BDD: Práctica final

UNIVERSIDAD DE GRANADA

Juan Manuel Consigliere Picco
Enrique González López

0. Índice

1. Enunciado del problema

Una multinacional andaluza desea implementar una base de datos distribuida para gestionar los empleados, clientes y proveedores de una de sus cadenas hoteleras. Los datos correspondientes a cada uno de los diferentes hoteles de la cadena, incluyendo datos de empleados, de clientes y de proveedores, estarán almacenados físicamente en cuatro localidades, dependiendo de la ciudad en la que esté ubicado el hotel. Las localidades de almacenamiento serán: Granada (para hoteles de todas las ciudades de las provincias de Granada y Jaén), Cádiz (para hoteles de todas las ciudades de las provincias de Cádiz y Huelva), Sevilla (para hoteles de todas las ciudades de las provincias de Sevilla y Córdoba), y Málaga (para hoteles de todas las ciudades de las provincias de Málaga y Almería).

Los principales requisitos de funcionamiento y de almacenamiento de la cadena se describen con la siguiente lista de especificaciones:

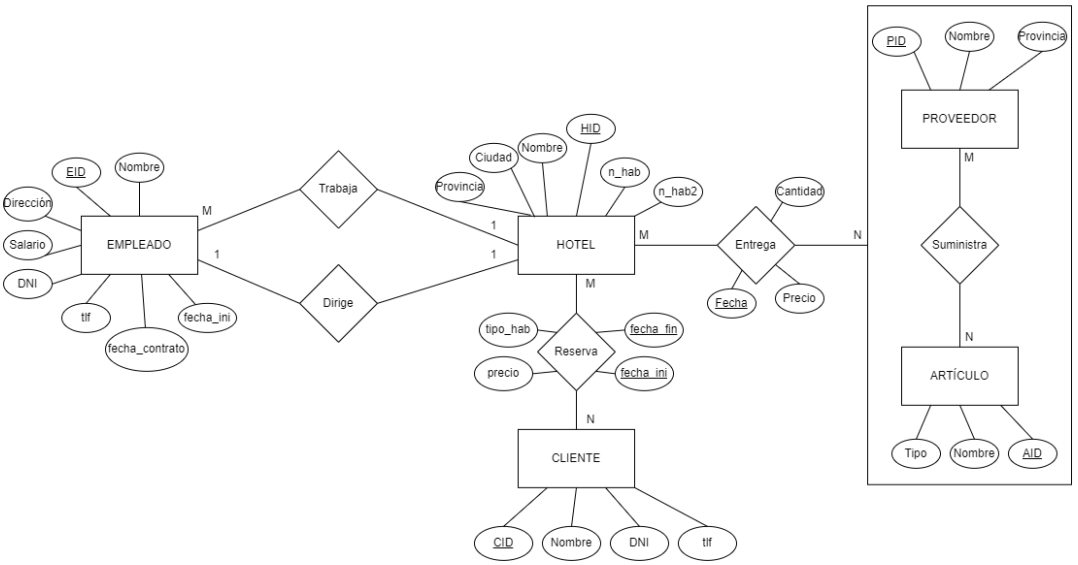
- **LOS HOTELES** se ubican (cada uno de ellos) en una ciudad de una provincia, se identifican por un código único, tienen un nombre, una capacidad medida en número de habitaciones sencillas y número de habitaciones dobles y un director que es empleado de la multinacional.
- **LOS EMPLEADOS** de la cadena se identifican mediante un código de empleado, que mantendrán mientras trabajen en la multinacional independientemente del hotel en el que estén asignados en un momento determinado. La administración de la cadena almacena para cada empleado el DNI, el nombre, el número de teléfono, la dirección actual, la fecha de comienzo de contrato con la multinacional, el salario, el hotel en el que está asignado actualmente y la fecha en la que inició su trabajo en él. Cada empleado, en un momento dado, sólo puede estar asignado a un hotel. Además, se mantendrá, para cada empleado, un registro histórico de todos los hoteles a los que haya sido asignado desde el inicio de su contrato con la multinacional, incluyendo el tiempo (fecha de inicio y fecha de fin) transcurrido en cada uno de dichos hoteles.
- **LOS PROVEEDORES** de la cadena están todos en ‘Granada’ y en ‘Sevilla’, de manera que, los proveedores de Granada suministran a hoteles de las provincias de Jaén, Granada, Málaga y Almería y los proveedores de Sevilla suministran a hoteles de las provincias de Córdoba, Sevilla, Cádiz y Huelva. De cada proveedor, se almacena un código de proveedor, su nombre, la ciudad en la que se encuentra, los artículos que suministra y los suministros proporcionados a los diferentes hoteles. De cada artículo se almacena el código de artículo, el nombre de artículo y el tipo de artículo. Un suministro consiste en un artículo, la cantidad suministrada, la fecha del suministro, y el precio por unidad del artículo en la fecha de suministro. Un artículo sólo puede ser suministrado como mucho por dos proveedores: uno de Granada y otro de Sevilla.

- **LOS CLIENTES** de los hoteles se identifican mediante un código de cliente, que se les asigna la primera vez que realizan una reserva en algún hotel de la cadena. Para cada cliente, se almacena el DNI, el nombre, un teléfono de contacto y las reservas realizadas en los hoteles de la multinacional. Cada reserva consiste en la fecha de entrada en el hotel, fecha de salida, tipo de habitación reservada (sencilla o doble) y el precio de la habitación por noche en el momento de la reserva. Un cliente puede realizar más de una reserva en un mismo hotel, pero en fechas diferentes.

Las aplicaciones, tanto de actualización como de consulta, que utilizan datos de hoteles (incluidos datos de empleados, reservas y suministros), se generan en cualquier localidad, pero referencian con una probabilidad del 0,95 a hoteles cercanos. Las aplicaciones que usan datos de proveedores (incluidos datos de suministros) si se generan en Jaén, Granada, Almería o Málaga, referencian siempre a proveedores de Granada, y si se generan en Córdoba, Cádiz, Sevilla o Huelva, referencian siempre a proveedores de Sevilla.

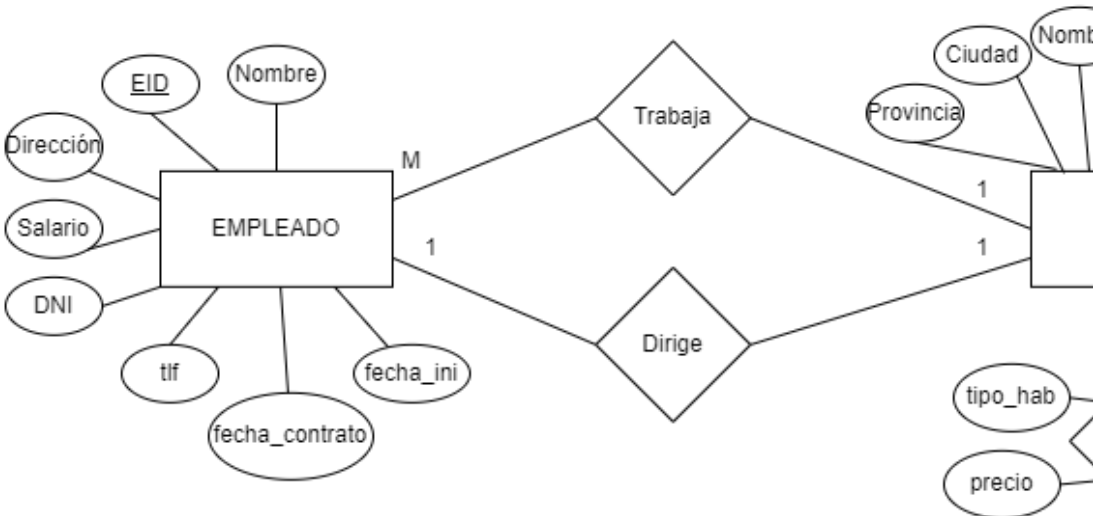
2. Diagrama Entidad/Relación

Primeramente voy a mostrar el diagrama al completo, y luego me iré parando poco a poco en cada parte que considere relevante.



Hemos puesto todos los atributos que se nos especifican en el enunciado, ni mas ni menos. Ahora voy a dividir en 3 partes mi explicación:

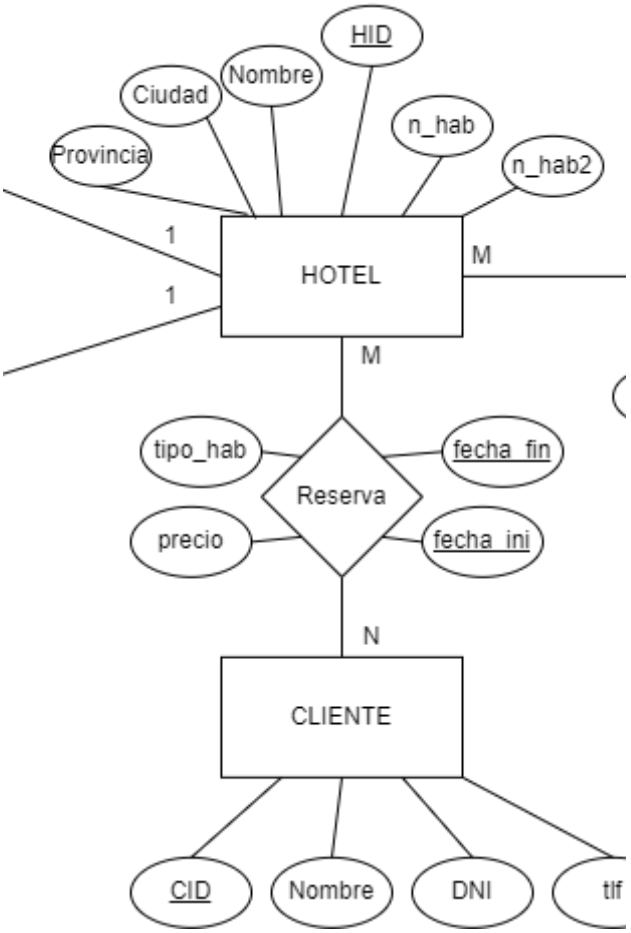
2.1. Empleados y directores



Empleado se conecta con Hotel mediante la relación Trabaja, que indica que un empleado trabaja en un solo hotel pero en un hotel pueden trabajar muchísimos empleados (relación uno a muchos).

Hay una relación Dirige, que es la que nos permite modelar el concepto de Director. Un director dirige un solo hotel, y un hotel solamente tiene un director (relación uno a uno). También vemos que el director es un empleado más del hotel.

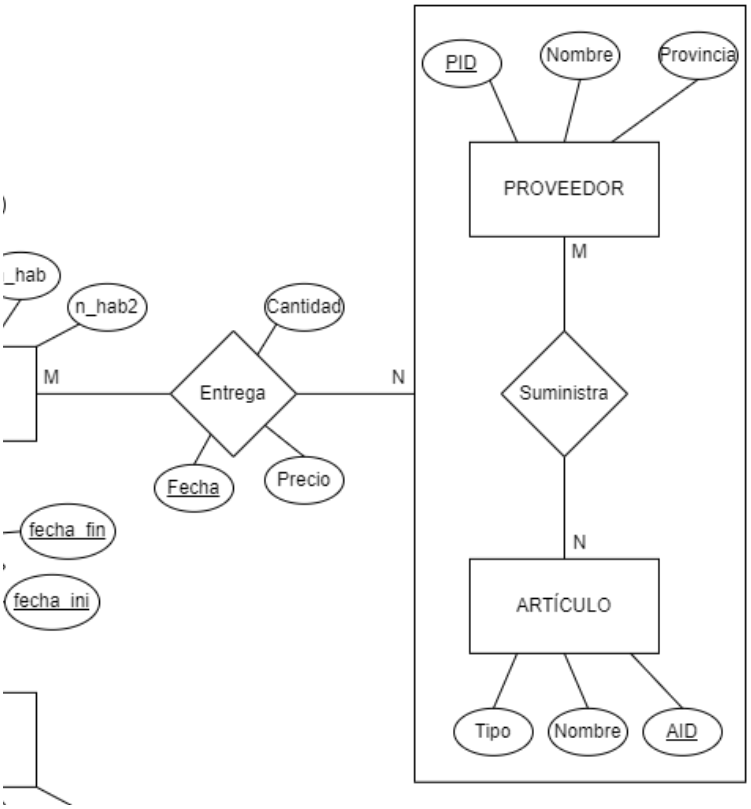
2.2. Hoteles y clientes



Entre hotel y cliente se encuentra la relación Reserva, la cual contiene como claves primarias fecha_ini y fecha_fin. Esto sirve para que la reserva al realizarse el paso a tablas contenga restricciones, y así no se dará el caso en el que un cliente tenga una reserva las mismas fechas en el mismo hotel. Otras restricciones las veremos en los procedures y los triggers.

La relación existente entre Hotel y cliente es de muchos a muchos, ya que un cliente puede reservar en varios hoteles y los hoteles tienen reservas de muchos clientes.

