Autores: Alonso BH, Juan Carlos GQ

Práctica 3. Implementación del diseño

1. Restricciones de llave única

2. Restricciones no nulas que se consideren oportunas

3. El director de una sucursal es empleado de la compañía

```
create or replace TRIGGER DIS_EMPLEADO_DIRECTOR BEFORE INSERT OR UPDATE ON sucursal FOR EACH ROW
DECLARE
numEmpleados NUMBER(2);
BEGIN
IF(INSERTING OR UPDATING('director')) THEN -- Se realizará este proceso si se están insertando los datos
        Lo primero será comprobar si ya hay id's en la tabla (Si existe ya ese pedido)
                IF(:NEW.director IS NOT NULL) THEN
                    SELECT COUNT(*) INTO numEmpleados
                            FROM empleado_v
                            WHERE cod_empleado =:NEW.director;
        IF(numEmpleados = 0) THEN
        RAISE_APPLICATION_ERROR(-20100,'El director debe ser empleado, el que has introducido no existe');
        END IF;
END IF;
END IF;
END;
```

4. Un empleado, al mismo tiempo

```
CREATE OR REPLACE TRIGGER DIS_DIRECTOR_SUCURSAL BEFORE INSERT OR UPDATE ON sucursal FOR EACH ROW
```

Autores: Alonso BH, Juan Carlos GQ

```
DECLARE
        numEmpleados NUMBER(2);
BEGIN
        IF(INSERTING OR UPDATING('director')) THEN -- Se realizará este proceso si se están insertando los datos
                IF(:NEW.director IS NOT NULL) THEN
                        SELECT COUNT(*) INTO numEmpleados
                                FROM empleado
                                WHERE cod_empleado =:NEW.director;
                IF(numEmpleados = 0) THEN   -- Si >0, hay ya director con ese número
                        RAISE_APPLICATION_ERROR(-20001,'El empleado no existe');
                END IF;
        END IF;
        END IF;
END;
```

5. Un empleado, al mismo tiempo, solamente puede trabajar en una sucursal

6. El salario de un empleado nunca puede disminuir

```
create or replace TRIGGER DIS_SALARIO_DISMINUIR BEFORE UPDATE ON Empleado FOR EACH ROW
DECLARE
PRAGMA AUTONOMOUS_TRANSACTION;
salario_viejo     number(8,3);
BEGIN
IF UPDATING('salario') THEN -- Si va a modificar el salario
SELECT salario INTO salario_viejo FROM empleado_v
WHERE cod_empleado = :NEW.cod_empleado;   -- seleccionar el salario antiguo de esta fila

IF (salario_viejo > :NEW.salario ) THEN --Si lo que se modifica es menor
```

```
        RAISE_APPLICATION_ERROR(-20017,'El salario tiene que ser superior a  '|| :OLD.salario || ' euros (el salario no puede disminuir
        NUNCA).');
END IF;
END IF;
END;
```

7. La sucursal

```
create or replace TRIGGER DIS_SUCURSAL_EXISTE BEFORE UPDATE OR INSERT ON empleado FOR EACH ROW
DECLARE
contador NUMBER(1);
BEGIN
IF (UPDATING('cod_sucursal') OR INSERTING) THEN
SELECT COUNT(cod_sucursal) INTO contador FROM sucursal_v
WHERE cod_sucursal = :NEW.cod_sucursal;

IF(contador=0) THEN
        RAISE_APPLICATION_ERROR(-20017,'La sucursal del empleado debe existir.');
END IF;
END IF;
END;
```

8. Los clientes solamente pueden ser de tipo

```
create or replace TRIGGER DIS_TIPO_CLIENTE BEFORE INSERT OR UPDATE ON Clientes FOR EACH ROW
DECLARE
BEGIN
IF (UPDATING ('tipo_cliente') OR INSERTING) THEN
IF (:NEW.tipo_cliente NOT IN('A','B','C')) THEN
        Si el tipo es diferente a A, B o C.
```

```
        RAISE_APPLICATION_ERROR(-20019,'El tipo de cliente debe ser A, B o C.');
END IF;
END IF;
END;
```

9. Un cliente puede solicitar

```
create or replace TRIGGER DIS_CLIENTE_PIDE_SUCURSAL
BEFORE INSERT OR UPDATE ON CLIENTE_SOLICITA_SUMINISTRO FOR EACH ROW
DECLARE
contador NUMBER(8);
es_correcta NUMBER (1); -- Para almacenar si la sucursal es correcta.
Cod_Sucursal sucursal.cod_sucursal%TYPE; -- Para almacenar la sucursal.
CA_sucursal sucursal.comunidad_autonoma%TYPE; --Almacenamos la ca de la sucursal
CA_cliente clientes.comunidad_autonoma%TYPE;   --Almacenamos la ca del cliente
delegacion VARCHAR2(50); -- Para almacenar la delegación del cliente.
CURSOR cursor_sucursal IS SELECT cod_sucursal, comunidad_autonoma FROM sucursal;    -- Necesitaremos un cursor para las sucursales.

BEGIN
es_correcta := 0;
SELECT comunidad_autonoma INTO CA_cliente
FROM clientes_v
WHERE cod_cliente = :NEW.cod_cliente;

GET_DELEGACION(delegacion , CA_cliente);

        EJECUTAMOS EL CURSOR CON LA DELEGACIÓN CORRECTA.

OPEN cursor_sucursal ; -- ABRIMOS EL CURSOR CON LA DELEGACION CORRECTA
FETCH cursor_sucursal INTO Cod_Sucursal , CA_sucursal ; -- LEEMOS EL PRIMER VALOR
```

Autores: Alonso BH, Juan Carlos GQ

```
WHILE cursor_sucursal %FOUND -- MIENTRAS LA ÚLTIMA FILA HAYA SIDO LEÍDA CON ÉXITO
LOOP
IF (delegacion = 'Madrid') THEN
        IF (CA_sucursal  = 'Castilla-León' OR CA_sucursal = 'Castilla-La Mancha' OR CA_sucursal = 'Aragón'
        OR CA_sucursal = 'Madrid' OR CA_sucursal = 'La Rioja') THEN
                IF (Cod_Sucursal = :NEW.cod_sucursal) THEN -- Si el cod_sucursal es válido
                es_correcta:= 1;    --   Asignamos a nuestro booleano el valor de true
                END IF;
        END IF;
END IF;

IF (delegacion = 'Barcelona')THEN
        IF (CA_sucursal = 'Cataluña' OR CA_sucursal= 'Baleares' OR CA_sucursal= 'País Valenciano'
        OR CA_sucursal= 'Murcia') THEN
        IF (Cod_Sucursal = :NEW.cod_sucursal) THEN -- SI LA SUCURSAL INSERTADA
                es_correcta:= 1;        -- COINCIDE CON UNA POSIBLE DEL CLIENTE
        END IF;
        END IF;
END IF;

IF (delegacion = 'La Coruña')THEN
        IF (CA_sucursal = 'Galicia' OR CA_sucursal= 'Asturias' OR CA_sucursal= 'Cantabria'
        OR CA_sucursal= 'País Vasco' OR CA_sucursal= 'Navarra') THEN

        IF (Cod_Sucursal = :NEW.cod_sucursal) THEN
                es_correcta:= 1;
        END IF;
        END IF;
END IF;
```

```
IF (delegacion = 'Sevilla') THEN
        IF (CA_sucursal = 'Canarias' OR CA_sucursal= 'Andalucía' OR CA_sucursal= 'Extremadura'
        OR CA_sucursal= 'Melilla' OR CA_sucursal= 'Ceuta') THEN

        IF (Cod_Sucursal = :NEW.cod_sucursal) THEN
                es_correcta:= 1;        -- Ponemos a true el contador
        END IF;
        END IF;
END IF;

FETCH cursor_sucursal INTO Cod_Sucursal, CA_sucursal; -- Avanzamos a la siguiente sucursal
END LOOP;
CLOSE cursor_sucursal ; -- CERRAMOS EL CURSOR.

IF (es_correcta = 0) THEN  -- SI LA SUCURSAL NO ES NINGUNA DE LAS POSIBLES.
RAISE_APPLICATION_ERROR(-20020, 'El cliente debe realizar el pedido en la sucursal de su delegación.');
END IF;
END;
```

