



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

PIZZERÍA BUONA

GESTIÓN DE DATOS: Ciencia de Datos. Grupo 19

Curso 2022/2023



Daniel Garijo, Javier Luque, Pablo Parrilla

Índice

1. Primera Parte. Cassandra	3
1.1. Primera Etapa. Cassandra, diagrama de clases	3
1.1.1. Diagrama de clases UML	3
1.1.2. Diagrama de clases orientado a la agregación	4
1.1.3. Descripción del caso con las modificaciones realizadas	4
1.2. Segunda Etapa. Diseño de la base de datos Cassandra	5
1.2.1. Workflow	5
1.2.2. Transformación de clases	6
1.2.3. Diagrama Chebotko: Queries	7
1.2.4. Diagrama Chebotko: Completo	11
1.3. Tercera Etapa. Creación y carga de la base de datos Cassandra	12
1.3.1. Instrucciones de creación de las tablas Cassandra	12
1.3.2. Consultas SQL para la carga de las tablas Cassandra	18
1.3.3. Carga de las tablas Cassandra	24
1.4. Cuarta Etapa. Resolución de consultas CQL	27
2. Segunda Parte. Neo4j	31
2.1. Primera Etapa. Diseño de la base de datos Neo4j	31
2.1.1. Diagrama de clases	31
2.1.2. Diagrama Lógico Neo4j	32
2.1.3. Diagrama Lógico Neo4j (Grafo)	32
2.2. Segunda Etapa. Creación y carga de la base de datos Neo4j	33
2.2.1. Nodo Empleado (Dependiente y Repartidor)	33
2.2.2. Nodo Ingrediente	34
2.2.3. Nodo Suministro y Arcos Suministra y Recibe	35
2.2.4. Nodo Pizza	36
2.2.5. Arco Lleva	36
2.2.6. Nodo Cliente	37
2.2.7. Nodo Pedido y Arco Realiza	38
2.2.8. Arco Contiene	39
2.2.9. Nodo Membresía	40
2.2.10. Arco Tiene	40
2.2.11. Nodo Vehículo	41
2.2.12. Nodo Conducir y Arcos Conducido_por y Conduce	42
2.2.13. Nodo Transporte y Arco Recorre	43
2.2.14. Arco Entrega	44
2.3. Tercera Etapa. Consultas en Cypher	46

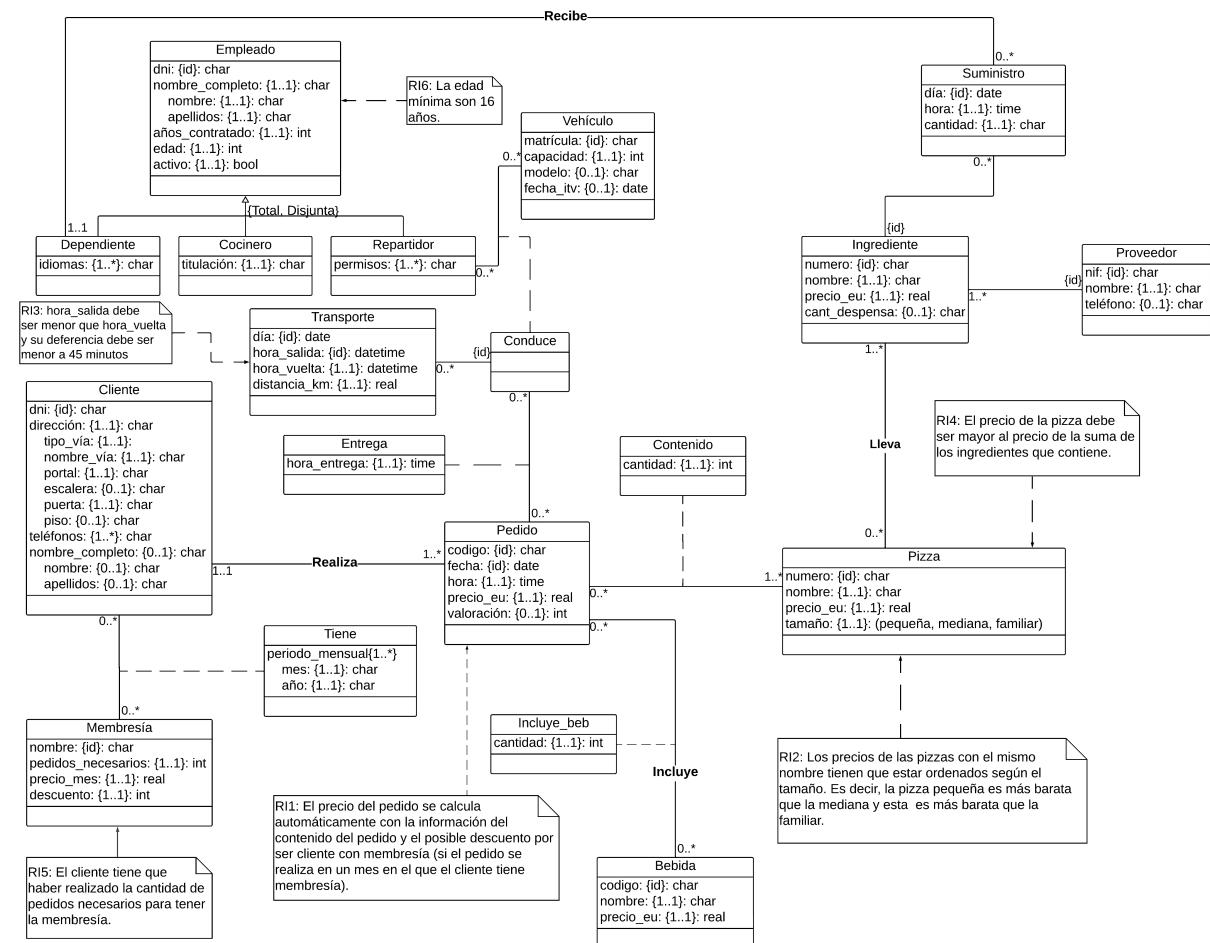
1. Primera Parte. Cassandra

1.1. Primera Etapa. Cassandra, diagrama de clases

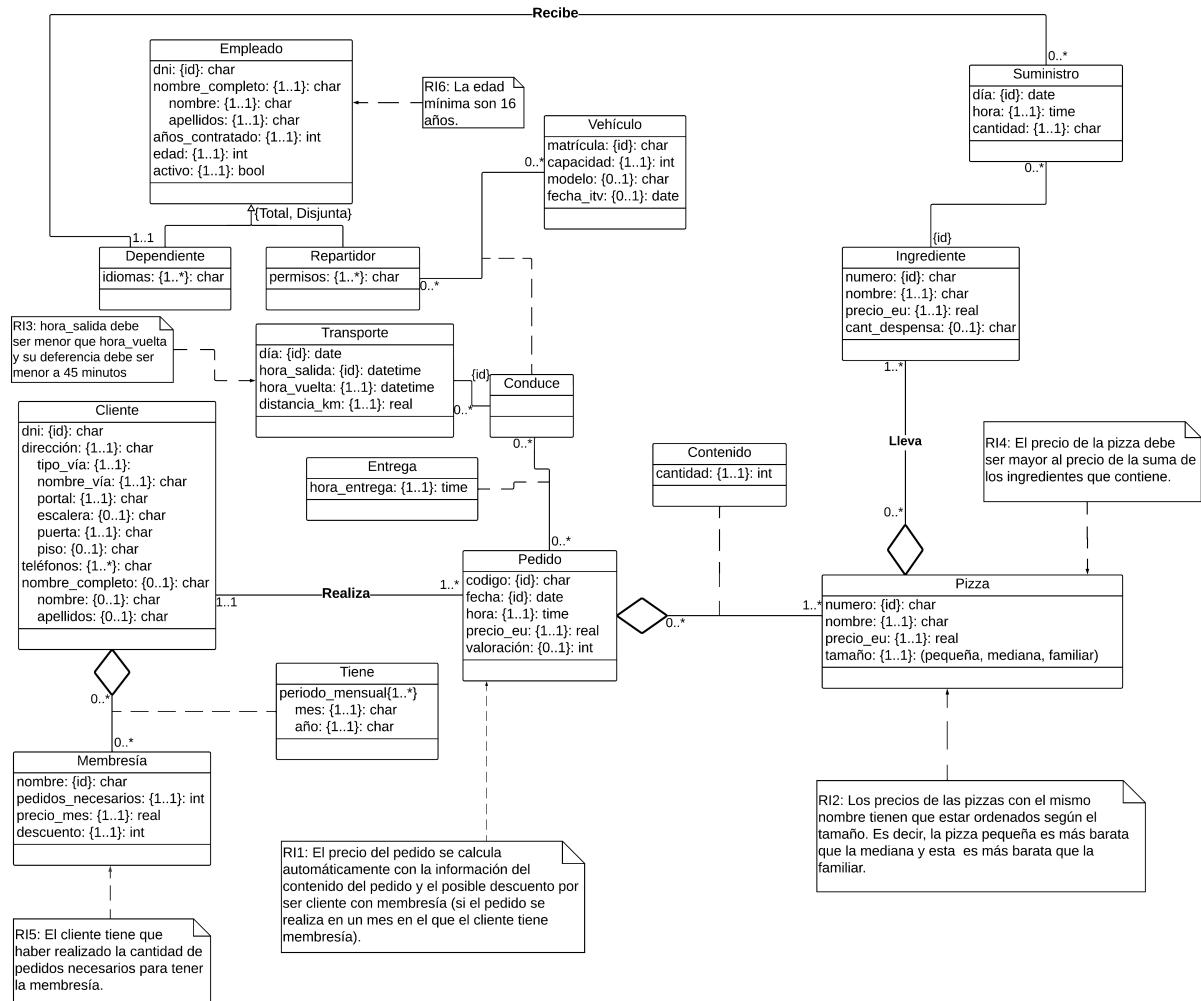
Como bien tratamos en la asignatura de Bases de Datos (BDA), nuestro caso busca relacionar una base de datos creada a partir de las acciones y objetos que mueve una pizzería (Pizzería Buona). Así pues, las clases que identificamos para realizar el diagrama de clases UML fueron: Empleado, Suministro, Vehículo, Cocinero, Dependiente, Repartidor, Transporte, Entrega, Pedido, Cliente, Contenido, Pizza, Membresía, Tiene, Bebida, Ingrediente y Proveedor.

De esta forma, creamos el diagrama de clases UML mediante la herramienta Lucid Chart para mostrar las clases, sus atributos y sus relaciones, incluyendo la herencia, la asociación y la composición para representar la estructura de la pizzería.

1.1.1. Diagrama de clases UML



1.1.2. Diagrama de clases orientado a la agregación



[Link a LucidChart](#)

1.1.3. Descripción del caso con las modificaciones realizadas

Como definición, sabemos que la agregación es una relación entre dos clases en la que una clase (clase contenedora) incluye a otra (clase contenida) como parte de su estructura. En la agregación, la clase contenida puede existir independientemente de la clase contenedora y puede estar contenida en otras clases también. De esta forma, podemos ver que la agregación se representa en el diagrama de clases con una línea con un rombo vacío en el extremo de la clase contenedora que apunta a la clase contenida, de manera que el rombo vacío representa la relación de "parte-de" indicando que la clase contenida es un componente de la clase contenedora.

A partir del diagrama de clases UML del trabajo de BDA, lo hemos modificado para obtener un diagrama de clases orientado a la agregación con la asociación entre las clases Cliente y Pedido. Así pues, se agrega la clase Membresía a la clase Cliente; y las clases Pizza y Bebida a la clase Pedido, cuando realmente se produce la agregación con las respectivas clases asociación Contenido e Incluye_beb. Por otra parte, también se agrega la clase Ingrediente a la clase Pizza.

Uniendo lo mencionado previamente, denotamos que una clase contenedora es la clase Pedido, siendo sus clases contenidas las clases Pizza (que a su vez es clase contenedora de la clase Ingrediente) y Bebida, y que hay otra clase contenedora que sería Cliente, teniendo la clase contenida Membresía.

