UNIVERSITAT POLITÈCNICA DE VALÈNCIA

PIZZERÍA BUONA

BASES DE DATOS: Ciencia de Datos. Grupo 19

Curso 2022/2023

Índice

1. Primera Entrega

1.1. Enunciado Ampliado

Se ha ampliado el enunciado original con las partes en negrita:

La pizzería "Buona" ha comenzado a desarrollar su sistema de información para el reparto a domicilio y, tras una primera fase de análisis, se han obtenido los siguientes requisitos de información.

La pizzería tiene contratados empleados de los que conoce su dni, nombre, apellidos**, años que llevan trabajando en la pizzaría, edad (debe ser mayor de 16 años), si están en activo** y categoría.

La categoría del empleado puede ser cocinero, dependiente o repartidor. De los cocineros se quiere almacenar su titulación**, de cada dependiente los idiomas que habla** y de cada repartidor los tipos de permiso de conducción que posee (debe tener al menos uno).

La pizzería dispone de un parque de vehículos de reparto identificados por su matrícula, de los que **se conoce su capacidad (cantidad de pizzas que pueden llevar) y** posiblemente también se conozca su modelo y la fecha de la próxima ITV.

La pizzería elabora diferentes pizzas. Cada pizza tiene un número que la identifica, un nombre, su precio **y su tamaño; que puede ser pequeña, mediana o familiar**. Además, hay que almacenar qué ingredientes necesita para su elaboración.

Cada ingrediente tiene un número de referencia que es único para cada proveedor, además de un nombre y de un precio. Además, se desea conocer la cantidad que queda en la despensa de la pizzería de cada ingrediente. Los proveedores se pueden identificar a través de su NIF y disponen de un nombre y, frecuentemente, de un número de teléfono. **Cada vez que un proveedor entrega un suministro de ingredientes a la pizzería, se guarda la cantidad de cada ingrediente llevada, el día y la hora de la entrega y el dependiente responsable de la recepción de la mercancía**.

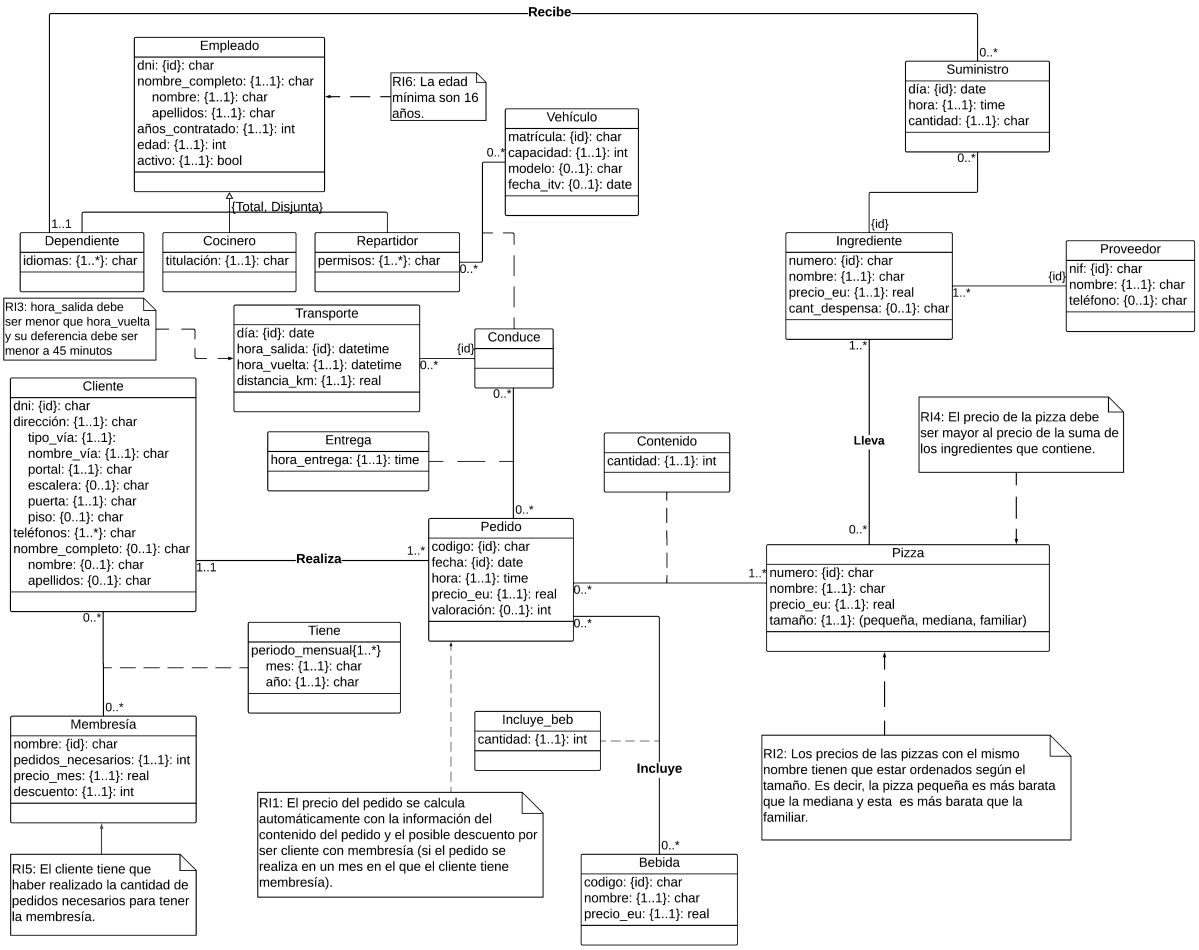
Cada cliente que realiza un pedido se identifica por su DNI y se desea poder almacenar también su nombre, apellidos, dirección y sus teléfonos de contacto, fijos y/o móviles (al menos uno). Cuando se realiza un pedido se le asigna un código de pedido que es único para cada día (fecha) y se debe almacenar también la hora del pedido, el cliente que lo realiza, el contenido del pedido (formado por el número de cada pizza solicitada y la cantidad de unidades de esa pizza que se desea) y el coste total del pedido que se calculará automáticamente a partir de los datos del pedido. **El pedido puede incluir también una bebida; cada bebida tiene un código que la identifica, un nombre y un precio.** Posteriormente se le asignará el pedido a uno o varios de los empleados repartidores (dependiendo del tamaño del pedido), que tomando cada uno de ellos un vehículo de los disponibles, realizarán la entrega del pedido. Se debe almacenar el vehículo en que cada repartidor ha hecho la entrega y la hora en que se ha entregado (está información se introducirá en el sistema cuando el repartidor vuelva a la pizzería).

