Programación en Base de Datos

PROGRAMACIÓN PL/SQL I

Semana 7

OBJETIVO DEL LABORATORIO

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 bases 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.

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.

MANEJO DE EXCEPCIONES EN PL/SQL.

FAVOR DE DESCARGAR EL ARCHIVO Semana07\_PL\_SQL\_1.sql,

DEFINICION:

Las excepciones se generan cuando suceden errores. Esta es una sección independiente del bloque PL/SQL.

UTILIDAD.

Permitir la depuración (gestión) del error de manera independiente, haciendo la lógica del programa más legible.

**DECLARACION DE EXCEPCIONES:**

PREDEFINIDOS

Se generan automáticamente, por el propio Oracle.

DEFINIDOS POR EL USUARIO

Son generadas mediante la orden RAISE.

Excepciones de Oracle - Predefinidos

|  |  |  |
| --- | --- | --- |
| Nombre de la Excepción | Código de Error | Descripción |
| NO\_DATA\_FOUND | ORA-01403 | La sentencia SELECT no devolvió datos. |
| TOO\_MANY\_ROWS | ORA-01422 | La sentencia SELECT devolvió más de una fila de datos. |
| INVALID\_CURSOR | ORA-01001 | Se produjo una operación de cursor ilegal. |
| ZERO\_DIVIDE | ORA-01476 | Se intentó dividir por cero. |
| DUP\_VAL\_ON\_INDEX | ORA-00001 | Se intentó insertar un valor duplicado. |
| INVALID\_NUMBER | ORA-01722 | Falla la conversión de una cadena de caracteres a numeros. |

**EJEMPLO DE EXCEPCION PREDEFINIDA**

SET SERVEROUTPUT ON;

**DECLARE**

v\_result NUMBER;

**BEGIN**

SELECT 10/2 INTO v\_result FROM dual;

DBMS\_OUTPUT.put\_line('El resultado es: '|| v\_result);

**EXCEPTION**

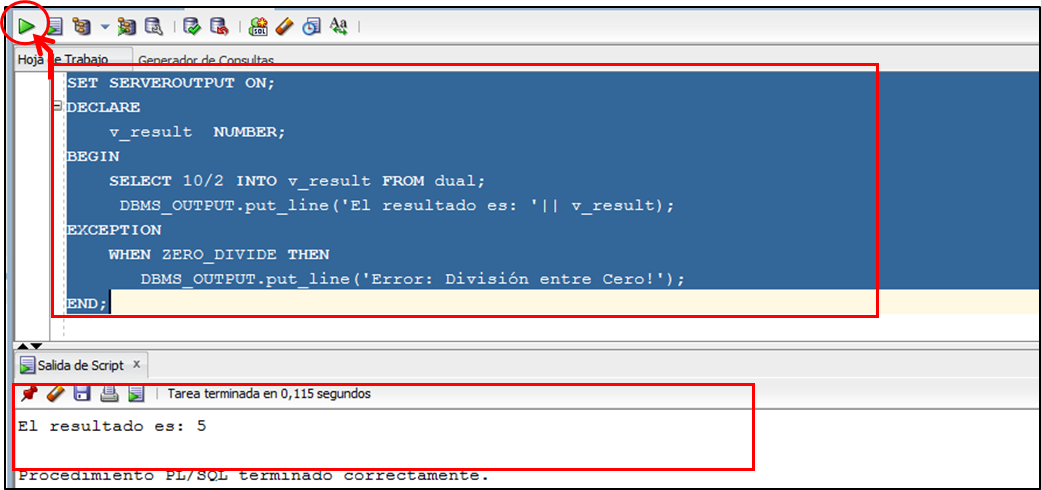
WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.put\_line('Error: División entre Cero!');

**END**;

En este ejemplo no se tendrá la salida del erro de excepción, dado que no se está dividiendo por cero, en el siguiente empleo veremos cómo reacciona la excepción cuando realiza una división por cero.

