UNIVERSIDAD NACIONAL ABIERTA
VICERRECTORADO ACADÉMICO
AREA: INGENIERÍA

TRABAJO PRÁCTICO

ASIGNATURA: Base de Datos

CÓDIGO: 311

FECHA DE ENTREGA AL ESTUDIANTE:
Al inicio del lapso académico, a través de la plataforma de Diseño Académico.

FECHA DE DEVOLUCIÓN POR PARTE DEL ESTUDIANTE:
Enviar al correo electrónico de la Unidad Académica y asesor del Centro Local, hasta el día 17/04/2021 sin prórroga.

NOMBRE DEL ESTUDIANTE:Samuel Jesús Rodriguez Figueroa

CÉDULA DE IDENTIDAD: 27.240.851

CORREO ELECTRÓNICO DEL ESTUDIANTE: samuelr76@gmail.com

TELÉFONO: 0424-1917803

CENTRO LOCAL: (0400) Aragua Sede CARRERA: 236

LAPSO ACADÉMICO: 2021-1

NUMERO DE ORIGINALES:

FIRMA DEL ESTUDIANTE:samuelrodriguez

UTILICE ESTA PÁGINA COMO CARÁTULA DE SU TRABAJO

RESULTADOS DE CORRECCIÓN:

OBJ. Nº		**5**	**6**	**7**	**8**
0:NL	**1:L**				

Especialista: Nelly Escorcha

Introducción

La base de datos es un conjunto almacenado de archivos que se relacionan entre ellos por medio de la metodología entidad relación. Cada archivo representa una entidad, en dicha entidad se pueden almacenar múltiples registros. Los registros están compuestos por múltiples campos de datos individuales. En este Trabajo práctico se desarrollara una base de datos partiendo desde el concepto hasta definirla y crearla con un gestor de base de datos relacional.

Este trabajo está compuesto de cinco partes, que vienen hacer los cinco objetivos indispensables para realizar el presente trabajo, y son los siguientes:

Objetivo 7.1, Es la etapa de obtención y análisis de requisitos, en ella se estudia, analiza y se define los requerimientos de datos para el Sistema de información. En esta etapa se estudian los diferentes usuarios que harán uso de la base de datos, con el fin de recabar los datos necesarios para cada y por tipo de usuario. Los datos obtenidos de los usuarios serán el eje central de datos por el cual se estructurará el Sistema de información de la base de datos desarrollar.

Objetivo 7.2, En esta etapa se estudiara el diseño conceptual de la base de datos, en ella se definirán las entidades y relaciones que serán necesarias para el desarrollo de la base de datos, todo esto desde una perspectiva conversacional pero debidamente estructurado. Esta etapa servirá como borrador para comenzar definir el sistema de la base de datos.

Objetivo 8.1, en esta etapa se elige en cual Sistema de gestión de base de datos relacional se desarrollará la base de datos del trabajo practico, para ello se estudiaran diferentes SGBDR del mercado y se compararan entre ellas para elegir la que más se adapte a los criterios de Trabajo practico, y a las posibilidades viables de su utilización.

Objetivo 8.3, en esta etapa se diseñara de forma lógica, documenta y escrita la estructura de los datos, orientado a su implementación en su implementación en SQL. Se estudiara la forma correcta en que se deberán relacionar las entidades para la correcta optimización de la base de datos. Se normalizará todas las entidades y atributos desarrollados hasta este punto, con el fin de formalizar y simplificar las entidades y sus atributos.

Objetivo 8.2 en esta etapa se implementa el diseño de la base de datos desarrollada en el trabajo práctico, hacia el sistema de gestión de base de datos relacional escogido. Para ello se implementara criterios de almacenamiento, criterios de seguridad y reglas de negocios que permitan mantener la base de datos en un estado consistente ante cualquier transacción.

Adjunto al trabajo estará el el backup de la base de datos desarrolladas en SQL, con la cual se podrá regenerar la base de datos completa desde su catalogo en el SGBDR postgres.

Estudio de Caso

En la exportadora de flores “FLORENCIA”, se lleva el inventario de flores exóticas mediante un libro Excel, el cual le ha funcionado bastante bien hasta el momento. Sin embargo, la gerencia informática, desea agregar nuevas funcionalidades y vaciar la información de este libro Excel a un sistema de B.D. relacional.

Para el desarrollo de las nuevas funcionalidades, se necesitará registrar en una B.D. los datos más sensibles que permitirán generar estadísticas tanto mensuales como anuales, comparativas con años anteriores para fines estratégicos de comercialización y también la presentación de un dashboard para el uso de la junta directiva, permitiéndoles tener un panorama general del área comercial.

Para el registro de la información anteriormente nombrada se requiere el desarrollo de una B.D. relacional, que garantice la integridad de los datos, y cuyas funcionalidades serán las siguientes:

- Registrar los datos del producto, vaciando la información contenida en la hoja Excel como son: los datos del Producto: Código, Cantidad, Costo, Precio unitario, Descripción, Tipo, impuestos, montos de ventas.

- Registrar los clientes: Razón social, Rif, Dirección, zona de venta, teléfonos, e-mail.

- Registrar a los vendedores: Nombre y apellidos, dirección, cédula, teléfono, e-mail, zona de venta.

- Registrar las ventas contenidas en la hoja Excel por mes y año para fines estadísticos: fecha de venta, monto, zona de venta.

La alta gerencia de la empresa de flores, considera el control de inventarios como el área más estratégica para su organización, así que decide priorizar *el diseño de la base de datos relacional que* de soporte al sistema de información (a desarrollar a futuro) cuyos requerimientos funcionales han sido descritos en los párrafos anteriores.

Solución

7.1 **Obtención y Análisis de Requisitos**

La empresa exportadora de flores “Florencia”, es una empresa del tipo compra y venta (Compra flores en el exterior y las vende a nivel nacional). Este tipo de empresa comprende una estructura organizacional más simple en comparación con empresas como los bancos ó aeropuertos, sin embargo, la estructura organizacional de esta empresa, al igual que todas las empresas requieres de entidades que se encarguen de las ventas de los productos, el almacenado de los artículos, control administrativo financiero sobre las ventas y atención especializada hacia el cliente objetivo.

Los objetivos de las empresas “Florencia” son las de expandir y aumentar su causal de venta a nivel nacional. En dicho sentido, la empresa desea desarrollar una base de datos con el fin de agilizar (automatizar) procesos habituales en sus sistema de ventas.

La base de datos a desarrollar debe cumplir con las necesidades y funcionalidades que la empresa necesita, las cuales se deben debatir y estudiar. Dichas funcionalidades y operación específicas deben estar pensadas para satisfacer las necesidades y usos de los usuarios (clientes).

La investigación y análisis de los requerimientos funcionales para el desarrollo de la base de datos que responda a los requerimientos de la empresa Florencia se deben realizar de la mano y en comunicación constante con la gerencia de la empresa, así como con los distintos departamentos, atendiendo además a la estructura y organización física de la empresa, con el fin último de desarrollar una lista de requerimientos de datos para la base de datos. Sin embargo, como este es un trabajo del tipo teórico práctico, se debe proceder a realizar los requerimientos funcionales para la base de datos de la empresa “Florencia” atendiendo a las pautas indicadas en la situación problema, y completada con datos que se observan y preveen para dicho sistema.

Los requerimientos funcionales y de datos que se desea implemente la base de datos, son lo siguiente:

1- Almacenar los informes de ventas de manera individual, pero organizada en su conjunto, de manera tal que se pueda acceder y realizar diferentes tipos de búsqueda atendiendo a criterios de fechas.
2- El control y almacenado de las flores se realicen por una serie de almacenes, los cuales estén asignados a vendedores especializados en zonas especificas del país.
3- Permitir un sistema de información y de datos relacionados tales que, en su conjuntos, sirva para desarrollar todos los datos y formulas orientados a la compra y venta de bienes.

Esta representa una vista muy general hacia las principales funcionalidades de la base de datos a desarrollar, es posible que en el estudio y desarrollo de la misma se descubran nuevas funciones y operaciones necesarias.

- ✓ **Una lista y descripción de cada uno de los usuarios que habrán de utilizar los datos contenidos en la base de datos a diseñar.**

Ya definida los requerimientos funcionales de la base de datos a desarrollar, toca identificar a todos los usuarios que harán uso de esta base de datos, pues, las bases de datos se desarrollan con el fin de servir y facilitar las operaciones a los usuarios comunes. Por lo cual se define la siguiente la tabla:

Lista de usuarios

Tipos de usuarios	Descripción
Cliente	Usuarios comunes que realizan las compras de las diferentes flores en la empresa. Usuarios externos al sistema empresarial.
Vendedor	Los diferentes empleados certificados por la empresa ó vendedores terceros que facilitan el proceso de ventas a los usuarios (clientes).
Administrador (almacén)	Empleado a cargo de la administración y control de los artículos en los almacenes.
Financiero	Empleado con preparación en finanzas, encargado de la visualización y control de las ventas, para generar gráficos y control de gatos.

✓ **Una lista de los requerimientos de datos de cada uno de los usuarios identificados (requerimientos de datos por vistas de usuarios).**

Atendiendo a los diferentes tipos de usuarios que harán uso de la base de datos, se procede a definir los requerimientos de datos para cada usuario en particular. La siguiente lista de datos definirá los datos que cada usuario utilizara dentro de la base de datos, así como los datos que los representen sobre la misma:

Lista integrada requerimientos de usuarios

Tipos de usuarios	Requerimiento de datos para el usuario (clasificados por vista de usuario)
Cliente	1- **Datos_personales(**Nombre y apellido, numero teléfono celular, email, sexo, cedula, dirección.**)** 2- **articulos_disponibles(**código, precio de venta, descuento, descripción, nombre**)** 3- **compra(**cedula_vendedor, cedula_cliente, estado, fecha, impuesto, total_monto, serial_comprobante**).** 4- **Mi_cuenta(**numero bancario , cedula , monto**).** 5- **Mis_pedidos(**nombre , cantidad_del_articulo, código_articulo, precio_del_pedido, serial_comprobante**).**
Vendedor	1- **articulo_almacen(**dni_almacen, nombre del articulo, descripción del articulo, precio de venta, cantidad del articulo, Estado del almacén (ubicacion), Municipio del almacén (ubicacion)**).** 2- **Cuales_almacenes_administro(**dni_almacen, cedula_vendedor**).** 3- **Mis_almacenes(**dni, estado , municipio**)** 4- **Mis_clientes(**cedula, nombre, nombre2, apellido1, apellido2, sexo, dirección, serial_comprobante, cedula_vendedor, cedula_cliente, estado, impuesto, fecha, total_monto, numero_bancario**).** 5- **Datos_personales(**Nombre y apellido, Ubicación,

	teléfono, email, sexo y cedula.**)** 6- **Mis_venta(**serial_comprobante, cedula_vendedor, cedula_cliente, estado, impuesto, fecha, total_monto, numero_bancario**).** 7- **Pedido_atencion(**dni_pedido, cedula, nombre, código_articulo, catidad_del_articulo, precio_del_pedido, serial_comprobante**)**
Administrador (almacén)	El administrador del almacén se denota como un usuario especial que tiene todos los permisos(insertar, eliminar, actualizar) para administrar sobre las entidades: 1- Almacen 2- Se_ubica 3- Administra_almacen 4- articulo
Financiero	El usuario **Financiero** se denota como un usuario especial que tiene todos los permisos(insertar, eliminar, actualizar) para administrar sobre las entidades: 1- venta 2- cuenta 3- pedido.

- ✓ **Una lista que integre los requerimientos de todos los usuarios (requerimientos de datos para dar soporte al sistema de información).**

Tipos de datos	Requerimiento de datos para el usuario (clasificados por vista de usuario)	Tipos de datos	Requerimiento de datos para el usuario (clasificados por vista de usuario)
Referentes a la identificación de los clientes y vendedores	**1-** Nombre y apellido, **2-** numero teléfono celular, **3-** email, **4-** sexo, **5-** cedula, **6-** dirección.**)**	**Referentes a los datos de almacén de artículo, articulo y administración sobre los almacenes**	**1-** dni_almacen, **2-** cedula_vendedor **1-** codigo **2-** precio_de_venta **3-** descuento **4-** descripcion **3-** nombre **4-** cantidad **5-** estado **6-** municipio
Referentes a los pedidos, ventas y cuenta	1- serial_comprobante 2- cedula_vendedor, 3- cedula_cliente, 4- estado, 5- impuesto, 6- fecha, 7- total_monto, 8- numero_bancario 9- dni_pedido, 10- cedula, 11- nombre, 12- código_articulo, 13- catidad_del_articulo 14- precio_del_pedido,	**Referentes a las ventas y datos monetarios.**	**1-** *serial_comprobante **2-** cedula_vendedor **3-** cedula_cliente **4-** estado **5-** impuesto **6-** fecha **7-** total_monto **8-** numero_bancario **9-** monto **10-** precio_del_pedido

Los datos recabados acerca de los diferentes usuarios de la base de datos, se realizo sobre una investigación prevista sobre la cantidad de datos necesarios para la base de datos. Es posible que en el transcurso del desarrollo de la base de datos se perciban nuevos datos útiles para la representación del sistema de información.

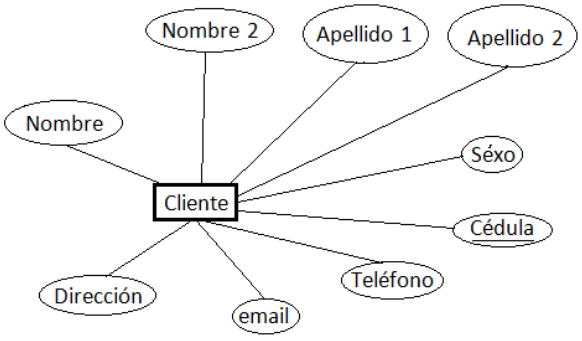
Los requerimientos funcionales de la base de datos y los requerimientos de datos por parte de los usuarios, representan la investigación previa realizada para el desarrollo de la base de datos. La base de datos a desarrollar deberá cumplir las funciones, operaciones (transacciones) que facilita (en parte automatiza) el manejo de la información en la empresa. Las operaciones antes indicadas se sustentan y operan sobre los requerimientos de datos por parte de los usuarios que harán vida de la base de datos.

7.2 **Diseño conceptual de la base de datos (BD)**

Atendiendo a las necesidades de datos, y en referencia a los usuarios de la base de datos, se perciben las siguientes entidades, para el desarrollo de la base de datos:

Entidad	Descripción de entidad
Cliente	Usuario común que realiza las compras en la empresa.
Vendedor	Usuario que facilita el proceso de venta
Almacén	Entidad que almacena los artículos (Las flores).
Articulo	Producto (bien), que la empresa dispone para la venta.
Pedido	Petición del cliente para realizar la compra de x articulo.
Cuenta	Cantidad de dinero que posee el cliente para realizar las distintas compras
Venta	Formulario para la realización de la venta.

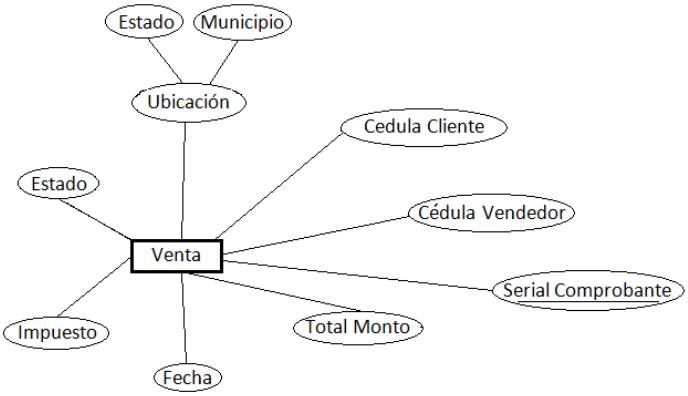
De las entidades antes mencionadas, Se especifican los atributos para cada entidad, las cuales, especifican los datos a los cuales la entidad hace referencia:



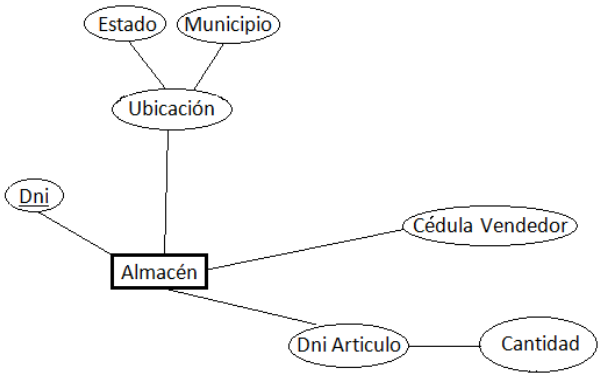
Para la entidad cliente, se elige al atributo “Cedula” Como candidata para ser la clave, dado que la cedula, es única para cada individuo.

Para la entidad “Venta”, se toma como la clave al “serial Comprobante”, ya que este debe ser único para identificar a todas las ventas individualmente.

El atributo Ubicación para la entidad venta, es un atributo compuesto, pues, la ubicación se divide en el estado y municipio.

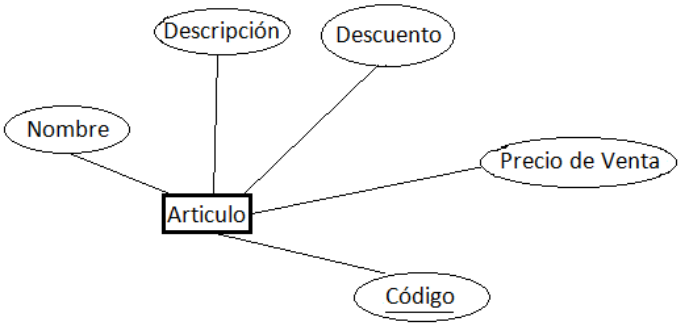


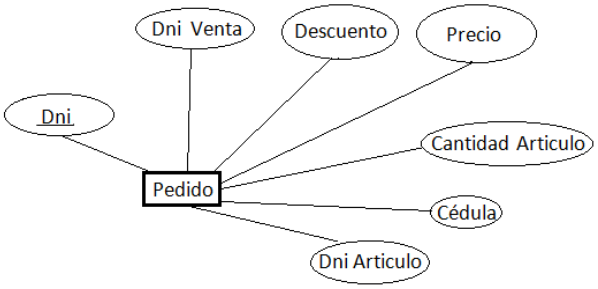
La ubicación es un atributo importante pues, especifica la zona en la cual se registró la venta, los cual es un dato valioso para el registro.



Para los almacenes, se elige el atributo “Dni”, un atributo destinado solamente a identificar al almacén. Ya que se espera pueda haber más de un almacén en la misma ubicación.

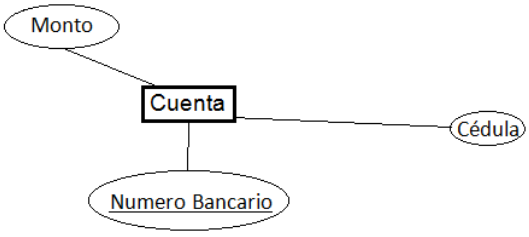
Para la entidad “Articulo” Se elige al atributo “código”, como la clave de la entidad. Pues, cada artículo debe tener su código que lo diferencie.

