PRÁCTICAS BDD DE UNA CADENA HOTELERA

Félix Ramírez García
Javier Gómez Luzón
2017-2018

UNIVERSIDAD
DE GRANADA

Departamento de
Lenguajes y Sistemas
Informáticos

PRÁCTICA 1

DISEÑO CONCEPTUAL Y LÓGICO GLOBAL DE LA BASE DE DATOS

El diseño se hace en base a las necesidades de una multinacional de cadenas hoteleras , donde se van a guardar datos de hoteles , empleados , clientes y proveedores.

El diagrama E/R es el que muestra la figura 1:

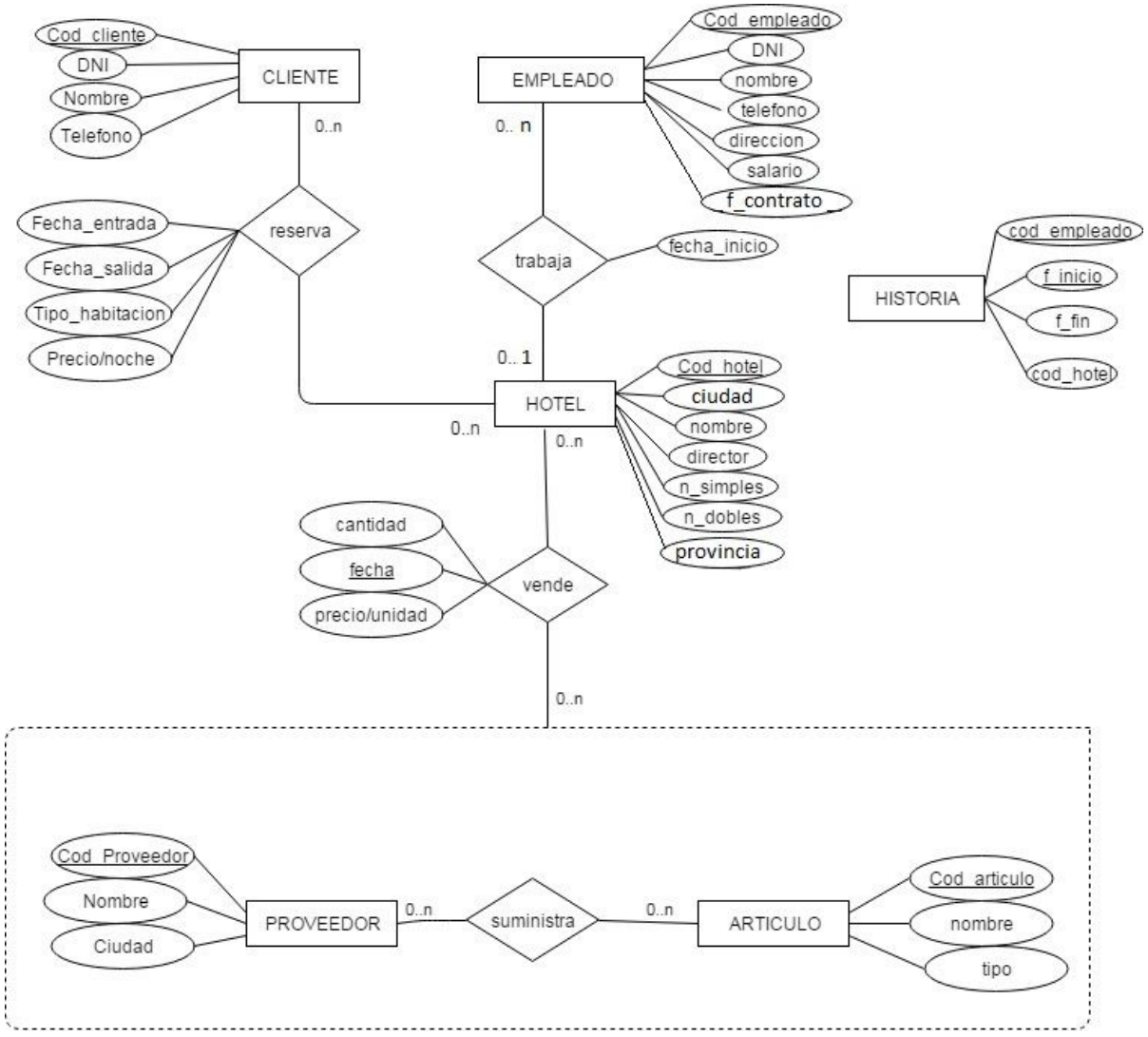

Figura 1 : Diseño E/R de una cadena hotelera

Donde las entidades del diagrama son:

CLIENTE
EMPLEADO
HOTEL
PROVEEDOR
ARTÍCULO
HISTORIA.

Y las relaciones del diagrama son:

-RESERVA, relaciona las entidades CLIENTE ,HOTEL.
-TRABAJA , relaciona las entidades EMPLEADO ,HOTEL
-SUMINISTRA , relaciona las entidades PROVEEDOR, ARTÍCULO.
-VENDE, relaciona la entidad HOTEL con la agregación de PROVEEDOR SUMINISTRA ARTÍCULO.

Otra opción de diseño podría ser no usar una agregación entre PROVEEDOR y ARTÍCULO, y conectar directamente PROVEEDOR con HOTEL a través de vende, pero esta opción es menos recomendable para garantizar una mayor independencia e integridad de los datos, ya que con la agregación en el paso a tablas, la relación VENDE tendria tambien el codigo del articulo .

Las tablas resultantes del diagrama entidad/relación de la Figura1 son:

-HOTEL (cod_hotel, nombre, n_dobles , n_simples, ciudad , provincia , director)
-EMPLEADO (cod_empleado, DNI, nombre, direccion , telefono, f_contrato , salario ,cod_hotel , fecha_inicio)
-CLIENTE (cod_cliente, DNI, nombre , telefono)
-RESERVA (cod_cliente, cod_hotel , fecha_entrada, fecha_salida , tipo_habitacion , precio)
-PROVEEDOR (cod_proveedor, nombre, ciudad)
-ARTÍCULO (cod_articulo , nombre, tipo, cod_proveedor)
-VENDE (cod_articulo , cod_proveedor, cod_hotel, fecha, cantidad , precio)
-HISTORIA (cod_empleado ,fecha_inicio,cod_hotel, fecha_fin)

PRÁCTICA 2

DISEÑO DE LA FRAGMENTACIÓN Y DE LA ASIGNACIÓN

Las tablas que deben fragmentarse son :

1. Tabla HOTEL

Fragmentamos la tabla **HOTEL** con una fragmentación horizontal primaria por ubicación, puesto que el enunciado nos indica que solamente tenemos 4 localidades con hoteles , por tanto es necesaria la fragmentación.

P = { P1: Granada , P2: Almería , P3: Jaén , P4: Málaga , P5: Cadiz ,P6: Sevilla ,P7: Córdoba ,P8: Huelva }
El total de las conjunciones que tenemos sería 2 ^ 8 conjunciones, que serían 256.

Conjunciones verdaderas
y1: ¬P1 ^ ¬P2 ^ ¬P3 ^ ¬P4 ^ ¬P5 ^ ¬P6 ^ ¬P7 ^ P8 (Huelva)
y2: ¬P1 ^ ¬P2 ^ ¬P3 ^ ¬P4 ^ ¬P5 ^ ¬P6 ^ P7 ^ ¬P8 (Córdoba)
y3: ¬P1 ^ ¬P2 ^ ¬P3 ^ ¬P4 ^ ¬P5 ^ P6 ^ ¬P7 ^ ¬P8 (Sevilla)
y4: ¬P1 ^ ¬P2 ^ ¬P3 ^ ¬P4 ^ P5 ^ ¬P6 ^ ¬P7 ^ ¬P8 (Cadiz)
y5: ¬P1 ^ ¬P2 ^ ¬P3 ^ P4 ^ ¬P5 ^ ¬P6 ^ ¬P7 ^ ¬P8 (Málaga)
y6: ¬P1 ^ ¬P2 ^ P3 ^ ¬P4 ^ ¬P5 ^ ¬P6 ^ ¬P7 ^ ¬P8 (Jaén)
y7: ¬P1 ^ P2 ^ ¬P3 ^ ¬P4 ^ ¬P5 ^ ¬P6 ^ ¬P7 ^ ¬P8 (Almería)
y8: P1 ^ ¬P2 ^ ¬P3 ^ ¬P4 ^ ¬P5 ^ ¬P6 ^ ¬P7 ^ ¬P8 (Granada)

Los fragmentos que obtenemos son los siguientes:

Hotel1= $SL_{(y8\ v\ y6)}$ (Hotel)
Hotel2= $SL_{(y4\ v\ y1)}$ (Hotel)
Hotel3= $SL_{(y3\ v\ y2)}$ (Hotel)
Hotel4= $SL_{(y5\ v\ y7)}$ (Hotel)

La asignación de fragmentos finales es la siguiente:
El fragmento Hotel1 se va a asignar a la localidad1
El fragmento Hotel2 se va a asignar a la localidad2
El fragmento Hotel3 se va a asignar a la localidad3
El fragmento Hotel4 se va a asignar a la localidad4

Reconstrucción
Hotel = Hotel1 UN Hotel2 UN Hotel3 UN Hotel4

2 .Tabla RESERVA

La tabla de RESERVA será fragmentada con una fragmentación horizontal derivada de la tabla HOTEL y lo haremos por código de hotel, puesto que cada reserva se hará en en un hotel de una determinada comunidad autónoma.

