

# **DATABASEMANAGEMENTSYSTEMS**

## **LABORATORY MANUAL & RECORD**

**B.TECH  
(IIYEAR–IISEM)  
(2021-22)**



## **DEPARTMENT OF COMPUTERSCIENCEANDENGINEERING**

# **MALLAREDDY COLLEGE OF ENGINEERING & TECHNOLOGY**

**(AutonomousInstitution–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 COMPUTER SCIENCE AND ENGINEERING**

### **Vision**

- To acknowledge quality education and instill high patterns of discipline making the student's technologically superior and ethically strong which involves the improvement in the quality of life in human race.

### **Mission**

- To achieve and impart holistic technical education using the best of infrastructure, outstanding technical and teaching expertise to establish the students to competent and confident engineers.
- Evolving the center of excellence through creative and innovative teaching learning practices for promoting academic achievement to produce international competitive and world class professionals.

## **PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)**

### **PEO1–ANALYTICALSKILLS**

1. To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing the irproductivity.

### **PEO2–TECHNICALSKILLS**

2. To facilitate the graduates with the technical skills that prepare them for immediate employment and pursue certification providing adeeper understanding ofthetechnologyinadvancedareasofcomputerscienceandrelatedfields,thusencouragi ngtopursuehighereducationandresearchbasedontheirinterest.

### **PEO3–SOFTSKILLS**

3. To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self confidence by communicating effectively, havingapositive attitude, get involved in team-work, being a leader, managing their careerandtheirlife.

### **PEO4–PROFESSIONALETHICS**

4. Tofacilitatethegraduateswiththeknowledgeofprofessionalandethicalresponsibilitiesby paying attention to grooming, being conservative with style, following dress codes, safetycodes, andadaptingthemselves totechnologicaladvancements.

## PROGRAM SPECIFIC OUTCOMES (PSOs)

Engineering Graduates will be able to:

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 th 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.

## INDEX

S. No	Topic	Page no	Date	Sign
1	<b>E-R Model:</b> Analyze the problem with the entities which identify data persisted in the database which contains entities, attributes.	8		
2	<b>Concept design with E-R Model:</b> Apply cardinalities for each relationship, identify strong entities and weak entities for relationships like generalization, aggregation, specialization.	17		
3	<b>Relation Model:</b> Represent attributes as columns in tables and different types of attributes like Composite, Multi-valued, and Derived.	19		
4	<b>Normalization</b>	23		
5	<b>Installation of MySql and MONGO DB.</b>	26& 34		
6	<b>Practicing DML commands</b> SELECT, INSERT, UPDATE, DELETE.	39		
7	<b>Querying</b> Queries using ANY, ALL, IN, INTERSECT, UNION	52		
8	<b>Querying</b> Using aggregate functions COUNT, SUM using GROUPBY andHAVING.	69		
9	<b>Querying</b> Using aggregate functions AVERAGE using GROUPBY andHAVING	69		
10	<b>TRIGGER</b>	77		
11	<b>Procedures</b> Creation, Execution and Modification of stored Procedure	82		
12	<b>Stored Procedures&amp;PI/SQL</b> Creation, Execution and Modification of stored Procedure	82		
13	DCL Commands	87		
	<b>CASESTUDY1</b>	92		

## **INTRODUCTION**

### **Hierarchical Model**

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

#### **The drawbacks of this model are:**

The hierarchical structure is not flexible to represent all the relationship proportions, which occur in the real world.

It cannot demonstrate the overall data model for the enterprise because of the non-availability of actual data at the time of designing the data model.

It cannot represent the Many-to-Many relationship.

### **Network Model**

It supports the One-To-One and One-To-Many types only. The basic objects in this model are Data Items, Data Aggregates, Records and Sets.

It is an improvement on the Hierarchical Model. Here multiple parent-child relationships are used. Rapid and easy access to data is possible in this model due to multiple access paths to the data elements.

### **Relational Model**

Does not maintain physical connection between relations. Data is organized in terms of rows and columns in a table.

The position of a row and/or column in a table is of no importance. The intersection of a row and column must give a single value.

### **Features of an RDBMS**

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

Retrieval of information stored in more than one table.

An RDBMS product has to satisfy at least seven of the 12 rules of Codd to be accepted as a full-fledged RDBMS.

## **RelationalDatabaseManagementSystem**

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

Simple example of Relational model is as follows:

### **StudentDetailsTable**

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

### **StudentMarksheetTable**

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

Here, both tables are based on students details. Common field in both tables is Rollno. So we can say both tables are related with each other through Rollno column.

### **Degree of Relationship**

One to One (1:1)

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

1) Many to Many (M:M)

The Degree of Relationship indicates the link between two entities for a specified occurrence of each.



**One to One Relationship:****(1:1)11****StudentHasRollNo.**

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

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

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)MM****StudentsAppearsTests**

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 systems are built.

Oracle has introduced added functionality to this by incorporating object-oriented capabilities. Now it is known as Object Relational Database Management System (ORDBMS). Object-oriented concept is added in Oracle 8.

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 stored in tables as data values.

### **2. The Rule of Guaranteed Access**

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

### **3. The Systematic Treatment of Null Values**

The RDBMS must be taken care of null values to represent missing or inapplicable information.

### **4. The Database Description Rule**

A description of database is maintained using the same logical structures with which data was defined by the RDBMS.

### **5. Comprehensive Data SubLanguage**

According to the rule the system must support data definition, view definition, data manipulation, integrity constraints, authorization and transaction management operations.

### **6. The View Updating Rule**

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

### **7. 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.

### **8. The Physical Independence Rule**

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

### **9. 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 a table into many more tables.

### **10. The Integrity Independence Rule**

The integrity constraints should be stored in the system catalog or in the database.

### **11. The Distribution Rule**

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

## **12. 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.

### **Object Relational Database Management System**

Oracle8 and later versions are supported object-oriented concepts. A structure once created can be reused is the fundamental of the OOP's concept. So we can say Oracle8 is supported Object Relational model, Object-oriented model both. Oracle products are based on a concept known as a client-server technology. This concept involves segregating the processing of an application between two systems. One performs all activities related to the database (server) and the other performs activities that help the user to interact with the application (client). A client or front-end database application also interacts with the database by requesting and receiving information from database server. It acts as an interface between the user and the database.

The database server or back end is used to manage the database tables and also respond to client requests.

### **Introduction to ORACLE**

ORACLE is a powerful RDBMS product that provides efficient and effective solutions for major database features. This includes:

Large databases and space management  
control Many concurrent database users

High transaction processing performance  
High availability

Controlled availability

Industry accepted  
standards Manageable security

Database enforced  
integrity Client/Server environment

Distributed database  
systems Portability

Compatibility

Connectivity

An ORACLE database system can easily take advantage of distributed processing by using its Client/Server architecture. In this architecture, the database system is divided into two parts:

**A front-end or client portion**

The client executes the database application that accesses database information and interacts with the user.

**A back-end or server portion**

The server executes the ORACLE software

and handles the functions required for concurrent, shared data access to ORACLE database.



## **ILLUSTRATION:ROADWAYTRAVELS**

**“RoadwayTravels”**isinbusinesssince

1977withseveralbusesconnectingdifferentplacesinIndia.ItsmainofficeislocatedinHyderabad.

Thecompanywantstocomputerizeitsoperationsinthefollowingareas:

Reservations

Ticketing

Cancellations

### **Reservations:**

Reservationsare directlyhandedbybookingoffice.reservationscanbemade60daysin advance in either cash or credit. In case the ticket is not available,a wait listed ticket is issuedtothecustomer.This ticketisconfirmedagainstthecancellation.

### **Cancellationand modification:**

Cancellations are also directly handed at the booking office. Cancellation chargeswillbecharged.

Waitlistedtickets thatdonotgetconfirmedarefullyrefunded.

## WEEK-1

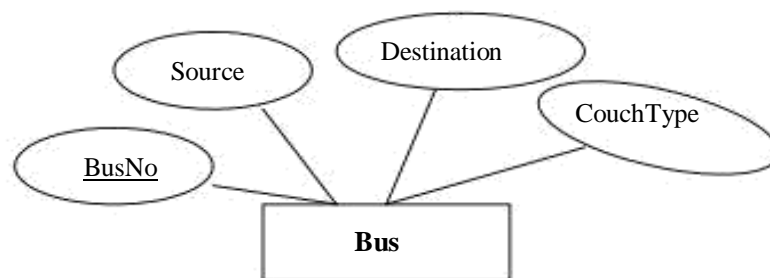
**AIM:**Analyze the problem and come with the entities.

**Identify what Data has to be persisted in the databases.**

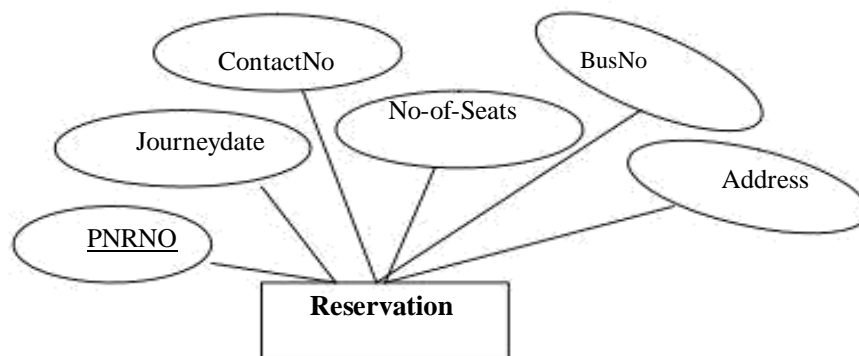
The Following are the entities:

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

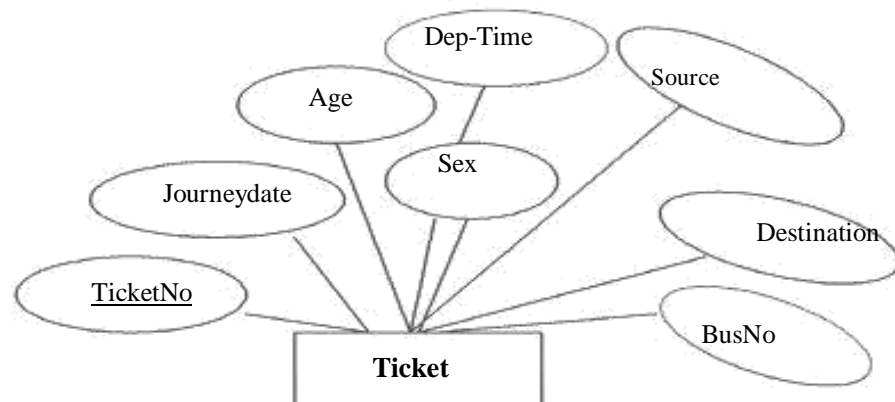
**The attributes in the Entities:**  
**us: (Entity)**



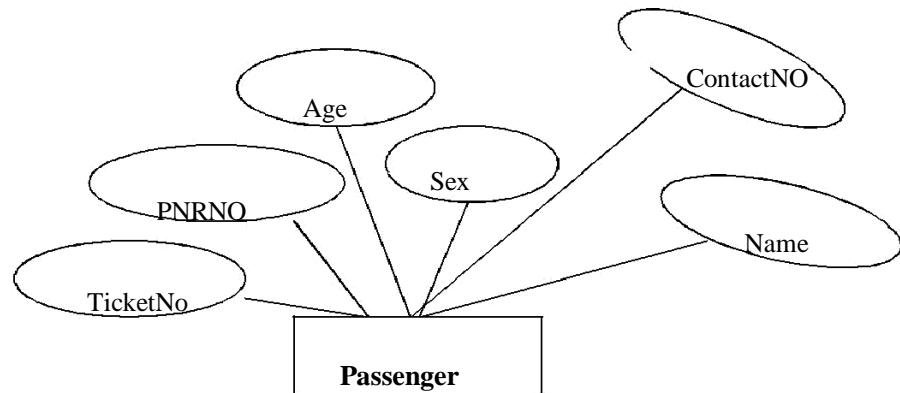
**Reservation (Entity)**



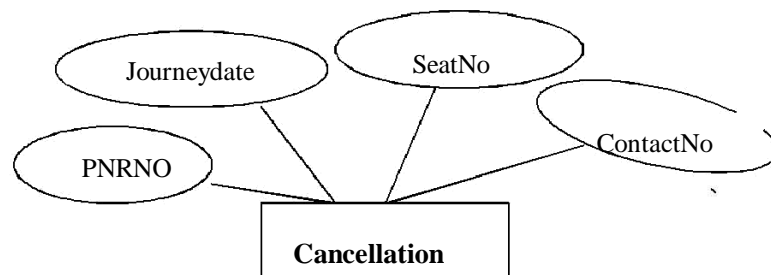
**Ticket: (Entity)**



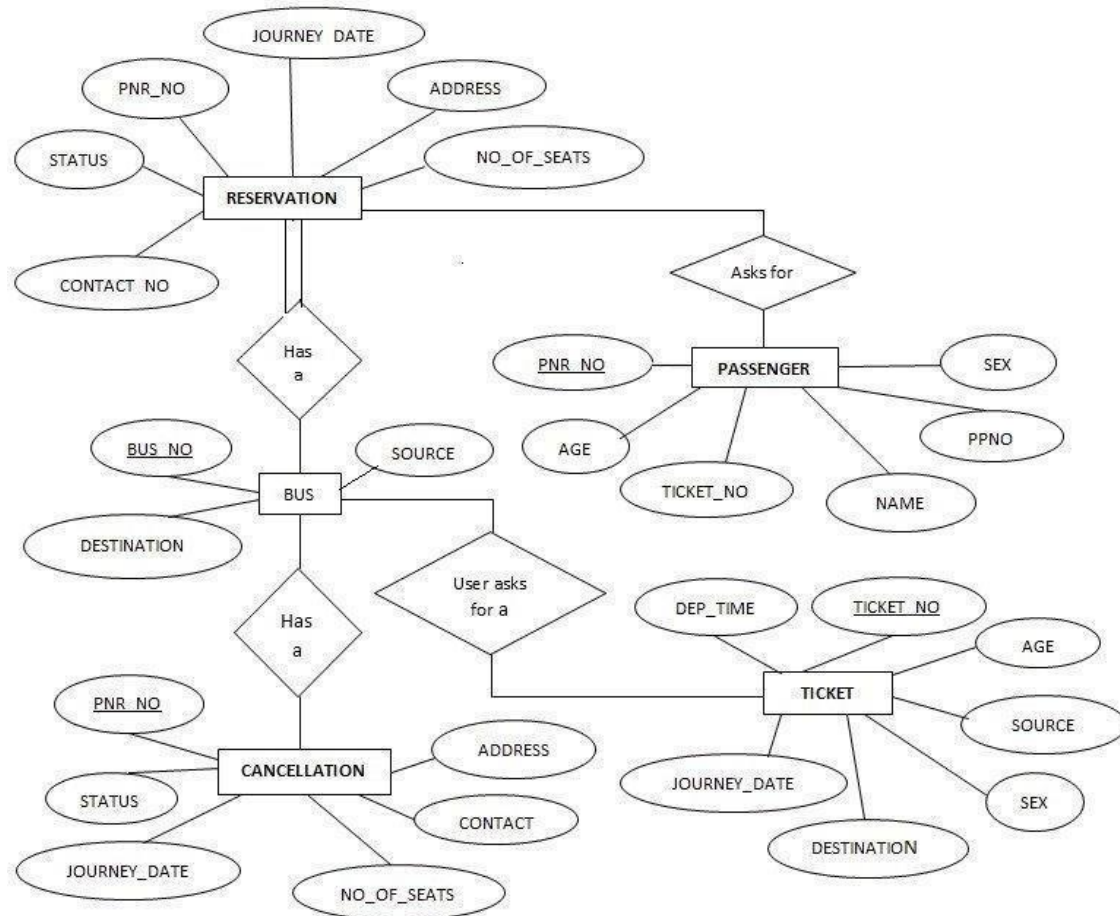
**Passenger:**



**Cancellation(Entity)**



### Concept design with E-R Model:





### **CASE STUDY 1:**

- Consider the following information about a university database:
- Professors have an SSN, a name, an age, a rank, and a research specialty.
- Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budget.
- Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.).
- Each project is managed by one professor (known as the project's principal investigator).
- Each project is worked on by one or more professors (known as the project's co-investigators).
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate students (known as the project's research assistants).
- When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.
- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professors work in one or more departments, and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working on their degree.
  - Each graduate student has another, more senior graduate student (known as a student advisor) who advises him or her on what courses to take.