Puede suceder que un repartidor salga de la pizzería con un vehículo para entregar varios pedidos seguidos sin tener que volver a la pizzería, por lo que cada vez que un repartidor coge un vehículo, se guardará el día y hora a la que sale de la pizzería y el día y hora a la que vuelve a la pizzería, guardando también la distancia recorrida en el trayecto en kilómetros. Los días de salida y llegada pueden ser diferentes si la entrega se produce a medianoche, ya que un repartidor no puede estar fuera de la pizzería con un vehículo más de 45 minutos (ida y vuelta), para que no se enfríen las pizzas. Podría suceder que un mismo repartidor coja el mismo transporte varias veces en un mismo día.

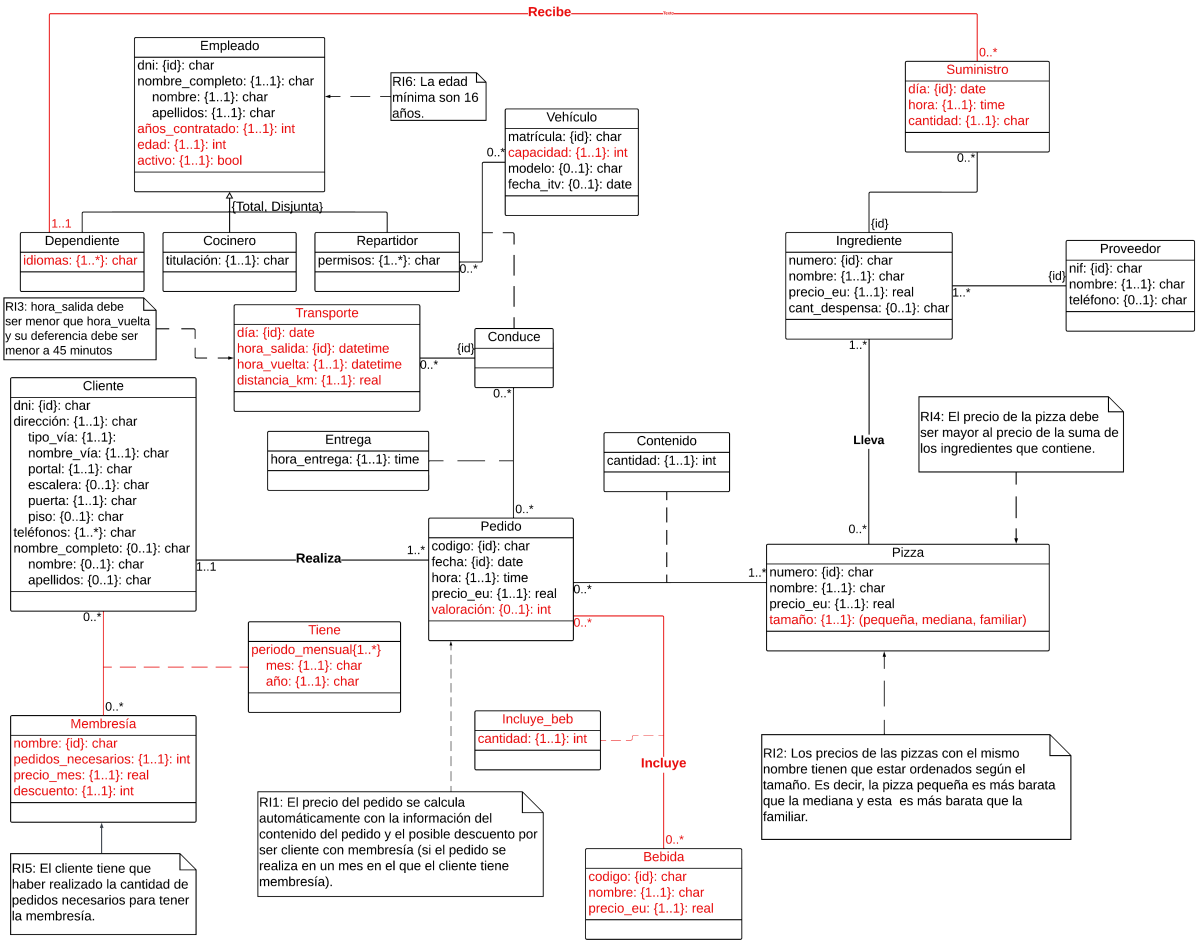
Una vez un cliente ha realizado al menos ciertos pedidos a la pizzería, tiene la opción de pagar una membresía mensual a cambio de un descuento en cada pedido que realice el cliente en ese mes. Cada membresía disponible se identifica con un nombre y se guarda los pedidos necesarios para poder desbloquearla, el precio mensual y el descuento que proporciona. Se guardará todos los meses en los que un cliente tuvo una membresía y de que tipo era.

Todos los precios que guardará la pizzería estarán en euros.

1.2. Diagrama de clases diseñado con el LucidChart



Aquí mostramos el diagrama destacando los cambios que hemos realizado en color rojo.



Link del diagrama en LucidChart: **Link a LucidChart**

1.3. RI no representadas en el diagrama

-RI1: El precio del pedido se calcula automáticamente con la información del contenido del pedido y el posible descuento por ser cliente con membresía (si el pedido se realiza en un mes en el que el cliente tiene membresía).

-RI2: Los precios de las pizzas con el mismo nombre tienen que estar ordenados según el tamaño. Es decir, la pizza pequeña es más barata que la mediana y esta más barata que la familiar.

-RI3: hora_salida debe ser menor que hora_vuelta y su diferencia debe ser menor a 45 minutos.

-RI4: El precio de la pizza debe ser mayor al precio de la suma de los ingredientes que contiene.

-RI5: El cliente tiene que haber realizado la cantidad de pedidos necesarios para tener la membresía.

-RI6: La edad mínima de los empleados son 16 años.

1.4. Lista de requisitos de proceso

-Dar de alta a un repartidor con un permiso de conducción A2, llamado 'Fernando Rodríguez García' con dni '48606924N' y 26 años.

-Dar de baja el vehículo con matrícula '5641CSX' modelo 'Opel Corsa'.

-Modificar el nombre completo de la empleada 'Aina Amelia Sánchez García' por 'Amelia Aina Sánchez García' con dni '24935804D'.

-Insertar una nueva pizza con nombre '7 quesos' con un precio_eu de 6, 8 y 12 euros, respectivamente a los tamaños y con números de código 'PI127', 'PI128', y 'PI129', respectivamente.

-Eliminar la bebida con código 'BE-06'.

-Borrar todas las pizzas que contengan piña.

1.5. Lista de consultas

-Obtener el DNI de los empleados los cuales no sean cocineros.

