DBS Labsheet-7

(Prof.R .Gururaj)

Introduction to PL-SQL

PL/SQL is **Oracle's** procedural language extension to **SQL**.

PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.

All the statements of a block are passed to oracle engine all at once which increases processing speed and decreases the traffic.

Disadvantages of SQL:

1. SQL doesn't provide the programmers with a technique of condition checking, looping and branching.
2. SQL statements are passed to Oracle engine one at a time which increases traffic and decreases speed.
3. SQL has no facility of error checking during manipulation of data.

Features of PL SQL

1. PL/SQL is basically a procedural language, which provides the functionality of decision making, iteration and many more features of procedural programming languages.
2. PL/SQL can execute a number of queries in one block using single command.
3. One can create a PL/SQL unit such as ***procedures***, ***functions***, packages, ***triggers***, and types, which are stored in the database for reuse by applications.
4. PL/SQL provides a feature to handle the exception which occurs in PL/SQL block known as exception handling block.
5. Applications written in PL/SQL are portable to computer hardware or operating system where Oracle is operational.
6. PL/SQL Offers extensive error checking.

We study more about: Stored Procedure; Triggers; Functions; and Cursors

Structure of PL/SQL Code:

PL/SQL extends SQL by adding constructs found in procedural languages, resulting in a structural language that is more powerful than SQL. The basic unit in PL/SQL is a block.

All PL/SQL programs are made up of blocks, which can be nested within each other.

Parts of PL-SQL Code:

Declare section starts with **DECLARE** keyword in which variables, constants, records as cursors can be declared which stores data temporarily. It basically consists definition of PL/SQL identifiers. This part of the code is optional.

Execution section starts with **BEGIN** and ends with **END** keyword. This is a mandatory section and here the program logic is written to perform any task like loops and conditional statements. It supports all DML commands, DDL commands and SQL*PLUS built-in functions as well.

Exception section starts with **EXCEPTION** keyword. This section is optional which contains statements that are executed when a run-time error occurs. Any exceptions can be handled in this section.

PL/SQL Stored Procedures:

A stored procedure is a **PL**/**SQL** block that **Oracle** stores **in the** database and can be called by name from an application.

When you create a stored procedure, **Oracle** parses the procedure and stores its parsed representation **in the** database.

1.A stored procedure is a named collection of procedural and SQL statements.

2.A procedure can be stored in the database.

3.Set of SQL statements that perform a business transaction can be encapsulated within a procedure and stored at the server.

4.Can be called by invocation as required.

5.This reduces the network traffic.

6.Helps in reducing the code duplication. Can be called by many applications.

Now create a simple table BOOK

With (bid int pk, title vc(10), price int)

Insert following tuples

<101, 'OPERATIONS', 300>; <107, 'DATABASES' , 370>;<128, 'NETWORKS', 175>

Note: For creating procedure the user must get permission from the admin.

Before creating a procedure, we must grant permissions to user

SQL> grant create procedure to <user>;

Example-1

// Procedure to perform updates to BOOK table

```
SQL> create or replace procedure proc1 as
  2  begin
  3  update Book set price=250 where price=300;
  4  dbms_output.put_line('  Update done  :  ');
  5  end;
  6  /
Procedure created.
```

//note that the numbers on the left of lines are line numbers generated by system just ignore.

This compiled block of code called ***proc1*** will be stored at DB server,

Note:

dbms_output.put_line(' Update done : ');

is a output statement like printf() in C.

First make server output on by executing the statement.

SQL> set serveroutput on;

To execute the procedure.

```
SQL> begin
  proc1;
   end;
 /
```

// modify the above code block to print string 'Proc1 executed'.

```
SQL> begin
  proc1;
   dbms_output.put_line(' Proc1 executed: ');
   end;
 /
```

// the above is calling a procedure in another block executed at SQL command prompt.

If executed, this will output:

Update done :

OR

SQL> exec proc1;

Will output same as previous approach.

Example:2

// Procedure with input parameter (bid) and prints the price of the book with bid sent as the parameter

```
SQL> create or replace procedure proc2(d in number) is
  n number;
  begin
  select price into n from BOOK where bid=d;
  dbms_output.put_line(' price  of  book with ID:'||d||'  is:  '||n);
  end;
  /
Procedure created.

SQL> exec proc2(107);
```

//Exercise:

1. Write a procedure sum_proc which takes three integers as input and prints the sum. Ex invocation is sum_proc(12,10,5);
   Output format: 'the sum of integers 12, 10, 5 is : 27'

Example-3:

// Procedure to illustrate the use of input and output parameters ;

```
SQL> create or replace procedure proc3(n in number, o out number) as
   begin
  select price into o from book where bid=n;
 end;
   /

Procedure created.
```

// Code block that calls the above procedure with bid, and stores the price of the book thrown by out param into a variable and print the same.

```
SQL> declare
pr number(5);
begin
proc3(101, pr);
 dbms_output.put_line('  price of 101 is  :  '|| pr);
 end;
/
salary of 101 is  :  300

PL/SQL procedure successfully completed.
```

Example:4

// Procedure to insert a new book record into Book table.

```
SQL> create or replace procedure proc4(id in number, name in varchar2, pr in nu
mber) as
 begin
insert into book values(id, name,pr);
dbms_output.put_line(' Book : '||id||  ':  name : '||name||' :  added to table');
 end;
/

Procedure created.

SQL> exec proc4(144,'Economics',570);
```