Reserva1 = Reserva $SJN_{cod\_hotel=\ cod\_hotel}$ (Hotel1)
Reserva2 = Reserva $SJN_{cod\_hotel=\ cod\_hotel}$ (Hotel2)
Reserva3 = Reserva $SJN_{cod\_hotel=\ cod\_hotel}$ (Hotel3)
Reserva4 = Reserva $SJN_{cod\_hotel=\ cod\_hotel}$ (Hotel4)

Asignación de fragmentos:
El fragmento Reserva1 se va a asignar a la localidad1
El fragmento Reserva2 se va a asignar a la localidad2
El fragmento Reserva3 se va a asignar a la localidad3
El fragmento Reserva4 se va a asignar a la localidad4

Reconstrucción
Reserva = Reserva1 UN Reserva2 UN Reserva3 UN Reserva4

3. Tabla PROVEEDOR

Fragmentamos la tabla **PROVEEDOR** con una fragmentación horizontal primaria por ciudad, puesto que el enunciado nos indica que solamente tenemos 2 localidades con proveedores , por tanto es necesaria la fragmentación.

P = { P1: Sevilla, P2: Granada}
El total de las conjunciones que tenemos sería 2 ^ 2 conjunciones, que serían 4.

Conjunciones verdaderas
y1: ¬P1 ^ P2 (Granada)
y2: P1 ^ ¬P2 (Sevilla)

Los fragmentos que obtenemos son los siguientes:
Proveedor1= SL_{y2} (Proveedor)
Proveedor2= SL_{y1} (Proveedor)

La asignación de fragmentos finales es la siguiente:
El fragmento Proveedor1 se va a asignar a la localidad1 y localidad4
El fragmento Proveedor2 se va a asignar a la localidad3 y localidad2

Reconstrucción
Hotel = Proveedor1 UN Proveedor2

4. Tabla VENDE

La tabla de VENDE será fragmentada con una fragmentación horizontal derivada de la tabla HOTEL y lo haremos por código de hotel, puesto que cada venta es realizada a un hotel especifico.

Vende1= Vende $SJN_{cod\_hotel= cod\_hotel}$(Hotel1)
Vende2= Vende $SJN_{cod\_hotel= cod\_hotel}$(Hotel2)
Vende3= Vende $SJN_{cod\_hotel= cod\_hotel}$(Hotel3)
Vende4= Vende $SJN_{cod\_hotel= cod\_hotel}$(Hotel4)

Asignación de fragmentos:
El fragmento Vende1 se va a asignar a la localidad1
El fragmento Vende2 se va a asignar a la localidad2
El fragmento Vende3 se va a asignar a la localidad3
El fragmento Vende4 se va a asignar a la localidad4

Reconstrucción
Vende= Vende1 UN Vende2 UN Vende3 UN Vende4

5. Tabla EMPLEADO

La tabla de EMPLEADO será fragmentada con una fragmentación horizontal derivada de la tabla HOTEL y lo haremos por código de hotel, puesto que cada empleado trabaja en un hotel especifico

$$\text{Empleado1} = \text{Empleado} \ltimes_{\text{cod\_hotel} = \text{cod\_hotel}} (\text{Hotel1})$$
$$\text{Empleado2} = \text{Empleado} \ltimes_{\text{cod\_hotel} = \text{cod\_hotel}} (\text{Hotel2})$$
$$\text{Empleado3} = \text{Empleado} \ltimes_{\text{cod\_hotel} = \text{cod\_hotel}} (\text{Hotel3})$$
$$\text{Empleado4} = \text{Empleado} \ltimes_{\text{cod\_hotel} = \text{cod\_hotel}} (\text{Hotel4})$$

Asignación de fragmentos:
El fragmento Empleado1 se va a asignar a la localidad1
El fragmento Empleado2 se va a asignar a la localidad2
El fragmento Empeado3 se va a asignar a la localidad3
El fragmento Empleado4 se va a asignar a la localidad4

Reconstrucción
Empleado= Empleado1 UN Empleado2 UN Empleado3 UN Empleado4

6. Tabla ARTICULO

La tabla de ARTICULO será fragmentada con una fragmentación horizontal derivada de la tabla PROVEEDOR y lo haremos por código de proveedor, puesto que cada artículo lo suministra un proveedor especifico

$$\text{Articulo1} = \text{Articulo} \ltimes_{\text{cod\_proveedor} = \text{cod\_proveedor}} (\text{Proveedor1})$$
$$\text{Articulo2} = \text{Articulo} \ltimes_{\text{cod\_proveedor} = \text{cod\_proveedor}} (\text{Proveedor2})$$

Asignación de fragmentos:
El fragmento Articulo1 se va a asignar a la localidad1 y localidad4
El fragmento Articulo2 se va a asignar a la localidad2 y localidad3

Reconstrucción
Articulo= Articulo1 UN Articulo2

7. Tabla HISTORIA

La tabla de HISTORIA será fragmentada con una fragmentación horizontal derivada de la tabla EMPLEADO y lo haremos por código de empleado.

Historia1= Historia SJN $_{cod\_empleado= cod\_empleado}$(Empleado1)
Historia2= Historia SJN $_{cod\_empleado= cod\_empleado}$(Empleado2)
Historia3= Historia SJN $_{cod\_empleado= cod\_empleado}$(Empleado3)
Historia4= Historia SJN $_{cod\_empleado= cod\_empleado}$(Empleado4)

Asignación de fragmentos:
El fragmento Historia1 se va a asignar a la localidad1
El fragmento Historia2 se va a asignar a la localidad2
El fragmento Historia3 se va a asignar a la localidad3
El fragmento Historia4 se va a asignar a la localidad4

