PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2413 — Bases de Datos

Proyecto semestral - Modelo Entidad Relación

Entrega 1

Trabajo grupal

Integrantes:

Anonymous

Fecha de entrega: 14 de Abril de 2024

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

Índice de contenido

1. Decisiones y consideraciones sobre el proyecto

- Como ha sido explicado en el ppt de la clase, hemos puesto los atributos que son llaves primarias de entidades fuertes en el diagrama subrayados, mientras que los atributos que son llaves parciales de entidades débiles con subrayado punteado (dashed).

- Hemos considerado el actualizar el estado del pedido en diferentes partes del flujo del diagrama, y nos hemos asegurado que en todos lados es posible actualizar su estado correctamente.

- El costo de despacho es fijo ya que sólo se hacen entregas dentro de la ciudad, y sólo hay una rebaja del descuento en cada despacho si el Cliente que realiza el Pedido tiene una suscripción, la que le otorga el beneficio del descuento.

- Para que se tenga registro de los platos que componen un pedido, se tiene la tabla de la relación se_compone que almacena el Platos.id y el Pedido.id, así se tiene una relación n a n, que identifica cada Plato y a qué Pedido corresponde.

- El modelo relacional ha sido pensado con psudotipos de datos (por ejemplo, STRING llevado a SQL debería ser VARCHAR(longitud)), pero algunos tipos de datos que utilizamos no estaban en los pdf oficiales de las clases, así que los escribimos en su notación SQL.

- Las FK han sido subrayadas en el modelo relacional (esquema), porque en los pdf oficiales de las clases hay ejemplos de llaves foráneas subrayas subrayadas.

- Pedido.estado es un tipo de dato CATEGORY, con opción de ser; pendiente, en preparación, entregado a despachador o entregado a cliente.

- Platos.tamano es un tipo de dato CATEGORY, con opción; individual, mediano y familiar.

- En la relación Administrador-administra-EmpresaDespacho, el atributo administra.accion es de tipo CATEGORY, y tiene como opciones; agregar, modificar o eliminar.

- Suscripcion.medio_de_pago es un tipo de dato CATEGORY, que tiene como opciones; Mastercard o Visa. Y Suscripcion.tipo_suscripcion también es un tipo de dato CATEGORY, y puede ser mensual o anual.

- Las evaluaciones, tanto de despechador como de cliente que se presentan en Pedidos, será anotado de 1 a 5 estrellas respectivamente.

- La empresa despachadora solo hace entregas dentro de la misma ciudad.

- Dentro de la entidad de Dirección, departamento adopta el valor 0 para casas, y los demás números positivos para el número de departamento.

- El atributo de ingredientes STRING[] en la entidad Plato es una lista de string, un tipo de dato de PostgreSQL.

- La entidad Suscripción provee de una rebaja en cada despacho de un 50 % a los Clientes que estén suscritos.

- La entidad de Plato puede considerarse como un menú de comidas.

- Hemos leído algunas Issues y, dado que se ha mencionado que el apartado de entrega de pedidos es de libre elección, hemos tomado la opción de entregar todo en un solo pedido por parte de la Empresa de Despacho.

- Notar que la entidad ServicioDespacho es equivalente a servicio despachador y la entidad EmpresaDespacho es equivalente a empresa despachadora.