10. Para cada cliente,

```
create or replace TRIGGER DIS_FECHA_PEDIDO BEFORE INSERT OR UPDATE ON cliente_solicita_suministro FOR EACH ROW
DECLARE
ultima_fecha cliente_solicita_suministro.fecha%TYPE;
BEGIN
IF (INSERTING) THEN -- Comprobamos la fecha del pedido
SELECT MAX(fecha) INTO ultima_fecha
        FROM cliente_solicita_suministro_v
        WHERE cod_cliente = :NEW.cod_cliente;
IF (ultima_fecha > :NEW.fecha) THEN -- Si es menor la fecha
```

```
        RAISE_APPLICATION_ERROR(-20021, 'No se puede crear un nuevo pedido antes de: ' || ultima_fecha);
END IF;
END IF;

        SI ESTAMOS ACTUALIZANDO
IF (UPDATING('fecha'))THEN
IF (:OLD.fecha > :NEW.fecha) THEN
        RAISE_APPLICATION_ERROR(-20022, 'No se puede crear un nuevo pedido antes de ' || ultima_fecha);
END IF;
END IF;
END;
```

No se puede suministrar un vino que no existe

11.a. Suministrar al cliente

```
create or replace TRIGGER DIS_VINSUMCLIEXISTE
BEFORE INSERT OR UPDATE ON cliente_solicita_suministro FOR EACH ROW
DECLARE
contador number(4);
BEGIN
IF ( INSERTING OR UPDATING ('cod_vino') ) THEN
                SELECT COUNT(*) INTO contador
                FROM vinos_v
                WHERE cod_vino = :NEW.cod_vino ;
IF (contador=0) THEN
        raise_application_error (-20050, 'El vino que quieres suministrar al cliente NO existe.');
END IF  ;
END IF;
END;
```

Autores: Alonso BH, Juan Carlos GQ

11.b. Suministro a sucursal

```
create or replace TRIGGER DIS_VINSUMSUCEXISTE BEFORE INSERT OR UPDATE ON sucursal_pedido_sucursal FOR EACH ROW
DECLARE
contador number(4);
BEGIN
IF (INSERTING OR UPDATING ('cod_vino')) THEN
                SELECT COUNT(*) INTO contador
                FROM vinos_v
                WHERE cod_vino = :NEW.cod_vino ;
IF contador=0 THEN
        raise_application_error (-20050, 'El vino que quieres suministrar a la sucursal NO existe.');
END IF;
END IF;
END;
```

12. Un vino solamente puede producirlo un productor.

```
create or replace TRIGGER DIS_VINOUNPROD BEFORE INSERT OR UPDATE ON vinos  FOR EACH ROW
DECLARE
contador number(4);
BEGIN
IF (INSERTING) THEN
                SELECT COUNT(*) INTO contador
                FROM vinos_v
                WHERE cod_vino = :NEW.cod_vino AND cod_productor <> :NEW.cod_productor;

IF (contador > 0) THEN
```

```
raise_application_error (-20056, 'ERROR: ESTE vino ya es PRODUCIDO por un productor, no puede ser producido por 2 productores.');
END IF  ;
END IF;
END;
```

13. Un vino no puede existir si no existe un productor que lo produzca.

```
create or replace TRIGGER DIS_VINO_TIENE_PRODUCTOR BEFORE INSERT OR UPDATE ON vinos FOR EACH ROW
DECLARE
contador number(8);
BEGIN
IF ( INSERTING OR UPDATING ('cod_productor') )  THEN
                SELECT COUNT(*) INTO contador
        FROM productor_v
        WHERE cod_productor = :NEW.cod_productor ;

IF (contador = 0) THEN
        raise_application_error (-20057, 'El productor del vino debe existir. ');
END IF  ;
END IF;
END;
```

14. El stock de un vino nunca puede ser negativo ni mayor que la cantidad producida.

```
create or replace TRIGGER DIS_STOCKMASPROD BEFORE INSERT OR UPDATE ON vinos FOR EACH ROW
BEGIN
IF INSERTING OR UPDATING ('cantidad_en_stock') THEN
                IF (:NEW.cantidad_en_stock < 0 OR :NEW.cantidad_en_stock > :NEW.cantidad_producida )
THEN
```

```
                raise_application_error (-20058, 'El stock de un vino no puede ser negativo ni MAYOR QUE  la cantidad producida del
                mismo.') ;
            END IF;
END IF;
END;
```

15. Los datos referentes a un vino solamente podrán eliminarse de la base de datos si la cantidad total suministrada de ese vino es 0 (o si ese vino no ha sido suministrado nunca).

```
create or replace TRIGGER DIS_BORRAR_VINO BEFORE DELETE ON vinos FOR EACH ROW
DECLARE
   PRAGMA AUTONOMOUS_TRANSACTION;
        num_sumini_cliente   number(4);
   num_sumini_sucursal       number(4);
BEGIN
        SELECT COUNT(1) INTO num_sumini_cliente FROM cliente_solicita_suministro_v
                WHERE cod_vino = :old.cod_vino ;
        SELECT COUNT(1) INTO num_sumini_sucursal FROM sucursal_pedido_sucursal_v
                WHERE cod_vino = :old.cod_vino;

        IF (num_sumini_cliente > 0 OR num_sumini_sucursal > 0)
   THEN  -- el vino ha sido suministrado
                raise_application_error (-20061, 'ERROR: No se puede borrar info de un vino SUMINISTRADO.') ;
        END IF;
END;
```

16. Los datos referentes a un productor solamente podrán eliminarse de la base de datos si para cada vino que produce, la cantidad total suministrada es 0 o no existe ningún suministro.

```
create or replace TRIGGER DIS_BORRAR_PRODUCTOR BEFORE DELETE ON productor  FOR EACH ROW
```

