	G. K. Gujar Memorial Charitable Trust's, **Dr. Ashok Gujar Technical Institute's** **Dr. Daulatrao Aher College of Engineering, Karad.**
	Department of Computer Science & Engineering

Database Engineering (DBE)

Lab Manual

Academic year (2023-2024) VI SEMESTER

Procedure for performing experiments

a) Explanation by the faculty using OHP/PPT covering the following aspects: 20 min.

1) Aim of the experiment
2) Software/Inputs required
3) Queries/Algorithm
4) Test Data/ Tables

b) Writing of source program by the students 20 min.

c) Compiling and execution of the program 60 min.

d) Assessment 20 min

Format of First Page

Name :

Student Roll No # :

Experiment # :

Experiment Title :

Date of Experiment :

Date of Submission :

Student Responsibilities

1. In the very beginning of the laboratory work, the student has to do programming individually. For this reason, regular attendance is strictly required.
2. Every laboratory session is divided into two parts. In the first part, the instructor will be lecturing on the test objective, procedure and data collection. In the second part, the students, organized in groups (individual student), are required to do the programming. In order to perform the experiment within the assigned period, and to gain the maximum benefit from the experiment, the students must familiarize themselves with the purpose, objective, and procedure of the experiment before coming to the laboratory. Relevant lecture notes and laboratory manual should be studied carefully and thoroughly.
3. At the end of the experiment, every student should submit the written algorithm & programs & testes result for approval by the instructor.
4. It should be understood that laboratory facilities and equipments (Hardware & Software) are provided to enhance the learning process.
5. The equipments must be properly cared after every laboratory session. Also, students should always take precautions to avoid any possible hazards. Students must follow laboratory regulations provided at the end of this section.

	G. K. Gujar Memorial Charitable Trust's, **Dr. Ashok Gujar Technical Institute's** **Dr. Daulatrao Aher College of Engineering, Karad.**
	Department of Computer Science & Engineering

EXPERIMENT LIST

Department: Computer Science and Engineering **Academic Year:** 2023-24

Class: Third Year (T.Y. B.Tech) **Semester:** VI

Subject: Database Engineering **Semester Duration:** 6 months

Expt. No.	TITLE OF THE EXPERIMENT
1	Introduction to Database Management System
2	Study and Draw E-R Model of DBMS
3	Installation & Demonstration of DBMS like MySql, Oracle etc. Reduce E-R model into tables
4	Implementation of program to modify data from database/data dictionary using JAVA and Oracle/SQL.
5	Study of Basic SQL Commands: DDL & DML queries.
6	Implementation of SQL Query constraints with referential integrity constraints.
7	Implementation of SQL Query Processing: program to show use of SQL Clauses-Order by, group by and between clauses.
8	Implementation of SQL Query Aggregate functions & set operations
9	Implementation of SQL query Join operations.
10	Implementation of String operations in SQL
11	Implementation of Views for any created table
12	Implementation of Static Hashing Technique using java
13	Study of NoSql

Experiment No.1

Title: Introduction to Database Management System

Aim: To Study the Database Management System (DBMS)

Prerequisites: Set Theory, Operating System, Data Structures

Theory:

- What is DBMS? Explain its Architecture with diagram
- Properties of DBMS
- Purpose of DBMS: Drawbacks of conventional file system
- Application of DBMS.

Conclusion: We have studied the Database management system, architecture and its user roles.

Questions:

1. What is Database Management System?
2. What are the Database System applications?
3. What is difference between Database System and File System?
4. Explain data abstraction, schema & data independence
5. Difference between Data Vs Information Vs Metadata
6. List and explain the various database users

Experiment No. 2

Title: Study and Draw E-R Model of DBMS
Aim: To learn and draw the E-R model of DBMS

Theory:

- Explain Entity Relationship model in detail
- Components of an E-R Model

An ERD typically consists of four different graphical components:

1. **Entity.** A data entity is anything real or abstract about which we want to store data. Entity types fall into five classes: roles, events, locations, tangible things or concepts. E.g. employee, payment, campus, book. Specific examples of an entity are called instances. E.g. the employee John Jones, Mary Smith's payment, etc.

2. **Relationship.** A data relationship is a natural association that exists between one or more entities. E.g. Employees process payments.

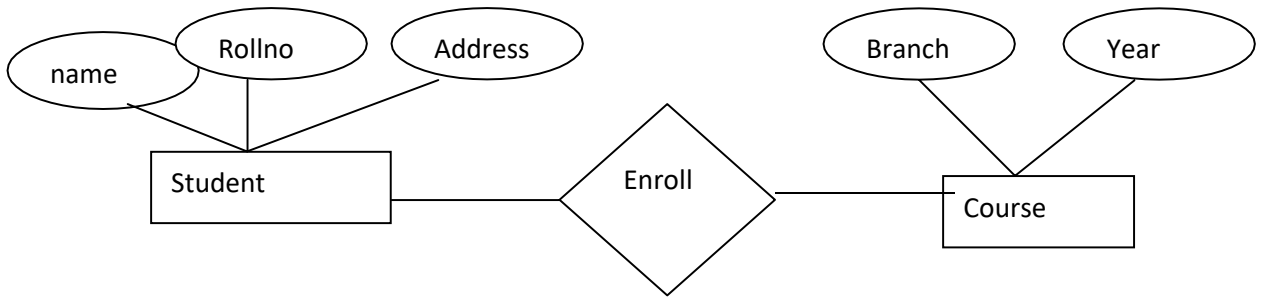
3. **Cardinality.** Defines the number of occurrences of one entity for a single occurrence of the related entity. E.g. an employee may process many payments but might not process any payments depending on the nature of her job.

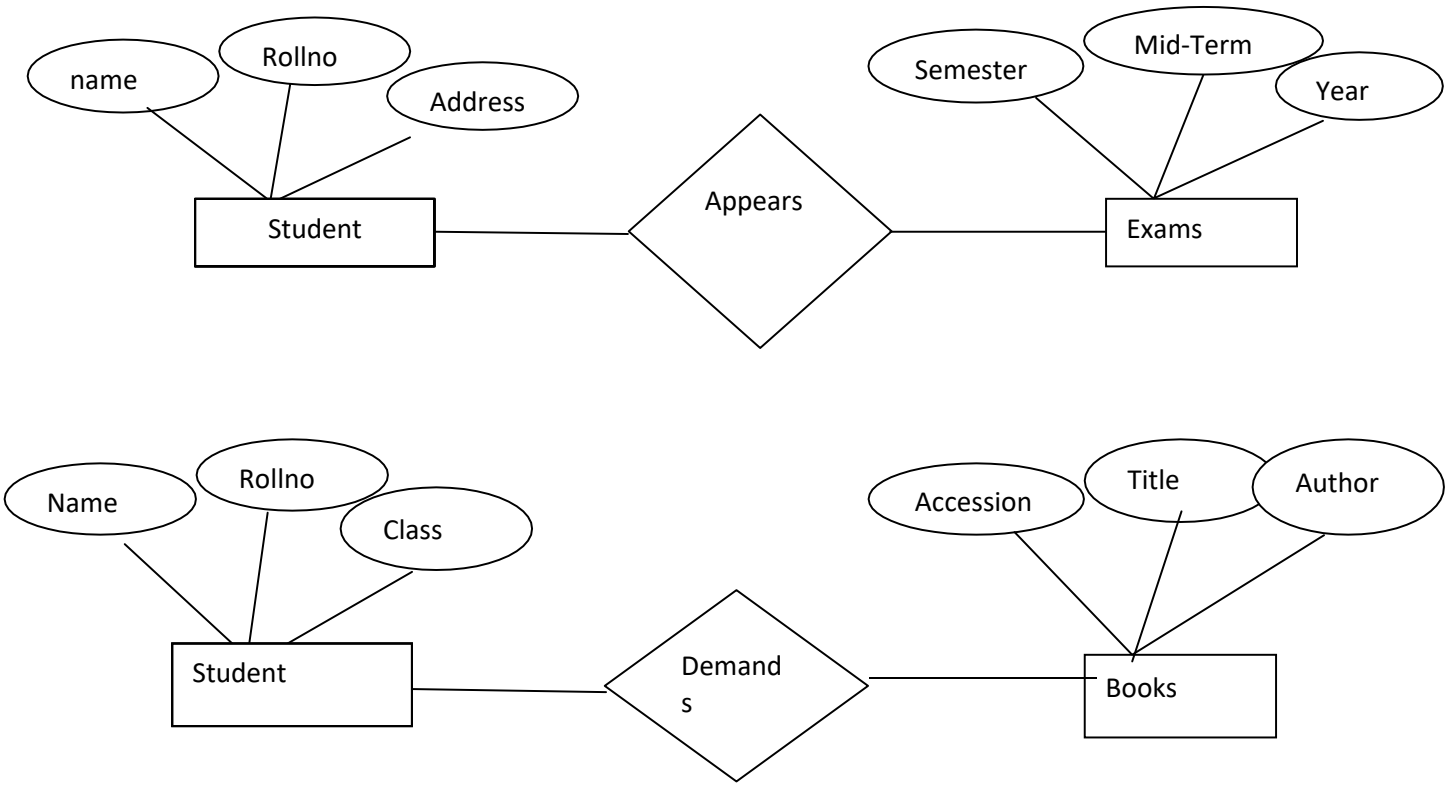
4. **Attribute**. A data attribute is a characteristic common to all or most instances of a particular entity. Synonyms include property, data element, field. E.g. Name, address, Employee Number, pay rate are all attributes of the entity employee. An attribute or combination of attributes that uniquely identifies one and only one instance of an entity is called a primary key or identifier. E.g. Employee Number is a primary key for Employee.

