SAI VIDYA INSTITUTE OF TECHNOLOGY

(Affiliated to VTU, Belagavi, Approved by AICTE, New Delhi and Govt. of Karnataka)
Accredited by NBA, New Delhi (CSE, ISE, ECE, MECH, CIVIL), NAAC-'A' Grade
Rajanukunte, Bengaluru – 560 064
Tel: 080-2846 8196, email: hodcse@saividya.ac.in web: www.saividya.ac.in

MOTTO

"Learn to lead"

VISION

Contribute dedicated, skilled, intelligent engineers and business administrators to architect strong India and the world.

MISSION

- *To provide quality education and skilled training to produce dedicated engineers and managers.*
- *To promote research, innovation, and ethical practices by creating supportive environment.*
- *To undertake collaborative projects with academia and industry that transform young minds into Socially Responsible Citizen and Globally Competent Professionals.*
- *To enhance personality traits which lead to entrepreneurship qualities among the students.*

DATABASE MANAGEMENT SYSTEM
[BCS403]

(As per Visvesvaraya Technological University Syllabus)

Compiled by:

Dr. Vinod Desai
Associate Professor
Dept. of CSE

Dr. Varun E
Associate Professor
Dept. of CSE

Name:_____

USN :_____

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

2023-24

Disclaimer

Trademark

Edition: 2023-24

Document Owners

The primary contacts for questions regarding this document are:

Authors: **1. Dr. Vinod Desai**

2. Dr. Varun E.

Department: **Computer Science & Engineering**

Contact email id:
1. vinod.desai@saividya.ac.in

2. varun.e@saividya.ac.in

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DEPARTMENT VISION

Contribute dedicated, skilled, intelligent Computer Engineers to architect strong India and the world.

DEPARTMENT MISSION

- *To promote activities that imparts high quality technical education in core domains of computer engineering and also interdisciplinary areas.*
- *To imbibe innovative projects and research based skills using software tools to solve real time problems for building modern India and the world.*
- *To create a conducive environment for lifelong learning with ethical responsibilities to produce globally competent Computer professionals and entrepreneurs.*

PROGRAM EDUCATIONAL OBJECTIVES

.

- *PEO 1: Graduates will have the expertise in analyzing real time problems and providing appropriate solutions related to Computer Science & Engineering.*
- *PEO 2: Graduates will have the knowledge of fundamental principles and innovative technologies to succeed in higher studies, and research.*
- *PEO 3: Graduates will continue to learn and to adapt technology developments combined with deep awareness of ethical responsibilities in profession.*

Program Outcomes

1 **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2 **Problem Analysis**: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3 **Design/Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4 **Conduct Investigations of Complex Problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5 **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6 **The Engineer and Society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7 **Environment and Sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8 **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9 **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10 **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11 **Project Management and Finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12 **Life-Long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Outcomes

PSO-1: Demostrate the knowledge and understating of working principles, design, implement, test and evaluate the hardware and software components of a computer system.

PSO-2: Apply standard Software Engineering practices and strategies project development.

PSO-3: Demonstrate the knowledge of Discrete Mathematics, Data management and Data Engineering.

Experiment 1

Create a table called Employee & execute the following.

Employee (EMPNO,ENAME,JOB, MANAGER_NO, SAL, COMMISSION)

1. Create a user and grant all permissions to the user.

2. Insert the any three records in the employee table contains attributes EMPNO,ENAME JOB, MANAGER_NO, SAL, COMMISSION and use rollback. Check the result.

3. Add primary key constraint and not null constraint to the employee table.

4. Insert null values to the employee table and verify the result

Solution

1. Creating the table and granting permissions:
   - Creating a new user

     SQL>CREATE USER user_name IDENTIFIED BY password;
   - Granting all permissions to the new user

     SQL>GRANT ALL PRIVILEGES ON to user_name;

- Creating the Employee table

```
SQL> CREATE TABLE Employee (
                        EMPNO INT,
                        ENAME VARCHAR(50),
                        JOB VARCHAR(50),
                        MANAGER_NO INT,
                        SAL DECIMAL(10, 2),
                        COMMISSION DECIMAL(10, 2)
                       );
```

2. Inserting three records into the Employee table and rolling back:
   - Inserting records:

     SQL>INSERT INTO Employee values(1, 'John Doe', 'Manager', 101, 20000.00, 2000.00);

     SQL>INSERT INTO Employee values(2, 'Jane Smith', 'Developer', 102, 45000.00, 1500.00);

SQL>INSERT INTO Employee values(3, 'Alice Johnson', 'Analyst', 103, 40000.00, 1000.00);

- Rolling back the transaction

SQL>ROLLBACK;

