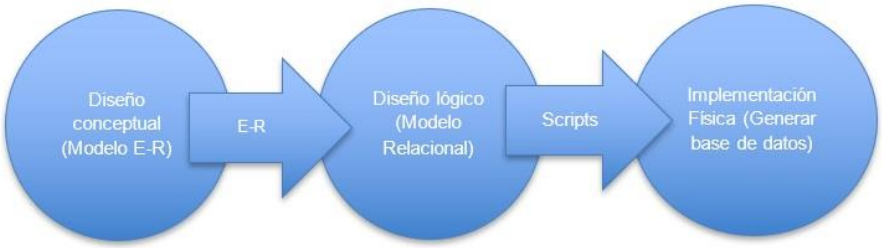
EXERCISES QUE TRABAJAREMOS EN EL CUE:

- 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 ENTIDAD RELACIÓN BASADO EN REGLAS DE NORMALIZACIÓN.
- EXERCISE 7: MODELO RELACIONAL A PARTIR DE UN MODELO ENTIDAD RELACIÓN.

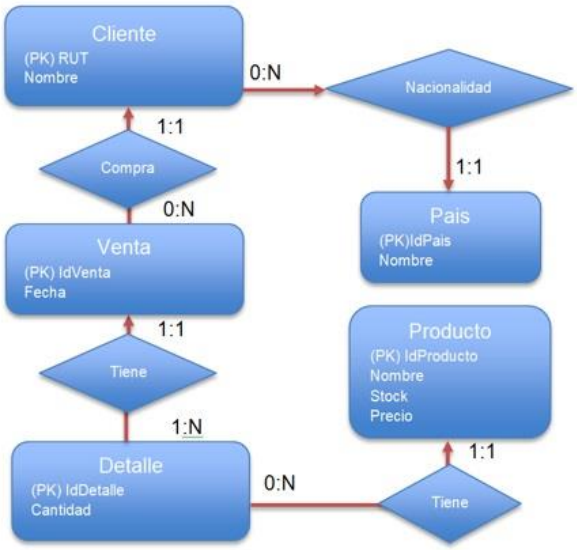
EXERCISE 1: MODELAR BASE DE DATOS RELACIONALES.

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.

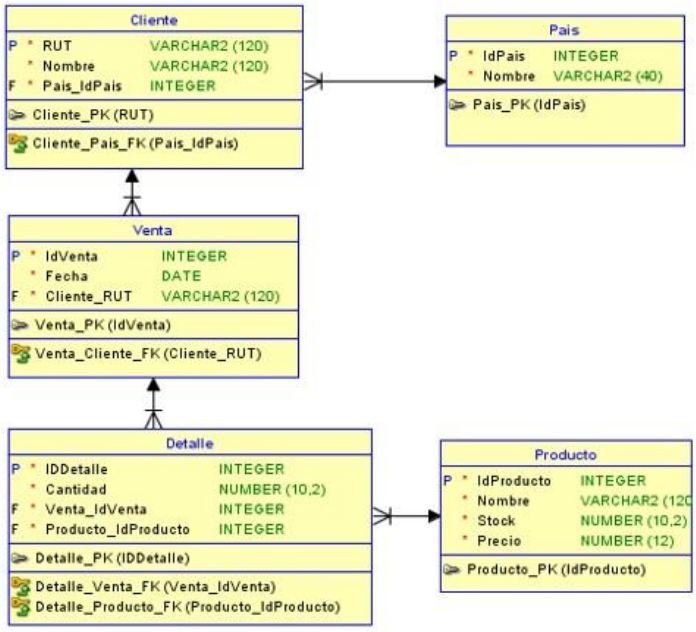


CREAR UN DISEÑO LÓGICO

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

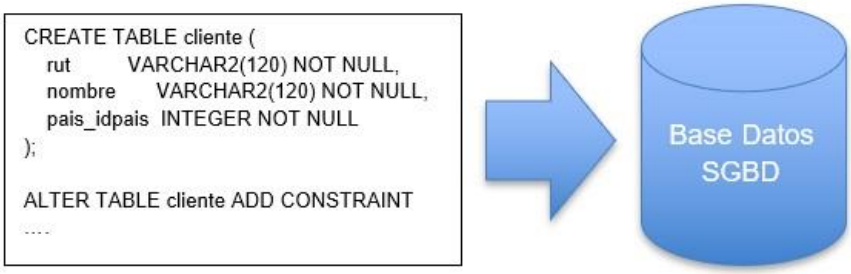
- Las entidades son implementadas como tablas.
- Se definen las características de cada atributo (tipo dato, permite nulo, …).
- Se crean las llaves foráneas para representar las relaciones.

El modelo anterior quedaría como este implementado en Data **modeler**:



IMPLEMENTACIÓN FÍSICA

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

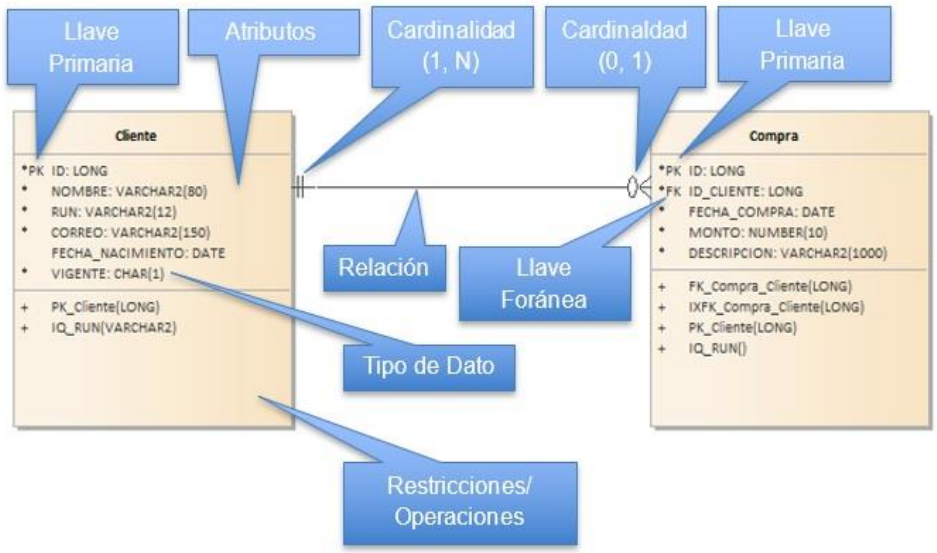
Es un subconjunto del conjunto de atributos comunes en una colección de entidades, la cual permite identificar inequívocamente cada una de las entidades pertenecientes a dicha colección. Asimismo, permiten distinguir entre sí, las relaciones de un conjunto de éstas.