2.3. Hoteles, proveedores y artículos



Primero voy a explicar la agregación entre Proveedor y Artículo. Es una relación de muchos a muchos aunque exista la restricción de que solamente puedan haber dos proveedores que suministren un artículo: uno en Granada y el otro en Sevilla. Por ahora un proveedor puede suministrar muchos artículos y un artículo puede ser suministrado por muchos proveedores.

Ahora hablemos de la relación Entrega. Esta relación es de muchos a muchos, ya que las entregas se realizan a muchos hoteles y los hoteles tienen muchas entregas. Tiene como clave primaria Fecha, ya que no pueden coincidir en fecha el mismo artículo del mismo proveedor del mismo hotel en la misma fecha.

3. Paso a tablas

El paso a tablas ha quedado de la siguiente manera:

- Empleado(EID, Nombre, Dirección, Salario, DNI, tlf, fecha_contrato, fecha_ini)
    - EID: ID del empleado. Clave primaria.
    - Nombre: Nombre del empleado.
    - Dirección: Dirección del empleado.
    - Salario: Salario del empleado.
    - DNI: DNI del empleado.
    - tlf: Teléfono del empleado.
    - fecha_contrato: Fecha de contrato del empleado.
    - fecha_ini: Fecha de inicio del empleado.

- Trabaja(**EID**, **HID**):
    - EID: ID del empleado. Clave externa de Empleado y clave primaria.
    - HID: ID del hotel. Clave externa de Hotel.

- Dirige(**EID**, **HID**):
    - EID: ID del empleado. Clave externa de Empleado y clave primaria.
    - HID: ID del hotel. Clave externa de Hotel.

- Hotel(HID, Nombre, Ciudad, Provincia, n_hab, n_hab2):
    - HID: ID del hotel. Clave primaria.
    - Nombre: Nombre del hotel.
    - Ciudad: Ciudad del hotel.
    - Provincia: Provincia del hotel.
    - n_hab: Número de habitaciones sencillas.
    - n_hab2: Número de habitaciones dobles.

- Cliente(CID, Nombre, DNI, tlf):
    - CID: ID del hotel. Clave primaria.
    - Nombre: Nombre del cliente.
    - DNI: DNI del cliente.
    - tlf: Teléfono del cliente.

- Reserva(**CID**, **HID**, fecha_fin, fecha_ini, tipo_hab, precio):
    - CID: ID del cliente. Clave externa de Cliente y clave primaria.
    - HID: ID del hotel. Clave externa de Hotel y clave primaria.
    - fecha_fin: Fecha de fin de reserva. Clave primaria.
    - fecha_ini: Fecha de inicio de reserva. Clave primaria.
    - tipo_hab: Tipo de habitación.
    - precio: Precio de la reserva.

- Proveedor(PID, Nombre, Provincia):
    - PID: ID del proveedor. Clave primaria.
    - Nombre: Nombre del proveedor.
    - Provincia: Provincia del proveedor.

- Artículo(AID, Nombre, Tipo):
    - AID: ID del artículo. Clave primaria.
    - Nombre: Nombre del artículo.
    - Tipo: Tipo de artículo.

- Suministra(**PID**, **AID**):
    - PID: ID del proveedor. Clave externa de Proveedor y clave primaria.
    - AID: ID del artículo. Clave externa de Artículo y clave primaria.

- Entrega(**PID**, **AID**, HID, Fecha, Precio, Cantidad):
    - PID: ID del proveedor. Clave externa de Proveedor y clave primaria.
    - AID: ID artículo. Clave externa de Artículo y clave primaria.
    - HID: ID del hotel. Clave externa de Hotel y clave primaria.
    - Fecha: Fecha de entrega. Clave primaria.
    - Precio: Precio de la entrega.
    - Cantidad: Cantidad de artículos de la entrega.

Estas son las tablas iniciales, pero hemos realizado fusión:

- Emp-Trab(**EID**, **HID**, Nombre, Dirección, Salario, DNI, tlf, fecha_contrato, fecha_ini): Es la fusión entre Empleado y Trabaja. Nos quedamos con el EID, HID, y todos los atributos de la tabla Empleado. Así podemos tener el hotel en el que trabaja cada empleado.
- Hotel-Dir(**HID**, **EID**, Nombre, Ciudad, Provincia, n_hab, n_hab2): Fusión entre Hotel y Dirige. Nos quedamos con EID, HID, y los atributos del hotel. Así podemos tener el director de cada hotel.
- Art-Sum(**PID**, **AID**, Nombre, Tipo): Fusión entre Artículo y Suministra. Nos quedamos con el PID, AID, y los atributos de Artículo. Así tendremos al artículo con el proveedor que lo suministra.

Finalmente nos hemos quedado con las tablas:

- Emp-Trab → Empleado
- Hotel-Dir → Hotel
- Cliente
- Reserva
- Proveedor
- Art-Sum → Entrega
- Entrega

Añadimos también, aunque no esté reflejado en el diagrama, una tabla Histórico que contiene EID, HID, la fecha de contrato inicial y la fecha de fin de contrato.

4. Fragmentación de tablas

Voy a dividir en varias partes este apartado. No podemos fragmentar Cliente en base a ningún criterio, y Artículo tampoco tiene ningún criterio sobre el que se pueda fragmentar, ya que hay muchos artículos y no hay ninguna manera de clasificarlos en base a otra tabla.

4.1. Fragmentación Hotel

Hemos decidido fragmentar los hoteles por su provincia correspondiente, ya que para cada BD se asignan 2 provincias (p.ej. Granada contiene los fragmentos de Granada y Jaén). Esto minimizará las consultas remotas.

Número de expresiones conjuntivas: 2^8

Conjunto de predicados simples:

P={
Provincia="Granada" p1
Provincia="Sevilla" p2
Provincia="Cádiz" p3
Provincia="Málaga" p4
Provincia="Córdoba" p5
Provincia="Huelva" p6
Provincia="Jaén" p7
Provincia="Almería" p8
}

Términos de predicado y esquema de Fragmentación:

p1^¬p2^¬p3^¬p4^¬p5^¬p6^¬p7^¬p8 Hotel1=SLp1(Hotel-Dir)
p2^¬p1^¬p3^¬p4^¬p5^¬p6^¬p7^¬p8 Hotel2=SLp2(Hotel-Dir)
p3^¬p2^¬p1^¬p4^¬p5^¬p6^¬p7^¬p8 Hotel3=SLp3(Hotel-Dir)
p4^¬p2^¬p3^¬p1^¬p5^¬p6^¬p7^¬p8 Hotel4=SLp4(Hotel-Dir)
p5^¬p2^¬p3^¬p4^¬p1^¬p6^¬p7^¬p8 Hotel5=SLp5(Hotel-Dir)
p6^¬p2^¬p3^¬p4^¬p5^¬p1^¬p7^¬p8 Hotel6=SLp6(Hotel-Dir)
p7^¬p2^¬p3^¬p4^¬p5^¬p6^¬p1^¬p8 Hotel7=SLp7(Hotel-Dir)
p8^¬p2^¬p3^¬p4^¬p5^¬p6^¬p7^¬p1 Hotel8=SLp8(Hotel-Dir)

Solamente valen los términos de predicado que restringen que un mismo hotel esté en una sola provincia a la vez.

4.2. Fragmentación Proveedor

Los proveedores los fragmentamos, ya que cada uno es asignado a una provincia de las disponibles (Granada ó Sevilla).

Número de expresiones conjuntivas: 2^2

Conjunto de predicados simples:

P={
Provincia=”Granada” p1
Provincia=”Sevilla” p2
}

Términos de predicado y esquema de Fragmentación:

p1^¬p2 Proveedor1=SLp1(Proveedor)
¬p1^p2 Proveedor2=SLp2(Proveedor)

Solamente valen los términos de predicado que restringen que un proveedor esté en una sola provincia a la vez.

4.3. Fragmentación Empleado

Resultado de la fragmentación derivada entre empleados y hoteles. Tiene sentido, ya que así tendremos información sobre qué empleados trabajan en los hoteles de cada provincia.

Esquema de Fragmentación:

Empleado1= Emp-Trab SJNhid Hotel1
Empleado2= Emp-Trab SJNhid Hotel2
Empleado3= Emp-Trab SJNhid Hotel3
Empleado4= Emp-Trab SJNhid Hotel4
Empleado5= Emp-Trab SJNhid Hotel5
Empleado6= Emp-Trab SJNhid Hotel6
Empleado7= Emp-Trab SJNhid Hotel7
Empleado8= Emp-Trab SJNhid Hotel8

De esta manera tendremos fragmentos con los empleados para cada hotel por separado.

4.4. Fragmentación Entrega

Resultado de la fragmentación derivada entre entregas y hoteles. Tiene sentido, ya que así tendremos información sobre qué entregas se dirigen hacia los hoteles de cada provincia.

Esquema de Fragmentación:

Entrega1= Entrega SJNhid Hotel1
Entrega2= Entrega SJNhid Hotel2
Entrega3= Entrega SJNhid Hotel3
Entrega4= Entrega SJNhid Hotel4
Entrega5= Entrega SJNhid Hotel5
Entrega6= Entrega SJNhid Hotel6
Entrega7= Entrega SJNhid Hotel7
Entrega8= Entrega SJNhid Hotel8

De esta manera tendremos las entregas de cada hotel distinto con su fecha, cantidad, etc.

4.5. Fragmentación Reserva

Resultado de la fragmentación derivada entre reservas y hoteles. Tiene sentido, ya que así tendremos información sobre qué reservas se realizan en los hoteles de cada provincia.

Esquema de Fragmentación:

Reserva1= Reservas SJNhid Hotel1
Reserva2= Reservas SJNhid Hotel2
Reserva3= Reservas SJNhid Hotel3
Reserva4= Reservas SJNhid Hotel4
Reserva5= Reservas SJNhid Hotel5
Reserva6= Reservas SJNhid Hotel6
Reserva7= Reservas SJNhid Hotel7
Reserva8= Reservas SJNhid Hotel8

De esta manera tendremos las reservas de cada hotel distinto con su fecha, cantidad, etc.

5. Asignación de tablas

Voy a dividir en varias partes este apartado, al igual que el anterior. Las bases de datos y las localidades son las siguientes:

- Papel1 → Granada
- Papel2 → Cádiz
- Papel3 → Sevilla
- Papel4 → Málaga

5.1. Asignación de Hotel

Asignamos los fragmentos Hotel1 y Hotel7 a la localidad de Granada, puesto que desde ella se controlan los datos de los hoteles de las provincias de Granada y Jaén.

Asignamos los fragmentos Hotel3 y Hotel6 a la localidad de Cádiz, puesto que desde ella se controlan los datos de los hoteles de las provincias de Cádiz y Huelva.

Asignamos los fragmentos Hotel2 y Hotel5 a la localidad de Sevilla, puesto que desde ella se controlan los datos de los hoteles de las provincias de Sevilla y Córdoba.

Asignamos los fragmentos Hotel4 y Hotel8 a la localidad de Málaga, puesto que desde ella se controlan los datos de los hoteles de las provincias de Málaga y Almería.

5.2. Asignación de Proveedor

Asignamos el fragmento Proveedor1 a la localidad de Granada y replicamos este fragmento en la localidad de Málaga.

Asignamos el fragmento Proveedor2 a la localidad de Sevilla y replicamos este fragmento en la localidad de Cádiz.

Hemos replicado los fragmentos Proveedor1 (en Málaga y Proveedor2 (en Cádiz), para que sus datos se encuentren en todas las localidades sobre las que trabajan.

5.3. Asignación de Empleado

Asignamos los fragmentos Empleado1 y Empleado7 a la localidad de Granada, puesto que desde ella se controlan los datos de los hoteles de las provincias de Granada y Jaén.

Asignamos los fragmentos Empleado3 y Empleado6 a la localidad de Cádiz, puesto que desde ella se controlan los datos de los hoteles de las provincias de Cádiz y Huelva.

Asignamos los fragmentos Empleado2 y Empleado5 a la localidad de Sevilla, puesto que desde ella se controlan los datos de los hoteles de las provincias de Sevilla y Córdoba.

Asignamos los fragmentos Empleado4 y Empleado8 a la localidad de Málaga, puesto que desde ella se controlan los datos de los hoteles de las provincias de Málaga y Almería.

5.4. Asignación de Entrega

Asignamos los fragmentos Entrega1 y Entrega7 a la localidad de Granada, puesto que desde ella se controlan los datos de los hoteles de las provincias de Granada y Jaén.

Asignamos los fragmentos Entrega3 y Entrega6 a la localidad de Cádiz, puesto que desde ella se controlan los datos de los hoteles de las provincias de Cádiz y Huelva.

Asignamos los fragmentos Entrega2 y Entrega5 a la localidad de Sevilla, puesto que desde ella se controlan los datos de los hoteles de las provincias de Sevilla y Córdoba.

Asignamos los fragmentos Entrega4 y Entrega8 a la localidad de Málaga, puesto que desde ella se controlan los datos de los hoteles de las provincias de Málaga y Almería.

