DataBase Management System - 22319)

Unit IV : PL/SQL Programming.

(Weightage - 16 marks)

Pranjal Save (SY-comps)

4.1 Introduction to PL/SQL, Advantages of PL/SQL, The PL/SQL Block structure, datatypes, Variables Constants.

Questions

Q1 Draw block structure of PL-SQL. (2m) ? Imp

Q2 State two advantages of PL/SQL. (2m) ? Imp

Q3 Datatypes used in PL/SQL. (2m) ?

Introduction - PL/SQL

What is PL/SQL?

⇒ PL/SQL: Procedural Language Structured query Language.

- Stands for Procedural Language extension to SQL
- In Oracle Co-operations's standard data access language for relational databases.
- Seamlessly integrates procedural constructs with SQL.

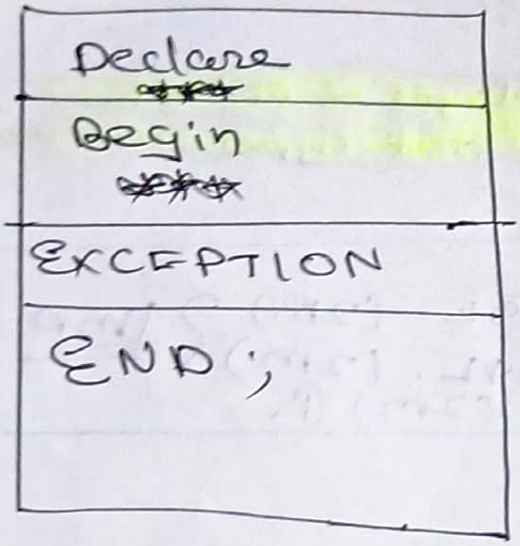
SQL + PL = PL/SQL

If/else switch case + SQL (DDL DML TCL etc) = PL/SQL (PL/SQL)

Advantages of PL/SQL (2m)

* It promotes reusability.
* It supports readibility
* Work can be divided into smaller modules so that it can be manageable and also enhances the readability of code.
* It is secure, as codes hides the internal details of user.
* It improves performance against running SQL queries multiple times.
* Block structure, Better performance, Procedural language capability.
* PL/SQL handles errors or exception effectively during the execution of PL/SQL program.

PL/SQL Block Structure



Declare
- Variable, cursors, user-defined exceptions.

BEGIN
- SQL statements.
- PL/SQL statement.

EXCEPTION
- Actions to perform when error occurs.

End
closing of program.

Name the Datatypes used in PL/SQL? (DM)

- CHAR
- VARCHAR
- Date
- Number
- Long
- Int

Helloworld program to implement Basic Structure

```
Declare
  message varchar2(50) := 'Hellow, World!';
BEGIN.
  dbms_output_line(message);
END;
```