Components of E-R model: (Draw all sign conventions)

1. Rectangle: Represents Entity Set
2. Diamond: Represents Relationship
3. Ellipse : Represents attributes
4. Dashed Ellipse: Represents derived attributes
5. Double Ellipse: Represents multi valued attributes
6. Lines: Link to entity set and relationship
7. Double Lines: Total Participation

E-R Diagram for student activities at college:





Conclusion: We have studied the concepts of E-R Model and how to draw it.

Questions:

1) What is E-R Model? Draw it with an example
2) What is E-R diagram and its components?
3) Draw an ER diagram for the University database, Car Insurance Company and Bank Management System.
4) Draw E-R diagram and create schema in SQL for relation: Customer buying a car from car manufacturing company. If car has an accident then Insurance compony is liable to pay for damage recovery of the car
5) Draw E-R diagram and create schema in SQL for following relations in college management system
   Instructor (i_id, name, dept_name, salary)

   Teaches (id, course_id, section_id, sem, year)

   Student (sid, name, dept_name, tot_credits)

   Department (dept_name, building, budget)

   Course (course_id, title, dept_name, credits)

   Advisor(sid, i_id)

Experiment No. 3

Title: Installation & Demonstration of DBMS like MySql, Oracle etc. Reduce E-R model into tables

Aim: To Install & Demonstrate of DBMS like MySql, Oracle etc. and Reduce E-R model into tables (Practice session: Students should be allowed to choose appropriate DBMS software, install it, configure it and start working on it. Create sample tables, execute some queries, use SQLPLUS features, use PL/SQL features like cursors on sample database. Students should be permitted to practice appropriate User interface creation tool and Report generation tool)

Steps:

Welcome to Oracle Database 10g Express Edition (Oracle Database XE)! This tutorial gets you quickly up and running using Oracle Database XE by creating a simple application. This guide covers the following topics:

- Logging in as the Database Administrator
- Unlocking the Sample User Account
- Logging in as the Sample User Account
- Creating a Simple Application
- Running Your New Application
- Using the Oracle Database XE Menus

1. Logging in as the Database Administrator:

The first thing you need to do is to log in as the Oracle Database XE Administrator. Follow these steps:

1. Open the Database Home Page login window:

On Windows, from the Start menu, select **Programs**(or All Programs),then Oracle Database 10g Express Edition, and then **Go To Database Home Page**.

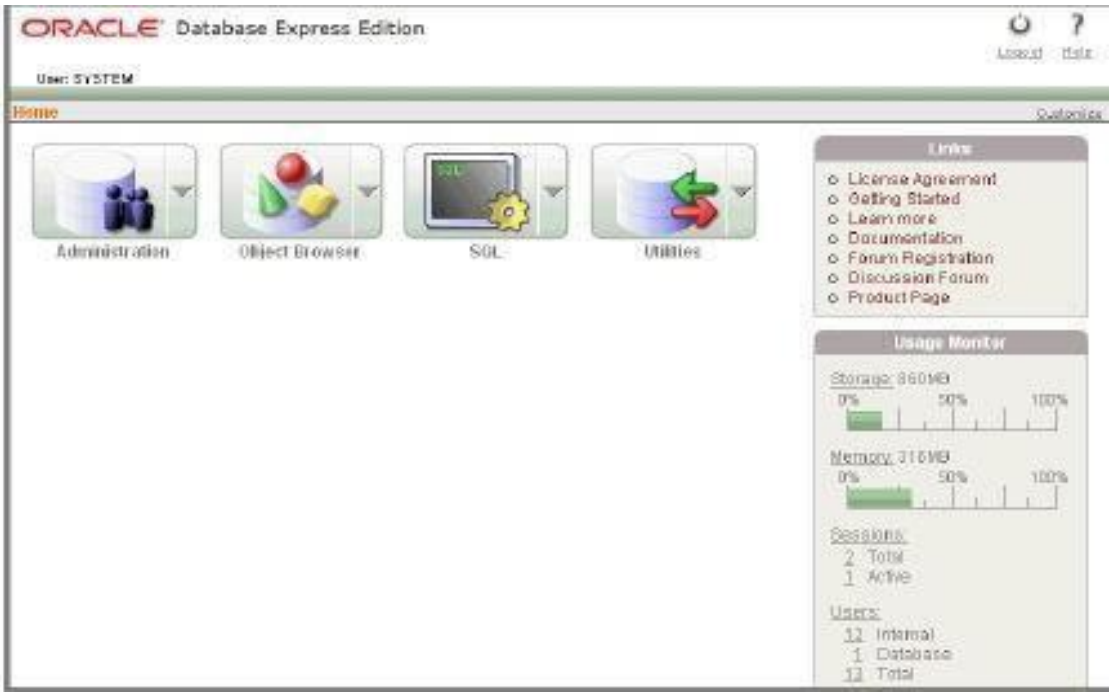
- On Linux, click the Application menu (on Gnome) or the K menu (on KDE), then point to Oracle Database 10g Express Edition, and then Go To Database Home Page.

2. At the Database Home Page login window, enter the following information:

- Username: Enter system for the user name.
- Password: Enter the password that was specified when Oracle Database XE was installed.

3.Click Login

The Oracle Database XE home page appears.



2. Unlocking the Sample User Account

To create your application, you need to log in as a database user. Oracle Database XE comes with a sample database user called HR. This user owns a number of database tables in a sample schema that can be used to create applications for a fictional Human Resources department. However, for security reasons, this users account is locked. You need to unlock this account before you can build a sample application.

To unlock the sample user account:

1. Make sure you are still logged on as the database administrator, as described in the previous section.
2. Click the Administration icon, and then click Database Users.
3. Click the HR schema icon to display the user information for HR.

- Under Manage Database User, enter the following settings:
- Password and Confirm Password: Enter hr for the password.
- Account Status: Select Unlocked.
- Roles: Ensure that both CONNECT and RESOURCE are enabled.

4. Click Alter User.

Now you are ready to create your first application.

3. Logging in as the Sample User Account

To log in as the sample user account:

1. Log out from the database administrator account by clicking Logout in the upper right corner of the Database Home Page.
2. In the window, click Login.
3. In the Login window, enter hr for both the user name and password.
4. Click Login.

The Database Home Page appears.

4. Creating a Simple Application

Creating an application is an easy way to view and edit your database data. You create this application based on the EMPLOYEES table, which is part of the HR schema.

To create an application based on the EMPLOYEES table:

1. On the Database Home Page, click the Application Builder icon.
2. Click the Create button.
3. Under Create Application, select Create Application and click Next.
4. Under Create Application:

a. Name: Enter MyApp.

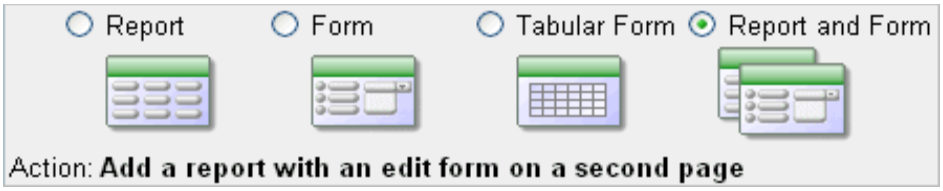
b. Accept the remaining defaults.

c. Click Next.

Next, add pages to your application.

5. Under Add Page:

a. For Select Page Type, select Report and Form.

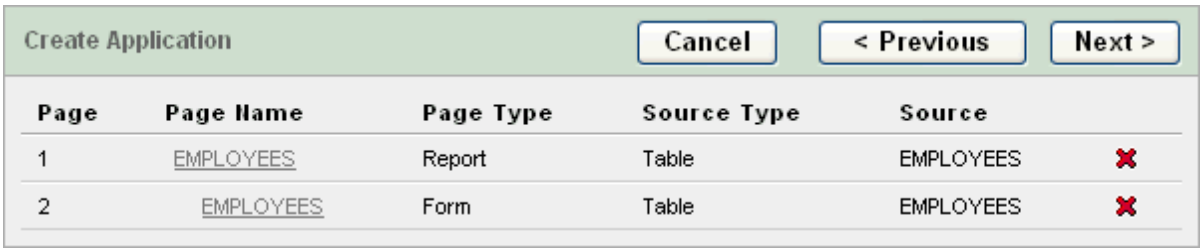


Notice that Action describes the type of page you are adding.

b. Next to the Table Name field, click the up arrow, and then select EMPLOYEES from the Search Dialog window.

c. Click Add Page.