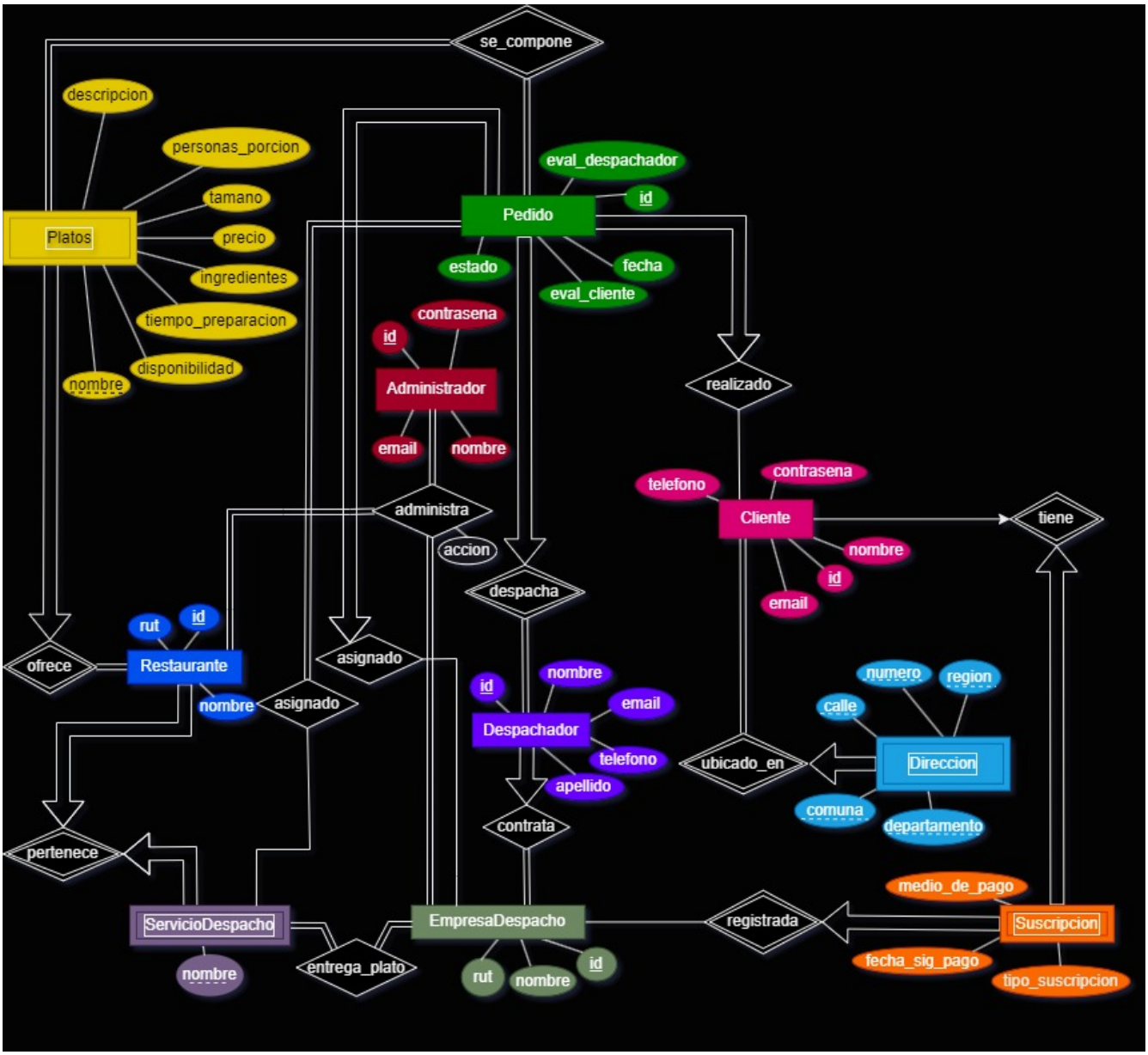
- Un usuario solo puede tener un email y un teléfono registrado dentro la app, otro usuario no puede tener el mismo teléfono o email.

- El despachador se guiará por GPS, no por código postal para entregar el pedido.

- Dos servicios despachadores de un restaurante no pueden tener el mismo nombre, ya que sólo existe un único servicio despachador por restaurante.

- Se asume que el Servicio de Despacho es el servicio que brinda el restaurante para entregar las compras realizadas en línea, no obstante, quien se hace cargo del envío hacia el cliente es la Empresa de Despacho mediante el despachador que tiene contratado.

2. Diagrama Entidad - Relación



3. Esquema o Modelo Relacional

3.1. Entidades

- Cliente(id: INT, email: STRING, nombre: STRING, telefono: INT, contraseña: STRING)

- Direccion(Cliente.id: INT, region: STRING, comuna: STRING,
calle: STRING, numero: INT, departamento: INT)

- Pedido(id: INT, Cliente.id: INT, Despachador.id: INT,
estado: CATEGORY, eval_cliente: INT, eval_despachador: INT, fecha: TIMESTAMP)

- Suscripcion(id: INT,Cliente.id: INT,EmpresaDespacho.id: INT, fecha_sig_pago: DATE,
  medio_de_pago: CATEGORY,
tipo_suscripcion: STRING)

- EmpresaDespacho(id: INT, rut: STRING, nombre: STRING)

