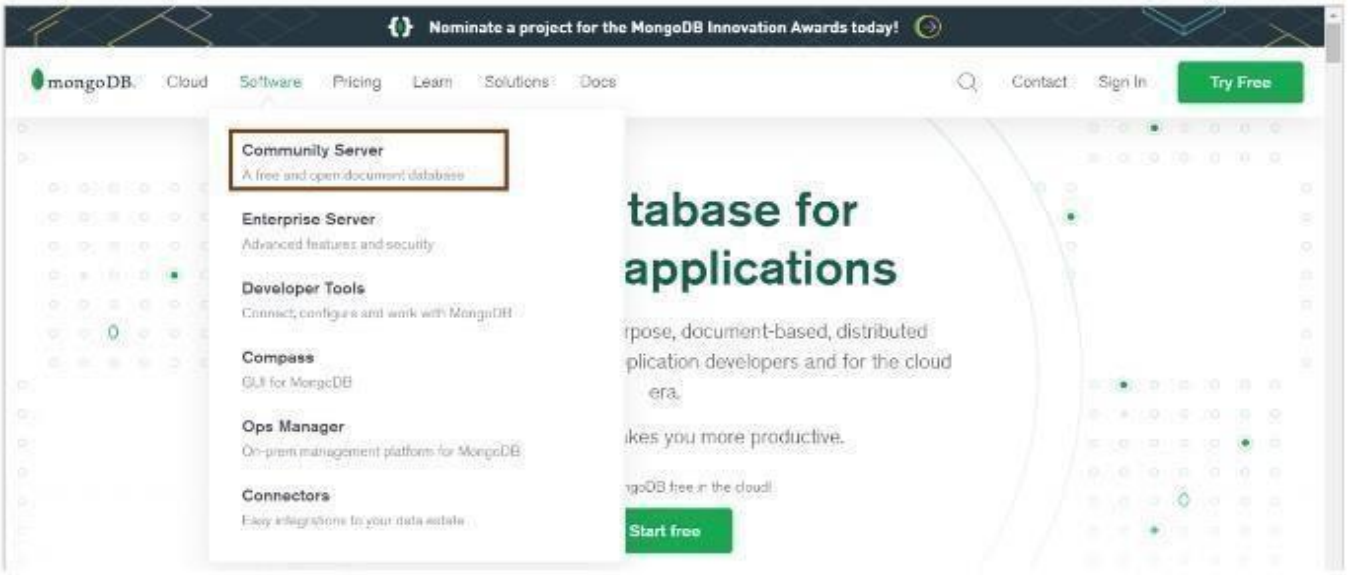
5(b) :Installation of MangoDB

Navigate to the download site:

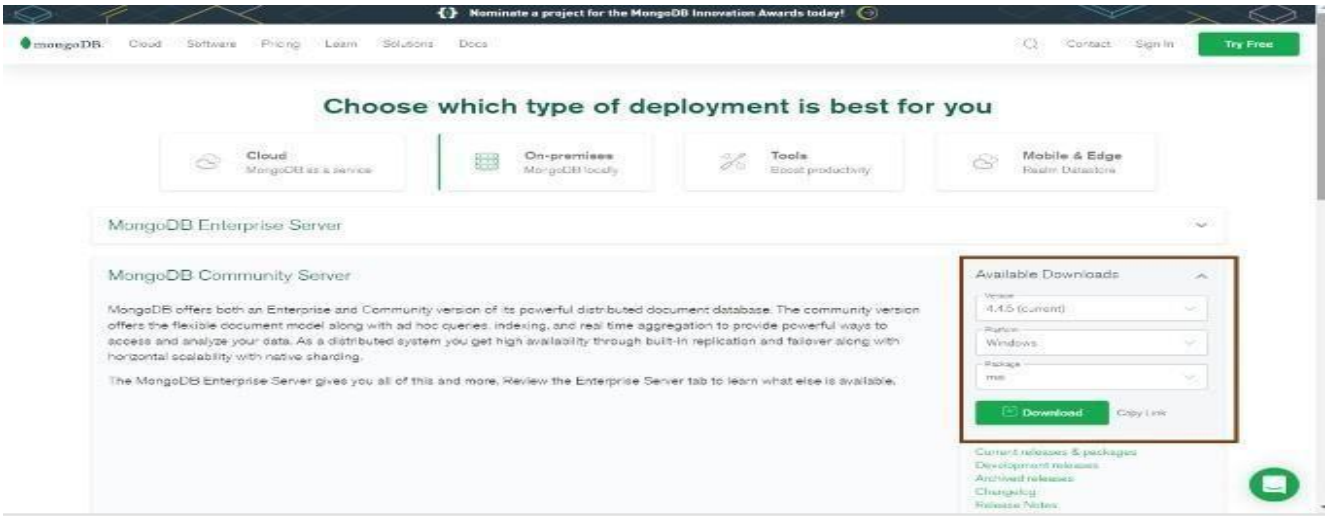
Navigate to the official MongoDB website https://www.mongodb.com/

