



UNIVERSIDAD DE GRANADA

Base de datos distribuida para cadena de hoteles multinacional andaluza

4º Sistemas de Información
Grado en Ingeniería Informática
Bases de datos distribuidas

Autores :

**Espejo Muñoz, Gabriel
García Romero, Carmen**

Febrero 2022

Práctica 1: Diseño conceptual y lógico global de la base de datos	3
1.1 Descripción del problema	3
1.2 Diagrama de Entidad-Relación	4
1.3 Descripción de las entidades y relaciones	5
1.4 Diseño del modelo lógico	6
Práctica 2: Diseño de la fragmentación y de la asignación	8
2.1 Hoteles	8
2.2 Proveedores	8
2.2 Suministro	9
2.4 Empleados	11
2.5 Replicación de datos	11
2.6 Localidades asociadas a la base de datos	11
Práctica 3: Implementación del diseño	11
3.1 Creación de tablas	12
3.2 Garantización de privilegios	15
3.3 Creación de vistas	16
3.4 Disparadores	26
Práctica 4: Implementación de las actualizaciones	31
4.1 Procedimientos	31
Práctica 5: Consultas	40
Consulta 1	40
Consulta 2	41
Consulta 3	42

Práctica 1:

Diseño conceptual y lógico global de la base de datos

1.1 Descripción 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.

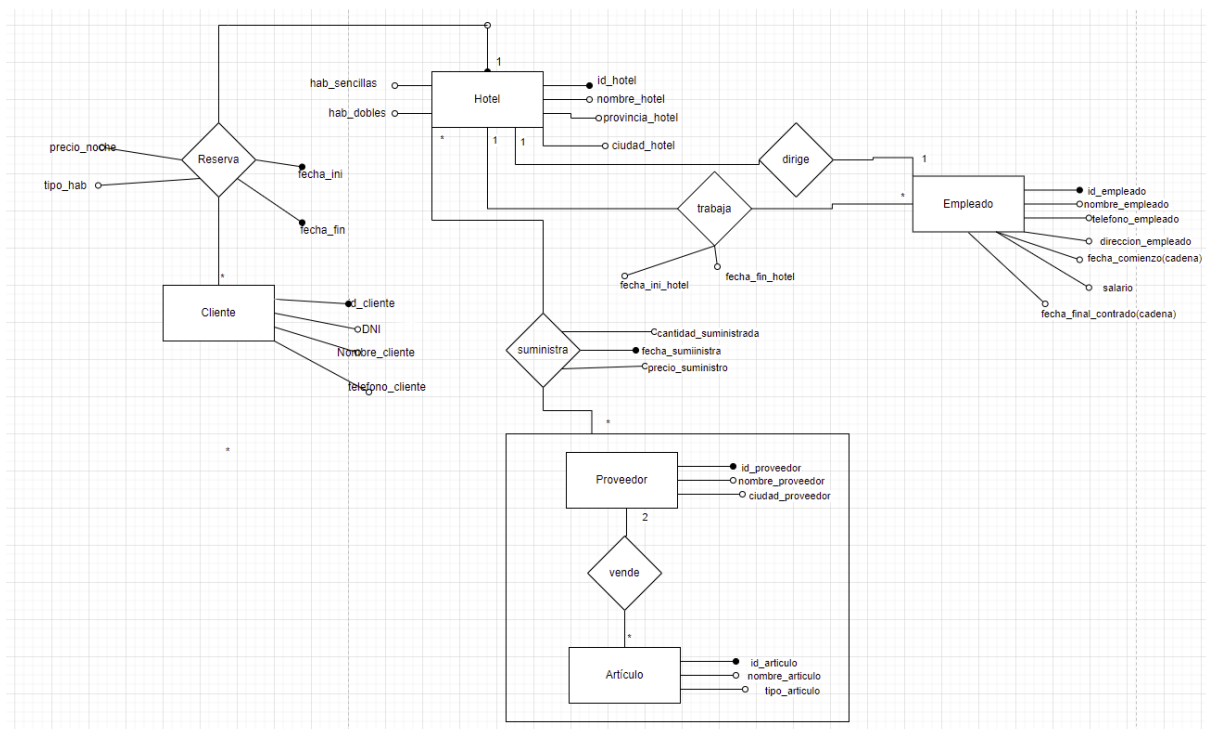
LOS EMPLEADOS de la cadena se identifican mediante un código de empleado, que mantendrán mientras trabajen en la multinacional independientemente del hotel en el que estén asignados en un momento determinado. La administración de la cadena almacena para cada empleado el DNI, el nombre, el número de teléfono, 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 del 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.

1.2 Diagrama de Entidad-Relación



Tal y como se puede entender de la descripción del problema, tenemos 5 entidades: Cliente, Hotel, Proveedor, Artículo y Empleado. Cada una de las entidades están compuestas por varios atributos, entre ellos una clave primaria. La dirección de un hotel la hemos representado con la relación dirige entre empleado y hotel, de tal manera que se pueda asociar fácilmente el código del director con el del hotel. Una posible alternativa sería crear una entidad Director que hereda sus atributos de Empleado, ya que un director es un tipo de

empleado que dirige un hotel. Entendemos que, al no tener el director más atributos que el empleado, es más eficiente hacerlo en forma de relación.

La relación entre cliente y hotel se llama Reserva tiene 4 atributos, entre los que destaca una clave primaria compuesta por fecha_ini y fecha_fin. Esta clave se pone para que no pueda haber una reserva con las mismas condiciones en un hotel por dos clientes diferentes. En un caso real, sería conveniente crear una entidad Habitación que se relacione con cliente por un lado y con hotel por otro para poder hacer esta restricción en una habitación en lugar de en el hotel entero.

La relación suministra es una relación entre hotel y la relación tiene (relación entre proveedores y artículos). A esto se le llama agregación, que se pone para evitar ciclos. Cabe destacar la clave fecha_suminstra, que, al igual que en reserva, se utiliza para que un artículo de un proveedor concreto en un hotel se pueda suministrar varias veces, aunque no se podría hacer en la misma fecha.

Por último, la relación trabaja conecta los empleados con un hotel, añadiendo entre ellos la fecha de inicio y de finalización de contrato en el hotel del empleado.

1.3 Diseño del modelo lógico

Una vez que hemos diseñado nuestro modelo E/R, tenemos que obtener el modelo lógico de la base de datos, haciendo el paso a tablas para el modelo de datos relacional. A raíz de las entidades y las relaciones explicadas anteriormente obtendremos las siguientes tablas (quedan subrayadas las claves primarias de las tablas):

1.3.1 Descripción de las entidades y relaciones

Nuestro modelo E/R se compone de las siguientes entidades y relaciones:

Entidades

- **Hotel** : almacena los datos de los distintos hoteles de cada ciudad , pertenecientes a la cadena hotelera.
- **Cliente**: almacena los datos de los distintos clientes que realicen reservas en los distintos hoteles de la cadena hotelera.
- **Proveedor**: almacena los datos de los distintos proveedores que distribuyen los distintos artículos a los diferentes hoteles de la cadena.
- **Artículo**: almacena los datos de los distintos artículos que son distribuidos por los proveedores a los distintos hoteles.
- **Empleado_contrato**: almacena un histórico de los empleados que han trabajado en la cadena hoteles en algún momento.

Relaciones

- **Trabaja:** almacena el historial de cada empleado que trabaja en un hotel determinado.
- **Reserva:** almacena los datos de las distintas reservas que haga cada cliente en los diferentes hoteles de la cadena hotelera.
- **Suministro:** almacena los datos de los suministros que hagan los proveedores a lo casa hotel de la cadena, así como la fecha del suministro.
- **Dirige:** relación entre empleado y hotel que sirve para asociar el código del director de un hotel (empleado) con el hotel que dirige.

1.3.2 Paso a tablas

La entidad Cliente contendrá a todos los clientes dados de alta en algún hotel de la cadena. Como clave primaria (PK) tendremos el código de cada cliente.

Cliente

(codigo_cliente, DNI, Nombre_cliente, Telefono)

En la tabla Empleado_contrato recogeremos los registros de los empleados que hayan trabajado en algún hotel de la cadena en algún momento, a modo de histórico. La clave primaria será el código del empleado y las fechas de inicio y final del contrato en un hotel.

Empleado_contrato

(codigo_empleado, codigo_hotel, fecha_inicio_hotel, fecha_final_hotel)

La clave primaria es el código del empleado. Un empleado solo podrá estar trabajando en un único hotel, pero podrá aparecer en más de un registro en el histórico si se da el caso de que dicho empleado haya trabajado en otros hoteles en otro momento.

Empleado_trabaja

(codigo_empleado, DNI, nombre, direccion, telefono, fecha_inicio_cadena, salario, codigo_hotel, fecha_inicio_hotel)

Las tablas Hotel y Dirige serán unificadas, creando así Hotel-Dirige, que tendrá como clave primaria el código del hotel, clave que comparten ambas tablas. En cada hotel, solo podrá haber un empleado como Director.

Hotel_dirige

(codigo_hotel, nombre, habitaciones_dobles, habitaciones_sencillas, ciudad, provincia, director)

Con la relación Reserva haremos una tabla donde recoger los datos de los clientes que realicen dichas reservas junto con el hotel donde las realicen. Como clave primaria tendremos la fecha de entrada y salida, el código del hotel y el código del cliente

Reserva

(entrada, salida, habitacion, precio, codigo, codigo_cliente)

Unificaremos la tabla Artículo y la relación Vende, recogiendo en dicha tabla las ventas realizadas por los proveedores y el código del artículo que se ha suministrado. Las claves primaria serán el código del artículo y el proveedor.

Articulo vende

(codigo_articulo, nombre, tipo, codigo_proveedor)

Para la entidad Proveedor, recogeremos los datos del mismo en la tabla Proveedor, siendo su código identificador la clave primaria de dicha tabla.

Proveedor

(codigo_proveedor, nombre, ciudad)

Con la relación Suministro unificaremos las tablas Artículo y Hotel, obteniendo la tabla T_Suministro donde recogeremos los datos de los diferentes suministros realizados por un proveedor a un hotel. La clave primaria sera el código del artículo, el proveedor, el hotel y la fecha de dicho suministro.