## **What is SQL and SQL\*Plus**

Oracle was the first company to release a product that used the English-based Structured Query Language or SQL. This language allows end users to manipulate information of table (primary database object). To use SQL you need not to require any programming experience. SQL is a standard language common to all relational databases. SQL is database language used for storing and retrieving data from the database. Most Relational Database Management Systems provide extension to SQL to make it easier for application developer. A table is a primary object of database used to store data. It stores data in form of rows and columns.

SQL\*Plus is an Oracle tool (specific program) which accepts SQL commands and PL/SQL blocks and executes them. SQL \*Plus enables manipulations of SQL commands and PL/SQL blocks. It also performs additional tasks such as calculations, store and print query results in the form of reports, list column definitions of any table, access and copy data between SQL databases and send messages to and accept responses from the user. SQL \*Plus is a character based interactive tool, that runs in a GUI environment. It is loaded on the client machine.

To communicate with Oracle, SQL 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 Savepoint

### **4. Data Control Language**

Grant and Revoke

Before we take a look on above-mentioned commands we will see the data types available in Oracle.

### **Oracle Internal Datatypes**

When you create a table in Oracle, a few items should be important, not only do you have to give each table a name (e.g. employee, customer), you must also list all the columns or fields (e.g. First\_name, Mname, Last\_name) associated with the table. You also have to specify what type of information that table will hold to the database. For example, the column Empno holds numeric information. An Oracle database can hold many different types of data.

### **Datatype Description**

**Char(Size)** Stores fixed-length character data to store

alphanumeric values, with a maximum size of 2000 bytes. Default and minimum size is 1 byte.

**Varchar2(Size)** Stores variable-length character data to store alphanumeric

values, with a maximum size of 4000 bytes.

**char(Size)** Stores fixed-length character data of length size characters or bytes, depending on the choice of national character set. Maximum size is determined by the number of bytes required storing each character with an upper limit of 2000 bytes. Default and minimum size is 1 character or 1 byte, depending on the character set.

**Nvarchar2(Size)** Stores variable-

length character string having maximum length size characters or bytes, depending on the choice of national character set. Maximum size is determined by the number of bytes required to store each character, with an upper limit of 4000 bytes.

**Long** Stores variable-length character data up to 2GB (Gigabytes). Its length would be restricted based on memory space available in the computer.

**Number [p,s]** Number having precision p and scale s. The precision p indicates total number of digit varies from 1 to 38. The scale s indicates number of digit in fraction part varies from -84 to 127.

**Date** Stores dates from January 1, 4712 B.C. to December 31, 4712 A.D. Oracle predefine format of Date datatype is DD-MON-YYYY.

**Raw (Size)** Stores binary data of length size. Maximum size is 2000 bytes. One must have to specify size with RAW type data, because by default it does not specify any size. **LongRawStore binary data of variable length up to 2GB (Gigabytes).**

### **LOBS-LARGE OBJECTS**

LOB is used to store unstructured information such as sound and video clips, pictures up to 4 GB size.

**CLOB** A Character Large Object containing fixed-width multi-byte characters. Varying-

width character sets are not supported. Maximum size is 4GB.

**NCLOB** A National Character Large Object containing fixed-width multi-byte characters.

Varying-

width character sets are not supported. Maximum size is 4GB. Stores national character set data.

**BLOB** To store a Binary Large Object such as graphics, video clips and sound files.

Maximum size is 4GB.

**BFILE** Contains a locator to a large Binary File stored outside the database.

Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4GB. Apart from Oracle internal datatypes, users can create their own datatype, which is used in data base and other database object. We will discuss it in the later part.

The following are tabular representation of the above entities and relationships

**BUS:**

<b><u>COLUMNNAME</u></b>	<b><u>DATATYPE</u></b>	<b><u>CONSTRAINT</u></b>
BusNo	varchar2(10)	<b>PrimaryKey</b>
Source	varchar2(20)	
Destination	varchar2(20)	
CouchType	varchar2(20)	

**Reservation:**

<u>COLUMNNAME</u>	<u>DATATYPE</u>	<u>CONSTRAINT</u>
PNRNo	number(9)	<b>PrimaryKey</b>
Journeydate	Date	
No-of-seats	integer(8)	
Address	varchar2(50)	
ContactNo	Number(9)	Shouldbeequalto10 numbersandnotallow otherthannumeric
BusNo	varchar2(10)	<b>Foreignkey</b>
Seatno	Number	

**Ticket:**

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

**Passenger:**

<u>COLUMNNAME</u>	<u>DATATYPE</u>	<u>CONSTRAINT</u>
PNRNo	Number(9)	<b>PrimaryKey</b>
TicketNo	Number(9)	Foreignkey
Name	varchar2(15)	
Age	integer(4)	
Sex	char(10)	(Male/Female)
Contactno	Number(9)	Shouldbeequalto10numbers andnotallowotherthan numeric

**Cancellation:**

<u>COLUMNNAME</u>	<u>DATATYPE</u>	<u>CONSTRAINT</u>
PNRNo	Number(9)	Foriegn-key
Journey-date	Date	
Seatno	Integer(9)	
Contact_No	Number(9)	Shouldbeequalto10numbers andnotallowotherthan numeric

## WEEK 2

Concept design with E-R Model and apply cardinalities for each relationship. Identify strong entities and weak entities for relationships like generalization, aggregation, specialization.

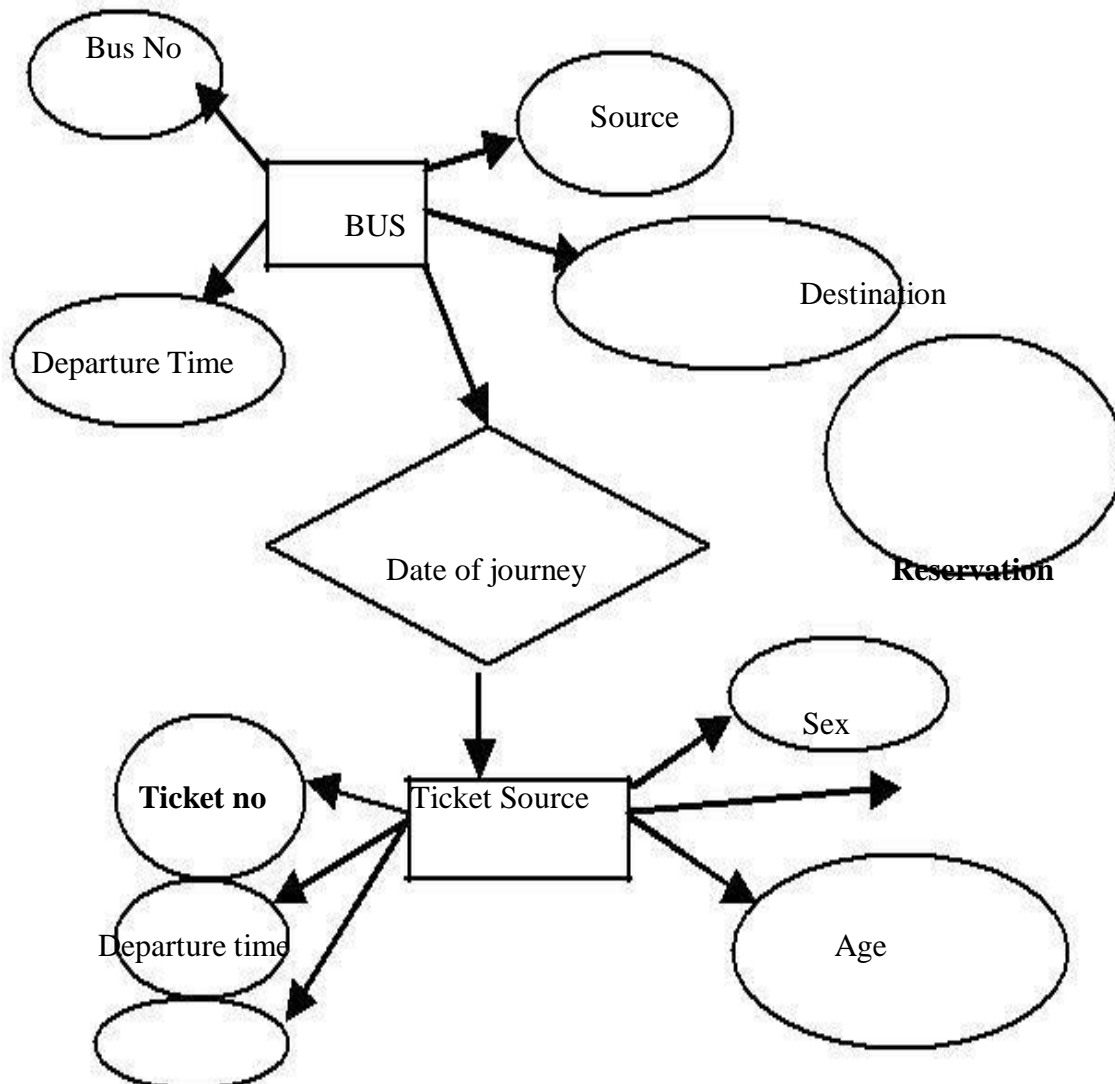
### Objectives:

Student will able to learn data structures in terms of entity types, relationship types and attributes or classes, associations and attributes.

### Outcomes:

Student gains the ability to describe the data requirements for a new information system in a direct and easy to understand graphical notation.

### E R diagram:



### **VIVA QUESTIONS**

1. Draw an E-R Diagram For an ATM System.
2. Draw an E-R Diagram For school mgmt system.
3. Draw an E-R Diagram For Roadways Travels Systems.
4. Draw an E-R Diagram For Bank Mgmt System.
5. Explain many to many and many to one relationship.



## WEEK 3

### AIM

**Relation Model represents attributes as columns in tables and different types of attributes like composite, Multi-valued and Derived.**

### Objectives:

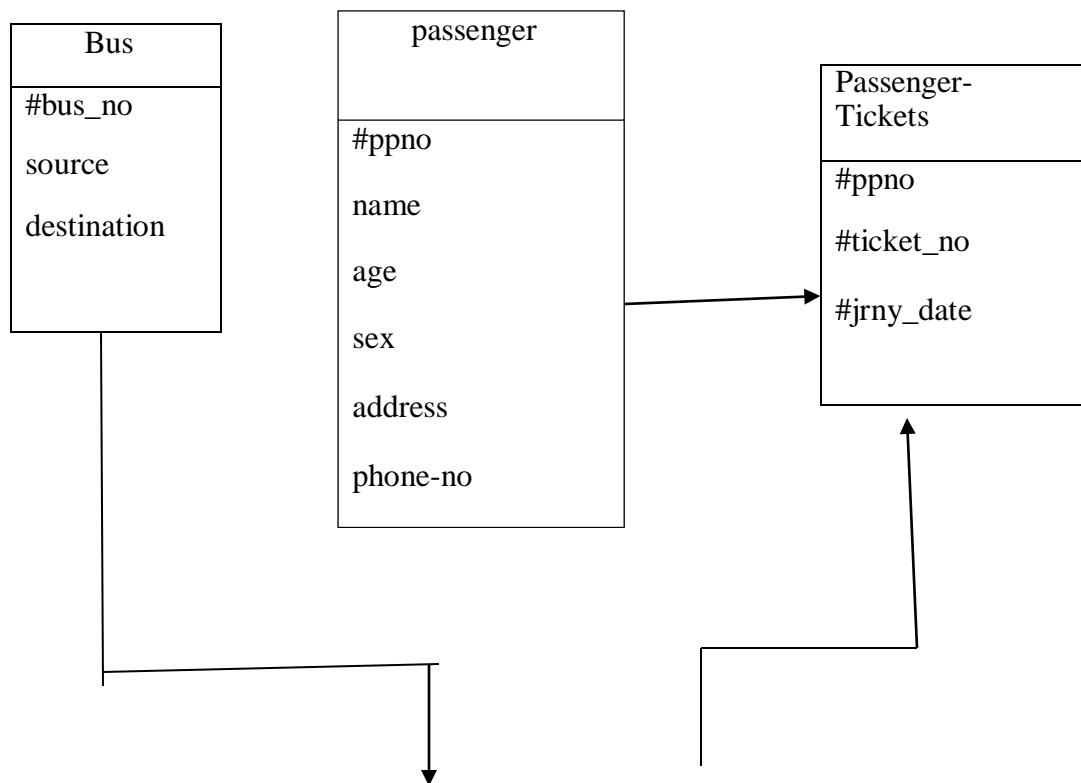
Student will able to learn the structural components of the relational data model.  
Student will able to learn to map ER models into relational models.

### Outcomes:

Student gains the ability

- To describe the Model Structure.
- To define Properties of Relations.
- To define Domains.
- To implement Notation to Describe the Relational Schema
- To Represent an ER Model as a Relational Model.

