

Tarea BD06

Actividad 1

La creación del procedimiento:

```
CREATE OR REPLACE PROCEDURE CambiarAgentesFamilia(id_FamiliaOrigen NUMBER,
id_FamiliaDestino NUMBER) AS

    v_familia_origen VARCHAR(255);
    v_familia_destino VARCHAR(255);
    v_numero_agentes NUMBER := 0;
    v_NumAgentes NUMBER := 0;
    v_hay_errores BOOLEAN := FALSE;
    v_FamiliaOrigenCount NUMBER := 0;
    v_FamiliaDestinoCount NUMBER := 0;

BEGIN

    --comprobamos que las familias existen

    SELECT COUNT(*) INTO v_FamiliaOrigenCount FROM familias WHERE identificador =
id_FamiliaOrigen;

    SELECT COUNT(*) INTO v_FamiliaDestinoCount FROM familias WHERE identificador =
id_FamiliaDestino;

    -- Comprobar la existencia de la familia de origen

    IF v_FamiliaOrigenCount = 0 THEN

        RAISE_APPLICATION_ERROR(-20001, 'La familia de origen especificada no existe.');
```

END IF;

-- Comprobar la existencia de la familia de destino

IF v_FamiliaDestinoCount = 0 THEN

RAISE_APPLICATION_ERROR(-20002, 'La familia de destino especificada no existe.');

END IF;

--y no son iguales

IF id_FamiliaOrigen = id_FamiliaDestino THEN

v_hay_errores := TRUE;

RAISE_APPLICATION_ERROR(-20001, 'Las familias no pueden ser iguales');

END IF;

```

--buscamos nombres de las familias ya que existen

SELECT nombre INTO v_familia_origen FROM familias WHERE identificador =
id_FamiliaOrigen;

SELECT nombre INTO v_familia_destino FROM familias WHERE identificador =
id_FamiliaDestino;

-- si no tengo errores Cambio de los agentes de la familia origen a la familia destino

IF v_hay_errores = FALSE THEN

    UPDATE agentes SET familia = id_FamiliaDestino WHERE familia = id_FamiliaOrigen;

    v_NumAgentes := SQL%ROWCOUNT;

    DBMS_OUTPUT.PUT_LINE('Se han trasladado ' || v_NumAgentes || ' agentes de la familia
' || v_familia_origen || ' a la familia ' || v_familia_destino);

END IF;

END;

/

```

Ejecución del procedimiento:

```

DECLARE

BEGIN

    CambiarAgentesFamilia(11, 111);

EXCEPTION

    WHEN OTHERS THEN

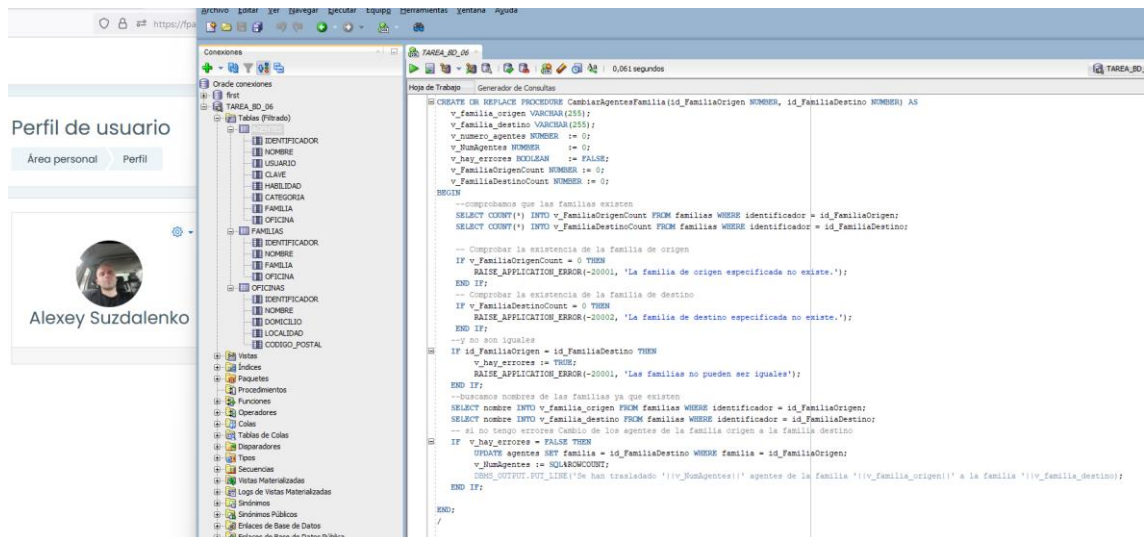
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLCODE || ' - ' || SQLERRM);

END;

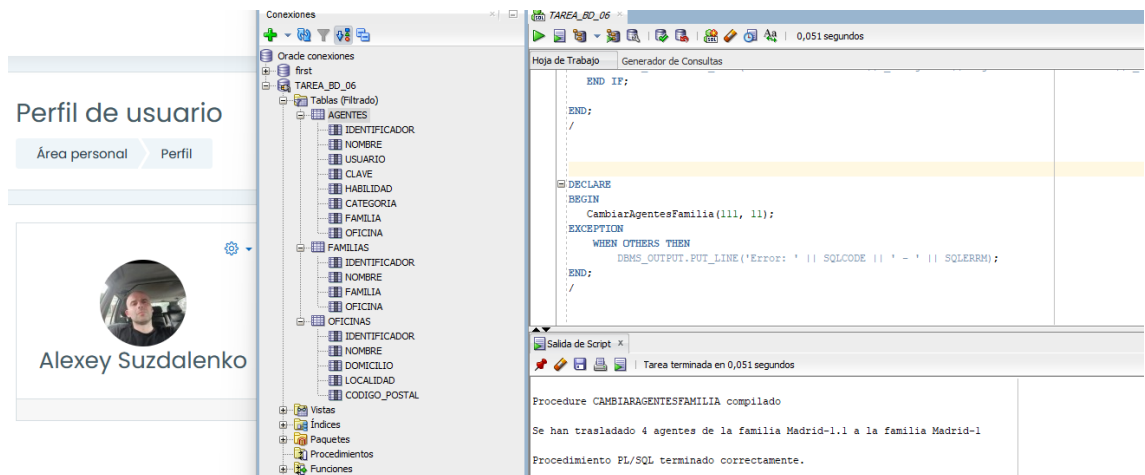
/