T suministro

(codigo_articulo, cantidad, precio, fecha_suministro, codigo, codigo_proveedor)

Práctica 2:

Diseño de la fragmentación y de la asignación

2.1 Hoteles

En primer lugar, la entidad Hotel, es una fuerte candidata para ser fragmentada. Los hoteles los fragmentamos mediante una fragmentación horizontal primaria a partir del atributo provincia. Todos los hoteles están distribuidos en ocho provincias, de tal forma que la fragmentación quedaría como sigue:

Conjunto de predicados P simple:

$P = \{$ provincia = 'Granada',
provincia = 'Jaén',
provincia = 'Cádiz',
provincia = 'Huelva',
provincia = 'Sevilla',
provincia = 'Córdoba',
provincia = 'Málaga',
provincia = 'Almería'
 $\}$

Obtenemos 8 fragmentos, por simplicidad usaremos la siguiente notación:

p1: provincia = 'Granada'
p2: provincia = 'Jaén'
p3: provincia = 'Cádiz'
p4: provincia = 'Huelva'
p5: provincia = 'Sevilla'
p6: provincia = 'Córdoba'
p7: provincia = 'Málaga'
p8: provincia = 'Almería'

El total de expresiones conjuntivas: $2^8 = 256$ expresiones conjuntivas. De las cuales las que se evalúan en verdadero para nuestro caso:

$y_1 = \{ p_1 \vee \text{not } p_2 \vee \text{not } p_3 \vee \text{not } p_4 \vee \text{not } p_5 \vee \text{not } p_6 \vee \text{not } p_7 \vee \text{not } p_8 \}$
 $y_2 = \{ \text{not } p_1 \vee p_2 \vee \text{not } p_3 \vee \text{not } p_4 \vee \text{not } p_5 \vee \text{not } p_6 \vee \text{not } p_7 \vee \text{not } p_8 \}$
 $y_3 = \{ \text{not } p_1 \vee \text{not } p_2 \vee p_3 \vee \text{not } p_4 \vee \text{not } p_5 \vee \text{not } p_6 \vee \text{not } p_7 \vee \text{not } p_8 \}$
 $y_4 = \{ \text{not } p_1 \vee \text{not } p_2 \vee \text{not } p_3 \vee p_4 \vee \text{not } p_5 \vee \text{not } p_6 \vee \text{not } p_7 \vee p_8 \}$
 $y_5 = \{ \text{not } p_1 \vee p_2 \vee \text{not } p_3 \vee \text{not } p_4 \vee p_5 \vee \text{not } p_6 \vee \text{not } p_7 \vee \text{not } p_8 \}$
 $y_6 = \{ \text{not } p_1 \vee \text{not } p_2 \vee \text{not } p_3 \vee \text{not } p_4 \vee \text{not } p_5 \vee p_6 \vee \text{not } p_7 \vee \text{not } p_8 \}$
 $y_7 = \{ \text{not } p_1 \vee \text{not } p_2 \vee \text{not } p_3 \vee \text{not } p_4 \vee p_5 \vee p_6 \vee \text{not } p_7 \vee \text{not } p_8 \}$
 $y_8 = \{ \text{not } p_1 \vee \text{not } p_2 \vee \text{not } p_3 \vee \text{not } p_4 \vee \text{not } p_5 \vee \text{not } p_6 \vee \text{not } p_7 \vee p_8 \}$

Por simplicidad no incluimos el resto de expresiones, ya que los hoteles se agrupan por pares en cada una de nuestras Localidades:

- Localidad “Granada”: hoteles de todas las ciudades de las provincias de Granada y Jaén.
- Localidad “Cádiz”: hoteles de todas las ciudades de las provincias de Cádiz y Huelva.
- Localidad “Sevilla”: hoteles de todas las ciudades de las provincias de Sevilla y Córdoba.
- Localidad “Málaga”: hoteles de todas las ciudades de las provincias de Málaga y Almería.

Lo que se corresponden con el siguiente esquema de fragmentación:

$Hotel_1 = SL_{y_1} (Hotel)$

$Hotel_2 = SL_{y_2} (Hotel)$

$Hotel_3 = SL_{y_3} (Hotel)$

$Hotel_4 = SL_{y_4} (Hotel)$

$Hotel_5 = SL_5 (Hotel)$

$Hotel_6 = SL_{y_6} (Hotel)$

$Hotel_7 = SL_{y_7} (Hotel)$

$Hotel_8 = SL_{y_8} (Hotel)$

2.2 Proveedores

Los proveedores los fragmentamos mediante una fragmentación horizontal primaria mediante el atributo provincia, teniendo en cuenta que en este caso solo hay dos proveedores, uno en la provincia de Granada, que suministra a los hoteles de Jaén, Granada, Málaga y Almería; y otro en la provincia de Sevilla, suministra a los hoteles de Sevilla, Cordoba, Cadiz y Huelva.

Conjunto de predicados simples:

$P = \{ \text{Provincia_proveedor} = 'Granada'$
 $\text{Provincia_proveedor} = 'Sevilla' \}$

Por simplicidad usaremos la siguiente notación:

$p1: \text{provincia_proveedor} = 'Granada'$

$p2: \text{provincia_proveedor} = 'Sevilla'$

El total de expresiones conjuntivas: $2^2 = 4$ expresiones conjuntivas. De las cuales las que se evalúan en verdadero:

$y_1 = \{ p1 \vee p2 \} \Rightarrow$ falso ya que un proveedor no puede pertenecer a las dos provincias

$y_1 = \{ \text{not } p1 \vee \text{not } p2 \} \Rightarrow$ falso ya que los proveedores tienen que pertenecer a Granada o Sevilla

De forma que nos quedamos con estas expresiones:

$$y_2 = p_1 \vee \text{not } p_2$$

$$y_3 = \text{not } p_1 \vee p_2$$

De forma que los fragmentos con proveedores de Granada irán a la localidad de Granada y los fragmentos con la Provincia de Sevilla irán a la localidad de Sevilla.

2.3 Suministro

Para la tabla Suministra se realizará una fragmentación horizontal derivada , ya que queremos que cada suministro vaya en la localidad en la que se encuentra el hotel al que corresponde el propio suministro. De forma que los suministros se repartirán en las cuatro localidades existentes. Por ejemplo, en el caso de los hoteles de la provincia de Jaén, alojados en la localidad de Granada, sus suministros irán almacenados en dicha localidad también.

$$\text{ProveedorSuministra}_1 = \text{ProveedorSuministra SJN}_{\text{id_hotel} = \text{id_hotel}} \text{Hotel}_1 \text{ (Granada)}$$

$$\text{ProveedorSuministra}_2 = \text{ProveedorSuministra SJN}_{\text{id_hotel} = \text{id_hotel}} \text{Hotel}_2 \text{ (Jaén)}$$

$$\text{ProveedorSuministra}_3 = \text{ProveedorSuministra SJN}_{\text{id_hotel} = \text{id_hotel}} \text{Hotel}_3 \text{ (Cadiz)}$$

$$\text{ProveedorSuministra}_4 = \text{ProveedorSuministra SJN}_{\text{id_hotel} = \text{id_hotel}} \text{Hotel}_4 \text{ (Huelva)}$$

$$\text{ProveedorSuministra}_5 = \text{ProveedorSuministra SJN}_{\text{id_hotel} = \text{id_hotel}} \text{Hotel}_5 \text{ (Sevilla)}$$

$$\text{ProveedorSuministra}_6 = \text{ProveedorSuministra SJN}_{\text{id_hotel} = \text{id_hotel}} \text{Hotel}_6 \text{ (Córdoba)}$$

$$\text{ProveedorSuministra}_7 = \text{ProveedorSuministra SJN}_{\text{id_hotel} = \text{id_hotel}} \text{Hotel}_7 \text{ (Málaga)}$$

$$\text{ProveedorSuministra}_8 = \text{ProveedorSuministra SJN}_{\text{id_hotel} = \text{id_hotel}} \text{Hotel}_8 \text{ (Almería)}$$

2.4 Empleados

Los empleados los fragmentamos mediante una fragmentación horizontal derivada usando el atributo `id_hotel`, porque nos interesa que los datos de los empleados vayan junto a los datos de las tablas de los hoteles en los que trabajan.

$$\text{Empleado}_i = \text{Empleado SJ}_{\#id_hotel = id_hotel} \text{Hotel}_i$$

2.5 Replicación de datos

Las tablas de Cliente y Artículo no se fragmentan, ya que hay un criterio lógico claro y las operaciones realizadas sobre estas tablas serán operaciones básicas, de forma que estas relaciones se podrán replicar en cada una de nuestras localidades si fuera conveniente.

2.6 Localidades asociadas a la base de datos

1 → Granada

2 → Cádiz

3 → Sevilla

4 → Málaga

Práctica 3:

Implementación del diseño

3.1 Creación de tablas

Las tablas se crearán en cada una de las localidades, siguiendo la fragmentación especificada anteriormente. A tener en cuenta que la tabla PROVEEDOR solo se creará en las localidades Granada y Sevilla, como se especifica en el apartado de fragmentación.

- **HOTEL_DIRIGE**

```
CREATE TABLE HOTEL_DIRIGE (  
    codigo INT PRIMARY KEY,  
    nombre VARCHAR(20) NOT NULL,  
    habitaciones_dobles INT,  
    habitaciones_sencillas INT,  
    ciudad INT,  
    provincia INT,  
    director INT  
);
```

- **CLIENTE**

```
CREATE TABLE CLIENTE (  
    codigo_cliente INT NOT NULL,  
    dni INT NOT NULL ,  
    nombre VARCHAR ( 20 ) NOT NULL ,  
    telefono INT NOT NULL ,  
    PRIMARY KEY (codigo_cliente)  
);
```

- **EMPLEADO_CONTRATO**

```
CREATE TABLE EMPLEADO_CONTRATO (  
    codigo_empleado INT ,  
    codigo INT NOT NULL REFERENCES HOTEL_DIRIGE(codigo),  
    fecha_inicio_hotel DATE NOT NULL ,  
    fecha_final_hotel DATE NOT NULL ,  
    PRIMARY KEY (codigo_empleado, fecha_inicio_hotel,  
    fecha_final_hotel));
```

- EMPLEADO_TRABAJA

```
CREATE TABLE EMPLEADO_TRABAJA (  
    codigo_empleado int,  
    dni INT NOT NULL ,  
    nombre VARCHAR ( 20 ) NOT NULL ,  
    direccion VARCHAR ( 50 ),  
    telefono INT ,  
    fecha_inicio_cadena DATE NOT NULL ,  
    salario FLOAT ( 10 ) NOT NULL ,  
    codigo INT NOT NULL REFERENCES HOTEL_DIRIGE(codigo),  
    fecha_inicio_hotel DATE NOT NULL,  
    PRIMARY KEY (codigo_empleado));
```

- RESERVA

```
CREATE TABLE RESERVA (  
    entrada DATE NOT NULL ,  
    salida DATE NOT NULL ,  
    habitacion VARCHAR ( 10 ) NOT NULL ,  
    precio FLOAT ( 10 ) NOT NULL ,  
    codigo INT NOT NULL REFERENCES HOTEL_DIRIGE(codigo),  
    codigo_cliente INT NOT NULL REFERENCES CLIENTE(codigo_cliente),  
    PRIMARY KEY (codigo_cliente, entrada, salida,  
    codigo)  
);
```

- T_SUMINISTRO

```
CREATE TABLE T_SUMINISTRO (  
    codigo_articulo INT NOT NULL ,  
    cantidad INT NOT NULL ,  
    precio FLOAT ( 10 ) NOT NULL ,  
    fecha_suministro DATE NOT NULL ,  
    codigo INT ,  
    codigo_proveedor INT NOT NULL ,  
    PRIMARY KEY (codigo_articulo, codigo_proveedor, codigo,  
    fecha_suministro)  
);
```

Las tablas PROVEEDOR y ARTICULO_VENDE serán creads solamente en las Localidades 1 y 3, respectivas localidad de los Proveedores Granada y Sevilla.