Output

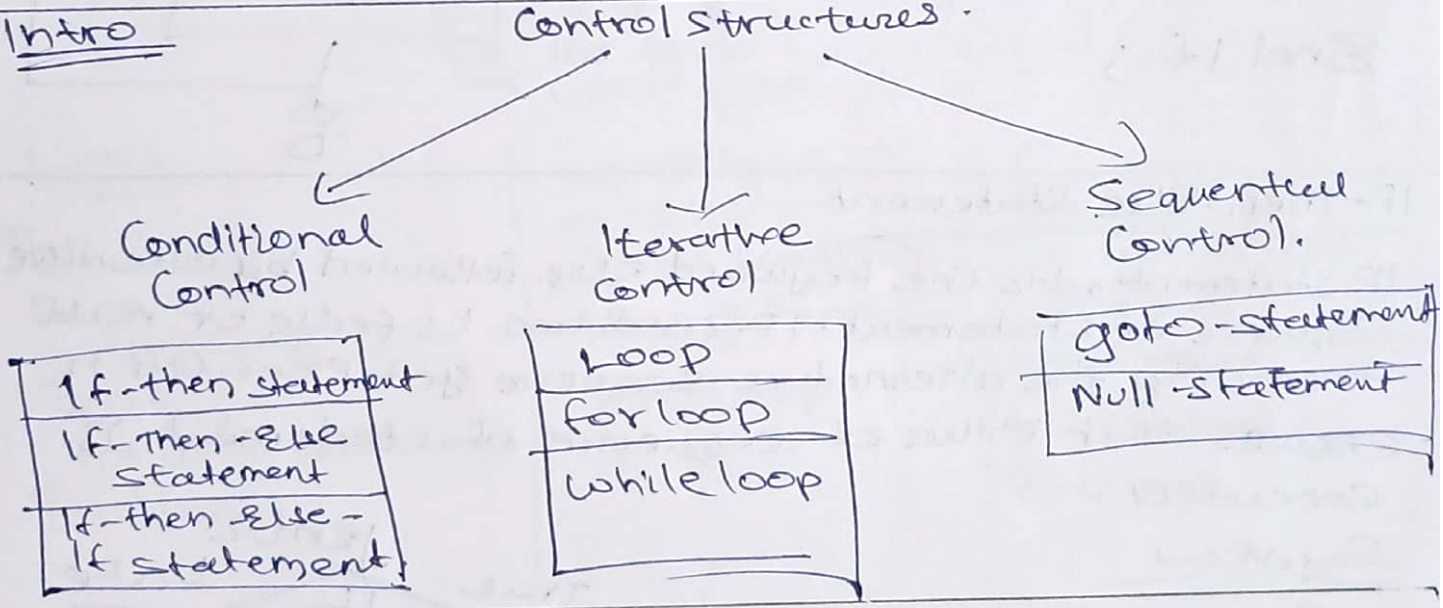
Hello, world!

= Control structures: Conditional Control, Iterative Control, Sequential Control.

stions

- state syntax of while loop command (2m)
- WAP PL/SQL to print odd no. from 1 to 10. (2m)
- Explain for loop syntax in PL/SQL with example of printing 10 to 1 reverse numbers. (6m)
- Write PL/SQL code to print reverse of a number. (4m)
- Explain Conditional Control in PL/SQL with example. (4m)
- Explain any one control structure in PL/SQL with example. (4m)
- WAP PL/SQL to print n even numbers using for loop. (6 m).
- WAP PL/SQL program to calculate factorial of given number. (6 m).
- PL/SQL If-then-else. (2m).

Intro



Control Structures

Decision making structures requires that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optinally, other statements to be executed if the condition is determined false

Conditional Control

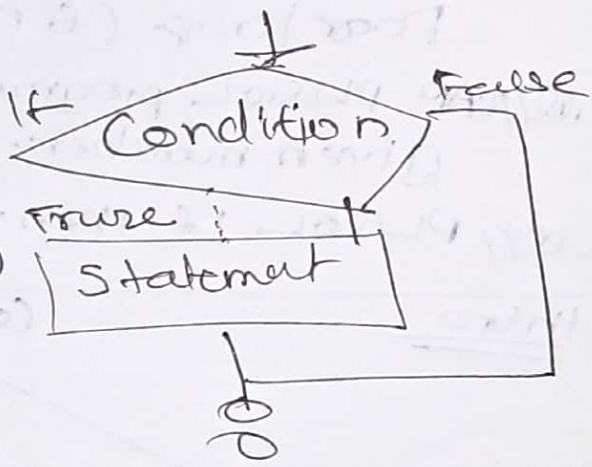
- (1) If - Then statement.
- (2) If - then - Else statement.
- (3) If - Then - Else - If statement.

IF - Then Statement

The If statement associates a condition with a sequence of statements enclosed by the keywords Then and End If.
If statement is true it get executed or condition is false then program does not processes If statement.
It is used in Begin part of Block structure.