```

Captura procedimiento:



Captura ejecución:



Actividad 2.

De las restricciones pedidas se implementan en el esquema a la hora de crear tablas agentes:

La longitud de la clave de un agente no puede ser inferior a 6.

La habilidad de un agente debe estar comprendida entre 0 y 9 (ambos inclusive).

La categoría de un agente sólo puede ser igual a 0, 1 o 2.

CREATE TABLE agentes (

identificador NUMBER(6) NOT NULL PRIMARY KEY,

nombre VARCHAR2(60) NOT NULL,

usuario VARCHAR2(20) NOT NULL UNIQUE,

clave VARCHAR2(20) NOT NULL CHECK (LENGTH(clave) >= 6),

```

    habilidad    NUMBER(1) NOT NULL CHECK (habilidad BETWEEN 0 AND 9),
    categoria    NUMBER(1) NOT NULL CHECK (categoria IN (0, 1, 2)),
    familia      NUMBER(6) REFERENCES familias,
    oficina      NUMBER(6) REFERENCES oficinas
);

```

Y el en trigger se implementan todas las restricciones:

```

CREATE OR REPLACE TRIGGER restricciones_agentes
BEFORE INSERT OR UPDATE ON agentes
FOR EACH ROW
DECLARE
    v_oficina_exist NUMBER := 0;
    v_familia_exist NUMBER := 0;
BEGIN
    -- Verificar longitud de la clave
    IF LENGTH(:NEW.clave) < 6 THEN
        RAISE_APPLICATION_ERROR(-20005, 'La longitud de la clave debe ser al menos 6
caracteres.');
```

END IF;

```

    -- Verificar habilidad
    IF :NEW.habilidad NOT BETWEEN 0 AND 9 THEN
        RAISE_APPLICATION_ERROR(-20006, 'La habilidad del agente debe estar entre 0 y 9.');
```

END IF;

```

    -- Verificar categoría
    IF :NEW.categoria NOT IN (0, 1, 2) THEN
        RAISE_APPLICATION_ERROR(-20007, 'La categoría del agente solo puede ser 0, 1 o 2.');
```

END IF;

```

    -- Verificar categoría y pertenencia

```

```

IF :NEW.categoria = 2 THEN

    IF :NEW.familia IS NOT NULL THEN

        RAISE_APPLICATION_ERROR(-20008, 'Un agente de categoría 2 no puede pertenecer a
ninguna familia.');
```

END IF;

```

    IF :NEW.oficina IS NULL THEN

        RAISE_APPLICATION_ERROR(-20009, 'Un agente de categoría 2 debe pertenecer a una
oficina.');
```

END IF;

```

ELSIF :NEW.categoria = 1 THEN

    IF :NEW.oficina IS NOT NULL THEN

        RAISE_APPLICATION_ERROR(-20010, 'Un agente de categoría 1 no puede pertenecer a
ninguna oficina.');
```

