PROGRAMANDO CON PL/SQL EN ORACLE

PL/SQL (Procedural Language/Structured Query Language) es un lenguaje de programación que se encuentra agregado en Oracle, el cual incluye las siguientes características:

- Declaración de variables.
- Estructuras modulares.
- Estructuras de control de flujo y toma de decisiones.
- Gestión de excepciones
- Instrucciones SQL

El código PL/SQL se puede dividir en dos ramas: en código anónimo (no tienen nombre asignado) y en subprogramas (tienen nombre asignado).
Los subprogramas se pueden dividir en triggers, procedimientos almacenados, funciones y paquetes

```
CREATE USER user1 IDENTIFIED BY 123
        DEFAULT TABLESPACE users
        TEMPORARY TABLESPACE temp
        QUOTA UNLIMITED ON users;

CREATE ROLE PROGRAMADOR;

GRANT CREATE session, CREATE any table, CREATE any view,
        CREATE any procedure,
        ALTER any table, ALTER any procedure,
       DROP any table, DROP any view, DROP any procedure
       TO PROGRAMADOR;

--Consultar los privilegios de un rol
select * from dba_sys_privs where grantee = 'PROGRAMADOR';
SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE='PROGRAMADOR' ;

--ASIGNAR UN ROL A UN USUARIO
GRANT PROGRAMADOR TO USER1;

--CONSULTAR LOS ROLES
select * from dba_roles;

--Para ver los roles asignado a un rol
select role, granted_role from role_role_privs;

--roles asignados a un usuario en concreto
select grantee, granted_role from dba_role_privs where grantee = upper ('&USUARIO') order by
grantee;

--roles del usuario conectado en ese momento
select username, granted_role from user_role_privs;
```

```
--ASIGNARLE PERMISOS A UN ROL PARA QUE PUEDA REALIZAR ACCIONES EN TABLAS DE OTRO USUARIO
GRANT select,insert,update,delete ON HABITANTE.tblDepartamentos TO PROGRAMADOR;

--VER LOS PRIVILEGIOS DE SELECT, INSERT, UPDATE O DELETE DE UN ROL O USUARIO
select * from dba_tab_privs where grantee = upper('&NOMBRE_ROL_O_USUARIO') order by grantee;

--AGREGAR UN NUEVO PRIVILEGIO AL ROL CREADO
GRANT CREATE TRIGGER TO programador;

--ELIMINAR UN PRIVILEGIO DE UN ROL O USUARIO
REVOKE CREATE TRIGGER FROM programador;

--ELIMINAR PERMISO DE REALIZAR SELECT EN TABLA DE OTRO USUARIO
REVOKE SELECT ON HABITANTE.TBLDEPARTAMENTOS FROM PROGRAMADOR

--ELIMINAR TODOS LOS PRIVILEGIOS DE SELECT, UPDATE, INSERT Y DELETE ASIGNADOS A TABLAS DE OTRO USUARIO
REVOKE ALL ON HABITANTE.TBLDEPARTAMENTOS FROM PROGRAMADOR

Ejemplo # 1 – Crear un bloque anónimo que contenga una variable de tipo varchar2 y a la cual se le asigne un nombre personal para luego imprimir su valor
--habilita la salida por pantalla desde la consola
SET SERVEROUTPUT ON;

--crea un bloque anónimo con un mensaje en pantalla
DECLARE
