MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part A-04 & A-05**

PART: A) ASSIGNMENT NO: 04, 05

AIM: **Unnamed PL/SQLcode block**: Use of Control structure and Exception handling is mandatory.

OBJECTIVES:

- To develop basic, intermediate and advanced Database programming skills.
- To learn the concept of procedural language.
- To learn Control structure and Exception handling.

APPRATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative
- Database: MySQL/ Oracle 11g Database.

THEORY:

PL/SQL Introduction:

- PL SQL basically stands for "Procedural Language extensions to SQL".
- It is Extension of Structured Query Language (SQL) that is used in Oracle.
- Unlike SQL, PL/SQL allows the programmer to write code in procedural format.
- It combines the data manipulation power of SQL with the processing power of procedural language to create a super powerful SQL queries.
- It allows the programmers to instruct the compiler 'what to do' through SQL and 'how to do' through its procedural way.
- PL/SQL is a completely portable, high-performance transaction-processing language.
- PL/SQL provides a built-in, interpreted and OS independent programming environment.

DECLARE **Variable declaration** **(Optional)**
BEGIN **Program Execution** **SQL Statement** **(Mandatory)**
EXCEPTION **Exception handling** **(Optional)**
END; **(Mandatory)**

A. Control Statements IF:

- PL/SQL supports the programming language features like conditional statements and iterative statements.
- Its programming constructs are similar to how you use in programming languages like Java and C++.
- There are different syntaxes for the IF-THEN-ELSE statement.

Syntax: (IF-THEN statement):

```
IF condition
    THEN
        Statement: It is executed when condition is true}
END IF;
```

Syntax: (IF-THEN-ELSE statement):

```
IF condition
    THEN
        {statements to execute when condition is TRUE...}
    ELSE
        {statements to execute when condition is FALSE...}
    END IF;
```

Syntax: (IF-THEN-ELSIF statement):

```
IF condition1
    THEN
        {statements to execute when condition1 is TRUE...}
        ELSIF condition2
```

THEN

{statements **to execute when** condition2 **is TRUE**...}

END IF;

Syntax: (IF-THEN-ELSIF-ELSE statement):

IF condition1

THEN

{statements **to execute when** condition1 **is TRUE**...}

ELSIF condition2

THEN

{statements **to execute when** condition2 **is TRUE**...}

ELSE

{statements **to execute when** both condition1 and condition2 are **FALSE**...}

END IF;

B. PL/SQL Loop:

- The PL/SQL loops are used to repeat the execution of one or more statements for specified number of times.
- These are also known as iterative control statements.
- **Syntax for a basic loop:**

LOOP

Sequence of statements;

END LOOP;

- **Types of PL/SQL Loops**

1. **Basic Loop / Exit Loop**

LOOP

statements;

EXIT;

{or EXIT **WHEN** condition;}

END LOOP;

2. **While Loop**

WHILE <condition>

LOOP statements;

END LOOP;

Important steps to follow when executing a while loop:

✓ Initialise a variable before the loop body.

✓ Increment the variable in the loop.

- ✓ EXIT WHEN statement and EXIT statements can be used in while loops but it's not done oftenly.

3. **For Loop**

```
FOR counter IN initial_value .. final_value LOOP
        LOOP statements;
    END LOOP;
```

- ✓ initial_value : Start integer value
- ✓ final_value : End integer value

Important steps to follow when executing a while loop:

- ✓ The counter variable is implicitly declared in the declaration section, so it's not necessary to declare it explicity.
- ✓ The counter variable is incremented by 1 and does not need to be incremented explicitly.
- ✓ EXIT WHEN statement and EXIT statements can be used in FOR loops but it's not done oftenly.

4. **GOTO Statement**

```
BEGIN
    ...
        GOTO insert_row;
    ...
    <<insert_row>>
    INSERT INTO emp VALUES ...
END;
```

C. Exception Handling

- An exception is an error condition during a program execution. PL/SQL supports programmers to catch such conditions using EXCEPTION block in the program and an appropriate action is taken against the error condition. There are two types of exceptions.