**PRUEBA DESDE ORACLE:**



**Probando la excepción dividiendo por cero.**

SET SERVEROUTPUT ON;

**DECLARE**

v\_result NUMBER;

**BEGIN**

SELECT 10/0 INTO v\_result FROM dual;

DBMS\_OUTPUT.put\_line('El resultado es: '|| v\_result);

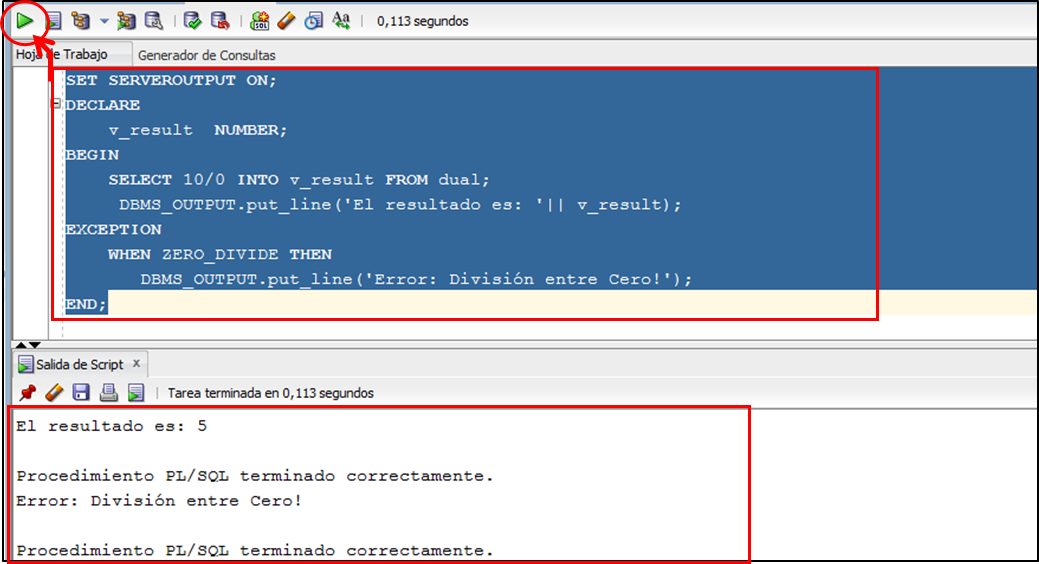
**EXCEPTION**

**WHEN** **ZERO\_DIVIDE** THEN

DBMS\_OUTPUT.put\_line('Error: División entre Cero!');

**END**;

PRUEBA DESDE ORACLE:



En un bloque PL/SQL, realice la búsqueda simple por código de empleado. En caso no existiese, usando excepciones envíe un mensaje ‘Código de empleado no existe’ mediante el gestor **NO\_DATA\_FOUND**.

SET SERVEROUTPUT ON;

**DECLARE**

CODIGO STAFF.ID%TYPE;

NOMBRE STAFF.NAME%TYPE;

**BEGIN**

CODIGO:=111;

SELECT ID ,NAME INTO CODIGO,NOMBRE FROM STAFF

WHERE ID=CODIGO;

DBMS\_OUTPUT.put\_line('Codigo: '|| CODIGO);

DBMS\_OUTPUT.put\_line('Nombre: '|| NOMBRE);