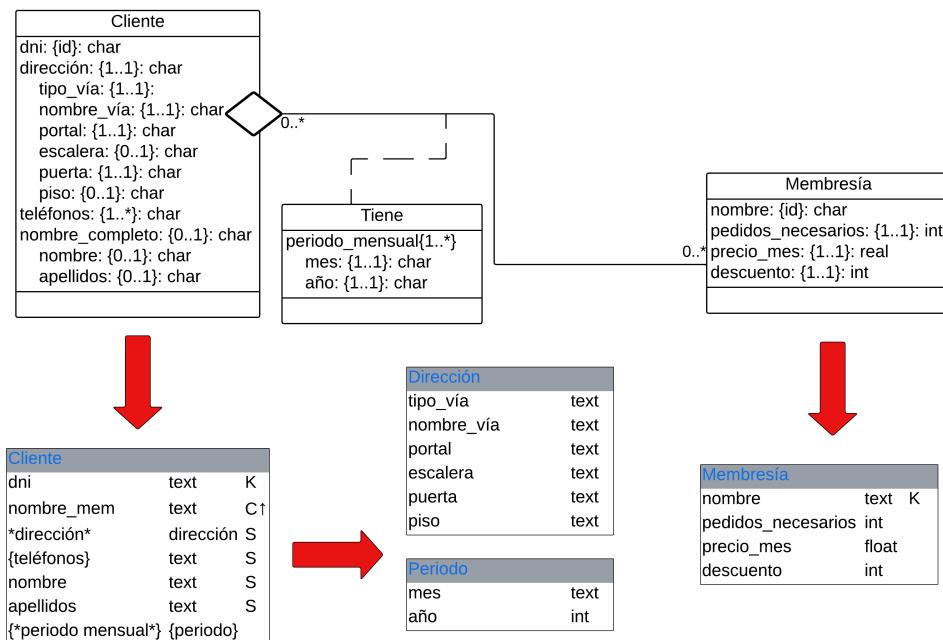
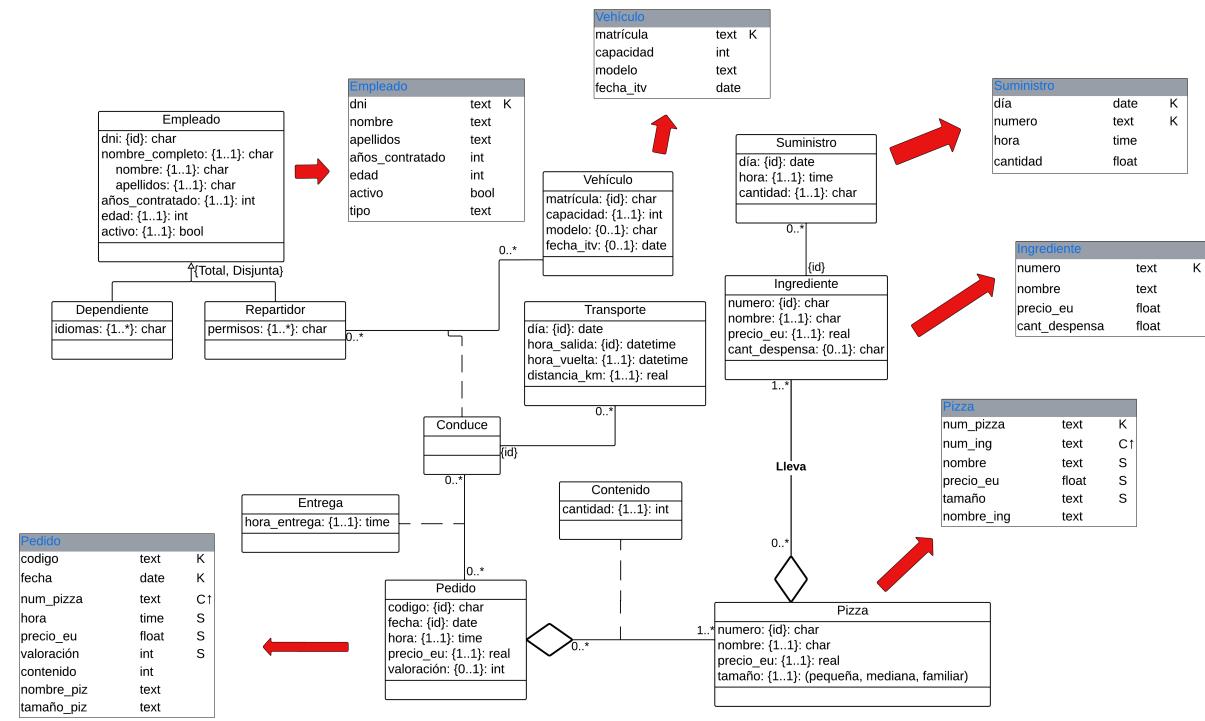
1.2. Segunda Etapa. Diseño de la base de datos Cassandra

En esta segunda etapa, hemos realizado el workflow a partir del diagrama de clases orientado a la agregación anterior y, teniendo en cuenta este, hemos realizado el diagrama Chebotko sobre las tablas de la pizzeria obtenidas.

1.2.1. Workflow

1. Mostrar los pedidos ordenados por fecha y hora descendente.
 - 1.1. Mostrar sus pizzas ordenadas por la cantidad de cada una descendente.
2. Mostrar los clientes ordenados por apellidos y nombre ascendente.
 - 2.1. Mostrar sus membresías ordenadas por período mensual descendente.
 - 2.2. Mostrar sus pedidos ordenados por valoración descendente.
3. Mostrar los empleados que lleven más de 5 años contratados.
4. Mostrar los vehículos ordenados por matrícula ascendente.
 - 4.1. Mostrar sus repartidores (los que han usado el vehículo) ordenados por apellidos y nombre ascendente.
5. Mostrar los tipos de pizzas ordenadas por su nombre ascendente.
 - 5.1. Mostrar sus ingredientes ordenados por su nombre y número ascendente.
 - 5.1.1. Mostrar sus suministros ordenados por día y hora descendente.

1.2.2. Transformación de clases



1.2.3. Diagrama Chebotko: Queries

▪ Tarea Q1

- RT1: Clase Pedido.
- RT4: Atributos ordenación fecha, hora y código.
- RT5: Atributos estáticos clase Pedido.

Pedido_por_fecha_y_hora			
agrupa	int	K	
fecha	date	C↓	
hora	text	C↓	
codigo	text	C↑	
precio_eu	float		
valoración	int		

▪ Tarea Q1_1

- RT1: clase asociación Contenido.
- RT2: búsqueda igualdad cod_ped y fecha_ped.
- RT4: atributo ordenación cantidad.
- RT5: atributos clave num_pizza.

Pizza_por_cantidad			
cod_ped	text	K	
fecha_ped	date	K	
cantidad	int	C↓	
num_pizza	text	C↑	
nombre	text		
tamaño	text		
precio_eu	float		

▪ Tarea Q2

- RT1: clase Cliente.
- RT4: atributos ordenación apellidos y nombre.
- RT5: atributos clave dni.

Cliente_por_apellidos_y_nombre			
agrupa	int	K	
apellidos	text	C↑	
nombre	text	C↑	
dni	text	C↑	
dirección	dirección		
{teléfonos}	text		

■ Tarea Q2_1

- RT1: clase asociación Tiene.
- RT2: búsqueda igualdad dni.
- RT4: atributos ordenación año y mes.
- RT5: atributos clave nombre_mem.

Membresía_por_año_y_mes		
dni	text	K
año	int	C↓
mes	text	C↓
nombre	text	C↑
pedidos_necesarios	int	
precio_mes	float	
descuento	int	

■ Tarea Q2_2

- RT1: asociación Realiza y clase Pedido.
- RT2: búsqueda de igualdad dni.
- RT4: atributo de ordenación valoración.
- RT5: atributos claves codigo y fecha.

Pedido_por_valoración		
dni	text	K
valoración	int	C↓
codigo	text	C↑
fecha	date	C↑
hora	time	
precio_eu	float	

■ Tarea Q3

- RT1: clase Empleado.
- RT3: búsqueda por rango años_contratado > 5.
- RT5: atributos clave dni.

Empleado_más_cinco_años		
agrupa	int	K
años_contratado	int	C↑
dni	text	C↑
nombre	text	
apellidos	text	
edad	int	
activo	bool	
tipo	text	

■ Tarea Q4

- RT1: clase Vehículo.
- RT4: atributo de ordenación matrícula.
- RT5: atributos clave matrícula.

Vehiculo_por_matricula		
agrupa	int	K
matrícula	text	C↑
capacidad	int	
modelo	text	
fecha_itv	date	

■ Tarea Q4_1

- RT1: clase asociación Conduce.
- RT2: búsqueda de igualdad matrícula.
- RT4: atributo de ordenación apellidos y nombre.
- RT5: atributos clave dni.

Repartidor_por_apellidos_y_nombre		
matrícula	text	K
apellidos	text	C↑
nombre	text	C↑
dni	text	C↑
años_contratado	int	
edad	int	
activo	bool	
tipo	text	

■ Tarea Q5

- RT1: clase Pizza.
- RT4: atributo de ordenación nombre.

Pizza_por_nombre		
agrupa	int	K
nombre	text	C↑
num_pizza	text	C↑

■ Tarea Q5_1

- RT1: agregación Lleva y clase Ingrediente.
- RT2: búsqueda de igualdad nom_pizza.
- RT4: atributos de ordenación nombre y numero.
- RT5: atributos de clave numero.

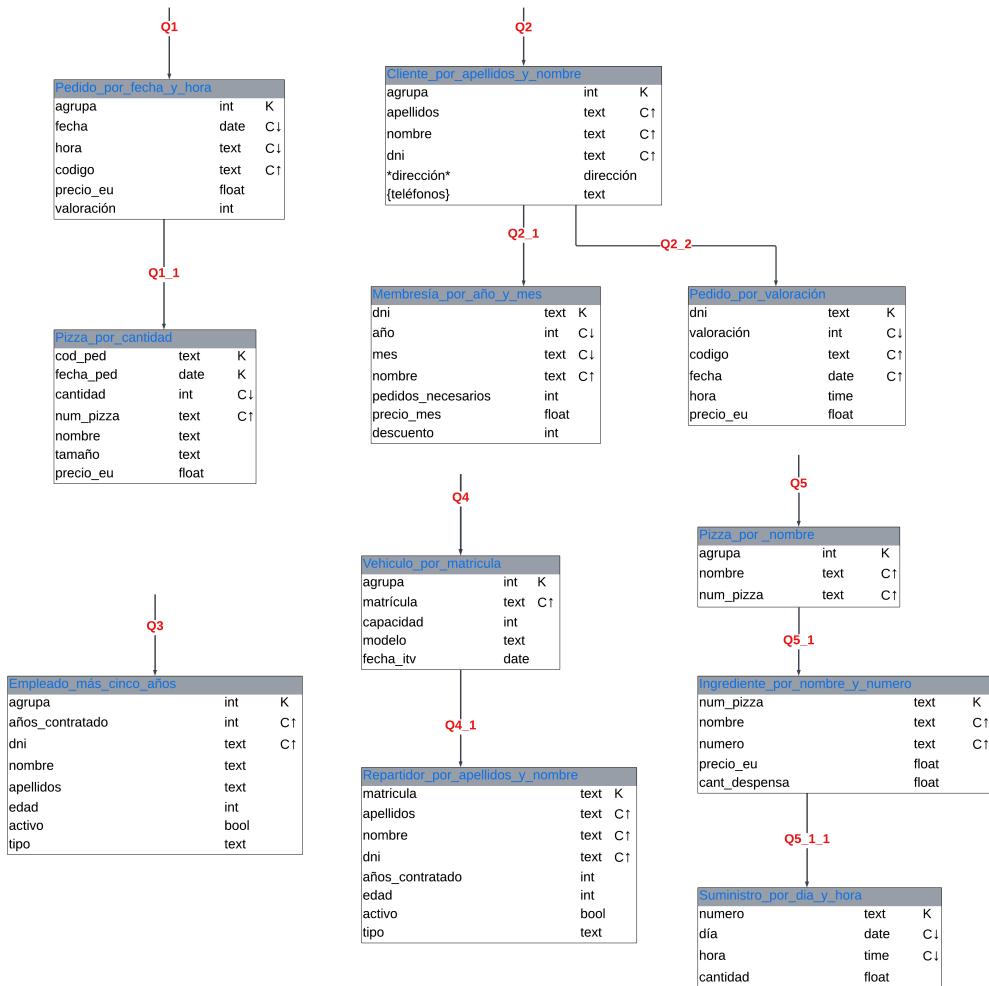
Ingrediente_por_nombre_y_numero		
num_pizza	text	K
nombre	text	C↑
numero	text	C↑
precio_eu	float	
cant_despensa	float	

■ Tarea Q5_1_1

- RT1: asociación clase Ingrediente y clase Suministro.
- RT2: búsqueda de igualdad numero y dia.
- RT4: atributo de ordenación dia y hora.
- RT5: atributos de clave dia y numero.

Suministro_por_dia_y_hora		
numero	text	K
día	date	C↓
hora	time	C↓
cantidad	float	

1.2.4. Diagrama Chebotko: Completo



1.3. Tercera Etapa. Creación y carga de la base de datos Cassandra

En esta tercera etapa, se ha realizado la creación y carga de información en la base de datos Cassandra. Para ello, se han creado las tablas obtenidas en la etapa anterior y se han recabado los datos desde Oracle mediante consultas SQL. El resultado de estas consultas se ha exportado a ficheros CSV (codificación UTF-8) para posteriormente cargarlos a su correspondiente tabla Cassandra.

1.3.1. Instrucciones de creación de las tablas Cassandra

Para empezar a crear las tablas, primero es necesario crear un *keyspace*.

```
CREATE KEYSPACE pizzeria WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '3'};
```

En la creación de tablas, se han utilizado dos tipos definidos: *direccion_t* y *periodo_t*.

```
CREATE TYPE direccion_t (
    tipo_via VARCHAR,
    nombre_via VARCHAR,
    portal VARCHAR,
    escalera VARCHAR,
    puerta VARCHAR,
    piso VARCHAR
);

CREATE TYPE periodo_t (
    anyo INT,
    mes INT
);
```

■ Tabla Empleado

```
CREATE TABLE empleado (
    dni VARCHAR PRIMARY KEY,
    nombre VARCHAR,
    apellidos VARCHAR,
    anyos_contratado INT,
    edad INT,
    activo BOOLEAN,
    tipo VARCHAR
);
```

■ Tabla Vehículo

```
CREATE TABLE vehiculo (
    matricula VARCHAR PRIMARY KEY,
    capacidad INT,
    modelo VARCHAR,
    fecha_itv DATE
);
```

- Tabla Suministro

```
CREATE TABLE suministro (
    dia DATE,
    num_ing VARCHAR,
    hora TIME,
    cantidad FLOAT,
    PRIMARY KEY (dia, num_ing)
);
```

- Tabla Ingrediente

```
CREATE TABLE ingrediente (
    numero VARCHAR PRIMARY KEY,
    nombre VARCHAR,
    precio_eu FLOAT,
    cant_dispensa FLOAT
);
```

- Tabla Pizza

```
CREATE TABLE pizza (
    num_pizza VARCHAR,
    num_ing VARCHAR,
    nombre VARCHAR STATIC,
    precio_eu FLOAT STATIC,
    tamanyo VARCHAR STATIC,
    nombre_ing VARCHAR,
    PRIMARY KEY (num_pizza, num_ing)
)
WITH CLUSTERING ORDER BY (num_ing ASC);
```

- Tabla Pedido

```
CREATE TABLE pedido (
    codigo VARCHAR,
    fecha DATE,
    num_piz VARCHAR,
    hora TIME STATIC,
    precio_eu FLOAT STATIC,
    valoracion INT STATIC,
    cantidad INT,
    nombre_piz VARCHAR,
    tamanyo_piz VARCHAR,
    PRIMARY KEY ((codigo, fecha), num_piz)
)
WITH CLUSTERING ORDER BY (num_piz ASC);
```