- **PROVEEDOR**

```
CREATE TABLE PROVEEDOR (  
    codigo_proveedor INT NOT NULL ,  
    nombre VARCHAR ( 20 ) NOT NULL ,  
    ciudad VARCHAR ( 20 ) NOT NULL ,  
    PRIMARY KEY (codigo_proveedor)  
);
```

- **ARTICULO_VENDE**

```
CREATE TABLE ARTICULO_VENDE (  
    codigo_articulo INT NOT NULL ,  
    nombre VARCHAR ( 20 ) NOT NULL ,  
    tipo CHAR ( 1 ) NOT NULL ,  
    codigo_proveedor INT NOT NULL REFERENCES PROVEEDOR(codigo_proveedor),  
    PRIMARY KEY (codigo_articulo, codigo_proveedor)  
);
```

3.2 Garantización de privilegios

Los privilegios se concederán en cada una de las localidades. Esto nos permitirá hacer consultas, insertar, borrar y actualizar registros desde cualquier localidad que nos interese hacia otra. Por seguridad, en un caso real se restringiría estas acciones, pero por comodidad y fines pedagógicos, en esta práctica vamos a darle privilegios a todas las localidades.

- GRANT Localidad 1: Granada

```
Grant SELECT , INSERT , DELETE , UPDATE ON PROVEEDOR TO transformer2, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON EMPLEADO_TRABAJA TO transformer2, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON EMPLEADO_CONTRATO TO transformer2, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON RESERVA TO transformer2, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON CLIENTE TO transformer2, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON T_SUMINISTRO TO transformer2, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON HOTEL_DIRIGE TO transformer2, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON ARTICULO_VENDE TO transformer2, transformer4;
```

- GRANT Localidad 2: Cadiz

```
Grant SELECT , INSERT , DELETE , UPDATE ON PROVEEDOR TO transformer1, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON EMPLEADO_TRABAJA TO transformer1, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON EMPLEADO_CONTRATO TO transformer1, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON RESERVA TO transformer1, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON CLIENTE TO transformer1, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON T_SUMINISTRO TO transformer1, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON HOTEL_DIRIGE TO transformer1, transformer3, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON ARTICULO_VENDE TO transformer1, transformer4;
```

- GRANT Localidad 3: Sevilla

```
Grant SELECT , INSERT , DELETE , UPDATE ON PROVEEDOR TO transformer1, transformer2, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON EMPLEADO_TRABAJA TO transformer1, transformer2, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON EMPLEADO_CONTRATO TO transformer1, transformer2, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON RESERVA TO transformer1, transformer2, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON CLIENTE TO transformer1, transformer2, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON T_SUMINISTRO TO transformer1, transformer2, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON HOTEL_DIRIGE TO transformer1, transformer2, transformer4;
Grant SELECT , INSERT , DELETE , UPDATE ON ARTICULO_VENDE TO transformer1, transformer2, transformer4;
```

- GRANT Localidad 4: Málaga

```
Grant SELECT , INSERT , DELETE , UPDATE ON PROVEEDOR TO transformer1, transformer2, transformer3;
Grant SELECT , INSERT , DELETE , UPDATE ON EMPLEADO_TRABAJA TO transformer1, transformer2, transformer3;
Grant SELECT , INSERT , DELETE , UPDATE ON EMPLEADO_CONTRATO TO transformer1, transformer2, transformer3;
Grant SELECT , INSERT , DELETE , UPDATE ON RESERVA TO transformer1, transformer2, transformer3;
Grant SELECT , INSERT , DELETE , UPDATE ON CLIENTE TO transformer1, transformer2, transformer3;
Grant SELECT , INSERT , DELETE , UPDATE ON T_SUMINISTRO TO transformer1, transformer2, transformer3;
Grant SELECT , INSERT , DELETE , UPDATE ON HOTEL_DIRIGE TO transformer1, transformer2, transformer3;
Grant SELECT , INSERT , DELETE , UPDATE ON ARTICULO_VENDE TO transformer1, transformer2;
```

3.3 Creación de vistas

Localidad 1: Granada

- Vistas globales. Tabla: EMPLEADO_TRABAJA

```
CREATE VIEW EMPLEADO AS
SELECT * FROM EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer2.EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer3.EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer4.EMPLEADO_TRABAJA;
```

- Vistas globales. Tabla: EMPLEADO_CONTRATO

```
CREATE VIEW HISTORICO_EMPLEADOS AS
SELECT * FROM EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer2.EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer3.EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer4.EMPLEADO_CONTRATO;
```

- Vistas globales. Tabla: CLIENTE

```
CREATE VIEW CLIENTES AS
SELECT * FROM CLIENTE
UNION
SELECT * FROM transformer2.CLIENTE
UNION
SELECT * FROM transformer3.CLIENTE
UNION
SELECT * FROM transformer4.CLIENTE;
```


- Vistas globales. Tabla: RESERVA

```
CREATE VIEW RESERVAS AS
SELECT * FROM RESERVA
UNION
SELECT * FROM transformer2.RESERVA
UNION
SELECT * FROM transformer3.RESERVA
UNION
SELECT * FROM transformer4.RESERVA;
```

- Vistas globales. Tabla: PROVEEDOR

```
CREATE VIEW PROVEEDORES AS
SELECT * FROM PROVEEDOR
UNION
SELECT * FROM transformer3.PROVEEDOR;
```

- Vistas globales. Tabla: HOTEL_DIRIGE

```
CREATE VIEW HOTEL AS
SELECT * FROM HOTEL_DIRIGE
UNION
SELECT * FROM transformer2.HOTEL_DIRIGE
UNION
SELECT * FROM transformer3.HOTEL_DIRIGE
UNION
SELECT * FROM transformer4.HOTEL_DIRIGE;
```

- Vistas globales. Tabla: ARTICULO

```
CREATE VIEW ARTICULO AS
SELECT * FROM ARTICULO_VENDE
UNION
SELECT * FROM transformer3.ARTICULO_VENDE;
```

- Vistas globales. Tabla: T_SUMINISTRO

```
CREATE VIEW SUMINISTROS AS  
SELECT * FROM SUMINISTRO  
UNION  
SELECT * FROM transformer2.SUMINISTRO  
UNION  
SELECT * FROM transformer3.SUMINISTRO  
UNION  
SELECT * FROM transformer4.SUMINISTRO;
```

Localidad 2: Cadiz

- Vistas globales. Tabla: EMPLEADO_TRABAJA

```
CREATE VIEW EMPLEADO AS
SELECT * FROM EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer1.EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer3.EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer4.EMPLEADO_TRABAJA;
```

- Vistas globales. Tabla: EMPLEADO_CONTRATO

```
CREATE VIEW HISTORICO_EMPLEADOS AS
SELECT * FROM EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer1.EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer3.EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer4.EMPLEADO_CONTRATO;
```

- Vistas globales. Tabla: CLIENTE

```
CREATE VIEW CLIENTES AS
SELECT * FROM CLIENTE
UNION
SELECT * FROM transformer1.CLIENTE
UNION
SELECT * FROM transformer3.CLIENTE
UNION
SELECT * FROM transformer4.CLIENTE;
```

- Vistas globales. Tabla: RESERVA

```
CREATE VIEW RESERVAS AS
SELECT * FROM RESERVA
UNION
SELECT * FROM transformer1.RESERVA
UNION
SELECT * FROM transformer3.RESERVA
UNION
SELECT * FROM transformer4.RESERVA;
```

- Vistas globales. Tabla: HOTEL_DIRIGE

```
CREATE VIEW HOTEL AS
SELECT * FROM HOTEL_DIRIGE
UNION
SELECT * FROM transformer1.HOTEL_DIRIGE
UNION
SELECT * FROM transformer3.HOTEL_DIRIGE
UNION
SELECT * FROM transformer4.HOTEL_DIRIGE;
```

- Vistas globales. Tabla: ARTICULO

```
CREATE VIEW ARTICULO AS
SELECT * FROM ARTICULO_VENDE
UNION
SELECT * FROM transformer3.ARTICULO_VENDE;
```

- Vistas globales. Tabla: T_SUMINISTRO

```
CREATE VIEW SUMINISTROS AS
SELECT * FROM SUMINISTRO
UNION
SELECT * FROM transformer2.SUMINISTRO
UNION
SELECT * FROM transformer3.SUMINISTRO
UNION
SELECT * FROM transformer4.SUMINISTRO;
```

Localidad 3: Sevilla

- Vistas globales. Tabla: EMPLEADO_TRABAJA

```
CREATE VIEW EMPLEADO AS
SELECT * FROM EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer2.EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer1.EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer4.EMPLEADO_TRABAJA;
```

- Vistas globales. Tabla: EMPLEADO_CONTRATO

```
CREATE VIEW HISTORICO_EMPLEADOS AS
SELECT * FROM EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer2.EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer1.EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer4.EMPLEADO_CONTRATO;
```

- Vistas globales. Tabla: CLIENTE

```
CREATE VIEW CLIENTES AS
SELECT * FROM CLIENTE
UNION
SELECT * FROM transformer2.CLIENTE
UNION
SELECT * FROM transformer1.CLIENTE
UNION
SELECT * FROM transformer4.CLIENTE;
```

- Vistas globales. Tabla: RESERVA

```
CREATE VIEW RESERVAS AS
SELECT * FROM RESERVA
UNION
SELECT * FROM transformer2.RESERVA
UNION
SELECT * FROM transformer1.RESERVA
UNION
SELECT * FROM transformer4.RESERVA;
```

- Vistas globales. Tabla: PROVEEDOR

```
CREATE VIEW PROVEEDORES AS
SELECT * FROM PROVEEDOR
UNION
SELECT * FROM transformer1.PROVEEDOR;
```

- Vistas globales. Tabla: HOTEL_DIRIGE

```
CREATE VIEW HOTEL AS
SELECT * FROM HOTEL_DIRIGE
UNION
SELECT * FROM transformer2.HOTEL_DIRIGE
UNION
SELECT * FROM transformer1.HOTEL_DIRIGE
UNION
SELECT * FROM transformer4.HOTEL_DIRIGE;
```

