Geethanjali College of Engineering and Technology

Cheeryal (V), Keesara (M), Ranga Reddy District – 501 301 (T.S)

DATABASE MANAGEMENT SYSTEMS

LABORATORY MANUAL

DEPARTMENT OF

COMPUTER SCIENCE & ENGINEERING

Lab In Charge

G.SWAPNA
Asst. Prof.

HOD-CSE

Dr. Nagender Kumar Suryadevara
Professor.

Geethanjali College of Engineering and Technology

Department of COMPUTER SCIENCE & ENGINEERING

(Name of the Lab Course) : OPERATING SYSTEMS

(JNTU CODE): **Programme : UG**

Branch: **CSE A,B, C & D** **Version No : 2**

Year: **II** **Updated on : 3/6/2015**

Semester: **I** **No.of pages : 85**

Classification status (Unrestricted / Restricted)

Distribution List : Department, Lab, Library, Lab In Charge

Prepared by :	**Modified by :**
1) Name : Madhuri Agarwal Gupta	**1) Name : Dr. Nagender Kumar Suryadevara**
2) Sign :	**2) Sign :**
3) Design : Asst. Prof.	**3) Design: Professor**
4) Date :	**4) Date :**

Verified by :	*** For Q.C Only.**
1) Name : Dr. Nagender Kumar Suryadevara	**1) Name :**
2) Sign :	**2) Sign :**
3) Design : Professor	**3) Design :**
4) Date :	**4) Date :**

Approved by : (HOD)

1) Name :

2) Sign :

3) Date :

List of Lab Exercises

S.No.	Name of the programs	Page No.
1	**E-R Model:** Analyze the problem with the entities which identify data persisted in the database which contains entities, attributes.	
2	**Concept design with E-R Model:** Apply cardinalities for each relationship, identify strong entities and weak entities for relationships like generalization, aggregation, specialization.	
3	**Relation Model:** Represent attributes as columns in tables and different types of attributes like Composite, Multi-valued, and Derived.	
4	**Normalization**	
5	**Installation of MySql and practicing DDL commands.**	
6	**Practicing DML commands** SELECT, INSERT, UPDATE, DELETE.	
7	**Querying** Queries using ANY, ALL, IN, INTERSECT, UNION	
8	**Querying** Using aggregate functions COUNT, SUM using GROUPBY and HAVING.	
9	**Querying** Using aggregate functions AVERAGE using GROUPBY and HAVING	
10	**Procedures** Creation, Execution and Modification of stored Procedure	
11	**Cursors** A cursor on the given data	

ADDITIONAL PROGRAMS

Vision of the Department

To produce globally competent and socially responsible computer science engineers contributing to the advancement of engineering and technology which involves creativity and innovation by providing excellent learning environment with world class facilities.

Mission of the Department

1. To be a center of excellence in instruction, innovation in research and scholarship, and service to the stake holders, the profession, and the public.

2. To prepare graduates to enter a rapidly changing field as a competent computer science engineer.

3. To prepare graduate capable in all phases of software development, possess a firm understanding of hardware technologies, have the strong mathematical background necessary for scientific computing, and be sufficiently well versed in general theory to allow growth within the discipline as it advances.

4. To prepare graduates to assume leadership roles by possessing good communication skills, the ability to work effectively as team members, and an appreciation for their social and ethical responsibility in a global setting.

PEOs and POs

PROGRAM EDUCATIONAL OBJECTIVES

1. To provide graduates with a good foundation in mathematics, sciences and engineering fundamentals required to solve engineering problems that will facilitate them to find employment in industry and / or to pursue postgraduate studies with an appreciation for lifelong learning.

2. To provide graduates with analytical and problem solving skills to design algorithms, other hardware / software systems, and inculcate professional ethics, inter-personal skills to work in a multi-cultural team.

3. To facilitate graduates to get familiarized with the art software / hardware tools, imbibing creativity and innovation that would enable them to develop cutting-edge technologies of multi-disciplinary nature for societal development.

PROGRAM OUTCOMES

1. An ability to apply knowledge of mathematics, science and engineering to develop and analyze computing systems.
2. an ability to analyze a problem and identify and define the computing requirements appropriate for its solution under given constraints.
3. An ability to perform experiments to analyze and interpret data for different applications.
4. An ability to design, implement and evaluate computer-based systems, processes, components or programs to meet desired needs within realistic constraints of time and space.
5. An ability to use current techniques, skills and modern engineering tools necessary to practice as a CSE professional.
6. An ability to recognize the importance of professional, ethical, legal, security and social issues and addressing these issues as a professional.
7. An ability to analyze the local and global impact of systems /processes /applications /technologies on individuals, organizations, society and environment.
8. An ability to function in multidisciplinary teams.
9. An ability to communicate effectively with a range of audiences.
10. Demonstrate knowledge and understanding of the engineering, management and economic principles and apply them to manage projects as a member and leader in a team.
11. A recognition of the need for and an ability to engage in life-long learning and continuing professional development
12. Knowledge of contemporary issues.
13. An ability to apply design and development principles in producing software systems of varying complexity using various project management tools.
14. An ability to identify, formulate and solve innovative engineering problems.

Mapping of Lab Course with Programme Educational Objectives

S.No	Course component	code	course	Semester	PEO 1	PEO 2	PEO 3
1	Professional core		DBMS	II	√	√	

Mapping of Lab Course outcomes with Programme outcomes:

Pos		1	2	3	4	5	6	7	8	9	10	11	12	13	14	
DBMS	**Exp no.**															
CO1: Student gains the ability to Construct E-R diagrams.	**1**		H	**H**	L						H			H	L	
CO 2: Student gains the ability to describe the data requirements for a new information system in a direct and easy to understand graphical notation.	**2,3**		H	H	L						H			H	L	
CO 3: Student gains the ability to implement Notation to Describe the Relational Schema and to Represent an ER Model as a Relational Model.	**2,3**		H	H	L						H			H	L	
CO 4: Students are able to build the database that does not have redundant data.	**4**		L	L								H	L		L	
CO 5: Students have the ability to Categorize the main	**5,6,7,8**		L		H	H					H			L	H	Professional core

database objects, Review and create the table structure with constraints.																
CO 6: Students are able to perform transactions like updating, deleting, inserting and selecting data from a data base.	**6,7,8**				**H**	**H**					H			L	H	
CO 7: Student are able to implement queries using ANY, ALL, IN, INTERSECT, UNION and aggregate operators.	**6,7,8,9**		L	L		H					H			L	H	
CO 8: Students have the ability to change database manager from a passive system to an active one.	**10**										L		L	H	H	
CO 9: Students are able to implement procedures , cursors and triggers.	**10,11,12**			L							L			H	H	

DBMS
Lab Manual

INTRODUCTION

What is Oracle?

Oracle is the name of the database management system that comes from Oracle Corporation.

