Programación en Base de Datos

PROGRAMACIÓN PL/SQL I

Semana 6:

OBJETIVO DE LA EXPERIENCIA

Construye programas estructurados utilizando el lenguaje PL/SQL dentro del manejador de base de datos Oracle.

MARCO TEÓRICO

El lenguaje utilizado para acceder a las bases de datos relacionales es el llamado Lenguaje Estructurado de Consulta (SQL, Structured Query Language), que es muy flexible y transparente, es decir, sencillo y poderoso. SQL es un lenguaje de cuarta generación, lo cual quiere decir que describe lo que quiere hacerse. Ayudará de manera factible al estudiante a relacionar el mundo real con las base de datos a desarrollar usando programación orientado a negocio.

PL / SQL es un lenguaje de procedimiento diseñado específicamente para incluir declaraciones SQL dentro de su sintaxis. Las unidades de programa PL / SQL son compiladas por el servidor de Oracle Database y almacenadas dentro de la base de datos. Y en tiempo de ejecución, tanto PL / SQL como SQL se ejecutan dentro del mismo proceso del servidor, brindando una eficiencia óptima. PL / SQL hereda automáticamente la solidez, seguridad y portabilidad de la base de datos Oracle.

RECURSOS

Hardware

* Sistema. Oracle en Windows requiere un PC Intel x86, AMD64 o Intel EM64T
* Memoria. Al menos 1 GB de RAM y el doble en virtual.
* Espacio en disco duro. Al menos 6 GB para la instalación (algunas instalaciones requieren menos). Además, necesitamos poder almacenar 500 MB en la carpeta TEMP del sistema.
* Tarjeta gráfica. Debe de ser capaz de mostrar 1024 por 768 píxeles como mínimo y 256 colores.

Software

**Sistema Operativo**.

* + Windows 2003 Server y 2003 Server R2.
  + Windows XP Professional.
  + Windows Vista, pero no la versión Home Edition.
  + Windows Server 2008 y 2012. No la versión Server Core.

Compiladores. Se usan para la gente que crean aplicaciones en Oracle usando lenguajes como Pro C, Pro COBOL, JAVA.

* + Visual C++.NET 2005 8.0 o Intel 10.1 C, .Net Express.

Navegador. Para configurar algunos servicios de Oracle. Debe de ser navegador moderno (Internet Explorer 6 o superior, Firefox 2.0 o superior, Safari 3.1 o superior, Chrome 3.0 o superior)

Usar dirección IP única en la máquina en la que se instala Oracle. Es decir, no usar DHCP para direccionar la IP en el servidor de Oracle. No es un requisito obligatorio, pero es muy recomendable.

Implementa programas estructurados usando bloques y excepciones, utilizando la programación PL/SQL.

INTRODUCCIÓN A ORACLE PL/SQL

FAVOR DE DESCARGAR DICHO ARCHIVO Semana06\_PL\_SQL\_1.sql,

PL / SQL es un lenguaje de programación de base de datos potente pero directo. Es fácil de escribir y leer, y viene con muchas optimizaciones y funciones de seguridad listas para usar.

**ESTRUCTURAS DE BLOQUES PL/SQL**

Todo programa implementado en PL/SQL tiene la siguiente estructura:

**SET SERVEROUTPUT ON;**

**DECLARE**

**<Declaración de variable>**

**BEGIN**

[**Cuerpo del programa**

**. . . . . . . . . . . . . . . . . .**

**. . . . . . . . . . . . . . . . . .**

**. . . . . . . . . . . . ..** ]

**END;**

**DEFINICIONES:**

**SET SERVEROUTPUT ON / OFF:**

La configuración **SERVEROUTPUT** controla si SQL \* Plus imprime la salida generada por el paquete DBMS\_OUTPUT de los procedimientos PL / SQL.

|  |  |
| --- | --- |
| **SECCION** | **DESCRIPCION** |
| **DECLARE** | Esta es el área donde se definen las variables de Oracle |
| **BEGIN** | Este representa el inicio del programa o bloque. |
| **END** | Este representa el fin del programa o bloque. |

