	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No. 08
PROBLEM STATEMENT	Implimentation of Triggers and Cursors in SQL/Oracle.			

ALGORITHM & CODE :

TRIGGERS:

In Oracle, triggers are PL/SQL blocks that execute automatically when a specified event occurs on a table, view, schema, or database. Oracle supports various types of triggers, including:

1. DML Triggers: Triggered by Data Manipulation Language (DML) events like INSERT, UPDATE, and DELETE.

2. DDL Triggers: Triggered by Data Definition Language (DDL) events like CREATE, ALTER and DROP.

3. INSTEAD OF TRIGGERS: Defined on views to redirect DML operations to the underlying tables.

4. Compound Triggers: Allow combining multiple timing points in a single trigger.

Syntax for creating a Trigger (Oracle SQL Example):

```
CREATE OR REPLACE TRIGGER trigger_name
BEFORE|AFTER|INSTEAD OF
INSERT OR UPDATE OR DELETE
ON table_name
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :
	Signature of Student Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

FOR EACH ROW
BEGIN
-- Triggers logic here
END;

Examples:

1. BEFORE INSERT Trigger: Automatically assigns a default value to a column before inserting a new row.

```
CREATE OR REPLACE TRIGGER before_insert_example
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
    :NEW.hire_date := SYSDATE; -- Set the current system date before insertion END;
```

Example Usage:

```
INSERT INTO employees (emp_id, name) VALUES (101,
                                    'John Doe');
```

The hire_date column will be automatically set to the current date.

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :
	Signature of Student Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.
OBLEM STATEMENT				

LGORITHM & CODE :

2. AFTER UPDATE Triggers: Logs changes made to a table into an audit table.

```
CREATE OR REPLACE TRIGGER after_update_example
AFTER UPDATE ON employees
FOR EACH ROW
BEGIN
    INSERT INTO employee_audit (emp_id, od_salary,
new_salary, change_date)
    VALUES (:OLD.emp_id , :OLD.salary , :NEW.
Salary, SYSDATE);
END;
/
```

Example Usage:

```
UPDATE employees SET salary = 60000 WHERE emp_id = 101;
```

3. BEFORE DELETE Trigger: Prevents deletion of employees in a specific department.

```
CREATE OR REPLACE TRIGGER before_delete_example
BEFORE DELETE ON employees
FOR EACH ROW
```

INPUT GIVEN		
OUTPUT OBTAINED		
REMARKS		
GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

OBLEM STATEMENT

LGORITHM & CODE :

```
BEGIN
    IF: OLD.department = 'HR' THEN
        RAISE_APPLICATION_ERROR(-20001,'Cannot delete
employees from HR department.');
    END IF;
END;
/
```

Example Usage:

```
DELETE FROM employees WHERE department = 'HR';
```

This will throw an error, Preventing the deletion.

4. INSTEAD OF Trigger (for Views): Allows updates on a view by redirecting changes to the underlying tables.

```
CREATE OR REPLACE TRIGGER instead_of_example
INSTEAD OF INSERT ON emp_view
FOR EACH ROW
BEGIN
    INSERT INTO employees (emp_id, name, Salary)
    VALUES (:NEW.emp_id, :NEW.name, :NEW.Salary);
END;
/
```

Example Usage:

```
INSERT INTO emp_view (emp_id, name, Salary) VALUES
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty / Date : / Signature of Student / Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

OBLEM STATEMENT

GORITHM & CODE :

(102, 'Alice', 70000);

This will insert the data into the employees table instead.

When to Use Triggers?

- Audit logging
- Enforcing business rules
- Automatically updating derived values
- Restricting certain operations

CURSORS :

A Cursor in Oracle is a pointer to a SQL result set. It is used to retrieve and Process multiple rows one at a time.

Types of Cursors

Implicit Cursor - Automatically created for SELECT, INSERT, UPDATE or DELETE statements.

Explicit Cursor - Defined by the User for handling query results row by row.

Cursor FOR Loop - A Simpler way to iterate through an explicit Cursor.

NPUT GIVEN

UTPUT OBTAINED

EMARKS

GRADE :	Signature of Faculty Date :	Signature of Student Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.
PROBLEM STATEMENT				

ALGORITHM & CODE :

Parameterized Cursor - Accepts Parameters to filter results dynamically.

REF Cursor - A dynamic cursor that allows fetching data dynamically at runtime.

