Exercise 3 PL/SQL Exercise - triggers

1. Using triggers to maintain business rules

Suppose that the Middlesex Transport Authority (MTA) has a rule stating that a bus driver's salary cannot be changed by more than 20% of the original salary. Create a trigger 'salary_change_monitoring' to enforce this constraint. The trigger fires whenever there is an update to the Busdriver table and outputs a suitable error message when the rule is violated.

In this case, we can define a trigger on the **Busdriver** table in the following way:

```
create or replace trigger salary_change_monitoring
before update on busdriver
for each row
begin
    if ((:new.bdsalary/:old.bdsalary) >= 1.2) or
    ((:old.bdsalary/:new.bdsalary) >= 1.2)
    then
    RAISE_APPLICATION_ERROR(-20002, 'Warning: Large percentage change in salary
    prohibited.');
    end if;
end;
/
```

Example update on busdriver giving Jane Brown an increase >20%:

```
Update busdriver
 set salary = 3600
where dname = 'Jane Brown';

Update busdriver
     *

Update busdriver
         *
ERROR at line 1:
ORA-20002: Warning: Large percentage change in salary prohibited.
ORA-06512: at "PATRICIA.SALARY_CHANGE_MONITORING", line 5
ORA-04088: error during execution of trigger 'PATRICIA.SALARY_CHANGE_MONITORING'
```

2. Creating triggers to prevent updates and deletions

In the BusDrivers' database, we can see that the rows in the Depot table are often referenced by many child rows in a number of other tables (e.g., Bus, Cleaner and Busdriver). Although there are FOREIGN KEY constraints declared on the child tables to maintain the referential integrity, we can still define a trigger in the parent table (i.e., Depot) to stop any attempt to change the name of the depot and/or to remove any of the depot rows. This is corresponding to the business rule stating that once a depot is established, it will be there 'for ever' and will not be allowed to change name (although unrealistic, we assume that such a rule is necessary).

Write appropriate PL/SQL statements to create the trigger. Note that the trigger you create is a **statement level trigger** so the 'for each row 'statement should not be used. After the trigger is created, try to change the name of some depots and delete a row from depot, and see what will happen.

The trigger we need is a **statement-level trigger**. The triggering events are UPDATE of DNAME and DELETE. The proper PL/SQL statements are as follows:

```
create or replace trigger Depot_trigger
before update of dname or delete on depot
begin
RAISE_APPLICATION_ERROR (-20501, 'You are not allowed to change the value of
DNAME or delete a depot row');
end;
/

  Trigger created.
```

Having created the trigger, if you try to perform an update operation on gname such as:

```
Update depot
set dname = 'Crouch End'
where dname = 'Hornsey';
```

The following error warns you that the operation is prohibited.

```
Update depot
       *
ERROR at line 1:
ORA-20501: You are not allowed to change the value of DNAME or delete a depot row
ORA-06512: at "PATRICIA.DEPOT_TRIGGER", line 2
ORA-04088: error during execution of trigger 'PATRICIA.DEPOT_TRIGGER'
```

3. Creating triggers to maintain data validity

A CHECK constraint is similar to a validation rule and is an option in the CREATE TABLE command whereby you can specify what data may be entered into a particular column. So if we wanted to add a constraint in an Cleaner table that salary must be within certain limits we could create the table thus:

```
create table Cleaner
(cno            varchar2(5),
cname           varchar2(20),
csalary                 number(6,2),
dno             varchar2(5),
constraint pk_clno primary key(cno),
constraint fk_deno1 foreign key(dno) references depot(dno),
check (csalary >0 and csalary <5000)      );
```

Here any salary that is less than zero and greater than 5000 will cause a violation of the constraint.

Applying the CHECK constraint, however, we would not know whether the salary is greater than 5000 or smaller than 0 (i.e., a negative number).

We can create a trigger instead of a CHECK constraint, which can tell us how the restriction on ‘csalary’ is violated. Whenever the value of ‘csalary’ is beyond the valid range (0 – 5000), the trigger will generate an error message informing users whether it is greater than 5000 or a negative number. (If a check constraint already exists it must be dropped first.)

Write PL/SQL statements to create the trigger, and use some SQL **update** and **insert** statements to test it.

The trigger we need is a row-level trigger. The triggering events are UPDATE of salary and INSERT. The proper PL/SQL statements are as follows:

```
create or replace trigger cleaner_salary_trigger
before update of csalary or insert on cleaner
for each row
when((new.csalary < 0) or (new.csalary > 5000))
begin
if :new.csalary < 0 then
      RAISE_APPLICATION_ERROR(-20511, ‘The salary cannot be negative.’);
end if;
if :new.csalary > 5000 then
      RAISE_APPLICATION_ERROR(-20512, ‘The salary cannot exceed 5000.’);
end if;
end;
/

Trigger created.
```

Having created the trigger, if you try to perform an update operation or insert a cleaner row with a salary not between 0 and 5000, an error occurs. For example, when executing

```
UPDATE cleaner
SET csalary = 6000
WHERE cno = '114';
```

```
ERROR at line 3:
ORA-20512: The salary cannot exceed 5000.
ORA-06512: at "PATRICIA.CLEANER_SALARY_TRIGGER", line 6
ORA-04088: error during execution of trigger 'PATRICIA.CLEANER_SALARY_TRIGGER'
```