PL/SQL

Eng Ayah Alrifai ^^

1. All about PL/SQL.
2. How to call Function / Procedure using JDBC.

PL/SQL Block structure

```
DECLARE
    Declaration statments;
BEGIN
    Execution statments;
  EXCEPTION
      Exception handling statment
END;
```

Declare Variable in PL/SQL

```
Variable_name [CONSTANT] data_type [NOT NULL] [:= | DEFULT initval]
```

```
DECLARE
    Var1 integer := 5;
    Var2 Varchar DEFAULT 'Ayah';
    Var3 integer;
BEGIN
    Var3 := Var1 + 10;
    dbms_output.putline ('Val of V3 is' || Var3);
END;
```

⇒ Pi constant number := 3.141592;

↳ can not change the value

If statement PL/SQL

```
IF Condition
THEN
        --Block of statements
ELSE
        --Block of statements
END IF;
```

```
IF Condition
THEN
    --Block of statements
ELSIF
    --Block of statements
END IF;
```

```
DECLARE
    Var integer := 50;
BEGIN
    IF (Var = 10) THEN
        dbms_output.put_line('Value is 10');
    ELSIF (Var = 20) THEN
        dbms_output.put_line('Value is 20');
    ELSIF (Var > 20 and Var < 50) THEN
        dbms_output.put_line('Var between (20,50)');
    ELSE
        dbms_output.Put_line('Exact value of Var is ' ||Var);
    END IF;
END;
```

witch statement PL/SQL

```
CASE [expression]
    WHEN cond1 THEN Block_of_statement
    WHEN cond2 THEN Block_of_statement
    WHEN cond3 THEN Block_of_statement
    ELSE Block_of_statement
END CASE;
```

```
DECLARE
    $char char(1) := 'A';
BEGIN
    CASE $char
        WHEN 'A' THEN print('Ayah');
        WHEN 'B' THEN Print('Bana');
        WHEN 'S' THEN Print('Sham');
        ELSE print('No such name');
    END CASE;
END;
```

Exit Loop PL/SQL

```
Loop
    //Block of statements
    EXIT;
END Loop;
```

```
Loop
    //Block of statements
    EXIT WHEN Condition;
END Loop;
```

WHILE Loop PL/SQL

```
WHILE Condition
    Loop
        //Block_of_statement
    END Loop;
```

```
DECLARE
    num integer := 1
BEGIN
    WHILE num <= 10
        Loop
            Print(num); num := num+1;
        END Loop;
END;
```

FOR Loop PL/SQL

```
FOR loop_counter IN [REVERSE] start_value .. end_value
    Loop
        //Block_of_statements
    END Loop;
```

* No need to declare loop_counter.
* The counter variable is incremented by 1
* Can use EXIT, EXIT WHEN

Continue PL/SQL

```
WHILE condition
    Loop
        ||Block_of_statements
        CONTINUE;
        ||Block_of_statements
    END Loop;
```

Loop label PL/SQL

```
<< outer_label >>
FOR i IN 1..10
    Loop
        F<< inner_loop >>
        FOR j IN 1..10
        Loop
            print ('i = " || i || ' j = ' || j );
        END Loop inner_loop
    END Loop outer_loop
```

GoTo PL/SQL

```
GoTo label_name;

<< label_name >>
|| statements
```

* Can't transfer control into an IF stat, Loop stat or sub block
* Can't transfer control from one IF stat cluse to another or from CASE stat WHEN cluse to another
* Can't transfer control from an outerblock to sub block
* Can't transfer control out of a sub-program.
* Can't transfer control into an Exception handler.

orted Procedure PL/SQL

CREATE [OR REPLACE] PROCEDURE proc-name [list of parameters]

IS | AS

// Declaration Block

BEGIN

// Execution Block

EXCEPTION

// Exception Block

END;

(Param-name IN Param-type,
Param-name OUT Param-type,
Param-name IN OUT Param-type)

IN : can't update value (read only)

OUT : Can update value (write only)

IN out : read and write

* EXEC proc-name();

