**Roll No…………….. Total No. of Pages:……**

**ST-2 (SET-II)**

**4th SEMESTER 2023-24**

**22CS007- Database Management System**

**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. What is the primary purpose of a stored procedure in DBMS?
   1. To store and organize data in a database
   2. To retrieve data from the database
   3. To define the structure of the database
   4. **To encapsulate a series of database operations**
2. Which of the following statements is true about stored procedures?
   1. Stored procedures cannot have input parameters
   2. Stored procedures cannot return values
   3. **Stored procedures can be reused and shared by multiple applications**
   4. Stored procedures can only be executed by the database administrator
3. Which of the following is not a type of trigger in DBMS?
   1. Insert trigger
   2. Update trigger
   3. Delete trigger
   4. **Search trigger**
4. Which component of a package in DBMS defines the interface and public entities?
   1. Package body
   2. **Package specification**
   3. Package constructor
   4. Package signature
5. What is the purpose of a package constructor in DBMS?
   1. To define the implementation of the package's procedures and functions
   2. **To initialize the package's variables and state**
   3. To provide an entry point for executing the package's functionality
   4. To create instances of the package for concurrent usage
6. The property of a schedule that states that the result of executing concurrent transactions is the same as executing them serially is known as:
   1. Consistency
   2. Atomicity
   3. **Serializability**
   4. Durability
7. Which of the following is not a concurrency control mechanism in DBMS?
   1. Locking
   2. Timestamp ordering
   3. Multiversion concurrency control
   4. **Rollback and recovery**
8. Which parameter mode is used when the procedure needs to both receive and send data to the calling program?
   1. IN
   2. OUT
   3. **INOUT**
   4. REF
9. Which parameter mode is used when the procedure needs to pass a reference to a variable?
   1. IN
   2. OUT
   3. INOUT
   4. **REF**
10. The \_\_\_\_\_\_\_\_\_\_ Statement is used for creating the package body.
    1. CREATE
    2. CREATE PACKAGE
    3. CREATE BODY
    4. **CREATE PACKAGE BODY**

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

1. Which of the following are the advantages of PL/SQL Packages?
   1. Modularity
   2. Easier Application Design
   3. Information Hiding
   4. Added Functionality,Better Performance
   5. **All mentioned above**
2. Under which circumstance do you design database triggers?
   1. To duplicate the functionality of other triggers.
   2. To replicate built-in constraints in the Oracle server such as primary key and foreign key.
   3. **When a specific operation is performed, related actions are performed.**
   4. None of the above
3. Which of the following is used to declare a record?
4. **%ROWTYPE**
5. %TYPE
6. Both A & B
7. None of the above
8. Which of the following pass parameters can be referenced by procedure?
9. IN, OUT
10. **IN, INOUT**
11. OUT, INOUT
12. None of the Above
13. What is a condition predicate in a DML trigger?
14. Specify a WHEN-LOGGING-ON condition in the trigger body.
15. Use the NEW and OLD qualifiers in the trigger body as a condition.
16. **Combine several DBM triggering events into one in the trigger body.**
17. Specify a SHUTDOWN or STARTUP condition in the trigger body.

**SECTION-C(Coding Question) (4x5 marks=20 marks)**

1. Write a PL/SQL program that creates a procedure which have parameters with default value.

Solution:

**declare**

**procedure p\_print**

**(i\_str1\_tx VARCHAR2 :='hello',**

**i\_str2\_tx VARCHAR2 :='world',**

**i\_end\_tx VARCHAR2 :='!' ) is**

**begin**

**DBMS\_OUTPUT.put\_line(i\_str1\_tx||','**

**||i\_str2\_tx||i\_end\_tx);**

**end;**

**begin**

**p\_print('Hi','there','all'); -- both parameters**

**p\_print('Hi','people'); -- without the last**

**p\_print('Hi'); -- only the first**

**p\_print(); -- no parameters**

**p\_print; -- no parenthesis**

**end;**

**/**

1. Write a PL/SQL program that delete an employee based on their employee\_id.

Solution:

**-- Create the "employee" table**

**CREATE TABLE employee (**

**employee\_id NUMBER PRIMARY KEY,**

**first\_name VARCHAR2(50),**

**last\_name VARCHAR2(50),**

**department VARCHAR2(50),**

**salary NUMBER**

**);**

**-- Insert sample records into the "employee" table**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (1, 'John', 'Doe', 'HR', 50000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (2, 'Jane', 'Smith', 'Finance', 60000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (3, 'Michael', 'Johnson', 'IT', 70000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (4, 'Merry', 'Agarwal', 'IT', 50000);**

**CREATE OR REPLACE PROCEDURE delete\_employee\_by\_id(p\_employee\_id NUMBER) AS**

**BEGIN**

**DELETE FROM employee**

**WHERE employee\_id = p\_employee\_id;**

**IF SQL%ROWCOUNT > 0 THEN**

**DBMS\_OUTPUT.PUT\_LINE('Employee deleted successfully.');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE('Employee ID not found. No employee deleted.');**

**END IF;**

**EXCEPTION**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE('An error occurred.');**

**END;**

**/**

1. Write a PL/SQL program that Create Trigger to prevent deletion of employees with a salary greater than 90000.

Solution:

**-- Create the "employee" table**

**CREATE TABLE employee (**

**employee\_id NUMBER PRIMARY KEY,**

**first\_name VARCHAR2(50),**

**last\_name VARCHAR2(50),**

**department VARCHAR2(50),**

**salary NUMBER**

**);**

**-- Insert sample records into the "employee" table**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (1, 'John', 'Doe', 'HR', 50000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (2, 'Jane', 'Smith', 'Finance', 60000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (3, 'Michael', 'Johnson', 'IT', 70000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (4, 'Merry', 'Agarwal', 'IT', 95000);**

**CREATE OR REPLACE TRIGGER trg\_prevent\_employee\_deletion**

**BEFORE DELETE ON employee**

**FOR EACH ROW**

**BEGIN**

**IF :old.salary > 90000 THEN**

**RAISE\_APPLICATION\_ERROR(-20003, 'Employees with a salary greater than 90,000 cannot be deleted.');**

**END IF;**

**END;**

**/**

**DELETE FROM employee where id=4;**

1. Write a PL/SQL program to create Package with a procedure to update the department of an employee.

Solution:

**-- Create the "employee" table**

**CREATE TABLE employee (**

**employee\_id NUMBER PRIMARY KEY,**

**first\_name VARCHAR2(50),**

**last\_name VARCHAR2(50),**

**department VARCHAR2(50),**

**salary NUMBER**

**);**

**-- Insert sample records into the "employee" table**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (1, 'John', 'Doe', 'HR', 50000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (2, 'Jane', 'Smith', 'Finance', 60000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (3, 'Michael', 'Johnson', 'IT', 70000);**

**INSERT INTO employee (employee\_id, first\_name, last\_name, department, salary)**

**VALUES (4, 'Merry', 'Agarwal', 'IT', 20000);**

**CREATE OR REPLACE PACKAGE employee\_mgmt AS**

**PROCEDURE update\_employee\_department(employee\_id NUMBER, new\_department VARCHAR2);**

**END employee\_mgmt;**

**/**

**CREATE OR REPLACE PACKAGE BODY employee\_mgmt AS**

**PROCEDURE update\_employee\_department(employee\_id NUMBER, new\_department VARCHAR2) IS**

**BEGIN**

**UPDATE employee**

**SET department = new\_department**

**WHERE employee\_id = employee\_id;**

**COMMIT;**

**END;**

**END employee\_mgmt;**

**/**