Example: The passenger tables look as below. This is an example. You can add more attributes based on your E-R model. This is not a normalized table. Passenger



Tickets
#tickets_no
no of tkts
From_place
T0_place
#Bus_no
#jrny_date

Name	Age	Sex	Address	<u>Passport</u>
				<u>ID</u>

**Note:** The student is required to submit a document by Represent relationships in a tabular fashion to the lab teacher.

## 2. Concept design with E-R model

Relate the entities appropriate for each relationship. Identify strong entities and weak entities (if any). Indicate the type of relationships (total/partial). In this we will design the different E-R diagram for different entities and also the whole “Roadway Travels”.

E-R diagram: An entity-relationship(ER) diagram is a specified graphic that illustrates the interrelationships between entities and database. We can express the overall logical structure of database graphically with an E-R diagram.

## 3. Relational Model and Normalization

Represent all the entities (Strong, Weak) in tabular fashion. Represent relationships in a tabular fashion. There are different ways of representing relationships as tables based on the cardinality. Represent attributes as columns in tables or as tables based on the requirement. In this we will represent the different entities, attributes of different keys in a tabular fashion or manner.

## Relational Model:

The relational model is a depiction of how each piece of stored information relates to the other stored information. It shows how tables are linked, what type of the links are between tables, what keys are used, what information is referenced between tables. It's an essential part of developing a normalized database structure to prevent repeat and redundant data storage.

Different types of keys:

- A super key is a set of one or more attributes which; taken collectively, allow us to identify uniquely an entity in the entity set.
- A primary key is a candidate key(there may be more than one) chosen by the DB designer to identify entities in an entity set.
- A super key may contain extraneous attributes, and we are often interested in the smallest super key. A super key for which no subset is a super key is called a candidate key.
- An entity does not possess sufficient attributes to form a primary key is called a weak entity set. One that does have a primary key is called a strong entity set.
- A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables.

## 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 logical or structural problems, namely data anomalies. In this we will write the normalization tables that is entities of "Roadway Travels."

**Normalization:** In relational databases, normalization is a process that eliminates redundancy, organizes data efficiently; Normalization is the process of efficiently organizing data in a database. There are two goals of the normalization process: eliminating redundant data(for example, storing the same data in more than one table) and ensuring data dependencies make sense(only storing related data in a table). Both of these are worthy goals as they reduce the amt of space a database consumes and ensure that data is logically stores.

The Normal Form: the database community has developed a series of guidelines for ensuring that databases are normalized. These are referred to as normal forms and are numbered from one ( the lowest form to normalization, referred to as first form or 1NF) through five(fifth normal form or 5NF). In practical applications, you'll often see 1NF, 2NF, and 3NF along with occasional 4NF. Fifth normal form is very rarely seen and won't be discussed in this article. It's important to point out that they are guidelines and guidelines only. Occasionally, it becomes necessary to stray from them to meet practical business requirements. However, when variations take place, it's extremely important to evaluate any possible requirements they could have on your system and account for possible inconsistencies. That said, let's explore the normal form.

### **VIVA QUESTIONS**

1. What is relational model and its importance.
2. Explain the difference between candidate key and primary key.
3. What is a super key.
4. Differentiate among all types of keys with example.
5. Explain the need of foreign key.

## **WEEK 4**

### **AIM**

#### **Normalization of tables**

#### **Objectives:**

Student will be able to learn to avoid problems that are associated with updating redundant data.

#### **Outcomes:**

Student gains the knowledge to build The database that does not have redundant data.

A basic objective of the first normal form defined by Edgar Frank "Ted" Codd in 1970 was to permit data to be queried and manipulated using a "universal data sub-language" grounded in first-order logic. (SQL is an example of such a data sub-language, albeit one that Codd regarded as seriously flawed.)

The objectives of normalization beyond 1NF (First Normal Form) were stated as follows by Codd:

1. To free the collection of relations from undesirable insertion, update and deletion dependencies;
2. To reduce the need for restructuring the collection of relations, as new types of data are introduced, and thus increase the life span of application programs;
3. To make the relational model more informative to users;
4. To make the collection of relations neutral to the query statistics, where these statistics are liable to change as time goes by.

Querying and manipulating the data within a data structure which is not normalized, such as the following non-1NF representation of customers' credit card transactions, involves more complexity than is really necessary:

Customer	Transactions		
	Tr. ID	Date	Amount
Jones	12890	14-Oct-2003	-87
	12904	15-Oct-2003	-50
Wilkinson	r. ID	Date	Amount
	12898	14-Oct-2003	-21
Stevens	Tr. ID	Date	Amount
	12907	15-Oct-2003	-18
	14920	20-Nov-2003	-70
	15003	27-Nov-2003	-60

To each customer corresponds a repeating group of transactions. The automated evaluation of any query relating to customers' transactions therefore would broadly involve two stages:

1. Unpacking one or more customers' groups of transactions allowing the individual transactions in a group to be examined, and
2. Deriving a query result based on the results of the first stage

For example, in order to find out the monetary sum of all transactions that occurred in October 2003 for all customers, the system would have to know that it must first unpack the *Transactions* group of each customer, then sum the *Amounts* of all transactions thus obtained where the *Date* of the transaction falls in October 2003.

One of Codd's important insights was that this structural complexity could always be removed completely, leading to much greater power and flexibility in the way queries could be formulated (by users and applications) and evaluated (by the DBMS). The normalized equivalent of the structure above would look like this:

<b>Customer</b>	<b>Tr. ID</b>	<b>Date</b>	<b>Amount</b>
Jones	12890	14-Oct-2003	-87
Jones	12904	15-Oct-2003	-50
Wilkins	12898	14-Oct-2003	-21
Stevens	12907	15-Oct-2003	-18
Stevens	14920	20-Nov-2003	-70
Stevens	15003	27-Nov-2003	-60

Now each row represents an individual credit card transaction, and the DBMS can obtain the answer of interest, simply by finding all rows with a *Date* falling in October, and summing their *Amounts*. The data structure places all of the values on an equal footing, exposing each to the DBMS directly, so each can potentially participate directly in queries; whereas in the previous situation some values were embedded in lower-level structures that had to be handled specially. Accordingly, the normalized design lends itself to general-purpose query processing, whereas the unnormalized design does not.

### **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-5(a)**

**AIM:**Installation of My SQL and practicing DDL & DML commands.

### **1.StepsforinstallingMySQL**

#### **Step1**

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

#### **Step2**

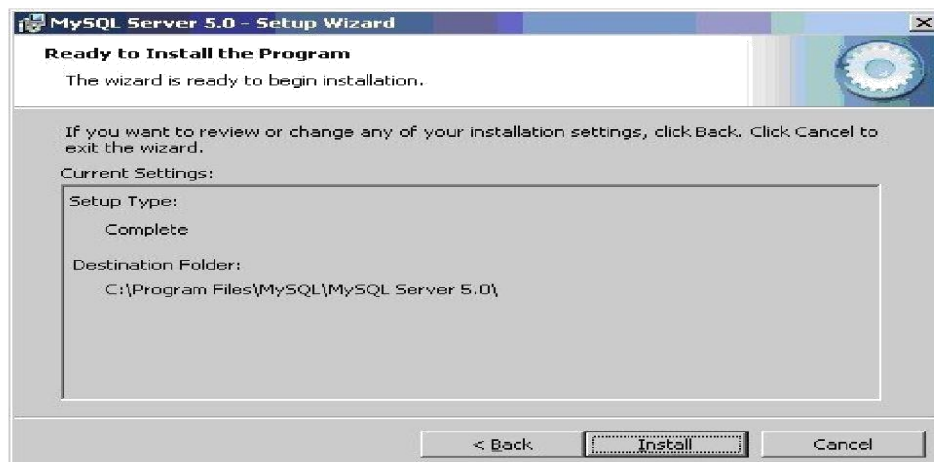
This is MySQL Server 5.0 setup wizard. The setup wizard will install MySQL Server 5.0 release 5.0.45 on your computer. To continue, click **next**.





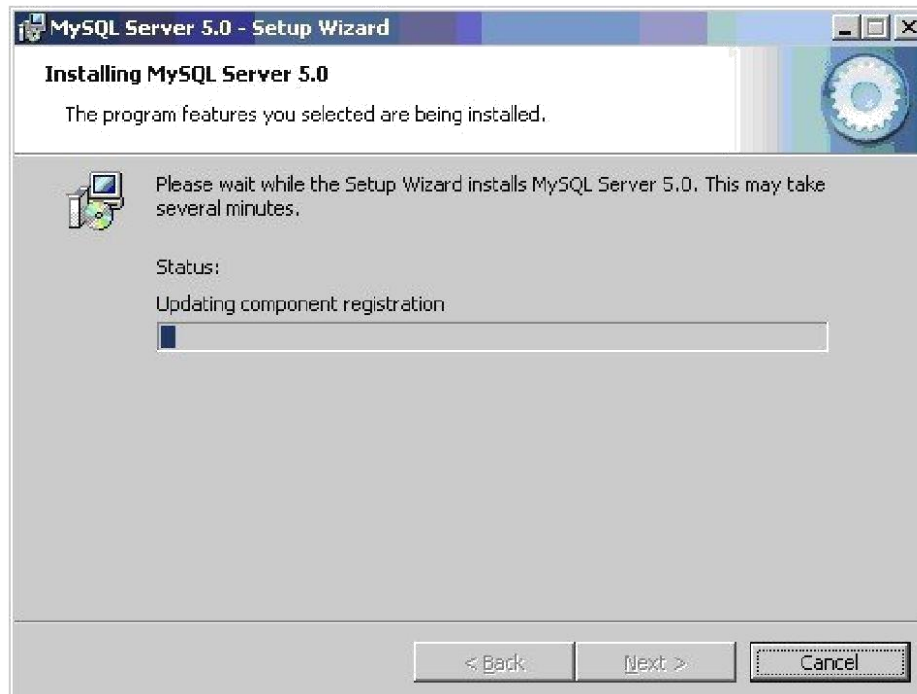
### Step3

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



### Step4

This wizard is ready to begin installation. Destination folder will be in **C:\Program Files\MySQL\MySQL Server 5.0\**. To continue, click **next**.

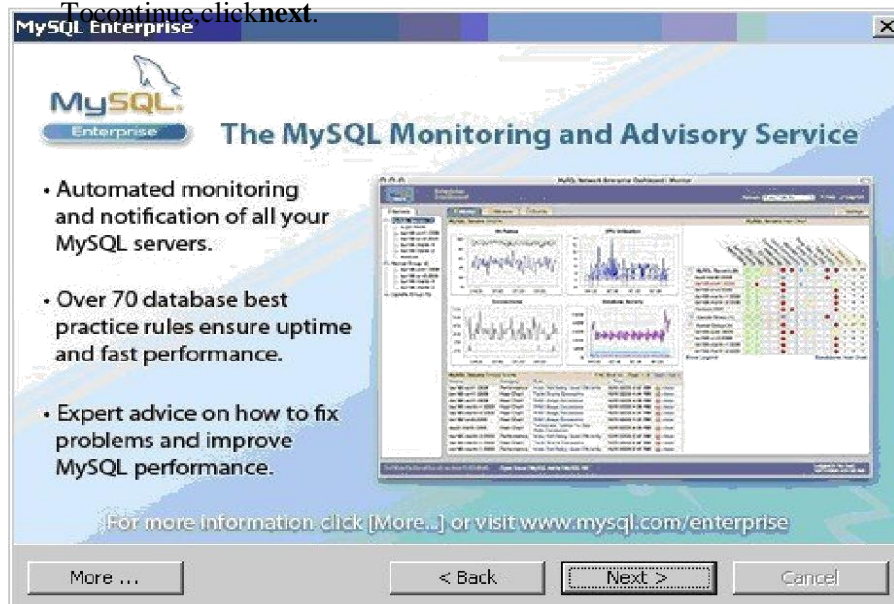


## Step5



### Step6

To continue, click next.



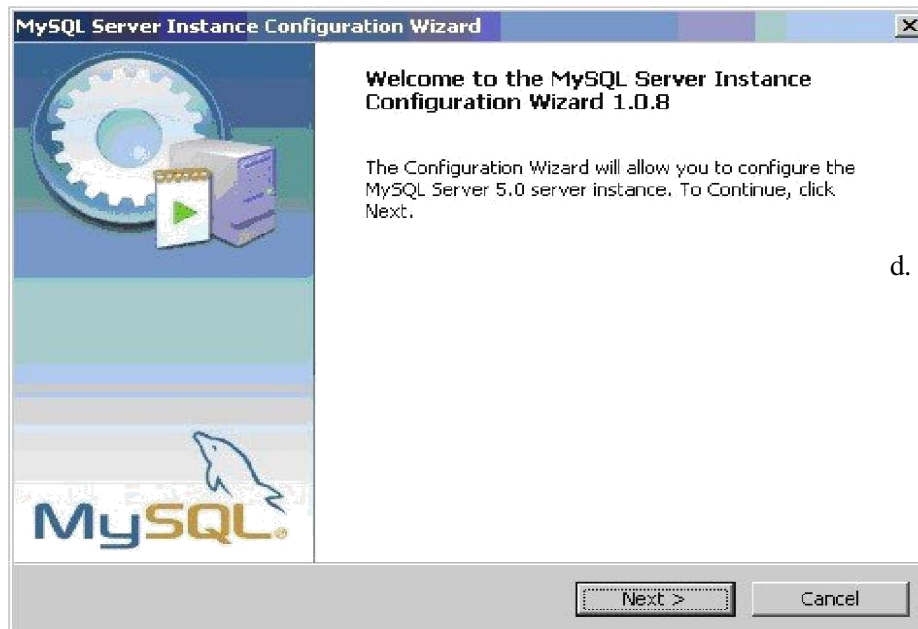
### Step7

To continue, click next.



### Step8

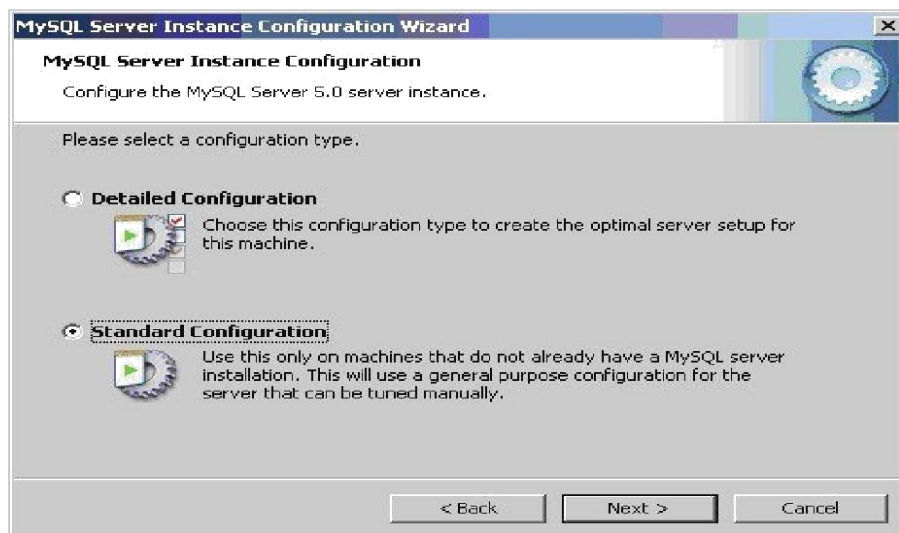
Wizard Completed. Setup has finished installing MySQL 5.0. **Check** the configure the MySQLserver nowtocontinue.Click**Finish**toexitthewizard



d.