-Obtener el nombre, los apellidos y la dirección de los clientes que han valorado el pedido.

-Obtener el dni de los empleados, el nombre, los apellidos y la edad de los trabajadores que se hayan desplazado entre 20 y 25 km y que sean veteranos (3 años de contrato como mínimo)

-Obtener el número y el nombre de las pizzas cuyo precio sea mayor a 7 euros.

-Obtener el NIF y el nombre de los proveedores que no tengan asignado un teléfono.

-Obtener el DNI de los empleados, sus apellidos, la edad de estos y los años que llevan contratados, así como el total de empleados.

-Obtener el número de pizzas vendidas durante el puente de diciembre (5-10 de diciembre de este año 2022).

-Obtener el nombre, el DNI y la cantidad de suministro que han recibido los dependientes del proveedor con NIF 'E65830492'.

-Obtener los ingredientes que contiene cada pizza de manera individual y los NIF de los mismos.

-Obtener el código y el precio de los pedidos cuya distancia en el transporte haya sido mayor que 1,5 km ordenado por hora.

-Obtener los pedidos con una valoración de 4 (como mínimo) y que contenga al menos 5 pizzas iguales.

-Obtener la media de las valoraciones de los pedidos que han sido entregados en más de media hora.

-Obtener el nombre y la cantidad de despensa de los ingredientes que se implementen en más de una pizza.

-Obtener los tipos de pizzas encargadas por clientes con membresía.

-Obtener el día y la cantidad de suministros que haya recibido cualquier dependiente que hable inglés o alemán.

-Obtener todos los pedidos los cuales contengan pizzas cuyos ingredientes no provengan del proveedor de código 'B31957936'.

-Obtener los cocineros con un mínimo de 10 años contratados.

2. Segunda Entrega

2.1. Esquema relacional textual

BEBIDA(cod_beb: char(5), nombre_beb: varchar2(20), precio_beb_eu: float(3,2)
CP: {cod_beb}
VNN: {nombre_beb, precio_beb_eu}

CLIENTE(dni_cli :char(9), nombre_cli: varchar2(20), apellidos_cli: varchar2(20), tipo_via_dir_cli: varchar2(10), nombre_via_dir_cli: varchar2(20), portal_dir_cli: varchar2(3), esc_dir_cli: varchar2(3), puerta_dir_cli: varchar2(3), piso_dir_cli: varchar2(2))
CP: {dni_cli}
VNN: {tipo_via_dir_cli, nombre_via_dir_cli, portal_dir_cli, puerta_dir_cli}

COCINERO(dni_coc: char(9), titulacion: varchar2(20))
CP: {dni_coc}
VNN: {titulacion}
CAj: {dni_coc} → EMPLEADO(dni_emp)
Borrado en cascada, modificación en cascada

CONDUCE(dni_rep: char(9), matricula_veh: char(7))
CP: {dni_rep, matricula_veh}
CAj: {dni_rep} → REPARTIDOR(dni_rep)
Borrado restringido, modificación en cascada
CAj: {matricula_veh} → VEHICULO(matricula_veh)
Borrado restringido, modificación en cascada

CONTENIDO(cod_ped: char(5), fecha_ped: date, cod_piz: char(5), cantidad: int(2)
CP: {cod_ped, fecha_ped, cod_piz}
VNN: {cantidad}
CAj: {cod_ped, fecha_ped} → PEDIDO(cod_ped, fecha_ped)
Borrado restringido, modificación en cascada
CAj: {cod_piz} → PIZZA(cod_piz)
Borrado restringido, modificación en cascada

DEPENDIENTE(dni_dep: char(9))
CP: {dni_dep}
CAj: {dni_dep}→ EMPLEADO(dni_emp)
Borrado en cascada, modificación en cascada

EMPLEADO(dni_emp: char(9), nombre_emp: varchar2(20), apellidos_emp: varchar2(40), anyos_contrato_emp: int(2), edad_emp: int(2), activo_emp: bool)

CP: {dni_emp}
VNN: {nombre_emp, apellidos_emp, anyos_contrato_emp, edad_emp, activo_emp}
RI1: (edad_emp $\geq$ 16)

ENTREGA(dni_rep: char(9), matricula_veh: char(7), cod_ped: char(5), fecha_ped: date, hora_entrega_ped: time)

CP: {dni_rep, matricula_veh, cod_ped, fecha_ped}
VNN: {hora_entrega_ped}
CAj: {dni_rep, matricula_veh} $\rightarrow$ CONDUCE(dni_rep, matricula_veh)
Borrado restringido, modificación en cascada
CAj: {cod_ped, fecha_ped} $\rightarrow$ PEDIDO(cod_ped, fecha_ped)
Borrado restringido, modificación en cascada

HABLA_DEP(dni_dep: char(9), idioma_dep: varchar2(15))

CP: {dni_dep, idioma_dep}
CAj: {dni_dep} $\rightarrow$ DEPENDIENTE(dni_dep)
Borrado en cascada, modificación en cascada

INCLUYE_BEB(cod_ped: char(5), fecha_ped: date, cod_beb: char(5), cantidad: int(2))

CP: {cod_ped, fecha_ped, cod_ped}
VNN: {cantidad}
CAj: {cod_ped, fecha_ped} $\rightarrow$ PEDIDO(cod_ped, fecha_ped)
Borrado restringido, modificación en cascada
CAj: {cod_beb} $\rightarrow$ BEBIDA(cod_beb)
Borrado restringido, modificación en cascada

INGREDIENTE(nif_prov: char(9), cod_ing: char(5), nombre_ing: varchar2(25), precio_ing_eukg: float(5,2), ing_despensa_kg: float(5,2))

CP: {nif_prov, cod_ing}
VNN: {nombre_ing, precio_ing_eukg}
CAj: {nif_prov} $\rightarrow$ PROVEEDOR(nif_prov)
Borrado restringido, modificación en cascada

LLEVA(cod_piz: char(5), cod_ing: char(5), nif_prov: char(9))

CP: {cod_piz, cod_ing, nif_prov}
CAj: {cod_piz} $\rightarrow$ PIZZA(cod_piz)
Borrado restringido, modificación en cascada
CAj: {nif_prov, cod_ing} $\rightarrow$ INGREDIENTE(nif_prov, cod_ing)
Borrado restringido, modificación en cascada

MEMBRESIA(nombre_mem: varchar2(10), pedidos_mem: int(2), precio_mes_mem_eu: float(4,2), descuento_mem_per: int(2))

CP: {nombre_mem}
VNN: {pedidos_mem, precios_mes_mem_eu, descuento_mem_per}

MESES_MEM: (dni_cli: char(9), nombre_mem:varchar2(10), mes_mem:varcahr(10), anyo_mem:char(4))

