Documentación de la práctica de Gestión de Bases de Datos

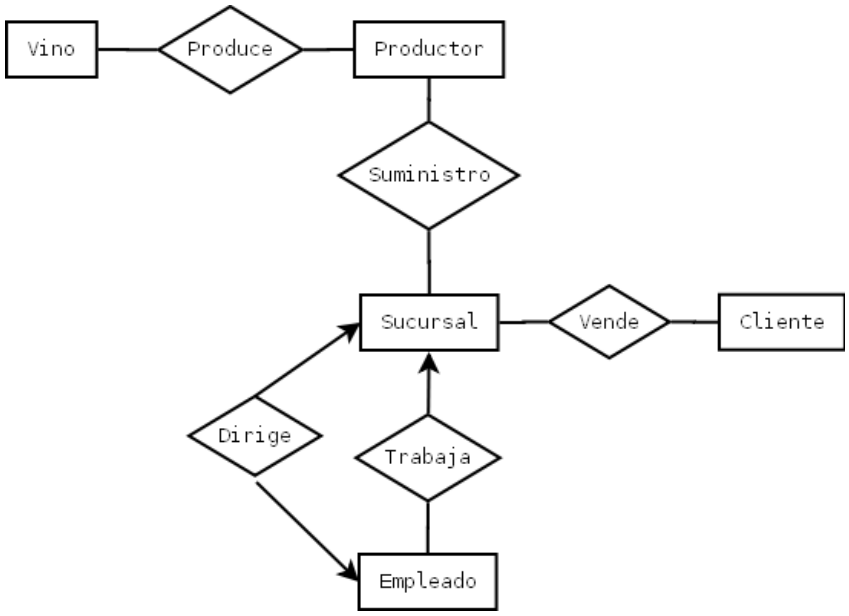
Alumnos:
Francisco Santolalla Quiñonero-76439251Q
Carlos Jesús Fernández Basso-75927137C

Índice:

1. Diseño conceptual y lógico de la base de datos

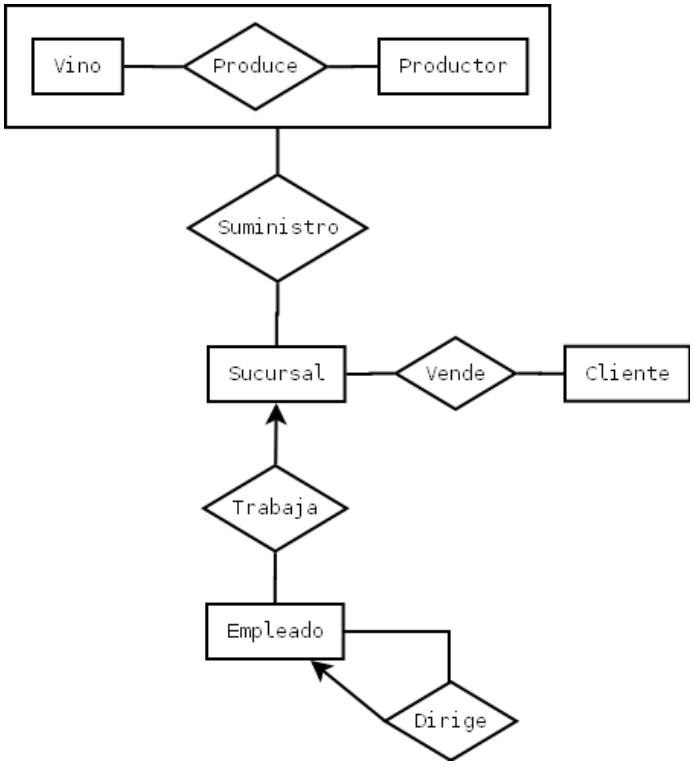
1.1 Diagrama E/R valorados y desechados

- Primer diagrama que valoramos:



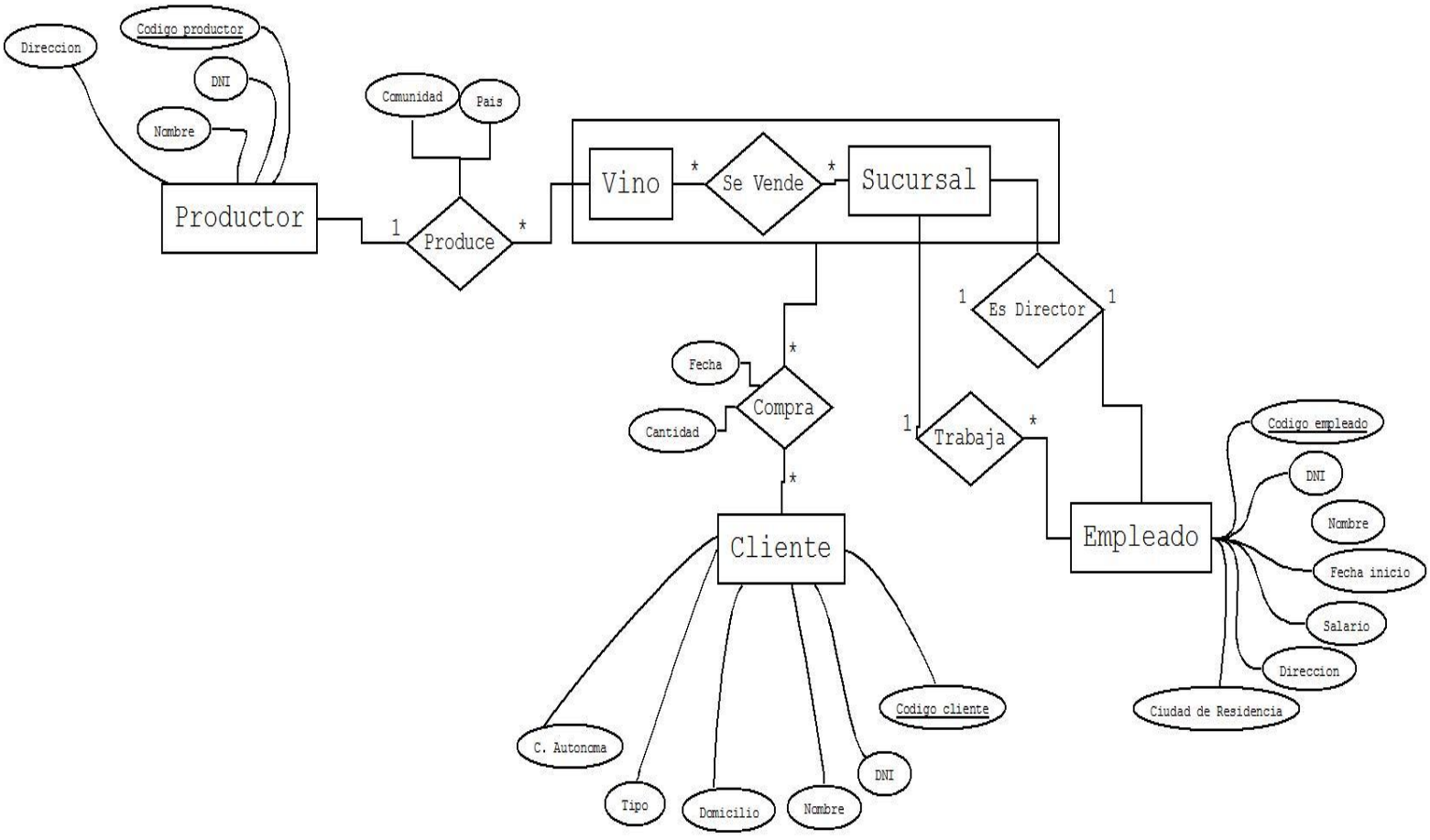
Nos encontramos con que presentaba dos defectos muy importantes: No relaciona al cliente, ni al suministro con el vino.

- Segundo diagrama que valoramos:



Nos encontramos con que a pesar de relacionar el suministro con el vino, seguía sin relacionar este con el cliente.

1.2 Diagrama E/R Final



Explicación del diagrama E/R:

Un empleado trabaja en una sucursal, y en una sucursal trabajan muchos empleados, por eso es una relación de uno a muchos.

Una sucursal tiene un director y solo uno, y un director solo puede ser director de una sucursal, esta relación es de uno a uno.

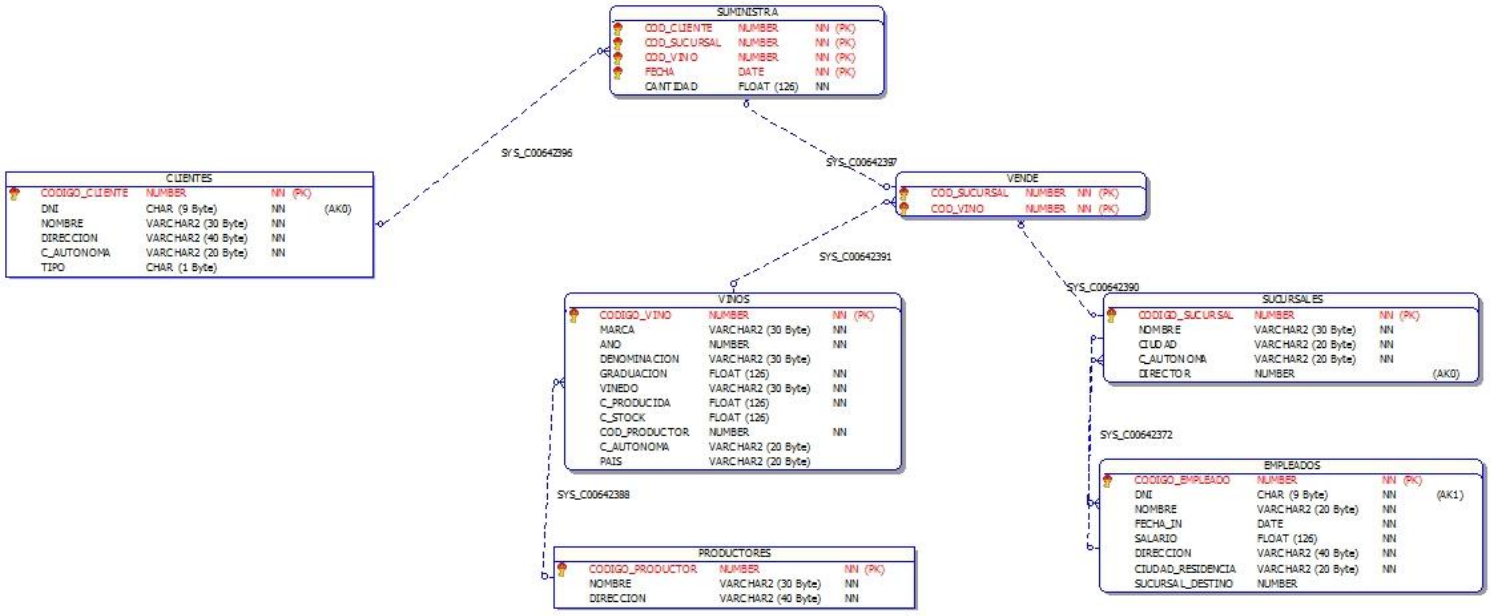
Un productor produce muchos vinos, pero un vino solo puede ser producido por un productor, esta relación es de uno a muchos. Esta relación tiene como atributos la comunidad y el país en el que se produce dicho vino.