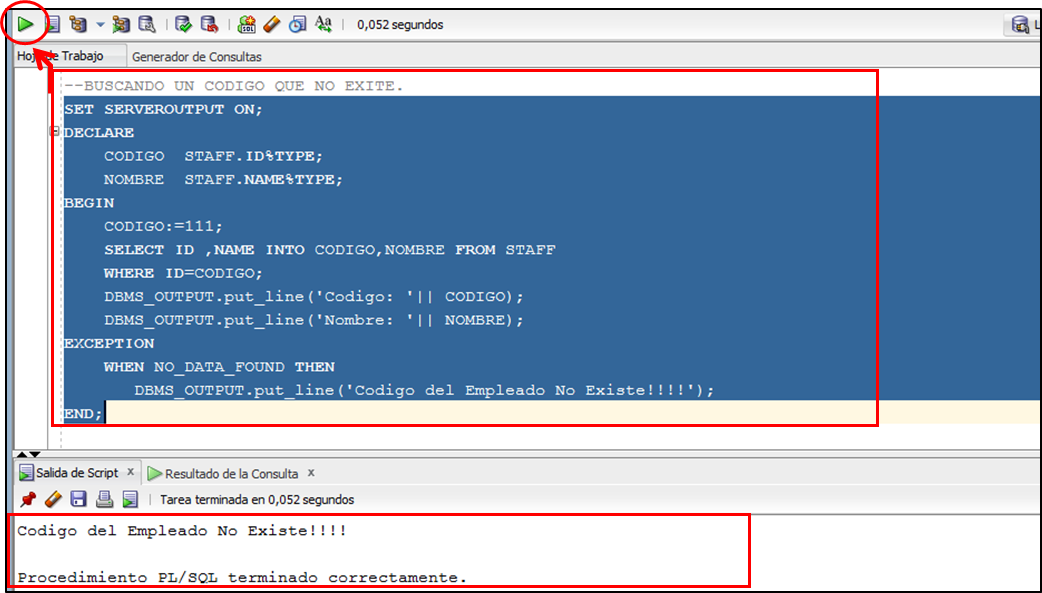
**EXCEPTION**

**WHEN** **NO\_DATA\_FOUND** THEN

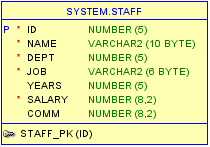
DBMS\_OUTPUT.put\_line('Codigo del Empleado No Existe!!!!');

**END;**

**PRUEBA DESDE ORACLE:**



**OBSERVACION:**

Abra notado que el código se usan un elemento **%TYPE**, este atribute permite a la variable heredar el tipo de dato de un determinado campo de tabla.

**DECLARE**

**CODIGO STAFF.ID%TYPE;**

**NOMBRE STAFF.NAME%TYPE;**

En este ejemplo, la variable CODIGO hereda el tipo de dato que tiene el campo ID de la tabla STAFF. Lo mismo pasa con la variable NOMBRE hereda el tipo de dato del campo NAME. En conclusión la variable CODIGO será del tipo NUMBER (5) y la variable NOMBRE será del tipo VARCHAR2 (10).

En un bloque PL/SQL, realice la búsqueda simple con la inicial del nombre del empleado. Usando excepciones predefinidas envíe los siguientes mensajes ‘Ningún registro encontrado’ o ‘Más de una Fila encontrada’. Utilice los gestores NO\_DATA\_FOUND y TOO\_MANY\_ROWS.

SET SERVEROUTPUT ON;

**DECLARE**

CODIGO STAFF.ID**%TYPE**;

NOMBRE STAFF.NAME**%TYPE**;

**BEGIN**

SELECT ID, NAME INTO CODIGO, NOMBRE FROM STAFF

WHERE NAME LIKE **&NOMBRE** ||'%';

DBMS\_OUTPUT.put\_line('Codigo: '|| CODIGO);

DBMS\_OUTPUT.put\_line('Nombre: '|| NOMBRE);

**EXCEPTION**

**WHEN NO\_DATA\_FOUND** THEN

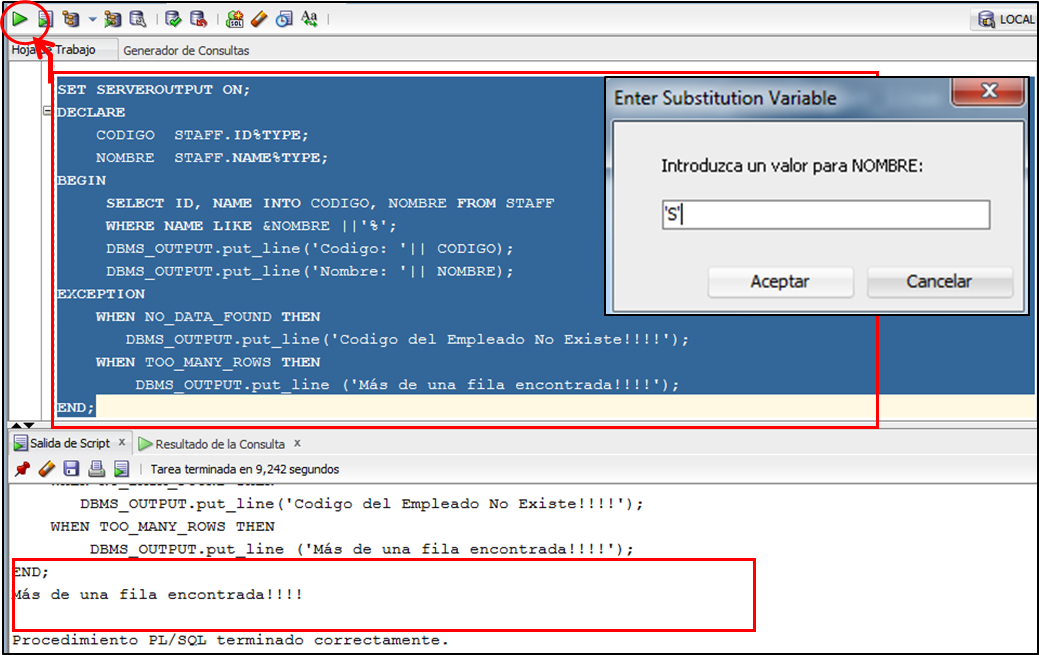
DBMS\_OUTPUT.put\_line('Codigo del Empleado No Existe!!!!');