Two new pages display at the top of the page, under Create Application.



d. Click Next

6. On the Tabs panel, accept the default (One Level of Tabs) and click Next.

7. On the Shared Components panel, accept the default (No) and click Next.

This option enables you to import shared components from another application. Shared components are common elements that can display or be applied on any page within an application.

8. For Authentication Scheme, Language, and User Language Preference Derived From, accept the defaults and click Next.

9. For the theme, select Theme 2 click Next. Themes are collections of templates that you can use to define the layout and style of an entire application.

10. Confirm your selections. To return to a previous wizard page, click Previous.

To accept your selections, click Create.

After you click Create, the following message displays at the top of the page:

Application created successfully.

5. Running Your New Application:

To run your application:

1. Click the Run Application icon.

In the log in page, enter hr for both the User Name and Password. Your application appears, showing the EMPLOYEES table.

3. Explore your application.

You can query the EMPLOYEES table, if you want. To manage the application, use the Developer toolbar at the bottom on the page.

The Developer toolbar offers a quick way to edit the current page, create a new page, control, or component, view session state, or toggle debugging or edit links on and off.

4. To exit your application and return to Application Builder, click Edit Page 1 on the Developer toolbar.

5. To return to the Database Home Page, select the Home breadcrumb at the top of the page.

Congratulations! You have just created your first application using Oracle Database XE.

6. Using the Oracle Database XE Menus:

You can use the Oracle Database XE menus to perform basic functions with Oracle Database XE. To see the menus, do the following:

■ On Windows, from the Start menu, select Programs (or All Programs) and then Oracle Database 10g Express Edition.

■ On Linux, click the Application menu (on Gnome) or the K menu (on KDE) and then point to Oracle Database 10g Express Edition. The following menu items are available:

■ Get Help: Displays the following selections:

- Go To Online Forum: Displays the online forum for discussions about Oracle Database XE.
- Read Documentation: Displays the Oracle Database XE documentation library on the Internet.
- Read Online Help: Displays the Oracle Database XE online help. This help is only available if the database is started.

■Register For Online Forum: Allows you to register for the Oracle Database XE online forum.

■Backup Database: In NOARCHIVELOG mode (the default), shuts down the database, backs it up, and then restarts it. In ARCHIVELOG mode, performs an online backup of the database. For more information on backups, refer to Oracle Database Express Edition 2 Day DBA.

■ Get Started: Link to this tutorial.

■ Go To Database Home Page: Displays the Oracle Database XE Home Page in your default browser. "Logging in as the Database Administrator" on page 1 explains how to log into this home page as a database administrator.

■ Restore Database: Shuts down and then restores the database to the most recent backup. For more information on restoring a database, refer to Oracle Database Express Edition 2 Day DBA.

■ Run SQL Command Line: Starts the SQL Command Line utility for Oracle Database XE. To connect to the database, issue the following command at the **SQL prompt that appears**:

Connect username/password

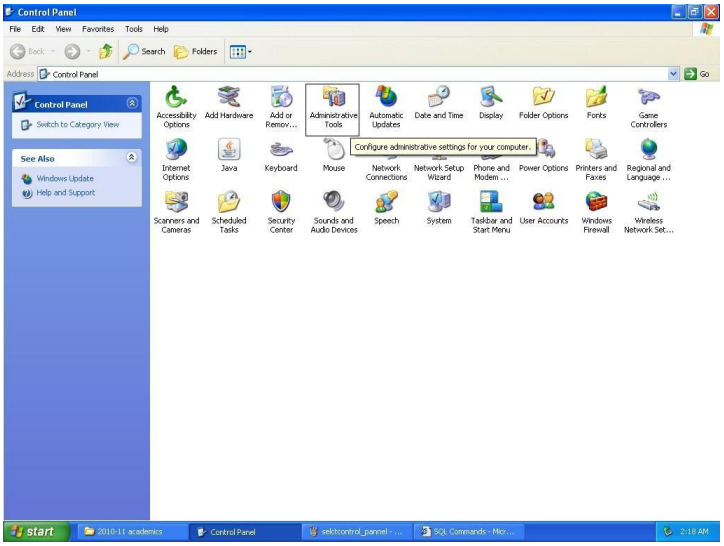
where username is the user name, such as sys, system, or another account name, and password is the password that was assigned when Oracle Database XE was installed. The get help, you can enter the command help at the SQL prompt, once you have connected to the database.

■ Start Database: Starts Oracle Database XE. By default, the database is started for you after installation and every time your computer is restarted. However, if you think the database is not running you can use this menu item to start it.

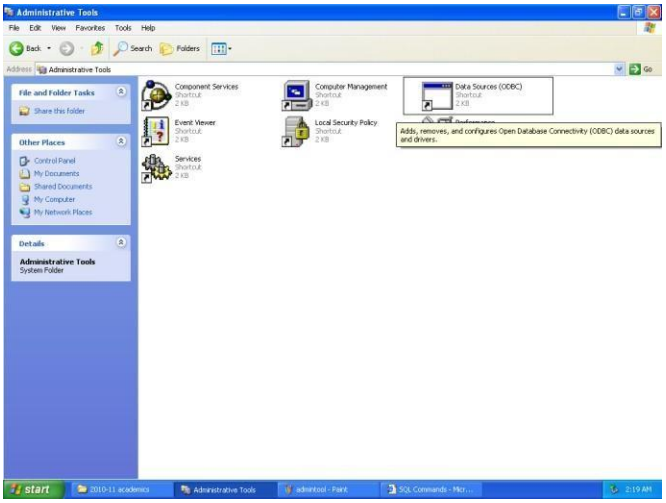
Stop Database: Stops Oracle Database XE.

How to create ODBC connectivity

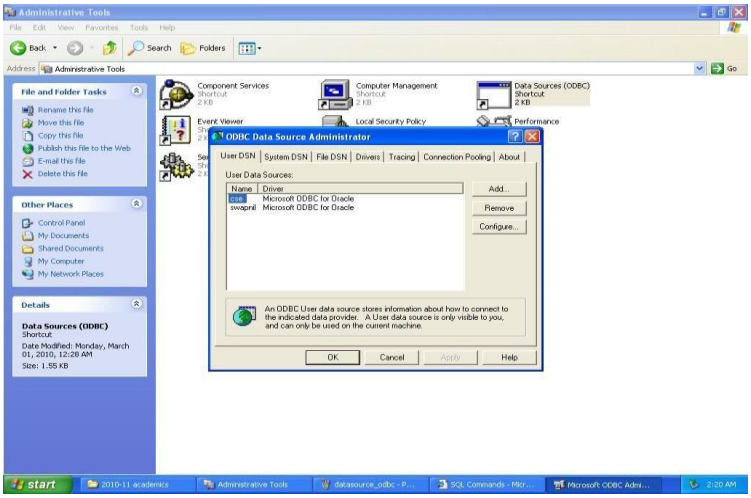
STEP 1: go to control panel and open Administrative tools



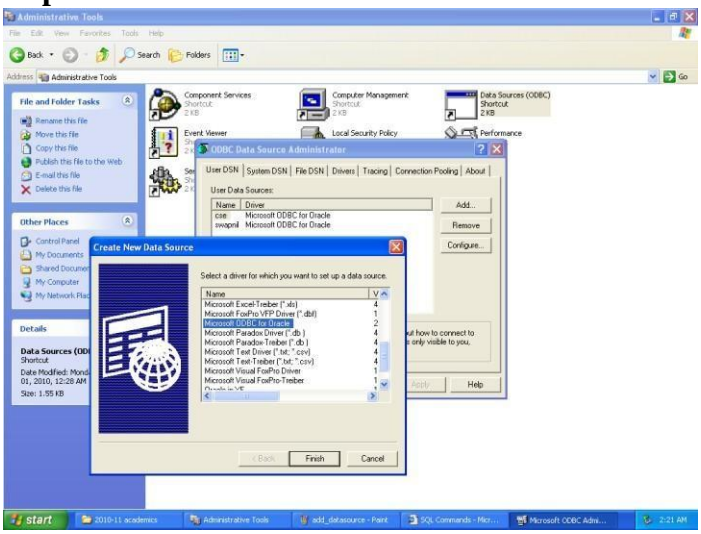
Step 2: select data source (ODBC) and open



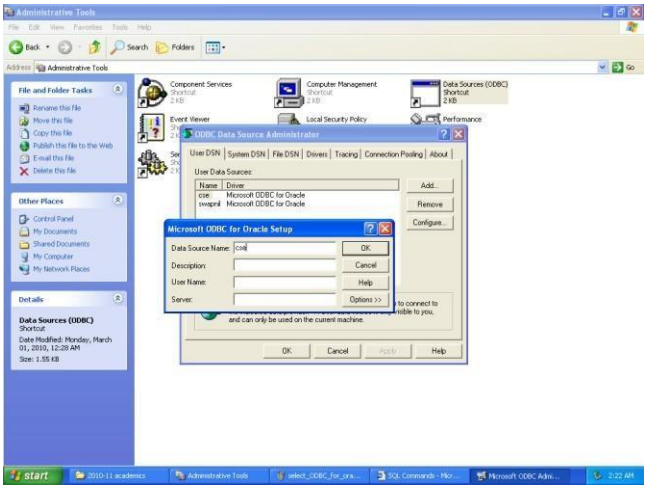
Step 3: Click Add…



Step 4: Select Microsoft ODBC for Oracle then click Finish.