Reconstrucción
Historia= Historia1 UN Historia2 UN Historia3 UN Historia4

PRÁCTICA 3

IMPLEMENTACIÓN DEL DISEÑO

CREACIÓN DE TABLAS

```
CREATE TABLE HOTEL  (
        cod_hotel INT NOT NULL,
        nombre VARCHAR(30) NOT NULL,
        n_dobles INT NOT NULL ,
        n_simples INT NOT NULL ,
        ciudad VARCHAR(30) NOT NULL,
        provincia VARCHAR(30) NOT NULL,
        director INT NOT NULL,
        PRIMARY KEY(cod_hotel)
);


CREATE TABLE EMPLEADO  (
        cod_empleado INT NOT NULL,
        DNI INT NOT NULL,
        nombre VARCHAR(30) NOT NULL,
        telefono INT,
```

```
        direccion VARCHAR(30) ,
        salario INT NOT NULL,
        cod_hotel INT NOT NULL,
        fecha_inicio date NOT NULL,
        f_contrato date NOT NULL,
        PRIMARY KEY(cod_empleado) ,
        FOREIGN KEY(cod_hotel) REFERENCES HOTEL(cod_hotel)
);

CREATE TABLE CLIENTE(
        cod_cliente INT NOT NULL,
        DNI INT NOT NULL,
        nombre VARCHAR(30) NOT NULL,
        telefono INT,
        PRIMARY KEY(cod_cliente)
);


CREATE TABLE RESERVA(
        cod_cliente INT NOT NULL,
        cod_hotel INT NOT NULL,
        fecha_entrada DATE NOT NULL,
        fecha_salida DATE NOT NULL,
        tipo_habitacion VARCHAR(30) NOT NULL,
        precio INT NOT NULL,
        PRIMARY KEY(cod_cliente , cod_hotel , fecha_entrada),
        FOREIGN KEY (cod_cliente) REFERENCES CLIENTE (cod_cliente),
        FOREIGN KEY(cod_hotel) REFERENCES HOTEL(cod_hotel)
);


CREATE TABLE PROVEEDOR(
        cod_proveedor INT NOT NULL,
        nombre VARCHAR(30) NOT NULL,
        ciudad VARCHAR(30) NOT NULL,
        PRIMARY KEY(cod_proveedor)
);

CREATE TABLE ARTICULO(
        cod_articulo INT NOT NULL,
        cod_proveedor INT NOT NULL,
        nombre VARCHAR(30) NOT NULL,
        tipo VARCHAR(30) NOT NULL,
        PRIMARY KEY(cod_articulo , cod_proveedor),
        FOREIGN KEY (cod_proveedor) REFERENCES PROVEEDOR(cod_proveedor)
);
```

```
CREATE TABLE VENDE(
        cod_articulo INT NOT NULL,
        cod_proveedor INT NOT NULL,
        cod_hotel INT NOT NULL,
        fecha DATE NOT NULL,
        cantidad INT NOT NULL,
        precio INT NOT NULL,
        PRIMARY KEY(cod_articulo , cod_proveedor , cod_hotel, fecha),
        FOREIGN KEY (cod_articulo, cod_proveedor) REFERENCES
ARTICULO(cod_articulo, cod_proveedor),
   FOREIGN KEY (cod_hotel) REFERENCES HOTEL(cod_hotel)
);

CREATE TABLE HISTORIA(
        cod_empleado INT NOT NULL,
        cod_hotel INT NOT NULL,
        fecha_inicio DATE NOT NULL,
        fecha_fin DATE NOT NULL,
        PRIMARY KEY(cod_empleado , fecha_inicio),
        FOREIGN KEY (cod_empleado) REFERENCES EMPLEADO(cod_empleado),
        FOREIGN KEY (cod_hotel) REFERENCES HOTEL(cod_hotel)
);
```

CONCESIÓN DE PRIVILEGIOS:

```
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe1.HOTEL TO jafe2, jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe2.HOTEL TO jafe1, jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe3.HOTEL TO jafe1, jafe2, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe4.HOTEL TO jafe1, jafe2, jafe3;

GRANT SELECT, INSERT, DELETE, UPDATE ON jafe1.EMPLEADO TO jafe2, jafe3,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe2.EMPLEADO TO jafe1, jafe3,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe3.EMPLEADO TO jafe1, jafe2,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe4.EMPLEADO TO jafe1, jafe2,jafe3;

GRANT SELECT, INSERT, DELETE, UPDATE ON jafe1.CLIENTE TO jafe2,jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe2.CLIENTE TO jafe1,jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe3.CLIENTE TO jafe1,jafe2, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe4.CLIENTE TO jafe1,jafe2, jafe3;

GRANT SELECT, INSERT, DELETE, UPDATE ON jafe1.RESERVA TO jafe2,jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe2.RESERVA TO jafe1,jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe3.RESERVA TO jafe2,jafe1, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe4.RESERVA TO jafe2,jafe3, jafe1;
```

GRANT SELECT, INSERT, DELETE, UPDATE ON jafe1.PROVEEDOR TO jafe2,jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe2.PROVEEDOR TO jafe1,jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe3.PROVEEDOR TO jafe2,jafe1, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe4.PROVEEDOR TO jafe2,jafe3, jafe1;

GRANT SELECT, INSERT, DELETE, UPDATE ON jafe1.ARTICULO TO jafe2, jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe2.ARTICULO TO jafe1, jafe3, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe3.ARTICULO TO jafe2, jafe1, jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe4.ARTICULO TO jafe2, jafe3, jafe1;

GRANT SELECT, INSERT, DELETE, UPDATE ON jafe1.VENDE TO jafe2, jafe3,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe2.VENDE TO jafe1, jafe3,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe3.VENDE TO jafe2, jafe1,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe4.VENDE TO jafe2, jafe3,jafe1;

GRANT SELECT, INSERT, DELETE, UPDATE ON jafe1.HISTORIA TO jafe2, jafe3,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe2.HISTORIA TO jafe1, jafe3,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe3.HISTORIA TO jafe2, jafe1,jafe4;
GRANT SELECT, INSERT, DELETE, UPDATE ON jafe4.HISTORIA TO jafe2, jafe3,jafe1;

CREACIÓN DE VISTAS

create view vista_hotel as
*SELECT * from jafe1.HOTEL union*
*SELECT * from jafe2.HOTEL union*
*SELECT * from jafe3.HOTEL union*
*SELECT * from jafe4.HOTEL;*

create view vista_reserva as
*SELECT * from jafe1.RESERVA union*
*SELECT * from jafe2.RESERVA union*
*SELECT * from jafe3.RESERVA union*
*SELECT * from jafe4.RESERVA;*

create view vista_proveedor as
*SELECT * from jafe1.PROVEEDOR union*
*SELECT * from jafe2.PROVEEDOR union*
*SELECT * from jafe3.PROVEEDOR union*
*SELECT * from jafe4.PROVEEDOR;*

create view vista_vende as
*SELECT * from jafe1.VENDE union*

```
SELECT * from jafe2.VENDE union
SELECT * from jafe3.VENDE union
SELECT * from jafe4.VENDE;

create view vista_empleado as
SELECT * from jafe1.EMPLEADO union
SELECT * from jafe2.EMPLEADO union
SELECT * from jafe3.EMPLEADO union
SELECT * from jafe4.EMPLEADO;


create view vista_articulo as
SELECT * from jafe1.ARTICULO union
SELECT * from jafe2.ARTICULO union
SELECT * from jafe3.ARTICULO union
SELECT * from jafe4.ARTICULO;

create view vista_historia as
SELECT * from jafe1.HISTORIA union
SELECT * from jafe2.HISTORIA union
SELECT * from jafe3.HISTORIA union
SELECT * from jafe4.HISTORIA;
```

DISPARADORES

1. El número de reservas de un hotel, nunca podrá exceder su capacidad, es decir, nunca se podrá exceder el número de habitaciones sencillas, el número de habitaciones dobles y el número total de habitaciones.

```
create or replace trigger n_reservas
before insert or update of cod_hotel on  RESERVA for each row
DECLARE
  simple_hotel number;
  doble_hotel number;
  total_simple number;
  total_doble number;

  total number;
begin
  select n_simples into simple_hotel from vista_hotel where cod_hotel=:new.cod_hotel;
  select n_dobles into doble_hotel from vista_hotel where cod_hotel=:new.cod_hotel;
  select count(*) into total_simple from vista_reserva where TIPO_HABITACION='Simple'
AND cod_hotel=:new.cod_hotel;
  select count(*) into total_doble from vista_reserva where TIPO_HABITACION='Doble' AND
cod_hotel=:new.cod_hotel;
```

```
  if :new.tipo_habitacion='Simple' then
    total:=simple_hotel+1;
    if total>total_simple then
          raise_application_error(-20904, 'No se pueden superar todas las habitaciones
simples ');
          end if;
  elsif :new.tipo_habitacion='Doble' then
    total:=doble_hotel+1;
    if total>total_doble then
          raise_application_error(-20004, 'No se pueden superar todas las habitaciones dobles
reservadas');
          end if;
  end if;
end;
```

2. La fecha de entrada de un cliente en un hotel nunca podrá ser posterior a la de salida

```
create or replace trigger fecha_reserva
before insert or update of FECHA_ENTRADA, FECHA_SALIDA ON RESERVA FOR EACH
ROW
begin
  IF :new.FECHA_ENTRADA > :new.FECHA_SALIDA then
    raise_application_error(-20005, 'La fecha de entrada no puede superar a la de salida');
  end if;
end;
```

