**Phase 7: Advanced Database Programming and Auditing**

This phase emphasizes advanced programming techniques in PL/SQL to ensure system efficiency, data security, and functionality enhancement. Below is the step-by-step implementation of Phase 7 for the Course Management System (CMS):

**1. Problem Statement Development**

* **Triggers**: To enforce business rules and automate workflows (e.g., logging attendance updates).
* **Cursors:** To process multi-row queries efficiently (e.g., calculate average grades).
* **Packages:** To group related procedures and improve code re-usability.
* **Auditing:** To monitor and restrict access to sensitive data.

1. **Trigger Implementation**
2. **BEFORE Trigger**

Enforces attendance status entry.

CREATE OR REPLACE TRIGGER before\_attendance\_insert

BEFORE INSERT ON ATTENDANCE

FOR EACH ROW

BEGIN

IF :NEW.Status NOT IN ('Present', 'Absent') THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Invalid status. Must be Present or Absent.');

END IF;

END;

/

1. **Compound Trigger**

* Handles multi-row SUBMISSION updates for ensuring deadlines.

CREATE OR REPLACE TRIGGER submission\_deadline

BEFORE INSERT OR UPDATE ON SUBMISSION

FOR EACH ROW

DECLARE

due\_date ASSIGNMENT.Due\_Date%TYPE;

BEGIN

SELECT Due\_Date

INTO due\_date

FROM ASSIGNMENT

WHERE Assignment\_ID = :NEW.Assignment\_ID;

IF :NEW.Submission\_Date > due\_date THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Submission past the due date.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Assignment ID not found.');

END;

/

**3. Cursor Usage**

* It will Calculate the average grade for each student.

DECLARE

CURSOR student\_grades IS

SELECT g.Student\_ID, s.Name, AVG(g.Grade\_Value) AS Avg\_Grade

FROM GRADE g

JOIN STUDENT s ON g.Student\_ID = s.Student\_ID

GROUP BY g.Student\_ID, s.Name;

v\_student\_id STUDENT.Student\_ID%TYPE;

v\_student\_name STUDENT.Name%TYPE;

v\_avg\_grade NUMBER;

BEGIN

OPEN student\_grades;

LOOP

FETCH student\_grades INTO v\_student\_id, v\_student\_name, v\_avg\_grade;

EXIT WHEN student\_grades%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Student ID: ' || v\_student\_id || ' | Name: ' || v\_student\_name || ' | Average Grade: ' || v\_avg\_grade);

END LOOP;

CLOSE student\_grades;

END;

/

**4. Attributes (%TYPE and %ROWTYPE)**

Example: Reuse attributes for efficient code.

DECLARE

v\_student\_rec STUDENT%ROWTYPE;

BEGIN

SELECT \* INTO v\_student\_rec FROM STUDENT WHERE Student\_ID = 1;

DBMS\_OUTPUT.PUT\_LINE('Student Name: ' || v\_student\_rec.Name);

END;

/

**5. Package Development**

Package Specification:

Defines related procedures for better organization.

CREATE OR REPLACE PACKAGE cms\_package AS

PROCEDURE log\_audit(p\_table\_name VARCHAR2, p\_action\_type VARCHAR2);

PROCEDURE update\_course\_capacity(p\_course\_id INT);

END cms\_package;

/

**Package Body:**

Implements the procedures.

CREATE OR REPLACE PACKAGE BODY cms\_package AS

PROCEDURE log\_audit(p\_table\_name VARCHAR2, p\_action\_type VARCHAR2) IS

BEGIN

INSERT INTO AUDIT\_LOG (Table\_Name, Action\_Type, Changed\_By, Change\_Date)

VALUES (p\_table\_name, p\_action\_type, USER, SYSDATE);

END log\_audit;

PROCEDURE update\_course\_capacity(p\_course\_id INT) IS

BEGIN

UPDATE COURSE

SET Seats\_Available = Seats\_Available - 1

WHERE Course\_ID = p\_course\_id;

END update\_course\_capacity;

END cms\_package;

/

1. **Auditing and Restrictions**

Auditing Example: Track changes to sensitive data.

CREATE OR REPLACE TRIGGER audit\_sensitive\_data

AFTER UPDATE OR DELETE ON STUDENT

FOR EACH ROW

DECLARE

v\_action\_type VARCHAR2(10);

BEGIN

IF DELETING THEN

v\_action\_type := 'DELETE';

ELSIF UPDATING THEN

v\_action\_type := 'UPDATE';

END IF;

INSERT INTO AUDIT\_LOG (Table\_Name, Action\_Type, Changed\_By, Change\_Date)

VALUES ('STUDENT', v\_action\_type, USER, SYSDATE);

END;

/

**Restriction Example**: Limit access based on roles.

BEGIN

IF SYS\_CONTEXT('USERENV', 'SESSION\_USER') != 'ADMIN\_ROLE' THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Unauthorized access.');

END IF;

END;

/

**End**