Name: Urooba Gohar Roll No: 22P-9216 Section: BSCS-6A

# Database Systems Labtask 9

# **DML Trigger Tasks:**

Q1. Create a trigger that automatically updates an employee's bonus table when a new record is added to the employees table. The bonus is set to 10% of the inserted salary. Create a table employee\_bonus and populate it on each insert command.

Ans: First we create employees table:

```
--22P-9216

create table employees(
employeeid int primary key,
name varchar(100),
salary decimal(10,2)
);
```

Then we create employee\_bonus table:

```
--22P-9216
create table employee_bonus(
employeeid int primary key,
bonus_amt decimal(10,2)
);
```

Now we create the trigger:

```
--22P-9216

create or replace trigger update_bonus
after insert on employees
for each row

begin
    insert into employee_bonus(employeeid, bonus_amt) values
    (:new.employeeid, :new.salary*0.10);
end;
/
```

Trigger UPDATE\_BONUS compiled

Now to test the trigger, we insert values into the employees table:

```
--22P-9216
insert into employees(employeeid, name, salary) values (111, 'urooba gohar', 60000)
```

And view all from employee\_bonus:

```
--22P-9216
select * from employee_bonus
```

This is what we see:



Q2. Create a trigger that checks the new salary value being updated in the employees table. If the new salary is greater than a threshold (say 10,000), display an error message to the user.

Ans: First we create the trigger:

```
--22P-9216

Create or replace trigger check_salary
before update of salary on employees
for each row

begin
    if :new.salary > 10000 then
        RAISE_APPLICATION_ERROR(-20001, 'error. salary cannot be greater than 10,000');
end if;
end;
//
```

#### Trigger CHECK\_SALARY compiled

Now to test the trigger, we update employees table:

```
--22P-9216
update employees
set salary=11000 where employeeid=111
```

The result is:

```
Error starting at line: 2 in command -

UPDATE employees

SET salary = 11000

WHERE employeeid = 111

Error report -

ORA-20001: error. salary cannot be greater than 10,000

ORA-06512: at "P229216_LAB3.CHECK_SALARY", line 3

ORA-04088: error during execution of trigger 'P229216_LAB3.CHECK_SALARY'
```

Q3. Create a trigger that logs every deleted record from the Employees table into a Deleted\_Employees\_Log table.

Ans: First we create the deleted employees log table:

```
--22P-9216
create table deleted_employees_log(
employeeid number(20),
name varchar2(100),
salary decimal(10,2)
```

Now we create the trigger:

```
--22P-9216

create or replace trigger log_record

after delete on employees

for each row

begin

insert into deleted_employees_log(employeeid, name, salary)

values (:old.employeeid, :old.name, :old.salary);

end;

/
```

Trigger LOG\_RECORD compiled

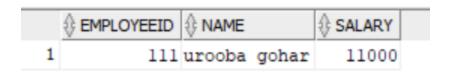
Now to test the trigger, we delete from employees:

```
--22P-9216
delete from employees where employeeid=111;
```

Now we view all from the table:

```
--22P-9216
select * from deleted_employees_log
```

This is the result:



Q4. Create a trigger that logs the old and new values of a salary whenever an UPDATE occurs in the employees table.

Ans: First we create the salary\_update\_log table:

```
--22P-9216
create table salary_update_log(
employeeid number(20),
name varchar2(100),
old_salary decimal(10,2),
new_salary decimal(10,2)
```

Now create the trigger:

```
--22P-9216

create or replace trigger salary_log
after update of salary on employees
for each row
begin
insert into salary_update_log(employeeid, name, old_salary, new_salary)
values (:old.employeeid, :old.name, :old.salary, :new.salary);
end;
//
```

Trigger SALARY\_LOG compiled

To test the trigger, insert values into employees table:

```
--22P-9216
insert into employees (employeeid, name, salary)
values (112, 'sara', 8000);
```

Update employees:

```
--22P-9216
update employees
set salary=9500
where employeeid=112;
```

View all from the table:

```
--22P-9216
select * from salary_update_log;
```

This is the result:

		<b>♦ NAME</b>	♦ OLD_SALARY	NEW_SALARY
1	112	sara	8000	9500

\_\_\_\_

# **DDL Trigger Tasks:**

Q1. Create a trigger that logs every new table created in the database into an Audit\_Log table, including the table name, creation time and user name.

Ans: First we create the audit\_log table:

```
--22P-9216
create table audit_log(
table_name varchar2(20),
creation_time timestamp default current_timestamp,
user_name varchar2(20)
```

Now create the trigger:

```
--22P-9216

create or replace trigger audit_trigger
after create on database
declare
    table_name_new varchar2(20);

begin
    insert into audit_log(table_name, user_name)
    values (table_name_new, user);
end;

/

Trigger AUDIT_TRIGGER compiled
```

Now we test the trigger:

```
--22P-9216
create table t1(
id number,
name varchar2(20)
```

View all from the table:

```
--22P-9216
select * from audit_log
```

This is the result:

```
        $\psi$ TABLE_NAME
        $\psi$ CREATION_TIME
        $\psi$ USER_NAME

        1 (null)
        17-APR-25 03.59.08.122000000 AM P229216_LAB3
```

Q2. Create a trigger that prevents changes (ALTER statements) to the employees table after business hours (e.g., 6 PM to 8 AM).