1. **System-defined exceptions**
2. **User-defined exceptions**

```
DECLARE
    <declarations section>
        BEGIN
            <executable command(s)>
```

```
    EXCEPTION
        <exception handling goes here >
WHEN exception1 THEN
        exception1-handling-statements
WHEN exception2 THEN
        exception2-handling-statements
WHEN exception3 THEN
        exception3-handling-statements
WHEN others THEN
        exception3-handling-statements
END;
```

D. Raising Exceptions

- Exceptions are raised by the database server aquatically whenever there is any internal database error, but exceptions can be raised explicitly by the programmer by using the command RAISE. Following is the simple syntax for raising an exception:

```
DECLARE
    exception_name EXCEPTION;
BEGIN
    IF condition THEN
        RAISE exception_name;
END IF;
EXCEPTION
    WHEN exception_name THEN   statement;
END;
```

IMPLEMENTATION:

Consider Tables:

1. Borrower (Roll_no, Name, Date of Issue, Name of Book, Status)
2. Fine (Roll_no, Date, Amt)

- Accept Roll_no and Name of Book from user.
- Check the number of days (from date of issue).
- If days are between 15 to 30 then fine amount will be Rs 5per day.
- If no.of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs 5 per day.
- After submitting the book, status will change from I to R.

- If condition of fine is true, then details will be stored into fine table.
- Also handles the exception by named exception handler or user define exception handler.

OR

Write a **PL/SQL code block** to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

Note: Instructor will frame the problem statement for writing PL/SQL block in line with above statement.

CONCLUSION:

QUESTIONS:

1. What are the advantages of PLSQL over SQL?
2. List Different Pre-defined Exceptions.
3. Explain User-defined exceptions.

MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part A-06**

PART: A) ASSIGNMENT NO: 06

AIM: **Named PL/SQL Block:** PL/SQL Stored Procedure and Stored Function.

OBJECTIVES:

- To develop basic, intermediate and advanced Database programming skills.
- To learn the concept of procedural language.
- To learn stored procedure and stored function in PL/SQL.

APPRATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative
- Database: MySQL/ Oracle 11g Database.

THEORY:

A. PL/SQL Stored Procedure

- The PL/SQL stored procedure or simply a procedure is a PL/SQL block which performs one or more specific tasks. It is just like procedures in other programming languages.
- A procedure may or may not return any value
- The procedure contains a header and a body.

✓ **Header:** The header contains the name of the procedure and the parameters or variables passed to the procedure.

✓ **Body:** The body contains a declaration section, execution section and exception section similar to a general PL/SQL block.

- **Procedures: Passing Parameters**

❖ **IN parameters:**

✓ The IN parameter can be referenced by the procedure or function.

✓ This parameter is used for giving input to the subprograms.

- ✓ It is a read-only variable inside the subprograms, their values cannot be changed inside the subprogram

❖ **OUT parameters:**

- ✓ The OUT parameter cannot be referenced by the procedure or function.
- ✓ This parameter is used for getting output from the subprograms.
- ✓ It is a read-write variable inside the subprograms, their values can be changed inside the subprograms.

❖ **INOUT parameters:**

- ✓ The INOUT parameter can be referenced by the procedure or function.
- ✓ This parameter is used for both giving input and for getting output from the subprograms.
- ✓ It is a read-write variable inside the subprograms, their values can be changed inside the subprograms.

- **PL/SQL Create Procedure**

Syntax for creating procedure:

```
CREATE [OR REPLACE] PROCEDURE procedure_name  [ (parameter [,parameter]) ]
 IS
        [Declaration_section]
BEGIN
         Executable_section
        [EXCEPTION
         Exception_section]
END [procedure_name];
 /
```

Syntax for drop procedure

```
DROP PROCEDURE procedure_name
```

B. PL/SQL Function

- The PL/SQL Function is very similar to PL/SQL Procedure.
- The main difference between procedure and a function is, a function must always return a value, and on the other hand a procedure may or may not return a value.
- Except this, all the other things of PL/SQL procedure are true for PL/SQL function too.