- ServicioDespacho(nombre: INT, Restaurante.id: INT,
nombre: STRING)

- Despachador(id: INT,EmpresaDespacho.id: INT,
telefono: INT, nombre: STRING, apellido: STRING, email: STRING)

- Restaurante(id: INT, rut: STRING, nombre: STRING)

- Administrador(id: INT, nombre: STRING, contrasena: STRING, email: STRING)

- Platos(Restaurante.id: INT: INT, nombre: STRING, descripcion: STRING, tamano: CATEGORY,
  precio: INT, personas_porcion: INT, disponibilidad: BOOL, tiempo_preparacion: INTERVAL,
  ingredientes: STRING[])

3.2. Relaciones

Aquí se detallan las relaciones entre dos entidades de la forma Entidad_A-relación-Entidad_B

- Administrador-administra-EmpresaDespacho(Administrador.id: INT, EmpresaDespacho.id: INT, Restaurante.id: INT, accion: CATEGORY)

- Pedido-se_compone-Platos(Platos.id: INT, Pedido.id: INT)

- Pedido-asignado-Servicio(Pedido.id: INT, ServicioDespacho.id: INT)

- Pedido-asignado-EmpresaDespacho(Pedido.id: INT, EmpresaDespacho.id: INT)

- ServicioDespacho-entrega_plato-EmpresaDespacho(EmpresaDespacho.id: INT, ServicioDespacho.id: INT, Pedido.id: INT)

4. Preguntas

4.1. Identifique las entidades débiles en el modelo E/R y explique para cada una de ellas por qué son débiles.

- ServicioDespacho: Es débil porque pertenece a un restaurante; según enunciado, cada restaurant debe tener un servicio despachador, por lo que ServicioDespacho pertenece a un Restaurante en específico, siendo este último una entidad fuerte con su RUT (registro único tributario) como identificador único.

- Platos: Es débil porque, dado que cada plato es ofrecido por un Restaurante en específico, el plato a servir depende del Restaurante, así que un plato será siempre dependiente de un restaurante que lo prepare (la entidad Platos es un menú de comida).

- Direccion: Es débil porque una dirección, en este sistema, depende del Cliente (entidad fuerte),
no pueden existir clientes sin dirección porque ahí les llegará el pedido, y una dirección no puede existir si no es de un Cliente.

- Suscripción: Similar a lo anterior, es débil porque una suscripción lo es de un Cliente, que es una entidad fuerte.

4.2. Identifique/defina las llaves primarias, parciales en el modelo E/R y justifique su uso.

4.2.1. Respuesta a llaves primarias:

Llaves primarias de Cliente: id, elegimos el atributo id de Cliente de tipo INT, porque es el identificador único más pequeño, y no se repite por ser AUTO GENERATE, además es no nulo por tener la propiedad NOT NULL (en SQL).

Llaves primarias de Direccion: Cliente.id, y todos los otros atributos, porque nos parece que la llave primaria necesariamente estar compuesta de todos los atributos, dado que un cliente puede tener múltiples direcciones, cada una diferente de la otra por tener cada una de ellas un id diferente.

Llaves primarias de Pedido: Pedido.id, nos parece que el id del pedido y el id del cliente son suficientes para identificarlo de manera única, y elegirlo como llave primaria (PK) compuesta.

Llaves primarias de Suscripcion: EmpresaDespacho.id y Cliente.id, ya que así podemos identificar a qué cliente corresponde una suscripción y, a su vez, a qué empresa está suscrito dicho cliente.

Llaves primarias de EmpresaDespacho: id, al principio nos parecía que el RUT era más adecuado que en vez de elegir
un id, pero dado que este puede cambiar por razones legales, decidimos no hacerlo y optar por un id autogenerado.

Llaves primarias de ServicioDespacho: Elegimos Restaurante.id y nombre, ya que por definición de nuestro desarrollo, dos servicios despachadores de un restaurante no pueden tener el mismo nombre.

Llaves primarias de Despachador: Elegimos id, por las mismas razones que elegimos id anteriormente.

Llaves primarias de Restaurante: id, si bien RUT nos parecía más natural que elegir un id, leímos en internet que la id es más segura en la mayoría de los casos y también que el RUT puede cambiar en algunos casos. Además, el id, según lo visto en clases, se usa prácticamente en el 99 % de las veces.