- Tabla Membresía

```
CREATE TABLE membresia (
    nombre VARCHAR PRIMARY KEY,
    pedidos_necesarios INT,
    precio_mes FLOAT,
    descuento INT
);
```

- Tabla Cliente

```
CREATE TABLE cliente (
    dni VARCHAR,
    nombre_mem VARCHAR,
    direccion FROZEN<direccion_t> STATIC,
    telefonos SET<VARCHAR> STATIC,
    nombre VARCHAR STATIC,
    apellidos VARCHAR STATIC,
    periodo_mes SET<FROZEN<periodo_t>>,
    PRIMARY KEY (dni, nombre_mem)
)
WITH CLUSTERING ORDER BY (nombre_mem ASC);
```

- Tabla Pedido por fecha y hora (Q1)

```
CREATE TABLE pedido_por_fecha_y_hora (
    agrupa INT,
    fecha DATE,
    hora TIME,
    codigo VARCHAR,
    precio_eu FLOAT,
    valoracion INT,
    PRIMARY KEY (agrupa, fecha, hora, codigo)
)
WITH CLUSTERING ORDER BY (fecha DESC, hora DESC, codigo ASC);
```

- Tabla Pizza por cantidad (Q1_1)

```
CREATE TABLE pizza_por_cantidad (
    cod_ped VARCHAR,
    fecha_ped DATE,
    cantidad INT,
    num_piz VARCHAR,
    nombre VARCHAR,
    tamano VARCHAR,
    precio_eu FLOAT,
    PRIMARY KEY ((cod_ped, fecha_ped), cantidad, num_piz)
)
WITH CLUSTERING ORDER BY (cantidad DESC, num_piz ASC);
```

- Tabla Cliente por apellidos y nombre (Q2)

```
CREATE TABLE cliente_por_apellidos_y_nombre (
    agrupa INT,
    apellidos VARCHAR,
    nombre VARCHAR,
    dni VARCHAR,
    direccion FROZEN<direccion_t>,
    telefonos SET<VARCHAR>,
    PRIMARY KEY (agrupa, apellidos, nombre, dni)
)
WITH CLUSTERING ORDER BY (apellidos ASC, nombre ASC, dni ASC);
```

- Tabla Membresía por año y mes (Q2_1)

```
CREATE TABLE membresia_por_anyo_y_mes (
    dni VARCHAR,
    periodo_mes FROZEN<periodo_t>,
    nombre VARCHAR,
    pedidos_necesarios INT,
    precio_mes FLOAT,
    descuento INT,
    PRIMARY KEY(dni, periodo_mes, nombre)
)
WITH CLUSTERING ORDER BY (anyo DESC, mes DESC, nombre ASC);
```

- Tabla Pedido por valoración (Q2_2)

```
CREATE TABLE pedido_por_valoracion (
    dni VARCHAR,
    valoracion INT,
    codigo VARCHAR,
    fecha DATE,
    hora TIME,
    precio_eu FLOAT,
    PRIMARY KEY (dni, valoracion, codigo, fecha)
)
WITH CLUSTERING ORDER BY (valoracion DESC, codigo ASC, fecha ASC);
```

- Tabla Empleado más de cinco años contratado (Q3)

```
CREATE TABLE empleado_mas_cinco_anuos (
    agrupa INT,
    anyos_contratado INT,
    dni VARCHAR,
    nombre VARCHAR,
    apellidos VARCHAR,
    edad INT,
    activo BOOLEAN,
    tipo VARCHAR,
    PRIMARY KEY (agrupa, anyos_contratado, dni)
)
WITH CLUSTERING ORDER BY (anyos_contratado ASC, dni ASC);
```

- Tabla Vehículo por matrícula (Q4)

```
CREATE TABLE vehiculo_por_matricula (
    agrupa INT,
    matricula VARCHAR,
    capacidad INT,
    modelo VARCHAR,
    fecha_itv DATE,
    PRIMARY KEY (agrupa, matricula)
)
WITH CLUSTERING ORDER BY (matricula ASC);
```

- Tabla Repartidor por apellidos y nombre (Q4_1)

```
CREATE TABLE repartidor_por_apellidos_y_nombre (
    matricula VARCHAR,
    apellidos VARCHAR,
    nombre VARCHAR,
    dni VARCHAR,
    anyos_contratado INT,
    edad INT,
    activo BOOLEAN,
    PRIMARY KEY (matricula, apellidos, nombre, dni)
)
WITH CLUSTERING ORDER BY (apellidos ASC, nombre ASC, dni ASC);
```

- Tabla Pizza por nombre (Q5)

```
CREATE TABLE pizza_por_nombre (
    agrupa INT,
    nombre VARCHAR,
    num_piz VARCHAR,
    PRIMARY KEY (agrupa, nombre, num_piz)
)
WITH CLUSTERING ORDER BY (nombre ASC, num_piz ASC);
```

- Tabla Ingrediente por nombre y numero (Q5_1)

```
CREATE TABLE ingrediente_por_nombre_y_numero (
    num_piz VARCHAR,
    nombre VARCHAR,
    numero VARCHAR,
    precio_eu FLOAT,
    cant_despensa FLOAT,
    PRIMARY KEY (num_piz, nombre, numero)
)
WITH CLUSTERING ORDER BY (nombre ASC, numero ASC);
```

- Tabla Suministro por día y hora (Q5_1_1)

```
CREATE TABLE suministro_por_dia_y_hora (
    numero VARCHAR,
    dia DATE,
    hora TIME,
    cantidad FLOAT,
    PRIMARY KEY (numero, dia, hora)
)
WITH CLUSTERING ORDER BY (dia DESC, hora DESC);
```

1.3.2. Consultas SQL para la carga de las tablas Cassandra

Algunas aclaraciones sobre las consultas:

1. En los booleanos se ha modificado la salida de 1,0 a ‘true’, ‘false’.
2. Para los empleados, se ha añadido la columna tipo indicando si el empleado es repartidor o dependiente (no hay cocineros).
3. En el diagrama original, Ingrediente es una clase débil de Proveedor, y la clase Proveedor ha sido podada. Por tanto, el código del ingrediente, que por sí mismo puede repetirse, pasa a ser la concatenación de dicho código con el nif del proveedor del ingrediente; asegurando así unicidad.
4. Para las horas, se han añadido los segundos con la concatenación *hora* || ‘:00’, para que el formato sea el correcto para el tipo TIME de Cassandra.
5. Para el tipo definido *direccion_t*, existen posibles valores nulos en los atributos **esc_dir_cli** y **pisos_dir_cli**, y Cassandra no admite valores nulos en atributos de tipos definidos por el usuario. Por lo tanto, para los posibles valores de estos atributos que sean nulos, se ha cambiado el valor por la cadena de texto ‘null’, y que de esta forma, funcione a modo de valor nulo a pesar de que no lo sea estrictamente.
Además, se han utilizado concatenaciones con el objetivo de dar el formato correcto para que Cassandra pueda leer el tipo definido *direccion_t*.
6. Originalmente, los teléfonos de los clientes estaban en una tabla aparte, ya que un cliente podía tener más de 1 teléfono. Por lo tanto, se han recogido todos los teléfonos de cada cliente dándole el formato adecuado del tipo SET.
7. Al igual que con el tipo definido *direccion_t*, se han utilizado concatenaciones para darle el formato adecuado para el tipo *periodo_t*.

Además, en el caso de la consulta de Cliente, al tener agregada la clase Membresía y existir clientes que no tienen membresía, el valor de este tipo definido podría ser nulo. Por lo tanto, se ha considerado esta posibilidad en la consulta.

8. En relación con el apartado anterior, como el nombre de la membresía puede ser nulo y en la tabla agregada Cliente se trata de un clave de clúster; se sustituye el posible valor por el valor ‘Ninguna’.

▪ Consulta Empleado

```
SELECT e.dni_emp AS dni,
       e.nombre_emp AS nombre,
       e.apellidos_emp AS apellidos,
       e.anyos_contrato_emp AS anyos_contratado,
       e.edad_emp AS edad,
       CASE
           WHEN e.activo_emp = 1 THEN 'true'
           WHEN e.activo_emp = 0 THEN 'false'
       END AS activo,
       CASE
           WHEN r.dni_rep IS NOT NULL THEN 'repartidor'
           WHEN d.dni_dep IS NOT NULL THEN 'dependiente'
       END AS tipo
FROM empleado e
LEFT JOIN repartidor r ON e.dni_emp = r.dni_rep
LEFT JOIN dependiente d ON e.dni_emp = d.dni_dep
WHERE r.dni_rep IS NOT NULL OR d.dni_dep IS NOT NULL;
```

- Consulta Vehículo

```
SELECT matricula_veh AS matricula,
       capacidad_veh AS capacidad,
       modelo_veh AS modelo,
       fecha_itv_veh AS fecha_itv
  FROM vehiculo;
```

- Consulta Suministro

```
SELECT dia_sum AS dia,
       cod_ing || nif_prov AS num_ing,
       hora_sum || ':00' AS hora,
       cantidad_sum_kg AS cantidad
  FROM suministro;
```

- Consulta Ingrediente

```
SELECT cod_ing || nif_prov AS numero,
       nombre_ing AS nombre,
       precio_ing_eukg AS precio_eu,
       ing_despensa_kg AS cant_dispensa
  FROM ingrediente;
```

- Consulta Pizza

```
SELECT p.cod_piz AS num_pizza,
       l.cod_ing || l.nif_prov AS num_ing,
       p.nombre_piz AS nombre,
       p.precio_piz_eu AS precio_eu,
       p.tamanyo_piz AS tamanyo,
       i.nombre_ing AS nombre_ing
  FROM pizza p, lleva l, ingrediente i
 WHERE p.cod_piz = l.cod_piz AND
       l.cod_ing = i.cod_ing AND
       l.nif_prov = i.nif_prov;
```

- Consulta Pedido

```
SELECT p.cod_ped AS codigo,
       p.fecha_ped AS fecha,
       c.cod_piz AS num_piz,
       p.hora_ped || ':00' AS hora,
       p.precio_ped_eu AS precio_eu,
       p.valoracion_ped AS valoracion,
       c.cantidad AS cantidad,
       pi.nombre_piz AS nombre_piz,
       pi.tamanyo_piz AS tamnyo_piz
  FROM pedido p, contenido c, pizza pi
 WHERE p.cod_ped = c.cod_ped AND
       p.fecha_ped = c.fecha_ped AND
       c.cod_piz = pi.cod_piz;
```

▪ Consulta Membresía

```

SELECT nombre_mem AS nombre,
       pedidos_mem AS pedidos_necesarios,
       precio_mes_mem_eu AS precio_mes,
       descuento_mem_per AS descuento
  FROM membresia;

```

▪ Consulta Cliente

```

SELECT dni, nombre_mem, direccion, telefonos, nombre, apellidos,
      '{' || LISTAGG(periodo_mes, ',') WITHIN GROUP (ORDER BY periodo_mes) || '}'
  FROM (
SELECT c.dni_cli AS dni,
       COALESCE(m.nombre_mem, 'Ninguna') AS nombre_mem,
       '{tipo_via:' || c.tipo_via_dir_cli ||
       ',nombre_via:' || c.nombre_via_dir_cli ||
       ',portal:' || c.portal_dir_cli ||
       CASE
           WHEN c.esc_dir_cli IS NULL THEN ',escalera:null'
           ELSE ',escalera:' || c.esc_dir_cli
        END ||
       ',puerta:' || c.puerta_dir_cli ||
       CASE
           WHEN c.piso_dir_cli IS NULL THEN ',piso:null'
           ELSE ',piso:' || c.piso_dir_cli
        END ||
       '}' AS direccion,
       '{' || LISTAGG(t.tlf_cli, ',') WITHIN GROUP (ORDER BY t.tlf_cli)
       || '}' AS telefonos,
       c.nombre_cli AS nombre,
       c.apellidos_cli AS apellidos,
       CASE
           WHEN m.nombre_mem IS NULL THEN NULL
           ELSE '{anyo:' || m.anyo_mem ||
                 ',mes:' || m.mes_orden || '}'
        END AS periodo_mes
  FROM cliente c
  LEFT JOIN meses_mem m ON c.dni_cli = m.dni_cli
  LEFT JOIN tlf_cliente t ON c.dni_cli = t.dni_cli
 GROUP BY c.dni_cli, m.nombre_mem, c.tipo_via_dir_cli,
          c.nombre_via_dir_cli, c.portal_dir_cli, c.esc_dir_cli,
          c.puerta_dir_cli, c.piso_dir_cli, c.nombre_cli,
          c.apellidos_cli, m.anyo_mem, m.mes_orden)
 GROUP BY dni, nombre_mem, direccion, telefonos, nombre, apellidos;

```

- Consulta Pedido por fecha y hora (Q1)

```

SELECT 1 AS agrupa,
       fecha_ped AS fecha,
       hora_ped || ':00' AS hora,
       cod_ped AS codigo,
       precio_ped_eu AS precio_eu,
       valoracion_ped AS valoracion
  FROM pedido
 ORDER BY fecha DESC, hora DESC;

```

- Consulta Pizza por cantidad (Q1_1)

```

SELECT c.cod_ped,
       c.fecha_ped,
       cantidad,
       c.cod_piz AS num_piz,
       nombre_piz AS nombre,
       tamanyo_piz AS tamanyo,
       precio_piz_eu AS precio_eu
  FROM pizza p, contenido c, pedido pe
 WHERE p.cod_piz = c.cod_piz AND
       pe.cod_ped = c.cod_ped AND
       pe.fecha_ped = c.fecha_ped
 ORDER BY cantidad DESC;