5.5. Asignación de Reserva

Asignamos los fragmentos Reserva1 y Reserva7 a la localidad de Granada, puesto que desde ella se controlan los datos de los hoteles de las provincias de Granada y Jaén.

Asignamos los fragmentos Reserva3 y Reserva6 a la localidad de Cádiz, puesto que desde ella se controlan los datos de los hoteles de las provincias de Cádiz y Huelva.

Asignamos los fragmentos Reserva2 y Reserva5 a la localidad de Sevilla, puesto que desde ella se controlan los datos de los hoteles de las provincias de Sevilla y Córdoba.

Asignamos los fragmentos Reserva4 y Reserva8 a la localidad de Málaga, puesto que desde ella se controlan los datos de los hoteles de las provincias de Málaga y Almería.

5.6. Consideraciones finales

Los clientes los hemos decidido replicar en todas las localidades, ya que todas van a llamar a esa tabla regularmente por las reservas que se realicen y los nuevos clientes que se creen.

Los artículos los replicamos en todas las localidades, ya que de ahí vienen los proveedores y el producto natural no requeriría de referencias remotas.

La tabla Histórico la hemos decidido alojar en Sevilla, y al ser una tabla a la que no es muy común acceder a ella pues todas las localidades accederán remotamente.

Hemos alojado los distintos datos de la misma provincia en sus provincias correspondientes (Hotel, Empleado, Entrega y Reserva), ya que así tenemos cierta autonomía local en las localidades.

6. Tablas, grants, drops y vistas

En donde está NN ó N van los números correspondientes para cada tabla, y lo mismo para los grants y drops. En las tablas hemos decidido juntar las dos provincias de cada localidad en una sola tabla. Las tablas en cada localidad se llaman así:

- Papel1:
    - Hotel17
    - Empleado17
    - Entrega17
    - Reserva17
    - Proveedor1
    - Cliente
    - Articulo

- Papel2:
    - Hotel36
    - Empleado36
    - Entrega36
    - Reserva36
    - Proveedor2
    - Cliente
    - Articulo

- Papel3:
    - Hotel25
    - Empleado25
    - Entrega25
    - Reserva25
    - Proveedor2
    - Cliente
    - Articulo
    - Historico

- Papel4:
    - Hotel48
    - Empleado48
    - Entrega48
    - Reserva48
    - Proveedor1
    - Cliente
    - Articulo

6.1. Tablas

```
CREATE TABLE HOTELNN(
	HID NUMBER NOT NULL,
	EID NUMBER,
	NOMBRE VARCHAR(50) NOT NULL,
	CIUDAD VARCHAR(20) NOT NULL,
	PROVINCIA VARCHAR(10) NOT NULL,
	N_HAB NUMBER NOT NULL,
	N_HAB2 NUMBER NOT NULL,
	PRIMARY KEY(HID)
);

CREATE TABLE PROVEEDORN(
	PID NUMBER NOT NULL,
	NOMBRE VARCHAR(50) NOT NULL,
	PROVINCIA VARCHAR(10) NOT NULL,
	PRIMARY KEY(PID)
);

CREATE TABLE ARTICULO(
	PID NUMBER NOT NULL,
	AID NUMBER NOT NULL,
	NOMBRE VARCHAR(50) NOT NULL,
	TIPO VARCHAR(50) NOT NULL,
	PRIMARY KEY(PID, AID)
);

CREATE TABLE ENTREGANN(
	PID NUMBER NOT NULL,
	AID NUMBER NOT NULL,
	HID NUMBER NOT NULL REFERENCES HOTELNN(HID),
	FECHA DATE NOT NULL,
	PRECIO NUMBER NOT NULL,
	CANTIDAD NUMBER NOT NULL,
	PRIMARY KEY(PID, AID, HID, FECHA),
	FOREIGN KEY(PID,AID) REFERENCES ARTICULO(PID, AID)
);

CREATE TABLE EMPLEADONN(
	EID NUMBER NOT NULL,
	HID NUMBER NOT NULL REFERENCES HOTELNN(HID),
	NOMBRE VARCHAR(50) NOT NULL,
	DIRECCION VARCHAR(70) NOT NULL,
	SALARIO NUMBER NOT NULL,
	DNI NUMBER NOT NULL,
	TLF NUMBER NOT NULL,
	FECHA_CONTRATO DATE NOT NULL,
	FECHA_INI DATE NOT NULL,
	PRIMARY KEY(EID)
);
```

```
CREATE TABLE CLIENTE(
      CID NUMBER NOT NULL,
      NOMBRE VARCHAR(50) NOT NULL,
      DNI NUMBER NOT NULL,
      TLF NUMBER NOT NULL,
      PRIMARY KEY(CID)
);

CREATE TABLE RESERVANN(
      CID NUMBER NOT NULL REFERENCES CLIENTE(CID),
      HID NUMBER NOT NULL REFERENCES HOTELNN(HID),
      FECHA_FIN DATE NOT NULL,
      FECHA_INI DATE NOT NULL,
      TIPO_HAB VARCHAR(20) NOT NULL,
      PRECIO NUMBER NOT NULL,
      PRIMARY KEY(CID, HID, FECHA_FIN, FECHA_INI)
);

CREATE TABLE HISTORICO(
      EID NUMBER NOT NULL,
      HID NUMBER NOT NULL,
      FECHA_INI DATE NOT NULL,
      FECHA_FIN DATE NOT NULL,
      PRIMARY KEY(EID, HID)
);
```

6.2. Grants

```
GRANT DELETE, INSERT, UPDATE, SELECT
ON ARTICULO
TO PAPELN,PAPELN,PAPELN;
commit;

GRANT DELETE, INSERT, UPDATE, SELECT
ON CLIENTE
TO PAPELN,PAPELN,PAPELN;
commit;

GRANT DELETE, INSERT, UPDATE, SELECT
ON EMPLEADONN
TO PAPELN,PAPELN,PAPELN;
commit;

GRANT DELETE, INSERT, UPDATE, SELECT
ON ENTREGANN
TO PAPELN,PAPELN,PAPELN;
commit;
```

```
GRANT DELETE, INSERT, UPDATE, SELECT
ON HOTELNN
TO PAPELN,PAPELN,PAPELN;
commit;

GRANT DELETE, INSERT, UPDATE, SELECT
ON PROVEEDORN
TO PAPELN,PAPELN,PAPELN;
commit;

GRANT DELETE, INSERT, UPDATE, SELECT
ON RESERVANN
TO PAPELN,PAPELN,PAPELN;
commit;

GRANT DELETE, INSERT, UPDATE, SELECT
ON HISTORICO
TO PAPELN,PAPELN,PAPELN;
commit;
```

6.3. Drops

```
DROP TABLE HISTORICO;

DROP TABLE EMPLEADONN;

DROP TABLE RESERVA25;

DROP TABLE CLIENTE;

DROP TABLE ENTREGANN;

DROP TABLE HOTELNN;

DROP TABLE ARTICULO;

DROP TABLE PROVEEDORN;
```

6.4. Vistas

Hemos creado las vistas para poder consultar las tablas en conjunto y poder realizar una correcta recuperación sin repetidos. Estas ya tienen los números correspondientes, y están en todas las localidades.

Para hacer drop solamente debemos escribir `DROP VIEW NOMBREVISTA;`, y se borrará de la localidad la vista escogida. Las vistas disponibles son:

```sql
CREATE OR REPLACE VIEW EMPLEADO AS
SELECT * FROM PAPEL1.EMPLEADO17
UNION
SELECT * FROM PAPEL2.EMPLEADO36
UNION
SELECT * FROM PAPEL3.EMPLEADO25
UNION
SELECT * FROM PAPEL4.EMPLEADO48;

-- RESERVA
CREATE OR REPLACE VIEW RESERVA AS
SELECT * FROM PAPEL1.RESERVA17
UNION
SELECT * FROM PAPEL2.RESERVA36
UNION
SELECT * FROM PAPEL3.RESERVA25
UNION
SELECT * FROM PAPEL4.RESERVA48;

-- HOTEL
CREATE OR REPLACE VIEW HOTEL AS
SELECT * FROM PAPEL1.HOTEL17
UNION
SELECT * FROM PAPEL2.HOTEL36
UNION
SELECT * FROM PAPEL3.HOTEL25
UNION
SELECT * FROM PAPEL4.HOTEL48;

-- ENTREGA
CREATE OR REPLACE VIEW ENTREGA AS
SELECT * FROM PAPEL1.ENTREGA17
UNION
SELECT * FROM PAPEL2.ENTREGA36
UNION
SELECT * FROM PAPEL3.ENTREGA25
UNION
SELECT * FROM PAPEL4.ENTREGA48;

-- PROVEEDOR: SOLAMENTE RECUPERAMOS PROVEEDOR1 Y PROVEEDOR2 DE GRANADA
-- Y SEVILLA PARA EVITAR REPETIDOS
CREATE OR REPLACE VIEW PROVEEDOR AS
SELECT * FROM PAPEL1.PROVEEDOR1
UNION
SELECT * FROM PAPEL3.PROVEEDOR2;

-- HISTORIAL: SE CREA PARA ACCEDER FÁCILMENTE A LA TABLA DE LA
-- LOCALIDAD DE SEVILLA
CREATE OR REPLACE VIEW HISTORIAL AS
SELECT * FROM PAPEL3.HISTORICO;
```

7. Disparadores

Las restricciones de llave única, de integridad referencial y no nulas las hemos comprobado tanto en procedures como en triggers. Algunos triggers se han comprobado en los procedure, ya que no se podía de otra manera distinta. Las N y NN son números y se especifican en la explicación del procedure.

El trigger 16 lo hemos puesto en el procedure directamente, ya que de por sí se necesitaba comprobar que la localidad fuera Granada o Sevilla, y allí podíamos poner el código de error.

7.1. DIRECTOR_HOTEL(7)

Este disparador está asociado a la tabla HotelNN, con previo aviso a la inserción de una nueva tupla. Se encarga de comprobar que el empleado que vamos a asignar como director existe, en el caso de que no exista se muestra un mensaje de error correspondiente. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER DIRECTOR_HOTEL
BEFORE INSERT OR UPDATE ON PAPELN.HOTELNN FOR EACH ROW
DECLARE
    ECOUNT NUMBER;
BEGIN
    IF(:OLD.EID != :NEW.EID) THEN
        -- Comprobamos que el director es un empleado.
        SELECT COUNT (*)
        INTO ECOUNT
        FROM PAPELN.EMPLEADONN
        WHERE EID = :NEW.EID;

        IF (ECOUNT = 0) THEN
            RAISE_APPLICATION_ERROR(-20076,'Este empleado no existe');
        END IF;
    END IF;
END;
```

7.2. UN_SOLO_HOTEL(8, 9)

Este disparador está asociado a la tabla HotelNN, con previo aviso a la inserción de una nueva tupla. Se encarga de controlar que cada director tenga asociado solo un hotel, primero comprueba que el hotel existe y posteriormente si dicho empleado ya es director de otro hotel. En cualquiera de los dos casos negativos se muestra un mensaje de error informando del problema. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER UN_SOLO_HOTEL
BEFORE INSERT ON PAPELN.HOTELNN FOR EACH ROW
DECLARE
    HCOUNT NUMBER;
```

```
    ECOUNT NUMBER;
BEGIN
    -- Comprobamos que el hotel no exista.
    SELECT COUNT(*)
    INTO HCOUNT
    FROM HOTEL
    WHERE HID = :NEW.HID;

    IF (HCOUNT > 0) THEN
        RAISE_APPLICATION_ERROR(-20077,'El hotel ya existe');
    END IF;

    -- Comprobamos que el empleado no dirija otros hoteles
    SELECT COUNT (*)
    INTO ECOUNT
    FROM HOTEL
    WHERE EID = :NEW.EID;

    IF (ECOUNT > 0) THEN
        RAISE_APPLICATION_ERROR(-20078,'Este empleado ya dirige un
hotel.');
    END IF;
END;
```

7.3. INSERTAR_EMPLEADO(11)

Este disparador está asociado a la tabla EMPLEADONN, con previo aviso a la inserción de una nueva tupla. Se encarga de comprobar que cuando añadimos un nuevo empleado, la fecha del contrato es anterior o igual a la fecha en la que el empleado comienza a trabajar, en caso contrario se muestra un mensaje de error. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER INSERTAR_EMPLEADO
BEFORE INSERT ON PAPELN.EMPLEADONN FOR EACH ROW
BEGIN
    IF(:NEW.FECHA_CONTRATO > :NEW.FECHA_INI) THEN
        RAISE_APPLICATION_ERROR(-20079,'Fecha de contrato mayor a la de
inicio');
    END IF;
END;
```

7.4. SALARIO_MAYOR(10)

Este disparador está asociado a la tabla EMPLEADONN, con previo aviso a la actualización de una tupla. Se encarga de comprobar que el salario de un empleado no se disminuye al actualizar sus datos, en ese caso se muestra un mensaje de error. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER SALARIO_MAYOR
BEFORE UPDATE ON PAPELN.EMPLEADONN FOR EACH ROW
BEGIN
    IF(:NEW.SALARIO < :OLD.SALARIO) THEN
        RAISE_APPLICATION_ERROR(-20080,'El salario no puede
disminuir.');
    END IF;
END;
```