Step 5: Enter Data Source Name(in prevision example it is "cse") and click Ok

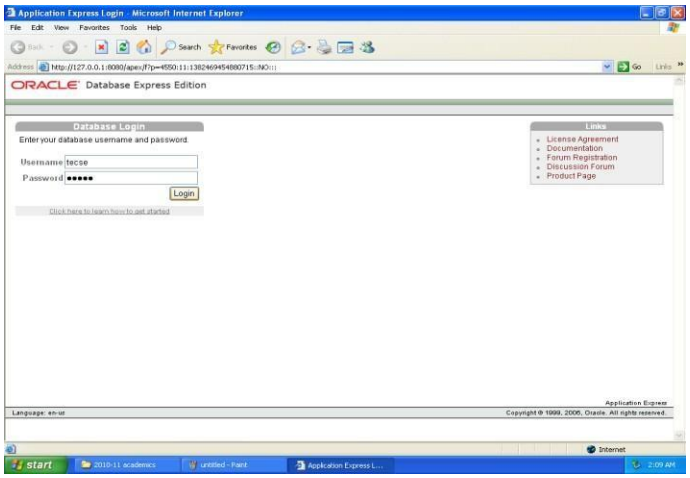


After compilation it will create DSN for oracle. This DSN is used during java program for connection con = DriverManager.getConnection("jdbc:odbc:**cse**","tecse","tecse");

How to run SQL queries in oracle 10 g.

Step1 : Select Oracle Database 10g Express Edition - Go To Database Home Page from Start menu

Step 2: login to oracle 10 g using username and password

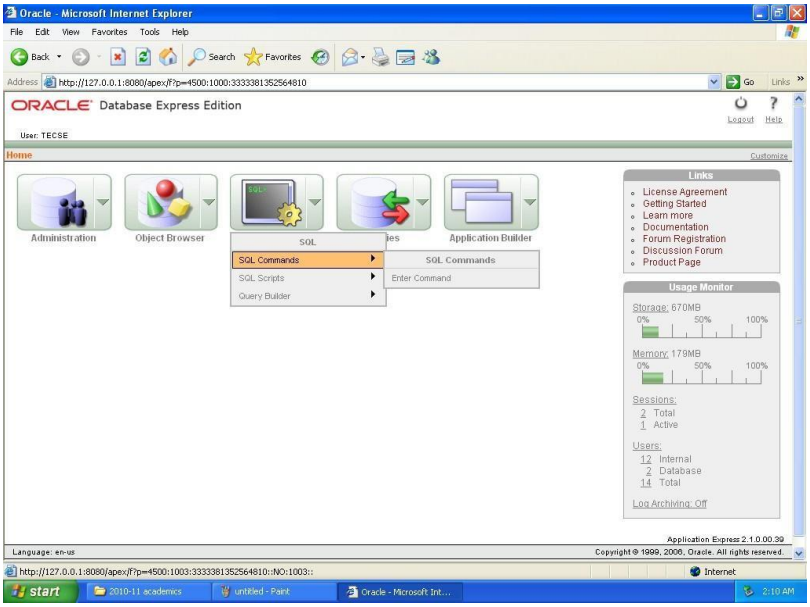


In our example we used username: tecse and password; tecse

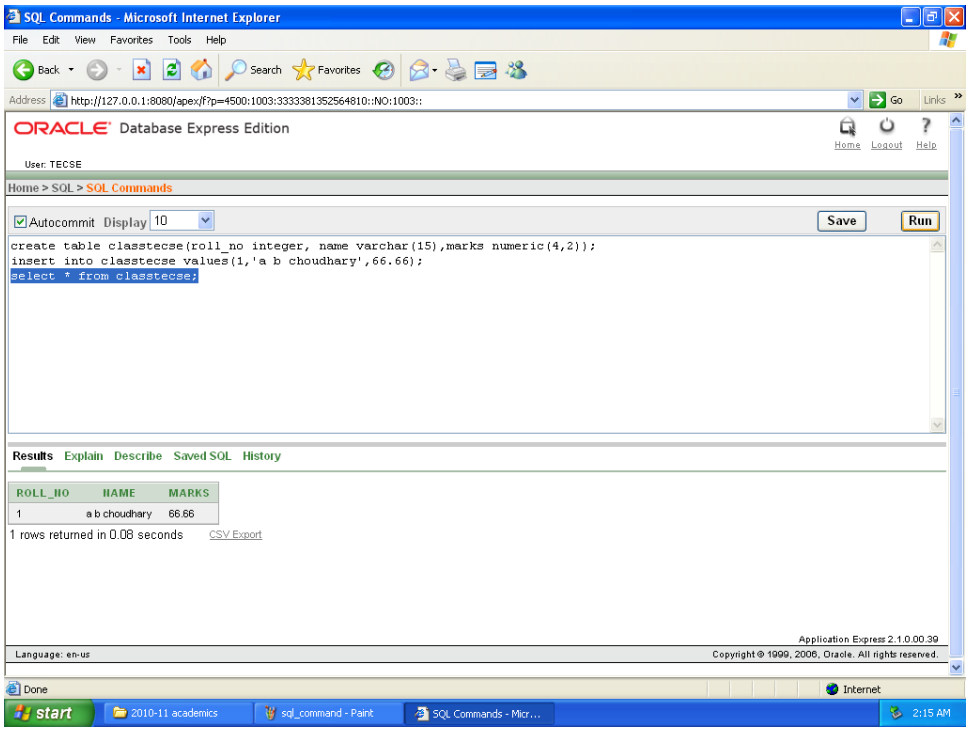
i.e. con = DriverManager.getConnection("jdbc:odbc:cse","tecse","tecse");

Passsword given at the time of installation is for user system which can used for create new users.

Step 3 : select SQL – SQL Commands – Enter Command



Step 4: type query then select and click Run button. You will get result in result window



e.g. select * from classtecse

Also we create different users and assign privileges through create user.

Role Name	Description
CONNECT	Enables a user to connect to the database. Grant this role to any user or application that needs database access.

Role Name	Description
RESOURCE	Enables a user to create certain types of schema objects in his own schema. Grant this role only to developers and to other users that must create schema objects. This role grants a subset of the *create object* system privileges. For example, it grants the CREATE TABLE system privilege, but does not grant the CREATE VIEW system privilege. It grants only the following privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
DBA	Enables a user to perform most administrative functions, including creating users and granting privileges; creating and granting roles; creating and dropping schema objects in other users' schemas; and more. It grants all system privileges, but does not include the privileges to start up or shut down the database. It is by default granted to user SYSTEM.

Conclusion: We have install Oracle Database 10g Express Edition (Oracle Database XE) successfully.

Experiment No.4

Title: Implementation of program to modify data from database/data dictionary using JAVA and Oracle/SQL.

Aim: To understand the connectivity to a database table using java. In java database is accessed through java database connectivity (JDBC).

Requirements: Mysql database software, Java software.

Theory/ Design:

The four main components required for implementation of JDBC are application, driver manager, data source specific drivers and corresponding data sources. The application program establishes/terminates connection with data. Driver manager will load JDBC drivers and pass JDBC function calls from the application to the driver. The data source processes commands from the driver and return the results.

1. Drivers for the database are loaded by using Class.for Name.

2. The getConnection() function of DriverManager class of JDBC is used to create the connection object. The URL specifies the machine name (db.onlinebook.edu)

```
public static void Sample(String DB_id, String U-id, String Pword)
{
StringURL =”jdbc:oracle:oci8:@db.onlinebook.edu:100:onbook_db”;
Class.forName(“oracle.jdbc.driver.OracleDriver”);
Connection cn=DriverManager.getConnection(URL,U_id,Pword);
Statement st = Cn.createStatement();
try {
st.executeUpdate(“INSERT INTO AUTHOR VALUES('A010','SMITH',JONES','TEXAS')”);
 }
catch(SQLException se)
{
System.out.println(“Tuple cannot be inserted.”+se);
}
ResultSet rs = st.execute Query(“SELECT Aname,State from AUTHOR WHERE City = 'Seatle'”);
while(rs.next())
{
System.out.println(rs.getString(1) + “ “+rs.getString(2));
}
 st.close();
Cn.close();
}
```

Database Servers:-

A database server is a computer program that provides database services to other computer programs or computers, as defined by the client-server model. The term may also refer to a computer dedicated to running such a program. Database management systems frequently provide database server functionality, and some DBMS's (e.g., MySQL) rely exclusively on the client-server model for database access. In a master-slave model, database master servers are central and primary locations of data while database slave servers are synchronized backups of the master acting as proxies.

