TAREA 6

Adur Marques

BASES DE DATOS 24/04/2022

Ejercicio 1

```
CREATE OR REPLACE FUNCTION generarUsuario(numero NUMBER, longitud NUMBER, relleno CHAR, cadena VARCHAR2)
RETURN VARCHAR2 IS
    longitudNumero INTEGER;
    cadenaFinal VARCHAR2(20);
    i NUMBER;
BEGIN
    IF (longitud>15) THEN
     RAISE_APPLICATION_ERROR(-20011, 'La longitud introducida ('||longitud||') excede la longitud máxima que es 15');
    END IF;

    longitudNumero :=longitud - LENGTH(numero) - LENGTH(cadena);
    cadenaFinal := numero;

    FOR i IN 1..longitudNumero LOOP
        cadenaFinal := cadenaFinal || relleno;
    END LOOP;

    cadenaFinal := cadenaFinal || upper(cadena);

    RETURN cadenaFinal;
END;
```

```
/*
Resultado esperado:
12345#ID
*/
BEGIN
DBMS_OUTPUT.put_line( A: generarUsuario ( numero: 12345, longitud: 10, relleno: '#', cadena: 'id'));
END;

/*
Resultado esperado:
ORA-20011: El ancho introducido (20) excede la longitud máxima que es 15
*/
BEGIN
DBMS_OUTPUT.put_line( A: generarUsuario ( numero: 12345, longitud: 20, relleno: '#', cadena: 'id'));
END;
```

Ejercicio 2

```
CREATE OR REPLACE PROCEDURE actualizaAgente(idCat number) IS
    CURSOR cAgentes IS
        SELECT identificador, nombre, usuario,habilidad, categoria
        FROM agentes where categoria=idCat;
        ag_rec cAgentes%ROWTYPE;
    BEGIN
        OPEN cAgentes;
            FETCH cAgentes INTO ag_rec ;

            IF (cAgentes % FOUND = FALSE) THEN
                RAISE_APPLICATION_ERROR(-20012,'No Hay agentes con la categoria ' || idCat );
            END IF;

            WHILE cAgentes % FOUND LOOP
                DBMS_OUTPUT.put_line(
                    A: 'El agente ' || ag_rec.nombre || ' ha cambiado a usuario ' ||
                    generarUsuario( NUMERO: ag_rec.IdENTIFICADOR, LONGITUD: 12, RELLENO: '-', CADENA: 'H'|| ag_rec.habilidad)
                );

                UPDATE agentes set USUARIO = generarUsuario( NUMERO: ag_rec.IdENTIFICADOR, LONGITUD: 12, RELLENO: '-', CADENA: 'H' || ag_rec.habilidad)
                WHERE identificador = ag_rec.IDENTIFICADOR;

                FETCH cAgentes INTO ag_rec ;
            END LOOP;

            DBMS_OUTPUT.put_line( A: 'Se han actualizado '|| cagentes%ROWCOUNT || ' Agentes');
    CLOSE cAgentes;
END;
```

```
/*
 Resultado esperado:
 -- El Agente Diosdado Sánchez Hernández ha cambiado a usuario 211-----H8
 -- El Agente JesÃ°s Baños Sancho ha cambiado a usuario 111-----H8
 -- El Agente Salvador Romero Villegas ha cambiado a usuario 1111----H7
 -- El Agente JosÃ® Javier Bermádez Hernández ha cambiado a usuario 1112----H7
 -- El Agente Alfonso Bonillo Sierra ha cambiado a usuario 1113----H7
 -- El Agente Silvia Thomas BarrÃ³s ha cambiado a usuario 1121----H7
 -- Se han actualizado ó Agentes
*/
BEGIN
    actualizaAgente( idCat: 1);
END;

/*
 Resultado esperado:
 -- ORA-20012: No Hay agentes con la categoría 3
*/
BEGIN
 actualizaAgente( idCat: 3);
END;
```