        nombre VARCHAR2(100):=’Emilson Omar Acosta’;
BEGIN
        DBMS_OUTPUT.PUT_LINE (nombre);
END;

--crear un bloque anónimo para sumar dos números y concatenar el resultado con un mensaje en pantalla
DECLARE
        numero1 NUMBER(4,2):=10.1;
        numero2 NUMBER(4,2):=20.2;
BEGIN
        DBMS_OUTPUT.PUT_LINE('LA SUMA ES: '||TO_CHAR(numero1+numero2));
END;

--crear un bloque anónimo que obtenga el nombre del departamento con ID=1 y lo almacene en una variable para luego imprimir su valor en pantalla
DECLARE
        nombre VARCHAR2(50);
BEGIN
```

```
        SELECT NOMBREDEPARTAMENTO INTO nombre FROM USER1.TBLDEPARTAMENTOS
        WHERE IDDEPARTAMENTO=1;

        DBMS_OUTPUT.PUT_LINE('EL NOMBRE ES: '||nombre);
END;

--crear un bloque anónimo que obtenga el nombre y el id de país del departamento con ID=1 y los
almacene en variables para luego imprimir sus valores en pantalla
DECLARE
        nombre VARCHAR2(50);
        idpais number(38);
BEGIN
        SELECT NOMBREDEPARTAMENTO, IDPAIS INTO nombre, idpais FROM
        USER1.TBLDEPARTAMENTOS WHERE IDDEPARTAMENTO=1;

        DBMS_OUTPUT.PUT_LINE('EL NOMBRE ES: '||nombre||chr(13)||'CODIGO PAÍS:
        '||to_char(idpais));
END;

--crear un bloque anónimo que obtenga el nombre del habitante con ID=1 y lo guarde en una
variable que sea del mismo tipo de dato que la columna "NOMBREHABITANTE"
DECLARE
        nombre USER1.TBLHABITANTES.NOMBREHABITANTE%TYPE;
BEGIN
        SELECT NOMBREHABITANTE INTO nombre FROM USER1.TBLHABITANTES WHERE
        IDHABITANTE=1;

        DBMS_OUTPUT.PUT_LINE('EL NOMBRE ES: '||nombre);
END;

--crear un bloque anónimo que obtenga el todos los datos del departamento con ID=1 y lo guarde
en una variable que se haya mapeado con la misma estructura de un registro de la tabla
TBLDEPARTAMENTOS
DECLARE
        registro TBLDEPARTAMENTOS%ROWTYPE;
BEGIN
        SELECT * INTO registro FROM TBLDEPARTAMENTOS
        WHERE IDDEPARTAMENTO=1;

        DBMS_OUTPUT.PUT_LINE(registro.IDDEPARTAMENTO||' '||
        registro.NOMBREDEPARTAMENTO||' '||registro.IDPAIS);
END;

--crear un bloque anónimo que obtenga el todos los datos del departamento con ID=1 y el nombre
del país para ese departamento
DECLARE
        nombreDepto TBLDEPARTAMENTOS.NOMBREDEPARTAMENTO%TYPE;
        nombrePais_ TBLPAISES.NOMBREPAIS%TYPE;
```

```
BEGIN
        SELECT nombreDepartamento, nombrePais INTO
        nombreDepto, nombrePais_
        FROM TBLDEPARTAMENTOS INNER JOIN TBLPAISES
        ON TBLDEPARTAMENTOS.IDPAIS=TBLPAISES.IDPAIS
        WHERE IDDEPARTAMENTO=1;

        DBMS_OUTPUT.PUT_LINE('El nombre del departamento es:'||
        nombreDepto||' y el nombre del pais es:'||nombrePais_);
END;
```