Oracle9i is the latest product released by Oracle Corporation. Unlike Oracle8i, which is only a database management system, Oracle9i is a collection of following software:

- Oracle9i Application Server – Oracle9iAS
- Oracle9i Database Server – Oracle9iDB
- Oracle9i Developer Suite – Oracle9iDS

In simple words Oracle9i is a platform and not a simple database management system. Oracle9iDB is the database management system that is used to store and access data. Oracle is by far the most widely used relational database management system (RDBMS).

Oracle Corporation is second largest software company next to Microsoft. Oracle Corporation has been targeting Internet programming with the caption - software powers the internet.

This book is about Oracle Database Server. It doesn't discuss about other products in Oracle9i.

Oracle Corporation is also into Enterprise Resource Planning (ERP). It has Oracle Applications that includes Oracle Financials etc.

Oracle Database Server

Oracle database server is one of the databases that are widely used in client/server computing as back-end. Front-end programs that are written using application development tools such as Visual basic access Oracle and submit SQL commands for execution. Oracle8i onwards oracle is trying to provide extra facilities that are required to be an internet database.

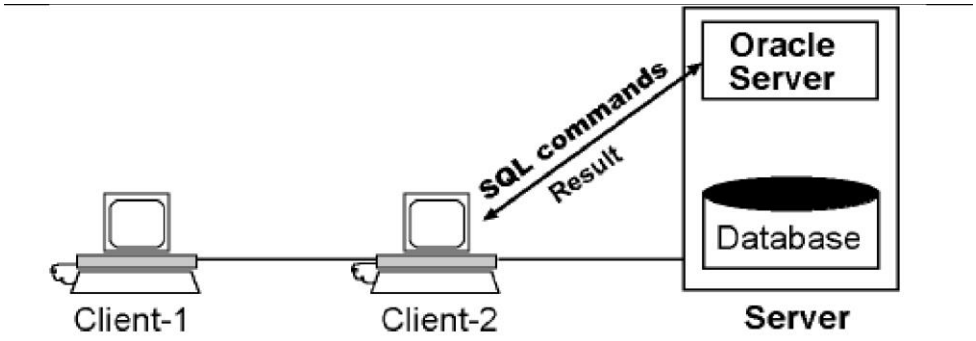


Figure 1: Oracle Server as Server in Client/Server computing model.

Oracle8i provides special features to support various types of data that is to be stored in web sites. Oracle supports both OLTP (online transaction processing) applications as well data warehouse applications, which contain a very large database (VLDB).

One of the biggest advantages of Oracle has been its presence on around 100 different platforms. Oracle is quite scalable, which means it can scale up and down very easily as the requirements change.

Oracle also provides Java Virtual Machine (JVM) as part of database. This enables oracle to run java programs. In fact, starting from Oracle8i, oracle can run programs written either in PL/SQL or Java.

Oracle Instance

Oracle instance is a collection of memory structures and processed that are used to manage oracle database. Each oracle database is to be accessed by one or more Oracle instances. If two or more instances are accessing the same database, it is called as parallel server architecture. In order to start using an oracle database, we must first start Oracle instance. Oracle instance will them open the database and make it available to users. It is beyond the scope of this book to discuss what Oracle instance actually contains. Please read "Oracle Concepts" manual for complete information about oracle instance. In nutshell every oracle installation contains at least one Oracle Instance and one oracle database.

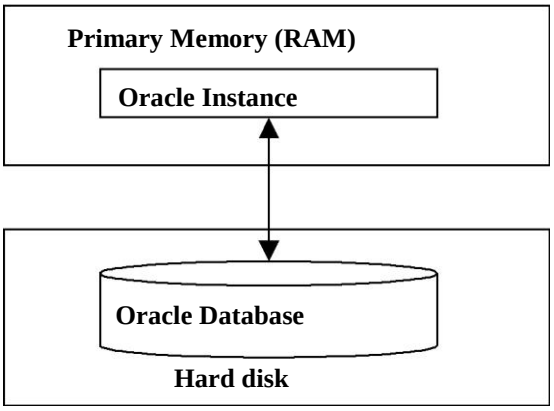


Figure 2: Oracle Instance and Oracle Database.

What Is Personal Oracle?

Personal Oracle is one of the flavors of Oracle. This is not a product that is used by production system (systems where real data is stored). This is more like a learning tool. It runs on desktop PCs. In personal oracle, oracle instance, oracle database and client application all run on the same machine (see figure 3). Whereas in Oracle database server, only oracle instance and database reside on the server and client applications run on clients.

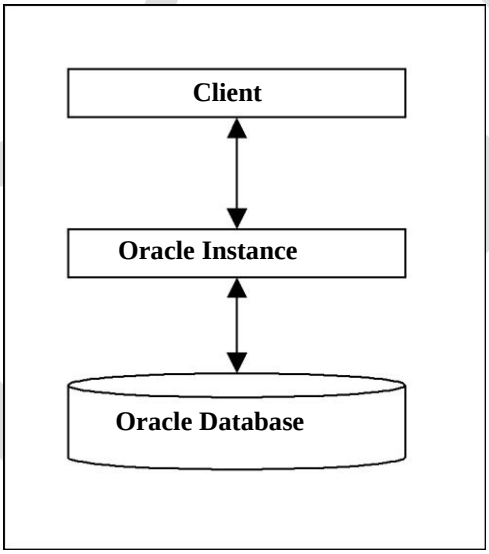


Figure 3: Personal Oracle.

It is also possible to develop an applications using Personal Oracle on you desktop/laptop and deploy them in a client/server environment.

Starting up Database

Before we access oracle database, we must start oracle database. Starting up oracle database means starting oracle instance and associating oracle instance with an oracle database so that oracle instance can access the database. The process is very length and complicated. Several steps are involved in it. But fortunately we do not have to know all that happens when a database starts. We just need to select an option or two to startup database. Generally you do not have to startup database in case of Oracle Server running on Windows NT/Windows 2000 as oracle server automatically starts in this case. However, if you ever have to start oracle database on Windows NT/Windows 2000, follow the steps given below:

1. Start services program using Administrative Tools -> Service in Windows/2000 or Control Panel -> Service on Windows NT.
2. If service OracleServiceOracle8i has not yet started, click on it with right button and select start option from popup menu.

   The exact name of the service depends on the name you have given to oracle instance at the time of installing it.

Note: Starting and shutting down the database is the job of Database Administrator. As this books assumes that you are an application developer, it doesn't get into those details.

Starting up database in Personal Oracle

Unlike Oracle Server in Personal Oracle, Oracle Instance doesn't start on its own. The Oracle Instance must be explicitly started. The following are the steps to start oracle on Personal Oracle:

1. Select start database option in Personal Oracle8i for windows menu.
2. When a dialog box is displayed wait until the message Oracle Instance Started appears.
3. Click on Close button to close the dialog box.

Starting SQL*PLUS

Sql*plus is a tool that comes along with Oracle. It is used to issue SQL and SQL*PLUS commands. It provides command line interface through which we can enter SQL and SQL*PLUS command.