3. Un cliente nunca podrá tener más de una reserva en hoteles distintos para las mismas fechas.

```
create or replace trigger x_reservas
before insert or update of FECHA_ENTRADA, FECHA_SALIDA on RESERVA for each row
declare
          cont number;
begin
          select count(*) into cont from vista_reserva where
((:new.FECHA_ENTRADA>=:old.FECHA_ENTRADA AND
:new.FECHA_ENTRADA<=:old.FECHA_SALIDA)  or
(:new.FECHA_SALIDA>=:old.FECHA_ENTRADA AND
:new.FECHA_SALIDA<=:old.FECHA_SALIDA)) and :new.COD_HOTEL!=:old.COD_HOTEL;
          IF cont>0 then
                    raise_application_error(-20006, 'Ya tiene reservas en las mismas fechas');
          end if;
end;
```

4. **El director de un hotel es un empleado de la multinacional.**

```
create or replace trigger director_es_empleado
before insert or update of DIRECTOR on HOTEL for each row
declare
	cont number;
begin
	select count(*) into cont from vista_empleado where cod_empleado=:new.director;
	if cont=0 then
		raise_application_error(-20007, 'Ese director no es un empleado');
	end if;
end;
```

5. **Un empleado, al mismo tiempo, sólo puede ser director de un hotel.**

```
create or replace trigger un_director
before insert or update of DIRECTOR on HOTEL for each row
declare
	cont number;
begin
	select count(*) into cont from vista_hotel where  :new.director=director AND
DIRECTOR!=0;
	if cont>0 then
		raise_application_error(-20008, 'Ese empleado ya es director de un hotel');
	end if;
end;
```

6. **El hotel donde trabaja un empleado debe existir.**

```
create or replace trigger existe_hotel
before insert or update of cod_hotel on EMPLEADO for each row
declare
	aux number;
begin
	select count(*) into aux from vista_hotel where cod_hotel=:new.cod_hotel;
	if aux=0 then
		raise_application_error(-20009, 'El hotel no existe');
	end if;
end;
```

7. **El salario de un empleado nunca podrá disminuir.**

```
create or replace trigger salario_no_disminuye
before insert or update of SALARIO on EMPLEADO for each row
begin
        if :old.salario < :new.salario then
                raise_application_error(-20010, 'El salario no puede disminuir');
        end if;
end;
```

8. *La fecha de inicio de un empleado en un hotel será siempre igual o posterior a la fecha de inicio de su contrato con la multinacional.*

```
create or replace trigger fecha_contrato_posterior
before insert or update of FECHA_INICIO, F_CONTRATO on EMPLEADO for each row
begin

        if :new.fecha_inicio < :new.f_contrato then
                raise_application_error(-20011, 'La fecha de contrato debe ser igual o
superior a la fecha de inicio');
        end if;
end;
```

9. La fecha de inicio de un empleado en un hotel será siempre igual o posterior a la fecha de fin en el hotel al que estaba asignado anteriormente.

```
create or replace trigger f_nuevo_contrato_posterior1
before insert or update of FECHA_INICIO, FECHA_FIN on HISTORIA for each row
declare
        aux date;
begin
        select max(fecha_fin) into aux from vista_historia where
:new.cod_empleado=cod_empleado;
        if aux > :new.fecha_inicio then
                raise_application_error(-20012, 'La fecha de contrato debe ser igual o
superior a la fecha de fin del hotel anterior');
        end if;
end;

create or replace trigger f_nuevo_contrato_posterior2
before insert or update of FECHA_INICIO on EMPLEADO for each row
declare
        aux date;
begin
        select max(fecha_fin) into aux from vista_historia where
:new.cod_empleado=cod_empleado;
```

```
        if aux > :new.fecha_inicio then
                raise_application_error(-20112, 'La fecha de contrato debe ser igual o
superior a la fecha de fin del hotel anterior');
        end if;
end;
```

10. El tipo de un artículo será ‘A’, ‘B’, ‘C’ o ‘D’.

```
create or replace trigger tipo_articulo
before insert or update of TIPO on ARTICULO for each row
declare
        bool BOOLEAN:=FALSE;
begin
        if :new.TIPO = 'A' then
                bool:=TRUE;
         elsif :new.TIPO = 'B' then
                bool := TRUE;
        elsif :new.TIPO = 'C' then
                bool := TRUE;
        elsif :new.TIPO='D' then
                bool:=TRUE;
        end if;
        if bool = FALSE then
                raise_application_error(-20013, 'El tipo de articulo debe ser A,B,C o D');
        end if;
end;
```

11. El precio por unidad de un artículo para un suministro determinado a un hotel determinado, nunca podrá ser menor que el de ese mismo artículo en suministros anteriores a ese mismo hotel.

```
create or replace trigger precio_nuevo_mayor
before insert or update of PRECIO on VENDE for each row
declare
        aux number;
begin
        select max(precio) into aux from vista_vende where :new.cod_hotel=cod_hotel and
:new.cod_articulo=cod_articulo;
        if aux < :new.precio then
                raise_application_error(-20014, 'El precio del articulo no puede ser menor');
```

```
        end if;
end;
```

12. Un artículo sólo puede ser suministrado, como mucho, por dos proveedores, uno de Granada y otro de Sevilla

```
create or replace trigger dos_proveedores_articulo
before insert or update of COD_PROVEEDOR on ARTICULO for each row
declare
        count_proveedores number;
        lugar1 VARCHAR(30);
        lugar2 VARCHAR(30);
        our_proveedor number;
begin
        select count(*) into count_proveedores from vista_articulo where
cod_articulo=:new.cod_articulo;
        if count_proveedores=2 then
                raise_application_error(-20115, 'Solo puede haber dos proveedores uno de
Granada y otro de Sevilla');
        end if;
        if count_proveedores=1 then
                select ciudad into lugar1 from vista_proveedor where
cod_proveedor=:new.cod_proveedor;
                select cod_proveedor into our_proveedor from vista_articulo where
cod_articulo=:new.cod_articulo;
                select ciudad into lugar2 from vista_proveedor where
cod_proveedor=our_proveedor;
                if lugar1=lugar2 then
                        raise_application_error(-20215, 'Solo puede haber dos proveedores
uno de Granada y otro de Sevilla');
                end if;
        end if;
end;
```

13. Ningún proveedor será de otra ciudad distinta a Granada o a Sevilla.

```
create or replace trigger prov_granada_sevilla
before insert or update of COD_PROVEEDOR on PROVEEDOR for each row
begin
        if :new.ciudad!='Granada' and :new.ciudad!='Sevilla' then
                raise_application_error(-20016, 'Los proveedores solo pueden ser de
Granada o Sevilla');
        end if;
end;
```

14. Ningún hotel de las provincias de Granada, Jaén, Málaga o Almería podrán solicitar artículos a proveedores de Sevilla.

```
create or replace trigger restriccion_pedir_articulos1
before insert or update of COD_HOTEL on VENDE for each row
declare
        ciudad_hotel VARCHAR(30);
        ciudad_proveedor VARCHAR(30);
        aux1 BOOLEAN:=FALSE;
begin
        select provincia into ciudad_hotel from vista_hotel where cod_hotel=:new.cod_hotel;
        select ciudad into ciudad_proveedor from vista_proveedor where
cod_proveedor=:new.cod_proveedor;
        if ciudad_hotel='Granada' then
                aux1:=TRUE;
        elsif ciudad_hotel='Jaen' then
                aux1:=TRUE;
        elsif ciudad_hotel='Malaga' then
                aux1:=TRUE;
        elsif ciudad_hotel='Almeria' then
                aux1:=TRUE;
        end if;
        if aux1=TRUE then
                if ciudad_proveedor='Sevilla' then
                        raise_application_error(-20017, 'Los hoteles de Granada, Jaen,
Malaga y Almeria no pueden pedir articulos a Sevilla');
                end if;
        end if;
end;
```

15. Ningún hotel de las provincias de Córdoba, Sevilla, Cádiz o Huelva podrán solicitar artículos a proveedores de Granada.

```
create or replace trigger restriccion_pedir_articulos2
before insert or update of COD_HOTEL on VENDE for each row
declare
        ciudad_hotel VARCHAR(30);
        ciudad_proveedor VARCHAR(30);
        aux1 BOOLEAN:=FALSE;
begin
        select provincia into ciudad_hotel from vista_hotel where cod_hotel=:new.cod_hotel;
        select ciudad into ciudad_proveedor from vista_proveedor where
cod_proveedor=:new.cod_proveedor;
```

```
        if ciudad_hotel='Cordoba' then
                aux1:=TRUE;
        elsif ciudad_hotel='Sevilla' then
                aux1:=TRUE;
        elsif ciudad_hotel='Cadiz' then
                aux1:=TRUE;
        elsif ciudad_hotel='Huelva' then
                aux1:=TRUE;
        end if;
        if aux1=TRUE then
                if ciudad_proveedor='Granada' then
                        raise_application_error(-20018, 'Los hoteles de Cordoba, Sevilla,
Cadiz y Huelva no pueden pedir articulos a Granada');
                end if;
        end if;
end;
```