**TIPOS DE BLOQUE.**



**TIPOS DE VARIABLES.**

|  |  |
| --- | --- |
| **Tipo de dato** | **Descripción** |
| BINARY\_INTEGER | Tipo base para los enteros entre -2147483647 y +2147483647 |
| NUMBER [(precisión, escala)] | Tipo base para los números de punto fijo y flotante. |
| CHAR [(longitud máxima)] | Tipo base para los datos de caracteres de longitud Fija, hasta 2000 bytes. Si no especifica una longitud Máxima, la longitud será 1 por defecto |
| LONG | Tipo base para los datos de caracteres de longitud Variable, hasta 2 Gigabytes |
| LONG RAW | Tipo base para los datos binarios, hasta 2 Gigabytes. |
| VARCHAR2(longitud máxima) | Tipo base para los datos de caracteres de longitud Variable, hasta 4000 bytes. |
| DATE | Tipo base para fechas y horas |
| BOOLEAN | Tipo base que almacena uno de los tres posibles valores que se obtienen de un cálculo lógico: TRUE, FALSE y NULL |
| RAW (size) | Tipo de dato binario RAW, hasta 2000 bytes |
| LONG RAW | Lo mismo que lo anterior pero hasta 2 Gigabytes. |
| CLOB | Tipo que almacena simples caracteres, hasta 4 Gigabytes. Significa ‘Character Large Object” |
| BFILE | Contiene un puntero a un archivo externo a la base de datos. Hasta 4 Gigabytes. |

Ejemplo de declaración de variables:

**BLOQUES ANONIMOS.**

**DECLARE**

AREA NUMBER:=0;

BASE NUMBER;

ALTURA NUMBER;

Nombre Varchar2 (100);

FechaNac Date;

Observaciones:

En la declaración de variables se usan inicializaciones es decir, el caso de la variable AREA, esta variable usa un indicador **:=** significa asignación, toda variable con respecto a otra declarada debe estar separada por un punto y coma ”**;**”.

**Ejemplo 01:**

Un ejemplo más completo acerca de la declaración y propósito de las variables:

**SET SERVEROUTPUT ON;**

**DECLARE**

BASE NUMBER:=10;

ALTURA NUMBER:=5;

AREA NUMBER:=0;

**BEGIN**

AREA :=( BASE \* ALTURA) / 2;

DBMS\_OUTPUT.PUT\_LINE ('EL AREA ES: ' || AREA);

**END;**

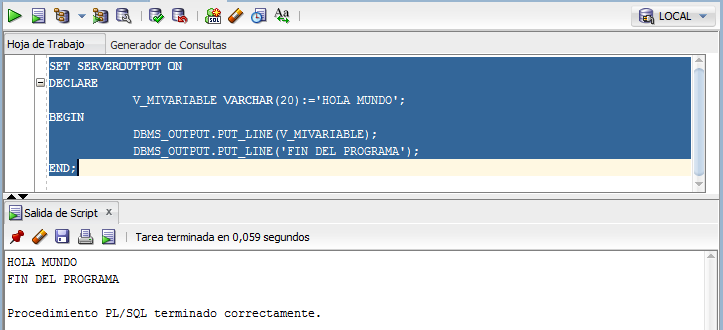
Este código debe ser ejecutado en un solo bloque de selección.

**Comando de salida:**

**DBMS\_OUTPUT.PUT\_LINE**

Con el podemos realizar las impresiones por la ventana de consola de Oracle.

**PRUEBA DESDE ORACLE PL/SQL:**



Este ejemplo muestra dos mensajes: “HOLA MUNDO” y “FIN DEL PROGRAMA”.

**Ejemplo 2**:

SET SERVEROUTPUT ON;

DECLARE

V\_NUM1 NUMBER (4, 2):= 10.2;

V\_NUM2 NUMBER (4, 2):= 20.1;

BEGIN

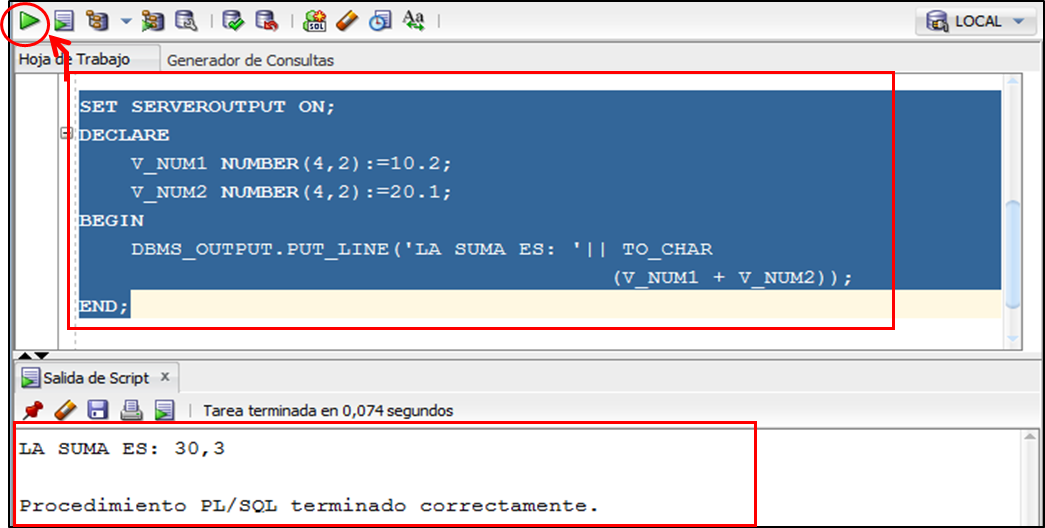
DBMS\_OUTPUT.PUT\_LINE ('LA SUMA ES: '|| TO\_CHAR