- Vistas globales. Tabla: ARTICULO

```
CREATE VIEW ARTICULO AS
SELECT * FROM ARTICULO_VENDE
UNION
SELECT * FROM transformer1.ARTICULO_VENDE;
```

- Vistas globales. Tabla: T_SUMINISTRO

```
CREATE VIEW SUMINISTROS AS
SELECT * FROM T_SUMINISTRO
UNION
SELECT * FROM transformer1.T_SUMINISTRO
UNION
SELECT * FROM transformer2.T_SUMINISTRO
UNION
SELECT * FROM transformer4.T_SUMINISTRO;
```

Localidad 4: Málaga

- Vistas globales. Tabla: EMPLEADO_TRABAJA

```
CREATE VIEW EMPLEADO AS
SELECT * FROM EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer1.EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer3.EMPLEADO_TRABAJA
UNION
SELECT * FROM transformer2.EMPLEADO_TRABAJA;
```

- Vistas globales. Tabla: EMPLEADO_CONTRATO

```
CREATE VIEW HISTORICO_EMPLEADOS AS
SELECT * FROM EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer1.EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer3.EMPLEADO_CONTRATO
UNION
SELECT * FROM transformer2.EMPLEADO_CONTRATO;
```

- Vistas globales. Tabla: CLIENTE

```
CREATE VIEW CLIENTES AS
SELECT * FROM CLIENTE
UNION
SELECT * FROM transformer1.CLIENTE
UNION
SELECT * FROM transformer3.CLIENTE
UNION
SELECT * FROM transformer2.CLIENTE;
```

- Vistas globales. Tabla: RESERVA

```
CREATE VIEW RESERVAS AS
SELECT * FROM RESERVA
UNION
SELECT * FROM transformer2.RESERVA
UNION
SELECT * FROM transformer1.RESERVA
UNION
SELECT * FROM transformer4.RESERVA;
```


- Vistas globales. Tabla: HOTEL_DIRIGE

```
CREATE VIEW HOTEL AS
SELECT * FROM HOTEL_DIRIGE
UNION
SELECT * FROM transformer1.HOTEL_DIRIGE
UNION
SELECT * FROM transformer3.HOTEL_DIRIGE
UNION
SELECT * FROM transformer2.HOTEL_DIRIGE;
```

- Vistas globales. Tabla: ARTICULO

```
CREATE VIEW SUMINISTROS AS
SELECT * FROM T_SUMINISTRO
UNION
SELECT * FROM transformer1.T_SUMINISTRO
UNION
SELECT * FROM transformer2.T_SUMINISTRO
UNION
SELECT * FROM transformer3.T_SUMINISTRO;
```

- Vistas globales. Tabla: T_SUMINISTRO

```
CREATE VIEW SUMINISTROS AS
SELECT * FROM T_SUMINISTRO
UNION
SELECT * FROM transformer1.T_SUMINISTRO
UNION
SELECT * FROM transformer2.T_SUMINISTRO
UNION
SELECT * FROM transformer4.T_SUMINISTRO;
```

3.4 Disparadores

Los disparadores nos sirven para poner restricciones cuando insertamos o actualizamos una tabla. Esto “simula” de alguna manera el funcionamiento de las claves externas, aunque hemos metido muchas más funcionalidades que eso para cumplir con los requisitos de la práctica.

- Disparador asociado a la tabla: EMPLEADO-TRABAJA

```
create or replace TRIGGER EMPLEADO_TRABAJA_TRIGGER
  BEFORE INSERT OR UPDATE ON EMPLEADO_TRABAJA FOR EACH ROW
DECLARE
  contarEmpleados NUMBER ;
BEGIN
  if INSERTING THEN
    SELECT COUNT (*) INTO contarEmpleados FROM EMPLEADO WHERE CODIGO_EMPLEADO = :NEW.CODIGO_EMPLEADO;
    if contarEmpleados > 0 then
      raise_application_error(-20004, 'TAL EMPLEADO YA EXISTE');
    END IF ;
  END IF ;
END ;
```

- Disparador asociado a la tabla: HOTEL-DIRIGE

```
create or replace TRIGGER HOTEL_DIRIGE_TRIGGER
  BEFORE INSERT OR UPDATE ON HOTEL_DIRIGE FOR EACH ROW
DECLARE
  contarEmpleados NUMBER ;
  contarHoteles NUMBER;
  contarEmpleadoDirige NUMBER;
BEGIN
  IF INSERTING THEN
    SELECT COUNT (*) INTO contarHoteles FROM HOTEL WHERE CODIGO = :NEW.CODIGO;
    IF contarHoteles > 0 THEN
      raise_application_error(-20004, 'TAL HOTEL YA EXISTE');
    END IF ;
  END IF ;
END ;
```

- Disparador asociado a la tabla: CLIENTE

```
create or replace TRIGGER CLIENTE_TRIGGER
  BEFORE INSERT OR UPDATE ON CLIENTE FOR EACH ROW
DECLARE
  contarClientes NUMBER ;
BEGIN
  IF INSERTING THEN
    SELECT COUNT (*) INTO contarClientes FROM CLIENTE WHERE CODIGO_CLIENTE = :NEW.CODIGO_CLIENTE;
    IF contarClientes > 0 THEN
      raise_application_error(-20004, 'TAL CLIENTE YA EXISTE');
    END IF ;
  END IF ;
END ;
```

- Disparador asociado a la tabla: ARTICULOS

```
create or replace TRIGGER ARTICULO_VENDE_TRIGGER
  BEFORE INSERT OR UPDATE ON ARTICULO_VENDE FOR EACH ROW
DECLARE
  numArticulos NUMBER ;
  numProveedores NUMBER;
  provinciaProveedor_nuevo VARCHAR2(20);
  provinciaProveedor_almacenado VARCHAR2(20);
  codigo_proveedor_existe NUMBER;
BEGIN
  if INSERTING THEN
    if :NEW.TIPO != 'A' and :NEW.TIPO != 'B' and :NEW.TIPO != 'C' and :NEW.TIPO != 'D' then
      raise_application_error( -20004 , 'EL TIPO DE TAL ARTICULO NO ES VALIDO' );
    END IF ;
    SELECT COUNT (*) INTO numProveedores FROM PROVEEDOR WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and CIUDAD = 'Granada' ;
    if numProveedores = 0 then
      SELECT COUNT (*) INTO numProveedores FROM PROVEEDOR WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and CIUDAD = 'Sevilla' ;
      if numProveedores = 0 then
        raise_application_error(-20004,'TAL PROVEEDOR NO ES NI DE GRANADA NI DE SEVILLA');
      ELSE
        SELECT CIUDAD INTO provinciaProveedor_nuevo FROM PROVEEDOR WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and CIUDAD = 'Sevilla' ;
      END IF ;
    ELSE
      SELECT CIUDAD INTO provinciaProveedor_nuevo FROM PROVEEDOR WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and CIUDAD = 'Granada' ;
    END IF ;
    SELECT COUNT (*) INTO numArticulos FROM ARTICULO WHERE CODIGO_ARTICULO = :NEW.CODIGO_ARTICULO;
    if numArticulos = 2 then
      raise_application_error(-20004,'TAL ARTICULO YA LE SUMINISTRAN 2 PROVEEDORES');
    elsif numArticulos = 1 then
      SELECT CODIGO_PROVEEDOR INTO codigo_proveedor_existe FROM ARTICULO WHERE CODIGO_ARTICULO = :NEW.CODIGO_ARTICULO;
      SELECT CIUDAD INTO provinciaProveedor_almacenado FROM PROVEEDOR WHERE CODIGO_PROVEEDOR = codigo_proveedor_existe;
      if provinciaProveedor_nuevo = provinciaProveedor_almacenado then
        raise_application_error(-20004,'TAL ARTICULO YA LE SUMINISTRAN UN PROVEEDOR DE LA MISMA CIUDAD QUE LA DEL NUEVO PROVEEDOR');
      END IF ;
    END IF ;
  END IF ;
END ;
```

- Disparador asociado a la tabla: T_SUMINISTRO

```
create or replace TRIGGER SUMINISTRO_TRIGGER
BEFORE INSERT OR UPDATE ON t_suministro FOR EACH ROW
DECLARE
cnt NUMBER;
existeSuministro NUMBER;
cntProveedorArticulo NUMBER;
cntArticulo NUMBER;
cntProveedor NUMBER;
cntHotel NUMBER;
maxPrecioAnterior NUMBER;
ciudadProveedor VARCHAR(20);
ciudadHotel VARCHAR(20);

BEGIN

IF :new.codigo_articulo is NULL OR :new.codigo_proveedor is NULL OR :new.codigo is NULL
OR :new.FECHA_SUMINISTRO is NULL OR :new.CANTIDAD is NULL OR :new.precio is NULL THEN
    raise_application_error(-20004,'ERROR: VALORES NO VALIDOS');
END IF;

IF :new.precio <1 THEN
    raise_application_error(-20004,'ERROR: PRECIO INCORRECTO, INTRODUZCA UN VALOR
    MAYOR QUE 0');
END IF;

IF :new.CANTIDAD <-1 THEN
    raise_application_error(-20004,'ERROR: INTRODUZCA UNA CANTIDAD MAYOR O IGUAL QUE 0');
END IF;

SELECT COUNT(*) INTO cntHotel FROM HOTEL WHERE CODIGO = :new.codigo;
IF cntHotel = 0 THEN
    raise_application_error(-20004,'ERROR: EL HOTEL NO EXISTE');
END IF;

SELECT COUNT(*) INTO cntArticulo FROM ARTICULO WHERE CODIGO_ARTICULO = :new.codigo_articulo;
IF cntArticulo = 0 THEN
    raise_application_error(-20004,'ERROR: EL ARTICULO NO EXISTE');
END IF;

SELECT COUNT(*) INTO cntProveedor FROM PROVEEDORES WHERE CODIGO_PROVEEDOR = :new.codigo_proveedor;
IF cntProveedor = 0 THEN
    raise_application_error(-20004,'ERROR: EL PROVEEDOR NO EXISTE');
END IF;
```