Un vino se puede vender en muchas sucursales, y una sucursal puede vender varios vinos, por eso esta relación es de muchos a muchos.

De la relación entre vinos y sucursales hacemos una agregación, y esta la relacionamos con los clientes, de manera que un cliente puede comprar muchos vinos en una sucursal, y una sucursal que vende un vino se lo puede vender a varios clientes, por lo que es una relación de muchos a muchos. Esta relación tiene dos atributos: cantidad y fecha del suministro.

1.3 Tablas resultantes del diagrama E/R

- Diagrama de tablas



- Tablas resultantes de las entidades:

Entidad: Productores	
Datos de Productores	
Atributo	**Descripción**
CODIGO_PRODUCTOR	Código que identifica a los productores
NOMBRE	Nombre del productor
DIRECCIÓN	Dirección del productor
Claves Candidatas	
CÓDIGO_PRODUCTOR	

Entidad: Vinos	
Datos de Vinos	
Atributo	**Descripción**
CODIGO_VINO	Código que identifica a cada vino
MARCA	Marca que representa al vino
ANO	Año de cosecha
DENOMINACION	Procedencia del vino
GRADUACION	Cantidad de alcohol que tiene el vino
VINEDO	Es una plantación para la producción de vino.
C_PRODUCIDA	Cantidad de vino producida
C_STOCK	Cantidad de vino que esta disponible para su venta
COD_PRODUCTOR	Código del productor que fabrica el vino
C_AUTONOMA	Comunidad autónoma de procedencia
PAIS	País de procedencia
Claves Candidatas	
CODIGO_VINO	
Restricciones adicionales	
C_PRODUCIDA=>0 C_STOCK=>0 COD_PRODUCTOR (FK PRODUCTOR)	

Entidad: Sucursales	
Datos de Sucursales	
Atributo	**Descripción**
CODIGO _SUCURSAL	Código de la sucursal
NOMBRE	Nombre de la sucursal
CIUDAD	Ciudad de la sucursal
C_AUTONOMA	Comunidad autónoma de la sucursal
DIRECTOR	Director de la sucursal
Claves Candidatas	
CODIGO_SUCURSAL	
Restricciones adicionales	
El director de una sucursal es único, por lo que no puede haber un mismo director para varias sucursales. DIRECTOR(FK EMPLEADOS)	

Entidad: Clientes	
Datos de Clientes	
Atributo	**Descripción**
CODIGO_CLIENTE	Código del cliente
DNI	Documento nacional de identidad
NOMBRE	Nombre del cliente
DIRECCIÓN	Dirección del cliente
C_AUTONOMA	Comunidad autónoma en la que vive el cliente
TIPO	Tipo de cliente
Claves Candidatas	
CODIGO_CLIENTE	
Restricciones adicionales	
El tipo del cliente debe ser A o B o C. El DNI debe ser único, no puede haber dos iguales.	

Entidad: Empleados	
Datos de Empleados	
Atributo	**Descripción**
CODIGO_EMPLEADO	Código del empleado
DNI	Documento nacional de identidad
NOMBRE	Nombre del empleado
FECHA_IN	Fecha de inicio del contrato
SALARIO	Salario del empleado
DIRECCION	Dirección del empleado
CIUDAD_RESIDENCIA	Ciudad donde vive el empleado
SUCURSAL_DESTINO	Sucursal en la que trabaja
Claves Candidatas	
CODIGO_EMPLEADO	
Restricciones adicionales	
El DNI deber ser único, no puede haber dos iguales. SUCURSAL_DESTINO(FK SUCURSALES)	

- Tablas resultantes de las relaciones:

Entidad: Vende	
Datos de Vende	
Entidad	**Cardinalidad**
Sucursal	*
Vino	*
Claves Candidatas	
Cod_sucursal(sucursal) Cod_vino(vinos)	

Entidad: Suministra	
Datos de Suministra	
Entidad	**Cardinalidad**
Vende	*
Clientes	*
Claves Candidatas	
(Cod_sucursal,Cod_vino)(Vende) Cod_cliente(Clientes)	

- Relaciones entre entidades, a las que no les ha hecho falta crearles tablas:

La relación entre sucursales y empleados es de uno a muchos, por lo que no es necesario crear una tabla para esta relación. Simplemente para tenerla en nuestra BD, ponemos en la tabla empleados un atributo que dice la sucursal en la que trabaja.

De la relación entre sucursal y qué empleado dirige esa sucursal, simplemente hemos puesto en la tabla sucursales un atributo que referencia al empleado que la dirige.

En la relación produce entre vinos y productores, lo mismo que entre sucursales y empleados, en vinos tenemos un atributo que nos dice qué productor lo produce.

