

EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: MODELAR BASE DE DATOS RELACIONALES.
- EXERCISE 2: RELACIONANDO ENTIDADES.
- EXERCISE 3: MODELO ENTIDAD – RELACIÓN.
- EXERCISE 4: FORMAS NORMALES.
- EXERCISE 5: OPTIMIZACIÓN DEL MODELO.
- EXERCISE 6: ESTANDARIZACIÓN DE MODELO RELACIONAL BASADO EN REGLAS DE NORMALIZACIÓN.
- EXERCISE 7: MODELO RELACIONAL A PARTIR DE UN MODELO ENTIDAD - RELACIÓN.

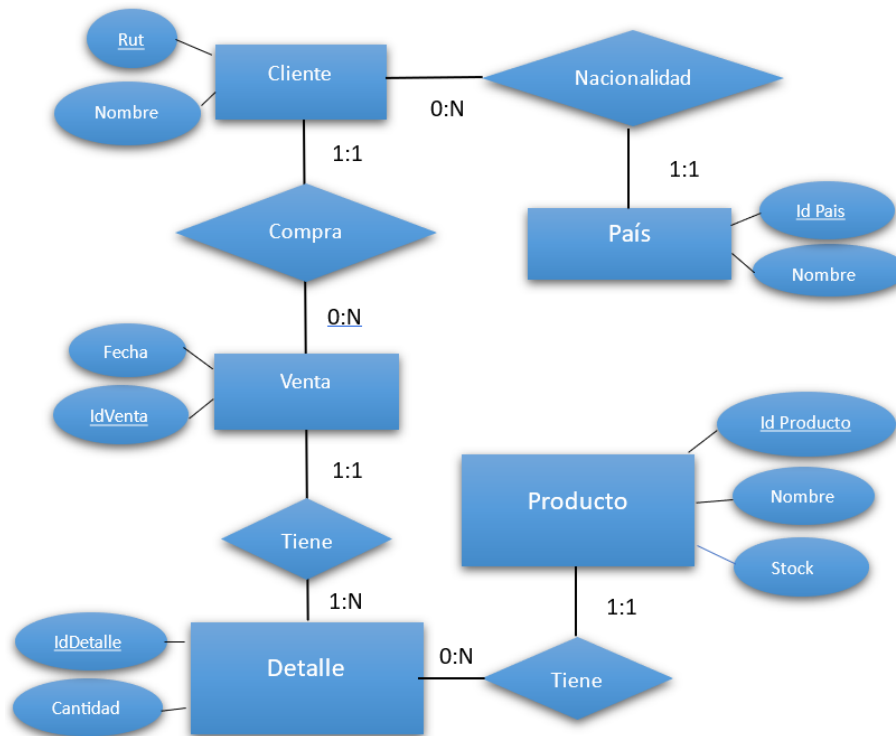
EXERCISE 1: MODELAR BASE DE DATOS RELACIONALES.

Para implementar una base de datos, se deben seguir estos pasos básicos:



CREAR UN DISEÑO CONCEPTUAL

Al diseño conceptual se le llama Modelo Entidad – Relación. Lo que busca es aclarar, de forma gráfica, las entidades involucradas y sus relaciones.

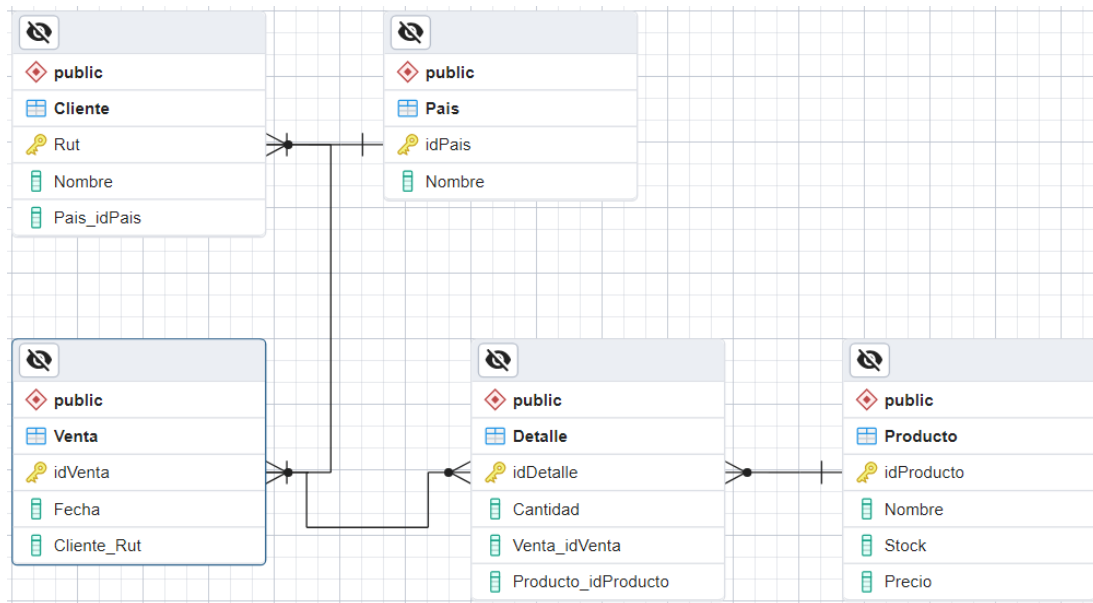


CREAR UN DISEÑO LÓGICO

Al modelo lógico se le llama **Modelo Relacional**, y es la traducción del modelo E-R.

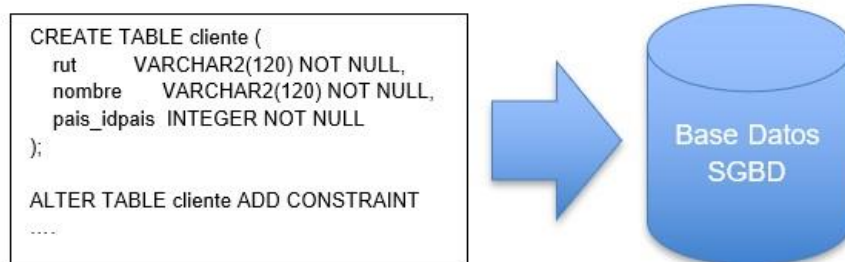
- Las entidades son implementadas como tablas.
- Se definen las características de cada atributo (tipo dato, permite nulo, ...).
- Se crean las llaves foráneas para representar las relaciones.

El modelo anterior quedaría como este implementado en pgAdmin:



IMPLEMENTACIÓN FÍSICA

A partir del Modelo Relacional, se generan los scripts para crear la base de datos en algún sistema gestor de base de datos.



EXERCISE 2: RELACIONANDO ENTIDADES.

La dependencia o asociación entre los conjuntos de entidades es llamada participación. Imaginemos que tenemos una entidad Cliente y sus Compras: los conjuntos de dichas entidades participan en el conjunto de relaciones cliente-compra, y están relacionadas o hay dependencias participativas, ya que un cliente puede realizar compras. Al definir una, o un conjunto, de relaciones entre estas dos entidades, podemos obtener información de participación, tal como: las compras realizadas por un cliente, compras mensuales de un cliente, los clientes con mayor cantidad de compras, entre otros.

RESTRICCIONES:

Son reglas que se deben cumplir sobre las entidades y relaciones.

CARDINALIDAD

Dado un conjunto de relaciones en el que participan dos o más entidades, la cardinalidad indica el número de registros con los que puede estar relacionado un registro.

- Uno a Uno (1, 1): un registro de una entidad X se relaciona con solo un registro en una entidad Y.
- Uno a Muchos (1, N): un registro en una entidad X se relaciona con cero o muchos registros en una entidad Y; los registros de Y solamente se relacionan con un registro en X.
- Muchos a Uno (N, 1): un registro de una entidad X se relaciona exclusivamente con un registro de la entidad Y. Pero, un registro de la entidad en Y se puede relacionar con 0 o muchos registros de la entidad en X.
- Muchos a Muchos (N, M): un registro de la entidad X se puede relacionar con 0 o muchos registros de la entidad en B, y viceversa.

PARTICIPACIÓN

Dado un conjunto de relaciones R, en el cual participa un conjunto de entidades A, la participación puede ser de dos tipos:

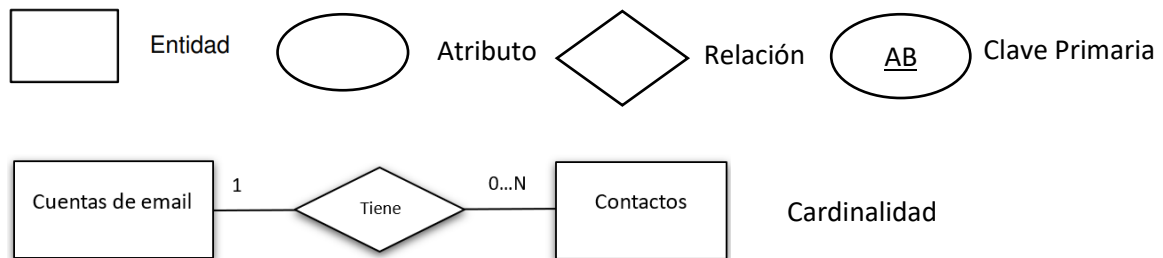
- *Total*: cuando cada entidad en A participa en, al menos, una relación de R.
- *Parcial*: cuando al menos una entidad en A, NO participa en alguna relación de R.

CLAVES

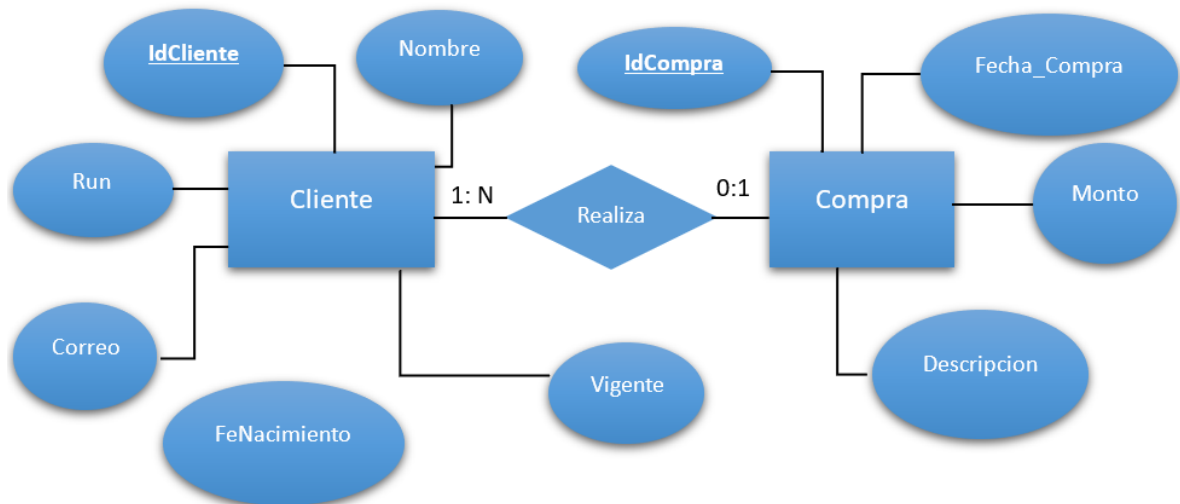
Es un subconjunto del conjunto de atributos comunes en una colección de entidades, la cual permite identificar inequívocamente cada una de las entidades pertenecientes a dicha colección. Asimismo, permiten distinguir entre sí, las relaciones de un conjunto de éstas.

- *Clave o Llave Primaria:* subconjunto de atributos de una entidad, que permite distinguir unívocamente un registro dentro de una entidad; los valores para éstos no se pueden repetir por cada entidad.

Elementos gráficos del modelo Entidad - Relación



Ejemplo de un Modelo ER:



EXERCISE 3: DISEÑO DEL MODELO ENTIDAD - RELACIÓN.

Como se mencionó anteriormente, su objetivo es aclarar de forma gráfica las entidades involucradas y sus relaciones, esto es un diseño conceptual.

Las entidades pueden ser de 2 tipos:

- Entidades Físicas: un auto, una casa, una persona, un empleado.
- Entidades Conceptuales: un trabajo, un curso, un préstamo.

Un conjunto de entidades, o tipo entidad, es un conjunto de aquellas que comparten las mismas propiedades, por ejemplo: conjunto de empleados, compañías, clientes, autos, etc.

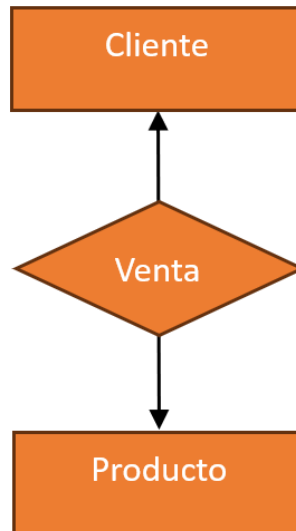
PASOS PARA CREAR UN MODELO ENTIDAD - RELACIÓN

1. Identificar las entidades.
2. Crear las relaciones.
3. Indicar cardinalidad de las relaciones.
4. Se definen los atributos de las entidades.
5. Se identifica los atributos claves de cada entidad.

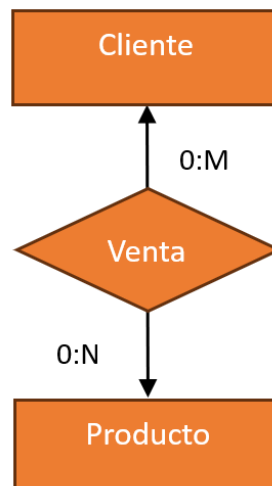
Vamos a desarrollar un ejemplo en el que necesitamos diseñar un modelo Entidad-Relación (ER) para registrar la venta de productos. Los requisitos incluyen tener un registro con el nombre de cada cliente, donde la venta debe contener información sobre la fecha y el monto total. Además, se desea identificar el país de residencia de cada cliente, y los detalles del producto que se vende deben incluir el IdProducto, Nombre, Stock y Precio.