- Disparador asociado a la tabla: ARTICULO

```
SELECT COUNT(*) INTO cntProveedorArticulo FROM ARTICULO
WHERE CODIGO_PROVEEDOR = codigo_proveedor AND CODIGO_ARTICULO = :new.codigo_articulo;

IF cntProveedorArticulo= 0 THEN
    raise_application_error(-20004,'ERROR: EL PROVEEDOR INSERTADO NO PROVEE DICHO ARTICULO');
END IF;

SELECT MAX(precio) INTO maxPrecioAnterior
FROM SUMINISTRO
WHERE CODIGO = :new.codigo
AND CODIGO_ARTICULO = :new.codigo_articulo;

IF maxPrecioAnterior > :new.precio THEN
    raise_application_error(-20004,'ERROR: EL PRECIO DEL NUEVO SUMINISTROS NO PUEDE
    SER INFERIOR AL DEL ANTIGUO SUMINISTRO');
END IF;

SELECT CIUDAD INTO ciudadProveedor FROM PROVEEDORES
WHERE CODIGO_PROVEEDOR=:new.codigo_proveedor;

SELECT CIUDAD INTO ciudadHotel FROM HOTEL
WHERE CODIGO = :new.codigo;

IF ciudadProveedor='Sevilla' AND
(ciudadHotel='Granada' OR ciudadHotel='Malaga' OR ciudadHotel= 'Almeria'
OR ciudadHotel='Jaen') THEN
    raise_application_error(-20004,'ERROR: EL PROVEEDOR DE SEVILLA NO PUEDE PROVEER
    A UN HOTEL DE LAS PROVINCIAS GRANADA, MALAGA, ALMERIA O JAEN');

ELSE IF ciudadProveedor='Granada' AND
(ciudadHotel='Sevilla' OR ciudadHotel='Cadiz' OR ciudadHotel= 'Cordoba'
OR ciudadHotel='Huelva') THEN
    raise_application_error(-20004,'ERROR: EL PROVEEDOR DE GRANADA NO PUEDE PROVEER
    A UN HOTEL DE LAS PROVINCIAS SEVILLA, HUELVA, CADIZ O CORDOBA');
END IF;
END IF;
END;
```

- Disparador asociado a la tabla: RESERVA

```
create or replace TRIGGER RESERVA_TRIGGER
  BEFORE INSERT OR UPDATE ON RESERVA FOR EACH ROW
DECLARE
  contarReservas NUMBER;
  contar NUMBER;
  contarCliente NUMBER;
BEGIN

  IF :new.CODIGO_CLIENTE is NULL OR :new.entrada is NULL OR :new.salida is NULL OR
    :new.codigo is NULL THEN
    raise_application_error(-20004,'NO SE PERMITEN VALORES DE ENTRADA A NULL, POR FAVOR INTRODUZCA
    VALORES VALIDOS');
  END IF;

  SELECT COUNT(*) INTO contarCliente FROM CLIENTES WHERE CODIGO_CLIENTE = :new.CODIGO_CLIENTE;

  IF contarCliente = 0 THEN
    raise_application_error(-20004, 'TAL CLIENTE NO EXISTE');
  END IF;

  SELECT COUNT(*) INTO contarReservas FROM RESERVAS WHERE CODIGO_CLIENTE = :new.CODIGO_CLIENTE
  AND entrada = :new.entrada AND salida = :new.salida;

  IF contarReservas = 0 THEN
    raise_application_error(-20004, 'TAL CLIENTE NO TIENE RESERVAS EN LA FECHA INDICADA');
  END IF;
END;
```

Práctica 4:

Implementación de las actualizaciones

4.1 Procedimientos

Los procedimientos nos permiten tanto hacer parte de la función de un disparador, por tanto hemos añadido algunas de las restricciones que requería el enunciado, como realizar de forma más práctica y desde la primera localidad acciones como la inserción, modificación o eliminación de tuplas.

- **Procedimiento para insertar un empleado:**

```
create or replace PROCEDURE
insertarEmpleado(cod_emp NUMBER , dni NUMBER , direccion VARCHAR , fecha_inicio_cadena DATE ,
fecha_inicio_hotel DATE , salario FLOAT , nombre VARCHAR ,telefono NUMBER , codigo NUMBER ) IS
    contar NUMBER ;
    contarMaximaFechaFinContrato DATE;
BEGIN
    if cod_emp is null OR dni is NULL OR fecha_inicio_cadena is NULL OR
    fecha_inicio_hotel is NULL OR salario is NULL OR nombre is NULL OR codigo is NULL THEN
        raise_application_error(-20004 , 'LOS UNICOS DATOS QUE PUEDEN ESTAR A NULL SON DIRECCION Y TELEFONO,
        LOS DEMAS DEBEN SER VALORES VALIDOS' );
    END IF ;
    SELECT COUNT (*) INTO contar FROM HOTEL WHERE CODIGO = codigo;
    IF contar = 0 THEN
        raise_application_error(-20004,'EL HOTEL QUE SE HA INSERTADO NO EXISTE');
    END IF ;
    IF fecha_inicio_hotel < fecha_inicio_cadena THEN
        raise_application_error(-20004,'LA FECHA DE INICIO DEL CONTRATO DEL HOTEL ES INFERIOR
        A LA FECHA DE INICIO DEL CONTRATO EN LA CADENA DE HOTELES Y ESO NO PUEDE OCURRIR');
    END IF ;
    SELECT MAX (FECHA_FINAL_HOTEL) INTO contarMaximaFechaFinContrato FROM HISTORICO_EMPLEADOS
    WHERE CODIGO_EMPLEADO = cod_emp;
    IF contarMaximaFechaFinContrato is not NULL THEN
        IF contarMaximaFechaFinContrato > fecha_inicio_hotel THEN
            raise_application_error(-20004,'LA FECHA DE INICIO DE CONTRATO ES INFERIOR A LA ULTIMA FECHA DE FINAL
            DE CONTRATO DE ESTE EMPLEADO');
        END IF ;
    END IF ;
    SELECT COUNT (*) INTO contar FROM HOTEL_DIRIGE WHERE CODIGO = codigo;
    IF contar > 0 THEN
        INSERT INTO TRANSFORMER1.EMPLEADO_TRABAJA VALUES (cod_emp, dni, nombre, direccion, telefono, fecha_inicio_cadena,salario,
        codigo, fecha_inicio_hotel);
    ELSE
        SELECT COUNT (*) INTO contar FROM TRANSFORMER2.HOTEL_DIRIGE WHERE CODIGO = cod_emp;
        IF contar > 0 THEN
            INSERT INTO TRANSFORMER2.EMPLEADO_TRABAJA VALUES (cod_emp, dni, nombre, direccion, telefono, fecha_inicio_cadena,salario,
            codigo, fecha_inicio_hotel);
        ELSE
            SELECT COUNT (*) INTO contar FROM TRANSFORMER3.HOTEL_DIRIGE WHERE CODIGO = codigo;
            IF contar > 0 THEN
                INSERT INTO TRANSFORMER3.EMPLEADO_TRABAJA VALUES (cod_emp, dni, nombre, direccion, telefono, fecha_inicio_cadena,salario,
                codigo, fecha_inicio_hotel);
```

```
ELSE
    SELECT COUNT (*) INTO contar FROM TRANSFORMER4.HOTEL_DIRIGE WHERE CODIGO = codigo;
    IF contar > 0 THEN
        INSERT INTO TRANSFORMER4.EMPLEADO_TRABAJA VALUES (cod_emp, dni, nombre, direccion, telefono, fecha_inicio_cadena,salario,
            codigo, fecha_inicio_hotel);
    END IF ;
END IF ;
END IF ;
END IF ;
END ;
```


- Procedimiento para eliminar un empleado:

```
create or replace PROCEDURE
eliminarEmpleado(cod_emp NUMBER , fecha_final_hotel DATE ) IS
contar NUMBER ;
fecha_inicio DATE;
hotel_trabajaba NUMBER;
BEGIN
    IF cod_emp is NULL OR fecha_final_hotel is NULL THEN
        raise_application_error( -20004 , 'DEBE DE INSERTAR VALORES CORRECTOS, LOS VALORES NULL NO SON CORRECTOS' );
    END IF ;
    SELECT COUNT (*) INTO contar FROM TRANSFORMER1.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
    IF contar > 0 THEN
        SELECT CODIGO, FECHA_INICIO_HOTEL INTO hotel_trabajaba, fecha_inicio
        FROM TRANSFORMER1.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
        IF fecha_final_hotel < fecha_inicio THEN
            raise_application_error(-20004,'TAL FECHA FINAL DE CONTRATO ES MENOR QUE LA FECHA DE INICIO DE CONTRATO');
        END IF ;
        INSERT INTO TRANSFORMER1.EMPLEADO_CONTRATO VALUES (cod_emp,hotel_trabajaba, fecha_inicio, fecha_final_hotel);
        DELETE FROM TRANSFORMER1.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
    ELSE
        SELECT COUNT (*) INTO contar FROM TRANSFORMER2.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
        IF contar > 0 THEN
            SELECT CODIGO, FECHA_INICIO_HOTEL INTO hotel_trabajaba, fecha_inicio
            FROM TRANSFORMER2.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
            IF fecha_final_hotel < fecha_inicio THEN
                raise_application_error(-20004,'TAL FECHA FINAL DE CONTRATO ES MENOR QUE LA FECHA DE INICIO DE CONTRATO');
            END IF ;
            INSERT INTO TRANSFORMER2.EMPLEADO_CONTRATO VALUES (cod_emp,hotel_trabajaba, fecha_inicio, fecha_final_hotel);
            DELETE FROM TRANSFORMER2.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
        ELSE
            SELECT COUNT (*) INTO contar FROM TRANSFORMER3.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
            IF contar > 0 THEN
                SELECT CODIGO, FECHA_INICIO_HOTEL INTO hotel_trabajaba, fecha_inicio
                FROM TRANSFORMER3.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
                IF fecha_final_hotel < fecha_inicio THEN
                    raise_application_error(-20004,'TAL FECHA FINAL DE CONTRATO ES MENOR QUE LA FECHA DE INICIO DE CONTRATO');
                END IF ;
                INSERT INTO TRANSFORMER3.EMPLEADO_CONTRATO VALUES (cod_emp,hotel_trabajaba, fecha_inicio, fecha_final_hotel);
                DELETE FROM TRANSFORMER3.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
            ELSE
                SELECT COUNT (*) INTO contar FROM TRANSFORMER4.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
                IF contar > 0 THEN
                    SELECT CODIGO, FECHA_INICIO_HOTEL INTO hotel_trabajaba, fecha_inicio
                    FROM TRANSFORMER4.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
                    IF fecha_final_hotel < fecha_inicio THEN
                        raise_application_error(-20004,'TAL FECHA FINAL DE CONTRATO ES MENOR QUE LA FECHA DE INICIO DE CONTRATO');
                    END IF ;
                    INSERT INTO TRANSFORMER4.EMPLEADO_CONTRATO VALUES (cod_emp,hotel_trabajaba, fecha_inicio, fecha_final_hotel);
                    DELETE FROM TRANSFORMER4.EMPLEADO_TRABAJO WHERE CODIGO_EMPLEADO = cod_emp;
                ELSE
                    raise_application_error(-20004,'TAL EMPLEADO NO EXISTE');
                END IF ;
            END IF ;
        END IF ;
    END IF ;
END ;
```