CP: {dni_cli, nombre_mem, mes_mem, anyo_mem}
CAj: {dni_cli, nombre_mem} $\rightarrow$ TIENE_MEM(dni_cli, nombre_mem)
Borrado restringido, modificación en cascada
RI2: (mes_mem IN ("Enero", "Febrer", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre", "Noviembre", "Diciembre"))
RI3: (anyo_mem > 2010)

PEDIDO(cod_ped: char(5), fecha_ped: date, hora_ped: time, precio_ped_eu: float(5,2), valoracion_ped: int(1), dni_cli: char(9))

CP: {cod_ped, fecha_ped}
VNN: {hora_ped, precio_ped_eu, dni_cli}
CAj: {dni_cli} $\rightarrow$ CLIENTE(dni_cli)
Borrado restringido, modificación en cascada

PERMISO_REP(dni_rep(9): char(9), permiso_rep: varchar2(10))
CP: {dni_rep, permiso_rep}
CAj: {dni_rep} → REPARTIDOR(dni_rep)
Borrado en cascada, modificación en cascada

PIZZA(cod_piz: char(5), nombre_piz: varchar2(20), precio_piz_eu: float(4,2), tamanyo_piz: varchar2(8))
CP: {cod_piz}
VNN: {nombre_piz, precio_piz_eu, tamanyo_piz}
RI4: (tamanyo_piz ∈ ("pequeña", "mediana", "familiar"))

PROVEEDOR(nif_prov: char(9), nombre_prov: varchar2(20), tlf_prov: char(9))
CP: {nif_prov}
VNN: {nombre_prov}

REPARTIDOR(dni_rep(9))
CP: {dni_rep}
CAj: {dni_rep} → EMPLEADO(dni_emp)
Borrado en cascada, modificación en cascada

SUMINISTRO(nif_prov: char(9), cod_ing: char(5), dia_sum: date, hora_sum: time, cantidad_sum_kg: float(6,2), dni_dep: char(9))
CP: {nif_prov, cod_ing, dia_sum}
VNN: {hora_sum, cantidad_sum_kg, dni_dep}
CAj: {dni_dep} → DEPENDIENTE(dni_dep)
Borrado restringido, modificación en cascada
CAj: {nif_prov, cod_ing} → INGREDIENTE(nif_prov, cod_ing)
Borrado restringido, modificación en cascada

TIENE_MEM(dni_cli: char(9), nombre_mem:varchar2(10))
CP: {dni_cli, nombre_mem}
CAj: {dni_cli} → CLIENTE(dni_cli)
Borrado en cascada, modificación en cascada
CAj: {nombre_mem} → MEMBRESIA(nombre_mem)

TLF_CLIENTE(dni_cli: char(9), tlf_cli: char(9))
CP: {dni_cli, tlf_cli}
CAj: {dni_cli} → CLIENTE(dni_cli)
Borrado en cascada, modificación en cascada

TRANSPORTE(dni_rep: char(9), matricula_veh: char(7), dia_trans: date, hora_salida_trans: datetime, hora_llegada_trans: datetime, distancia_trans_km: float(5,2))
CP: {dni_rep, matricula_veh, dia_trans}
VNN: {hora_llegada_trans, distancia_trans_km}
CAj: {dni_rep, matricula_veh} → CONDUCE(dni_rep, matricula_veh)
Borrado restringido, modificación en cascada

VEHICULO(matricula_veh: char(7), capacidad_veh: int(3), modelo_veh: varchar2(40), fecha_itv_veh: date)
CP: {matricula_veh}
VNN: {capacidad_veh}

2.2. RI no representadas en el esquema

-RI5: Todo valor que aparezca en el atributo dni_dep de DEPENDIENTE debe aparecer en el atributo dni_dep de HABLA_DEP.

-RI6: Todo valor que aparezca en el atributo dni_rep de REPARTIDOR debe aparecer en el atributo dni_rep de PERMISO_REP.

-RI7: Todo valor que aparezca en el atributo dni_emp de EMPLEADO debe aparecer en el atributo dni_dep de DEPENDIENTE, dni_coc de COCINERO o dni_rep de REPARTIDOR.

-RI8: No puede haber un mismo valor en el atributo dni_dep de DEPENDIENTE y en el dni_coc de COCINERO; ni en el dni_dep de DEPENDIENTE y en el dni_rep de REPARTIDOR; ni en el atributo dni_coc de COCINERO y en el dni_rep de REPARTIDOR.

-RI9: Todo valor que aparezca en el atributo cod_piz de PIZZA debe aparecer en el atributo cod_piz de LLEVA.

-RI10: Toda tupla de valores de los atributos cod_ped y fecha_ped de PEDIDO debe aparecer en los atributos cod_ped y fecha_ped de CONTENIDO.

-RI11: Todo valor que aparezca en el atributo dni_cli de CLIENTE debe aparecer en el atributo dni_cli de TLF_CLIENTE.

-RI12: Todo valor que aparezca en el atributo dni_cli de CLIENTE debe aparecer en el atributo dni_cli de PEDIDO.

-RI13: Todo valor que aparezca en el atributo nif_prov de PROVEEDOR debe aparecer en el atributo nif_prov de INGREDIENTE.

-RI14: El precio del pedido se calcula automáticamente con la información del contenido del pedido y el posible descuento por ser cliente con membresía (si el pedido se realiza en un mes en el que el cliente tiene membresía).

-RI15: Los precios de las pizzas con el mismo nombre tienen que estar ordenados según el tamaño. Es decir, la pizza pequeña es más barata que la mediana y esta más barata que la familiar.