1. Identificar las entidades:
 - Físicas: Cliente, Producto.
 - Conceptuales: Venta.

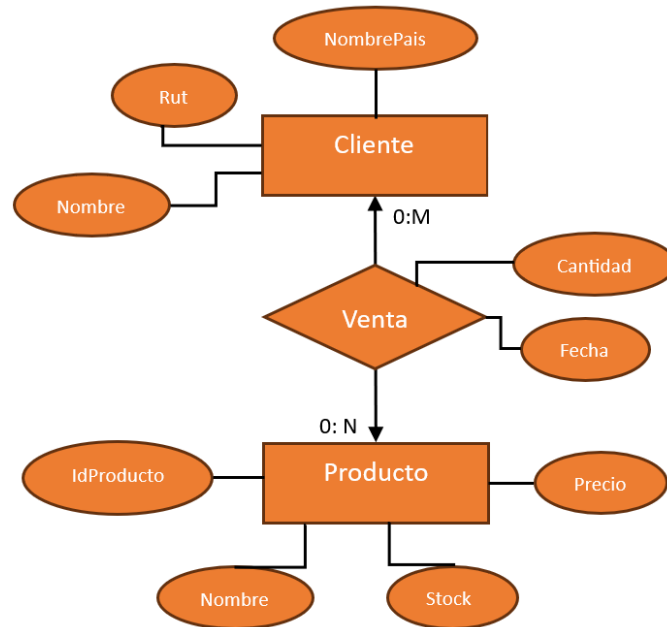
2. Crear las relaciones:



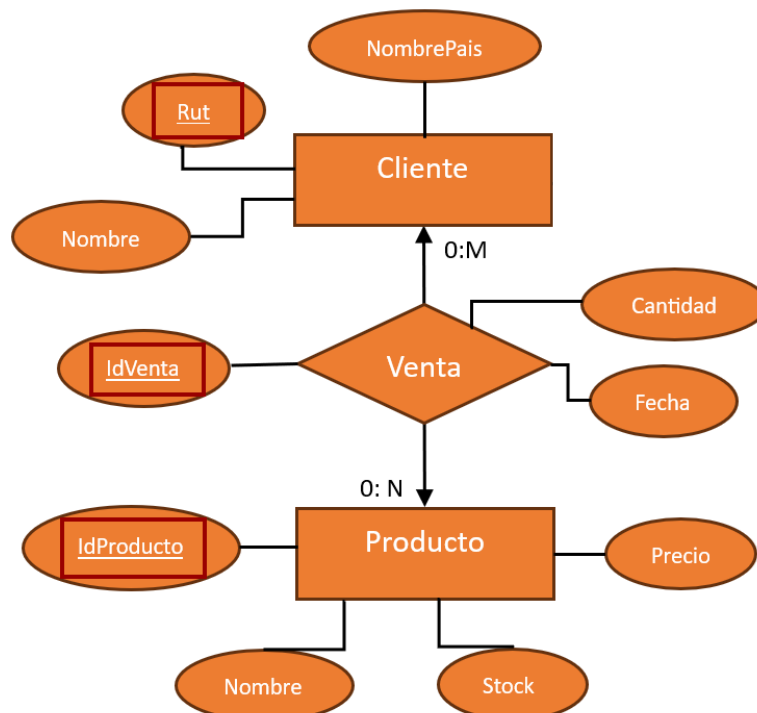
3. Indicar cardinalidad de las relaciones:



4. Se definen los atributos de las entidades:



5. Se identifica el atributo clave de cada entidad:



EXERCISE 4: FORMAS NORMALES.

La normalización de base de datos es una técnica de modelado, que consiste en designar y aplicar una serie de reglas a las relaciones obtenidas tras el paso del Modelo Entidad – Relación, al Relacional.

PRIMERA FORMA NORMAL (1FN)

Una tabla está en primera forma normal si sus atributos contienen valores atómicos, esto quiere decir que tienen que ser indivisibles.

En el siguiente ejemplo, podemos ver que no se cumple la 1FN para el atributo “Emails”, en los registros destacados.

EMPLEADO				
ID (PK)	NOMBRE	PUESTO	SALARIO	EMAILS
1	José Sánchez	Jefe de área	3000	juan@test.com; jefe1@test.com
2	Juan Pérez	Administrativo	1500	jsanchez@test.com
3	Ana Díaz	Administrativo	1500	adiaz@test.com; admin@test.com

Para solucionarlo, existen 2 opciones:

1. Duplicar registros con valores repetidos:
 - Se elimina el atributo “Email” que incumplía la condición.
 - Se incluye un nuevo atributo “Email” que sí sea indivisible. Por lo que se crea una nueva clave primaria con éste.
 - La nueva clave primaria será “ID, Email”.

Empleado				
IdEmpleado (Pk)	Email (Pk)	Nombre	Puesto	Salario
1	juan@test.com	Juan Pérez	Jefe de área	3000
1	jefe1@test.com	Juan Pérez	Jefe de área	3000
2	jsanchez@test.com	José Sánchez	Administrativo	1500
3	adiaz@test.com	Ana Díaz	Administrativo	1500
3	admin@test.com	Ana Díaz	Administrativo	1500

2. Separar atributo "Email" en otra tabla:

- Se crea una nueva tabla Empleados (b) que no contenga el atributo "Email".

Empleado			
IdEmpleado (Pk)	Nombre	Puesto	Salario
1	Juan Pérez	Jefe de área	3000
1	Juan Pérez	Jefe de área	3000
2	José Sánchez	Administrativo	1500

- Se crea una nueva tabla EMAILS con clave primaria ID-Email. Las tablas Emails y Empleados se relacionan por el campo ID.

Email	
Email (Pk)	IdEmpleado
juan@test.com	1
jefe1@test.com	1
jsanchez@test.com	2
adiaz@test.com	3
admin@test.com	3

Una tabla está en 2FN si:

- Está en 1FN.
- Todos los atributos que no son clave primaria tienen dependencia funcional completa, con respecto a todas las claves existentes en el esquema.
- Para recuperar un atributo no clave, se necesita acceder por la clave completa, no por una subclave.

La 2FN aplica a las relaciones con claves primarias, compuestas por dos o más atributos.

Empleado				
IdEmpleado (Pk)	Email (Pk)	Nombre	Puesto	Salario
1	juan@test.com	Juan Pérez	Jefe de área	3000
1	jefe1@test.com	Juan Pérez	Jefe de área	3000
2	jsanchez@test.com	José Sánchez	Administrativo	1500
3	adiaz@test.com	Ana Díaz	Administrativo	1500
3	admin@test.com	Ana Díaz	Administrativo	1500

En la tabla de Empleados (a), se pueden ver las dependencias de los atributos:

- ID → Nombre, puesto, salario.
- Puesto → Salario.

Observamos que los atributos, nombre, puesto y salario dependen únicamente del campo ID, por lo que no cumple la 2FN. Para solucionarlo, se debe:

- Actuar sobre los atributos con dependencias incompletas.
- Eliminar los atributos con dependencias incompletas.

Empleado			
IdEmpleado (Pk)	Nombre	Puesto	Salario
1	Juan Pérez	Jefe de área	3000
2	José Sánchez	Administrativo	1500

Crear nueva tabla con los atributos y la clave de la que depende.

Emails	
Email (Pk)	IdEmpleado
juan@test.com	1
jefe1@test.com	1
jsanchez@test.com	2
adiaz@test.com	3
admin@test.com	3

Se llega a la misma solución que con la 1FN.

TERCERA FORMA NORMAL (3FN)

Una tabla está en 3FN si:

- Está en 2FN.
- Todos los atributos que no son clave primaria no dependen transitivamente de esta.

Por lo tanto, hay que buscar dependencias funcionales entre atributos que no estén en la clave.

EMPLEADOS			
ID (Pk)	Nombre	Puesto	Salario
1	Juan Pérez	Jefe de área	3000
2	José Sánchez	Administrativo	1500
3	Ana Díaz	Administrativo	1500

Las dependencias son:

- ID → Nombre, Puesto.
- Puesto → Salario.

Observamos como la dependencia Puesto → Salario tiene dependencia transitiva con la clave primaria ID.

Para solucionarlo, se debe:

- Actuar sobre los atributos con dependencias transitivas.
- Separar en una tabla adicional los atributos que tienen dependencia transitiva con la clave (Salario), y establecer como PK el campo que define la transitividad (Puesto).

Puestos	
Puesto (Pk)	Salario
Jefe de área	3000
Administrativo	1500

Empleados		
ID (Pk)	Nombre	Puesto (FK)
1	Juan Pérez	Jefe de área
2	José Sánchez	Administrativo
3	Ana Díaz	Administrativo

Se añade el campo "Puesto" como Foreign Key, y ahora se encuentra en 3FN.

EXERCISE 5: OPTIMIZACIÓN DEL MODELO.

Cuando se obtiene un modelo normalizado, esto se refiere a que, desde el punto de vista de la teoría, se cuenta con un modelo optimizado en la redundancia de los datos, e integridad.

En la práctica, algunas veces se debe desnormalizar el modelo para darle solución a la velocidad de respuesta a determinadas consultas, mantener datos de forma histórica, o impedir que se modifiquen transacciones realizadas de forma indirecta.

Para ilustrar lo referente a esto, analicemos el siguiente ejemplo:

Una empresa X desea almacenar sus ventas diarias con detalle de lo vendido, desde el punto de vista de las formas normales. El modelo resumido quedaría de la siguiente forma:

VENTAS	
Folio (PK)	Fecha
1	10-10-2018
2	01-12-2019
3	15-08-2020

DETALLE_VENTAS			
ID (PK)	Folio (FK)	ID Artículo (FK)	Cantidad
1	1	10	1
2	1	20	10
3	1	30	2
4	2	40	1
5	2	50	1
6	3	30	1
7	3	20	1
8	3	50	1
9	3	40	1

ARTICULOS			
ID (PK)	Artículo	Stock	Valor Unitario
10	Alicates	20	1000
20	Tornillo 2"	100	300
30	Pilas	300	1200
40	Palanca	30	50000
50	Pala	80	15000

PROBLEMAS DEL MODELO NORMALIZADO

- Para obtener el total de una venta, debería relacionar las tres tablas, y multiplicar la cantidad vendida por el valor unitario del artículo.
- Si el valor unitario de un producto cambia, alteraría el valor de las ventas ya realizadas.
- Si el nombre de un artículo cambia, se modificaría el nombre de lo ya vendido.

Desde el punto de vista del diseño, se deben analizar todos estos escenarios posteriores a la normalización, con el objetivo de ver si el modelo normalizado da solución a todos los requerimientos del negocio.

Lo expuesto anteriormente, puede que sea un problema para los requerimientos de algún negocio de ventas, puesto que se realizan pocas ventas, y el costo de realizar las sumas no sea significativo; por otra parte, se generan reglas de negocio que impiden modificar o eliminar artículos ya vendidos, pero, lo que en cualquier caso sería un error desde el punto de vista de cualquier negocio de ventas, es que el valor unitario no se encuentre en el detalle, ya que todo artículo puede incrementar o disminuir su valor con el tiempo.

La solución será desnormalizar el modelo, para así solucionar este problema agregando el valor unitario del artículo a la tabla detalle venta.

DETALLE_VENTAS				
ID (PK)	Folio (FK)	ID Articulo (FK)	Cantidad	Unitario
1	1	10	1	1000
2	1	20	10	300
3	1	30	2	1200
4	2	40	1	50000
5	2	50	1	15000
6	3	30	1	1200
7	3	20	1	300
8	3	50	1	15000
9	3	40	1	50000

CALCULAR EL ORDEN DE LAS CONSULTAS.

Para el caso anterior, supongamos que es una empresa que realiza miles de ventas diarias, lo que implica que puede ser significativo y costoso obtener el total de una venta; para esto deberíamos calcular el orden de cada consulta, y así estimar como poder optimizarlas.

El orden se calcula obteniendo los datos promedio de cada tabla, y multiplicándolos a medida que se van relacionando. En el caso de las ventas, para obtener el orden de una consulta que calcule el total, vamos a suponer que:

- Las ventas promedio diarias son: 1.500.
- Cada venta tiene un promedio de artículos vendidos de: 15 ítems.
- El inventario promedio de artículos es de: 15.000.

Orden para obtener el siguiente listado diario de las ventas:

- Folio, Fecha, Total.
- $1.500 * 15 = 22.500$.

Esto quiere decir que se deben realizar 22.500 transacciones promedio, para obtener un listado de las ventas diarias, además, como el total de la venta se calcula Cantidad * Unitario, y el resultado se debe sumar por venta, podemos agregar que:

- Se realizan 22.500 multiplicaciones y 1.500 sumas.

Si los tiempos de respuesta no son los adecuados, podemos optimizar el modelo desnormalizándolo de la siguiente forma:

- Agregar el total de la venta a la tabla ventas.

VENTAS		
Folio (PK)	Fecha	Total
1	10-10-2018	6200
2	01-12-2019	65000
3	15-08-2020	66500

- Ahora, el orden para la misma consulta sería 1.500, y los cálculos aritméticos se reducen a 0.

Cuando se crean varias fuentes de datos que producen el mismo resultado, siempre se debe definir una que representa la verdad absoluta, con la finalidad de tener cómo solucionar posibles inconsistencias. Para el caso anterior, se podría definir Ventas Total como el valor verdadero si se producen diferencias al sumar el detalle de venta.