(V\_NUM1 + V\_NUM2)) ;

END;

Este ejemplo muestra la suma de los valores 10.2 + 20.1 cuyo resultado es: LA SUMA ES: 30,3

**PRUEBA DESDE ORACLE PL/SQL:**

****

**SENTENCIA CONTROL IF/THEN/ELSE**

En Oracle, la instrucción IF-THEN-ELSE se usa para ejecutar código cuando una condición es VERDADERA, o ejecutar código diferente si la condición se evalúa como FALSA.

**Sintaxis:**

**IF** (Operación Lógica) **THEN**

<Respuesta por bloque verdadero>

**[ ELSEIF THEN** <- Respuesta por bloque verdadero>

<Respuesta por bloque Verdadero> **]**

**ELSE**

<Respuesta por bloque Falso>

**END IF;**

**Ejemplo 01**:

**SET SERVEROUTPUT ON;**

**DECLARE**

**Edad Int := 18;**

**BEGIN**

**IF (Edad>=18) THEN**

**DBMS\_OUTPUT.PUT\_LINE ('UD. Es Mayor de Edad');**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE ('UD. Es Menor de Edad.');**

**END IF;**

**END;**

Este ejemplo muestra dos mensajes: “**Ud. Es Mayor de Edad**” cuando de edad es mayor o igual 18, en caso contrario “**Ud. Es Menor de Edad**”.

**Ejemplo 02:**

**SET SERVEROUTPUT ON;**

**DECLARE**

**A Int := 10; B Int := 5; C Int := 17;**

**Mayor Int;**

**BEGIN**

**IF (A>=B AND A>=C) THEN Mayor:= A; END IF;**

**IF (B>=A AND B>=C) THEN Mayor:= B; END IF;**

**IF (C>=A AND C>=B) THEN Mayor:= C; END IF;**

**DBMS\_OUTPUT.PUT\_LINE ('EL MAYOR ES: ' || Mayor) ;**

**END;**

Este ejemplo muestra el mayor de 3 números, es decir el programa decidirá entre los valores **10, 5 y 17** quién de ellos es el valor, la respuesta desde luego es **17**.

**Consideraciones que debe tomar en cuenta.**

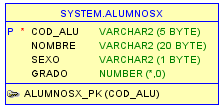
* La asignación de un valor sobre una variable se realiza mediante el operador**:=**, ejemplo: **v\_mivariable:=20**;
* Toda instrucción de PLSQL finaliza con punto y coma (**;**).
* Si un query devuelve una sola fila y con un campo puede ser asignado a una variable, ejemplo:

**SELECT COUNT (\*) INTO MI\_VARIABLE FROM EMPLEADO;**

* Si un query devuelve una sola fila y con 2 o más campos también puede ser asignado a varias variables, ejemplo:

**SELECT Nombre, Edad INTO MI\_VAR1, MI\_VAR2**

**FROM EMPLEADO;**

**Ejemplo 03:**

Se tiene la siguiente tabla de datos:

**CREATE TABLE Alumnos (**

**Cod\_alu varchar2 (5) NOT NULL,**

**Nombre varchar2 (20) NULL,**

**Sexo varchar2 (1) NULL,**

**Grado int NULL,**

**PRIMARY KEY (Cod\_alu)**

**);**

**INSERT INTO AlumnosX VALUES ('A0001','PEDRO','M', 1);**

**INSERT INTO AlumnosX VALUES ('A0002','ANA','F', 2);**

**INSERT INTO AlumnosX VALUES ('A0003','JUAN','M', 3);**

**INSERT INTO AlumnosX VALUES ('A0004','MARIA','F', 2);**

**INSERT INTO AlumnosX VALUES ('A0005','NOE','M', 2);**

**COMMIT;**

Se pide un programa PL/SQL el cual permita contar, cuantos varones y damas existen en dicha tabla.