My SQL:-

The Structured Query Language (SQL) is a very popular database language, and its standardization makes it quite easy to store, update and access data. One of the most powerful SQL servers out there is called MySQL and surprisingly enough, its free.

Some of the features of MySQL Include: Handles large databases, in the area of 50,000,000+ records. No memory leaks. Tested with a commercial memory leakage detector (purify). A privilege and password system which is very flexible and secure, and which allows host-based verification. Passwords are secure since all password traffic when connecting to a server is encrypted.

PL/SQL:-

PL/SQL is a database-oriented programming language that extends Oracle SQL with procedural capabilities. We will review in this lab the fundamental features of the language and learn how to integrate it with SQL to help solve database problems.SQL statements are defined in term of constraints we wish to fix on the result of a query. Such a language is commonly referred to as*declarative*. This contrasts with the so called procedural languages where a program specifies a list of operations to be performed sequentially to achieve the desired result. PL/SQL adds *selective* (i.e. if...then...else...) and *iterative*constructs(i.e.loops) to SQL. PL/SQL is most useful to write *triggers* and *stored procedures*. Stored procedures are units of procedural code stored in a compiled form within the database.

JDBC:-JDBC stands for "Java DataBase Connectivity". It is an API (Application Programming Interface) which consists of a set of Java classes, interfaces and exceptions and a specification to which both JDBC driver vendors and JDBC developers (like you) adhere when developing applications.

JDBC is a very popular data access standard. RDBMS (Relational Database Management Systems) or third-party vendors develop drivers which adhere to the JDBC specification. Other developers use these drivers to develop applications which access those databases e.g. you'll use ConnectorJ JDBC driver to access MySQL database. Since the drivers adhered to JDBC specification, the JDBC application developers can replace one driver for their application with another better one without having to rewrite their application. If they had used some proprietary API provided by some RDBMS vendor, they will not have been able to change the driver and/or database without having to rewrite the complete application.

Who develops JDBC Specification?

SUN prepares and maintains the JDBC specification. Since JDBC is just a specification (suggestions for writing and using JDBC drivers), third-party vendors develop JDBC drivers adhering to this specification. JDBC developers then use these drivers to access data sources.

Why use JDBC?
JDBC is there only to help you (a Java developer) develop data access applications without having to learn and use proprietary APIs provided by different RDBMS vendors. You just have to learn JDBC and then you can be sure that you'll be able to develop data access applications which can access different RDBMS using different JDBC drivers.

JDBC Architecture
We'll divide it into 2 parts:

JDBC API (java.sql & javax.sql packages)

- JDBC Driver Types

JDBC API
The JDBC API is available in the java.sql and javax.sql packages. Following are important JDBC classes, interfaces and exceptions in the java.sql package:

- DriverManager - Loads JDBC drivers in memory. Can also be used to open connections to a data source.
- Connection - Represents a connection with a data source. Is also used for creatingStatement, PreparedStatement and CallableStatement objects.
- Statement - Represents a static SQL statement. Can be used to retrieve ResultSetobject/s.
- PreparedStatement - Higher performance alternative to Statement object, represents a precompiled SQL statement.
- CallableStatement - Represents a stored procedure. Can be used to execute stored procedures in a RDBMS which supports them.
- ResultSet - Represents a database result set generated by using a SELECT SQL statement.
- SQLException - An exception class which encapsulates database base access errors.

JDBC Driver Types
There are 4 types of JDBC drivers. Commonest and most efficient of which are type 4 drivers. Here is the description of each of them:

- JDBC Type 1 Driver - They are JDBC-ODBC Bridge drivers. They delegate the work of data access to ODBC API. They are the slowest of all. SUN provides a JDBC/ODBC driver implementation.
- JDBC Type 2 Driver - They mainly use native API for data access and provide Java wrapper classes to be able to be invoked using JDBC drivers.
- JDBC Type 3 Driver - They are written in 100% Java and use vendor independent Net-protocol to access a vendor independent remote listener. This listener in turn maps the vendor independent calls to vender dependent ones. This extra step adds complexity and decreases the data access efficiency.
- JDBC Type 4 Driver - They are also written in 100% Java and are the most efficient among all driver types.

Introduction to ODBC

The Open Database Connectivity (ODBC) interface is a call-based application programming interface defined by Microsoft Corporation as a standard interface to database management systems on Windows operating systems. In addition, ODBC is now widely used on many non-Windows platforms, such as UNIX and Macintosh.

Software requirements

To write ODBC applications for Adaptive Server Enterprise, you need:

- Adaptive Server Enterprise, versions 12.0 and above.
- A C compiler capable of creating programs for your environment.
- ODBC Software Development Kit.
- On non-Windows platforms, there are open source projects like unixODBC and iOBDC that release the required headers and libraries to build ODBC applications.

Problem Statement:

1. Implementation of program to modify data from database using JAVA with backend Oracle/SQL.
2. Implementation of program to create, insert and delete in database using JAVA with backend Oracle/SQL.

Conclusion: We have implemented the JAVA program to connect the SQL database by using JDBC & ODBC connection

Experiment No. 5

Title: Study of Basic SQL Commands: DDL & DML queries.

Aim: To implement DDL & DML queries of SQL

Theory / Design:

- Introduction to SQL & basic queries.

Basic SQL:

Each record has a unique identifier or primary key. SQL, which stands for Structured Query Language, is used to communicate with a database. Through SQL one can create and delete tables. Here are some commands:

- CREATE TABLE - creates a new database table
- ALTER TABLE - alters a database table
- DROP TABLE - deletes a database table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

SQL also has syntax to update, insert, and delete records.

- SELECT - get data from a database table
- UPDATE - change data in a database table
- DELETE - remove data from a database table
- INSERT INTO - insert new data in a database table

1. SELECT

The SELECT is used to query the database and retrieve selected data that match the specific criteria that you specify:

SELECT *column1* [, *column2*, ...]
FROM *tablename*
WHERE *condition*

The conditional clause can include these operators

- = Equal
- > Greater than
- < Less than
- >= Greater than or equal
- <= Less than or equal

- <> Not equal to
- LIKE pattern matching operator

SELECT * FROM *tablename*

returns all the data from the table.

Use single quotes around text values (most database systems will also accept double quotes). Numerical values should not be enclosed in quotes.

LIKE matches a pattern. The wildcard *%* is used to denote 0 or more characters.

- 'A%' : matches all strings that start with *A*
- '%a' : matches all strings that end with *a*
- '%a%' : matches all strings that contain an *a*

2. CREATE TABLE

The CREATE TABLE statement is used to create a new table. The format is:

CREATE TABLE *tablename*
(*column1 data type*,
column2 data type,
column3 data type);

- char(size): Fixed length character string.
- varchar(size): Variable-length character string. Max size is specified in parenthesis.
- number(size): Number value with a max number of columns specified in parenthesis
- date: Date value
- number(size,d): A number with a maximum number of digits of "size" and a maximum number of "d" digits to the right of the decimal

3. INSERT VALUES

Once a table has been created data can be inserted using INSERT INTO command.

INSERT INTO *tablename*
(col1, ... , coln)
VALUES (val1, ... , valn)

4. UPDATE
To change the data values in a pre existing table, the UPDATE command can be used.

UPDATE *tablename*

SET *colX = valX* [, *colY = valY*, ...]
WHERE *condition*

5. DELETE
The DELETE command can be used to remove a record(s) from a table.

DELETE FROM *tablename*
WHERE *condition*

To delete all the records from a table without deleting the table do

DELETE * FROM *tablename*

6. DROP
To remove an entire table from the database use the DROP command.

DROP TABLE *tablename*

7. ORDER BY
ORDER BY clause can order column name in either ascending (ASC) or descending (DESC) order.

ORDER BY *col_name* ASC

8. AND / OR
AND and OR can join two or more conditions in a WHERE clause. AND will return data when all the conditions are true. OR will return data when any one of the conditions is true.

9. IN
IN operator is used when you know the exact value you want to return for at least one of the columns

SELECT * FROM *table_name* WHERE *col_name* IN (*val1*, *val2*, ...)

10. BETWEEN / AND
The BETWEEN ... AND operator selects a range of data between two values. These values can be numbers, text, or dates.

SELECT * FROM *table_name* WHERE *col_name* BETWEEN *val1* AND *val2*

Implementation:

//PROGRAM FOR BASIC QUERY

CREATE TABLE QUERY: (DDL)

create table teit(rollno int,name char(30),percentage int)

output:

Table created.

INSERT QUERY: (DML)

insert into teit values(7,'POOJA',65)

OUTPUT:

1 row(s) inserted.