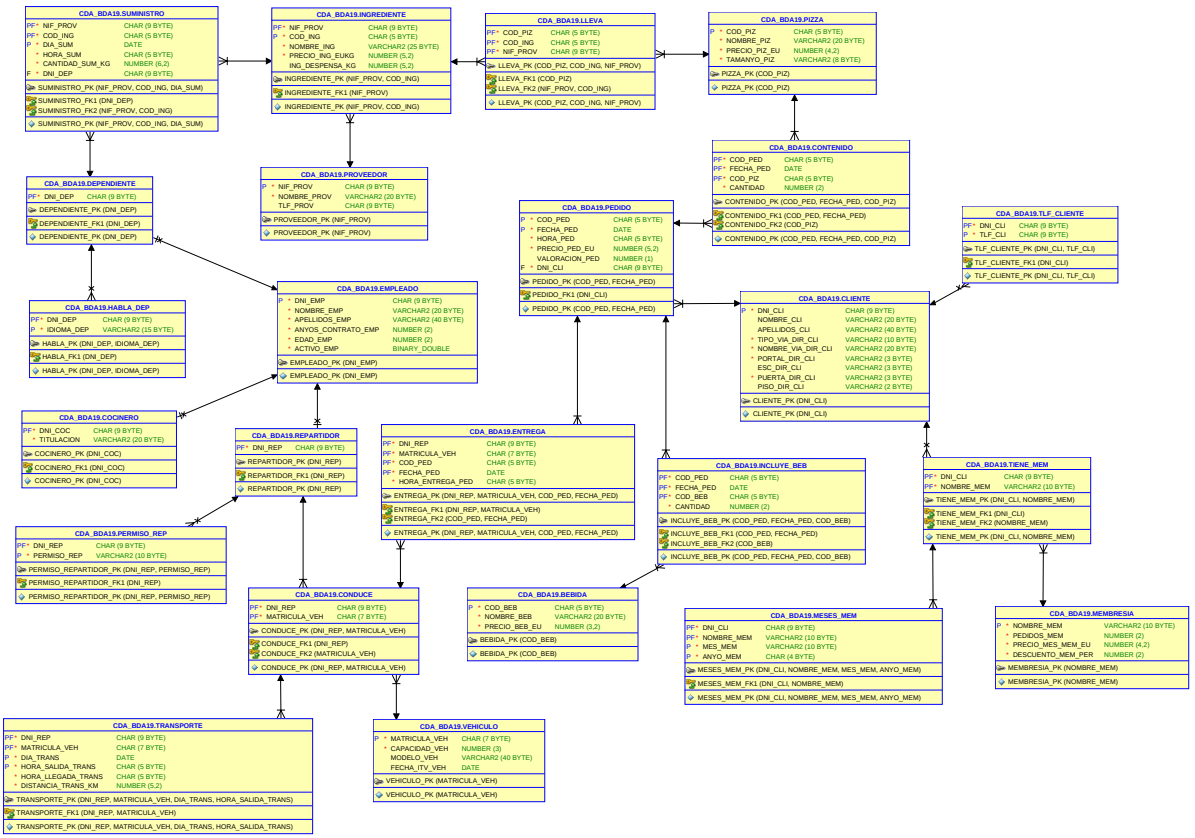
-RI16: hora_salida debe ser menor que hora_vuelta y su diferencia debe ser menor a 45 minutos.

-RI17: El precio de la pizza debe ser mayor al precio de la suma de los ingredientes que contiene.

-RI18: El cliente tiene que haber realizado la cantidad de pedidos necesarios para tener la membresía.

-RI19: La edad mínima de los empleados son 16 años.

2.3. Diagrama Oracle



2.4. Normalización

De acuerdo a la Teoría de Normalización, se puede confirmar que se cumple dicha teoría en esta base de datos, ya que se cumples las tres formas de acuerdo a los siguientes casos:

- Se cumple la definición de la Primera Forma Normal(1FN), ya que los atributos de la relación R sólo pueden tomar valores simples e indivisibles. De esta forma, 1FN establece que cada tabla en la base de datos debe tener una clave primaria única, y que todas las columnas de la tabla deben estar relacionadas con la clave primaria.

- También se cumple la definición de la Segunda Forma Normal (2FN), debido a que la relación R está en 1FN y todos sus atributos no-primos dependen totalmente de cualquier de R. Es decir, 2FN establece que cada columna de la tabla debe estar relacionada con la clave primaria y que ninguna de las columnas de la tabla puede depender de ninguna otra columna de la misma tabla.

- Por último, se cumple la definición de la Tercera Forma Normal (3FN), puesto que la relación R está en 2FN y no se dan dependencias funcionales entre los atributos no-primos. Así pues, 3FN establece que no hay columnas que dependan parcialmente de la clave primaria. Esto quiere decir que todas las columnas deben estar totalmente relacionadas con la clave primaria.

3. Tercera Entrega

3.1. Carga de datos

Para la carga de datos hemos utilizado el programa Python del repositorio, que contiene funciones que generan archivos con código SQL que insertan valores a las tablas. En las tablas más pequeñas se han modificado algunas cosas "a mano" o se han insertado directamente los datos, siempre y cuando la cantidad de filas que se fueran a insertar fuera razonablemente baja.

En el siguiente repositorio de GitHub se incluye dicho programa Python; archivos de texto con nombres y apellidos, que se han usado para mezclarlos y generar nombres aleatorios; un archivo csv de proveedores generado por Mockaroo, que posteriormente se ha tratado en Python; y un archivo CSV generado desde sqldeveloper para cargar ciertas claves primarias para la generación de datos de una tabla.

Repositorio GitHub: **Link**

3.2. Informes generados

Como parte de la tarea, hemos generado informes de variada dificultad para hacer un seguimiento de los datos que resulten más interesantes de la base de datos dispuestos de manera clara y concisa. Los informes realizados son los siguientes:

3.2.1. Primer informe: Cocineros con un mínimo de 10 años contratados

Con objeto de conocer a los empleados que trabajan en la cocina de las pizzerías desde hace 10 o más años, se desea obtener el DNI de dichos cocineros con un mínimo de 10 años de experiencia, la titulación máxima que han obtenido en sus estudios, los años que lleva contratado, la edad del empleado y si lleva tiempo activo en nuestra pizzería o no.

COCINEROS CONTRATADOS DESDE HACE 10 AÑOS O MÁS

DNI	TITULACION	AÑOS_CONTRAT	EDAD	TIEMPO ACTIVO
75897037G	FP	10	28	1.0
52590691H	FP	10	36	1.0
97751950H	Grado Medio	12	32	1.0
99616397Q	Grado Superior	13	40	0.0
15546422D	Grado Medio	14	31	1.0
24494772W	Grado Superior	15	39	0.0
84697196H	FP	16	51	1.0
46171555K	Grado Medio	16	32	1.0
81291930V	FP	16	53	1.0
11230819B	FP	17	33	1.0
20296318Z	FP	18	35	1.0
52777557D	Grado Superior	19	36	1.0
40432702J	Grado Medio	20	51	0.0
35186431L	FP	23	54	1.0
84345805K	Grado Medio	27	51	1.0
41264754V	FP	27	46	1.0

3.2.2. Segundo informe: Tipos de pizzas encargadas por clientes con membresía

En beneficio de nuestros clientes más leales (aquellos que pagan la cuota de membresía mensual), nuestra pizzería quiere recordarles cuáles de nuestras pizzas han sido solicitadas por todos ellos. Así pues, se desea un informe que muestre el código de las pizzas, el nombre de estas y sus precios en euros. Además, se anhela como requisito que se filtren las pizzas por un único proveedor y, que de esta forma, no se repitan los nombres y los códigos.