```

- Consulta Cliente por apellidos y nombre (Q2)

```

SELECT 1 AS agrupa,
       c.apellidos_cli AS apellidos,
       c.nombre_cli AS nombre,
       c.dni_cli AS dni,
       '{tipo_via:' || c.tipo_via_dir_cli || '
       ',nombre_via:' || c.nombre_via_dir_cli || '
       ',portal:' || c.portal_dir_cli || '
CASE
      WHEN c.escalera_dir_cli IS NULL THEN ',escalera:null'
      WHEN c.escalera_dir_cli IS NOT NULL THEN ',escalera:' || c.escalera_dir_cli
END ||
       ',puerta:' || c.puerta_dir_cli ||
CASE
      WHEN c.piso_dir_cli IS NULL THEN ',piso:null'
      WHEN c.piso_dir_cli IS NOT NULL THEN ',piso:' || c.piso_dir_cli
END ||
       '}' AS direccion,
       '{' || LISTAGG(t.tlf_cli, ',') WITHIN GROUP (ORDER BY t.tlf_cli)
       || '}' AS telefonos
  FROM cliente c
 LEFT JOIN tlf_cliente t ON c.dni_cli = t.dni_cli
 GROUP BY c.apellidos_cli, c.nombre_cli, c.dni_cli, c.tipo_via_dir_cli,
          c.nombre_via_dir_cli, c.portal_dir_cli, c.escalera_dir_cli,
          c.puerta_dir_cli, c.piso_dir_cli
 ORDER BY apellidos, nombre;

```

- Consulta Membresía por año y mes (Q2_1)

```

SELECT mes.dni_cli AS dni,
       '{anyo:' || mes.anyo_mem || ',mes:' || mes.mes_orden || '}' AS periodo_mes,
       mes.nombre_mem AS nombre,
       m_pedidos_mem AS pedidos_necesarios,
       m.precio_mes_mem_eu AS precio_mes,
       m.descuento_mem_per AS descuento
  FROM membresia m, meses_mem mes
 WHERE m.nombre_mem = mes.nombre_mem
 ORDER BY mes.anyo_mem DESC, mes.mes_orden DESC;

```

- Consulta Pedido por valoración (Q2_2)

```

SELECT dni_cli AS dni,
       valoracion_ped AS valoracion,
       cod_ped AS codigo,
       fecha_ped AS fecha,
       hora_ped AS hora,
       precio_ped_eu AS precio_eu
  FROM pedido
 WHERE valoracion_ped IS NOT NULL
 ORDER BY valoracion DESC;

```

- Consulta Empleado más de cinco años contratado (Q3)

```

SELECT 1 AS agrupa,
       e.anyos_contrato_emp AS anyos_contratado,
       e.dni_emp AS dni,
       e.nombre_emp AS nombre,
       e.apellidos_emp AS apellidos,
       e.edad_emp AS edad,
       CASE
           WHEN e.activo_emp = 1 THEN 'true'
           WHEN e.activo_emp = 0 THEN 'false'
       END AS activo,
       CASE
           WHEN r.dni_rep IS NOT NULL THEN 'repartidor'
           WHEN d.dni_dep IS NOT NULL THEN 'dependiente'
       END AS tipo
  FROM empleado e
 LEFT JOIN repartidor r ON e.dni_emp = r.dni_rep
 LEFT JOIN dependiente d ON e.dni_emp = d.dni_dep
 WHERE e.anyos_contrato_emp > 5
   AND (r.dni_rep IS NOT NULL OR d.dni_dep IS NOT NULL)
 ORDER BY anyos_contratado;

```

- Consulta Vehículo por matrícula (Q4)

```
SELECT 1 AS agrupa,
       matricula_veh AS matricula,
       capacidad_veh AS capacidad,
       modelo_veh AS modelo,
       fecha_itv_veh AS fecha_itv
  FROM vehiculo
 ORDER BY matricula;
```

- Consulta Repartidor por apellidos y nombre (Q4_1)

```
SELECT c.matricula_veh AS matricula,
       e.apellidos_emp AS apellidos,
       e.nombre_emp AS nombre,
       e.dni_emp AS dni_emp,
       e.anyos_contrato_emp AS anyos_contratado,
       e.edad_emp AS edad,
       CASE
           WHEN e.activo_emp = 1 THEN 'true'
           WHEN e.activo_emp = 0 THEN 'false'
       END AS activo
  FROM conduce c, empleado e
 WHERE c.dni_rep = e.dni_emp
 ORDER BY apellidos, nombre;
```

- Consulta Pizza por nombre (Q5)

```
SELECT 1 AS agrupa,
       nombre_piz AS nombre,
       cod_piz AS num_piz
  FROM pizza
 ORDER BY nombre;
```

- Consulta Ingrediente por nombre y numero (Q5_1)

```
SELECT l.cod_piz AS num_piz,
       i.nombre_ing AS nombre,
       i.cod_ing || i.nif_prov AS numero,
       i.precio_ing_eukg AS precio_eu,
       i.ing_despensa_kg AS cant_despensa
  FROM lleva l, ingrediente i
 WHERE l.cod_ing = i.cod_ing AND
       l.nif_prov = i.nif_prov
 ORDER BY nombre, numero;
```

- Consulta Suministro por día y hora (Q5_1_1)

```
SELECT i.cod_ing || i.nif_prov AS numero,
       s.dia_sum AS dia,
       s.hora_sum || ':00' AS hora,
       s.cantidad_sum_kg AS cantidad
  FROM suministro s, ingrediente i
 WHERE s.nif_prov = i.nif_prov AND
       s.cod_ing = i.cod_ing
 ORDER BY dia DESC, hora DESC;
```

1.3.3. Carga de las tablas Cassandra

Para realizar la carga de datos, una vez nos encontramos en el MobaXterm habiendo ejecutado el cqlsh desde la carpeta en la que se han guardado todos los ficheros CSV, todo el código a introducir por consola sigue el mismo formato:

```
cqlsh:pizzeria> COPY <nombre_tabla> (<atributo_1>, <atributo_2>, ..., <atributo_n>)
...   FROM '<fichero.csv'>
...   WITH HEADER = True;
```

- Tabla Empleado

```
COPY empleado (dni, nombre, apellidos, anyos_contratado, edad, activo,
              tipo)
  FROM 'empleado.csv'
 WITH HEADER = true;
```

- Tabla Vehículo

```
COPY vehiculo (matricula, capacidad, modelo, fecha_itv)
  FROM 'vehiculo.csv'
 WITH HEADER = true;
```

- Tabla Suministro

```
COPY suministro (dia, num_ing, hora, cantidad)
  FROM 'suministro.csv'
 WITH HEADER = true;
```

- Tabla Ingrediente

```
COPY ingrediente (numero, nombre, precio_eu, cant_dispensa)
  FROM 'ingrediente.csv'
 WITH HEADER = true;
```

- Tabla Pizza

```
COPY pizza (num_pizza, num_ing, nombre, precio_eu, tamanyo, nombre_ing)
  FROM 'pizza.csv'
 WITH HEADER = true;
```

- Tabla Pedido

```
COPY pedido (codigo, fecha, num_piz, hora, precio_eu, valoracion,  
            cantidad, nombre_piz, tamanyo_piz)  
FROM 'pedido.csv'  
WITH HEADER = true;
```

- Tabla Membresía

```
COPY membresia (nombre, pedidos_necesarios, precio_mes, descuento)  
FROM 'membresia.csv'  
WITH HEADER = true;
```

- Tabla Cliente

```
COPY cliente (dni, nombre_mem, telefonos, nombre, apellidos, periodo_mes)  
FROM 'cliente.csv'  
WITH HEADER = true;
```

- Tabla Pedido por fecha y hora (Q1)

```
COPY pedido_por_fecha_y_hora (grupa, fecha, hora, codigo, precio_eu,  
                               valoracion)  
FROM 'Q1.csv'  
WITH HEADER = true;
```

- Tabla Pizza por cantidad (Q1_1)

```
COPY pizza_por_cantidad (cod_ped, fecha_ped, cantidad, num_piz, nombre,  
                           tamanyo, precio_eu)  
FROM 'Q1_1.csv'  
WITH HEADER = true;
```

- Tabla Cliente por apellidos y nombre (Q2)

```
COPY cliente_por_apellidos_y_nombre (agrupa, apellidos, nombre, dni,  
                                       direccion, telefonos)  
FROM 'Q2.csv'  
WITH HEADER = true;
```

- Tabla Membresía por año y mes (Q2_1)

```
COPY membresia_por_anho_y_mes (dni, periodo_mes, nombre,  
                                 pedidos_necesarios, precio_mes, descuento)  
FROM 'Q2_1.csv'  
WITH HEADER = true;
```

- Tabla Pedido por valoración (Q2_2)

```
COPY pedido_por_valoracion (dni, valoracion, codigo, fecha, hora,  
                            precio_eu)  
FROM 'Q2_2.csv'  
WITH HEADER = true;
```

- Tabla Empleado más de cinco años contratado (Q3)

```
COPY empleado_mas_cinco_anhos (agrupa, anyos_contratado, dni, nombre,
                                 apellidos, edad, activo, tipo)
  FROM 'Q3.csv'
  WITH HEADER = true;
```

- Tabla Vehículo por matrícula (Q4)

```
COPY vehiculo_por_matricula (agrupa, matricula, capacidad, modelo,
                               fecha_itv)
  FROM 'Q4.csv'
  WITH HEADER = true;
```

- Tabla Repartidor por apellidos y nombre (Q4_1)

```
COPY repartidor_por_apellidos_y_nombre (matricula, apellidos, nombre, dni,
                                         anyos_contratado, edad, activo)
  FROM 'Q4_1.csv'
  WITH HEADER = true;
```

- Tabla Pizza por nombre (Q5)

```
COPY pizza_por_nombre (agrupa, nombre, num_piz)
  FROM 'Q5.csv'
  WITH HEADER = true;
```

- Tabla Ingrediente por nombre y numero (Q5_1)

```
COPY ingrediente_por_nombre_y_numero (num_piz, nombre, numero, precio_eu,
                                         cant_despensa)
  FROM 'Q5_1.csv'
  WITH HEADER = true;
```

- Tabla Suministro por día y hora (Q5_1_1)

```
COPY suministro_por_dia_y_hora (numero, dia, hora, cantidad)
  FROM 'Q5_1_1.csv'
  WITH HEADER = true;
```

1.4. Cuarta Etapa. Resolución de consultas CQL

En esta cuarta etapa, se resolverán en CQL algunas consultas de los accesos planteados en el Workflow de la etapa 2.

1. Mostrar los pedidos ordenados por fecha y hora descendenteamente.

```
33 SELECT *
34 FROM pedido_por_fecha_y_hora;
35
```

agrupa	fecha	hora	codigo	precio_eu	valoracion
1	2022-12-30	22:53:00.000000000	00002	78.69	9
1	2022-12-30	05:50:00.000000000	00001	89.45	<<null>>
1	2022-12-30	02:59:00.000000000	00003	44.36	7
1	2022-12-28	14:40:00.000000000	00001	82.91	9
1	2022-12-28	00:11:00.000000000	00002	28.8	8
1	2022-12-27	05:18:00.000000000	00002	40.1	<<null>>
1	2022-12-27	02:14:00.000000000	00001	144.48	<<null>>
1	2022-12-26	14:25:00.000000000	00001	29.79	5
1	2022-12-26	08:20:00.000000000	00002	115.94	8
1	2022-12-25	23:49:00.000000000	00001	109.47	4
1	2022-12-24	21:43:00.000000000	00002	112.28	<<null>>

2. Mostrar las pizzas de un pedido (el segundo del 30 de diciembre del 2022) ordenadas por la cantidad de cada una descendenteamente.

```
32 SELECT *
33 FROM pizza_por_cantidad
34 WHERE fecha_ped = '2022-12-30' AND cod_ped = '00002';
35
```

cod_ped	fecha_ped	cantidad	num_piz	nombre	precio_eu	tamano
00002	2022-12-30	4	PI081	Scampi	25.1	Familiar
00002	2022-12-30	1	PI001	Margherita	4.9	Pequeña

3. Mostrar los clientes ordenados por apellidos y nombre ascendentemente.

```
32 SELECT *
33 FROM cliente_por_apellidos_y_nombre;
34
```

agrupa	apellidos	nombre	dni	direccion	telefonos
1	Abarca Castán	Francisco Javier	15622920D	{tipo_via:'Avenida',nombre_v: {698558778}}	
1	Abarca Álvarez	Héctor	43175707S	{tipo_via:'Avenida',nombre_v: {629455704, 650356328, 6: 629455704}}	
1	Alapont Andreu	Pablo	23022870P	{tipo_via:'Calle',nombre_v: {646626292, 672015820, 6: 646626292}}	
1	Alapont Carboneres	Antonio	83445004S	{tipo_via:'Calle',nombre_v: {613494917, 658908660, 6: 613494917}}	
1	Alapont Fenollosa	Coral	26235107L	{tipo_via:'Calle',nombre_v: {624569812, 690033892}}	
1	Alapont Galán	Francisco	13855501W	{tipo_via:'Calle',nombre_v: {687694981}}	
1	Alapont García	María	41275284J	{tipo_via:'Calle',nombre_v: {694992558}}	
1	Alapont Magraner	Manuel	82288318E	{tipo_via:'Avenida',nombre_v: {621247084}}	
1	Alcayde Belenguer	Francisco	99134440R	{tipo_via:'Avenida',nombre_v: {633932403, 637542251}}	
1	Alcayde Bertó	Ángel	66050785Y	{tipo_via:'Avenida',nombre_v: {624566758, 630256693, 6: 624566758}}	
1	Alcayde Riu	Carla Denisse	50945165G	{tipo_via:'Calle',nombre_v: {619187752}}	

1 statement successfully executed in 564 ms. Retrieved 300 rows

1-300

4. Mostrar las membresías de un cliente (dni 49226378F) ordenadas por período mensual descendente.

```

32 SELECT *
33 FROM membresia_por_anho_y_mes
34 WHERE dni = '49226378F';

Results | Query Trace | 1-27 | X | C

```

dni	periodo_mes	nombre	descuento	pedidos_necesarios	precio_mes
49226378F	{anyo:2022,mes:8}	Bronce	10	10	20
49226378F	{anyo:2021,mes:7}	Plata	20	15	30
49226378F	{anyo:2021,mes:6}	Plata	20	15	30
49226378F	{anyo:2020,mes:12}	Plata	20	15	30
49226378F	{anyo:2019,mes:10}	Plata	20	15	30
49226378F	{anyo:2019,mes:8}	Plata	20	15	30
49226378F	{anyo:2018,mes:6}	Plata	20	15	30
49226378F	{anyo:2018,mes:5}	Plata	20	15	30
49226378F	{anyo:2017,mes:12}	Bronce	10	10	20
49226378F	{anyo:2017,mes:11}	Bronce	10	10	20
49226378F	{anyo:2017,mes:4}	Plata	20	15	30

