Oracle trigger

A trigger is a named PL/SQL block stored in the Oracle Database and executed automatically when a triggering event takes place. The event can be any of the following:

- A data manipulation language (DML) statement executed against a table e.g., INSERT, UPDATE, or DELETE. For example, if you define a trigger that fires before an INSERT statement on the customers table, the trigger will fire once before a new row is inserted into the customers table.
- A data definition language (DDL) statement executes e.g., CREATE or ALTER statement. These triggers are often used for auditing purposes to record changes of the schema.
- A system event such as startup or shutdown of the Oracle Database.
- A user event such as login or logout.

How to create a trigger in Oracle

CREATE TRIGGER statement:

```
CREATE [OR REPLACE] TRIGGER trigger_name
{
BEFORE | AFTER
}
triggering_event ON table_name
[FOR EACH ROW]
[FOLLOWS | PRECEDES another_trigger]
[ENABLE / DISABLE ]
[WHEN condition]
DECLARE
   declaration statements
BEGIN
   executable statements
EXCEPTION
   exception_handling statements
END;
```

A trigger has two main parts: header and body.

The following illustrates the trigger header:

```
CREATE [OR REPLACE] TRIGGER trigger_name
   {
BEFORE | AFTER
 } triggering_event ON table_name
   [FOR EACH ROW]
   [FOLLOWS | PRECEDES another_trigger]
   [ENABLE / DISABLE ]
   [WHEN condition]
```

And this is the trigger body:

```
DECLARE
   declaration statements
BEGIN
   executable statements
EXCEPTION
   exception_handling statements
END;
```

PL/SQL BLOCK FOR TRANSACTION OPERATIONS OF AN INVENTORY APPLICATION USING TRIGGERS

AIM

To write a PL / SQL block for transaction operations of an inventory application using triggers.

ALGORITHM

Step – I : Create tables Prod_Details and Inv_Trans as follows:

```
SQL> create table Prod_Details (item_code number(4) constraint pk_ic primary key, item_name varchar2(30) constraint nn_in not null, min_qty number(4) constraint chk_mq check (min_qty > 0), reorder_qty number(4) constraint chk_rq check (reorder_qty > 0), qty_in_hand number(4) constraint chk_qih check(qty_in_hand > 0));
SQL> create table Inv_Trans (trans_date date, item_code number(4), trans_mode char(1), trans_qty number(4) constraint chk_tq check (trans_qty > 0), constraint pk_td_ic primary key (trans_date, item_code), constraint fk_ic foreign key (item_code) references prod_details (item_code) on delete cascade);
```

Step – II : Type ed in the SQL Editor to open a notepad window. Key in the trigger code and save it in a file (say E:\SKS\MC9217\Program_9\Reorder_Trig.txt). The syntax for creating a trigger is:

```
Create or replace Trigger <Trigger_name>

[before/after] [insert/update/delete] on <table-name>

[for each statement / for each row] [when <condition>];
```

Step – III : The trigger is fired before every insert or update operation in Prod_Details table, when the quantity in hand for a particular item goes below the minimum quantity required for that item. Execute the trigger as shown below:

```
SQL> @ E:\SKS\MC9217\Program_9\Reorder_Trig.txt
```

```
Trigger created.
```

Step – IV : Insert records in the Prod_Deatils table by using the Insert command as given below:

```
SQL> insert into prod_details values (&item_code,
     '&item_name', &min_qty, &reorder_qty, &qty_in_hand);
```

Step – V : In the notepad window, key in the PL/SQL procedure and save it in a file (say E:\Prince\MC9217\Program_9\PLSQL_Trans.txt).

Step – VI : The PL / SQL procedure inserts a new record in the Inv_Trans table and updates Qty_in_hand in the Prod_Details table according to the transaction mode.

Step – VII : Execute the PL / SQL procedure as shown below:

```
SQL> @ E:\SKS\MC9217\Program_9\PLSQL_Trans.txt
```

Step – VIII : Input the Item Code, Transaction Mode, and Transaction Quantity values to execute a transaction.

Step – IX : Execute the following command for the dbms_output.put_line (a function to display a line of text) used in the PL / SQL procedure to take effect.

```
SQL> set serveroutput on
```

create table Prod_Details (item_code number(4) constraint pk_ic primary key, item_name varchar2(30) constraint nn_in not null, min_qty number(4) constraint chk_mq check (min_qty > 0), reorder_qty number(4) constraint chk_rq check (reorder_qty > 0), qty_in_hand number(4) constraint chk_qih check(qty_in_hand > 0));

Reorder_trig.txt

```
create or replace trigger reorder_trig before insert or update of qty_in_hand on prod_details for each row when (new.qty_in_hand < new.min_qty)

begin

 raise_application_error(-20001, 'Quantity in hand for any item should be greater than the minimum quantity level for that item');

end;

/
```

PLSQL_Trans.txt

```
declare
 ic inv_trans.item_code%type;
 tm inv_trans.trans_mode%type;
 tq inv_trans.trans_qty%type;
 qih prod_details.qty_in_hand%type;
begin
 ic := &item_code;
 tm := '&transaction_mode';
 tq := &transaction_quantity;
 select qty_in_hand into qih from prod_details where item_code=ic;
 if upper(tm) = 'P' then
  qih := qih + tq;
 elsif upper(tm) = 'S' then
  qih := qih - tq;
 else
  dbms_output.put_line('The transaction mode should be either P or S');
 end if;
 if upper(tm) = 'P' or upper(tm) = 'S' then
  update prod_details set qty_in_hand = qih where item_code = ic;
  insert into inv_trans values (sysdate, ic, tm, tq);
 end if;
 dbms_output.put_line('Item Code            : ' || to_char (ic));
```

```
 dbms_output.put_line('Transaction Mode     : ' || upper (tm));
 dbms_output.put_line('Transaction Quantity : ' || to_char (tq));
 dbms_output.put_line('Quantity In Hand     : ' || to_char (qih));
end;
/
```