- *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.

ÍNDICES

Es un subconjunto del conjunto de atributos comunes en una colección de entidades, los cuales son ordenados mediante un índice; todas las claves, a su vez, también tienen uno.

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, …

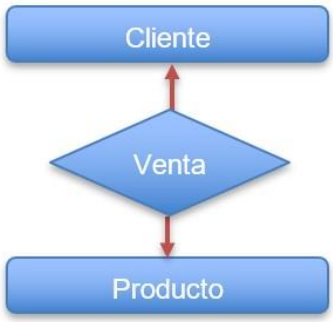
PASOS PARA CREAR UN MODELO ENTIDAD - RELACIÓN

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 la llave primaria de cada entidad.
6. Se normaliza el modelo.
7. Se optimiza el modelo.

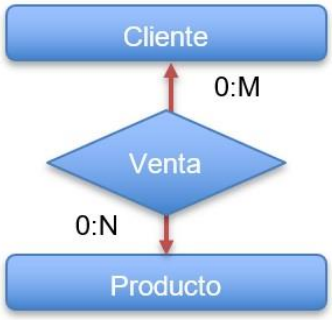
Realizaremos un ejemplo, en el cual se requiere construir un modelo ER para el registro de la venta de productos. Se debe contar con un registro del nombre de cada cliente; la venta debe tener la fecha y el monto total; y además, se debe identificar el país de cada cliente; los datos del producto que se vende son: IdProducto, Nombre, Stock y Precio.

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

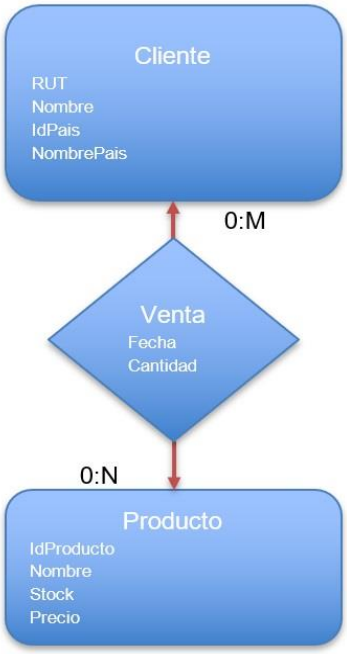
2. Crear las relaciones:



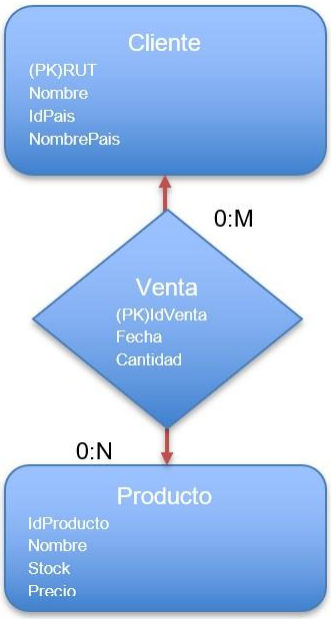
3. Indicar cardinalidad de las relaciones:



4. Se definen los atributos de las entidades:

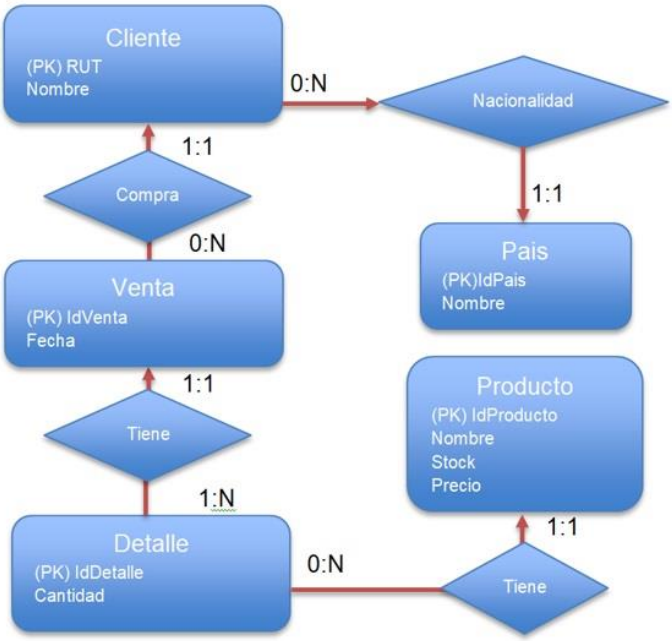


5. Se identifica la llave primaria de cada entidad:

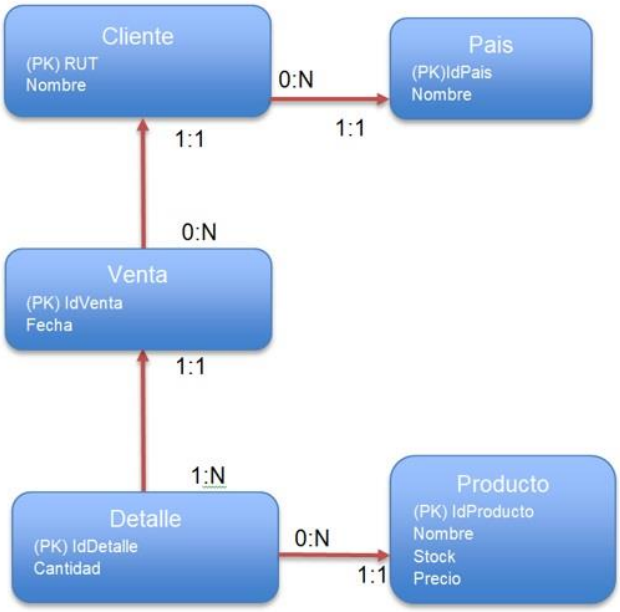


6. Se normaliza el modelo:

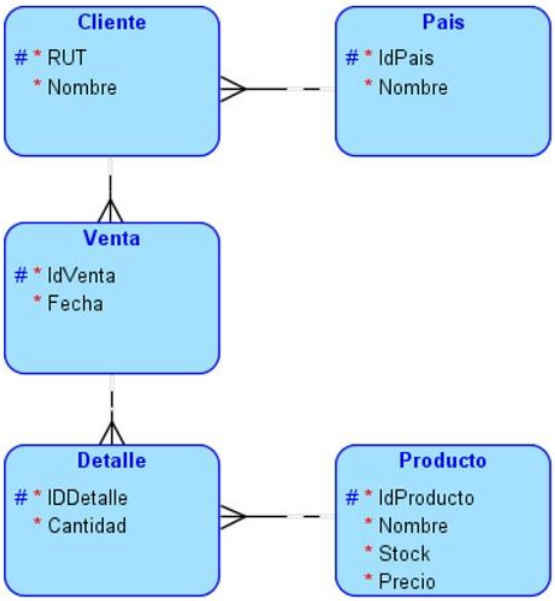
   El proceso de normalización se verá en la siguiente sección.



El cual, por simplicidad, lo representaremos de la siguiente forma:



Y el mismo modelo, esta vez creado en Data modeler de Oracle:



7. Se optimiza el modelo:

   El proceso de optimización generalmente se debe a una desnormalización del modelo, lo que se verá en la siguiente sección.

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 Articulo (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)	**Articulo**	**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 ENTIDAD - RELACIÓN BASADO EN REGLAS DE NORMALIZACIÓN.

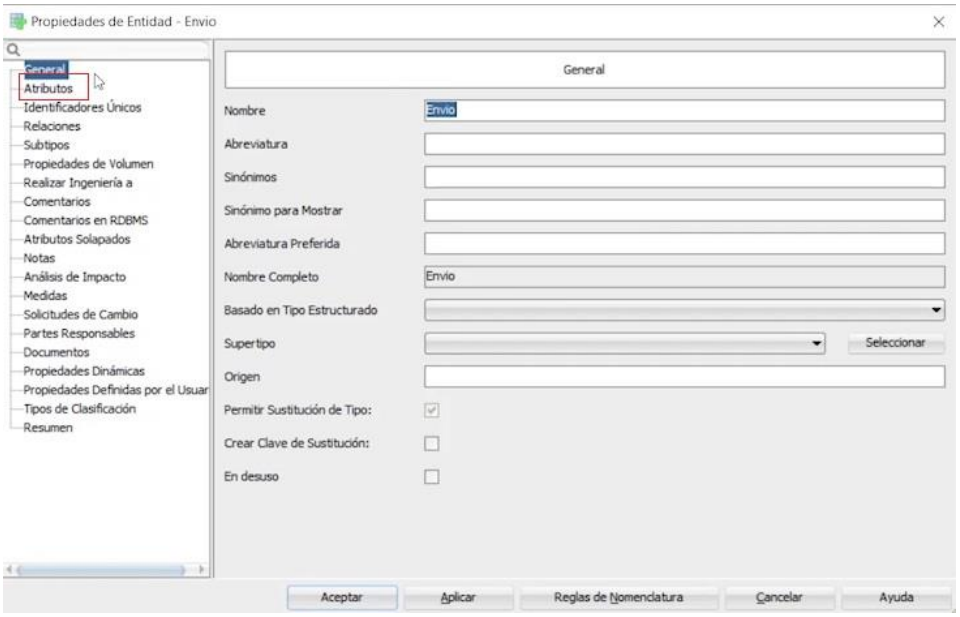
Para continuar con las ejemplificaciones, trabajaremos basándonos en un ejercicio propuesto:

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.

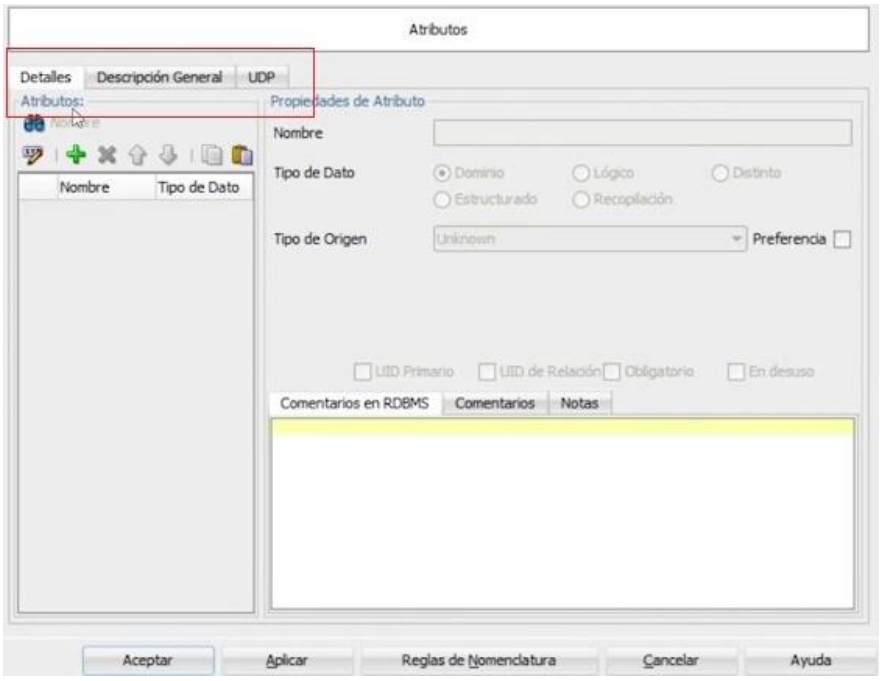
Para realizar la ejemplificación, crearemos un Modelo Entidad - Relación normalizado. Trabajaremos en la ventana Logical del programa Data Modeler.

