Para la realización de los ejercicios de esta unidad nos basaremos en el caso de estudio expuesto en los contenidos de la misma en el Anexo I. La tarea que te pedimos que realices consta de 2 actividades:

Ejercicio 1

Crear un procedimiento que permita cambiar a todos los agentes de una familia determinada (familia origen) a otra familia (familia destino).

El procedimiento tendrá la siguiente cabecera CambiarAgentesFamilia (id_FamiliaOrigen, id_FamiliaDestino), donde cada uno de los argumentos corresponde a un identificador de Familia. Cambiará la columna Identificador de Familia de todos los agentes, de la tabla AGENTES, que pertenecen a la Familia con código id_FamiliaOrigen por el código id_FamiliaDestino

Previamente comprobará que ambas familias existen y que no son iguales.

Para la comprobación de la existencia de las familias se puede utilizar un cursor variable, o contar el número de filas y en caso de que no exista, se visualizará el mensaje correspondiente mediante una excepción del tipo RAISE_APPLICATION_ERROR. También se mostrará un mensaje en caso de que ambos argumentos tengan el mismo valor.

El procedimiento visualizará el mensaje "Se han trasladado XXX agentes de la familia XXXXXX a la familia ZZZZZZ" donde XXX es el número de agentes que se han cambiado de familia, XXXXXX es el nombre de la familia origen y ZZZZZZ es el nombre de la familia destino.

SOLUCION

```
CREATE OR REPLACE PROCEDURE mover_familia(id_origen NUMBER, id_destino NUMBER) IS
    familia_origen familias%ROWTYPE;
    familia_destino familias%ROWTYPE;
    TYPE cursor_familias IS REF CURSOR RETURN familias%ROWTYPE;
    cFamilias cursor_familias;

    --Función auxiliar que nos devuelve 0 si una familia no es hija de otra y 1 en caso contrario
    FUNCTION es_hija(origen familias%ROWTYPE, destino familias%ROWTYPE) RETURN NUMBER IS
        madre familias%ROWTYPE;
    BEGIN
        IF (destino.familia IS NULL) THEN
            RETURN 0;
        ELSIF (destino.familia = origen.identificador) THEN
```

```
            RETURN 1;
        ELSE
            SELECT * INTO madre FROM familias WHERE identificador = destino.familia;
            RETURN es_hija(origen, madre);
        END IF;
    END;
BEGIN
    --Comprobamos si la familia origen existe y si existe la guardamos en familia_origen
    OPEN cFamilias FOR SELECT * FROM familias WHERE identificador = id_origen;
    FETCH cFamilias INTO familia_origen;
    IF (cFamilias%FOUND = FALSE) THEN
        RAISE_APPLICATION_ERROR(-20011, 'La familia origen no existe');
    END IF;

    --Comprobamos si la familia destino existe y si existe la guardamos en familia_destino
    OPEN cFamilias FOR SELECT * FROM familias WHERE identificador = id_destino;
    FETCH cFamilias INTO familia_destino;
    IF (cFamilias%FOUND = FALSE) THEN
        RAISE_APPLICATION_ERROR(-20012, 'La familia destino no existe');
    END IF;

    --Comprobamos si la familia destino es hija de la familia origen
    IF (es_hija(familia_origen, familia_destino) = 1) THEN
        RAISE_APPLICATION_ERROR(-20013, 'La familia destino es hija (directa o indirecta) de la f
    END IF;

    --Actualizamos la familia al identificador de la familia destino
    --y ponemos la oficina a NULL por si acaso era la familia raíz de la oficina
    UPDATE familias SET familia = id_destino, oficina = NULL WHERE identificador = id_origen;
     COMMIT;
```

Ejercicio 2.

Queremos controlar algunas restricciones a la hora de trabajar con 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.
- ✓ Si un agente tiene categoría 2 no puede pertenecer a ninguna familia y debe pertenecer a una oficina.
- ✓ Si un agente tiene categoría 1 no puede pertenecer a ninguna oficina y debe pertenecer a una familia.
- ✓ Todos los agentes deben pertenecer a una oficina o a una familia pero nunca a ambas a la vez.

Se pide crear un disparador para asegurar estas restricciones. El disparador deberá lanzar todos los errores que se puedan producir en su ejecución mediante errores que identifiquen con un mensaje adecuado por qué se ha producido dicho error.

Algunas de las restricciones implementadas con el disparador se pueden incorporar a la definición del esquema de la tabla utilizando el Lenguaje de Definición de Datos (Check, Unique,..).Identifica cuáles son y con qué tipo de restricciones las implementarías.

SOLUCION

```
CREATE OR REPLACE TRIGGER integridad_agentes
BEFORE INSERT OR UPDATE ON agentes
FOR EACH ROW
BEGIN
   --Comprobamos que el usuario y la clave no son iguales
   IF (:new.usuario = :new.clave) THEN
      RAISE_APPLICATION_ERROR(-20021, 'El usuario y la clave deben ser diferentes');
   END IF;

   --Comprobamos que la habilidad del agente está comprendida entre 0 y 9
   IF (:new.habilidad < 0 OR :new.habilidad > 9) THEN
      RAISE_APPLICATION_ERROR(-20022, 'La habilidad del agente es errónea');
   END IF;
```

```
    --Comprobamos que la categoria del agente está comprendida entre 0 y 2
    IF (:new.categoria < 0 OR :new.categoria > 2) THEN
        RAISE_APPLICATION_ERROR(-20023, 'La categoría del agente es errónea');
    END IF;

    --Si un agente pertenece directamente a una oficina su categoria debe ser 2
    IF (:new.oficina IS NOT NULL and :new.categoria != 2) THEN
        RAISE_APPLICATION_ERROR(-20024, 'La categoría de un agente que pertenece a una oficina di
    END IF;

    --Si un agente no pertenece directamente a una oficina su categoria debe ser distinta de 2
    IF (:new.oficina IS NULL and :new.categoria = 2) THEN
        RAISE_APPLICATION_ERROR(-20025, 'La categoría de un agente que no pertenece a una oficina
    END IF;

    --No puede haber agentes huérfanos ni con dos padres (oficina y familia)
    IF (:new.familia IS NULL and :new.oficina IS NULL) THEN
        RAISE_APPLICATION_ERROR(-20026, 'Un agente no puede ser huérfano');
    ELSIF (:new.familia IS NOT NULL and :new.oficina IS NOT NULL) THEN
        RAISE_APPLICATION_ERROR(-20027, 'Un agente no puede tener dos padres');
    END IF;
END;
/
```