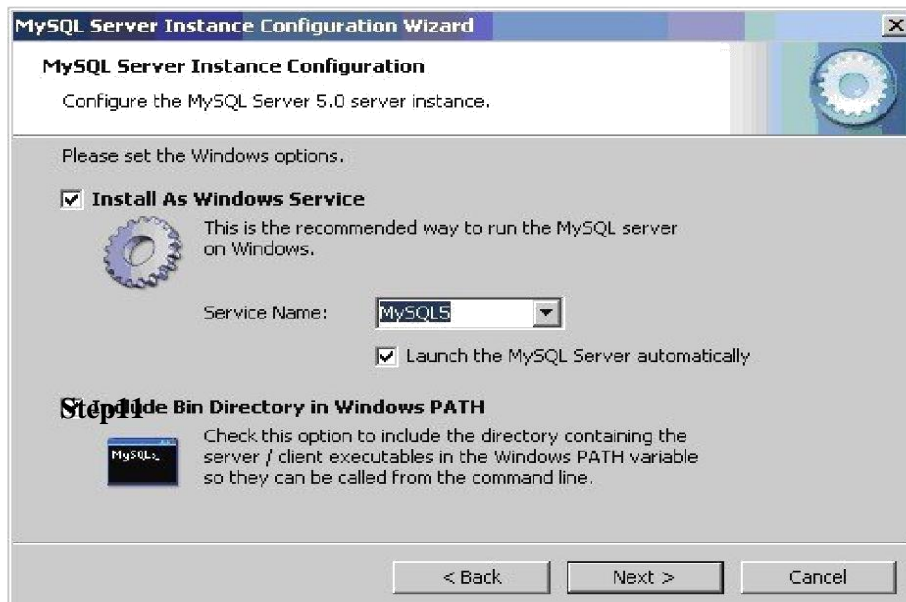
### Step9

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

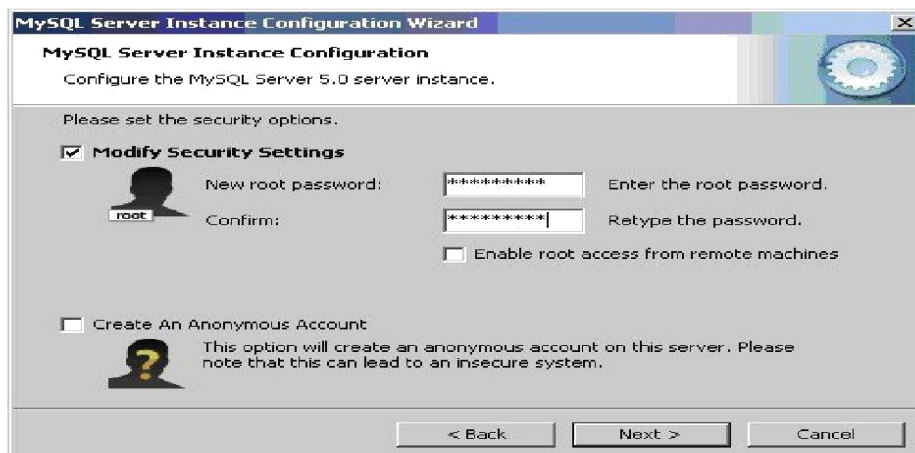


### Step10

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



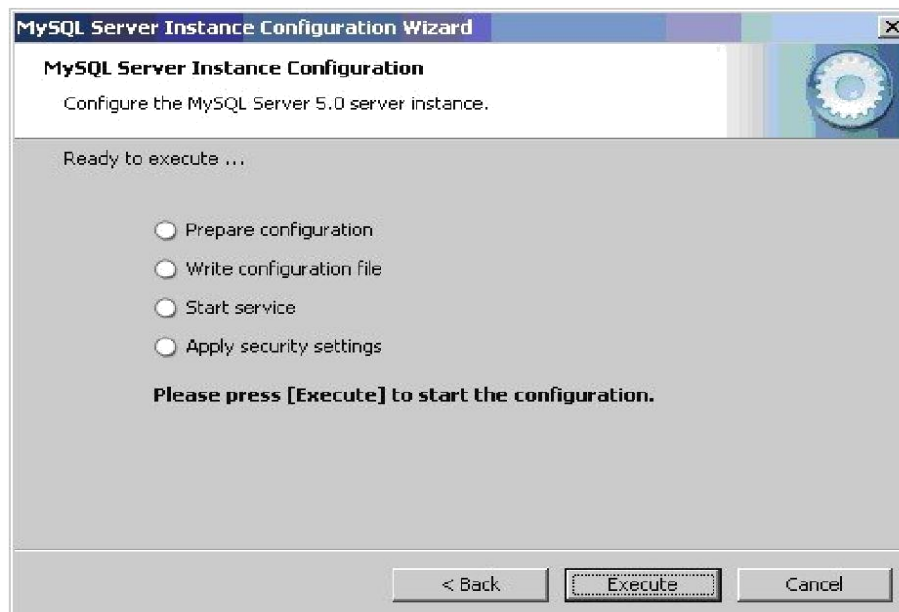
Check on the **install as windows service** and **include bin directory in windows path**. To continue, click **next**.



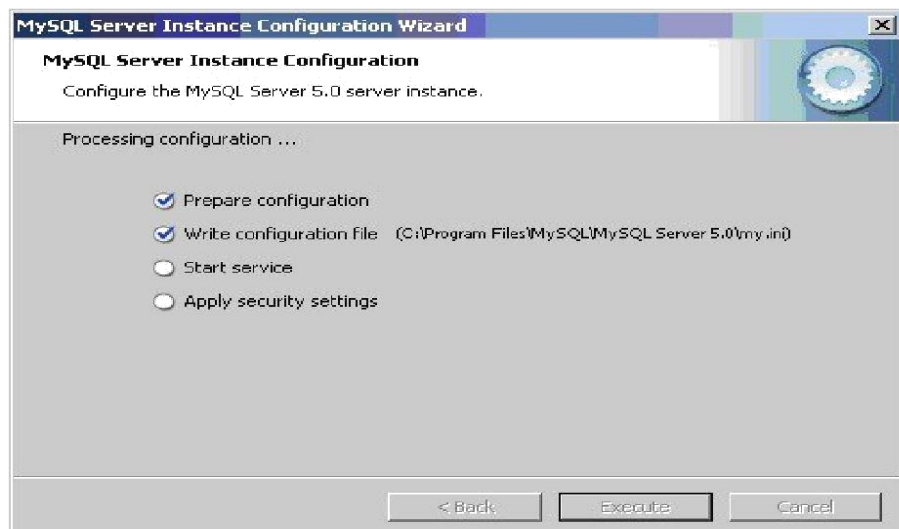
## Step 12

Please set these security options by entering the root password and confirm retype the password.

continue,clicknext.



**Step13**  
Readytoexecute?Clicksexcutetocontinue.

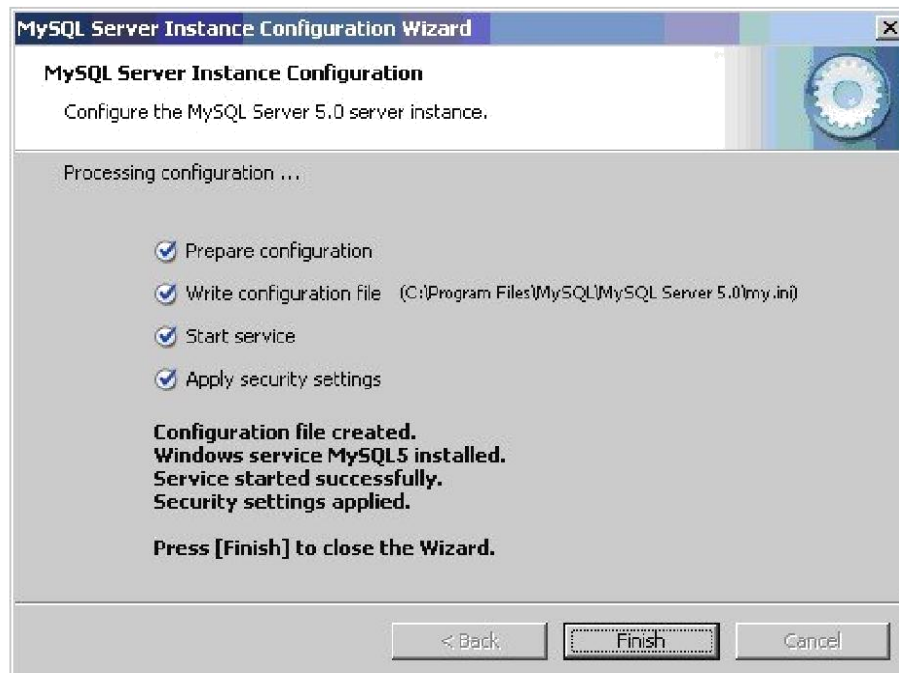


**Step14**  
Processingconfigurationinprogress.



## Step15

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



## WEEK-5(b)

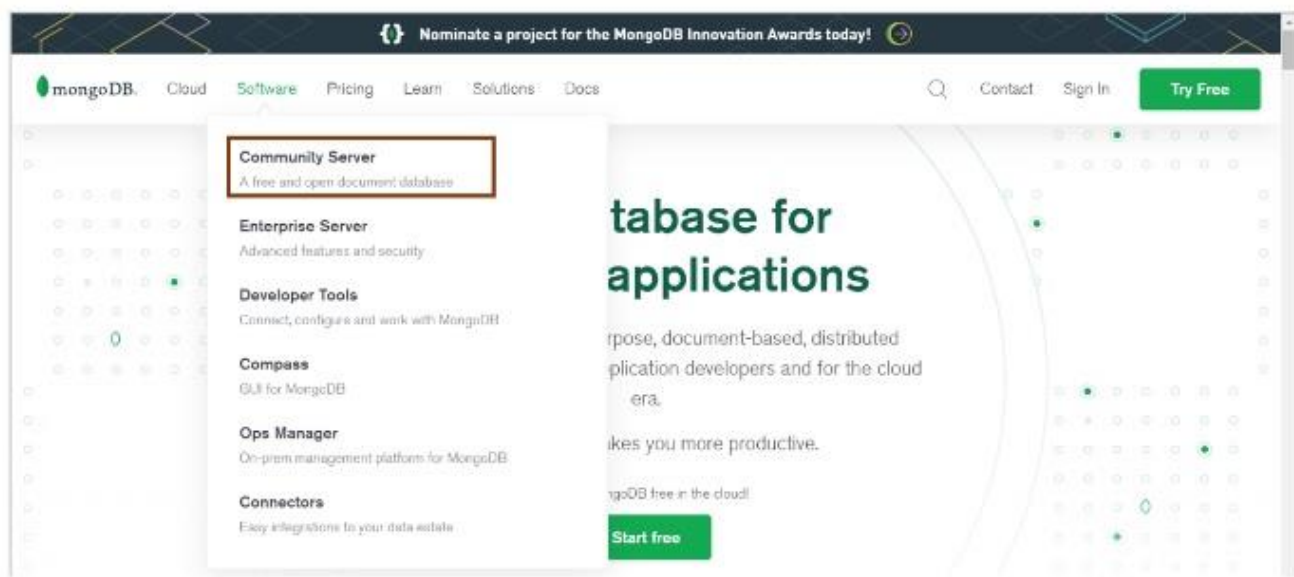
### MongoDB

Navigate to the download site:

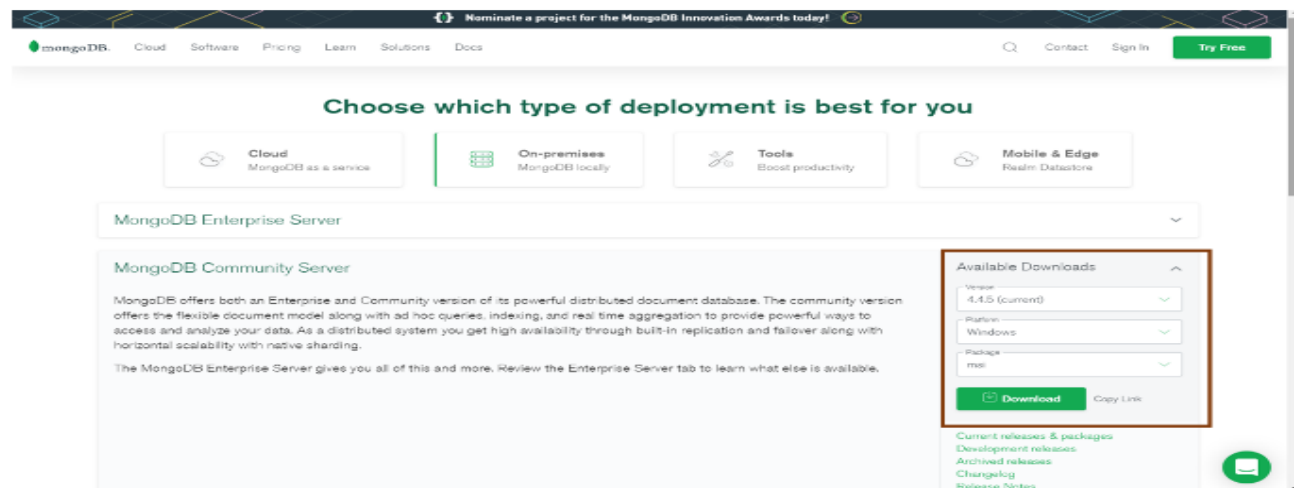
Navigate to the official MongoDB [website](https://www.mongodb.com/) <https://www.mongodb.com/>