El problema principal de la desnormalización es que se deben agregar procesos que mantengan la consistencia de los datos; para el caso del ejemplo anterior, la suma del detalle debería cuadrar con el total para que los datos sean consistentes, y esto se traduce en mayores costos de desarrollo.

OTRO CASO

Supongamos un sistema para emitir licencias de conducir, si creamos un modelo normalizado, tendríamos algo como esto.

Entidades - Atributos:

- Persona (RUT, Nombre, FechaNacimiento, ...)
- Licencia (Folio, FechaEmision, Clase, ...)

El problema de este modelo es que, si cambia algún dato de la entidad Persona, por ejemplo: el nombre, alteraría las licencias emitidas con anterioridad, y esto significa que si requiere reimprimir una licencia que se obtuvo previo al cambio de nombre, ésta saldría con un nombre distinto al que fue emitida, lo que no puede ocurrir.

Para solucionarlo, se desnormaliza el modelo colocando datos redundantes con la finalidad que se mantenga el histórico.

- Persona (RUT, Nombre, FechaNacimiento, ...)
- Licencia (Folio, NombreConductor, FechaNacimiento, FechaEmision, Clase, ...)

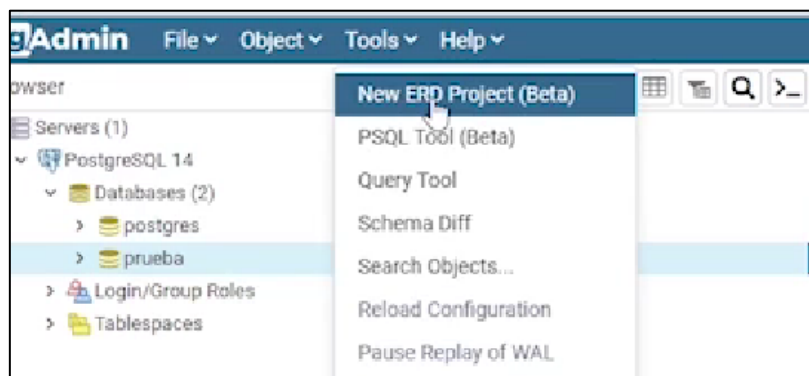
EXERCISE 6: ESTANDARIZACIÓN DEL MODELO RELACIONAL BASADO EN REGLAS DE NORMALIZACIÓN.

Para continuar con las ejemplificaciones, trabajaremos basándonos en un ejercicio propuesto:

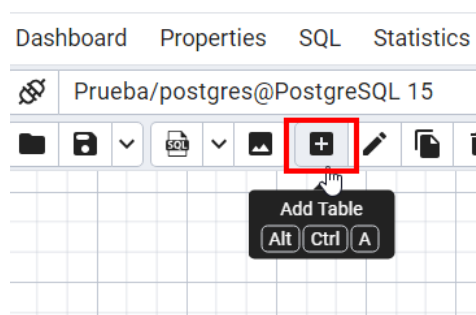
“Una empresa de transportes quiere gestionar envíos de pedido a clientes. Podemos plantearnos un diseño inicial de una tabla, la cual contendrá toda la información, código del envío, camión de transporte, datos del cliente y cada artículo del envío, incluyendo sus características. Un envío puede despachar varios pedidos, y cada pedido puede incluir varios artículos”.

Para realizar la ejemplificación, crearemos un Modelo Relacional normalizado.

Iniciamos el proceso de representación gráfica de nuestras tablas, las cuales corresponden a las dos principales entidades identificadas en el enunciado: "Envío" y "Pedido". Para plasmarlas visualmente, comenzaremos creando una base de datos. Una vez creada, nos dirigiremos a la sección "Tools" y luego seleccionaremos la opción "New ERD Project". Esta funcionalidad nos facilitará la creación de las tablas necesarias.



A continuación, haremos clic en “Add Table”



Se abrirá una ventana en la que podremos agregar los atributos. Si observamos la imagen a continuación, veremos cuatro pestañas que indican las acciones que podemos realizar. Seleccionaremos la pestaña "Columns".

Table: Envio (public)

General **Columns** Advanced Constraints

Name Envio

Schema public

Comment

Al hacer clic en "Columns", se abrirá una nueva interfaz. Al hacer clic en el icono "+" podremos añadir un nuevo atributo para la tabla.

Table: Envio (public) ×

General **Columns** Advanced Constraints

Columns +

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
⋮		Select an item...			<input type="checkbox"/>	<input type="checkbox"/>	

El signo "Delete row", sirve para eliminar un atributo.

Table: Envio (public)

General **Columns** Advanced Constraints

Columns

	Name	Data type
⋮		Select an item...

Delete row

Mientras que los "puntos" se utilizan para reorganizar los atributos según el orden deseado para su visualización.

Table: Envio (public)

General Columns Advanced

Columns

		Name	
⋮	✎	id	
⋮	✎	prueba	

A continuación, ingresaremos el nombre del atributo; en este caso, será "CodigoEnvio". El tipo de dato predeterminado es "char", y podemos elegir si este atributo será la "Primary key" o si será un campo obligatorio ("Not NULL"). Dado que se trata de la clave primaria, seleccionaremos "Primary key".

Table: Envio (public)

General Columns Advanced Constraints







Columns

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
⋮	✎	CodigoEnvio	char			<input type="checkbox"/>	<input checked="" type="checkbox"/>

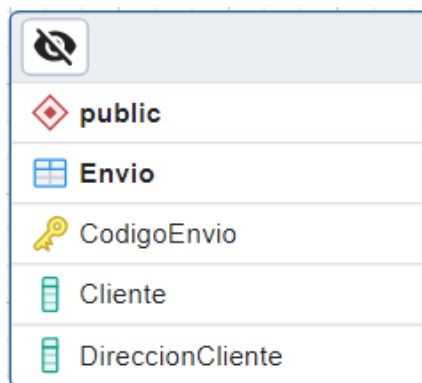
Inmediatamente, añadimos los atributos "Cliente" y "DireccionCliente", indicándolos como campos obligatorios. Finalmente, seleccionamos "Guardar" para completar el ingreso de los atributos.

Table: Envio (public)

General Columns Advanced Constraints

Columns							
		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
		CodigoEnvio	"char" v			<input type="checkbox"/>	<input checked="" type="checkbox"/>
		Cliente	"char" v			<input checked="" type="checkbox"/>	<input type="checkbox"/>
		DireccionCliente	"char" v			<input checked="" type="checkbox"/>	<input type="checkbox"/>

Con esto, podemos observar nuestra primera tabla junto con sus atributos asociados.



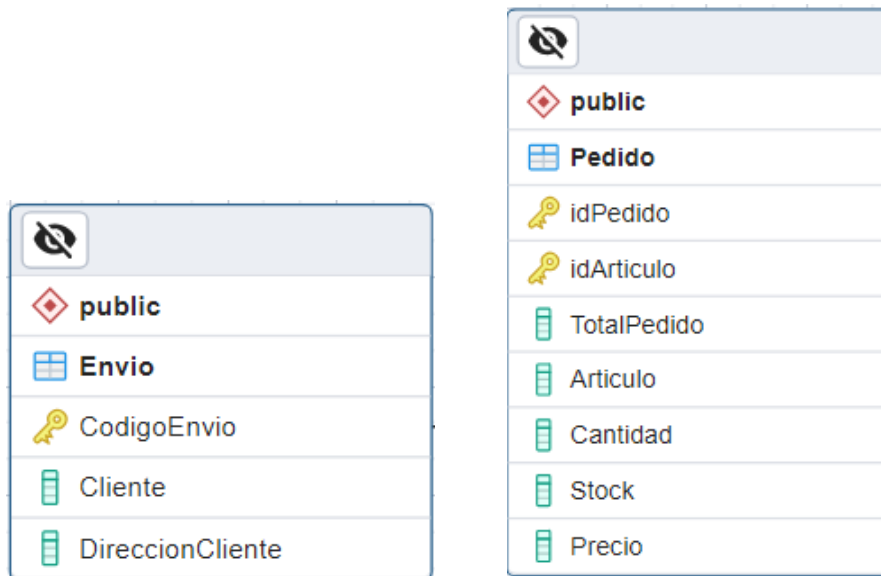
Continuaremos con la tabla "Pedido", que incluye los atributos: "idArticulo", "TotalPedido", "Articulo", "Cantidad", "Stock", "Precio" y "IdPedido". En la tabla "Pedido", se observa la formación de una clave compuesta, ya que los atributos "idPedido" y "idArticulo" serán necesarios para establecer nuestra "primary key". Aunque la mayoría de los atributos dependen de esta clave, es importante señalar que los atributos "Stock" y "Precio" dependen de "IdArticulo".

General Columns Advanced Constraints							
		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
		idArticulo	"char" v				
		idPedido	"char" v				
		Articulo	"char" v				
		Cantidad	"char" v				
		Stock	"char" v				
		Precio	"char" v				

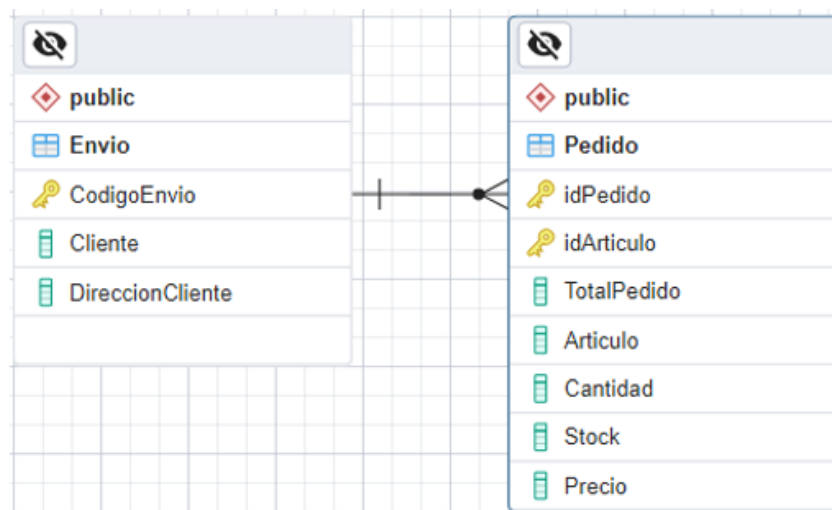
public
Pedido
idPedido
idArticulo
TotalPedido
Articulo
Cantidad
Stock
Precio

Como buena práctica, dejaremos las llaves como el primer atributo de la tabla.

Ahora, nuestras dos tablas se presentan de la siguiente manera:



A continuación, procederemos a establecer las relaciones entre las tablas. En este contexto, un "Envío" puede contener una cantidad ilimitada de "Pedidos".



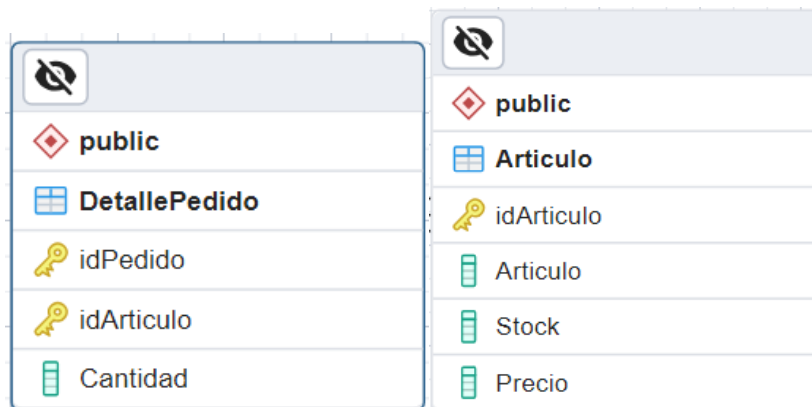
Hasta este punto, nuestro modelo ha alcanzado la primera forma normal.

Continuaremos evaluando si estas tablas cumplen con la segunda forma normal. La tabla "Envío" satisface la segunda forma normal, ya que cada atributo depende de su respectiva llave primaria "CodigoEnvío", y no hay dependencias parciales.

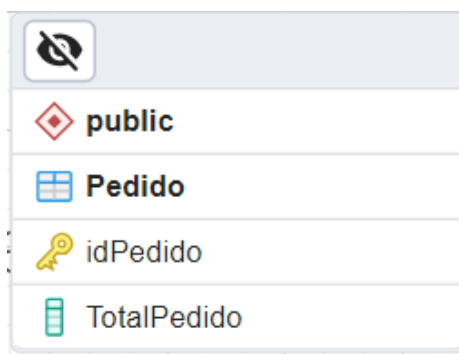
En cuanto a la tabla "Pedido", se identifica una dependencia parcial del atributo "Artículo" respecto a la clave primaria compuesta "idPedido" e "idArtículo".

Para resolver esta situación, incorporaremos dos tablas en nuestro modelo. La primera se denominará "DetallePedido", que contendrá la llave compuesta con "idPedido" e "idArtículo". Su único atributo será "Cantidad", ya que este depende de la llave compuesta. Ahora, la llave compuesta determina completamente todos los atributos sin dependencias parciales, cumpliendo con la segunda forma normal.