1. Implicit Cursor Example:

Oracle automatically creates an implicit cursor for INSERT, UPDATE, DELETE or SELECT INTO statements.

```
DECLARE
   V_total NUMBER;
BEGIN
   SELECT COUNT(*) INTO V_total FROM employees;
   DBMS_OUTPUT.PUT_LINE ('Total Employees: ' || V_total);
END;
```

2. Explicit Cursor Example

Used when a query returns multiple rows, allowing row-by-row processing.

```
DECLARE
   --Declare cursor
   CURSOR emp_cursor IS
      SELECT emp_id, name, salary FROM employees;
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date :
	Signature of Student Date :

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
-- Variable to Store fetched values
   V_emp_id employees.emp_id%TYPE;
   V_name employees.name%TYPE;
   V_salary employees.Salary%TYPE;

BEGIN
  --Open Cursor
    OPEN emp_cursor;
    -- fetch rows one by one
  LOOP
    FETCH emp_cursor INTO V_emp_id, V_name, V_Salary;
    EXIT WHEN emp_cursor%NOTFOUND; -- Exit if no more rows
    DBMS_OUTPUT.PUT_LINE('ID:' || V_emp_id || ', Name:' || V_name
|| ', Salary:' || V_salary);
      END LOOP;
  --Close Cursor
    CLOSE emp_cursor;
END;
```

3. Cursor FOR LOOP Example

A Simpler way to use an explicit Cursor without manually opening, fetching, and closing.

```
DECLARE
    CURSOR emp_cursor IS
        SELECT emp_id,name, salary FROM employees;
```

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	
GRADE :	Signature of Faculty Date : / Signature of Student Date :

	Subject :		Software :	
			Hardware :	
	Branch :	Semester :	Page No.	Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
BEGIN
    FOR emp_rec IN emp_cursor LOOP
      DBMS_OUTPUT.PUT_LINE('ID: '||emp_rec.emp_id||',
Name: '||emp_rec.name||', Salary: '||emp_rec.salary);
    END LOOP;
END;
```

4. Parameterized Cursor Example

A cursor that accepts Parameter

```
DECLARE
    CURSOR emp_cursor(p_min_salary NUMBER) IS
      SELECT emp_id, name, salary from employees WHERE
salary > p_min_salary;
BEGIN
    For emp_rec IN emp_cursor(50000) LOOP
      DBMS.OUTPUT.PUT_LINE('ID: '||emp_rec.emp_id||
', Name: '||emp_rec.name||', Salary: '||emp_rec.salary);
    END LOOP;
END;
```

5. REF Cursor Example (Dynamic Cursor):

A REF CURSOR allows fetching records dynamically at runtime.

INPUT GIVEN	
OUTPUT OBTAINED	
REMARKS	

GRADE :	Signature of Faculty Date :	Signature of Student Date :

Subject :	Software :
	Hardware :
Branch : / Semester :	Page No. / Prog No.

PROBLEM STATEMENT

ALGORITHM & CODE :

```
DECLARE
    TYPE emp_ref_cursor IS REF CURSOR;
    v_emp_cursor emp_ref_cursor;
    v_emp_id employees.emp_id%TYPE;
    v_name employees.name%TYPE;
    v_salary employees.salary%TYPE;
BEGIN
  -- open REF Cursor dynamically
OPEN v_emp_cursor FOR SELECT emp_id, name,
salary FROM employees
WHERE salary > 60000;
LOOP
  FETCH v_emp_cursor INTO v_emp_id, v_name,
v_salary;
  EXIT WHEN v_emp_cursor%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE('ID:' || v_emp_id ||', Name:
' || v_name ||', Salary: ' || v_salary);
END LOOP;
-- Close Cursor
  CLOSE v_emp_cursor;
END;
```

PUT GIVEN

TPUT OBTAINED

MARKS

ADE :

Signature of Faculty
Date :

Signature of Student
Date :

Subject :		Software :	
		Hardware :	
Branch :	Semester :	Page No.	Prog No.

OBLEM STATEMENT

ORITHM & CODE :

Cursor Attributes:

Attribute	Description
%FOUND	Returns TRUE if a row is fetched.
%NOTFOUND	Returns TRUE if no row is found.
%ROWCOUNT	Returns the number of fetched rows.
%ISOPEN	Returns TRUE if the cursor is open

Example:

```
If emp_cursor %FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('Record found');
END IF;
```

When to use Cursor?

- Fetching multiple rows one by one
- Processing large result sets in PL/SQL
- Performing row by row operations
- Dynamically retrieving record in stored Procedures.

IVEN

OBTAINED

KS

Signature of Faculty Date :	Signature of Student Date :