Base de Datos II

Sintaxis a Aprender

- Crear usuario y asignar privilegios

```
ALTER SESSION SET"_oracle_script"=TRUE;
CREATE USER FGARCIA IDENTIFIED BY 123;

GRANT CONNECT, RESOURCE, dba TO FGARCIA ;
GRANT UNLIMITED TABLESPACE TO FGARCIA ;

ALTER PROFILE DEFAULT LIMIT PASSWORD_REUSE_TIME UNLIMITED;
ALTER PROFILE DEFAULT LIMIT PASSWORD_LIFE_TIME UNLIMITED;

-- REVOKE
-- DROP USER FGARCIA CASCADE;


CREATE ROLE DESARROLLADOR;
GRANT CREATE SESSION, CREATE TABLE, ALTER ANY TABLE, DROP ANY TABLE,
INSERT ANY TABLE, UPDATE ANY TABLE, DELETE ANY TABLE, SELECT ANY TABLE
TO DESARROLLADOR

SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE = 'DESARROLLADOR';

GRANT DESARROLLADOR TO FGARCIA;
```

- Excepciones

```
SET SERVEROUTPUT ON

DECLARE
    empl   employees%rowtype;
BEGIN
    SELECT
        *
    INTO
        empl
    FROM
        employees
    WHERE
        employee_id > 1;

    dbms_output.put_line(empl.first_name);
EXCEPTION
    WHEN ex1 THEN
        NULL;
    WHEN ex2 THEN
        NULL;
    WHEN OTHERS THEN
        NULL;
```

```
END;
/

--Excepciones predefinidas

SET SERVEROUTPUT ON

DECLARE
    empl employees%rowtype;
BEGIN
    SELECT
        *
    INTO empl
    FROM
        employees
    WHERE
        employee_id > 1;

    dbms_output.put_line(empl.first_name);
EXCEPTION
-- NO_DATA_FOUND   ORA-01403
-- TOO_MANY_ROWS
-- ZERO_DIVIDE
-- DUP_VAL_ON_INDEX
    WHEN no_data_found THEN
        dbms_output.put_line('ERROR, EMPLEADO INEXISTENTE');
    WHEN too_many_rows THEN
        dbms_output.put_line('ERROR, DEMASIADOS EMPLEADO');
    WHEN OTHERS THEN
        dbms_output.put_line('ERROR INDEFINIDO');
END;
/


SET SERVEROUTPUT ON
DECLARE
   MI_EXCEP EXCEPTION;
   PRAGMA EXCEPTION_INIT(MI_EXCEP,-937);
   V1 NUMBER;
   V2 NUMBER;
BEGIN
    SELECT EMPLOYEE_ID,SUM(SALARY) INTO V1,V2 FROM EMPLOYEES;
    DBMS_OUTPUT.PUT_LINE(V1);
EXCEPTION
   WHEN MI_EXCEP THEN
       DBMS_OUTPUT.PUT_LINE('FUNCION DE GRUPO INCORRECTA');
   WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('ERROR INDEFINIDO');
END;


CREATE TABLE ERRORS
    ( codigo      NUMBER
                    CONSTRAINT errors_code_nn NOT NULL

    , mensaje    VARCHAR2(256)
    )



SET SERVEROUTPUT ON
```

```
DECLARE
   EMPL EMPLOYEES%ROWTYPE;
   CODE NUMBER;
   MESSAGE VARCHAR2(100);
BEGIN
   SELECT * INTO EMPL FROM EMPLOYEES;
    DBMS_OUTPUT.PUT_LINE(EMPL.SALARY);
EXCEPTION
   WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE(SQLCODE);
        DBMS_OUTPUT.PUT_LINE(SQLERRM);
        CODE:=SQLCODE;
        MESSAGE:=SQLERRM;
        INSERT INTO ERRORS VALUES (CODE,MESSAGE);
        COMMIT;
END;

SELECT * FROM ERRORS;


/

SET SERVEROUTPUT ON
DECLARE
  REG REGIONS%ROWTYPE;
  REG_CONTROL REGIONS.REGION_ID%TYPE;
BEGIN
   REG.REGION_ID:=5;
   REG.REGION_NAME:='Africa';
   SELECT REGION_ID INTO REG_CONTROL FROM REGIONS
   WHERE REGION_ID=REG.REGION_ID;
   DBMS_OUTPUT.PUT_LINE('LA REGION YA EXISTE');
EXCEPTION
   WHEN NO_DATA_FOUND  THEN
        INSERT INTO REGIONS VALUES (REG.REGION_ID,REG.REGION_NAME);
        COMMIT;
END;

/

DECLARE
   reg_max EXCEPTION;
   regn NUMBER;
   regt varchar2(200);
BEGIN
   regn:=101;
   regt:='ASIA';
   IF regn > 100 THEN
         RAISE reg_max;
    ELSE
       insert into regions values (regn,regt);
       commit;
      END IF;
EXCEPTION
  WHEN reg_max THEN
    DBMS_OUTPUT.PUT_LINE('La region no puede ser mayor de 100.');
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error indefinido');
END;
/
```

```
DECLARE
   regn NUMBER;
   regt varchar2(200);
BEGIN
   regn:=101;
   regt:='ASIA';
   iF regn > 100 THEN
       -- EL CODIGO DEBE ESTAR ENTRE -20000 Y -20999
       RAISE_APPLICATION_ERROR(-20001,'LA ID NO PUEDE SER MAYOR DE 100');
    ELSE
       insert into regions values (regn,regt);
       commit;
    END IF;
END;
```