SINTAXIS DE CONDICIÓN IF

```
IF (CONDICIÓN) THEN
        <INSTRUCCIONES>
ELSIF (CONDICIÓN) THEN
        <INSTRUCCIONES>
ELSE
        <INSTRUCCIONES>
END IF;
```

```
--crear un bloque anónimo que obtenga el nombre del habitante con ID=1 y lo guarde en una variable que sea del mismo tipo de dato que la columna “NOMBREHABITANTE” y luego compare si el nombre comienza con la letra “R”
DECLARE
        nombre USER1.TBLHABITANTES.NOMBREHABITANTE%TYPE;
BEGIN
        SELECT NOMBREHABITANTE INTO nombre FROM USER1.TBLHABITANTES WHERE
        IDHABITANTE=1;

        IF (SUBSTR(nombre,1,1)='R') THEN
                DBMS_OUTPUT.PUT_LINE('EL NOMBRE ES: '||nombre);
        ELSE
                DBMS_OUTPUT.PUT_LINE('EL NOMBRE NO CUMPLE LA CONDICIÓN');
        END IF;
END;
```

SINTAXIS DE CASE

```
CASE
        WHEN CONDICION1 THEN
                <INSTRUCCIONES>
        WHEN CONDICION2 THEN
                <INSTRUCCIONES>
        ELSE
                <INSTRUCCIONES>
END CASE;
```

```
--crear un bloque anónimo que obtenga el nombre del habitante con ID=1 y lo guarde en una variable que sea del mismo tipo de dato que la columna “NOMBREHABITANTE” y luego compare si el nombre comienza con la letra “R” o “D”
DECLARE
        nombre USER1.TBLHABITANTES.NOMBREHABITANTE%TYPE;
BEGIN
        SELECT NOMBREHABITANTE INTO nombre FROM USER1.TBLHABITANTES WHERE
        IDHABITANTE=1;

        CASE
                WHEN (SUBSTR(nombre,1,1)='R') THEN
                        DBMS_OUTPUT.PUT_LINE('EL NOMBRE ES: '||nombre);
                WHEN (SUBSTR(nombre,1,1)='D') THEN
                        DBMS_OUTPUT.PUT_LINE('EL NOMBRE ES: '||nombre);
                ELSE
                        DBMS_OUTPUT.PUT_LINE('EL NOMBRE NO CUMPLE LA CONDICIÓN');
        END CASE;
END;
```

SINTAXIS DE LOOP

```
LOOP
        <INSTRUCCIONES>
        EXIT WHEN CONDICION_SALIDA
END LOOP
```

```
--crear un bloque anónimo que imprima la fecha y hora del sistema de 5 días distintos y enumere los valores que se han imprimido
DECLARE
        iteracion NUMBER:=0;
BEGIN
        LOOP
                DBMS_OUTPUT.PUT_LINE(to_char(iteracion +1)||': '||TO_CHAR(SYSDATE +
                iteracion,'YYYY-MM-DD HH:MI:SS')||chr(13));

                iteracion:= iteracion +1;
                EXIT WHEN (iteracion >4);
        END LOOP;
END;
```

SINTAXIS DE WHILE

```
WHILE CONDICION LOOP
        <INSTRUCCIONES>
END LOOP
```

```
--crear un bloque anónimo que imprima la fecha y hora del sistema de 5 días distintos y enumere los valores que se han imprimido
DECLARE
        iteracion NUMBER:=0;
```

```
BEGIN
        WHILE (iteracion <5) LOOP
                DBMS_OUTPUT.PUT_LINE(to_char(iteracion +1)||': '||TO_CHAR(SYSDATE +
                iteracion,'YYYY-MM-DD HH:MI:SS')||chr(13));

                iteracion:= iteracion +1;
        END LOOP;
END;
```

SINTAXIS DE FOR

```
FOR VARIABLE IN INICIO..FINAL LOOP
        <INSTRUCCIONES>
END LOOP;
```

--crear un bloque anónimo que imprima la fecha y hora del sistema de 5 días distintos y enumere los valores que se han imprimido

```
DECLARE
        linea NUMBER:=0;
BEGIN
        FOR repeticion IN 1..5 LOOP
                DBMS_OUTPUT.PUT_LINE(to_char(linea+1)||': '||TO_CHAR(SYSDATE +
                linea,'YYYY-MM-DD HH:MI:SS')||chr(13));

                linea:=linea+1;
        END LOOP;
END;
```

SENTENCIA DE CONTROL GOTO

--crear un bloque anónimo que haga uso de la sentencia GOTO en la cual se haga uso de un contador y una variable llamada “total” que vaya almacenando la suma del valor que posee actualmente más el valor de la variable contador. Hacer esto mientras la variable total sea menor o igual que 20

```
DECLARE
        V_TOTAL NUMBER(9):=0;
        V_CONTADOR NUMBER(6):=0;
        LINEA NUMBER(3):=0;

BEGIN
        <<calcular>>
        V_CONTADOR:=V_CONTADOR+1;
        V_TOTAL:=V_TOTAL+V_CONTADOR;

        IF V_TOTAL <=20 THEN
                LINEA:=LINEA+1;
                DBMS_OUTPUT.PUT_LINE(LINEA||' CONTADOR ES: '||V_CONTADOR||
                   ' EL VALOR DE V_TOTAL ES: '|| V_TOTAL );
                GOTO calcular;
        END IF;
```

```
END;
```

