PL/SQL Day 4| Assigment

Harshit Kushmakar| 16896

Triggers

Practice 1

• Implement the following business rule with the help of a trigger named TR_CHECK_DEPT and a procedure named SECURE_DML. Changes to data in the dept table, will be allowed only in the month of March.

• Create a procedure called SECURE_DML that prevents the DML statement from executing in any other month than March. In case, a user tries to modify the table in any other month, the procedure should display a message "You can modify or add a department only at the end of a financial year"

• Create a statement level triggerTR_CHECK_DEPT on the dept table that calls the above procedure.

• Test it by inserting a new record in the dept table.

```
set serveroutput on
CREATE OR REPLACE PROCEDURE SECURE_DML_Kushmakar
AS
BEGIN
IF TO_CHAR(SYSDATE,'MON') <> 'MAR' THEN
DBMS_OUTPUT.PUT_LINE('ONLY ALLOWED');
END IF;
END;
```

```
/



CREATE OR REPLACE TRIGGER TR_M
BEFORE INSERT ON E FOR EACH ROW
BEGIN
SECURE_DML_Kushmakar;
END;
/

INSERT INTO employee1(ID,NAME) VALUES(13,'ANIL');
SELECT * FROM employee1
```

Practice 2

• Enforce referential integrity with a trigger named TR_CASCADE_CHANGE. When the value of DEPTNO changes in the Dept table, cascade the update to the corresponding rows in the EMP table.

• Test it by updating the value of a deptno from the dept table.

```
CREATE OR REPLACE TRIGGER TR_CAS_Kushmakar
AFTER UPDATE ON employee1 FOR EACH ROW
BEGIN
UPDATE employee1 SET DEPID = :NEW.ID WHERE DEPID = :OLD.ID;
```

```
END;
/
```

```
DECLARE
NEWID employee1.DEPID%TYPE := '&NEWID';
OLDID employee1.DEPID%TYPE := '&OLDID';
BEGIN
UPDATE employee1 SET ID = NEWID WHERE ID = OLDID;
END;
/
```

Practice 3

- **Create a trigger named TR_CHECK_COMM to implement the following business rule. In EMP table, employee having job as 'Salesman' should receive a commission. A Salesman must receive a commission of not less than Rs. 100. Employees who are not sales persons are not entitled to get commission (comm value should be NULL).**
- **Test it by inserting a record in the emp table.**

```
CREATE OR REPLACE TRIGGER TR_CHECK_Kushmakar
BEFORE INSERT ON employee1 FOR EACH ROW
BEGIN
```

```
IF :NEW.JOB = 'HR' AND :NEW.COMM < 100 THEN
RAISE_APPLICATION_ERROR(-20010,'COMM MUST BE GREATER');
ELSIF :NEW.JOB <> 'HR' AND :NEW.COMM > 0 THEN
RAISE_APPLICATION_ERROR(-20010,'COMM NOT FOUND');
END IF;
END;
/
```

```
INSERT INTO employee1(ID,NAME) VALUES(15,'PG');
SELECT * FROM employee1;
```

Practice 4

• While modifying the EMP table, ensure that the salary is in the valid range as specified in the SALGRADE table (between lowest losal and highest hisal) with the help of a trigger named TR_VALIDATE_SAL.

• Test it by updating the salary value of an existing record in the emp table.

```
create or replace trigger TR_VALIDATE_SAL_Kushmakar
before update on employee1 for each row
```

```
begin
 if(FIND_SAL_GRADE_Kushmakar(:new.salary)<>'0')
 then
 DBMS_OUTPUT.PUT_LINE('Record inserted');
 else
 raise_application_error(-20000,'Data is not as per buisness
rule');
 end if;
end;

------------------------------------------------------4
CREATE TABLE SALARYLOG_Kushmakar(
EMPNO NUMBER,
NEW_GRAD VARCHAR2(2),
OLD_SAL NUMBER,
NEW_SAL NUMBER
);



INSERT INTO SALARYLOG_Kushmakar VALUES(1,'A',10000,12000);
```

Practice5

• Create a table named salaryLog with the appropriate columns and insert the empno, new grade, old salary and new salary values in salaryLog table if the grade of an employee changes.

• Create a trigger named TR_CHECK_GRADE that will be fired when a user modifies the EMP table. It will check whether the grade has changed by making use of the SALGRADE table. (Grade is dependent on Salary.) If grade is changed, the trigger will log the corresponding employee number, old salary, new salary and new grade into salaryLog table.

• Test the working of the trigger by firing an appropriate DML query.

```
CREATE OR REPLACE TRIGGER TR_CHECK_GRADE_Kushmakar

BEFORE UPDATE ON employee1 FOR EACH ROW

DECLARE

OLD_GRADE SAL_GRADE_Kushmakar%TYPE;

NEW_GRADE SAL_GRADE_Kushmakar%TYPE;

BEGIN

SELECT GRADE INTO OLD_GRADE FROM SAL_GRADE_Kushmakar WHERE
START_RANGE < :OLD_SALARY;

SELECT GRADE INTO NEW_GRADE FROM SAL_GRADE_Kushmakar WHERE
START_RANGE < :NEW_SALARY;

IF NEW_GRADE <> OLD_GRADE THEN

insert into SALARYLOG_Kushmakar
values(:new.id,:old.salary,:new.salary,FIND_SAL_GRADE_16845(:new.salary));

 DBMS_OUTPUT.PUT_LINE('Record inserted in salaryLog_16845');

 end if;
```

```
end;
/
```