Cross-check the Specifications and Download MongoDB

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

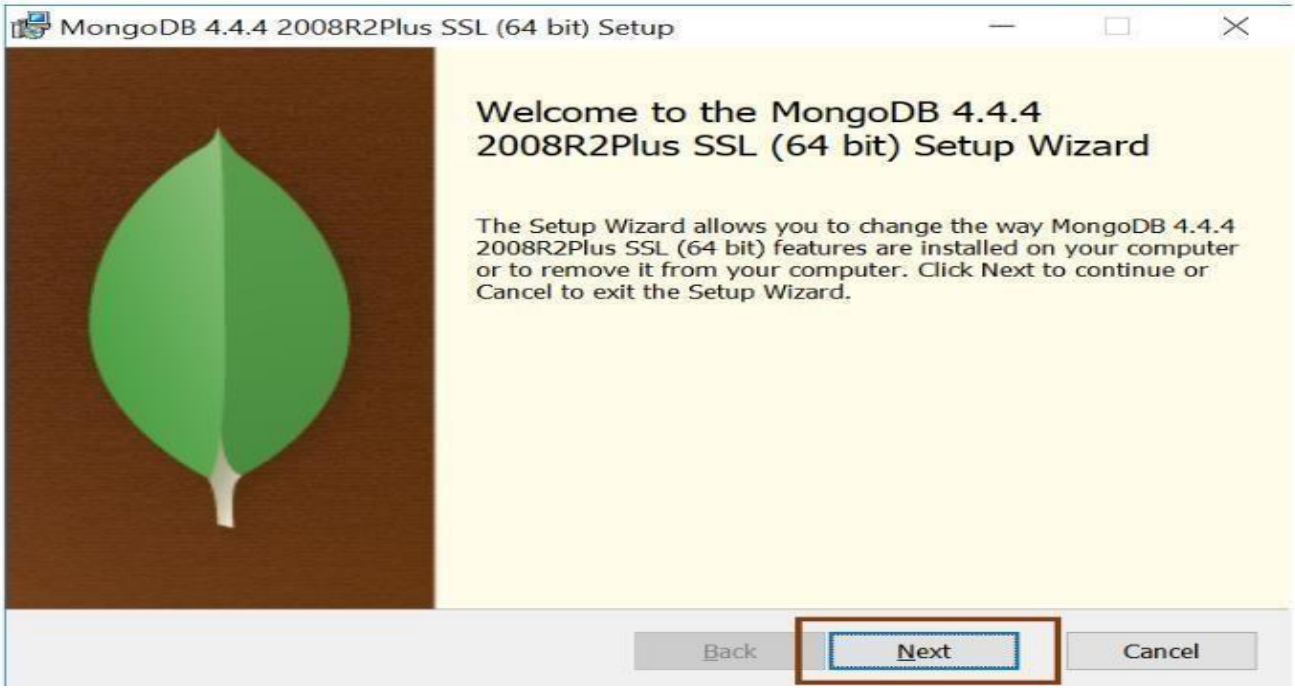


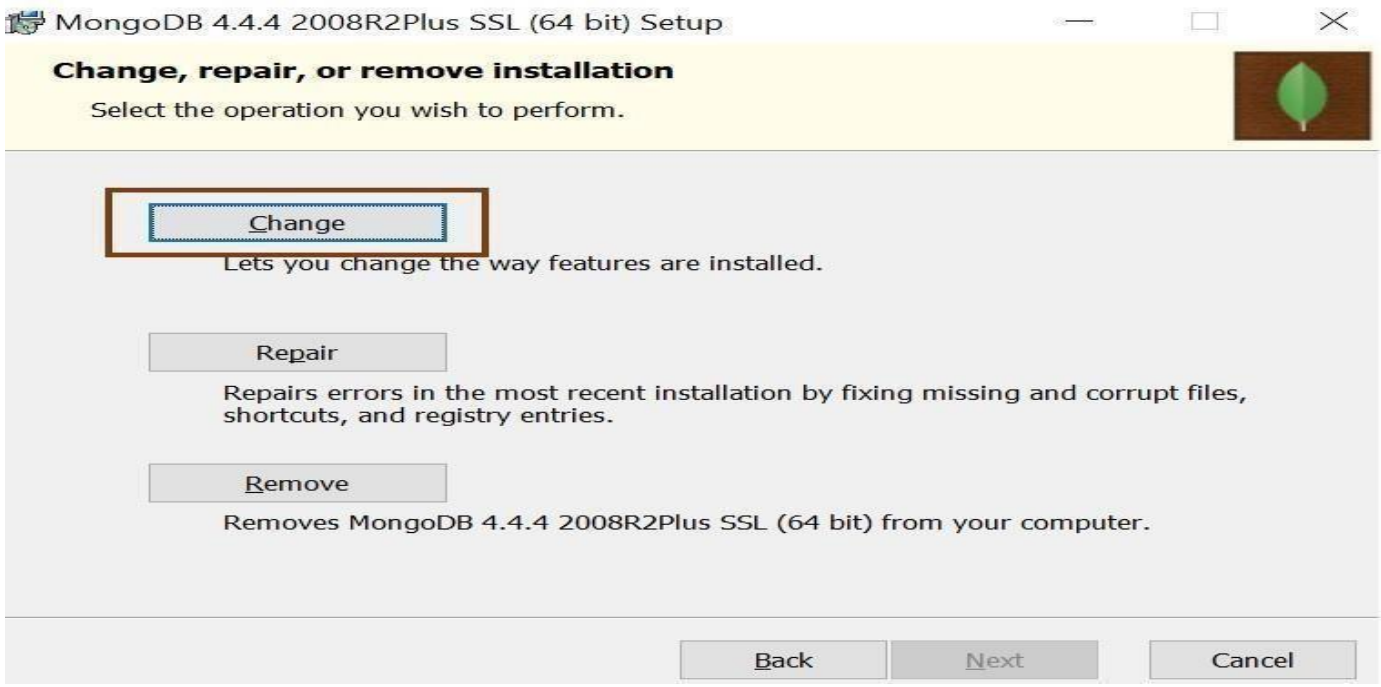
At the time of writing, the latest version is 4.4.5. Ensure that the platform is Windows, and the packageis MSI. Go ahead and click on download.



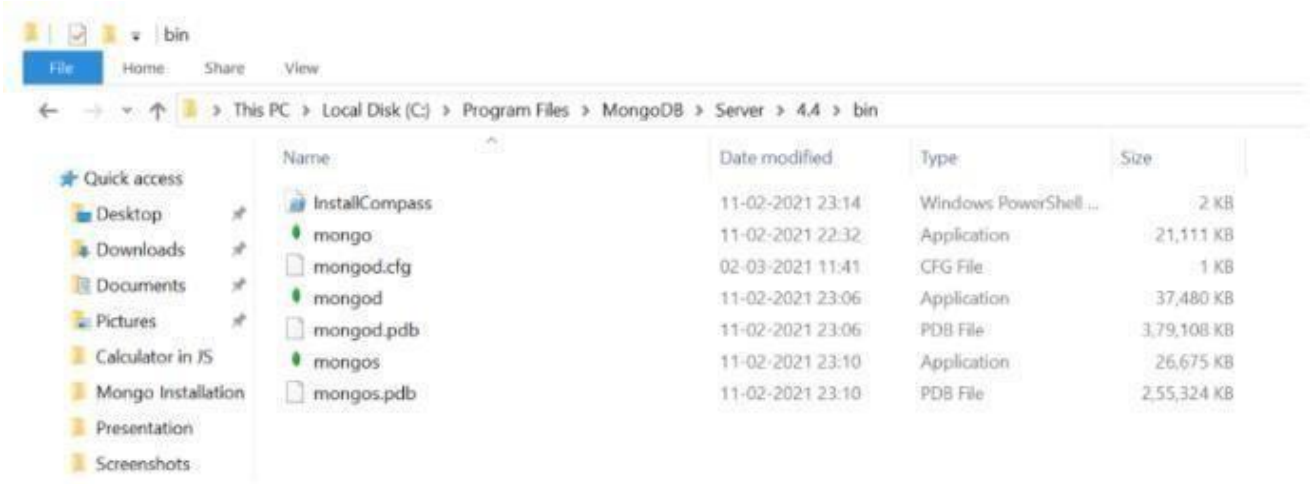
Mongo DB Installation:

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





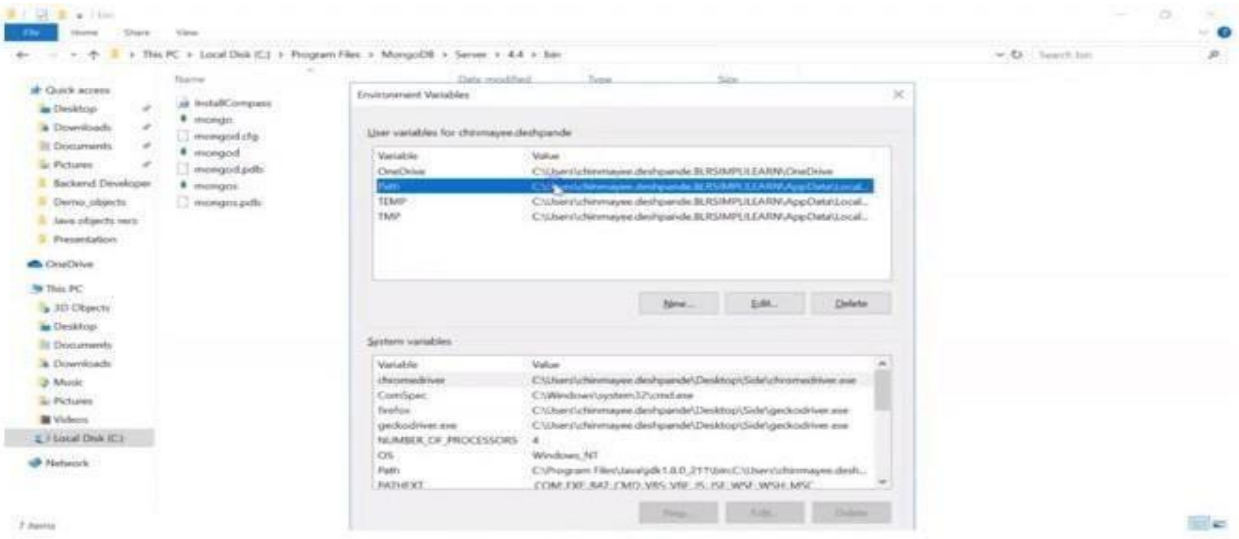
On completing the installation successfully, you will find the software package in your Cdrive. C:\Program Files\MongoDB\Server\4.4\bin.

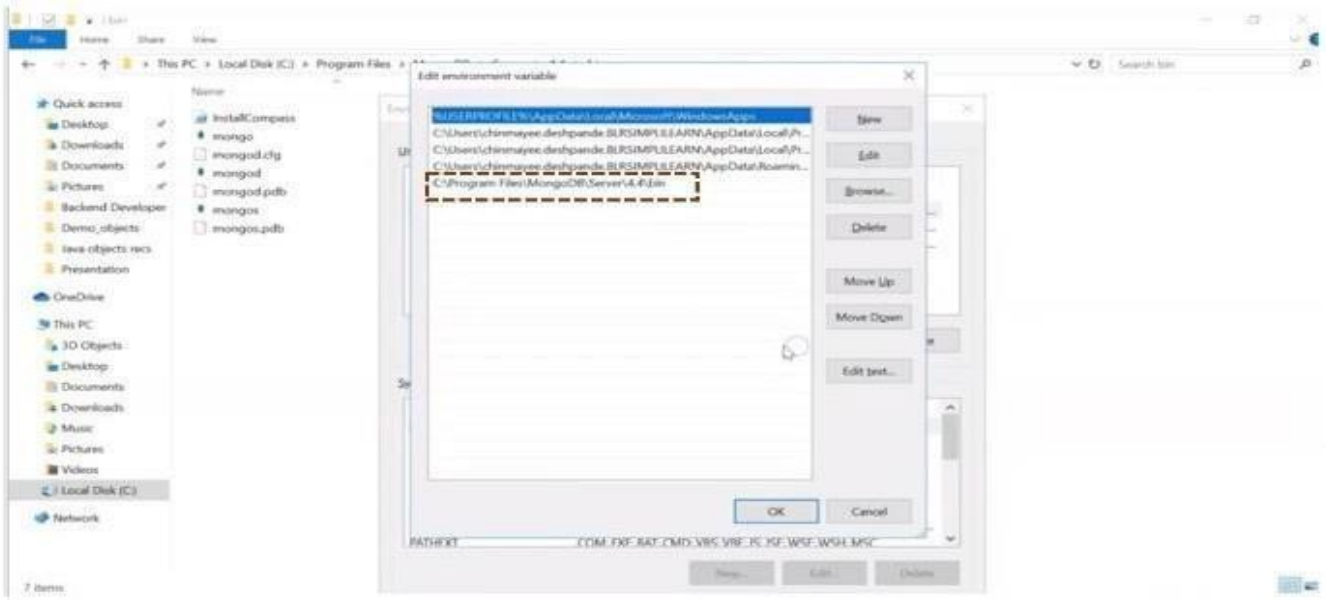


You can see that there are mongo and mongod executable files. The mongod file is the daemonprocess that does the background jobs like accessing, retrieving, and updating the database.

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:

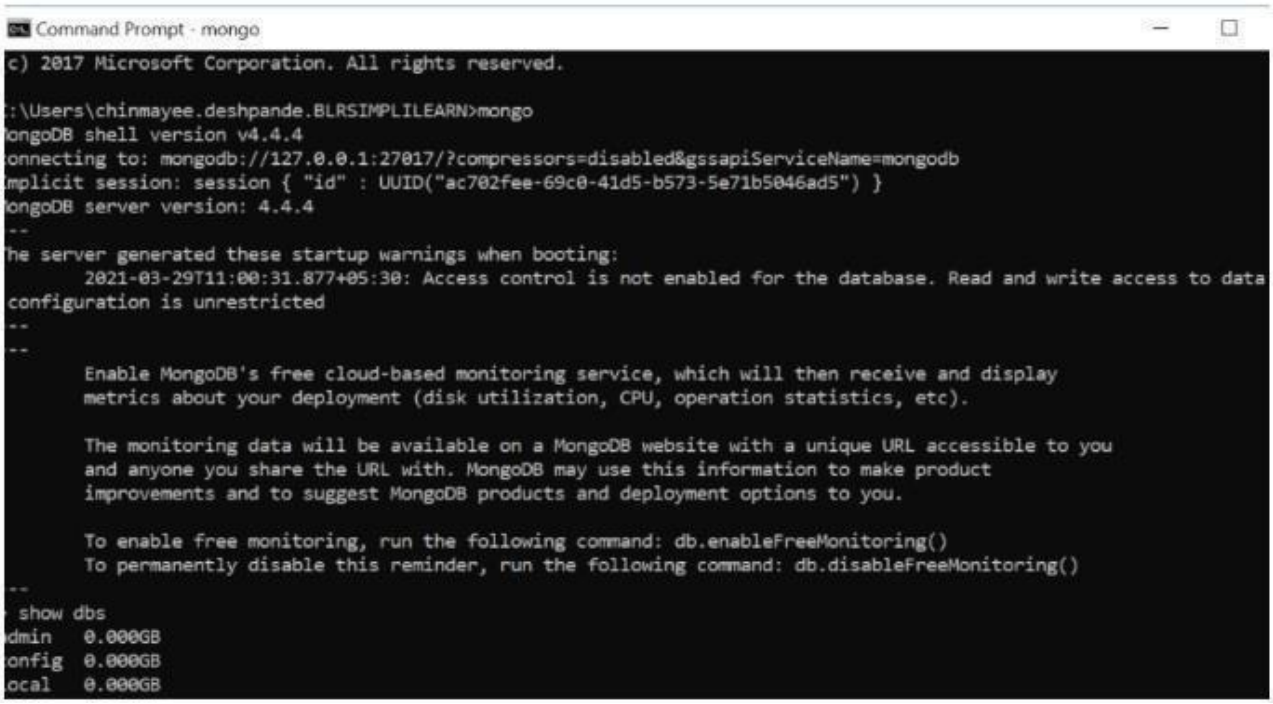
After creating an environment path, you can open the command prompt and justtype in mongo and press enter.

```
Command Prompt - mongo
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\chinmayee.deshpande.BLRSIMPLILEARN>mongo
MongoDB shell version v4.4.4
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("ac702fee-69c0-41d5-b573-5e71b5046ad5") }
MongoDB server version: 4.4.4
---
The server generated these startup warnings when booting:
        2021-03-29T11:00:31.877+05:30: Access control is not enabled for the database. Read and write access to data
 configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

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

Verify the Setup

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



With that, you have successfully installed and set up MongoDB on your Windows system.

Creating Tables and altering the Tables

Mysql>Create table passenger2(passportId Integer Primary Key,Name varchar(10) Not Null,Age Integer Not Null,Sex char,Address varchar(20) Not Null);

Mysql> desc passenger2;