TIPOS DE PIZZAS ENCARGADAS POR CLIENTES CON MEMBRESÍA

COD_PIZ	NOMBRE_PIZ	PRECIO_PIZ_EU
PI001	Margherita	4,9
PI013	Cipola	5,05
PI043	Siciliana	5,1
PI022	Carcciofa	5,25
PI010	Salami	5,25
PI004	Napoli	5,25
PI019	Funghi	5,25
PI007	Napolitana	5,4
PI049	Giardinera	5,4
PI028	Romana	5,5
PI034	Salami e prosciutto	5,5
PI052	Tropical	5,5
PI016	Prosciutto	5,5
PI055	Tonno	5,5
PI025	Bologna	5,5

3.2.3. Tercer informe: Desplazamiento de los empleados

Se desea un informe que nos muestre el dni de los empleados, el nombre, los apellidos y la edad de los trabajadores que se hayan desplazado entre 20 y 25 km y que sean veteranos (3 años de contrato como mínimo)

Repartidores que se desplazan 20-25 km

DNI del empleado	Apellidos y Nombre del empleado	Edad del Empleado
20073711R	Orón Blasco Carla	47
32354115S	Garijo Espinosa Luis	55
20432834W	Juan Añó Pablo	24
70604624Y	Navarro Pallardó Francisco Javier	51
23601605V	Juan Alonso Jorge	33
76110246A	Blasco Parrilla Aina Amelia	55
60609369E	Hidalgo Martínez Marián	57
66366441X	Maamri Belenguer Anselmo	47
35357688H	Peris Rodríguez Fanny	51
83901649H	Blasco Añó Adam	53
88281426C	Silvestre López Yassmina	24
55139522N	Raga López Alejandro	51
88663057N	González Hernández Denyse	37
11589562T	Rallo Lladró Joan	34
54953241P	Martínez Palop Francisco Javier	48
62205460M	López Fuentes Fanny	33
34467533D	Verdú Hidalgo Aina Amelia	51
14672153Q	Tortola González Violeta	59

3.2.4. Cuarto informe: Plus de antiguedad

Como premio a nuestros trabajadores, la Pizzería Buona ha decidido implementar un plus de antigüedad que se vea reflejado en la nómina. Para ello, se desea un informe que muestre el DNI de los empleados, sus apellidos, la edad de estos y los años que llevan contratados, así como el total de empleados. Para una consulta más dinámica y sencilla, se desea implementar una manera de obtener los anteriores datos de quiénes han trabajado más de unos años especificados

Empleados según sus años contratados

DNI	Apellidos del empleado	Años trabajados	Edad
85142008X	Martínez Muñoz	23	40
41264754V	Cortés Rico	27	46
84345805K	García Tallec	27	51
11230819B	González Tortola	17	33
34467533D	Verdú Hidalgo	22	51
81291930V	Portillo Nebot	16	53
20296318Z	Hidalgo Chirivella	18	35
18917398J	Raga García	22	56
46171555K	Benetó Muñoz	16	32
52777557D	Martínez García	19	36
40432702J	Espinosa Juan	20	51
14672153Q	Tortola González	28	59
35186431L	Sanchis Galán	23	54
84697196H	Álvarez Raga	16	51

Los empleados que han trabajado más de los años pedidos son 14

3.2.5. Quinto informe: Ingredientes y proveedores por pizza

Para una mayor información de cara a la compra de ingredientes, se desea obtener los ingredientes que contiene cada pizza de manera individual y los NIF de los mismos.

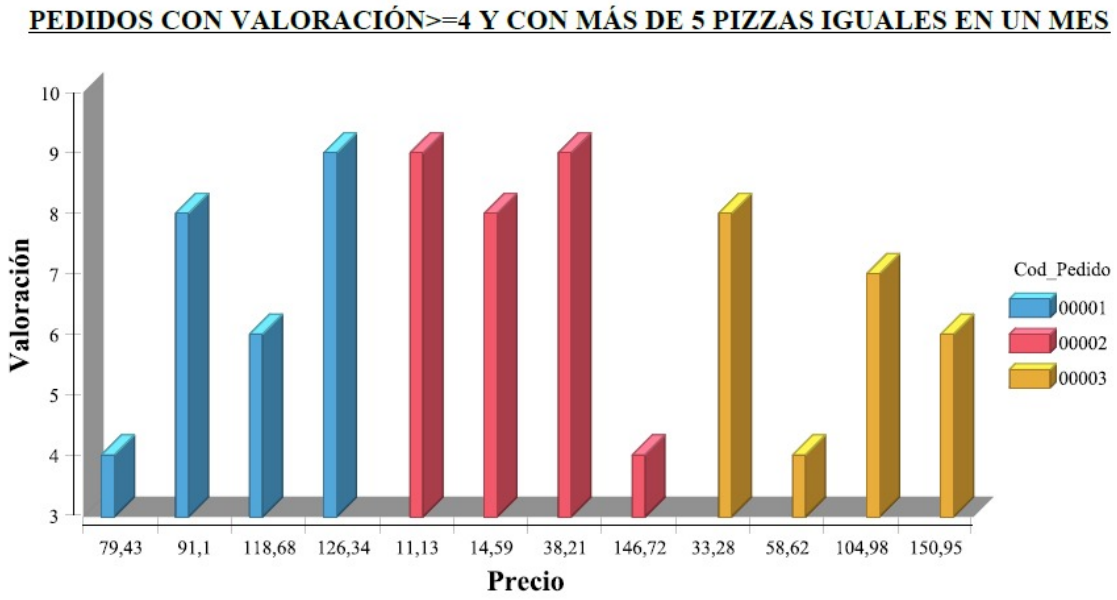
Empleados según sus años contratados

DNI	Apellidos del empleado	Años trabajados	Edad
85142008X	Martínez Muñoz	23	40
41264754V	Cortés Rico	27	46
84345805K	García Tallec	27	51
11230819B	González Tortola	17	33
34467533D	Verdú Hidalgo	22	51
81291930V	Portillo Nebot	16	53
20296318Z	Hidalgo Chirivella	18	35
18917398J	Raga García	22	56
46171555K	Benetó Muñoz	16	32
52777557D	Martínez García	19	36
40432702J	Espinosa Juan	20	51
14672153Q	Tortola González	28	59
35186431L	Sanchis Galán	23	54
84697196H	Álvarez Raga	16	51

Los empleados que han trabajado más de los años pedidos son 14

3.2.6. Sexto informe: Pedidos con una valoración de 4 (como mínimo) y que contenga al menos 5 pizzas iguales

Se sospecha que no se utiliza todo el espacio del vehículo cuando los repartidores se van a entregar pedidos, por ello se desea realizar una consulta para obtener aquellos pedidos que se han realizado a lo largo de un mes con una valoración mínima (en este caso un 4) y filtrar dichos envíos por aquellos que contengan más de 5 pizzas del mismo tamaño y mismo tipo(ya sea Barbacoa, Margarita...). Cabe destacar que se requiere resolver la consulta mediante un gráfico donde se especifique la valoración, el recio y el código de cada pedido recibido.