EXEC proc-nam(P1, P2, P3);

الـ Procedure طا بيعمل return لإمتي بعمل من خلال OUT برجع القيم يلي بدي إياها

Function PL/SQL

CREATE [OR REPLACE] FUNCTION fun-name [list-of-Parameters]
RETURN return-dataType
IS | AS

// Declaration Block

BEGIN

// Execution Block

RETURN return-variable

EXCEPTION

Same as Procedure

* result := fun-name();

* sqled dual. *, fun-name() FROM dual;

plicit Cursor PL/SQL

[1] CURSOR cursor_name IS select_statement;

[3] FETCH cursor_name INTO [list of Variables] ;

dependent on selection statemad

[2] OPEN cursor_name ;

[4] CLOSE cursor_name ;

```
DECLARE
    S_name   Student.Name%TYPE; ***
    S_age    Student.Age%TYPE;
    S_grade  Student.Grade%TYPE;
    CURSOR student_cursor IS (SELECT Name, Age, Grade FROM Student);
BEGIN
    OPEN student_cursor
    LOOP
        FETHC student_cursor INTO S_name, S_age, S_grade; **
        Print('Name:' || S_name || ' Age:' || S_age || ' Grade:' || S_grade);
    END_Loop
    CLOSE student_cursor;
END;
```

TableName.ColName%TYPE [***]

Table ال column بهاي ال DataType نفس ال Variable ال بهاي الطريقة بعطي

CURSOR DECLIR بال Select بال يلي الترتيب نفس الترتيب يكون لازم [**]

Exception Handling PL/SQL

```
.....
EXCEPTION
    WHEN ex_name1 THEN
        // Error Handling statements
    WHEN ex_name2 THEN
        // Error Handling statements
```

RAISE Exception PL/SQL

```
.....
BEGIN
    // Block of statements
    RAISE defined_exception_name
    // Block of statments
EXCEPTION
    WHEN define_exception_Name THEN
        // Error Handling statement
END;
```

User defined Exception

```
DECLARE
    .....
    ex_invalid_rollNo EXCEPTION;
BEGIN
    // Block of statements
    RAISE ex_invalid_rollNo;
    // Block of statements
EXCEPTION
    WHEN ex_invalid_rollNo THEN
        // Error handling statments
END;
```

Triggers PL/SQL

```
CREATE [OR REPLACE] TRIGGER trigger_name
  { BEFORE | AFTER | INSTED OF }
  { INSERT [OR] | UPDATE [OR] | DELETE }
  [OF col_name]
  ON table_name
  [REFERENCING OLD As o NEW As n]
  [FOR EACH ROW]
  WHEN (condition)
  BEGIN
    // SQL Statements
  END;
```

بستخدمها لل View

بختار واحده أو أكثر و بيبنهن OR

بكتب هاد الـ Section في حال بدي الـ trigger يشتغل على column معين

بكتب هاد الجزء في حال كنت محتاجه القيمة القديمة و القيمة الجديدة

بكتب هاد الجزء في حال كنت محتاجه أعمل run لل trigger عكل row [تعديل | إنشاء | إنحذف]

Oracle PL/SQL Package

A Package is a schema object that groups logically related PL/SQL types, Variable and subprograms.

الـ Pkg كأنه بعمل group بحيث ممكن نفس إسم الـ function أستخدمه كل مره بختار الـ Pkg

...ll Procedure using JDBC

```
CallStatement call = connection.prepareCall("CALL Pkg_name.Proc_name(?,?)");
Call.setString(1,"1000"); // set input value
Call.registerOutPutParameter(2, TYPE.VARCHAR);
Call.executeUpdate();
String name = call.getString(2);
Call.close;
```

(?, ?)

[1]	[2]
IN	OUT
VARCHAR	VARCHAR

registerOutPutParameter → For output

getString(2) → get output

Call Function Using JDBC

Same as Procedure but Prepare call argument is

{ ? = CALL Pkg_name.Proc_name(?,?) }
[1] [2] [3]

[1] → return data (output)

So use RegistrationOutPutParameter