**SET SERVEROUTPUT ON;**

**DECLARE**

**Cm Int := 0; Cf Int := 0;**

**BEGIN**

**SELECT COUNT (SEXO) INTO Cm FROM ALUMNOSX WHERE SEXO='M';**

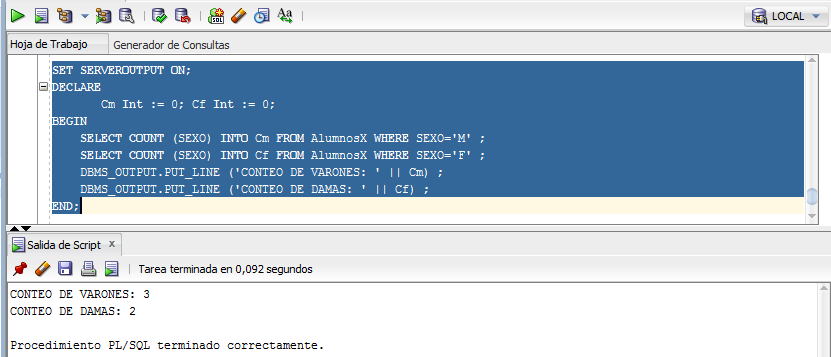
**SELECT COUNT (SEXO) INTO Cf FROM ALUMNOSX WHERE SEXO='F';**

**DBMS\_OUTPUT.PUT\_LINE ('CONTEO DE VARONES: ' || Cm) ;**

**DBMS\_OUTPUT.PUT\_LINE ('CONTEO DE DAMAS: ' || Cf) ;**

**END;**

**PRUEBA DESDE ORACLE PL/SQL:**



**Sentencia CASE:**

Selección de casos, omite usar varias sentencia **IF**.

Sintaxis:

**Ejemplo:**

**CASE**

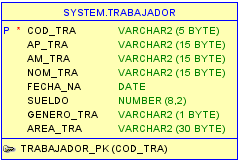
**WHEN <CONDICION1> THEN**

**WHEN <CONDICION2> THEN**

**ELSE**

**<Caso contrario a todos>**

**END CASE;**

Se tiene la siguiente tabla de datos:

**CREATE TABLE TRABAJADOR (**

**COD\_TRA VARCHAR2 (5) NOT NULL,**

**AP\_TRA VARCHAR2 (15) NULL,**

**AM\_TRA VARCHAR2 (15) NULL,**

**NOM\_TRA VARCHAR2 (15) NULL,**

**FECHA\_NA DATE NULL,**

**SUELDO NUMERIC (8,2) NULL,**

**GENERO\_TRA VARCHAR2 (1) NULL,**

**AREA\_TRA VARCHAR2 (30) NULL,**

**PRIMARY KEY (COD\_TRA));**

Se pide un código el cual pueda determinar si la planilla puede o no ser pagada, según la condiciones:

**SET SERVEROUTPUT ON;**

**DECLARE**

**Total Numeric (10, 2):= 0;**

**BEGIN**

**SELECT SUM (SUELDO) INTO Total FROM TRABAJADOR;**

**CASE**

**WHEN Total<10000 THEN**

**DBMS\_OUTPUT.PUT\_LINE ('LA PLANILLA PUEDE SER PAGADA');**

**WHEN (Total>=10001) AND (Total<=15000) THEN**

**DBMS\_OUTPUT.PUT\_LINE ('LA PLANILLA AUN PUEDE SER PAGADA') ;**

**ELSE**

**DBMS\_OUTPUT.PUT\_LINE ('LA PLANILLA NO PUEDE SER PAGADA') ;**

**END CASE;**

**END;**

**BUCLES: FOR/WHILE/LOOP**

**Sentencia: LOOP**

Permite generar bucles, el bucle se genera cuando la condición es falsa.

Sintaxis:

**LOOP**

----

<Bloque de instrucciones>

----

**EXIT WHEN** <Condicion\_Salida>

**END LOOP**

**Ejemplo 01:**

Imprimiendo los primeros 10 números:

SET SERVEROUTPUT ON;

DECLARE

V\_NUM NUMBER := 0;

BEGIN

**LOOP**

V\_NUM:=V\_NUM+1;

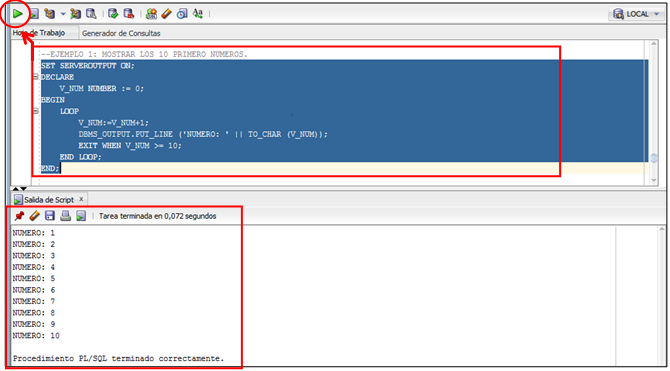
DBMS\_OUTPUT.PUT\_LINE ('NUMERO: ' || TO\_CHAR (V\_NUM)) ;