- Procedimiento para modificar el salario de un empleado:

```
create or replace PROCEDURE modificarSalarioEmpleado(codigo NUMBER , salario_p FLOAT ) IS
    contar NUMBER ;
    salarioAntiguo FLOAT;
    contarEmpleado NUMBER;
BEGIN

    IF codigo is NULL or salario_p is NULL THEN
        raise_application_error( -20004 , 'NO PUEDE HABER PARAMETROS A NULL' );
    END IF ;

    SELECT COUNT (*) INTO contarEmpleado FROM EMPLEADO WHERE CODIGO_EMPLEADO = codigo;
    IF contarEmpleado = 0 THEN
        raise_application_error(-20004,'EL CODIGO DE EMPLEADO NO EXISTE');
    END IF ;

    SELECT SALARIO INTO salarioAntiguo FROM EMPLEADO WHERE CODIGO_EMPLEADO = codigo;
    IF salarioAntiguo > salario_p THEN
        raise_application_error(-20004,'EL NUEVO SALARIO ES INFERIOR AL SALARIO ACTUAL');
    ELSIF salarioAntiguo = salario_p THEN
        raise_application_error(-20004,'EL NUEVO SALARIO ES IGUAL AL SALARIO ACTUAL');
    END IF ;

    SELECT COUNT (*) INTO contar FROM EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;
    IF contar > 0 THEN
        UPDATE EMPLEADO_TRABAJA SET SALARIO = salario_p WHERE CODIGO_EMPLEADO = codigo;
    ELSE
        SELECT COUNT (*) INTO contar FROM transformer2.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;
        IF contar > 0 THEN
            UPDATE transformer2.EMPLEADO_TRABAJA SET SALARIO = salario_p WHERE CODIGO_EMPLEADO = codigo;
        ELSE
            SELECT COUNT (*) INTO contar FROM transformer3.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;
            IF contar > 0 THEN
                UPDATE transformer3.EMPLEADO_TRABAJA SET SALARIO = salario_p WHERE CODIGO_EMPLEADO = codigo;
            ELSE
                SELECT COUNT (*) INTO contar FROM transformer4.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;
                IF contar > 0 THEN
                    UPDATE transformer4.EMPLEADO_TRABAJA SET SALARIO = salario_p WHERE CODIGO_EMPLEADO = codigo;
                END IF ;
            END IF ;
        END IF ;
    END IF ;
END ;
```

Procedimiento para trasladar un empleado:

```
create or replace PROCEDURE
trasladarEmpleado(cod_emp NUMBER , fecha_final_hotel_act DATE , codigo_hotel_nuevo NUMBER ,
fecha_inicio_hotel_nu DATE , direccion VARCHAR , telefono NUMBER ) IS
    contar NUMBER ;
    contar2 NUMBER;
    contarHotel NUMBER;
    contarEmpleado NUMBER;
    nombre_empleado VARCHAR(20);
    dni_empleado NUMBER;
    fecha_inicio_cadena DATE;
    fecha_inicio_hotel DATE;
    salario_empleado FLOAT;
    codigo_hotel_viejo NUMBER;

BEGIN

    IF cod_emp is NULL OR fecha_final_hotel_act is NULL OR codigo_hotel_nuevo is NULL
    OR fecha_inicio_hotel_nu is NULL THEN
        raise_application_error( -20004 , 'LOS UNICOS VALORES QUE SE PERMITEN QUE ESTEN A NULL SON LA DIRECCION
        Y EL TELEFONO, LOS DEMAS DEBEN SER VALORES CORRECTOS' );
    END IF ;
    SELECT COUNT (*) INTO contar FROM EMPLEADO WHERE CODIGO_EMPLEADO = cod_emp;
    IF contar > 0 THEN
        SELECT COUNT (*) INTO contar2 FROM HOTEL WHERE CODIGO = codigo_hotel_nuevo;
        IF contar2 > 0 THEN
            SELECT DNI, NOMBRE, FECHA_INICIO_CADENA, SALARIO, FECHA_INICIO_HOTEL, CODIGO
            INTO dni_empleado, nombre_empleado, fecha_inicio_cadena, salario_empleado, fecha_inicio_hotel, codigo_hotel_viejo
            FROM EMPLEADO WHERE CODIGO_EMPLEADO = cod_emp;
            IF codigo_hotel_viejo = codigo_hotel_nuevo THEN
                raise_application_error(-20004,'NO PUEDE REALIZARSE UN TRASLADO DE UN HOTEL AL MISMO, MODIFIQUE EL HOTEL AL QUE VA HA SER  TRASLADADO');
            END IF ;
            IF fecha_final_hotel_act < fecha_inicio_hotel THEN
                raise_application_error(-20004,'LA FECHA DE FINAL DE CONTRATO ES INFERIOR A LA FECHA DE INICIO DEL CONTRATO');
            END IF ;
            IF fecha_inicio_hotel_nu < fecha_final_hotel_act THEN
                raise_application_error(-20004,'LA FECHA DE INICIO DE CONTRATO DEL NUEVO TRASLADO ES INFERIOR A LA FECHA DE FIN DEL CONTRATO DEL ANTERIOR TRASLADO');
            END IF ;
            SELECT COUNT (*) INTO contarEmpleado FROM transformer1.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = cod_emp;
            IF contarEmpleado > 0 THEN
                DELETE FROM transformer1.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = cod_emp;
                INSERT INTO transformer1.EMPLEADO_CONTRATO VALUES (cod_emp, codigo_hotel_viejo, fecha_inicio_hotel, fecha_final_hotel_act);
            ELSE
                SELECT COUNT (*) INTO contarEmpleado FROM transformer2.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = cod_emp;
                IF contarEmpleado > 0 THEN
                    DELETE FROM transformer2.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = cod_emp;
                    INSERT INTO transformer2.EMPLEADO_CONTRATO VALUES (cod_emp, codigo_hotel_viejo, fecha_inicio_hotel, fecha_final_hotel_act);
                ELSE
                    SELECT COUNT (*) INTO contarEmpleado FROM transformer3.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = cod_emp;
```

```

IF contarEmpleado > 0 THEN
    DELETE FROM transformer4.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = cod_emp;
    INSERT INTO transformer4.EMPLEADO_CONTRATO VALUES (cod_emp, codigo_hotel_viejo, fecha_inicio_hotel, fecha_final_hotel_act);
END IF ;
END IF ;
END IF ;
END IF ;

SELECT COUNT (*) INTO contarHotel FROM HOTEL_DIRIGE WHERE CODIGO = codigo_hotel_nuevo;
IF contarHotel > 0 THEN
    INSERT INTO transformer1.EMPLEADO_TRABAJA VALUES (cod_emp, dni_empleado, nombre_empleado, direccion, telefono,
        fecha_inicio_cadena, salario_empleado, codigo_hotel_nuevo, fecha_inicio_hotel_nu);
ELSE
    SELECT COUNT (*) INTO contarHotel FROM transformer2.HOTEL_DIRIGE WHERE CODIGO = codigo_hotel_nuevo;
    IF contarHotel > 0 THEN
        INSERT INTO transformer2.EMPLEADO_TRABAJA VALUES (cod_emp, dni_empleado, nombre_empleado, direccion, telefono,
            fecha_inicio_cadena, salario_empleado, codigo_hotel_nuevo, fecha_inicio_hotel_nu);
    ELSE
        SELECT COUNT (*) INTO contarHotel FROM transformer3.HOTEL_DIRIGE WHERE CODIGO = codigo_hotel_nuevo;
    IF contarHotel > 0 THEN
        INSERT INTO transformer3.EMPLEADO_TRABAJA VALUES (cod_emp, dni_empleado, nombre_empleado, direccion, telefono,
            fecha_inicio_cadena, salario_empleado, codigo_hotel_nuevo, fecha_inicio_hotel_nu);
    ELSE
        SELECT COUNT (*) INTO contarHotel FROM transformer4.HOTEL_DIRIGE WHERE CODIGO = codigo_hotel_nuevo;
    IF contarHotel > 0 THEN
        INSERT INTO transformer4.EMPLEADO_TRABAJA VALUES (cod_emp, dni_empleado, nombre_empleado, direccion, telefono,
            fecha_inicio_cadena, salario_empleado, codigo_hotel_nuevo, fecha_inicio_hotel_nu);
    END IF ;
    END IF ;
    END IF ;
    END IF ;
ELSE
    raise_application_error(-20004,'TAL HOTEL NO EXISTE');
END IF ;
ELSE
    raise_application_error(-20004,'TAL EMPLEADO NO EXISTE');
END IF ;
END ;

```

- Procedimiento para insertar un hotel:

```
create or replace PROCEDURE
insertarHOTEL(CODIGO_P NUMBER , CIUDAD_P VARCHAR , PROVINCIA_P VARCHAR ,
NOMBRE_P VARCHAR , NUMERO_HABITACIONES_SIMPLE_P NUMBER , NUMERO_HABITACIONES_DOBLE_P NUMBER ) IS
    contar NUMBER ;
BEGIN
    IF CODIGO_P is NULL OR CIUDAD_P is NULL OR PROVINCIA_P is NULL OR NOMBRE_P is NULL
    OR NUMERO_HABITACIONES_SIMPLE_P is NULL OR NUMERO_HABITACIONES_DOBLE_P is NULL THEN
        raise_application_error( -20004 , 'NO PUEDEN INSERTARSE DATOS COMO NULL, TODOS LOS PARAMETROS DEBEN DE SER VALIDOS' );
    END IF ;
    IF PROVINCIA_P = 'Granada' or PROVINCIA_P = 'Jaen' THEN
        INSERT INTO transformer1.HOTEL_DIRIGE VALUES (CODIGO_P, NOMBRE_P, NUMERO_HABITACIONES_DOBLE_P, NUMERO_HABITACIONES_SIMPLE_P, CIUDAD_P, PROVINCIA_P, NULL );
    ELSIF PROVINCIA_P = 'Cadiz' or PROVINCIA_P = 'Huelva' THEN
        INSERT INTO transformer2.HOTEL_DIRIGE VALUES (CODIGO_P, NOMBRE_P, NUMERO_HABITACIONES_DOBLE_P, NUMERO_HABITACIONES_SIMPLE_P, CIUDAD_P, PROVINCIA_P, NULL );
    ELSIF PROVINCIA_P = 'Sevilla' or PROVINCIA_P = 'Cordoba' THEN
        INSERT INTO transformer3.HOTEL_DIRIGE VALUES (CODIGO_P, NOMBRE_P, NUMERO_HABITACIONES_DOBLE_P, NUMERO_HABITACIONES_SIMPLE_P, CIUDAD_P, PROVINCIA_P, NULL );
    ELSIF PROVINCIA_P = 'Malaga' or PROVINCIA_P = 'Almeria' THEN
        INSERT INTO transformer4.HOTEL_DIRIGE VALUES (CODIGO_P, NOMBRE_P, NUMERO_HABITACIONES_DOBLE_P, NUMERO_HABITACIONES_SIMPLE_P, CIUDAD_P, PROVINCIA_P, NULL );
    ELSE
        raise_application_error(-20004,'TAL PROVINCIA NO EXISTE EN LA BASE DE DATOS, POR FAVOR PRUEBE CON Granada, Jaen, Cadiz, Huelva, Sevilla, Cordoba, Malaga o Almeria');
    END IF ;
END ;
```