Autores: Alonso BH, Juan Carlos GQ

```
DECLARE
   PRAGMA AUTONOMOUS_TRANSACTION;
         suministros_a_cliente                  number(8);
         suministros_a_sucursal                 number(8);
   cod_vino_recuperado vinos.cod_vino%TYPE ;   -- guardamos cada vino recorrido por el cursor
         CURSOR c1 IS SELECT cod_vino FROM vinos_v
      WHERE cod_productor = :OLD.cod_productor ;

BEGIN
   OPEN c1;
   FETCH c1 INTO cod_vino_recuperado;
   WHILE c1 %FOUND
         --FOR registro IN c1  -- iteramos sobre cada vino producido por el productor cod_prod
         LOOP
                  -- compruebo que no se ha suministrado a Cli o a Sucursal en las tablas asociadas:
                  SELECT COUNT(*) INTO suministros_a_cliente
         FROM cliente_solicita_suministro_v WHERE cod_vino_recuperado = cod_vino;
                  SELECT COUNT(*) INTO suministros_a_sucursal
         FROM sucursal_pedido_sucursal_v WHERE cod_vino_recuperado = cod_vino;
                  -- si alguna de las variables es mayor que 0 (> 0), entonces es porque el vino
                  --   ha sido suministrado en algun momento, luego su productor NO se
                  --   podrá eliminar.
                  IF ( (suministros_a_cliente > 0) OR (suministros_a_sucursal > 0) ) THEN
                           raise_application_error (-20070, 'No se puede borrar un Productor que tiene alguno de sus vinos SUMINISTRADOS (a un
Cliente o una Sucursal, da igual).' ) ;


      END IF;

      FETCH c1 INTO cod_vino_recuperado;
```

```
        END LOOP;

   CLOSE c1;
END;
```

17. Una sucursal no puede realizar pedidos a sucursales de su misma delegación.

```
create or replace TRIGGER DIS_SUCURSALES_IGUALES_17
BEFORE INSERT OR UPDATE ON  sucursal_pedido_sucursal  FOR EACH ROW
DECLARE
        ca1 varchar2(30);
   ca2 varchar2(30);
   delegacion1 varchar2(30);
   delegacion2 varchar2(30);
BEGIN
        SELECT comunidad_autonoma INTO ca1
   FROM sucursal_v WHERE :NEW.cod_sucursal1 = cod_sucursal;
   GET_DELEGACION (delegacion1, ca1);

   SELECT comunidad_autonoma INTO ca2
   FROM sucursal_v WHERE :NEW.cod_sucursal2 = cod_sucursal;
   GET_DELEGACION (delegacion2, ca2);

   IF (delegacion1 = delegacion2) THEN
      raise_application_error (-20071, 'Una sucursal no puede pedir a otra sucursal de SU MISMA Delegacion.') ;
        END IF;
END;
```

18. La cantidad total de cada vino que las sucursales piden a otras sucursales, no puede exceeder la cantidad total que de ese vino solicitan los clientes.

Autores: Alonso BH, Juan Carlos GQ

```
create or replace TRIGGER DIS_SUMINISTRADA
BEFORE INSERT OR UPDATE ON sucursal_pedido_sucursal FOR EACH ROW
DECLARE
   PRAGMA AUTONOMOUS_TRANSACTION;
        cantidad_sucursal number(8);          -- cantidad pedida del vino X por las sucursales
        cantidad_cliente number(8);           -- ídem de arriba para los cliente
        cantidad_antigua  number(8);   --variable auxiliar para la zona de UPDATING
BEGIN
        IF (INSERTING) THEN
                SELECT SUM(CANTIDAD) INTO cantidad_sucursal
                FROM sucursal_pedido_sucursal_v
                WHERE cod_vino = :new.cod_vino;
                SELECT SUM(CANTIDAD) INTO cantidad_cliente
                FROM cliente_solicita_suministro_v
                WHERE cod_vino = :new.cod_vino;

                IF (cantidad_sucursal + :new.cantidad > cantidad_cliente ) THEN
                        raise_application_error (-20071, 'Error al insertar: la cantidad total de un vino suministrada entre sucursales debe ser
MENOR O IGUAL a la suministra de ese mismo vino a CLIENTES.') ;
      END IF;
        END IF;

        IF ( UPDATING ('CANTIDAD' ) ) THEN
                SELECT SUM(CANTIDAD) INTO cantidad_sucursal
        FROM sucursal_pedido_sucursal_v
        WHERE cod_vino = :new.cod_vino;
                SELECT SUM(CANTIDAD) INTO cantidad_cliente
        FROM cliente_solicita_suministro_v
        WHERE cod_vino = :new.cod_vino;
```

```
            -- como estamos actualizando el campo en Sucursal_Pedido_Sucursal, significa
      -- que tenemos que restar a la suma de cantidades la cantidad antigua de esta
      -- fila, sumarle la nueva, y ver si es menor que en cliente_solicita_suministro, tal que:
      -- capturar el valor antiguo de la fila a modificar:
      SELECT cantidad INTO cantidad_antigua
      FROM sucursal_pedido_sucursal_v
      WHERE cod_sucursal1 = :new.cod_sucursal1 AND cod_sucursal2 = :new.cod_sucursal2
         AND cod_vino = :new.cod_vino;

      -- ahora se comprueba:
            IF ( cantidad_sucursal -  cantidad_antigua + :new.cantidad > cantidad_cliente ) THEN
                  raise_application_error (-20071, 'Error al Actualizar: la cantidad total de un vino suministrada entre sucursales debe ser
MENOR O IGUAL a la suministra de ese mismo vino a CLIENTES.') ;
            END IF;

         END IF;  -- fin del updating
END;
```

19. La sucursal a la que otra se dirige para hacer pedidos de vinos que ella no distribuye, tiene que suministrar directamente el vino que se solicita; es decir, si, por ejemplo, una sucursal de Andalucía requiere vino de Rioja, tiene que solicitarlo, necesariamente, a una sucursal de la delegación de Madrid.

```
create or replace TRIGGER DIS_SUC_PEDIDO_DISTRIB_DIRECT
BEFORE INSERT OR UPDATE ON  sucursal_pedido_sucursal  FOR EACH ROW
DECLARE
   delegacion_sucursal_pedida varchar2(30);
   delegacion_vino varchar2(30);
   ca_receptora varchar2(30);
   ca_vino  varchar2(30);
```

```
BEGIN
        -- sacar la CA de la sucursal a la que se pide (la 'receptora')
        SELECT comunidad_autonoma INTO ca_receptora
      FROM SUCURSAL_V WHERE :NEW.cod_sucursal2 = cod_sucursal;
        GET_DELEGACION (delegacion_sucursal_pedida, ca_receptora);
   -- sacar la CA del vino
        SELECT comunidad_autonoma INTO ca_vino
      FROM VINOS_V WHERE :NEW.cod_vino = cod_vino;
   GET_DELEGACION (delegacion_vino, ca_vino );

   -- ¿son la misma delegación? => NO -> pues error
   IF (delegacion_sucursal_pedida <> delegacion_vino) THEN
      raise_application_error (-20080, 'La CA del vino no está dentro de la delegación de la sucursal a la que se le ha hecho el pedido.' );
   END IF;
END;
```