Empezamos a graficar nuestras tablas, correspondientes a las tres entidades que identificamos desde el enunciado: “Envío”, “Pedido” y DetallePedido”. Al graficarlas, se despliega una ventana que permite agregarle sus atributos.

Al costado izquierdo de esta ventana, se muestra una lista con las actividades que podemos realizar, seleccionamos “Atributos”.

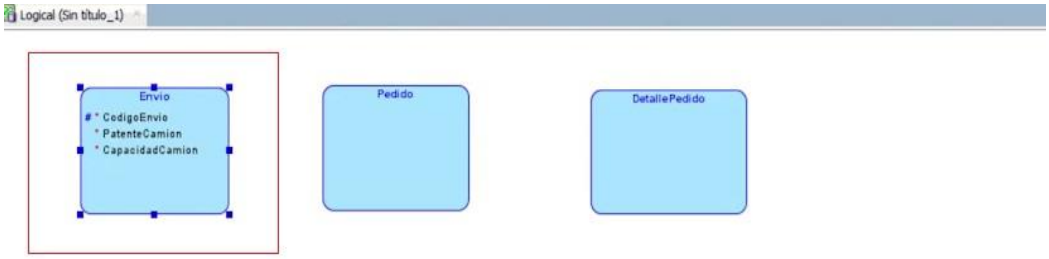


Al hacer clic en “Atributos”, nos muestra una nueva ventana. En la pestaña de “Detalles”, aparece una subsección al lado izquierdo, llamada “Atributos”. Cuando hacemos clic sobre el signo “+” de color verde, podemos agregar un nuevo atributo para la entidad; el signo “x” de color rojo, sirve para eliminar un atributo; mientras que las flechas de color celeste “arriba y abajo”, sirven para reordenar los atributos según el orden en el que queremos que se desplieguen.



En la subsección de la derecha, se escribe el nombre de la propiedad, para el primer caso será "CódigoEnvío"; el tipo de dato queda en "Dominio" por defecto; el tipo de origen se deja tal cual; y se puede seleccionar si ese atributo será el "UID primario", o si solo será un campo obligatorio. En este caso, es la llave primaria, así que seleccionamos "UID primario". De inmediato agregamos los atributos "PatenteCamión" y "CapacidadCamión", y los haremos obligatorios. Finalmente, seleccionaremos "Aceptar" para poder terminar el ingreso de los atributos.

De esta forma, ya podemos observar nuestra primera entidad con sus atributos correspondientes:



Continuaremos con la tabla "Pedido", que tiene los atributos "Cliente", "DirecciónCliente" y "TotalPedido". Para este caso, crearemos un cuarto atributo para utilizar como llave primaria, y lo llamaremos "IdPedido".

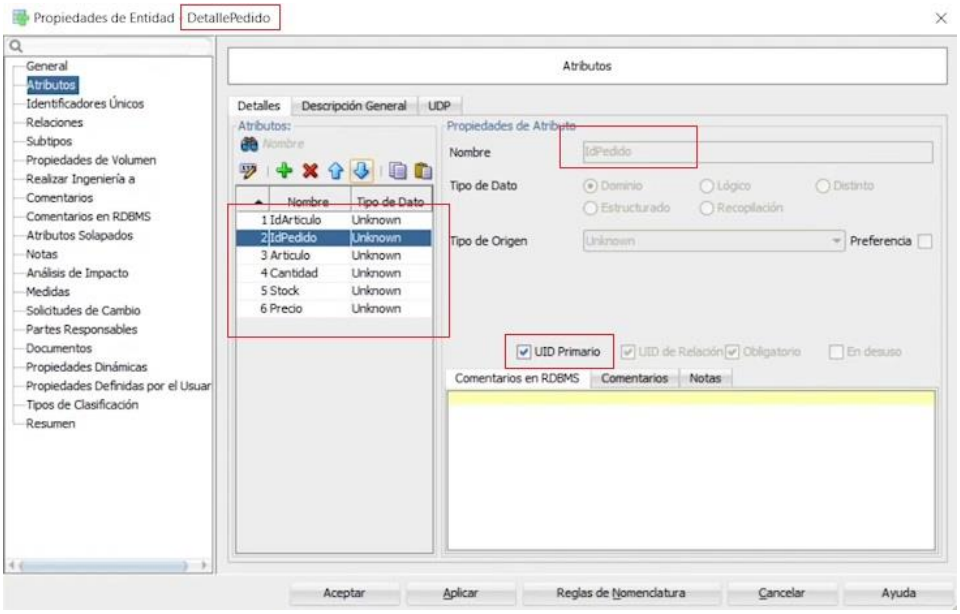
> Como buena práctica, dejaremos las llaves como el primer atributo de la tabla.

Finalmente, a la entidad "DetallePedido" le agregaremos los cuatro atributos: "Artículo", "Cantidad", "Stock" y "Precio". Agregamos un quinto atributo llamado "IdArticulo" como llave primaria, sin embargo, esta entidad tendrá una llave primaria compuesta entre los atributos "IdArticulo" y "IdPedido". Esto lo veremos al realizar las relaciones.

Comenzaremos a generar las relaciones entre las entidades. Un “Envío” puede contener “N” cantidad de “Pedidos”, y un “Pedido” puede tener más de un “DetallePedido”.