3. Adding primary key and not null constraints:

- Adding primary key constraint

SQL>ALTER TABLE Employee ADD CONSTRAINT PK_Employee_EMPNO PRIMARY KEY (EMPNO);

- Adding not null constraints

SQL>ALTER TABLE Employee MODIFY (EMPNO INT NOT NULL,
ENAME VARCHAR(50) NOT NULL,
JOB VARCHAR(50) NOT NULL,
SAL DECIMAL(10, 2) NOT NULL);

4. Inserting NULL values and verifying the result:

- Inserting NULL values

SQL>INSERT INTO Employee VALUES (4, NULL, 'Intern', NULL, 30000.00, NULL);

- Verifying the result

SELECT * FROM Employee;

Experiment 2

Create a table called Employee that contain attributes EMPNO, ENAME, JOB, MGR, SAL & execute the following.

1. **Add a column commission with domain to the Employee table.**
2. **Insert any five records into the table.**
3. **Update the column details of job**
4. **Rename the column of Employ table using alter command.**
5. **Delete the employee whose Empno is 105.**

Solution:

SQL>CREATE TABLE Employee (

EMPNO INT PRIMARY KEY,

ENAME VARCHAR(50),

JOB VARCHAR(50),

MGR INT,

SAL DECIMAL(10, 2));

2. Insert five records into the table

SQL>INSERT INTO Employee VALUES (101, 'John Doe', 'Manager', NULL, 5000.00);

SQL>INSERT INTO Employee VALUES (102, 'Jane Smith', 'Salesperson', 101, 3000.00);

SQL>INSERT INTO Employee VALUES (103, 'Michael Johnson', 'Salesperson', 101, 3200.00);

SQL>INSERT INTO Employee VALUES (104, 'Emily Brown', 'Clerk', 102, 2500.00);

SQL>INSERT INTO Employee VALUES (105, 'David Williams', 'Analyst', 101, 4500.00);

3. Update the column details of job

SQL> UPDATE Employee SET JOB = 'Senior Salesperson' WHERE EMPNO = 103;

4. Rename the column of Employee table using ALTER command

SQL>ALTER TABLE Employee

RENAME COLUMN EMPNO TO Employee_ID;

5. Delete the employee whose Empno is 105

```
SQL> DELETE FROM Employee
 WHERE Employee_ID = 105;
```

Experiment 3

Queries using aggregate functions (COUNT,AVG,MIN,MAX,SUM),Group by,Orderby.

Employee(E_id, E_name, Age, Salary)

1. Create Employee table containing all Records E_id, E_name, Age, Salary.

2. Count number of employee names from employeetable

3. Find the Maximum age from employee table.

4. Find the Minimum age from employeetable.

5. Find salaries of employee in Ascending Order.

6. Find grouped salaries of employees.

1. Create Employee table containing all Records E_id, E_name, Age, Salary.

```
SQL> CREATE TABLE Employee (
                                E_id INT PRIMARY KEY,
                                E_name VARCHAR(25),
                                Age INT,
                                Salary DECIMAL(10, 2)
                            );
SQL> insert into Employee values(101,'Shravya',19,20000);
SQL> insert into Employee values(102,'keerthi',32,15000);
SQL> insert into Employee values(103,'bhuvan',18,30000);
SQL> insert into Employee values(104,'akash',55,50000);
SQL> insert into Employee values(105,'Sumanth',20,20000);
```

2. Count number of employee names from employee table:

```
SELECT COUNT(E_name)  FROM Employee;
```

3. Find the Maximum age from employee table:

```
SELECT MAX(Age) FROM Employee;
```

4. Find the Minimum age from employee table:

```
SELECT MIN(Age) FROM Employee;
```

5. Find salaries of employees in Ascending Order:

```
SELECT E_name, Salary FROM Employee ORDER BY Salary ASC;
```

6. Find grouped salaries of employees:

```
SELECT Salary, COUNT(*) AS Num_of_Employees
FROM Employee GROUP BY Salary;
```

Experiment 4

Create a row level trigger for the customers table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old & new Salary.

CUSTOMERS(ID,NAME,AGE,ADDRESS,SALARY)