**WHEN TOO\_MANY\_ROWS** THEN

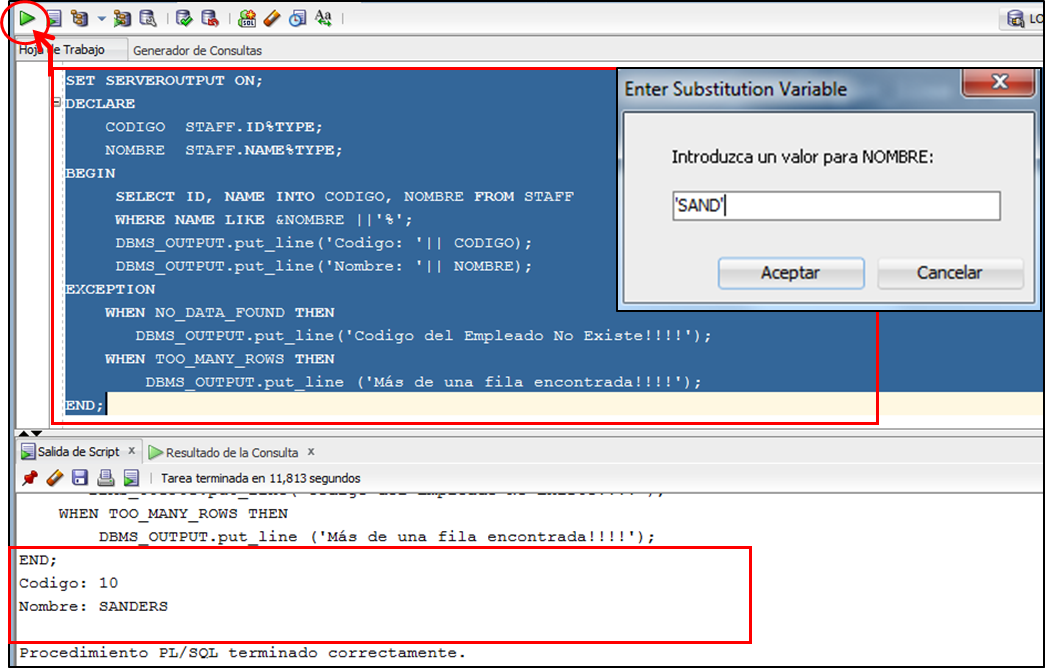
DBMS\_OUTPUT.put\_line ('Más de una fila encontrada!!!!');

**END;**

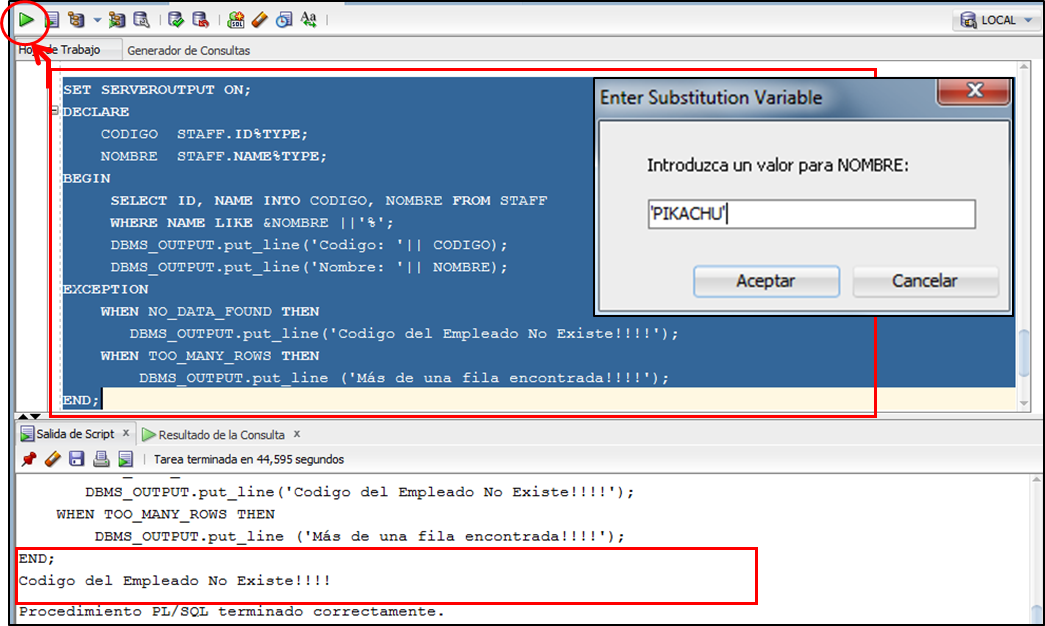
PRUEBA DESDE ORACLE: PROBANDO CON NOMBRES CON VARIAS CONCIDENCIAS.