- Procedimiento para cambiar el director de un hotel

```
create or replace PROCEDURE
cambiarDirectorHotel(codigo NUMBER , codigo_director NUMBER ) IS
    contar NUMBER ;
    contarEmpleadoDirige NUMBER;
BEGIN
    IF codigo is NULL OR codigo_director is NULL THEN
        raise_application_error( -20004 , 'NO PUEDE INSERTAR VALORES A NULL, LOS VALORES DEBEN DE SER VALIDOS' );
    END IF ;
    SELECT COUNT (*) INTO contarEmpleadoDirige FROM HOTEL WHERE director = codigo_director;
    IF contarEmpleadoDirige > 0 THEN
        raise_application_error(-20004, 'TAL EMPLEADO YA DIRIGE UN HOTEL');
    END IF ;
    SELECT COUNT (*) INTO contar FROM EMPLEADO WHERE CODIGO_EMPLEADO = codigo_director;
    IF contar = 0 THEN
        raise_application_error(-20004, 'TAL EMPLEADO NO EXISTE');
    END IF ;
    SELECT COUNT (*) INTO contar FROM transformer1.HOTEL_DIRIGE WHERE CODIGO = codigo;
    IF contar > 0 THEN
        UPDATE transformer1.HOTEL_DIRIGE SET director = codigo_director WHERE CODIGO = codigo;
    ELSE
        SELECT COUNT (*) INTO contar FROM transformer2.HOTEL_DIRIGE WHERE CODIGO = codigo;
        IF contar > 0 THEN
            UPDATE transformer2.HOTEL_DIRIGE SET director = codigo_director WHERE CODIGO = codigo;
        ELSE
            SELECT COUNT (*) INTO contar FROM transformer3.HOTEL_DIRIGE WHERE CODIGO = codigo;
            IF contar > 0 THEN
                UPDATE transformer3.HOTEL_DIRIGE SET director = codigo_director WHERE CODIGO = codigo;
            ELSE
                SELECT COUNT (*) INTO contar FROM transformer4.HOTEL_DIRIGE WHERE CODIGO = codigo;
                IF contar > 0 THEN
                    UPDATE transformer4.HOTEL_DIRIGE SET director = codigo_director WHERE CODIGO = codigo;
                ELSE
                    raise_application_error(-20004, 'TAL HOTEL NO EXISTE, POR FAVOR INTRODUCZA UN HOTEL EXISTENTE EN NUESTRA BASE DE DATOS');
                END IF ;
            END IF ;
        END IF ;
    END IF ;
END ;
```

- Procedimiento para dar de alta a un cliente:

```
create or replace PROCEDURE
altaCliente(CODIGO_CLIENTE NUMBER , DNI NUMBER , NOMBRE VARCHAR , TELEFONO NUMBER ) IS
contar NUMBER ;
BEGIN
  IF CODIGO_CLIENTE is NULL OR DNI is NULL OR NOMBRE is NULL OR TELEFONO is NULL THEN
    raise_application_error( -20004 , 'NO SE PERMITEN VALORES DE ENTRADA A NULL, POR FAVOR INTRODUZCA VALORES VALIDOS' );
  END IF ;
  INSERT INTO transformer1.CLIENTE VALUES (CODIGO_CLIENTE, DNI, NOMBRE, TELEFONO);
  INSERT INTO transformer2.CLIENTE VALUES (CODIGO_CLIENTE, DNI, NOMBRE, TELEFONO);
  INSERT INTO transformer3.CLIENTE VALUES (CODIGO_CLIENTE, DNI, NOMBRE, TELEFONO);
  INSERT INTO transformer4.CLIENTE VALUES (CODIGO_CLIENTE, DNI, NOMBRE, TELEFONO);
END ;
```

- Procedimiento para dar de alta un proveedor:

```
create or replace PROCEDURE
altaProveedor(CODIGO_PROVEEDOR_P NUMBER , NOMBRE VARCHAR , CIUDAD VARCHAR) IS
contarProveedor NUMBER ;
BEGIN
  IF CODIGO_PROVEEDOR_P is NULL OR NOMBRE is NULL OR CIUDAD is NULL THEN
    raise_application_error( -20004 , 'NO SE PERMITEN VALORES DE ENTRADA A NULL, POR FAVOR INTRODUZCA VALORES VALIDOS' );
  END IF ;
  SELECT COUNT (*) INTO contarProveedor FROM PROVEEDOR WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
  IF contarProveedor > 0 THEN
    raise_application_error(-20004,'YA EXISTE EL CODIGO DEL PROVEEDOR QUE SE ESTÁ INTENTANDO INSERTAR');
  END IF ;
  IF CIUDAD = 'Granada' THEN
    INSERT INTO transformer1.PROVEEDOR VALUES (CODIGO_PROVEEDOR_P, NOMBRE, CIUDAD);
  ELSIF CIUDAD = 'Sevilla' THEN
    INSERT INTO transformer3.PROVEEDOR VALUES (CODIGO_PROVEEDOR_P, NOMBRE, CIUDAD);
  ELSE
    raise_application_error(-20004,'EL PROVEEDOR A INSERTAR NO TIENE UNA CIUDAD VALIDA');
  END IF ;
END ;
```

- **Procedimiento para dar de baja un proveedor:**

```
create or replace PROCEDURE
bajaProveedor(CODIGO_PROVEEDOR_P NUMBER ) IS
contar NUMBER ;
num_suministro NUMBER;
cantidad_articulo_suministrado number;
BEGIN
  IF CODIGO_PROVEEDOR_P is NULL THEN
    raise_application_error( -20004 , 'NO SE PERMITEN VALORES A NULL, DEBES INTRODUCIR UN VALOR VALIDO' );
  END IF ;
  SELECT COUNT (*) INTO contar FROM transformer1.PROVEEDOR WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
  IF contar > 0 THEN
    SELECT COUNT (*) INTO num_suministro FROM SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
    if num_suministro > 0 then
      SELECT COUNT (*) into cantidad_articulo_suministrado FROM SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P AND CANTIDAD > 0 ;
      if cantidad_articulo_suministrado > 0 then
        raise_application_error(-20004,'EL PROVEEDOR NO SE PUEDE BORRAR YA QUE TIENE CANTIDAD DE ARTICULO SUMINISTRADA MAYOR QUE CERO');
      END IF ;
    END IF ;
  END IF ;

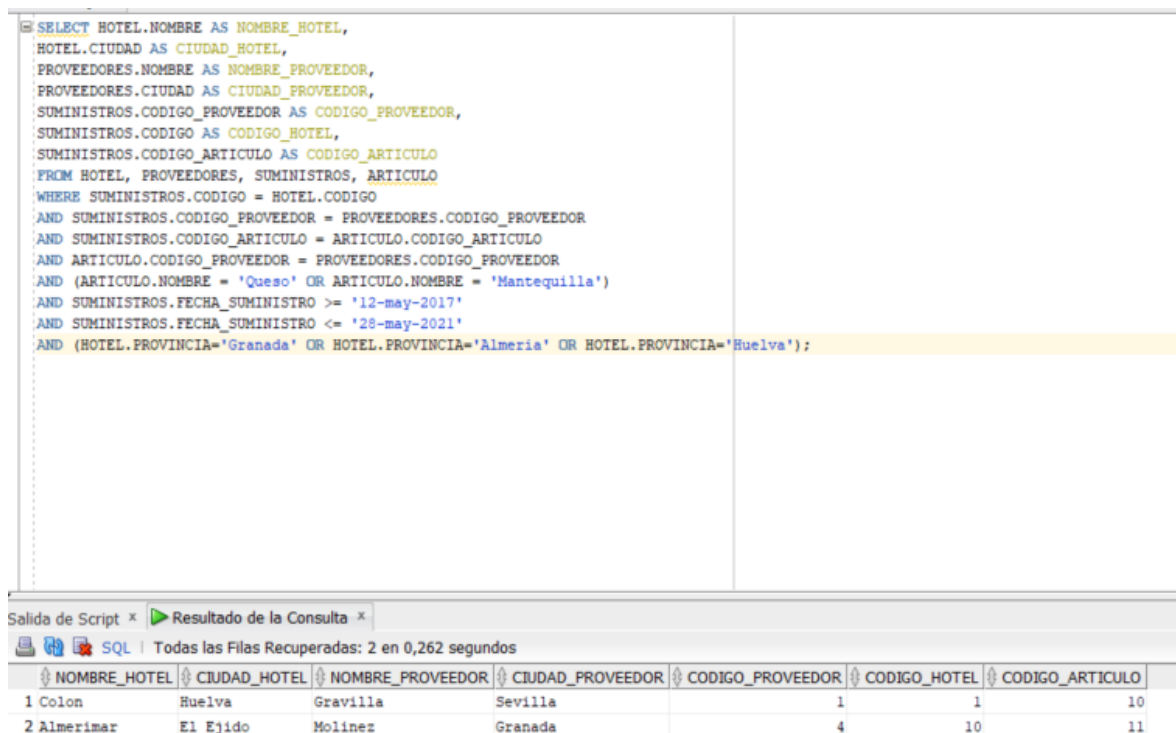
  DELETE FROM transformer1.SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
  DELETE FROM transformer2.SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
  DELETE FROM transformer3.SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
  DELETE FROM transformer4.SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
  DELETE FROM transformer1.ARTICULO_VENDE WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
  DELETE FROM transformer1.PROVEEDOR WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
  ELSE
    SELECT COUNT (*) INTO contar FROM transformer3.PROVEEDOR WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
    IF contar > 0 THEN
      SELECT COUNT (*) INTO num_suministro FROM SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
      if num_suministro > 0 then
        SELECT COUNT (*) into cantidad_articulo_suministrado FROM SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P AND CANTIDAD > 0 ;
        if cantidad_articulo_suministrado > 0 then
          raise_application_error(-20004,'EL PROVEEDOR NO SE PUEDE BORRAR YA QUE TIENE CANTIDAD DE ARTICULO SUMINISTRADA MAYOR QUE CERO');
        END IF ;
      END IF ;
    END IF ;

    DELETE FROM transformer1.SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
    DELETE FROM transformer2.SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
    DELETE FROM transformer3.SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
    DELETE FROM transformer4.SUMINISTRO WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
    DELETE FROM transformer3.ARTICULO_VENDE WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
    DELETE FROM transformer3.PROVEEDOR WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
    ELSE
      raise_application_error(-20004,'EL PROVEEDOR A ELIMINAR NO EXISTE');
    END IF ;
  END IF ;
END ;
```