SELECT QUERY: (DML)

select*from teit

OUTPUT:

ROLLNO	NAME	PERCENTAGE
7	POOJA	65

ALTER QUERY: for adding column

alter table teit add (age int)

OUTPUT:

Table altered.

ROLLNO	NAME	PERCENTAGE	AGE
7	POOJA	65	-
10	ANUJA	62	-
24	RUTUJA	63	-

UPDATE QUERY: (DML)

update teit set age=23 where rollno=7

1 row(s) updated.

ROLLNO	NAME	PERCENTAGE	AGE
7	POOJA	65	23
10	ANUJA	62	21
24	RUTUJA	63	21

RENAME QUERY: (DDL)

Rename column name:-

select name as stname from teit

OUTPUT:

STNAME
POOJA
ANUJA
RUTUJA

Rename table name:-

rename teit to studentdb

OUTPUT:

Statement processed.

ALTER QUERY: for deleting column

alter table studentdb drop column age

OUTPUT:

Table dropped.

ROLLNO	NAME	PERCENTAGE
7	POOJA	65
10	ANUJA	62
24	RUTUJA	63

TRUNCATE QUERY:

truncate table studentdb

OUTPUT:

Table truncated.

DROP QUERY:

drop table studentdb

OUTPUT:

Table dropped.

Problem: Perform DDL & DML queries for Bank management system

Conclusion:
We have studied the implementation of DDL & DML commands for SQL.

Experiment No. 6

Title : Implementation of SQL Query constraints with referential integrity constraints.
Aim : To implement SQL Query constraints with referential integrity constraints

Theory :

- **PRIMARY KEY**: A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint create a unique index for accessing the table faster
- **UNIQUE**: The UNIQUE constraint in Mysql does not allow to insert a duplicate value in a column.
- **NOT NULL**: In Mysql NOT NULL constraint allows to specify that a column can not contain any NULL value.
- **FOREIGN KEY**: A FOREIGN KEY in mysql creates a link between two tables by one specific column of both table. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY.
- **CHECK**: The CHECK constraint determines whether the value is valid or not from a logical expression.
- **DEFAULT**: While inserting data into a table, if no value is supplied to a column, then the column gets the value set as DEFAULT

Implementation:

❖ Primary Key Constraint

- **Column Level**

```
create table cust(c_name char(20),c_id int primary key,c_add char(25))

insert into cust values('Mrunali',3,'Vaduj')

select *from cust
```

C_NAME	C_ID	C_ADD
Mrunali	3	Vaduj

Table Level:

```
create table cust1(c_name char(20),c_id int,c_add char(25),primary key(c_id,c_name))

insert into cust1 values('Sheetal',3,'Vaduj')

select *from cust1
```

C_NAME	C_ID	C_ADD
Sheetal	3	Vaduj

❖ Foreign Key

- **Column Level**

```
create table emp(c_name char(20),c_id int references cust,e_add char(25))
```

insert into emp values('Sheetal',3,'RAVIVAR PETH')

select *from emp

C_NAME	C_ID	E_ADD
Sheetal	3	RAVIVAR PETH

- **Table Level**

create table emp1(c_name char(20),c_id int,e_add char(25),foreign key(c_name,c_id)references cust1(c_name,c_id))

insert into emp1 values('Sheetal',3,'RAVIVAR PETH')

insert into emp1 values('Sheetal',3,'karve Road')

select *from emp1

C_NAME	C_ID	E_ADD
Sheetal	3	RAVIVAR PETH
Sheetal	3	karve Road

❖ **Unique Key**

- **Column Level**

create table acc1(acc_no int unique,branch_name char(10),bal int)

insert into acc1 values(null,'karad',2000)

select *from acc1

ACC_NO	BRANCH_NAME	BAL
-	karad	2000

- **Table Level**

create table acc2(acc_no int,branch_name char(10),bal int,unique(acc_no,branch_name)

insert into acc2 values(null,'karad',2000)

insert into acc2 values(3332,'karad',2200)

select *from acc2

ACC_NO	BRANCH_NAME	BAL
-	karad	2000
3332	karad	2200

❖ Null Value Concept

create table Branch(B_no varchar(10),B_name varchar(25))

insert into Branch(B_no,B_name)values('B1',null)

insert into Branch(B_no,B_name)values('B2','')

select *from Branch where B_name=''

select *from Branch where B_name is null

B_NO	B_NAME
B1	-
B1	-
B2	-

❖ Not Null Value Concept

create table account(acc_no int not null,branch_name char(10),bal int)

insert into account values(3332,'karad',2000)

insert into account values(4567,'satara',3000)

select *from account

ACC_NO	BRANCH_NAME	BAL
3332	karad	2000
4567	satara	3000

Check Constraint

- **Column Level-**

create table act(acc_no int,branch_name char(10)check(branch_name=upper(branch_name)),bal int)

insert into act values(3332,'KARAD',2000)

insert into act values(4567,'SATARA',5000)

select *from act

ACC_NO	BRANCH_NAME	BAL
3332	KARAD	2000
4567	SATARA	5000

- **Table Level-**

create table abc(acc_no int,branch_name char(10),bal int,c_name char(30),check(branch_name=upper(branch_name)),check(c_name=upper(c_name)))

insert into abc values(2222,'KARAD',1000,'ABC')

insert into abc values(3232,'SATARA',5000,'MABC')

select *from abc

ACC_NO	BRANCH_NAME	BAL	C_NAME
2222	KARAD	1000	ABC
3232	SATARA	5000	MABC

- **ON DELETE CASCADE**

create t able customer(cust_id int primary key,cust_name char(15),cust_addr char(20))

Table created.

insert into customer values(1,'sona','karad')

1 row(s) inserted.

insert into customer values(2,'mona','karad')

1 row(s) inserted.

insert into customer values(3,'anu','satara')

1 row(s) inserted.

insert into customer values(4,'ravi','pune')

1 row(s) inserted.

insert into customer values(5,'mahi','sangli')

1 row(s) inserted.

select *from customer

CUST_ID	CUST_NAME	CUST_ADDR
1	sona	karad
2	mona	karad
3	anu	satara
4	ravi	pune
5	mahi	sangli

create table cust(c_id int references customer(cust_id)ON DELETE CASCADE,c_name char(20),c_add char(30)

insert into cust values(1,'sona','karad')

insert into cust values(2,'sham','nagpur')

insert into cust values(3,'sam','solapur')

insert into cust values(4,'naina','kolhapur')

select *from cust

C_ID	C_NAME	C_ADD
1	sona	karad
2	sham	nagpur
3	sam	solapur
4	naina	kolhapur

delete from customer where cust_id=3

select *from customer

CUST_ID	CUST_NAME	CUST_ADDR
1	sona	karad
2	mona	karad
4	ravi	pune
5	mahi	sangli

select *from cust

C_ID	C_NAME	C_ADD
1	sona	karad
2	sham	nagpur
4	naina	kolhapur

- **ON DELETE SET NULL**

create table cust2(c_id int ,c_name char(20),c_add char(30),foreign key(c_id)references customer(cust_id)on delete set null)

Table created.
insert into cust2 values(2,'sham','nagpur')

1 row(s) inserted.
insert into cust2 values(4,'naina','kolhapur')
1 row(s) inserted.
select *from cust2

C_ID	C_NAME	C_ADD
2	sham	nagpur
4	naina	kolhapur

delete from cust2 where c_id=2
1 row(s) deleted.
select *from cust2

C_ID	C_NAME	C_ADD
4	naina	kolhapur

Problem statement: Implement all constraint to create database for college management System.

Conclusion:
Thus we have studied and implemented all the SQL Query constraints with referential integrity constraints

Experiment No. 7

Title: Implementation of SQL Query Processing: program to show use of SQL Clauses-Order by, group by and between clauses.

Aim: To implement SQL Query Processing: program to show use of SQL Clauses-Order by, group by and between clauses.

Theory:

- Write the syntax in SQL for following clauses: Order by, group by, having and between clauses

Implementation:

select * from dari

STUD_ROLLNO	STUD_NAME	ADDRESS
1	Daryappa	jaysingpur
2	pravin	pandarpur
3	Ganesh	phalthan

select * from sdev

ROLLNO	NAME	ID
12	abc	430
8	abc	220
9	amg	520
2	ass	250

Order By :-

1. Ascending

select rollno from sdev order by rollno

ROLLNO
2
8
9
12

2. Descending

select rollno from sdev order by rollno desc

ROLLNO
12
9
8
2

Group By :- having

select name, count(rollno)from sdev group by name;

NAME	COUNT(ROLLNO)
amg	1
abc	2

Between Clause:

Select rollno from sdev where ID between 200 and 400;

Problem statement:

1) Write a program which uses following SQL Commands with check constraint for employee payroll

a. Between
b. Distinct
c. Rollup

2) Write a program which uses following SQL Commands for employee payroll

a. Group by (using Having clause)

b. Order by

Conclusion:

We have studied and implemented the SQL Clauses-Order by, group by and between.