7.5. INSERTAR_HISTORICO(12)

Este disparador está asociado a la tabla HISTORICO, con previo aviso a la inserción de una nueva tupla. Se encarga de comprobar que la fecha en la que un empleado comienza a trabajar en un hotel debe de ser igual o posterior a la fecha de fin en el hotel en el que estaba asignado previamente, en caso contrario se muestra un mensaje de error. Este trigger solamente se encuentra en Papel3 (Sevilla).

```
CREATE OR REPLACE TRIGGER INSERTAR_HISTORICO
BEFORE INSERT ON PAPEL3.HISTORICO FOR EACH ROW
DECLARE
BEGIN


    IF(:NEW.FECHA_INI > :NEW.FECHA_FIN) THEN
        RAISE_APPLICATION_ERROR(-20081,'La fecha de contrato nueva no
puede ser antes que la final.');
    END IF;
END;
```

7.6. ESPACIO_RESERVA(5)

Este disparador está asociado a la tabla RESERVANN, con previo aviso a la inserción de una nueva tupla. Se encarga de comprobar que la fecha de entrada de una reserva no es posterior a la de salida, en caso de error se muestra un mensaje informando del problema. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER ESPACIO_RESERVA
BEFORE INSERT OR UPDATE ON PAPELN.RESERVANN FOR EACH ROW
DECLARE
    NUM_HABS NUMBER;
BEGIN
    IF :NEW.FECHA_FIN < :NEW.FECHA_INI THEN
```

```
        RAISE_APPLICATION_ERROR(-20082,'Fecha de salida no puede ser
menor a la de entrada');
    END IF;
END;
```

7.7. RESERVA_DISPONIBLE(4, 6)

Este disparador está asociado a la tabla RESERVANN, con previo aviso a la inserción de una nueva tupla. Se encarga de comprobar que una reserva se encuentra disponible, primero se comprueba que el cliente que está realizando la reserva no tiene otra reserva durante las mismas fechas solicitadas, después se comprueba que el hotel al que se le va a realizar la reserva cuenta con habitaciones disponibles. Si alguno de estos dos procesos fallase se muestra un mensaje de error. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER RESERVA_DISPONIBLE
BEFORE INSERT ON PAPELN.RESERVANN FOR EACH ROW
DECLARE
    NHABITACIONES NUMBER;
    RESERVAS NUMBER;
    RESERVASHOTEL NUMBER;
BEGIN
    -- Comprobamos que no haya reservas en esas fechas por el mismo
cliente.
    SELECT COUNT(*)
    INTO RESERVAS
    FROM RESERVA
    WHERE   CID=:NEW.CID
                AND (FECHA_INI<=:NEW.FECHA_INI OR
FECHA_INI<=:NEW.FECHA_FIN)
                AND (FECHA_FIN>=:NEW.FECHA_INI OR
FECHA_FIN>=:NEW.FECHA_FIN);

    IF(RESERVAS > 0) THEN
        RAISE_APPLICATION_ERROR(-20083,'El cliente ya tiene una reserva
en esas fechas.');
    END IF;

    IF (:NEW.TIPO_HAB = 'Sencilla') THEN
        SELECT N_HAB
        INTO NHABITACIONES
        FROM PAPEL3.HOTEL25
        WHERE HID = :NEW.HID;

    ELSIF (:NEW.TIPO_HAB = 'Doble') THEN
        SELECT N_HAB2
        INTO NHABITACIONES
        FROM PAPEL3.HOTEL25
        WHERE HID = :NEW.HID;
    ELSE
```

```
        RAISE_APPLICATION_ERROR(-20084,'Tipo de habitación no
válida.');
    END IF;

    -- Comprobamos que hayan habitaciones libres.
    SELECT COUNT(*)
    INTO RESERVASHOTEL
    FROM PAPEL3.RESERVA25
    WHERE   HID=:NEW.HID
            AND (FECHA_INI<=:NEW.FECHA_INI OR
FECHA_INI<=:NEW.FECHA_FIN)
            AND (FECHA_FIN>=:NEW.FECHA_INI OR
FECHA_FIN>=:NEW.FECHA_FIN);

    IF(RESERVASHOTEL = NHABITACIONES) THEN
        RAISE_APPLICATION_ERROR(-20085,'No queda espacio en el hotel en
esas fechas.');
    END IF;
END;
```

7.8. EXISTE_PROVEEDOR(EXTRA)

Este disparador está asociado a la tabla PROVEEDORN, con previo aviso a la inserción de una nueva tupla. Se comprueba que el proveedor que queremos añadir no exista, en caso contrario se informará del problema con un mensaje de error. De este trigger hay uno por localidad. Si se quiere insertar desde Granada, Jaén, Málaga o Almería podemos comprobar desde Papel1.Proveedor1 ó Papel4.Proveedor1, y si queremos insertar desde Córdoba, Sevilla, Cádiz o Huelva podemos comprobar desde Papel2.Proveedor2 ó Papel3.Proveedor2.

```
CREATE OR REPLACE TRIGGER EXISTE_PROVEEDOR
BEFORE INSERT ON PAPELN.PROVEEDORN FOR EACH ROW
DECLARE
AUXCOUNT NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO AUXCOUNT
    FROM PROVEEDOR
    WHERE PID = :NEW.PID;
    IF(AUXCOUNT = 1) THEN
        SELECT COUNT(*)
        INTO AUXCOUNT
        FROM PAPELN.PROVEEDORN
        WHERE PID = :NEW.PID;

        IF(AUXCOUNT = 0) THEN
            RAISE_APPLICATION_ERROR(-20068,'Proveedor existente.');
        END IF;
    END IF;
END;
```

7.9. BORRAR_PROVEEDOR(19)

Este disparador está asociado a la tabla PROVEEDORN, con previo aviso al borrado de una tupla. Se encarga de comprobar que el proveedor que queremos eliminar no tiene ningún suministro asignado, en caso contrario se muestra un mensaje de error. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER BORRAR_PROVEEDOR
BEFORE DELETE ON PAPELN.PROVEEDORN FOR EACH ROW
DECLARE
      PROVEEDOR_ACTIVO NUMBER;
BEGIN
      SELECT COUNT(*)
    INTO PROVEEDOR_ACTIVO
      FROM ENTREGA
      WHERE PID = :OLD.PID;

      IF(PROVEEDOR_ACTIVO > 0) THEN
            RAISE_APPLICATION_ERROR(-20087,'Este proveedor no puede
eliminarse, sigue activo');
    END IF;
END;
```

7.10. TIPO_ARTICULO(13)

Este disparador está asociado a la tabla ARTICULO, con previo aviso a la inserción de una nueva tupla. Se encarga de comprobar que el nuevo artículo sea de los tipos A,B,C o D, si no es de ninguno de esos se muestra un mensaje de error. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER TIPO_ARTICULO
BEFORE INSERT ON PAPELN.ARTICULO FOR EACH ROW
BEGIN
      IF(:NEW.TIPO != 'A' AND :NEW.TIPO != 'B' AND :NEW.TIPO != 'C' AND
:NEW.TIPO != 'D') THEN
            RAISE_APPLICATION_ERROR(-20088,'El tipo de articulo
introducido no se admite en la base de datos.');
    END IF;
END;
```

7.11. UNICO_ARTICULO(15)

Este disparador está asociado a la tabla ARTICULO, con previo aviso a la inserción de una nueva tupla. Se comprueba que un artículo solo pueda ser suministrado por un

proveedor de Granada y otro de Sevilla, si no se cumple esa condición se muestra un mensaje de error. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER UNICO_ARTICULO
BEFORE INSERT ON PAPELN.ARTICULO FOR EACH ROW
DECLARE
    ACOUNT NUMBER;
    PROVID NUMBER;
    LOCALIDAD VARCHAR(20);
    LOCALIDADNUEVO VARCHAR(20);
BEGIN
    SELECT COUNT (*)
    INTO ACOUNT
    FROM ARTICULO
    WHERE AID = :NEW.AID;

    IF (ACOUNT = 1) THEN
        SELECT PID
        INTO PROVID
        FROM ARTICULO
        WHERE AID = :NEW.AID;

        SELECT PROVINCIA
        INTO LOCALIDAD
        FROM PROVEEDOR
        WHERE PID = PROVID;

        SELECT PROVINCIA
        INTO LOCALIDADNUEVO
        FROM PROVEEDOR
        WHERE PID = :NEW.PID;

        IF(LOCALIDAD = LOCALIDADNUEVO) THEN
            RAISE_APPLICATION_ERROR(-20089,'Los artículos los
suministra un solo proveedor por provincia.');
        END IF;
    ELSIF(ACOUNT > 1) THEN
        RAISE_APPLICATION_ERROR(-20090,'Ya hay un proveedor por
provincia que suministra el artículo.');
    END IF;
END;
```

7.12. BORRAR_ARTICULO(20)

Este disparador está asociado a la tabla ARTICULO, con previo aviso al borrado de una tupla. Se encarga de comprobar que el artículo que queremos eliminar no tiene ningún suministro asignado, en caso contrario se muestra un mensaje de error. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER BORRAR_ARTICULO
BEFORE DELETE ON PAPELN.ARTICULO FOR EACH ROW
DECLARE
      ARTICULO_ACTIVO NUMBER;
BEGIN

      SELECT COUNT(*) INTO ARTICULO_ACTIVO
      FROM ENTREGA
      WHERE AID=:OLD.AID AND CANTIDAD != 0;

      IF(ARTICULO_ACTIVO != 0) THEN
            RAISE_APPLICATION_ERROR(-20091,'Este articulo no puede
eliminarse, sigue activo');
    END IF;
END;
```

7.13. PRECIO_ENTREGA(14)

Este disparador está asociado a la tabla ENTREGANN, con previo aviso a la inserción de una nueva tupla. Se encarga de comprobar que el precio por unidad de un artículo en un suministro a un hotel no sea inferior al precio por unidad de un artículo de un suministro anterior al mismo hotel, si no se cumple esta condición se muestra un mensaje de error. De este trigger hay uno por localidad.

```
CREATE OR REPLACE TRIGGER PRECIO_ENTREGA
BEFORE INSERT ON PAPELN.ENTREGANN FOR EACH ROW
DECLARE
    PRECIO_AUX NUMBER;
BEGIN
    BEGIN
        SELECT PRECIO
        INTO PRECIO_AUX
        FROM ENTREGA
        WHERE   AID = :NEW.AID
                AND HID = :NEW.HID
                AND PID = :NEW.PID
        ORDER BY FECHA ASC;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            PRECIO_AUX := 0;
    END;

    IF(:NEW.PRECIO < PRECIO_AUX) THEN
        RAISE_APPLICATION_ERROR(-20092,'El precio del artículo del
suministro no puede ser inferior.');
    END IF;
END;
```

7.14. PRODUCTOS_N(17, 18)

Este disparador está asociado a la tabla ENTREGANN, con previo aviso a la inserción de una nueva tupla. Se encarga de comprobar que los hoteles de Córdoba, Sevilla, Cádiz y Huelva no soliciten suministros a proveedores de Granada y que los hoteles de Granada, Jaén, Málaga o Almería no soliciten suministros a proveedores de Sevilla, y en caso contrario se muestra un mensaje de error. De este trigger hay uno por localidad. Si se quiere insertar desde Granada, Jaén, Málaga o Almería podemos comprobar desde Papel1.Proveedor1 ó Papel4.Proveedor1, y si queremos insertar desde Córdoba, Sevilla, Cádiz o Huelva podemos comprobar desde Papel2.Proveedor2 ó Papel3.Proveedor2.

```
CREATE OR REPLACE TRIGGER PRODUCTOS_N
BEFORE INSERT ON PAPELN.ENTREGANN FOR EACH ROW
DECLARE
      PROVEEDOR_N NUMBER;
BEGIN
      SELECT COUNT(*) INTO PROVEEDOR_N
      FROM PAPEL1.PROVEEDOR1
      WHERE PID=:NEW.PID;

      IF(PROVEEDOR_N != 0) THEN
            RAISE_APPLICATION_ERROR(-20093,'Córdoba, Sevilla, Cádiz Y
Huelva/Granada, Jaén, Málaga o Almería, no pueden solicitar este
proveedor');
    END IF;
END;
```