To start SQL*PLUS, take the steps given below:

1. Select start->programs->Oracle - Oracle8i.
   Oracle8i is the name of the instance. It may be different on your system.
2. Then select Application Development -> SQL Plus.
3. When Log On dialog box is displayed, enter username, password and Host string. Use tab key to move from one field to another. For more information about each of these fields, see next section.
4. Click on OK.
5. If the information supplied is valid then you enter into Oracle and SQL*PLUS will display SQL> prompt.

Username, Password and Host String

Oracle is a multi-user database. Whoever is access the database must log on to database? To log on we have to supply username and password. When the given username and password are recognized by Oracle, it will allow us to access data. A user can access only the data that belongs to his/her and not the data of others. However, it is possible for a user to grant privileges to others so that other can access his/her data. Creation of users and management of overall security is the responsibility of Database Administrator (DBA). DBA is the person who makes sure that database is functioning smoothly. He is responsible for operations such as taking backup of the database, recovering the database in the event of failure, fine tuning database to get best performance. So, if you want to have a new account under your name, please consult administrator of your database.

Username & Password

Every user who wants to access oracle database must have an account in the database. These accounts are created by DBA. Each account is associated with username and password.
Oracle comes with a set of predefined accounts. The following are the usernames and passwords of these accounts.

Username	Password
system	manager

sys	change_on_install
Scott	tiger
Demo	demo

Note: when you enter into oracle using either system or sys then you become DBA. That means you get special privileges to perform major operations such as creating users etc.

Host String

Host string is a name that is used to access oracle server that is running on a different machine from client. This is required only when you are trying to access oracle server that is not on the current machine. That means, you never need to use host string for Personal Oracle as client and oracle always run on the same machine in Personal Oracle. Host string is required when you are trying to connect to Oracle Server running on remote machine. Host string is actually called as net service name. Net service name is a name that is stored in TNSNAMES.ORA file on the client to provide the following information.

Host	Name of the machine or IP address of the machine on Which oracle server is running?
Instance	Name of the Oracle Instance running on the remote machine.
Port Number	Port number of the listener, a program that takes Requests from clients. Port number is an integer that Uniquely identifies the program on the server.

How to enter SQL statements?

SQL*PLUS allow to types of command to entered at the prompt - SQL and SQL*PLUS.
SQL commands include commands of ANSI/ISO SQL and extra commands added to ANSI SQL by oracle.
The following are the rules to be followed while entering SQL commands.

1. An SQL statement may be entered in multiple lines.
2. It is not possible to break a word across lines.
3. SQL statement must be terminated by semicolon (;).

The following is an example of SQL command. What this command does is not important at this moment.

```
SQL>  select  ccode, name
  2   from  courses
  3   where  fee  >  5000;
```

In the above command, we entered the command in three lines. When you enter semicolon and press enter key then SQL*PLUS will take it as the end of the command. Also note that you have to press enter key at the end of each line.

Note: Both SQL and SQL*PLUS commands are NOT case sensitive.

How to enter SQL*PLUS statements?

SQL*Plus statements are available only in SQL*PLUS. They are not part of standard SQL. SQL*Plus commands are mainly used for two purposes – editing SQL commands and formatting result of query.

The following rules are to be followed while entering these commands.

1. The entire command must be entered on a single line.
2. No need to terminate command with semicolon (;).
3. Commands can be abbreviated. However, the amount of abbreviation is not fixed. Some commands are abbreviated to one letter some are abbreviated to 2 and so on.

The following example show how to use CLEAR SCREEN command of SQL*PLUS.

```
SQL>clear  screen
```

Or it can be abbreviated to

```
SQL>clrscr
```

Common Errors

The following are the common errors that you get while you are trying to log on to Oracle.

Ora-01017: invalid username/password; login denied

WEEK 1

AIM

E-R Model

Analyze the problem with the entities which identify data persisted in the database which contains entities and attributes.

Objectives:

Student will able to learn the Entity-Relationship(ER) modeling to develop a conceptual model of data.

Outcomes:

Student gains the ability

- About business rules, notations and constructs.
- To Construct E-R diagrams including:
  - Entities (strong, weak, associative)
  - Attributes (simple, multi-valued, derived)
  - Relations (unary, binary, ternary)

Roadway Travels System

Requirements Analysis

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.

The company wants to computerize its operations in the following areas:

1. Reservations and Ticketing
2. 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.

Following steps are involved in the process:

1. Analyzing the problem and identifying the Entities and Relationships
2. E-R Model
3. Normalised Relational Model
4. Creating the database
5. Querying.
6. Triggers and Stored procedures on the tables.

1) E-R Model:

Analyze the problem carefully and come up with the entities in it. Identify what data has to be persisted in the database. This contains the entities, attributes etc. In this we wqill analyze different types of entities with attributes of "Roadways Travels".

Entity: An Entity is an object or concept about which you want to store information

Relationship: A relationship is an association between several entities.

Attributes: An object is characterized by its properties or attributes. In relational database systems attributes correspond to fields.

The Road Way Travels Consists Of The Following Entities:

- ✓ BUS
- ✓ Ticket
- ✓ Passenger
- ✓ Reservation
- ✓ Cancellation/modification

These Entities have following attributes

Bus:

- Bus_id
- Bus_name
- Bus_type
- Bus_totalseats

Ticket:

- Ticket_booking
- Ticket_datejourney
- Ticket_to
- Ticket_from
- Ticket_id
- Ticket_no of tickets

Passenger:

- Pid
- Pname
- Pgender

- Page
- precancel

VIVA QUESTIONS

1. Explain entity, relation and attributes?
2. What is the difference between weak entity and strong entity?
3. What are the different types of attributes?
4. Explain difference between multi valued and single valued attributes?
5. What is the difference between entity and entity set?

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.