Práctica 5:

Consultas

- **Consulta 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 2021 y el 28 de mayo de 2021.



```
SELECT HOTEL.NOMBRE AS NOMBRE_HOTEL,
HOTEL.CIUDAD AS CIUDAD_HOTEL,
PROVEEDORES.NOMBRE AS NOMBRE_PROVEEDOR,
PROVEEDORES.CIUDAD AS CIUDAD_PROVEEDOR,
SUMINISTROS.CODIGO_PROVEEDOR AS CODIGO_PROVEEDOR,
SUMINISTROS.CODIGO AS CODIGO_HOTEL,
SUMINISTROS.CODIGO_ARTICULO AS CODIGO_ARTICULO
FROM HOTEL, PROVEEDORES, SUMINISTROS, ARTICULO
WHERE SUMINISTROS.CODIGO = HOTEL.CODIGO
AND SUMINISTROS.CODIGO_PROVEEDOR = PROVEEDORES.CODIGO_PROVEEDOR
AND SUMINISTROS.CODIGO_ARTICULO = ARTICULO.CODIGO_ARTICULO
AND ARTICULO.CODIGO_PROVEEDOR = PROVEEDORES.CODIGO_PROVEEDOR
AND (ARTICULO.NOMBRE = 'Queso' OR ARTICULO.NOMBRE = 'Mantequilla')
AND SUMINISTROS.FECHA_SUMINISTRO >= '12-may-2021'
AND SUMINISTROS.FECHA_SUMINISTRO <= '28-may-2021'
AND (HOTEL.PROVINCIA='Granada' OR HOTEL.PROVINCIA='Almeria' OR HOTEL.PROVINCIA='Huelva');
```

NOMBRE_HOTEL	CIUDAD_HOTEL	NOMBRE_PROVEEDOR	CIUDAD_PROVEEDOR	CODIGO_PROVEEDOR	CODIGO_HOTEL	CODIGO_ARTICULO
1 Colon	Huelva	Gravilla	Sevilla	1	1	10
2 Almerimar	El Ejido	Molinez	Granada	4	10	11

- **Código**

```
SELECT HOTEL.NOMBRE AS NOMBRE_HOTEL,
HOTEL.CIUDAD AS CIUDAD_HOTEL,
PROVEEDORES.NOMBRE AS NOMBRE_PROVEEDOR,
PROVEEDORES.CIUDAD AS CIUDAD_PROVEEDOR,
SUMINISTROS.CODIGO_PROVEEDOR AS CODIGO_PROVEEDOR,
SUMINISTROS.CODIGO AS CODIGO_HOTEL,
SUMINISTROS.CODIGO_ARTICULO AS CODIGO_ARTICULO
FROM HOTEL, PROVEEDORES, SUMINISTROS, ARTICULO
WHERE SUMINISTROS.CODIGO = HOTEL.CODIGO
AND SUMINISTROS.CODIGO_PROVEEDOR = PROVEEDORES.CODIGO_PROVEEDOR
AND SUMINISTROS.CODIGO_ARTICULO = ARTICULO.CODIGO_ARTICULO
AND ARTICULO.CODIGO_PROVEEDOR = PROVEEDORES.CODIGO_PROVEEDOR
AND (ARTICULO.NOMBRE = 'Queso' OR ARTICULO.NOMBRE = 'Mantequilla')
AND SUMINISTROS.FECHA_SUMINISTRO >= '12-may-2021'
AND SUMINISTROS.FECHA_SUMINISTRO <= '28-may-2021'
AND (HOTEL.PROVINCIA='Granada' OR HOTEL.PROVINCIA='Almeria' OR HOTEL.PROVINCIA='Huelva');
```


- **Consulta 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 ARTICULO.NOMBRE AS NOMBRE_ARTICULO,
HOTEL.NOMBRE AS NOMBRE_HOTEL,
HOTEL.CIUDAD AS CIUDAD_HOTEL,
SUMINISTROS.CANTIDAD AS CANTIDAD
FROM HOTEL, SUMINISTROS, ARTICULO
WHERE SUMINISTROS.CODIGO = HOTEL.CODIGO
AND SUMINISTROS.CODIGO_PROVEEDOR = 3
AND SUMINISTROS.CODIGO_ARTICULO = ARTICULO.CODIGO_ARTICULO
AND ARTICULO.CODIGO_PROVEEDOR = 3
AND ( HOTEL.PROVINCIA = 'Jaen' OR HOTEL.PROVINCIA = 'Almeria');

```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 5 en 0,191 segundos

	NOMBRE_ARTICULO	NOMBRE_HOTEL	CIUDAD_HOTEL	CANTIDAD
1	Pollo	Alcaza	Almeria	300
2	Pavo	Alcaza	Almeria	100
3	Cordero	Santo Reino	Jaen	150
4	Pescadilla	Santo Reino	Jaen	150
5	Pescadilla	Santo Reino	Jaen	100

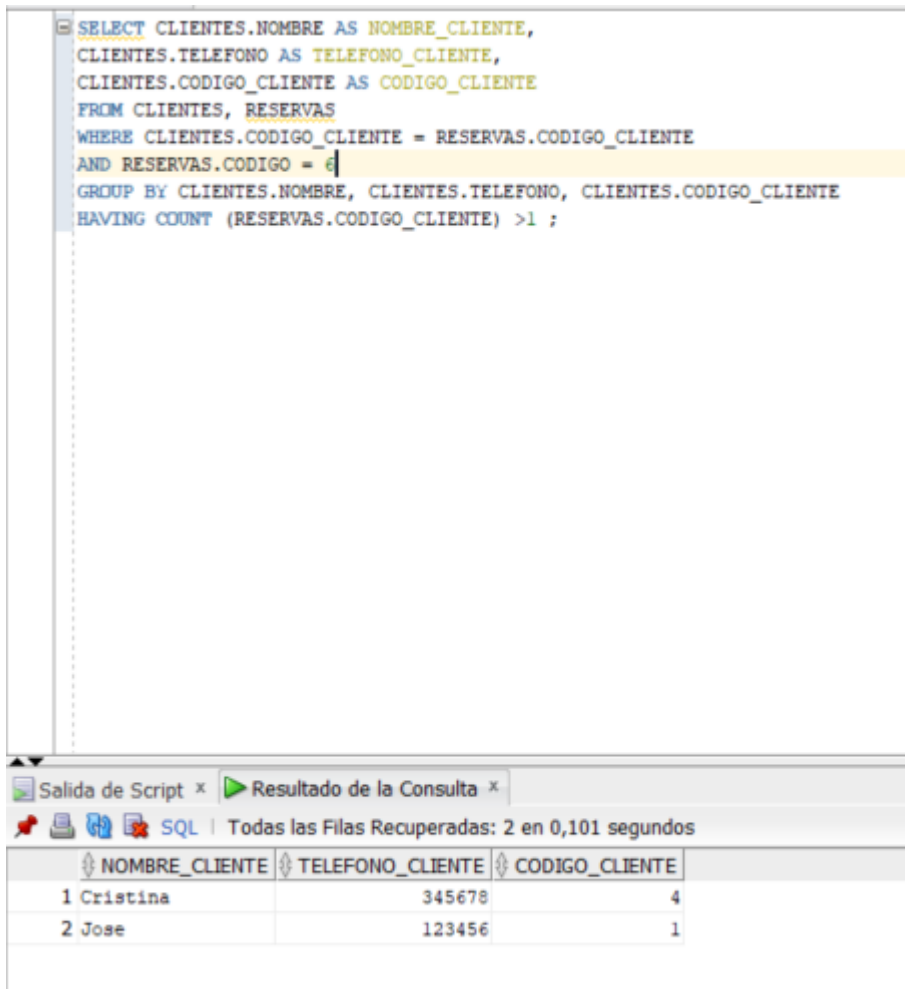
- **Código:**

```

SELECT ARTICULO.NOMBRE AS NOMBRE_ARTICULO,
HOTEL.NOMBRE AS NOMBRE_HOTEL,
HOTEL.CIUDAD AS CIUDAD_HOTEL,
SUMINISTROS.CANTIDAD AS CANTIDAD
FROM HOTEL, SUMINISTROS, ARTICULO
WHERE SUMINISTROS.CODIGO = HOTEL.CODIGO
AND SUMINISTROS.CODIGO_PROVEEDOR = 3
AND SUMINISTROS.CODIGO_ARTICULO = ARTICULO.CODIGO_ARTICULO
AND ARTICULO.CODIGO_PROVEEDOR = 3
AND ( HOTEL.PROVINCIA = 'Jaen' OR HOTEL.PROVINCIA = 'Almeria');

```

- **Consulta 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 CLIENTES.NOMBRE AS NOMBRE_CLIENTE,
CLIENTES.TELEFONO AS TELEFONO_CLIENTE,
CLIENTES.CODIGO_CLIENTE AS CODIGO_CLIENTE
FROM CLIENTES, RESERVAS
WHERE CLIENTES.CODIGO_CLIENTE = RESERVAS.CODIGO_CLIENTE
AND RESERVAS.CODIGO = 6
GROUP BY CLIENTES.NOMBRE, CLIENTES.TELEFONO, CLIENTES.CODIGO_CLIENTE
HAVING COUNT (RESERVAS.CODIGO_CLIENTE) >1 ;

```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 2 en 0,101 segundos

	NOMBRE_CLIENTE	TELEFONO_CLIENTE	CODIGO_CLIENTE
1	Cristina	345678	4
2	Jose	123456	1

- **Código:**

```

SELECT CLIENTES.NOMBRE AS NOMBRE_CLIENTE,
CLIENTES.TELEFONO AS TELEFONO_CLIENTE,
CLIENTES.CODIGO_CLIENTE AS CODIGO_CLIENTE
FROM CLIENTES, RESERVAS
WHERE CLIENTES.CODIGO_CLIENTE = RESERVAS.CODIGO_CLIENTE
AND RESERVAS.CODIGO = 6
GROUP BY CLIENTES.NOMBRE, CLIENTES.TELEFONO, CLIENTES.CODIGO_CLIENTE
HAVING COUNT (RESERVAS.CODIGO_CLIENTE) > 1 ;

```