Syntax to create a function:

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
        < function_body >
END [function_name];
/
```

Syntax for removing your created function:

```
DROP FUNCTION function_name;
```

IMPLEMENTATION:

Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks899and 825 category is Higher Second Class.

Write a PL/SQLblock to use procedure created with above requirement.

Stud_Marks(name, total_marks)

Result(Roll,Name, Class)

Note: Instructor will frame the problem statement for writing stored procedure and function in line with above statement.

CONCLUSION:

QUESTIONS:

1. What is a Stored Procedure?
2. Describe the use of %ROWTYPE and %TYPE in SQL?
3. Explain IN, OUT, IN-OUT mode in stored procedure.
4. What is a Stored Function?
5. What is difference between stored functions and stored procedures?

MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part A-07**

PART: A) ASSIGNMENT NO: 07

AIM: **Cursors:** (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor)

OBJECTIVES:

- To develop basic, intermediate and advanced Database programming skills.
- To learn the concept of procedural language.
- To study cursor programming and different cursor operations.

APPRATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative
- Database: MySQL/ Oracle 11g Database.

THEORY:

A. PL/SQL Cursor

- When an SQL statement is processed, Oracle creates a memory area known as context area.
- A cursor is a pointer to this context area.
- It contains all information needed for processing the statement.
- In PL/SQL, the context area is controlled by Cursor.
- A cursor contains information on a select statement and the rows of data accessed by it.
- A cursor can hold more than one row, but can process only one row at a time. The set of rows the cursor holds is called the active set.

PL/SQL Implicit Cursors

- The implicit cursors are automatically generated by Oracle while an SQL statement is executed, if you don?t use an explicit cursor for the statement.
- These are created by default to process the statements when DML statements like INSERT, UPDATE, DELETE etc. are executed.

- Orcale provides some attributes known as Implicit cursor?s attributes to check the status of DML operations.

  %FOUND, %NOTFOUND, %ROWCOUNT and %ISOPEN.

Attributes	Return Value
%FOUND **SQL%FOUND**	The return value is TRUE, if the DML statements like INSERT, DELETE and UPDATE affect at least one row and if SELECT ….INTO statement return at least one row.
	The return value is FALSE, if DML statements like INSERT, DELETE and UPDATE do not affect row and if SELECT….INTO statement do not return a row
%NOTFOUND **SQL%NOTFOUND**	The return value is FALSE, if DML statements like INSERT, DELETE and UPDATE at least one row and if SELECT ….INTO statement return at least one row.
	The return value is TRUE, if a DML statement likes INSERT, DELETE and UPDATE do not affect even one row and if SELECT ….INTO statement does not return a row.
%ROWCOUNT **SQL%ROWCOUNT**	Return the number of rows affected by the DML operations INSERT, DELETE, UPDATE, SELECT.
%ISOPEN **SQL%ISOPEN**	It always returns FALSE for implicit cursors, because the SQL cursor is automatically closed after executing its associated SQL statements.

PL/SQL Explicit Cursors

- The Explicit cursors are defined by the programmers to gain more control over the context area.
- These cursors 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.
- General Syntax for creating a cursor:

  CURSOR cursor_name **IS** select_statement;;

- Steps:
  1. Declare the cursor to initialize in the memory.
  2. Open the cursor to allocate memory.
  3. Fetch the cursor to retrieve data.
  4. Close the cursor to release allocated memory.

❖ **1) Declare the cursor:**

It defines the cursor with a name and the associated SELECT statement.

CURSOR name **IS** SELECT statement;

- **2) Open the cursor:**

It is used to allocate memory for the cursor and make it easy to fetch the rows returned by the SQL statements into it.

OPEN cursor_name;

- **3) Fetch the cursor:**

It is used to access one row at a time. You can fetch rows from the above-opened cursor as follows:

FETCH cursor_name INTO variable_list;

- **4) Close the cursor:**

It is used to release the allocated memory. The following syntax is used to close the above-opened cursors.

CLOSE cursor_name;

IMPLEMENTATION:

Write a PL/SQL block of code using parameterized Cursor that will merge the data available in the newly created table N_Roll Call with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.

Note: Instructor will frame the problem statement for writing PL/SQL block using all types of Cursors in line with above statement.

CONCLUSION:

QUESTIONS:

1. What are different types of cursor? Explain each type with syntax.
2. What are the different attributes of cursor?
3. What is the parameterized cursor?

MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student:	**Class:**
Semester/Year:	**Roll No:**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part A-08**

PART: A) ASSIGNMENT NO: 08

AIM: **Database Trigger** (All Types: Row level and Statement level triggers, Before and After Triggers).

OBJECTIVES:

- To develop basic, intermediate and advanced Database programming skills.
- To learn the concept of procedural language.
- To learn various Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers).

APPRATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative
- Database: MySQL/ Oracle 11g Database.

THEORY:

PL/SQL Trigger

- A database trigger is a stored procedure that automatically executes whenever an event occurs. The event may be insert-delete-update operations.
- Trigger is invoked by Oracle engine automatically whenever a specified event occurs.
- Trigger is stored into database and invoked repeatedly, when specific condition match.
- Triggers could be defined on the table, view, schema, or database with which the event is associated.
- A procedure is executed explicitly from another block via a procedure call with passing arguments,
- While a trigger is executed (or fired) implicitly whenever the triggering event (DML: INSERT, UPDATE, or DELETE) happens, and a trigger doesn't accept arguments.
- Triggers are written to be executed in response to any of the following events.

- ✓ A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).
- ✓ A database definition (DDL) statement (CREATE, ALTER, or DROP).
- ✓ A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

- **Type of Triggers**

✓ **BEFORE Trigger:** BEFORE trigger execute before the triggering DML statement (INSERT, UPDATE, DELETE) execute. Triggering SQL statement is may or may not execute, depending on the BEFORE trigger conditions block.

✓ **AFTER Trigger:** AFTER trigger execute after the triggering DML statement (INSERT, UPDATE, DELETE) executed. Triggering SQL statement is execute as soon as followed by the code of trigger before performing Database operation.

✓ **ROW Trigger:** ROW trigger fire for each and every record which are performing INSERT, UPDATE, DELETE from the database table. If row deleting is define as trigger event, when trigger file, deletes the five rows each times from the table.

✓ **Statement Trigger:** Statement trigger fire only once for each statement. If row deleting is define as trigger event, when trigger file, deletes the five rows at once from the table.

- **Syntax of Trigger**