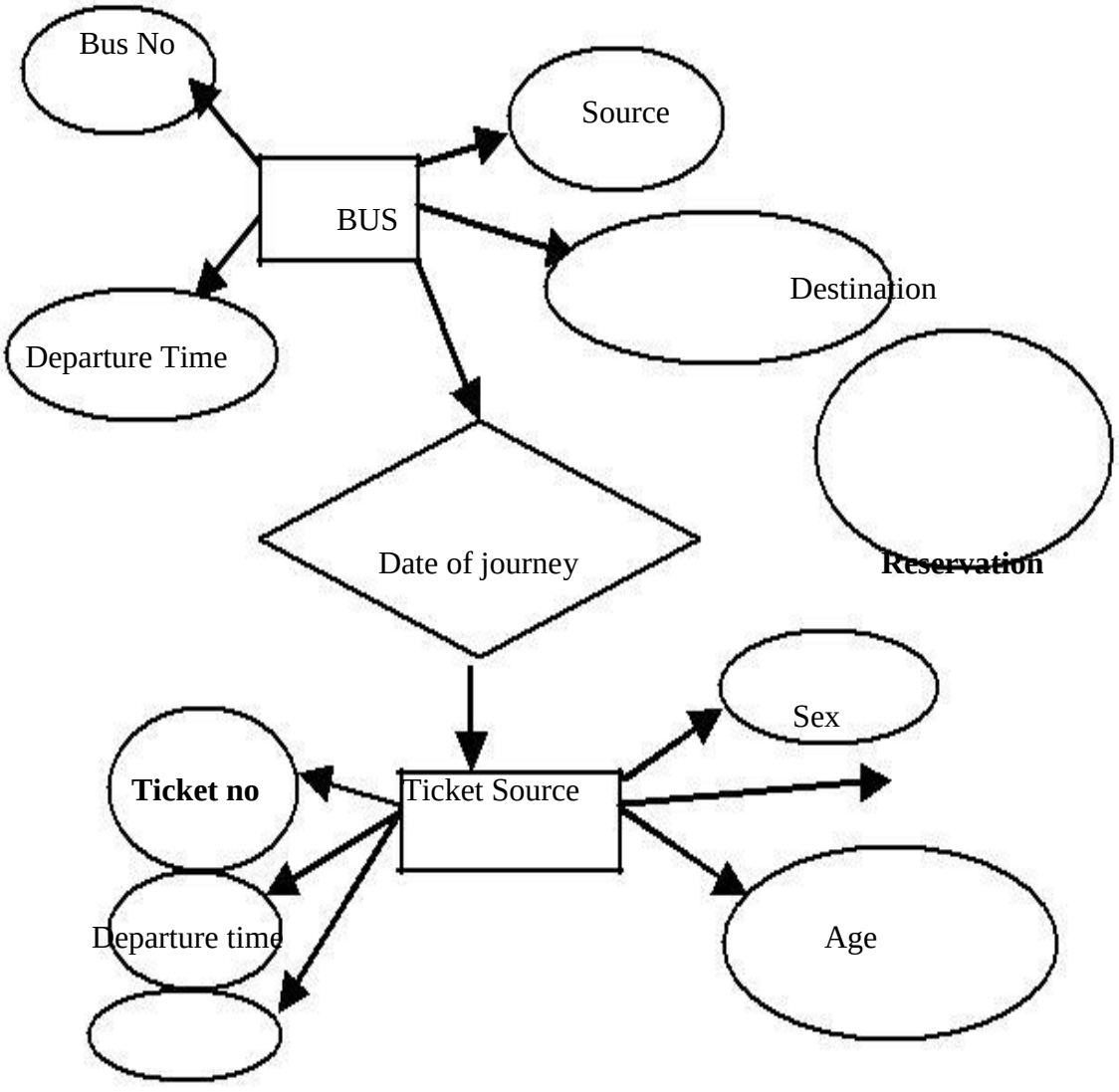
Objectives:

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:

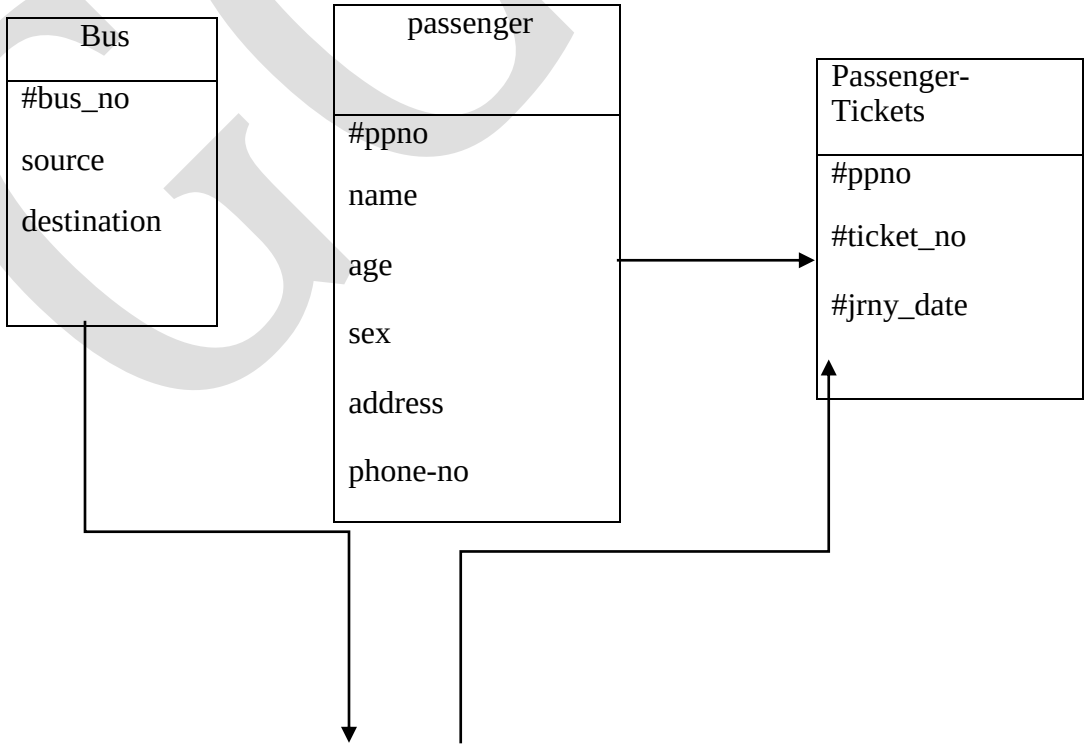
Student will able to learn the structural components of the relational data model.
Student will able to learn to map ER models into relational models.

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	Passport ID

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 week 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. W e 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 posses sufficient attributes to form a primary ket 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 tablet). 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 INF) through five(fifth normal form of SNF). In practical applications, you'll often see INF, 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 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
	r. ID	**Date**	**Amount**
Wilkinson	12898	14-Oct-2003	−21
	Tr. ID	**Date**	**Amount**
Stevens	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:

Customer	Tr. ID	Date	Amount
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

AIM

INSTALLATION OF MYSQL and Practicing DDL commands

Objectives:

Student will able to learn DDL Statements to Create and Manage Tables.

Outcomes:

Student gains the ability to
• Categorize the main database objects
• Review the table structure
• List the data types that are available for columns
• Create a simple table
• Describe how constraints are created at the time of table creation
• Describe how schema objects work.

Requirements:

MYSQL, software for database that is "MYSQL.exe" required space for installing MYSQL server.

Description: In this we will install the MYSQL server and also practice the "DDL" commands.

Installation of MySQL and Practicing DDL commands:

Installation of MySQL:

Why MySQl: MySQL is undoubtedly the most popular and widely-used open source database:

- It is simple to setup and use.
- It is recognised as one of the fastest databse engines.
- Most linux(and many Windows-based)web hosts offer MySQL.
- MySQL is closely integrated with PHP, which makes it an ideal cndidate for many web applications.

Why Install MySQL Locally?

Installing MySQL on your development PC allows you to safely create and test a web application without affecting the data or systems on your live website.

Installing MySQL on Windows: In this section you will learn how to install MySQL 5.0 on windows system the MySQl 3.21 was first version for the windows. Windows installer of MySQL includes auto installer with configuration Wizard that support for easy installation.