8. Procedimientos.

8.1. INSERT_ARTICULO(14)

Este procedimiento recibe como argumentos los atributos de cada artículo y realiza un INSERT en cada una de las tablas ARTICULO de todas las localidades.

```
CREATE OR REPLACE PROCEDURE INSERT_ARTICULO(    COD_PROVEEDOR NUMBER,
                                                COD_ARTICULO NUMBER,
                                                NOMBRE_ARTICULO VARCHAR,
                                                TIPO_ARTICULO VARCHAR)IS

ARTCOUNT NUMBER;

BEGIN
    INSERT INTO PAPEL1.ARTICULO
    VALUES (COD_PROVEEDOR, COD_ARTICULO, NOMBRE_ARTICULO,
TIPO_ARTICULO);

    INSERT INTO PAPEL2.ARTICULO
    VALUES (COD_PROVEEDOR, COD_ARTICULO, NOMBRE_ARTICULO,
TIPO_ARTICULO);

    INSERT INTO PAPEL3.ARTICULO
    VALUES (COD_PROVEEDOR, COD_ARTICULO, NOMBRE_ARTICULO,
TIPO_ARTICULO);

    INSERT INTO PAPEL4.ARTICULO
    VALUES (COD_PROVEEDOR, COD_ARTICULO, NOMBRE_ARTICULO,
TIPO_ARTICULO);
END;
```

8.2. DELETE_ARTICULO(15)

Este procedimiento recibe como argumento el código del artículo que queremos eliminar y lo borra de las tablas ARTICULO de todas las localidades.

```
CREATE OR REPLACE PROCEDURE DELETE_ARTICULO( COD_ART NUMBER ) IS

ARTCOUNT NUMBER;

BEGIN
      SELECT COUNT (*)
      INTO ARTCOUNT
      FROM ARTICULO
      WHERE COD_ART = AID;

      IF(ARTCOUNT > 0) THEN
            DELETE FROM PAPEL1.ENTREGA17 WHERE COD_ART = AID;
```

```
            DELETE FROM PAPEL4.ENTREGA48 WHERE COD_ART = AID;

            DELETE FROM PAPEL2.ENTREGA36 WHERE COD_ART = AID;

            DELETE FROM PAPEL3.ENTREGA25 WHERE COD_ART = AID;

            DELETE FROM PAPEL1.ARTICULO WHERE COD_ART = AID;

            DELETE FROM PAPEL2.ARTICULO WHERE COD_ART = AID;

            DELETE FROM PAPEL3.ARTICULO WHERE COD_ART = AID;

            DELETE FROM PAPEL4.ARTICULO WHERE COD_ART = AID;
    ELSE
            RAISE_APPLICATION_ERROR(-20005,'Este codigo de articulo no
se encuentra registrado.');
    END IF;
END;
```

8.3. INSERT_CLIENTE(7)

Este procedimiento recibe como argumentos los atributos de cada cliente y realiza un INSERT en cada una de las tablas CLIENTE de todas las localidades.

```
CREATE OR REPLACE PROCEDURE INSERT_CLIENTE( COD_CLIENTE NUMBER,
                                            NOMBRE_CLIENTE VARCHAR,
                                            DNI_CLIENTE NUMBER,
                                            TLF_CLIENTE NUMBER) IS

CLIENTCOUNT NUMBER;

BEGIN
    SELECT COUNT (*)
    INTO CLIENTCOUNT
    FROM PAPEL1.CLIENTE
    WHERE   COD_CLIENTE = CID;

    IF(CLIENTCOUNT > 0) THEN
        RAISE_APPLICATION_ERROR(-20006,'El cliente ya existe.');
    ELSE
        INSERT INTO PAPEL1.CLIENTE
        VALUES (COD_CLIENTE, NOMBRE_CLIENTE, DNI_CLIENTE, TLF_CLIENTE);

        INSERT INTO PAPEL2.CLIENTE
        VALUES (COD_CLIENTE, NOMBRE_CLIENTE, DNI_CLIENTE, TLF_CLIENTE);

        INSERT INTO PAPEL3.CLIENTE
        VALUES (COD_CLIENTE, NOMBRE_CLIENTE, DNI_CLIENTE, TLF_CLIENTE);
```

```
        INSERT INTO PAPEL4.CLIENTE
        VALUES (COD_CLIENTE, NOMBRE_CLIENTE, DNI_CLIENTE, TLF_CLIENTE);
    END IF;
END;
```

8.4. INSERT_EMPLEADO(1)

Este procedimiento recibe como argumentos los atributos de cada empleado y realiza un INSERT en la tabla EMPLEADO de la localidad correspondiente, esta la obtenemos a partir del COD_HOTEL introducido como argumento, ya que la localidad de dicho hotel será la misma en la que almacenaremos a este empleado.

```
CREATE OR REPLACE PROCEDURE INSERT_EMPLEADO(     COD_EMPLEADO NUMBER,
                                                 COD_HOTEL NUMBER,
                                                 NOMBRE_EMP VARCHAR,
                                                 DIRECCION_EMP VARCHAR,
                                                 SALARIO_EMP NUMBER,
                                                 DNI_EMP NUMBER,
                                                 TLF_EMP NUMBER,
                                                 FECHA_CONT_EMP DATE,
                                                  FECHA_INI_EMP DATE) IS

LOCALIDAD VARCHAR(20);
ECOUNT NUMBER;

BEGIN
    SELECT COUNT(*)
    INTO ECOUNT
    FROM EMPLEADO
    WHERE EID = COD_EMPLEADO;

    IF(ECOUNT = 0) THEN
        SELECT PROVINCIA
        INTO LOCALIDAD
        FROM HOTEL
        WHERE HID = COD_HOTEL;

        IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
            INSERT INTO PAPEL1.EMPLEADO17
            VALUES (COD_EMPLEADO, COD_HOTEL, NOMBRE_EMP ,
DIRECCION_EMP, SALARIO_EMP, DNI_EMP, TLF_EMP, FECHA_CONT_EMP,
FECHA_INI_EMP);


        ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
            INSERT INTO PAPEL2.EMPLEADO36
            VALUES (COD_EMPLEADO, COD_HOTEL, NOMBRE_EMP ,
DIRECCION_EMP, SALARIO_EMP, DNI_EMP, TLF_EMP, FECHA_CONT_EMP,
FECHA_INI_EMP);
```

```
        ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
            INSERT INTO PAPEL3.EMPLEADO25
            VALUES (COD_EMPLEADO, COD_HOTEL, NOMBRE_EMP ,
DIRECCION_EMP, SALARIO_EMP, DNI_EMP, TLF_EMP, FECHA_CONT_EMP,
FECHA_INI_EMP);
        ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
            INSERT INTO PAPEL4.EMPLEADO48
            VALUES (COD_EMPLEADO, COD_HOTEL, NOMBRE_EMP ,
DIRECCION_EMP, SALARIO_EMP, DNI_EMP, TLF_EMP, FECHA_CONT_EMP,
FECHA_INI_EMP);
        ELSE
            RAISE_APPLICATION_ERROR(-20007,'LOCALIDAD NO VÁLIDA');
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20008,'Empleado ya existe.');
    END IF;
END;
```

8.5. CAMBIAR_SUCURSAL_EMPLEADO(4)

Este procedimiento recibe como argumentos COD_EMPLEADO, COD_HOTEL, FECHA_CONTRATO_FINAL, FECHA_INI_EMP y realiza un UPDATE a los datos del empleado que queramos cambiar de hotel y un INSERT en el histórico registrando este cambio.

```
CREATE OR REPLACE PROCEDURE CAMBIAR_SUCURSAL_EMPLEADO(
                                                COD_EMPLEADO NUMBER,
                                                COD_HOTEL NUMBER,
                                                FECHA_CONTRATO_FINAL DATE,
                                                FECHA_INI_EMP DATE) IS

FECHA_CONTRATO_ANTIGUA DATE;
HOTEL_ANTIGUO NUMBER;
LOCALIDAD VARCHAR(20);
ECOUNT NUMBER;

BEGIN
    SELECT HID
    INTO HOTEL_ANTIGUO
    FROM EMPLEADO
    WHERE EID = COD_EMPLEADO;

    IF(HOTEL_ANTIGUO != COD_HOTEL) THEN
        SELECT COUNT(*)
        INTO ECOUNT
        FROM EMPLEADO
        WHERE EID = COD_EMPLEADO;

        IF(ECOUNT > 0) THEN
            SELECT PROVINCIA
```

```
            INTO LOCALIDAD
            FROM HOTEL
            WHERE HID = HOTEL_ANTIGUO;

            IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
                SELECT FECHA_CONTRATO
                INTO FECHA_CONTRATO_ANTIGUA
                FROM PAPEL1.EMPLEADO17
                WHERE COD_EMPLEADO = EID;

                UPDATE PAPEL1.EMPLEADO17
                SET HID = COD_HOTEL,
                    FECHA_CONTRATO = FECHA_INI_EMP,
                    FECHA_INI = FECHA_INI_EMP
                WHERE   COD_EMPLEADO = EID;

                INSERT INTO PAPEL3.HISTORICO
                VALUES (COD_EMPLEADO, HOTEL_ANTIGUO,
FECHA_CONTRATO_ANTIGUA, FECHA_CONTRATO_FINAL);

            ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
                SELECT FECHA_CONTRATO
                INTO FECHA_CONTRATO_ANTIGUA
                FROM PAPEL2.EMPLEADO36
                WHERE COD_EMPLEADO = EID;

                UPDATE PAPEL2.EMPLEADO36
                SET HID = COD_HOTEL,
                    FECHA_CONTRATO = FECHA_INI_EMP,
                    FECHA_INI = FECHA_INI_EMP
                WHERE   COD_EMPLEADO = EID;

                INSERT INTO PAPEL3.HISTORICO
                VALUES (COD_EMPLEADO, HOTEL_ANTIGUO,
FECHA_CONTRATO_ANTIGUA, FECHA_CONTRATO_FINAL);

            ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
                SELECT FECHA_CONTRATO
                INTO FECHA_CONTRATO_ANTIGUA
                FROM PAPEL3.EMPLEADO25
                WHERE COD_EMPLEADO = EID;

                UPDATE PAPEL3.EMPLEADO25
                SET HID = COD_HOTEL,
                    FECHA_CONTRATO = FECHA_INI_EMP,
                    FECHA_INI = FECHA_INI_EMP
                WHERE   COD_EMPLEADO = EID;

                INSERT INTO PAPEL3.HISTORICO
                VALUES (COD_EMPLEADO, HOTEL_ANTIGUO,
FECHA_CONTRATO_ANTIGUA, FECHA_CONTRATO_FINAL);
```

```
            ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
                SELECT FECHA_CONTRATO
                INTO FECHA_CONTRATO_ANTIGUA
                FROM PAPEL4.EMPLEADO48
                WHERE COD_EMPLEADO = EID;

                UPDATE PAPEL4.EMPLEADO48
                SET HID = COD_HOTEL,
                    FECHA_CONTRATO = FECHA_INI_EMP,
                    FECHA_INI = FECHA_INI_EMP
                WHERE   COD_EMPLEADO = EID;

                INSERT INTO PAPEL3.HISTORICO
                VALUES (COD_EMPLEADO, HOTEL_ANTIGUO,
FECHA_CONTRATO_ANTIGUA, FECHA_CONTRATO_FINAL);
            ELSE
                RAISE_APPLICATION_ERROR(-20009,'LOCALIDAD NO VÁLIDA');
            END IF;
        ELSE
            RAISE_APPLICATION_ERROR(-20010,'Empleado no existe.');
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20011,'Es el mismo hotel.');
    END IF;
END;
```

8.6. SALARIO_EMPLEADO(3)

Este procedimiento recibe como argumentos COD_EMPLEADO y SALARIO_EMP, realiza un UPDATE sobre los datos del empleado cambiando su salario por SALARIO_EMP.

```
CREATE OR REPLACE PROCEDURE SALARIO_EMPLEADO(   COD_EMPLEADO NUMBER,
                                                SALARIO_EMP NUMBER) IS

LOCALIDAD VARCHAR(20);
HOTEL_EMP NUMBER;
ECOUNT NUMBER;

BEGIN
    SELECT COUNT(*)
    INTO ECOUNT
    FROM EMPLEADO
    WHERE EID = COD_EMPLEADO;

    IF(ECOUNT > 0) THEN
        SELECT HID
        INTO HOTEL_EMP
        FROM EMPLEADO
        WHERE EID = COD_EMPLEADO;
```

```
        SELECT PROVINCIA
        INTO LOCALIDAD
        FROM HOTEL
        WHERE HID = HOTEL_EMP;

        IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
            UPDATE PAPEL1.EMPLEADO17
            SET SALARIO = SALARIO_EMP
            WHERE   COD_EMPLEADO = EID;

        ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
            UPDATE PAPEL2.EMPLEADO36
            SET SALARIO = SALARIO_EMP
            WHERE   COD_EMPLEADO = EID;

        ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
            UPDATE PAPEL3.EMPLEADO25
            SET SALARIO = SALARIO_EMP
            WHERE   COD_EMPLEADO = EID;

        ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
            UPDATE PAPEL4.EMPLEADO48
            SET SALARIO = SALARIO_EMP
            WHERE   COD_EMPLEADO = EID;
        ELSE
            RAISE_APPLICATION_ERROR(-20012,'LOCALIDAD NO VÁLIDA');
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20013,'Empleado no existe.');
    END IF;
END;
```