Ejercicio 3

```
CREATE OR REPLACE PROCEDURE copiarFamilia(idOrigen NUMBER, idDestino NUMBER) IS
    familiaOrigen Familias % ROWTYPE;
    familiaDestino Familias % ROWTYPE;
    TYPE cursorFamilias IS
        REF CURSOR RETURN familias % ROWTYPE;
    cFamilias cursorFamilias;
    nuevoNombre VARCHAR2(100);

    -- Función auxiliar
    FUNCTION fu_nombre(padre number, destino number)
    RETURN VARCHAR2 IS
        nombrePadre varchar2(100);
    BEGIN
        SELECT nombre INTO nombrePadre FROM FAMILIAS WHERE IDENTIFICADOR = padre;
        RETURN nombrePadre ||'-'|| destino;
    END;

    BEGIN
        -- Comprobamos si la familia destino no existe
        OPEN cFamilias FOR
            SELECT * FROM familias WHERE identificador = idDestino;
        FETCH cFamilias INTO familiaDestino;

        IF (cFamilias % FOUND = TRUE) THEN
            RAISE_APPLICATION_ERROR(-20013, 'La familia destino existe');
        END IF;

        -- Comprobamos si la familia origen existe
        OPEN cFamilias FOR
            SELECT * FROM familias WHERE identificador = idOrigen;
        FETCH cFamilias INTO familiaOrigen;

        IF (cFamilias % FOUND = FALSE) THEN
            RAISE_APPLICATION_ERROR(-20011, 'La familia origen no existe');
        END IF;
```

```
        nuevoNombre:=fu_nombre( padre: familiaOrigen.familia, destino: idDestino);

        INSERT INTO familias (identificador, nombre, familia, oficina)
        VALUES (idDestino, nuevoNombre, familiaOrigen.familia, familiaOrigen.oficina);

        COMMIT;
        DBMS_OUTPUT.put_line('Se ha copiado con exito la familia ['|| idOrigen || '] a la familia ['|| idDestino || '] ');
    END;
```

```
/*
ERROR: 'La familia origen no existe'

-- Resultado esperado:
-- ORA-20011: La familia origen no existe
*/
BEGIN
    copiarFamilia( idOrigen: 33, idDestino: 900);
END;

/*
ERROR: 'La familia destino existe'

-- Resultado esperado:
-- ORA-20013: La familia destino existe
*/
BEGIN
    copiarFamilia( idOrigen: 111, idDestino: 112);
END;

/*
Comprobar que funciona

-- Resultado esperado:
-- Se ha copiado con exito la familia [111] a la familia [901]
-- Madrid-1.2-901
*/
DECLARE
    familiaOrigen number:= 111;
    familiaDestino number:= 901;
    nuevoNombre varchar(100);
BEGIN
    copiarFamilia( idOrigen: familiaOrigen, idDestino: familiaDestino);
    SELECT nombre INTO nuevoNombre FROM familias WHERE identificador = familiaDestino;
    DBMS_OUTPUT.put_line(nuevoNombre);
END;
```