Llaves primarias de Administrador: Elegimos id, por las mismas razones que elegimos id anteriormente.

Llaves primarias de Platos: Elegimos nombre y Restaurante.id, muchos restaurantes pueden preparar platos con mismos nombres.

4.2.2. Respuesta a llaves parciales:

Llaves parciales: Según el libro oficial de la bibliografía del curso "Fundamentos de sistemas de bases de datos", con autor "Elmasri, Ramez.", en la página 68 se señala que "Un tipo de entidad débil normalmente tiene una clave parcial, que es el conjunto de atributos que pueden identificar sin lugar a dudas las entidades débiles que están relacionadas con la misma entidad propietaria." Así que, considerando esta definición para las llaves de las entidades débiles, tenemos lo siguiente;

Llaves parciales de Suscripcion: Cliente.id y EmpresaDespacho.id, ya que sirven para identificar una Suscripción de forma única, dado que nuestro planteamiento involucra necesariamente que un Cliente sólo puede tener un teléfono y email asociado, y ningún otro Cliente puede tener ese mismo email o teléfono.

Llaves parciales de ServicioDespacho: Elegimos Restaurante.id y nombre, ya que por definición de nuestro desarrollo, dos servicios despachadores de un restaurante no pueden tener el mismo nombre.

Llaves parciales de Platos: nombre, Restaurante.id, así se puede distinguir un plato de otro y además a qué Restaurante pertenece el plato mencionado.

Llaves parciales de Direccion: Cliente.id, calle, numero, region, comuna, departamento (todos los atributos), porque es la única forma de diferenciar una tupla de otra.

Llaves parciales de Pedido: Pedido.id, Cliente.id, Despachador.id, debe incluirse Cliente.id ya que la existencia de Pedido depende de la entidad fuerte Cliente mediante una FK que la referencia, y también se debe considerar Pedido.id porque de lo contrario, no podríamos diferenciar un pedido de otro que es realizado por un mismo cliente.

4.3. ¿Qué entidades tienen llaves compuestas? Ejemplifique para cada una de ellas la necesidad de tener una llave compuesta

- La llave compuesta de Platos está conformada de Platos.nombre y Restaurante.id, ambas son necesarias para identificar de forma única los platos, ya que un restaurante puede preparar varios platos y un mismo plato puede ser ofrecido por varios restaurantes, o sea por si solas no sirven para identificar a un plato. Esto también se debe a que los platos dependen de la existencia de un restaurante.

-La llave compuesta de ServicioDespacho consta de ServicioDespacho.nombre y Restaurante.id, recordar que, por enunciado, un servicio despachador sólo puede pertenecer a un sólo restaurante.

- La llave compuesta de Suscripción debe ser Cliente.id y EmpresaDespacho.id, con eso identificamos a que cliente pertenece dicha suscripción, como también a qué empresa está suscrito.

- También, la llave compuesta de Direccion consta de Cliente.id y todos sus atributos, dado que si bien todos los atributos juntos diferencian una tupla de otra, dos clientes pueden residir en el mismo domicilio a despachar el pedido, por lo que es necesario distinguirlos mediante Cliente.id y sus otros atributos.

Fuera de eso, ninguna otra entidad tiene llaves compuestas; las llaves son, o bien id's autogenerados que son PK, o bien es un id autogenerado que es PK pero también es a su vez FK relacionando a otra PK de otra tabla.

No creemos que fuera necesario usar llaves compuestas porque cada id identifica completamente a la tabla en cuestión de forma única, ya sea una entidad débil o fuerte.

4.4. Identifique la cardinalidad para cada una de las relaciones del modelo E/R, justificando su decisión.

Cliente-tiene-Suscripcion: (0,1) a (1,1), porque un cliente puede o no tener suscripción, pero una suscripción específica sólo puede ser de un cliente en particular, no puede ser de más de uno.

Suscripcion-registrada-EmpresaDespacho: (1,1) a (0,n), porque una suscripción puede sólo estar registrada en una empresa despachadora en particular, pero una de esas empresas puede tener múltiples suscripciones, o ninguna.

Cliente-ubicado_en-Direccion: (1,n) a (1,1), porque un cliente puede tener como mínimo una dirección al registrarse en el sistema, pero puede tener múltiples direcciones, mientras que una dirección en particular sólo puede pertenecer a un cliente (en nuestra propuesta del programa lo concebimos así, por motivos de seguridad).