- Relaciones entre entidades, a las que sí ha hecho falta crearles tablas:
    - Vende: de la relación vende entre vinos y sucursales, al ser una relación de muchos a muchos hemos tenido que crear la tabla vende en la que queda reflejada esta relación.
    - Suministra: igualmente de la relación entre clientes y la agregación vinos-sucursales, hemos creado la tabla suministra que refleja todos los suministros de los vinos por las sucursales hacia los clientes, a la que le hemos añadido los atributos de la relación: fecha y cantidad

1.4 Posibles alternativas al diagrama E/R elegido

De las relaciones entre productor-vino, empleado-sucursal, director-sucursal, no ha habido ninguna duda, y desde el principio vimos que quedaban bien reflejadas como dicta nuestro diagrama E/R.

Respecto a la relación suministros, que guardan las compras de todos los clientes a través de sucursales de un determinado vino, hemos contemplado varias posibilidades, las cuales explicaremos a continuación:

1) Hacer una relación ternaria entre vinos-sucursales-clientes. El problema de una relación ternaria es la ineficiencia de tener una tabla con tantísimos atributos, tanto para la búsqueda de datos como para el tamaño que estos ocupan. Esta opción no fue valorada pero había que dejar claro por qué no sirve.

2) Hacer una agregación de la relación entre vinos y clientes, y esta agregación relacionarla con las sucursales. Esta opción no nos gustó por el hecho de que un cliente no compra un vino directamente, sino que lo compra en una sucursal. Por ello preferimos hacer la relación directamente entre vinos y sucursales. Esta opción es válida, pero no la vimos tan clara como la otra.

3) Hacer una agregación de la relación entre clientes y sucursales, y esta agregación relacionarla con los vinos. Esta solución parecía la más correcta al principio, por el hecho de que un cliente solo puede comprar en las sucursales de la delegación que le corresponde, pero debido a que esta restricción no iba a quedar reflejada en las tablas (sino en un disparador) acabamos viéndolo más claro si relacionábamos primero sucursales-vinos, y esta relación la relacionábamos con los clientes.

4) Definitiva Creamos la relación entre sucursales y vinos, de esta relación hacemos una agregación y la relacionamos con los clientes. Elegimos esta porque consideramos que quedaba más clara y lógica que las otras, ya que antes de comprar nada un cliente, tiene que existir la sucursal que venda el vino

2. Implementación del diseño

2.1 Restricciones de integridad en las tablas

- Tabla clientes:

```
CREATE TABLE clientes(
codigo_cliente INTEGER PRIMARY KEY NOT NULL,
dni CHAR(9) NOT NULL UNIQUE,
nombre VARCHAR2(30) NOT NULL,
direccion VARCHAR2(40) NOT NULL,
c_autonomaVARCHAR2(20) NOT NULL,
tipo CHAR(1) CHECK (tipo in ('A', 'B', 'C'))
);
```

Restricciones de integridad:

-el tipo del cliente debe ser A o B o C.

-el dni debe ser único, no puede haber dos iguales.

- Tabla empleados:

```
CREATE TABLE empleados(
codigo_empleado INT NOT NULL PRIMARY KEY,
dni CHAR(9) NOT NULL UNIQUE,
nombre VARCHAR2(20) NOT NULL,
fecha_in DATE NOT NULL,
salario FLOAT NOT NULL CHECK (salario>=0),
direccion VARCHAR2(40) NOT NULL,
ciudad_residenciaVARCHAR2(20) NOT NULL,
sucursal_destino INT,
 FOREIGN KEY (sucursal_destino) REFERENCES sucursales(codigo_sucursal)
);
```

Restricciones de integridad:

-el dni deber ser único, no puede haber dos iguales.

-la sucursal_destino esta referenciada de la tabla sucursales, por lo que debe existiruna sucursal con ese código.

- Tabla vinos:

```
CREATE TABLE vinos(
codigo_vino INT NOT NULL PRIMARY KEY,
marca VARCHAR2(30) NOT NULL,
ano INT NOT NULL,
denominacion VARCHAR2(30),
graduacion FLOAT NOT NULL,
vinedo VARCHAR2(30) NOT NULL,
c_producida FLOAT NOT NULL CHECK (0<=c_producida),
c_stock FLOAT CHECK (0<=c_stock),
cod_productor INT NOT NULL,
c_autonomaVARCHAR2(20),
pais VARCHAR2(20),
FOREIGN KEY (cod_productor) REFERENCES productores(codigo_productor)
);
```

Restricciones de integridad:

-la c_producida y la c_stock no pueden ser negativas.
-el cod_productor esta referenciado de la tabla productores por lo que debe existir un productor con ese código.

- Tabla sucursales:

```
CREATE TABLE sucursales(
codigo_sucursal INT NOT NULL PRIMARY KEY,
nombre VARCHAR2(30) NOT NULL,
ciudad VARCHAR2(20) NOT NULL,
c_autonomaVARCHAR2(20) NOT NULL,
director INT UNIQUE,
 FOREIGN KEY (director) REFERENCES empleados(codigo_empleado)
);
```

Restricciones de integridad:

-el director de una sucursal es único, por lo que no puede haber un mismo director para varias sucursales.
-el director esta referenciado de la tabla empleados por lo que debe existir un empleado con ese código.