USO DE INSTRUCCIONES DDL EN PL/SQL

```
--crear un bloque anónimo que cree una tabla llamada LOGS y con los campos id de tipo NUMBER, nombreUsuario de tipo VARCHAR2(50), fechaMod de tipo DATE y descripción de tipo VARCHAR2(100)
BEGIN
	EXECUTE IMMEDIATE 'CREATE TABLE USER1.LOGS(id NUMBER PRIMARY KEY,
	nombreUsuario VARCHAR2(50), fechaMod DATE, descripcion VARCHAR2(100))';
END;
```

USO DE INSTRUCCIONES DML EN PL/SQL

```
--crear un bloque anónimo que haga uso de la instrucción INSERT INTO en la tabla LOGS creada anteriormente y en la cual se ingresen tres registros
BEGIN
	INSERT INTO LOGS VALUES(1, 'Emilson', SYSDATE, 'Se elimino la tabla');
	INSERT INTO LOGS VALUES(2, 'Emilson', TO_DATE('02-19-2018 19:21:01','MM-DD-YYYY'),
	'Se elimino la tabla');
	INSERT INTO LOGS VALUES(3, 'Emilson', TO_DATE('02-19-2018 19:21:01','MM-DD-YYYY
	HH24:MI:SS'), 'Se elimino la tabla');
	COMMIT;
END;

--crear un bloque anónimo que inserte 2 registros en la tabla TBLHABITANTES, y forzar a que genere un error en la llave primaria duplicada, para luego gestionar este error por medio del control de excepciones y deshacer los cambios en caso de que suceda el error.
DECLARE
BEGIN
	INSERT INTO TBLHABITANTES values(20, 'Maria', 3);

	INSERT INTO TBLHABITANTES values(21, 'Luis', 3);
	COMMIT;
	EXCEPTION
	WHEN DUP_VAL_ON_INDEX THEN
		DBMS_OUTPUT.PUT_LINE('Error');
		ROLLBACK;
	WHEN OTHERS THEN
		IF (SQLCODE=-2) THEN
			DBMS_OUTPUT.PUT_LINE('Error en llave primaria'||SQLCODE||
			' '||SQLERRM);
		END IF;
END;
```

SINTAXIS DE CURSORES

```
DECLARE
	CURSOR <nombre cursor> IS <QUERY>;
BEGIN
	OPEN MICURSOR;
```

```
        FETCH MICURSOR INTO VARIABLES_PLSQL
        CLOSE MICURSOR;
END;
```

```
--crear un bloque anónimo que imprima todos los nombres de departamentos (USO DE CURSORES)
DECLARE
        CURSOR departamentos IS SELECT * FROM USER1.TBLDEPARTAMENTOS;
        registro USER1.TBLDEPARTAMENTOS%ROWTYPE;
BEGIN
        OPEN departamentos;
        LOOP
                FETCH departamentos INTO registro;
                EXIT WHEN departamentos%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(registro.nombredepartamento);
        END LOOP;
        CLOSE departamentos;
END;
```