**SE**GUNDA PRUEBA EN ORACLE: PROBANDO CON UN NOMBRE EXISTENTE ‘SAND’



**TERCERA PRUEBA EN ORACLE: PROBANDO CON UN NOMBRE NO EXISTENTE.**



En un bloque PL/SQL, realice una inserción de registros a la tabla STAFF. Usando excepciones predefinidas envíe los siguientes mensajes ‘Ya existe un departamento con el código ingresado’ o ‘Error desconocido’. Utilice los gestores DUP\_VAL\_ON\_INDEX y OTHERS.

SET SERVEROUTPUT ON;

**DECLARE**

**BEGIN**

**INSERT INTO** STAFF (ID, NAME, DEPT, JOB, YEARS, SALARY, COMM)

**VALUES** (**10**,'PIKACHU', 20,'MGR', 10, 2000.55, 250.99);

DBMS\_OUTPUT.PUT\_LINE ('Dpt. se ingreso correctamente!!!');

**COMMIT**;

**EXCEPTION**

**WHEN DUP\_VAL\_ON\_INDEX THEN**

DBMS\_OUTPUT.PUT\_LINE ('Ya existe un Dpt. con dicho Codigo');

**WHEN OTHERS THEN**

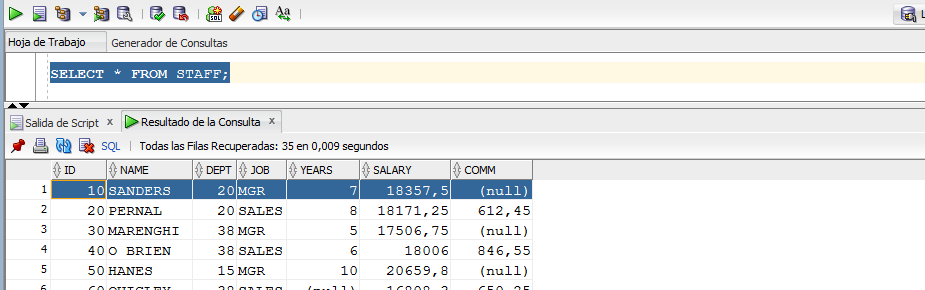
DBMS\_OUTPUT.PUT\_LINE ('Error desconocido');

**END;**

PRUEBA DESDE ORACLE. INGRESANDO UN REGISTRO NUEVO, SOLO QUE EL CODIGO YA EXISTE EN LA TABLA STAFF. ROMPIENDO LA REGLA DEL PRIMARY KEY.