Para generar la llave compuesta en la tabla “DetallePedido”, podremos ver que aparece el atributo “IdPedido”. Ahora lo marcamos como llave primaria, lo subimos dentro de la lista de atributos, y estará listo.

Se ha generado la llave compuesta:



Hasta ahora, nuestro modelo ha cumplido con la primera forma normal. Continuaremos viendo si estas tablas cumplen con la segunda forma normal.

Las tablas “Envío” y “Pedido” cumplen con la segunda forma normal, pues cada uno de los atributos de éstas dependen de sus respectivas llaves primarias. Para el caso de la tabla “DetallePedido”, esta cuenta con una llave compuesta, por lo que todos sus atributos dependen

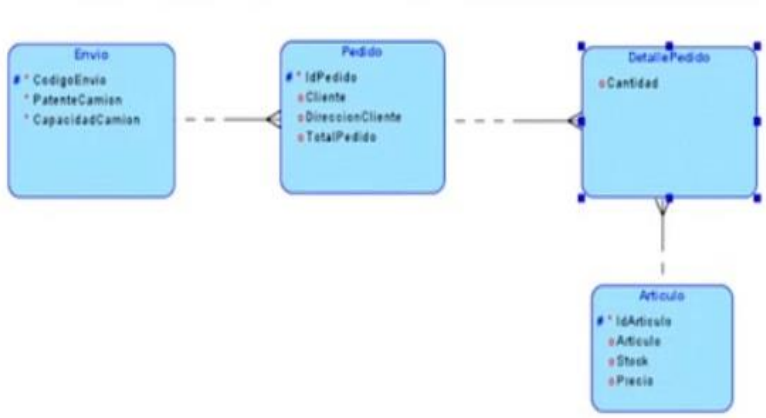
de dicha llave, menos los atributos “Stock” y “Precio”, que dependen respectivamente de las llaves “IdPedido” e “IdArticulo”.

Para subsanar esta problemática, nos vemos en la obligación de crear otra tabla en nuestro modelo, a la que llamaremos “Artículo”, y esta tendrá el atributo “IdArticulo” que será llave primaria y los atributos “Stock” y “Precio”.

Haremos una relación con la tabla “DetallePedido”, que puede contener muchos artículos, y eliminamos el atributo “IdArticulo” que contenía antes, manteniendo el referente a la tabla “Articulo” “IdArticulo1”.

Eliminaremos todos los atributos que se repiten, dejando solo: “IdArticulo1”, “IdPedido”, y “Cantidad”.

Ahora, ya podemos observar que nuestro modelado cumple de forma completa con la **segunda forma normal**.



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

Comenzamos revisando la tabla “Envío”, en la cual podemos observar que “PatenteCamión” no forma parte de la llave primaria, pero sí de “CapacidadCamión”. Debido a este detalle, la tabla no cumple con la tercera forma normal, obligándonos a crear otra para normalizarla.

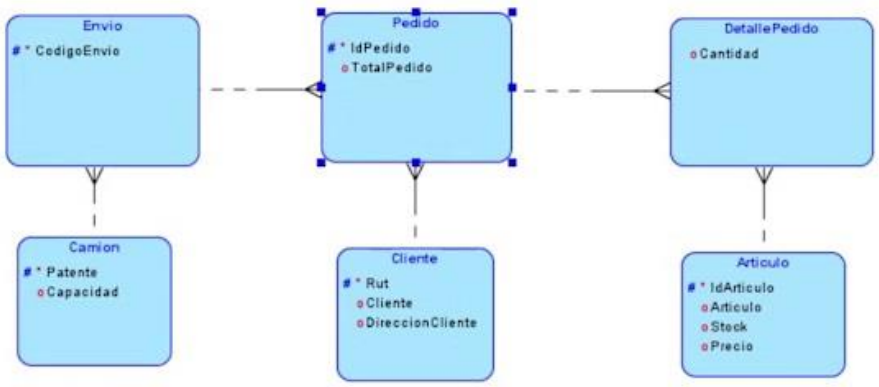
Creamos una nueva entidad con el nombre “Camión”, y los atributos “Patente” y “Capacidad”. “Patente” actuaría como la llave primaria.

Ahora nos falta establecer la relación con esta nueva tabla, por lo que indicaremos que un camión puede efectuar varios envíos. En la entidad “Envío” eliminamos los atributos redundantes, dejando solo “CódigoEnvío” y “Patente”.

Esta situación se presenta también en la tabla “Pedido”. Para subsanarlo, crearemos una nueva entidad de nombre “Cliente” con los atributos: “DirecciónCliente”, “Cliente” y “Rut”, siendo este último la llave primaria.

Posteriormente, graficamos la relación que establece que un cliente puede efectuar más de un pedido. Por último, lo que nos resta hacer es eliminar los atributos redundantes, dejando solo: “IdPedido”, “TotalPedido”, “CódigoEnvío” y “Rut”.

Finalmente, nuestro modelo quedará de la siguiente forma:



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

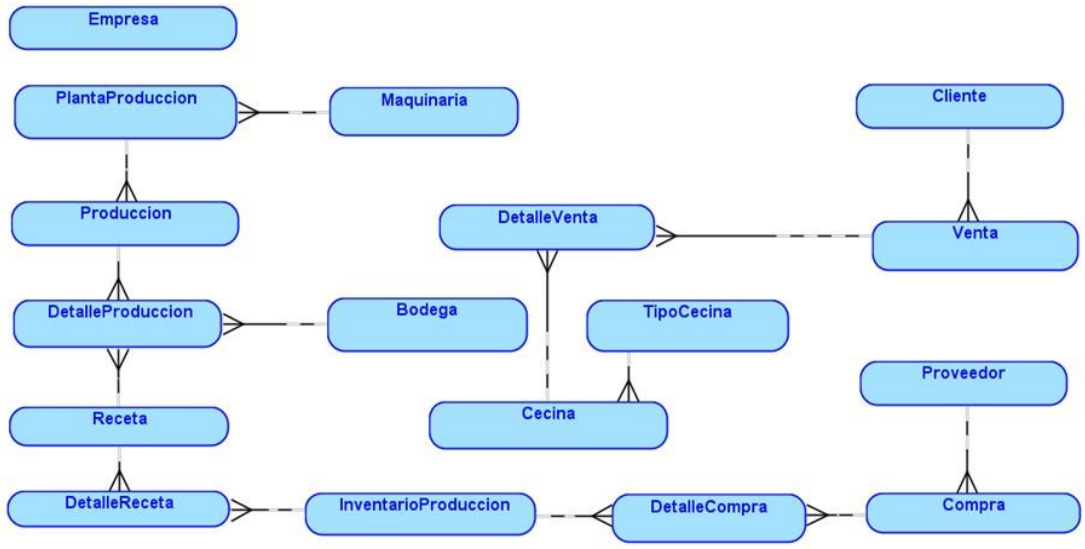
Basándonos en el modelo de la empresa de cecinas que desarrollamos en la primera sesión práctica, lo completaremos definiendo, de manera intuitiva, los atributos de las tablas o entidades del ejercicio.

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,

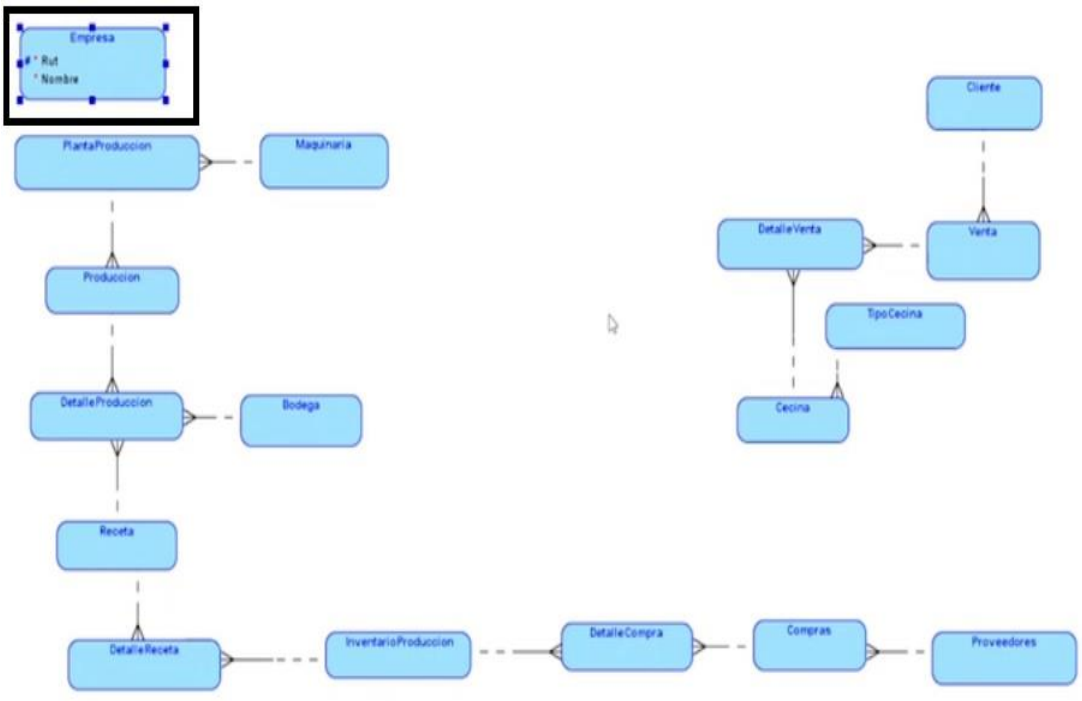
compra de materias primas e insumos a proveedores, planta productiva, y máquinas donde se produce cada partida y su clasificación en lotes. En su producción de cecinas, se requiere conocer la merma producida por kilo de cerdo según el tipo de éstas. Las partidas son almacenadas en bodegas, y es necesario 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 esta ejemplificación, usaremos el programa **Data Modeler**, e iremos a archivos y abriremos el ejercicio que ya habíamos desarrollado.

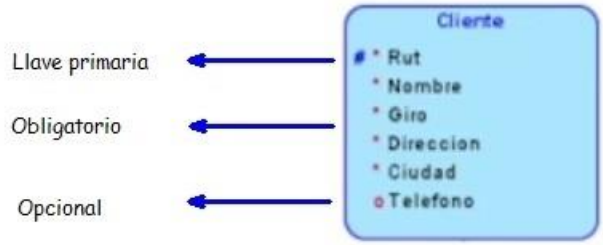
En la entidad “Empresa”, agregaremos el atributo Rut que será nuestra clave primaria. También incluiremos el atributo *nombre*.



Definiremos todos los atributos de las tablas maestras, que son aquellas con mayor nivel jerárquico. Estas tablas son: “Maquinaria”, “Bodega”, ”Cliente”, “Proveedor” y “Cecina”.