1 selected statement successfully executed in 55 ms. Retrieved 27 rows

5. Mostrar los pedidos de un cliente (dni 43175707S) ordenados por valoración descendente.

```

32 SELECT *
33 FROM pedido_por_valoracion
34 WHERE dni = '43175707S';

Results | Query Trace | 1-27 | X | C

```

dni	valoracion	codigo	fecha	hora	precio_eu
43175707S	9	00001	2018-11-19	01:29:00.000000000	134.59
43175707S	9	00001	2019-01-24	16:56:00.000000000	60.99
43175707S	8	00002	2013-03-06	12:38:00.000000000	102.9
43175707S	4	00002	2018-06-02	04:53:00.000000000	149.2

6. Mostrar los empleados que lleven más de 5 años contratados.

```

32 SELECT *
33 FROM empleado_mas_cinco_anos;

Results | Query Trace | 1-21 | X | C

```

agrupa	anyos_contratado	dni	activo	apellidos	edad	nombre	tipo
1	8	25095384S	true	Maamri Fuentes	32	Nuria	dependiente
1	8	54953241P	true	Martínez Palop	48	Francisco Javier	repartidor
1	8	76110246A	true	Blasco Parrilla	55	Aina Amelia	repartidor
1	9	60609369E	true	Hidalgo Martínez	57	Marián	repartidor
1	9	91729766Q	true	Tortola Roqueta	58	Marcos	dependiente
1	10	10501008J	true	Jiménez García	47	Nicolás	dependiente
1	10	17557304R	false	Vericat Corbí	45	Antonio	dependiente
1	10	55139522N	true	Raga López	51	Alejandro	repartidor
1	10	89273090Q	true	Carles Vicedo	48	José Manuel	repartidor
1	12	35357688H	false	Peris Rodríguez	51	Fanny	repartidor
1	13	70604624Y	true	Navarro Pallarés	51	Francisco Javier	repartidor

2 statements successfully executed in 340 ms. Retrieved 21 rows

7. Mostrar los vehículos ordenados por matrícula ascendenteamente.

```
32 SELECT *
33 FROM vehiculo_por_matricula;
```

Results	Query Trace			
agrupa	matricula	capacidad	fecha_itv	modelo
1	1768DTH	8	2028-05-18	Peugeot Speedfight 50
1	1930HJV	8	2027-05-11	Aprilia SXR 50
1	1967CSZ	200	2023-12-22	Fiat Doblò Combi SX 1.6 Multijet
1	3077CGL	50	2027-11-22	Ford Fiesta
1	3146NDY	50	2025-04-02	Ford Fiesta
1	3503HWW	50	2027-06-19	Ford Fiesta
1	4208WXP	200	2025-10-06	Fiat Fiorino Cargo
1	4723HSP	200	2023-02-14	Fiat Doblò Combi SX 1.6 Multijet
1	5641CSX	50	2024-06-28	Opel Corsa
1	5940VJT	200	2024-09-06	Fiat Fiorino Cargo
1	6142V7D	200	2028-05-26	Opel Combo Life 1.5 TD Edition F

2 statements successfully executed in 593 ms. Retrieved 15 rows

8. Mostrar los repartidores de un vehículo (matrícula 1768DTH); es decir, los que han usado el vehículo, ordenados por apellidos y nombre ascendentemente.

```
32 SELECT *
33 FROM repartidor_por_apellidos_y_nombre
34 WHERE matricula = '1768DTH';
```

Results	Query Trace					
matricula	apellidos	nombre	dni	activo	anyos_contratado	edad
1768DTH	Blasco Añó	Adam	83901649H	true	5	53
1768DTH	Carles Vicedo	Jose Manuel	89273090Q	true	10	48
1768DTH	Hidalgo Martínez	Marián	60609369E	true	9	57
1768DTH	López Fuentes	Fanny	62205460M	false	3	33
1768DTH	Maamri Belenguer	Anselmo	66366441X	true	14	47
1768DTH	Martínez Palop	Francisco Javier	54953241P	true	8	48
1768DTH	Orón Blasco	Carla	20073711R	true	5	47
1768DTH	Silvestre López	Yassmina	88281426C	true	5	24
1768DTH	Tortola González	Violeta	14672153Q	true	28	59
1768DTH	Verdú Silvestre	Coral	34625495F	true	2	21

1 statement successfully executed in 76 ms. Retrieved 10 rows

9. Mostrar los tipos de pizzas ordenadas por su nombre ascendentemente.

```
32 SELECT *
33 FROM pizza_por_nombre;
```

Results	Query Trace	
agrupa	nombre	num_piz
1	Al huevo	PI112
1	Al huevo	PI113
1	Al huevo	PI114
1	Barbacoa	PI109
1	Barbacoa	PI110
1	Barbacoa	PI111
1	Barbacoa vegana	PI100
1	Barbacoa vegana	PI101
1	Barbacoa vegana	PI102
1	Bologna	PI025
1	Rolonna	PI026

1 statement successfully executed in 314 ms. Retrieved 126 rows

10. Mostrar los ingredientes de una pizza (numero PI110) ordenados por su nombre y número ascendenteamente.

```
32 SELECT *
33 FROM ingrediente_por_nombre_y_numero
34 WHERE num_piz = 'PI110';

Results | Query Trace | 1-15 | X | C | F |
```

num_piz	nombre	numero	cant_despensa	precio_eu
PI110	Carne	IN002S1984725A	27.8	8.22
PI110	Carne	IN003P5301473G	13.4	8.9
PI110	Carne	IN004A92379420	14.8	7
PI110	Orégano	IN001G78224827	0.56	30.2
PI110	Orégano	IN001P5301473G	1.43	34.85
PI110	Orégano	IN001W1481724H	1.32	35.65
PI110	Orégano	IN002C67931174	1.07	36.7
PI110	Orégano	IN002F91358427	1.1	32.34
PI110	Orégano	IN002S4978189H	0.64	31.41
PI110	Orégano	IN003E65830492	1.42	29.37
PI110	Orégano	IN004F65830492	0.76	31.45

1 statement successfully executed in 62 ms. Retrieved 15 rows

11. Mostrar los suministros de un ingrediente (carne número IN004A92379420) ordenados por día y hora descendenteamente.

```
32 SELECT *
33 FROM suministro_por_dia_y_hora
34 WHERE numero = 'IN004A92379420';

Results | Query Trace | 1-15 | X | C | F |
```

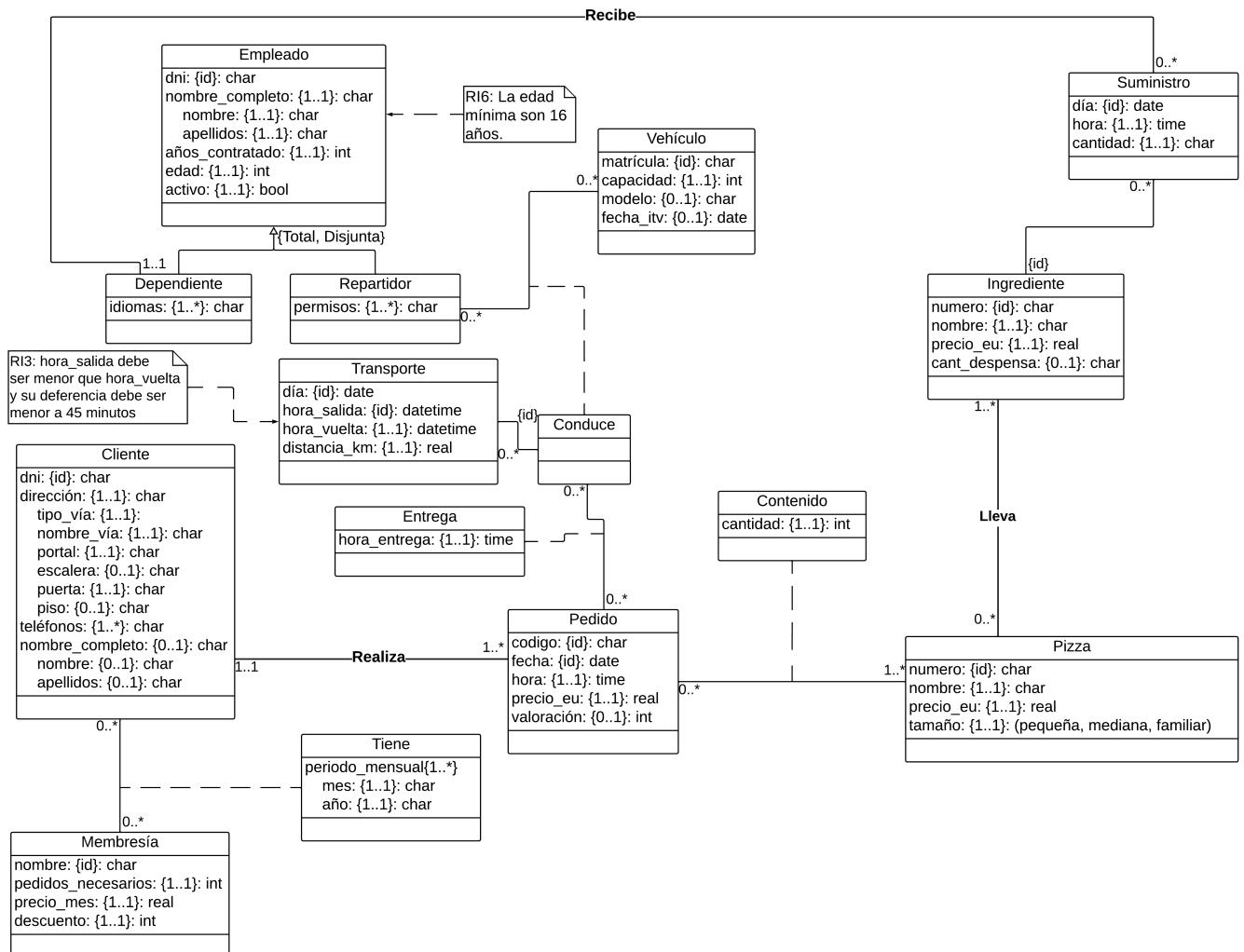
numero	dia	hora	cantidad
IN004A92379420	2021-11-21	13:20:00.000000000	26.18
IN004A92379420	2020-05-30	17:51:00.000000000	39.77
IN004A92379420	2019-02-24	08:10:00.000000000	45.1
IN004A92379420	2019-01-10	00:49:00.000000000	32.65
IN004A92379420	2016-01-28	06:48:00.000000000	13.79
IN004A92379420	2013-02-16	16:55:00.000000000	45.88

2. Segunda Parte. Neo4j

2.1. Primera Etapa. Diseño de la base de datos Neo4j

En esta primera etapa de la segunda parte, se va a proceder al diseño de la base de datos Neo4j. Este diseño se va a realizar a partir del diagrama de clases utilizado también para el diseño de la base de datos Cassandra. En esta ocasión no se contempla la agregación de las clases utilizada con Cassandra, pero sí que se mantiene la poda sugerida. Así, en la figura siguiente se puede observar dicho diagrama.

2.1.1. Diagrama de clases

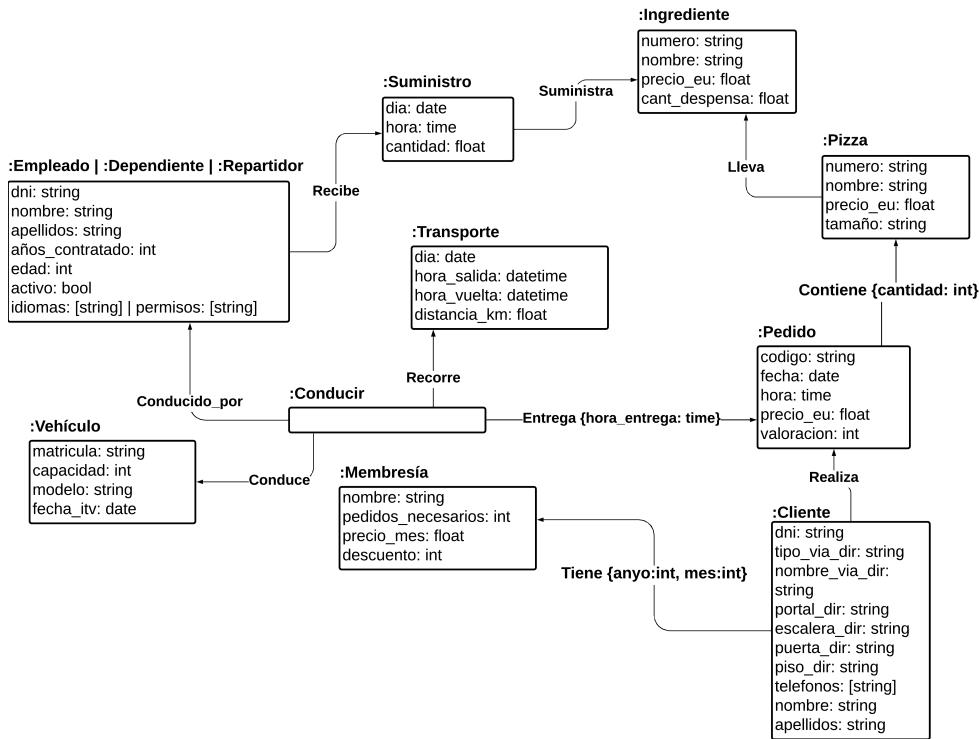


Así pues, obtenemos el diagrama lógico de la base de datos Neo4j, con los atributos de las clases y a modo de grafo ¹ ².

