CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE &ENGINNERING

Bachelor of Engineering (Computer Science & Engineering)

Database Management System and CST-227

Prepared By: Ms. Heena Arora

Database Management System

Course Outcome

CO Number	Title	Level
CO1	Learn the fundamentals of database systems design and draw ER diagram for the real world problems.	Understand
CO2	Design and query database using SQL.	Apply
CO3	Analyze and apply concepts of normalization to relational database design	Understand
CO4	Learn the concept of transaction, concurrency and recovery.	Remember

CONTROL SRUCTURE

Introduction to PL/SQL

Different types of Statements

Cursors, Views, Triggers

Syntax for creating triggers

Types of triggers

Need of PL/SQL

- Acts as a host language for stored procedures and triggers.
- Provides the capability to add middle tier business logic to client/server applications.
- Provides Portability of code from one environment to another
- Improves performance of multi-query transactions.
- Provides error handling

Structure of PL/SQL

- PL/SQL is Block Structured
- A block is the basic unit from which all PL/SQL programs are built. A block can be named (functions and procedures) or anonymous
- Sections of block
- Declaration Section
- Executable Section
- Exception Section

Example for Structure of PL/SQL

```
DECLARE
                    a number;
                    text1 varchar2(20);
        text2 varchar2(20) := "HI";
BEGIN
        --------- --------- ---------
END;
```

PL/SQL Control Structures

- PL/SQL Control structures includes:
- Sequential statements
- Iterative statements

PL/SQL Control Structure

- **Conditional Controls**
  - IF....THEN....END IF;
  - IF....THEN...ELSE....END IF;
  - IF....THEN...ELSIF....THEN....ELSE....END IF;

PL/SQL Control Structure : Loop

- This is used to repeatedly execute a set of statements. This is the simplest form of looping structures.
    - LOOP
    - Statements;
    - EXIT [WHEN condition];
    - END LOOP;

PL/SQL Control Structure : NestedLoop

- It is possible to have a loop within another loop. When a loop is placed within another loop it is called as nested loop. The inner loop is executed for each iteration of outer loop.
- The following example will display table up to 10 for numbers from 1 to 5.
    - LOOP ...
        - LOOP ...
            - EXIT outerloop WHEN ...
    - –
        - END LOOP; ...
    - END LOOP;

Nested Loop: Example

```
declare
    i number(2);
    j number(2);
begin
        i := 1;
loop
        j:= 1;
        loop
                dbms_output.put_line(i || '*' || j || '=' || i * j);
                j := j + 1;
                exit when j > 10;
        end loop;
        i := i + 1;
        exit when i > 5;
end loop;
end;
```

PL/SQL Control Structure : While

- Executes a series of statements as long as the given condition is true.
-
- WHILE condition LOOP
- Statements;
- END LOOP;

While : Example

```
declare
        i number(2) := 1;
    begin
        while i <= 10 loop
            dbms_output.put_line(i);
            i := i + 1;
        end loop;
    end;
```

PL/SQL Control Structure : For

- This looping structure is best suited to cases where we have to repeatedly execute a set of statements by varying a variable from one value to another.

- FOR counter IN [REVERSE] lower range .. Upper range
- LOOP
- Statements;
- END LOOP;

For : Example

```
Declare
                i int=1;
        Begin
                for i in 1..10 loop
                        dbms_output.put_line(i);
                end loop;
        end;
```

Cursors

- Cursor is used to point the context area.
- There are two types of cursors.
- Explicit Cursors
- Implicit Cursors

Implicit Cursors

- Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement. Programmers cannot control the implicit cursors and the information in it.

Explicit Cursors

- Explicit cursors are programmer defined cursors for gaining more control over the **context area**. An explicit cursor should be defined in the declaration section of the PL/SQL Block. It is created on a SELECT Statement which returns more than one row.

