CSE216 July 2022
Practice Problems for Triggers

Task 1: Create a trigger that will be executed **after** an insertion of a record in student table and display the following messages:

First line: "The CGPA of Student name _____ has been inserted"

Second line: "The CGPA is ____"

Test the trigger by inserting the sample data given in the book.

Task 2: Create a trigger that will be executed **before** an insertion of a record in student table and display the following messages:

First line: "The CGPA of Student name _____ will be inserted"

Second line: "The CGPA will be ____"

Test the trigger by inserting the sample data given in the book.

Task 3-1: Create a trigger that will be executed **before** an insertion or deletion of a record in student table and display the following message:

"Record has been inserted or deleted"

Test the trigger by inserting the sample data given in the book and deleting the same records.

Task 3-2: Create a trigger that will be executed **after** an insertion or deletion of a record in student table and display the following message:

"Record has been inserted or deleted"

Test the trigger by inserting the sample data given in the book and deleting the same records.

Task 4: Create a trigger that will be executed **after** an update in CGPA of a record in student table and display the following message:

"CGPA has been updated"

Insert 4 records as follows:

Abid 3.25, Zahid 3.5, Karim 3.7, Sohel 3.8

Test the trigger by increasing the CGPA of the students by 10% for the students with CGPA <= 3.7.

Task 5: Create a trigger that will be executed **before** an update in CGPA of a record in student table and display the following message:

"CGPA of student ------ has been updated with new CGPA ----"

"The previous CGPA of the student was ------ "

Test the trigger by decreasing the CGPA of the students by 10% for the students with CGPA >= 3.5.

Task 6: Write a PL/SQL trigger LOG_CGPA_UPDATE. This trigger will log all updates done on the CGPA column of STUDENTS table. The trigger will save current user's name, current system date, previous CGPA and Changed CGPA in a log table named LOG_TABLE_CGPA_UPDATE.

Task 7: Write a PL/SQL trigger BACKUP_MANAGEMENT. This trigger will save all records that are deleted from the STUDENTS table into a backup table named STUDENTS_DELETED.

Task 8: Write a trigger that will save a student records in a table named LOW_CGPA_STUDENTS which contain only one column to store student's names. The trigger will work before an updateoperation or an insert operation. Whenever the update operation results in a CGPA value less than 2.0, the trigger will be fired and the trigger will save the students name in the LOW_CGPA_STUDENTS table. Similarly, when an insert operation inserts a new row with CGPA less than 2.0, the corresponding row must be saved in the LOW_CGPA_STUDENTS table.

Task 9: Write a trigger that will be fired when DML operations (INSERT, UPDATE, and DELETE statements) are performed on emp_tab table. You can choose what combination of operations should fire the trigger.

Because the trigger uses the BEFORE keyword, it can access the new values before they go into the table, and can change the values if there is an easily-corrected error by assigning to :NEW.column_name. You might use the AFTER keyword if you want the trigger to query or

change the same table, because triggers can only do that after the initial changes are applied and the table is back in a consistent state.

Because the trigger uses the FOR EACH ROW clause, it might be executed multiple times, such as when updating or deleting multiple rows. You might omit this clause if you just want to record the fact that the operation occurred, but not examine the data for each row.

For this trigger, you have to create the following table and then create the trigger.

```
CREATE TABLE Emp_tab (
   Empno     NUMBER NOT NULL,
   Ename     VARCHAR2(10),
   Job       VARCHAR2(9),
   Mgr       NUMBER(4),
   Hiredate  DATE,
   Sal       NUMBER(7,2),
   Comm      NUMBER(7,2),
   Deptno    NUMBER(2) NOT NULL);
```

```
CREATE OR REPLACE TRIGGER Print_salary_changes
  BEFORE DELETE OR INSERT OR UPDATE ON Emp_tab
  FOR EACH ROW
WHEN (new.Empno > 0)
DECLARE
    sal_diff number;
BEGIN
    sal_diff  := :new.sal  - :old.sal;
    dbms_output.put('Old salary: ' || :old.sal);
    dbms_output.put('  New salary: ' || :new.sal);
    dbms_output.put_line('  Difference ' || sal_diff);
END;
/
```