```
mysql> create table passenger3(passportId integer primary key,name varchar(10) not null,Age Integer not null,
Sex char,Address varchar(20) not null);
Query OK, 0 rows affected (0.03 sec)

mysql> desc passenger3;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

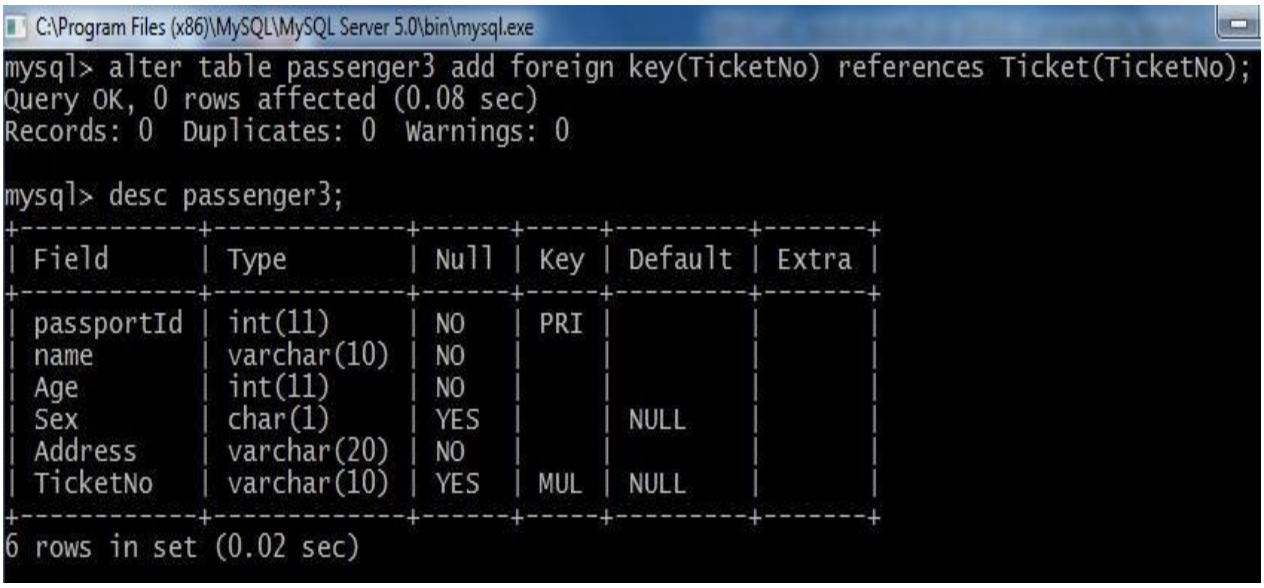
USING ALTER COMMAND

Adding Extra column to Existing Table

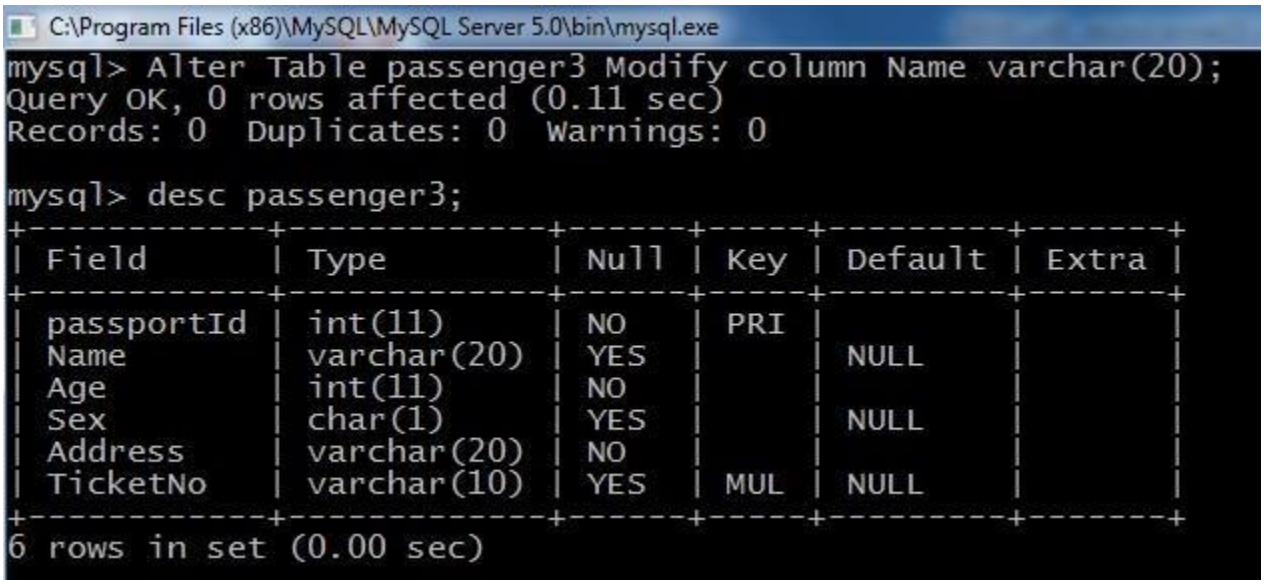
Mysql>Alter table passenger3 add column TicketNo varchar(10);

```
mysql> Alter table passenger3  add column TicketNo  varchar(10);
Query OK, 0 rows affected (0.14 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc passenger3;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
| TicketNo   | varchar(10) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Mysql>Alter Table passenger3 add Foreign key(TicketNo) references Ticket(TicketNo);



Mysql>Alter Table passenger3 Modify column Name varchar(20);



Mysql>Alter table passenger drop foreign key fk1;