- Bloques Anónimos

```
-- Estructura Básica
/* Para imprimir en pantalla usamos*/
SET SERVEROUTPUT ON;
DECLARE
  /*Aquí se declararán las variables a utilizar.
    Este parte es opcional*/
  NOMBRE_VARIABLE TIPO_VARIABLE;
  -- para asignar valor a una varible se utiliza :=
BEGIN
  /*Instrucciones*/
  DBMS_OUTPUT.PUT_LINE('Lo que queremos imprimir en pantalla'||NOMBRE_VARIABLE)
END;
```

- Cursores

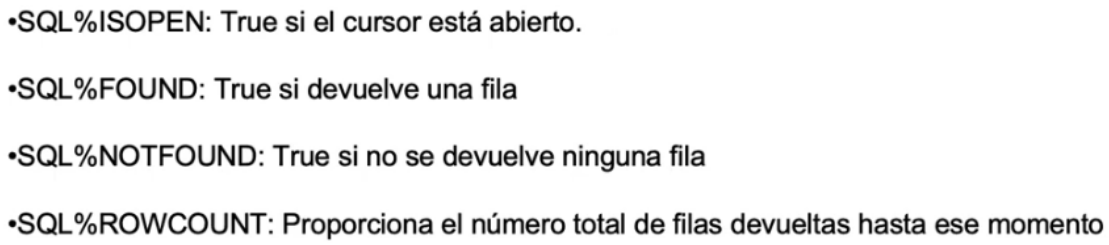
```
DECLARE
  CURSOR NOMBRE-DEL-CURSOR IS SELECT NOMBRE-COLUMNAS FROM NOMBRE-TABLA
BEGIN
  SENTENCAIAS DEL BLOQUE;
END;

-----------------------

DECLARE
  CURSOR NOMBRE-DEL-CURSOR(NOMBRE DE procedimiento IN TIPO DE DATO) IS
  SELECT NOMBRE-COLUMNAS FROM NOMBRE-TABLA
  WHERE CONDICION[INCLUYENDO LA VARIABLE]
BEGIN
  SENTENCAIAS DEL BLOQUE;
END;
```

  - Ejemplo de cursores

```
DECLARE
    CURSOR C1 IS SELECT * FROM REGIONS;
    V1 REGIONS%ROWTYPE;
```

```
BEGIN
    OPEN C1; --abrimos el cursor
    LOOP -- usamos este loop para que recorra todas las filas del cursor
    FETCH C1 INTO V1; -- recorremos el cursor de una fila en una
    EXIT WHEN C1%NOTFOUND; --
    dbms_output.put_line(V1.REGION_NAME);
    END LOOP;
    CLOSE C1;
END;

--si usamos un ciclo for no teneemos que abrir el cursor y ponerle la condicion de salida
DECLARE
    CURSOR C1 IS SELECT * FROM REGIONS;
BEGIN
    for i in C1 LOOP
    dbms_output.put_line(i.REGION_NAME);
    END LOOP;
END;

-- PODEMOS OMITIR LA DECLARACION
BEGIN
    FOR i IN (SELECT * FROM REGIONS) LOOP
        dbms_output.put_line(i.REGION_NAME);
    END LOOP;
END;

--ejemplo con parametros
DECLARE
    CURSOR C1(SALARIO NUMBER) IS -- LE PASAMOS PARAMETROS AL CURSOR
    SELECT * FROM EMPLOYEES WHERE SALARY > SALARIO;
    EMPL EMPLOYEES%ROWTYPE;
BEGIN
    FOR i IN C1(10000) LOOP
        dbms_output.put_line(i.FIRST_NAME || ' <-> ' || i.SALARY);
    END LOOP;
END;

--- Para modificar datos con un cursor
DECLARE
    CURSOR CUR IS SELECT * FROM EMPLOYEES FOR UPDATE; --
    EMPL EMPLOYEES%ROWTYPE;
BEGIN
    OPEN CUR;
    LOOP
        FETCH CUR INTO EMPL;
        EXIT WHEN CUR%NOTFOUND;
        IF EMPL.COMMISSION_PCT IS NOT NULL THEN
            UPDATE EMPLOYEES SET SALARY=SALARY*1.10 WHERE CURRENT OF CUR;
        ELSE
            dbms_output.put_line('ERROR INESPERADO');
        END IF;
    END LOOP;
END;
```