3.3. Procedimientos

3.3.1. Diferencia precio pizzas

Se ha implementado el siguiente procedimiento que indica la diferencia de dinero entre la pizza familiar y la mediana y entre la mediana y la pequeña.

```
create or replace procedure diferencia_precios_pizza(
    p_pizza pizza.nombre_piz %type)
is
cursor precios is
    select precio_piz_eu, tamanyo_piz
    from pizza
    where nombre_piz = p_pizza;
    v_precio_f pizza.precio_piz_eu %type;
    dif_fm pizza.precio_piz_eu %type;
    v_precio_m pizza.precio_piz_eu %type;
    dif_mp pizza.precio_piz_eu %type;
    v_precio_p pizza.precio_piz_eu %type;
begin
    for precio in precios loop
        if precio.tamanyo_piz = 'Familiar' then
            v_precio_f := precio.precio_piz_eu;
        elsif precio.tamanyo_piz = 'Mediana' then
            v_precio_m := precio.precio_piz_eu;
        else
            v_precio_p := precio.precio_piz_eu;
        end if;
    end loop;
    dif_fm := v_precio_f - v_precio_m;
    dif_mp := v_precio_m - v_precio_p;
    dbms_output.put_line('La pizza mediana cuesta '||to_char(dif_mp)||' € más que la pequeña y la familiar
    cuesta '||to_char(dif_fm)||' € más que la mediana');
end;
```

3.3.2. Trayectos de repartidor con transporte

Se ha implementado el siguiente procedimiento que indica todas las direcciones a las que ha ido un repartidor con un vehículo y el día en el que fue.

```
create or replace procedure donde_fue(
    p_dni_rep conduce.dni_rep %type,
    p_matricula_veh conduce.matricula_veh %type)
is
    cursor pedidos is select dni_cli, fecha_ped
                      from pedido p
                      where exists (select *
                                    from entrega
                                    where p.cod_ped = cod_ped and p.fecha_ped = fecha_ped
                                          and p_dni_rep = dni_rep and p_matricula_veh = matricula_veh);
    v_tipo_via_dir_cli cliente.tipo_via_dir_cli %type;
    v_nombre_via_dir_cli cliente.nombre_via_dir_cli %type;
    v_portal_dir_cli cliente.portal_dir_cli %type;
begin
    for pedido in pedidos loop
        select tipo_via_dir_cli, nombre_via_dir_cli, portal_dir_cli
        into v_tipo_via_dir_cli, v_nombre_via_dir_cli, v_portal_dir_cli
        from cliente
        where dni_cli = pedido.dni_cli;
        dbms_output.put_line('Fue el '||to_char(pedido.fecha_ped)||' a '||v_tipo_via_dir_cli||'
```

```
'||v_nombre_via_dir_cli||' '||v_portal_dir_cli);
    end loop;
end;
```

3.3.3. Proveedores por ingrediente

Se ha implementado este procedimiento que dado un ingrediente muestra los proveedores que lo ofrecent con su teléfono.

```
create or replace procedure provs_ing(
    p_ingrediente ingrediente.nombre_ing%type)
is
    v_nombre_prov proveedor.nombre_prov%type;
    v_tlf_prov proveedor.nombre_prov%type;
    cursor provs is
        select nif_prov
        from ingrediente
        where nombre_ing = p_ingrediente;
begin
    dbms_output.put_line('Ofrecen este ingrediente los siguientes proveedores');
    for prov in provs loop
        select tlf_prov, nombre_prov into v_tlf_prov, v_nombre_prov
        from proveedor
        where nif_prov = prov.nif_prov;
        dbms_output.put_line('Proveedor '||v_nombre_prov||' con teléfono '||v_tlf_prov);
    end loop;
end;
```

3.4. Funciones

3.4.1. Valoración media

Se ha implementado la siguiente función que devuelve la media de todas las valoraciones que ha hecho un cliente.

```
create or replace function media_valoracion(
    p_dni_cli pedido.dni_cli%type)
    return number
is
    v_media number;
    v_valoracion pedido.valoracion_ped%type;
begin
    select valoracion_ped into v_media
    from pedido
    where p_dni_cli = dni_cli;
    select avg(valoracion_ped) into v_media
    from pedido
    where p_dni_cli = dni_cli;
    return v_media;
exception when no_data_found then
    raise_application_error(-20202, 'Cliente inexistente');
end;
```