**EXIT WHEN V\_NUM >= 10;**

**END LOOP;**

END;

**PRUEBA DESDE ORACLE.**



**Ejemplo 02:**

Muestra la tabla de multiplicar del 5

SET SERVEROUTPUT ON;

DECLARE

TABLA NUMBER := 5;

I NUMBER := 0;

BEGIN

**LOOP**

I := I + 1;

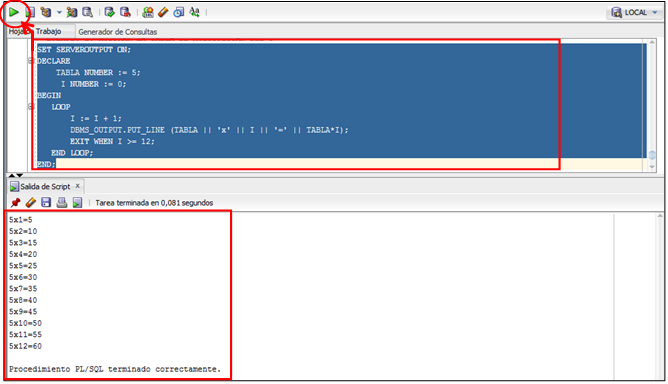
DBMS\_OUTPUT.PUT\_LINE (TABLA || 'x' || I || '=' || TABLA\*I);

**EXIT WHEN I >= 12;**

**END LOOP;**

END;

**PRUEBA DESDE ORACLE:**



**SENTENCIA: WHILE El bucle se genera cuando la condición es verdadera.**

Sintaxis:

**WHILE** <Condición Loop>

**<Bloque de instrucciones a repetir>**

**END LOOP**;

**Ejercicio 01:**

Programa que muestra los 10 primero números.

SET SERVEROUTPUT ON;

**DECLARE**

V\_NUM NUMBER := 1;

**BEGIN**

**WHILE** V\_NUM<11 **LOOP**

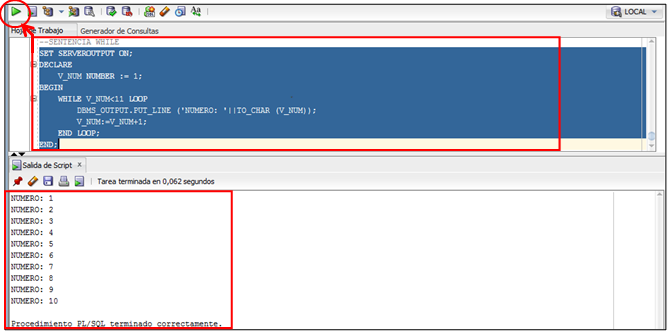
DBMS\_OUTPUT.PUT\_LINE ('NUMERO: '||TO\_CHAR (V\_NUM));

V\_NUM:=V\_NUM+1;

**END LOOP;**

**END;**

**PRUEBA DESDE ORACLE:**



**Ejercicio 02:**

Programa que convierte un valor Decimal **100** en Binario: **1100100**

SET SERVEROUTPUT ON;

**DECLARE**

N Int := 100; Q Int; R Int;

Bin Varchar2 (30):='';

**BEGIN**

Q := N;

**WHILE** Q>0 **LOOP**

N := Q; Q := TRUNC (Q/2); R := N-(Q \* 2);

IF R=0 THEN Bin:='0' || Bin; END IF;

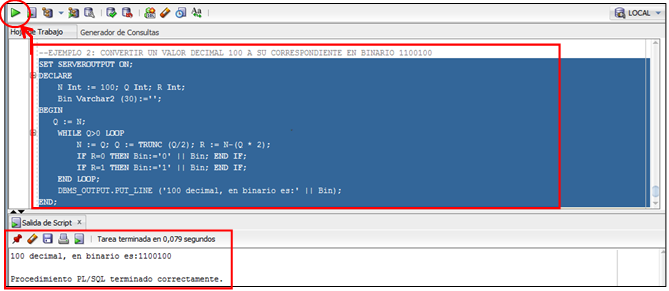
IF R=1 THEN Bin:='1' || Bin; END IF;

**END LOOP;**

DBMS\_OUTPUT.PUT\_LINE ('100 decimal, en binario es:' || Bin);

**END**;

**PRUEBA DESDE ORACLE:**



**Ejercicio 03:**

Programa que convierte un valor Decimal 100 en Octal: 144

SET SERVEROUTPUT ON;

**DECLARE**

N Int:=100; Q Int; R Int;

Oct Varchar2 (30):='';

**BEGIN**

Q:=N;

**WHILE** Q>0 **LOOP**

N:=Q; Q:=TRUNC(Q/8); R:=N-(Q \* 8);

