Base de Datos Distribuidas

Escuela Técnica Superior de Ingeniería Informática y Teleco.
Universidad de Granada
Granada, España

2017-2018

UNIVERSIDAD DE GRANADA

Autores:
Iván Rodriguez Millán
Rshad Zhran

Tabla de Contenido

1. Introducción

En este trabajo se pretende aprender cómo crear una base de datos distribuida.

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 suministro 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 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. Pasos a seguir

Podemos decir que empezamos desde `0` a crearla, ya que hemos seguido los siguientes pasos:

1. Diseño del modelo E/R.
2. Diseño del modelo Lógico.
3. Diseño de las Fragmentaciones y la Asignación.
4. Implementación del Diseño.
5. Implementación de Triggers y Procedimientos.
6. Implementación de Actualizaciones.
7. Implementación de Consultas.

3. Diseño del modelo E/R

3.1 Explicación del Modelo E/R

En en esta sección, vamos a explicar las alternativas que tuvimos al diseñar el modelo, las entidades creadas, las relaciones, cardinalidades y participación.

3.1.1 Relación1

Proveedor ----- Vende ------- Articulo

Cardinalidad

En la relación binaria Vende, que tiene como entidades participantes Proveedor y Articulo. Esta relación describe la acción de venta realizada de un Proveedor a un Articulo, con cardinalidad **Proveedor {2} - {m} Articulo** :

- Una fila de la entidad **Proveedor** puede relacionarse con varias de la entidad **Articulo**, mientras una fila de la entidad **Articulo**, puede relacionarse relacionarse como máximo con 2 filas de la entidad **Proveedor**.

  ¿ Por qué sólo 2 de Proveedor ?
  Sabemos que si la cardinalidad es mayor que 1, entonces puede expresarse como **'mucho',** Pero nos dimos cuenta de que si tenemos la cardinalidad entre **Proveedor** y **Articulo,** entonces no podremos fusionar la tabla **Articulo** con **Vende**, ya que así **Vende** tendrá Llave diferente (En la que aparece la llave de Proveedor) al de **Articulo**, y en este caso no se pueden fusionar.

Participación

En esta relación tenemos las participaciones de la siguiente forma:

Proveedor {P:1} ------- Vende -------- {P:1} Articulo

En este caso, como es una relación binaria, entonces puede haber 2 tipos de participaciones:

- Total
- Parcial

En este caso, tenemos participaciones diferentes en los 2 lados de la relación. Participación **Total** en el lado de **Proveedor**, es decir, un articulo **tiene que** estar vendido por uno ó más proveedores, y **Parcial** en el lado de **Articulo**, es decir, un proveedor **puede** vender a uno ó más artículos ó a ninguno.

3.1.2 Relación2

Agregación(Proveedor --- Vende ---- Articulo) ---- Suministro ---- Hotel

Una vez que tenemos la relación **Proveedor --- Vende ---- Articulo** que describe el proceso de de venta de articulo/s por un/os proveedor/es, podemos empezar a diseñar la relación **Suministro.**

Suministro describe una relación entre un **artículo** tal, vendido por un **proveedor** tal que se suministra por un hotel.

¿ Por qué elegimos este diseño ?
En nuestro problema, los sinónimos {Suministra,Suministrado,Suministro}, siempre están relacionados con las entidades **{Hotel,Proveedor,Articulo}**. Una de las posibilidades puede ser una relación ternaria con estas 3 entidades. Pero en nuestro problema, una de las restricciones es sólo tener relaciones binarias. Entonces para solucionar este problema, aprovechamos la relación **Proveedor --- Vende ---- Articulo** , ya que no deja de formar una parte de un suministro, y podemos considerarla como una entidad, utilizando la técnica de **Agregación**.

Cardinalidad
Vemos que las cardinalidades son de la siguiente forma:

- **Agregación(Proveedor --- Vende ---- Articulo) {n}**
- **Hotel {m}**

Es una relación con cardinalidad de mucho a mucho, donde un hotel puede suministrar varios artículos y un artículo puede ser suministrado por varios hoteles.

Participación:
En esta relación tenemos la participación de la siguiente forma:

- **Agregación(Proveedor --- Vende ---- Articulo) {P:0}**
- **Hotel {P:0}**

Entonces es una relación en la que tenemos participaciones **parciales** en los dos lados de la relación. Es decir un Hotel **puede** suministrar a un articulo tal, vendido por un/os proveedor/es y un articulo vendido por un/os proveedor/es **puede** ser suministrado por un/os hotel/es.

3.1.3 Relación3

Empleado ----- Dirige ------ Hotel

Es una relación binaria, que describe una relación de que un empleado trabaja en un hotel pero para un puesto determinado de trabajar y es ser Director.

Cardinalidad

Las cardinalidades en esta relación son de la siguiente forma ..

Empleado {1} ----- Dirige ----- Hotel {1}

Entonces es una relación 1:1 en la que un hotel sólo puede estar dirigido por un sólo empleado y un empleado sólo puede dirigir a un único hotel.

Participación

Las participaciones en esta relación son de la siguiente forma …

Empleado {P:1} ----- Dirige ----- Hotel {P:0}

Entonces aquí tenemos participaciones de diferentes tipos, total y parcial. Total en el lado de Empleado y Parcial en el lado de Hotel. Es decir, un Empleado **puede** dirigir a un hotel, pero un Hotel **tiene** que estar dirigido por un Empleado.

3.1.4 Relación4

Empleado ----- Trabaja ----- Hotel

Es una relación binaria en la la que se describe el concepto de que un empleado trabaja para un hotel.

Cardinalidad

En esta relación, las cardinalidades son de la siguiente forma ..

Empleado {n} ---- Trabaja ---- Hotel {1}

Entonces es mucho a 1, es decir un empleado, sólo puede trabajar en un único hotel en una fecha determinada, mientras en un hotel, puede trabajar varios empleados en una fecha determinada.

Participación

Las participaciones en esta relación son de la siguiente forma …

Empleado {P:0} ----- Trabaja ----- Hotel {P:1}

Entonces es participación parcial por parte del Empleado y Total por parte del hotel. Es decir, un empleado **tiene** que trabajar en un hotel, mientras en un hotel **puede** trabajar un/os empleado/s, en una fecha determinada.

3.1.5 Relación5

Cliente ---- Reserva ---- Hotel

Esta relación binaria, describe el proceso de realizar una reserva por un cliente en un hotel.

Cardinalidad
En esta relación, las cardinalidades son de la siguiente forma ..

Cliente{m} ----- Reserva ------ Hotel{n}

Es decir, mucho a mucho. Un cliente puede reservar en varios hoteles y en un hotel puede reservar varios clientes.

Participación
Las participaciones en esta relación son de la siguiente forma ...

Cliente{P:0} ----- Reserva ------ Hotel{P:0}

Entonces es participación parcial en los 2 lados de la relación. Es decir, un cliente **puede** reservar en un/os hotel/es, y en un hotel **pueden** reservar un/os clientes.

4. Diseño del modelo Lógico

Una vez diseñado el modelo E/R, tenemos que obtener el modelo lógico de la base de datos, es decir, hacer el paso a tablas, para el modelo de datos relacional. A continuación procedemos con los siguientes pasos para realizar esta tarea:

1. Paso a Tablas.
2. Fusión de Tablas.

4.1 Descripción entidades y relaciones

Cómo vemos, en nuestro modelo E/R, tenemos las siguiente tablas y relaciones:

4.1.1 Entidades:

- **Empleado**: Para almacenar los datos de los empleados que trabajan en la empresa hotelera.

- **Hotel**: Para almacenar los datos de los distintos hoteles miembro de la empresa hotelera.

- **Cliente**: Para almacenar los datos de los distintos clientes que realicen reservas en los distintos hoteles de la cadena hotelera.

- **Proveedor**: Para almacenar los datos de los distintos proveedores que distribuyen los distintos artículos a los diferentes hoteles de la cadena hotelera.

- **Artículo**: Para almacenar los datos de los distintos artículos que puedan distribuir los distintos proveedores a los diferentes hoteles de la cadena hotelera.

- **Empleado-Contrato**: Para almacenar un histórico de los empleados que han trabajado en la cadena hotelera.

4.1.2 Relaciones:

- **Trabaja**: Para almacenar un historial de cada empleado con respecto a cuando ha trabajado en un hotel, junto con la fechas de inicio y final del contrato.

- **Dirige**: Para almacenar los datos referentes a los empleados que dirigen los diferentes hoteles de la cadena hotelera . Simplemente hará falta el código del empleado que dirige el hotel y el código del hotel que es dirigido.

- **Reserva**: Para almacenar los datos de las distintas reservas que hagan los clientes en los diferentes hoteles de la cadena hotelera. Con información como fecha de inicio de la reserva y de final, así como el tipo de habitación y el precio.

- **Vende**: Para almacenar las ventas que realizan los proveedores y que artículos pertenecen a cada venta.

- **Suministro**: Para almacenar los distintos suministros que hagan los proveedores a los distintos hoteles con los artículos que haya de por medio. También se deben de almacenar datos importantes como la fecha del suministro, la cantidad del artículo que se suministra y el precio por unidad del artículo.

4.1.3 Descripción de las tablas y sus atributos

NOTA: Los nombres de los atributos que aparecen a continuación no son exactos como aparecen en el modelo. Por ejemplo, sin barra baja.

EMPLEADO:

- **Código del empleado**, que nos servirá como identificativo en todo el sistema para cada empleado. Será un dato de tipo numérico,
- **DNI** del empleado que será un dato de tipo numérico de 8 dígitos.
- **Dirección** del empleado que será un dato de tipo String.
- **Fecha de inicio de contrato del empleado con la multinacional** que será un dato de tipo Date. (Esta fecha no es la del inicio de contrato en el hotel, si no la de inicio de contrato con la cadena hotelera que es diferente).
- **Salario** del empleado que será un dato de tipo numérico en coma flotante (float o double).
- **Nombre** del empleado que será un dato de tipo String.
- **Teléfono** del empleado que será un dato de tipo numérico.

EMPLEADO-CONTRATO:

- **Código del empleado**, que nos servirá como identificativo en todo el sistema para cada empleado. Será un dato de tipo numérico,
- **Código del hotel**, que nos servirá como identificativo en todo el sistema para cada hotel. Será un dato de tipo numérico.
- **Fecha de inicio de contrato del empleado en un hotel** que será un dato de tipo Date. Esta fecha es la de inicio del contrato de un empleado en un hotel en particular, para nada tiene que ver con la fecha de inicio de contrato con la multinacional. Esta fecha será diferente cada vez que el empleado trabaje en un hotel distinto.
- **Fecha de final de contrato del empleado en un hotel** que será un dato de tipo Date. Esta fecha es la de final del contrato de un empleado en un hotel en particular, para nada tiene que ver con la fecha de inicio de contrato con la multinacional. Esta fecha será diferente cada vez que el empleado termine de trabajar en un hotel, obviamente no puede ser menor que la fecha de inicio de contrato o que la fecha de inicio de contrato con la cadena hotelera.

HOTEL:

- **Código del hotel**, que nos servirá como identificativo en todo el sistema para cada hotel. Será un dato de tipo numérico.
- **Ciudad** en donde está el hotel. Será un dato de tipo String.
- **Provincia** en donde está el hotel. Será un dato de tipo String.
- **Nombre** del hotel que será un dato de tipo String.
- **Número de habitaciones simples** que será un dato de tipo numérico.
- **Número de habitaciones dobles** que será un dato de tipo numérico.

CLIENTE:

- **Código del cliente**, que nos servirá como identificativo en todo el sistema para cada cliente. Será un dato de tipo numérico.
- **Nombre** del cliente que será un dato de tipo String.
- **DNI** del cliente que será un dato de tipo numérico con máximo 8 dígitos.
- **Teléfono** del cliente que será un dato de tipo numérico.

PROVEEDOR:

- **Código del proveedor**, que nos servirá como identificativo en todo el sistema para cada proveedor. Será un dato de tipo numérico.
- **Nombre** del proveedor que será un dato de tipo String.
- **Ciudad** del proveedor que será un dato de tipo String.
- **Provincia** del proveedor que será un dato de tipo String.

ARTÍCULO:

- **Código del artículo**, que nos servirá como identificativo en todo el sistema para cada artículo. Será un dato de tipo numérico.
- **Nombre** del artículo que será un dato de tipo String.
- **Tipo** del artículo que será un dato de tipo String.

TRABAJA:

- **Código del empleado** perteneciente al empleado que trabaja en un hotel en un momento marcado por una fecha en concreto.
- **Código del hotel** en el que trabaja el empleado en un momento dado.
- **Fecha de inicio de contrato del empleado** en un momento dado para estar en un hotel determinado.
- **Fecha de fin de contrato del empleado** en un momento dado para estar en un hotel determinado.

DIRIGE:

- **Código del empleado** que hace la función de director de un hotel en particular.
- **Código del hotel** que dirige un empleado en un momento dado.

RESERVA:

- **Código del cliente** que realiza la reserva en un hotel en un momento dado.
- **Fecha de inicio de la reserva** realizada por un cliente para un hotel.
- **Fecha fin de la reserva** realizada por un cliente para un hotel.
- **Código del hotel** al que un cliente le realiza la reserva.

VENDE:

- ❏ **Código del proveedor** que provee un artículo a un hotel.
- ❏ **Código del artículo** que es provisto por un proveedor a un hotel.

SUMINISTRO:

- ❏ **Código del hotel** al que es suministrado una cantidad de artículos determinada por un proveedor en un momento dado.
- ❏ **Código del proveedor** que suministra una cantidad de artículos a un determinado hotel en un momento dado.
- ❏ **Código del artículo** que es suministrado por un proveedor a un hotel en un momento dado.
- ❏ **Fecha de suministro** que será un dato de tipo Date. Corresponde a la fecha en la que es realizado el suministro.
- ❏ **Cantidad** de artículos que se suministran a un determinado hotel. Será un dato de tipo numérico.
- ❏ **Precio por unidad** de un artículo que es suministrado por un proveedor. Será un dato de tipo numérico.

4.2 Paso a Tablas

Para la tabla Empleado y la tabla Trabaja realizamos una propagación y se nos quedaría de la siguiente forma:

En donde la clave primaria (PK) es el código del empleado, la fecha de inicio del contrato con el hotel correspondiente y la fecha de final de contrato. En cada momento un empleado solo podrá estar trabajando en un único hotel, pero podrá darse en forma de historial que un empleado salga más de una vez por que haya trabajado en distintas fechas para otros hoteles.