- Tabla productores:

```
CREATE TABLE productores(
codigo_productor INTEGER NOT NULL PRIMARY KEY,
nombre VARCHAR2(30) NOT NULL,
direccion VARCHAR2(40) NOT NULL
);
```

Restricciones de integridad: (ninguna destacable)

- Tabla vende (relación entre sucursales y vinos):

```
CREATE TABLE vende(
cod_sucursal INT,
cod_vino INT,
 PRIMARY KEY (cod_sucursal, cod_vino),
FOREIGN KEY (cod_sucursal) REFERENCES sucursales(codigo_sucursal),
 FOREIGN KEY (cod_vino) REFERENCES vinos(codigo_vino)
);
```

Restricciones de integridad:

-el cod_sucursal esta referenciado de la tabla sucursales por lo que debe existir una sucursal con ese código.

-el cod_vino esta referenciado de la tabla vinos por lo que debe existir un vino con ese código.

- Tabla suministra:

```
CREATE TABLE suministra(
cod_cliente INT,
cod_sucursal INT,
cod_vino INT,
fecha DATE NOT NULL,
cantidad FLOAT NOT NULL CHECK (cantidad>0),
 FOREIGN KEY (cod_cliente) REFERENCES clientes(codigo_cliente),
 FOREIGN KEY (cod_sucursal, cod_vino) REFERENCES vende(cod_sucursal, cod_vino),
PRIMARY KEY (cod_cliente, cod_sucursal, cod_vino, fecha)
);
```

Restricciones de integridad:

-el cod_cliente esta referenciado de la tabla clientes por lo que debe existir un cliente con ese código.

-el cod_sucursal y el cod_vinoestan referenciados de la tabla vende por lo que debe existir una sucursal con ese código que venda un vino con ese codigo.

-la cantidad suministrada no puede ser negativa.

Nota: las restricciones obvias como las de primarykey, y las de notnull no las hemos comentado aunque estén en las tablas.

2.2 Restricciones de integridad en los disparadores

- Trigger para empleados:

```
CREATE OR REPLACE TRIGGER salario_empleado
 BEFORE UPDATE OF salario ON empleados FOR EACH ROW
BEGIN
IF(:old.salario> :new.salario) THEN
dbms_output.put_line('Estas intentando reducirle el sueldo a tu empleado en tiempos de
crisis, accion no permitida');
raise_application_error(-20008, 'No le puede reducir el sueldo al empleado');
END IF;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
dbms_output.put_line('No se encontro el empleado que quiere modificar');
END;
```

Restricciones de integridad:

-al actualizar el salario de un empleado, este no puede disminuir.

- Trigger para sucursales:

```
CREATE OR REPLACE TRIGGER director_sucursal
 BEFORE UPDATE OF director OR INSERT ON sucursales FOR EACH ROW
DECLARE
cont INT;
BEGIN
  IF INSERTING THEN
     SELECT COUNT(*) INTO cont FROM empleados WHERE (:new.codigo_sucursal =
empleados.sucursal_destino AND :new.director = empleados.codigo_empleado);
     IF (cont = 0 AND :new.director IS NOT NULL) THEN
raise_application_error(-20009, 'El director de la sucursal debe ser un empleado de esa
sucursal');
END IF;
  ELSE
     SELECT COUNT(*) INTO cont FROM empleados WHERE (:old.codigo_sucursal =
empleados.sucursal_destino AND :new.director = empleados.codigo_empleado);
     IF (cont = 0 AND :new.director IS NOT NULL) THEN
raise_application_error(-20009, 'El director de la sucursal debe ser un empleado de esa
sucursal');
END IF;
  END IF;
END;
```

Restricciones de integridad:

-El director de una sucursal debe ser un empleado de esa sucursal.

- Trigger para empleados:

```
CREATE OR REPLACE TRIGGER insertar_empleado
 BEFORE INSERT ON empleados FOR EACH ROW
DECLARE
cont INT;
BEGIN
  SELECT COUNT(*) INTO cont FROM sucursales WHERE
(codigo_sucursal=:new.sucursal_destino);
IF (cont = 0) THEN
raise_application_error(-20029, 'Para introducir un empleado, este debe trabajar en una
sucursal que exista, y no existe la sucursal indicada');
  END IF;
END;
```

Restricciones de integridad:

-La sucursal donde trabaja un empleado debe existir.

- Trigger para vinos:

```
CREATE OR REPLACE TRIGGER borrar_vino
 BEFORE DELETE OR INSERT ON vinos FOR EACH ROW
DECLARE
cont INT;
BEGIN
 IF DELETING THEN
   SELECT count(*) INTO cont FROM suministra WHERE (suministra.cod_vino =
:old.codigo_vino);
IF (cont> 0) THEN
raise_application_error(-20020, 'No puede borrar este vino, porque ha sido suministrado');
END IF;
 ELSE
:new.c_stock := :new.c_producida;
   SELECT count(*) INTO cont FROM productores WHERE (productores.codigo_productor =
:new.cod_productor);
IF (cont = 0) THEN
raise_application_error(-20020, 'No puede insertar este vino porque el productor de este vino
no exite.');
END IF;
 END IF;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
dbms_output.put_line('No se encontro el vino que quiere modificar');
END;
```

Restricciones de integridad:

-Para borrar un vino este no puede haber sido suministrado nunca.
-Al insertar un vino pongo su c_stock igual a su c_producida.
-El productor de un vino debe existir.