Attribute	Type	Description
%ISOPEN	Boolean	Evaluates to TRUE if the cursor is open
%NOTFOUND	Boolean	Evaluates to TRUE if the most recent fetch does not return a row
%FOUND	Boolean	Evaluates to TRUE if the most recent fetch returns a row; complement of %NOTFOUND
%ROWCOUNT	Number	Evaluates to the total number of rows returned so far

```
--crear un cursor que obtenga todos los departamentos guardados y que muestre el nombre de pais
DECLARE
         CURSOR cursor_datos IS SELECT IDDEPARTAMENTO,
         NOMBREDEPARTAMENTO, TBLPAISES.IDPAIS, NOMBREPAIS
         FROM TBLDEPARTAMENTOS
         INNER JOIN TBLPAISES ON TBLDEPARTAMENTOS.IDPAIS=
         TBLPAISES.IDPAIS;
         IDDEPTO TBLDEPARTAMENTOS.IDDEPARTAMENTO%TYPE;
         NOMBDEPTO TBLDEPARTAMENTOS.NOMBREDEPARTAMENTO%TYPE;
         IDPAIS_ TBLDEPARTAMENTOS.IDPAIS%TYPE;
         NOMBPAIS TBLPAISES.NOMBREPAIS%TYPE;
BEGIN
        OPEN cursor_datos;
```

```
        LOOP
                FETCH cursor_datos INTO IDDEPTO, NOMBDEPTO, IDPAIS_,
                NOMBPAIS;
                EXIT WHEN cursor_datos%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE('EL NUMERO DE REGISTROS ES '
                ||cursor_datos%ROWCOUNT);
                DBMS_OUTPUT.PUT_LINE('EL id del depto es:'||
                IDDEPTO||' EL NOMBRE ES: '||
                NOMBDEPTO||' Y EL NOMBRE DE PAIS ES:'||
                NOMBPAIS);
        END LOOP;
END;

--crear un bloque anónimo que imprima el nombre del departamento y cuyo código de
departamento es recibido como parámetro
DECLARE
        CURSOR departamentos (idDepto NUMBER:=1) IS
         SELECT * FROM USER1.TBLDEPARTAMENTOS WHERE idDepartamento=idDepto;
        registro USER1.TBLDEPARTAMENTOS%ROWTYPE;
BEGIN
        OPEN departamentos;
        LOOP
                FETCH departamentos INTO registro;
                EXIT WHEN departamentos%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE(registro.nombredepartamento);
        END LOOP;
        CLOSE departamentos;
END;
--crear un bloque anónimo que haga uso de cursores y obtenga todos los departamentos. Se debe
imprimir el nombre, id del departamento y el nombre del país al que pertenece
DECLARE
        CURSOR cDat IS SELECT nombredepartamento, iddepartamento, nombrepais
        FROM TBLDEPARTAMENTOS INNER JOIN TBLPAISES ON
        TBLDEPARTAMENTOS.IDPAIS=TBLPAISES.IDPAIS;
        registro cDat%ROWTYPE;
BEGIN
        OPEN cDat;
        LOOP
                FETCH cDat INTO registro;
                EXIT WHEN cDat%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE('El nombre del depto es:
                '||registro.nombredepartamento);
                DBMS_OUTPUT.PUT_LINE('El id del depto es: '||registro.iddepartamento);
                DBMS_OUTPUT.PUT_LINE('El nombre pais es: '||registro.nombrepais);
        END LOOP;
        CLOSE cDat;
END;
```