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

```
create or replace trigger eliminar_proveedor
before delete on PROVEEDOR for each row
declare
        cont number;
begin
        select count(*) into cont from vista_vende where cantidad>0 and
cod_proveedor=:new.cod_proveedor;
        if cont>0 then
                raise_application_error(-20019, 'No se puede eliminar un proveedor si tiene
ventas realizadas');
        end if;
end;
```

17. Los datos referentes a un artículo, sólo podrán eliminarse de la base de datos, si la cantidad total suministrada de ese artículo es 0, o no existe ningún suministro.

```
create or replace trigger eliminar_articulo
before delete on ARTICULO for each row
declare
        cont number;
begin
        select count(*) into cont from vista_vende where cantidad>0 and
cod_articulo=:new.cod_articulo;
```

```
        if cont>0 then
                raise_application_error(-20020, 'No se puede eliminar un proveedor si tiene
ventas realizadas');
        end if;
end;
```

PRÁCTICA 4

IMPLEMENTACIÓN DE ACTUALIZACIONES

PROCEDIMIENTOS

1. Dar de alta a un nuevo empleado. Argumentos: Código de empleado, DNI, nombre, dirección actual, teléfono, fecha de inicio de contrato, salario, código del hotel en el que va a trabajar, y fecha de inicio en el hotel.

```
create or replace PROCEDURE alta_empleado(empleado number, dni varchar, nombre
varchar, telefono number,direccion varchar, fechaini date, salario number, c_hotel number,
f_ini_hotel date) IS
        comunidad varchar(30);
        cont number;
        cont2 number;
begin
        select count(*) into cont from VISTA_EMPLEADO where
COD_EMPLEADO=empleado;
        select count(*) into cont2 from vista_hotel where cod_hotel=c_hotel;
        if cont != 0 then
                raise_application_error(-20021,'El empleado ya está registrado en la base de
datos');
        end if;
        if cont2 = 0 then
                raise_application_error(-20060,'El hotel no existe');
        end if;
        select PROVINCIA into comunidad from VISTA_HOTEL where COD_HOTEL =
c_hotel;
        if comunidad = 'Granada' OR comunidad='Jaen' THEN
                INSERT into jafe1.EMPLEADO values (empleado, dni, nombre, telefono,
direccion, salario ,c_hotel, f_ini_hotel, fechaini);
        elsif comunidad='Cadiz' OR comunidad='Huelva' THEN
                INSERT into jafe2.EMPLEADO values (empleado, dni, nombre, telefono,
direccion, salario ,c_hotel, f_ini_hotel, fechaini);
        elsif comunidad='Sevilla' OR comunidad='Cordoba' THEN
                INSERT into jafe3.EMPLEADO values (empleado, dni, nombre, telefono,
direccion, salario ,c_hotel, f_ini_hotel, fechaini);
```

```
        elsif comunidad='Malaga' OR comunidad='Almeria' THEN
                INSERT into jafe4.EMPLEADO values (empleado, dni, nombre, telefono,
direccion, salario ,c_hotel, f_ini_hotel, fechaini);
        end if;
end;
```

2. Dar de baja a un empleado. Argumento: Código de empleado y fecha de baja. Esta operación requiere, antes de proceder a la eliminación del empleado, crear el registro histórico correspondiente.

```
create or replace PROCEDURE baja_empleado(empleado number , fecha_baja date) IS
        comunidad varchar(30);
        comunidad2 varchar(30);
        codhotel number;
        condir number;
        cont number;
        fechaini date;
begin
        select count(*) into cont from VISTA_EMPLEADO where COD_EMPLEADO =
empleado;
        if cont=0 then
                raise_application_error(-20022,'El empleado no existe');
        end if;
        select count(*) into condir from VISTA_HOTEL where DIRECTOR = empleado;
        select COD_HOTEL into codhotel from VISTA_EMPLEADO where
COD_EMPLEADO=empleado;
        select PROVINCIA into comunidad from VISTA_HOTEL where COD_HOTEL =
codhotel;
        select fecha_inicio into fechaini from VISTA_EMPLEADO where
COD_EMPLEADO=empleado;

        if comunidad = 'Granada' OR comunidad='Jaen' THEN
                delete jafe1.EMPLEADO where COD_EMPLEADO = empleado;
        elsif comunidad='Cadiz' OR comunidad='Huelva' THEN
                delete jafe2.EMPLEADO where COD_EMPLEADO = empleado;
        elsif comunidad='Sevilla' OR comunidad='Cordoba' THEN
                delete jafe3.EMPLEADO where COD_EMPLEADO = empleado;
        elsif comunidad='Malaga' OR comunidad='Almeria' THEN
                delete jafe4.EMPLEADO where COD_EMPLEADO = empleado;
        end if;

        if condir != 0 then
                select PROVINCIA into comunidad2 from VISTA_HOTEL where DIRECTOR
        = empleado;
                if comunidad2 = 'Granada' OR comunidad2='Jaen' THEN
```

```
                update JAFE1.HOTEL set DIRECTOR=0 where (DIRECTOR =
            empleado);
            elsif comunidad2='Cadiz' OR comunidad2='Huelva' THEN
                update JAFE2.HOTEL set DIRECTOR = 0 where (DIRECTOR =
            empleado);
            elsif comunidad2='Sevilla' OR comunidad2='Cordoba' THEN
                update JAFE3.HOTEL set DIRECTOR = 0 where (DIRECTOR =
            empleado);
            elsif comunidad2='Malga' OR comunidad2='Almeria' THEN
                update JAFE4.HOTEL set DIRECTOR = 0 where (DIRECTOR =
            empleado);
            end if;
        end if;
        INSERT into jafe1.HISTORIA values (empleado, codhotel, fechaini, fecha_baja);
        INSERT into jafe2.HISTORIA values (empleado, codhotel, fechaini, fecha_baja);
        INSERT into jafe3.HISTORIA values (empleado, codhotel, fechaini, fecha_baja);
        INSERT into jafe4.HISTORIA values (empleado, codhotel, fechaini, fecha_baja);
end;
```

3. Modificar el salario de un empleado. Argumentos: Código del empleado y nuevo salario.

```
create or replace procedure modificar_salario (empleado number, nuevo_salario number) IS
        comunidad varchar(30);
        hotel number;
begin
        select COD_HOTEL into hotel from VISTA_EMPLEADO where
COD_EMPLEADO=empleado;
        select PROVINCIA into comunidad from VISTA_HOTEL where COD_HOTEL = hotel;
        if comunidad = 'Granada' OR comunidad='Jaen' THEN
                update jafe1.EMPLEADO set SALARIO = nuevo_salario where
        COD_EMPLEADO = empleado;
        elsif comunidad='Cadiz' OR comunidad='Huelva' THEN
                update jafe2.EMPLEADO set SALARIO = nuevo_salario where
        COD_EMPLEADO = empleado;
        elsif comunidad='Sevilla' OR comunidad='Cordoba' THEN
                update jafe3.EMPLEADO set SALARIO = nuevo_salario where
        COD_EMPLEADO = empleado;
        elsif comunidad='Malaga' OR comunidad='Almeria' THEN
                update jafe4.EMPLEADO set SALARIO = nuevo_salario where
        COD_EMPLEADO = empleado;
        end if;
end;
```