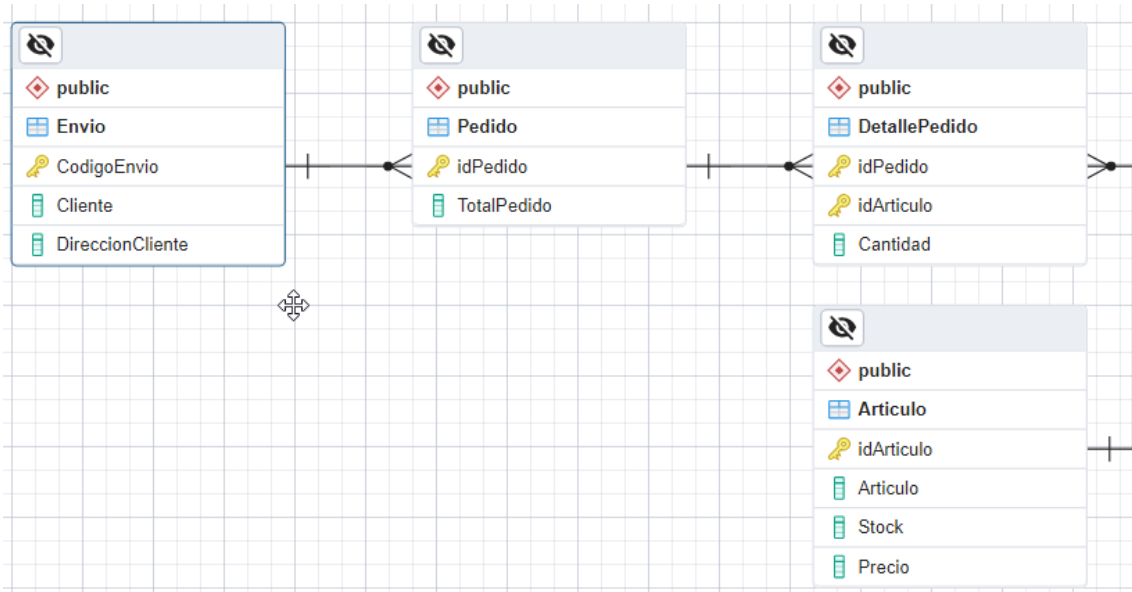
La segunda tabla será "Artículo", que tendrá el atributo "IdArtículo" como llave primaria, y los atributos "Artículo", "Stock" y "Precio".



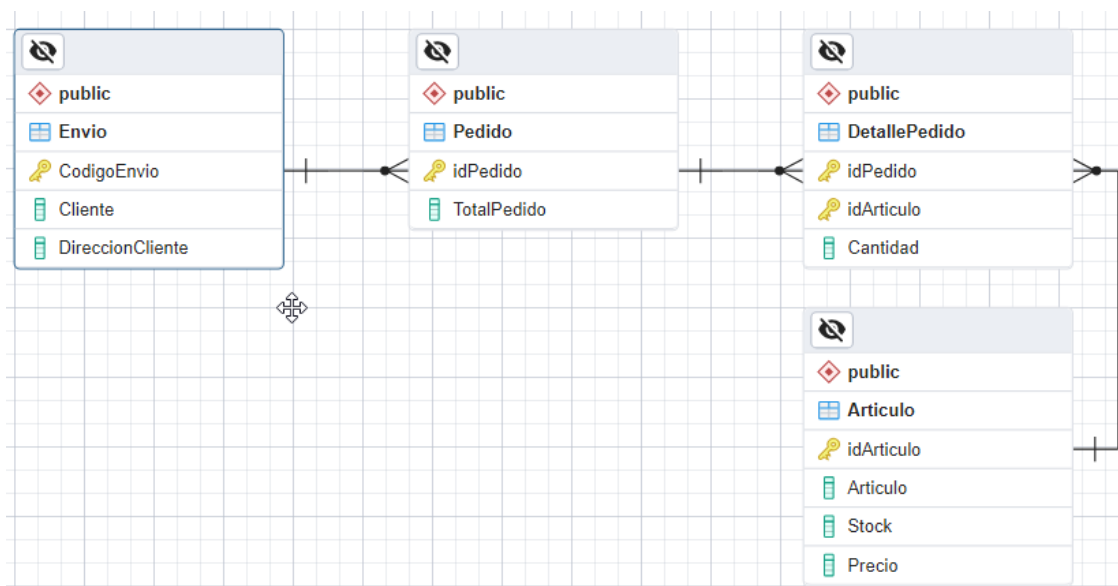
Finalmente, la tabla "Pedido" cuenta con la llave primaria "idPedido", determinando completamente a todos los atributos sin dependencias parciales, cumpliendo con la 2NF. La estructura final es la siguiente:



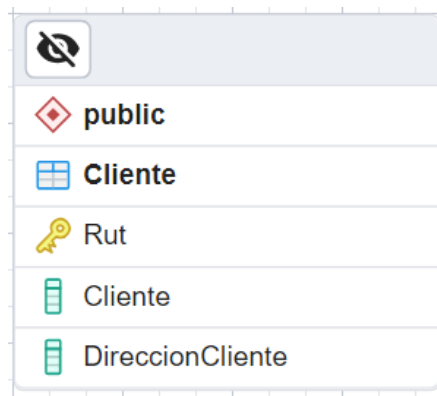
Ahora, procederemos a establecer las relaciones con las dos tablas nuevas. Un pedido puede tener más de un detalle de pedido, y un artículo puede estar en muchos detalles de pedidos. De esta manera, observamos que nuestro modelado cumple de forma completa con la segunda forma normal.



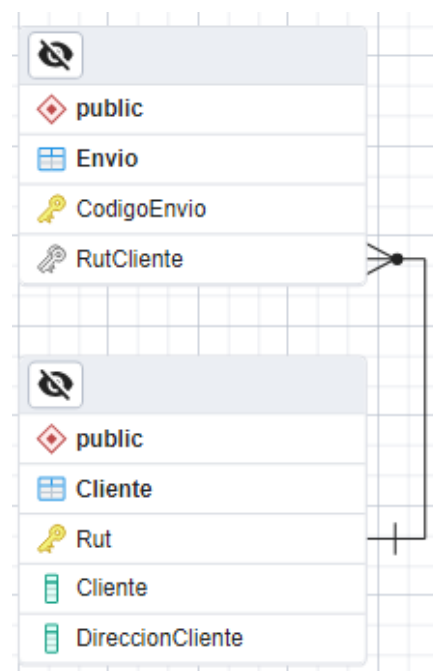
Continuaremos analizando si nuestro modelo cumple con la tercera forma normal:



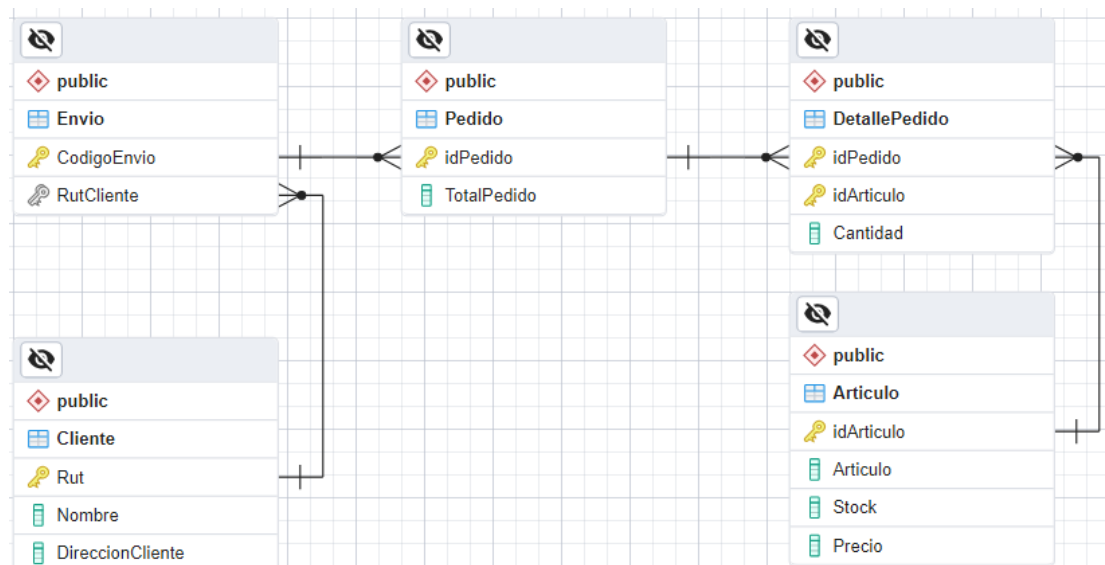
En esta ocasión, al revisar la tabla "Envío", identificamos que "DireccionCliente" depende funcionalmente de "Cliente". Por lo tanto, "DireccionCliente" está transitivamente dependiente de la clave primaria "CodigoEnvio". Debido a este detalle, la tabla no cumple con la tercera forma normal, lo que nos obliga a crear una nueva tabla llamada "Cliente" con los atributos "Rut" como llave primaria, "Cliente" y "DireccionCliente".



Eliminaremos de la tabla "Envío" los atributos correspondientes al cliente y posteriormente, graficamos la relación en donde estableceremos que un cliente puede tener más de un Envío.



Con estos ajustes, hemos corregido la dependencia funcional identificada en la tabla "Envío", llevándola a la Tercera Forma Normal (3NF). Finalmente, nuestro modelo quedará de la siguiente forma:

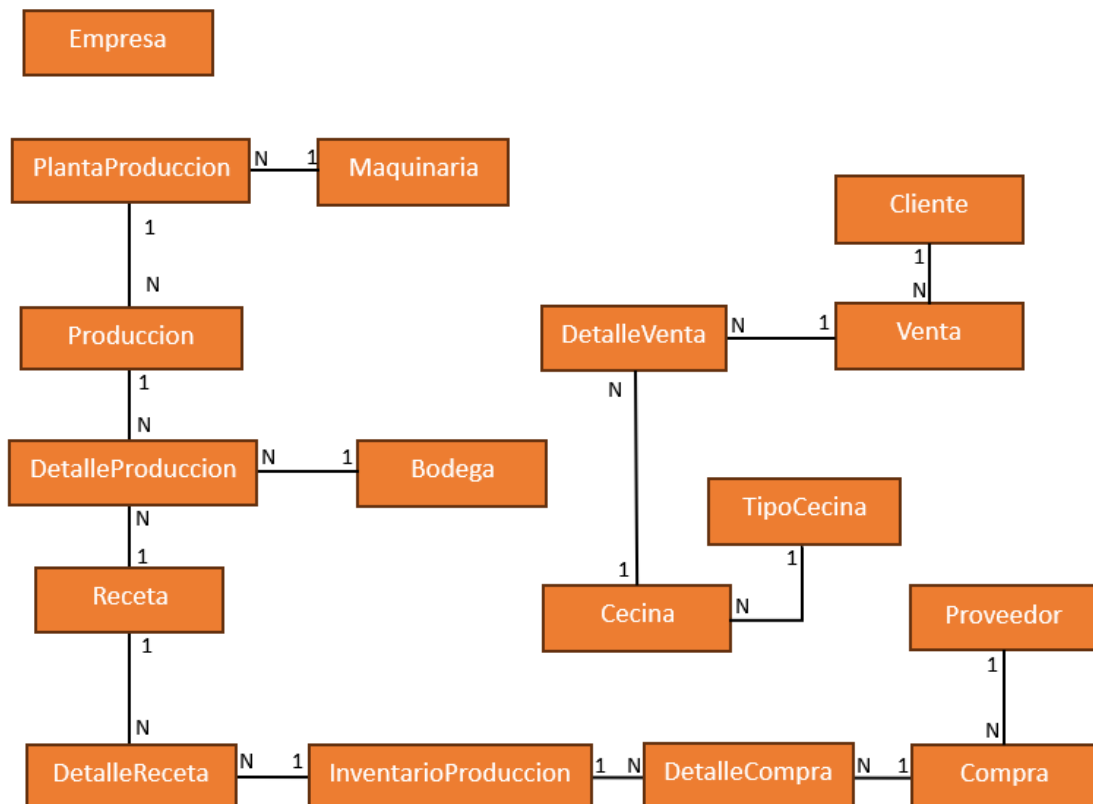


EXERCISE 7: MODELO RELACIONAL A PARTIR DE UN MODELO ENTIDAD - RELACIÓN.

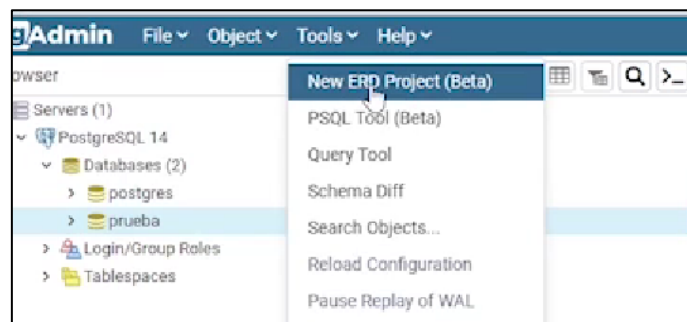
ENUNCIADO DEL EJERCICIO:

Una empresa de cecinas de cerdo desea poder mantener un registro de la producción de sus cecinas y materias primas consumidas. Necesitan tener control de toda su línea de producción, compra de materias primas e insumos a proveedores, planta productiva, y máquinas donde se produce cada partida y su clasificación en lotes. En su producción de cecinas, se requiere conocer la merma producida por kilo de cerdo según el tipo de éstas. Las partidas son almacenadas en bodegas, y es necesario ver cuáles deben ser despachadas primero desde la bodega, según su fecha de fabricación. Se debe tener registrado, además, a quien le fue vendido cada lote.