```
CREATE [OR REPLACE ] TRIGGER trigger_name
        {BEFORE | AFTER | INSTEAD OF }
        {INSERT [OR] | UPDATE [OR] | DELETE}
        [OF col_name]
        ON table_name
        [REFERENCING OLD AS o NEW AS n]
        FOR EACH ROW | FOR EACH STATEMENT [ WHEN Condition ]
        WHEN (condition)
DECLARE
        Declaration-statements
BEGIN
        Executable-statements
EXCEPTION
        Exception-handling-statements
END;
```

- ✓ **CREATE [OR REPLACE] TRIGGER trigger_name**: It creates or replaces an existing trigger with the trigger_name.
- ✓ **{BEFORE | AFTER | INSTEAD OF} :** This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
- ✓ {**INSERT [OR] | UPDATE [OR] | DELETE}:** This specifies the DML operation.
- ✓ **[OF col_name]:** This specifies the column name that would be updated.
- ✓ **[ON table_name]:** This specifies the name of the table associated with the trigger.
- ✓ **[REFERENCING OLD AS o NEW AS n]:** This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
- ✓ **[FOR EACH ROW]:** This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
- ✓ **WHEN (condition):** This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers

IMPLEMENTATION:

Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

Note: Instructor will Frame the problem statement for writing PL/SQLblock for all types of Triggers in line with above statement.

CONCLUSION:

QUESTIONS:

1. What is a trigger?
2. What are Benefits of Triggers?
3. What are Row triggers and Statement triggers?
4. Why are we using Before and After triggers?
5. What is Insert, Update and Delete triggers?