Cross-check the Specifications and Download MongoDB

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



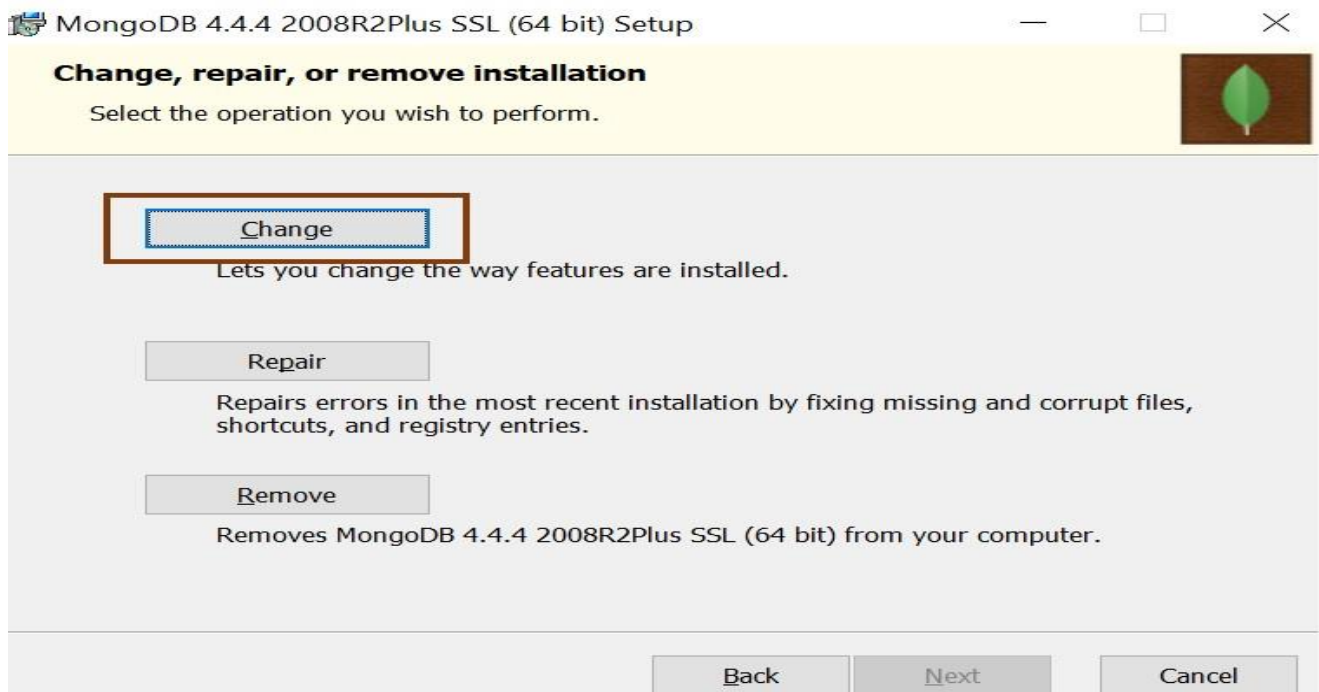
At the time of writing, the latest version is 4.4.5. Ensure that the platform is Windows, and the package is MSI. Go ahead and click on download.



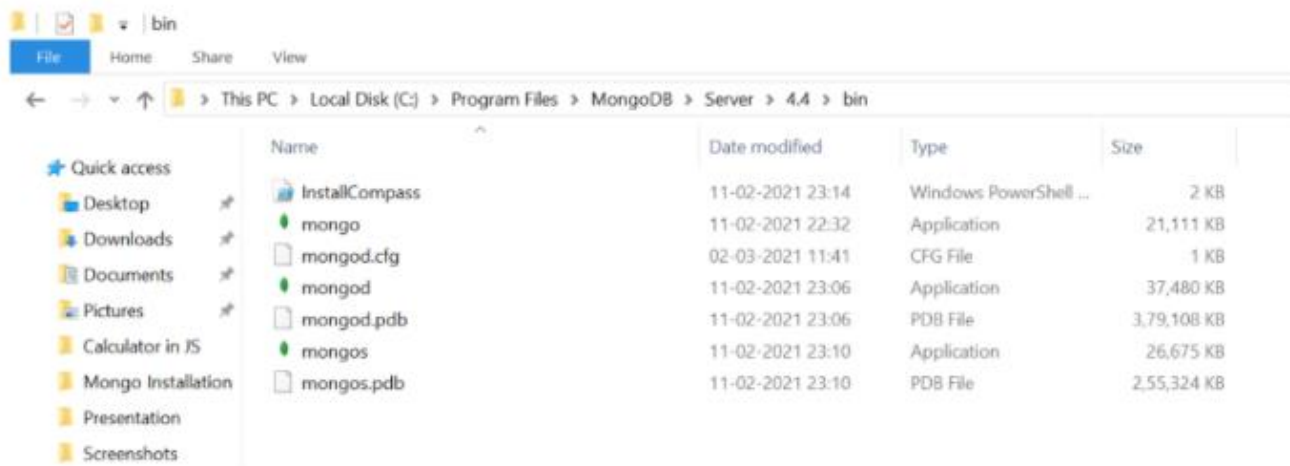


## Mongo DB Installation:

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



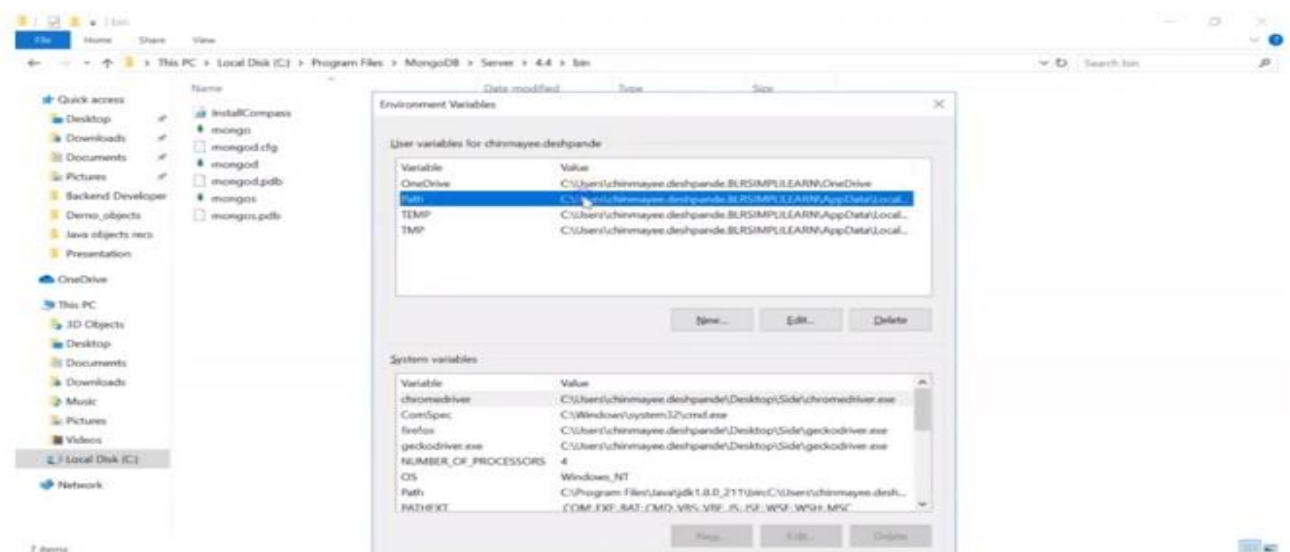
On completing the installation successfully, you will find the software package in your C drive. C:\Program Files\MongoDB\Server\4.4\bin.

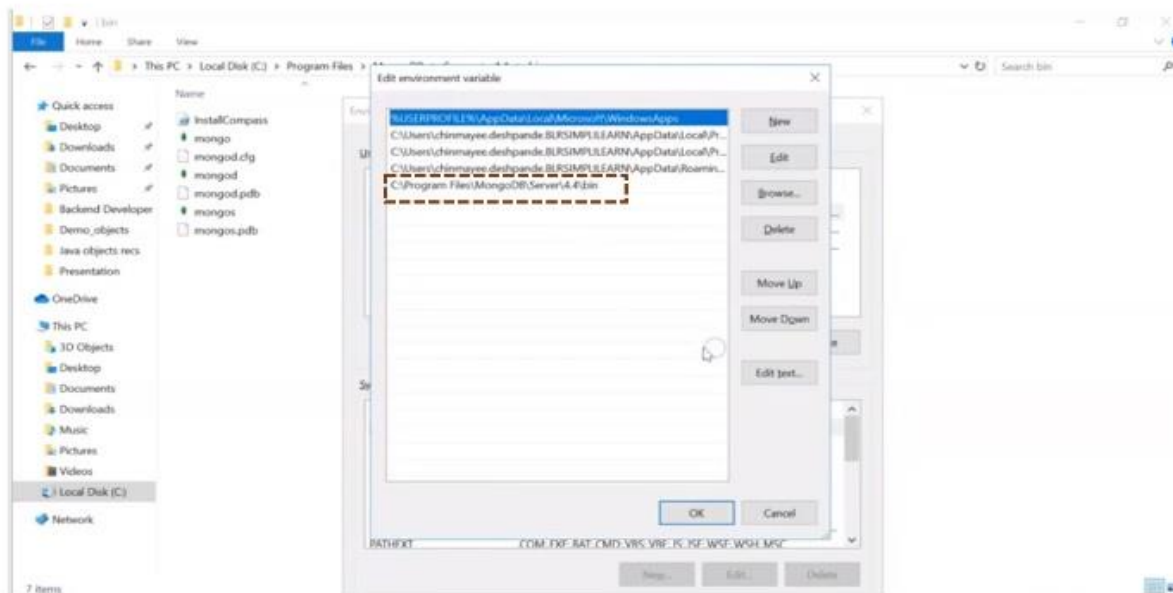


You can see that there are mongo and mongod executable files. The mongod file is the daemon process that does the background jobs like accessing, retrieving, and updating the database.

### Create an Environment Variable:

It's best practice to create an environment variable for the executable file so that you don'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 just type 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: mongod://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongod
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.

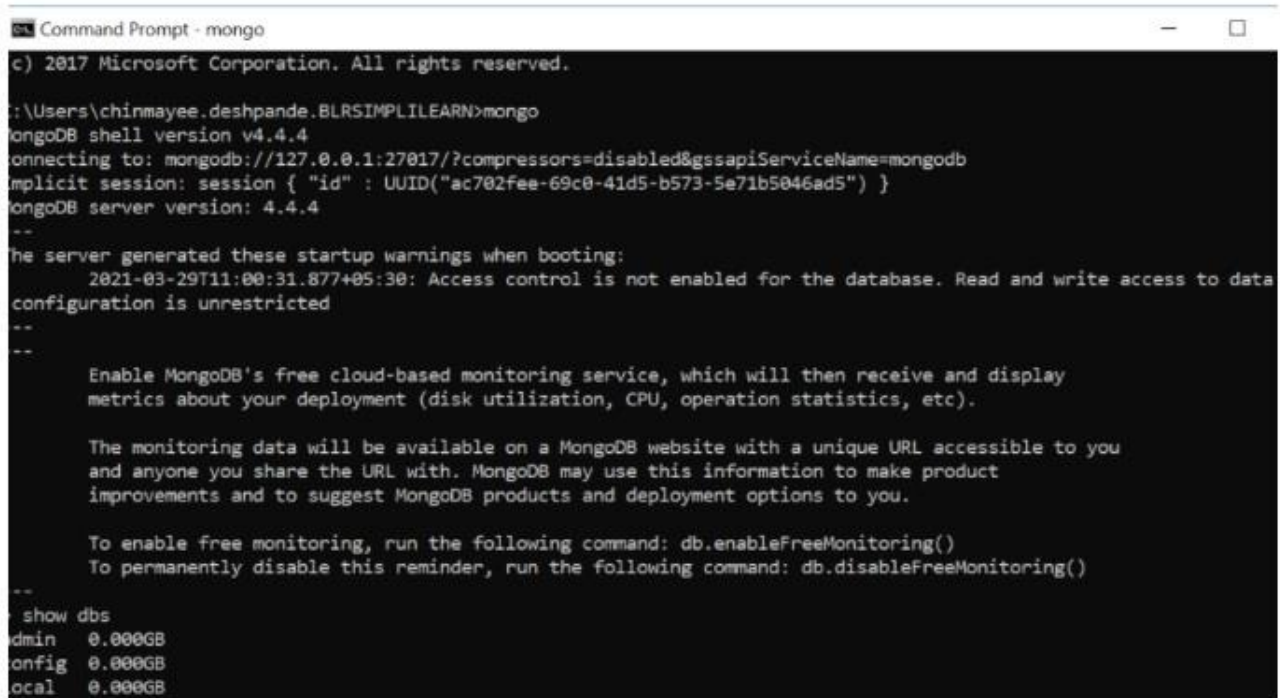
  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB

```

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.



```
Command Prompt - mongo

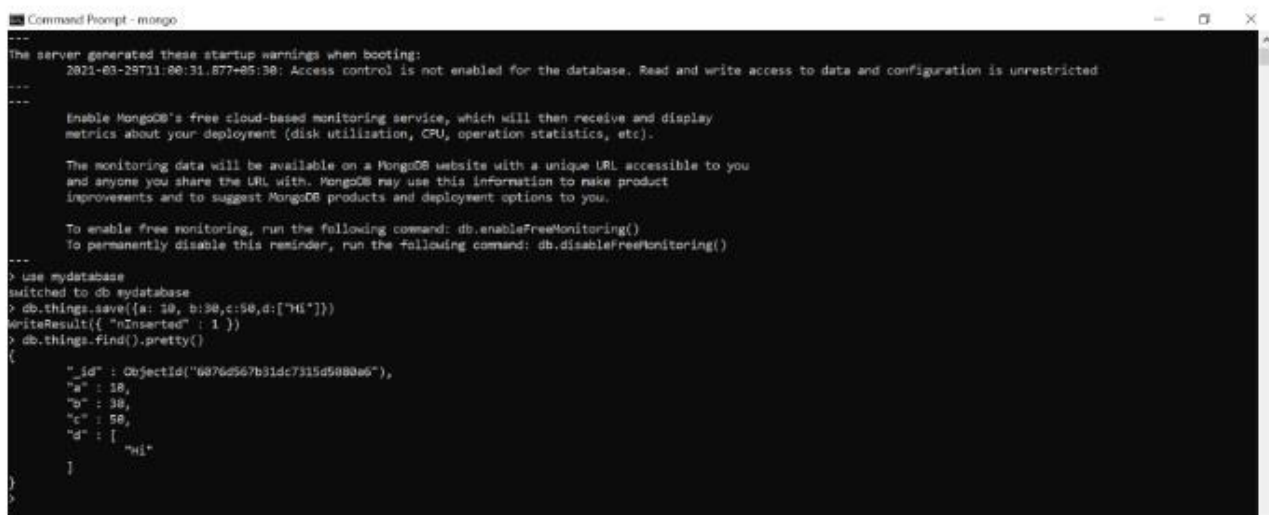
c) 2017 Microsoft Corporation. All rights reserved.