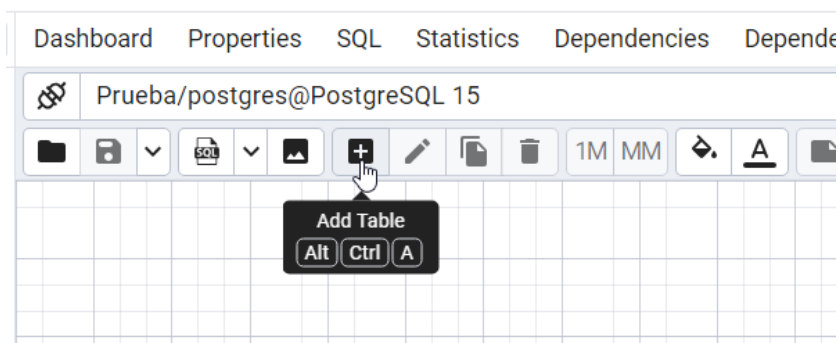
El modelo Entidad – Relación es el siguiente:



Para realizar la demostración práctica y dar inicio a nuestro proceso de modelado, utilizaremos la aplicación **pgAdmin** de PostgreSQL. Al iniciar la aplicación, se nos pedirá la contraseña previamente registrada. Luego, tendremos la opción de acceder a las bases de datos disponibles, entre las cuales destacan "postgres" y "prueba". En esta instancia, optaremos por la base de datos denominada "prueba". A continuación, nos dirigiremos a la pestaña "Tools" y seleccionaremos la opción "New ERD Project", donde podremos llevar a cabo la creación de nuestro modelo relacional.



Vamos a construir un modelo relacional sin atributos, donde las tablas fundamentales son: Empresa, Cecinas, Producción y Materia Prima. Para empezar, accederemos a la opción "Add Table" o mediante el atajo de teclado "Alt + Ctrl + A".



En primer lugar, crearemos la tabla denominada "Empresa", con visibilidad pública, y confirmaremos la acción haciendo clic en "OK".

New table ×

GeneralColumnsAdvancedConstraints

Name

Empresa

Schema

public

×

▼

Comment

×

Close

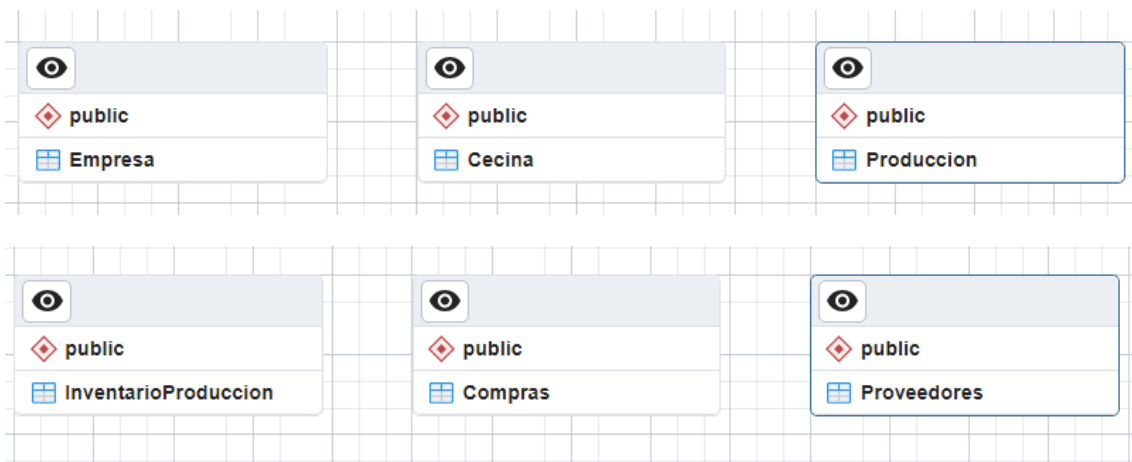
↺

Reset

Save

Seguiremos el mismo procedimiento para crear las tablas "Cecina", "Producción" e "InventarioProducción" (la cual representa la materia prima en la empresa).

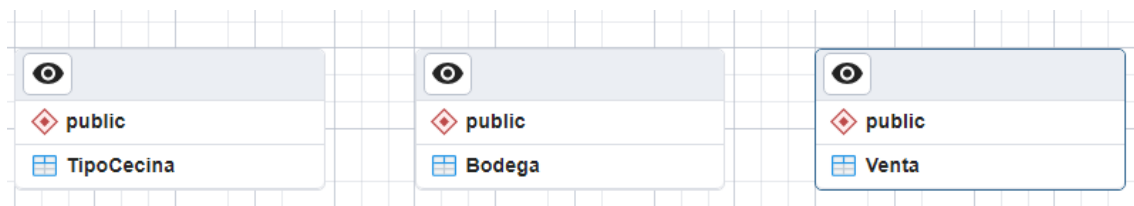
Las próximas tablas que incluiremos en nuestro diagrama son "Compras" y "Proveedores". Las tablas "Materias Primas" e "Insumos" serán integradas en la tabla "InventarioProducción", quedando configurado de la siguiente manera:



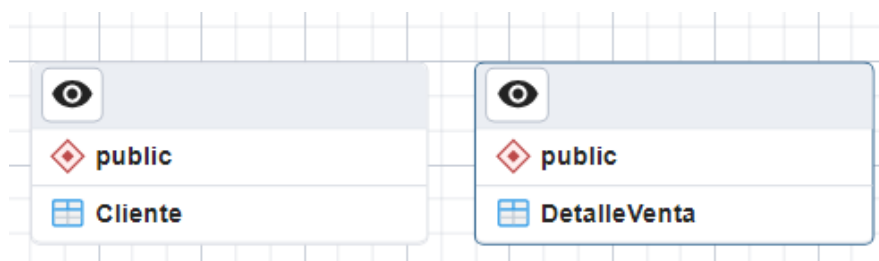
Las tablas que se identifican en el enunciado son la "Planta Productiva" y la "Maquinaria"; en este contexto, consideraremos el "Lote" como un atributo en lugar de una tabla.



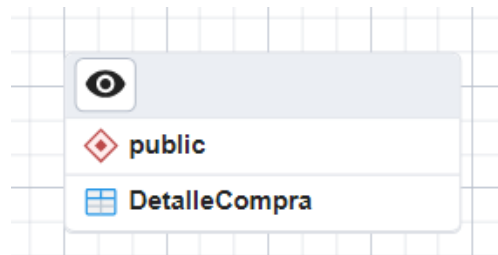
Al continuar examinando el enunciado, observamos que aún existen tablas adicionales que podemos crear, tales como "Tipo de Cecina", "Bodega" y "Venta".



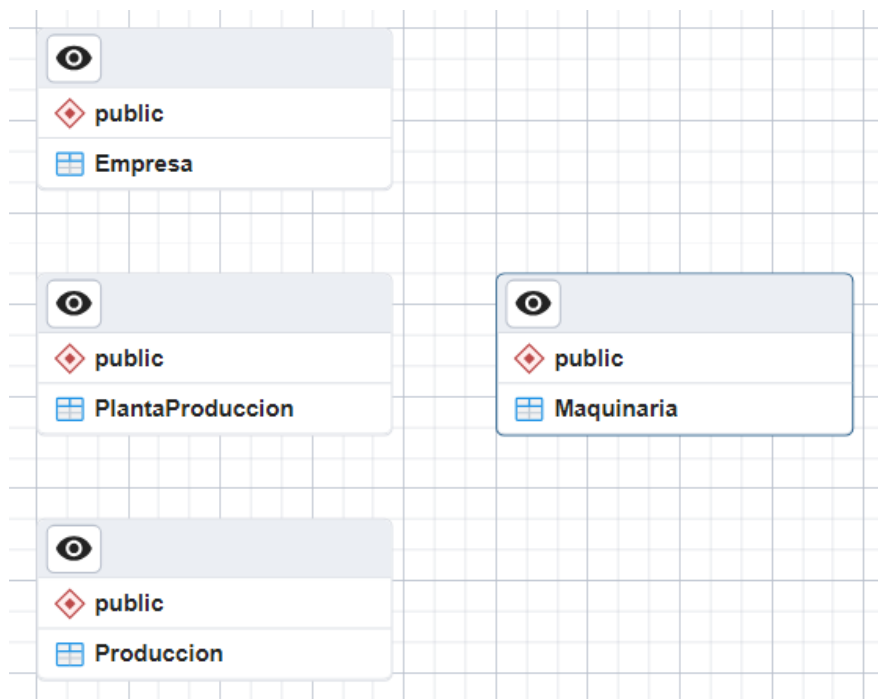
Ahora que hemos creado todas estas tablas, el siguiente paso es establecer relaciones entre ellas. Por ejemplo, en el caso de la tabla "Venta", esta será solicitada por un cliente. Por lo tanto, debemos crear la tabla "Cliente" y posteriormente establecer la relación entre el cliente y la venta, creando así la tabla "Detalle Venta".



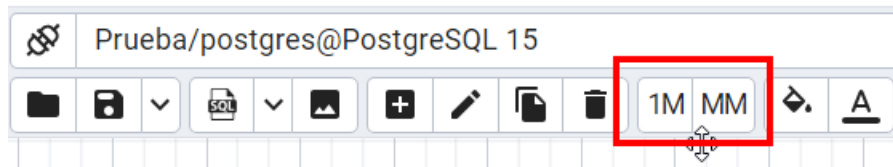
Además, incorporaremos una tabla implícita llamada "DetalleCompra", teniendo en cuenta la actividad de compra.



Ahora nos centraremos en establecer las relaciones entre las tablas de nuestro diagrama. Comenzaremos con la tabla "Empresa", que actuará como la tabla principal de este diagrama. Directamente debajo de esta tabla, estableceremos la conexión con la tabla "PlantaProduccion", la cual está relacionada tanto con la "Maquinaria" como con la "Producción". La estructura resultante será la siguiente:



Accederemos al menú superior, donde encontraremos los símbolos "1M" y "MM", que indican relaciones de uno a muchos o muchos a muchos, respectivamente.

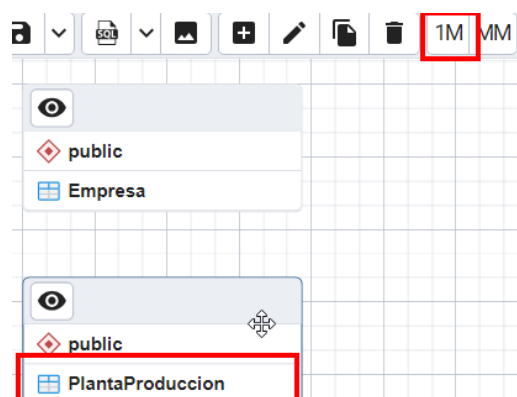


Para establecer las relaciones, comenzaremos por añadir atributos a las tablas que utilizaremos. Iniciaremos haciendo doble clic en la tabla "PlantaProducción", nos dirigiremos a la pestaña "Columns" y agregaremos la primera columna llamada "id". Repetiremos este proceso para la tabla "Maquinaria".

Table: PlantaProduccion (public)





General				Columns	Advanced	Constraints
Columns						
		Name	Data type	Len		
		id	"char"			

Posteriormente, seleccionaremos la tabla "PlantaProduccion" y activaremos la relación uno a muchos desde el menú superior. En este punto, estableceremos la relación con la tabla "Maquinaria" utilizando el atributo "id".



One to many relation ✕

General



Local Table	<div> (public) PlantaProduccion</div>
Local Column	<div> id</div>
Referenced Table	<div> (public) Maquinaria</div>
Referenced Column	<div> id</div>

✕ Close ↺ Reset 💾 Save

Ahora haremos la relación de uno a muchos entre "Produccion" a "PlantaProduccion".

One to many relation





General

Local Table	<div> (public) Produccion</div>
Local Column	<div> id</div>
Referenced Table	<div> (public) PlantaProduccion</div>
Referenced Column	<div> id</div>

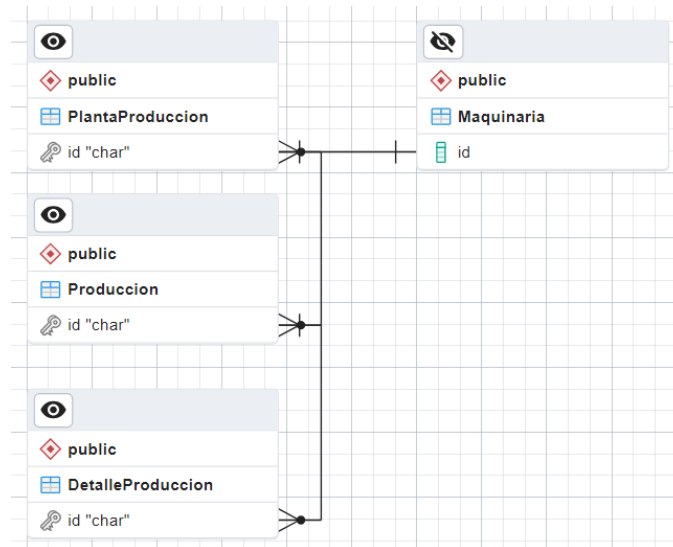
Dado que cada producción requerirá un detalle de productos y no habíamos incluido esta tabla en el diagrama principal, crearemos la tabla llamada "DetalleProduccion" con la columna predeterminada "id". Posteriormente, estableceremos la relación con la tabla "Produccion".