- Trigger para productores:

```
CREATE OR REPLACE TRIGGER borrar_productor
 BEFORE DELETE ON productores FOR EACH ROW
DECLARE
cont INT;
BEGIN
  SELECT COUNT(*) INTO cont FROM suministra, vinos WHERE (vinos.cod_productor =
:old.codigo_productor AND suministra.cod_vino = vinos.codigo_vino);
IF (cont> 0) THEN
raise_application_error(-20021, 'No puede borrar este productor, porque ha realizado
suministros');
END IF;
EXCEPTION
 WHEN NO_DATA_FOUND THEN
dbms_output.put_line('No se encontro el empleado que quiere modificar');
-- WHEN OTHERS THEN
--   SHOW ERROR TRIGGER borrar_vino;
END;
```

Restricciones de integridad:

-Para borrar un productor este no puede haber suministrado un vino suyo nunca.

- Trigger para suministra:

```
create or replace
TRIGGER suministro_posible
 BEFORE INSERT OR UPDATE OR DELETE ON suministra FOR EACH ROW
DECLARE
cont INT;
stock_futurovinos.c_stock%TYPE;
suma_suministrossuministra.cantidad%TYPE;
s_comunidadsucursales.c_autonoma%TYPE;
c_comunidadclientes.c_autonoma%TYPE;
permitido BOOLEAN;
BEGIN
  IF INSERTING THEN
--Para suministrar un vino este debe existir
SELECT COUNT(*) INTO cont FROM vinos WHERE (:new.cod_vino = vinos.codigo_vino);
IF (cont = 0) THEN
raise_application_error(-20022, 'El vino que quiere suministrar no exite.');
    END IF;

    --compruebo que ese cliente no tenga suministros mas actuales
SELECT COUNT(*) INTO cont FROM suministra WHERE (:new.cod_cliente =
suministra.cod_cliente AND :new.fecha<suministra.fecha);
IF (cont> 0) THEN
raise_application_error(-20022, 'Este cliente tiene suministros mas actuales, introduzca una
fecha valida');
    END IF;
```

```
    --Resto al stock de vinos la cantidad suministrada, y compruebo si hay suficiente stock para el suministro
SELECT (c_stock) INTO stock_futuro FROM vinos WHERE (vinos.codigo_vino= :new.cod_vino);
stock_futuro:= stock_futuro - :new.cantidad;
    IF (stock_futuro< 0) THEN
raise_application_error(-20022, 'NO hay suficiente vino en el stock para tramitar este suministro');
    END IF;

    -- Compruebo si ese cliente esta comprando en una de las sucursales que pertenecen a la delegacion de su comunidad
                SELECT c_autonoma INTO c_comunidad FROM clientes WHERE (clientes.codigo_cliente= :new.cod_cliente);
                SELECT c_autonoma INTO s_comunidad FROM sucursales WHERE ( sucursales.codigo_sucursal = :new.cod_sucursal);

permitido := FALSE;
                IF ((c_comunidad = 'Andalucia' OR c_comunidad = 'Canarias' OR c_comunidad = 'Ceuta' OR c_comunidad = 'Extremadura' OR c_comunidad = 'Melilla') AND (s_comunidad = 'Andalucia' OR s_comunidad = 'Canarias' OR s_comunidad = 'Ceuta' OR s_comunidad = 'Extremadura' OR s_comunidad = 'Melilla')) THEN
permitido := TRUE;
    ELSIF ((c_comunidad = 'Aragon' OR c_comunidad = 'Castilla-Leon' OR c_comunidad = 'Castilla-La Mancha' OR c_comunidad = 'La Rioja' OR c_comunidad = 'Madrid') AND (s_comunidad = 'Aragon' OR s_comunidad = 'Castilla-Leon' OR s_comunidad = 'Castilla-La Mancha' OR s_comunidad = 'La Rioja' OR s_comunidad = 'Madrid')) THEN
                        permitido := TRUE;
                ELSIF ((c_comunidad = 'Asturias' OR c_comunidad = 'Cantabria' OR c_comunidad = 'Galicia' OR c_comunidad = 'Navarra' OR c_comunidad = 'Pais Vasco') AND (s_comunidad = 'Asturias' OR s_comunidad = 'Cantabria' OR s_comunidad = 'Galicia' OR s_comunidad = 'Navarra' OR s_comunidad = 'Pais Vasco')) THEN
                        permitido := TRUE;
                ELSIF ((c_comunidad = 'Cataluna' OR c_comunidad = 'Baleares' OR c_comunidad = 'Murcia' OR c_comunidad = 'Pais Valenciano') AND (s_comunidad = 'Cataluna' OR s_comunidad = 'Baleares' OR s_comunidad = 'Murcia' OR s_comunidad = 'Pais Valenciano')) THEN
                        permitido := TRUE;
                ELSIF (permitido = FALSE) THEN
raise_application_error(-20012,'El cliente esta intentando comprar en una sucursal que no pertenece a la delegacion que le corresponde.');
                END IF;

    --compruebo si ese cliente tiene otro suministro de ese mismo vino en esa misma sucursal ese mismo dia
SELECT COUNT(*) INTO cont FROM suministra WHERE (:new.cod_cliente = suministra.cod_cliente AND :new.fecha = suministra.fecha AND :new.cod_sucursal = suministra.cod_sucursal AND :new.cod_vino = suministra.cod_vino);
IF (cont> 0) THEN  --si esto ocurre, sumo las cantidades de ambos suministros y pongo esa cantidad en la tupla que vamos a insertar, y elimino la tupla que habia
SELECT (cantidad) INTO suma_suministros FROM suministra WHERE (:new.cod_cliente = suministra.cod_clienteAND :new.fecha = suministra.fecha AND :new.cod_sucursal = suministra.cod_sucursal AND :new.cod_vino = suministra.cod_vino);
```

```
suma_suministros:=suma_suministros+ :new.cantidad;
DELETE FROM suministra WHERE (:new.cod_cliente = suministra.cod_clienteAND :new.fecha =
suministra.fecha AND :new.cod_sucursal = suministra.cod_sucursal AND :new.cod_vino =
suministra.cod_vino);
:new.cantidad := suma_suministros;
    END IF;
    --Le resto al stock del vino la cantidad que hemos suministrado
UPDATE vinos SET c_stock=stock_futuro WHERE (vinos.codigo_vino= :new.cod_vino);

ELSIF UPDATING THEN
    --Si cambia la cantidad del suministro, resto al stock de vino la nueva cantidad y le sumo la
cantidad que tenía el suministro
suma_suministros:= :new.cantidad - :old.cantidad;
SELECT (c_stock) INTO stock_futuro FROM vinos WHERE (vinos.codigo_vino= :new.cod_vino);
--Compruebo si hay suficiente stock
IF ( suma_suministros>stock_futuro ) THEN
raise_application_error(-20022, 'NO hay suficiente vino en el stock para tramitar este
suministro (update)');
END IF;
stock_futuro:= stock_futuro - suma_suministros;
    UPDATE vinos SET c_stock=stock_futuro WHERE (vinos.codigo_vino= :new.cod_vino);
ELSE
   --Si borro un suministro, entonces tengo que sumar al stock de vinos la cantidad que se
habia suministrado
SELECT (c_stock) INTO stock_futuro FROM vinos WHERE (vinos.codigo_vino= :old.cod_vino);
stock_futuro:= stock_futuro+ :old.cantidad;
   UPDATE vinos SET c_stock=stock_futuro WHERE (vinos.codigo_vino= :old.cod_vino);

END IF;
END;
```