Syntax

```
If (Condition)
Then
Statement (. . . . . )
End If ;
```

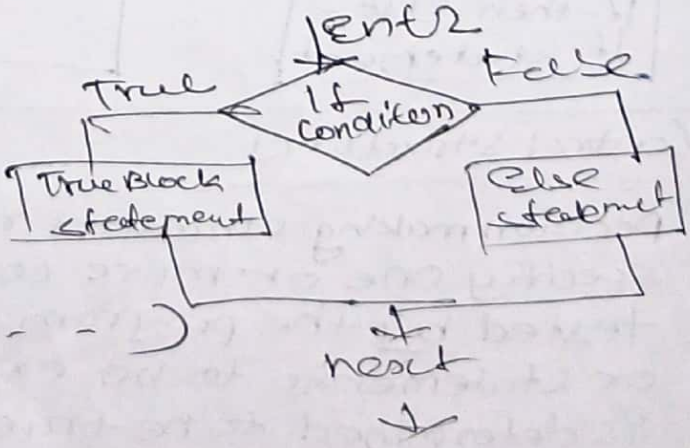


If - Then - Else Statement

IF statement adds the keyword Else followed by alternative sequence of statement. If condition is false or null then only the alternative sequence get executed It ensures that either of sequence of statement is executed.

Syntax

```
IF condition
Then
statement. . . . . ->
Else,
   statement (. . . . . )
Stop. End If.
```

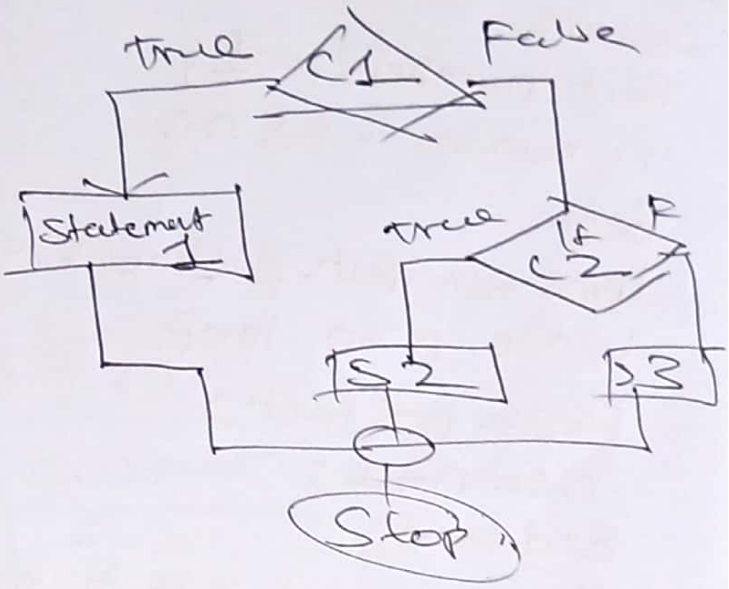


then-Else-IF (Nested)

allows to choose various alternatives.

Syntax

```
If Condition 1
    then
    (Statement)
Else If Condition 2
    then
    (Statement 2)
then End If.
```



Iteractive

The PL/SQL Loops are used to repeat the execution of one or more statements for specified number of times. These are known as iterative control statement

- Loop
- for loop
- while loop.

Loop

Syntax

```
Loop
  statemenes
  exit;
  (or when condition)
Endloop;
```

whileloop

```
while (condition)
  Loop statements;
  Endloop
```

for loop

```
for counter IN
initial value.....
final value Loop
    Statenent.
Endloop;
```

Programs

Q/ PL/SQL program to calculate factorial of given number ⇒6marks

⇒
```
Declare
 fact number := 1;
 n number := &n;
 Begin
 dbms_output.putline('Enter value of n');
  while n>0 loop
  fact:= n* fact;
   n:= n-1;
  End loop;
  dbms_output.put_line('n! = '||fact);

 End;
```

Q// Output

~~dbms_out~~ Enter value of n 2

n! = 2

Q// PL/SQL Program to print n even numbers using for loop. (6marks)