One to many relation

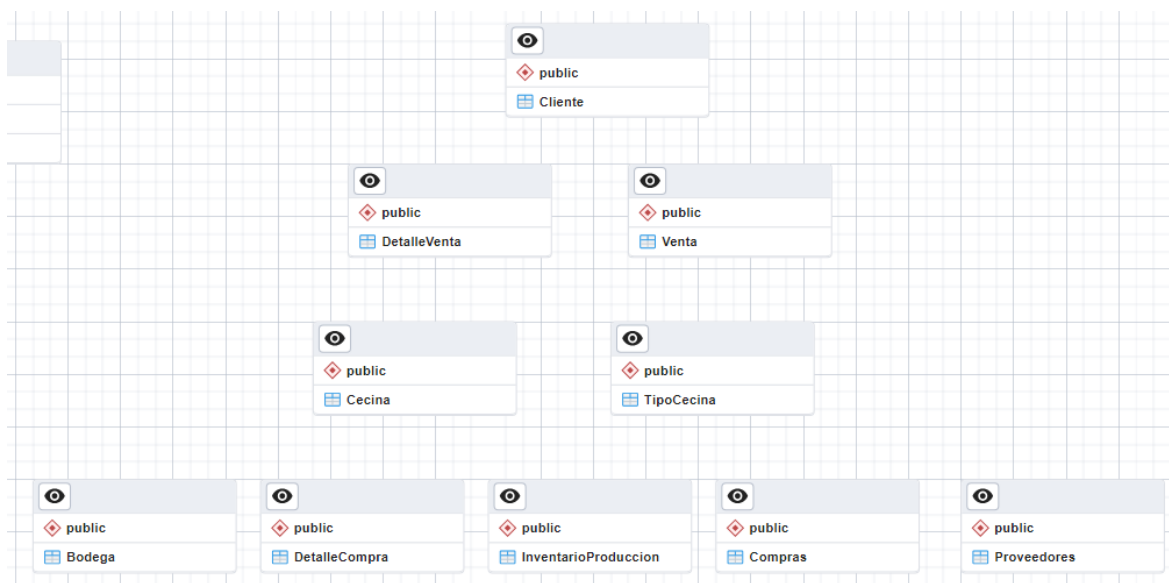
General

Local Table	<div> (public) DetalleProduccion</div>
Local Column	<div> id</div>
Referenced Table	<div> (public) Produccion</div>
Referenced Column	<div> id</div>

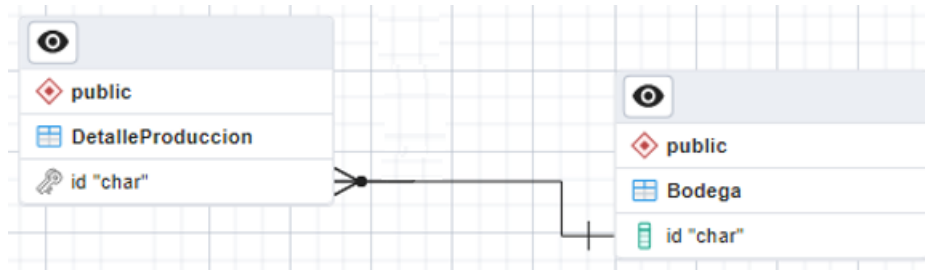
Hasta el momento nuestro diagrama va quedando de la siguiente manera:



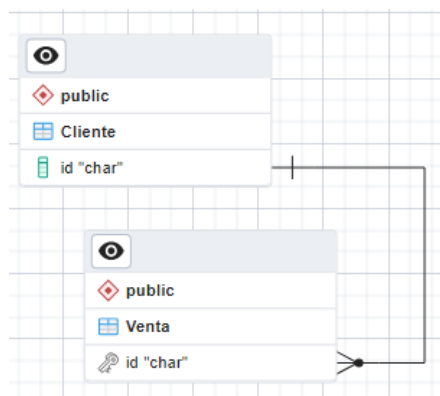
Hasta ahora, hemos observado la existencia de tablas que están contenidas por otras. Por ejemplo, las ventas están vinculadas a clientes, y un cliente puede tener múltiples compras, que, desde la perspectiva de una empresa, se traducen en ventas. Estas ventas pueden estar asociadas a muchos productos, a su vez relacionados con detalles de ventas, y múltiples cecinas pueden estar vinculadas a un tipo de cecina. Por lo tanto, procederemos a reorganizar nuestro diagrama de la siguiente manera:



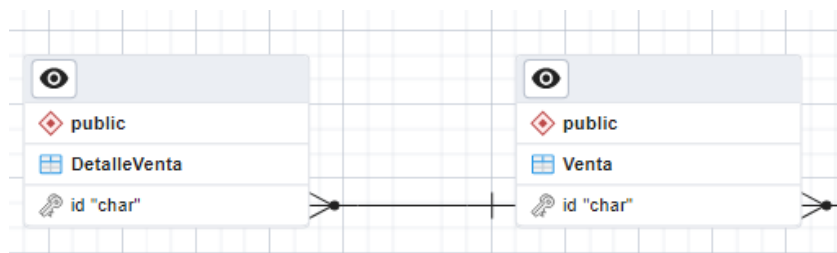
Comenzaremos estableciendo las relaciones correspondientes. Una bodega puede contener múltiples detalles de producción, por lo que procederemos a crear la relación utilizando el atributo por defecto de cada tabla.



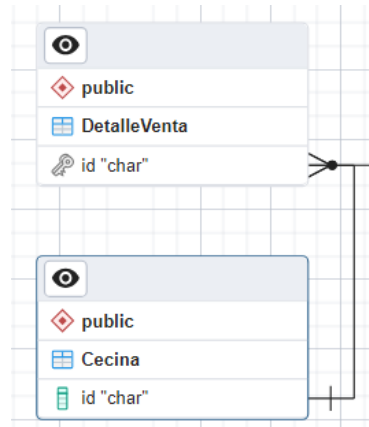
En cuanto al cliente, sabemos que puede generar varias ventas. Para reflejar esta relación, estableceremos una relación de uno a muchos entre estas dos tablas.



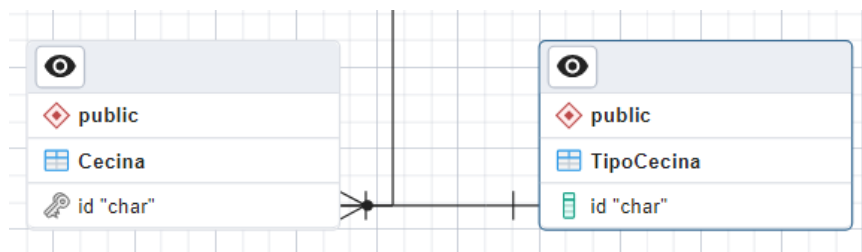
Además, considerando que una venta puede tener varios detalles de venta, continuaremos trabajando en la relación entre estas dos tablas.



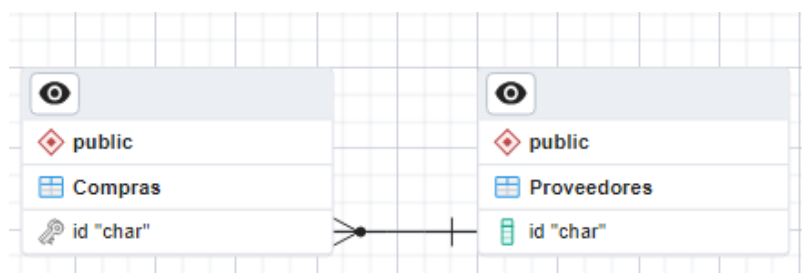
Además, una cecina puede tener varios detalles de venta, por lo que avanzaremos en esta relación.



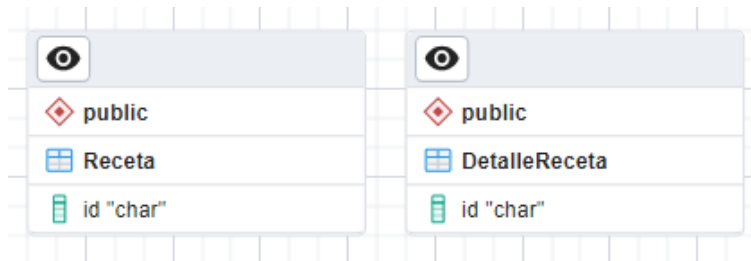
Asimismo, múltiples cecinas pueden estar agrupadas bajo un mismo tipo de cecina. La relación entre estas dos tablas se vería de la siguiente manera:



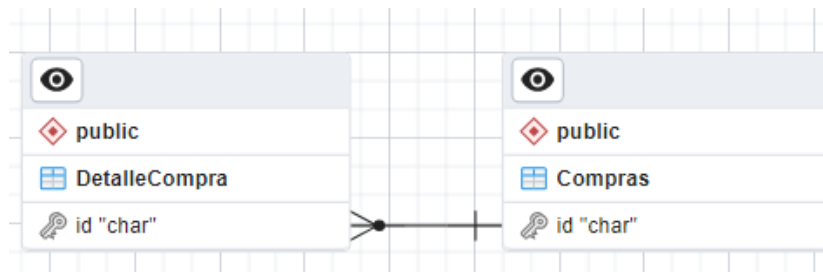
Con estas conexiones, hemos vinculado al cliente con las ventas y hemos establecido la relación entre la planta de producción y su maquinaria. Continuaremos vinculando a los proveedores con las compras, donde un proveedor puede tener varias transacciones de compra.



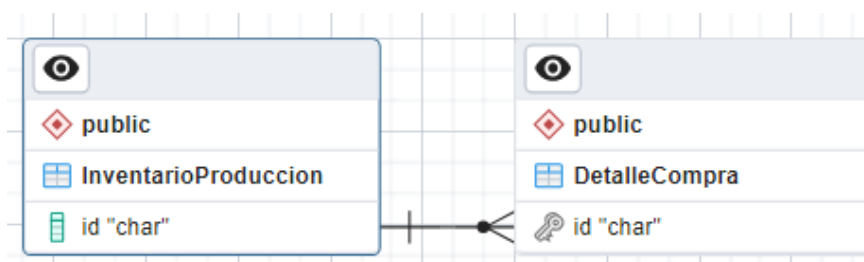
Aún debemos establecer la relación entre las tablas "InventarioProduccion" y "DetalleCompra". Cada vez que se realiza una compra, ambas tablas tendrán un detalle específico de la transacción. Este detalle se enviará al inventario de producción, y se registrarán todos los ingredientes en una nueva tabla denominada "DetalleReceta", que a su vez estará asociada con otra tabla llamada "Receta".



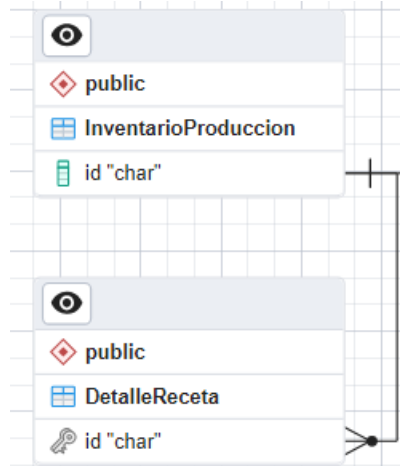
Estas nuevas tablas serán fundamentales para conectar a los proveedores con la producción, por lo que crearemos nuevas relaciones. Primero, estableceremos la relación entre las tablas "Compra" y "DetalleCompra", ya que una compra puede tener varios detalles de compra.



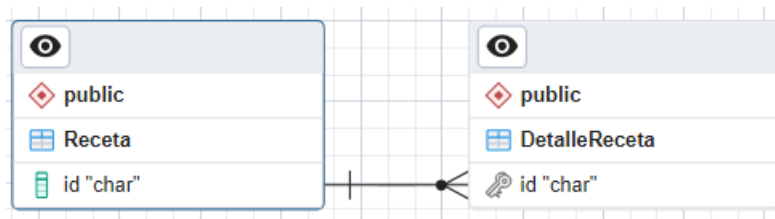
Continuaremos con el inventario de producción, que contendrá múltiples detalles de compras asociados.



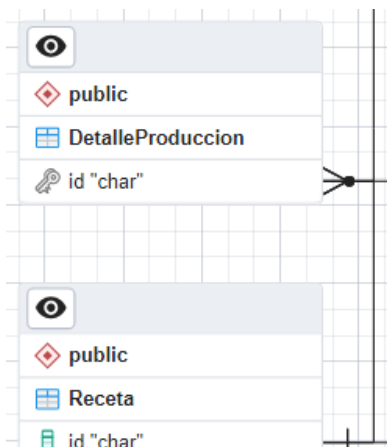
Además, es importante tener en cuenta que el inventario de producción puede contener varios detalles de receta.



Continuamos con la tabla "Receta". Una receta puede estar presente en muchos detalles de recetas, por lo que estableceremos esta relación.

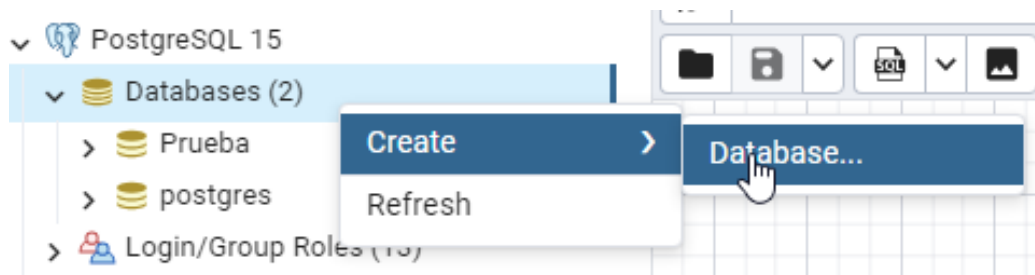


Finalmente, concluimos con la conexión entre "Receta" y "DetalleProduccion". De esta manera, una receta puede estar asociada a varios detalles de producción.




De esta forma, hemos representado visualmente las relaciones internas y externas entre las tablas de grandes agrupaciones, como en el caso de una empresa que interactúa con clientes y proveedores.