- propiedades



- Procedimientos

```
CREATE OR REPLACE PROCEDURE [ESQUEMA].NOMBRE_PROCEDIMIENTO
(NOMBRE_PARAMETRO [IN|OUT|IN OUT] TIPO_DATO) [IS|AS]
/*
  DECLARACION DE VARIABLES;
  DECLARACION DE CONSTANTES;
  DECLARACION DE CURSORES;
*/
BEGIN
  -- CUERPO DEL BLOQUE PL/SQL
  EXCEPTION  -- (OPCIONAL)
    WHEN (NOMBRE DE EXCEPCION) THEN
      -- ACCION A REALIZAR
END;
```

  - Ejemplos de Procedimientos

```
--REPLACE REEMPLAZA EL PROCEDIMIENTO SI YA EXISTE
--EL PROCEDIMIENTO SE PUEDE CREAR SOLO CON EL CREATE SIN EL REPLACE
CREATE OR REPLACE PROCEDURE PR1
IS
    X NUMBER:=10;
BEGIN
    dbms_output.put_line(X);
END;

-- LOS PROCEDIMIENTO SE LLAMAN DESDE OTRO LADO PARA EJECUTARSE.
BEGIN
    PR1;
END;

SELECT * FROM USER_OBJECTS WHERE OBJECT_TYPE='PROCEDURE'; -- esto es para ver los procedimientos

SELECT TEXT FROM USER_SOURCE WHERE NAME='PR1'; -- aca lo pasamos a texto para ver el "codigo"

------------------------
---- procedimientos con parametros
-- LOS PARAMETROS DE ENTRADA SON SOLO DE LECTURAS
CREATE OR REPLACE PROCEDURE CAL_TAX
(EMPL IN EMPLOYEES.EMPLOYEE_ID%TYPE, T1 IN NUMBER, R1 OUT NUMBER)
IS
    TAX NUMBER:=0;
    SALARIO EMPLOYEES.SALARY%TYPE:=0;
BEGIN
```

```
    SELECT SALARY INTO SALARIO FROM EMPLOYEES WHERE EMPLOYEE_ID=EMPL;
    R1:=SALARIO*T1/100;
    dbms_output.put_line('SALARIO: '|| SALARIO);
    dbms_output.put_line('IMPUESTO (R1): '|| R1);

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('DATOS NO ENCONTRADOS');
END;

SET SERVEROUTPUT ON
DECLARE
    A NUMBER:=120;
    B NUMBER:=5;
    R NUMBER:=0; --SIEMPRE ES CERO EN LA VARIABLE DE SALIDA
BEGIN
    dbms_output.put_line('R [INICIAL]: '|| R);
    CAL_TAX(A,B,R);
    dbms_output.put_line('R [DESPUES DE ENTRAR AL PROCEDIMIENTO]: '|| R);
END;

---------------------

SET SERVEROUTPUT ON
-- LOS PARAMETROS DE ENTRADA-SALIDA SI PUEDEN TENER UNA ASIGNACI�N
CREATE OR REPLACE PROCEDURE CAL_TAX
(EMPL IN EMPLOYEES.EMPLOYEE_ID%TYPE, T1 IN OUT NUMBER)
IS
    TAX NUMBER:=0;
    SALARIO EMPLOYEES.SALARY%TYPE:=0;
BEGIN
    SELECT SALARY INTO SALARIO FROM EMPLOYEES WHERE EMPLOYEE_ID=EMPL;
    T1:=SALARIO*T1/100;
    dbms_output.put_line('SALARIO: '|| SALARIO);
    dbms_output.put_line('IMPUESTO (T1): '|| T1);

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('DATOS NO ENCONTRADOS');
END;

SET SERVEROUTPUT ON
DECLARE
    A NUMBER:=120;
    B NUMBER:=5;
BEGIN
    dbms_output.put_line('B [ANTES DE ENTRAR AL PROCEDIMIENTO]: '|| B);
    CAL_TAX(A,B);
    dbms_output.put_line('B [ES T1]: '|| B);
END;


-----------------
-- prcedimiento con un cursor
CREATE OR REPLACE PROCEDURE VISUALIZAR
IS
    CURSOR C1 IS SELECT FIRST_NAME, SALARY FROM EMPLOYEES;
BEGIN
    FOR i IN C1 LOOP
    dbms_output.put_line(i.FIRST_NAME || '  ->  ' || i.SALARY);
    END LOOP;
```

```
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            dbms_output.put_line('DATOS NO ENCONTRADOS');
END;

EXECUTE VISUALIZAR;
```

