DBMS Lab 5
2019

General Instruction

1. [P] marked questions are for your practice, need not be submitted. There are also some practice problems at the end. [B] marked questions are bonus. You will get 1 mark extra for bonus.
2. Make a single pdf file using screen shots. Work out the questions in the order given and also arrange them in the same order in the submitted pdf.
3. Error messages are also output, we need to check if you are getting correct error messages or not.

Trigger

A.University database

1. Create a table hostel with attributes (name, location). Insert values ('alpha','NE'), ('beta','SE'),('gamma','NW'),('delta','SW'). Create a table student_hostel (student_id, hostel_name, room_number). student_id refers to id in student table, hostel_name refers to name in hostel table. room _number is alpha-numeric type. Possible values are A1 to A47, B1 to B47. Write a trigger so that, whenever a new student is added to student table the student gets a room allocated. Rooms are allocated in sequence: hostel names as given before and room numbers as given above.
2. Create a table inst_dept by joining two existing table instructor and department.
3. Write a *trigger* for stopping any update anomalies on budget and building in your database.
    a. User can update *budget* and/or *building* of any individual tuple in *inst_dept* table or *department* table. Any update on *budget or building* in any tuple in any table should automatically reflects on *budget or building* of corresponding department for all redundant occurrence of it (even in other tables).
    b. Validate your trigger by executing following DML
        i. Update *budget* to '1,50,000' of instructor whose name is *"Srinivasan"* in inst_dept table.
        ii. Update *budget and building* of *"Physics"* department to *'1,00,000'* and *'Amstrong'*

B. Canteen database

1. Create a database *canteen*. Create a table *menu* with attributes *id int not null*, and *name varchar(50) not null, type* that can take value between *'healthy'*, and '*unhealthy'*. Create another table *customerorder* with attribute *id not null*, and *count int not null*. Create a table *price* that will contain *id* of a dish in the *menu* and *amount float*.
2. Create a trigger *init* that will initiate the *count* to zero for each entry in the *menu* table. Insert one record to menu table, and show how *init* works.
3. Drop the trigger *init* and show the effect.
4. Create a trigger *init_price* that will check the *type* of the dish in the menu, and will automatically initialize a price in the correct table. For healthy food price is 10, and for unhealthy foods price is 15.
5. Create trigger *price_checker* that will raise error, and display message 'price can not be negative' if accidentally someone tries to enter a negative price for some dish.
6. Modify the table *menu* to add an attribute *spicy* which can take value either *'Y'* or *'N'*. Create the attribute *spicy* as varchar but write a trigger *spicy_checker* to check the validity. If invalid then raise an error and throw a message: 'wrong spicy level!'.