USO DE INSTRUCCIÓN SELECT Y GESTIÓN DE EXCEPCIONES

```
--crear un bloque anónimo que obtenga todos los registros de la tabla TBLDEPARTAMENTOS y en caso de error crear una excepción que muestre un mensaje en pantalla
DECLARE
	registro USER1.TBLDEPARTAMENTOS%ROWTYPE;
BEGIN
	SELECT * INTO registro FROM USER1.TBLDEPARTAMENTOS;
	DBMS_OUTPUT.PUT_LINE(registro.NOMBREDEPARTAMENTO);
	DBMS_OUTPUT.PUT_LINE('cantidad de registros: '||SQL%ROWCOUNT);
	EXCEPTION
	WHEN OTHERS THEN
		DBMS_OUTPUT.PUT_LINE('SE RETORNAN VARIOS REGISTROS '||SQLERRM);
END;

--crear un bloque anónimo que inserte un registro en la tabla TBLDEPARTAMENTOS, este registro debe tener un ID igual a 4 y el nombre del departamento debe ser igual a OLANCHO, en caso de haber algún error se debe gestionar la excepción para mostrar un mensaje
DECLARE
	nombre VARCHAR2(5);
BEGIN
	nombre:='Islas de la Bahia';
	INSERT INTO TBLDEPARTAMENTOS VALUES (5,'OLANCHO',1);
	INSERT INTO TBLDEPARTAMENTOS VALUES (6, nombre,1);

	EXCEPTION
	WHEN DUP_VAL_ON_INDEX THEN
		DBMS_OUTPUT.PUT_LINE('Ingrese un ID distinto, este ID ya existe');
		ROLLBACK;
	WHEN VALUE_ERROR THEN
		DBMS_OUTPUT.PUT_LINE('El tamaño de los datos son invalidos');
	WHEN OTHERS THEN
		IF (SQLCODE=-12899) THEN
			DBMS_OUTPUT.PUT_LINE('Error de tamaño en los datos');
	END IF;
END;

--crear un bloque anónimo que inserte 3 registros en la tabla habitantes y que en uno de los registros se produzca un error para utilizar el rollback hacia un savepoint
DECLARE
	 punto_rest NUMBER:=0;
BEGIN
	 INSERT INTO TBLHABITANTES values(21, 'Maria', 3);
	 SAVEPOINT insercion1;
	 punto_rest:=1;
	 COMMIT;
	 INSERT INTO TBLHABITANTES values(22, 'Luis', 3);
	 SAVEPOINT insercion2;
```

```
            punto_rest:=2;
            COMMIT;
            INSERT INTO TBLHABITANTES values(21, 'CARLA', 3);
            SAVEPOINT insercion3;
            punto_rest:=3;
            COMMIT;

            EXCEPTION
             WHEN DUP_VAL_ON_INDEX THEN
               if (punto_rest=1) then
                 DBMS_OUTPUT.PUT_LINE('Error en el insert 1');
                 ROLLBACK TO SAVEPOINT insercion1;
               end if;
               if (punto_rest=2) then
                 DBMS_OUTPUT.PUT_LINE('Error en el insert 2');
                 ROLLBACK TO SAVEPOINT insercion2;
               end if;
               if (punto_rest=3) then
                 DBMS_OUTPUT.PUT_LINE('Error en el insert 3');
                 ROLLBACK TO SAVEPOINT insercion3;
               end if;
             WHEN OTHERS THEN
               DBMS_OUTPUT.PUT_LINE('Error en llave primaria'||SQLCODE||
               ' '||SQLERRM);
END;

--crear un bloque anónimo en el cual se haga uso de una tabla para almacenar los registros que
obtiene una consulta que hace uso del BULK COLLECT
DECLARE
        TYPE t_reg IS TABLE OF TBLDEPARTAMENTOS%ROWTYPE INDEX BY