Restricciones de integridad:

-Para suministrar un vino este debe existir.

-Compruebo que ese cliente no tenga suministros más actuales, en tal caso no se puede insertar el suministro.

-Resto al stock de vinos la cantidad suministrada, y compruebo si hay suficiente stock para el suministro, en caso de que no haya suficiente entonces no se podrá insertar el suministro.

-Compruebo si ese cliente está comprando en una de las sucursales que pertenecen a la delegación de su comunidad, si no es así entonces no se puede insertar el suministro.

-Compruebo si ese cliente tiene otro suministro de ese mismo vino en esa misma sucursal ese mismo día, en ese caso sumo las cantidades de ambos suministros y lo dejo como un registro.

Modificaciones de otras tablas:

-Le resto al stock del vino la cantidad que hemos suministrado.

-Si cambia la cantidad del suministro, resto al stock de vino la nueva cantidad y le sumo la cantidad que tenía el suministro

-Si borro un suministro, entonces tengo que sumar al stock de vinos la cantidad que se había suministrado.

3.Implementación de actualizaciones

```
--1 Dar de alta a un empleado

CREATE OR REPLACE PROCEDURE alta_empleado (cod_empleado IN INT, dni IN CHAR, nombre
IN VARCHAR2, fecha_in IN DATE, salario IN FLOAT, direccion IN VARCHAR2, ciudad IN
VARCHAR2, cod_sucursal IN INT)
IS
BEGIN
   INSERT INTO empleados VALUES (cod_empleado, dni, nombre, fecha_in, salario, direccion,
ciudad, cod_sucursal);

END;
/

--2 Dar de baja a un empleado
--Si el empleado es el director de una sucursal entonces pongo el director de la sucursal a NULL

CREATE OR REPLACE PROCEDURE baja_empleado (cod_empleado IN INT)
IS
BEGIN
   UPDATE sucursales SET director=NULL WHERE director=cod_empleado;
DELETE FROM empleados WHERE codigo_empleado=cod_empleado;
END;
/

--3 Modificar el salario de un empleado

CREATE OR REPLACE PROCEDURE salario_empleado (cod_empleado IN INT, salario_nuevo IN
FLOAT)
IS
BEGIN
   UPDATE empleados SET salario=salario_nuevo WHERE codigo_empleado=cod_empleado;
END;
/

--4 Dar de alta una nueva sucursal

CREATE OR REPLACE PROCEDURE alta_sucursal (cod_sucursal IN INT, nombre IN VARCHAR2,
ciudad IN VARCHAR2, comunidad IN VARCHAR2)
IS
BEGIN
   INSERT INTO sucursales VALUES (cod_sucursal, nombre, ciudad, comunidad, NULL);
END;
/
```

```
--5 Cambiar director de una sucursal

CREATE OR REPLACE PROCEDURE director_sucursal (cod_sucursal IN INT, cod_director IN INT)
IS
BEGIN
   UPDATE sucursales SET director=cod_directorWHERE  (codigo_sucursal=cod_sucursal);
END;
/

--6 Dar de alta a un nuevo cliente

CREATE OR REPLACE PROCEDURE alta_cliente (cod_cliente IN INT, dni IN CHAR, nombre IN
VARCHAR2, direccion IN VARCHAR2, tipo IN CHAR, comunidad IN VARCHAR2)
IS
BEGIN
  INSERT INTO clientes VALUES (cod_cliente, dni, nombre, direccion, comunidad, tipo);

END;
/

--7 Dar de alta o actualizar un suministro

CREATE OR REPLACE PROCEDURE alta_suministro (cliente IN INT, sucursal IN INT, vino IN INT,
fecha_nueva IN DATE, cantidad_nueva IN FLOAT)
IS
cont INT;
BEGIN
      INSERT INTO suministra VALUES (cliente, sucursal, vino, fecha_nueva, cantidad_nueva);
END;
/

--8 Devolver suministro
--Si no se especifica la fecha se borraran todo los suministros

CREATE OR REPLACE PROCEDURE devolver_suministro (cliente IN INT, sucursal IN INT, vino IN
INT, fecha_opcional IN DATE DEFAULT NULL)
IS
BEGIN
IF(fecha_opcional IS NULL) THEN
       DELETE FROM suministra WHERE (cod_cliente=cliente AND cod_sucursal=sucursal AND