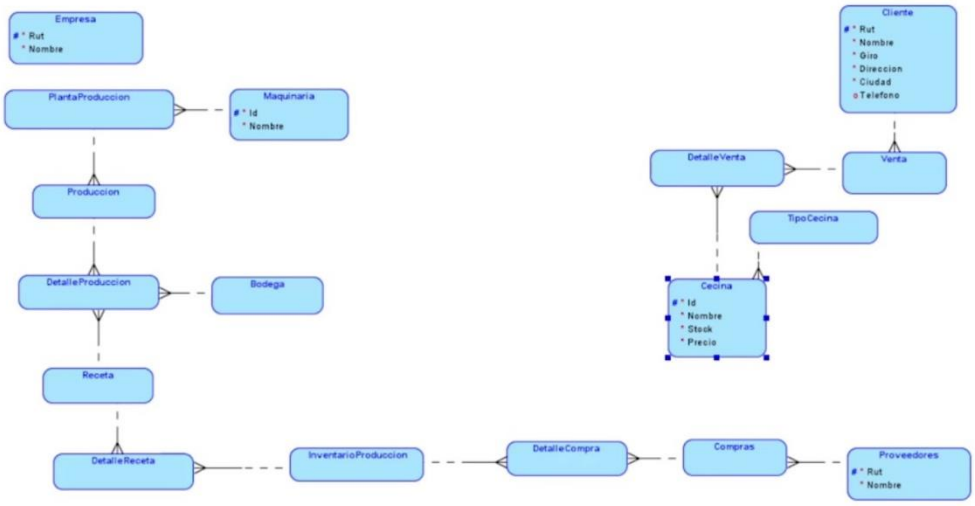
A la tabla “Maquinaria” le agregaremos los atributos id, como clave primaria y nombre. Por su parte, la tabla “Cliente” tendrá el atributo rut como clave primaria, nombre, giro, dirección que le indicaremos que es obligatorio, ciudad y teléfono.

A medida que creamos nuestra tabla, podemos ir observando que al lado de cada atributo aparecen símbolos. La almohadilla o “gato” es el símbolo de la llave primaria, mientras que el “asterisco” significa que el atributo es obligatorio. El circulo por su parte, representa un atributo opcional.



La tabla “Proveedor”, tendrá los atributos rut como llave primaria y nombre. La entidad “Cecina, tendrá el atributo id, el cual marcaremos como su primary key, además del atributo nombre, stock y precio.

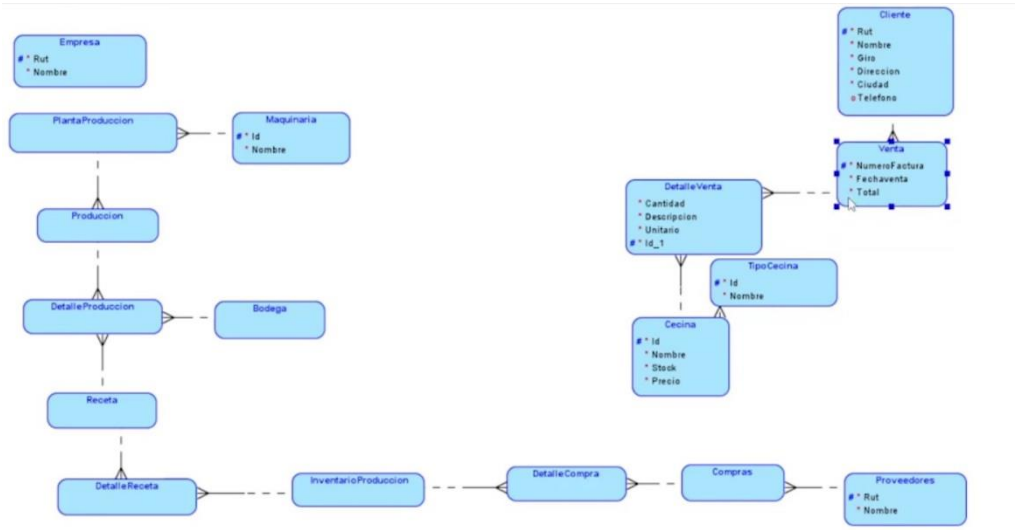
Hasta el momento, nuestra tabla se ve de la siguiente forma:



Continuaremos definiendo los atributos de la entidad “Venta”, que serán: numeroFactura, como primary key, fechaVenta y total. Por su parte, la tabla “DetalleVenta” tendrá los atributos cantidad, descripción, y unitario (estos dos últimos que serán obligatorios). Además, para incorporar una llave primaria, agregaremos el atributo ID.

Para el caso de la entidad “TipoCecina”, le agregaremos los atributos id como llave primaria, y nombre como dato obligatorio.

Observamos nuevamente nuestro modelo para ver cómo va tomando forma:



Continuamos con la tabla “Compras”, que contará con los atributos númeroFactura, como primary key, además de fecha y total, de carácter obligatorio.

La entidad “Bodega” tendrá el atributo id, que será su clave primaria, y el atributo nombre de tipo obligatorio. Para el caso de la tabla “DetalleCompra”, esta tendrá el atributo id como llave primaria, además de cantidad, descripción y unitario, todos de carácter obligatorio.

Continuando con la tabla “PlantaProducción”, le definiremos un id y un nombre como atributos. Mientras que la entidad “Producción”, tendrá los atributos id como llave primaria, fechaInicio de carácter obligatorio, y fechaTermino de carácter opcional.

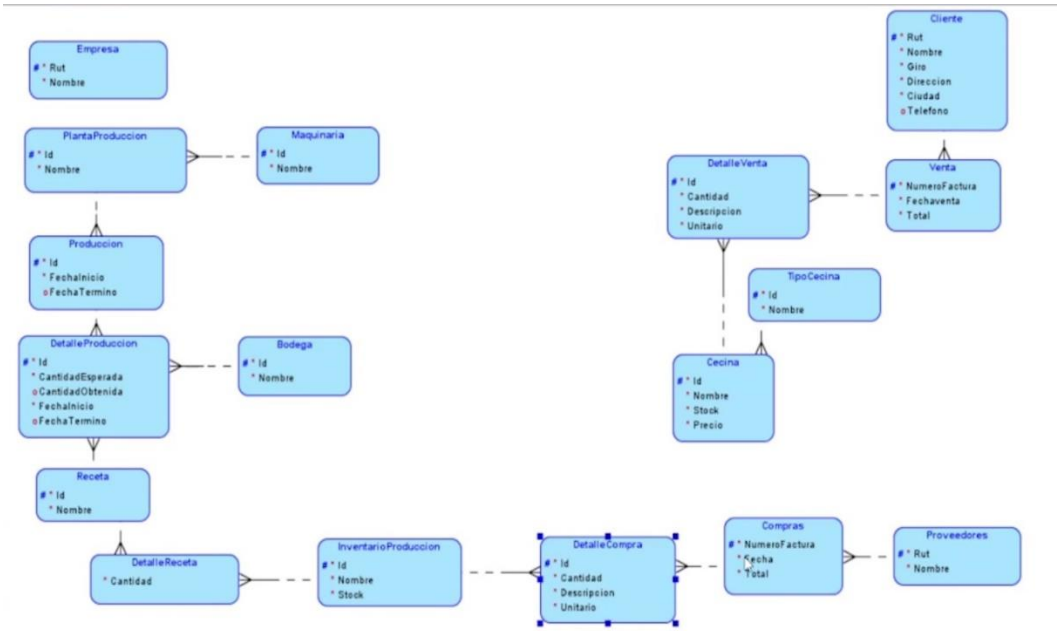
Continuamos con la tabla “DetalleProducción”, la cual contendrá los atributos id, CantidadEsperada, CantidadObtenida, FechaInicio y FechaTermino. La tabla “Receta”, por su parte, contendrá los atributos id como primary key, y nombre.