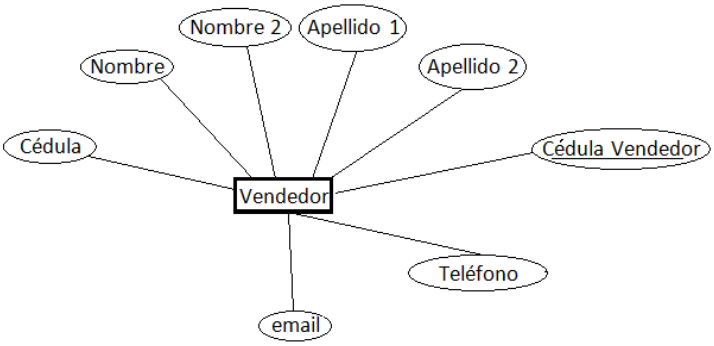




Para la entidad “Pedido” se toma como clave principal al “Dni”, pues dicho dato se define solo para diferencia a un determinado pedido.

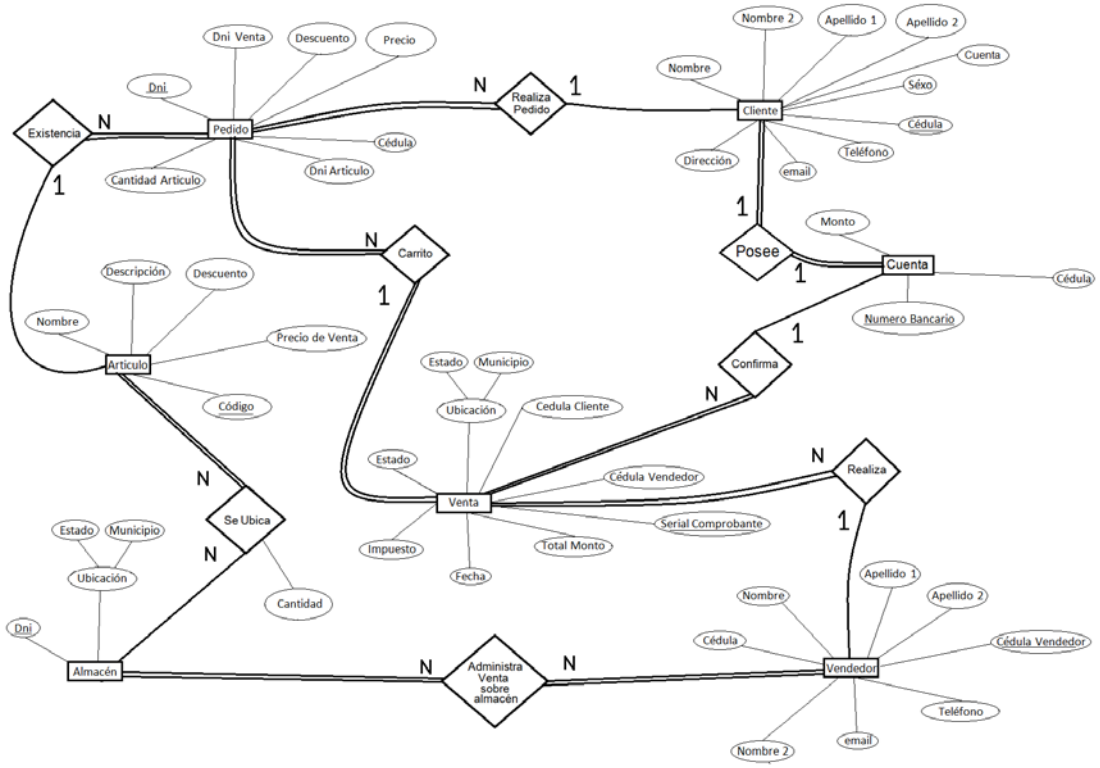
La entidad cuenta se añade con el fin, de completar las entidades, para un sistema de compra.





Establecidos ya las entidades y sus atributos, falta especificar la forma en que se relacionan, esto se realiza acorde a los requerimientos de sistema planteados en el objetivo 7.1, por lo cual, se deben especificar relaciones entre las entidades que permitan realizar las transacciones indicadas para el sistema.

Se diseña el siguiente esquema Entidad-Relacion (ó se propone):



El diagrama entidad-relación sigue las normas del modelo de diagrama de entidad relación explicado por elmasri. En el diagrama se detallan los tipos de relaciones y sus motivos, sustentado en los requisitos del sistema. Las líneas doble de relación recalcan la dependencia obligatoria por parte de un lado de la relación para adquirir validez en su entidad. Por ejemplo, “Almacén” necesita ser operado por uno o varios vendedores para mantener un flujo constante de artículos sobre los almacenes,

y los vendedores necesitan de los almacenes para despachar los artículos vendidos a los clientes, estas representan una relación de dependencia obligaría de parte de ambos lados, por lo que se requiere líneas dobles al ambos lados de la relación.

Se define que cada cliente posee una cuenta ó monedero virtual sobre el cual pueden efectuar las compras en la empresa “Florencia”, dicha cuenta está alojado en el mismo sistema de la base de datos. Además, no es obligatorio que la entidad cliente realice pedidos, es por ello que no tiene una dependencia obligatoria sobre la entidad pedido.

La entidad venta, requiere de relaciones exactas para poder realizar (confirmar) el proceso de venta. Entre ellas encontramos a la relaciones carrito, la cual se encargar de registrar varios pedidos y relacionarlos con una sola venta, además, se debe verificar la cuenta del cliente para confirmar si este puede pagar la cantidad de artículos deseados, Finamente se debe registrar al vendedor y cliente para la venta realizada.

Siguiendo la notación de diagrama entidad, se puede comprender las distintas relaciones y restricciones de cardinalidad entre las distintas entidades. Todas las relaciones, entidades y atributos, se encuentran anexadas al final de trabajo práctico.

8.1. Elección con criterios de Ingeniero de un Sistema de Gestión de Bases de Datos

Existen diferentes sistemas gestores de base de datos en el mercado, lo que hace necesarios elegir de acuerdo a diferentes criterios la conveniencia de algún SGBD para nuestra base de datos. Para poder optar por algún u otro SGBDR se deben primero elegir de una serie de SGBDR del mercado para aplicar criterios de comparación. Es por ello que de acuerdo a su popularidad y destacados rendimientos en bases de datos profesionales, se visualizan las siguientes 3 opciones de SGBDR:

1- **Microsoft Access:** Es un SGBD incluido en el paquete ofimático denominado Microsoft 365, sucesor de Embedded Basic. Dicho SGBD se basa en el paradigma de las tablas relacionales, dicha base de datos es capaz de almacenar datos del tipo Excel, SharePoint, música, videos entre otros. Sin embargo, es un software privado, por lo cual hay que pagar licencias por periodo de usos, ó en dado caso comprar la licencia para el desarrollo de una base de datos empresarial.

2- **MySQL:** es un sistema de gestión de base de datos relacional que cuenta con licencia pública y licencia privada, sin embargo, aunque cuenta con una licencia privada, no sucede que la versión publica este limitada en varios aspectos técnicos, por otro lado, cuenta una variedad de aplicaciones sobre control y administración de la base de datos, con su principal poder de procesamientos sobre las peticiones de cargado de datos, de manera tal que es la base de datos de código abierto más popular del mundo.

3- **PostgreSQL:** es un sistema de gestión de base de datos relacional y de código abierto, publicado bajo licencia PostgresSQL, se destaca en ser 100% gratis, fácil de descargar e instalar, poseyendo además un gran catalogo de tutoriales e instrucciones en internet. Aunque sea gratis posee un amplio conjunto de aplicaciones destinadas al monitoreo y control de la base de datos que rivaliza con los SGBDR privados del mercado.

La base de datos que se va a desarrollar en este trabajo práctico es de escala pequeña, con solo 8 entidades y de relaciones y restricciones básicas, por lo cual, se podría utilizar cualquier SGBDR para realizar dicha base de datos. Sin embargo el SGBD Microsoft Access es de licencia privada, por lo cual, quedaría fuera de nuestro alcance.

MySQL es el SGBDR mas popular, sin embargo está diseñado para una comunidad y sustentadas en varios aspecto por la comunidad. Al ser un SGBDR muy grande y con muchas aplicaciones de control, causa que la instalación sea muy pesada y tediosa al momento de configurar todos los apartados, sin mencionar que se requiere de internet para la instalación.

PostgreSQL es la opción ideal para el desarrollo de base de datos a escala pequeña, dado que presenta todas las herramientas útiles para el manejo de la base de datos, todas sus aplicaciones están optimizadas y comprimidas de manera que la descargar e instalación del programa sea de manera muy sencilla y rápida.

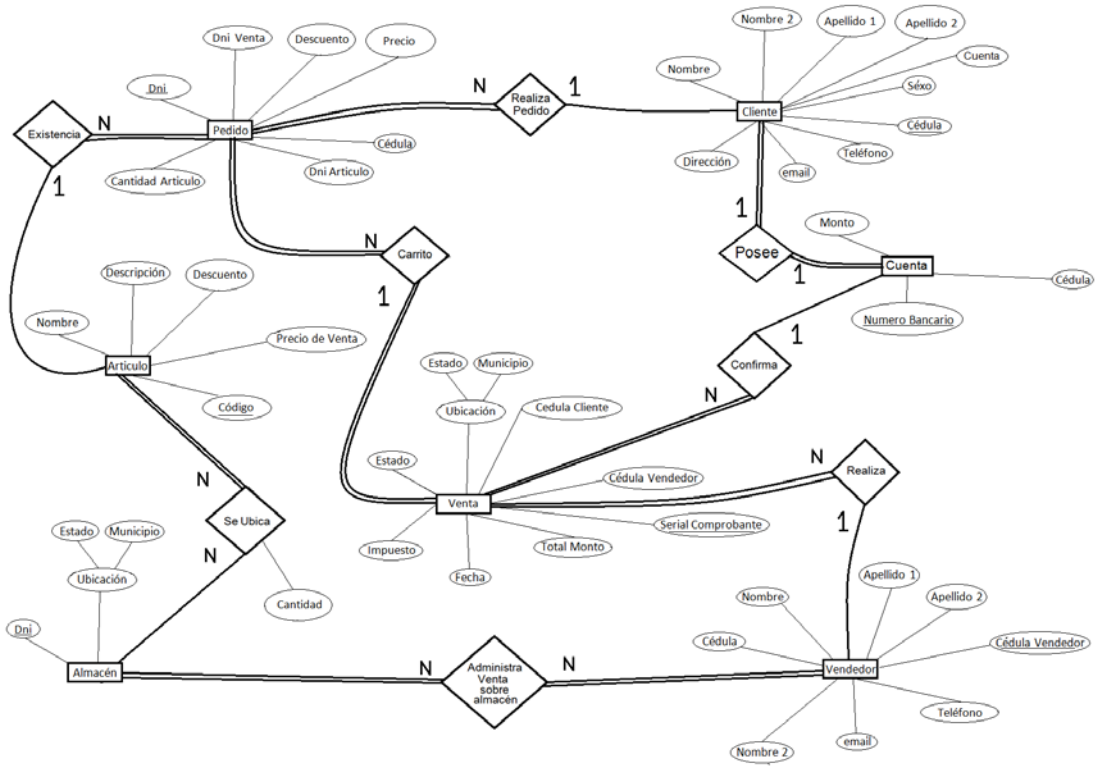
Por todo lo mencionado se elige como SGBDR para el desarrollo de la base de datos de este trabajo practico a PostgreSQL, por su fácil manejo y su amplio conjunto de herramientas visuales para el control de la base de datos.

En el anexo de este trabajo se encuentra documentado las características detalladas de cada sistema de gestión de base de datos mencionados, así como un cuadro comparativo en donde se evidencia de acuerdo a varios criterios la selección del SGBDR.

8.3 Diseño Lógico de la base de datos (¿qué se guarda?)

En esta etapa se procede a trasformar el diseño conceptual de la base de datos, al diseño lógico. Para ello se procede a transformar todas las entidades y relaciones indicada en el diseño conceptual hacia tablas de datos, en las cuales se especifiquen todos los atributos y atributos claves y atributos de foreing keys.

Para ello se toma de referencia el diseño conceptual de la base de datos:



Se sigue el planteamiento de convertir a tablas primero las entidades marcada en el diagrama entidad relación conceptual, luego de acuerdo a las necesidades suscitadas en el proceso se transformaran algunas que otras relaciones a tablas, con el fin de representar correctamente las relaciones entre las entidades.

Las entidades sobre el diagrama entidad relación tiene la siguiente forma:

Cliente							
<u>Cédula</u>	Nombre	Nombre 2	Apellido 1	Apellido 2	Sexo	Dirección	email

Cuenta		
<u>Numero Bancario</u>	Cédula	Monto

Pedido						
<u>Dni</u>	Dni Venta	Descuento	Precio	Cédula	Dni Articulo	Cantidad Articulo

Venta							
<u>Serial Comprobante</u>	Cedula Vendedor	Cedula cliente	Estado	Impuesto	Fecha	Total Monto	Ubicación

Venta- Ubicación			
Serial Comprobante	Ubicación	Estado	Municipio

Para la entidad venta se utilizaron dos tablas con el fin de almacenar de forma correcta el atributo Ubicación, pues, no se pueden almacenar atributos de valor compuesto en las tablas de relaciones.

La entidad “Venta-Ubicación” registra la ubicación sobre la cual se realizó la venta. El atributo ubicación asigna un numero relacionado con el estado y municipio, determinado por el análisis de requisitos de datos. Sin embargo, se observa que el atributo “Ubicación” de la entidad venta queda confuso y no define la dependencia como una forening keys debidamente, esto se revisara en las etapas de normalización de la base de datos.

Vendedor						
Cédula	Nombre	Nombre 2	Apellido 1	Apellido 2	Teléfono	email

Almacén		
Dni	Dni Articulo	Ubicación

Almacén-Ubicación			
Dni Almacén	Ubicación	Estado	Municipio

La entidad almacén se divide en tres tablas, pues posee dos atributos de valores compuestos, lo cual hace necesario, diseñar dos tablas mas, con el fin de guardar de manera correcta los datos.

Para la nueva entidad “Almacén-Articulo” se combinas los atributos “Dni Almacén” y “Dni Articulo” para hacer referencia directa a la cantidad exacta de cada artículo ubicado en cada almacén.

Articulo				
Código	Precio de venta	Descuento	Descripción	Nombre

Ya desarrollados las tablas para las entidades previstas en los requisitos, toca revisar las relaciones, sobre todo aquellas que son de N:M(Muchos a muchos) ó 1:1 (uno a uno).

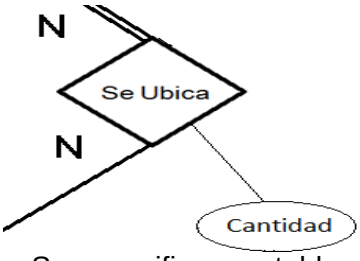
La relación:

Es de muchos a muchos, y en ella se involucran las entidades Vendedor y almacén, pues se prevé que en un mismo almacén varios vendedores puedan hacer uso del mismo. Para evitar filas repetitivas en las cuales un mismo almacén hace referencia a varios vendedores, se debe diseñar una nueva tabla en la cual se especifiquen todos los vendedores asignados a un mismo almacén ó todos los almacenes asignados a un vendedor. Para diseñar esta tabla se tendrá que hacer uso de dos atributos combinados, con el fin de crear un atributo clave que diferencia inequívocamente la relación de un almacén con un vendedor, la tabla quedaría de la siguiente forma:

Administra almacén		
Dni Almacén	Cedula vendedor	Dni

En la cual, para las combinaciones de las claves “Dni Almacén” y “Cedula Vendedor” ubican inequívocamente el “Dni” Dni hace referencia al almacén, que en este caso es administrado por el vendedor de la cedula tal.

La relación:



Relaciona muchos artículos con muchos almacenes, quiere decir que en un almacén puede haber muchos artículos diferentes ó que en muchos almacenes puede haber muchos artículos. Además dicha relación especifica la cantidad del determinado artículo que hay en el almacén.

Se especifica una tabla compuesta por dos claves, las cuales hacen referencia directa hacia la cantidad del determinado artículo que hay en un determinado almacén.

Se ubica		
Dni Almacén	Dni Articulo	Cantidad

La relación 1:1 entre “Cliente” y “Cuenta” se puede representar fácilmente con una foreing keys en el lado del cliente, tal como se muestra en el diagrama entidad para la entidad cliente.

Todas las demás relaciones que quedan, son del tipo 1:N ó N:1, y todas ellas se pueden representar por medio de atributos foreing keys.

Para la entidad venta, se añade el atributo “Numero Bancario” con el objetivo de confirma el saldo de la cuenta del cliente con numero de “Cedula”:

Venta								
Serial Comprobante	Cedula Vendedor	Cedula cliente	Estado	Impuesto	Fecha	Total Monto	Ubicación	Numero Bancario

Para la entidad pedido, se modifica el atributo Dni venta, pues la clave principal de la entidad Venta es el “numero de comprobante”, esto para hacer referencia correcta a la entidad venta.

Pedido						
Dni	Serial Comprobante	Descuento	Precio	Cédula	Dni Articulo	Cantidad Articulo

De esta forma quedan representadas todas las entidades y relaciones del diagrama “entidad-relación”, por medio de las tablas relacionales, queda representada su forma lógica por la cual se vincularan las tablas al momento de pasarlo al plano físico de la base de datos. Sin embargo, todavía faltaría aplicar el método formal para desarrollar las vinculaciones entre las entidades, para corroborar que las relaciones y las entidades estén debidamente planteadas para un óptimo desempeño de Sistema de información. Para ello se procede a desarrollar el modelo lógico de la base de datos a sus respectivas 3 formas normales.

Normalización del sistema lógico de la Base de Datos

El proceso de normalizar una base de datos consiste en formalizar las relaciones planteadas en las tablas relacionales de acuerdo a los criterios que Boyce-Codd definió como las 3 formas normales en la que se pueda representar una base de datos. Cada etapa de la normalización se basa en algún criterio puntual que ayuda definir la base de datos en algún estado consistente:

- **Primera forma normal:** Se centra en la atomización de los datos (atributos). Consiste en que todos los atributos deben ser únicos e individuales (no pueden haber atributos compuestos).
- **Segunda forma normal:** Se centra en la dependencia funcional de los atributos. Consiste en que cada clave haga referencia inequívoca hacia un conjunto de datos.
- **Tercera forma normal:** Se centra en corregir las dependencias transitivas de los datos. Se denota como una etapa muy ligada con la segunda forma normal, pues se basa principalmente en el principio de la dependencia funcional de los datos sobre la clave principal, sin embargo, aquí se verifica (como una etapa final de acabado) que no existan dependencias parciales de datos (ya existente, o generadas por el uso de las etapas anteriores).

