Savitribai Phule Pune University

S.Y. B.C.A. (Science) (Semester-III) Practical Examination

BCA 235: s(Database Management Systems II Laboratory)

Duration: 3Hrs. **Max Marks: 35+15=50**

Note: - 1. Read the questions carefully and insert data in the database accordingly.
2. Insert sufficient number of records in the database.
3. No query should generate empty output.
4. For count queries output should be more than 2 records. (If asked)

Create the following database in 3NF using PostgresSQL. [Total Marks: 10]

Q1) Consider the following database of Movie_Actor_Producer.
Movie (m_name varchar (25), release_year integer, budget money)
Actor (a_name char (30), city varchar(30))
Producer (producer_id integer, pname char (30), p_address varchar (30))

Relationship:

Movie and Actor related with many-to-many relationship with descriptive attributes role and charges. Producer and Movie related with many-to-many relationship.

Constraints: Primary key, release_year should not be null.

Create a View: **[10]**

1. To display actor details acted in movie 'Sholey'.
2. To display producer name who have produced more than two movies.

Q.2) Using above database solve following questions: **[Total Marks: 20]**

1. Write a trigger before inserting charges into relationship table. Charges should not be more than 30 lakh. Display appropriate message. **[10]**
2. Write a stored function to accept actor name as an input parameter and display names of movies in which that actor has acted. Display error message for an invalid actor name. **[10]**

Q.3) External Viva **[05]**

Q.4) Internal Evaluation **[15]**

MOVIE_ACTOR_PRODUCER

CREATE TABLE Movie (m_name VARCHAR(25), release_year INTEGER NOT NULL, budget DECIMAL, PRIMARY KEY (m_name, release_year));

CREATE TABLE Actor (a_name CHAR(30), city VARCHAR(30), PRIMARY KEY (a_name));

CREATE TABLE Producer (producer_id INTEGER, pname CHAR(30), p_address VARCHAR(30), PRIMARY KEY (producer_id));

CREATE TABLE Movie_Actor (m_name VARCHAR(25), release_year INTEGER, a_name CHAR(30), role VARCHAR(50), charges DECIMAL, PRIMARY KEY (m_name, release_year, a_name), FOREIGN KEY (m_name, release_year) REFERENCES Movie(m_name, release_year), FOREIGN KEY (a_name) REFERENCES Actor(a_name));

CREATE TABLE Movie_Producer (m_name VARCHAR(25), release_year INTEGER, producer_id INTEGER, PRIMARY KEY (m_name, release_year, producer_id), FOREIGN KEY (m_name, release_year) REFERENCES Movie(m_name, release_year), FOREIGN KEY (producer_id) REFERENCES Producer(producer_id));

INSERT INTO Movie VALUES ('Sholey', 1975, 5000000), ('Lagaan', 2001, 3000000), ('Taal', 1999, 2000000);

INSERT INTO Actor VALUES ('Amitabh Bachchan', 'Mumbai'), ('Aamir Khan', 'Mumbai'), ('Dharmendra', 'Pune'), ('Hema Malini', 'Delhi');

INSERT INTO Producer VALUES (1, 'Mr. Subhash Ghai', 'Mumbai'), (2, 'Yash Chopra', 'Pune');

INSERT INTO Movie_Actor VALUES ('Sholey', 1975, 'Amitabh Bachchan', 'Jai', 1000000), ('Sholey', 1975, 'Dharmendra', 'Veeru', 800000), ('Lagaan', 2001, 'Aamir Khan', 'Bhuvan', 1200000);

INSERT INTO Movie_Producer VALUES ('Sholey', 1975, 1), ('Lagaan', 2001, 2), ('Lagaan', 2001, 1) ('Taal', 1999, 1);

Q.1) Create a View:

CREATE VIEW Actors_In_Sholey AS SELECT ma.a_name, ma.role, ma.charges FROM Movie_Actor ma WHERE ma.m_name = 'Sholey' AND ma.release_year = 1975;

SELECT * FROM Actors_In_Sholey;

CREATE VIEW Producers_More_Than_Two_Movies AS SELECT p.pname FROM Producer p JOIN Movie_Producer mp ON p.producer_id = mp.producer_id GROUP BY p.pname HAVING COUNT(mp.m_name) > 2;

SELECT * FROM Producers_More_Than_Two_Movies;

Q.2) Using above database solve following questions:

```
CREATE OR REPLACE FUNCTION check_charges()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.charges > 3000000 THEN
    RAISE EXCEPTION 'Charges cannot be more than 30 lakhs.';
  END IF;
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER before_insert_charges
BEFORE INSERT ON Movie_Actor
FOR EACH ROW EXECUTE FUNCTION check_charges();

INSERT INTO Movie_Actor VALUES ('Sholey', 1975, 'Amitabh Bachchan', 'Lead', 4000000);
```

```
CREATE OR REPLACE FUNCTION movies_by_actor(actor_name CHAR(30))
RETURNS VOID AS $$
DECLARE
  movie_name VARCHAR(25);
BEGIN
  FOR movie_name IN
    SELECT ma.m_name
    FROM Movie_Actor ma
    WHERE ma.a_name = actor_name
  LOOP
    RAISE NOTICE 'Movie: %', movie_name;
  END LOOP;

  IF NOT FOUND THEN
    RAISE EXCEPTION 'Invalid actor name: %', actor_name;
  END IF;
END;
$$ LANGUAGE plpgsql;

SELECT movies_by_actor('Amitabh Bachchan');
```