Ex.No.: 13		**WORKING WITH TRIGGER** **<u>TRIGGER</u>**
Date:	**29.10.2024**	

Program 1

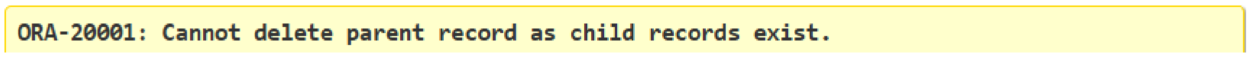
Write a code in PL/SQL to develop a trigger that enforces referential integrity by preventing the deletion of a parent record if child records exist.

```
CREATE OR REPLACE TRIGGER prevent_parent_deletion
BEFORE DELETE ON parent_table
FOR EACH ROW
DECLARE
   child_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO child_count
   FROM child_table
   WHERE parent_id = :OLD.parent_id;

   IF child_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Cannot delete parent record as child records
exist.');
   END IF;
END;

Testing of Trigger

DELETE FROM parent_table WHERE parent_id = 1;
```



```
ORA-20001: Cannot delete parent record as child records exist.
```

Program 2

Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.

```
CREATE OR REPLACE TRIGGER check_duplicate_value

BEFORE INSERT OR UPDATE ON table_name

FOR EACH ROW

DECLARE

  v_count NUMBER;

BEGIN

  -- Check if the new value already exists in the table

  SELECT COUNT(*) INTO v_count

  FROM table_name

  WHERE specific_column = :NEW.specific_column;


  -- If a duplicate is found, raise an error

  IF v_count > 0 THEN

        RAISE_APPLICATION_ERROR(-20002, 'Duplicate value detected in specific column.');

  END IF;

END;

/
```

Output:

```
ORA-20002: Duplicate value detected in specific column.
```

Program 3

Write a code in PL/SQL to create a trigger that restricts the insertion of new rows if the total of a column's values exceeds a certain threshold.

```
CREATE OR REPLACE TRIGGER restrict_insertion

BEFORE INSERT ON table_name

FOR EACH ROW

DECLARE

  v_total NUMBER;

  v_threshold CONSTANT NUMBER := 10000; -- Set your threshold here

BEGIN

  -- Calculate the total sum of the column values

  SELECT SUM(column_name) INTO v_total FROM table_name;



  -- Prevent insertion if the threshold is exceeded

  IF v_total + :NEW.column_name > v_threshold THEN

          RAISE_APPLICATION_ERROR(-20003, 'Cannot insert, total column value
exceeds threshold.');

  END IF;

END;

/
```

Output:

```
ORA-20003: Cannot insert, total column value exceeds threshold.
```

Program 4

Write a code in PL/SQL to design a trigger that captures changes made to specific columns and logs them in an audit table.

```
CREATE OR REPLACE TRIGGER log_column_changes

AFTER UPDATE ON table_name

FOR EACH ROW

BEGIN

  -- Check if specific columns have been modified

  IF :OLD.column_name1 != :NEW.column_name1 OR :OLD.column_name2 !=
:NEW.column_name2 THEN

        -- Insert the old and new values into the audit table

        INSERT INTO audit_table (user_id, change_time, old_value, new_value)

        VALUES (USER, SYSDATE, :OLD.column_name1 || ', ' || :OLD.column_name2,
:NEW.column_name1 || ', ' || :NEW.column_name2);

  END IF;

END;

/
```

Output:

```
User_ID    Change_Time        Old_Value          New_Value

SYSTEM     2024-09-19         OldValue1,         NewValue,
           10:05:00           OldValue2          AnotherNewValue
```

Program 5

Write a code in PL/SQL to implement a trigger that records user activity (inserts, updates, deletes) in an audit log for a given set of tables.

```
CREATE OR REPLACE TRIGGER audit_user_activity

AFTER INSERT OR UPDATE OR DELETE ON table_name

FOR EACH ROW

BEGIN

  IF INSERTING THEN

          INSERT INTO audit_log (user_id, operation, record_id, change_time)

          VALUES (USER, 'INSERT', :NEW.id_column, SYSDATE);


  ELSIF UPDATING THEN

          INSERT INTO audit_log (user_id, operation, record_id, change_time)

          VALUES (USER, 'UPDATE', :NEW.id_column, SYSDATE);


  ELSIF DELETING THEN

          INSERT INTO audit_log (user_id, operation, record_id, change_time)

          VALUES (USER, 'DELETE', :OLD.id_column, SYSDATE);

  END IF;

END;

/
```

Output:

User_ID	Operation	Record_ID	Change_Time
SYSTEM	INSERT	1	2024-09-19 10:10:00
SYSTEM	UPDATE	1	2024-09-19 10:15:00
SYSTEM	DELETE	1	2024-09-19 10:20:00

Program 6

Write a code in PL/SQL to implement a trigger that automatically calculates and updates a running total column for a table whenever new rows are inserted.

```
CREATE OR REPLACE TRIGGER update_running_total

AFTER INSERT ON table_name

FOR EACH ROW

BEGIN

  -- Update the running total column in the total_table

  UPDATE total_table

  SET running_total = running_total + :NEW.value_column

  WHERE total_id = :NEW.total_id;

END;

/
```

Output:

```
Total_ID   Running_Total

1          1500
```

Program 7

Write a code in PL/SQL to create a trigger that validates the availability of items before allowing an order to be placed, considering stock levels and pending orders.

```
CREATE OR REPLACE TRIGGER validate_item_availability

BEFORE INSERT ON orders

FOR EACH ROW

DECLARE

  v_stock_level NUMBER;

  v_pending_orders NUMBER;

BEGIN

  SELECT stock INTO v_stock_level FROM inventory WHERE item_id = :NEW.item_id;

  -- Check pending orders

  SELECT SUM(quantity) INTO v_pending_orders

  FROM orders

  WHERE item_id = :NEW.item_id AND status = 'Pending';

  -- Ensure stock is available for the order

  IF v_stock_level - v_pending_orders < :NEW.order_quantity THEN

          RAISE_APPLICATION_ERROR(-20004, 'Insufficient stock available for this
order.');

  END IF;

END;

/
```

Output:

```
ORA-20004: Insufficient stock available for this order.
```