3.4.2. Cálculo precio del pedido

Se ha implementado la siguiente función que calcula el precio de un pedido a partir de los contenidos de este y el posbile descuento por membresía y devuelve este valor, además de modficar el valor del precio en la tabla pedido por este calculado.

```
create or replace function precio_pedido(
    p_cod_ped pedido.cod_ped %type,
    p_fecha_ped pedido.fecha_ped %type)
return pedido.precio_ped_eu %type
is
    v_dni_cli pedido.dni_cli %type;
    v_precio pedido.precio_ped_eu %type := 0;
    v_precio_piz pizza.precio_piz_eu %type;
    v_precio_beb bebida.precio_beb_eu %type;
    v_anyo meses_mem.anyo_mem %type;
    v_num_mes number(2);
    v_nombre_mes meses_mem.mes_mem %type;
    v_nombre_mem meses_mem.nombre_mem %type;
    v_descuento membresia.descuento_mem_per %type := 0;
    v_existe_mem number;
    cursor cur_pizzas is
        select cod_piz, cantidad
        from contenido
        where cod_ped = p_cod_ped and fecha_ped = p_fecha_ped;
    cursor cur_bebidas is
        select cod_beb, cantidad
        from incluye_beb
        where cod_ped = p_cod_ped and fecha_ped = p_fecha_ped;
begin
    select dni_cli into v_dni_cli
    from pedido
    where cod_ped = p_cod_ped and fecha_ped = p_fecha_ped;
    for c_piz in cur_pizzas loop
        select precio_piz_eu into v_precio_piz
        from pizza
        where c_piz.cod_piz = cod_piz;
        v_precio := v_precio + (v_precio_piz * c_piz.cantidad);
    end loop;
    for c_beb in cur_bebidas loop
        select precio_beb_eu into v_precio_beb
        from bebida
        where c_beb.cod_beb = cod_beb;
        v_precio := v_precio + (v_precio_beb * c_beb.cantidad);
        end loop;
    select EXTRACT(MONTH FROM p_fecha_ped) into v_num_mes from dual;
    select TO_CHAR(p_fecha_ped, 'YYYY') into v_anyo from dual;
    if (v_num_mes = 1) then
        v_nombre_mes := 'Enero';
    elsif (v_num_mes = 2) then
        v_nombre_mes := 'Febrero';
    elsif (v_num_mes = 3) then
        v_nombre_mes := 'Marzo';
    elsif (v_num_mes = 4) then
        v_nombre_mes := 'Abril';
    elsif (v_num_mes = 5) then
        v_nombre_mes := 'Mayo';
    elsif (v_num_mes = 6) then
        v_nombre_mes := 'Junio';
    elsif (v_num_mes = 7) then
        v_nombre_mes := 'Julio';
    elsif (v_num_mes = 8) then
        v_nombre_mes := 'Agosto';
    elsif (v_num_mes = 9) then
```

```
        v_nombre_mes := 'Septiembre';
    elsif (v_num_mes = 10) then
        v_nombre_mes := 'Octubre';
    elsif (v_num_mes = 11) then
        v_nombre_mes := 'Noviembre';
    else
        v_nombre_mes := 'Diciembre';
    end if;
    select count(*) into v_existe_mem
    from meses_mem
    where dni_cli = v_dni_cli and mes_mem = v_nombre_mes and anyo_mem = v_anyo;
    if (v_existe_mem ¿0) then
        select nombre_mem into v_nombre_mem
        from meses_mem
        where dni_cli = v_dni_cli and mes_mem = v_nombre_mes and anyo_mem = v_anyo;
        select descuento_mem_per into v_descuento
        from membresia
        where nombre_mem = v_nombre_mem;
        v_precio := v_precio * ((100-v_descuento)/100);
    end if;
    update pedido
    set precio_ped_eu = v_precio
    where cod_ped = p_cod_ped and fecha_ped = p_fecha_ped;
    return v_precio;
end;
```

3.4.3. Eliminación suministro defectuoso

Se ha implementado la siguiente función que recibe un suministro (sus identificadores $\mathrm{cod}_ing, nif_provdia_sum)yloelimina..$

```
create or replace function suministro_defectuoso(
    p_nif_prov suministro.nif_prov %type,
    p_cod_ing suministro.cod_ing %type,
    p_dia_sum suministro.dia_sum %type)
return boolean
is
    v_existe number;
begin
    select count(*) into v_existe
    from suministro
    where p_nif_prov = nif_prov and p_cod_ing = cod_ing and p_dia_sum = dia_sum;
    if (v_existe>0) then
        delete from suministro
        where p_nif_prov = nif_prov and p_cod_ing = cod_ing and p_dia_sum = dia_sum;
    end if;
    return (v_existe>0);
end;
```

3.5. Disparadores

3.5.1. Variable controlada pedidos realizados por cliente

Se ha añadido el atributo num_ped_cli a la tabla pedido y se ha incializado con la cantidad de pedidos que ha realizado cada cliente. A continuación se han implementado tres disparadores para controlar esta variable si se insertan, modifican o eliminan pedidos.

Insertar

```
create or replace trigger insertar_pedido
after insert on pedido
for each row
begin
    update cliente
    set num_ped_cli = num_ped_cli + 1
    where dni_cli = :new.dni_cli;
end;
```

Eliminar

```
create or replace trigger borrar_pedido
after delete on pedido
for each row
begin
    update cliente
    set num_ped_cli = num_ped_cli - 1
    where dni_cli = :new.dni_cli;
end;
```

Modificar

```
create or replace trigger modificar_pedido
after update on pedido
for each row
begin
    update cliente
    set num_ped_cli = num_ped_cli - 1
    where dni_cli = :old.dni_cli;
    update cliente
    set num_ped_cli = num_ped_cli + 1
    where dni_cli = :new.dni_cli;
end;
```

3.5.2. Control pedidos para membresía

Se ha implementado el siguiente disparador que comprueba que el cliente ha realizado los pedidos necesarios para obtener una membresía antes de añadir una mensualidad para esta membresía.

```
create or replace trigger pedidos_membresia
before insert or update of nombre_mem on meses_mem
for each row
declare
    _num_mes char(2);
    v_num_ped cliente.num_ped_cli %type;
    v_fecha_ped pedido.fecha_ped %type;
    v_ped_nec membresia.pedidos_mem %type;
    ped_insuf exception;
begin
    if (:new.mes_mem = 'Enero') then
        v_num_mes := '1';
    elsif (:new.mes_mem = 'Febrero') then
        v_num_mes := '2';
    elsif (:new.mes_mem = 'Marzo') then
        v_num_mes := '3';
    elsif (:new.mes_mem = 'Abril') then
        v_num_mes := '4';
    elsif (:new.mes_mem = 'Mayo') then
        v_num_mes := '5';
```

```
    elsif (:new.mes_mem = 'Junio') then
        v_num_mes := '6';
    elsif (:new.mes_mem = 'Julio') then
        v_num_mes := '7';
    elsif (:new.mes_mem = 'Agosto') then
        v_num_mes := '8';
    elsif (:new.mes_mem = 'Septiembre') then
        v_num_mes := '9';
    elsif (:new.mes_mem = 'Octubre') then
        v_num_mes := '10';
    elsif (:new.mes_mem = 'Noviembre') then
        v_num_mes := '11';
    else
        v_num_mes := '12';
    end if;
    select to_date('01/'||v_num_mes||'/'||:new.anyo_mem, 'DD/MM/YYYY')
    into v_fecha_ped
    from dual;
    select count(*)
    into v_num_ped
    from pedido
    where dni_cli = :new.dni_cli and fecha_ped ¡v_fecha_ped;
    select pedidos_mem into v_ped_nec
    from membresia
    where :new.nombre_mem = nombre_mem;
    if (v_ped_nec ¿v_num_ped) then raise ped_insuf;
    end if;
exception
    when ped_insuf then
        raise_application_error(-20101, 'Este cliente no ha hecho los pedidos suficientes para obtener esta
        membresía');
end;
```

3.5.3. Control diferencia de horas de transporte

Para comprobar la restricción de integridad que indica que debe haber como mínimo una diferencia de 45 min entre la hora de llegada y la hora de salida del transporte, se ha implementado el siguiente disparador:

```
create or replace trigger diferencia_hora
before insert on transporte
for each row
begin
    if :new.hora_llegada_trans - :new.hora_salida_trans > '00:45':
 raise_application_ERROR(-20007, 'La hora de llegada debe ser 45 min después de la hora de salida como
mínimo');
    end if;
end;
```