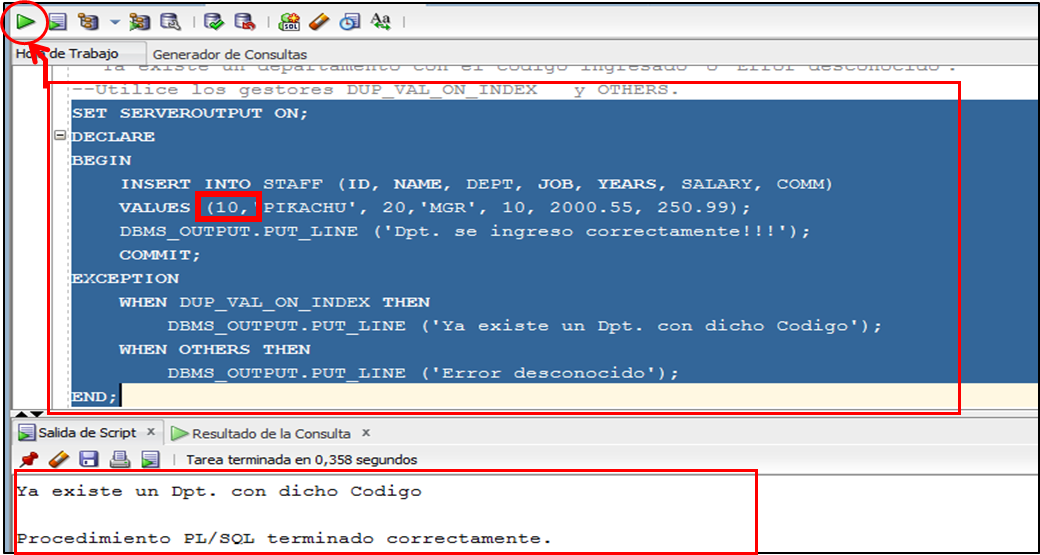
Antes de ejecutar el código, se recomienda realizar una consulta simple sobre la tabla STAFF, y verificar si el código o ID con el número 10, existe.

SELECT \* FROM STAFF;



Como se ve el código o ID con el valor 10 existe, si ejecuta el siguiente código, no saldrá un mensaje indicando que ya existe dicho código.

**PRUEBAR DESDE ORACLE:**



**ERRORES DE EXCEPCION, DEFINIDAS POR EL USUARIO.**

Este programa determinar si un número **v\_valor := -1;**  es positivo o negativo, mediante el uso de errores de excepción definidas por el usuario RAISE.

SET SERVEROUTPUT ON;

**DECLARE**

**v\_negativo EXCEPTION;**

v\_valor NUMBER;

**BEGIN**

**v\_valor := -1;**

**IF** v\_valor < 0 **THEN**

**RAISE** v\_negativo;

**END IF;**

DBMS\_OUTPUT.PUT\_LINE ('El valor ingresado es POSITIVO!!');

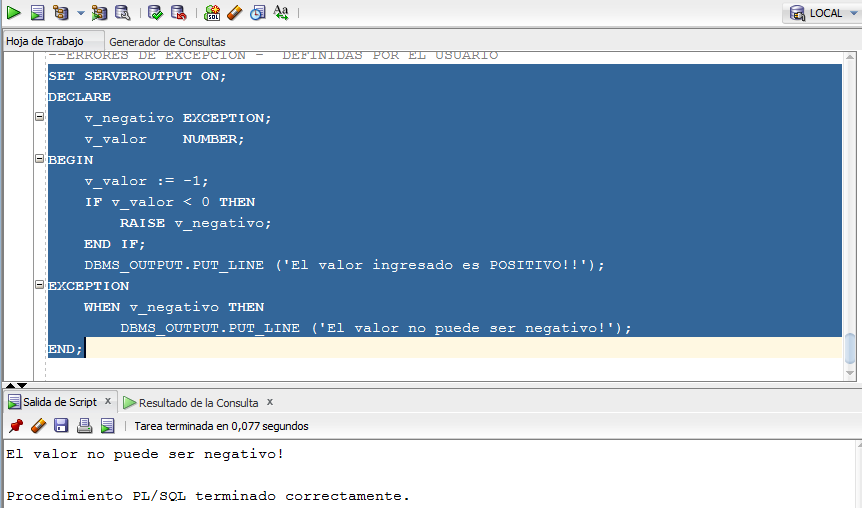
**EXCEPTION**

**WHEN v\_negativo THEN**

DBMS\_OUTPUT.PUT\_LINE ('El valor no puede ser negativo!');

**END;**

**PRUEBA DESDE ORACLE:**



En un bloque PL/SQL, realice la búsqueda con la inicial del nombre de persona (**usar variable de sustitución &)** y contabilizar los registros encontrados. Usando excepciones definidas por el usuario enviando los siguientes mensajes ‘**Ningún registro encontrado**’ – ‘**Una fila encontrada**’ o ‘**Más de una Fila encontrada**’. Utilice el gestor **RAISE**.