**CASE**

**WHEN** R=0 THEN Oct:='0' || Oct;

**WHEN** R=1 THEN Oct:='1' || Oct;

**WHEN** R=2 THEN Oct:='2' || Oct;

**WHEN** R=3 THEN Oct:='3' || Oct;

**WHEN** R=4 THEN Oct:='4' || Oct;

**WHEN** R=5 THEN Oct:='5' || Oct;

**WHEN** R=6 THEN Oct:='6' || Oct;

**WHEN** R=7 THEN Oct:='7' || Oct;

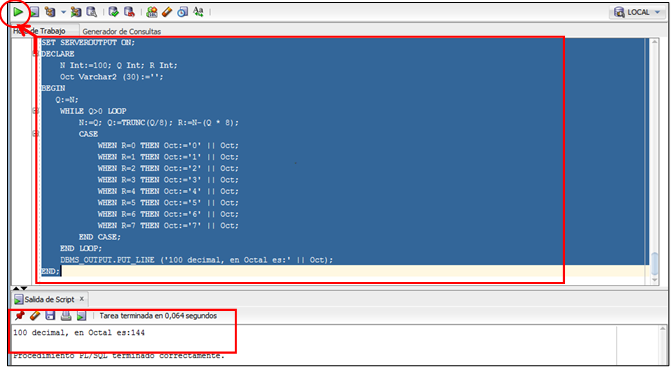
**END CASE;**

**END LOOP;**

DBMS\_OUTPUT.PUT\_LINE ('100 decimal, en Octal es:' || Oct);

**END;**

**PRUEBA DESDE ORACLE:**



**Ejercicio 04:**

Programa que convierte un valor Decimal 100 en Hexadecimal: 64

SET SERVEROUTPUT ON;

**DECLARE**

N Int:=100; Q Int; R Int; Hex Varchar2(30):='';

**BEGIN**

Q:=N;

**WHILE** Q>0 **LOOP**

N:=Q; Q:=TRUNC(Q/16);R:=N-(Q \* 16);