Experiment No. 8

Title : Implementation of SQL Query Aggregate functions & set operations
Aim : To Implement SQL Query Aggregate functions & set operations
Theory :
These functions are used to manipulating data items and returning the results.
1. Group functions or Aggregate functions.
2. Single Row or scalar function.
Group functions or Aggregate functions:
These functions operated a set of values or rows
i. Sum()
ii. Avg()
iii. Min()
iv. Max()
v. Count()

- Write down the syntax for at least 7 Aggregate functions & set operations

Implementation:

1. Create table

```
create table beit(roll_no int,name char(20),ph_no int)
```

ROLL_NO	NAME	PH_NO
15	vishwambhar	8805705733
30	ahwin	8805161129
40	jaydip	9561239599
55	ajinkya	96596235

2. AVG

```
select avg(roll_no)"Roll_no Avg" from beit
```

Roll_no Avg
35

3. MIN

```
select min(roll_no)"Roll_no Avg" from beit
```

Roll_no Avg
15

4. MAX

```
select max(roll_no)"Max Roll_no" from beit
```

Max Roll_no
55

5. COUNT

select count(ph_no)"Count Ph_no" from beit

Count Ph_no
4

6. SUM

select sum(ph_no)"SUM Ph_No" from beit

SUM Ph_No
27268702696

7. ABS

select abs(ph_no)"ABS Ph_No" from beit

ABS Ph_No
8805705733
8805161129
9561239599
96596235

8. POWER

select power(7,3)"POWER" from beit

POWER
343
343
343

9. ROUND

select round(34.56375,3)"ROUND" from beit

ROUND
34.564
34.564
34.564
34.564

10. SQRT

select sqrt(25)"ROOT" from beit

ROOT
5
5
5
5

11. EXP(n)

select exp(5)"EXPONENT" from beit

EXPONENT
148.413159102576603421115580040552279624
148.413159102576603421115580040552279624
148.413159102576603421115580040552279624
148.413159102576603421115580040552279624

12. GREATEST

select greatest(56,25)"GREATEST" from beit

GREATEST
56
56
56
56

13. LEAST

select least(56,25)"LEAST" from beit

LEAST
25
25
25
25

14. MOD(m,n)

select mod(56,5)"MOD" from beit

MOD
1
1
1
1

15. TRUNC

select trunc(56.53625,3)"TRUNC" from beit

TRUNC
56.536
56.536
56.536
56.536

16. FLOOR

select floor(56.53625)"floor" from beit

Floor
56
56
56
56

- **Set operations:**
  - **Queries:-**
  - select * from dari

STUD_ROLLNO	STUD_NAME	ADDRESS
1	Daryappa	jaysingpur
2	pravin	pandarpur
3	Ganesh	phalthan

- select * from sdev

ROLLNO	NAME	ID
12	abc	430
8	abc	220
9	amg	520
2	ass	250

Set operator

1. Union: If a={1,2,3}; b={2,4} then aUb: {1,2,3,4}

select rollno from sdev **union** select stud_rollno from dari

ROLLNO
1
2
3
8
9
12

2. Intersect: If a={1,2,3}; b={2,4} then a∩b: {2}

select rollno from sdev **intersect** select stud_rollno from dari

ROLLNO
2

3. Minus: (Set difference) If a={1,2,3}; b={2,4} then a-b: {1,3} & b-a: {4}

select rollno from sdev **minus** select stud_rollno from dari

ROLLNO
8
9
12

Problem statement: Create any database with application of all aggregate functionsand set operations

Conclusion:
We have studied and implemented all aggregate functions and set operations in the SQL database

Experiment No. 9

Title : Implementation of SQL query Join operations.

Aim : To implement SQL query Join operations

Theory/ Design :

- Explain the concept of join and its types in detail

MySQL JOINS are used to retrieve data from multiple tables. A MySQL JOIN is performed whenever two or more tables are joined in a SQL statement. There are different types of MySQL joins:

- MySQL INNER JOIN (or sometimes called simple join)
- MySQL NATURAL JOIN
- MySQL CROSS JOIN
- MySQL FULL OUTER JOIN
- MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

INNER JOIN (simple join) MySQL INNER JOINS return all rows from multiple tables where the join condition is met. Syntax:

SELECT columns From table1 Inner join table2 On table1.column=table2.column;

LEFT OUTER JOIN Another type of join is called a MySQL LEFT OUTER JOIN. This type of join returns all rows from the LEFT-hand table specified in the ON condition and only those rows from the other table where the joined fields are equal.

Syntax:

SELECT columns From table1 left join table2 On table1.column=table2.column;

RIGHT OUTER JOIN Another type of join is called a MySQL RIGHT OUTER JOIN. This type of join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where the joined fields are equal.

Syntax: SELECT columns From table1 Right join table2 On table1.column=table2.column;

Implementation:

Queries:-

select * from stud

STUD_ROLLNO	STUD_NAME	ADDRESS
1	Kishor	thane
2	rohit	dadar
3	Siddhant	afrika

select * from cust1

CUST_ID	CUST_NAME	ADDRESS
1	bhavesh	Kalyan
2	amol	karad
3	Kishor	thane

Inner Join

select * from stud inner join cust1 on stud.stud_name=cust1.cust_name

STUD_ROLLNO	STUD_NAME	ADDRESS	CUST_ID	CUST_NAME	ADDRESS
1	Kishor	thane	3	Kishor	thane

Natural Join

select * from stud natural join cust1

ADDRESS	STUD_ROLLNO	STUD_NAME	CUST_ID	CUST_NAME
thane	1	Kishor	3	Kishor

Cross Join

select * from stud cross join cust1

STUD_ROLLNO	STUD_NAME	ADDRESS	CUST_ID	CUST_NAME	ADDRESS
1	Kishor	thane	1	bhavesh	Kalyan
1	Kishor	thane	2	amol	karad
1	Kishor	thane	3	Kishor	thane
2	rohit	dadar	1	bhavesh	Kalyan
2	rohit	dadar	2	amol	karad
2	rohit	dadar	3	Kishor	thane
3	Siddhant	afrika	1	bhavesh	Kalyan
3	Siddhant	afrika	2	amol	karad
3	Siddhant	afrika	3	Kishor	thane

Outer Join

1. **Full outer join**

select * from stud full outer join cust1 on stud.stud_name=cust1.cust_name

STUD_ROLLNO	STUD_NAME	ADDRESS	CUST_ID	CUST_NAME	ADDRESS
1	Kishor	thane	3	Kishor	thane
3	Siddhant	afrika	-	-	-
2	rohit	dadar	-	-	-
-	-	-	2	amol	karad
-	-	-	1	bhavesh	Kalyan

2. Right outer join

select * from stud right outer join cust1 on stud.stud_name=cust1.cust_name

STUD_ROLLNO	STUD_NAME	ADDRESS	CUST_ID	CUST_NAME	ADDRESS
1	Kishor	thane	3	Kishor	thane
-	-	-	2	amol	karad
-	-	-	1	bhavesh	Kalyan

3. Left outer join

select * from stud left outer join cust1 on stud.stud_name=cust1.cust_name

STUD_ROLLNO	STUD_NAME	ADDRESS	CUST_ID	CUST_NAME	ADDRESS
1	Kishor	thane	3	Kishor	thane
3	Siddhant	afrika	-	-	-
2	rohit	dadar	-	-	-

Problem statement: Implement all joins for University management system.

Conclusion:
We have studied and implemented all SQL query Join operations

Experiment No. 10

Title : Implementation of String operations in SQL.

Aim : To implement String operations in SQL

Theory/ Design:

- Write the syntax for all string related operations

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;
LXV

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;
SMITH

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.

Implementation:
select lower ('kishor')"lower" from dual

lower
kishor

select upper ('kishor')"capittalised" from dual

capittalised
KISHOR

select substr('sekishor',3,6)"substring" from dual

substring
kishor

select translate('1 set523','143','7ag')"change" from dual

change
7 set52g

select initcap('DACOE')"titlecase" from dual

titlecase
Dacoe

select ASCII('DACOE')"ASCII" from dual

ASCII
68

select length('DACOE')"length" from dual

length
5

select lpad('DACOE',9,'*')"lpad" from dual

lpad
****DACOE

select rpad('DACOE',9,'*')"rpad" from dual

rpad
DACOE****

select ltrim('DACOE','D')"ltrim" from dual

ltrim
ACOE

select rtrim('DACOE','E')"rtrim" from dual

rtrim
DACO

select concat('RA','-ONE')"concate" from dual

concate
RA-ONE

Problem statement: Create database of employee payroll system and implement all string and date functions on the database.

Conclusion: We have studied and implemented all string and date functions in the SQL database

Experiment No. 11

Title : Implementation of Views for any created table

Aim : To understand and implement views in SQL table

Theory/ Design:

A view is the tailored presentation of data contained in one or more table and can also be said as restricted view to the data in the tables. A view is a "virtual table" or a "stored query" which takes the output of a query and treats it as a table. The table upon which a view is created is called as base table. A view is a logical table based on a table or another view. A view contains no data of its own but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called base tables. The view is stored as a SELECT statement in the data dictionary.