Aunque cada etapa comprender un criterio para corregir las tablas relacionales, también es cierto que cada etapa es una oportunidad para corregir (evidenciar) posibles errores de atributos redundantes o atributos faltantes.

Atendiendo a los criterios de cada formas de normalización, se procede a corregir el diseño lógico de las tablas relacionales, con el objetivo de que convertirla a su primera forma normal.

Primera forma normal del diseño lógico de la base de datos

En la primera forma normal se busca representar todas las tablas entidades de acuerdo al criterio de atomización de los datos (Cada atributos deber indivisible ó simple). La mayoría de este proceso se completo al momento de convertir el diagrama entidad relación conceptual al modelo lógicos de tablas relacionales (Pues se tuvieron que eliminar los atributos compuestos). Sin embargo, se procede a listar todas las tablas del modelo lógico para evidenciar posibles errores:

Cliente							
Cédula	Nombre	Nombre 2	Apellido 1	Apellido 2	Sexo	Dirección	email

Cuenta		
Numero Bancario	Cédula	Monto

Venta								
Serial Comprobante	Cedula Vendedor	Cedula cliente	Estado	Impuesto	Fecha	Total Monto	Ubicación	Numero Bancario

Venta- Ubicación			
Serial Comprobante	Ubicación	Estado	Municipio

Pedido						
Dni	Serial Comprobante	Descuento	Precio	Cédula	Código articulo	Cantidad Articulo

Articulo				
Código	Precio de venta	Descuento	Descripción	Nombre

Se ubica		
Dni Almacén	Dni Articulo	Cantidad

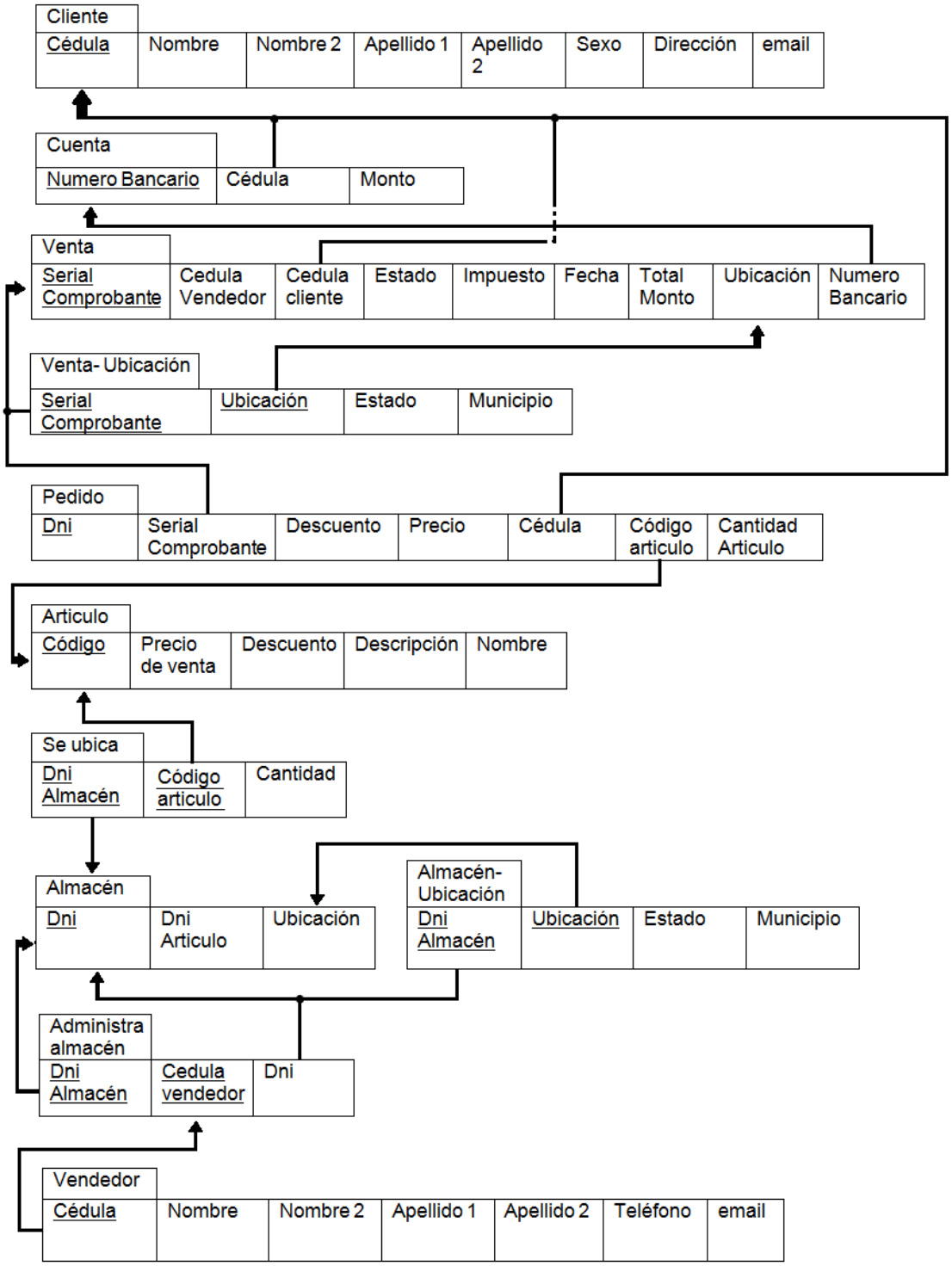
Almacén		
Dni	Dni Articulo	Ubicación

Almacén-Ubicación			
Dni Almacén	Ubicación	Estado	Municipio

Vendedor						
Cédula	Nombre	Nombre 2	Apellido 1	Apellido 2	Teléfono	email

Administra almacén		
Dni Almacén	Cedula vendedor	Dni

Como se observa todos los atributos están en su mínima expresión (todos son individuales), para finalizar y facilitar la siguiente etapa de normalización se especificaran todas las tablas con líneas direccionales de relación:


Cliente
Cédula
Nombre
Nombre 2
Apellido 1
Apellido 2
Sexo
Dirección
email
Cuenta
Numero Bancario
Cédula
Monto
Venta
Serial Comprobante
Cedula Vendedor
Cedula cliente
Estado
Impuesto
Fecha
Total Monto
Ubicación
Numero Bancario
Venta- Ubicación
Serial Comprobante
Ubicación
Estado
Municipio
Pedido
Dni
Serial Comprobante
Descuento
Precio
Cédula
Código articulo
Cantidad Articulo
Articulo
Código
Precio de venta
Descuento
Descripción
Nombre
Se ubica
Dni Almacén
Código articulo
Cantidad
Almacén
Dni
Dni Articulo
Ubicación
Almacén-Ubicación
Dni Almacén
Ubicación
Estado
Municipio
Administra almacén
Dni Almacén
Cedula vendedor
Dni
Vendedor
Cédula
Nombre
Nombre 2
Apellido 1
Apellido 2
Teléfono
email

Segunda forma normal del diseño lógico de la base de datos

La segunda forma normal se basa en el criterio de la dependencia funcional de los atributos sobre la clave principal. Quiere decir que la clave principal de una entidad identificara inequívocamente solo a aquellos atributos de la entidad que tienen relación (Sentido lógico) directa con este, ejemplo: la clave principal “Cedula” de la entidad cliente, identifica un conjunto de atributos inherentes a la identificación del cliente (Nombre, nombre 2, apellido 1 y apellido 2), no tendría sentido que unos de estos atributos fuera el código de un articulo. De la misma manera, para una clave principal combinada de dos atributos, se tiene que esta debería de identificar a un conjunto de atributos acorde con la función de la entidad.

Se comienza el análisis sobre la primera forma normal lograda en el paso anterior. Se observar que la entidad Venta contiene un conjunto de atributos dudosos, extraños, los cuales son “Ubicación”, el atributo ubicación se piensa que debe indicar en qué zona geográfica se realizo la venta, sin embargo, debe haber una forma de saber en qué lugar se hizo venta, y se observa que la entidad venta no obtiene la ubicación de ningún lado. El lugar en donde se realiza la venta debe depender sobre una ubicación específica, Ejemplo: una tienda. Se observa que el único lugar fijo en cuanto a área geográfica se refiere son los almacenes, quiere decir que cada almacén, debería con tener como datos prioritario su ubicación. De esta manera la entidad venta podría hacer referencia hacia los almacenes. Para lo cual se tendría que realizar una nueva relación entre la entidad “Venta” y “Almacén”, dado que las ventas se realizarían sobre las ubicaciones de los almacenes.

Se procede a modificar la tabla (entidad) Venta para que haga referencia al almacén sobre él se realiza la venta:

Venta								
Serial Comprobante	Cedula Vendedor	Cedula cliente	Estado	Impuesto	Fecha	Total Monto	Dni Almacén	Numero Bancario

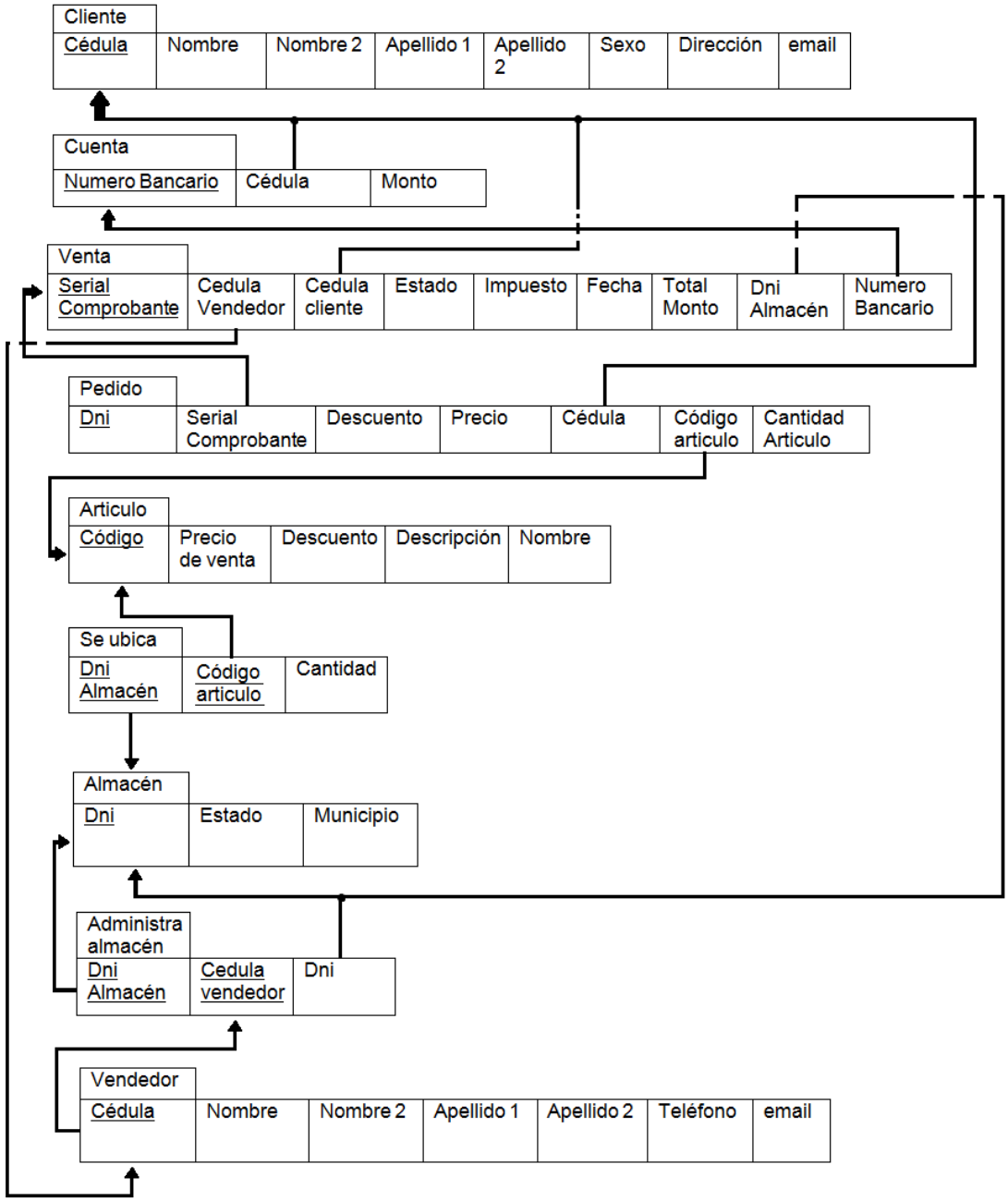
De lo anterior mente comentado se observa, que la entidad almacén debería almacenar como atributos propios su ubicación representado por los atributos “Estado” y “Almacén”:

Almacén			
Dni	Dni Articulo	Estado	Municipio

Además se observa que el atributo “Dni Articulo” contenido en la entidad almacén es redundante, pues la entidad “Se ubica” ya contiene esta información (el almacén y la cantidad de dicho artículo en x almacén), por lo tanto se quita ese atributo de la entidad almacén:

Almacén		
Dni	Estado	Municipio

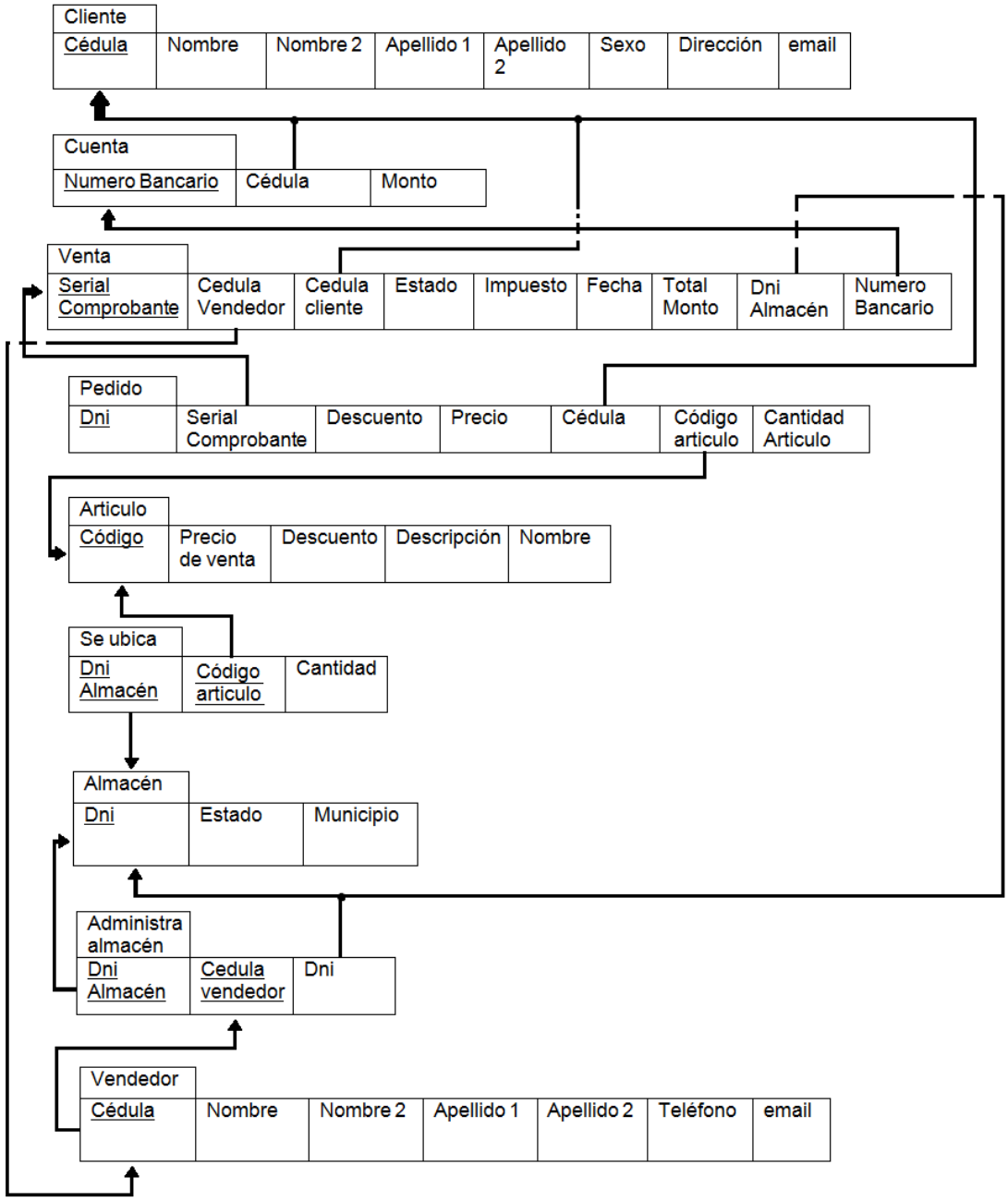
Estas dos entidades son las que presentaban errores en la dependencia de datos, por lo cual se procedío a modificarlas de acuerdo a criterios de especificación del sistema, atendiendo a las premisas de las etapas de normalización. Las tablas entidades relación en su segunda forma normal, quedaría de la siguiente forma:



Se observa que después de la segunda fase de normalización, se logro eliminar la entidad “Venta-Ubicación”, la cual no resultaba eficiente para el sistema de información en desarrollo.

Tercera forma normal del diseño lógico de la base de datos

La tercera forma normal se basa en eliminar todas las dependencias transitivas de datos que queden en las tablas. Sucede que luego de la segunda forma normal, quedan atributos sobre algunas tablas que dependan de otros atributos dentro de las tablas que no son claves principales. Se parte del sistema relacional en la segunda forma normal:



Tras un vistazo, se observa que todos los atributos sobre una entidad son completamente dependientes de la clave principal, esto de acuerdo a los criterios y requerimientos de datos del Sistema de información. Por lo cual se puede afirmar que ya se está en la tercera forma normal.

Puede suceder que a veces no haga falta aplicar el segundo o tercer método formal, esto debido a la similitud de sus criterios. Y como se vio en este caso no hubo falta aplicar el tercer criterio, pues no se hallaron dependencias transitivas en ninguna tabla. Al no hallar ninguna dependencia transitiva, se contacta que los requerimientos de datos y sus relaciones fueron debidamente realizados.

En el anexo de este trabajo se encuentra la documentación del modelo lógico, las entidades, atributos y transacciones.

8.2. Diseño Físico de la base de datos (¿cómo se guardará?)

✓ Traducción del Esquema Lógico Global para el SGBDR, seleccionado presentando el diseño de tablas (entidades) y relaciones así como las reglas de negocio (si éstas últimas no están establecidas en el contenido de las "ESPECIFICACIONES", el estudiante debe presentar situaciones hipotéticas, manteniendo la concordancia de ideas del planteamiento original).