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
```

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



```
Command Prompt - mongo

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 and 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()
--
use mydatabase
switched to db mydatabase
> db.things.save({a: 10, b: 30, c: 50, d: ["Hi"]})
WriteResult({ "nInserted" : 1 })
> db.things.find().pretty()
{
  "_id" : ObjectId("6076d567b31dc7315d5080a6"),
  "a" : 10,
  "b" : 30,
  "c" : 50,
  "d" : [
    "Hi"
  ]
}
```

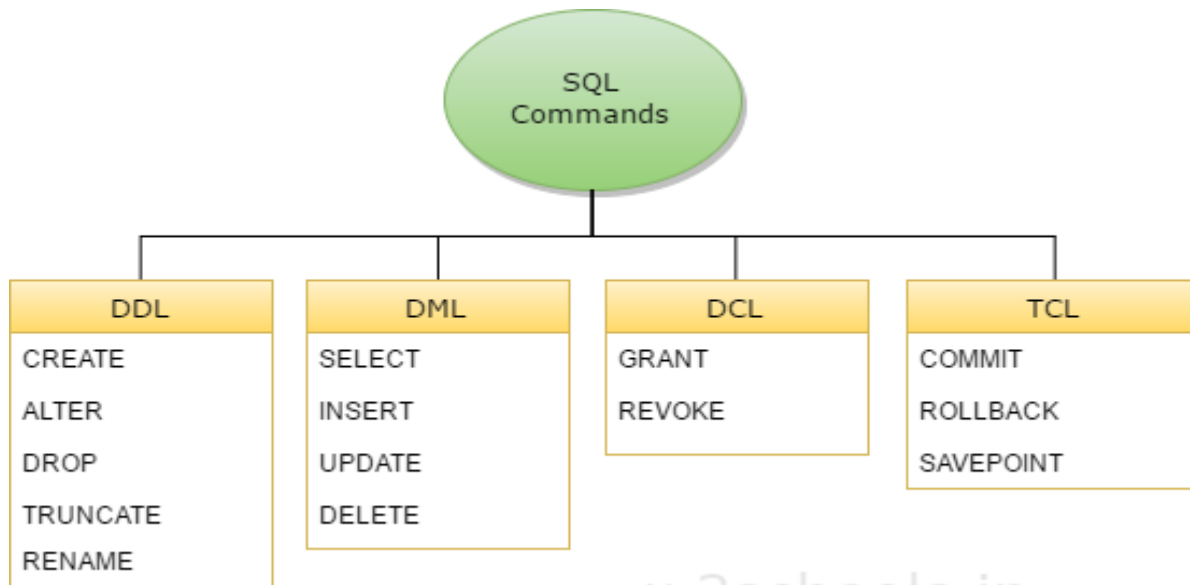
## WEEK-6

### PRACTISING DDL&DMLCOMMANDS

#### DataDefinitionLanguage

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:

- Createtablecommand
- Altertablecommand
- Truncatetablecommand
- Droptablecommand



w3schools.in

## 1. CREATION OF TABLES:

### SQL-CREATETABLE:

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

Syntax: CREATETABLEtablename(column\_nametypeconstraints,...)

```
SQL>CREATETABLESAILORS((SIDint(10)PRIMARYKEY,SNAMEVARCHAR(10),RATINGint(10),AGEint(10));
```

### **Table**

### **Created.Descc**

### **ommand**

TheDESCRIBEcommand is used to view the structure of a table as follows. SQL>DESC

SAILORS;

### **TESTRESULT**

Example1: Create a RESERVE table with fields (SID, BID, DAY) and display using DESCRIBE command.

Example2: Create a BOAT table with fields (BID, BNAME, COLOR) and display using DESCRIBE command

## 2. ALTER TABLE:

### **To ADD a column:**

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

EX: (Write your own Query)

TEST OUTPUT

### **To DROP a column:**

SYNTAX: ALTER TABLE <TABLENAME> DROP COLUMN <COLUMNNAME>;

EX: (Write your own Query)

TEST OUTPUT

### **To MODIFY a column:**

SYNTAX: ALTER TABLE <TABLENAME> MODIFY (<COLUMNNAME>  
<NEW DATATYPE>(<NEWSIZE>));

EX: (Write your own Query)

TEST OUTPUT

Example1:

SQL>ALTER TABLE SAILOR ADD(SNONUMBER(10));

TESTOUTPUT

### 3. **RENAME TABLE**

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

SQL>**RENAME** oldtablename

TO newtablename; EX: (Write your own Query)

TESTOUTPUT

### 4. **TRUNCATE TABLE**

Truncate command is used to delete all records from a table.

SQL>**TRUNCATE TABLE** tablename; EX: (

Write your own Query)

TESTOUTPUT

### 5. **DROP TABLE**

Drop command is used to remove an existing table permanently from database.

SQL>

**DROP TABLE** tablename; EX: (Wri

te your own Query)

TESTOUTPUT



## VIVA QUESTIONS

1. Define data and information.
2. Define Database 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 used drop table command?

## **DMLCOMMANDS**

### 1. ToRetrieve/DisplayDatafromTables:

#### a. SelectcommandisusedtoselectvaluesordatafromtableSYNT

AX

**SELECT\*FROMTABLENAME;**

Example:

SQL>SELECT\*FROMSAILORS;TE

STOUTPUT:

#### b. **The retrieving of specific columns from a table**

SQL>**SELECT**columnname1,columnname2,...columnnamen**FROM**tablename;EX:(Write your own Query)

TESTOUTPUT

#### c. **Elimination of duplicates from the select statement**

SQL>**SELECTDISTINCT**columnname1,columnname2,...columnnamenFROMtablename;

EX:(WriteyourownQuery)

TESTOUTPUT

**d. Selectingadatasetfromtabledata**

SQL>**SELECT**columnname1,columnname2,...columnnamenFROMtablenameWHEREsearc

hcondition;

EX:(WriteyourownQuery)

TESTOUTPUT

Example1:DisplayDataFromRESERVESTable

Example2:DisplayDataFromBOATSTable

## 2. INSERTING DATA INTO TABLE

Insert command is used to insert rows into the table. SYNTAX

AX:

**INSERT INTO** tablename(columnname1,columnname2,...columnnamen)

Example:

SQL>INSERT INTO SAILORS VALUES(22,'DUSTIN',7,45.0);

1 row created

SQL>INSERT INTO SAILORS VALUES(29,'BRUTUS',1,33.0);

1 row created

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

SYNTAX

INSERT INTO tablename(columnname1,columnname2,...columnnamen) VALUES(Value1,Value2,..Value n);

Example:

SQL>INSERT INTO SAILORS(SID,SNAME,RATING,AGE) VALUES(31,'LUBBER',8,55.5);

1 row created

Example 1: INSERT data into RESERVE table: T

OUTPUT:

Example 2: INSERT data into BOAT table: TE

OUTPUT:

## **UPDATE**

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

SQL> **UPDATE** tablename

**SET** column1=expression1, column2=expression2,...

**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> must match the data type.

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

Example 1: **UPDATE SAILORSS**

**SET** S.age=S.age+1, S.rating=S.rating-  
1 **Where** S.sid=34546;

TEST OUTPUT

Example 2: (Write your own Query)

TEST OUTPUT

## **DELETE**

In order to delete rows from a

table we use this command SQL> **DELETE**

**FROM** tablename **WHERE** condition;

Based on the conditions specified the rows get fetched from the table and get deleted in the table. Here the WHERE clause is optional.

Example 1: **DELETE S.AGE FROM SAILORSS where S.Sname='Smith';**

## TESTOUTPUT

Example2:DELETEFROMSAILORS;TE

STOUTPUT

## VIVAQUESTIONS

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?

## **KEYCONSTRAINTS**

Domain Integrity  
constraintsEntity Integrity  
constraintsReferentialIntegrityc  
onstraints

### **1. PRIMARY KEY&NOTNULL**

Example:

```
CREATETABLEsailors(sidinteger,  
                    sname varchar(32)  
                    ,rating integer NOT  
                    NULL,age real,  
                    PRIMARYKEY(sid));
```

Tablecreated.

**TestOutput:**

Example:PracticewithyourownQuery:

TestOutput

### **ImposingICusingALTER**

Example:AlterTableSailorsMODIFYsname varchar(32)NOTNULL;

**TestOutput**

Example:PracticewithyourownQuery:

TestOutput

### **2. DEFAULT**

```
CREATETABLEsailors(sidinteger,  
                    sname varchar(32)  
                    ,rating integer NOT  
                    NULL,age real DEFAULT  
                    25,PRIMARYKEY(sid));
```

Example:PracticewithyourownQuery:

TestOutput

### 3. UNIQUE

```
CREATE TABLE sailors (sid integer,  
                        sname varchar(32) UNIQUE  
                        ,rating integer,  
                        age real DEFAULT  
                        25, PRIMARY  
                        KEY(sid));
```

TestOutput

Example: PracticewithyourownQuery:

TestOutput

### 4. FOREIGNKEY

```
CREATE TABLE reserves ( sid integer not null, bid integer not null, day datetime not null, PRIMARY  
KEY(sid,bid,day), FOREIGN KEY(sid) REFERENCES sailors(sid));
```

Example: PracticewithyourownQuery:

TestOutput

#### Adding Foreign Key to an existing Table

```
Alter table reserves ADD Foreign Key (sid) REFERENCES sailors(sid);
```

Example: PracticewithyourownQuery:

TestOutput



## VIVA QUESTIONS

- 1) Difference between UNIQUE and PRIMARY KEY
- 2) When do you use Composite Primary key?
- 3) Difference between Candidate Key & Primary Key
- 4) What is the Prerequisite for a key to be used as a Foreign Key
- 5) What is a Referential Integrity?
- 6) Give Two practical examples for Referential Integrity?

## **WEEK 7**

### **AIM**

#### **QUERYING**

##### **QUERIES USING ANY, ALL, IN, INTERSECT, UNION**

#### **Objectives:**

Student will able to learn to operate on multiple result sets to return a single result set.  
Student will able to learn to perform nested Queries.

#### **Outcomes:**

Student gains the knowledge to implement queries using ANY, ALL, IN, INTERSECT, UNION.

6. Create the table using following attributes **TICKET** (TICKET\_NO: NUMERIC (9): **PK**, JOURNEY\_DATE: DATE, AGE: INT (4), SEX: CHAR (10): MALE/FEMALE, SOURCE: VARCHAR2 (50), DEP\_TIME: VARCHAR2 (50))

#### **Creating table**

##### **Syntax**

Create table <table name> (col1 datatype,col2 datatype,col3 datatype)

##### **Query**

CREATE TABLE TICKET (TICKET\_NO NUMBER (9) PRIMARY KEY , JOURNEY\_DATE DATE, AGE NUMBER (4), SEX CHAR (10), SOURCE VARCHAR2 (50), DEP\_TIME VARCHAR2 (50))

#### **Describing table**

##### **Query**

Desc TICKET

## **Output**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
<a href="#">TICKET</a>	<a href="#">TICKET_NO</a>	Number	-	9	0	1	-	-	-
	<a href="#">JOURNEY_DATE</a>	Date	7	-	-	-	-	-	-
	<a href="#">AGE</a>	Number	-	4	0	-	-	-	-
	<a href="#">SEX</a>	Char	10	-	-	-	-	-	-
	<a href="#">SOURCE</a>	Varchar2	50	-	-	-	-	-	-
	<a href="#">DEP_TIME</a>	Varchar2	50	-	-	-	-	-	-

### **Inserting records into “TICKET” table**

#### **Syntax**

Insert into <table name> values (val 1,val2,val3)

#### **Query**

insert into TICKET values(1203,'10/FEB/11',19,'MALE','HYDERABAD','10.30 AM');

insert into TICKET values(1213,'10/FEB/11',19,'FEMALE','HYDERABAD','10.30AM');

insert into TICKET values(1201,'13/FEB/11',20,'FEMALE','HYDERABAD','11.30AM');

insert into TICKET values(1202,'14/FEB/11',20,'MALE','TIRUPATHI','11.00 AM'); insert into TICKET values(1205,'14/FEB/11',20,'MALE','HYDERABAD','11.00 AM');

#### **Display table**

##### **Syntax**

Select <select list> from <table name>

##### **Query**

select \* from TICKET;

## Output

TICKET_NO	JOURNEY_DATE	AGE	SEX	SOURCE	DEP_TIME
1203	10-FEB-11	19	MALE	HYDERABAD	10.30 AM
12					
12					
12					
12					

7.Create the table using following attributes **PASSENGER\_TICKETS** (PPNO: VARCHAR2 (15): **PK**, TICKET\_NO: NUMERIC (9))

### Creating table

#### Syntax

Create table <table name> (col1 datatype,col2 datatype,col3 datatype)

#### Query

CREATE TABLE **PASSENGER\_TICKETS** (PPNO VARCHAR2 (15) PRIMARYKEY, TICKET\_NO NUMBER (9))

#### Describing tableQuery

Desc PASSENGER\_TICKETS

#### Output

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comments
<a href="#">PASSENGER_TICKETS</a>	<a href="#">PPNO</a>	Varchar2	15	-	-	1	-	-	-
	<a href="#">TICKET_NO</a>	Number	-	9	0	-	✓	-	-

## **Inserting records into “PASSENGER\_TICKETS” table**

### **Syntax**

Insert into <table name> values (val 1,val2,val3)

### **Query**

```
insert into PASSENGER_TICKETS values(1,1203);insert into
PASSENGER_TICKETS values(2,1213); insert into
PASSENGER_TICKETS values(3,1201); insert into
PASSENGER_TICKETS values(4,1202); insert into
PASSENGER_TICKETS values(5,1205);
```

**Display table**

### **Syntax**

Select <select list> from <table name>

### **Query**