8.7. DELETE_EMPLEADO(2)

Este procedimiento recibe como argumentos COD_EMPLEADO y FECHA_FIN, realiza un DELETE de los datos del empleado y un INSERT en el histórico registrando la baja.

```
CREATE OR REPLACE PROCEDURE DELETE_EMPLEADO( COD_EMPLEADO NUMBER,
                                             FECHA_FIN DATE) IS
EMPCOUNT NUMBER;
HOT NUMBER;
LOCALIDAD VARCHAR(20);
FECHA_I DATE;

BEGIN
    SELECT COUNT (*)
    INTO EMPCOUNT
    FROM EMPLEADO
    WHERE COD_EMPLEADO = EID;
```

```
    IF(EMPCOUNT = 0) THEN
        RAISE_APPLICATION_ERROR(-20014,'Este codigo de empleado no se
encuentra registrado.');
    END IF;

    SELECT HID
    INTO HOT
    FROM EMPLEADO
    WHERE COD_EMPLEADO = EID;

    SELECT FECHA_INI
    INTO FECHA_I
    FROM PAPEL1.EMPLEADO17
    WHERE COD_EMPLEADO = EID;

    SELECT PROVINCIA
    INTO LOCALIDAD
    FROM HOTEL
    WHERE HOT = HID;

    IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
        INSERT INTO PAPEL3.HISTORICO
        VALUES(COD_EMPLEADO, HOT, FECHA_I, FECHA_FIN);

        DELETE FROM PAPEL1.EMPLEADO17 WHERE COD_EMPLEADO = EID;

    ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
        INSERT INTO PAPEL3.HISTORICO
        VALUES(COD_EMPLEADO, HOT, FECHA_I, FECHA_FIN);

        DELETE FROM PAPEL2.EMPLEADO36 WHERE COD_EMPLEADO = EID;

    ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
        INSERT INTO PAPEL3.HISTORICO
        VALUES(COD_EMPLEADO, HOT, FECHA_I, FECHA_FIN);

        DELETE FROM PAPEL3.EMPLEADO25 WHERE COD_EMPLEADO = EID;

    ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
        INSERT INTO PAPEL3.HISTORICO
        VALUES(COD_EMPLEADO, HOT, FECHA_I, FECHA_FIN);

        DELETE FROM PAPEL4.EMPLEADO48 WHERE COD_EMPLEADO = EID;
    ELSE
        RAISE_APPLICATION_ERROR(-20015,'LOCALIDAD NO VÁLIDA');
    END IF;
END;
```

8.8. INSERT_UPDATE_ENTREGA(12)

Este procedimiento recibe como argumentos COD_PROVEEDOR, COD_ARTICULO, COD_HOTEL, FECHA_EN, PRECIO_EN, CANTIDAD_EN. Si ya se encuentra registrada alguna entrega con los mismos COD_PROVEEDOR, COD_ARTICULO, COD_HOTEL, FECHA_EN, el procedimiento hace un UPDATE de dicha entrega, si no un INSERT de dicha entrega en la tabla ENTREGA de la localidad correspondiente, la cual la obtenemos a partir de COD_HOTEL, ya que deben estar hotel y entrega en la misma localidad.

```
CREATE OR REPLACE PROCEDURE INSERT_UPDATE_ENTREGA(
                                                    COD_PROVEEDOR NUMBER,
                                                    COD_ARTICULO NUMBER,
                                                    COD_HOTEL NUMBER,
                                                    FECHA_EN DATE,
                                                    PRECIO_EN NUMBER,
                                                    CANTIDAD_EN NUMBER) IS

UPDATECOUNT NUMBER;
LOCALIDAD VARCHAR(20);
PRECIO_AUX NUMBER;
BEGIN
    SELECT COUNT (*)
    INTO UPDATECOUNT
    FROM ENTREGA
    WHERE COD_HOTEL = HID
          AND COD_ARTICULO = AID
          AND COD_PROVEEDOR = PID
          AND FECHA_EN = FECHA;

    SELECT PROVINCIA
    INTO LOCALIDAD
    FROM HOTEL
    WHERE COD_HOTEL = HID;

    IF(UPDATECOUNT = 0) THEN
        IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
            INSERT INTO PAPEL1.ENTREGA17
            VALUES (COD_PROVEEDOR, COD_ARTICULO, COD_HOTEL, FECHA_EN ,
PRECIO_EN, CANTIDAD_EN);

        ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
            INSERT INTO PAPEL2.ENTREGA36
            VALUES (COD_PROVEEDOR, COD_ARTICULO, COD_HOTEL, FECHA_EN ,
PRECIO_EN, CANTIDAD_EN);

        ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
            INSERT INTO PAPEL3.ENTREGA25
            VALUES (COD_PROVEEDOR, COD_ARTICULO, COD_HOTEL, FECHA_EN ,
PRECIO_EN, CANTIDAD_EN);
```

```
        ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
            INSERT INTO PAPEL4.ENTREGA48
            VALUES (COD_PROVEEDOR, COD_ARTICULO, COD_HOTEL, FECHA_EN ,
PRECIO_EN, CANTIDAD_EN);

        ELSE
            RAISE_APPLICATION_ERROR(-20016,'LOCALIDAD NO VÁLIDA');
        END IF;
    ELSE
        SELECT PRECIO
        INTO PRECIO_AUX
        FROM ENTREGA
        WHERE   COD_HOTEL = HID
                AND COD_ARTICULO = AID
                AND COD_PROVEEDOR = PID
        ORDER BY FECHA ASC;

        IF(PRECIO_AUX <= PRECIO_EN) THEN
            IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
                UPDATE PAPEL1.ENTREGA17
                SET PRECIO = PRECIO_EN,
                    CANTIDAD = (CANTIDAD+CANTIDAD_EN)
                WHERE   COD_HOTEL = HID
                        AND COD_ARTICULO = AID
                        AND COD_PROVEEDOR = PID
                        AND FECHA_EN = FECHA;
            ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
                UPDATE PAPEL2.ENTREGA36
                SET PRECIO = PRECIO_EN,
                    CANTIDAD = (CANTIDAD+CANTIDAD_EN)
                WHERE   COD_HOTEL = HID
                        AND COD_ARTICULO = AID
                        AND COD_PROVEEDOR = PID
                        AND FECHA_EN = FECHA;

            ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
                UPDATE PAPEL3.ENTREGA25
                SET PRECIO = PRECIO_EN,
                    CANTIDAD = (CANTIDAD+CANTIDAD_EN)
                WHERE   COD_HOTEL = HID
                        AND COD_ARTICULO = AID
                        AND COD_PROVEEDOR = PID
                        AND FECHA_EN = FECHA;

            ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
                UPDATE PAPEL4.ENTREGA48
                SET PRECIO = PRECIO_EN,
                    CANTIDAD = (CANTIDAD+CANTIDAD_EN)
                WHERE   COD_HOTEL = HID
                        AND COD_ARTICULO = AID
                        AND COD_PROVEEDOR = PID
```

```
                        AND FECHA_EN = FECHA;
            ELSE
                RAISE_APPLICATION_ERROR(-20017,'LOCALIDAD NO VÁLIDA');
            END IF;
        ELSE
            RAISE_APPLICATION_ERROR(-20018,'El precio no puede
reducirse.');
        END IF;
    END IF;
END;
```

8.9. DELETE_ENTREGA(13)

Este procedimienton recibe como argumentos COD_ARTICULO, COD_HOTEL y FECHA_SUM y realiza un DELETE de dicha entrega de la tabla ENTREGA de la localidad correspondiente, la cual la obtenemos a partir de COD_HOTEL, ya que deben estar hotel y entrega en la misma localidad..

```
CREATE OR REPLACE PROCEDURE DELETE_ENTREGA( COD_ARTICULO NUMBER,
                                            COD_HOTEL NUMBER,
                                            FECHA_SUM DATE) IS

ENTCOUNT NUMBER;
LOCALIDAD VARCHAR(20);

BEGIN
      SELECT COUNT (*)
      INTO ENTCOUNT
      FROM ENTREGA
      WHERE COD_HOTEL = HID AND COD_ARTICULO = AID;
      IF(ENTCOUNT = 0) THEN
            RAISE_APPLICATION_ERROR(-20019,'Esta entrega no se encuentra
registrada.');
    END IF;
      SELECT PROVINCIA
      INTO LOCALIDAD
      FROM HOTEL
      WHERE COD_HOTEL = HID;

      IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
            IF (FECHA_SUM IS NULL) THEN
                  DELETE FROM PAPEL1.ENTREGA17 WHERE COD_HOTEL = HID AND
COD_ARTICULO = AID;
            ELSE
                  DELETE FROM PAPEL1.ENTREGA17 WHERE COD_HOTEL = HID AND
COD_ARTICULO = AID AND FECHA = FECHA_SUM;
            END IF;

    ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
```

```
            IF (FECHA_SUM IS NULL) THEN
                DELETE FROM PAPEL2.ENTREGA36 WHERE COD_HOTEL = HID AND
COD_ARTICULO = AID;
            ELSE
                DELETE FROM PAPEL2.ENTREGA36 WHERE COD_HOTEL = HID AND
COD_ARTICULO = AID AND FECHA = FECHA_SUM;
            END IF;

    ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
            IF (FECHA_SUM IS NULL) THEN
                DELETE FROM PAPEL3.ENTREGA25 WHERE COD_HOTEL = HID AND
COD_ARTICULO = AID;
            ELSE
                DELETE FROM PAPEL3.ENTREGA25 WHERE COD_HOTEL = HID AND
COD_ARTICULO = AID AND FECHA = FECHA_SUM;
            END IF;

    ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
            IF (FECHA_SUM IS NULL) THEN
                DELETE FROM PAPEL4.ENTREGA48 WHERE COD_HOTEL = HID AND
COD_ARTICULO = AID;
            ELSE
                DELETE FROM PAPEL4.ENTREGA48 WHERE COD_HOTEL = HID AND
COD_ARTICULO = AID AND FECHA = FECHA_SUM;
            END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20020,'LOCALIDAD NO VÁLIDA');
    END IF;
END;
```

8.10. INSERT_HOTEL(5)

Este procedimiento recibe como argumentos los atributos correspondientes a un hotel y realiza un insert en la tabla HOTEL de la localidad correspondiente siendo esta la especificada en PROVINCIA_HOTEL.

```
CREATE OR REPLACE PROCEDURE INSERT_HOTEL(    COD_HOTEL NUMBER,
                                             COD_DIRECTOR NUMBER,
                                             NOMBRE_HOTEL VARCHAR,
                                             CIUDAD_HOTEL VARCHAR,
                                             PROVINCIA_HOTEL VARCHAR,
                                             NU_HAB NUMBER,
                                             NU_HAB2 NUMBER) IS

HCOUNT NUMBER;

BEGIN
    IF (PROVINCIA_HOTEL = 'Granada' OR PROVINCIA_HOTEL = 'Jaén') THEN
        INSERT INTO PAPEL1.HOTEL17
```

```
        VALUES (COD_HOTEL, COD_DIRECTOR, NOMBRE_HOTEL, CIUDAD_HOTEL,
PROVINCIA_HOTEL, NU_HAB, NU_HAB2);

    ELSIF (PROVINCIA_HOTEL = 'Cádiz' OR PROVINCIA_HOTEL = 'Huelva')
THEN
        INSERT INTO PAPEL2.HOTEL36
        VALUES (COD_HOTEL, COD_DIRECTOR, NOMBRE_HOTEL, CIUDAD_HOTEL,
PROVINCIA_HOTEL, NU_HAB, NU_HAB2);

    ELSIF (PROVINCIA_HOTEL = 'Sevilla' OR PROVINCIA_HOTEL = 'Córdoba')
THEN
        INSERT INTO PAPEL3.HOTEL25
        VALUES (COD_HOTEL, COD_DIRECTOR, NOMBRE_HOTEL, CIUDAD_HOTEL,
PROVINCIA_HOTEL, NU_HAB, NU_HAB2);

    ELSIF (PROVINCIA_HOTEL = 'Málaga' OR PROVINCIA_HOTEL = 'Almería')
THEN
        INSERT INTO PAPEL4.HOTEL48
        VALUES (COD_HOTEL, COD_DIRECTOR, NOMBRE_HOTEL, CIUDAD_HOTEL,
PROVINCIA_HOTEL, NU_HAB, NU_HAB2);

    ELSE
        RAISE_APPLICATION_ERROR(-20021,'LOCALIDAD NO VÁLIDA');
    END IF;
END;
```