Ya en la etapa del modelo físico, se procede a describir la base de datos en su modelo lógico por medio del lenguaje SQL, apoyado claro en el sistema de gestión de base de datos relacional escogido, que en este caso es PostgreSQL. Se procederá a describir todas las entidades, atributos y relaciones por medio de lenguaje de definición SQL. Teniendo presente para ello la documentación del modelo lógico del anexo.

Antes de desarrollar la base de datos en el SGBDR se procederá a especificar las reglas de negociaciones, las cuales definen las reglas y restricciones de las relaciones entre las distintas entidades de la base de datos, estas restricciones están orientadas a mantener la base de datos en un estado consistente (que todos los datos se relacionan de modo coherente). Es importante especificar las reglas de negociaciones de manera conceptual (conversacional), pues estas serán la guía para posteriormente definirlas en la base de datos, por medio de lenguaje SQL. La siguiente lista muestra un conjunto de normas destinadas a mantener la base de datos en un estado consistente, normas que se adecuan con los requerimientos de datos del sistema:

1- El monto de una compra no debe exceder el saldo de la cuenta del cliente (por ellos la entidad "venta" se relaciona con la entidad "Cuenta").
2- Cuando la entidad estado de la entidad venta se marque como verdadera se deberá descontar la el monto de la venta en la cuenta del cliente.
3- La cantidad de un artículo pedido por un cliente no debe exceder la suma de las cantidades que se dispone de ese artículo en todos los almacenes (para los almacenes en los cuales el vendedor que atiende al cliente tiene acceso).
4- Cuando la entidad estado de la entidad venta se marque como verdadera (se confirma una venta), se debe proceder a restar las cantidades de los artículos comprados de los almacenes, ubicados en la entidad "Se ubica", (solo de los almacenes en los cuales el vendedor que realizó la venta está autorizado).
5- Esta regla ya esta vincula al SGBDR o ya es natural en este tipo de base de datos, pero se recalca que todos los datos foreing keys, en esta base de datos

deben de ser not null, es decir, que no pueden haber referencias vacías, en el caso de rellenar la entidad “administra almacén” no puede suceder que se haga referencia a un almacén ó a un vendedor que no existen todavía en sus respectivas tablas, por lo cual como se menciona una es una entidad que debe estar compuesta por dos atributos, los cuales deben hacer referencias cada una entidad diferentes (dni_almacen a la entidad almacen ‘y’ cedula_vendedor a la entidad ‘Vendedor’). De igual la forma para apertura una venta a un cliente se debe primero a establecer una venta relaciona con el cliente un con un “serial_comprobante” que identifica inequívocamente a una venta, este tipo de apertura de venta se debe marca con el atributo de estado false, para especificar que el cliente todavía tiene que realizar pedidos para proceder y poder cumplir con la venta. Recalcando las restricciones de que un cliente no puede realizar una compra si no se ha realizado pedidos validos que contenga el valor del pedido realizado.

Con estas reglas de negociones definidas se procede entonces a traducir reglas de negocio para el SGBDR, el cual es PostgresSQL. Todas las referencias a entidades, tanto como las reglas de negocio se definen con SQL.

Para la restricciones 1 el cual especifica que el monto de una compra no debe exceder el saldo de la cuenta de cliente, se tiene que para especificar esta restricción se debe de desarrollar un trigger que intercepte el momento en cuando una venta se marca como positiva, en dicho momento se comprueba el monto de la venta con el saldo de la cuenta del cliente, en caso de excederse se debe proceder a negar la venta, y marcar la venta como un estado falso (venta no concretada). para ellos se desarrolla primero la función que comprueba el estado de la venta, seguidamente para comprobar el saldo de la cuenta, dicha función se define en sql como:

```
CREATE FUNCTION public.verificar_monto()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
offsetCon1 integer;
borrar integer;
offsetCon2 integer;
CantidadPer integer;
BEGIN
CantidadPer := (select catidad_del_articulo
    from pedido where serial_Comprobante = NEW.serial_Comprobante limit 1);
offsetCon1 := 0;
offsetCon2 := 0;
if(NEW.estado = TRUE) THEN
    NEW.total_Monto := (select sum(precio_del_Pedido)
    from pedido where serial_Comprobante = NEW.serial_Comprobante);
END IF;
if(NEW.total_Monto = 0) THEN
    NEW.estado := FALSE;
    RAISE NOTICE 'El cliente debe realizar pedidos, para poder realizar la Venta';
ELSIF (NEW.total_Monto > (select monto from Cuenta where Numero_Bancario = NEW.numero_Bancario))
    THEN
    RAISE NOTICE 'Monto insuficiente para realizar la venta';
    NEW.estado := FALSE;
ELSE
    UPDATE Cuenta SET monto = monto - NEW.total_Monto
    WHERE Numero_Bancario = NEW.numero_Bancario;
```

En caso de que el monto se correcto y se descuente la cantidad de la cuenta del cliente, se debe de aprovechar la oportunidad para eliminar los artículos comprados de los almacenes, en los cuales el vendedor de la venta tiene derechos de administración, por lo cual se especifica el siguiente código que ejecuta el trabajo:

```
<<establecer>>
LOOP
   IF((select catidad_del_articulo from pedido offset offsetCon1 limit 1) > 0) then
   if(offsetCon2 = 0) then
       CantidadPer := (select catidad_del_articulo from pedido offset offsetCon1 limit 1);
   END IF;
           if ((select cantidad from se_ubica where
               Dni_Articulo =
                (select codigo_articulo from pedido offset offsetCon1 limit 1)
                and cantidad >0 and dni_almacen = (select dni_almacen
       from Administra_Almacen where cedula_vendedor = (select
       cedula_vendedor from venta where serial_comprobante = NEW.serial_comprobante
       )
       offset offsetCon2 limit 1))>0 )      then

       borrar:=    (select dni_almacen from se_ubica where     Dni_Articulo =
                (select codigo_articulo from pedido offset offsetCon1 limit 1)
                and cantidad >0 and dni_almacen = (select dni_almacen
       from Administra_Almacen where cedula_vendedor = (select
       cedula_vendedor from venta where serial_comprobante = NEW.serial_comprobante
       )
       offset offsetCon2 limit 1));
```

```
57
58                 if((select cantidad from se_ubica where
59                 Dni_Articulo = (select codigo_articulo from pedido offset offsetCon1 limit 1)
60                     and dni_almacen = borrar)
61                    < CantidadPer) then
62                    CantidadPer := CantidadPer - (select cantidad from se_ubica where
63                 Dni_Articulo = (select codigo_articulo from pedido offset offsetCon1 limit 1)
64                     and dni_almacen = borrar);
65                 UPDATE se_ubica set cantidad = 0 where (Dni_Articulo =
66                     (select codigo_articulo from pedido offset offsetCon1 limit 1)
67                      AND Dni_Almacen = borrar);
68                     offsetCon2 := offsetCon2 + 1;
69                 ELSE
70                 UPDATE se_ubica set cantidad = cantidad - CantidadPer where (Dni_Articulo =
71                     (select codigo_articulo from pedido offset offsetCon1 limit 1)
72                       AND Dni_Almacen = borrar);
73                     offsetCon1 := offsetCon1 + 1;
74                     offsetCon2 :=0;
75             --  EXIT establecer;
76                 end if;
77             ELSE
78         --      update se_ubica set cantidad = offsetCon2;
79                 offsetCon2 := 0;
80                 offsetCon1 := offsetCon1 + 1;
81                 RAISE NOTICE 'Esta es el problema';
82             end if;
83     ELSE
84         EXIT establecer;
85     end if;
86 end loop establecer;
87 END IF;
88 RETURN NEW;
89 END
90 $BODY$;
91
92 ALTER FUNCTION public.verificar_monto()
93     OWNER TO postgres;
```

Para que la operación tenga efecto inmediato luego de que se inserte o actualice un registro venta con estado positivo, se debe de especificar un trigger, tal como el que muestra a continuación:

```
5 CREATE TRIGGER verificar_monto
6     BEFORE INSERT OR UPDATE
7     ON public.venta
8     FOR EACH ROW
9     EXECUTE PROCEDURE public.verificar_monto();
```

La tercera restricción específica que un cliente no puede realizar un pedido de un artículo que no existe, en todo caso no se puede realizar un pedido por una cantidad de un determinado artículo mayor de la cantidad que se dispone en todos los almacenes para dicho artículo, para ellos se especifica la siguiente función:

```
CREATE FUNCTION public.control_cantidad_pedido()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
DECLARE
cantidadArt integer;
offsetCon1 integer;
BEGIN
offsetCon1 := 0;
cantidadArt := 0;
    <<Cantidad_almacen_por_vendedor>>
loop
    if((select sum(cantidad)
    from Se_Ubica where Dni_Articulo = NEW.codigo_articulo and dni_almacen =(select dni_almacen
        from Administra_Almacen where cedula_vendedor = (select
        cedula_vendedor from venta where serial_comprobante = NEW.serial_comprobante
        )
        offset offsetCon1 limit 1)) > 0) then
       cantidadArt := cantidadArt + ((select sum(cantidad)
    from Se_Ubica where Dni_Articulo = NEW.codigo_articulo and dni_almacen =(select dni_almacen
        from Administra_Almacen where cedula_vendedor = (select
        cedula_vendedor from venta where serial_comprobante = NEW.serial_comprobante
        )
        offset offsetCon1 limit 1)));
        offsetCon1 := offsetCon1 + 1;
    else
        exit Cantidad_almacen_por_vendedor;
    end if;
end loop Cantidad_almacen_por_vendedor;
insert into probar2(resultado_cantidad,resultado_offset) values (cantidadArt,offsetCon1);
if(NEW.catidad_del_articulo > cantidadArt)
then
    RAISE NOTICE 'Cantidad del articulo pedido\n
                supera las cantidad disponible en almacen';
    NEW.catidad_del_articulo := 0;
    DELETE from pedido where catidad_del_articulo = 0;
ELSIF ((select sum(catidad_del_articulo)
        from pedido where (codigo_articulo = NEW.codigo_articulo
        and serial_Comprobante = NEW.serial_Comprobante)) > cantidadArt) then
        NEW.catidad_del_articulo := 0;
    RAISE NOTICE 'Las cantidades de articulos pedidos excede la capacidad de los almacenes';
END IF;
--insert into probar(resultado) values (cantidadArt);
RETURN NEW;
END
$BODY$;

ALTER FUNCTION public.control_cantidad_pedido()
    OWNER TO postgres;

```

Dicha función controla y mantiene una concordancia con los pedidos realizados, y la cantidad de un determinado artículo en el almacén.

Se añade el siguiente trigger para que llame a la función antes indicadas, en el momento en que se inserten nuevos pedidos:

```
CREATE TRIGGER para_control_cantidad_pedido
    BEFORE INSERT
    ON public.pedido
    FOR EACH ROW
    EXECUTE PROCEDURE public.control_cantidad_pedido();
```

Además de las reglas de negocio antes indicadas, es importante señalar, que no deben existir pedidos en los cuales la cantidad exigida sea nula, es decir aquellos pedidos en los cuales se pide 0 cantidad de un determinado artículo, para ello se especifica la siguiente función:

```
CREATE FUNCTION public.eliminar_pedido()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    DELETE from pedido where catidad_del_articulo = 0;
RETURN NULL;
END
$BODY$;

ALTER FUNCTION public.eliminar_pedido()
    OWNER TO postgres;
```

Dicha función debe tomar efecto luego de que se inserten nuevos pedidos, se verifica que no hallan pedidos de artículos nulos:

```
CREATE TRIGGER eliminar_pedidos
    AFTER INSERT
    ON public.pedido
    FOR EACH ROW
    EXECUTE PROCEDURE public.eliminar_pedido();
```

✓ **Diseño de la representación física, presentando para cada archivo: organización y justificación de la misma, índices, redundancias controladas (de ser necesario mejorar las condiciones de servicio y debe justificarlas), necesidades de espacio de almacenamiento (sumatoria de la capacidad de almacenamiento requerida por cada una de las entidades definitivas).**

El diseño físico de la base de datos comprende la forma en que se guardan los registros en los archivos, como se relacionan entre ellos para facilitar operaciones de carga de datos. Dentro de las organizaciones de registro se deben cumplir los criterios de identificación ó la no repetición de atributos para varios registros, estos según criterios de la entidad claro está.

En este sentido, para el desarrollo de la base de datos en estudio se plantea la organización de registros desordenados (ficheros heap), dicha organización como su nombre lo indica se basa en que los nuevos registros insertado en los archivos se

colocan siempre al final de mismo, esta funcionalidad permite una rápida ejecución sobre las operaciones de inserciones (por la simplicidad operacional de la misma), sin embargo aumenta la duración para los procesos de carga de datos a medida que los registros en el archivo aumenten en cantidad. Dicho método de organización es práctico y funcional para bases de datos pequeñas y casos de estudios, pues si se prevén de 1.000 a 10.000 registros por archivos, este no tendrá gran impacto sobre el tiempo de ejecución en el CPU. Por lo dicho anteriormente se plantea el método “ficheros heap”, como criterio para la organización de los registros en los archivos.

Cabe mencionar que dicho método de organización es el que viene por defecto en la mayoría de los SGBDR, por lo cual, se procederá a identificar su funcionalidad sobre algunas tablas de ejemplos.

Para la entidad “cliente” se tiene que ese cumple con las restricciones de integridad de identificación, pues este posee una única clave principal, la cual es la cedula de cada empleado, pues dicho atributo no se puede repetir para dos empleados diferentes. En dicho sentido no se presentaran redundancias en dicho archivo, pues además se contacta que solo se almacenan valores referentes a los clientes. La tabla cliente con tres registros de cliente se vería de la siguiente forma:

Data Output

	cedula [PK] integer	nombre character varying (10)	nombre2 character varying (10)	apellido1 character varying (15)	apellido2 character varying (15)	sexo character (1)	direccion character varying (50)	email character varying (30)
1	27240851	sam	pamo	Rodriguez	Figueroa	H	Edo Aragua, municipio Sant...	alfredr76@gmail.com
2	45000000	abel	ramirez	Rodriguez	martin	H	Edo Aragua, Los limones	abelRmH@gmail.com
3	13875332	Rut	Norberta	Rodriguez	Figueroa	M	Edo Miranda, altos de pipe	noriberisuru@gmail.com

Como se observar en la tabla ‘cliente’, los registros introducidos se almacenaron en el mismo orden en los cuales fueron insertados. El SGBDR postgreSQL lleva un conteo de la cantidad de registro por archivo, en el caso de la tabla cliente dicho conteo (índice) sirve como soporte para las operaciones de cargado de datos.

Para visualizar el tamaño especifico que la base de datos Postgre otorga por defecto a la entidad cliente se hace uso de la siguiente función:

```
select pg_size_pretty(pg_table_size('cliente'));
```

La función ‘pg_table_size’ da como resultado el tamaño de la entidad cliente, y la función ‘pg_size_pretty’ convierte dicho valor a su equivalente en bytes, resultado para esta operación es la siguiente:

Data Output

	pg_size_pretty text
1	8192 bytes

Lo cual quiere decir que la tabla entidad ‘cliente’ tiene un espacio asignado por defecto del 8,192 kilo bytes. Cada registro puede tener un espacio de 120 bytes, para lo cual se pueden introducir hasta 68 registros en la entidad ‘cliente’ sin la necesidad de aumentar el tamaño del registro.

En caso de que algún archivo requiera más espacio de almacenamiento, el SGBDR está capacitado para ampliar el tamaño del registro de manera predeterminada.

Estimado que en la base de datos actual con un peso de 8192 bytes, con capacidad de hasta 68 registros, crezca hasta 5.000 clientes, se puede calcular el peso correspondiente por medio de la siguiente tabla de tres:

$$68_{registros} \text{----} 8.192_{bytes}$$
$$5.000_{registros} \text{---} x_{bytes}$$

Para un sistema de venta como en el estudio, se estima que en la práctica puedan haber de 100 hasta 5.000 clientes, incluso más, por lo cual se estimaría un aumento del tamaño del registros hasta 602.352,9412 bytes, unos 603 kilo bytes para redondear.

El estudio realizado sobre la relación 'Cliente' es un ejemplo de la manera en como todas las entidad en la base de datos se ordenan y agrupan. El tamaño de cada entidad de acuerdo al tamaño individual de sus elementos se encuentra documentados en el anexo del trabajo práctico.

✓ **Diseño de los mecanismos de seguridad, presentando las vistas de usuarios y reglas de acceso (debe haber concordancia entre estas vistas de usuario y las definidas por el estudiante en 7.1).**

Los mecanismos de seguridad de la base de datos comprenden todo un conjunto de restricciones las cuales se deben de cumplir con el objetivo de que no sucedan transacciones sobre la base de datos que no estuvieron diseñadas al momento de su desarrollo, ejemplo de transacciones indeseadas pueden ser aquellas en las cuales usuarios no autorizados inserten, actualicen o borren registros sin el debido consentimiento del administrador de la base de datos, de la misma forma no se debe permitir que usuarios no autorizados puedan visualizar registros de diferentes tablas.

Para poder implementar mecanismo de seguridad (restricciones), establecer que datos puede ver un usuario sobre la base de datos, primero se debe especificar la función de cada usuario sobre la base de datos, y sobre qué datos este opera. Para ello ya contamos con la lista de usuario y requerimientos del apartado 7.1, por lo cual, se procede utilizarla y en dado caso ampliarla, con el objetivo de que se adecue a la base de datos desarrollada hasta ahora:

Tipos de usuarios	Descripción
Cliente	Usuarios comunes que realizan las compras de artículos en la empresa Florencia. 1-Los clientes deben poder visualizar sus datos personales. 2. deben poder realizar ellos mismo los pedidos (por lo cual deben de tener permiso de inserción sobre la tabla pedido). 3. deben poder inicializar un proceso de compra, por cual debería poder inserta un registro de venta sobre la entidad venta en la cual el estado sea false, y cuando el

	cliente lo crea necesario poder finalizar y procesar la venta cambiando en estado de la venta a true. 4. el usuario cliente debería poder ver todos los datos de la venta completada, a modo de factura. 5. el usuario debe poder visualizar todos los pedidos que ha realizado hasta el momento.
Vendedor	Los diferentes empleados certificados por la empresa ó vendedores terceros que facilitan el proceso de ventas a los usuarios (clientes). 1. Los vendedores deben porder visualizar sus datos personales. 2. Los vendedores deber poder visualizar la tabla que específica sobre que almacenes ellos tienen control de administración. 3. Los vendedores deben poder visualizar los datos acerca de los almacenes, así de cómo cuantos artículos y de que tipos hay por almacén. 4. Los vendedores deben poder visualizar todas aquellas ventas sobre las cuales ellas realizaron. 5. Los vendedores pueden visualizar la lista de pedido realizados por los clientes, de los cuales ellos atendieron y se relacionan por las ventas realizadas o en proceso.
Administrador (almacén)	Empleado a cargo de la administración y control de los artículos en los almacenes. 1. Este empleado debe poseer control de administración especial sobre la entidades almacen, se_ubica y administra_almacen. Pues dicho empleado estará a cargo del suministro de los almacenes, del mismo modo tendrán control para delegar que vendedores pueden usar tales almacenes.
Financiero	Empleado con preparación en finanzas, encargado de la visualización y control de las ventas, para generar gráficos y control de gastos. 1. Dichos empleados deben poseer control especial sobre la entidad venta, pedido y cuenta, pues son todas las entidades que tienen datos en referencia a las cantidades monetarias, sin embargo es necesario restringir las operaciones de actualización y borrado, pues todos los registros almacenados se deben de guardar como datos históricos.