EmpresaDespacho-contrata-Despachador: (1,n) a (1,1), porque una empresa puede contratar mínimo a un despachador para funcionar, pero puede tener hasta n, mientras que un despachador en particular sólo puede ser contratado por una empresa, ya que trabaja con horario laboral completo.

Despachador-despacha-Pedido: (1,n) a (1,1), porque un despachador puede despachar mínimo un pedido, pero puede despachar muchos más, mientras que un pedido sólo puede ser despachado por despachador.

Pedido-realizado-Cliente: (1,1) a (0,n), porque un pedido puede sólo existir si es realizado por un cliente, pero no puede ser realizado por más de uno, porque cada Pedido es único, así que por definición del sistema en nuestra propuesta, dos usuarios o más no pueden ser parte de realizar un mismo pedido en específico. Y es a (0,n) porque un cliente puede registrarse en el sistema y no realizar pedidos, o realizar n pedidos.

Pedido-se_compone-Platos: (1,n) a (1,n), porque un pedido puede componerse de varios platos diferentes de distintos restaurantes, y un plato puede ser parte componente de varios pedidos diferentes, ya que los platos se repiten y se crean varios iguales.

Administrador-administra-Restaurante: (1,n) a (1,n), porque un administrador puede administrar como mínimo un restaurante según enunciado, ya que no tendría sentido que no existieran restaurantes en el sistema, y puede administrar hasta n restaurantes. Por el otro lado, es a (1,n) porque un restaurante puede ser administrado como mínimo por un administrador, pero hasta un máximo de n, ya que todos los administradores deben poder administrar todos los restaurantes.

Restaurante-pertenece-ServicioDespacho: (1,1) a (1,1), porque cada restaurante debe tener un servicio despachador por enunciado, y por otro lado, es a (1,1) porque un ServicioDespacho puede pertenecer sólo a un mismo restaurante, según nuestro planteamiento del proyecto.

Administrador-administra-EmpresaDespacho: (1,n) a (1,n), porque el administrador puede administrar desde mínimo 1 hasta n empresas, mientras que una empresa mínimamente debe ser administrada por un administrador, pero todos los administradores deben poder administrar todas las n empresas, todo esto por enunciado del proyecto.

Restaurante-ofrece-Platos: (1,n) a (1,1), porque un restaurante puede ofrecer como mínimo un plato, pero no hay límite tope, por lo que puede ofrecer hasta n platos, y un plato puede ser ofrecido por un solo restaurante en específico, cada restaurante tiene sus propios platos y formas de prepararlos.

ServicioDespacho-asignado-Pedido: La relación es (0,n) a (1,n), porque a un servicio despachador se le pueden asignar cero o varios pedidos, y por el otro lado, un Pedido puede ser asignado a varios servicios despachadores, pero mínimo a un servicio despachador.

EmpresaDespacho-asignado-Pedido: La relación es (0,n) a (1,1), porque a una empresa despachadora se le pueden asignar cero o varios pedidos, y por el otro lado, un Pedido puede ser asignado a sólo una empresa despachadora, ya que por enunciado sólo la EmpresaDespacho puede despachar. Además, nuestra propuesta asume que el pedido completo será despachado por la EmpresaDespacho mediante su despachador.

ServicioDespacho-entrega_plato-EmpresaDespacho: (1,n) a (1,n), porque un servicio despachador puede hacer entrega de mínimo un plato, pero puede entregarle hasta n platos según el pedido. De igual forma, la relacion es a (1,n) porque a la empresa despachadora le pueden ser entregados 1 o más platos desde el servicio despachador propio de cada restaurante.

4.5. ¿Se uso jerarquía de clases en el modelo E/R? ¿Qué razones llevaron a que usara jerarquía de clases?

Pensamos en implementar un enfoque donde nos pudiera ayudar a organizar mejor el diagrama, pero no nos pareció que fuera necesario, porque las entidades definidas en nuestra propuesta son bastante diferentes entre sí, tanto en sus atributos como en el espacio relacional en el que están situadas,así que implementar jerarquía de clases no hubiera sido de gran ayuda.