4. Trasladar de hotel a un empleado. Argumentos: Código de empleado, fecha fin en el actual hotel, código del hotel al que es trasladado, fecha de inicio en el nuevo hotel y, opcionalmente, nueva dirección y nuevo teléfono. Si estos dos últimos argumentos no se indican, los valores para los atributos correspondientes deben ponerse a nulo. Los valores para el resto de atributos (DNI, nombre, fecha de contrato y salario) no se modifican. Esta operación requiere crear el registro histórico correspondiente.

```
create or replace procedure trasladar_emp(codemp number, fechafin date, codhotel
number, fechainiciohotel date , dir varchar2 , telef number) IS
        fech date;
        sal number;
        dni number;
        nombre varchar(30);
        comunidad varchar(30);
        comunidadv varchar(30);
        hotel number;
        cont number;
        direc varchar(30);
        telefon number;
        fech_contrato date;
begin
        if dir!=null then
                direc := dir;
        else
                direc:= null;
        end if;
        if telef!=null then
                telefon:=telef;
        else
                telefon:=null;
        end if;
        select count(*) into cont from VISTA_EMPLEADO where COD_EMPLEADO =
codemp;
        select COD_HOTEL into hotel from VISTA_EMPLEADO where COD_EMPLEADO =
codemp;
        if cont =0 then
                raise_application_error(-20023, 'El empleado no existe');
        end if;
        select PROVINCIA into comunidadv from VISTA_HOTEL where COD_HOTEL =
hotel;
        select F_CONTRATO, FECHA_INICIO, SALARIO, DNI, NOMBRE into
fech_contrato, fech, sal,dni, nombre  from VISTA_EMPLEADO where
COD_EMPLEADO=codemp;
```

```
        select PROVINCIA into comunidad from VISTA_HOTEL where
COD_HOTEL=codhotel;
        if comunidadv = 'Granada' OR comunidadv='Jaen' THEN
                delete jafe1.EMPLEADO where COD_EMPLEADO=codemp;
        elsif comunidadv='Cadiz' OR comunidadv='Huelva' THEN
                delete jafe2.EMPLEADO where COD_EMPLEADO=codemp;
        elsif comunidadv='Sevilla' OR comunidadv='Cordoba' THEN
                delete jafe3.EMPLEADO where COD_EMPLEADO=codemp;
        elsif comunidadv='Malaga' OR comunidadv='Almeria' THEN
                delete jafe4.EMPLEADO where COD_EMPLEADO=codemp;
        end if;

        if comunidadv = 'Granada' OR comunidadv='Jaen' THEN
                insert into jafe1.EMPLEADO values (codemp ,dni , nombre, telefon, direc, sal,
codhotel, fechainiciohotel, fech_contrato);
        elsif comunidadv='Cadiz' OR comunidadv='Huelva' THEN
                insert into jafe2.EMPLEADO values (codemp ,dni , nombre, telefon, sal, direc,
codhotel, fechainiciohotel, fech_contrato);
        elsif comunidadv='Sevilla' OR comunidadv='Cordoba' THEN
                insert into jafe3.EMPLEADO values (codemp ,dni , nombre, telefon, sal, direc,
codhotel, fechainiciohotel, fech_contrato);
        elsif comunidadv='Malaga' OR comunidadv='Almeria' THEN
                insert into jafe4.EMPLEADO values (codemp ,dni , nombre, telefon, sal, direc,
codhotel, fechainiciohotel, fech_contrato);
        end if;

        INSERT into jafe1.HISTORIA values (codemp, codhotel, fech, fechafin);
        INSERT into jafe2.HISTORIA values (codemp, codhotel, fech, fechafin);
        INSERT into jafe3.HISTORIA values (codemp, codhotel, fech, fechafin);
        INSERT into jafe4.HISTORIA values (codemp, codhotel, fech, fechafin);
end;
```

5. Dar de alta un nuevo hotel. Argumentos: Código de hotel, nombre, ciudad en la que se ubica, provincia, número de habitaciones sencillas y número de habitaciones dobles.

```
create or replace procedure alta_hotel (codhotel number, nombre varchar, ciudad varchar,
provincia varchar, simples number, dobles number) IS
        cont number;
begin
        select count(*) into cont from VISTA_HOTEL where COD_HOTEL = codhotel;
        if cont != 0 then
                raise_application_error(-20025,'El hotel ya existe');
        else
                if provincia= 'Granada' OR provincia='Jaen' THEN
```

```
                INSERT into jafe1.HOTEL values (codhotel, nombre, dobles, simples,
ciudad, provincia,0);
        elsif provincia='Cadiz' OR provincia='Huelva' THEN
                INSERT into jafe2.HOTEL values (codhotel, nombre, dobles, simples,
ciudad, provincia,0);
        elsif provincia='Sevilla' OR provincia='Cordoba' THEN
                INSERT into jafe3.HOTEL values (codhotel, nombre, dobles, simples,
ciudad, provincia,0);
        elsif provincia='Malaga' OR provincia='Almeria' THEN
                INSERT into jafe4.HOTEL values (codhotel, nombre, dobles, simples,
ciudad, provincia,0);
        end if;
    end if;
end;
```

6. Cambiar el director de un hotel. Esta operación debe servir también para nombrar director inicial de un hotel. Argumentos: Código de sucursal y código del nuevo (o del primer) director.

```
create or replace procedure cambiar_director (codhotel number, codemp number) IS
    cont number;
    contv number;
    comunidad varchar(30);

begin
    select count(*) into cont from VISTA_EMPLEADO where COD_EMPLEADO =
codemp;
    if cont = 0 then
        raise_application_error(-20026, 'El director no es un empleado');
    end if;
    select count(*) into contv from VISTA_HOTEL where COD_HOTEL= codhotel;
    if contv =0 then
        raise_application_error(-20027, 'El hotel no existe');
    end if;
    select PROVINCIA into comunidad from VISTA_HOTEL where COD_HOTEL =
codhotel;
    if comunidad = 'Granada' OR comunidad='Jaen' THEN
        UPDATE jafe1.HOTEL set DIRECTOR = codemp;
    elsif comunidad='Cadiz' OR comunidad='Huelva' THEN
        UPDATE jafe2.HOTEL set DIRECTOR = codemp;
    elsif comunidad='Sevilla' OR comunidad='Cordoba' THEN
        UPDATE jafe3.HOTEL set DIRECTOR = codemp;
    elsif comunidad='Malaga' OR comunidad='Almeria' THEN
```

```
            UPDATE jafe4.HOTEL set DIRECTOR = codemp;
        end if;
end;
```

7. Dar de alta a un nuevo cliente. Argumentos: Código de cliente, DNI, nombre, y teléfono.

```
create or replace procedure alta_cliente (codcli number, dni number, nombre varchar, telef
number) IS
        cont number;
begin
        select count(*) into cont from CLIENTE where COD_CLIENTE = codcli;
        if cont !=0 then
            raise_application_error ( -20028, 'El cliente ya existe');
        end if;
        INSERT into jafe1.CLIENTE values (codcli, dni, nombre, telef);
        INSERT into jafe2.CLIENTE values (codcli, dni, nombre, telef);
        INSERT into jafe3.CLIENTE values (codcli, dni, nombre, telef);
        INSERT into jafe4.CLIENTE values (codcli, dni, nombre, telef);
end;
```

8. Dar de alta o actualizar una reserva. Argumentos: Código de cliente, código de hotel, tipo de habitación, fecha de entrada, fecha de salida, y precio de la habitación.

```
create or replace PROCEDURE alta_reserva(codcli number, codhotel number,
tipo_habitacion varchar, f_entrada date, f_salida date, price number) IS
        cont1 number;
        cont2 number;
        comunidad1 number;
begin
        select count(*) into cont1 from CLIENTE where COD_CLIENTE = codcli;
        select count(*) into cont2 from VISTA_HOTEL where COD_HOTEL = codhotel;

        if cont1 =0 then
            raise_application_error(-20029,'El cliente no existe');
        end if;

        if cont2 =0 then
            raise_application_error(-20030,'El hotel no existe');
        end if;
```

```
        select PROVINCIA into comunidad1 from VISTA_HOTEL where COD_HOTEL =
codhotel;
        if tipo_habitacion!= 'Sencilla' and tipo_habitacion!= 'Doble' then
                raise_application_error(-20031,'El tipo de habitacion no existe');
        end if;

        if comunidad1 = 'Granada' OR comunidad1='Jaen' THEN
                insert into jafe1.RESERVA values(codcli, codhotel, f_entrada, f_salida,
tipo_habitacion, price);
        elsif comunidad1='Cadiz' OR comunidad1='Huelva' THEN
                insert into jafe2.RESERVA values(codcli, codhotel, f_entrada, f_salida,
tipo_habitacion, price);
        elsif comunidad1='Sevilla' OR comunidad1='Cordoba' THEN
                insert into jafe3.RESERVA values(codcli, codhotel, f_entrada, f_salida,
tipo_habitacion, price);
        elsif comunidad1='Malaga' OR comunidad1='Almeria' THEN
                insert into jafe4.RESERVA values(codcli, codhotel, f_entrada, f_salida,
tipo_habitacion, price);
        end if;
end;
```