Los tipos de usuarios se definen como roles dentro de la base de datos. Para los nuevos usuarios de la base de datos, se le debe asignar un rol, y de acuerdo al rol que tenga, se le asignaran todas las restricciones que posea dicho rol. En dicho sentido se crean los roles y se procede a definir todas las restricciones por roles, luego dichas restricciones se le asigna a los usuarios por medio de los roles asignados.

Los roles clientes y vendedores son los roles que mas poseen restricción, pues se denotan como tipos de usuarios que mas harán uso de la base de datos, la mayor parte de los usuarios de la base de datos están conformado por los vendedores y clientes. A cada usuario de le asigna un ‘usuario’ y una’ contraseña’ con los cuales podrán acceder a la base de datos, la primera restricciones es que los usuarios de los roles clientes y vendedores deben ser sus respectivas cedulas, y la contraseña puede ser cualquiera, la razón por la cual el usuario debe de ser la cedula, es porque la cedula por cada usuario es única, y ellas permite identificar y corroborar restricciones para cada usuario individual. En el caso de los clientes de debe especificar que un usuario identificado como cliente no pueda acceder a los datos personales de otros clientes, sino mas bien que la vista de usuarios para cada cliente individual atienda a la restricciones de que cada usuario debe tener acceso solo sobre sus datos, sus pedidos y sus compras, así como de su cuenta (bancaria). Del mismo modo para los vendedores.

Los roles almacenista y financiero son roles que poseen menos restricciones pues se denotan como usuario de acceso especial a la base de datos, ellos atienden a las entidades que fueron señaladas en los requerimientos de datos previos. Dichos roles como es el caso del rol financiero puede tener acceso sobre todas los registro de la entidad venta y pedido así como de las cuentas bancarias. De esta misma forma el usuario almacenista tiene control de actualización inserción y borrado de la entidad se_ubica, pero este no debe poder eliminar almacenes de la entidad almacén, pues estaría eliminando la referencia hacia almacenes reales.

Seguidamente se procede a especificar las restricciones de vista y operacionales sobre las tablas para los roles. Comenzando para el rol clientes.

El rol cliente comprende las siguientes vistas:

clientes

Type	Name	Database
Table	public.pedido	Florencia Base de datos TP-311
View	public.articulos_disponibles	Florencia Base de datos TP-311
View	public.compra	Florencia Base de datos TP-311
View	public.datos_personales	Florencia Base de datos TP-311
View	public.mi_cuenta	Florencia Base de datos TP-311
View	public.mis_pedidos	Florencia Base de datos TP-311

Las cuales comprenden las vistas permitidas y necesarias para el rol cliente. Para comprender mejor cuales datos se permiten en dichas vitas se muestran sus definiciones en SQL.

```sql
CREATE OR REPLACE VIEW public.articulos_disponibles
 AS
 SELECT articulo.codigo,
    articulo.precio_de_venta,
    articulo.descuento,
    articulo.descripcion,
    articulo.nombre
   FROM articulo;

ALTER TABLE public.articulos_disponibles
    OWNER TO postgres;

GRANT SELECT ON TABLE public.articulos_disponibles TO clientes;
GRANT ALL ON TABLE public.articulos_disponibles TO postgres;
```

```sql
CREATE OR REPLACE VIEW public.compra
 AS
 SELECT venta.cedula_vendedor AS serial_factura,
    venta.cedula_cliente,
    venta.estado,
    venta.fecha,
    venta.impuesto,
    venta.total_monto,
    venta.serial_comprobante
   FROM venta
  WHERE (venta.cedula_vendedor = (( SELECT USER AS "user"))::integer);

ALTER TABLE public.compra
    OWNER TO postgres;

GRANT SELECT ON TABLE public.compra TO clientes;
GRANT ALL ON TABLE public.compra TO postgres;
```

```sql
CREATE OR REPLACE VIEW public.mi_cuenta
 AS
 SELECT cuenta.numero_bancario,
    cuenta.cedula,
    cuenta.monto
   FROM cuenta
  WHERE (cuenta.cedula = (( SELECT USER AS "user"))::integer);

ALTER TABLE public.mi_cuenta
    OWNER TO postgres;

GRANT SELECT ON TABLE public.mi_cuenta TO clientes;
GRANT ALL ON TABLE public.mi_cuenta TO postgres;
```

```
CREATE OR REPLACE VIEW public.mis_pedidos
 AS
 SELECT p1.cedula AS serial_factura,
    a1.nombre,
    p1.catidad_del_articulo,
    p1.codigo_articulo,
    p1.precio_del_pedido,
    p1.serial_comprobante
   FROM pedido p1,
    articulo a1
  WHERE ((p1.cedula = (( SELECT CURRENT_USER AS "current_user"))::integer)
          AND ((a1.nombre)::text = (( SELECT articulo.nombre
           FROM articulo
          WHERE (articulo.codigo = p1.codigo_articulo)))::text));

ALTER TABLE public.mis_pedidos
    OWNER TO postgres;

GRANT SELECT ON TABLE public.mis_pedidos TO clientes;
GRANT ALL ON TABLE public.mis_pedidos TO postgres;
```

```
CREATE OR REPLACE VIEW public.datos_personales
 AS
 SELECT cliente.cedula,
    cliente.nombre,
    cliente.nombre2,
    cliente.apellido1,
    cliente.apellido2,
    cliente.sexo,
    cliente.direccion,
    cliente.email
   FROM cliente
  WHERE (cliente.cedula = (( SELECT CURRENT_USER AS "current_user"))::integer);

ALTER TABLE public.datos_personales
    OWNER TO postgres;

GRANT SELECT ON TABLE public.datos_personales TO clientes;
GRANT ALL ON TABLE public.datos_personales TO postgres;
```

Para finalizar la entidad cliente tiene permiso de:

```
GRANT DELETE ON TABLE public.pedido TO clientes;
GRANT INSERT(codigo_articulo) ON public.pedido TO clientes;
GRANT INSERT(catidad_del_articulo) ON public.pedido TO clientes;
```

Las cuales le otorgan privilegios para insertar el código y la cantidad del artículo que desee comprar. De igual forma le permite eliminar pedidos no deseados, siempre y cuando sean sus pedidos.

Para el rol vendedores se especifican la siguiente lista de vistas de usuarios:

Type	Name	Database
View	public.articulo_almacen	Florencia Base de datos TP-311
View	public.cuales_almacenes_administro	Florencia Base de datos TP-311
View	public.mis_almacenes	Florencia Base de datos TP-311
View	public.mis_clientes	Florencia Base de datos TP-311
View	public.mis_datos_vendedor	Florencia Base de datos TP-311
View	public.mis_ventas	Florencia Base de datos TP-311
View	public.pedido_atencion	Florencia Base de datos TP-311

Las cuales comprenden las vistas permitidas y necesarias para el rol vendedores. Para comprender mejor cuales datos se permiten en dichas vitas se muestran sus definiciones en SQL.

```
CREATE OR REPLACE VIEW public.articulo_almacen
 AS
 SELECT seub.dni_almacen,
    art.nombre,
    art.descripcion,
    art.precio_de_venta,
    seub.cantidad,
    a1.estado,
    a1.municipio
   FROM articulo art,
    se_ubica seub,
    almacen a1,
    administra_almacen aa1
  WHERE ((art.codigo = seub.dni_articulo) AND (seub.dni_almacen = a1.dni)
   AND (a1.dni = aa1.dni_almacen) AND (aa1.cedula_vendedor = (CURRENT_USER)::integer));
ALTER TABLE public.articulo_almacen
    OWNER TO postgres;

GRANT SELECT ON TABLE public.articulo_almacen TO vendedores;
GRANT ALL ON TABLE public.articulo_almacen TO postgres;
```

```
5  CREATE OR REPLACE VIEW public.cuales_almacenes_administro
6   AS
7   SELECT administra_almacen.dni_almacen,
8      administra_almacen.cedula_vendedor
9     FROM administra_almacen
10    WHERE (administra_almacen.cedula_vendedor = (CURRENT_USER)::integer);
11
12 ALTER TABLE public.cuales_almacenes_administro
13     OWNER TO postgres;
14
15 GRANT SELECT ON TABLE public.cuales_almacenes_administro TO vendedores;
16 GRANT ALL ON TABLE public.cuales_almacenes_administro TO postgres;
```

```
CREATE OR REPLACE VIEW public.mis_almacenes
 AS
 SELECT a1.dni,
    a1.estado,
    a1.municipio
   FROM almacen a1,
    administra_almacen aa1
  WHERE ((a1.dni = aa1.dni_almacen) AND (aa1.cedula_vendedor = (CURRENT_USER)::integer));

ALTER TABLE public.mis_almacenes
    OWNER TO postgres;

GRANT SELECT ON TABLE public.mis_almacenes TO vendedores;
GRANT ALL ON TABLE public.mis_almacenes TO postgres;
```

```
CREATE OR REPLACE VIEW public.mis_clientes
 AS
 SELECT cli.cedula,
    cli.nombre,
    cli.nombre2,
    cli.apellido1,
    cli.apellido2,
    cli.sexo,
    cli.direccion,
    cli.email,
    vent.serial_comprobante,
    vent.cedula_vendedor,
    vent.cedula_cliente,
    vent.estado,
    vent.impuesto,
    vent.fecha,
    vent.total_monto,
    vent.numero_bancario
   FROM cliente cli,
    venta vent
  WHERE ((cli.cedula = vent.cedula_cliente)
         AND (vent.cedula_vendedor = (CURRENT_USER)::integer));

ALTER TABLE public.mis_clientes
    OWNER TO postgres;

GRANT SELECT ON TABLE public.mis_clientes TO vendedores;
GRANT ALL ON TABLE public.mis_clientes TO postgres;
```

```sql
CREATE OR REPLACE VIEW public.mis_datos_vendedor
 AS
 SELECT vendedor.cedula,
    vendedor.nombre,
    vendedor.nombre2,
    vendedor.apellido1,
    vendedor.apellido2,
    vendedor.telefono,
    vendedor.email
   FROM vendedor
  WHERE (vendedor.cedula = ((CURRENT_USER)::integer)::numeric);

ALTER TABLE public.mis_datos_vendedor
    OWNER TO postgres;

GRANT SELECT ON TABLE public.mis_datos_vendedor TO vendedores;
GRANT ALL ON TABLE public.mis_datos_vendedor TO postgres;
```

```sql
CREATE OR REPLACE VIEW public.mis_ventas
 AS
 SELECT vent.serial_comprobante,
    vent.cedula_vendedor,
    vent.cedula_cliente,
    vent.estado,
    vent.impuesto,
    vent.fecha,
    vent.total_monto,
    vent.numero_bancario
   FROM venta vent
  WHERE (vent.cedula_vendedor = (CURRENT_USER)::integer);

ALTER TABLE public.mis_ventas
    OWNER TO postgres;

GRANT SELECT ON TABLE public.mis_ventas TO vendedores;
GRANT ALL ON TABLE public.mis_ventas TO postgres;
```

```sql
CREATE OR REPLACE VIEW public.pedido_atencion
 AS
 SELECT ped.dni,
    ped.cedula,
    art.nombre,
    ped.codigo_articulo,
    ped.catidad_del_articulo,
    ped.precio_del_pedido,
    ped.serial_comprobante
   FROM pedido ped,
    articulo art,
    venta vent
  WHERE ((art.codigo = ped.codigo_articulo) AND (ped.cedula = vent.cedula_cliente) AND
         (ped.serial_comprobante = vent.serial_comprobante) AND (vent.cedula_vendedor = (CURRENT_USER)::integer));
ALTER TABLE public.pedido_atencion
    OWNER TO postgres;

GRANT SELECT ON TABLE public.pedido_atencion TO vendedores;
GRANT ALL ON TABLE public.pedido_atencion TO postgres;
```

El rol vendedores comprendes solo permisos de vistas, pues este solo de encarga de atender al cliente, mas no tiene autoridad para modificar los pedidos de los clientes, ni tiene autoridad para modificar la tabla venta, tampoco los valores referidos a los almacenes, dado que los almacenes están bajo la administración de los almacenistas.

El rol almacenista tiene control sobre las siguientes tablas (relaciones):

almacenista

Type	Name	Database
Table	public.administra_almacen	Florencia Base de datos TP-311
Table	public.almacen	Florencia Base de datos TP-311
Table	public.articulo	Florencia Base de datos TP-311
Table	public.se_ubica	Florencia Base de datos TP-311
Table	public.vendedor	Florencia Base de datos TP-311

Para comprender mejor el alcance de su control sobre estas relaciones se especifican su autorización sobre estas en sql:

```
GRANT UPDATE, INSERT, SELECT ON TABLE public.administra_almacen TO almacenista;
GRANT SELECT ON TABLE public.almacen TO almacenista;
GRANT INSERT, SELECT ON TABLE public.articulo TO almacenista;
GRANT INSERT, SELECT ON TABLE public.se_ubica TO almacenista;
GRANT SELECT ON TABLE public.vendedor TO almacenista;
```

El rol contador (financista) tiene control sobre las siguientes tablas (relaciones):

contador

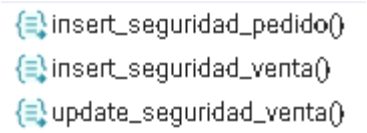
Type	Name	Database
Table	public.cuenta	Florencia Base de datos TP-311
Table	public.pedido	Florencia Base de datos TP-311
Table	public.venta	Florencia Base de datos TP-311

Para comprender mejor el alcance de su control sobre estas relaciones se especifican su autorización sobre estas en sql:

```
GRANT SELECT ON TABLE public.venta TO contador;
GRANT SELECT ON TABLE public.cuenta TO contador;
GRANT SELECT ON TABLE public.pedido TO contador;
```

Todas las restricciones de vista, y de acceso a las relaciones comprenden una parte del mecanismo de seguridad de la base de datos, orientada a la integridad de datos y control sobre el acceso a las tablas.

En complementación a los accesos de seguridad especificados, se diseñan los siguientes trigger con el objetivo, de evitar entradas ilegales por parte de usuarios mal intencionados:



Dichas restricciones controlan que los usuarios (clientes) no puedan realizar pedidos en los cuales su número de cedula está mal introducido, además controla que los usuarios (clientes) no puedan inicializar más de dos ventas al mismo tiempo, seguido de que los usuarios (clientes) no pueden actualizar ventas de las cuales no son propietarios (es decir venta de otros clientes). El código de estas restricciones es las siguientes:

```
CREATE FUNCTION public.insert_seguridad_pedido()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
if(current_role = clientes) then
    if(new.serial_comprobante = null) then
        new.serial_comprobante = (select serial_comprobante from venta where
                                  cedula = current_user and estado = false);
        exit insert_seguridad_pedido;
    end if;
    if(new.serial_comprobante = null) then
        raise notice 'Cliente no ha aperturado una venta, no puede hacer pedido';
        delete from pedido where serial_comprobante = null;
        exit insert_seguridad_pedido;
    end if;
    if(new.cedula = null) then
        new.cedula = cast(current_user as integer);
    end if;
end if;
RETURN NEW;
END
$BODY$;

ALTER FUNCTION public.insert_seguridad_pedido()
    OWNER TO postgres;
```

```
CREATE FUNCTION public.insert_seguridad_venta()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
if(new.cedula_cliente = null) then
    if(current_role = clientes) then
        raise notice 'Comprobacion automatica de cedula cliente';
        new.cedula_cliente = cast(current_user as integer);
        exit insert_seguridad_venta;
    elsif(current_user = postgres) then
        delete from venta where cedula_cliente = null;
        exit insert_seguridad_venta;
    end if;
end if;
if(new.cedula_vendedor = null) then
    new.cedula_vendedor = (select cedula from vendedor limit 1);
    if(new.cedula_vededor = null) then
        delete from venta where cedula_vendedor = null;
        exit insert_seguridad_venta;
    end if;
end if;
if(new.serial_comprobante = null) then
    new.serial_comprobante = 100000000000 + new.cedula_cliente +
    (select count(*) from venta where cedula_cliente
    = current_user and estado = true);
end if;
if(new.numero_bancario = null) then
    new.numero_bancario = (select Numero_Bancario from cuenta
                           where cedula = new.cedula);
    if(new.numero_bancario = null) then
        raise notice 'El cliente no posee cuenta bancaria, no puede realiza compras';
        delete from venta where numero_bancario = null;
        exit insert_seguridad_venta;
    end if;
end if;
if((select count(*) from venta where cedula_cliente
    = current_user and estado = false) > 1) then
    raise notice 'No se puede realizar dos compras al mismo tiempo,
    se elimnará la ultima';
    delete from venta where serial_comprobante = new.serial_comprobante;
end if;
RETURN NEW;
END
$BODY$;

ALTER FUNCTION public.insert_seguridad_venta()
    OWNER TO postgres;
```

```
CREATE FUNCTION public.update_seguridad_venta()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
if(current_role = clientes) then
    update venta set estado = false where total_monto = null;

end if;

RETURN NEW;
END
$BODY$;

ALTER FUNCTION public.update_seguridad_venta()
    OWNER TO postgres;
```

Seguidamente se especifican los trigger para estas tres funciones:

```
CREATE TRIGGER trigger_insert_seguridad_venta
    AFTER INSERT
    ON public.venta
    FOR EACH ROW
    EXECUTE PROCEDURE public.insert_seguridad_venta();
```

```
CREATE TRIGGER trigger_insert_seguridad_pedido
    AFTER INSERT
    ON public.pedido
    FOR EACH ROW
    EXECUTE PROCEDURE public.insert_seguridad_pedido();
```

```
CREATE TRIGGER trigger_insert_seguridad_venta
    AFTER INSERT
    ON public.venta
    FOR EACH ROW
    EXECUTE PROCEDURE public.update_seguridad_venta();
```

- ✓ Modelo Físico:

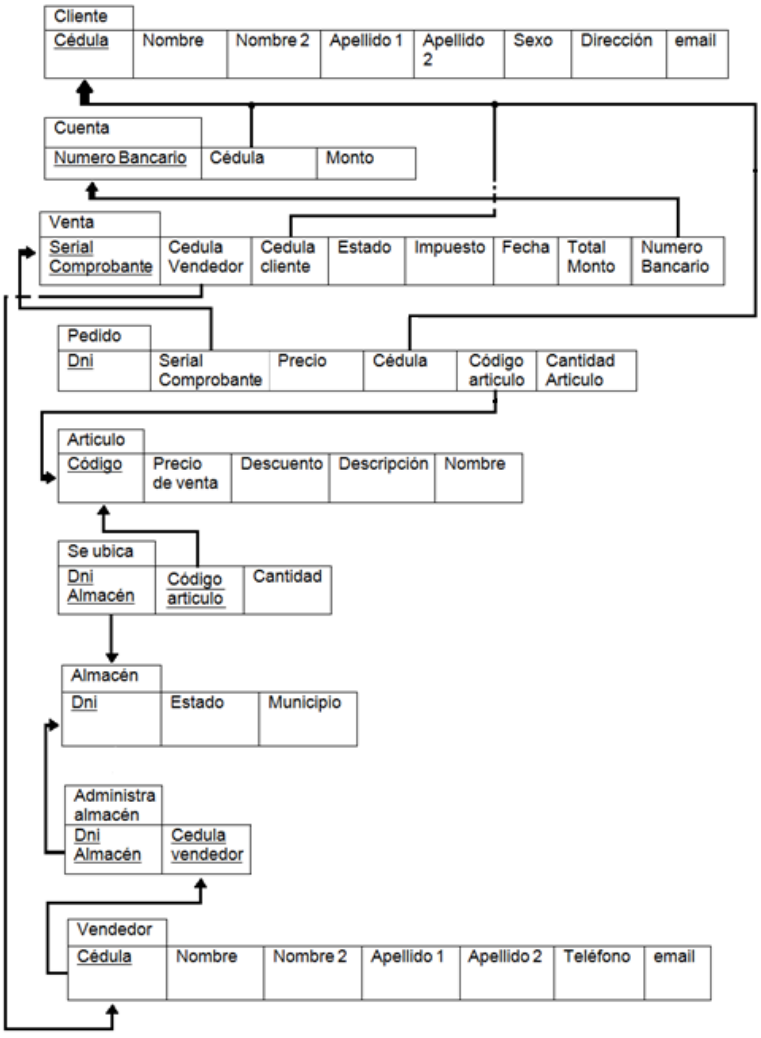
- Gráfico en 3 FN contentivo de entidades, atributos, claves, relaciones (definitivos)
- Construido con SGBD seleccionado y adecuado a las características del mismo

Ya en la parte del modelo físico se debe proceder a desarrollar la base de datos en el SGBDR, para dicho desarrollo se debe apoyar sobre las especificaciones y requerimientos de datos especificados en el desarrollo del trabajo, especial mente sobre el diseño y la formalización de las relaciones en su etapa lógica, la tercera forma normal lograda en el diseño lógico representa el esquema relación y todos los atributos útiles que dispondrá la base de datos a desarrollar.