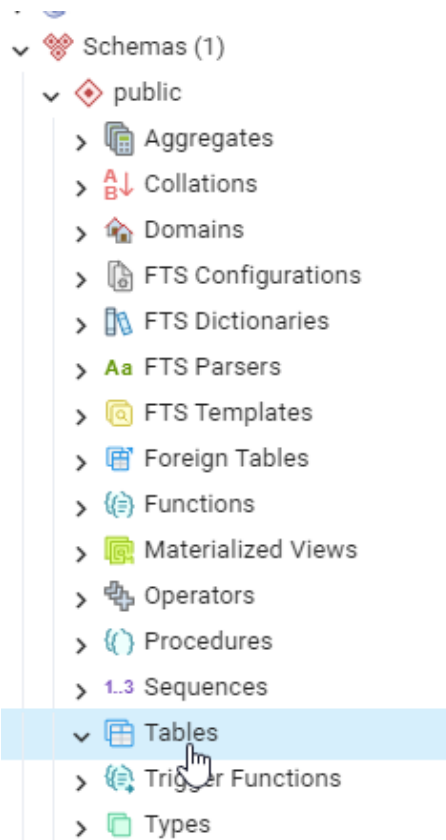
Ahora que hemos finalizado la creación de nuestro modelo relacional, procederemos a detallar los pasos necesarios para convertirlo a SQL y obtener el script que incluye todas nuestras tablas creadas. Tenemos dos alternativas: podemos crear una nueva base de datos o utilizar una ya existente. En este caso práctico, elegiremos la opción de utilizar la base de datos denominada "Prueba".



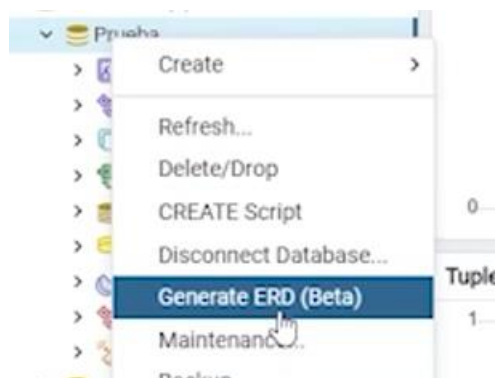
Es importante destacar que, si se elige crear una nueva base de datos, esta se generará inicialmente vacía, es decir sin tablas.

Create - Database

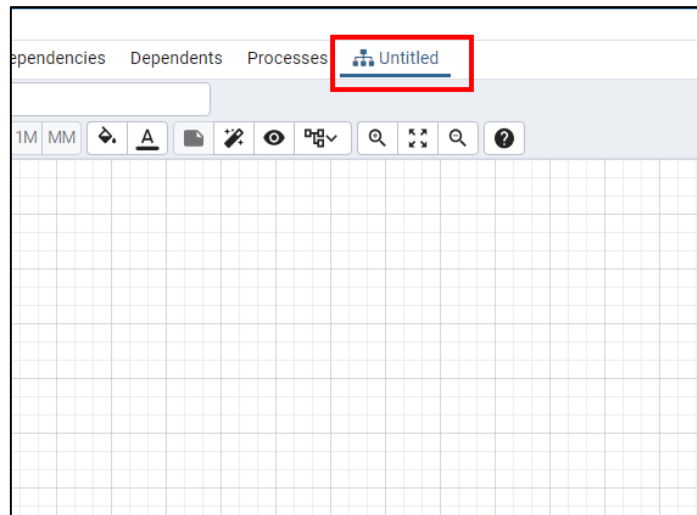
General	Definition	Security	Parameters	Advanced	SQL
Database	Pruebas				
OID					
Owner	 postgres				
Comment					



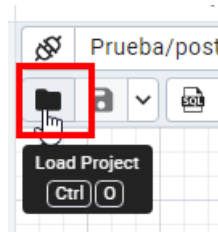
Ahora, haremos clic en la base de datos que vamos a utilizar y seleccionaremos la opción "Generate ERD".

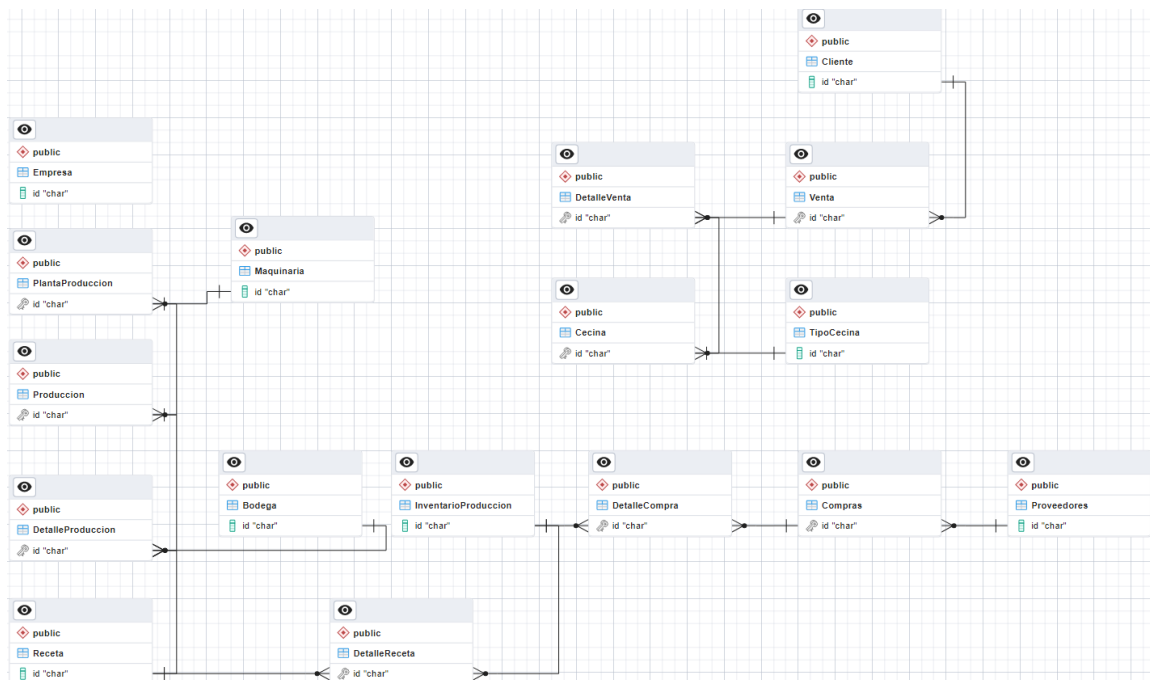


Esto abrirá una nueva pestaña donde podremos diseñar diversos modelos relacionales.

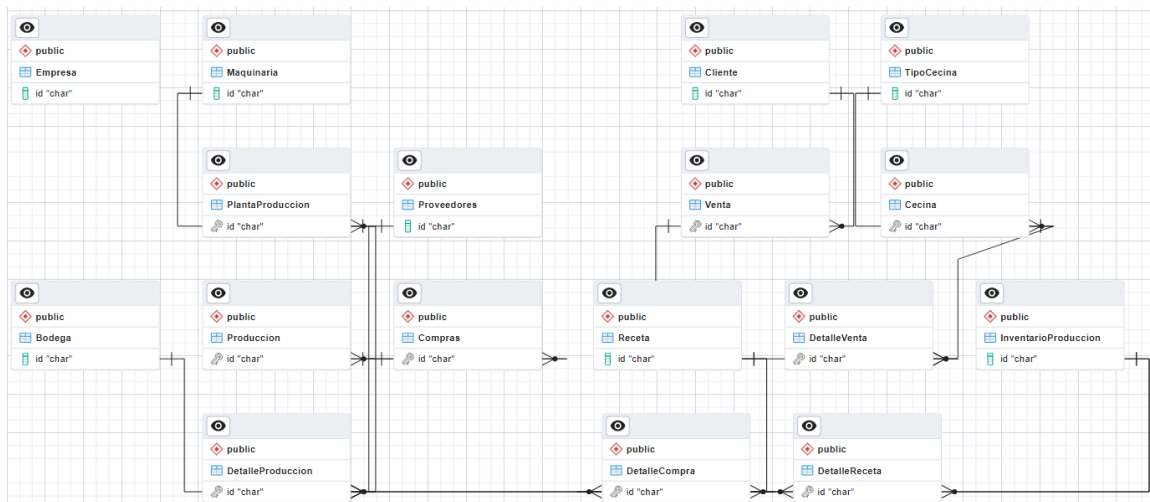
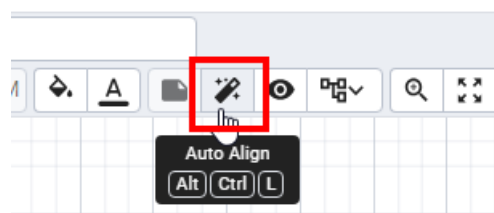


Después, podemos acceder a nuestro modelo relacional previamente creado simplemente haciendo clic en el ícono de archivo y buscar el diagrama que guardamos.





Si deseamos alinear nuestras tablas de manera más organizada, podemos hacer uso de la herramienta "Auto Align", lo que nos permitirá ordenar nuestro modelo de manera más eficiente.









Con esto realizado, ahora necesitaremos agregar más atributos a nuestras tablas. Comenzaremos con la tabla "Empresa", a la cual añadiremos el atributo "Rut" como nuestra clave primaria. También incluiremos el atributo "Nombre".

Table: Empresa (public)

General Columns Advanced Constraints

Columns







		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
			Rut	character varying v		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
			Nombbre	character varying v		<input type="checkbox"/>	<input type="checkbox"/>

Definiremos los atributos de las tablas maestras, que ocupan el nivel jerárquico más alto. Estas son: "Maquinarias", "Bodega", "Cliente" y "Proveedores". La tabla "Maquinaria" tendrá un atributo llamado "id", que servirá como clave primaria y no podrá ser nulo. Además, agregaremos el atributo "Nombre".

Table: Maquinaria (public)

General Columns Advanced Constraints

Columns



















		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
			id	numeric v		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
			Nombre	character varying v		<input type="checkbox"/>	<input type="checkbox"/>

La tabla de clientes contendrá los siguientes atributos: "Rut" como clave primaria, "Nombre", "Giro", "Dirección" (obligatorio), "Ciudad" y "Teléfono".


Table: Cliente (public)









General Columns Advanced Constraints

Columns

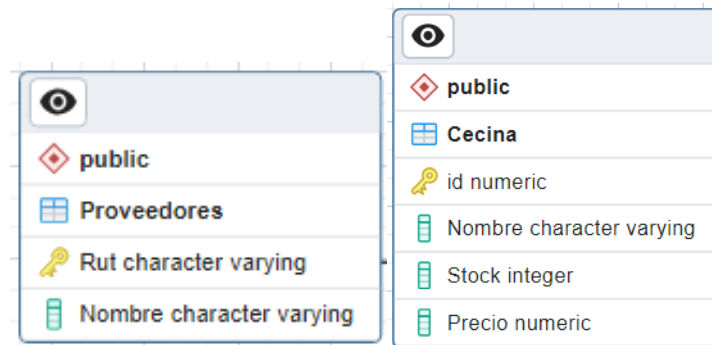
		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?
		 Rut	character varying v			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
		 Nombre	character varying v			<input type="checkbox"/>	<input type="checkbox"/>
		 Giro	character varying v			<input type="checkbox"/>	<input type="checkbox"/>
		 Direccion	character varying v			<input checked="" type="checkbox"/>	<input type="checkbox"/>
		 Ciudad	character varying v			<input type="checkbox"/>	<input type="checkbox"/>
		 Telefono	numeric v			<input type="checkbox"/>	<input type="checkbox"/>

Al observar la tabla de clientes, notaremos que los atributos definidos como clave primaria se representarán con el símbolo de una llave amarilla, mientras que los demás atributos se mostrarán en verde.

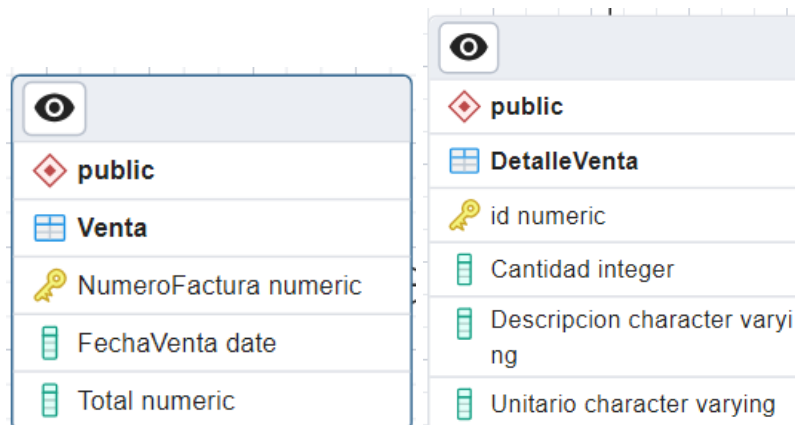


 public
 Cliente
 Rut character varying
 Nombre character varying
 Giro character varying
 Direccion character varying
 Ciudad character varying
 Telefono numeric

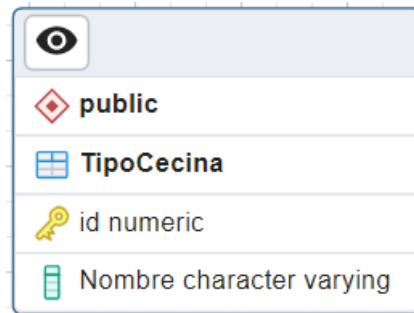
Continuamos ahora con la tabla "Proveedores", la cual contendrá los atributos "Rut" como clave primaria y "Nombre". En cuanto a la tabla "Cecina", dispondrá del atributo "id", que designaremos como su clave primaria, junto con "Nombre", "Stock" y "Precio".