        PLS_INTEGER;
        registro t_reg;
BEGIN
        SELECT * BULK COLLECT INTO registro FROM TBLDEPARTAMENTOS;
        DBMS_OUTPUT.PUT_LINE(registro(1).NOMBREDEPARTAMENTO);
        DBMS_OUTPUT.PUT_LINE('cantidad de registros: '||SQL%ROWCOUNT);
        EXCEPTION
                WHEN OTHERS THEN
                DBMS_OUTPUT.PUT_LINE('SE RETORNAN VARIOS REGISTROS '||SQLERRM);
END;

DECLARE
          TYPE reg IS RECORD
          (
            IDHAB TBLHABITANTES.IDHABITANTE%TYPE,
            NOMBHAB TBLHABITANTES.NOMBREHABITANTE%TYPE,
            NOMBMUN TBLMUNICIPIOS.NOMBREMUNICIPIO%TYPE
          );
```

```
          TYPE t_reg IS TABLE OF reg
          INDEX BY PLS_INTEGER;
          registro t_reg;
BEGIN
          select IDHABITANTE, NOMBREHABITANTE,
          NOMBREMUNICIPIO BULK COLLECT INTO registro from tblhabitantes
          INNER JOIN TBLMUNICIPIOS ON TBLHABITANTES.IDMUNICIPIO=
          TBLMUNICIPIOS.IDMUNICIPIO
          ORDER BY IDHABITANTE ASC;

          DBMS_OUTPUT.PUT_LINE('LA CANTIDAD DE REGISTROS ES: '
          ||SQL%ROWCOUNT);

          FOR iteracion IN 1..SQL%ROWCOUNT LOOP
            DBMS_OUTPUT.PUT_LINE('El id del habitante: '
            ||registro(iteracion).idhab);
            DBMS_OUTPUT.PUT_LINE('El nombre del habitante: '
            ||registro(iteracion).nombhab);
            DBMS_OUTPUT.PUT_LINE('El nombre del habitante: '
            ||registro(iteracion).nombmun);
          END LOOP;
END;

--arreglos asociativos
DECLARE
        TYPE t_reg IS TABLE OF VARCHAR2(50) INDEX BY VARCHAR2(50);
        registro t_reg;
BEGIN
        registro('DEPTO1'):='Olancho';
        registro('DEPTO2'):='Islas de la Bahía';
        registro('DEPTO3'):='Choluteca';
        registro('DEPTO4'):='Lempira';

        DBMS_OUTPUT.PUT_LINE(registro.first);
        DBMS_OUTPUT.PUT_LINE(registro.count);
        DBMS_OUTPUT.PUT_LINE(registro.next('DEPTO3'));
        DBMS_OUTPUT.PUT_LINE(registro('DEPTO4'));
END;

DECLARE
        TYPE rDeptos IS RECORD
        (       codDepto    NUMBER,
                nombeDepto VARCHAR2(100),
                nombrePais VARCHAR2(100)
         );
        TYPE deptos IS TABLE OF rDeptos INDEX BY BINARY_INTEGER ;
        tDeptos deptos;
BEGIN
```

```
        SELECT * BULK COLLECT INTO tDeptos FROM TBLDEPARTAMENTOS INNER JOIN TBLPAISES
        ON TBLDEPARTAMENTOS.IDPAIS=TBLPAISES.IDPAIS;
        DBMS_OUTPUT.PUT_LINE(tDeptos (1).codDepto);
        DBMS_OUTPUT.PUT_LINE(tDeptos (1).nombreDepto);
        DBMS_OUTPUT.PUT_LINE(tDeptos (1).nombrePais);
END;

DECLARE
        TYPE t_reg IS TABLE OF TBLDEPARTAMENTOS%ROWTYPE INDEX BY VARCHAR2(50);
        registro t_reg;
BEGIN
        registro('ID').IDDEPARTAMENTO:=5;
        registro('ID').NOMBREDEPARTAMENTO:='Olancho';

        DBMS_OUTPUT.PUT_LINE(registro('ID').NOMBREDEPARTAMENTO);
END;
```