Ans: First we create the trigger:

```
--22P-9216
| create or replace trigger stop_alter_employees
| before ddl on database
| begin
| if ora_sysevent='alter'
| and ora_dict_obj_name='employees'
| and ora_dict_obj_type='table' then
| if to_number(to_char(sysdate, 'HH24')) < 8 or to_number(to_char(sysdate, 'HH24')) >= 18 then
| raise_application_error(-20001, 'alter on employees not allowed after hours');
| end if;
| end;
| end;
```

Now we test it:

```
--22P-9216
alter table employees add(after_hours_test number);
```

This is the output:

```
Error starting at line : 2 in command -
ALTER TABLE employees ADD (after_hours_test NUMBER)
Error report -
ORA-00604: error occurred at recursive SQL level 1
ORA-20001: ALTER on EMPLOYEES not allowed after hours.
ORA-06512: at line 7
```

Q3. Create a trigger that logs every DROP operation on any table in the database to a Drop\_Log table, recording the user who performed the action and the time it occurred.

Ans: First we create the drop log table:

```
--22P-9216

create table drop_log(
log_id number primary key,
inserted_by varchar2(20),
insertion_time date
```

Now create the trigger:

```
--22P-9216

create or replace trigger drop_trigger
after drop on database
begin
insert into drop_log(log_id, inserted_by, insertion_time)
values (1, user, sysdate);
end;
/
```

Trigger DROP\_TRIGGER compiled

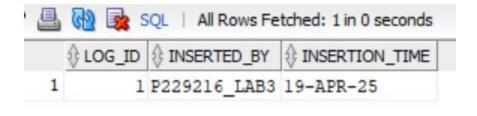
Now we test the trigger by dropping the students table:

```
--22P-9216
drop table students
```

Now view all:

```
--22P-9216
select * from drop_log
```

This is the result:



Q4. Create a trigger that prevents dropping the Audit\_Log table under any circumstance and display a warning message instead.

Ans: First we create the trigger:

```
--22P-9216

create or replace trigger prevent_drop_trigger

before drop on schema

begin

if ora_dict_obj_name='audit_log' and ora_dict_obj_type='table' then

raise_application_error(-20001, 'cannot drop audit_log table');

end if;

end;

/
```

Trigger PREVENT\_DROP\_TRIGGER compiled

### **System/Database Trigger Tasks:**

Q1. Create a trigger that logs the time and status when the database starts into a System\_Logs table.

Ans: First we create the system logs table:

```
--22P-9216
create table System_Logs(
log_time date default sysdate,
status varchar2(50)
```

Now we create the trigger:

```
--22P-9216
Create or replace trigger log_startup
after startup on database
begin
insert into System_Logs (status)
values('database started');
end;
/
Trigger LOG_STARTUP compiled
```

Q2. Create a trigger that tracks the login attempts of users and logs unsuccessful attempts into a Failed Logins table.

Ans: First we create the users table:

```
--22P-9216
create table Users(
   username varchar2(50) primary key,
   password varchar2(50)
);
```

Now we create a login attempts table:

```
--22P-9216
create table Login_Attempts(
    username varchar2(50),
    password varchar2(50)
);
```

Now we create failed logins table:

Now create the trigger:

```
--22P-9216

create or replace trigger failed_login

before insert on Login_Attempts

for each row

declare

new_count number;

begin

select count(*) into new_count

from Users

where username=:new.username and password=:new.password;

if new_count=0 then

insert into Failed_Logins (username, reason)

values(:new.username, 'invalid username or password');

end if;

end;

/
```

```
Trigger FAILED_LOGIN compiled
```

Q3. Create a trigger that logs every successful logout along with the session duration into a User\_Activity\_Log table.

Ans: First we create the login\_sessions table:

Now we create the user activity log table:

Now we create a logout\_event table:

Now create the trigger:

```
Z---,-----
 --22P-9216
create or replace trigger trg log logout
 after insert on Logout Event
 for each row
 declare
     v login time date;
     v_session_time interval day to second;
BEGIN
     SELECT MAX(login_time)
     INTO v login time
     FROM Login Sessions
     WHERE username = : NEW.username;
     v_session_time := NUMTODSINTERVAL(:NEW.logout_time - v_login_time, 'DAY');
     INSERT INTO User Activity Log (
         username, login time, logout time, session duration
     ) VALUES (
         :NEW.username,
         v login time,
         :NEW.logout time,
         v_session_time
     );
 END:
```

# **Instead Of Trigger Tasks:**

Q1. Create a view that joins Employees and Departments, and write an INSTEAD OF INSERT trigger that correctly distributes new data into both the Employees and Departments tables.

Ans: First we create the departments table:

```
--22P-9216
create table Departments(
Department_ID int primary key,
Department_Name varchar2(100)
);
```

Then we create the employees table:

Then we create view:

```
--22P-9216

create view Employee_Department_View as

select e.Employee_ID, e.Employee_Name, e.Department_ID, d.Department_Name
from Employees e

join Departments d
on e.Department_ID = d.Department_ID;
```

Then we create the trigger:

```
--22P-9216

create or replace trigger Employee_Department_Insert

instead of insert on Employee_Department_View

for each row

begin

insert into Departments (Department_Name)

values(:NEW.Department_Name);

insert into Employees (Employee_Name, Department_ID)

values(:new.Employee_Name, (select Department_ID from Departments

where Department_Name = :new.Department_Name));
end;
```

Trigger EMPLOYEE DEPARTMENT INSERT compiled

Q2. Create a view that shows employee salaries, and write an INSTEAD OF UPDATE trigger to prevent any salary updates that reduce the employee's salary by more than 20%.

Ans: First we create the view:

```
--22P-9216

create view employee_salaries as

select employee_id, employee_name, salary

from employees;
```

Then we create the trigger:

```
--22P-9216
create trigger prevent_reduction
instead of update on employee_salaries
for each row
begin
   if :new.salary < :old.salary*0.8 then
        raise_application_error(-20001, 'salary reduction exceeds 20% limit');
   else
        update employees set salary=:new.salary
        where employee_id=:old.employee_id;
end if;
end;</pre>
```

Trigger PREVENT\_REDUCTION compiled