20. La fecha del pedido de una sucursal S1 a otra S2 de un determinado vino, tiene que ser posterior a la fecha del último pedido que S1 haya cursado a S2 de ese mismo vino.

```
create or replace TRIGGER DIS_PEDIDO_SUCURSAL_FECHA BEFORE INSERT OR UPDATE ON sucursal_pedido_sucursal FOR EACH
ROW
DECLARE
  fecha_ultima sucursal_pedido_sucursal.fecha%TYPE;
  contador NUMBER(8);
BEGIN
   IF (INSERTING) THEN -- Comprobamos que se ha realizado un pedido
      SELECT COUNT(*) INTO contador
      FROM sucursal_pedido_sucursal_v
      WHERE cod_sucursal1= :NEW.cod_sucursal1
         AND cod_sucursal2= :NEW.cod_sucursal2
```

```
        AND cod_vino= :NEW.cod_vino;

      IF (contador > 0) THEN --Es porque hay pedidos
        SELECT MAX(fecha) INTO fecha_ultima-- Almacenamos la fecha del último pedido
        FROM  sucursal_pedido_sucursal_v
        WHERE cod_sucursal1= :NEW.cod_sucursal1
          AND cod_sucursal2= :NEW.cod_sucursal2
          AND cod_vino= :NEW.cod_vino
        GROUP BY cod_vino;

        IF (fecha_ultima > :NEW.fecha) THEN
          RAISE_APPLICATION_ERROR(-20037, 'La fecha debe ser superior o igual a: ' || fecha_ultima);
        END IF;
      END IF;
    END IF;

    IF (UPDATING('fecha')) THEN
      IF (:OLD.fecha > :NEW.fecha) THEN
        RAISE_APPLICATION_ERROR(-20038, 'La fecha debe ser superior o igual a: ' || :OLD.fecha);
      END IF;
    END IF;
END;
```

21. La fecha de pedido de un vino de una sucursal S1 a otra S2, tiene que ser posterior a la última fecha de solicitud de suministro de ese mismo vino recibida en S1 por un cliente. Por ejemplo, si un cliente de Andalucía solicita suministro de vino de Rioja a la sucursal S1 en fecha F, y esa solicitud es la última que S1 ha recibido de vino de Rioja, el pedido de S1 a la sucursal de la delegación de Madrid correspondiente tiene que ser de fecha posterior a F.

```
create or replace TRIGGER DIS_FECHA_SUCUR_CLIENTE BEFORE INSERT OR UPDATE ON sucursal_pedido_sucursal  FOR EACH
ROW
```

```
DECLARE
  fecha_ultima sucursal_pedido_sucursal.fecha%TYPE;
BEGIN
    IF (INSERTING) THEN
        SELECT MAX(fecha) INTO fecha_ultima-- Almacenamos la última fecha del pedido
        FROM cliente_solicita_suministro_v
        WHERE (cod_sucursal = :NEW.cod_sucursal1
        AND cod_vino = :NEW.cod_vino);

        IF (fecha_ultima > :NEW.fecha) THEN -- No puede ser anterior
            RAISE_APPLICATION_ERROR(-20039, 'Error trigger 21: La fecha debe ser superior o igual a: ' || fecha_ultima);
        END IF;
    END IF;

    -- SI ACTUALIZAMOS LA FECHA COMPROBAMOS QUE SEA POSTERIOR.
    IF (UPDATING('fecha')) THEN
        IF (:OLD.fecha > :NEW.fecha) THEN
            RAISE_APPLICATION_ERROR(-20050, 'La fecha debe ser superior o igual a: ' || :OLD.fecha);
        END IF;
    END IF;
END;
```

Práctica 4: Implementación de actualizaciones

1. Dar de alta a un nuevo empleado.

```
create or replace PROCEDURE PR_NUEVO_EMPLEADO(codigo NUMBER, dni_dado VARCHAR2,
    nombre_dado VARCHAR2, fecha_dada DATE, salario_dado NUMBER, direccion_dada VARCHAR2,
    sucursal_dada NUMBER)
```

Autores: Alonso BH, Juan Carlos GQ

```
IS
   ca_asociada sucursal.COMUNIDAD_AUTONOMA%TYPE;
   delegacion VARCHAR(50);
   contador number(2);

BEGIN

   SELECT COUNT(*) INTO contador FROM empleado_v where cod_empleado = codigo;

   IF (contador > 0) THEN
      RAISE_APPLICATION_ERROR(-20052, 'YA EXISTE UN EMPLEADO CON ESE CÓDIGO');
   ELSE  -- NO EXISTE, SEGUIMOS.
      -- OBTENER LA DELEGACION DE LA SUCURSAL DEL EMP PARA SABER DONDE
      -- (EN QUÉ LOCALIDAD) LO TENEMOS QUE INSERTAR
      SELECT comunidad_autonoma INTO ca_asociada FROM sucursal_v WHERE cod_sucursal = sucursal_dada;
      GET_DELEGACION(delegacion, ca_asociada);  -- OBTENEMOS LA DELEGACIÓN

      CASE
         -- SI LA DELEGACIÓN PERTENECE A MADRID
         WHEN delegacion = 'Madrid' THEN
         INSERT INTO juanal1.empleado
         VALUES (codigo, dni_dado, nombre_dado, fecha_dada, salario_dado, direccion_dada,
         sucursal_dada);

         -- SI LA DELEGACIÓN PERTENECE A BARCELONA
         WHEN delegacion = 'Barcelona' THEN
         INSERT INTO juanal2.empleado
         VALUES (codigo, dni_dado, nombre_dado, fecha_dada, salario_dado, direccion_dada,
         sucursal_dada);
```

```
        -- SI LA DELEGACIÓN PERTENECE A LA CORUÑA
        WHEN delegacion = 'La Coruña' THEN
        INSERT INTO juanal3.empleado
        VALUES (codigo, dni_dado, nombre_dado, fecha_dada, salario_dado, direccion_dada,
        sucursal_dada);

        -- SI LA DELEGACIÓN PERTENECE A SEVILLA
        WHEN delegacion = 'Sevilla' THEN
        dbms_output.put_line('asdkfjsdaf');
        INSERT INTO juanal4.empleado
        VALUES (codigo, dni_dado, nombre_dado, fecha_dada, salario_dado, direccion_dada,
        sucursal_dada);
      END CASE;
      DBMS_OUTPUT.PUT_LINE('EL EMPLEADO ' || nombre_dado ||' HA SIDO DADO DE ALTA DE FORMA CORRECTA.');

   END IF;
END PR_NUEVO_EMPLEADO;
```