4.6. Si se quisiera minimizar el tiempo de despacho total y para ello se levanta la restricción que indica que el pedido solo lo puede despachar una empresa ¿cambia en algo el modelo? En caso de ser afirmativo haga el nuevo modelo.

Sí, cada plato podría entregarse por diferentes empresas despachadoras, y aunque eso aumentaría considerablemente el costo del pedido, podría reducir el tiempo de despacho total, suponiendo que se hacen varias entregas del mismo pedido, según estén listos los platos.

Ahora, en caso de que se siga manteniendo nuestra propuesta del proyecto, en la que decidimos que toda empresa despachadora debe entregar el pedido sólo cuando estén todos los Platos en la empresa de despacho listos para ser despachados, también se podría reducir el tiempo de espera si es que los restaurantes, mediante sus ServicioDespacho entregan el pedido más rápido porque están geográficamente más cerca. Así, a algunas empresas despachadoras les llegarían los Platos antes que a otras, y podrían despachar antes. Entonces, el diagrama entidad-relación y el modelo relacional no cambiarían en nada, porque nuestras relaciones no capturan directamente que una empresa despachadora despacha un solo pedido, sino que se asume que en la relación "entrega_plato", diferentes platos le llegan a la EmpresaDespacho, pero platos que son del mismo pedido, así que solo esa EmpresaDespacho hace el despacho del pedido total mediante el despachador. Pero, si fueran muchas EmpresasDespacho que entregaran el pedido al Cliente, entonces las multiplicidades, las relaciones, las entidades y los atributos de estas no variarían a excepción de la multiplicidad de la relación "Pedido-despacha-Despachador", que pasaría de ser (1,1) a (1,n), a ser (1,n) a (1,n), porque ahora un despachador seguiría pudiendo entregar 1 o más pedidos, pero un pedido podría ser entregado por uno o más despachadores, ya que cada EmpresaDespacho tiene sus propios despachadores, así que eso permitiría que un pedido fuera repartido parcialmente por varias empresas. No hace falta hacer un nuevo diagrama para sólo agregar este cambio, esperamos que la explicación del cambio de la multiplicidad sea suficiente para satisfacer lo preguntado.

4.7. ¿El modelo resuelve los temas vistos en clase? como: Fidelidad, Redundancia, Anomalías, Simplicidad, Buena elección de llaves primarias.

Fidelidad: Hemos cumplido con la fidelidad al contexto del problema, ya que cada relación entre entidades ha sido pensada de tal forma que sea lo suficientemente necesaria y que también se distinga de todas las demás, para no ser redundantes al hacer dos relaciones similares en lugares cercanos del diagrama.

Redundancia: No hemos agregado atributos que sean redundantes a las entidades, todos los atributos tienen una razón de ser, ya sea para poder hacer consultas mediante el atributo RUT de, por ejemplo, la entidad Restaurante, o bien puede que no existan atributos en las entidades que puedan contener información ya presente en las relaciones.

Anomalías: No hay anomalías, se puede agregar, editar y eliminar las tablas (entidades) sin problemas, porque toda entidad que depende de otra se actualizará en cascada mediante el código SQL si es que la tabla de la que depende cambia. También, las relaciones están definidas de tal forma que no hay atributos conflictivos.

Simplicidad: Se han creado las entidades y atributos mínimamente indispensables para nuestro planteamiento del problema. Además, no se han realizado relaciones complejas innecesariamente. Puede ser que el RUT en algunas entidades parezca innecesario para el problema, pero nos parece poco realista no plantear una entidad comercial de este tipo sin el RUT, es común consultar por el RUT en vez del id.

Buena elección de llaves primarias: Elegimos las llaves primarias de las entidades fuertes mediante la id autogenerada, eso es seguro y evita que colisionen los id, ya que son autogenerados y no pueden ser null (más adelante agregaremos estas cláusulas al código SQL, usaremos el tipo de dato SERIAL en vez de INT). Ahora bien, en otras entidades débiles decidimos crear como PK un nuevo atributo que a su vez es FK referenciado a otra PK de otra tabla, ya sea directamente, o mediante más de una relación entre entidades débiles entre medio, pero que finalmente siempre depende de una entidad fuerte con una PK de tipo id autogenerado y no nulo en ningún caso, por lo que no hay problema en la elección de las llaves primarias según nuestra consideración.