- Working with an explicit cursor involves four steps:
  - Declaring the cursor for initializing in the memory
  - Opening the cursor for allocating memory
  - Fetching the cursor for retrieving data
  - Closing the cursor to release allocated memory

Implicit Cursor

```
DECLARE
  l_rows number(5);
BEGIN
  UPDATE employee SET salary = salary + 1000;
  IF SQL%NOTFOUND THEN
     dbms_output.put_line('None of the salaries where updated');
  ELSIF SQL%FOUND THEN l_rows := SQL%ROWCOUNT;        dbms_output.put_line('Salaries for ' || l_rows ||
        'employees are updated');
  END IF;
END
```

Explicit Cursor

```
DECLARE
    l_employees employees%ROWTYPE;
    CURSOR l_c (p_low NUMBER DEFAULT 0, p_high NUMBER DEFAULT 99) is
        SELECT * FROM employees WHERE job_id > p_low AND job_id < p_high;
BEGIN
    OPEN l_c(3,20);
    LOOP
      FETCH l_c INTO l_employees;
      EXIT WHEN l_c%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(l_employees.last_name || l_employees.job_id );
    END LOOP;
    CLOSE l_c;
END;
```

SQL Views

- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

- CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition

Triggers

- Triggers are events which fires out automatically when some events occurred at the system.
- Triggers occurred when we perform deletion, insertion, updating on the table.

USE OF TRIGGERS

- Database triggers can be used to perform any of the following:
- Audit data modification
- Log events transparently
- Enforce complex business rules
- Derive column values automatically
- Implement complex security authorizations
- Maintain replicate tables

SYNTAX OF TRIGGERS

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE|AFTER} triggering_event ON table_name
[FOR EACH ROW]
[WHEN condition]
DECLARE
Declaration statements
BEGIN
Executable statements
EXCEPTION
Exception-handling statements
END;
```

TYPES OF TRIGGERS

- Triggers may be called BEFORE or AFTER the following events:
- INSERT, UPDATE and DELETE.
- The before/after options can be used to specify when the trigger body should be fired with respect to the triggering statement. If the user indicates a BEFORE option, then Oracle fires the trigger before executing the triggering statement. On the other hand, if an AFTER is used, Oracle fires the trigger after executing the triggering statement.

STATEMENT TRIGGERS

Statement Trigger

```
CREATE OR REPLACE TRIGGER mytrig1 BEFORE DELETE OR INSERT OR UPDATE ON employee
BEGIN
IF (TO_CHAR(SYSDATE, 'day') IN ('sat', 'sun')) OR
 (TO_CHAR(SYSDATE,'hh:mi') NOT BETWEEN '08:30' AND '18:30')
THEN
RAISE_APPLICATION_ERROR(-20500, 'table is secured');
END IF;
END;
```

ROW TRIGGERS

Row Trigger:-

```
CREATE OR REPLACE TRIGGER mytrig2
AFTER DELETE OR INSERT OR UPDATE ON employee
FOR EACH ROW
BEGIN
IF DELETING THEN
INSERT INTO xemployee (emp_ssn, emp_last_name,emp_first_name, deldate)
VALUES (:old.emp_ssn, :old.emp_last_name,:old.emp_first_name, sysdate);
ELSIF INSERTING THEN
```

CONT..

```
INSERT INTO nemployee (emp_ssn, emp_last_name,emp_first_name, adddate)
VALUES (:new.emp_ssn, :new.emp_last_name,:new.emp_first_name, sysdate);
ELSE
INSERT INTO uemployee (emp_ssn, emp_address, up_date)
VALUES (:old.emp_ssn, :new.emp_address, sysdate);
END IF;
END;
```

Assessment Pattern

S.No.	Item	Number/semester	Marks	System
1	MSTs	2	24 (12 each)	Combined tests
2	Quiz	1	4	Once online
3	Surprise test	1	3	Teacher decides
4	Assignments	3 (one per unit)	4	By teacher as per the dates specified
5	Tutorials	Depending on classes	3	In tutorial classes
6	Attendance	Above 90%	2	
Internal (division as mentioned above points 1-6)			**40**	
External			**60**	
Total			**100**	

REFERENCES

- “Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles” by Narasimha Karumanchi
- “Data Structures Using C” by Aaron M. Tenenbaum
- “Data Structures and Algorithms” by Alfred V. Aho
- “Fundamentals of Data Structures in C by Ellis Horowitz, Sartaj Sahni, and Susan Anderson-Freed

THANK YOU

For queries

Email: Heenae7725@cumail.in