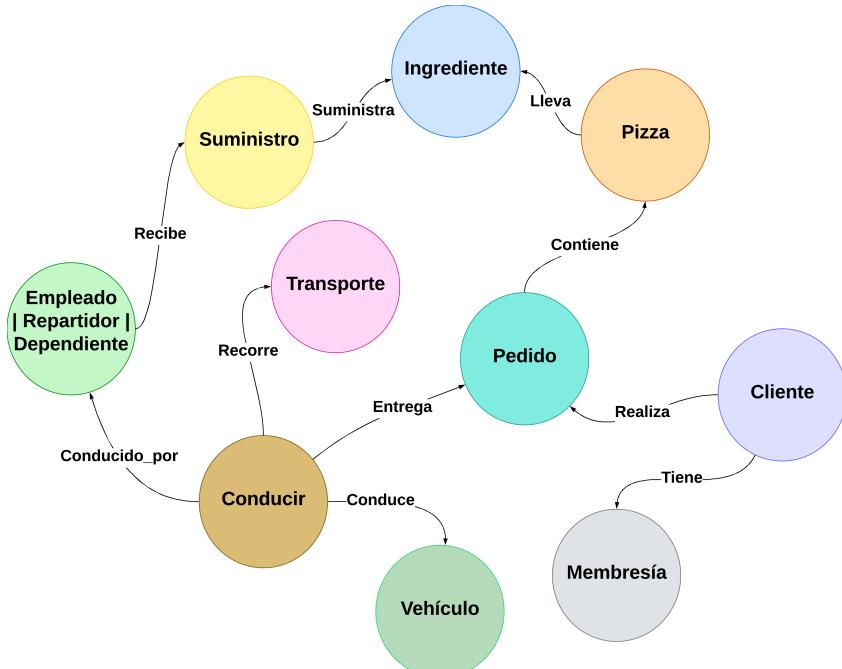
¹Se han cambiado el tipo de dato de algunos atributos de acuerdo a cambios realizados también en el diseño de la base de datos Cassandra

²Se han cambiado los nombres de las clases de forma verbal al infinitivo para poder usar los verbos conjugados en los nombres de los arcos

2.1.2. Diagrama Lógico Neo4j



2.1.3. Diagrama Lógico Neo4j (Grafo)



2.2. Segunda Etapa. Creación y carga de la base de datos Neo4j

En esta segunda etapa se va a proceder a cargar la base de datos en Neo4j. Para ello, se han realizado consultas en Oracle para obtener los archivos CSV que posteriormente se cargarán. En estas consultas no se ha utilizado en ningún momento SELECT * y siempre se seleccionan los atributos uno a uno. Esto se ha hecho, por una parte, para cambiar los nombres de los atributos para que tengan los mismos que las propiedades de los nodos que se implementarán en Neo4j; y por otra parte, para que sea más comprensible visualmente cuales son los atributos que se exportan en cada CSV.

Así, se mostrará la consulta utilizada e inmediatamente seguido el código Cypher. Por último, cada vez que se complete la carga de un grupo de nodos y/o arcos que van conjuntos en el mismo código, se mostrará el diagrama lógico en forma de grafo de cómo queda la base de datos tras cada carga.

2.2.1. Nodo Empleado (Dependiente y Repartidor)

- Consulta Dependiente

```
SELECT e.dni_emp AS dni,
       e.nombre_emp AS nombre,
       e.apellidos_emp AS apellidos,
       e.anyos_contrato_emp AS anyos_contratado,
       e.edad_emp AS edad,
       CASE
           WHEN e.activo_emp = 1 THEN 'true'
           WHEN e.activo_emp = 0 THEN 'false'
       END AS activo
  FROM empleado e
RIGHT JOIN dependiente d ON e.dni_emp = d.dni_dep;
```

- Carga Dependiente

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/dependiente.csv' AS X
CREATE (d:Empleado:Dependiente)
SET d.dni = X.DNI,
    d.nombre = X.NOMBRE,
    d.apellidos = X.APELLIDOS,
    d.anyos_contratado = toInteger(X.ANYOS_CONTRATADO),
    d.edad = toInteger(X.EDAD),
    d.activo = X.ACTIVO = 'true',
    d.idiomas = []
```

- Consulta Idiomas

```
SELECT dni_dep AS dni,
       idioma_dep AS idiomas
  FROM habla_dep;
```

- Carga Idiomas

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/idiomas.csv' AS X
MATCH (d:Dependiente{dni:X.DNI})
SET d.idiomas = d.idiomas + X.IDIOMAS
```

- Consulta Repartidor

```

SELECT e.dni_emp AS dni,
       e.nombre_emp AS nombre,
       e.apellidos_emp AS apellidos,
       e.anyos_contrato_emp AS anyos_contratado,
       e.edad_emp AS edad,
       CASE
           WHEN e.activo_emp = 1 THEN 'true'
           WHEN e.activo_emp = 0 THEN 'false'
       END AS activo
  FROM empleado e
RIGHT JOIN repartidor r ON e.dni_emp = r.dni_rep;
    
```

- Carga Repartidor

```

LOAD CSV WITH HEADERS FROM 'file:///pizzeria/repartidor.csv' AS X
CREATE (r:Empleado:Repartidor)
SET r.dni = X.DNI,
    r.nombre = X.NOMBRE,
    r.apellidos = X.APELLIDOS,
    r.anyos_contratado = toInteger(X.ANYOS_CONTRATADO),
    r.edad = toInteger(X.EDAD),
    r.activo = X.ACTIVO = 'true',
    r.permisos = []
    
```

- Consulta Permisos

```

SELECT dni_rep AS dni,
       permiso_rep AS permisos
  FROM permiso_rep;
    
```

- Carga Permisos

```

LOAD CSV WITH HEADERS FROM 'file:///pizzeria/permisos.csv' AS X
MATCH (r:Repartidor{dni:X.DNI})
SET r.permisos = r.permisos + X.PERMISOS
    
```

- Grafo



2.2.2. Nodo Ingrediente

- Consulta Ingrediente

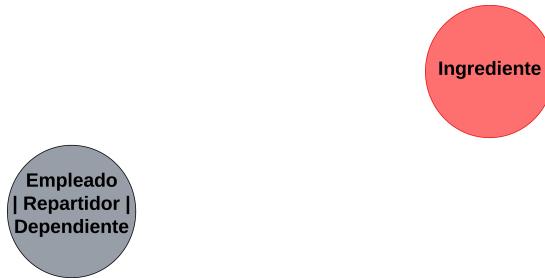
```

SELECT nif_prov || cod_ing AS numero,
       nombre_ing AS nombre,
       precio_ing_eukg AS precio_eu,
       ing_despensa_kg AS cant_despensa
  FROM ingrediente;
    
```

- Carga Ingrediente

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/ingrediente.csv' AS X
CREATE (i:Ingrediente)
SET i.numero = X.NUMERO,
    i.nombre = X.NOMBRE,
    i.precio_eu =toFloat(X.PRECIO_EU),
    i.cant_despensa =toFloat(X.CANT_DESPENSA)
```

- Grafo



2.2.3. Nodo Suministro y Arcos Suministra y Recibe

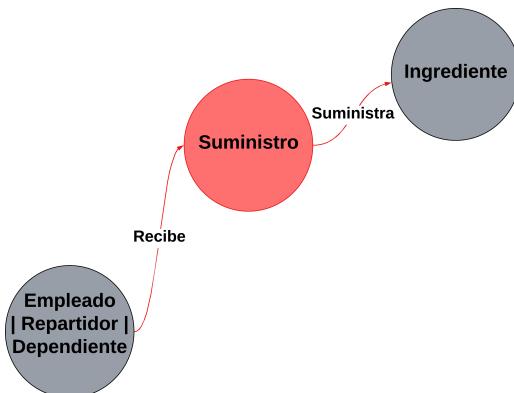
- Consulta Suministro / Suministra / Recibe

```
SELECT nif_prov || cod_ing AS numero,
       dia_sum AS dia,
       hora_sum AS hora,
       cantidad_sum_kg AS cantidad,
       dni_dep AS dni
  FROM suministro;
```

- Carga Suministro / Suministra / Recibe

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/suministro.csv' AS X
MATCH (d:Dependiente{dni:X.DNI}),(i:Ingrediente{numero:X.NUMERO})
CREATE (d)-[rr:Recibe]->(s:Suministro)-[rs:Suministra]->(i)
SET s.numero = date(X.DIA),
    s.nombre = time(X.HORA),
    s.precio_eu = toFloat(X.CANTIDAD)
```

- Grafo



2.2.4. Nodo Pizza

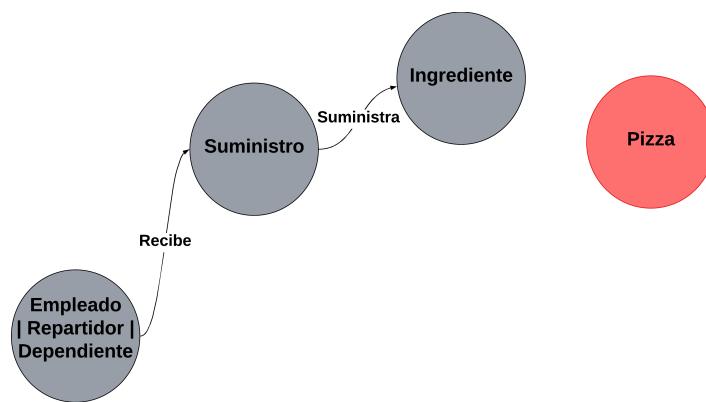
- Consulta Pizza

```
SELECT cod_piz AS numero,
       nombre_piz AS nombre,
       precio_piz_eu AS precio_eu,
       tamanyo_piz AS taamnyo
  FROM pizza;
```

- Carga Pizza

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/pizza.csv' AS X
CREATE (p:Pizza)
SET p.numero = X.NUMERO,
p.nombre = X.NOMBRE,
p.precio_eu =toFloat(X.PRECIO_EU),
p.tamanyo = X.TAMANYO
```

- Grafo



2.2.5. Arco Lleva

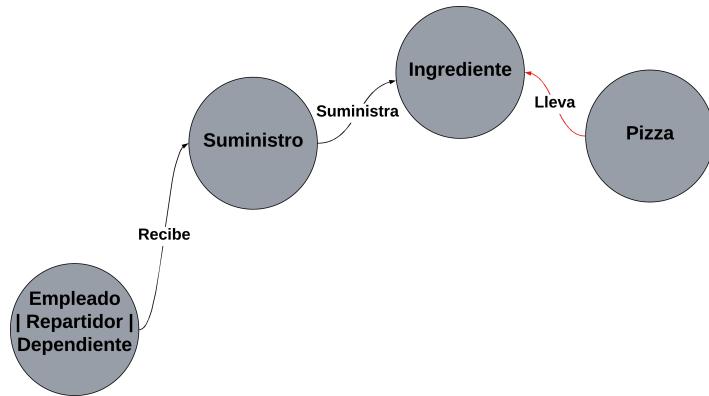
- Consulta Lleva

```
SELECT cod_piz,
       nif_prov || cod_ing AS cod_ing
  FROM lleva;
```

- Carga Lleva

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/lleva.csv' AS X
MATCH (p:Pizza{numero:X.COD_PIZ}),(i:Ingrediente{numero:X.COD_ING})
CREATE (p)-[r:Lleva]->(i)
```

- Grafo



2.2.6. Nodo Cliente

- Consulta Cliente

```

SELECT dni_cli AS dni,
       nombre_cli AS nombre,
       apellidos_cli AS apellidos,
       tipo_via_dir_cli AS tipo_via_dir,
       nombre_via_dir_cli AS nombre_via_dir,
       portal_dir_cli AS portal_dir,
       esc_dir_cli AS escalera_dir,
       puerta_dir_cli AS puerta_dir,
       piso_dir_cli AS piso_dir
  FROM cliente;
  
```

- Carga Cliente

```

LOAD CSV WITH HEADERS FROM 'file:///pizzeria/cliente.csv' AS X
CREATE (c:Cliente)
SET c.dni = X.DNI,
    c.nombre = X.NOMBRE,
    c.apellidos = X.APELLIDOS,
    c.tipo_via_dir = X.TIPO_VIA_DIR,
    c.nombre_via_dir = X.NOMBRE_VIA_DIR,
    c.portal_dir = X.PORTAL_DIR,
    c.escalera_dir = X.ESCALERA_DIR,
    c.puerta_dir = X.PUERTA_DIR,
    c.piso_dir = X.PISO_DIR,
    c.telefonos = []
  
```

- Consulta Teléfonos

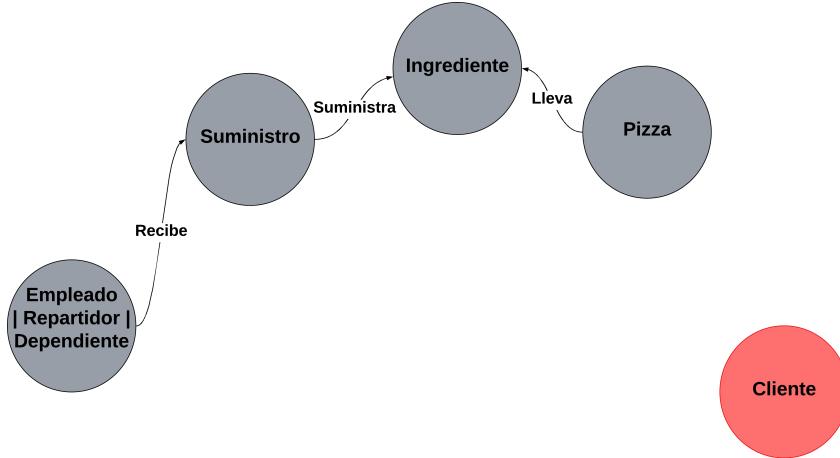
```

SELECT dni_cli AS dni,
       tlf_cli AS telefonos
  FROM tlf_cliente;
  