9. Anular una reserva. Argumentos: Código de cliente, código de hotel, fecha de entrada y fecha de salida.

```
create or replace procedure baja_reserva (codcli number, codhotel number, fech date,
f_entrada date, f_salida date) IS
        cont number;
        comunidad varchar(30);
begin
        select count(*) into cont from VISTA_RESERVA where COD_CLIENTE=codcli AND
COD_HOTEL=codhotel AND FECHA_ENTRADA = f_entrada AND
FECHA_SALIDA=f_salida;
        if cont = 0 OR f_entrada = null OR f_salida = null then
                raise_application_error(-20032, 'No existe la reserva');
        end if;
        select PROVINCIA into comunidad from VISTA_HOTEL where COD_HOTEL =
codhotel;
        if comunidad = 'Granada' OR comunidad='Jaen' THEN
                delete jafe1.RESERVA where COD_CLIENTE=codcli AND
COD_HOTEL=codhotel AND FECHA_ENTRADA = f_entrada AND FECHA_SALIDA =
f_salida;
        elsif comunidad='Cadiz' OR comunidad='Huelva' THEN
                delete jafe2.RESERVA where COD_CLIENTE=codcli AND
COD_HOTEL=codhotel AND FECHA_ENTRADA = f_entrada AND FECHA_SALIDA =
f_salida;
```

```
        elsif comunidad='Sevilla' OR comunidad='Cordoba' THEN
                delete jafe3.RESERVA where COD_CLIENTE=codcli AND
COD_HOTEL=codhotel AND FECHA_ENTRADA = f_entrada AND FECHA_SALIDA =
f_salida;
        elsif comunidad='Malaga' OR comunidad='Almeria' THEN
                delete jafe4.RESERVA where COD_CLIENTE=codcli AND
COD_HOTEL=codhotel AND FECHA_ENTRADA = f_entrada AND FECHA_SALIDA =
f_salida;
        end if;
end;
```

10. Dar de alta a un nuevo proveedor. Argumentos: Código de proveedor, nombre, y ciudad.

```
create or replace procedure alta_proveedor (cod_p number, nom varchar, ciudad varchar) IS
        cntp number;
BEGIN
        select count(*) into cntp from VISTA_PROVEEDOR where
COD_PROVEEDOR=cod_p;
        if cntp!=0 then
                raise_application_error(-20033, 'El proveedor ya esta dado de alta en la base
de datos');
        end if;
        if ciudad = 'Granada' THEN
                insert into jafe1.PROVEEDOR values(cod_p, nom, ciudad);
                insert into jafe4.PROVEEDOR values(cod_p, nom, ciudad);
        elsif ciudad='Sevilla' THEN
                insert into jafe3.PROVEEDOR values(cod_p, nom, ciudad);
                insert into jafe2.PROVEEDOR values(cod_p, nom, ciudad);
        else
                raise_application_error(-20034, 'La ciudad del proveedor solo puede ser
Granada o Sevilla');
        end if;
END;
```

11. Dar de baja a un proveedor. Argumento: Código de proveedor.

```
create or replace procedure baja_proveedor(cod_p number) IS
        cntp number;
        lugar varchar(30);
BEGIN
        select count(*) into cntp from VISTA_PROVEEDOR where
COD_PROVEEDOR=cod_p;
        if cntp=0 then
```

```
            raise_application_error(-20035, 'El proveedor que quiere eliminar no se
encuentra en la base de datos');
        end if;
        select CIUDAD into lugar from VISTA_PROVEEDOR where
COD_PROVEEDOR=cod_p;
        if lugar = 'Granada' THEN
            delete from jafe1.PROVEEDOR where COD_PROVEEDOR=cod_p;
            delete from jafe4.PROVEEDOR where COD_PROVEEDOR=cod_p;
        elsif lugar = 'Sevilla' THEN
            delete from jafe3.PROVEEDOR where COD_PROVEEDOR=cod_p;
            delete from jafe2.PROVEEDOR where COD_PROVEEDOR=cod_p;
        end if;
END;
```

12. Dar de alta o actualizar un suministro. Argumentos: Código de artículo, código de proveedor, código del hotel que solicita el suministro, fecha de suministro, cantidad a suministrar (puede ser negativa, lo cual significa que se devuelve el suministro) , y precio por unidad. Si existiera una solicitud de suministro del mismo hotel y del mismo artículo al mismo proveedor y en la misma fecha, se modificará , en la forma adecuada, la cantidad a suministrar.

```
create or replace PROCEDURE alta_suministro(cod_a number, codpro number,codhotel
number, fech date, cant number, precio number) IS
        cont1 number;
        cont2 number;
        cont3 number;
        fech1 date;
        comunidad1 varchar(30);
        comunidad2 varchar(30);
        zonapro int;
        zonahotel int;
        cont4 number;
        cantini number;
        cantfin number;
begin
        select count(*) into cont1 from VISTA_PROVEEDOR where COD_PROVEEDOR =
codpro;
        select count(*) into cont2 from VISTA_HOTEL where COD_HOTEL = codhotel;
        select count(*) into cont3 from VISTA_ARTICULO where COD_ARTICULO = cod_a;
        if cont1 =0 then
            raise_application_error(-20036,'El proveedor no existe');
        end if;
        if cont2 =0 then
            raise_application_error(-20037,'La hotel no existe');
        end if;
```

```
        if cont3 =0 then
                raise_application_error(-20038,'El articulo no existe');
        end if;

        select PROVINCIA into comunidad1 from VISTA_HOTEL where COD_HOTEL =
codhotel;

        if comunidad1 = 'Granada' OR comunidad1='Jaen' THEN
                zonahotel:=1;
        elsif comunidad1='Cadiz' OR comunidad1='Huelva' THEN
                zonahotel:=2;
        elsif comunidad1='Sevilla' OR comunidad1='Cordoba' THEN
                zonahotel:=2;
        elsif comunidad1='Malaga' OR comunidad1='Almeria' THEN
                zonahotel:=1;
        end if;

        select CIUDAD into comunidad2 from VISTA_PROVEEDOR where
COD_PROVEEDOR = codpro;
        if comunidad2 = 'Granada' THEN
                zonapro:=1;
        elsif comunidad2='Sevilla' THEN
                zonapro:=2;
        end if;

        if zonapro != zonahotel then
                raise_application_error(-20040,'EL hotel no puede hacer un pedido a ese
proveedor');
        end if;
        select count(*) into cont4 from VISTA_VENDE where COD_PROVEEDOR = codpro
AND COD_HOTEL = codhotel AND COD_ARTICULO = cod_a AND FECHA = fech;

        if cont4 = 0 then
                if comunidad1 = 'Granada' OR comunidad1='Jaen' THEN
                        insert into jafe1.VENDE values (cod_a, codpro, codhotel, fech, cant,
precio);
                elsif comunidad1='Cadiz' OR comunidad1='Huelva' THEN
                        insert into jafe2.VENDE values (cod_a, codpro, codhotel, fech, cant,
precio);
                elsif comunidad1='Sevilla' OR comunidad1='Cordoba' THEN
                        insert into jafe3.VENDE values (cod_a, codpro, codhotel, fech, cant,
precio);
                elsif comunidad1='Malaga' OR comunidad1='Almeria' THEN
                        insert into jafe4.VENDE values (cod_a, codpro, codhotel, fech, cant,
precio);
                end if;
```

```
        else
                select CANTIDAD into cantini from VISTA_VENDE where
COD_PROVEEDOR=codpro AND COD_HOTEL=codhotel AND COD_ARTICULO = cod_a
AND FECHA=fech;
                cantfin := cantini + cant;
                if cantfin >= 0 then
                        if comunidad1 = 'Granada' OR comunidad1='Jaen' THEN
                                UPDATE jafe1.VENDE set CANTIDAD= cantfin where
COD_PROVEEDOR = codpro and COD_HOTEL = codhotel and COD_ARTICULO=cod_a
and FECHA=fech;
                        elsif comunidad1='Cadiz' OR comunidad1='Huelva' THEN
                                UPDATE jafe2.VENDE set CANTIDAD= cantfin where
COD_PROVEEDOR = codpro and COD_HOTEL = codhotel and COD_ARTICULO=cod_a
and FECHA=fech;
                        elsif comunidad1='Sevilla' OR comunidad1='Cordoba' THEN
                                UPDATE jafe3.VENDE set CANTIDAD= cantfin where
COD_PROVEEDOR = codpro and COD_HOTEL = codhotel and COD_ARTICULO=cod_a
and FECHA=fech;
                        elsif comunidad1='Malaga' OR comunidad1='Almeria' THEN
                                UPDATE jafe4.VENDE set CANTIDAD= cantfin where
COD_PROVEEDOR = codpro and COD_HOTEL = codhotel and COD_ARTICULO=cod_a
and FECHA=fech;
                        end if;
                else
                        raise_application_error(-20041, 'La cantidad pedida no es válida');
                end if;
        end if;
end;
```