Empleado-Trabaja
[**código_empleado,fecha_inicio_contrato,fecha_final_contrato**,dni_empleado,salario ,nombre_empleado, telefono_empleado,código_hotel]

Para las tablas Hotel y Dirige realizamos igualmente una propagación juntándolas. De tal forma que nos quedamos con Hotel-Dirige y como clave primaria tendría el codigo del hotel ya que ambas tablas comparten la clave primaria. En cada momento únicamente puede haber un empleado dirigiendo un hotel.

Hotel-dirige
[**código_hotel**,ciudad, provincia, nombre_hotel, número_habitaciones_simples, número_habitaciones_dobles, código_empleado_dirige]

Con la entidad Cliente hacemos una tabla que contendrá a todos los clientes dados de alta en hoteles de nuestra cadena. Como clave primaria tendremos el codigo del cliente.

Cliente
[**código_cliente**,nombre,dni,telefono]

Con la relación Reserva hacemos una tabla en donde recogemos los distintos datos de los clientes y las reservas hechas por ellos, asociados al hotel donde se ha realizado. Como clave primaria tendremos el código del cliente, la fecha de inicio de la reserva, la fecha de final de reserva y el código del hotel, nos servirán para discriminar entre reservas del mismo cliente.

Reserva
[**código_cliente**,**fecha_inicio_reserva**,**fecha_final_reserva, codigo_hotel** ,tipo_habitacion, precio]

Para la entidad proveedor necesitaremos reflejar los datos del mismo. De tal forma que tenemos como identificador de la tabla el código del proveedor para discriminar a los proveedores en todo el sistema.

Proveedor
[**codigo_proveedor**, nombre_proveedor ,ciudad, provincia]

Para la relación Vende y la entidad Articulo se hará propagación, en donde se reflejan las ventas realizadas por los proveedores y el código del artículo que se ha provisto tendremos una tabla en donde la clave primaria estará formada por el código del proveedor y el código del artículo.

Articulo-Vende
[**codigo_proveedor**,**codigo_articulo,** nombre_articulo ,tipo]

Para la relación **suministro** entre la agregación de artículo y proveedor y la entidad hotel, tendremos una tabla en donde se recogerán los datos de los diferentes suministros realizados por un proveedor a un hotel con un artículo de por medio. De tal forma que la clave primaria de esta tabla será código del hotel, código del proveedor y el código del artículo.

Suministro
[**codigo_hotel,codigo_proveedor, codigo_articulo,** fecha_suministro, cantidad_articulo, precio_por_unidad_articulo]

En el caso de la entidad **Empleado-Contrato** tendremos una tabla que recogerá los datos de los contratos de los diferentes empleados que han ido pasando por la cadena hotelera. De tal forma, que la clave primaria de esta tabla estará compuesta por el código del empleado, código del hotel, fecha de inicio del contrato y fecha final del contrato.

Empleado-Contrato
[codigo_hotel,codigo_empleado, fecha_inicio_contrato,fecha_final_contrato]

5. Diseño de la fragmentación y de la asignación

En primer lugar se va ha proceder con el diseño de la fragmentación. Este proceso abarcará desde determinar qué tablas o relaciones deben fragmentarse y con que tipo hasta poder realizar el diseño distribuido.

Para la fragmentación debe de tenerse en cuenta datos como por ejemplo como referencian las aplicaciones a los datos, por ejemplo en nuestro caso tenemos que las aplicaciones que utilizan datos de hoteles (incluidos datos de empleados, reservas y suministros) se generan en cualquier localidad, pero referencian con una probabilidad del 95 % a hoteles cercanos. Luego está claro que todas estas tablas deberán fragmentarse de acuerdo a la cercanía del hotel.

En primer lugar por tanto miramos a la entidad **Hotel-Dirige** que parece una firme candidata a ser fragmentada, y puesto que lo que nos están pidiendo es que fragmentemos de tal forma que al referenciar en una localidad se mostrarán al 95% los hoteles más cercanos, lo que debemos hacer es fragmentar horizontalmente quedándonos con los hoteles cercanos a cada localidad.

Por tanto la fragmentación sería la siguiente:

Conjunto de predicados simple

P = {Provincia=Jaén (P1), Provincia=Granada (P2), Provincia=Sevilla (P3), Provincia=Córdoba (P4), Provincia=Cádiz (P5), Provincia=Huelva (P6), Provincia=Málaga (P7), Provincia=Almería (P8)}.

Términos de predicado

Y1 = P1 AND P2 AND P3 AND P4 AND P5 AND P6 AND P7 AND P8;

Y2 = P1 AND P2 AND P3 AND P4 AND P5 AND P6 AND P7 AND NOT P8;
Y1 = P1 AND P2 AND P3 AND P4 AND P5 AND P6 AND NOT P7 AND P8;

Y1 = P1 AND P2 AND P3 AND P4 AND P5 AND NOT P6 AND P7 AND P8;

.

.

.

.

.

.

Nosotros nos quedaremos con las siguientes:

Yx = P1 AND NOT P2 AND NOT P3 AND NOT P4 AND NOT P5 AND NOT P6 AND NOT P7 AND NOT P8;

Yx+1 = NOT P1 AND P2 AND NOT P3 AND NOT P4 AND NOT P5 AND NOT P6 AND NOT P7 AND NOT P8;

Yx+2 = NOT P1 AND NOT P2 AND P3 AND NOT P4 AND NOT P5 AND NOT P6 AND NOT P7 AND NOT P8;

Yx+3 = NOT P1 AND NOT P2 AND NOT P3 AND P4 AND NOT P5 AND NOT P6 AND NOT P7 AND NOT P8;

Yx+4 = NOT P1 AND NOT P2 AND NOT P3 AND NOT P4 AND P5 AND NOT P6 AND NOT P7 AND NOT P8;

Yx+5 = NOT P1 AND NOT P2 AND NOT P3 AND NOT P4 AND NOT P5 AND P6 AND NOT P7 AND NOT P8;

Yx+6 = NOT P1 AND NOT P2 AND NOT P3 AND NOT P4 AND NOT P5 AND NOT P6 AND P7 AND NOT P8;

Yx+7 = NOT P1 AND NOT P2 AND NOT P3 AND NOT P4 AND NOT P5 AND NOT P6 AND NOT P7 AND P8;

En donde Yx pertenece a Jaén, Yx+1 a Granada, Yx+2 a Sevilla, Yx+3 a Córdoba, Yx+4 a Cádiz, Yx+5 a Huelva, Yx+6 a Málaga e Yx+7 a Almería.

De tal forma que se distribuyen de la siguiente forma:

Localidad	**Provincia1**	**Provincia2**
Granada	Granada	Jaén
Sevilla	Sevilla	Córdoba
Cádiz	Cádiz	Huelva
Málaga	Málaga	Almería

En dónde “Localidad” se refiere a la localidad en donde estarán almacenados los diferentes fragmentos, y por “Provincia1” y “Provincia2” se entienden que los fragmentos que contengan dichas provincias irán almacenados a dicha localidad.
De tal forma que los fragmentos con la provincia de Granada ó Jaén irán a la localidad de Granada.
Los fragmentos con la provincia de Sevilla ó Córdoba irán a la localidad de Sevilla.
Y así sucesivamente.

Así tendríamos las siguientes consultas:

Hotel1 = SLprovincia=Jaén (HOTEL);

Hotel2 = SLprovincia=Granda (HOTEL);

Hotel3 = SLprovincia=Sevilla (HOTEL);

Hotel4 = SLprovincia=Córdoba (HOTEL);

Hotel5 = SLprovincia=Cádiz (HOTEL);

Hotel6 = SLprovincia=Huelva (HOTEL);

Hotel7 = SLprovincia=Málaga (HOTEL);

Hotel8 = SLprovincia=Almería (HOTEL);

De igual forma tendríamos fragmentada la tabla **Proveedor** por el atributo provincia, pero con un pequeño matiz, y es que en este caso solamente existen proveedores en las provincias de Granada y Sevilla de manera que un proveedor de Granada suministra a los hoteles de Jaén, Granada, Málaga y Almería, y un proveedor de Sevilla suministra a los hoteles de Sevilla, Córdoba, Cádiz y Huelva.

Por tanto y de acuerdo a las aplicaciones y lo matizado anteriormente debemos de fragmentar horizontalmente la tabla de proveedores y asignarlas a las localidades de Granada o Sevilla dependiendo su provincia de procedencia.

Conjunto de predicados simples

P = {Provincia=Granada (P1), Provincia=Sevilla (P2)};

Términos de predicado

Y1 = P1 AND P2

Y4 = NOT P1 AND NOT P2

En nuestro caso nos quedamos con los siguientes términos de predicado:

Y2 = P1 AND NOT P2

Y3 = NOT P1 AND P2

Localidad	**Provincia**
Granada	Granada
Sevilla	Sevilla

En donde 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.

Para la tabla **Suministro** se va a proceder con una fragmentación derivada, esto es así porque queremos que cada suministro vaya a parar a la misma localidad en la que se encuentra el hotel al que corresponda el suministro. Así los suministros irán a parar a las cuatro localidades existentes, y en caso de los hoteles que estén alojados en localidades cercanas a sus provincias como por ejemplo los hoteles de la provincia de Jaén que están alojados en la localidad de Granada, para sus suministros también estarán almacenados en la localidad del hotel correspondiente.

Para que todo quede mucho más claro se procede a mostrar un esquema de como quedarían encuadrados los suministros:

En este primer ejemplo tenemos los suministros correspondientes a los hoteles de Jaén. Este suministro iría a la misma localidad donde estuviesen los hoteles de Jaén.

- Suministro1 = Suministro **SJN cod_hotel** (Hotel1)

En este segundo ejemplo tenemos los suministros correspondientes a los hoteles de Granada. Este suministro iría a la misma localidad donde estuviesen los hoteles de Granada.

- Suministro2 = Suministro **SJN cod_hotel** (Hotel2)

En este tercer ejemplo tenemos los suministros correspondientes a los hoteles de Sevilla. Este suministro iría a la misma localidad donde estuviesen los hoteles de Sevilla.

- Suministro3 = Suministro **SJN cod_hotel** (Hotel3)

En este cuarto ejemplo tenemos los suministros correspondientes a los hoteles de Córdoba. Este suministro iría a la misma localidad donde estuviesen los hoteles de Córdoba.

- Suministro4 = Suministro **SJN cod_hotel** (Hotel4)

En este quinto ejemplo tenemos los suministros correspondientes a los hoteles de Cádiz. Este suministro iría a la misma localidad donde estuviesen los hoteles de Cádiz.

- Suministro5 = Suministro **SJN cod_hotel** (Hotel5)

En este sexto ejemplo tenemos los suministros correspondientes a los hoteles de Huelva Este suministro iría a la misma localidad donde estuviesen los hoteles de Huelva.

- Suministro6 = Suministro **SJN cod_hotel** (Hotel6)

En este séptimo ejemplo tenemos los suministros correspondientes a los hoteles de Málaga Este suministro iría a la misma localidad donde estuviesen los hoteles de Málaga.

- ❏ Suministro7 = Suministro **SJN cod_hotel** (Hotel7)

En este octavo y último ejemplo tenemos los suministros correspondientes a los hoteles de Almería Este suministro iría a la misma localidad donde estuviesen los hoteles de Almería.

- ❏ Suministro8 = Suministro **SJN cod_hotel** (Hotel8)

Para la tabla **Reserva** se realiza el mismo procedimiento que para Suministro, de tal forma que cada reserva(i) iría a la misma localidad que su hotel(i), como sugiere el siguiente ejemplo:

- ❏ Reserva i = Reserva **SJN cod_hotel** (Hotel i)

Para la tabla **Empleado-Trabaja** se procede igual que en los dos casos anteriores, en donde cada Empleado-Trabaja irá al Hotel correspondiente como sugiere el siguiente ejemplo:

- ❏ Empleado-Trabaja i = Empleado-Trabaja **SJN cod_hotel** (Hotel i)

Para la tabla **Empleado-Contrato** contaremos con una fragmentación derivada, de tal forma que cada histórico de cada empleado irá a parar a la localidad perteneciente a los datos del hotel. Por tanto, será de la siguiente forma:

- ❏ Empleado-Contrato i = Empleado-Contrato **SJN cod_hotel** (Hotel i)

Para la tabla **Articulo-Vende** contaremos con una fragmentación derivada, de tal forma que cada artículo acompañado de sus datos y del proveedor que lo provee, irá a parar a la localidad perteneciente a los datos del proveedor. En este caso tan solo hay 2 posibilidades, o la localidad de Granada o la de Sevilla:

- ❏ Articulo-Vende i = Articulo-Vende **SJN codigo_proveedor** (Proveedor i)

La tabla Empleado-Contrato tendrá una fragmentación derivada, de tal forma que cada histórico de cada empleado irá a parar a la localidad del hotel correspondiente donde estuvo trabajando.

- Empleado-Contrato i = Empleado-Contrato **SJN cod-hotel** (Hotel i)

REPLICACIÓN

Para la replicación la componente principal que vamos a tener en cuenta es la volatilidad de los datos de las tablas, de tal forma que tengamos datos replicados en distintas localidades y esos datos apenas sufran actualizaciones.

Se procederá a replicar la tabla **Cliente**, la cuál es necesaria en cada localidad para la tabla Reserva que requiere de sus datos para poder hacer dichas reservas. A su vez se puede intuir que será una tabla poco volátil, ya que simplemente se tiene la posibilidad de dar de alta a un nuevo cliente o dar de baja, acciones que no serán tan dinámicas como las descritas anteriormente en la parte de fragmentación. Luego todo esto ha sido lo que nos ha llevado a replicar la tabla Cliente, una elección que parece más que lógica con todo lo descrito.

Localidad asociada a la base de datos:

1 -> Granada.
2 -> Cádiz.
3 -> Sevilla.
4 -> Málaga.