```
mysql> Alter table passenger2  add column TicketNo  varchar(10);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table passenger2 add constraint fk1 foreign key(TicketNo) reference
s Ticket(TicketNo);
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> Alter table passenger2 drop foreign key fk1;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc passenger2;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
| TicketNo   | varchar(10) | YES  | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)
```

Mysql> Alter table passenger2 Drop column TicketNo;

```
mysql> Alter table passenger2 drop column ticketNo;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc passenger2;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| passportId | int(11)     | NO   | PRI |         |       |
| name       | varchar(10) | NO   |     |         |       |
| Age        | int(11)     | NO   |     |         |       |
| Sex        | char(1)     | YES  |     | NULL    |       |
| Address    | varchar(20) | NO   |     |         |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

Viva Questions:

1. What is DDL (Data Definition Language)?
2. What is VDL (View Definition Language)?
3. What is SDL (Storage Definition Language)?
4. What is DML (Data Manipulation Language)?
5. What is DML Compiler?
6. What is PL/SQL?

WEEK-12

Aim: Practicing DCL(Grant, Revoke) and TCL Commands(Commit, Rollback and Savepoint)

MySQL CREATE USER Example

The following are the step required to create a new user in the MySQL server database.

Step 1: Open the MySQL server by using the **mysql client tool**.

Step 2: Enter the password for the account and press Enter.

Enter **Password**: ********

Step 3: Execute the following command to show all users in the current MySQL server.

mysql> **select** user **from** mysql.user;

We will get the output as below:

```
+------------------+
| user             |
+------------------+
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| root             |
+------------------+
4 rows in set (0.00 sec)
```

Step 4: Create a new user with the following command.

1. mysql> **create** user peter@localhost identified **by** 'jtp12345';

Now, run the command to show all users again.

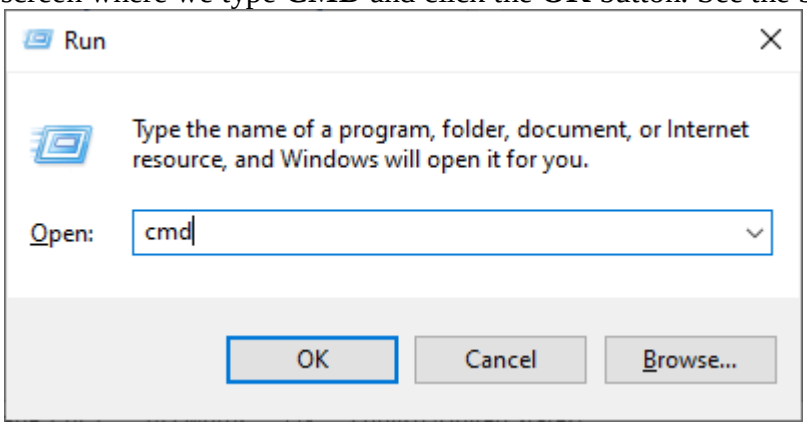
```
+------------------+
| user             |
+------------------+
| mysql.infoschema |
| mysql.session    |
| mysql.sys        |
| peter            |
| root             |
+------------------+
5 rows in set (0.00 sec)
```

Next, we need to **provide all privileges** to the newly created user account 'William' by executing the query as follows:

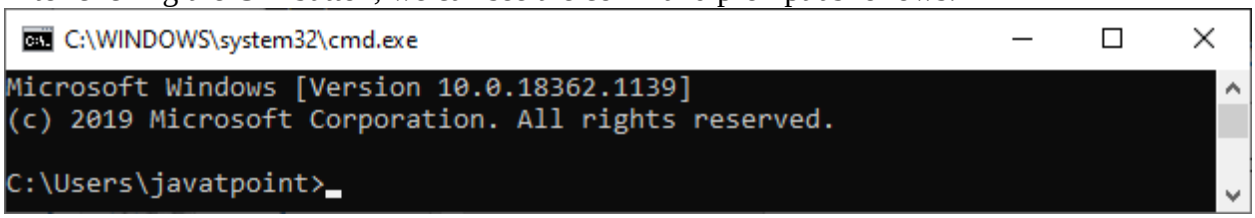
1. mysql> **GRANT** ALL **PRIVILEGES ON** * . * **TO** 'william'@'localhost';

   If this query is successful, we should get the below message:

   If we want to login to a different user account, we need to **open the command prompt** by executing the **RUN** command in the search box or pressing the **Windows + R** shortcut key. It will display the screen where we type **CMD** and click the **OK** button. See the below screenshot:



   After clicking the **OK** button, we can see the command prompt as follows:

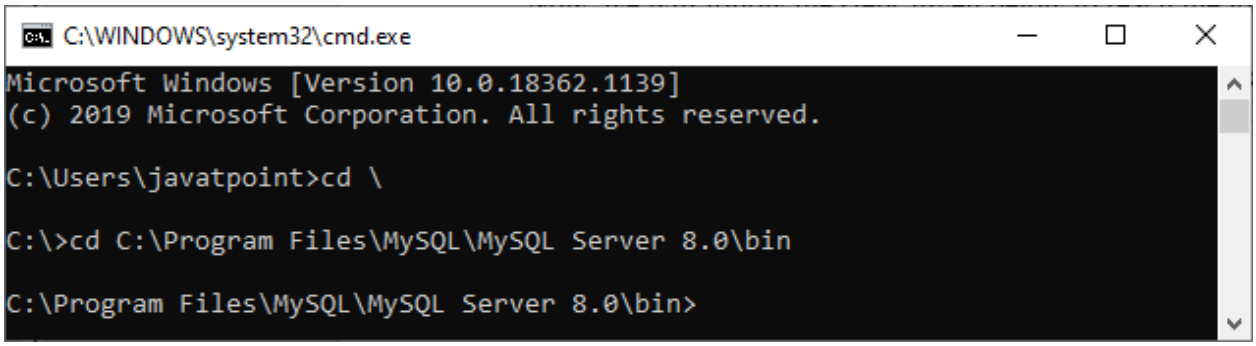


   Now, we will follow the steps given below to reach the location of the bin directory:

   Step 1: Suppose we have installed MySQL in the **C folder** on our device. Then, copy that folder and paste it into our command prompt and press the **Enter** key.