```
select * from TICKET;
```

### **Output**

PPNO	TICKET_NO
1	1203
2	1213
3	1201
4	1202
5	1205

8. Create the table using following attributes RESERVATION (PNR\_NO: NUMERIC (9): FK, JOURNEY\_DATE: DATE, NO\_OF\_SEATS: INT (8), ADDRESS: VARCHAR2 (50), CONTACT\_NO: NUMERIC (10), STATUS: CHAR (3): YES/NO)

→ FOREIGN KEY(PNR\_NO ) REFERENCES PASSENGER\_TICKECTS(PPNO);

### Creating table

#### Syntax

Create table <table name> (col1 datatype,col2 datatype,col3 datatype)

#### Query

create table reservation (pnr\_no varchar2(15),journey\_date date,no\_of\_seats number(8),address varchar2(50),contact\_no number(10),status char(3),foreign key(pnr\_no)references passenger\_tickets(ppno));

### Describing table

#### Query

Desc reservation

#### Output

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVATION	PNR_NO	Varchar2	15	-	-	-	✓	-	-
	JOURNEY_DATE	Date	7	-	-	-	✓	-	-
	NO_OF_SEATS	Number	-	8	0	-	✓	-	-
	ADDRESS	Varchar2	50	-	-	-	✓	-	-
	CONTACT_NO	Number	-	10	0	-	✓	-	-
	STATUS	Char	3	-	-	-	✓	-	-

### Inserting records into “reservation” table

#### Syntax

Insert into <table name> values (val 1,val2,val3)

→ FOREIGN KEY(PNR\_NO ) REFERENCES PASSENGER\_TICKECTS(PPNO);

### Creating table

#### Syntax

Create table <table name> (col1 datatype,col2 datatype,col3 datatype)

#### Query

create table reservation (pnr\_no varchar2(15),journey\_date date,no\_of\_seats number(8),address varchar2(50),contact\_no number(10),status char(3),foreign key(pnr\_no)references passenger\_tickets(ppno));

### Describing table

#### Query

Desc reservation

#### Output

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVATION	PNR_NO	Varchar2	15	-	-	-	✓	-	-
	JOURNEY_DATE	Date	7	-	-	-	✓	-	-
	NO_OF_SEATS	Number	-	8	0	-	✓	-	-
	ADDRESS	Varchar2	50	-	-	-	✓	-	-
	CONTACT_NO	Number	-	10	0	-	✓	-	-
	STATUS	Char	3	-	-	-	✓		



## Inserting records into “reservation” table

### Syntax

Insert into <table name> values (val 1,val2,val3)

### Query

```
insert into reservation values(1,'10/feb/11',5,'amberpet','7416944004','yes'); insert into
reservation values(1,'11/feb/11',8,'amberpet','7416944004','yes'); insert into reservation
values(2,'11/feb/11',8,'b.b nagar','7207204221','yes'); insert into reservation
values(2,'14/feb/11',2,'b.b nagar','7207204221','yes'); insert into reservation
values(3,'14/feb/11',3,'ecil','00000000','yes');
```

```
insert into reservation values(4,'14/feb/11',4,'nagaram','9700135300','yes'); insert into
reservation values(5,'16/feb/11',1,'b.b nagar','8143528258','yes'); insert into reservation
values(5,'15/feb/11',7,'b.b nagar','8143528258','yes');
```

### Display table

### Syntax

Select <select list> from <table name>

### Query

```
select * from reservation;
```

### Output

PNR_NO	JOURNEY_DATE	NO_OF_SEATS	ADDRESS	CONTACT_NO	STATUS
1	10-FEB-11	5	amberpet	7416944004	yes
1	11-FEB-11	8	amberpet	7416944004	yes
2	11-FEB-11	8	b.b nagar	7207204221	yes
2	14-FEB-11	2	b.b nagar	7207204221	yes
3	14-FEB-11	3	ecil	0	yes
4	14-FEB-11	4	nagaram	9700135300	yes
5	16-FEB-11	1	b.b nagar	8143528258	yes
5	15-FEB-11	7	b.b nagar	8143528258	yes

Create the table using following attributes **CANCELLATION** (PNR\_NO: NUMERIC (9): **FK**, JOURNEY\_DATE: DATE, NO\_OF\_SEATS: INT (8), ADDRESS: VARCHAR2 (50), CONTACT\_NO: NUMERIC (9), STATUS: CHAR (3): YES/NO)

➔ FOREIGN KEY(PNR\_NO ) REFERENCES PASSENGER\_TICKECTS(PPNO);

### Creating table

#### Syntax

Create table <table name> (col1 datatype,col2 datatype,col3 datatype)

#### Query

create table cancellation (pnr\_no varchar2(15),journey\_date date,no\_of\_seats number(8),address varchar2(50),contact\_no number(10),status char(3),foreignkey(pnr\_no)references passenger\_tickets(ppno));

### Describing table

#### Query

Desc reservation

#### Output

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
<a href="#">CANCELLATION</a>	<a href="#">PNR_NO</a>	Varchar2	15	-	-	-	✓	-	-
	<a href="#">JOURNEY_DATE</a>	Date	7	-	-	-	✓	-	-
	<a href="#">NO_OF_SEATS</a>	Number	-	8	0	-	✓	-	-
	<a href="#">ADDRESS</a>	Varchar2	50	-	-	-	✓	-	-
	<a href="#">CONTACT_NO</a>	Number	-	10	0	-	✓	-	-
	<a href="#">STATUS</a>	Char	3	-	-	-	✓	-	-

## **Inserting records into “CANCELLATION” table**

### **Syntax**

Insert into <table name> values (val 1,val2,val3)

### **Query**

```
insert into cancellation values(1,'10/feb/11',5,'amberpet','7416944004','yes'); insert into
cancellation values(1,'11/feb/11',8,'amberpet','7416944004','yes'); insert into cancellation
values(2,'11/feb/11',8,'b.b nagar','7207204221','yes'); insert into cancellation
values(2,'14/feb/11',2,'b.b nagar','7207204221','yes'); insert into cancellation
values(3,'14/feb/11',3,'ecil','00000000','yes');
insert into cancellation values(4,'14/feb/11',4,'nagaram','9700135300','yes'); insert into
cancellation values(5,'16/feb/11',1,'b.b nagar','8143528258','yes'); insert into cancellation
values(5,'15/feb/11',7,'b.b nagar','8143528258','yes');
```

### **Display table**

### **Syntax**

Select <select list> from <table name>

### **Query**

```
select * from cancellation
```

### **Output**

PNR_NO	JOURNEY_DATE	NO_OF_SEATS	ADDRESS	CONTACT_NO	STATUS
1	10-FEB-11	5	amberpet	7416944004	yes
1	11-FEB-11	8	amberpet	7416944004	yes
2	11-FEB-11	8	b.b nagar	7207204221	yes
2	14-FEB-11	2	b.b nagar	7207204221	yes
3	14-FEB-11	3	ecil	0	yes

4	14-FEB-11	4	nagaram	9700135300	yes
5	16-FEB-11	1	b.b nagar	8143528258	yes
5	15-FEB-11	7	b.b nagar	8143528258	yes

26. Write a trigger on passenger to display messages ‘1 Record is inserted’, ‘1 record is deleted’, ‘1 record is updated’ when insertion, deletion and updation are done on passenger respectively.

27. Display unique PNR\_NO of all passengers.

#### Query

```
select distinct(pnr_no) from reservation;
```

#### Output

PNR_NO
1
3
5
2
4

28. Display all the names of male passengers.

#### Query

```
select name from passenger where sex='MALE'
```

#### Output

NAME
TIRUMALAY
NAGARAJU
AVS.RAVI

29. Display the ticket numbers and names of all the passengers.

**Query**

```
select p.name,t.ticket_no from passenger p,passenger_tickets t where  
t.ppno=p.ppno
```

NAME	TICKET_NO
TIRUMALAY	1203
SUPRIYA	1213
AMULYA	1201
NAGARAJU	1202
AVS.RAVI	1205

**Output**

30. Find the ticket numbers of the passengers whose name start with 't' and ends with 'y'.

**Query**

```
select t.ticket_no from passenger p,passenger_tickets t where p.name like 'T%Y' and  
t.ppno=p.ppno;
```

**Output**

TICKET_NO
1203

31. Find the names of passengers whose age is between 15 and 20.

**Query**

```
select name from passenger where age between 15 and 20
```

**Output**

NAME
TIRUMALAY
SUPRIYA
AMULYA
NAGARAJU
AVS.RAVI

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

**Query**

```
select name from passenger where name like 'A%';
```

**Output**

NAME
AMULYA
AVS.RAVI

33. Display the sorted list of passengers names.

**Query**

```
select name from passenger order by name;
```

**Output**

NAME
AMULYA
AVS.RAVI
NAGARAJU
SUPRIYA
TIRUMALAY

34. Write a query to display the information present in the PASSENGER and CANCELLATION tables. (Use UNION Operator).

**Query**

```
select * from passenger p,cancellation c where p.ppno=c.pnr_nounion  
select * from passenger p1,cancellation c1 where p1.ppno=c1.pnr_no
```

### Output

PP NO	NAME	AGE	SEX	ADDRESS	PNR_NO	JOURNEY_DATE	NO_OF_SEATS	ADDRESS	CONTACT_NO	STATUS
1	TIRUMALAY	19	MAL E	AMBER PET	1	10-FEB-11	5	amberpet	7416944004	yes
1	TIRUMALAY	19	MAL E	AMBER PET	1	11-FEB-11	8	amberpet	7416944004	yes
2	SUPRIYA	20	FEMALE	B.B NAGAR	2	11-FEB-11	8	b.b nagar	7207204221	yes
2	SUPRIYA	20	FEMALE	B.B NAGAR	2	14-FEB-11	2	b.b nagar	7207204221	yes
3	AMULYA	20	FEMALE	ECIL	3	14-FEB-11	3	ecil	0	yes
	NAGARAJU	20	MAL E	NAGARAM	4	14-FEB-11	4	nagaram	9700135300	yes
5	AVS.RAVI	20	MAL E	B.B NAGAR	5	15-FEB-11	7	b.b nagar	8143528258	yes
5	AVS.RAVI	20	MAL E	B.B NAGAR	5	16-FEB-11	1	b.b nagar	8143528258	yes

35. Display the number of tickets booked for each PNR\_NO using GROUP BY clause. (Use GROUP BY on PNR\_NO).

### Query

```
select pnr_no,sum(no_of_seats) from reservation group by pnr_no;
```

### Output

PNR_NO	SUM(NO_OF_SEATS)
1	13
3	3
5	8
2	10
4	4



36. Find the distinct PNR numbers that are present.

**Query**

Select distinct (pnr\_no) from reservation;

**Output**

PNR_NO
1
3
5
2
4

37. Find the number of tickets booked by a passenger where the number of seats is greater than 5. (Use GROUP BY, WHERE and HAVING clauses).

**Query**

select pnr\_no,sum(no\_of\_seats) from reservation group by pnr\_no having sum(no\_of\_seats) > 5

**Output**

PNR_NO	SUM(NO_OF_SEATS)
1	13
5	8
2	10

38. Find the total number of cancelled seats.

**Query**

select sum(no\_of\_seats) from cancellation;

**Output**

SUM(NO_OF_SEATS)
38

## **VIVA QUESTIONS**

1. What is the syntax for create command?
2. What is the difference between primary key and foreign key?
3. What is the command to display data from a table?
4. What are the types of clause used in mysql ?

## WEEK 8 & 9:

**Querying Using Aggregate functions (COUNT, SUM, AVERAGE using GROUPBY and HAVING) Queries using Aggregate functions (COUNT, SUM, AVG, MAX and MIN), GROUP BY, HAVING and Creation and dropping of Views.**

### Objectives:

Student will be able to learn to perform mathematical operations that return a single value, calculated from values in a column.

### Outcomes:

Student gains the knowledge to perform aggregate operations on the database appropriately.

**Aggregate operators:** In addition to simply retrieving data, we often want to perform some computation or summarization. SQL allows the use of arithmetic expressions. We now consider a powerful class of constructs for computing aggregate values such as MIN and SUM.

**1. Count:** COUNT followed by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (\*) indicates all the tuples of the column.

**Syntax:** COUNT (Column name)

**Example:** SELECT COUNT (Sal) FROM emp;

**2. SUM:** SUM followed by a column name returns the sum of all the values in that column.

**Syntax:** SUM (Column name)

**Example:** SELECT SUM (Sal) From emp;

**3. AVG:** AVG followed by a column name returns the average value of that column values.

**Syntax:** AVG (n1,n2..)

**Example:** Select AVG(10, 15, 30) FROM DUAL;

**4. MAX:** MAX followed by a column name returns the maximum value of that column.

**Syntax:** MAX (Column name)

**Example:** SELECT MAX (Sal) FROM emp; SQL> select

deptno,max(sal) from emp group by deptno;

DEPTNO	MAX(SAL)
--------	----------

----	-----
------	-------

10	5000
----	------

20	3000
----	------

30	2850
----	------

SQL> select deptno,max(sal) from emp group by deptno having max(sal)<3000;DEPTNO

MAX(SAL)
----------

----	-----
------	-------

30	2850
----	------

**5. MIN:** MIN followed by column name returns the minimum value of that column.

**Syntax:** MIN (Column name)

**Example:** SELECT MIN (Sal) FROM emp;

SQL>select deptno,min(sal) from emp group by deptno having min(sal)>1000;DEPTNO

MIN(SAL)
----------

----	-----
------	-------

10	1300
----	------

**VIEW:** In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

A view is a virtual table, which consists of a set of columns from one or more tables. It is similar to a table but it does not store in the database. View is a query stored as an object.

**Syntax:**        CREATE VIEW view\_name AS SELECT set of fields FROM  
                  relation\_name WHERE (Condition)

### **I. Example:**

```
SQL>CREATE VIEW employee AS SELECT empno,ename,job FROM EMPWHERE job =  
      'clerk';
```

View created.

```
SQL> SELECT * FROM EMPLOYEE; EMPNO
```

	ENAME	JOB
---	----	-----
7369	SMITH	CLERK
7876	ADAMS	CLERK
7900	JAMES	CLERK
7934	MILLER	CLERK

## 2.Example:

```
CREATE VIEW [Current Product List] AS SELECT
ProductID,ProductName
FROM Products
WHERE Discontinued=No
```

**DROP VIEW:** This query is used to delete a view , which has been already created.

**Syntax:** DROP VIEW View\_name;

**Example :** SQL> DROP VIEW EMPLOYEE;

View dropped

**Queries using Conversion functions (to\_char, to\_number and to\_date), string functions (Concatenation, lpad, rpad, ltrim, rtrim, lower, upper, initcap, length, substr and instr), date functions (Sysdate, next\_day, add\_months, last\_day, months\_between, least, greatest, trunc, round, to\_char, to\_date)**

**1. Conversion functions:**  
**To\_char:** TO\_CHAR (number) converts n to a value of VARCHAR2 data type, using the optional number format fmt. The value n can be of type NUMBER, BINARY\_FLOAT, or BINARY\_DOUBLE.

SQL>select to\_char(65,'RN')from dual;

**To\_number :** TO\_NUMBER converts expr to a value of NUMBER data type.

SQL> Select to\_number('1234.64') from Dual;1234.64

**To\_date:**TO\_DATE converts char of CHAR, VARCHAR2, NCHAR,or NVARCHAR2 data type to a value of DATE data type.