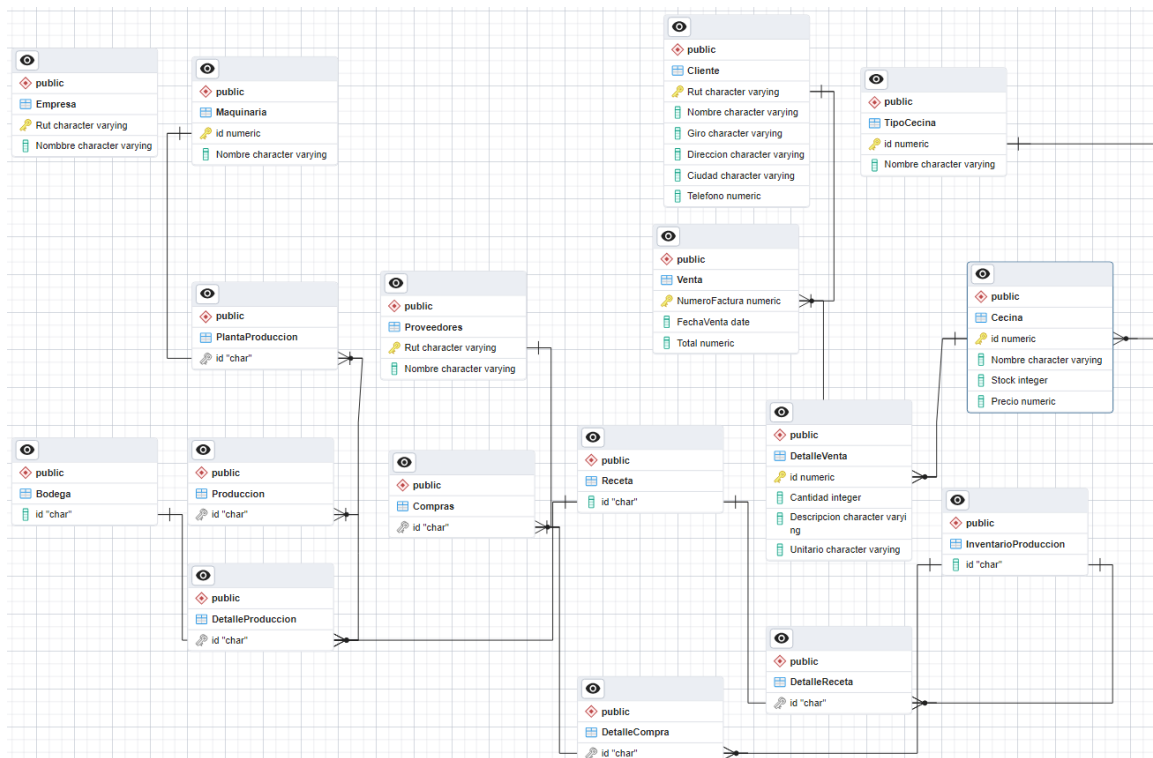
A continuación, procederemos a definir los atributos de la tabla "Venta", que serán: "NumeroFactura" como clave primaria, "fechaVenta" y "total". En cuanto a la tabla "DetalleVenta", contará con los atributos "cantidad", "descripción" y "unitario" (estos dos últimos serán obligatorios). Para incorporar una clave primaria, añadiremos el atributo "id".



En el caso de la tabla "TipoCecina", añadiremos los atributos "id" como clave primaria y "nombre" como dato obligatorio.

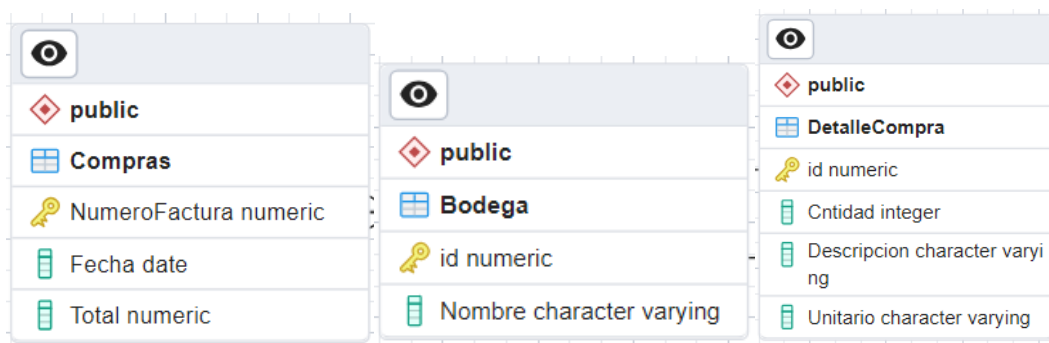


Revisemos nuevamente nuestro modelo para apreciar cómo va tomando forma.



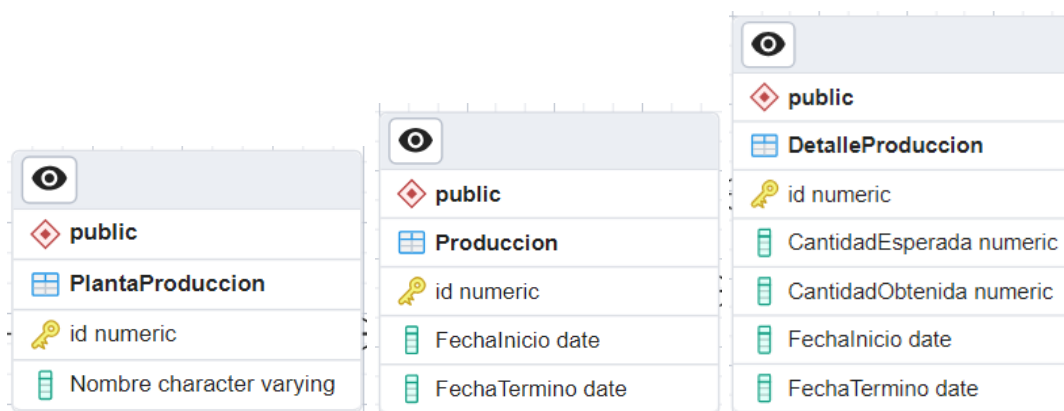
Continuamos con la tabla "Compras", que incluirá los atributos "NumeroFactura" como clave primaria, junto con "fecha" y "total" como datos obligatorios.

La tabla "Bodega" se caracterizará por tener el atributo "id" como clave primaria y el atributo "nombre" como un dato obligatorio. Para la tabla "DetalleCompra", definiremos el atributo "id" como llave primaria, así como los atributos "cantidad", "descripción" y "unitario", todos de carácter obligatorio.



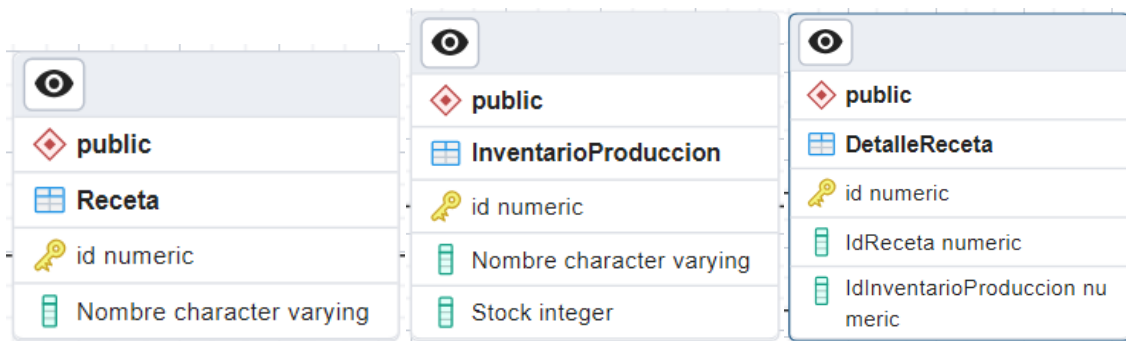
Ahora, avanzaremos con la tabla "PlantaProducción", en la que estableceremos los atributos "id" y "nombre". Respecto a la tabla "Producción", contará con los atributos "id" como clave primaria, "fechaInicio" de carácter obligatorio y "fechaTermino" de carácter opcional.

Siguiendo con la tabla "DetalleProducción", incorporaremos los atributos "id", "CantidadEsperada", "CantidadObtenida", "FechaInicio" y "FechaTermino".



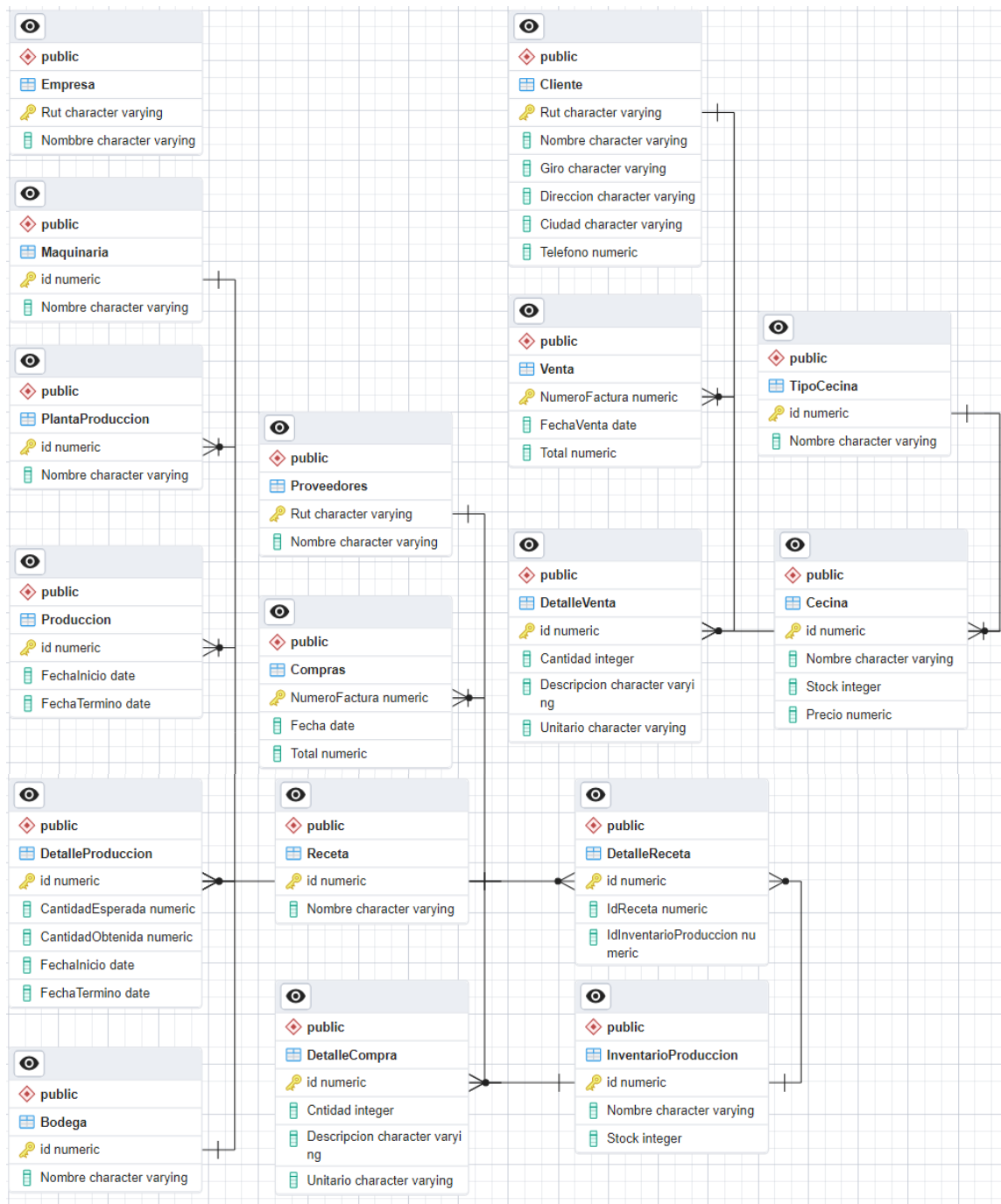
La tabla "Receta", por su parte, albergará los atributos "id" como clave primaria y "nombre". En cuanto a la tabla "InventarioProduccion", se distinguirá por contar con un atributo "id" como clave primaria, además de "nombre" y "stock", ambos de carácter obligatorio.

Finalmente, respecto a la tabla "DetalleReceta", conservaremos su atributo "id" y agregaremos los atributos "idReceta" e "IdInventarioProduccion".

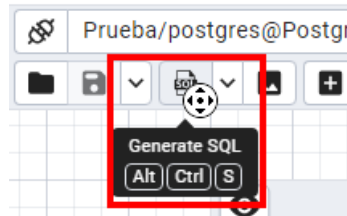


En todas las tablas anteriores, se debe tener siempre la precaución de renombrar las llaves foráneas cuando coincidan en nombre, para así hacer más entendible nuestro modelo, a pesar de que éstas no se ven reflejadas gráficamente.

Ahora podemos visualizar todas las tablas junto con sus respectivos atributos:



Ahora que hemos asociado todos los atributos a cada tabla, procederemos a generar el SQL de nuestras tablas creadas. Para ello, nos dirigimos al menú superior y seleccionaremos la opción "Generate SQL".



Este proceso nos proporcionará un código script para la creación de todas nuestras tablas con sus atributos correspondientes.

```
Query  Query History
6  CREATE TABLE IF NOT EXISTS public."Empresa"
7  (
8      "Rut" character varying NOT NULL,
9      "Nombbre" character varying,
10     PRIMARY KEY ("Rut")
11 );
12
13 CREATE TABLE IF NOT EXISTS public."PlantaProduccion"
14 (
15     id numeric NOT NULL,
16     "Nombre" character varying,
17     PRIMARY KEY (id)
18 );
19
20 CREATE TABLE IF NOT EXISTS public."Maquinaria"
21 (
22     id numeric NOT NULL,
23     "Nombre" character varying,
24     PRIMARY KEY (id)
25 );
26
27 CREATE TABLE IF NOT EXISTS public."Produccion"
28 (
29     id numeric NOT NULL,
30     "FechaInicio" date NOT NULL,
31     "FechaTermino" date,
32     PRIMARY KEY (id)
33 );
```