SET SERVEROUTPUT ON;

**DECLARE**

**mensaje1** **EXCEPTION**; **mensaje2** **EXCEPTION**; **mensaje3** **EXCEPTION**;

cant NUMBER; inicial VARCHAR2(1);

**BEGIN**

**SELECT** count(\*) **INTO** cant FROM STAFF where **NAME** like **&**inicial||'%';

**IF** cant=0 THEN

**RAISE** mensaje1;

**ELSIF** cant=1 THEN

**RAISE** mensaje2;

**ELSE**

**RAISE** mensaje3;

**END IF;**

**EXCEPTION**

**WHEN** **mensaje1** THEN

DBMS\_OUTPUT.put\_line('Ningun registro encontrado!!!!');

**WHEN** **mensaje2** THEN

DBMS\_OUTPUT.put\_line('Una fila encontrada!!!!');

**WHEN** **mensaje3** THEN

DBMS\_OUTPUT.put\_line('Mas de una fila encontrada!!!!');

**END;**

**TRATAMIENTO DE EXCEPCIONES.**

Cuando se genera una excepción, el control pasa a la sección de excepciones del Bloque, la cual está compuesta de uno o varios Gestores de excepciones, formada por la cláusula WHEN, quien identifica la excepción correspondiente.

**Sintaxis:**

**DECLARE**

v\_error1 **EXCEPTION**;

v\_error2 **EXCEPTION**;

**BEGIN**

**RAISE** v\_error1;

……

**RAISE** v\_error2;

….

**EXCEPTION**

**WHEN** v\_error1 **THEN**

…..

**WHEN** v\_error2 **THEN**

…..

**WHEN** OTHERS **THEN**

dbms\_output.put\_line( SQLCODE || SQLERRM);

**END;**

**FUNCIONES PARA IDENTIFICAR EXCEPCIONES.**

**SQLCODE**: devuelve un valor numérico, que corresponde al código de error SQL no se puede referenciar directamente, debe ser asignado a una variable PL/SQL tipo Number.

|  |  |
| --- | --- |
| **Valor de SQLCODE** | **Descripción** |
| 0 | No se encontró ninguna excepción. |
| 1 | Excepción definida por el usuario |
| +100 | Excepción NO\_DATA\_FOUND |
| Negativo | Otro número de error del servidor Oracle. |

**SQLERRM**: devuelve el mensaje asociado con el número del error. Tipo Varchar2.

**Ejemplo 1: identificación de excepciones: SQLCODE – SQLERRM**

Este ejemplo devuelve un error ya que no se puede dividir por cero.

Recorte de pantalla

SET SERVEROUTPUT ON;

**DECLARE**

err\_num NUMBER;

err\_msg VARCHAR2(255);

result NUMBER;

**BEGIN**

SELECT 1/0 INTO result FROM DUAL;

**EXCEPTION**

WHEN OTHERS THEN

err\_num := SQLCODE;

err\_msg := SQLERRM;

DBMS\_OUTPUT.put\_line('Error:'||TO\_CHAR(err\_num));

DBMS\_OUTPUT.put\_line(err\_msg);

**END;**

**Ejemplo 2: identificación de excepciones: SQLCODE – SQLERRM**

Este ejemplo devuelve un error ya que en la tabla STAFF no se puede grabar un ID con Null, dado que el campo ID es Not Null.



SET SERVEROUTPUT ON;

**DECLARE**

err\_num NUMBER;

err\_msg VARCHAR2(255);

result NUMBER;

**BEGIN**

INSERT INTO STAFF (ID) VALUES (NULL);

**EXCEPTION**

WHEN OTHERS THEN

err\_num := SQLCODE;

err\_msg := SQLERRM;

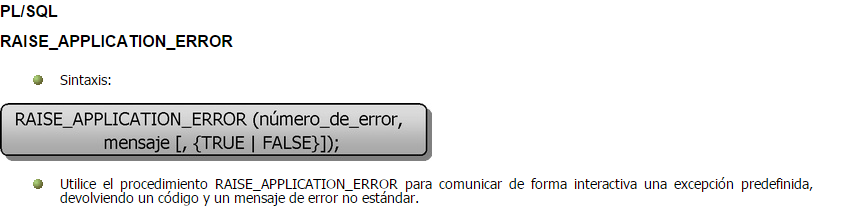
DBMS\_OUTPUT.put\_line('Error:'||TO\_CHAR(err\_num));

DBMS\_OUTPUT.put\_line(err\_msg);

**END;**

**RAISE\_APPLICATION\_ERROR**

Utilice el procedimiento RAISE\_APPLICATON\_ERROR para comunicar de forma interactiva una excepción predefinida, devolviendo un código y un mensaje de error no estándar.