6. Implementación del diseño

6.1 Creación de tablas

La creación de las tablas se realiza en cada localidad donde se ha indicado en el apartado anterior que se va a fragmentar o replicar. Hay que prestar especial atención a la tabla **PROVEEDOR** ya que esta solamente se creará en las localidades de Granada y Sevilla como se especifica en la parte de fragmentación.

```
CREATE TABLE EMPLEADO_CONTRATO (codigo_empleado INT,
                                fecha_inicio_contrato_hotel DATE NOT NULL,
                                fecha_final_contrato_hotel DATE NOT NULL,
                                codigo_hotel INT NOT NULL REFERENCES HOTEL_DIRIGE(cod_hotel),
                                PRIMARY KEY (codigo_empleado, fecha_inicio_contrato_hotel,
fecha_final_contrato_hotel));


CREATE TABLE EMPLEADO_TRABAJA (codigo_empleado INT,
                               dni INT NOT NULL,
                               direccion VARCHAR(50),
                               fecha_inicio_contrato_cadena DATE NOT NULL,
        fecha_inicio_contrato_hotel DATE NOT NULL
                               salario FLOAT(10) NOT NULL,
                               nombre VARCHAR(20) NOT NULL,
                               telefono INT,
                               codigo_hotel INT NOT NULL REFERENCES   HOTEL_DIRIGE(cod_hotel),
                               PRIMARY KEY (codigo_empleado));


Donde el asterisco significa que varía entre 1 y 4.
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));



Donde el asterisco significa que varía entre 1 y 4.
CREATE TABLE RESERVA* (          fecha_inicio_reserva DATE NOT NULL,
                              fecha_final_reserva DATE NOT NULL,
                              tipo_habitacion VARCHAR(10) NOT NULL,
                              precio_habitacion FLOAT(10) NOT NULL,
                              codigo_hotel INT NOT NULL REFERENCES HOTEL_DIRIGE(cod_hotel),
                              codigo_cliente INT NOT NULL REFERENCES CLIENTE*(codigo_cliente),
                              PRIMARY KEY (codigo_cliente, fecha_inicio_reserva, fecha_final_reserva,
codigo_hotel));
```

Donde el asterisco significa que va en las localidades 1 y 3.

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

Donde el asterisco significa que varía entre 1 y 4.

```
CREATE TABLE SUMINISTRO* (codigo_articulo INT NOT NULL,
                          codigo_hotel INT,
                          codigo_proveedor INT NOT NULL ,
                          fecha_suministro DATE NOT NULL,
                          cantidad INT NOT NULL,
                          precio_por_unidad FLOAT(10) NOT NULL,
                          PRIMARY KEY (codigo_articulo, codigo_proveedor, codigo_hotel,
fecha_suministro),
                          FOREIGN KEY (codigo_articulo, codigo_proveedor) REFERENCES
ARTICULO_VENDE(codigo_articulo, codigo_proveedor),
                          FOREIGN KEY (codigo_hotel) REFERENCES HOTEL_DIRIGE(cod_hotel)
                          );
```

```
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));
```

6.2 Garantizando privilegios

GRANT DE LA LOCALIDAD DE GRANADA

```
Grant SELECT, INSERT, DELETE, UPDATE ON EMPLEADO_CONTRATO TO gbdd2, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON RESERVA1 TO gbdd2, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON CLIENTE1 TO gbdd2, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON PROVEEDOR1 TO gbdd2, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON EMPLEADO_TRABAJA TO gbdd2, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON SUMINISTRO1 TO gbdd2, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON HOTEL_DIRIGE TO gbdd2, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON ARTICULO_VENDE TO gbdd2, gbdd3, gbdd4;
```

GRANT DE LA LOCALIDAD DE CÁDIZ

```
Grant SELECT, INSERT, DELETE, UPDATE ON EMPLEADO_TRABAJA TO gbdd1, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON EMPLEADO_CONTRATO TO gbdd1, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON RESERVA2 TO gbdd1, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON CLIENTE2 TO gbdd1, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON SUMINISTRO2 TO gbdd1, gbdd3, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON HOTEL_DIRIGE TO gbdd1, gbdd3, gbdd4;
```

GRANT DE LA LOCALIDAD DE SEVILLA

```
Grant SELECT, INSERT, DELETE, UPDATE ON PROVEEDOR3 TO gbdd1, gbdd2, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON EMPLEADO_TRABAJA TO gbdd1, gbdd2, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON EMPLEADO_CONTRATO TO gbdd1, gbdd2, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON RESERVA3 TO gbdd1, gbdd2, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON CLIENTE3 TO gbdd1, gbdd2, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON SUMINISTRO3 TO gbdd1, gbdd2, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON HOTEL_DIRIGE TO gbdd1, gbdd2, gbdd4;

Grant SELECT, INSERT, DELETE, UPDATE ON ARTICULO_VENDE TO gbdd1, gbdd2, gbdd4;
```

GRANT DE LA LOCALIDAD DE MÁLAGA

```
Grant SELECT, INSERT, DELETE, UPDATE ON EMPLEADO_TRABAJA TO gbdd1, gbdd2, gbdd3;

Grant SELECT, INSERT, DELETE, UPDATE ON EMPLEADO_CONTRATO TO gbdd1, gbdd2, gbdd3;

Grant SELECT, INSERT, DELETE, UPDATE ON RESERVA4 TO gbdd1, gbdd2, gbdd3;

Grant SELECT, INSERT, DELETE, UPDATE ON CLIENTE4 TO gbdd1, gbdd2, gbdd3;

Grant SELECT, INSERT, DELETE, UPDATE ON SUMINISTRO4 TO gbdd1, gbdd2, gbdd3;

Grant SELECT, INSERT, DELETE, UPDATE ON HOTEL_DIRIGE TO gbdd1, gbdd2, gbdd3;
```

6.3 Creación de vistas

LOCALIDAD DE GRANADA

VISTAS GLOBALES DE LA TABLA EMPLEADO-TRABAJA

```
CREATE VIEW EMPLEADO AS
SELECT * FROM EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD2.EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD3.EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD4.EMPLEADO_TRABAJA;
```

VISTAS GLOBALES DE LA TABLA HISTORICO-EMPLEADO

```
CREATE VIEW HISTORICO_EMPLEADOS AS
SELECT * FROM EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD2.EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD3.EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD4.EMPLEADO_CONTRATO;
```

VISTAS GLOBALES DE LA TABLA CLIENTE

```
CREATE VIEW CLIENTE AS
SELECT * FROM CLIENTE1
UNION
SELECT * FROM GBDD2.CLIENTE2
UNION
SELECT * FROM GBDD3.CLIENTE3
UNION
SELECT * FROM GBDD4.CLIENTE4;
```

VISTAS GLOBALES DE LA TABLA RESERVA

```
CREATE VIEW RESERVA AS
SELECT * FROM RESERVA1
UNION
SELECT * FROM GBDD2.RESERVA2
UNION
SELECT * FROM GBDD3.RESERVA3
UNION
SELECT * FROM GBDD4.RESERVA4;
```

VISTAS GLOBALES DE LA TABLA PROVEEDOR

```
CREATE VIEW PROVEEDOR AS
SELECT * FROM PROVEEDOR1
UNION
SELECT * FROM GBDD3.PROVEEDOR3;
```

VISTAS GLOBALES DE LA TABLA HOTEL-DIRIGE

```
CREATE VIEW HOTEL AS
SELECT * FROM HOTEL_DIRIGE
UNION
SELECT * FROM GBDD2.HOTEL_DIRIGE
UNION
SELECT * FROM GBDD3.HOTEL_DIRIGE
UNION
SELECT * FROM GBDD4.HOTEL_DIRIGE;
```

VISTAS GLOBALES DE LA TABLA ARTÍCULO-VENDE

```
CREATE VIEW ARTICULO AS
SELECT * FROM ARTICULO_VENDE
UNION
SELECT * FROM GBDD3.ARTICULO_VENDE;
```

VISTAS GLOBALES DE LA TABLA SUMINISTRO

```
CREATE VIEW SUMINISTRO AS
SELECT * FROM SUMINISTRO1
UNION
SELECT * FROM GBDD2.SUMINISTRO2
UNION
SELECT * FROM GBDD3.SUMINISTRO3
UNION
SELECT * FROM GBDD4.SUMINISTRO4;
```

VISTAS GLOBALES DE LA TABLA EMPLEADO-TRABAJA

```
CREATE VIEW EMPLEADO AS
SELECT * FROM EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD1.EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD3.EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD4.EMPLEADO_TRABAJA;
```

VISTAS GLOBALES DE LA TABLA HISTORICO-EMPLEADO

```
CREATE VIEW HISTORICO_EMPLEADOS AS
SELECT * FROM EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD1.EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD3.EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD4.EMPLEADO_CONTRATO;
```

VISTAS GLOBALES DE LA TABLA CLIENTE

```
CREATE VIEW CLIENTE AS
SELECT * FROM CLIENTE2
UNION
SELECT * FROM GBDD1.CLIENTE1
UNION
SELECT * FROM GBDD3.CLIENTE3
UNION
SELECT * FROM GBDD4.CLIENTE4;
```

VISTAS GLOBALES DE LA TABLA RESERVA

```
CREATE VIEW RESERVA AS
SELECT * FROM RESERVA2
UNION
SELECT * FROM GBDD1.RESERVA1
UNION
SELECT * FROM GBDD3.RESERVA3
UNION
SELECT * FROM GBDD4.RESERVA4;
```

VISTAS GLOBALES DE LA TABLA HOTEL-DIRIGE

```
CREATE VIEW HOTEL AS
SELECT * FROM HOTEL_DIRIGE
UNION
SELECT * FROM GBDD1.HOTEL_DIRIGE
UNION
SELECT * FROM GBDD3.HOTEL_DIRIGE
UNION
SELECT * FROM GBDD4.HOTEL_DIRIGE;
```

VISTAS GLOBALES DE LA TABLA SUMINISTRO

```
CREATE VIEW SUMINISTRO AS
SELECT * FROM SUMINISTRO2
UNION
SELECT * FROM GBDD1.SUMINISTRO1
UNION
SELECT * FROM GBDD3.SUMINISTRO3
UNION
SELECT * FROM GBDD4.SUMINISTRO4;
```

LOCALIDAD DE SEVILLA

VISTAS GLOBALES DE LA TABLA EMPLEADO-TRABAJA

```
CREATE VIEW EMPLEADO AS
SELECT * FROM EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD1.EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD2.EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD4.EMPLEADO_TRABAJA;
```

VISTAS GLOBALES DE LA TABLA HISTORICO-EMPLEADO

```
CREATE VIEW HISTORICO_EMPLEADOS AS
SELECT * FROM EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD1.EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD2.EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD4.EMPLEADO_CONTRATO;
```

VISTAS GLOBALES DE LA TABLA CLIENTE

```
CREATE VIEW CLIENTE AS
SELECT * FROM CLIENTE3
UNION
SELECT * FROM GBDD1.CLIENTE1
UNION
SELECT * FROM GBDD2.CLIENTE2
UNION
SELECT * FROM GBDD4.CLIENTE4;
```

VISTAS GLOBALES DE LA TABLA RESERVA

```
CREATE VIEW RESERVA AS
SELECT * FROM RESERVA3
UNION
SELECT * FROM GBDD1.RESERVA1
UNION
SELECT * FROM GBDD2.RESERVA2
UNION
SELECT * FROM GBDD4.RESERVA4;
```

VISTAS GLOBALES DE LA TABLA HOTEL-DIRIGE

```
CREATE VIEW HOTEL AS
SELECT * FROM HOTEL_DIRIGE
UNION
SELECT * FROM GBDD1.HOTEL_DIRIGE
UNION
SELECT * FROM GBDD2.HOTEL_DIRIGE
UNION
SELECT * FROM GBDD4.HOTEL_DIRIGE;
```

VISTAS GLOBALES DE LA TABLA PROVEEDOR

```
CREATE VIEW PROVEEDOR AS
SELECT * FROM PROVEEDOR3
UNION
SELECT * FROM GBDD1.PROVEEDOR1;
```

VISTAS GLOBALES DE LA TABLA ARTÍCULO-VENDE

```
CREATE VIEW ARTICULO AS
SELECT * FROM ARTICULO_VENDE
UNION
SELECT * FROM GBDD1.ARTICULO_VENDE;
```

VISTAS GLOBALES DE LA TABLA SUMINISTRO

```
CREATE VIEW SUMINISTRO AS
SELECT * FROM SUMINISTRO3
UNION
SELECT * FROM GBDD2.SUMINISTRO2
UNION
SELECT * FROM GBDD1.SUMINISTRO1
UNION
SELECT * FROM GBDD4.SUMINISTRO4;
```

VISTAS GLOBALES DE LA TABLA EMPLEADO-TRABAJA

```
CREATE VIEW EMPLEADO AS
SELECT * FROM EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD1.EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD3.EMPLEADO_TRABAJA
UNION
SELECT * FROM GBDD2.EMPLEADO_TRABAJA;
```

VISTAS GLOBALES DE LA TABLA HISTORICO-EMPLEADO

```
CREATE VIEW HISTORICO_EMPLEADOS AS
SELECT * FROM EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD1.EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD3.EMPLEADO_CONTRATO
UNION
SELECT * FROM GBDD2.EMPLEADO_CONTRATO;
```

VISTAS GLOBALES DE LA TABLA CLIENTE

```
CREATE VIEW CLIENTE AS
SELECT * FROM CLIENTE3
UNION
SELECT * FROM GBDD1.CLIENTE1
UNION
SELECT * FROM GBDD3.CLIENTE3
UNION
SELECT * FROM GBDD2.CLIENTE2;
```

VISTAS GLOBALES DE LA TABLA RESERVA

```
CREATE VIEW RESERVA AS
SELECT * FROM RESERVA3
UNION
SELECT * FROM GBDD1.RESERVA1
UNION
SELECT * FROM GBDD3.RESERVA3
UNION
SELECT * FROM GBDD2.RESERVA2;
```

VISTAS GLOBALES DE LA TABLA HOTEL-DIRIGE

```
CREATE VIEW HOTEL AS
SELECT * FROM HOTEL_DIRIGE
UNION
SELECT * FROM GBDD1.HOTEL_DIRIGE
UNION
SELECT * FROM GBDD3.HOTEL_DIRIGE
UNION
SELECT * FROM GBDD2.HOTEL_DIRIGE;
```

VISTAS GLOBALES DE LA TABLA SUMINISTRO

```
CREATE VIEW SUMINISTRO AS
SELECT * FROM SUMINISTRO4
UNION
SELECT * FROM GBDD1.SUMINISTRO1
UNION
SELECT * FROM GBDD3.SUMINISTRO3
UNION
SELECT * FROM GBDD2.SUMINISTRO2;
```

6.4 Implementación de los Disparadores

LOCALIDAD DE GRANADA

Disparador asociado a la tabla EMPLEADO-TRABAJA, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo empleado que se vaya a insertar no existe ya. En caso de que exista el empleado, se lanza una excepción denotando tal inconveniente.

```
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, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo hotel que se vaya a insertar no existe ya. En caso de que exista el hotel, se lanza una excepción denotando tal inconveniente.