Ejercicio 4

```
CREATE OR REPLACE TRIGGER numeroAgentes
BEFORE INSERT OR DELETE ON agentes
FOR EACH ROW
DECLARE
 TYPE cursorOficinas IS REF CURSOR RETURN oficinas % ROWTYPE;
    cOficinas cursorOficinas;
 TYPE cursorFamilias IS REF CURSOR RETURN familias % ROWTYPE;
    cFamilias cursorFamilias;
 rtFamilias Familias % ROWTYPE;
 rtOficinas Oficinas % ROWTYPE;
BEGIN
    IF inserting THEN
        --1. La familias y la oficina no pueden ser las dos nulas
        IF (:new.familia IS NULL AND :new.oficina IS NULL) THEN
            raise_application_error(-20021, 'No pueden ser nulas a la vez la familia y la oficina');
        END IF;

        --2. La oficina y la categoria no pueden tener valor las dos
        IF (:new.familia IS NOT NULL AND :new.oficina IS NOT NULL) THEN
            raise_application_error(-20022, 'No pueden tener valor la familia y la oficina a la vez. Debes elegir una');
        END IF;

        --3. Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina.
        IF (:new.categoria = 2 AND :new.oficina IS NULL) THEN
            RAISE_APPLICATION_ERROR(-20023, 'Si el agente pertenece a la categoria 2, debe tener una oficina asociada.');
        END IF;

        --4. Si un agente tiene categoría 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia.
        IF (:new.categoria = 1 AND :new.familia IS NULL) THEN
            RAISE_APPLICATION_ERROR(-20024, 'Si el agente pertenece a la categoria 1, debe tener una familia asociada.');
        END IF;

        --5. Al insertar un agente en la tabla AGENTES se actualizará, incrementándose en 1,
        UPDATE familias SET numagentes = numagentes+1 WHERE identificador=: new.familia OR oficina=: new.oficina;
    END IF;
```

```
    IF deleting THEN
        --6. Antes de eliminar un agente en la tabla AGENTES se actualizará¡, decrementándose en 1, el campo NumAgentes de la tabla FAMILIAS
        UPDATE familias SET numagentes = numagentes- 1 WHERE identificador=: old.familia or oficina=: old.oficina;
    END IF;
END;

--1. La familias y la oficina no pueden ser las dos nulas
INSERT INTO
 AGENTES (identificador, Nombre, Usuario, Clave, Habilidad, Categoria, Familia, oficina)
VALUES (316, 'Federico', 'Fede11', 'Fed11', 4, 0, null, null);

--2. La oficina y la categoria no pueden tener valor las dos
INSERT INTO
 AGENTES (identificador, Nombre, Usuario, Clave, Habilidad, Categoria, Familia, oficina)
VALUES (316, 'Federico', 'Fede11', 'Fed11', 4, 0, 11, 1);

--3. Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina.
INSERT INTO AGENTES (identificador, Nombre, Usuario, Clave, Habilidad, Categoria, Familia, oficina)
VALUES (316, 'Federico', 'Fede11', 'Fed11', 4, 2, 4, null);

--4. Si un agente tiene categoria 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia.
INSERT INTO AGENTES (identificador, Nombre, Usuario, Clave, Habilidad, Categoria, Familia, oficina)
    VALUES (316, 'Federico', 'Fede11', 'Fed11', 4, 1, null, 4); --Insertar Agente de categoría 2

SELECT * FROM FAMILIAS WHERE oficina = 2; -- comprobamos qué valor tiene el campo numAgentes antes de insertar
INSERT INTO AGENTES (identificador, Nombre, Usuario, Clave, Habilidad, Categoria, Familia, oficina)
    VALUES (317, 'Elisa', 'Elis2', 'El2', 4, 2, null, 2);

SELECT * FROM AGENTES WHERE identificador = 317; -- comprobamos que el agente está insertado correctamente
SELECt * FROM FAMILIAS WHERE oficina = 2; --comprobamos que el valor del campo numAgentes se ha incrementado

--Insertar Agente de categoría 1
SELECt * FROM FAMILIAS WHERE identificador = 11; -- comprobamos quÃ© valor tiene el campo numAgentes antes de insertar

INSERT INTO AGENTES (identificador, Nombre, Usuario, Clave, Habilidad, Categoria, Familia, oficina)
 VALUES (316, 'Federico', 'Fede11', 'Fed11', 4, 1, 11, null);
```

```sql
SELECT * FROM AGENTES WHERE identificador = 316; -- comprobamos que el agente estÃ¡ insertado correctamente

SELECT * FROM familias WHERE identificador = 11; --comprobamos que el valor del campo numAgentes se ha incrementado

SELECT * FROM familias WHERE identificador = 11; -- comprobamos quÃ© valor tiene el campo numAgentes antes de eliminar

DELETE agentes WHERE identificador = 316; --Eliminar Agente 316

SELECT * FROM agentes WHERE identificador = 316;

SELECT * FROM familias WHERE identificador = 11; --comprobamos que el valor del campo numAgentes se ha decrementado
```