- Funciones

```
CREATE OR REPLACE FUNCTION [ESQUEMA].NOMBRE_FUNCION
(NOMBRE_PARAMETRO [IN|OUT|IN OUT] TIPO_DATO) RETURN TIPO_DATO_RETORNO
[IS|AS]
/*
  DECLARACION DE VARIABLES;
  DECLARACION DE CONSTANTES;
  DECLARACION DE CURSORES;
*/
BEGIN
  -- CUERPO DEL BLOQUE PL/SQL
  EXCEPTION  -- (OPCIONAL)
    WHEN (NOMBRE DE EXCEPCION) THEN
      -- ACCION A REALIZAR
END;

-- LLAMADA A LA FUNCION
DECLARE
  VARIABLES QUE COINCIDAN CON LAS DE LA FUNCION
  ** LAS VARIABLES DE SALIDA SIEMPRE SE INICIALIZAN EN CERO
BEGIN
  CUERPO DEL BLOQUE ANONIMO
END;

/* Crear una funci□n que tenga como par□metro un n□mero de
departamento y que devuelve la suma de los salarios de dicho
departamento. La imprimimos por pantalla.

Si el departamento no existe debemos generar una excepci□n
con dicho mensaje
Si el departamento existe, pero no hay empleados dentro,
tambi□n debemos generar una excepci□n para indicarlo */


CREATE OR REPLACE FUNCTION salarios_dept (
    dep NUMBER
) RETURN NUMBER IS
    salary    NUMBER;
    depart    departments.department_id%TYPE;
    num_emple NUMBER;
BEGIN
    SELECT
        department_id
    INTO depart
    FROM
        departments
    WHERE
        department_id = dep;

    SELECT
        COUNT(*)
```

```
    INTO num_emple
    FROM
        employees
    WHERE
        department_id = dep;

    IF dep > 0 THEN
        SELECT
            SUM(salary)
        INTO salary
        FROM
            employees
        WHERE
            department_id = dep
        GROUP BY
            department_id;

    ELSE
        raise_application_error(-20730, 'El departamento existe, pero no hay empleados ' || dep);
    END IF;

    RETURN salary;
EXCEPTION
    WHEN no_data_found THEN
        raise_application_error(-20730, 'No existe el departamento ' || dep);
END;




SET SERVEROUTPUT ON
DECLARE
    sal  NUMBER;
    dept NUMBER := 20;
BEGIN
    sal := salarios_dept(dept);
    dbms_output.put_line('El salario total del departamento '
                         || dept
                         || ' es: '
                         || sal);
END;
```