Solution

```
SQL>create table CUSTOMERS (
                                        ID INT PRIMARY KEY,
                                       NAME varchar2(20) NOT NULL,
                                        AGE INT,
                                       ADDRESS varchar2(25) ,
                                       SALARY Decimal);
insert into CUSTOMERS values (111, 'kavana', 20 , 'Bengaluru', 29000)
SQL>insert into CUSTOMERS values (222, 'Darshan', 35 , 'Hassan', 15000)

SQL> CREATE OR REPLACE TRIGGER salary_difference_trigger
        AFTER INSERT OR UPDATE OR DELETE ON CUSTOMERS
        FOR EACH ROW
         DECLARE
              old_salary NUMBER;
            new_salary NUMBER;
BEGIN
  IF INSERTING THEN
    DBMS_OUTPUT.PUT_LINE('New record inserted. Salary: ' || :NEW.SALARY);
  ELSIF UPDATING THEN
    old_salary := :OLD.SALARY;
    new_salary := :NEW.SALARY;
    DBMS_OUTPUT.PUT_LINE('Record updated. Old Salary: ' || old_salary || ', New Salary: '
|| new_salary);
    DBMS_OUTPUT.PUT_LINE('Salary Difference: ' || (new_salary - old_salary));
```

```
    ELSIF DELETING THEN
        DBMS_OUTPUT.PUT_LINE('Record deleted. Salary: ' || :OLD.SALARY);
    END IF;
END;
/
```

Trigger created.

Experiment 5

Create cursor for Employee table & extract the values from the table. Declare the variables, Open the cursor & extract the values from the cursor. Close the cursor.

Employee (E_id, E_name, Age, Salary)

Solution

```
CREATE TABLE Employee (
   E_id INT PRIMARY KEY,
   E_name VARCHAR(255),
   Age INT,
   Salary DECIMAL(10, 2)
);

-- Declare variables
DECLARE
   E_id Employee.E_id%TYPE;
   E_name Employee.E_name%TYPE;
   Age Employee.Age%TYPE;
   Salary Employee.Salary%TYPE;

-- Declare cursor
CURSOR employee_cursor IS
   SELECT E_id, E_name, Age, Salary
   FROM Employee;

-- Open the cursor
BEGIN
   OPEN employee_cursor;

   -- Fetch data from cursor
   LOOP
```

```
        FETCH employee_cursor INTO E_id, E_name, Age, Salary;
        EXIT WHEN employee_cursor%NOTFOUND;

        -- Output or use the fetched values
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || E_id || ', Name: ' || E_name || ', Age: ' ||
Age || ', Salary: ' || Salary);
    END LOOP;

    -- Close the  cursor
    CLOSE employee_cursor;
END;
```

Experiment 6

Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

Solution

```
DECLARE
   v_count NUMBER;
   CURSOR c_new_rollcall IS
      SELECT id, name, roll
      FROM N_RollCall2;
BEGIN
   FOR new_rec IN c_new_rollcall LOOP
      -- Check if the record already exists in O_RollCall2
      SELECT COUNT(*)
      INTO v_count
      FROM  O_RollCall2
      WHERE  id = new_rec.id;

      -- If record doesn't exist, insert it
      IF v_count = 0 THEN
         INSERT INTO O_RollCall2 (id, name, roll)
         VALUES (new_rec.id, new_rec.name, new_rec.roll);
         DBMS_OUTPUT.PUT_LINE('Record inserted: ' || new_rec.id);
      ELSE
         DBMS_OUTPUT.PUT_LINE('Record skipped: ' || new_rec.id);
      END IF;
   END LOOP;
   COMMIT;
```

```
END;

select * from N_RollCall2;
select * from O_RollCall2;

INSERT INTO N_RollCall2 (id, name, roll)
VALUES (1121, 'satya12333', 111111111);


delete from o_rollcall2 where id=121;
```

Experiment no: 7

Install an Open Source NoSQL Data base MangoDB & perform basic CRUD(Create, Read, Update & Delete) operations. Execute MangoDB basic Queries using CRUD operations.

Solution

Installation:

- Download MongoDB:

  Visit the official MongoDB website and download the MongoDB Community Server according to your operating system.

- Install MongoDB:

Follow the installation instructions provided for your operating system.

- Start MongoDB:

After installation, start the MongoDB server. On most systems, you can do this by running mongod in your terminal or command prompt.

Basic CRUD Operations:

1. Connect to MongoDB:

   Open a new terminal or command prompt window and run the mongo command to open the MongoDB shell.

2. Create Database:

   To create a new database, use the use command followed by the database name.

   For example: use mydatabase

3. Create Collection:

   Collections in MongoDB are analogous to tables in relational databases. To create a collection, you can simply start adding documents to it. MongoDB will create the

collection automatically when you insert the first document.

4. Insert Document (Create Operation):

To insert a document into a collection, use the insertOne or insertMany method. For example:

db.collectionName.insertOne({ key: value })

5. Read Document (Read Operation):

To retrieve documents from a collection, use the find method. For example:

db.collectionName.find()

6. Update Document (Update Operation):

To update a document in a collection, use the updateOne or updateMany method.

For example: db.collectionName.updateOne({ key: value }, { $set: { key: newValue } })

7. Delete Document (Delete Operation):

To delete a document from a collection, use the deleteOne or deleteMany method.

For example: db.collectionName.deleteOne({ key: value })