PRACTICING DDL COMMANDS:

Creating a table:

To store data into the database we must create a table with same structure as the data we have to store into the database. To create a table use the create table Command that has the following syntax.

Syntax: SQL>Create table <tablE name> (<colname>< datatype>[constraints],<colname> [constraints]……. colname>< datatype>[constraints],[table level constraints])

student table:

Sid	int	pK
Sname	Varchar(30)	Notnull
Course	Varchar(30)	default 'cpp'

Query for creating Student Table:

Create table student (Sid int constraint sid_PK Primary Key, Sname varchar(30) not null. Course varchar(30) constraint default-course default'CPP')

Altering Table structure:

After creating table we may want to add or remove columns and constraints or change the data type of a column to perform all these operations, we have to use alter table that has the following syntax,

1. **Alter column:** this option is used to change the datatype of a column or add or remove not null constraint.

Syntax: Alter Tabl<tablEname>AlterColumn<colname><datatype>Null/NotNull

Q: Sp_help'student' //Execute

EX: The following example changes the data types of sname column of student table to char with maximum length 40 removes the not Null Column constraint available on it.

Q: Alter Table Student Alter Column Sname Char (40) Null

Q: sp_help'student //Execute

3. **Drop Column**: this option is used to delete columns from the table

Syntax: Alter Table<tablEName>Drop Column<Column List>

Ex: The following example deletes the columns Fname and address from the table student.

Q: Alter table Drop Column Fname, Address.

VIVA QUESTIONS

1. Write the syntax for all DDL commands?
2. What is the difference between drop and truncate command?
3. What is DDL interpreter?
4. What is the command to delete columns from table?

WEEK 6

AIM

Practicing DML commands

Objectives:

Student will able to learn commands that make changes in relational database and transaction management.

Outcomes:

Student gains the knowledge to perform transactions like updating, deleting, inserting and selecting data from a data base.

SELECT INSERT,UPDATE,DELETE.

5. Data Manipulation Language commands

1**) Inserting rows into the table:** To insert rows into the table we have to use **insert Command** that has the following syntax

Syntax: insert [into] <tablename>values ([values>)

→While using thei syntax of insert command we must provide value for every column. Whenever we want to insert null into a column then use the keyword **null** and to insert default value for a column use the keyword **default.**

Ex: The following examples insert rows into the table **student**

Insert student Values (1001,'A','Dotnet')

Insert student Values (1002,'B','Defaultt)

Insert student Values (1003,'C','SQL')

Select * from student //Execute this line

Ex: The following examples insert rows into the table **marks**

Insert Marks Values (1001, 77, 54, 67, null, null, null)

Insert Marks Values (1002, 70, 45, 56, null, null, null)

Insert Marks Values (1005, 78, 65, 89, null, null, null)

Select * from marks //select this line and execute

2) Creating a table from another table: When we have to create a new table from existing table then use the following Syntax of Select statement.

Syntax: Select*|<column list>into<new table name>from <old table name>[where<condition>]

Ex: the following example creates a table with name **student3** from the table student.

Q: Select * into student3 from student //Execute

3) Updating Rows in the table: When we have to modify the existing data in the table then we have to use **update Command** that has the following syntax

Syntax: Update <tablEname> Set<column>=<value>,<colname>=<value>…[Where<condition>]

Ex: The following examples updates name and course of the student with id 1004 to **S** and **DOTNET**

Q: Update student set name-'S', course=''DOTNET' where sid=1004

Q: Select * from student

EX: The following example updates marks table by calculating total and average for all rows.

Q: update marks set total=C+CPP+SQL,aveg=(C+CPP+SQL)/3.0

Q: Select * from marks

Ex:the following example updates marks table by calculating the grade on average marks.

Q: Update marks set grade=

Case

When Aveg>=70 then 'distinction'

When **Aveg**>=60 and Aveg<70 then 'first class'

When Aveg>=50 and Aveg<60 then'Second class'

When Aveg>=35 And Aveg<50 then'third class'

Else'Fail'

End

4) Deleting Rows from the table: To delete rows from table use the delete Command that has the following syntax:

Syntax: Delete Rows from Table Delete[from]<tblEname>[where<condition>]

Ex: The following example deletes the student record with si4-1004 from marks table

Q: Delete marks where sid=1004

Q: Select * from marks

EX: The following example deletes all rows from the marks.

Q: Delete marks

Output

RATING	MINAGE
8	25.5
7	35
3	25.5

4. Create the table using following attributes **BUS (**BUSNO: VARCHAR2 (10): **PK**, SOURCE: VARCHAR2 (50), DESTINATION: VARCHAR2 (50)**)**

Creating table

Syntax

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

Query

Create table create table BUS (BUSNO VARCHAR2(10) Primary Key, SOURCE VARCHAR2 (50), DESTINATION VARCHAR2 (50));

Describing table

Query

Desc bus

Output

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
BUS	BUSNO	Varchar2	10	-	-	1	-	-	-
	SOURCE	Varchar2	50	-	-	-	✓	-	-
	DESTINATION	Varchar2	50	-	-	-	✓	-	-

1 - 3

Inserting records into "Bus" table

Syntax

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

Query

insert into bus values(1234,'hyderabad','tirupathi');

insert into bus values(2345,'hyderabad','tirupathi');

insert into bus values(23,'hyderabad','kolkata');

insert into bus values(45,' tirupathi ','banglore');

insert into bus values(34,'hyderabad','chennai');

Display table

Syntax

Select <select list> from <table name>

Query

Select * from bus;

Output

BUSNO	SOURCE	DESTINATION
1234	hyderabad	tirupathi
2345	hyderabad	tirupathi
23	hyderabad	kolkata

45	tirupathi	banglore
34	hyderabad	chennai

5. Create the table using following attributes PASSENGER (PPNO: VARCHAR2 (15): PK, NAME: VARCHAR2 (15), AGE: INT (4), SEX: CHAR (10): MALE/FEMALE, ADDRESS: VARCHAR2 (50))

Creating table

Syntax

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

Query

CREATE TABLE PASSENGER (PPNO VARCHAR2 (15) PRIMARY KEY, NAME VARCHAR2 (15), AGE NUMBER (4), SEX CHAR (10), ADDRESS VARCHAR2 (50));

Describing table

Query

Desc PASSENGER

Output

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PASSENGER	PPNO	Varchar2	15	-	-	1	-	-	-
	NAME	Varchar2	15	-	-	-	✓	-	-
	AGE	Number	-	4	0	-	✓	-	-
	SEX	Char	10	-	-	-	✓	-	-
	ADDRESS	Varchar2	50	-	-	-	✓	-	-

1 - 5

Inserting records into "PASSENGER" table

Syntax

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

Query

insert into PASSENGER values(1,'TIRUMALAY',19,'MALE','AMBERPET');

insert into PASSENGER values(2,'SUPRIYA',20,'FEMALE','B.B NAGAR');

insert into PASSENGER values(3,'AMULYA',20,'FEMALE','ECIL');

insert into PASSENGER values(4,'NAGARAJU',20,'MALE','NAGARAM');

insert into PASSENGER values(5,'AVS.RAVI',20,'MALE','B.B NAGAR');

Display table

Syntax

Select <select list> from <table name>

Query

Select * from PASSENGER;

Output

PPNO	NAME	AGE	SEX	ADDRESS
1	TIRUMALAY	19	MALE	AMBERPET
2	SUPRIYA	20	FEMALE	B.B NAGAR
3	AMULYA	20	FEMALE	ECIL
4	NAGARAJU	20	MALE	NAGARAM
5	AVS.RAVI	20	MALE	B.B NAGAR

VIVA QUESTIONS

1. What is the syntax for insert command?
2. Define Key constraint?
3. What is the difference between NULL Values and NOT NULL Values?
4. What is the command to display table?