- Paquetes

```
/* PAQUETES */
-- Crear un encabezado de paquete

CREATE OR REPLACE PACKAGE PACK1
IS
  V1 NUMBER:=100;
  V2 VARCHAR2(100);
END;
/


BEGIN
  PACK1.V1:=PACK1.V1+10;
  DBMS_OUTPUT.PUT_LINE(PACK1.V1);
END;
```

```
/*CREAR BODY PAQUETE*/



CREATE OR REPLACE PACKAGE PACK1
IS
  PROCEDURE CONVERTIR_CADENA (NAME VARCHAR2, tipo_conversion CHAR);
END;
/


CREATE OR REPLACE PACKAGE BODY PACK1
IS
FUNCTION UP(NAME VARCHAR2)
RETURN VARCHAR2
IS
BEGIN
    RETURN UPPER(NAME);
END UP;

FUNCTION DO(NAME VARCHAR2)
RETURN VARCHAR2
IS
BEGIN
    RETURN LOWER(NAME);
END DO;

PROCEDURE CONVERTIR_CADENA (NAME VARCHAR2, tipo_conversion CHAR)
 IS
 BEGIN
    IF tipo_conversion='MAY' THEN
       DBMS_OUTPUT.PUT_LINE(UP(NAME));
    ELSIF tipo_conversion='MIN' THEN
       DBMS_OUTPUT.PUT_LINE(DO(NAME));
    ELSE
       DBMS_OUTPUT.PUT_LINE('EL PARAMETRO DEBE SER MAY o MIN');
   END IF;
END CONVERTIR_CADENA;

END PACK1;



SET SERVEROUTPUT ON
BEGIN
 PACK1.CONVERTIR_CADENA('AAAAA','MIN');

END;


/*Agregar Funcion**/


CREATE OR REPLACE PACKAGE PACK1
IS
  PROCEDURE CONVERTIR_CADENA (NAME VARCHAR2, tipo_conversion CHAR);
  FUNCTION FN_CONVERTIR (NAME VARCHAR2, tipo_conversion CHAR) RETURN VARCHAR2 ;
END;
/
```

```
CREATE OR REPLACE PACKAGE BODY PACK1
IS
FUNCTION UP(NAME VARCHAR2) -- esta solo se puede usar en el cuerpo del paquete
RETURN VARCHAR2
IS
BEGIN
    RETURN UPPER(NAME);
END UP;

FUNCTION DO(NAME VARCHAR2) -- es/ta solo se puede usar en el cuerpo del paquete
RETURN VARCHAR2
IS
BEGIN
    RETURN LOWER(NAME);
END DO;
--paquete
PROCEDURE CONVERTIR_CADENA (NAME VARCHAR2, tipo_conversion CHAR)
 IS
 BEGIN
    IF tipo_conversion='MAY' THEN
       DBMS_OUTPUT.PUT_LINE(UP(NAME));
    ELSIF tipo_conversion='MIN' THEN
       DBMS_OUTPUT.PUT_LINE(DO(NAME));
    ELSE
       DBMS_OUTPUT.PUT_LINE('EL PARAMETRO DEBE SER MAY o MIN');
   END IF;
END CONVERTIR_CADENA;
-- funcion
FUNCTION FN_CONVERTIR (NAME VARCHAR2, tipo_conversion CHAR) return VARCHAR2
 IS
 BEGIN
    IF tipo_conversion='MAY' THEN
       return(UP(NAME));
    ELSIF tipo_conversion='MIN' THEN
       return(DO(NAME));
    ELSE
       DBMS_OUTPUT.PUT_LINE('EL PARAMETRO DEBE SER MAY o MIN');
   END IF;
END FN_CONVERTIR;


END PACK1;


SET SERVEROUTPUT ON
DECLARE
  V1 VARCHAR2(100);
BEGIN
 V1:=PACK1.FN_CONVERTIR('AAAAA','MIN');
 DBMS_OUTPUT.PUT_LINE(V1);
END;



SELECT
    first_name,PACK1.FN_CONVERTIR(FIRST_NAME,'MIN'),PACK1.FN_CONVERTIR(LAST_NAME,'MIN')
FROM
    employees;
```

```
/* Sobrecarga */

CREATE OR REPLACE
PACKAGE PACK2 AS


  FUNCTION COUNT_EMPLOYEES(ID NUMBER) RETURN NUMBER;
  FUNCTION COUNT_EMPLOYEES(ID VARCHAR2) RETURN NUMBER;
END PACK2;
/
CREATE OR REPLACE
PACKAGE BODY PACK2 AS

  FUNCTION COUNT_EMPLOYEES(ID NUMBER) RETURN NUMBER AS
  X NUMBER;
  BEGIN
    SELECT COUNT(*) INTO X FROM EMPLOYEES WHERE DEPARTMENT_ID=ID;
    RETURN X;
  END COUNT_EMPLOYEES;

  FUNCTION COUNT_EMPLOYEES(ID VARCHAR2) RETURN NUMBER AS
  X NUMBER;
  BEGIN
    SELECT COUNT(*) INTO X FROM EMPLOYEES A, DEPARTMENTS B
         WHERE DEPARTMENT_NAME=ID
         AND A.DEPARTMENT_ID=B.DEPARTMENT_ID;

    RETURN X;
  END COUNT_EMPLOYEES;

END PACK2;



BEGIN
  DBMS_OUTPUT.PUT_LINE(PACK2.COUNT_EMPLOYEES('Marketing'));
END;
```