⇒
```
Declare
 A number = &A;
 B number = &B;
Begin
for I in A..... B loop
 If(Mod(I,2)=0) then
  dbms_output.put_line(I);
End if;
End loop;
End;
```

Output

1 10

2
4
6
8

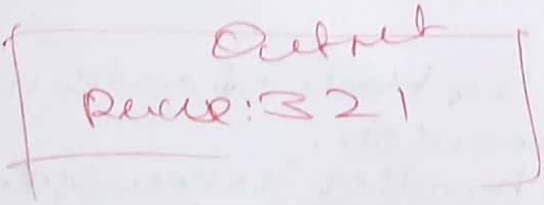
5 SQL Program to print odd. no from 1 to 10

```
declare
A number:= 1;
B number:= 10;
Begin
for I in A .... B for loop
If(Mod(I,2)=1) then
dbms_output.put_line(I);
End if;
End loop;
End;
```

Q// Write PLSQL code to print reverse of a number.

⇒

```
Declare
number := &n;
rev := 0;
r number ;
Begin
dbms_output.put_line('Enter number'
S:=123
r = mod(n,10);
rev=(rev*10)+r;
n:=n/10;
end loop;
dbms_output.putline('Reverse = '||rev);
End;
```



Q// PL SQL with example of printing 1 to 10 reverse numbers.

⇒
```
Declare
num1 number;
Begin
num1 = 01
for num1 in reverse 1..... 10
Loop
DBMS_output.put_line(num1);
Endloop;
End;
```

VIMP

Questions

43 : Exception Handling

Predefined User defined

* Exception Handling with suitable example & syntax.

⇒ Exception is a warning or error condition which may interrupt the program execution.

* It can be defined as an error situation, arising during program execution.
* An exception is raised when an error occurs, the normal execution is stopped and control transfers to exception handler code.
* Exception can be predefined or can be user defined.
* Example Zero DIVIDE & storage error are predefined exceptions.
* User defined exception can be written in declare Section of any PL/SQL block, subprogram, or package.
* Exception Handlers is a block of code written to handle the raised exception.
* After an exception handler is executed the control passes back to next statement after the statement where exception was raised.
* Handling the exception ensures the PL/SQL block do not exit

Syntax

declare
/Declaration section
Begin
Begin section

Exception
when Exception_one then.
Exception handler code to handle the error.
when Exception _two then.

when Exception_three

when other then.

End;

4.4 Cursors: Implicit and Explicit Cursors, Declaring, Open and closing of Cursors, Fetching a Record from the Cursor, Cursor for loop, Parameterized Cursors.

Questions

Q1 Write step by step syntax to create, open and close cursor in PL/SQL. (4M)

Q2 Explain cursor with example. (4M)

Q3 Explain implicit & explicit cursors. (4M)

Q4 Explain the steps for cursor implementation with syntax and example. (4M)

Q5

* When an SQL statement is executed or processed, Oracle creates an context area. A cursor is pointer to context area.
* It contains all information needed for processing the statement.
* In PL/SQL, context area is controlled by cursor.
* A cursor contains information on a select statement and rows of data accessed by it.
* A cursor is reffered to a program to fetch and process the row returns by SQL statement, one at a time.
* It is a database object to retrieve data from result set on row at an time.
* It is temporary worked area created in system when SQL statement is executed.
* It is useful when we want to manipulate the record of table in single fon method, in other word one row at a time.
* In short, cursor can hold more than one row but can pass only one row at time.
* The set of row holds by cursor is called the active set.

There are two types of cursors

Implicit

Explicit.

Implicit cursor

* These types of cursors are generated and used by the system during the manipulation of DML query.
* An implicit cursor is also generated by system when a single row is selected by select command.
* Programmers cannot control implicit cursors.

Oracle provides some attributes known to be explicit cursors attribute..

%found = atleast one record processed
%ROWCOUNT = no. of rows processed.
%ISOPEN = Used with explicit.
%Not found = no rows processed.