```

- Carga Teléfonos

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/telefonos.csv' AS X
MATCH (c:Cliente{dni:X.DNI})
SET c.telefonos = c.telefonos + X.TELEFONOS
```

- Grafo



2.2.7. Nodo Pedido y Arco Realiza

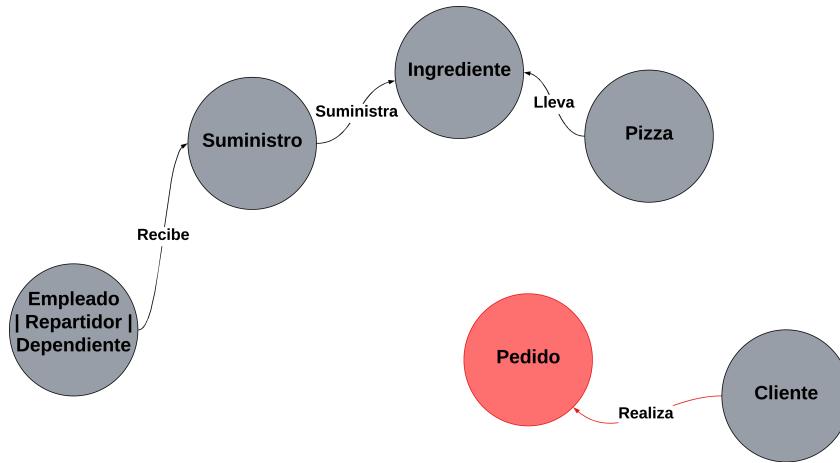
- Consulta Pedido / Realiza

```
SELECT cod_ped AS codigo,
       fecha_ped AS fecha,
       hora_ped AS hora,
       precio_ped_eu AS precio_eu,
       valoracion_ped AS valoracion,
       dni_cli AS dni
  FROM pedido;
```

- Carga Pedido / Realiza

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/pedido.csv' AS X
MATCH (c:Cliente{dni:X.DNI})
CREATE (c)-[r:Realiza]->(p:Pedido)
SET p.codigo = X.CODIGO,
    p.fecha = date(X.FECHA),
    p.hora = time(X.HORA),
    p.precio_eu = toFloat(X.PRECIO_EU),
    p.valoracion = toInteger(X.VALORACION)
```

- Grafo



2.2.8. Arco Contiene

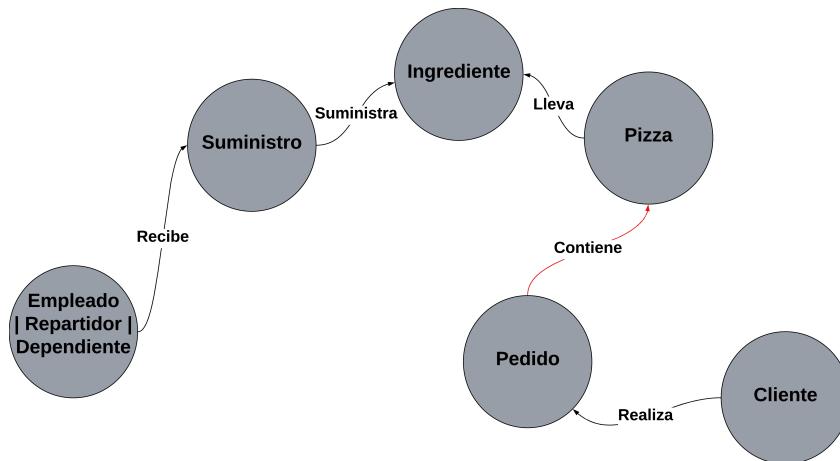
- Consulta Contiene

```
SELECT cod_ped AS codigo,
       fecha_ped AS fecha,
       cod_piz AS numero,
       cantidad
  FROM contenido;
```

- Carga Contiene

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/contenido.csv' AS X
MATCH (p:Pedido{codigo:X.CODIGO,fecha:date(X.FECHA)}),
      (pi:Pizza{numero:X.NUMERO})
CREATE (p)-[r:Contiene]->(pi)
SET r.cantidad = toInteger(X.CANTIDAD)
```

- Grafo



2.2.9. Nodo Membresía

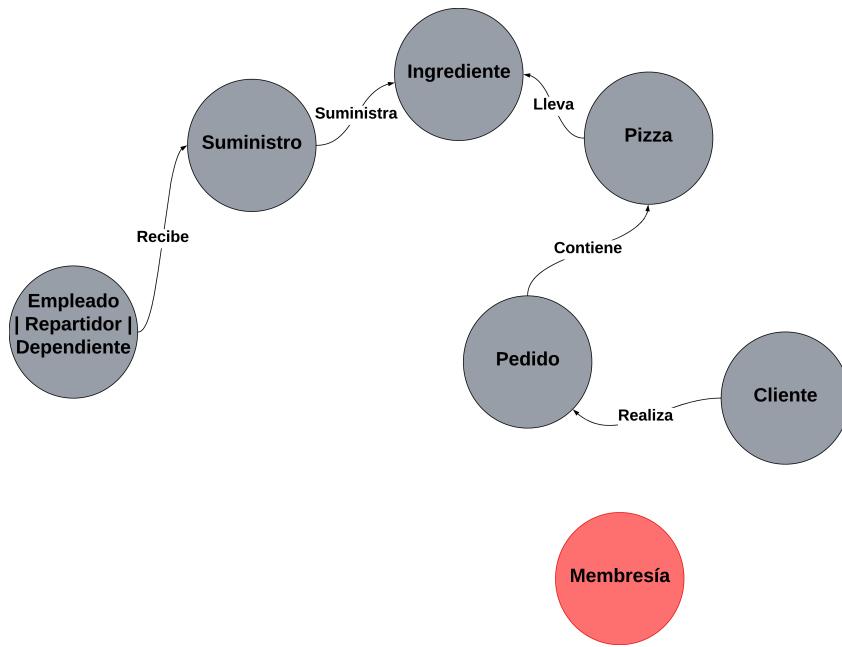
- Consulta Membresía

```
SELECT nombre_mem AS nombre,
       pedidos_mem AS pedidos_necesarios,
       precio_mes_mem_eu AS precio_mes,
       descuento_mem_per AS descuento
  FROM membresia;
```

- Carga Membresía

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/membresia.csv' AS X
CREATE (m:Membresia)
SET m.nombre = X.NOMBRE,
    m.pedidos_necesarios = toInteger(X.PEDIDOS_NECESSARIOS),
    m.precio_mes = toFloat(X.PRECIO_MES),
    m.descuento = toInteger(X.DESCUENTO)
```

- Grafo



2.2.10. Arco Tiene

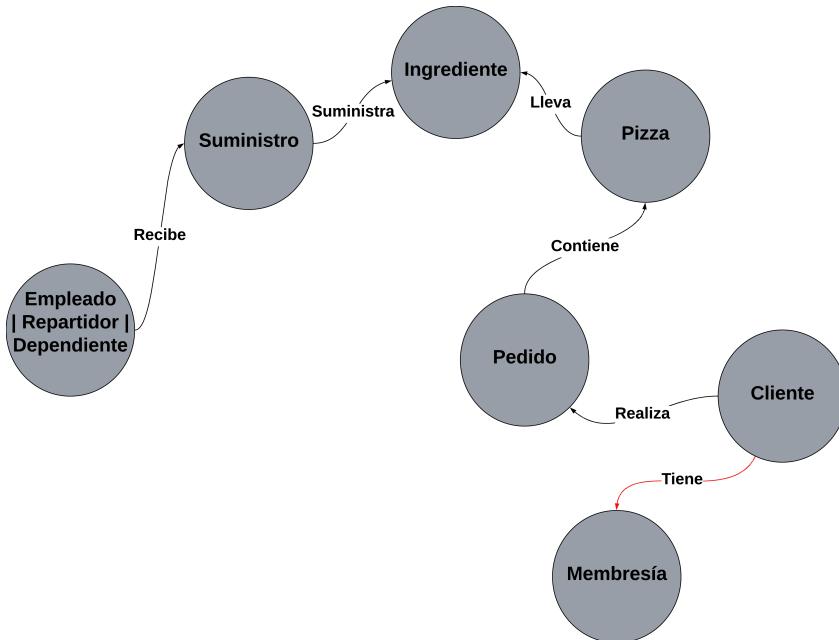
- Consulta Tiene

```
SELECT dni_cli AS dni,
       nombre_mem AS nombre,
       mes_orden AS mes,
       anyo_mem AS anyo
  FROM meses_mem;
```

- Carga Tiene

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/tiene.csv' AS X
MATCH (m:Membresia{nombre:X.NOMBRE}),(c:Cliente{dni:X.DNI})
CREATE (c)-[r:Tiene]->(m)
SET r.anio = toInteger(X.AÑO),
    r.mes = toInteger(X.MES)
```

- Grafo



2.2.11. Nodo Vehículo

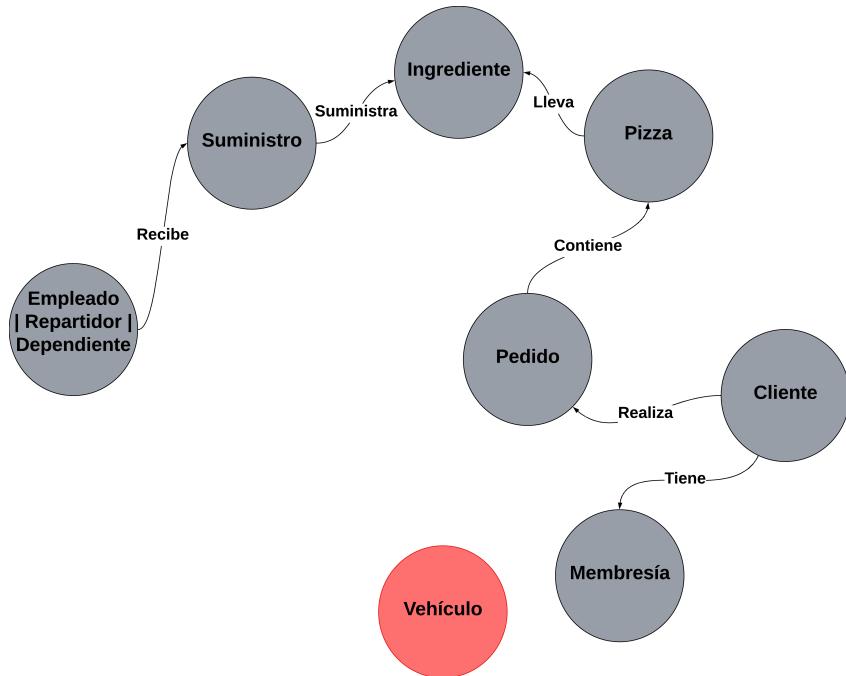
- Consulta Vehículo

```
SELECT matricula_veh AS matricula,
       capacidad_veh AS capacidad,
       modelo_veh AS modelo,
       fecha_itv_veh AS fecha_itv
  FROM vehiculo;
```

- Carga Vehículo

```
LOAD CSV WITH HEADERS FROM 'file:///pizzeria/vehiculo.csv' AS X
CREATE (v:Vehiculo)
SET v.matricula = X.MATRICULA,
    v.capacidad = toInteger(X.CAPACIDAD),
    v.modelo = X.MODELO,
    v.fecha_itv = date(X.FECHA_ITV)
```

- Grafo



2.2.12. Nodo Conducir y Arcos Conducido_por y Conduce

- Consulta Conducir / Conducido_por / Conduce

```

SELECT dni_rep AS dni,
       matricula_veh AS matricula
FROM conduce;

```

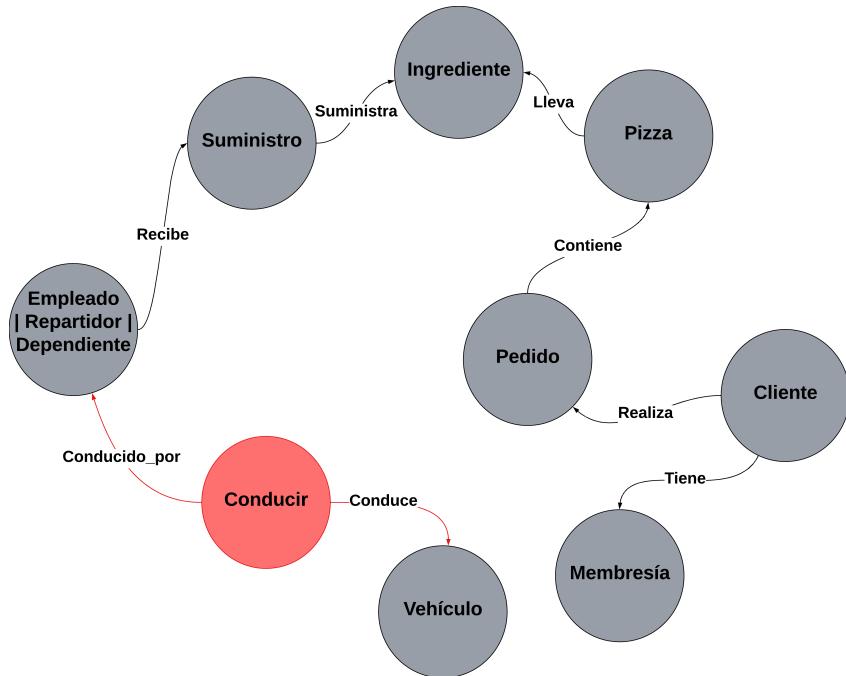
- Carga Conducir / Conducido_por / Conduce

```

LOAD CSV WITH HEADERS FROM 'file:///pizzeria/conduce.csv' AS X
MATCH (r:Repartidor{dni:X.DNI}),(v:Vehiculo{matricula:X.MATRICULA})
CREATE (v)->[rc:Conducir]-(c:Conducir)-[rcp:Conducido_por]->(r)

```

- Grafo



2.2.13. Nodo Transporte y Arco Recorre

- Consulta Transporte / Recorre