2. Dar de baja a un empleado.

```
create or replace PROCEDURE PR_QUITAR_EMPLEADO (codigo NUMBER)
IS
   contador1 NUMBER(5);
   delegacion VARCHAR(50);
   delegacion2 VARCHAR(50);
   ca_sucursal sucursal.comunidad_autonoma%TYPE;   -- CA de la sucursal donde trabaja EMP
   ca_sucursal2 sucursal.comunidad_autonoma%TYPE;  -- CA de la sucursal donde EMP es director

BEGIN
   -- VAMOS A COMPROBAR PRIMERO SI EL EMPLEADO EXISTE EN LA COMPAÑÍA
```

```
    SELECT COUNT(*) INTO contador1 FROM empleado_v WHERE cod_empleado = codigo;

    IF (contador1 = 0) THEN -- el código de emp no existe
       RAISE_APPLICATION_ERROR(-20076, 'Lo sentimos, no existe el empleado que quieres borrar. ');
    ELSE
       -- EXISTE EL EMPLEADO A BORRAR

       -- ahora sacamos la delegacion en la que trabaja el empleado y así detectaremos en qué
       --   fragmento hay que borrar
       SELECT s.comunidad_autonoma INTO ca_sucursal
          FROM sucursal_v s, empleado_v e
          WHERE s.cod_sucursal = e.cod_sucursal AND e.cod_empleado = codigo;

       GET_DELEGACION(delegacion, ca_sucursal);  -- OBTENEMOS LA DELEGACIÓN DONDE SE GUARDA EL EMP

       CASE
       -- SI LA DELEGACIÓN = MADRID
       WHEN delegacion = 'Madrid' THEN

          -- AHORA, SI ES DIRECTOR DE UNA SUCURSAL LA PONEMOS A NULL Y MOSTRAMOS EL MENSAJE
          SELECT COUNT(*) INTO contador1 FROM sucursal_v WHERE director = codigo;

          IF (contador1 > 0) THEN    -- hay una sucursal que tiene como director al empleado que queremos borrar
             -- OBTENEMOS LA DELEGACIÓN EN LA QUE ÉL ES EL DIRECTOR.
             SELECT s.comunidad_autonoma  INTO ca_sucursal2
             FROM sucursal_v s
             WHERE s.director = codigo;

             GET_DELEGACION(delegacion2, ca_sucursal2);  -- OBTENEMOS LA DELEGACIÓN DONDE ES DIRECTOR
```

```
        IF (delegacion2 = 'Madrid') THEN -- SI LA DELEGACIÓN ES MADRID
           UPDATE juanal1.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'Barcelona') THEN -- SI LA DELEGACIÓN ES BARCELONA
           UPDATE juanal2.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'La Coruña') THEN -- SI LA DELEGACIÓN ES LA CORUÑA
           UPDATE juanal3.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'Sevilla') THEN -- SI LA DELEGACIÓN ES SEVILLA
           UPDATE juanal4.sucursal SET director = null WHERE director = codigo;
        END IF;
        DBMS_OUTPUT.PUT_LINE('LA SUCURSAL DE LA QUE ERA DIRECTOR HA QUEDADO VACÍA');
     END IF;

     -- si era director, ya lo hemos borrado de la sucursal, ahora lo borramos a él
     DELETE FROM juanal1.empleado WHERE cod_empleado = codigo;

  WHEN delegacion = 'Barcelona' THEN
     -- SI ES DIRECTOR DE UNA SUCURSAL LA PONEMOS A NULL Y MOSTRAMOS EL MENSAJE

     --IF (contador1 <> 0) THEN
     -- BUSCAMOS LA SUCURSAL DE LA QUE ES DIRECTOR Y LA MODIFICAMOS.
     SELECT COUNT(*) INTO contador1 FROM sucursal_v WHERE director = codigo;

     IF (contador1 > 0) THEN    -- hay una sucursal que tiene como director al empleado que queremos borrar
        -- OBTENEMOS LA DELEGACIÓN EN LA QUE ÉL ES EL DIRECTOR.
        SELECT s.comunidad_autonoma  INTO ca_sucursal2
        FROM sucursal_v s
        WHERE s.director = codigo;
```

```
        GET_DELEGACION(delegacion2, ca_sucursal2);  -- OBTENEMOS LA DELEGACIÓN DONDE ES DIRECTOR

        IF (delegacion2 = 'Madrid') THEN -- SI LA DELEGACIÓN ES MADRID
           UPDATE juanal1.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'Barcelona') THEN -- SI LA DELEGACIÓN ES BARCELONA
           UPDATE juanal2.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'La Coruña') THEN -- SI LA DELEGACIÓN ES LA CORUÑA
           UPDATE juanal3.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'Sevilla') THEN -- SI LA DELEGACIÓN ES SEVILLA
           UPDATE juanal4.sucursal SET director = null WHERE director = codigo;
        END IF;
        DBMS_OUTPUT.PUT_LINE('LA SUCURSAL DE LA QUE ERA DIRECTOR HA QUEDADO VACÍA');

     END IF;
     --END IF;

     -- si era director, ya lo hemos borrado de la sucursal, ahora lo borramos a él
     DELETE FROM juanal2.empleado WHERE cod_empleado = codigo;

  WHEN delegacion = 'La Coruña' THEN
     -- SI ES DIRECTOR DE UNA SUCURSAL LA PONEMOS A NULL Y MOSTRAMOS EL MENSAJE

     --IF (contador1 <> 0) THEN
     -- BUSCAMOS LA SUCURSAL DE LA QUE ES DIRECTOR Y LA MODIFICAMOS.
     SELECT COUNT(*) INTO contador1 FROM sucursal_v WHERE director = codigo;
     IF (contador1 > 0) THEN    -- hay una sucursal que tiene como director al empleado que queremos borrar
```

```
        -- OBTENEMOS LA DELEGACIÓN EN LA QUE ÉL ES EL DIRECTOR.
        SELECT s.comunidad_autonoma  INTO ca_sucursal2
        FROM sucursal_v s
        WHERE s.director = codigo;

        GET_DELEGACION(delegacion2, ca_sucursal2);  -- OBTENEMOS LA DELEGACIÓN DONDE ES DIRECTOR

        IF (delegacion2 = 'Madrid') THEN -- SI LA DELEGACIÓN ES MADRID
          UPDATE juanal1.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'Barcelona') THEN -- SI LA DELEGACIÓN ES BARCELONA
          UPDATE juanal2.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'La Coruña') THEN -- SI LA DELEGACIÓN ES LA CORUÑA
          UPDATE juanal3.sucursal SET director = null WHERE director = codigo;
        END IF;
        IF (delegacion2 = 'Sevilla') THEN -- SI LA DELEGACIÓN ES SEVILLA
          UPDATE juanal4.sucursal SET director = null WHERE director = codigo;
        END IF;

      DBMS_OUTPUT.PUT_LINE('LA SUCURSAL DE LA QUE ERA DIRECTOR HA QUEDADO VACÍA');
      END IF;

      -- si era director, ya lo hemos borrado de la sucursal, ahora lo borramos a él
      DELETE FROM juanal3.empleado WHERE cod_empleado = codigo;

    WHEN delegacion = 'Sevilla' THEN
      -- SI ES DIRECTOR DE UNA SUCURSAL LA PONEMOS A NULL Y MOSTRAMOS EL MENSAJE
```

```
--IF (contador1 <> 0) THEN
-- BUSCAMOS LA SUCURSAL DE LA QUE ES DIRECTOR Y LA MODIFICAMOS.
SELECT COUNT(*) INTO contador1 FROM sucursal_v WHERE director = codigo;
IF (contador1 > 0) THEN    -- hay una sucursal que tiene como director al empleado que queremos borrar

   -- OBTENEMOS LA DELEGACIÓN EN LA QUE ÉL ES EL DIRECTOR.
   SELECT s.comunidad_autonoma  INTO ca_sucursal2
   FROM sucursal_v s
   WHERE s.director = codigo;

   GET_DELEGACION(delegacion2, ca_sucursal2);  -- OBTENEMOS LA DELEGACIÓN DONDE ES DIRECTOR

   IF (delegacion2 = 'Madrid') THEN -- SI LA DELEGACIÓN ES MADRID
     UPDATE juanal1.sucursal SET director = null WHERE director = codigo;
   END IF;
   IF (delegacion2 = 'Barcelona') THEN -- SI LA DELEGACIÓN ES BARCELONA
     UPDATE juanal2.sucursal SET director = null WHERE director = codigo;
   END IF;
   IF (delegacion2 = 'La Coruña') THEN -- SI LA DELEGACIÓN ES LA CORUÑA
     UPDATE juanal3.sucursal SET director = null WHERE director = codigo;
   END IF;
   IF (delegacion2 = 'Sevilla') THEN -- SI LA DELEGACIÓN ES SEVILLA
     UPDATE juanal4.sucursal SET director = null WHERE director = codigo;
   END IF;

DBMS_OUTPUT.PUT_LINE('LA SUCURSAL DE LA QUE ERA DIRECTOR HA QUEDADO VACÍA');
END IF;

-- si era director, ya lo hemos borrado de la sucursal, ahora lo borramos a él
DELETE FROM juanal4.empleado WHERE cod_empleado = codigo;
```

```
        END CASE;

        DBMS_OUTPUT.PUT_LINE('EL EMPLEADO ' || codigo ||  ' FUE ELIMINADO CORRECTAMENTE');

    END IF;
END PR_QUITAR_EMPLEADO;
```