En el transcurso de la definición lógica hasta las implementaciones de seguridad, vistas de usuarios y requerimientos de espacio como de ordenamientos de las entidades, se han notado algunos atributos redundantes, de los cuales se podrían prescindir, entre ellos se encontramos:

1- Atributo ‘descuento’ de la relación pedido, dicho atributo ya se encuentra en la relación articulo, y no seria optimo que dicho atributo se repitiera en dos tablas, pues para su fin, que es el de calcular, el precio del los pedidos se podría hacer referencia simplemente al atributo ‘descuento’ de la relación articulo.
2- Atributo ‘Dni Almacen’ de la relacion venta, dicho atributo termina siendo ambiguo e ineficaz, pues en las funcionalidades del sistema de base de datos, se espera que un cliente pueda realizar pedido para artículos que se encuentren en diferentes almacenes. Es decir un cliente podría comprar todas las unidades de un determinado artículo de todos los almacenes en una sola compra.
3- Atributo ‘dni’ de la relación administra_almacen, la entidad administra_almacen relaciones a los vendedores con los respectivos almacenes sobre los cuales pueden operar, para ello basta solamentes con los atributos ‘cedula_vendedor’ y ‘dni_almacen’ por la conjunciones de esos dos atributos genera claves únicas que no se repiten en toda la tabla, por lo cual basta para especificar las relaciones de administración de los vendedores sobre las tablas almacen.

Dichos atributos se eliminan de sus respectivas entidades, y quedaría la tercera forma normal definitiva la cual serviría de guía para el desarrollo de la base de datos en el SGBDR:



El dominio de los atributos esta especificado en la documentación de los atributos, tanto en su documentación conceptual como en su documentación física, su tamaño en bytes y su función dentro de las entidades y del sistema en conjunto de la base de datos. Es por ello que aquí se procederá a definir todas las tablas (entidades o relaciones) en lenguaje SQL, el cual corresponde al estándar SQL y es el que soporta el sistema de gestión de base de datos posgretSQL.

Para la entidad cliente se tiene la siguiente definición:

```
CREATE TABLE public.cliente
(
    cedula integer NOT NULL,
    nombre character varying(10) COLLATE pg_catalog."default" NOT NULL,
    nombre2 character varying(10) COLLATE pg_catalog."default" NOT NULL,
    apellido1 character varying(15) COLLATE pg_catalog."default" NOT NULL,
    apellido2 character varying(15) COLLATE pg_catalog."default" NOT NULL,
    sexo character(1) COLLATE pg_catalog."default",
    direccion character varying(50) COLLATE pg_catalog."default",
    email character varying(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT cliente_pkey PRIMARY KEY (cedula),
    CONSTRAINT cliente_cedula_check CHECK (cedula > 5000000 AND cedula < 50000000)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.cliente
    OWNER to postgres;
```

Para la entidad pedido se tiene la siguiente definición:

```sql
CREATE TABLE public.pedido
(
    dni integer NOT NULL DEFAULT nextval('pedido_dni_seq'::regclass),
    serial_comprobante numeric(12,0),
    precio_del_pedido numeric(12,2),
    cedula integer NOT NULL,
    codigo_articulo numeric(7,0) NOT NULL,
    catidad_del_articulo integer NOT NULL,
    CONSTRAINT pedido_pkey PRIMARY KEY (dni),
    CONSTRAINT pedido_cedula_fkey FOREIGN KEY (cedula)
        REFERENCES public.cliente (cedula) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT pedido_codigo_articulo_fkey FOREIGN KEY (codigo_articulo)
        REFERENCES public.articulo (codigo) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.pedido
    OWNER to postgres;
```

Para la entidad venta se tiene la siguiente definición:

```
CREATE TABLE public.venta
(
    serial_comprobante numeric(12,0) NOT NULL,
    cedula_vendedor integer NOT NULL,
    cedula_cliente integer,
    estado boolean DEFAULT false,
    impuesto integer DEFAULT 16,
    fecha timestamp without time zone DEFAULT now(),
    total_monto numeric(12,2) DEFAULT 0,
    numero_bancario numeric(20,0),
    CONSTRAINT venta_pkey PRIMARY KEY (serial_comprobante),
    CONSTRAINT venta_cedula_cliente_fkey FOREIGN KEY (cedula_cliente)
        REFERENCES public.cliente (cedula) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT venta_cedula_vendedor_fkey FOREIGN KEY (cedula_vendedor)
        REFERENCES public.vendedor (cedula) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT venta_numero_bancario_fkey FOREIGN KEY (numero_bancario)
        REFERENCES public.cuenta (numero_bancario) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT venta_cedula_vendedor_check CHECK
           (cedula_vendedor > 5000000 AND cedula_vendedor < 50000000),
    CONSTRAINT venta_cedula_cliente_check
            CHECK (cedula_cliente > 5000000 AND cedula_cliente < 50000000)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.venta
    OWNER to postgres;
```

Para la entidad cuenta se tiene la siguiente definición:

```
CREATE TABLE public.cuenta
(
    numero_bancario numeric(20,0) NOT NULL,
    cedula integer NOT NULL,
    monto numeric(12,2),
    CONSTRAINT cuenta_llave_principal PRIMARY KEY (numero_bancario),
    CONSTRAINT cuenta_cedula_key UNIQUE (cedula),
    CONSTRAINT cuenta_cedula_fkey FOREIGN KEY (cedula)
        REFERENCES public.cliente (cedula) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE,
    CONSTRAINT cuenta_numero_bancario_check CHECK (numero_bancario >=
  '10000000000000000000'::numeric AND numero_bancario
                                  <= '99999999999999999999'::numeric)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;
```

Para la entidad artículo se tiene la siguiente definición:

```
CREATE TABLE public.articulo
(
    codigo numeric(7,0) NOT NULL,
    precio_de_venta numeric(12,2) NOT NULL,
    descuento numeric(12,2) NOT NULL,
    descripcion text COLLATE pg_catalog."default",
    nombre character varying(20) COLLATE pg_catalog."default",
    CONSTRAINT articulo_pkey PRIMARY KEY (codigo),
    CONSTRAINT articulo_codigo_check CHECK
        (codigo >= 1000000::numeric AND codigo <= 9999999::numeric)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.articulo
    OWNER to postgres;
```

Para la entidad almacén se tiene la siguiente definición:

```
CREATE TABLE public.almacen
(
    dni integer NOT NULL DEFAULT nextval('almacen_dni_seq'::regclass),
    estado character varying(12) COLLATE pg_catalog."default" NOT NULL,
    municipio character varying(20) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT almacen_pkey PRIMARY KEY (dni),
    CONSTRAINT almacen_estado_check CHECK (estado::text = 'Aragua'::text OR estado::text
    = 'miranda'::text OR estado::text = 'Bolivar'::text OR estado::text = 'merida'::text
      OR estado::text = 'trujillo'::text OR estado::text = 'portuguesa'::text
      OR estado::text = 'monagas'::text OR estado::text = 'zulia'::text OR estado::text =
                                                          'distrito Capital'::text)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.almacen
    OWNER to postgres;
```

Para la entidad se_ubica se tiene la siguiente definición:

```
CREATE TABLE public.se_ubica
(
    dni_almacen integer NOT NULL DEFAULT nextval('se_ubica_dni_almacen_seq'::regclass),
    dni_articulo numeric(7,0) NOT NULL,
    cantidad integer NOT NULL,
    CONSTRAINT se_ubica_pkey PRIMARY KEY (dni_almacen, dni_articulo),
    CONSTRAINT se_ubica_cantidad_check CHECK (cantidad >= 0)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.se_ubica
    OWNER to postgres;
```

Para la entidad administra_almacen se tiene la siguiente definición:

```
CREATE TABLE public.administra_almacen
(
    dni_almacen integer NOT NULL
    DEFAULT nextval('administra_almacen_dni_almacen_seq'::regclass),
    cedula_vendedor integer NOT NULL,
    CONSTRAINT administra_almacen_dni_almacen_fkey FOREIGN KEY (dni_almacen)
        REFERENCES public.almacen (dni) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE CASCADE
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.administra_almacen
    OWNER to postgres;
```

Para la entidad vendedor se tiene la siguiente definición:

```
CREATE TABLE public.vendedor
(
    cedula numeric(10,0) NOT NULL,
    nombre character varying(10) COLLATE pg_catalog."default" NOT NULL,
    nombre2 character varying(10) COLLATE pg_catalog."default" NOT NULL,
    apellido1 character varying(15) COLLATE pg_catalog."default" NOT NULL,
    apellido2 character varying(15) COLLATE pg_catalog."default" NOT NULL,
    telefono numeric(12,0),
    email character varying(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT vendedor_pkey PRIMARY KEY (cedula),
    CONSTRAINT vendedor_cedula_check CHECK (cedula > 5000000::numeric
                                            AND cedula < 50000000::numeric)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE public.vendedor
    OWNER to postgres;
```

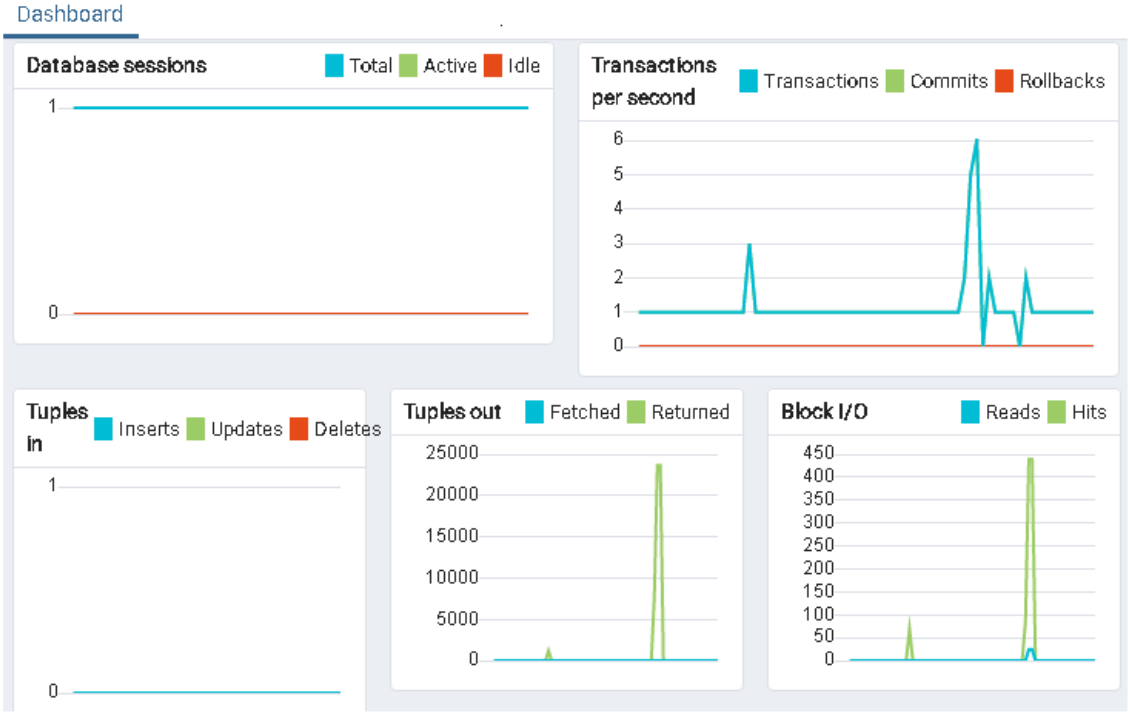
Estas fueron las definiciones para todas entidades que conforman la base principal de la ‘base de datos’, sobre ellas operaran todas las transacciones de inserción y actualización de las base de datos, así como de las transacciones triggers.

✓ **Sugerencias para monitorear y afinar el sistema de acuerdo al SGBDR seleccionado.**

El sistema de gestión de base de datos relacional escogido es PosgretSql, dicho SGBDR comprenden una lista de funcionalidades útiles para el mantenimiento, el desarrollo y la ampliación de base de datos establecidas.

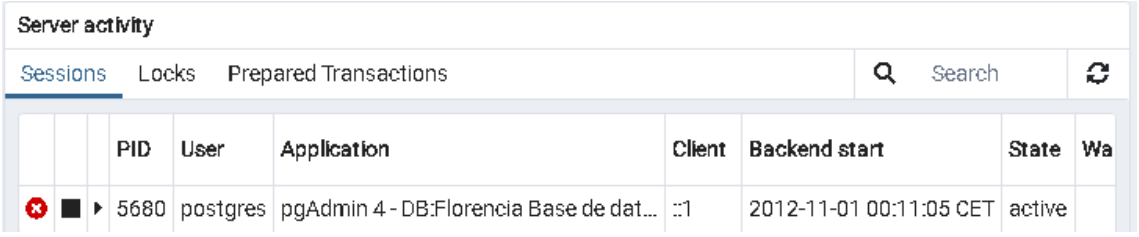
Entre las funcionalidades para el mantenimientos de una base de datos se tienen las siguientes:

1-monitorizacion de la base de datos por medio del 'DashBoard':



El DashBoard muestra en tiempo real, y en forma grafica, la cantidad de trafico realizado sobre la base de datos. En la imagen se puede apreciar los diferentes cuadros gráficos 'database sessions', 'transacciones por segundo', 'tuplas', 'tuples out' y 'bloques de entrada/salida', dichas cuadro permiten visualizar de forma sencilla u superficial el estado actual de la base de datos, y en caso necesario se podrían tomar una que otra acción al respecto.

Juntamente con el Dashboard se muestran las actividades del servidor, las cuales también son útiles para registrar las operaciones del administrador de la base de datos:



Para el mantenimiento de la base de datos, también comprender una opción de estadísticas, la cual es aplicable a todas las tablas, vista de usuario, transacciones y en la base de datos completa, dicha función permite visualizar datos de carácter estadísticos (en forma historial) de todas las operaciones realizadas sobre dicho objeto. La siguiente imagen muestra la estadística para la entidad cliente:

Statistics

Statistics	Value	Statistics	Value	Statistics	Value
Sequential scans	1	Index blocks read	1	Table size	8192 bytes
Sequential tuples read	4	Index blocks hit	0	Toast table size	
Index scans	0	Toast blocks read		Indexes size	16 kB
Index tuples fetched	0	Toast blocks hit			
Tuples inserted	0	Toast index blocks read			
Tuples updated	0	Toast index blocks hit			
Tuples deleted	0	Last vacuum			
Tuples HOT updated	0	Last autovacuum			
Live tuples	0	Last analyze			
Dead tuples	0	Last autoanalyze			
Heap blocks read	1	Vacuum counter	0		
Index blocks read	1	Autovacuum counter	0		
Index blocks hit	0	Analyze counter	0		
Toast blocks read		Autoanalyze counter	0		

Para el desarrollo de la base de datos se cuentan con dos funcionalidades muy importantes, las cuales son ‘Dependencies’ y ‘Dependents’ ellas funcionan con todos los objetos de la base de datos, y te permiten visualizar la dependencia del objeto, y además que objetos dependen de él, respectivamente. Dicha herramienta es muy importante, pues en toda base de datos se tiene que una entidad se vincula como muchas otras entidades, con varios triggers, y con diferentes vistas, por lo cual sirve para guiarte y cerciorarse de las diferentes relaciones que existen en la base de datos, con lo cual se puede tomar un plan de acción para la modificación y creación de diferentes transacciones de mejoramiento. Igual dicha utilidad sirve para que los administradores de bases de datos externos a la creación de la base de datos, puedan comprender rápidamente las diferentes relaciones entre los objetos de la base de datos.