```
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 COD_HOTEL = :NEW.COD_HOTEL;

        IF contarHoteles > 0 THEN
            raise_application_error(-20004, 'TAL HOTEL YA EXISTE');
        END IF;
    END IF;
END;
```

Disparador asociado a la tabla CLIENTE1, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo cliente que se vaya a insertar no existe ya. En caso de que exista el cliente, se lanza una excepción denotando tal inconveniente.

```
create or replace TRIGGER CLIENTE_TRIGGER
    BEFORE INSERT OR UPDATE ON CLIENTE1 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 ARTICULO-VENDE, con previo aviso a la inserción de una nueva tupla. En este disparador se comprueban varias cosas, en primer lugar que el tipo de artículo sea el adecuado (A, B, C o D). En segundo lugar se comprueba de que provincia es el proveedor que va a proveer dicho artículo, en caso de no ser ni de Granada ni de Sevilla lanzaríamos una excepción. En caso de ser de alguna de las dos localidades, cogemos su provincia y la guardamos en la variable **provinciaProveedor_nuevo**, después de esto comprobamos que el artículo no esté suministrado ya por dos proveedores, en caso de que sí lo esté, lanzamos una excepción. En caso de que solamente esté suministrado por 1 proveedor, comprobamos con la variable en donde guardamos previamente la provincia del proveedor nuevo que no sean iguales las provincias del nuevo proveedor con la del viejo, en caso contrario lanzamos una excepción.

```
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 PROVINCIA = 'Granada';

        if numProveedores = 0 then

            SELECT COUNT(*) INTO numProveedores FROM PROVEEDOR
```

```
        WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and PROVINCIA = 'Sevilla';

        if numProveedores = 0 then
            raise_application_error(-20004,'TAL PROVEDDOR NO ES NI DE GRANADA NI DE SEVILLA');

        ELSE
            SELECT PROVINCIA INTO provinciaProveedor_nuevo FROM PROVEEDOR
            WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and PROVINCIA = 'Sevilla';
        END IF;
    ELSE
        SELECT PROVINCIA INTO provinciaProveedor_nuevo
        FROM PROVEEDOR
        WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and PROVINCIA = '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 PROVINCIA 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 PROVINCIA
QUE LA DEL NUEVO PROVEEDOR');
            END IF;
        END IF;
    END IF;
END;
```

Disparador asociado a la tabla EMPLEADO-TRABAJA, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo empleado que se vaya a insertar no existe ya. En caso de que exista el empleado, se lanza una excepción denotando tal inconveniente.

```
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, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo hotel que se vaya a insertar no existe ya. En caso de que exista el hotel, se lanza una excepción denotando tal inconveniente.

```
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 COD_HOTEL = :NEW.COD_HOTEL;

        IF contarHoteles > 0 THEN
            raise_application_error(-20004, 'TAL HOTEL YA EXISTE');
        END IF;
    END IF;
END;
```

Disparador asociado a la tabla CLIENTE2, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo cliente que se vaya a insertar no existe ya. En caso de que exista el cliente, se lanza una excepción denotando tal inconveniente.

```
create or replace TRIGGER CLIENTE_TRIGGER
    BEFORE INSERT OR UPDATE ON CLIENTE2 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 EMPLEADO-TRABAJA, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo empleado que se vaya a insertar no existe ya. En caso de que exista el empleado, se lanza una excepción denotando tal inconveniente.

```
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, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo hotel que se vaya a insertar no existe ya. En caso de que exista el hotel, se lanza una excepción denotando tal inconveniente.

```
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 COD_HOTEL = :NEW.COD_HOTEL;

        IF contarHoteles > 0 THEN
            raise_application_error(-20004, 'TAL HOTEL YA EXISTE');
        END IF;
    END IF;
END;
```

Disparador asociado a la tabla CLIENTE3, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo cliente que se vaya a insertar no existe ya. En caso de que exista el cliente, se lanza una excepción denotando tal inconveniente.

```
create or replace TRIGGER CLIENTE_TRIGGER
    BEFORE INSERT OR UPDATE ON CLIENTE3 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 ARTICULO-VENDE, con previo aviso a la inserción de una nueva tupla. En este disparador se comprueban varias cosas, en primer lugar que el tipo de artículo sea el adecuado (A, B, C o D). En segundo lugar se comprueba de que provincia es el proveedor que va a proveer dicho artículo, en caso de no ser ni de Granada ni de Sevilla lanzaríamos una excepción. En caso de ser de alguna de las dos localidades, cogemos su provincia y la guardamos en la variable **provinciaProveedor_nuevo**, después de esto comprobamos que el artículo no esté suministrado ya por dos proveedores, en caso de que sí lo esté, lanzamos una excepción. En caso de que solamente esté suministrado por 1 proveedor, comprobamos con la variable en donde guardamos previamente la provincia del proveedor nuevo que no sean iguales las provincias del nuevo proveedor con la del viejo, en caso contrario lanzamos una excepción.

```
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 PROVINCIA = 'Granada';

        if numProveedores = 0 then

            SELECT COUNT(*) INTO numProveedores FROM PROVEEDOR
```

```
        WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and PROVINCIA = 'Sevilla';

        if numProveedores = 0 then
            raise_application_error(-20004,'TAL PROVEDDOR NO ES NI DE GRANADA NI DE SEVILLA');

        ELSE
            SELECT PROVINCIA INTO provinciaProveedor_nuevo FROM PROVEEDOR
            WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and PROVINCIA = 'Sevilla';
        END IF;
    ELSE
        SELECT PROVINCIA INTO provinciaProveedor_nuevo
        FROM PROVEEDOR
        WHERE CODIGO_PROVEEDOR = :NEW.CODIGO_PROVEEDOR and PROVINCIA = '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 PROVINCIA 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 PROVINCIA
QUE LA DEL NUEVO PROVEEDOR');
            END IF;
        END IF;
    END IF;
END;
```

Disparador asociado a la tabla EMPLEADO-TRABAJA, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo empleado que se vaya a insertar no existe ya. En caso de que exista el empleado, se lanza una excepción denotando tal inconveniente.

```
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, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo hotel que se vaya a insertar no existe ya. En caso de que exista el hotel, se lanza una excepción denotando tal inconveniente.

```
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 COD_HOTEL = :NEW.COD_HOTEL;

        IF contarHoteles > 0 THEN
            raise_application_error(-20004, 'TAL HOTEL YA EXISTE');
        END IF;
    END IF;
END;
```

Disparador asociado a la tabla CLIENTE4, con previo aviso a la inserción de una nueva tupla.En este caso el disparador simplemente comprueba que el nuevo cliente que se vaya a insertar no existe ya. En caso de que exista el cliente, se lanza una excepción denotando tal inconveniente.

```
create or replace TRIGGER CLIENTE_TRIGGER
    BEFORE INSERT OR UPDATE ON CLIENTE4 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;
```

Nota: Como se puede comprobar en esta parte no está implementada toda la funcionalidad que se pedía en la parte de implementación del diseño. Esto es debido a que la funcionalidad restante se ha tenido que realizar en los procedimientos. En cada procedimiento se explicará que se ha hecho minuciosamente, y porqué se ha hecho de esa forma, metiendo funcionalidad que en principio se pedía en los Disparadores.

7. Implementación de las actualizaciones

Procedimiento dedicado para la inserción de un nuevo empleado denominado **INSERTAREMPLEADO.** El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ En segundo lugar se comprueba que el hotel que se inserte para indicar donde trabaja el empleado exista, en caso de que no sea así se lanzará una excepción.

- ❏ En tercer lugar se comprueba que la fecha de inicio de contrato en el hotel no sea menor que la fecha de inicio de contrato en la cadena hotelera, en caso de que sea así se lanzará una excepción.

- ❏ En cuarto lugar se comprueba que la fecha de inicio de contrato del empleado que se inserta no sea menor a la anterior fecha final de contrato de este mismo empleado. En caso de no poseer un anterior contrato al actual no se comprueba.

- ❏ Como paso final se comprueba en que localidad debe de insertarse la tupla del nuevo empleado, ya que como hemos explicado anteriormente la tabla de **empleado-trabaja** está fragmentada.

```
create or replace PROCEDURE
insertarEmpleado(codigo NUMBER, dni NUMBER, direccion VARCHAR, fecha_inicio_contrato_cadena DATE,
fecha_inicio_contrato_hotel DATE, salario FLOAT, nombre VARCHAR,telefono NUMBER, codigo_hotel NUMBER) IS

contar NUMBER;
contarMaximaFechaFinContrato DATE;

BEGIN


if codigo is null OR dni is NULL OR fecha_inicio_contrato_cadena is NULL OR
fecha_inicio_contrato_hotel is NULL OR salario is NULL OR nombre is NULL OR codigo_hotel 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 COD_HOTEL = codigo_hotel;

IF contar = 0 THEN
    raise_application_error(-20004,'EL HOTEL QUE SE HA INSERTADO NO EXISTE');
END IF;

IF fecha_inicio_contrato_hotel < fecha_inicio_contrato_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_CONTRATO_HOTEL) INTO contarMaximaFechaFinContrato FROM HISTORICO_EMPLEADOS
```

```
    WHERE CODIGO_EMPLEADO = codigo;

    IF contarMaximaFechaFinContrato is not NULL THEN
        IF contarMaximaFechaFinContrato > fecha_inicio_contrato_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 COD_HOTEL = codigo_hotel;

    IF contar > 0 THEN
        INSERT INTO GBDD1.EMPLEADO_TRABAJA VALUES(codigo, dni, direccion, fecha_inicio_contrato_cadena,salario,
nombre,
        telefono, codigo_hotel, fecha_inicio_contrato_hotel);
    ELSE
        SELECT COUNT(*) INTO contar FROM GBDD2.HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel;

        IF contar > 0 THEN
            INSERT INTO GBDD2.EMPLEADO_TRABAJA VALUES(codigo, dni, direccion,
fecha_inicio_contrato_cadena,salario, nombre,
            telefono, codigo_hotel, fecha_inicio_contrato_hotel);
        ELSE
            SELECT COUNT(*) INTO contar FROM GBDD3.HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel;

            IF contar > 0 THEN
                INSERT INTO GBDD3.EMPLEADO_TRABAJA VALUES(codigo, dni, direccion,
fecha_inicio_contrato_cadena,salario, nombre,
                telefono, codigo_hotel, fecha_inicio_contrato_hotel);

            ELSE
                SELECT COUNT(*) INTO contar FROM GBDD4.HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel;

                IF contar > 0 THEN
                    INSERT INTO GBDD4.EMPLEADO_TRABAJA VALUES(codigo, dni, direccion,
fecha_inicio_contrato_cadena,salario, nombre,
                        telefono, codigo_hotel, fecha_inicio_contrato_hotel);
                END IF;
            END IF;
        END IF;
    END IF;
END;
```

Procedimiento dedicado para la eliminación de un empleado denominado **ELIMINAREMPLEADO.**

El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ En segundo lugar se comprueba en qué localidad está almacenado el registro del empleado que se va a dar de baja, y una vez que se tiene la localidad se inserta el nuevo registro histórico en esa localidad y se elimina de la tabla **Empleado-Trabaja** el registro actual.

- ❏ Cabe decir que también se comprueba, una vez que sabemos la localidad en la que está el registro almacenado, que la fecha de inicio del contrato sea menor a la fecha de final del contrato que se haya insertado.

```
create or replace PROCEDURE
eliminarEmpleado(codigo NUMBER, fecha_final_contrato DATE) IS

contar NUMBER;
fecha_inicio DATE;
hotel_trabajaba NUMBER;

BEGIN
    IF codigo is NULL OR fecha_final_contrato 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 GBDD1.EMPLEADO_TRABAJA
    WHERE CODIGO_EMPLEADO = codigo;

    IF contar > 0 THEN

        SELECT CODIGO_HOTEL, FECHA_INICIO_CONTRATO_HOTEL
        INTO  hotel_trabajaba, fecha_inicio
        FROM GBDD1.EMPLEADO_TRABAJA
        WHERE CODIGO_EMPLEADO = codigo;

        IF fecha_final_contrato < 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 GBDD1.EMPLEADO_CONTRATO
        VALUES(codigo, fecha_inicio,fecha_final_contrato, hotel_trabajaba);

        DELETE FROM GBDD1.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;
    ELSE
        SELECT COUNT(*)
        INTO contar
        FROM GBDD2.EMPLEADO_TRABAJA
        WHERE CODIGO_EMPLEADO = codigo;


        IF contar > 0 THEN

            SELECT CODIGO_HOTEL, FECHA_INICIO_CONTRATO_HOTEL
```

```
            INTO  hotel_trabajaba, fecha_inicio
            FROM GBDD2.EMPLEADO_TRABAJA
            WHERE CODIGO_EMPLEADO = codigo;

            IF fecha_final_contrato < 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 GBDD2.EMPLEADO_CONTRATO
            VALUES(codigo, fecha_inicio,fecha_final_contrato, hotel_trabajaba);


            DELETE FROM GBDD2.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;

        ELSE
            SELECT COUNT(*)
            INTO contar
            FROM GBDD3.EMPLEADO_TRABAJA
            WHERE CODIGO_EMPLEADO = codigo;
            IF contar > 0 THEN

                SELECT CODIGO_HOTEL, FECHA_INICIO_CONTRATO_HOTEL
                INTO  hotel_trabajaba, fecha_inicio
                FROM GBDD3.EMPLEADO_TRABAJA
                WHERE CODIGO_EMPLEADO = codigo;

                IF fecha_final_contrato < 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 GBDD3.EMPLEADO_CONTRATO
                VALUES(codigo, fecha_inicio,fecha_final_contrato, hotel_trabajaba);


                DELETE FROM GBDD3.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;
            ELSE
                SELECT COUNT(*)
                INTO contar
                FROM GBDD4.EMPLEADO_TRABAJA
                WHERE CODIGO_EMPLEADO = codigo;

                IF contar > 0 THEN
                    SELECT CODIGO_HOTEL, FECHA_INICIO_CONTRATO_HOTEL
                    INTO hotel_trabajaba, fecha_inicio
                    FROM GBDD4.EMPLEADO_TRABAJA
                    WHERE CODIGO_EMPLEADO = codigo;

                    IF fecha_final_contrato < 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 GBDD4.EMPLEADO_CONTRATO
                    VALUES(codigo, fecha_inicio,fecha_final_contrato, hotel_trabajaba);

                    DELETE FROM GBDD4.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;
                ELSE
                    raise_application_error(-20004,'TAL EMPLEADO NO EXISTE');
                END IF;
            END IF;
        END IF;
    END IF;
```

```
END;
```

Procedimiento dedicado para la modificación del salario de un empleado denominado **MODIFICARSALARIOEMPLEADO.**

El procedimiento consta de los siguientes pasos:

- En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- En segundo lugar se comprueba que el empleado exista.

- En tercer lugar se comprueba que el salario sea mayor que el antiguo salario del empleado.

- Y por último se comprueba en que localidad está para realizar la actualización del registro correspondiente.