3. Modificar el salario de un empleado.

```
create or replace PROCEDURE PR_UPT_SALARIO (codigo empleado.cod_empleado%TYPE, nuevo_salario empleado.salario%TYPE)
IS
    contador NUMBER(8);
    delegacion VARCHAR2(50);
    ca Sucursal.comunidad_autonoma%TYPE;
BEGIN

    -- VEMOS SI EL EMPLEADO EXISTE EN LA COMPAÑÍA.
    SELECT COUNT (*) INTO contador FROM empleado_v WHERE cod_empleado = codigo;

    IF (contador = 0) THEN -- SI EL CODIGO DE EMPLEADO NO EXISTE
        RAISE_APPLICATION_ERROR(-20109, 'No existe empleado con ese código. ');
    ELSE
        -- OBTENEMOS LA DELEGACIÓN EN LA QUE TRABAJA EL EMPLEADO.
        SELECT s.comunidad_autonoma INTO ca
            FROM sucursal_v s, empleado_v emp
            WHERE s.cod_sucursal = emp.cod_sucursal
                AND emp.cod_empleado = codigo;

        GET_DELEGACION(delegacion, ca);  -- OBTENEMOS LA DELEGACIÓN
```

```
IF (delegacion = 'Madrid') THEN
   UPDATE juanal1.empleado SET salario = nuevo_salario WHERE cod_empleado = codigo;
END IF;

IF (delegacion = 'Barcelona') then
   UPDATE juanal1.empleado SET salario = nuevo_salario WHERE cod_empleado = codigo;
END IF;

IF (delegacion = 'La Coruña') then
   UPDATE juanal3.empleado SET salario = nuevo_salario WHERE cod_empleado = codigo;
END IF;

IF (delegacion = 'Sevilla') then
   UPDATE juanal4.empleado SET salario = nuevo_salario WHERE cod_empleado = codigo;
END IF;

/*
CASE
-- SI LA DELEGACIÓN PERTENECE A MADRID
WHEN delegacion = 'Madrid' THEN
   UPDATE juanal1.empleado SET salario = nuevo_salario WHERE cod_empleado = codigo;

-- SI LA DELEGACIÓN PERTENECE A BARCELONA
WHEN delegacion = 'Barcelona' THEN
   UPDATE juanal2.empleado SET salario = nuevo_salario WHERE cod_empleado = codigo;

-- SI LA DELEGACIÓN PERTENECE A LA CORUÑA
WHEN delegacion = 'La Coruña' THEN
   UPDATE juanal3.empleado SET salario = nuevo_salario WHERE cod_empleado = codigo;
```

```
        -- SI LA DELEGACIÓN PERTENECE A SEVILLA
        WHEN delegacion = 'Sevilla' THEN
            UPDATE juanal4.empleado SET salario = nuevo_salario WHERE cod_empleado = codigo;
        ELSE null;
        END CASE;
        */
        DBMS_OUTPUT.PUT_LINE('EL SALARIO PARA EL EMPLEADO ' || codigo ||  ' HA SIDO MODIFICADO CORRECTAMENTE a ' ||
nuevo_salario);
    END IF;

END PR_UPT_SALARIO;
```