Program
to count no. of rows updated. (add salary of every emp by 5000)

```
Declare
  total_rows number;
Begin
update customers set salary=salary+5000;
 If sql%notfound then.
     dbms_output.putline('No records
                          updated');
 else if sql%found Then.
   total_rows = sql%rowcount;
dbms_output.put_line('total rows' || 'customers
 updated');
 End if;
 End;
```

Explicit Cursors

* An Explicit cursor is defined in declaration section of PL/SQL Block.
* It is created on a SELECT statement which statement returns more than one row.
* It is a type of cursor is created by user when the select command returns more than one row and only one row is to be processed at a time.

* An explicit cursor can more from one row to another row in result set.
* An explicit cursor uses a pointer that hold the records of row.

Def Syntax

CURSOR cursorname IS select_statement;

cursorname → Suitable name

select_statement → query.

DOFL process Code

Steps for Curs

① a) Declaring cursor;
Cursor is declared in declared section;

Syntax: cursor <cursor_name> is <select>

Exp: cursor a is select ename from emp_5 where emp_no=5;

② Opening cursor:
After the declaring the cursor, need to be open.

Syntax: open <cursor_name>;

Exp: open a;

③ fetching a record from cursor;
Once we declared and opened the we need to get records or rows from cursor.
Records are accessed using FETCH statement.

Syntax: fetch <cursor_name> into <variable_list>

Exp: fetch a into name;

④ Closing cursor
Once all processing is over cursor is closed.
using close cursor.

Syntax: close cursor_name;

Exp: close a;

Code

```
Declare
cursor emp_cursor is select * from emp
 emp_record employees%ROWtype;
 Begin
 open emp_cursor;
   Loop
 FETCH emp_cursor INTO emp_record;
 EXIT when emp_cursor%NOTfound;
 dbms_output.put_line( first_name||
 ' emp_record.lastname);
 End loop
 Close emp_cursor;
 End;
```

Procedures: Advantages, Creating, Executing and Deleting a ~~function~~ Stored Procedure

Subprogram is a program unit/module that performs particular task.

These subprogram are combined to form larger program.

* This is basically the 'Modular design'
* A subprogram can be invoked by another subprogram or program which is called the calling program.
* A stored procedure or in simple procedure is named PL/SQL A-S proc which performs one or more specified task.
* Example, Insert & Delete procedures are created.

Creating a Procedure

Create (or replace) Procedure procedure_name
[(parameter_name [IN | OUT | IN OUT] type (---))]
{IS/AS}
BEGIN.
<procedure_body>
End procedure_name;.

Insertion deletion update
Procedure
IS/AS ⇒ methods.

where,
procedure name specifies name.
(or replace) - modifying.

* The optional parameter list contains name, mode types of parameter.
IN - value will be passed inside and out will return value outside of procedure.
* procedure body contains executable part.
* AS/IS - for creating as for declare procedure
* Procedure is used in Database
  - Schema level
    - Package (table) creation
    - PL/SQL Block.

For Drop
Drop procedure
procedure_name;

4.6 Functions: Advantages, Creating, and Deletion of function.

Question

* Explain function in PL/SQL with scalable example. (6M/4M)
* State any two advantages of function in PL/SQL.

Functions

* The PL/SQL Procedure.
* The main difference between procedure and function is, a function must always returns a value & procedure may or may not return a value.
* Except this, all others thing of PL/SQL procedure are likely to PL/SQL function.
* A function is logically grouped set of SQL and PL/SQL statements that perform a specific task.
* Function must return value to caller.
* A function can return only one value to calling PL/SQL block.
* Unlike procedure, procedure can be called recursively so multiple values can be passed..

Syntax

```
Create (or replace) function functionname
[parameter value [IN | out | INOUT] type [...]]
Return return_datatype
{IS/AS}
Begin
<function_body>
End <functionname>;
```

Drop fun
fun_name

Database Triggers: Use of Database Triggers How to apply triggers, types of triggers, Syntax for Creating & Deleting triggers.

estion:

Q1/ Define database trigger. How to create and delete trigger?