**CASE**

**WHEN** R=0 THEN Hex:='0' || Hex;

**WHEN** R=1 THEN Hex:='1' || Hex;

**WHEN** R=2 THEN Hex:='2' || Hex;

**WHEN** R=3 THEN Hex:='3' || Hex;

**WHEN** R=4 THEN Hex:='4' || Hex;

**WHEN** R=5 THEN Hex:='5' || Hex;

**WHEN** R=6 THEN Hex:='6' || Hex;

**WHEN** R=7 THEN Hex:='7' || Hex;

**WHEN** R=8 THEN Hex:='8' || Hex;

**WHEN** R=9 THEN Hex:='9' || Hex;

**WHEN** R=10 THEN Hex:='A' || Hex;

**WHEN** R=11 THEN Hex:='B' || Hex;

**WHEN** R=12 THEN Hex:='C' || Hex;

**WHEN** R=13 THEN Hex:='D' || Hex;

**WHEN** R=14 THEN Hex:='E' || Hex;

**WHEN** R=15 THEN Hex:='F' || Hex;

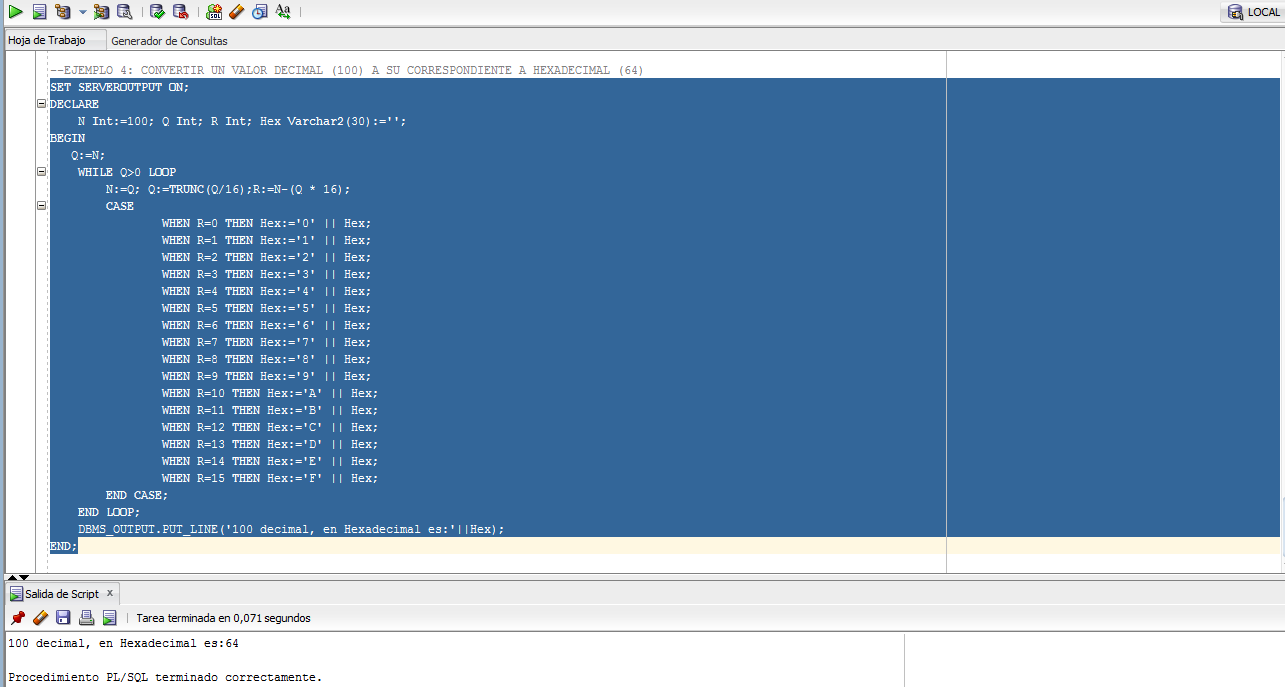
**END CASE;**

**END LOOP;**

DBMS\_OUTPUT.PUT\_LINE('100 decimal, en Hexadecimal es:'||Hex);

**END;**

**PRUEBA DESDE ORACLE:**



0

**SENTENCIA: FOR Genera un ciclo repetitivo desde una valor inicial hasta un valor final.**

**Sintaxis**:

**FOR variable IN inicio ... final LOOP**

**<Bloque instrucciones del Bucle>**

**END LOOP;**

**Ejemplo 01:**

Programa que genera los 10 primero números.

SET SERVEROUTPUT ON;

**BEGIN**

**FOR** V\_NUM **IN** **1**..**10** **LOOP**

DBMS\_OUTPUT.PUT\_LINE ('NUMERO: '||TO\_CHAR (V\_NUM));

**END LOOP;**

**END;**

Nota: La variable utilizada como contador en el FOR no requiere ser especificada en el **DECLARE**.

**PRUEBA DESDE ORACLE.**



**Ejemplo 02:**

Programa que calcula el valor factorial de 5

SET SERVEROUTPUT ON

**DECLARE**

FACT NUMERIC :=1;

I Int; N Int:=5;

**BEGIN**

**FOR** I IN 1..N LOOP

FACT:=FACT\*I;

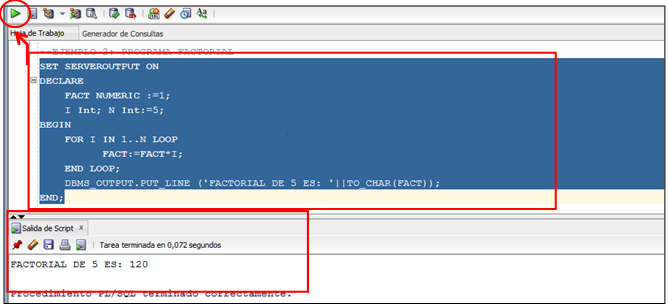
**END LOOP;**

DBMS\_OUTPUT.PUT\_LINE ('FACTORIAL DE 5 ES: '||TO\_CHAR(FACT));

**END;**

**Nota**: La variable utilizada como contador en el FOR es definida por el usuario, especificada en el DECLARE.

**PRUEBA DESDE ORACLE:**



**Usando variables de sustitución &:**

**Ejemplo:**

SET SERVEROUTPUT ON

**DECLARE**

AREA NUMBER:=0;

BASE NUMBER;

ALTURA NUMBER;

**BEGIN**

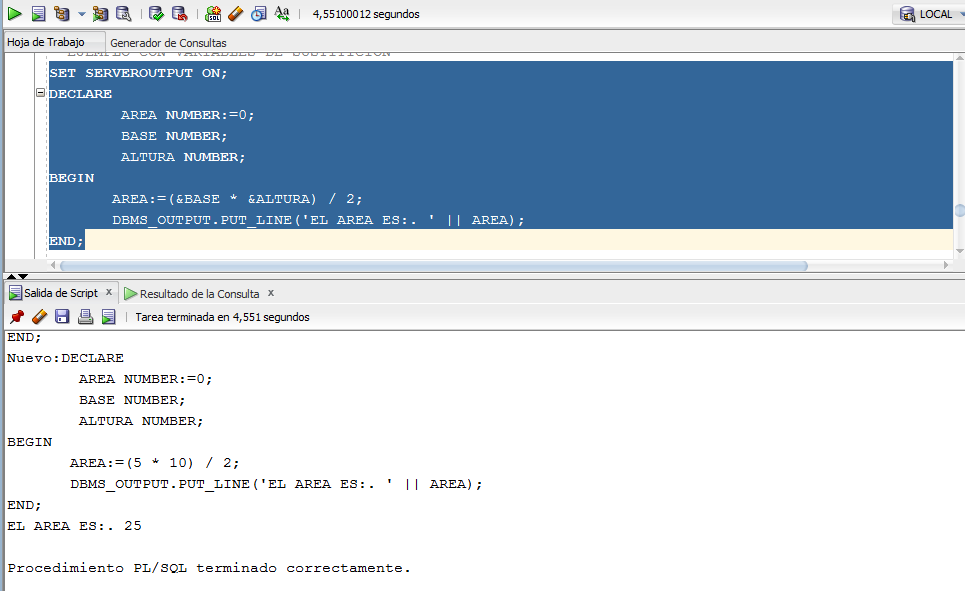
AREA:=(**&**BASE \* **&**ALTURA) / 2;