cod_vino=vino);
    ELSE
       DELETE FROM suministra WHERE (cod_cliente = cliente AND cod_sucursal=sucursal AND
cod_vino=vino AND fecha=fecha_opcional);
    END IF;
END;
/
```

```
--9 Dar de alta un nuevo vino

createorreplace
PROCEDURE alta_vino (vino IN INT, marca IN VARCHAR2, ano_nuevo IN INT, denominacion in
VARCHAR2 DEFAULT NULL, grado IN FLOAT, vinedo_pro IN VARCHAR2, c_producida IN FLOAT,
productor IN INT, c_autonoma IN VARCHAR2,pais IN VARCHAR2)
IS
BEGIN
   INSERT INTO vinos VALUES (vino, marca, ano_nuevo, denominacion, grado, vinedo_pro,
c_producida, c_producida, productor, c_autonoma, pais);

END;
/


--10 Dar de baja un vino

CREATE OR REPLACE PROCEDURE baja_vino (cod_vino IN INT)
IS
BEGIN
    DELETE FROM vinos WHERE codigo_vino=cod_vino;
END;
/

--11 Dar de alta un productor

CREATE OR REPLACE PROCEDURE alta_productor(productor IN INT, nombre IN VARCHAR,
direccion IN VARCHAR)
IS
BEGIN
   INSERT INTO productores VALUES (productor, nombre, direccion);
END;
/


--12 Dar de baja un productor
--Si se puede dar de baja al productor, elimina todos sus vinos

CREATE OR REPLACE PROCEDURE baja_productor(productor IN INT)
IS
BEGIN
   DELETE FROM vinos WHERE (cod_productor = productor);
   DELETE FROM productores WHERE (codigo_productor=productor);
END;
/
```

4.Implementación de consultas

- ## Macro1:

-- 1 Listar los clientes(nombre y direccion) de Andalucia o Castilla-La Mancha y las sucursales(nombre y ciudad) a los que se le ha suministrado el vino "Tablas de Daimiel" entre las fechas dadas

```
-- Lectura
acceptfechaIni DATE FORMAT dd/mm/yyyyprompt 'Fecha de inicio(dd/mm/yyyy):';
acceptfechaFin DATE FORMAT dd/mm/yyyyprompt 'Fecha final(dd/mm/yyyy):';
-- Consulta
SELECT c.nombre, c.direccion, s.nombre, s.ciudad
FROM
        (SELECT * FROM clientes WHERE c_autonoma IN ('Andalucia', 'Castilla-La Mancha')) c,
        sucursales s,
        (SELECT * FROM suministra WHERE cod_vino IN (SELECT codigo_vino FROM vinos
WHERE marca='Tablas de Daimiel') AND (fecha BETWEEN '&fechaIni' AND '&fechaFin')) sm
WHERE
        c.codigo_cliente=sm.cod_cliente AND s.codigo_sucursal=sm.cod_sucursal;
```

- ## Macro2:

--2 Recibo el cod_productor y muestro: codigo, marca, año cosecha de los vinos que el produce, y tambien la cantidad total suministrada de cada vino a clientes de Extremadura, Baleares, Pais Vasco

```
-- Lectura
acceptcod_product INT prompt 'Diga el codigo del productor:';

-- Consulta
SELECT v.codigo_vino, v.marca, v.ano, cantidad_total
FROM (SELECT cod_vino, SUM(cantidad) cantidad_total FROM suministra WHERE cod_cliente
IN (SELECT codigo_cliente FROM clientes WHERE c_autonoma IN
        ('Extremadura', 'PaisValenciano', 'Baleares')) GROUP BY cod_vino) sm,
   (SELECT * FROM vinos WHERE cod_productor = &cod_product) v
WHERE
sm.cod_vino=v.codigo_vino;
```

- **Macro3:**

--3 Recibo el codigo de una sucursal, muestro codigo, nombre de sus clientes, su tipo,la cantidad media de vinos de importacion que se le ha suministrado a cada uno de ellos.

```
-- Lectura
acceptcod_sucur INT prompt 'Introduzca el codigo de la sucursal:';
-- Consulta
SELECT
        c.codigo_cliente, c.nombre, c.tipo, avg(sm.cantidad)
FROM
        clientes c,
        (SELECT * FROM suministra WHERE (cod_sucursal = &cod_sucur AND cod_vino IN
      (SELECT codigo_vino FROM vinos WHERE (vinos.pais<> 'Espana')))) sm
WHERE c.codigo_cliente=sm.cod_cliente
GROUP BY c.codigo_cliente, c.nombre, c.tipo;
```