8.11. UPDATE_DIRECTOR(6)

Este procedimiento recibe como argumento COD_HOTEL y COD_NUEVODIR, comprueba que el hotel de código COD_HOTEL existe y que el empleado de código COD_NUEVODIR existe y no es director de otro hotel, y después de las comprobaciones realiza un update sobre la tabla HOTEL de la localidad correspondiente al hotel de código COD_HOTEL cambiando el director.

```
CREATE OR REPLACE PROCEDURE UPDATE_DIRECTOR(    COD_HOTEL NUMBER,
                                                COD_NUEVODIR NUMBER) IS

HIDCOUNT NUMBER;
DCOUNT NUMBER;
LOCALIDAD VARCHAR(20);

BEGIN
      SELECT COUNT (*)
      INTO HIDCOUNT
    FROM HOTEL
    WHERE COD_HOTEL = HID;

    SELECT COUNT (*)
    INTO DCOUNT
    FROM HOTEL
```

```
    WHERE EID = COD_NUEVODIR;

    IF(HIDCOUNT > 0) THEN
        IF(DCOUNT = 0) THEN
            SELECT PROVINCIA
            INTO LOCALIDAD
            FROM HOTEL
            WHERE COD_HOTEL = HID;

            IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
                UPDATE PAPEL1.HOTEL17
                    SET EID = COD_NUEVODIR
                    WHERE COD_HOTEL = HID;

            ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
                UPDATE PAPEL2.HOTEL36
                    SET EID = COD_NUEVODIR
                    WHERE COD_HOTEL = HID;

            ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
                UPDATE PAPEL3.HOTEL25
                    SET EID = COD_NUEVODIR
                    WHERE COD_HOTEL = HID;

            ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
                UPDATE PAPEL4.HOTEL48
                    SET EID = COD_NUEVODIR
                    WHERE COD_HOTEL = HID;
            ELSE
                RAISE_APPLICATION_ERROR(-20022,'Localidad no válida.');
            END IF;
        ELSE
             RAISE_APPLICATION_ERROR(-20023,'Este empleado ya dirige
un hotel.');
        END IF;
    ELSE
         RAISE_APPLICATION_ERROR(-20024,'Hotel no existe.');
    END IF;
END;
```

8.12. INSERT PROVEEDOR(10)

Este procedimiento recibe como argumentos los atributos correspondientes a un hotel y realiza un insert en la tabla PROVEEDOR de la localidad correspondiente siendo esta la especificada en LOCALIDAD.

```
CREATE OR REPLACE PROCEDURE INSERT_PROVEEDOR(   COD_PROVEEDOR NUMBER,
                                                NOMBRE_PRO VARCHAR,
                                                LOCALIDAD VARCHAR) IS
```

```
PCOUNT NUMBER;

BEGIN
    IF (LOCALIDAD = 'Granada') THEN
        INSERT INTO PAPEL1.PROVEEDOR1
        VALUES (COD_PROVEEDOR, NOMBRE_PRO , LOCALIDAD);

        INSERT INTO PAPEL4.PROVEEDOR1
        VALUES (COD_PROVEEDOR, NOMBRE_PRO , LOCALIDAD);

    ELSIF (LOCALIDAD = 'Sevilla') THEN
        INSERT INTO PAPEL3.PROVEEDOR2
        VALUES (COD_PROVEEDOR, NOMBRE_PRO , LOCALIDAD);

        INSERT INTO PAPEL2.PROVEEDOR2
        VALUES (COD_PROVEEDOR, NOMBRE_PRO , LOCALIDAD);
    ELSE
        RAISE_APPLICATION_ERROR(-20025,'LOCALIDAD NO VÁLIDA');
    END IF;
END;
```

8.13. DELETE_PROVEEDOR(11)

Este procedimiento recibe como argumento COD_PRO y realiza un DELETE en la tabla PROVEEDOR de la localidad correspondiente siendo esta la cual la obtenemos realizando un SELECT sobre la vista PROVEEDOR.

```
CREATE OR REPLACE PROCEDURE DELETE_PROVEEDOR(COD_PRO NUMBER) IS

PROCOUNT NUMBER;
LOCALIDAD VARCHAR(20);

BEGIN
      SELECT COUNT (*)
      INTO PROCOUNT
      FROM PROVEEDOR
      WHERE COD_PRO = PID;

    IF(PROCOUNT = 0) THEN
        RAISE_APPLICATION_ERROR(-20026,'Este codigo de proveedor no se
encuentra registrado.');
    END IF;

      SELECT PROVINCIA
      INTO LOCALIDAD
    FROM PROVEEDOR
    WHERE COD_PRO = PID;

      IF(LOCALIDAD = 'Granada') THEN
            DELETE FROM PAPEL1.PROVEEDOR1 WHERE COD_PRO = PID;
```

```
            DELETE FROM PAPEL4.PROVEEDOR1 WHERE COD_PRO = PID;

      ELSIF (LOCALIDAD = 'Sevilla') THEN
            DELETE FROM PAPEL2.PROVEEDOR2 WHERE COD_PRO = PID;
            DELETE FROM PAPEL3.PROVEEDOR2 WHERE COD_PRO = PID;
    ELSE
        RAISE_APPLICATION_ERROR(-20027,'LOCALIDAD NO VÁLIDA');
    END IF;
END;
```

8.14. INSERT_UPDATE_RESERVA(8)

Este procedimiento si recibe como argumentos los atributos correspondientes a una reserva, NUEVO_INICIO Y NUEVO_FINAL. Si NUEVO_INICIO y NUEVO_FINAL son nulos el procedimiento realiza un INSERT en la tabla RESERVA de la localidad correspondiente, la cual la obtenemos a través del COD_HOTEL, ya que deben estar en la misma localidad. Si no son nulos se hace un UPDATE de la reserva en la tabla RESERVA de la localidad correspondiente, la cual obtenemos de la forma anterior.

```
CREATE OR REPLACE PROCEDURE INSERT_UPDATE_RESERVA( COD_CLIENTE NUMBER,
                                                   COD_HOTEL NUMBER,
                                                   FECHA_INICIO DATE,
                                                   FECHA_FINAL DATE,
                                                    NUEVO_INICIO DATE,
                                                   NUEVO_FINAL DATE,
                                                   TIPO_HABITACION VARCHAR,
                                                   PRECIO_RESERVA FLOAT) IS

UPDATECOUNT NUMBER;
RESERVAS NUMBER;
NHABITACIONES NUMBER;
RESERVASHOTEL NUMBER;
LOCALIDAD VARCHAR(20);
HABITACION_ANTERIOR VARCHAR(20);

BEGIN
    IF(NUEVO_INICIO IS NULL OR NUEVO_FINAL IS NULL) THEN
        SELECT PROVINCIA
        INTO LOCALIDAD
        FROM HOTEL
        WHERE HID = COD_HOTEL;

        IF LOCALIDAD='Granada' OR LOCALIDAD='Jaén' THEN
            INSERT INTO PAPEL1.RESERVA17
            VALUES (COD_CLIENTE, COD_HOTEL, FECHA_FINAL, FECHA_INICIO,
TIPO_HABITACION, PRECIO_RESERVA);

        ELSIF (LOCALIDAD='Cádiz' OR LOCALIDAD='Huelva') THEN
            INSERT INTO PAPEL2.RESERVA36
```

```
            VALUES (COD_CLIENTE, COD_HOTEL, FECHA_FINAL, FECHA_INICIO,
TIPO_HABITACION, PRECIO_RESERVA);

        ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
            INSERT INTO PAPEL3.RESERVA25
            VALUES (COD_CLIENTE, COD_HOTEL, FECHA_FINAL, FECHA_INICIO,
TIPO_HABITACION, PRECIO_RESERVA);

        ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
            INSERT INTO PAPEL4.RESERVA48
            VALUES (COD_CLIENTE, COD_HOTEL, FECHA_FINAL, FECHA_INICIO,
TIPO_HABITACION, PRECIO_RESERVA);

        ELSE
            RAISE_APPLICATION_ERROR(-20028,'LOCALIDAD NO VÁLIDA');
        END IF;
    ELSIF (NUEVO_INICIO IS NOT NULL AND NUEVO_FINAL IS NOT NULL) THEN
        SELECT COUNT (*)
        INTO UPDATECOUNT
        FROM RESERVA
        WHERE   CID = COD_CLIENTE
                AND COD_HOTEL = HID
                AND FECHA_INICIO = FECHA_INI
                AND FECHA_FINAL = FECHA_FIN;

        IF UPDATECOUNT > 0 THEN
            SELECT COUNT(*)
            INTO RESERVAS
            FROM RESERVA
            WHERE   (FECHA_INI<=NUEVO_INICIO OR FECHA_INI<=NUEVO_FINAL)
                    AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL);

            SELECT TIPO_HAB
            INTO HABITACION_ANTERIOR
            FROM RESERVA
            WHERE   CID = COD_CLIENTE
                    AND COD_HOTEL = HID
                    AND FECHA_INI = FECHA_INICIO
                    AND FECHA_FIN = FECHA_FINAL;

            IF (TIPO_HABITACION = 'Sencilla') THEN
                SELECT N_HAB
                INTO NHABITACIONES
                FROM HOTEL
                WHERE HID = COD_HOTEL;

            ELSIF (TIPO_HABITACION = 'Doble') THEN
                SELECT N_HAB2
                INTO NHABITACIONES
                FROM HOTEL
```

```
                WHERE HID = COD_HOTEL;
            END IF;

            IF(RESERVAS <= 1) THEN
                SELECT PROVINCIA
                INTO LOCALIDAD
                FROM HOTEL
                WHERE HID = COD_HOTEL;

                IF LOCALIDAD='Granada' OR LOCALIDAD='Jaén' THEN
                    IF(HABITACION_ANTERIOR != TIPO_HABITACION) THEN
                        IF (TIPO_HABITACION = 'Sencilla') THEN
                            SELECT COUNT(*)
                            INTO RESERVASHOTEL
                            FROM PAPEL1.RESERVA17
                            WHERE   (FECHA_INI<=NUEVO_INICIO OR
FECHA_INI<=NUEVO_FINAL)
                                    AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL)
                                    AND TIPO_HAB = 'Sencilla';

                            IF(RESERVASHOTEL < NHABITACIONES) THEN
                                UPDATE PAPEL1.RESERVA17
                                SET FECHA_FIN = NUEVO_FINAL,
                                    FECHA_INI = NUEVO_INICIO,
                                    TIPO_HAB = TIPO_HABITACION,
                                    PRECIO = PRECIO_RESERVA
                                WHERE   COD_CLIENTE = CID
                                        AND COD_HOTEL = HID;
                            ELSE
                                RAISE_APPLICATION_ERROR(-20029,'No
queda espacio en el hotel en esas fechas.');
                            END IF;
                        ELSIF (TIPO_HABITACION = 'Doble') THEN
                            SELECT COUNT(*)
                            INTO RESERVASHOTEL
                            FROM PAPEL1.RESERVA17
                            WHERE   (FECHA_INI<=NUEVO_INICIO OR
FECHA_INI<=NUEVO_FINAL)
                                    AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL)
                                    AND TIPO_HAB = 'Doble';

                            IF(RESERVASHOTEL < NHABITACIONES) THEN
                                UPDATE PAPEL1.RESERVA17
                                SET FECHA_FIN = NUEVO_FINAL,
                                    FECHA_INI = NUEVO_INICIO,
                                    TIPO_HAB = TIPO_HABITACION,
                                    PRECIO = PRECIO_RESERVA
                                WHERE   COD_CLIENTE = CID
                                        AND COD_HOTEL = HID;
```

```
                                ELSE
                                    RAISE_APPLICATION_ERROR(-20030,'No
queda espacio en el hotel en esas fechas.');
                                END IF;
                            END IF;
                        ELSE
                            UPDATE PAPEL1.RESERVA17
                            SET FECHA_FIN = NUEVO_FINAL,
                                FECHA_INI = NUEVO_INICIO,
                                TIPO_HAB = TIPO_HABITACION,
                                PRECIO = PRECIO_RESERVA
                            WHERE   COD_CLIENTE = CID
                                    AND COD_HOTEL = HID;
                        END IF;

                    ELSIF (LOCALIDAD='Cádiz' OR LOCALIDAD='Huelva') THEN
                        IF(HABITACION_ANTERIOR != TIPO_HABITACION) THEN
                            IF (TIPO_HABITACION = 'Sencilla') THEN
                                SELECT COUNT(*)
                                INTO RESERVASHOTEL
                                FROM PAPEL2.RESERVA36
                                WHERE   (FECHA_INI<=NUEVO_INICIO OR
FECHA_INI<=NUEVO_FINAL)
                                        AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL)
                                        AND TIPO_HAB = 'Sencilla';

                                IF(RESERVASHOTEL < NHABITACIONES) THEN
                                    UPDATE PAPEL2.RESERVA36
                                    SET FECHA_FIN = NUEVO_FINAL,
                                        FECHA_INI = NUEVO_INICIO,
                                        TIPO_HAB = TIPO_HABITACION,
                                        PRECIO = PRECIO_RESERVA
                                    WHERE   COD_CLIENTE = CID
                                            AND COD_HOTEL = HID;
                                ELSE
                                    RAISE_APPLICATION_ERROR(-20031,'No
queda espacio en el hotel en esas fechas.');
                                END IF;
                            ELSIF (TIPO_HABITACION = 'Doble') THEN
                                SELECT COUNT(*)
                                INTO RESERVASHOTEL
                                FROM PAPEL2.RESERVA36
                                WHERE   (FECHA_INI<=NUEVO_INICIO OR
FECHA_INI<=NUEVO_FINAL)
                                        AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL)
                                        AND TIPO_HAB = 'Doble';

                                IF(RESERVASHOTEL < NHABITACIONES) THEN
                                    UPDATE PAPEL2.RESERVA36
```

```
                                    SET FECHA_FIN = NUEVO_FINAL,
                                        FECHA_INI = NUEVO_INICIO,
                                        TIPO_HAB = TIPO_HABITACION,
                                        PRECIO = PRECIO_RESERVA
                                    WHERE   COD_CLIENTE = CID
                                            AND COD_HOTEL = HID;
                                ELSE
                                    RAISE_APPLICATION_ERROR(-20032,'No
queda espacio en el hotel en esas fechas.');
                                END IF;
                            END IF;
                        ELSE
                            UPDATE PAPEL2.RESERVA36
                            SET FECHA_FIN = NUEVO_FINAL,
                                FECHA_INI = NUEVO_INICIO,
                                TIPO_HAB = TIPO_HABITACION,
                                PRECIO = PRECIO_RESERVA
                            WHERE   COD_CLIENTE = CID
                                    AND COD_HOTEL = HID;
                        END IF;

                    ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba')
THEN
                        IF(HABITACION_ANTERIOR != TIPO_HABITACION) THEN
                            IF (TIPO_HABITACION = 'Sencilla') THEN
                                SELECT COUNT(*)
                                INTO RESERVASHOTEL
                                FROM PAPEL3.RESERVA25
                                WHERE   (FECHA_INI<=NUEVO_INICIO OR
FECHA_INI<=NUEVO_FINAL)
                                        AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL)
                                        AND TIPO_HAB = 'Sencilla';

                                IF(RESERVASHOTEL < NHABITACIONES) THEN
                                    UPDATE PAPEL3.RESERVA25
                                    SET FECHA_FIN = NUEVO_FINAL,
                                        FECHA_INI = NUEVO_INICIO,
                                        TIPO_HAB = TIPO_HABITACION,
                                        PRECIO = PRECIO_RESERVA
                                    WHERE   COD_CLIENTE = CID
                                            AND COD_HOTEL = HID;
                                ELSE
                                    RAISE_APPLICATION_ERROR(-20033,'No
queda espacio en el hotel en esas fechas.');
                                END IF;
                            ELSIF (TIPO_HABITACION = 'Doble') THEN
                                SELECT COUNT(*)
                                INTO RESERVASHOTEL
                                FROM PAPEL3.RESERVA25
```

```
                                WHERE   (FECHA_INI<=NUEVO_INICIO OR
FECHA_INI<=NUEVO_FINAL)
                                    AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL)
                                    AND TIPO_HAB = 'Doble';

                                IF(RESERVASHOTEL < NHABITACIONES) THEN
                                    UPDATE PAPEL3.RESERVA25
                                    SET FECHA_FIN = NUEVO_FINAL,
                                        FECHA_INI = NUEVO_INICIO,
                                        TIPO_HAB = TIPO_HABITACION,
                                        PRECIO = PRECIO_RESERVA
                                    WHERE   COD_CLIENTE = CID
                                            AND COD_HOTEL = HID;
                                ELSE
                                    RAISE_APPLICATION_ERROR(-20034,'No
queda espacio en el hotel en esas fechas.');
                                END IF;
                            END IF;
                        ELSE
                            UPDATE PAPEL3.RESERVA25
                            SET FECHA_FIN = NUEVO_FINAL,
                                FECHA_INI = NUEVO_INICIO,
                                TIPO_HAB = TIPO_HABITACION,
                                PRECIO = PRECIO_RESERVA
                            WHERE   COD_CLIENTE = CID
                                    AND COD_HOTEL = HID;
                        END IF;

                    ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería')
THEN
                        IF(HABITACION_ANTERIOR != TIPO_HABITACION) THEN
                            IF (TIPO_HABITACION = 'Sencilla') THEN
                                SELECT COUNT(*)
                                INTO RESERVASHOTEL
                                FROM PAPEL4.RESERVA48
                                WHERE   (FECHA_INI<=NUEVO_INICIO OR
FECHA_INI<=NUEVO_FINAL)
                                    AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL)
                                    AND TIPO_HAB = 'Sencilla';

                                IF(RESERVASHOTEL < NHABITACIONES) THEN
                                    UPDATE PAPEL4.RESERVA48
                                    SET FECHA_FIN = NUEVO_FINAL,
                                        FECHA_INI = NUEVO_INICIO,
                                        TIPO_HAB = TIPO_HABITACION,
                                        PRECIO = PRECIO_RESERVA
                                    WHERE   COD_CLIENTE = CID
                                            AND COD_HOTEL = HID;
                                ELSE
```

```
                                    RAISE_APPLICATION_ERROR(-20035,'No
queda espacio en el hotel en esas fechas.');
                                END IF;
                            ELSIF (TIPO_HABITACION = 'Doble') THEN
                                SELECT COUNT(*)
                                INTO RESERVASHOTEL
                                FROM PAPEL4.RESERVA48
                                WHERE   (FECHA_INI<=NUEVO_INICIO OR
FECHA_INI<=NUEVO_FINAL)
                                        AND (FECHA_FIN>=NUEVO_INICIO OR
FECHA_FIN>=NUEVO_FINAL)
                                        AND TIPO_HAB = 'Doble';

                                IF(RESERVASHOTEL < NHABITACIONES) THEN
                                    UPDATE PAPEL4.RESERVA48
                                    SET FECHA_FIN = NUEVO_FINAL,
                                        FECHA_INI = NUEVO_INICIO,
                                        TIPO_HAB = TIPO_HABITACION,
                                        PRECIO = PRECIO_RESERVA
                                    WHERE   COD_CLIENTE = CID
                                            AND COD_HOTEL = HID;
                                ELSE
                                    RAISE_APPLICATION_ERROR(-20036,'No
queda espacio en el hotel en esas fechas.');
                                END IF;
                            END IF;
                        ELSE
                            UPDATE PAPEL4.RESERVA48
                            SET FECHA_FIN = NUEVO_FINAL,
                                FECHA_INI = NUEVO_INICIO,
                                TIPO_HAB = TIPO_HABITACION,
                                PRECIO = PRECIO_RESERVA
                            WHERE   COD_CLIENTE = CID
                                    AND COD_HOTEL = HID;
                        END IF;
                    ELSE
                        RAISE_APPLICATION_ERROR(-20037,'LOCALIDAD NO
VÁLIDA');
                    END IF;
                ELSE
                    RAISE_APPLICATION_ERROR(-20038,'El cliente ya tiene una
reserva en esas fechas.');
                END IF;
            ELSE
                RAISE_APPLICATION_ERROR(-20039,'No existe ninguna reserva
en esas fechas.');
            END IF;
        END IF;
END;
```