CREACIÓN DE TRIGGERS

SINTAXIS

```
CREATE OR REPLACE TRIGGER <NOMBRE_TRIGGER>
        INSTEAD OF|BEFORE|AFTER
        INSERT|DELETE|UPDATE OF <COLUMNAS>
        ON <TABLA>
        FOR EACH ROW
        WHEN <CONDICION>
DECLARE
BEGIN
        <CODIGO>
END;
```

Existen 2 variables que se pueden utilizar en los triggers, estas son:
:New y :Old
Estas variables se pueden utilizar en base a la siguiente tabla:

Data Operations	Old Value	New Value
INSERT	NULL	Inserted value
UPDATE	Value before update	Value after update
DELETE	Value before delete	NULL

```
--creación de secuencias
CREATE SEQUENCE sec_tblDeptos
 START WITH 1
 INCREMENT BY 1;

DROP SEQUENCE sec_tblDeptos;
```

```
ALTER SEQUENCE sec_tblDeptos
  INCREMENT BY 2;

--crear secuencia para la tabla LOGS y luego utilizar esa secuencia para gestionar un campo autoincrementable en la llave primaria de la tabla LOGS
GRANT CREATE SEQUENCE TO PROGRAMADOR;

CREATE SEQUENCE incremento_tbl_logs
  START WITH 1
  INCREMENT BY 1;

CREATE OR REPLACE TRIGGER LOGS_SEC
        BEFORE INSERT ON LOGS
        FOR EACH ROW
DECLARE
BEGIN
        :NEW.ID:=incremento_tbl_logs.NEXTVAL;
END;
INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION) VALUES (USER, SYSDATE, 'SE INSERTÓ UN NUEVO REGISTRO');

--crear un trigger que se ejecute después de la inserción de un dato en la tabla TBLDEPARTAMENTOS y que guarde un registro en la tabla LOGS con la información de la inserción
GRANT CREATE ANY TRIGGER, ALTER ANY TRIGGER, DROP ANY TRIGGER TO PROGRAMADOR;

CREATE OR REPLACE TRIGGER tg_tblDepartamentos
        AFTER INSERT ON TBLDEPARTAMENTOS
        FOR EACH ROW
DECLARE
BEGIN
        INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION) VALUES
        ('EMILSON',SYSDATE,'Se insertó un nuevo registro con valores id='
        ||:NEW.iddepartamento||' nombre='||:NEW.nombredepartamento||
        ' idpais='||:NEW.idpais);
END;

INSERT INTO TBLDEPARTAMENTOS (IDDEPARTAMENTO, NOMBREDEPARTAMENTO, IDPAIS)
VALUES (5,'Yoro',1);
```

```
--crear un trigger que se ejecute antes de actualizar un dato en la tabla TBLDEPARTAMENTOS
CREATE OR REPLACE TRIGGER tgUpdate_tblDepartamentos
        BEFORE UPDATE ON TBLDEPARTAMENTOS
        FOR EACH ROW
DECLARE
BEGIN
        INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION) VALUES
        (USER,SYSDATE,'Se modificó un registro el cual tenía los valores id='
        ||:OLD.iddepartamento||' nombre='||:OLD.nombredepartamento||
        ' idpais='||:OLD.idpais);
END;

UPDATE TBLDEPARTAMENTOS SET NOMBREDEPARTAMENTO='Atlántida' WHERE
NOMBREDEPARTAMENTO='Yoro';

--crear un trigger que se ejecute antes de eliminar un registro en la tabla TBLHABITANTES y que almacene en la tabla LOGS un registro de los datos eliminados
CREATE OR REPLACE TRIGGER tgrEliminarHabitante
        BEFORE DELETE ON TBLHABITANTES
        FOR EACH ROW
        BEGIN
                INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION)
                VALUES(USER, SYSDATE, 'SE ELIMINO UN REGISTRO DE LA TABLA
                TBLHABITANTES Y LOS DATOS ELIMINADOS SON IDHABITANTE='||
                :OLD.IDHABITANTE||' NOMBRE='||:OLD.NOMBREHABITANTE||' IDMUN='||
                :OLD.IDMUNICIPIO);
        END;

--crear un trigger que se ejecute después de actualizar un registro en la tabla TBLMUNICIPIOS y que almacene en la tabla LOGS un registro los cambios sucedidos, almacenando tanto los valores viejos como los valores nuevos.
CREATE OR REPLACE TRIGGER tgrActualizaMun
        AFTER UPDATE ON TBLMUNICIPIOS
        FOR EACH ROW
        BEGIN
                INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION)
                VALUES(USER, SYSDATE, 'SE ACTUALIZO UN REGISTRO DE LA TABLA
                TBLMUNICIPIOS, LOS DATOS SON nomMunNew='||
                :NEW.NOMBREMUNICIPIO||' nomMunOld='||:OLD.NOMBREMUNICIPIO);
        END;

--crear un trigger que se ejecute antes de una inserción, modificación o eliminación de un registro en la tabla TBLPAISES
CREATE OR REPLACE TRIGGER tgrTblPaises
BEFORE UPDATE OR INSERT OR DELETE ON TBLPAISES
FOR EACH ROW
DECLARE
BEGIN
```

```
 IF INSERTING THEN
   INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION) VALUES
   (USER,SYSDATE,'Se insertó un registro el cual tiene los valores id='
   ||:NEW.idpais||' nombre='||:NEW.nombrepais);
 ELSIF UPDATING THEN
   INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION) VALUES
   (USER,SYSDATE,'Se modificó un registro el cual tenía los valores id='
   ||:OLD.idpais||' nombre='||:OLD.nombrepais);
 ELSIF DELETING THEN
   INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION) VALUES
   (USER,SYSDATE,'Se eliminó un registro el cual tenía los valores id='
   ||:OLD.idpais||' nombre='||:OLD.nombrepais);
 END IF;

 EXCEPTION
   WHEN OTHERS THEN
         INSERT INTO LOGS (NOMBREUSUARIO, FECHAMOD, DESCRIPCION) VALUES
         (USER,SYSDATE,'La operación no se pudo realizar');
END;
```