4. Trasladar de sucursal a un empleado.

```
create or replace PROCEDURE PR_TRASLADO_EMP (codigo_emp empleado.cod_empleado%TYPE, sucursal_nueva empleado.cod_sucursal
%TYPE, direccion_nueva empleado.direccion%TYPE DEFAULT NULL )
IS
    contador NUMBER(8);
    deleg_antigua VARCHAR2(50);
    deleg_nueva VARCHAR2(50);
    ca_suc1 sucursal.comunidad_autonoma%TYPE;
    ca_suc2 sucursal.comunidad_autonoma%TYPE;
    DNI_ empleado.DNI%TYPE;
    nombre_ empleado.nombre%TYPE;
    fecha_ empleado.fecha_contrato%TYPE;
    salario_ empleado.salario%TYPE;

BEGIN
```

```
SELECT dni  INTO DNI_  FROM empleado_v WHERE cod_empleado = codigo_emp;
SELECT nombre INTO nombre_ FROM empleado_v WHERE cod_empleado = codigo_emp;
SELECT fecha_contrato INTO fecha_ FROM empleado_v WHERE cod_empleado = codigo_emp;
SELECT salario INTO salario_ FROM empleado_v WHERE cod_empleado = codigo_emp;

/*CURSOR CU_EMPLEADO IS -- CURSOR PARA GUARDAR LOS DATOS DEL EMPLEADO QUE NO SE MODIFICARÁN.
   SELECT DNI, nombre, fecha_contrato, salario
   FROM empleado_v
   WHERE cod_empleado = codigo_emp;
--fila_empleado CU_EMPLEADO%ROWTYPE;
*/

-- COMPROBAMOS QUE EL EMPLEADO EXISTE
SELECT COUNT(*) INTO contador FROM empleado_v WHERE cod_empleado = codigo_emp;

IF (contador = 0) THEN -- SI NO EXISTE EL EMPLEADO
   RAISE_APPLICATION_ERROR(-20076,'El empleado no existe. ');

ELSE  -- EL EMPLEADO EXISTE, PODEMOS CONTINUAR...
   --OPEN CU_EMPLEADO; -- ABRIMOS EL CURSOR
   --FETCH CU_EMPLEADO INTO fila_empleado; -- LEEMOS LOS DATOS DEL EMPLEADO
   --IF (CU_EMPLEADO%FOUND) THEN -- SI SE HA ENCONTRADO UN EMPLEADO (fila guardada)

   -- Obtener la delegación/localidad ACTUAL (ca_suc1) del Empleado
   --   para saber su  ubicación y poder eliminarle de ahí.
   SELECT comunidad_autonoma INTO ca_suc1 FROM sucursal_v s, empleado_v emp
   WHERE emp.cod_sucursal = s.cod_sucursal AND cod_empleado = codigo_emp;
   GET_DELEGACION(deleg_antigua, ca_suc1);  -- OBTENEMOS LA DELEGACIÓN

   DBMS_OUTPUT.PUT_LINE ('Localidad actual del empleado: ' || deleg_antigua || '.' );
```

```
-- Comprobar, antes que nada, si existe la sucursal a la que se va a trasladar
SELECT COUNT(*) INTO contador FROM sucursal_v s
WHERE s.cod_sucursal = sucursal_nueva;

IF (contador = 0) THEN  -- SI NO EXISTE LA NUEVA SUCURSAL.
   RAISE_APPLICATION_ERROR(-20110, 'LA SUCURSAL NUEVA NO EXISTE.');
ELSE
   -- AHORA QUE SABEMOS QUE LA SUCURSAL NUEVA EXISTE, SACAMOS
   --   LA DELEG/LOCALIDAD ASOCIADA
   SELECT comunidad_autonoma INTO ca_suc2 FROM sucursal_v s
   WHERE s.cod_sucursal = sucursal_nueva;
   GET_DELEGACION(deleg_nueva, ca_suc2);  -- OBTENEMOS LA DELEGACIÓN

   -- SI LA SUCURSAL NUEVA ESTÁ EN LA DELEGACIÓN DEL EMPLEADO, NO
   -- HAY QUE CAMBIARLO DE LOCALIDAD, SOLO ALTERAR EL CAMPO 'SUCURSAL'
   -- DE LA FILA ASOCIADA A ESE EMPLEADO
   IF (deleg_antigua = deleg_nueva) THEN
      CASE
         -- SI LA ANTIGUA DELEGACIÓN ES MADRID
         WHEN deleg_antigua = 'Madrid' THEN
            UPDATE juanal1.empleado SET cod_sucursal=sucursal_nueva
            WHERE cod_empleado=codigo_emp;

         -- SI LA ANTIGUA DELEGACIÓN ES BARCELONA
         WHEN deleg_antigua = 'Barcelona' THEN
            UPDATE juanal2.empleado SET cod_sucursal=sucursal_nueva
            WHERE cod_empleado=codigo_emp;

         -- SI LA ANTIGUA DELEGACIÓN ES LA CORUÑA
```

```
        WHEN deleg_antigua = 'La Coruña' THEN
            UPDATE juanal3.empleado SET cod_sucursal=sucursal_nueva
            WHERE cod_empleado=codigo_emp;

        -- SI LA ANTIGUA DELEGACIÓN ES SEVILLA
        WHEN deleg_antigua = 'Sevilla' THEN
            UPDATE juanal4.empleado SET cod_sucursal=sucursal_nueva
            WHERE cod_empleado=codigo_emp;

    END CASE;

ELSE  -- LA SUCURSAL NUEVA ES DE OTRA DELEGACIÓN => REUBICAR EMPLEADO

    -- OPERACIONES DE MODIFICACIÓN 1: BORRAR AL EMPLEADO DE SU LOCALIDAD ANTIGUA
    CASE
        -- SI LA ANTIGUA DELEGACIÓN ES MADRID
        WHEN deleg_antigua = 'Madrid' THEN
            DELETE FROM juanal1.empleado WHERE cod_empleado = codigo_emp;

        -- SI LA ANTIGUA DELEGACIÓN ES BARCELONA
        WHEN deleg_antigua = 'Barcelona' THEN
            DELETE FROM juanal2.empleado WHERE cod_empleado = codigo_emp;

        -- SI LA ANTIGUA DELEGACIÓN ES LA CORUÑA
        WHEN deleg_antigua = 'La Coruña' THEN
            DELETE FROM juanal3.empleado WHERE cod_empleado = codigo_emp;

        -- SI LA ANTIGUA DELEGACIÓN ES SEVILLA
        WHEN deleg_antigua = 'Sevilla' THEN
            DELETE FROM juanal4.empleado WHERE cod_empleado = codigo_emp;
```

```
END CASE;

-- OPERACIÓN DE MODIFICACION 2.-
-- INSERTAR AL EMPLEADO EN LA DELEGACIÓN NUEVA.
DBMS_OUTPUT.PUT_LINE ('Va a ir a la localidad ' || deleg_nueva || '.' );
CASE

   -- SI LA NUEVA DELEGACIÓN ES MADRID
   WHEN deleg_nueva = 'Madrid' THEN
      INSERT INTO juanal1.empleado (cod_empleado, DNI, nombre, fecha_contrato, salario, direccion, cod_sucursal)
      VALUES (codigo_emp, DNI_, nombre_,
      fecha_, salario_,
      direccion_nueva, sucursal_nueva);

   -- SI LA NUEVA DELEGACIÓN ES BARCELONA
   WHEN deleg_nueva = 'Barcelona' THEN
      INSERT INTO juanal2.empleado (cod_empleado, DNI, nombre, fecha_contrato, salario, direccion, cod_sucursal)
      VALUES (codigo_emp, DNI_,nombre_,  fecha_,
      salario_, direccion_nueva, sucursal_nueva);

   -- SI LA NUEVA DELEGACIÓN ES LA CORUÑA
   WHEN deleg_nueva = 'La Coruña' THEN
      INSERT INTO juanal3.empleado (cod_empleado, DNI, nombre, fecha_contrato, salario, direccion, cod_sucursal)
      VALUES (codigo_emp, DNI_,nombre_,  fecha_,
      salario_, direccion_nueva, sucursal_nueva);

   -- SI LA NUEVA DELEGACIÓN ES SEVILLA
   WHEN deleg_nueva = 'Sevilla' THEN
      INSERT INTO juanal4.empleado (cod_empleado, DNI, nombre, fecha_contrato, salario, direccion, cod_sucursal)
```

```
                VALUES (codigo_emp, DNI_,nombre_,  fecha_,
                salario_, direccion_nueva, sucursal_nueva);

            END CASE;
          END IF;
       END IF;     -- existe la sucursal NUEVA

       --END IF;   -- hemos captado la fila
       --CLOSE CU_EMPLEADO;

   END IF;
   DBMS_OUTPUT.PUT_LINE('EL TRASLADO DEL EMPLEADO ' || codigo_emp || '  A LA SUCURSAL ' || sucursal_nueva || ' HA
FINALIZADO CORRECTAMENTE.' );
END PR_TRASLADO_EMP;
```

5. Dar de alta una nueva sucursal.

```
create or replace PROCEDURE PR_NUEVA_SUC
(codigo  sucursal.cod_sucursal%TYPE, nombre_dado sucursal.nombre%TYPE,
ciudad_dada sucursal.ciudad%TYPE, ca_dada sucursal.comunidad_autonoma%TYPE,
director_dado empleado.cod_empleado%TYPE DEFAULT NULL)
IS
   delegacion VARCHAR(50);
   contador NUMBER(1);
BEGIN
   -- COMPROBAMOS SI ESA SUCURSAL YA EXISTE, EN ESE CASO NO SE PUEDE INSERTAR Y SALTA ERROR
   SELECT COUNT(*) INTO contador FROM sucursal_v where cod_sucursal = codigo;

   IF (contador > 0) THEN      -- ESTA SUCURSAL YA EXISTE
      RAISE_APPLICATION_ERROR(-20065, 'La sucursal asociada a este código ya existe.');
```

```
    ELSE

        -- OBTENEMOS LA DELEGACIÓN DE LA SUCURSAL.
        GET_DELEGACION(delegacion, ca_dada);

        CASE
            -- SI LA DELEGACIÓN PERTENECE A MADRID
            WHEN delegacion = 'Madrid' THEN
                INSERT INTO juanal1.sucursal
                VALUES (codigo, nombre_dado, ciudad_dada, ca_dada, director_dado);

            -- SI LA DELEGACIÓN PERTENECE A BARCELONA
            WHEN delegacion = 'Barcelona' THEN
                INSERT INTO juanal2.sucursal
                VALUES (codigo, nombre_dado, ciudad_dada, ca_dada, director_dado);

            -- SI LA DELEGACIÓN PERTENECE A LA CORUÑA
            WHEN delegacion = 'La Coruña' THEN
                INSERT INTO juanal3.sucursal
                VALUES (codigo, nombre_dado, ciudad_dada, ca_dada, director_dado);

            -- SI LA DELEGACIÓN PERTENECE A SEVILLA
            WHEN delegacion = 'Sevilla' THEN
                INSERT INTO juanal4.sucursal
                VALUES (codigo, nombre_dado, ciudad_dada, ca_dada, director_dado);

        END CASE;
        DBMS_OUTPUT.PUT_LINE ('LA NUEVA SUCURSAL ' || nombre_dado || ' HA SIDO DADA DE ALTA SATISFACTORIAMENTE.');

    END IF;
```