DBMS\_OUTPUT.PUT\_LINE ('EL AREA ES:. ' || AREA);

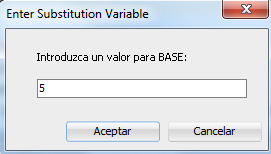
**END;**

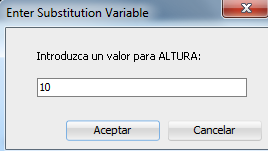
La ejecución de este código Oracle pedirá dos valores, solo aquellas variables que usen &, es decir Base y Altura, cuando estos valores hayan sido ingresados desde teclado, Oracle procederá a realizar el cálculo del área de triangulo.

**PRUEBA DESDE ORACLE:**



0





CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA

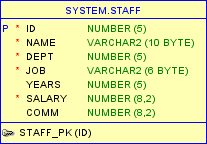
El objetivo de este capítulo es comprender la lógica de Programación en Base de Datos, de esta manera el estudiante se compenetra más con entorno PL/SQL de Oracle, al analizar la información, resultado de las programaciones realizadas de manera exitosa. Mediante el uso de las herramientas **PL/SQL**.

ACTIVIDAD VIRTUAL

Ingresa a la plataforma virtual, luego revisa y analiza el tema desarrollado para realizar las actividades propuestas:

De acuerdo al material presentado Semana 06, Responda a las siguientes Preguntas:

**SE TIENE LA SIGUIENTE TABLA: STAFF CON LA SIGUIENTE ESTRUCTURA.**



**CREATE TABLE** STAFF (

ID Numeric (5, 0) **PRIMARY KEY** NOT NULL,

NAME Varchar2 (10) NOT NULL,

DEPT Numeric (5, 0) NOT NULL,

JOB Varchar2 (6) NOT NULL,

YEARS Numeric (5, 0) NULL,

SALARY Numeric (8, 2) NOT NULL,

COMM Numeric (8, 2) NULL

);

IMPORTANTE

**LA ESTRUCTURA Y DATOS DE LA TABLA STAFF, SE ENCUENTRAN**

**EN EL ARCHIVO SCRIPT DE BASE DE DATOS.**

**S06\_PL\_SQL\_1.sql,**

**FAVOR DE DESCARGAR DICHO ARCHIVO**

**.**

**RESOLVER:**

Crear los siguientes programas PL/SQL.

1. Leer dos valores, y mostrar de ellos las cuatro operaciones básicas.
2. Leer un número entero, y mostrar de él, su tabla de multiplicar.
3. De la tabla STAFF se pide el conteo de todas las comisiones, tanto nulas como no nulas, visualizar el conteo de ambos casos.
4. Se desea pagar una planilla de sueldos, crear un programa que obtenga la suma de los salarios según el tipo de trabajo (**JOB**) usar aquí variable de sustitución, los tipos de JOB son: **MGR**, **SALES** y **CLERK**, si la suma de los salarios supera los 160000, entonces **LA PLANILLA NO ES PAGABLE**, en caso contrario **LA PLANILLA ES PAGABLE**.
5. Por razones de reducción de personal, se pide saber la contabilización de las personas por años de servicios, según la siguiente tabla.

|  |  |
| --- | --- |
| **CRITERIO** | **RESULTADO** |
| Entre 1 y 4 años | CONTINUAR LABORANDO |
| Entre 5 y 6 años | PENDIENTE DE JUBILACION |
| Entre 7 a más años | DESPEDIDOS. |

**Ejemplo de Salida de datos:**

Entre 1 y 4 años: 3 Trabajadores, CONTINUAR LABORANDO

Entre 5 y 6 años: 5 Trabajadores, PENDIENTE DE JUBILACION

Entre 7 a más años 9 Trabajadores, DESPEDIDOS.