```
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 GBDD2.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;

        IF contar > 0 THEN
            UPDATE GBDD2.EMPLEADO_TRABAJA SET SALARIO = salario_p WHERE CODIGO_EMPLEADO = codigo;
        ELSE
```

```
            SELECT COUNT(*) INTO contar FROM GBDD3.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;

            IF contar > 0 THEN
               UPDATE GBDD3.EMPLEADO_TRABAJA SET SALARIO = salario_p WHERE CODIGO_EMPLEADO = codigo;
            ELSE
                SELECT COUNT(*) INTO contar FROM GBDD4.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;

                IF contar > 0 THEN
                    UPDATE GBDD4.EMPLEADO_TRABAJA SET SALARIO = salario_p WHERE CODIGO_EMPLEADO = codigo;
                END IF;
            END IF;
        END IF;
    END IF;
END;
```

Procedimiento dedicado para trasladar a un empleado de hotel denominado **TRASLADAREMPLEADO.**

El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ En segundo lugar se comprueba tanto que el empleado exista como que el hotel al que se va a trasladar exista, si estos datos no son correctos lanzaremos la excepción correspondiente.

- ❏ En tercer lugar se guardan los datos referentes al empleado en variables para que más adelante podamos guardarlos en un nuevo registro, o para que en caso de que no se pueda realizar el traslado por algún motivo podamos volver a insertar el registro donde estaba.

- ❏ Si el hotel que se va a trasladar es el mismo que en el que está, se lanzará una excepción.

- ❏ Si la fecha de inicio del nuevo contrato es inferior a la fecha final del contrato anterior se lanzará una excepción.
- ❏ Si la fecha de final del contrato anterior es menor que la fecha inicio del contrato anterior se lanzará una excepción.

- ❏ Tras las anteriores comprobaciones se mira en que localidad está el empleado guardado, para eliminarlo e insertarlo en la tabla de históricos.

- ❏ Una vez tenemos al empleado eliminado de su anterior hotel, y guardado en la tabla de históricos, procedemos a insertarlo en la nueva tabla correspondiente a la localidad del nuevo hotel.

```
create or replace PROCEDURE
trasladarEmpleado(codigo NUMBER, fecha_final_contrato_hotel_act DATE, codigo_hotel_nuevo NUMBER,
```

```
fecha_inicio_contrato_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 codigo is NULL OR fecha_final_contrato_hotel_act is NULL OR codigo_hotel_nuevo is NULL
    OR fecha_inicio_contrato_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 = codigo;

    IF contar > 0 THEN

        SELECT COUNT(*)
        INTO contar2
        FROM HOTEL
        WHERE COD_HOTEL = codigo_hotel_nuevo;

        IF contar2 > 0 THEN

            SELECT DNI, NOMBRE, FECHA_INICIO_CONTRATO_CADENA, SALARIO,
            FECHA_INICIO_CONTRATO_HOTEL, CODIGO_HOTEL
            INTO dni_empleado, nombre_empleado, fecha_inicio_cadena, salario_empleado,
            fecha_inicio_hotel, codigo_hotel_viejo
            FROM EMPLEADO
            WHERE CODIGO_EMPLEADO = codigo;

            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_contrato_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_contrato_hotel_nu < fecha_final_contrato_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 GBDD1.EMPLEADO_TRABAJA
            WHERE CODIGO_EMPLEADO = codigo;

            IF contarEmpleado > 0 THEN

                DELETE FROM GBDD1.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;

                INSERT INTO GBDD1.EMPLEADO_CONTRATO
                VALUES (codigo, fecha_inicio_hotel,
                fecha_final_contrato_hotel_act, codigo_hotel_viejo);
```

```
        ELSE

            SELECT COUNT(*) INTO contarEmpleado FROM GBDD2.EMPLEADO_TRABAJA
            WHERE CODIGO_EMPLEADO = codigo;

            IF contarEmpleado > 0 THEN

                DELETE FROM GBDD2.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;

                INSERT INTO GBDD2.EMPLEADO_CONTRATO
                VALUES (codigo, fecha_inicio_hotel,
                fecha_final_contrato_hotel_act, codigo_hotel_viejo);

            ELSE

                SELECT COUNT(*) INTO contarEmpleado FROM GBDD3.EMPLEADO_TRABAJA
                WHERE CODIGO_EMPLEADO = codigo;

                IF contarEmpleado > 0 THEN

                    DELETE FROM GBDD3.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;

                    INSERT INTO GBDD3.EMPLEADO_CONTRATO
                    VALUES (codigo, fecha_inicio_hotel,
                    fecha_final_contrato_hotel_act, codigo_hotel_viejo);

                ELSE
                    SELECT COUNT(*) INTO contarEmpleado FROM GBDD4.EMPLEADO_TRABAJA
                    WHERE CODIGO_EMPLEADO = codigo;

                    IF contarEmpleado > 0 THEN

                        DELETE FROM GBDD4.EMPLEADO_TRABAJA WHERE CODIGO_EMPLEADO = codigo;


                        INSERT INTO GBDD4.EMPLEADO_CONTRATO
                        VALUES (codigo, fecha_inicio_hotel,
                        fecha_final_contrato_hotel_act, codigo_hotel_viejo);
                    END IF;
                END IF;
            END IF;
        END IF;

        SELECT COUNT(*) INTO contarHotel FROM HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel_nuevo;

        IF contarHotel > 0 THEN

            INSERT INTO GBDD1.EMPLEADO_TRABAJA
            VALUES (codigo, dni_empleado, direccion, fecha_inicio_cadena,
            salario_empleado, nombre_empleado, telefono, codigo_hotel_nuevo,
            fecha_inicio_contrato_hotel_nu);

        ELSE

            SELECT COUNT(*) INTO contarHotel FROM GBDD2.HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel_nuevo;

            IF contarHotel > 0 THEN

                INSERT INTO GBDD2.EMPLEADO_TRABAJA
                VALUES (codigo, dni_empleado, direccion, fecha_inicio_cadena,
                salario_empleado, nombre_empleado, telefono, codigo_hotel_nuevo,
                fecha_inicio_contrato_hotel_nu);

            ELSE

                SELECT COUNT(*) INTO contarHotel FROM GBDD3.HOTEL_DIRIGE WHERE COD_HOTEL =
codigo_hotel_nuevo;

                IF contarHotel > 0 THEN
```

```
                    INSERT INTO GBDD3.EMPLEADO_TRABAJA
                    VALUES (codigo, dni_empleado, direccion, fecha_inicio_cadena,
                    salario_empleado, nombre_empleado, telefono, codigo_hotel_nuevo,
                    fecha_inicio_contrato_hotel_nu);

                ELSE
                    SELECT COUNT(*) INTO contarHotel FROM GBDD4.HOTEL_DIRIGE WHERE COD_HOTEL =
codigo_hotel_nuevo;

                    IF contarHotel > 0 THEN
                        INSERT INTO GBDD4.EMPLEADO_TRABAJA
                        VALUES (codigo, dni_empleado, direccion, fecha_inicio_cadena,
                        salario_empleado, nombre_empleado, telefono, codigo_hotel_nuevo,
                        fecha_inicio_contrato_hotel_nu);
                    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 dedicado para la inserción de un nuevo hotel denominado **INSERTARHOTEL.**
El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ En segundo lugar se comprueba en que localidad irá el nuevo hotel dependiendo de la provincia. En caso de que se inserte una provincia no válida, se lanzará una excepción.

```
create or replace PROCEDURE
insertarHOTEL(COD_HOTEL_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 COD_HOTEL_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 GBDD1.HOTEL_DIRIGE
      VALUES(COD_HOTEL_P , CIUDAD_P , PROVINCIA_P , NOMBRE_P , NUMERO_HABITACIONES_SIMPLE_P ,
NUMERO_HABITACIONES_DOBLE_P , NULL );
```

```
    ELSIF PROVINCIA_P  = 'Cadiz' or PROVINCIA_P  = 'Huelva' THEN
      INSERT INTO GBDD2.HOTEL_DIRIGE
      VALUES(COD_HOTEL_P , CIUDAD_P , PROVINCIA_P , NOMBRE_P , NUMERO_HABITACIONES_SIMPLE_P ,
NUMERO_HABITACIONES_DOBLE_P , NULL);

    ELSIF PROVINCIA_P  = 'Sevilla' or PROVINCIA_P  = 'Cordoba' THEN
      INSERT INTO GBDD3.HOTEL_DIRIGE
      VALUES(COD_HOTEL_P , CIUDAD_P , PROVINCIA_P , NOMBRE_P , NUMERO_HABITACIONES_SIMPLE_P ,
NUMERO_HABITACIONES_DOBLE_P , NULL);


    ELSIF PROVINCIA_P  = 'Malaga' or PROVINCIA_P  = 'Almeria' THEN
      INSERT INTO GBDD4.HOTEL_DIRIGE
      VALUES(COD_HOTEL_P , CIUDAD_P , PROVINCIA_P , NOMBRE_P , NUMERO_HABITACIONES_SIMPLE_P ,
NUMERO_HABITACIONES_DOBLE_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 dedicado para cambiar el director de un hotel denominado **CAMBIARDIRECTORHOTEL.**

El procedimiento consta de los siguientes pasos:

- En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- En segundo lugar se comprueba que el empleado no dirija ya a un hotel, en caso de que dirija a otro se lanzará una excepción.

- En tercer lugar comprobar que el empleado exista, si no existe se lanzará una excepción.

- Y por último se comprueba en qué localidad está dicho hotel, para realizar la actualización correspondiente del director.

- También saltará una excepción en el caso de que se inserte un hotel no existente en la base de datos.

```
create or replace PROCEDURE
cambiarDirectorHotel(codigo_hotel NUMBER, codigo_director NUMBER) IS

contar NUMBER;
contarEmpleadoDirige NUMBER;

BEGIN

    IF codigo_hotel 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 CODIGO_EMPLEADO_DIRIGE = 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 GBDD1.HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel;

    IF contar > 0 THEN
        UPDATE GBDD1.HOTEL_DIRIGE SET CODIGO_EMPLEADO_DIRIGE = codigo_director
        WHERE COD_HOTEL = codigo_hotel;
    ELSE
        SELECT COUNT(*) INTO contar FROM GBDD2.HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel;

        IF contar > 0 THEN
            UPDATE GBDD2.HOTEL_DIRIGE SET CODIGO_EMPLEADO_DIRIGE = codigo_director
            WHERE COD_HOTEL = codigo_hotel;
        ELSE
            SELECT COUNT(*) INTO contar FROM GBDD3.HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel;

            IF contar > 0 THEN
               UPDATE GBDD3.HOTEL_DIRIGE SET CODIGO_EMPLEADO_DIRIGE = codigo_director
               WHERE COD_HOTEL = codigo_hotel;
            ELSE
                SELECT COUNT(*) INTO contar FROM GBDD4.HOTEL_DIRIGE WHERE COD_HOTEL = codigo_hotel;

                IF contar > 0 THEN
                    UPDATE GBDD4.HOTEL_DIRIGE SET CODIGO_EMPLEADO_DIRIGE = codigo_director
                    WHERE COD_HOTEL = codigo_hotel;

                ELSE
                    raise_application_error(-20004, 'TAL HOTEL NO EXISTE,
                    POR FAVOR INTRODUZCA UN HOTEL EXISTENTE EN NUESTRA BASE DE DATOS');
                END IF;
            END IF;
        END IF;
    END IF;
END;
```

Procedimiento dedicado para dar de alta a un nuevo cliente en la cadena hotelera denominado **ALTACLIENTE.**

El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ En último lugar se inserta el cliente de manera duplicada en todas las localidades.

```
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 GBDD1.CLIENTE1 VALUES(CODIGO_CLIENTE, DNI, NOMBRE, TELEFONO);

    INSERT INTO GBDD2.CLIENTE2 VALUES(CODIGO_CLIENTE, DNI, NOMBRE, TELEFONO);

    INSERT INTO GBDD3.CLIENTE3 VALUES(CODIGO_CLIENTE, DNI, NOMBRE, TELEFONO);

    INSERT INTO GBDD4.CLIENTE4 VALUES(CODIGO_CLIENTE, DNI, NOMBRE, TELEFONO);

END;
```

Procedimiento dedicado para insertar o actualizar una reserva ya existente denominado **INSERTARACTUALIZARRESERVA.** La diferencia entre insertar o actualizar una reserva ya existente está en que si se le pasa las fechas antiguas a una reserva se interpretará como que se quiere actualizar la reserva, en caso de que se inserten ambas fechas a **NULL** se hará una reserva nueva.
El procedimiento consta de los siguientes pasos:

- En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- En segundo lugar se comprueba que la fecha de inicio de reserva nueva no sea mayor a la fecha final de reserva nueva. En caso de no ser así se lanzará una excepción.

- En tercer lugar se comprueba en que localidad está almacenado irá almacenado el registro de la nueva reserva, en caso de que el hotel no exista en nuestra base de datos se lanzará una excepción explicando que ese hotel no existe y que la reserva no puede llevarse a cabo.

- En cuarto lugar se comprueba si se va a actualizar una reserva anterior ya hecha o por lo contrario se va a realizar una nueva reserva observando si están a NULL las fechas de la reserva anterior. En caso de ser una actualización se guardarán los datos de la reserva anterior por si hubiera algún problema poder insertarla de nuevo y se eliminará la reserva anterior .

- En quinto lugar se comprueba que la reserva nueva no interfiere con alguna reserva ya hecha por el mismo cliente en algún hotel. En caso de que interfiera se lanzará una excepción y se volverá a insertar la antigua reserva si se está actualizando.

❏ En sexto lugar se comprueba que el hotel en el que se quiere hacer una reserva tiene habitaciones libres del tipo que el cliente necesita. En caso contrario se lanzará una excepción y se volverá a insertar la antigua reserva si se está actualizando.