RAISE\_APPLICATION\_ERROR (**num\_error**, **mje\_error**);

**Donde:**

**Num\_error**: número comprendido entre -20000 y -20999

**Mje\_error**: mensaje de error personalizado a mostrar.

Este ejemplo de error de excepción, que genera división por cero, de manera temporal asigna dicho error al código -20001 más un mensaje “**Ud. Hizo una división entre cero!!!”** correspondiente.

SET SERVEROUTPUT ON;

**Declare**

numero number;

**Begin**

numero:= 12/0;

DBMS\_OUTPUT.put\_line ('RESULTADO:' || TO\_CHAR (numero));

**Exception**

**WHEN** ZERO\_DIVIDE then

**RAISE\_APPLICATION\_ERROR**(-20001,'Ud. hizo una división entre cero!!!');

**End**;

El siguiente código se intenta buscar un código 7369, en la tabla STAFF no existe, se genera una excepcion al no encontrar dicho registro, este error usa al código -20099 y le asigna de manera temporal el mensaje de error: “**NO SE ENCONTRO NINGUN REGISTRO!!!**”

SET SERVEROUTPUT ON;

**DECLARE**

Codigo **STAFF.ID%TYPE**;

Nombre **STAFF.NAME%TYPE**;

**BEGIN**

Codigo:=7369;

**SELECT ID, NAME INTO Codigo, Nombre FROM STAFF where ID= Codigo;**

DBMS\_OUTPUT.put\_line ('Codigo: ' || Codigo);

DBMS\_OUTPUT.put\_line ('Nombre: ' || Nombre);

**EXCEPTION**

**WHEN** NO\_DATA\_FOUND **THEN**

**RAISE\_APPLICATION\_ERROR** (-20099,'NO SE ENCONTRO NINGUN REGISTROS!!!');

**END;**

CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA

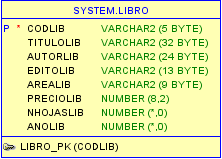
El objetivo de este capítulo es comprender la lógica de Programación en Base de Datos basado en el control de errores de Excepcion, de esta manera el estudiante se compenetra más con entorno **PL/SQL** de Oracle, al analizar la información, resultado de las programaciones realizadas de manera exitosa.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

ACTIVIDAD VIRTUAL

Ingresa a la plataforma virtual, luego revisa y analiza el tema desarrollado en la semana 6 para desarrollar las siguientes 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 LIBRO**(

CODLIB VARCHAR2(5) PRIMARY KEY NOT NULL,

TITULOLIB VARCHAR2 (32),

AUTORLIB VARCHAR2 (24),

EDITOLIB VARCHAR2 (13),

AREALIB VARCHAR2 (9),

PRECIOLIB NUMERIC (8, 2),

NHOJASLIB INT,

ANOLIB INT);

IMPORTANTE

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

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

**S7\_PL\_SQL\_1.sql,**

**FAVOR DE DESCARGAR DICHO ARCHIVO.**

**RESOLVER:**

Crear los siguientes programas PL/SQL con el control de ERRORES DE EXCEPCION:

1. Consultar un libro cualquiera (usar variables de sustitución &), use el campo código de libro para realizar la búsqueda, si existe, se debe mostrar todos sus datos, si en caso no existe, debe mostrarse un mensaje que diga: “**LIBRO NO EXISTE**”, y si hubiera muchas coincidencias, entonces debe mostrar el mensaje “**MUCHOS LIBROS COINCIDEN CON EL CRITERIO**”.
2. Intentar registrar un nuevo libro y sus demás datos, si el libro no existe debe mostrar el mensaje “**LIBRO GRABADO CON EXITO**”, si el libro existe debe mostrar el mensaje “**ERROR EL LIBRO YA EXISTE**”.
3. Usando **RAISE\_APPLICATION\_ERROR**, genere un error de excepción en una operación de división, es decir si en algún momento divide por cero, debe salir el mensaje “ERROR DIVISION POR CERO”, en caso contrario debe realizar la división y mostrar el resultado.