END PR_NUEVA_SUC;

6. Cambiar el director de una sucursal.

```
create or replace PROCEDURE PR_DIRECTOR_SUC (codigo_suc sucursal.cod_sucursal%TYPE, director_nuevo empleado.cod_empleado
%TYPE)
IS
   contador1 NUMBER(4);
   contador2 NUMBER(4);
   delegacion VARCHAR(50);
   ca_sucursal sucursal.comunidad_autonoma%TYPE;
BEGIN
   -- COMPROBAR SI EXISTE UNA SUCURSAL  CON ESE CODIGO
   SELECT COUNT (*) INTO contador1 FROM sucursal_v WHERE cod_sucursal = codigo_suc;
   IF(contador1 = 0) THEN
      RAISE_APPLICATION_ERROR(-20106, 'NO HAY UNA SUCURSAL CON ESE CÓDIGO');
   END IF;

   -- COMPROBAR SI EXISTE UN EMPLEADO CON ESE CODIGO
   SELECT COUNT (*) INTO contador2 FROM empleado_v WHERE cod_empleado = director_nuevo;
   IF(contador2 = 0) THEN
      RAISE_APPLICATION_ERROR(-20107, 'NO HAY NINGÚN EMPLEADO CON ESE CÓDIGO');
   END IF;

   IF (NOT (contador1=0 OR contador2=0) ) THEN
      -- OBTENEMOS LA DELEGACIÓN DE LA SUCURSAL PARA REALIZAR LA MODIFICACIÓN
      SELECT comunidad_autonoma INTO ca_sucursal FROM sucursal_v WHERE cod_sucursal = codigo_suc;
      GET_DELEGACION(delegacion, ca_sucursal);
```

```
    CASE
      -- SI LA DELEGACIÓN PERTENECE A MADRID
      WHEN delegacion = 'Madrid' THEN
        UPDATE juanal1.sucursal SET director = director_nuevo WHERE cod_sucursal = codigo_suc;

      -- SI LA DELEGACIÓN PERTENECE A BARCELONA
      WHEN delegacion = 'Barcelona' THEN
        UPDATE juanal2.sucursal SET director = director_nuevo WHERE cod_sucursal = codigo_suc;

      -- SI LA DELEGACIÓN PERTENECE A LA CORUÑA
      WHEN delegacion = 'La Coruña' THEN
        UPDATE juanal3.sucursal SET director = director_nuevo WHERE cod_sucursal = codigo_suc;

      -- SI LA DELEGACIÓN PERTENECE A SEVILLA
      WHEN delegacion = 'Sevilla' THEN
        UPDATE juanal4.sucursal SET director = director_nuevo WHERE cod_sucursal = codigo_suc;

    END CASE;

    DBMS_OUTPUT.PUT_LINE('EL DIRECTOR ' || director_nuevo || ' HA SIDO ASIGNADO A LA SUCURSAL ' || codigo_suc || '
ADECUADAMENTE.');
  END IF;
END PR_DIRECTOR_SUC ;
```

7. Dar de alta a un nuevo cliente.

```
create or replace PROCEDURE PR_NUEVO_CLIENTE (codigo clientes.cod_cliente%TYPE, dni_dado clientes.DNI%TYPE, nombre_dado
clientes.nombre%TYPE,
direccion_dada clientes.direccion%TYPE, tipo_dado clientes.tipo_cliente%TYPE , ca_dada clientes.comunidad_autonoma%TYPE)
IS
```

```
   contador NUMBER(8);
   delegacion VARCHAR(50);
BEGIN

   -- VEMOS SI EL CLIENTE EXISTE EN EL SISTEMA.
   SELECT COUNT (*) INTO contador FROM clientes_v WHERE cod_cliente = codigo;

   IF (contador <> 0) THEN -- SI EL CODIGO DEL CLIENTE EXISTE
      RAISE_APPLICATION_ERROR(-20072, 'El cliente que has introducido (su código) YA EXISTE');
   ELSE
      -- OBTENEMOS LA DELEGACIÓN EN LA QUE SE INCLUIRÁ EL cliente.
      GET_DELEGACION(delegacion, ca_dada);

      CASE
         -- SI LA DELEGACIÓN PERTENECE A MADRID
         WHEN delegacion = 'Madrid' THEN
            INSERT INTO juanal1.clientes (cod_cliente, nombre, direccion, tipo_cliente, comunidad_autonoma, dni) VALUES (codigo,
nombre_dado, direccion_dada, tipo_dado, ca_dada, dni_dado);

         -- SI LA DELEGACIÓN PERTENECE A BARCELONA
         WHEN delegacion = 'Barcelona' THEN
            INSERT INTO juanal2.clientes  (cod_cliente, nombre, direccion, tipo_cliente, comunidad_autonoma, dni)  VALUES (codigo,
nombre_dado, direccion_dada, tipo_dado, ca_dada, dni_dado);

         -- SI LA DELEGACIÓN PERTENECE A LA CORUÑA
         WHEN delegacion = 'La Coruña' THEN
            INSERT INTO juanal3.clientes  (cod_cliente, nombre, direccion, tipo_cliente, comunidad_autonoma, dni)  VALUES (codigo,
nombre_dado, direccion_dada, tipo_dado, ca_dada, dni_dado);

         -- SI LA DELEGACIÓN PERTENECE A SEVILLA
```

```
        WHEN delegacion = 'Sevilla' THEN
          INSERT INTO juanal4.clientes  (cod_cliente, nombre, direccion, tipo_cliente, comunidad_autonoma, dni)  VALUES (codigo, nombre_dado, direccion_dada, tipo_dado, ca_dada, dni_dado);

      END CASE;

      DBMS_OUTPUT.PUT_LINE('EL CLIENTE LLAMADO ' || nombre_dado || ' HA SIDO DADO DE ALTA CORRECTAMENTE.');

   END IF;

END PR_NUEVO_CLIENTE ;
```

8. Dar de alta o actualizar un suministro.

9. Dar de baja suministros.

10. Dar de alta un pedido de una sucursal.

11. Dar de baja pedidos de una sucursal.

12. Dar de alta un nuevo vino.

Autores: Alonso BH, Juan Carlos GQ

13. Dar de baja un vino

14. Dar de alta un productor.

15. Dar de baja un productor.