Advantages of a view:

a. Additional level of table security.
b. Hides data complexity.
c. Simplifies the usage by combining multiple tables into a single table

Creating and dropping view:

Syntax:

Create or replace view view_name AS SELECT column_name(s) FROM table_name
WHERE condition;
Drop view <view name>;

Example:

create table tv9(mno varchar2(50),brand varchar2(50),price varchar2(50));
Table created.

insert into tv9 values('ms78','samsung','10000');
1 row(s) inserted.

insert into tv9 values('89ru','lg','34000');
1 row(s) inserted.

insert into tv9 values('nve9','sharp','12000');
1 row(s) inserted.

insert into tv9 values('rusk9','onida','17000');
1 row(s) inserted.

select * from tv9

MNO	BRAND	PRICE
ms78	samsung	10000
89ru	lg	34000
nve9	sharp	12000
rusk9	onida	17000

4 rows returned in 0.05 seconds

create view tv9view AS
SELECT brand,price
FROM tv9
WHERE brand IS NOT NULL
AND price IS NOT NULL;

View created.

select*from tv9view

BRAND	PRICE
samsung	10000
lg	34000
sharp	12000
onida	17000

4 rows returned in 0.01 secor

- **Insert:**

INSERT INTO tv9view
(brand,price)
VALUES
('toshiba','8000');

1 row(s) inserted.
select*from tv9view

BRAND	PRICE
samsung	10000
lg	34000
sharp	12000
onida	17000
toshiba	8000

5 rows returned in 0.00 seconds

select * from tv9

MNO	BRAND	PRICE
ms78	samsung	10000
89ru	lg	34000
nve9	sharp	12000
rusk9	onida	17000
-	toshiba	8000

5 rows returned in 0.00 seconds

- **Update**

UPDATE tv9view
SET price = price + 10
where brand='toshiba'

1 row(s) updated.

select*from tv9view

BRAND	PRICE
samsung	10000
lg	34000
sharp	12000
onida	17000
toshiba	8010

5 rows returned in 0.00 seconds

select * from tv9

MNO	BRAND	PRICE
ms78	samsung	10000
89ru	lg	34000
nve9	sharp	12000
rusk9	onida	17000
-	toshiba	8010

5 rows returned in 0.01 seconds

- **Delete**

delete from tv9view where brand='toshiba'

1 row(s) deleted.

select*from tv9view

BRAND	PRICE
samsung	10000
lg	34000
sharp	12000
onida	17000

4 rows returned in 0.00 seconds

Problem statement:

Write the queries for the following

(i) Create a view emp from employee such that it contains only emp_no and emp_name and department.
(ii) Create a view dept from department with only dept_no and location.
(iii) Create a view that contains the details of employees who are managers only.
(iv) drop the views.

Conclusion: We have studied and implemented views with basic operations in the SQL database

Experiment No. 12

Title : Implementation of Hashing Technique on the created data.

Aim : To study and implement Hashing technique.

Theory:

Hashing is one of the way to store the Data into database. In the Hashing one hash function is there that will find address of the bucket where we have to store the particular record after calculation on the particular search key.
There are two types:

1) Static Hashing.

2) Dynamic Hashing.

1) Static Hashing:

In hash file organization we obtain the address of the disk block containing the desired record directly by computing a function on the search key value of record.

Let K denotes the set of all search key values & B denotes the set of all bucket address. A hash function h is a function from K to B Let h denotes a hash function.

To insert a record with search key K we compute h(K:) which gives the address of the bucket for that record.

Hash Functions:

Hash functions maps all search key values to the same bucket Ideal hash function distributes the stored key uniformly across all the buckets so that every bucket has some no. of records.

Handling of bucket overflows:

If the bucket does not have enough space a bucket overflow is said to occur bucket overflow can occur for several reasons.

a) Insufficient Buckets:

The no. of buckets which we denote must be chosen such that n b>h r|f r where nr denotes the total will be stored & f r denotes the no. of records that will fit in a bucket.

b) Skew:

Some buckets are assigns more records than are others so a bucket may overflow even when other bucket still have specials situation is called bucket skew.

c) Hash Indices:

A hash index organizes the search keys their associated pointers into a hash file structure.

2) Dynamic Hashing :

Several dynamic hashing techniques allows the hash functions to be modified dynamically to accurate the growth or shrinkage of database bucket address table.

Hashing is having following advantages-:

1) File Organization is based on the technique of hashing allow us to avoid accessing the index structure.
2) Hashing Provides a way to construct the index.
3) Here we obtain the address by calculating the hash function on the record using any desirable formula.
4) We can insert records using this hash index search key.
5) To insert we follow same procedure as of b –trees.
6) To Delete a record we also do look up and then searching it is to be deleted form file.
7) Store all The records in a file and store their indices block wise.

Example:

```
// CPP Program For Hashing

#include<iostream.h>

#include<fstream.h>

#include<conio.h>

#include<stdio.h>

#include<string.h>

class hash

{

public:

int lo_no,amt;

char name[20];

void get()

{

cout<<"\n Enter the name:";

cin>>name;

cout<<"\n Enter the loan no:";

cin>>lo_no;
```

```
cout<<"\n Enter the amount:";
cin>>amt;
}
void show()
{
cout<<"\n\n\n Name\tloan no\t Amount";
cout<<name<<"\t"<<lo_no<<"\t"<<amt<<endl;
}
}m,n;
int hash_fn(int);
void main()
{
fstream fp,fp1;
hash m,n;
int no,c;
char ch[50]="hash1";
clrscr();
cout<<"\n1:Enter the Record \n2:Serach the Record";
cout<<"\n Enter the choose:";
cin>>c;
if(c==1)
{
cout<<endl;
m.get();
n=m;
no=hash_fn(m.lo_no);
if(no==0)
{
```

```
strcat(ch,"0.txt");
}
if(no==1)
{
strcat(ch,"1.txt");
}
if(no==2)
{
strcat(ch,"2.txt");
}
if(no==3)
{
strcat(ch,"3.txt");
}  if(no==4)
{
strcat(ch,"4.txt");
}
if(no==5)
{
strcat(ch,"5.txt");
}
fp.open(ch,ios::app);
fp.write((char*)&n,sizeof(n));
fp.close();
}
if(c==2)
{
cout<<"Enter Laon no for searching:";
```

```
cin>>m.lo_no;
n.lo_no=m.lo_no;
no=hash_fn(m.lo_no);
if(no==0)
{
strcat(ch,"0.txt");
}
if(no==1)
{
strcat(ch,"1.txt");
}
if(no==2)
{
strcat(ch,"2.txt");
}
if(no==3)
{
strcat(ch,"3.txt");
}
if(no==4)
{
strcat(ch,"4.txt");
}
if(no==5)
{
strcat(ch,"5.txt");
}
fp1.open(ch,ios::in);
```

```
if(fp!=NULL)
{
while(fp1.read((char*)&m,sizeof(m)))
{
fp1.read((char*)&m,sizeof(m));
cout<<"in loop";
if(n.lo_no==m.lo_no)
{
m.show();
}}
fp.close();
}
else
{
cout<<"file not found";
}}
getch();
}
hash_fn(int no)
{
int k;
k=no%5;
return(k);
}
/*OUTPUT */
1:Enter the Record
2:Serach the Record
Enter the choose:1
```

Enter the name: rohit

Enter the loan no:11

Enter the amount:500

1:Enter the Record

2:Serach the Record

Enter the choose:2

Enter Laon no for searching:11

in loop

Name loan no Amount rohit 11 500

Problem statement: Write a java program to implement the hashing technique

Conclusion:

Thus we can insert & display data after using Hash indices by calculating Hash on the Records

Questions:

1. **What is Indexing and B+ tree index file?**
2. **What is Static Hash Functions?**
3. **What is Dynamic Hash Functions?**
4. **What is difference between Indexing and Hashing?**

Experiment No. 13

Title : Study of NoSQL

Aim : To study and implement NoSQL

Theory/Design:

- Explain NoSQL database in detail.
  Explain features of NoSQL database
- A comparison of different NoSQL databases (w.r.t. Database Tool, Data model, Transaction model, Query support)

- **Key-value store:** In these databases all the stored data is represented by a pair of key and value per record, meaning that each key is unique and it allows accessing record‟s This structure is also known as "hash table" where data retrieval is usually performed by using key to access value.
- **Document Store.** These databases are designed to manage data stored in documents that use different format standards, such as XML or JSON. This type of storage is more complex in comparison to storage used by Key-value Stores and enables data querying.
- **Column Family.** The database structure of this NoSQL type is similar to the standard Relational Database Management System (RDMS) since all the data is stored as sets of columns and rows. Columns, that store related data that is often retrieved together, may be grouped.
- **Graph Database.** These databases are mostly used when the stored data may be represented as a graph with interlinked elements such as, social networking, road maps or transport routes.

Conclusion: We have studied and understand the NoSQL databases and its various formats