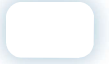


# Solución de la tarea para BD06.



Para la tarea propuesta en esta unidad proponemos una posible solución para la misma, aunque las implementaciones podrían diferir y seguir siendo totalmente válidas.

## ✓ Actividad 1:

```
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 cont
    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) d
    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;
```

END;

/



## 📌 Actividad 2:

```
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 ofic
    END IF;

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