```
create or replace PROCEDURE
insertaractualizarReserva(CODIGO_CLIENTE_P NUMBER,
FECHA_INICIO_RESERVA_P DATE,FECHA_FINAL_RESERVA_P DATE,
TIPO_HABITACION_P VARCHAR, PRECIO_P FLOAT, CODIGO_HOTEL_P NUMBER,
FECHA_INICIO_RESERVA_ANTIGUA_P DATE, FECHA_FINAL_RESERVA_ANTIGUA_P DATE) IS

contar NUMBER;
contarCliente NUMBER;
contarClientesEnEsaFecha NUMBER;
contarClienteAnterior NUMBER;
habitacionesTotales NUMBER;

codigo_cliente NUMBER;
F_inicio DATE;
F_final DATE;
T_hab VARCHAR(50);
precio FLOAT;
codigo_hotel NUMBER;

actualizacion NUMBER;
localidadAnterior NUMBER;

BEGIN

    IF CODIGO_CLIENTE_P is NULL OR FECHA_INICIO_RESERVA_P is NULL OR FECHA_FINAL_RESERVA_P is NULL
    OR TIPO_HABITACION_P is NULL OR PRECIO_P is NULL OR CODIGO_HOTEL_P is NULL THEN
        raise_application_error(-20004,'NO SE PERMITEN VALORES A NULL,
        TAN SOLO SE PERMITEN PARA EL CASO EN EL QUE VAYAS A INSERTAR UNA RESERVA NUEVA
        CON LAS FECHAS DE INICIO Y FINAL ANTIGUAS, INTRODUCE VALORES VALIDOS');
    END IF;

    IF FECHA_INICIO_RESERVA_P > FECHA_FINAL_RESERVA_P THEN
        raise_application_error(-20004,'LA RESERVA TIENE UNA FECHA DE INICIO SUPERIOR A LA FECHA FINAL DE LA
RESERVA');
    END IF;

    SELECT COUNT(*) INTO contar FROM GBDD1.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

IF contar > 0 THEN
    localidadAnterior := 1;
ELSE
    SELECT COUNT(*) INTO contar FROM GBDD2.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

    IF contar > 0 THEN
    localidadAnterior := 2;
    ELSE
        SELECT COUNT(*) INTO contar FROM GBDD3.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

        IF contar > 0 THEN
            localidadAnterior := 3;
        ELSE
            SELECT COUNT(*) INTO contar FROM GBDD4.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;
            IF contar > 0 THEN
                localidadAnterior := 4;
            ELSE
                raise_application_error(-20004,'EL HOTEL NO EXISTE EN NUESTRA BASE DE DATOS,
                POR FAVOR INSERTE UN HOTEL VALIDO');
            END IF;
```

```
            END IF;
        END IF;
    END IF;

    IF (FECHA_INICIO_RESERVA_ANTIGUA_P is not NULL) AND( FECHA_FINAL_RESERVA_ANTIGUA_P is not NULL) THEN

        IF FECHA_INICIO_RESERVA_P = FECHA_INICIO_RESERVA_ANTIGUA_P
        AND FECHA_FINAL_RESERVA_P = FECHA_FINAL_RESERVA_ANTIGUA_P THEN
            raise_application_error(-20004,'NO SE HAN MODIFICADO LAS FECHAS PARA REALIZAR LA RESERVA, POR FAVOR
            INTRODUZCA UNAS FECHAS DIFERENTES A LAS YA EXISTENTES');
        END IF;


        actualizacion := 1;

        SELECT CODIGO_CLIENTE,FECHA_INICIO_RESERVA, FECHA_FINAL_RESERVA, TIPO_HABITACION, PRECIO_HABITACION,
CODIGO_HOTEL
        INTO codigo_cliente,F_inicio,  F_final, T_hab, precio, codigo_hotel
        FROM RESERVA
        WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
        AND FECHA_INICIO_RESERVA = FECHA_INICIO_RESERVA_ANTIGUA_P
        AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_ANTIGUA_P;

        IF codigo_hotel != CODIGO_HOTEL_P THEN
            raise_application_error(-20004,'NO SE PUEDE HACER UN CAMBIO DE RESERVA CON DISTINTO HOTEL,
            PARA ESO PRIMERO SE HA DE DAR DE BAJA LA RESERVA EXISTENTE EN ESTE HOTEL');
        END IF;

        IF localidadAnterior = 1 THEN
            DELETE FROM GBDD1.RESERVA1
            WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
            AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_ANTIGUA_P;

        ELSIF localidadAnterior = 2 THEN

            DELETE FROM GBDD2.RESERVA2
            WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
            AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_ANTIGUA_P;

        ELSIF localidadAnterior = 3 THEN

            DELETE FROM GBDD3.RESERVA3
            WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
            AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_ANTIGUA_P;

        ELSIF localidadAnterior = 4 THEN

            DELETE FROM GBDD4.RESERVA4
            WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
            AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_ANTIGUA_P;
        END IF;

    ELSE

        actualizacion := 0;

    END IF;

      SELECT COUNT(*) INTO contarClienteAnterior FROM RESERVA
    WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
    AND (
            (FECHA_FINAL_RESERVA >= FECHA_FINAL_RESERVA_P
            AND FECHA_INICIO_RESERVA <= FECHA_FINAL_RESERVA_P)
            OR
            (FECHA_INICIO_RESERVA <= FECHA_INICIO_RESERVA_P
            AND FECHA_FINAL_RESERVA >= FECHA_INICIO_RESERVA_P)
            OR
            (FECHA_INICIO_RESERVA >= FECHA_INICIO_RESERVA_P
            AND FECHA_FINAL_RESERVA <= FECHA_FINAL_RESERVA_P)
```

```
      );

IF contarClienteAnterior = 0 THEN

SELECT COUNT(*) INTO contarClientesEnEsaFecha
  FROM RESERVA
  WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
  AND(
      (FECHA_FINAL_RESERVA >= FECHA_FINAL_RESERVA_P
      AND FECHA_INICIO_RESERVA <= FECHA_FINAL_RESERVA_P)
      OR
      (FECHA_INICIO_RESERVA <= FECHA_INICIO_RESERVA_P
      AND FECHA_FINAL_RESERVA >= FECHA_INICIO_RESERVA_P)
      OR
      (FECHA_INICIO_RESERVA >= FECHA_INICIO_RESERVA_P
      AND FECHA_FINAL_RESERVA <= FECHA_FINAL_RESERVA_P)
      )
  AND TIPO_HABITACION = TIPO_HABITACION_P;

IF TIPO_HABITACION_P = 'Doble' THEN
          SELECT NUMERO_HABITACIONES_DOBLE INTO habitacionesTotales FROM HOTEL WHERE COD_HOTEL =
CODIGO_HOTEL_P;

      ELSIF TIPO_HABITACION_P = 'Sencilla'THEN
          SELECT NUMERO_HABITACIONES_SIMPLE INTO habitacionesTotales FROM HOTEL WHERE COD_HOTEL =
CODIGO_HOTEL_P;

        ELSE
          raise_application_error(-20004,'EL HOTEL NO TIENE HABITACIONES DE ESTE TIPO, PRUEBA CON Doble O
Sencilla');
      END IF;

        IF habitacionesTotales = 0 THEN

          IF actualizacion = 1 THEN
              IF localidadAnterior = 1 THEN
                  INSERT INTO GBDD1.RESERVA1 VALUES(F_inicio, F_final,
                  T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

              ELSIF localidadAnterior = 2 THEN
                  INSERT INTO GBDD2.RESERVA2 VALUES(F_inicio, F_final,
                  T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

              ELSIF localidadAnterior = 3 THEN
                  INSERT INTO GBDD3.RESERVA3 VALUES(F_inicio, F_final,
                  T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

          ELSIF localidadAnterior = 4 THEN
              INSERT INTO GBDD4.RESERVA4 VALUES(F_inicio, F_final,
              T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);
          END IF;
      END IF;
      raise_application_error(-20004,'EL HOTEL NO TIENE HABITACIONES DE ESTE TIPO');

    END IF;

    IF (habitacionesTotales - contarClientesEnEsaFecha) > 0 THEN

      IF localidadAnterior = 1 THEN
          INSERT INTO GBDD1.RESERVA1 VALUES(FECHA_INICIO_RESERVA_P, FECHA_FINAL_RESERVA_P,
            TIPO_HABITACION_P, PRECIO_P, CODIGO_HOTEL_P, CODIGO_CLIENTE_P);

      ELSIF localidadAnterior = 2 THEN
          INSERT INTO GBDD2.RESERVA2 VALUES(FECHA_INICIO_RESERVA_P, FECHA_FINAL_RESERVA_P,
            TIPO_HABITACION_P, PRECIO_P, CODIGO_HOTEL_P, CODIGO_CLIENTE_P);

      ELSIF localidadAnterior = 3 THEN
          INSERT INTO GBDD3.RESERVA3 VALUES(FECHA_INICIO_RESERVA_P, FECHA_FINAL_RESERVA_P,
```

```
            TIPO_HABITACION_P, PRECIO_P, CODIGO_HOTEL_P, CODIGO_CLIENTE_P);

    ELSIF localidadAnterior = 4 THEN
        INSERT INTO GBDD4.RESERVA4 VALUES(FECHA_INICIO_RESERVA_P, FECHA_FINAL_RESERVA_P,
          TIPO_HABITACION_P, PRECIO_P, CODIGO_HOTEL_P, CODIGO_CLIENTE_P);
    END IF;

ELSE
    IF actualizacion = 1 THEN
        IF localidadAnterior = 1 THEN
            INSERT INTO GBDD1.RESERVA1 VALUES(F_inicio, F_final,
            T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

        ELSIF localidadAnterior = 2 THEN
            INSERT INTO GBDD2.RESERVA2 VALUES(F_inicio, F_final,
            T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

        ELSIF localidadAnterior = 3 THEN
            INSERT INTO GBDD3.RESERVA3 VALUES(F_inicio, F_final,
            T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

            ELSIF localidadAnterior = 4 THEN
                INSERT INTO GBDD4.RESERVA4 VALUES(F_inicio, F_final,
                T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);
            END IF;
        END IF;
        raise_application_error(-20004,'EL HOTEL ESTÁ OCUPADO PARA ESTA FECHA');
  END IF;
ELSE
    IF actualizacion = 1 THEN

        IF localidadAnterior = 1 THEN
            INSERT INTO GBDD1.RESERVA1 VALUES(F_inicio, F_final,
            T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

        ELSIF localidadAnterior = 2 THEN
            INSERT INTO GBDD2.RESERVA2 VALUES(F_inicio, F_final,
            T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

        ELSIF localidadAnterior = 3 THEN
            INSERT INTO GBDD3.RESERVA3 VALUES(F_inicio, F_final,
            T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);

        ELSIF localidadAnterior = 4 THEN
            INSERT INTO GBDD4.RESERVA4 VALUES(F_inicio, F_final,
            T_hab, precio, codigo_hotel, CODIGO_CLIENTE_P);
        END IF;
          END IF;
          raise_application_error(-20004,'EL CLIENTE YA TIENE UNA RESERVA EN ESTAS FECHAS');
      END IF;
END;
```

Procedimiento dedicado para dar de baja una reserva existente de un cliente denominado **DARBAJARESERVA.**

El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ En segundo lugar se comprueba que el cliente exista, en caso de que no sea así se lanzará una excepción.

- ❏ En tercer lugar se comprueba que el cliente tiene reserva hechas, en caso de que no sea así se lanzará una excepción.

- ❏ Por último se comprueba en que localidad estaba guardado el registro de la reserva para eliminarlo, si el código del hotel dado por parámetro no corresponde a un hotel existente se lanzará una excepción.

```
create or replace PROCEDURE
darBajaReserva(CODIGO_CLIENTE_P NUMBER,
FECHA_INICIO_RESERVA_P DATE,
FECHA_FINAL_RESERVA_P DATE,
CODIGO_HOTEL_P NUMBER) IS

contarReservas NUMBER;
contar NUMBER;
contarCliente NUMBER;

BEGIN

    IF CODIGO_CLIENTE_P is NULL OR FECHA_INICIO_RESERVA_P is NULL OR FECHA_FINAL_RESERVA_P is NULL OR
CODIGO_HOTEL_P 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 CLIENTE WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P;

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

SELECT COUNT(*) INTO contarReservas FROM RESERVA WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
AND FECHA_INICIO_RESERVA = FECHA_INICIO_RESERVA_P AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_P;

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

SELECT COUNT(*) INTO contar FROM GBDD1.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

IF contar > 0 THEN
    DELETE FROM GBDD1.RESERVA1
    WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
    AND FECHA_INICIO_RESERVA = FECHA_INICIO_RESERVA_P
    AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_P;
ELSE
    SELECT COUNT(*) INTO contar FROM GBDD2.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

    IF contar > 0 THEN
        DELETE FROM GBDD2.RESERVA2
        WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
        AND FECHA_INICIO_RESERVA = FECHA_INICIO_RESERVA_P
        AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_P;
    ELSE
        SELECT COUNT(*) INTO contar FROM GBDD3.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

        IF contar > 0 THEN
            DELETE FROM GBDD3.RESERVA3
            WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
            AND FECHA_INICIO_RESERVA = FECHA_INICIO_RESERVA_P
            AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_P;
        ELSE
            SELECT COUNT(*) INTO contar FROM GBDD4.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;
            IF contar > 0 THEN
                DELETE FROM GBDD4.RESERVA4
                WHERE CODIGO_CLIENTE = CODIGO_CLIENTE_P
                AND FECHA_INICIO_RESERVA = FECHA_INICIO_RESERVA_P
                AND FECHA_FINAL_RESERVA = FECHA_FINAL_RESERVA_P;
            ELSE
                raise_application_error(-20004, 'TAL HOTEL NO EXISTE,
                POR FAVOR INTRODUZCA UN HOTEL EXISTENTE EN NUESTRA BASE DE DATOS');
            END IF;
        END IF;
    END IF;
END IF;
```

```
END;
```

Procedimiento dedicado para la inserción de un nuevo proveedor denominado **ALTAPROVEEDOR.**

El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ En segundo lugar se comprueba que el proveedor nuevo no exista ya, en caso de que no sea así se lanzará una excepción.

- ❏ Y en último lugar se comprueba en que localidad irá el nuevo proveedor dependiendo de su provincia, en caso de que su provincia no sea correcta se lanzará una excepción.