La entidad “InventarioProduccion” tendrá un atributo id primary key, además de nombre y stock, ambos de carácter obligatorio.

Finalmente, a la tabla “DetalleReceta”, simplemente le renombraremos las llaves foráneas que contiene como idReceta y IdInventarioProduccion. Para que contenga un atributo propio, le colocaremos el atributo cantidad.

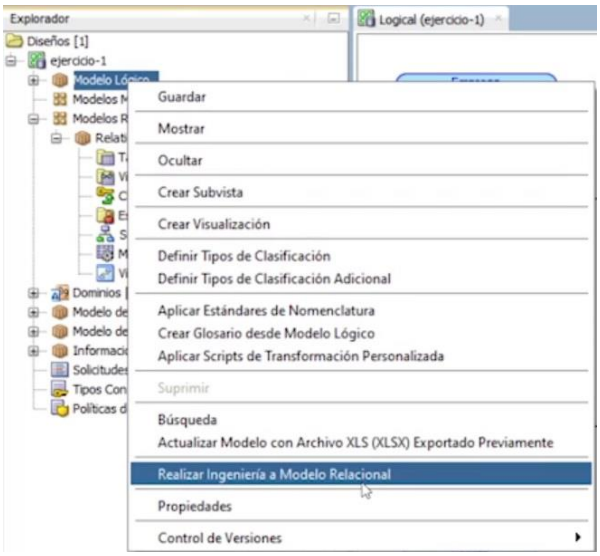
En todas las entidades anteriores, se debe tener siempre la precaución de renombrar las llaves foráneas cuando coincidan en nombre, para así hacer más entendible nuestro modelo, a pesar de que éstas no se ven reflejadas gráficamente.

Ya podemos observar todas las entidades con sus respectivos atributos:

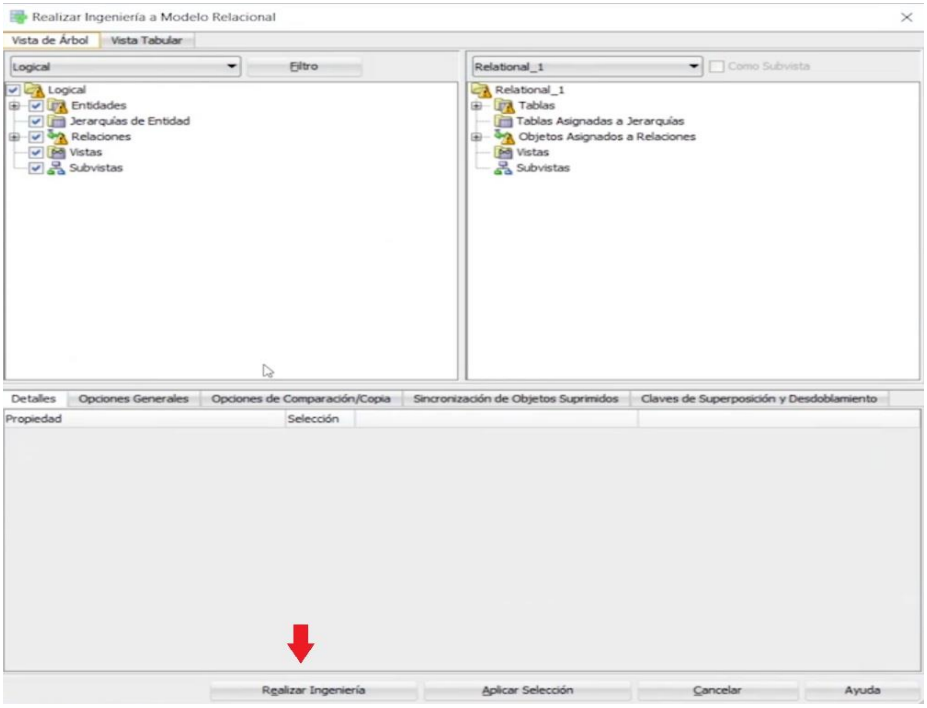


CONVIRTIENDO NUESTRO MODELO ER A MODELO RELACIONAL

Finalmente, convertiremos este **Modelo Entidad – Relación,** a un **Modelo Relacional**. Para ello, nos dirigiremos al ítem "modelo lógico", y seleccionaremos la opción "Realizar ingeniería a Modelo Relacional".



Se despliega una ventana, en la cual seleccionaremos la opción "realizar ingeniería".



Al hacer esto, se nos abre una nueva pestaña de nombre “Relational_1”, en la que podemos ver una nueva gráfica donde, además de tener sus relaciones, podemos ver el detalle de los atributos, tipo de datos, llaves primarias y foráneas, y también nos muestra de manera gráfica qué campos son requeridos, y cuáles no.