END IF;

```

    IF :NEW.familia IS NULL THEN

        RAISE_APPLICATION_ERROR(-20011, 'Un agente de categoría 1 debe pertenecer a una
familia.');
```

END IF;

```

END IF;

-- Verificar pertenencia a oficina o familia pero no a ambas

SELECT COUNT(*) INTO v_oficina_exist FROM oficinas WHERE identificador = :NEW.oficina;

SELECT COUNT(*) INTO v_familia_exist FROM familias WHERE identificador = :NEW.familia;

IF v_oficina_exist = 1 AND v_familia_exist = 1 THEN

    RAISE_APPLICATION_ERROR(-20012, 'Un agente no puede pertenecer a una oficina y a una
familia al mismo tiempo.');
```

END IF;

```


END;

/
```

Captura de creación trigger:

Perfil de usuario

Área personal Perfil



Alexey Suzdalenko

Oracle conexiones

TAREA_ID_06

Tablas (Filtrado)

AGENTES

IDENTIFICADOR

NOMBRE

USUARIO

CLAVE

HABILIDAD

CATEGORIA

FAMILIA

OFICINA

FAMILIAS

IDENTIFICADOR

NOMBRE

FAMILIA

OFICINAS

IDENTIFICADOR

NOMBRE

DOMICILIO

LOCALIDAD

CODIGO_POSTAL

Views

Indice

Paquetes

Procedimientos

Funciones

Operadores

Colas

Tablas de Colas

Disparadores

RESTRICCIONES_AGENTES

Tips

Secuencias

Views Materializadas

Log de Views Materializadas

Sinónimos

Sinónimos Públicos

Enlaces de Base de Datos

Enlaces de Base de Datos Públicos

Directorios

Ediciones

Application Express

Esquemas XML

Reportorio de Base de Datos XML

Programador

Property Graph

Propietario de Reciclos

Otros Usuarios

TAREA_ID_06

0,045 segundos

Hoja de Trabajo

Generador de Consultas

CREATE OR REPLACE TRIGGER RESTRICCIONES_AGENTES

BEFORE INSERT OR UPDATE ON Agentes

FOR EACH ROW

DECLARE

v_oficina_exist NUMBER := 0;

v_familia_exist NUMBER := 0;

BEGIN

-- Verificar longitud de la clave

IF LENGTH(:NEW.clave) < 6 THEN

RAISE_APPLICATION_ERROR(-20009, 'La longitud de la clave debe ser al menos 6 caracteres.');

END IF;

-- Verificar habilidad

IF :NEW.habilidad NOT BETWEEN 0 AND 9 THEN

RAISE_APPLICATION_ERROR(-20006, 'La habilidad del agente debe estar entre 0 y 9.');

END IF;

-- Verificar categoria

IF :NEW.categoria NOT IN (0, 1, 2) THEN

RAISE_APPLICATION_ERROR(-20007, 'La categoria del agente solo puede ser 0, 1 o 2.');

END IF;

-- Verificar categoria y pertenencia

IF :NEW.categoria = 2 THEN

IF :NEW.familia IS NOT NULL THEN

RAISE_APPLICATION_ERROR(-20008, 'Un agente de categoria 2 no puede pertenecer a ninguna familia.');

END IF;

IF :NEW.oficina IS NULL THEN

RAISE_APPLICATION_ERROR(-20009, 'Un agente de categoria 2 debe pertenecer a una oficina.');

END IF;

ELSEIF :NEW.categoria = 1 THEN

IF :NEW.oficina IS NOT NULL THEN

RAISE_APPLICATION_ERROR(-20010, 'Un agente de categoria 1 no puede pertenecer a ninguna oficina.');

END IF;

IF :NEW.familia IS NULL THEN

RAISE_APPLICATION_ERROR(-20011, 'Un agente de categoria 1 debe pertenecer a una familia.');

END IF;

END IF;

-- Verificar pertenencia a oficina o familia pero no a ambas

SELECT COUNT(*) INTO v_oficina_exist FROM oficinas WHERE identificador = :NEW.oficina;

SELECT COUNT(*) INTO v_familia_exist FROM familias WHERE identificador = :NEW.familia;

IF v_oficina_exist = 1 AND v_familia_exist = 1 THEN

RAISE_APPLICATION_ERROR(-20012, 'Un agente no puede pertenecer a una oficina y a una familia al mismo tiempo.');

END IF;

END IF;

END;

Salida de Script

Tarea terminada en 0,045 segundos

Trigger RESTRICCIONES_AGENTES compilado