```

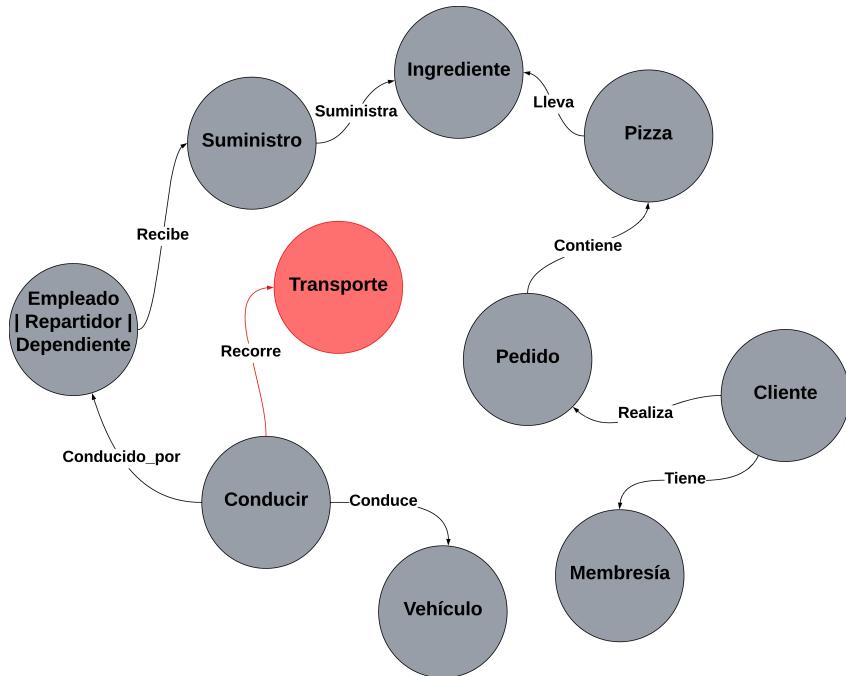
SELECT dni_rep AS dni,
       matricula_veh AS matricula,
       dia_trans AS dia,
       hora_salida_trans AS hora_salida,
       hora_llegada_trans AS hora_llegada,
       distancia_trans_km AS distancia_km
  FROM transporte;
  
```

- Carga Transporte / Recorre

```

LOAD CSV WITH HEADERS FROM 'file:///pizzeria/transporte.csv' AS X
MATCH (r:Repartidor{dni:X.DNI})->[rcp:Conducido_por]-
      (c:Conducir)-[rc:Conduce]->(v:Vehiculo{matricula:X.MATRICULA})
CREATE (c)-[rr:Recorre]->(t:Transporte)
SET t.dia = date(X.DIA),
    t.hora_salida = time(X.HORA_SALIDA),
    t.hora_llegada = time(X.HORA_LLEGADA),
    t.distancia_km = toFloat(X.DISTANCIA_KM)
  
```

- Grafo



2.2.14. Arco Entrega

- Consulta Entrega

```

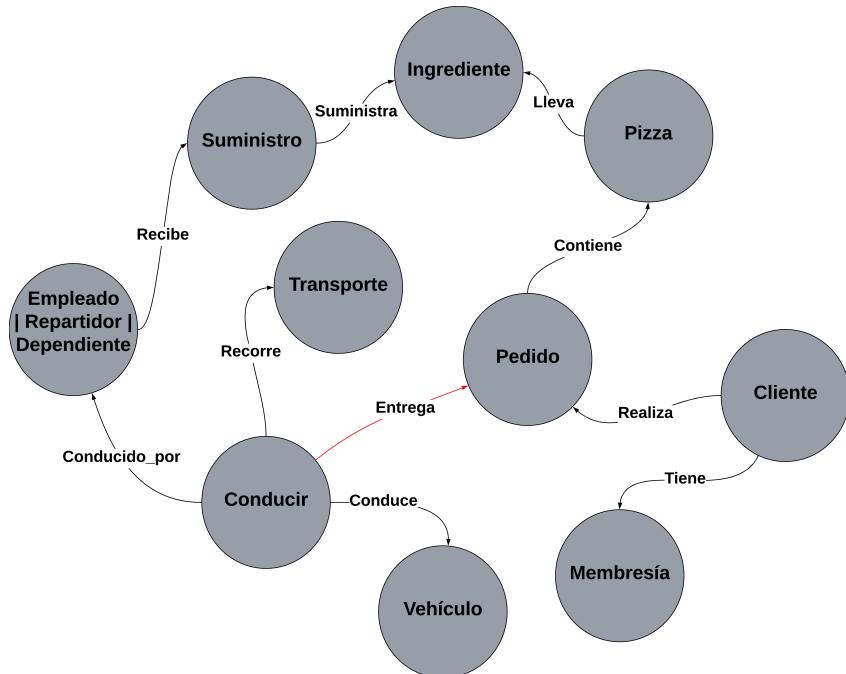
SELECT dni_rep AS dni,
       matricula_veh AS matricula,
       cod_ped AS codigo,
       fecha_ped AS fecha,
       hora_entrega_ped AS hora_entrega
  FROM entrega;
  
```

- Carga Entrega

```

LOAD CSV WITH HEADERS FROM 'file:///pizzeria/entrega.csv' AS X
MATCH (r:Repartidor{dni:X.DNI})<-[rcp:Conducido_por]->(c:Conducir)-[rc:Conduce]->(v:Vehiculo{matricula:X.MATRICULA}),
      (p:Pedido{codigo:X.CODIGO,fecha:date(X.FECHA)})
CREATE (c)-[e:Entrega]->(p)
SET e.hora_entrega = time(X.HORA_ENTREGA)
  
```

■ Grafo



2.3. Tercera Etapa. Consultas en Cypher

▪ Consulta Cypher 1

Mostrar el dni, nombre y apellidos de los 5 repartidores que más kilómetros han recorrido ordenados de más a menos kilómetros recorridos; y mostrar también la cantidad de kilómetros recorrida.

```
1 MATCH (r:Repartidor)-[:Conducido_por]-(:Conducir)-[:Recorre]->
  (t:Transporte)
2 WITH r, SUM(t.distancia_km) AS distanciaTotal
3 RETURN r.dni AS DNI, r.nombre AS Nombre, r.apellidos AS Apellidos,
       distanciaTotal AS Kilómetros
4 ORDER BY distanciaTotal DESC
5 LIMIT 5
```

table

	DNI	Nombre	Apellidos	Kilómetros
1	"14672153Q"	"Violeta"	"Tortola González"	4600.400000000002
2	"66366441X"	"Anselmo"	"Maamri Belenguer"	3785.529999999997
3	"70604624Y"	"Francisco Javier"	"Navarro Pallardó"	3439.310000000013
4	"35357688H"	"Fanny"	"Peris Rodríguez"	3272.549999999984
5	"62005460MA"	"Eduardo"	"El Águila Encantada"	2007.040000000002

▪ Consulta Cypher 2

Mostrar el dni, nombre y apellidos de los dependientes que hablen 2 idiomas o más y hayan recibido al menos 5 suministros.

```
1 MATCH (d:Dependiente)-[:Recibe]->(s:Suministro)
2 WITH d, COUNT(s) AS recibidos
3 WHERE SIZE(d.idiomas) ≥ 2 AND recibidos ≥ 5
4 RETURN d.dni AS DNI, d.nombre AS Nombre, d.apellidos AS Apellidos,
       recibidos AS Suministros
5 ORDER BY recibidos
```

table

	DNI	Nombre	Apellidos	Suministros
1	"17557304R"	"Antonio"	"Vericat Corbí"	33
2	"80501699C"	"Rubén"	"Bonifacio Bru"	35
3	"69658360P"	"Javier"	"Hernández Tudela"	43
4	"63330141D"	"Adrià"	"Lladró Escriche"	44

▪ Consulta Cypher 3

Mostrar el dni, nombre y apellidos de los repartidores activos que lleven menos de 10 años contratados cuyos pedidos entregados tengan al menos un pedido con 9 de valoración; y mostrar también cuantos pedidos tiene cada repartidor con 9 de valoración, ordenados por esta cantidad descendente.

```
1 MATCH (r:Repartidor)←[:Conducido_por]-(:Conducir)-[:Entrega]→  
  (p:Pedido)  
2 WHERE r.activo AND r.anyos_contratado < 10 AND p.valoracion = 9  
3 WITH r, COUNT(p) AS Pedidos  
4 RETURN r.dni AS DNI, r.nombre AS Nombre, r.apellidos AS Apellidos,  
      Pedidos  
5 ORDER BY Pedidos DESC
```

	DNI	Nombre	Apellidos	Pedidos
1	"60609369E"	"Marián"	"Hidalgo Martínez"	86
2	"34625495F"	"Coral"	"Verdú Silvestre"	73
3	"20073711R"	"Carla"	"Orón Blasco"	63
4	"54953241P"	"Francisco Javier"	"Martínez Palop"	59
5	--	--	--	--

▪ Consulta Cypher 4

Mostrar el nombre de los ingredientes que están en al menos 6 tipos de pizzas diferentes, mostrando también la cantidad de pizzas en las que están y en orden alfabético de los ingredientes.

```
1 MATCH (i:Ingrediente)←[:Lleva]-(p:Pizza)  
2 WITH i, count(DISTINCT p.nombre) AS Pizzas  
3 WHERE Pizzas ≥ 6  
4 RETURN DISTINCT i.nombre AS Nombre, Pizzas  
5 ORDER BY i.nombre
```

	Nombre	Pizzas
1	"Alcachofas"	6
2	"Champiñones"	13
3	"Jamón cocido"	15
4	"Orégano"	37
5	"Queso"	38

▪ Consulta Cypher 5

Mostrar el dni, nombre y apellidos de cada repartidor ordenados por apellidos, junto con la matrícula y el modelo del vehículo con el que han recorrido más kilómetros, mostrando también los kilómetros recorridos con ese transporte.

```

1 MATCH (r:Repartidor)←[:Conducido_por]-(c:Conducir)-[:Conduce]→(v:Vehiculo), (c)-[:Recorre]→
  (t:Transporte)
2 WITH r, v, SUM(t.distancia_km) AS kilometros
3 ORDER BY r.dni, kilometros DESC
4 WITH r, COLLECT({vehiculo: v, kilometros: kilometros})[0] AS datos
5 RETURN r.dni AS DNI, r.nombre AS Nombre, r.apellidos AS Apellidos, datos.vehiculo.matricula AS
  Matrícula, datos.vehiculo.modelo AS Modelo, datos.kilometros AS Kilómetros
6 ORDER BY r.apellidos
  
```

	DNI	Nombre	Apellidos	Matrícula	Modelo	Kilómetros
1	"83901649H"	"Adam"	"Blasco Añó"	"9317SFH"	"Ford Fiesta"	302.64
2	"76110246A"	"Aina Amelia"	"Blasco Parrilla"	"8981NCR"	"Fiat Doblo Combi SX 1.6 Multijet"	776.809999999998
3	"89273090Q"	"Jose Manuel"	"Carles Vicedo"	"3146NDY"	"Ford Fiesta"	387.470000000001
4	"32354115S"	"Luis"	"Garijo Espinosa"	"3077CGL"	"Ford Fiesta"	651.329999999998

▪ Consulta Cypher 6

Mostrar el dni, nombre y apellidos de los clientes que han tenido membresía en más de 5 meses y todas han sido del mismo tipo, mostrando también el tipo de la membresía.

```

1 MATCH (c:Cliente)-[:Tiene]→(m:Membresia)
2 WITH c, COLLECT(DISTINCT m.nombre) AS membresias, COUNT(m) AS cantidad
3 WHERE cantidad > 5 AND SIZE(membresias) = 1
4 RETURN c.dni AS DNI, c.nombre AS Nombre, c.apellidos AS Apellidos, membresias[0] AS Membresías
  
```

	DNI	Nombre	Apellidos	Membresías
1	"62862558Q"	"Almudena"	"Blasco García"	"Oro"
2	"66923263W"	"José"	"Comes López"	"Oro"
3	"89406011C"	"Natalia"	""	"Oro"
4	"78398992L"	"Carlos"	"Raga Sebastiá"	"Oro"
5	"42586274A"	"Natalia"	"Rodríguez López"	"Oro"

▪ Consulta Cypher 7

Mostrar el dni, nombre y apellidos de cada cliente que haya pedido una misma pizza al menos 10 veces, junto con el nombre y tamaño de la pizza y las veces que la ha pedido el cliente.

```

1 MATCH (p:Pizza)←[co:Contiene]-(:Pedido)←[:Realiza]-(c:Cliente)
2 WITH p, c, SUM(co.cantidad) AS pizzas
3 WHERE pizzas ≥ 10
4 RETURN c.dni AS DNI, c.nombre AS Nombre, c.apellidos AS Apellidos,
      p.nombre AS Pizza, p.tamano AS Tamaño, pizzas AS Cantidad
    
```

	DNI	Nombre	Apellidos	Pizza	Tamaño	Cantidad
1	"85712336F"	"Nicolás"	"Álvarez Pastor"	"Tutto"	"Mediana"	10
2	"90299071J"	"Pablo"	"Pedrós Medina"	"Tropical"	"Mediana"	12
3	"93026391Q"	"Carlos"	"García de León"	"Calzone"	"Pequeña"	10
4	"36270723E"	"Eurídice"	"de León Chirivella"	"Siciliana"	"Familiar"	10
5	"73968386H"	"Damián"	"García García"	"Tutto"	"Mediana"	10

▪ Consulta Cypher 8

Mostrar las 10 pizzas más vendidas a partir de las 10 de la noche hasta las 4 y media de la mañana.

```

1 MATCH (p:Pedido)-[co:Contiene]→(pi:Pizza)
2 WHERE p.hora > time("22:00") OR p.hora < time("04:30")
3 WITH pi, sum(co.cantidad) AS cantidadTotal
4 RETURN pi.numero AS Código, pi.nombre AS Nombre, pi.tamano AS
      Tamaño, pi.precio_eu AS Precio, cantidadTotal AS Vendidas
5 ORDER BY cantidadTotal DESC
6 LIMIT 10
    
```

	Código	Nombre	Tamaño	Precio	Vendidas
1	"PI112"	"Al huevo"	"Pequeña"	6.45	113
2	"PI121"	"Campagnola"	"Pequeña"	6.4	107
3	"PI004"	"Napoli"	"Pequeña"	5.25	102
4	"PI116"	"Carbonara"	"Mediana"	13.3	102