1. C:\Users\javatpoint>cd \
2. C:\>cd C:\Program Files\MySQL\MySQL Server 8.0\bin

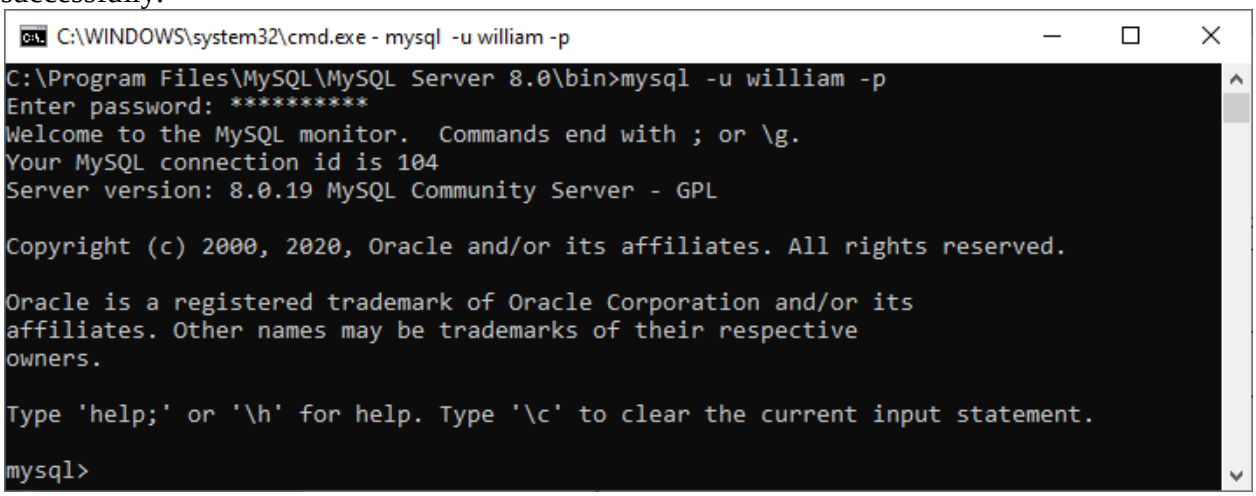
   After pressing the Enter key, we can see the below output:



Step 2: Next, we will use the syntax that we have discussed at the beginning (mysql -u username -p.). Write the 'William' as the user name & 'will123456' as the password created above and press the **Enter key** to access the account. See the below query:

mysql -u william -p
Enter **Password**: **********

After pressing the Enter key, we will get the below output that shows the username is opened successfully.



MySQL Grant Privilege

MySQL has a feature that provides many control options to the administrators and users on the database. We have already learned how to create a new user using CREATE USER statement in MySQL server. Now, we are going to learn about grant privileges to a user account. MySQL provides GRANT statements to give access rights to a user account.

GRANT Statement

The grant statement enables system administrators to ***assign privileges and roles*** to the MySQL user accounts so that they can use the assigned permission on the database whenever required.

Syntax

The following are the basic syntax of using the GRANT statement:

GRANT privilege_name(s)
ON object
TO user_account_name;

GRANT Statement Example

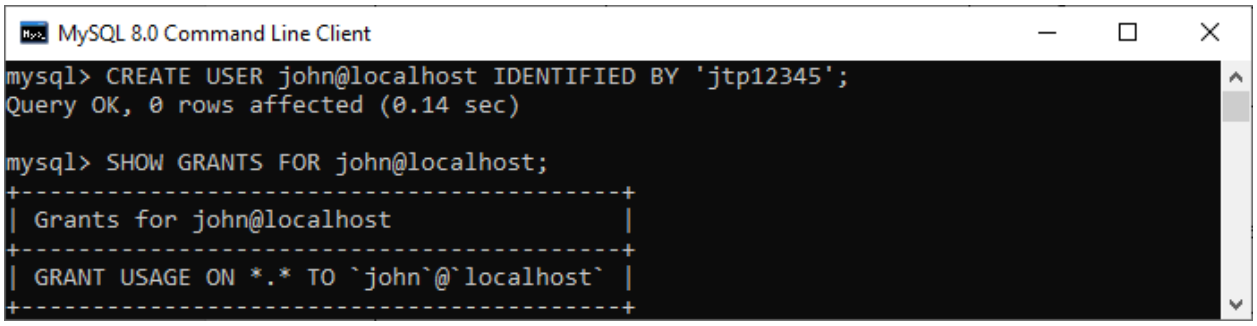
Let us understand the GRANT privileges through the example. First, we need to create a new user named "**john@localhost**" using the following statement:

mysql> **CREATE** USER john@localhost IDENTIFIED **BY** 'jtp12345';

Next, execute the SHOW GRANT statement to check the privileges assigned to john@localhost using the following query:

mysql> SHOW GRANTS **FOR** john@localhost;

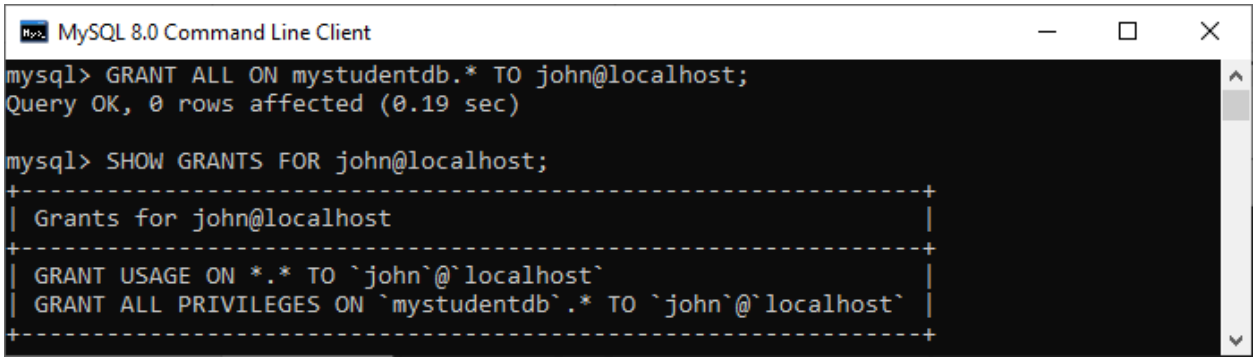
It will give the below output. Here, the **USAGE** means a user can log in to the database but does not have any privileges.



If we want to **assign all privileges** to all databases in the current server to john@localhost, execute the below statement:

1. mysql> **GRANT** ALL **ON** mystudentdb.* **TO** john@localhost;

Again, execute the SHOW GRANT statement to verify the privileges. After the successful execution, we will get the below output. Here all privileges are assigned to all databases in the current server to john@localhost.



REVOKE Statement

The revoke statement enables system administrators to ***revoke privileges and roles*** to the MySQL user accounts so that they cannot use the assigned permission on the database in the past.