8.15. DELETE_RESERVA(9)

Este procedimiento recibe como argumentos COD_CLIENTE, COD_HOTEL, FECHA_ENTRADA y FECHA_SALIDA. Si FECHA_ENTRADA y FECHA_SALIDA son nulos se realiza un DELETE de todas las reservas que tengan COD_HOTEL y COD_CLIENTE, de la tabla RESERVA de la localidad correspondiente, la cual la obtenemos a través del COD_HOTEL, ya que deben estar en la misma localidad. Si no son nulos solo se hace un DELETE de todas la reserva que tenga COD_CLIENTE, COD_HOTEL, FECHA_ENTRADA y FECHA_SALIDA de la tabla RESERVA de la localidad correspondiente, la cual obtenemos de la forma anterior.

```
CREATE OR REPLACE PROCEDURE DELETE_RESERVA( COD_CLIENTE NUMBER,
                                            COD_HOTEL NUMBER,
                                            FECHA_ENTRADA DATE,
                                            FECHA_SALIDA DATE) IS

RESCOUNT NUMBER;
LOCALIDAD VARCHAR(20);

BEGIN
      SELECT COUNT (*)
      INTO RESCOUNT
    FROM RESERVA
    WHERE COD_CLIENTE = CID AND COD_HOTEL = HID AND FECHA_ENTRADA =
FECHA_INI AND FECHA_SALIDA = FECHA_FIN;

    IF(RESCOUNT = 0) THEN
        RAISE_APPLICATION_ERROR(-20040,'Esta reserva no se encuentra
registrada.');
    END IF;

      SELECT PROVINCIA
      INTO LOCALIDAD
    FROM HOTEL
    WHERE COD_HOTEL = HID;

      IF (LOCALIDAD = 'Granada' OR LOCALIDAD = 'Jaén') THEN
            DELETE FROM PAPEL1.RESERVA17 WHERE COD_CLIENTE = CID AND
COD_HOTEL = HID AND FECHA_ENTRADA = FECHA_INI AND FECHA_SALIDA =
FECHA_FIN;

      ELSIF (LOCALIDAD = 'Cádiz' OR LOCALIDAD = 'Huelva') THEN
            DELETE FROM PAPEL2.RESERVA36 WHERE COD_CLIENTE = CID AND
COD_HOTEL = HID AND FECHA_ENTRADA = FECHA_INI AND FECHA_SALIDA =
FECHA_FIN;

      ELSIF (LOCALIDAD = 'Sevilla' OR LOCALIDAD = 'Córdoba') THEN
            DELETE FROM PAPEL3.RESERVA25 WHERE COD_CLIENTE = CID AND
COD_HOTEL = HID AND FECHA_ENTRADA = FECHA_INI AND FECHA_SALIDA =
FECHA_FIN;
```

```
      ELSIF (LOCALIDAD = 'Málaga' OR LOCALIDAD = 'Almería') THEN
            DELETE FROM PAPEL4.RESERVA48 WHERE COD_CLIENTE = CID AND
COD_HOTEL = HID AND FECHA_ENTRADA = FECHA_INI AND FECHA_SALIDA =
FECHA_FIN;
      ELSE
        RAISE_APPLICATION_ERROR(-20041,'LOCALIDAD NO VÁLIDA');
    END IF;
END;
```

9. Consultas.

9.1. Primera consulta

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 2021 y el 28 de mayo de 2021.

```
SELECT  HOTEL.NOMBRE AS NOMBRE_HOTEL,
        HOTEL.CIUDAD AS CIUDAD_HOTEL,
        ARTICULO.NOMBRE AS ARTICULO
FROM    HOTEL,
        PROVEEDOR,
        ARTICULO,
        ENTREGA
WHERE   (HOTEL.PROVINCIA = 'Granada' OR HOTEL.PROVINCIA = 'Huelva' OR
HOTEL.PROVINCIA = 'Almería')
        AND ENTREGA.HID = HOTEL.HID
        AND ENTREGA.PID = PROVEEDOR.PID
        AND ENTREGA.AID = ARTICULO.AID
        AND ARTICULO.PID = PROVEEDOR.PID
        AND (ARTICULO.NOMBRE = 'Queso' OR ARTICULO.NOMBRE =
'Mantequilla')
        AND ENTREGA.FECHA >= TO_DATE('12052021','DDMMYYYY')
        AND ENTREGA.FECHA <= TO_DATE('28052021','DDMMYYYY');
```

9.2. Segunda consulta

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.

```
ACCEPT COD_PROVEEDOR CHAR PROMPT 'Introduzca el PID deseado: ';
SELECT  ARTICULO.NOMBRE AS NOMBRE_ART,
        HOTEL.NOMBRE AS NOMBRE_HOTEL,
        HOTEL.CIUDAD AS CIUDAD_HOTEL,
        SUM(ENTREGA.CANTIDAD) AS CANTIDAD
FROM    HOTEL,
        PROVEEDOR,
        ARTICULO,
        ENTREGA
WHERE   PROVEEDOR.PID = &COD_PROVEEDOR
        AND (HOTEL.PROVINCIA = 'Jaén' OR HOTEL.PROVINCIA = 'Almería')
        AND ENTREGA.HID = HOTEL.HID
        AND ENTREGA.PID = PROVEEDOR.PID
        AND ENTREGA.AID = ARTICULO.AID
        AND ARTICULO.PID = PROVEEDOR.PID
GROUP BY ARTICULO.NOMBRE,HOTEL.NOMBRE,HOTEL.CIUDAD;
```

9.3. Tercera consulta

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

```
ACCEPT COD_HOTEL CHAR PROMPT 'Introduzca el HID deseado: ';
SELECT  CLIENTE.NOMBRE AS NOMBRE_CLIENTE,
        CLIENTE.TLF AS TLF_CLIENTE
FROM    RESERVA,
        CLIENTE
WHERE   RESERVA.CID = CLIENTE.CID
        AND RESERVA.HID = &COD_HOTEL
GROUP BY    CLIENTE.NOMBRE,
            CLIENTE.TLF
HAVING  COUNT(RESERVA.HID) > 1;
```