```
create or replace PROCEDURE
altaProveedor(CODIGO_PROVEEDOR_P NUMBER, NOMBRE VARCHAR, CIUDAD VARCHAR, PROVINCIA VARCHAR) IS

contarProveedor NUMBER;

BEGIN

    IF CODIGO_PROVEEDOR_P is NULL OR NOMBRE is NULL OR CIUDAD is NULL OR PROVINCIA 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 PROVINCIA = 'Granada' THEN
        INSERT INTO GBDD1.PROVEEDOR1 VALUES(CODIGO_PROVEEDOR_P, NOMBRE, CIUDAD, PROVINCIA);
    ELSIF PROVINCIA = 'Sevilla' THEN
        INSERT INTO GBDD3.PROVEEDOR3 VALUES(CODIGO_PROVEEDOR_P, NOMBRE, CIUDAD, PROVINCIA);
    ELSE
        raise_application_error(-20004,'EL PROVEEDOR A INSERTAR NO TIENE UNA PROVINCIA VALIDA');
    END IF;
END;
```

Procedimiento dedicado para la eliminación de un proveedor ya existente denominado **BAJAPROVEEDOR.** El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ En segundo lugar se comprueba en que localidad está guardado ese proveedor, una vez tenemos la localidad, vemos si tiene suministros hechos, en caso contrario borramos los datos correspondientes del proveedor, en caso de tener suministros pasamos a ver si esos suministros son con cantidades mayores que 0, en caso de ser así se lanza una excepción, en caso contrario se pasa a eliminar los registros correspondientes del proveedor.

- ❏ En último lugar decir que si el proveedor que se inserta no existe se lanzará una excepción.

```
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 GBDD1.PROVEEDOR1 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;

  DELETE FROM GBDD1.SUMINISTRO1 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

  DELETE FROM GBDD2.SUMINISTRO2 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

  DELETE FROM GBDD3.SUMINISTRO3 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

  DELETE FROM GBDD4.SUMINISTRO4 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

  DELETE FROM GBDD1.ARTICULO_VENDE WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

  DELETE FROM GBDD1.PROVEEDOR1 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
ELSE

  SELECT COUNT(*) INTO contar FROM GBDD3.PROVEEDOR3 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;


DELETE FROM GBDD1.SUMINISTRO1 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

DELETE FROM GBDD2.SUMINISTRO2 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

DELETE FROM GBDD3.SUMINISTRO3 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

DELETE FROM GBDD4.SUMINISTRO4 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;
```

```
      DELETE FROM GBDD3.ARTICULO_VENDE WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

      DELETE FROM GBDD3.PROVEEDOR3 WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

    ELSE
      raise_application_error(-20004,'EL PROVEEDOR A ELIMINAR NO EXISTE');
    END IF;
  END IF;
END;
```

Procedimiento dedicado para dar de alta o actualizar un suministro ya existente denominado **ALTAACTUALIZARSUMINISTRO.** La diferencia entre insertar un suministro nuevo y actualizar uno existente está en que si existe un suministro en esa fecha hacia el mismo hotel y por el mismo proveedor será una actualización, en caso contrario una inserción nueva.
El procedimiento consta de los siguientes pasos:

- En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- En segundo lugar se comprueba que el precio por unidad no sea menor que 1, después se comprueba que la cantidad no sea inferior a -1 (Será -1 en el caso de que se cancele el pedido.), después se comprobará que el hotel, el artículo y el proveedor existen, también se comprobará que el proveedor provee dicho artículo, se comprobará que el precio del nuevo suministro no sea inferior a antiguos suministros para ese hotel con ese artículo, también se comprobará que el proveedor provee a hoteles de su zona. En caso de que estas cosas no se cumplan se lanzará una excepción correspondiente a la negligencia cometida.

- Tras esto anterior se comprobará en que localidad hay que insertar o actualizar el suministro, y después se comprobará si es una inserción nueva o una actualización.

- En caso de que sea una actualización y la cantidad sea -1 simplemente se pondrá a -1 la cantidad, en caso de que la cantidad sea mayor que -1 y la cantidad anterior que hubiese sea -1 se pone la nueva cantidad y en caso de que la nueva cantidad sea mayor que -1 y la vieje mayor que -1 se suman ambas cantidades y se guarda. Para el caso en el que sea una nueva inserción, simplemente se inserta el nuevo registro.

```
create or replace PROCEDURE
altaActualizarSuministro(CODIGO_ARTICULO_P NUMBER,
CODIGO_PROVEEDOR_P NUMBER, CODIGO_HOTEL_P NUMBER, FECHA_SUMINSITRO_P DATE,
CANTIDAD_P NUMBER, PRECIO_UNIDAD_P FLOAT) IS

contar NUMBER;
existenciaSuministro NUMBER;
cantidadAnterior NUMBER;
contarProveedorArticulo NUMBER;
contarArticulo NUMBER;
contarProveedor NUMBER;
contarHotel NUMBER;
maximoPrecioAnterior NUMBER;

provinciaProveedor VARCHAR(20);
provinciaHotel VARCHAR(20);

BEGIN

  IF CODIGO_ARTICULO_P is NULL OR CODIGO_PROVEEDOR_P is NULL OR CODIGO_HOTEL_P is NULL
  OR FECHA_SUMINSITRO_P is NULL OR CANTIDAD_P is NULL OR PRECIO_UNIDAD_P is NULL THEN
    raise_application_error(-20004,'NO SE PERMITE QUE SE INSERTEN VALORES COMO NULL,
    LOS VALORES DEBEN DE SER VALIDOS');
  END IF;

  IF PRECIO_UNIDAD_P < 1 THEN
    raise_application_error(-20004,'NO SE PERMITE REALIZAR UN SUMINISTRO CON EL PRECIO INDICADO, DEBE DE PONER UN PRECIO
MAYOR O IGUAL A 1 COMO MINIMO');
  END IF;

  IF CANTIDAD_P < -1 THEN
    raise_application_error(-20004,'NO SE PERMITE PONER UNA CANTIDAD INFERIOR A 0, Y EN EL CASO DE QUE SE QUIERA DEVOLVER EL
SUMINISTRO, DEBE DE SER IGUAL A -1');
  END IF;

  SELECT COUNT(*) INTO contarHotel FROM HOTEL
  WHERE COD_HOTEL = CODIGO_HOTEL_P;

IF contarHotel = 0 THEN
  raise_application_error(-20004,'EL HOTEL NO EXISTE');
END IF;

SELECT COUNT(*) INTO contarArticulo FROM ARTICULO
WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

IF contarArticulo = 0 THEN
  raise_application_error(-20004,'EL ARTICULO NO EXISTE');
END IF;

SELECT COUNT(*) INTO contarProveedor FROM PROVEEDOR
WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

IF contarProveedor = 0 THEN
  raise_application_error(-20004,'EL PROVEEDOR NO EXISTE');
END IF;

SELECT COUNT(*) INTO contarProveedorArticulo FROM ARTICULO
WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
AND CODIGO_ARTICULO = CODIGO_ARTICULO_P;
```

```
IF contarProveedorArticulo = 0 THEN
   raise_application_error(-20004,'EL PROVEEDOR QUE SE HA INSERTADO NO PROVEE TAL ARTICULO');
END IF;

SELECT MAX(PRECIO_POR_UNIDAD) INTO maximoPrecioAnterior
FROM SUMINISTRO
WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
AND CODIGO_ARTICULO = CODIGO_ARTICULO_P;

IF maximoPrecioAnterior > PRECIO_UNIDAD_P THEN
   raise_application_error(-20004,'EL PRECIO DEL NUEVO SUMINISTRO NO PUEDE SER INFERIOR AL DE ANTIGUOS SUMINISTROS');
END IF;

SELECT PROVINCIA INTO provinciaProveedor
FROM PROVEEDOR
WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P;

SELECT PROVINCIA INTO provinciaHotel
FROM HOTEL
WHERE COD_HOTEL = CODIGO_HOTEL_P;

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

ELSIF provinciaProveedor = 'Granada' AND
(provinciaHotel = 'Sevilla' OR provinciaHotel = 'Cadiz' OR provinciaHotel = 'Cordoba' OR provinciaHotel = 'Huelva')
THEN
   raise_application_error(-20004,'EL PROVEEDOR DE LA PROVINCIA DE GRANADA NO PUEDE PROVEER A UN
   HOTEL DE LAS PROVINCIAS DE SEVILLA, HUELVA, CADIZ O CORDOBA');
END IF;

SELECT COUNT(*) INTO contar FROM GBDD1.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

IF contar > 0 THEN

   SELECT COUNT(*) INTO existenciaSuministro FROM GBDD1.SUMINISTRO1
   WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P
   AND CODIGO_HOTEL = CODIGO_HOTEL_P
   AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
   AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;

   IF existenciaSuministro > 0 THEN

      IF CANTIDAD_P < 0 THEN
   UPDATE GBDD1.SUMINISTRO1
   SET CANTIDAD = CANTIDAD_P,
   PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
   WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
   AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
   AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
ELSE
   SELECT CANTIDAD INTO cantidadAnterior FROM GBDD1.SUMINISTRO1 WHERE
   CODIGO_ARTICULO = CODIGO_ARTICULO_P
   AND CODIGO_HOTEL = CODIGO_HOTEL_P
   AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
   AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;

   IF cantidadAnterior > 0 THEN
      UPDATE GBDD1.SUMINISTRO1 SET CANTIDAD = cantidadAnterior + CANTIDAD_P
      ,PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
      WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
      AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
      AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
   ELSE
      UPDATE GBDD1.SUMINISTRO1 SET CANTIDAD = CANTIDAD_P
```

```
          ,PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
          WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
          AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
          AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
        END IF;
      END IF;
    ELSE
      INSERT INTO GBDD1.SUMINISTRO1
      VALUES(CODIGO_ARTICULO_P, CODIGO_HOTEL_P, CODIGO_PROVEEDOR_P, FECHA_SUMINSITRO_P,
      CANTIDAD_P, PRECIO_UNIDAD_P);
    END IF;

  ELSE
    SELECT COUNT(*) INTO contar FROM GBDD2.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

    IF contar > 0 THEN
      SELECT COUNT(*) INTO existenciaSuministro FROM GBDD2.SUMINISTRO2 WHERE
      CODIGO_ARTICULO = CODIGO_ARTICULO_P
      AND CODIGO_HOTEL = CODIGO_HOTEL_P
      AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
      AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;

    IF existenciaSuministro > 0 THEN

      IF CANTIDAD_P < 0 THEN
        UPDATE GBDD2.SUMINISTRO2
        SET CANTIDAD = CANTIDAD_P,
        PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
        WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
        AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
        AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
      ELSE
        SELECT CANTIDAD INTO cantidadAnterior FROM GBDD2.SUMINISTRO2 WHERE
        CODIGO_ARTICULO = CODIGO_ARTICULO_P
        AND CODIGO_HOTEL = CODIGO_HOTEL_P
        AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
        AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;

        IF cantidadAnterior > 0 THEN
          UPDATE GBDD2.SUMINISTRO2 SET CANTIDAD = cantidadAnterior + CANTIDAD_P
          ,PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
          WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
          AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
          AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
        ELSE
          UPDATE GBDD2.SUMINISTRO2 SET CANTIDAD = CANTIDAD_P
          ,PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
          WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
          AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
          AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
        END IF;
      END IF;
    ELSE
      INSERT INTO GBDD2.SUMINISTRO2
      VALUES(CODIGO_ARTICULO_P, CODIGO_HOTEL_P, CODIGO_PROVEEDOR_P, FECHA_SUMINSITRO_P,
      CANTIDAD_P, PRECIO_UNIDAD_P);
    END IF;
  ELSE
    SELECT COUNT(*) INTO contar FROM GBDD3.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

    IF contar > 0 THEN
      SELECT COUNT(*) INTO existenciaSuministro FROM GBDD3.SUMINISTRO3 WHERE
      CODIGO_ARTICULO = CODIGO_ARTICULO_P
      AND CODIGO_HOTEL = CODIGO_HOTEL_P
      AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
      AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;

      IF existenciaSuministro > 0 THEN
```

```
IF CANTIDAD_P < 0 THEN
  UPDATE GBDD3.SUMINISTRO3
  SET CANTIDAD = CANTIDAD_P,
  PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
  WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
  AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
  AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
ELSE
  SELECT CANTIDAD INTO cantidadAnterior FROM GBDD3.SUMINISTRO3 WHERE
  CODIGO_ARTICULO = CODIGO_ARTICULO_P
  AND CODIGO_HOTEL = CODIGO_HOTEL_P
  AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
  AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;

  IF cantidadAnterior > 0 THEN
    UPDATE GBDD3.SUMINISTRO3 SET CANTIDAD = cantidadAnterior + CANTIDAD_P
    ,PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
    WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
    AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
    AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
  ELSE
      UPDATE GBDD3.SUMINISTRO3 SET CANTIDAD = CANTIDAD_P
      ,PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
      WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
      AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
      AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
    END IF;
  END IF;

  ELSE
    INSERT INTO GBDD3.SUMINISTRO3
    VALUES(CODIGO_ARTICULO_P, CODIGO_HOTEL_P, CODIGO_PROVEEDOR_P, FECHA_SUMINSITRO_P,
    CANTIDAD_P, PRECIO_UNIDAD_P);
  END IF;
ELSE
  SELECT COUNT(*) INTO contar FROM GBDD4.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;
  IF contar > 0 THEN
    SELECT COUNT(*) INTO existenciaSuministro FROM GBDD4.SUMINISTRO4 WHERE
    CODIGO_ARTICULO = CODIGO_ARTICULO_P
    AND CODIGO_HOTEL = CODIGO_HOTEL_P
    AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
    AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;

    IF existenciaSuministro > 0 THEN

      IF CANTIDAD_P < 0 THEN
        UPDATE GBDD4.SUMINISTRO4
        SET CANTIDAD = CANTIDAD_P,
        PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
        WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
        AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
        AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
      ELSE
        SELECT CANTIDAD INTO cantidadAnterior FROM GBDD4.SUMINISTRO4 WHERE
        CODIGO_ARTICULO = CODIGO_ARTICULO_P
        AND CODIGO_HOTEL = CODIGO_HOTEL_P
        AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
        AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;

        IF cantidadAnterior > 0 THEN
          UPDATE GBDD4.SUMINISTRO4 SET CANTIDAD = cantidadAnterior + CANTIDAD_P
          ,PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
          WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
          AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
          AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
        ELSE
          UPDATE GBDD4.SUMINISTRO4 SET CANTIDAD = CANTIDAD_P
          ,PRECIO_POR_UNIDAD = PRECIO_UNIDAD_P
          WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
```

```
                AND CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
                AND FECHA_SUMINISTRO = FECHA_SUMINSITRO_P;
              END IF;
            END IF;

          ELSE
            INSERT INTO GBDD4.SUMINISTRO4
            VALUES(CODIGO_ARTICULO_P, CODIGO_HOTEL_P, CODIGO_PROVEEDOR_P, FECHA_SUMINSITRO_P,
            CANTIDAD_P, PRECIO_UNIDAD_P);
          END IF;
        END IF;
      END IF;
    END IF;
  END IF;
END;
```

Procedimiento dedicado para dar de baja un suministro ya existente denominado **BAJASUMINISTRO.**

El procedimiento consta de los siguientes pasos:

- En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- En segundo lugar se comprueba que tanto el hotel como el artículo existan, en caso contrario se lanza una excepción.

- Después se comprueba si se quiere eliminar un suministro en una fecha concreta o de lo contrario se quieren eliminar todos los suministros en donde tengan que ver el hotel y el artículo indicado.

- Si es en una fecha concreta, primero se comprueba que existe tal suministro, y en caso contrario se lanza una excepción, si existe tal suministro se elimina de la localidad correspondiente.

- Si es los suministros sin importar la fecha, primero se comprueba que se han hecho suministros hacia el hotel con el artículo indicado, en caso contrario se lanza una excepción. Si se han hecho los suministros se eliminan de la localidad correspondiente.