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
TICKET	TICKET_NO	Number	-	9	0	1	-	-	-
	JOURNEY_DATE	Date	7	-	-	-	✓	-	-
	AGE	Number	-	4	0	-	✓	-	-
	SEX	Char	10	-	-	-	✓	-	-
	SOURCE	Varchar2	50	-	-	-	✓	-	-
	DEP_TIME	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','10.30 AM');

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

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
1213	10-FEB-11	19	FEMALE	HYDERABAD	10.30 AM
1201	13-FEB-11	20	FEMALE	HYDERABAD	11.30 AM
1202	14-FEB-11	20	MALE	TIRUPATHI	11.00 AM
1205	14-FEB-11	20	MALE	HYDERABAD	11.00 AM

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) PRIMARY KEY, TICKET_NO NUMBER (9)**)**

Describing table

Query

Desc PASSENGER_TICKETS

Output

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
PASSENGER_TICKETS	PPNO	Varchar2	15	-	-	1	-	-	-
	TICKET_NO	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)

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),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
CANCELLATION	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 "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

Output

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

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_no

union

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	MALE	AMBERPET	1	10-FEB-11	5	amberpet	7416944004	yes
1	TIRUMALAY	19	MALE	AMBERPET	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
4	NAGARAJU	20	MALE	NAGARAM	4	14-FEB-11	4	nagaram	9700135300	yes
5	AVS.RAVI	20	MALE	B.B NAGAR	5	15-FEB-11	7	b.b nagar	8143528258	yes
5	AVS.RAVI	20	MALE	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 distict (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 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 following 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 doest 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)

1. Example:

```
SQL>CREATE VIEW employee AS SELECT empno,ename,job FROM EMP
    WHERE 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(‘ABCDEFGHIJ’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-o5.

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: Creation of INSERT TRIGGER, DELETE TRIGGER, UPDATE TRIGGER.

Objectives:

Student will able to learn to monitor a database and take initiate action when a condition occurs.

Outcomes:

Student gains the ability to change database manager from a passive system to an active one.

1)Write a trigger which ill check if the passenger is greater than 40 before a row is inserted into a passenger table.

2 SQL>CREATE OR REPLACE TRIGGER passenger_age

3 Before insert on passenger

4 FOR EACH ROW

5 DECLARE

6 BEGIN

7 IF(:new, Age<40)THEN

8 Raise-application-error (-20110,'invalid age');

9 END IF;

10 END;

Trigger created

SQL>insert into passenger values (81,'srinu', 10, 22);

Value inserted

SQL>insert into passenger values (25,'rksd', 5, 16);

ORA-20110: invalid age.

ORA-06512: at"gcet_550.passenger_age", line 4.

ORA-04088: error during execution of trigger'gcet_550.passenger_age'

VIVA QUESTIONS

1. What is Trigger?
2. What is Nested Trigger?
3. What are the types on Triggers?
4. What is the difference between For Trigger and After trigger?

WEEK 11:

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,stdFirstname
 FROM students WHERE stdId=in_customer_id;
 OPEN c1;
 FETCH c1 into v_id, v_name;
 Close c1;
 End;
 /
 PL/SQL procedure successfully completed.
```

VIVA QUESTIONS

1. What is difference between Function and Stored Procedure?
2. What is Stored Procedure?
3. What is PL/SQL?
4. Show how functions and procedures are called in a PL/SQL block?

WEEK 12

Cursors: A cursor on the given database.

Objectives:

Student will able to learn about cursors that enable traversal over the records in a database.

Outcomes:

Student gains the ability to implement cursors on the database.

We use a cursor when we have a SELECT statement that returns more than one row from the database. A cursor is basically a set of rows that we can access one at a time.

We retrieve the rows into the cursor using our SELECT statement and then fetch the rows from the cursor.

We may follow five steps when using a cursor.

Declare variables to store the column values from the SELECT statements.

1. Declare the cursor specifying our SELECT statement.
2. Open the cursor.
3. Fetch the rows from the cursor.
4. Close the cursor.

```
declare
        v_sno student.sno%TYPE;
        v_snname student.sname%TYPE;
        v_branch student.branch%TYPE;
        v_age student.age%TYPE;
        cursor MyCursor (c_sno number) is select sno,sname,branch,age from student
where sno=c_sno;--c_record MyCursor%rowtype;

begin
        if (NOT MyCursor%ISOPEN)then
                Open MyCursor (1201);
        end if;

        loop
```

```
            fetch MyCursor into v_sno,v_sname,v_branch,v_age;
            exit when MyCursor%NOTFOUND;
        dbms_output_line
("Record'||MyCursor%ROWCOUNT||':'||v_sno||''||v_sname||''||v_branch||''||v_age)
;
        endloop;

        if(MyCursor%ISOPEN)then
            close MyCursor;
        endif;

        if(NOT MyCursor%ISOPEN)then
            oprn MyCursor(1202);
        endif;

        loop
            fetch MyCursor into v_sno,v_sname,v_branch,v_age;
            exit when MyCursor%NOTFOUND;
        dbms_output.put_line ('Record'||MyCursor%ROWCOUNT||':'||v_sno||''||v_snam
e||''||v_branch||''||v_age);
        endloop;

        if (MyCursor%ISOPEN)then
            Close Mycursor;
        endif;
end;
```

VIVA QUESTIONS

1. Show code of a cursor for loop?
2. Explain uses of cursor?
3. What is Raise_application_error?
4. What is Cursor?

ADDITIONAL PROGRAMS

1) Queries on employee, department tables

syntax

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

Query

```
SQL> create table employee1(empsid
number,empname varchar(20),empjob
varchar(20),empsalary number,empjoindate
varchar(20));
```

Describing table

```
SQL> desc employee1;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
 EMPSID                                    NUMBER
 EMPNAME                                   VARCHAR2(20)
 EMPJOB                                    VARCHAR2(20)
 EMPSALARY                                 NUMBER
 EMPJOINDATE                               VARCHAR2(20)
```