Syntax

The following are the basic syntax of using the REVOKE statement:

REVOKE privilege_name(s)
ON object
FROM user_account_name

REVOKE Statement Example

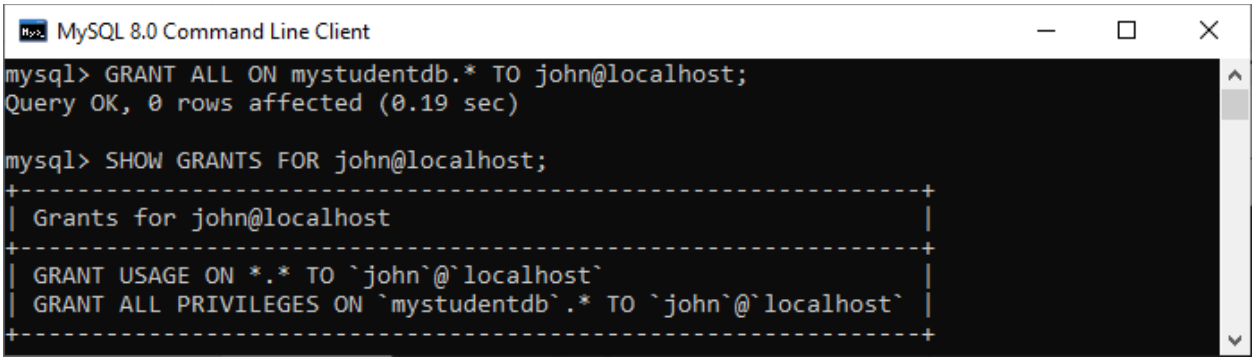
Let us understand the REVOKE privileges through the example. First, we need to create a new user named "**john@localhost**" using the following statement:

mysql> **CREATE** USER john@localhost IDENTIFIED **BY** 'jtp12345';

Next, assign all privileges to all databases in the current server to john@localhost, using the below statement:

mysql> **GRANT** ALL **ON** mystudentdb.* **TO** john@localhost;

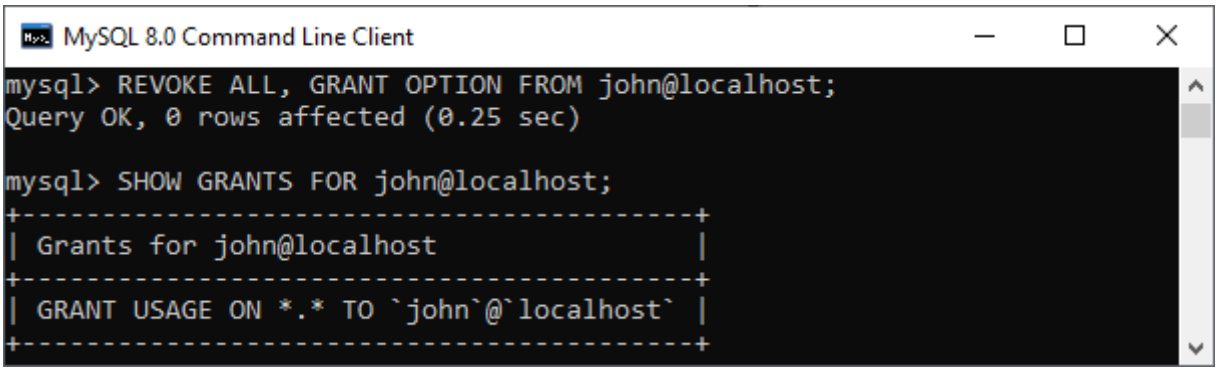
Next, execute the SHOW GRANT statement to verify the privileges. In the output, we can see that all privileges are assigned to all databases in the current server to john@localhost.



If we want to revoke all privileges assign to the user, execute the following statement:

mysql> **REVOKE** ALL, **GRANT OPTION FROM** john@localhost;

We will get the output below where we can see that a user can log in to the database without any privileges.



REVOKE selected privilege from a user account

Suppose we have provided grant privilege of SELECT, INSERT, and UPDATE command on mystudentdb to the user with the following statement:

mysql> **GRANT SELECT**, **UPDATE**, **INSERT ON** mystudentdb.* **TO** john@localhost;

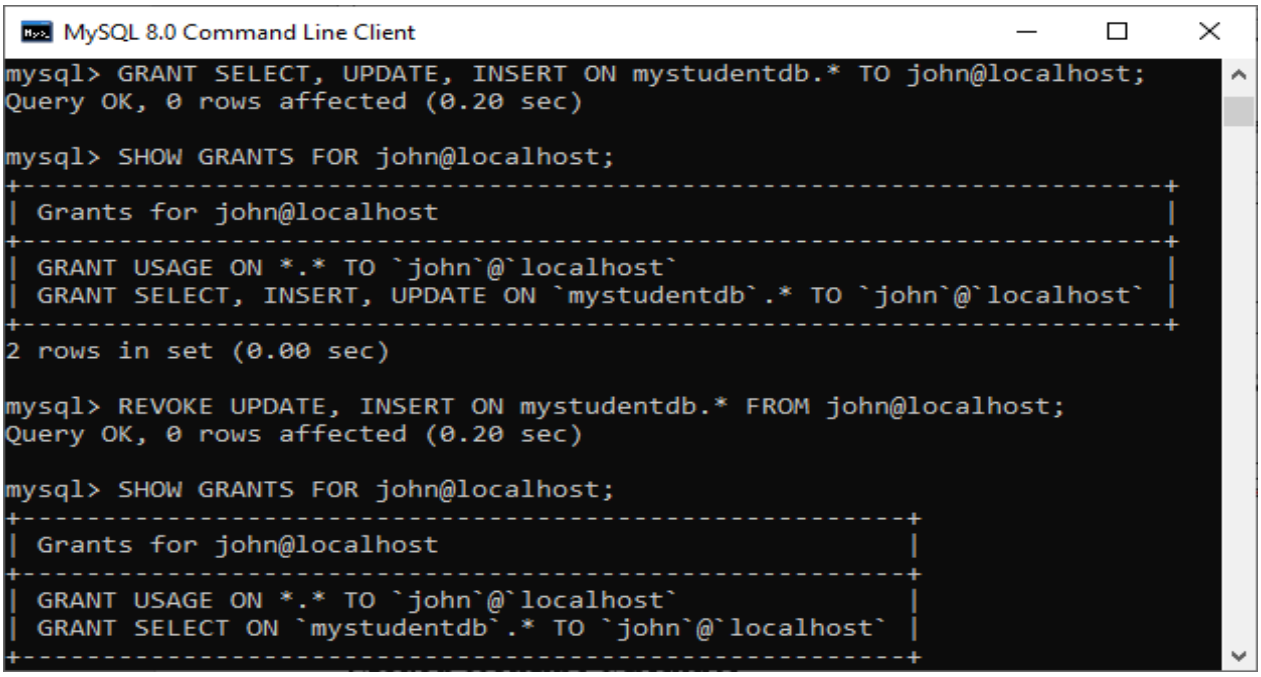
Next, display the GRANT privilege with the following statement:

mysql> SHOW GRANTS **FOR** john@localhost;

Finally, execute the REVOKE statement to remove UPDATE and INSERT privilege with the below statement:

mysql> **REVOKE UPDATE**, **INSERT ON** mystudentdb.* **FROM** john@localhost;

It will give the below output where only SELECT privilege is left.