Q2/ Write a trigger which invokes on deletion of record on emp table.

Q1/ State use of database triggers and state it's types.

* Triggers are stored programs, which are automatically executed or fired when some event occurs.
* Triggers are in fact, written to be executed in response to following events –
  - A DML statement.
  - DDL statement.
* Trigger is stored procedure which is called implicitly whenever, a insert, update or delete statement is fired.
* Triggers are procedures that are stored in the database and are implicitly run or fired when a DML events occurs on database, into specific table or updated columns.
* Trigger can be invoked when row is inserted

Uses of Triggers

- Auditing
- Data is generated by own.
- Prevent invalid transactions
- Reduce Redudancy.
- To edit data modification.
- Replicate table can be maintained.
- To auto increment field.
-

Types → Row level, Statement level.

Row level triggers.

Row level triggered executes once for each and every row in transaction.

Specially used for data auditing purpose.

"FOR Each Row" clause present in Create table

Example 15000 rows
↓↓
Time 15000 time execution

Statement level triggers.

Statement level execute only once for each single transaction.

Used to enforce all additional security on transactions performed on table.

"FOR EACH ROW" clause is omitted in Create Trigger.

Example 15000
↓↓
once only.

Syntax for Creating

Create (OR Replace) Trigger Trigger_name
[Before / After / Instead OF) // time
[Insert / Update / Delete) // event (OR)
on tablename [For Each Statement / for each row]
when (Condition)
PL/SQL Block.

Example

Create or replace Trigger X
Before Insert ON emp
for each row
Begin
if : new.sal < 0 then
raise_application_error
('salary should be
greater than 0');
end if;
End;
Trigger Created

Drop

Drop trigger trigger_name

r to prevent duplicate enrollment in course

te Trigger prevent_duplicate_enrollment
ore Insertion ON enrollment
R Each Row
BEGIN
If EXISTS(select 1 from enrollment
where stud_id = NEW.student id
AND course_id = NEW.course id)

Then,
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'Student is already
enrolled is done ";

End IF;

End;

Advantages of functions

- Security
- Performance
- memory allocation
- Productivity, integrity.
- Allows to reach great heights in modularity, maintainability & performance improvement.
- Work can be divided in smaller modules can be manageable

Q// Write PL/SQL Program which accept the customer id from user if user enters an invalid ID then raise the exception invalid_id is raised ~~during~~ use exception handling. (6m)

⇒ Declare
c_id number;
invalid_id_exception Exception;
BEGIN
c_id := &c_id;
if (c_id < 0) then
raise invalid_id_Exception;
End if;
EXCEPTION
when invalid_id_Exception THEN.
dbms_output.putline('Invalid customer id');
End;

Q// Write PL/SQL code to check whether specified employee is present in Emp table or not. Accept empno from user If employee does not exist display message using Exception handling. (6 marks)

⇒ Declare
no emp.empno %Type := &no;
BEGIN
Select empno into no from emp where empno = no;
DBMS_output.put_line('Emp no is present' || no);
Exception
when NO_DATA_FOUND then.
DBMS_OUTPUT.putline("Emp. no is not present");
End;

Write PL/SQL program, which accept the number from user enters an odd number then exception Invalid number is raised using user defined exception handling. (6M)

```
Declare
  user_number Number;
  Invalid number exception;
  pragma Exception_INIT(Invalid number, -20001);
Begin
  user_number := &user_number;
  IF MOD(user_number, 2) = 1
  THEN
    RAISE invalid number;
  Else
    DBMS_Output.put_line('Enter no. is even' ||
                         user_number);
  End if;
EXCEPTION
  When InvalidNumber then
    DBMS_____('ODD number');
```

Q// write a trigger which invokes on deletion of record on emp table. (4M)

⟹ create or replace trigger trg1 before delete on emp_details
column

declare
begin.
raise_application_error(-20000, 'cannot defect record');
End;

W-22

-200