Inserting records into table

```
SQL>insert  into  employee1
values(&empsid,&empname,&empjob,&empsalary,&empjoindate);
Enter value for empsid: 20
Enter value for empname: 'anil'
Enter value for empjob: 'java'
```

```
Enter value for empsalary: 20000
Enter value for empjoindate: '4-jan-2009'
old   1: insert into employee1
values(&empsid,&empname,&empjob,&empsalary,&empjoindate)
new   1: insert into employee1 values(20,'anil','java',20000,'4-jan-2009')
1 row created.
SQL> /
Enter value for empsid: 50
Enter value for empname: 'jyothi'
Enter value for empjob: 'faculty'
Enter value for empsalary: 30000
Enter value for empjoindate: '5-mar-2003'
old   1: insert into employee1
values(&empsid,&empname,&empjob,&empsalary,&empjoindate)
new   1: insert into employee1 values(50,'jyothi','faculty',30000,'5-mar-2003')
1 row created.
SQL> /
Enter value for empsid: 60
Enter value for empname: 'avinash'
Enter value for empjob: 'software'
Enter value for empsalary: 30000
Enter value for empjoindate: '5-apr-2009'
old   1: insert into employee1
values(&empsid,&empname,&empjob,&empsalary,&empjoindate)
new   1: insert into employee1 values(60,'avinash','software',30000,'5-apr-2009')
1 row created.
SQL> /
Enter value for empsid: 70
```

Enter value for empname: 'anitha'

Enter value for empjob: 'sql'

Enter value for empsalary: 40000

Enter value for empjoindate: '2-jun-2008'

old 1: insert into employee1

values(&empsid,&empname,&empjob,&empsalary,&empjoindate)

new 1: insert into employee1 values(70,'anitha','sql',40000,'2-jun-2008')

1 row created.

Display table

SQL> select * from employee1;

EMPSID	EMPNAME	EMPJOB	EMPSALARY	EMPJOINDATE
20	anil	java	20000	4-jan-2009
50	jyothi	faculty	30000	5-mar-2003
60	avinash	software	30000	5-apr-2009
70	anitha	sql	40000	2-jun-2008

SQL> create table department(dpsid

2 number,dpname varchar(20),dpjob

3 varchar(20),dpjoindate

4 varchar(20),dplocation varchar(20));

Table created.

SQL> desc department;

Name	Null?	Type
DPSID		NUMBER
DPNAME		VARCHAR2(20)

DPJOB VARCHAR2(20)

DPJOINDATE VARCHAR2(20)

DPLOCATION VARCHAR2(20)

SQL> insert into department values(&dpsid,&dpname,&dpjob,&dpjoindate,&dplocation);

Enter value for dpsid: 1

Enter value for dpname: 'cse'

Enter value for dpjob: 'dbms'

Enter value for dpjoindate: '4-jan-2002'

Enter value for dplocation: 'hyd'

old 1: insert into department values(&dpsid,&dpname,&dpjob,&dpjoindate,&dplocation)

new 1: insert into department values(1,'cse','dbms','4-jan-2002','hyd')

1 row created.

SQL> /

Enter value for dpsid: 2

Enter value for dpname: 'os'

Enter value for dpjob: 'co'

Enter value for dpjoindate: '5-apr-2003'

Enter value for dplocation: 'hyd'

old 1: insert into department values(&dpsid,&dpname,&dpjob,&dpjoindate,&dplocation)

new 1: insert into department values(2,'os','co','5-apr-2003','hyd')

1 row created.

SQL> /

Enter value for dpsid: 3

Enter value for dpname: 'ece'

Enter value for dpjob: 'stld'

Enter value for dpjoindate: '6-dec-2004'

Enter value for dplocation: 'hyd'

```
old  1: insert into department values(&dpsid,&dpname,&dpjob,&dpjoindate,&dplocation)
new  1: insert into department values(3,'ece','stld','6-dec-2004','hyd')
1 row created.
SQL> /
Enter value for dpsid: 4
Enter value for dpname: 'eee'
Enter value for dpjob: 'bee'
Enter value for dpjoindate: '6-jun-2006'
Enter value for dplocation: 'hyd'
old  1: insert into department values(&dpsid,&dpname,&dpjob,&dpjoindate,&dplocation)
new  1: insert into department values(4,'eee','bee','6-jun-2006','hyd')
1 row created.
SQL> select * from department;
```

DPSID	DPNAME	DPJOB	DPJOINDATE	DPLOCATION
1	cse	dbms	4-jan-2002	hyd
2	os	co	5-apr-2003	hyd
3	ece	stld	6-dec-2004	hyd
4	eee	bee	6-jun-2006	hyd

DDL&DML COMMANDS on Sailors, boats & reserves

DDL(Data Definition Language):-

```
SQL> alter table employee1 add emptime number;
Table altered.
SQL> desc employee1;
```

Name	Null?	Type
EMPSID		NUMBER
EMPNAME		VARCHAR2(20)
EMPJOB		VARCHAR2(20)
EMPSALARY		NUMBER
EMPJOINDATE		VARCHAR2(20)
EMPTIME		NUMBER

SQL> drop table employee1;

Table dropped.

SQL> desc employee1;

ERROR:

ORA-04043: object employee1 does not exist

1. Create the table using following attributes Sailors (**sid: number**, sname: varchar2(20), rating: number, age: number(4,2));

Creating table

Syntax

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

Query

Create table sailors (sid number, sname varchar2(20), rating number, age number(4,2));

Describing table

Query

Desc sailors

Output

Inserting records into table

Syntax

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

Query

insert into sailors values(22,'dustin',7,45.0);

insert into sailors values(29,'brutus',1,33.0);

insert into sailors values(31,'lubber',8,55.5);

insert into sailors values(32,'andy',8,25.5);

insert into sailors values(58,'rusty',10,35.0);

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
SAILORS	SID	Number	-	-	-	1	-	-	-
	SNAME	Varchar2	20	-	-	-	✓	-	-
	RATING	Number	-	-	-	-	✓	-	-
	AGE	Number	-	4	2	-	✓	-	-

1 - 4

insert into sailors values(64,'horatio',7,35.0);

insert into sailors values(71,'zobra',9,35.0);

insert into sailors values(74,'horatio',9,35.0);

insert into sailors values(85,'art',3,25.5);

insert into sailors values(95,'bob',3,63.5);

Display table

Syntax

Select <select list> from <table name>

Query

select * from sailors;

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment

Output

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	horatio	7	35
71	zobra	10	16
74	horatio	9	35
85	art	3	25.5
95	bob	3	63.5

2. Create the table using following attributes Boats (bid:number, bname:varchar22(10),color:varchar2(10));

Creating table

Syntax

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

Query

Create table Boats (bid number, bname varchar2(10),color varchar2(10));

Describing table

Query

Desc boats

Output

BOATS	BID	Number	-	-	-	1	-	-	-
	BNAME	Varchar2	10	-	-	-	✓	-	-
	COLOR	Varchar2	10	-	-	-	✓	-	-

1 - 3

Inserting records into table

Syntax

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

Query

insert into sailors values(101,'interlake',’blue’);

insert into sailors values(102,'interlake',’red’);

insert into sailors values(103,'clipper’,’green’);

insert into sailors values(104,'marine',’red’);