```
create or replace PROCEDURE
bajaSuministro(CODIGO_HOTEL_P NUMBER, CODIGO_ARTICULO_P NUMBER, FECHA_SUMINISTRO_P DATE) IS

contar NUMBER;
contarHotel NUMBER;
contarArticulo NUMBER;
contarSuministrosEsaFecha NUMBER;
contarSuministros NUMBER;

BEGIN
    IF CODIGO_HOTEL_P is NULL OR CODIGO_ARTICULO_P is NULL THEN
```

```
        raise_application_error(-20004,'EL UNICO VALOR QUE SE PERMITE QUE ESTE A NULL, ES LA FECHA DEL
SUMINISTRO,
        LOS DEMAS DEBEN SER VALORES CORRECTOS');
    END IF;

    SELECT COUNT(*) INTO contarHotel FROM HOTEL WHERE COD_HOTEL = CODIGO_HOTEL_P;

    IF contarHotel = 0 THEN
        raise_application_error(-20004,'EL HOTEL NO EXISTE');
    END IF;

    SELECT COUNT(*) INTO contarArticulo FROM ARTICULO WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

    IF contarArticulo = 0 THEN
        raise_application_error(-20004,'EL ARTICULO NO EXISTE');
    END IF;

    IF (FECHA_SUMINISTRO_P is not NULL) THEN

        SELECT COUNT(*) INTO contarSuministrosEsaFecha
        FROM SUMINISTRO
        WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P
        AND CODIGO_HOTEL = CODIGO_HOTEL_P
        AND FECHA_SUMINISTRO = FECHA_SUMINISTRO_P;

        IF contarSuministrosEsaFecha = 0 THEN
            raise_application_error(-20004,'NO EXISTE NINGUN SUMINISTRO REALIZADO EN ESA FECHA');
        END IF;

        SELECT COUNT(*) INTO contar FROM GBDD1.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

        IF contar > 0 THEN
            DELETE FROM GBDD1.SUMINISTRO1
            WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
            AND CODIGO_ARTICULO = CODIGO_ARTICULO_P
            AND FECHA_SUMINISTRO = FECHA_SUMINISTRO_P;
        ELSE
            SELECT COUNT(*) INTO contar FROM GBDD2.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

            IF contar > 0 THEN
                DELETE FROM GBDD2.SUMINISTRO2
                WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
                AND CODIGO_ARTICULO = CODIGO_ARTICULO_P
                AND FECHA_SUMINISTRO = FECHA_SUMINISTRO_P;
            ELSE
                SELECT COUNT(*) INTO contar FROM GBDD3.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

                IF contar > 0 THEN
                    DELETE FROM GBDD3.SUMINISTRO3
                    WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
                    AND CODIGO_ARTICULO = CODIGO_ARTICULO_P
                    AND FECHA_SUMINISTRO = FECHA_SUMINISTRO_P;
                ELSE
                    SELECT COUNT(*) INTO contar FROM GBDD4.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;
                    IF contar > 0 THEN
                        DELETE FROM GBDD4.SUMINISTRO4
                        WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
                        AND CODIGO_ARTICULO = CODIGO_ARTICULO_P
                        AND FECHA_SUMINISTRO = FECHA_SUMINISTRO_P;
                    END IF;
                END IF;
            END IF;
        END IF;

    ELSE


        SELECT COUNT(*) INTO contarSuministros
        FROM SUMINISTRO
```

```
        WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P
        AND CODIGO_HOTEL = CODIGO_HOTEL_P;

        IF contarSuministros = 0 THEN
            raise_application_error(-20004,'NO EXISTE NINGUN SUMINISTRO REALIZADO A ESTE HOTEL CON
            ESTE ARTICULO');
        END IF;

        SELECT COUNT(*) INTO contar FROM GBDD1.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

        IF contar > 0 THEN
            DELETE FROM GBDD1.SUMINISTRO1
            WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
            AND CODIGO_ARTICULO = CODIGO_ARTICULO_P;
        ELSE
            SELECT COUNT(*) INTO contar FROM GBDD2.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

            IF contar > 0 THEN
                DELETE FROM GBDD2.SUMINISTRO2
                WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
                AND CODIGO_ARTICULO = CODIGO_ARTICULO_P;
            ELSE
                SELECT COUNT(*) INTO contar FROM GBDD3.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;

                IF contar > 0 THEN
                    DELETE FROM GBDD3.SUMINISTRO3
                    WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
                    AND CODIGO_ARTICULO = CODIGO_ARTICULO_P;
                ELSE
                    SELECT COUNT(*) INTO contar FROM GBDD4.HOTEL_DIRIGE WHERE COD_HOTEL = CODIGO_HOTEL_P;
                    IF contar > 0 THEN
                        DELETE FROM GBDD4.SUMINISTRO4
                        WHERE CODIGO_HOTEL = CODIGO_HOTEL_P
                        AND CODIGO_ARTICULO = CODIGO_ARTICULO_P;
                    END IF;
                END IF;
            END IF;
        END IF;
    END IF;
END;
```

Procedimiento dedicado para dar de alta un nuevo artículo con el proveedor que lo provee denominado **ALTAARTICULO.**

El procedimiento consta de los siguientes pasos:

- ❏ En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- ❏ Se comprueba en que localidad se va a insertar el nuevo artículo dependiendo del proveedor.

- ❏ Si el proveedor no es válido se lanzará una excepción indicándolo.

```
create or replace PROCEDURE
altaArticulo(CODIGO_ARTICULO NUMBER, NOMBRE VARCHAR, TIPO VARCHAR, CODIGO_PROVEEDOR_P NUMBER) IS
```

```
contarProveedor NUMBER;

BEGIN

    IF CODIGO_ARTICULO is NULL OR  NOMBRE is NULL OR TIPO is NULL OR CODIGO_PROVEEDOR_P is NULL THEN
        raise_application_error(-20004,'NO SE PERMITEN DATOS DE ENTRADA A NULL, DEBEN DE SER DATOS VALIDOS');
    END IF;

    SELECT COUNT(*) INTO contarProveedor
    FROM PROVEEDOR
    WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
    AND PROVINCIA = 'Granada';

    IF contarProveedor > 0 THEN

        INSERT INTO GBDD1.ARTICULO_VENDE VALUES(CODIGO_ARTICULO, NOMBRE, TIPO, CODIGO_PROVEEDOR_P);

    ELSE
        SELECT COUNT(*) INTO contarProveedor
        FROM PROVEEDOR
        WHERE CODIGO_PROVEEDOR = CODIGO_PROVEEDOR_P
        AND PROVINCIA = 'Sevilla';

        IF contarProveedor > 0 THEN

            INSERT INTO GBDD3.ARTICULO_VENDE VALUES(CODIGO_ARTICULO, NOMBRE, TIPO, CODIGO_PROVEEDOR_P);

        ELSE

            raise_application_error(-20004,'EL PROVEEDOR NO ES VALIDO');

        END IF;
    END IF;
END;
```

Procedimiento dedicado para dar de baja un artículo ya existente denominado **BAJAARTICULO.**

El procedimiento consta de los siguientes pasos:

- En primer lugar se comprueban que los datos insertados sean correctos, esto quiere decir que no sean **NULL.** En caso de que no sean correctos se lanzará una excepción indicándolo.

- Se comprueba que el artículo existe, en caso contrario se lanza una excepción.

- Se comprueba si se han hecho suministros con este artículo, en caso contrario se elimina directamente, si se han hecho suministros se comprueba que las cantidades suministradas no hayan sido mayor a 0, en caso contrario se lanza una excepción.

```
create or replace PROCEDURE
bajaArticulo(CODIGO_ARTICULO_P NUMBER) IS

num_suministro NUMBER;
cantidad_articulo_suministrado NUMBER;
contarArticulo NUMBER;
```

```
BEGIN

    IF CODIGO_ARTICULO_P is NULL THEN
        raise_application_error(-20004,'NO PUEDE PASARSE EL CODIGO DEL ARTICULO A ELIMINAR COMO NULL,
        POR FAVOR INSERTE UN CODIGO DE ARTICULO VALIDO');
    END IF;

    SELECT COUNT(*) INTO contarArticulo FROM ARTICULO WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

    IF contarArticulo = 0 THEN
        raise_application_error(-20004,'EL ARTICULO A ELIMINAR NO EXISTE');
    END IF;


    SELECT COUNT(*) INTO num_suministro FROM SUMINISTRO WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

    IF num_suministro > 0 THEN

        SELECT COUNT(*) into cantidad_articulo_suministrado FROM SUMINISTRO WHERE CANTIDAD > 0
        AND CODIGO_ARTICULO = CODIGO_ARTICULO_P;

        IF cantidad_articulo_suministrado = 0 THEN

            DELETE FROM GBDD1.SUMINISTRO1 WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

            DELETE FROM GBDD2.SUMINISTRO2 WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

            DELETE FROM GBDD3.SUMINISTRO3 WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

            DELETE FROM GBDD4.SUMINISTRO4 WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

            DELETE FROM GBDD1.ARTICULO_VENDE WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

            DELETE FROM GBDD3.ARTICULO_VENDE WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;
        ELSE
            raise_application_error(-20004,'EL ARTICULO NO SE PUEDE BORRAR YA QUE TIENE UNA CANTIDAD DE
            ARTICULO SUMINISTRADA MAYOR QUE CERO');
        END IF;
    ELSE
        DELETE FROM GBDD1.ARTICULO_VENDE WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;

        DELETE FROM GBDD3.ARTICULO_VENDE WHERE CODIGO_ARTICULO = CODIGO_ARTICULO_P;
    END IF;
END;
```

8. Implementación de consultas

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

```
SELECT HOTEL.NOMBRE AS NOMBRE_HOTEL,
HOTEL.CIUDAD AS CIUDAD_HOTEL,
PROVEEDOR.NOMBRE AS NOMBRE_PROVEEDOR,
PROVEEDOR.CIUDAD AS CIUDAD_PROVEEDOR,
SUMINISTRO.CODIGO_PROVEEDOR AS CODIGO_PROVEEDOR,
SUMINISTRO.CODIGO_HOTEL AS CODIGO_HOTEL,
SUMINISTRO.CODIGO_ARTICULO AS CODIGO_ARTICULO
FROM HOTEL, PROVEEDOR, SUMINISTRO, ARTICULO
WHERE SUMINISTRO.CODIGO_HOTEL = HOTEL.COD_HOTEL
AND SUMINISTRO.CODIGO_PROVEEDOR = PROVEEDOR.CODIGO_PROVEEDOR
AND SUMINISTRO.CODIGO_ARTICULO = ARTICULO.CODIGO_ARTICULO
AND ARTICULO.CODIGO_PROVEEDOR = PROVEEDOR.CODIGO_PROVEEDOR
AND (ARTICULO.NOMBRE = 'Queso' OR ARTICULO.NOMBRE = 'Mantequilla')
AND SUMINISTRO.FECHA_SUMINISTRO >= '12-May-2017'
AND SUMINISTRO.FECHA_SUMINISTRO <= '28-May-2017';
```

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 ,
SUMINISTRO.CANTIDAD AS CANTIDAD
FROM HOTEL, SUMINISTRO, ARTICULO
WHERE SUMINISTRO.CODIGO_HOTEL = HOTEL.COD_HOTEL
AND SUMINISTRO.CODIGO_PROVEEDOR = 4
AND SUMINISTRO.CODIGO_ARTICULO = ARTICULO.CODIGO_ARTICULO
AND ARTICULO.CODIGO_PROVEEDOR = 4
AND (HOTEL.PROVINCIA = 'Jaen' OR HOTEL.PROVINCIA = 'Almeria');
```

CABE DECIR QUE NO SABEMOS COMO INGRESAR EN UNA CONSULTA MYSQL POR TECLADO UN PARÁMETRO.

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 CLIENTE.NOMBRE AS NOMBRE_CLIENTE,
CLIENTE.TELEFONO AS TELEFONO_CLIENTE,
CLIENTE.CODIGO_CLIENTE AS CODIGO_CLIENTE
FROM CLIENTE, RESERVA
WHERE CLIENTE.CODIGO_CLIENTE = RESERVA.CODIGO_CLIENTE
AND RESERVA.CODIGO_HOTEL = 6
group by CLIENTE.NOMBRE, CLIENTE.TELEFONO, CLIENTE.CODIGO_CLIENTE
HAVING COUNT(RESERVA.CODIGO_CLIENTE) > 1;
```

CABE DECIR QUE NO SABEMOS COMO INGRESAR EN UNA CONSULTA MYSQL POR TECLADO UN PARÁMETRO.