If autocommit mode is disabled within a session with SET autocommit = 0, the session always has a transaction open. A COMMIT or ROLLBACK statement ends the current transaction and a new one starts.

To use multiple-statement transactions, switch autocommit off with the SQL statement SET autocommit = 0 and end each transaction with COMMIT or ROLLBACK as appropriate. To leave autocommit on, begin each transaction with START TRANSACTION and end it with COMMIT or ROLLBACK. The following example shows two transactions. The first is committed; the second is rolled back.

```
mysql> CREATE TABLE customer (a INT, b CHAR (20), INDEX (a));
Query OK, 0 rows affected (0.00 sec)
mysql> -- Do a transaction with autocommit turned on.
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO customer VALUES (10, 'Heikki');
Query OK, 1 row affected (0.00 sec)
mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
mysql> -- Do another transaction with autocommit turned off.
mysql> SET autocommit=0;
Query OK, 0 rows affected (0.00 sec)
mysql> INSERT INTO customer VALUES (15, 'John');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO customer VALUES (20, 'Paul');
Query OK, 1 row affected (0.00 sec)
mysql> DELETE FROM customer WHERE b = 'Heikki';
Query OK, 1 row affected (0.00 sec)
mysql> -- Now we undo those last 2 inserts and the delete.
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT * FROM customer;
+------+--------+
| a    | b      |
+------+--------+
|   10 | Heikki |
+------+--------+
1 row in set (0.00 sec)
mysql>
```

The SAVEPOINT statement sets a named transaction savepoint with a name of ***identifier***. If the current transaction has a savepoint with the same name, the old savepoint is deleted and a new one is set.

The ROLLBACK TO SAVEPOINT statement rolls back a transaction to the named savepoint without terminating the transaction. Modifications that the current transaction made to rows after the savepoint was set are undone in the rollback

Syntax

Following is the syntax of the MySQL SAVEPOINT statement −

```
SAVEPOINT identifier
```

Example

MySQL saves the changes done after the execution of each statement. To save changes automatically, set the autocommit option as shown below −

```
SET autocommit=0;
```

Assume we have created a table in MySQL with name EMPLOYEES as shown below −

```
mysql> CREATE TABLE EMP(
   FIRST_NAME CHAR(20) NOT NULL,
   LAST_NAME CHAR(20),
   AGE INT,
   SEX CHAR(1),
   INCOME FLOAT);
Query OK, 0 rows affected (0.36 sec)
```

Let us insert 4 records in to it using INSERT statements as −

```
mysql> INSERT INTO EMP VALUES
  ('Krishna', 'Sharma', 19, 'M', 2000),
   ('Raj', 'Kandukuri', 20, 'M', 7000),
   ('Mohan', 'Rao', 25, 'F', 5000);
Query OK, 3 rows affected (0.49 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

Following transaction updates, the age values of all the employees in the emp table −

```
mysql>START TRANSACTION;
mysql>SELECT * FROM EMP;
mysql>UPDATE EMP SET AGE = AGE + 1;
mysql>SAVEPOINT samplesavepoint;
mysql>INSERT INTO EMP ('Mac', 'Mohan', 26, 'M', 2000);
mysql>ROLLBACK TO SAVEPOINT samplesavepoint;
mysql>COMMIT;
```

f you retrieve the contents of the table, you can see the updated values as −

```
mysql> SELECT * FROM EMP;
+------------+-----------+------+------+--------+
| FIRST_NAME | LAST_NAME | AGE  | SEX  | INCOME |
+------------+-----------+------+------+--------+
| Krishna    | Sharma    | 20   | M    | 2000   |
| Raj        | Kandukuri | 21   | M    | 7000   |
| Mohan      | Rao | 26   | F    | 5000   |
+------------+-----------+------+------+--------+
3 rows in set (0.07 sec)
```

VIVA QUESTIONS

1. What are DCL commands?
2. List out the uses of various DCL commands?
3. What is the difference between Commit, Rollback and Savepoint?
4. What is the difference between TCL & DCLcommands.
5. Who has the privilegeto access the DCLcommands.

CASE STUDY 1

Consider the following relations containing airline flight information:

Flights(flno: integer, from: string, to: string,

distance: integer, departs: time, arrives: time)

Aircraft(aid: integer, aname: string, cruisingrange: integer)

Certified(eid: integer, aid: integer)

Employees(eid: integer, ename: string, salary: integer)

Note that the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft (otherwise, he or she would not qualify as a pilot), and only pilots are certified to fly.

Write the following queries in relational algebra, tuple relational calculus, and domain relational

calculus.

1. Find the eids of pilots certified for some Boeing aircraft.

2. Find the names of pilots certified for some Boeing aircraft.

3. Find the aids of all aircraft that can be used on non-stop flights from Bonn to Madras.

4. Identify the flights that can be piloted by every pilot whose salary is more than $100,000.

(Hint: The pilot must be certified for at least one plane with a sufficiently large cruising

range.)

5. Find the names of pilots who can operate some plane with a range greater than 3,000

miles but are not certified on any Boeing aircraft.

6. Find the eids of employees who make the highest salary.

7. Find the eids of employees who make the second highest salary.

8. Find the eids of pilots who are certified for the largest number of aircraft.

9. Find the eids of employees who are certified for exactly three aircraft.

10. Find the total amount paid to employees as salaries.

CASESTUDY 2

Normalization Exercise

HEALTH HISTORY REPORT

PET ID	PET NAME		PET TYPE	PETAGE	OWNER	VISIT DATE	PROCEDURE
246	ROVER	DOG	12		SAM COOK	JAN 13/2002	01 - RABIES VACCINATION
						MAR 27/2002	10 - EXAMINE and TREAT WOUND
						APR 02/2002	05 - HEART WORM TEST
298	SPOT	DOG	2		TERRY KIM	JAN 21/2002	08 - TETANUS VACCINATION
						MAR 10/2002	05 - HEART WORM TEST
341	MORRIS	CAT	4		SAM COOK	JAN 23/2001	01 - RABIESVACCINATION
						JAN 13/2002	01 - RABIESVACCINATION
519	TWEEDY	BIRD	2		TERRY KIM	APR 30/2002	20 - ANNUAL CHECK UP
						APR 30/2002	12 - EYE WASH

Normal Forms:

Pet [pet_id, pet_name, pet_type, pet_age, owner, (visitdate, procedure_no,procedure_name)]

1NF:

Pet [pet_id, pet_name, pet_type, pet_age, owner]
Pet_Visit [pet_id, visitdate, procedure_no, procedure_name]

2NF:

Pet [pet_id, pet_name, pet_type, pet_age, owner] Pet_Visit [
pet_id, visitdate, procedure_no
] Procedure [procedure_no, procedure_name]