Display table

Syntax

Select <select list> from <table name>

Query

select * from boats;

Output

BID	BNAME	COLOR
101	interlake	blue
102	interlake	red
103	clipper	green
104	marine	red

3. Create the table using following attributes Reserves (sid: number, bid: number, day: date);

Creating table

Syntax

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

Query

Create table Reserves (sid number, bid number, day date);

Describing table

Query

Desc reserves

Output

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
RESERVES	SID	Number	-	-	-	1	-	-	-
	BID	Number	-	-	-	2	-	-	-
	DAY	Date	7	-	-	-	✓	-	-

Inserting records into table

Syntax

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

Query

insert into sailors values(22,101,'10/10/98');

insert into sailors values(22,102,'10/10/98');

insert into sailors values(22,103,'10/8/98');

insert into sailors values(22,104,'10/7/98');

insert into sailors values(31,102,’11/10/98’);

insert into sailors values(31,103,’10/6/98’);

insert into sailors values(31,104,’10/12/98’);

insert into sailors values(64,101,’9/5/98’);

Display table

Syntax

Select <select list> from <table name>

Query

select * from reserves;

Output

SID	BID	DAY
22	101	10/10/98
22	102	10/10/98
64	102	9/8/98
74	103	9/8/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98

1) Find the names of sailors who have reserved boat number 103 (Using AND).

Query

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

SNAME
horatio

Output

dustin
lubber

2) Find the sids of sailors who have reserved a red boat (Using AND).

Query

select r.sid from boats b,reserves r where b.bid=r.bid and b.color='red'

Output

SID
22
64
22
31
31

3) Find the names of sailors who have reserved a red boat (Using AND).

Query

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

Output

SNAME
dustin
horatio
dustin
lubber
lubber

4) Find the colors of boats reserved by Lubber (Using AND).

Query

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

Output

COLOR
green
red
red

5) Find the names of sailors who have reserved at least one boat.

Query

Select s.sname from sailors s, reserves r where s.sid=r.sid;

Output

SNAME
lubber
dustin
horatio

6) Find the ages of sailors whose name begins and ends with B and has at least three characters.

Query

Select s.age from sailors s where s.sname like 'b_%b';

Output

AGE
63.5

7) Find the names of sailors who have reserved a red or a green boat (Using AND,OR).

Query

select s.sname from sailors s,reserves r,boats b where s.sid=r.sid and r.bid=b.bid and (b.color='re' or b.color='green')

Output

SNAME
horatio
dustin
lubber

8) Find the names of sailors who have reserved both a red and a green boat.

Query

select s.sname from sailors s,reserves r,boats b

where s.sid =r.sid and r.bid=b.bid and b.color='red'

intersect

select s2.sname from sailors s2,reserves r2,boats b2

where s2.sid =r2.sid and r2.bid=b2.bid and b2.color='green'

Output

SNAME
dustin
horatio
lubber

9)Find the sids of sailors who have reserved red boats but not green boats.

Query

Select s1.sname from sailors s1, reserves r1, boats b1 where s1.sid=r1.sid and r1.bid=b1.bid and b1.color='red'

and s1.sid not in (select s2.sid from sailors s2,boats b2,reserves r2 where s2.sid=r2.sid and r2.bid=b2.bid and b2.color='green');

Output

SNAME
horatio

10) Find all sids of sailors who have a rating of 10 or reserved boat 104.

Query

Select s.sid from sailors s where s.rating=10

Union

Select r.sid from reserves r where r.bid=104;

Output

SID
22
31
58
71

11) Find the names of sailors who have reserved boat 103 (Using nested queries).

Query

Select s.sname from sailors s where s.sid in (select r.sid from reserves r where r.bid=103)

Output

SNAME
dustin
lubber
horatio

12) Find the names of sailors who have reserved a red boat (Using nested queries).

Query

```
select s.sname from sailors s where s.sid in (select r.sid from reserves r where r.bid
in(select b.bid from boats b  where b.color='red'));
```

Output

SNAME
dustin
lubber
horatio

13. Find the names of sailors who have not reserved a red boat (Using nested queries).

Query

```
Select s.sname from sailors s where s.sid not in (select r.sid from reserves r where
r.bid in (select b.bid from boats b where b.color=’red’));
```

Output

SNAME
brutus
andy
rusty
zobra
horatio
art
bob

14. Find the name of sailors who have reserved boat number 103 (Using correlated nested query).

Query

```
Select s.sname from sailors s where exists (select * from reserves r where
r.bid=103 and r.sid=s.sid);
```

Output

SNAME
dustin
lubber
horatio

15. Find sailors whose rating is better than some sailor called Horatio (Using ANY Operator).

Query

```
select s.sid from sailors s where s.rating > any(select s1.rating from sailors s1
where s1.sname='horatio');
```

Output

SID
31
32
58
71
74

16. Find sailors whose rating is better than every sailor called Horatio (Using ALL Operator).

Query

```
Select s1.sid from sailors s1 where s1.rating > all (select s2.rating from sailors s2
where s2.sname='horatio');
```

Output

SID
58
71

17. Find the sailors with the highest rating (Using ALL Operator).

Query

Select s1.sid from sailors s1 where s1.rating >= all (select s2.rating from sailors s2);

Output

SID
58
71

18. Find the names of sailors who have reserved both a red and a green boat (Using nested query).

Query

Select s1.sname from sailors s1, reserves r1, boats b1 where s1.sid=r1.sid and r1.bid=b1.bid and b1.color='red'

and s1.sid in (select s2.sid from sailors s2,boats b2,reserves r2 where s2.sid=r2.sid and r2.bid=b2.bid and b2.color='green');

Output

SNAME
lubber
dustin

19. Find the average age of all sailors.

Query

Select avg (s.age) from sailors s;

Output

AVG(S.AGE)
36.9

20. Find the average age of sailors with a rating of 10.

Query

Select avg (s.age) from sailors s where s.rating=10;

Output

AVG(S.AGE)
25.5

21. Find the name and age of the oldest sailor.

Query

Select s.sname,s.age from sailors s where s.age=(select max (s2.age) from sailors s2);

Output

SNAME	AGE
bob	63.5

22. Count the number of different sailor names.

Query

Select count(distinct(s.sname)) from sailors s;

Output

COUNT(DISTINCT(S.SNAME))
9

23. Find the names of sailors who are older than oldest sailor with a rating 10 (Using Aggregate Operators).

Query

Select s1.sname from sailors s1 where s1.age > (select max (s2.age) from sailors s2 where s2.rating =10);

Output

SNAME
dustin
lubber
bob

24. Find the age of the youngest sailor who is eligible to vote for each rating level with at least two such sailors.

Query

Select s.rating, min (s.age) as minage from sailors s where s.age>=18 group by s.rating having count (*) > 1;

25.Find the age of the youngest sailor for each rating level (Using Group By).

Query

Select s.rating, min (s.age) from sailors s group by s.rating;

Output

RATING	MIN(S.AGE)
1	33
8	25.5
7	35
3	25.5
10	16
9	35