Una vista sobre las ‘Dependencies’ y ‘Dependents’ , aplicada a la entidad cliente se muestra en la siguiente imagen:

Dependencies

Type	Name	Restriction
Schema	public	normal

Dependents

Type	Name	Restriction
Check	public.cliente_cedula_check	auto
Primary Key	public.cliente_pkey	auto
Check	public.cliente_cedula_check	normal
Foreign Key	public.cuenta.cuenta_cedula_fkey	normal
Foreign Key	public.pedido.pedido_cedula_fkey	normal
Foreign Key	public.venta.venta_cedula_cliente_fkey	normal
Rule	_RETURN ON public.datos_personales	normal
Rule	_RETURN ON public.mis_clientes	normal

Una vista sobre las ‘Dependencies’ y ‘Dependents’ , aplicada a la ‘view-articulos_disponibles’ se muestra en la siguiente imagen:

Dependencies

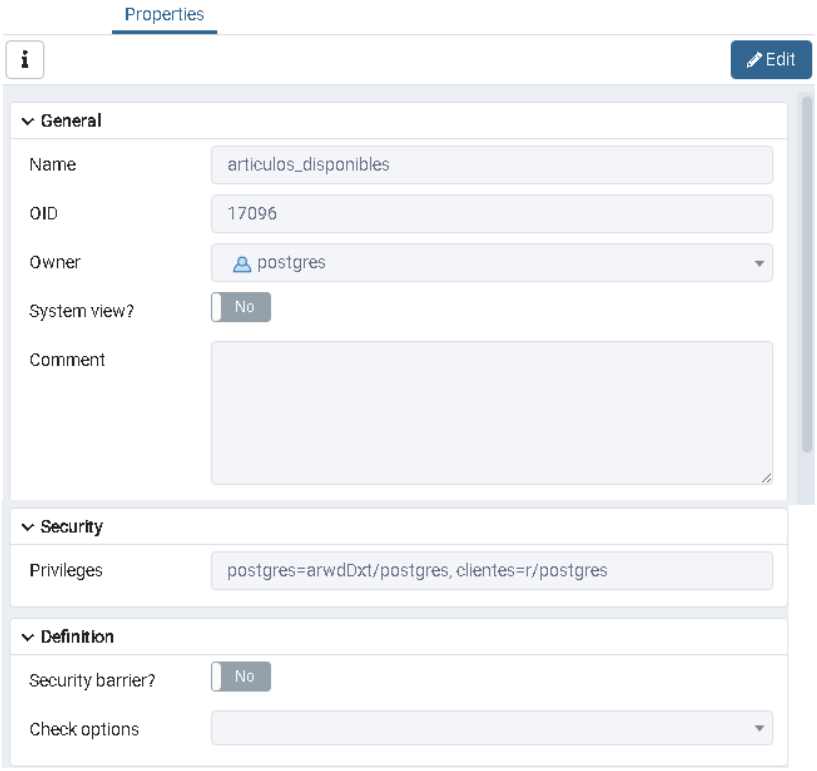
Type	Name	Restriction
Column	public.articulo.codigo	normal
Column	public.articulo.precio_de_venta	normal
Column	public.articulo.descuento	normal
Column	public.articulo.descripcion	normal
Column	public.articulo.nombre	normal
Schema	public	normal
Role	clientes	ACL

Dependents

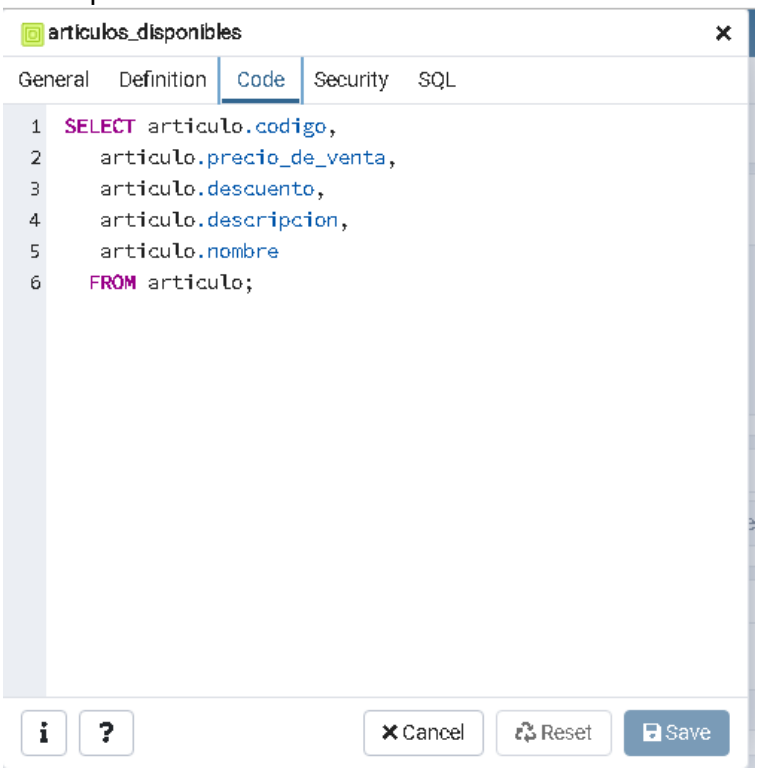
Type	Name	Restriction
Rule	_RETURN ON public.articulos_disponibles	normal

Una funcionalidad muy importante para el caso en que se necesita ampliar o modificar las entidades de la base de datos, es la herramienta ‘properties’, ella se puede aplicar sobre cualquier objeto que sea definible en SQL. La herramienta ‘properties’ te permite modificar una entidad de manera muy sencilla, y todo desde un apartado visual, puedes añadirle nuevos atributos, modificar atributos, todo ellos presionando simples botones. Si bien es cierto que todo ello se puede hacer con el comando ‘alter’ en SQL, la herramienta ‘properties’ brinda el apartado visual necesario para orientarse acerca de la composición de una entidad, vista de usuarios y objetos por demás.

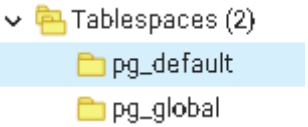
Una vista del apartado ‘properties’ sobre la vista de usuario ‘articulos_disponibles’ es la siguiente:



Si se clikea la opción de ‘edit’, se accede a un panel en el cual se puede modificar toda la composición del mismo:



Una funcionalidad también importante para el desarrollo de la base de datos, es el apartado ‘tabla space’, pues en dicho apartado se encuentra la referencia a los valores por default de las tablas, y el tamaño global de la base de datos. Todas las tablas tienen una dependencia con ‘pg_default’, esta, por definir el espacio habitual de cada tabla:



Dependents

Table	public.almacen	Florencia Base de datos TP-311
Table	public.articulo	Florencia Base de datos TP-311
Table	public.cliente	Florencia Base de datos TP-311
Table	public.cuenta	Florencia Base de datos TP-311
Table	public.pedido	Florencia Base de datos TP-311
Table	public.probar	Florencia Base de datos TP-311
Table	public.probar1	Florencia Base de datos TP-311
Table	public.probar2	Florencia Base de datos TP-311
Table	public.se_ubica	Florencia Base de datos TP-311
Table	public.vendedor	Florencia Base de datos TP-311
Table	public.venta	Florencia Base de datos TP-311
View	information_schema._pg_foreign_data_wr...	Florencia Base de datos TP-311
View	information_schema._pg_foreign_servers	Florencia Base de datos TP-311
View	information_schema._pg_foreign_table_co...	Florencia Base de datos TP-311

Documentación de los resultados de la obtención y análisis de requisitos.

Lista de requerimientos de datos de cada usuario	
Tipos de usuarios	Requerimiento de datos para el usuario (clasificados por vista de usuario)
Cliente	6- **Datos_personales(**Nombre y apellido, numero teléfono celular, email, sexo, cedula, dirección.**)** 7- **articulos_disponibles(**código, precio de venta, descuento, descripción, nombre**)** 8- **compra(**cedula_vendedor, cedula_cliente, estado, fecha, impuesto, total_monto, serial_comprobante**).** 9- **Mi_cuenta(**numero bancario , cedula , monto**).** 10- **Mis_pedidos(**nombre , cantidad_del_articulo, código_articulo, precio_del_pedido, serial_comprobante**).**
Vendedor	8- **articulo_almacen(**dni_almacen, nombre del articulo, descripción del articulo, precio de venta, cantidad del articulo, Estado del almacén (ubicacion), Municipio del almacén (ubicacion)**).** 9- **Cuales_almacenes_administro(**dni_almacen, cedula_vendedor**).** 10- **Mis_almacenes(**dni, estado , municipio**)** 11- **Mis_clientes(**cedula, nombre, nombre2, apellido1, apellido2, sexo, dirección, serial_comprobante, cedula_vendedor, cedula_cliente, estado, impuesto, fecha, total_monto, numero_bancario**).** 12- **Datos_personales(**Nombre y apellido, Ubicación, teléfono, email, sexo y cedula.**)** 13- **Mis_venta(**serial_comprobante, cedula_vendedor, cedula_cliente, estado, impuesto, fecha, total_monto, numero_bancario**).** 14- **Pedido_atencion(**dni_pedido, cedula, nombre, código_articulo, catidad_del_articulo, precio_del_pedido, serial_comprobante**)**
Administrador (almacén)	El administrador del almacén se denota como un usuario especial que tiene todos los permisos(insertar, eliminar, actualizar) para administrar sobre las entidades: 5- Almacen 6- Se_ubica 7- Administra_almacen 8- articulo
Financiero	El usuario **Financiero** se denota como un usuario especial que tiene todos los permisos(insertar, eliminar, actualizar) para administrar sobre las entidades: 4- venta 5- cuenta 6- pedido.

Lista Integrada de requerimientos de datos del sistema		
Datos	Datos	Datos
1- Nombre y apellido, **2-** numero teléfono celular, **3-** email, **4-** sexo, **5-** cedula, **6-** dirección.**)** **7-** dni_almacen, **8-** cedula_vendedor **9-** codigo **10-** precio_de_venta **11-** descuento **12-** descripcion **13-** nombre **14-** cantidad **15-** estado **16-** municipio	**17-** serial_comprobante **18-** cedula_vendedor, **19-** cedula_cliente, **20-** estado, **21-** impuesto, **22-** fecha, **23-** total_monto, **24-** numero_bancario **25-** dni_pedido, **26-** cedula, **27-** nombre, **28-** código_articulo, **29-** catidad_del_articulo precio_del_pedido,	**30-** *serial_comprobante **31-** cedula_vendedor **32-** cedula_cliente **33-** estado **34-** impuesto **35-** fecha **36-** total_monto **37-** numero_bancario **38-** monto **39-** precio_del_pedido

Documentación Conceptual

Documentación de entidad

Nombre de entidad	Cliente
Descripción de entidad	Usuario común que realiza las compras en la empresa. Esta tabla en la base de datos contendrá atributos que hagan referencia a datos personales e identificación de los clientes.
Atributos que contiene	**1-** *Cedula **2-** Nombre **3-** Nombre2 **4-** Apellido1 **5-** Apellido2 **6-** Sexo **7-** Dirección **8-** email

Nombre de entidad	pedido
Descripción de entidad	Petición del cliente para realizar la compra de x articulo. Esta tabla contendrá atributos referentes a la petición por parte de un cliente para realizar la compra de un determinad articulo. Además contendrá una referencia a la entidad venta (para cumplir con la norma de que una venta esta compuestas por uno o varios pedidos de diferentes articulos).
Atributos que contiene	**40-** *dni **41-** serial_comprobante **42-** precio_del_pedido **43-** cedula **44-** código_articulo **45-** catidad_del_articulo

Nombre de entidad	cuenta
Descripción de entidad	Cantidad de dinero que posee el cliente para realizar las distintas compras. Esta tabla contendrá atributos referentes a la identificación y estado del saldo del cliente.
Atributos que contiene	**1-** *numero_bancario **2-** cedula **3-** monto

Nombre de entidad	Venta
Descripción de entidad	Planilla final del proceso de venta. Dicha tabla contiene atributos referentes al contexto de la compra, como el cliente y el vendedor, como del monto total de la compra.
Atributos que contiene	**46-** *serial_comprobante **47-** cedula_vendedor **48-** cedula_cliente **49-** estado **50-** impuesto **51-** fecha **52-** total_monto **53-** numero_bancario

Nombre de entidad	articulo
Descripción de entidad	Muestra la variedad de artículos de los cuales dispone la empresa en sus distintos almacenes. Dicha tabla contiene atributos referentes a la descripción de un artículo (un tipo de flor).
Atributos que contiene	**54-** *codigo **55-** precio_de_venta **56-** descuento **57-** descripcion **58-** nombre

Nombre de entidad	almacen
Descripción de entidad	Entidad que almacena los artículos (Las flores). Dicha entidad muestra sobre sus atributos datos referentes únicamente al almacén, como su número de localización y su dirección.
Atributos que contiene	**1-** *dni **2-** estado **3-** municipio

Nombre de entidad	Se_ubica
Descripción de entidad	Entidad sirve de enlace entre las entidades ‘articulo’ y ‘almacen’. Esta entidad específica que articulo y la cantidad de la misma que se ubica en los almacén.
Atributos que contiene	**1-** *dni_almacen **2-** *dni_articulo **3-** cantidad

Nombre de entidad	Administra_almacen
Descripción de entidad	Entidad sirve de enlace entre las entidades ‘almacen’ y ‘vendedor. Esta entidad relaciona el dni de un almacén con el dni de un vendedor, la relación especifica que almacenes el vendedor puede operar.
Atributos que contiene	**1-** *dni_almacen **2-** *cedula_vendedor

Nombre de entidad	vendedor
Descripción de entidad	Usuario que facilita el proceso de venta. Esta entidad contiene atributos destinados a la identificación y localización de la persona asignada como un vendedor de la institución.
Atributos que contiene	**1-** *cedula **2-** nombre **3-** nombre2 **4-** apellido1 **5-** apellido2 **6-** telefono **7-** email

Documentación de atributo

Nombre de atributos	Nombre, nombre2, apellido1, apellido2
Descripción de los atributos	Estos atributos se utilizan en las entidades ‘cliente’ y ‘vendedor’, y sirven para identificar el nombre de las personas, las cuales se componen de cuatro partes, en el orden como esta ordenado. Además también se utiliza el atributo nombre en la entidad ‘articulo’ para representar el nombre por el cual se le conoce a un articulo.
Entidades que lo contienen	**1-** cliente **2-** vendedor **3-** articulo

Nombre de atributos	Dni, dni_almacen
Descripción de los atributos	Estos atributos identificadores seriales para las entidades pedido y almacén. Identifica según el orden de inserción los registros. Una lista de pedidos para una venta, está identificada según el orden de la realización de los pedidos. Los almacenes, precisamente por ser de naturaleza escasa, se pueden identificar según el orden de inserción.
Entidades que lo contienen	**1-** pedido **2-** almacen **3-** administra_almacen

Nombre de atributos	Cedula, email y direccion
Descripción de los atributos	Estos atributos son utilizados en las entidades ‘cliente’ y ‘vendedor’. Sirven para identificar a la persona (en la base de datos) por medio de numero único de identificación único. Los atributos email y dirección sirven para identificar a las persona de la cedula tal. La entidad pedido identifica quien realizo el pedido de acuerdo a su numero de cedula.
Entidades que lo contienen	**1-** cliente **2-** vendedor **3-** pedido **4-** administra_almacen **5-** venta **6-** cuenta

Nombre de atributos	Serial_comprobante
Descripción de los atributos	Este atributo identifica inequívocamente un proceso de venta. Cada venta contiene un identificador único. De esta manera se puede diferenciar todas las ventas.
Entidades que lo contienen	**1-** pedido **2-** venta

Nombre de atributos	Precio_del_pedido, total_monto, monto
Descripción de los atributos	Estos atributos aparecen respectivamente en las entidades 'pedido', 'venta' y 'cuenta'. Estos atributos representan cantidades monetarias. El atributo 'precio del pedido' muestra el costo para la cantidad de una pedido realizado. El atributo 'total_monto' muestra el valor total para una venta (el cual es la suma de todos los precios de los pedidos). El atributo 'monto' es la cantidad 'el saldo' que el cliente posee en su cuenta bancaria.
Entidades que lo contienen	**1-** pedido **2-** venta **3-** cuenta

Nombre de atributos	Código, código articulo
Descripción de los atributos	Este atributo identifica inequívocamente a un determinado articulo.
Entidades que lo contienen	**1-** pedido **2-** articulo **3-** se_ubica

Nombre de atributos	Estado, municipio
Descripción de los atributos	Estos atributos muestran la ubicación geográfica del almacén.
Entidades que lo contienen	**1-** almacen

Nombre de atributos	Numero bancario
Descripción de los atributos	Este atributo identifica inequívocamente la cuenta bancaria de un cliente.
Entidades que lo contienen	**1-** cuenta **2-** venta

Nombre de atributos	Descuento, impuesto
Descripción de los atributos	Estos atributos especifican cantidades que se descuenta o suman a las cantidades de compra. El atributo descuento se encuentra en la entidad ‘articulo’ y sirve para rebajar el precio del pedido. El atributo impuesto se encuentra en la entidad venta, y sirve para aumentar el precio de una venta.
Entidades que lo contienen	**1-** articulo **2-** venta

Nombre de atributos	Descuento, impuesto
Descripción de los atributos	Estos atributos especifican cantidades que se descuenta o suman a las cantidades de compra. El atributo descuento se encuentra en la entidad ‘articulo’ y sirve para rebajar el precio del pedido. El atributo impuesto se encuentra en la entidad venta, y sirve para aumentar el precio de una venta.
Entidades que lo contienen	**1-** articulo **2-** venta

Nombre de atributos	Estado
Descripción de los atributos	Este atributo se ubica en la entidad venta, y sirve para especificar el estado de una venta. Si el estado es igual a ‘false’, entonces se denota que la venta para el registro x esta aun sin completar, si el estado es igual a ‘true’ entonces quiere decir que la venta ya está realizada.
Entidades que lo contienen	**1-** venta

Nombre de atributos	fecha
Descripción de los atributos	Registra el momento del sistema en el cual se introdujo o comenzó una registro de venta en la entidad ‘venta’
Entidades que lo contienen	**1-** venta

Nombre de atributos	descripcion
Descripción de los atributos	Atributo que guarda una descripción en forma de texto sobre un determinado articulo.
Entidades que lo contienen	**1-** articulo

Documentación de relación (vinculo)

Nombre relación	Realiza pedido
Descripción relación:	Esta relación expresa que un cliente puede realizar pedidos, lo cual le da derecho de inserción sobre la tabla pedido.
Entidades involucradas en la relación:	**1-** cliente **2-** pedido

Nombre relación	posee
Descripción relación:	Esta relación expresa que cada cliente tiene una cuenta bancaria, de la cual podrá costear los pedidos.
Entidades involucradas en la relación:	**1-** cliente **2-** cuenta

Nombre relación	carrito
Descripción relación:	Esta relación expresa que una venta puede estar conformada por una lista de pedidos. Como en el caso de los centros comerciales, que se eligen varios víveres y luego se pagan en la caja.
Entidades involucradas en la relación:	**1-** pedido **2-** venta

Nombre relación	existencia
Descripción relación:	Esta relación comprueba que los articulo pedidos por los clientes verdaderamente sean validos (existan en la entidad ‘articulo’).
Entidades involucradas en la relación:	**1-** pedido **2-** articulo

Nombre relación	realiza
Descripción relación:	Esta relación denota que un vendedor realiza ventas, que esa es la función de los vendedores.
Entidades involucradas en la relación:	**1-** venta **2-** vendedor

Nombre relación	Se ubica
Descripción relación:	Esta relación entre las entidades ‘articulo’ y ‘almacén’, sirve para relacionar un determinado articulo con un determinado almacén y especificar la cantidad que se halla de este articulo en dicho almacén.
Entidades involucradas en la relación:	**1-** articulo **2-** almacén

