**Advanced Database Management System**

**Assignment : 2**

**Section: D**

## Suppose XYZ company has a rule stating that a employee’s salary cannot be changed by more than 20% of the original salary if his/her experience is less than 5 years. Create a trigger ‘salary\_change\_monitoring’ to enforce this constraint. The trigger fires whenever there is an update to the emp table and outputs a suitable error message when the rule is violated.

**Solution:**

CREATE OR REPLACE TRIGGER salary\_change\_monitoring BEFORE UPDATE OF SAL ON EMP

FOR EACH ROW DECLARE

years\_experience NUMBER; BEGIN

years\_experience := EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM :OLD.HIREDATE);

IF years\_experience < 5 THEN

IF :NEW.SAL > :OLD.SAL \* 1.2 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Salary cannot be changed by more than 20% of the original salary if experience is less than 5 years');

END IF;

END IF;

END;

1. Write database triggers that checks that qty\_on\_hnd does not become negative.

**Solution:**

## CREATE OR REPLACE TRIGGER check\_qty\_on\_hand BEFORE UPDATE OF QTY\_ON\_HND ON INVENTORY FOR EACH ROW

DECLARE BEGIN

IF :NEW.QTY\_ON\_HND < 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Quantity on hand cannot become negative'); END IF;

END;

1. A trigger occurs when ever an insert or update occurs in bookself table and insert the old and new data in the bookself\_audit table.

create table product\_master (

pno varchar2(5), dsc varchar2(10),

profit\_percent number(3), unit\_measure varchar2(10), qoh number(4), reorderlevel number(8), sell\_price number(8), cose\_price number(8)

);

create table bookself (

Title varchar2(6), publisher varchar2(6), rating varchar2(5)

);

create table bookself\_audit (

Title varchar2(6), publisher varchar2(6), old\_rating varchar2(5), new\_rating varchar2(5), audit\_dt date

)

### Solution:

CREATE OR REPLACE TRIGGER bookshelf\_change\_audit AFTER INSERT OR UPDATE ON bookself

FOR EACH ROW BEGIN

IF UPDATING THEN

INSERT INTO bookself\_audit (Title, publisher, old\_rating, new\_rating, audit\_dt) VALUES (

:OLD.Title,

:OLD.publisher,

:OLD.rating,

:NEW.rating,

SYSDATE

);

ELSIF INSERTING THEN

INSERT INTO bookself\_audit (Title, publisher, old\_rating, new\_rating, audit\_dt) VALUES (

:NEW.Title,

:NEW.publisher, NULL,

:NEW.rating, SYSDATE

); END IF; END;

1. Create 2 tables, LAB7\_RESERVATIONS table with 2 columns Create table LAB7\_RESERVATIONS

( flight\_id char(6) not null, Customer\_phone char(11) not null )

Create table LAB7\_flights ( flight\_id char(6) not null, Seats number(3) not null, Primary key (flight\_id) )

Insert into LAB7\_flights values (ACO529,120) Insert into LAB7\_flights values (ACO529,120)

Create a trigger LAB7\_RES\_TRG that will ensure that when a new row is inserted into the LAB7\_RESERVATIONS table, the flight id is in the LAB7\_FLIGHTS table and that the number of seats on this flight, SEATS is greater than 0. Here are the details of how the trigger should behave:

* + If flight id is not in the flights table it should raise application error ‘Invalid flight id’
  + If flight id is in the flights table, (for example AC0529) but SEATS = 0, then it should raise application error ‘Flight AC0529 has no seats left’
  + If flight id is in the flights table and SEATS > 0,then it should update the appropriate row in flights table by setting SEATS = SEATS – 1 for this flight.

### Solution:

CREATE OR REPLACE TRIGGER LAB7\_RES\_TRG BEFORE INSERT ON LAB7\_RESERVATIONS FOR EACH ROW

DECLARE

v\_seats LAB7\_FLIGHTS.seats%TYPE; BEGIN

SELECT seats INTO v\_seats FROM LAB7\_FLIGHTS

WHERE flight\_id = :NEW.flight\_id;

IF v\_seats IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid flight id'); ELSIF v\_seats = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Flight ' || :NEW.flight\_id || ' has no seats left'); ELSE

UPDATE LAB7\_FLIGHTS

SET seats = seats - 1

WHERE flight\_id = :NEW.flight\_id; END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid flight id'); END;