```
SQL>SELECT TO_DATE('January 15, 1989, 11:00 A.M.')FROM DUAL;TO_DATE('
15-JAN-89
```

-----

## 2. String functions:

**Concat:** CONCAT returns char1 concatenated with char2. Both char1 and char2 can be any of the datatypes

```
SQL>SELECT CONCAT('ORACLE','CORPORATION')FROM DUAL;
```

ORACLECORPORATION

**Lpad:** LPAD returns expr1, left-padded to length n characters with the sequence of characters in expr2.

```
SQL>SELECT LPAD('ORACLE',15,'*')FROM DUAL;*****ORACLE
```

**Rpad:** RPAD returns expr1, right-padded to length n characters with expr2, replicated as many times as necessary.

```
SQL>SELECT RPAD ('ORACLE',15,'*')FROM DUAL;
```

ORACLE\*\*\*\*\*

**Ltrim:** Returns a character expression after removing leading blanks.

```
SQL>SELECT LTRIM('SSMITHSS','S')FROM DUAL;MITHSS
```

**Rtrim:** Returns a character string after truncating all trailing blanks

```
SQL>SELECT RTRIM('SSMITHSS','S')FROM DUAL;SSMITH
```

**Lower:** Returns a character expression after converting uppercase character data to lowercase.

```
SQL>SELECT LOWER('DBMS')FROM DUAL;
```

dbms

**Upper:** Returns a character expression with lowercase character data converted to uppercase

```
SQL>SELECT UPPER('dbms')FROM DUAL;DBMS
```

**Length:** Returns the number of characters, rather than the number of bytes, of the given string expression, excluding trailing blanks.

```
SQL>SELECT LENGTH('DATABASE')FROM DUAL;8
```

**Substr:** Returns part of a character, binary, text, or image expression.

```
SQL>SELECT SUBSTR('ABCDEFGHJIJ'3,4)FROM DUAL;CDEF
```

**Instr:** The INSTR functions search string for substring. The function returns an integer indicating the position of the character in string that is the first character of this occurrence.

```
SQL>SELECT INSTR('CORPORATE FLOOR','OR',3,2)FROM DUAL;14
```

### **3. Date functions:**

**Sysdate:** SQL>SELECT SYSDATE FROM DUAL;29-DEC-08



**next\_day:**

SQL>SELECT NEXT\_DAY(SYSDATE,'WED')FROM DUAL;05-JAN-09

**add\_months:**

SQL>SELECT ADD\_MONTHS(SYSDATE,2)FROM DUAL;28-FEB-09

**last\_day:**

SQL>SELECT LAST\_DAY(SYSDATE)FROM DUAL;31-DEC-08

**months\_between:**

SQL>SELECT MONTHS\_BETWEEN(SYSDATE,HIREDATE)FROM EMP; 4

**Least:**

SQL>SELECT LEAST('10-JAN-07','12-OCT-07')FROM DUAL;10-JAN-07

**Greatest:**

SQL>SELECT GREATEST('10-JAN-07','12-OCT-07')FROM DUAL;10-JAN-07

**Trunc:**

SQL>SELECT TRUNC(SYSDATE,'DAY')FROM DUAL;28-DEC-08

**Round:**

```
SQL>SELECT ROUND(SYSDATE,'DAY')FROM  
      DUAL;28-DEC-08
```

**to\_char:**

```
SQL> select to_char(sysdate, "dd\mm\yy") from  
      dual;24-mar-05.
```

**to\_date:**

```
SQL> select to_date(sysdate, "dd\mm\yy") from  
      dual;24-mar-05.
```

**VIVA QUESTIONS**

1. What are aggregate functions?
2. What is the difference between LPAD and RPAD?
3. Define View?
4. What is the difference between group by and order by clause?

## Week-10

### TRIGGERS

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

- |               |
|---------------|
| BEFORE INSERT |
|---------------|

 –activated before data is inserted into the table.
- |              |
|--------------|
| AFTER INSERT |
|--------------|

 –activated after data is inserted into the table.
- |               |
|---------------|
| BEFORE UPDATE |
|---------------|

 –activated before data in the table is updated.
- |             |
|-------------|
| AFTER UPDAT |
|-------------|

 –activated after data in the table is updated.
- |               |
|---------------|
| BEFORE DELETE |
|---------------|

 –activated before data is removed from the table.
- |              |
|--------------|
| AFTER DELETE |
|--------------|

 –activated after data is removed from the table.

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

The events that fire a trigger include the following:

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

To View list of triggers;

**Show triggers;**

To remove a trigger for Database

**drop trigger trigger\_name;ex:**

**drop trigger ins\_sal;**

Types of Triggers:-

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

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

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

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

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

Sample:

```
CREATE TRIGGER trigger_name      trigger_time
                                trigger_event ON table_name
FOR EACH ROW
BEGIN
...END
;

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

Example:

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

Firing a trigger:

Question: Find the sum of salaries of all employees 1) First

to create a table **emp** with following columns

Field	Type
-------	------

empid	int(11)
ename	varchar(50)
sal	int(11)

WriteaQuery:

**TESTOUTPUT**

2) create variable/parameters **sal** as below at mysql prompt mysql

```
>set @sal=0;
```

3) now create trigger on emp

```
CREATE TRIGGER sal_sum after insert ON  
empFOREACHROWSET @sal=@sal+NEW.sal;
```

**TESTOUTPUT**

4) insert the values into table emp;

```
mysql> insert into emp  
values(1001,'suhaas',10000);mysql> insert into emp  
values(1002,'Dilraj',15000);mysql>insert into emp values(1  
003,'Riyanshi',25000);
```

Note: trigger is fired on after insert 5)  
check values in the table emp;

```
mysql>select * from emp;
```

**TESTOUTPUT**

6) checking value in the parameters sal m

```
ysql>select @sal as TotalSalary;
```

**TESTOUTPUT**

Note: whenever there is insert operation that value in the **sal** variable increases

## VIVA QUESTIONS

1. Define database triggers.
2. List out the uses of database triggers.
3. What are the parts of triggers and its 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 mean by trigger time?
8. Compare before trigger and after trigger.
9. What is the syntax for DROP trigger?
10. List out the some situations to apply before and after triggers.

**PROCEDURES**

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

**Objectives:**

Student will able to learn the features like reusability, maintainability and modularity.  
Student will able to learn to develop procedures and function for various operation.

**Outcomes:**

Student gains the knowledge to implement procedures and function for various operations.

```
CREATE PROCEDURE my Proc()BEGIN
```

```
SELECT COUNT (Tickets) FROM Ticket WHERE age>=40;End;
```

Procedures created

```
SQL>CREATE PROCEDURE myproc(in_customer_id INT)BEGIN
```

```
DECLARE v_id INT;
```

```
DECLARE v_name VARCHAR(30);
```

```
DECLARE c1 CURSOR FOR SELECT stdId,stdFirstnameFROM students
```

```
WHERE stdId=in_customer_id;
```

```
OPEN c1;
```

```
FETCH c1 into v_id, v_name;Close
```

```
c1;
```

```
End;
```

```
/
```

PL/SQL procedure successfully completed.



## PL/SQL

PL/SQL programs are written as lines of text using a specific set of characters:

- Upper- and lower-case letters A..Z and a..z
- Numerals 0..9
- Symbols ()+-\*/<>=!~^;:.'@%, "\$&\_[]{}?[]
- Tabs, spaces, and carriage returns

PL/SQL keywords are not case-sensitive, so lower-case letters are equivalent to corresponding upper-case letters except within string and character literals.

A line of PL/SQL text contains groups of characters known as lexical units:

- Delimiters (simple and compound symbols)
- Identifiers, which include reserved words
- Literals
- Comments

To improve readability, you can separate lexical units by spaces. In fact, you must separate adjacent identifiers by a space or punctuation. The following line is not allowed because the reserved words `END` and `IF` are joined:

```
IF x > y THEN high := x; END IF; -- not allowed, must be END IF
```

You cannot embed spaces inside lexical units except for string literals and comments. For example, the following line is not allowed because the compound symbol for assignment (`:=`) is split:

```
count := count + 1; -- not allowed, must be :=
```

To show structure, you can split lines using carriage returns, and indent lines using spaces or tabs. This formatting makes the first `IF` statement more readable.

```
IF x > y THEN max := x; ELSE max := y; END IF;
```

```
1) WRITEA PROGRAM TO PRINT THE HELLO  
WORLD BEGIN  
DBMS_OUTPUT.PUT_LINE('HELLO WORLD'); EN  
D;
```

**TEST OUTPUT**

```

2) WRITE A PROGRAM TO PRINT EVEN NUMBERS FROM 1 TO 100
DECLARE
  N NUMBER(3) := 0;
BEGIN
  WHILE
    N <= 100 LOOP
    N
    := N + 2; DBMS_OUTPUT.PUT_LINE
    (N); END LOOP;
END;

```

**TEST OUTPUT**

```

3) WRITE A PROGRAM TO ACCEPT A NUMBER AND FIND SUM OF THE DIGITS
DECLARE
  N NUMBER(5) := &N;
  S NUMBER := 0;
  R NUMBER(2) := 0;
BEGIN
  WHILE N
    != 0 LOOP
    R := MOD
    (N, 10); S := S + R;
    N := TRUNC(N / 10);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('SUM OF DIGITS OF GIVEN NUMBER IS ' || S);
END;

```

**TEST OUTPUT**

4) Write a program to accept a number and print it in reverse order

```

DECLARE
RE
N
NUMBER(5):=&N;
REVERSE
NUMBER(5):=0;
R:=0;
BEGIN
N:=&N;
WHILE N>0
LOOP
R:=(N%10);
N:=N/10;
REVERSE:=R*10+REVERSE;
END LOOP;
DBMS_OUTPUT.PUT_LINE('THE REVERSE OF A GIVEN NUMBER IS'||REVERSE);
END;
```

#### TEST OUTPUT

5) Write a program to accept the value of A, B & C and display which is greater

```

DECLARE
A NUMBER(4,2):=&A;
B NUMBER(4,2):=&B;
C NUMBER(4,2):=&C;
BEGIN
IF (A>B AND A>C) THEN
DBMS_OUTPUT.PUT_LINE('A IS GREATER'||A);
ELSIF B>C THEN
DBMS_OUTPUT.PUT_LINE('B IS GREATER'||B);
ELSE
DBMS_OUTPUT.PUT_LINE('C IS GREATER'||C);
END IF;
END;
```

### **VIVAQUESTIONS**

1. What is PL/SQL?
2. What is the basic structure of PL/SQL?
3. How is a process of PL/SQL compiled?
4. Mention what PL/SQL package consists of?
5. What are the benefits of PL/SQL packages?
6. What is the difference between FUNCTION,PROCEDURE AND PACKAGE inPL/SQL
7. Show how functions and procedures are called in a PL/SQL block?
8. What is Stored Procedure?
9. What is difference between Function and Stored Procedure?

## WEEK 13

**DCL (DATA CONTROL LANGUAGE):** Data Control Language statements are used to create roles, permissions, and referential integrity as well it is used to control access to database by securing it. DCL Commands are Grant and Revoke

**GRANT** - gives user's access privileges to database

**REVOKE** -

withdraw access privileges given with the GR

ANT command

### Checking of User Privileges, Grant set cm

**mysql>** create user mrcet\_cse;

Query OK, 0 rows affected (0.30 sec)

\*To Check where is the created user in our database

**mysql>** select user();

**TEST OUTPUT**

\*To Check what are the grants that the location is having/ mysql

**l>** show grants;

**TEST OUTPUT**

\*To Check what are the GRANTS having for created user mysql

**ql>** show grants for mrcet\_cse;

**TEST OUTPUT**

```
mysql>showtables;
```

## TESTOUTPUT

**\*ToFlush(RE-FRESH)theprivilegesmysql>flushprivileges;**  
QueryOK,0rowsaffected(0.08sec)

**\*Explanation:***Tocheckwhereistheuseri.eincaseifwecreateduser(Ex:mrcet\_cse)itwillbedisplayed as “%”. Rootuserisbydefaultsoit will beavailablein “Localhost”*

```
mysql>selecthost,userfrommysql.user;
```

## TESTOUTPUT

## VIVAQUESTIONS

1. WhatareDCLcommands?
2. ListouttheusesofvariousDCLcommands?
3. WhatarethedifferenttypesofCommands inSQL.
4. Whatis thedifferencebetweenTCL &DCLcommands.
5. Whohastheprivilegetoaccess theDCLcommands.

## CASE STUDY 1

Emp(eid:integer,ename:string,age:integer,salary:real)Works(

eid:integer,did:integer,pcttime:integer)

Dept(did:integer,dname:string,budget:real,managerid:integer)

1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?
2. Write the SQL statements required to create the above relations, including appropriate versions of all primary and foreign key integrity constraints.
3. Define the Dept relation in SQL so that every department is guaranteed to have a manager.
4. Write an SQL statement to add 'JohnDoe' as an employee with eid=101, age=32 and salary=15,000.
5. Write an SQL statement to give every employee a 10% raise.
6. Write an SQL statement to delete the 'Toy' department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.

## CASE STUDY 2

Suppliers(sid:integer,sname:string,address:string)Pa

rts(pid: integer, pname: string, color:

string)Catalog(sid: integer, pid: integer, cost:

real)Relational Algebra and Calculus 117

The key fields are underlined, and the domain of each field is listed after the field

name. Thus sid is the key for Suppliers, pid is the key for Parts, and sid and pid together form the key

for Catalog. The Catalog relation lists the prices charged for parts by Suppliers.

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

1. Find the names of suppliers who supply some red part.
2. Find the sids of suppliers who supply some red or green part.
3. Find the number of parts whose name has 5 letters.
4. Find the sids of suppliers who supply at least 3 parts.
5. Find the sids of suppliers who supply every part.



6. Find the sid of suppliers who supply every red part.
7. Find the sid of suppliers who supply every red or green part.
8. Find the sid of suppliers who supply every red part or supply every green part.
9. Find pairs of sid such that the supplier with the first sid charges more for some part than the supplier with the second sid.
10. Find the pid of parts that are supplied by at least two different suppliers.
11. Find the sid of supplier who supply costliest part.
12. Find the pid of parts supplied by every supplier at less than \$200. (If any supplier either does not supply the part or charges more than \$200 for it, the part is not selected.)

### CASE STUDY 3

Consider the following relations containing airline flight information: Flights

Flights(fno: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 IDs of employees who make the highest salary.
7. Find the IDs of employees who make the second highest salary.
8. Find the IDs of pilots who are certified for the largest number of aircraft.
9. Find the IDs of employees who are certified for exactly three aircraft.
10. Find the total amount paid to employees as salaries.
11. Is there a sequence of flights from Madison to Timbuktu? Each flight in the sequence is required to depart from the city that is the destination of the previous flight; the first flight must leave Madison, the last flight must reach Timbuktu, and there is no restriction on the number of intermediate flights. Your query must determine whether a sequence