13. Dar de baja suministros. Argumentos: Código del hotel que solicitó el suministro, código del artículo y, opcionalmente, fecha del suministro. Si no se indica la fecha de suministro, se darán de baja todos los suministros solicitados por el hotel de ese artículo al proveedor.

```
create or replace procedure baja_suministro (codhotel number, cod_a number, fech date) IS
        cont number;
        comunidad varchar(30);
begin
        select count(*) into cont from VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a AND FECHA = fech;

        if cont = 0 AND fech != null then
                raise_application_error(-20042, 'No existe el suministro');
```

```
        end if;

        select PROVINCIA into comunidad from VISTA_HOTEL where cod_hotel = codhotel;

        if comunidad = 'Granada' OR comunidad='Jaen' THEN
                if fech = null then
                        delete jafe1.VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a;
                else
                        delete jafe1.VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a AND FECHA = fech;
                end if;
        elsif comunidad='Cadiz' OR comunidad='Huelva' THEN
                if fech = null then
                        delete jafe2.VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a;
                else
                        delete jafe2.VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a AND FECHA = fech;
                end if;
        elsif comunidad='Cordoba' OR comunidad='Sevilla' THEN
                if fech = null then
                        delete jafe3.VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a;
                else
                        delete jafe3.VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a AND FECHA = fech;
                end if;
        elsif comunidad='Malaga' OR comunidad='Almeria' THEN
                if fech = null then
                        delete jafe4.VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a;
                else
                        delete jafe4.VENDE where cod_hotel=codhotel AND
COD_ARTICULO = cod_a AND FECHA = fech;
                end if;
        end if;
end;
```

14. Dar de alta un nuevo artículo. Argumentos: Código de artículo, nombre, tipo y código de proveedor.

```
create or replace procedure alta_articulo (cod_a number, nombre varchar, tipo varchar,
cod_p number) IS
        cntv number;
```

```
        countp number;
        city varchar(30);
        ciudad1 varchar(30);
        ciudad2 varchar(30);
        c_p number;
BEGIN
        select count(*) into cntv from VISTA_ARTICULO where cod_articulo=cod_a and
cod_proveedor=cod_p;
        if cntv!=0 then
                raise_application_error(-20043, 'El articulo ya existe en la base de datos');
        end if;

        select count(*) into countp from VISTA_ARTICULO where COD_ARTICULO=cod_a;
        if countp=1 then
                select cod_proveedor into c_p from VISTA_ARTICULO where
cod_articulo=cod_a;
                select ciudad into ciudad1 from vista_proveedor where
COD_PROVEEDOR=c_p;
                select ciudad into ciudad2 from VISTA_PROVEEDOR where
COD_PROVEEDOR=cod_p;
                if ciudad1=ciudad2 THEN
                        raise_application_error(-20143, 'El articulo ya existe en la base de
datos.');
                end if;
                if ciudad2='Granada' then
                        insert into jafe1.ARTICULO values (cod_a, cod_p, nombre, tipo);
                        insert into jafe4.ARTICULO values (cod_a, cod_p, nombre, tipo);
                elsif ciudad2='Sevilla' then
                        insert into jafe2.ARTICULO values (cod_a, cod_p, nombre, tipo);
                        insert into jafe3.ARTICULO values (cod_a, cod_p, nombre, tipo);
                end if;
        elsif countp=2 then
                        raise_application_error(-20243, 'No puede haber un articulo con
proveedores de la misma ciudadde');
        end if;
END;
```

15. Dar de baja un artículo. Argumentos: Código de artículo. Si es posible dar de baja el artículo, esta operación dará de baja también todos los suministros en los que aparezca dicho artículo.

```
create or replace PROCEDURE BAJA_ARTICULO(cod_a NUMBER) IS
        cnt NUMBER;
        lugar varchar2(30);
```

```
        c_p number;
BEGIN
        SELECT COUNT(*) INTO cnt FROM VISTA_ARTICULO WHERE COD_ARTICULO
= cod_a;
        if (cnt = 0) THEN
                raise_application_error(-20044, 'No existe el vino indicado');
        END if;
        delete from JAFE1.VENDE where COD_ARTICULO = cod_a;
        delete from JAFE4.VENDE where COD_ARTICULO = cod_a;
        delete from JAFE2.VENDE where COD_ARTICULO = cod_a;
        delete from JAFE3.VENDE where COD_ARTICULO = cod_a;
        DELETE FROM JAFE1.ARTICULO WHERE COD_ARTICULO = cod_a;
        DELETE FROM JAFE2.ARTICULO WHERE COD_ARTICULO = cod_a;
        DELETE FROM JAFE3.ARTICULO WHERE COD_ARTICULO = cod_a;
        DELETE FROM JAFE4.ARTICULO WHERE COD_ARTICULO = cod_a;
END;
```

PRÁCTICA 5

IMPLEMENTACIÓN DE CONSULTAS

1. “Listar los hoteles (nombre y ciudad) de las provincias de Granada, Huelva o Almería, y los proveedores (nombre y ciudad), a los que se le ha suministrado “Queso” o “Mantequilla” entre el 12 de mayo de 2017 y el 28 de mayo de 2017.

```
SELECT H.NOMBRE, H.CIUDAD, P.NOMBRE, P.CIUDAD FROM
VISTA_HOTEL H, VISTA_PROVEEDOR P, VISTA_VENDE S,
VISTA_ARTICULO AR
WHERE P.COD_PROVEEDOR = S.COD_PROVEEDOR AND
S.COD_HOTEL = H.COD_HOTEL AND AR.COD_ARTICULO =
S.COD_ARTICULO AND (PROVINCIA = 'Granada'
OR PROVINCIA = 'Huelva' OR PROVINCIA = 'Almeria') AND S.FECHA
> '12/05/17' AND S.FECHA < '28/05/17' AND (AR.NOMBRE = 'Queso'
OR AR.NOMBRE = 'Mantequilla');
```

2. Dado por teclado el código de un productor, “Listar los productos (nombre), los hoteles (nombre y ciudad) y la cantidad total de cada producto, suministrados por dicho productor a hoteles de las provincias de Jaén o Almería”.

```
SELECT AA.NOMBRE, H.NOMBRE, H.CIUDAD, S.CANTIDAD
FROM VISTA_PROVEEDOR P, VISTA_ARTICULO AA, VISTA_HOTEL
H, VISTA_VENDE S WHERE S.COD_ARTICULO =
AA.COD_ARTICULO AND AA.COD_PROVEEDOR =
P.COD_PROVEEDOR
AND S.COD_PROVEEDOR = P.COD_PROVEEDOR AND
S.COD_HOTEL = H.COD_HOTEL AND P.COD_PROVEEDOR =
&CodPro AND (H.CIUDAD = 'Jaen' OR H.CIUDAD = 'Almeria');
```

3. Dado por teclado el código de un hotel, “Listar los clientes (nombre y teléfono), que tengan registrada más de una reserva en dicho hotel”.

```
SELECT NOMBRE, TELEFONO FROM CLIENTE
WHERE COD_CLIENTE IN (SELECT COD_CLIENTE FROM (SELECT COD_CLIENTE,
COUNT(*) AS "COUNT" FROM VISTA_RESERVA WHERE cod_hotel = &CodHotel GROUP
BY cod_cliente) WHERE "COUNT" > 1);
```