Nombre relación	Administra almacen
Descripción relación:	Esta relación entre las entidades 'almacén' y 'vendedor', sirve para relacionar un determinado vendedor con un determinado almacén y de esta manera especificar sobre que almacenes el vendedor tiene acceso.
Entidades involucradas en la relación:	**1-** vendedor **2-** almacén

Documentación de cardinalidad de una relación

Nombre relación (vinculo)	Realiza pedido
Descripción de la restricción de cardinalidad:	Esta relación expresa que un cliente puede realizar muchos pedidos, o de la misma manera, muchos pedidos pueden ser realizados por un cliente. Para cumplir con este relación la entidad pedido posee un atributo 'cedula' el cual hace relación a la cedula del cliente que está haciendo el pedido.
Cardinalidad	1:N (de uno a muchos)
Nombre entidades involucradas	**1-** cliente **2-** pedido

Nombre relación (vinculo)	posee
Descripción de la restricción de cardinalidad:	Esta relación expresa que un cliente puede tener a lo sumo una cuenta bancaria, o una cuenta bancaria puede tener a lo sumo un cliente.
Cardinalidad	1:1 (de uno a uno)
Nombre entidades involucradas	**1-** cliente **2-** cuenta

Nombre relación (vinculo)	carrito
Descripción de la restricción de cardinalidad:	Esta relación expresa una venta está conformada por uno o muchos pedidos, o que muchos pedidos están relacionados a una venta.
Cardinalidad	N:1 (de muchos a uno)
Nombre entidades involucradas	**1-** pedido **2-** venta

Nombre relación (vinculo)	Existencia
Descripción de la restricción de cardinalidad:	Esta relación expresa un articulo puede estar referenciado en muchos pedidos, o que se pueden hacer muchos pedidos de un determinado articulo.
Cardinalidad	N:1 (de muchos a uno)
Nombre entidades involucradas	**1-** pedido **2-** articulo

Nombre relación (vinculo)	Realiza
Descripción de la restricción de cardinalidad:	Esta relación expresa que un vendedor puede efectuar muchas ventas, o que muchas ventas pueden ser realizadas por un solo vendedor.
Cardinalidad	N:1 (de muchos a uno)
Nombre entidades involucradas	**1-** venta **2-** vendedor

Nombre relación (vinculo)	Se ubica
Descripción de la restricción de cardinalidad:	Esta relación expresa que muchos artículos pueden estar ubicados en muchos almacenes, o que en muchos almacenes se encuentran muchos artículos. De esta manea un almacén puede almacenar muchos cantidades de artículos diferentes.
Cardinalidad	N:N (de muchos a muchos)
Nombre entidades involucradas	**1-** articulo **2-** almacén

Nombre relación (vinculo)	Administra almacen
Descripción de la restricción de cardinalidad:	Esta relación expresa que muchos vendedores pueden administrar muchos almacenes, o que muchos almacenes pueden ser administrados por varios vendedores. De esta manera un almacen puede ser administrado por mas de un vededor.
Cardinalidad	N:N (de muchos a muchos)
Nombre entidades involucradas	**1-** vendedor **2-** almacen

Nombre relación (vinculo)	Confirma
Descripción de la restricción de cardinalidad:	Esta relación expresa muchas ventas pueden ser realizadas por una misma cuenta bancaria, es decir, un vendedor puede realizar muchas compras.
Cardinalidad	N:1 (de muchos a uno)
Nombre entidades involucradas	**1-** venta **2-** cuenta

Documentación de transacción

Nombre transacción	**articulo_almacen**
Tipo (categoría) transacción:	visualizar
Descripción de la transacción: (comportamiento funcional)	Visualizar la cantidad y que artículos se encuentran disponibles en los almacenes donde administra un determinado vendedor**.** Dicha visualización solo lo pueden utilizar los vendedores, Pues el sistema toma la cedula del usuario actual del vendedor.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 15- dni_almacen 16- nombre del articulo 17- descripción del articulo 18- precio de venta 19- cantidad del articulo 20- Estado del almacén (ubicacion) 21- Municipio del almacén (ubicacion).
Frecuencia de utilización	La idea es que el vendedor pueda visualizar que artículos puede vender, por lo que se puede espera que recurra a dicha visualización al menos unas tres veces antes de completar una venta. Además se espera que un vendedor haga de 3 a 10 ventas por día.
Tiempo de respuesta estimado	Alrededor de 746 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 10 ventas por días, se puede tener que se hayan hecho 30 visualizaciones, lo que daría una productividad de: 30(746 msec)=22,38seg
Entidades involucradas	1- Articulo 2- Almacen 3- Se_ubica 4- Administra_almacen
Atributos involucrados	1- Articulo (nombre, descripción, precio de venta). 2- Se_ubica (dni_almacen,cantidad). 3- Almacen (estado, municipio). 4- Administra_almacen(dni_almacen, cedula_vendedor)
Usuarios involucrados	vendedores

Nombre transacción	articulos_disponibles
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el cliente pueda visualizar los artículos que se encuentra disponibles, con el fin de que pueda elegir que articulo comprar.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 11- codigo 12- precio de venta 13- descuento 14- descripcion 15- nombre
Frecuencia de utilización	La idea es que el cliente pueda visualizar que artículos están disponibles. Se espera que un cliente habitual realice hasta una vista de los artículos disponibles antes de proceder a realizar la petición.
Tiempo de respuesta estimado	Alrededor de 850 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 1.000 pedidos diarios, se tiene una productividad estimada de: 1.000(850 msec)=850,38seg=14,16min
Entidades involucradas	1- Articulo
Atributos involucrados	1- codigo 2- precio de venta 3- descuento 4- descripción 5- nombre
Usuarios involucrados	clientes

Nombre transacción	compra
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el cliente pueda visualizar los datos de la compra realizada, a modo de factura.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 16- cedula_vendedor 17- cedula_cliente 18- estado 19- fecha 20- impuesto 21- total_monto 22- serial_comprobante
Frecuencia de utilización	La idea es que el cliente pueda visualizar los datos de su venta una vez que esta esté culminada, o este en proceso, por lo que se espera que el cliente utilice esta vista por los menos 3 veces antes de completar una compra.

Tiempo de respuesta estimado	Alrededor de 850 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 1.000 ventas diarias, se tiene una productividad estimada de: 3.000(850 msec)=2550seg=42,5min
Entidades involucradas	1- venta
Atributos involucrados	1- cedula_vendedor 2- cedula_cliente 3- estado 4- fecha 5- impuesto 6- total_monto 7- serial_comprobante
Usuarios involucrados	clientes

Nombre transacción	**compra**
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el cliente pueda visualizar los datos de la compra realizada, a modo de factura.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 23- cedula_vendedor 24- cedula_cliente 25- estado 26- fecha 27- impuesto 28- total_monto 29- serial_comprobante
Frecuencia de utilización	La idea es que el cliente pueda visualizar los datos de su venta una vez que esta esté culminada, o este en proceso, por lo que se espera que el cliente utilice esta vista por los menos 3 veces antes de completar una compra.
Tiempo de respuesta estimado	Alrededor de 850 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 1.000 ventas diarias, se tiene una productividad estimada de: 3.000(850 msec)=2550seg=42,5min
Entidades involucradas	2- venta
Atributos involucrados	8- cedula_vendedor 9- cedula_cliente 10- estado 11- fecha 12- impuesto 13- total_monto 14- serial_comprobante
Usuarios involucrados	clientes

Nombre transacción	Cuales_almacenes_administro
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el vendedor pueda visualizar cuales son los almacenes sobre los que él tiene autorización de acceso.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 22- dni_almacen 23- cedula_vendedor
Frecuencia de utilización	La idea es que el vendedor pueda saber, o enterarse que almacenes él puede operar. Por lo que se espera que su visualización sea poca, tal vez una vez al dia.
Tiempo de respuesta estimado	Alrededor de 600 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 3 vistas diarias, se tiene una productividad estimada de: 3(600 msec)=1,8seg
Entidades involucradas	1- administra_almacen
Atributos involucrados	1- dni_almacen 2- cedula_vendedor
Usuarios involucrados	vendedores

Nombre transacción	Datos_personales
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el cliente pueda visualizar sus datos personales.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 1- cedula 2- nombre 3- nombre2 4- apellido1 5- apellido2 6- sexo 7- dirección 8- email
Frecuencia de utilización	El cliente podría ver sus datos personales 1 vez al día.
Tiempo de respuesta estimado	Alrededor de 600 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 1 vistas diarias, se tiene una productividad estimada de: 1(600 msec)=0,60seg
Entidades involucradas	1- cliente
Atributos involucrados	9- cedula 10- nombre

	11- nombre2 12- apellido1 13- apellido2 14- sexo 15- dirección 1- email
Usuarios involucrados	clientes

Nombre transacción	**Mi_cuenta**
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el cliente pueda sus visualizar datos bancarios.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 30- numero bancario 31- cedula 32- monto
Frecuencia de utilización	Es más frecuente que un cliente vea varias veces el saldo de su cuenta para confirmar, por lo que se espera que el cliente consulte estos datos por lo menos unas 5 veces día.
Tiempo de respuesta estimado	Alrededor de 500 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 100 clientes haciendo vista diarias, se espera una productividad estimada de: 500(500 msec)=250seg=4,16min
Entidades involucradas	1- cuenta
Atributos involucrados	1- numero bancario 2- cedula 3- monto
Usuarios involucrados	clientes

Nombre transacción	**Mis_almacenes**
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el vendedor pueda visualiza los datos sobre el almacén que el administra.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 2- dni 3- estado 4- municipio
Frecuencia de utilización	El vendedor podría necesitar ver los datos de los almacenes que el administra 1 vez al dia.
Tiempo de respuesta estimado	Alrededor de 500 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.

Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de una 1 vista al día se tiene la siguiente productividad estimada: 1(500 msec)=0,500 seg
Entidades involucradas	1- almacen 2- administra_almacen
Atributos involucrados	1- dni 2- estado 3- municipio
Usuarios involucrados	vendedores

Nombre transacción	**Mis_clientes**
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el vendedor pueda visualizar a todos los clientes que él ha atendido, como un histórico de las ventas realizadas.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 5- cedula 6- nombre 7- nombre2 8- apellido1 9- apellido2 10- sexo 11- dirección 12- serial_comprobante 13- cedula_vendedor 14- cedula_cliente 15- estado 16- impuesto 17- fecha 18- total_monto 19- numero_bancario
Frecuencia de utilización	El vendedor podría solicitar estos datos ocasionalmente para visualizar los clientes que el a atendido, alrededor de 1 vez al dia.
Tiempo de respuesta estimado	Alrededor de 1 seg, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de una 20 vendedore haciendo 1 vista al día se tiene la siguiente productividad estimada: 20(1 seg)=20 seg
Entidades involucradas	1- venta 2- pedido
Atributos involucrados	1- cedula 2- nombre 3- nombre2 4- apellido1 5- apellido2 6- sexo 7- dirección 8- serial_comprobante

	9- cedula_vendedor 10- cedula_cliente 11- estado 12- impuesto 13- fecha 14- total_monto 15- numero_bancario
Usuarios involucrados	vendedores

Nombre transacción	**Datos_personales**
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el vendedor pueda visualizar sus datos personales.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 16- cedula 17- nombre 18- nombre2 19- apellido1 20- apellido2 21- sexo 22- dirección 23- email
Frecuencia de utilización	El vendedor podría ver sus datos personales 1 vez al día.
Tiempo de respuesta estimado	Alrededor de 600 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 20 vendedores haciendo 1 vistas diaria, se tiene una productividad estimada de: 20(600 msec)=12 seg
Entidades involucradas	2- cliente
Atributos involucrados	1- cedula 2- nombre 3- nombre2 4- apellido1 5- apellido2 6- sexo 7- dirección 8- email
Usuarios involucrados	vendedores

Nombre transacción	Mis_pedidos
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el cliente pueda visualizar los pedidos que ha realizado
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 33- nombre 34- cantidad_del_articulo 35- código_articulo 36- precio_del_pedido 37- serial_comprobante
Frecuencia de utilización	El cliente podría necesitar ver sus pedidos realizados unas tres veces por compra diaria.
Tiempo de respuesta estimado	Alrededor de 600 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 100 clientes, haciendo 3 vista de pedidos diarios: 100(3)(600 msec)=180 seg=3 min
Entidades involucradas	1- pedido 2- articulo
Atributos involucrados	38- nombre 39- cantidad_del_articulo 40- código_articulo 41- precio_del_pedido 1- serial_comprobante
Usuarios involucrados	clientes

Nombre transacción	Mis_venta
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el vendedor pueda visualizar todas sus ventas, realizadas o en proceso.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 20- serial_comprobante 21- cedula_vendedor 22- cedula_cliente 23- estado 24- impuesto 25- fecha 26- total_monto 27- numero_bancario
Frecuencia de utilización	El vendedor podría necesitar ver ventas realizadas o en proceso unas 5 veces diarias.
Tiempo de respuesta estimado	Alrededor de 700 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de	Para un total de 20 vendedores, haciendo

transacción / intervalo de tiempo)	5 vista de ventas diarias, se tiene una productividad estimada de: 20(5)(700 msec)=70 seg=1,16 min
Entidades involucradas	1- venta
Atributos involucrados	1- serial_comprobante 2- cedula_vendedor 3- cedula_cliente 4- estado 5- impuesto 6- fecha 7- total_monto 8- numero_bancario
Usuarios involucrados	vendedores

Nombre transacción	**Pedido_atencion**
Tipo (categoría) transacción:	Visualizar (view)
Descripción de la transacción: (comportamiento funcional)	Que el vendedor pueda visualizar los pedidos realizados por su cliente.
Salida de la transacción:	Visualización de los siguiente atributos, en el mismo orden: 28- dni_pedido 29- cedula 30- nombre 31- código_articulo 32- catidad_del_articulo 33- precio_del_pedido 34- serial_comprobante
Frecuencia de utilización	El vendedor podría necesitar ver los pedidos realizados por sus clientes unas 5 veces diarias.
Tiempo de respuesta estimado	Alrededor de 800 msec, puede variar según el tipo de procesador, y la carga de tareas que este ejecutando.
Productividad estimada (cantidad de transacción / intervalo de tiempo)	Para un total de 20 vendedores, haciendo 5 vista de ventas diarias, se tiene una productividad estimada de: 20(5)(800 msec)=80 seg=1,33 min
Entidades involucradas	1- articulo 2- pedido 3- venta
Atributos involucrados	1- dni_pedido 2- cedula 3- nombre 4- código_articulo 5- catidad_del_articulo 6- precio_del_pedido 7- serial_comprobante
Usuarios involucrados	vendedores

Documentación de datos-modelo lógico

Documentación de entidad

Nombre de entidad	**Cliente**	
Descripción de entidad	Usuario común que realiza las compras en la empresa. Esta tabla en la base de datos contendrá atributos que hagan referencia a datos personales e identificación de los clientes.	
Tamaño registro (bytes/reg):	135(bytes/reg)	
Volumen estimado crecimiento (reg/año):	Se espera que los clientes aumenten en cantidades de 1.000 por año.	
Capacidad de almacenamiento requerida (bytes/año):	135 bytes x 1.000 = 135.000bytes = 135 kilo bytes.	
	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
1- Cedula	**Integer (**-2147483648 to +2147483647**)**	**principal**
2- Nombre	**CHARACTER** varying (10)	
3- Nombre2	**CHARACTER** varying (10)	
4- Apellido1	**CHARACTER** varying (15)	
5- Apellido2	**CHARACTER** varying (15)	
6- Sexo	**CHARACTER** varying (1)	
7- Dirección	**CHARACTER** varying (50)	
8- email	**CHARACTER** varying (30)	

Nombre de entidad	**pedido**
Descripción de entidad	Petición del cliente para realizar la compra de x articulo. Esta tabla contendrá atributos referentes a la petición por parte de un cliente para realizar la compra de un determinad articulo. Además contendrá una referencia a la entidad venta (para cumplir con la norma de que una venta esta compuestas por uno o varios pedidos de diferentes articulos).
Tamaño registro (bytes/reg):	25 (bytes/reg)
Volumen estimado crecimiento (reg/año):	Se espera que los clientes realicen alrededor de 10.000 pedidos por año y aumente a

	esta proporción.	
Capacidad de almacenamiento requerida (bytes/año):	25 bytes x 10.000 = 135.000bytes = 250 kilo bytes/año.	
	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
1- *dni	**Integer (**-2147483648 to +2147483647**)**	**principal**
2- serial_comprobante	**Numeric (12)**	Ajena
3- precio_del_pedido	**Numeric (12,2)**	
4- cedula	**Integer (**-2147483648 to +2147483647**)**	
5- código_articulo	**Numeric (7)**	Ajena
6- catidad_del_articulo	**Integer (**-2147483648 to +2147483647**)**	

Nombre de entidad	**cuenta**	
Descripción de entidad	Cantidad de dinero que posee el cliente para realizar las distintas compras. Esta tabla contendrá atributos referentes a la identificación y estado del saldo del cliente.	
Tamaño registro (bytes/reg):	16 (bytes/reg)	
Volumen estimado crecimiento (reg/año):	Cada cliente puede tener una sola cuenta, asi que el crecimiento de esta tabla es igual al de la tabla cliente; 1.000 por año.	
Capacidad de almacenamiento requerida (bytes/año):	16 bytes x 1.000 = 16.000bytes = 16 kilo bytes/año.	
	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
1- *numero_bancario	**Numeric (20)**	**principal**
2- cedula	**Integer (**-2147483648 to +2147483647**)**	Ajena
3- monto	**Numeric (12,2)**	

Nombre de entidad	venta	
Descripción de entidad	Planilla final del proceso de venta. Dicha tabla contiene atributos referentes al contexto de la compra, como el cliente y el vendedor, como del monto total de la compra.	
Tamaño registro (bytes/reg):	36 (bytes/reg)	
Volumen estimado crecimiento (reg/año):	La proporción de las ventas tiene que ser mucho menor a los pedidos, así que se espera que el incremento de las ventas sea de unos 2.000 por año.	
Capacidad de almacenamiento requerida (bytes/año):	36 bytes x 2.000 = 72.000bytes = 72 kilo bytes/año.	
	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
59- *serial_comprobante 1-	**Numeric (12)**	**principal**
60- cedula_vendedor 2-	**Integer (**-2147483648 to +2147483647**)**	Ajena
61- cedula_cliente 3-	**Integer (**-2147483648 to +2147483647**)**	Ajena
1- estado 2-	**boolean**	
62- impuesto 3-	**Integer (**-2147483648 to +2147483647**)**	
63- fecha **64-**	**TIMESTAMP** (with time zone)	
65- total_monto **66-**	**Numeric (12,2)**	
67- numero_bancario	**Numeric (20)**	Ajena

Nombre de entidad	articulo
Descripción de entidad	Muestra la variedad de artículos de los cuales dispone la empresa en sus distintos almacenes. Dicha tabla contiene atributos referentes a la descripción de un artículo (un tipo de flor).
Tamaño registro (bytes/reg):	83 bytes
Volumen estimado crecimiento (reg/año):	La proporción de aumento de los artículos, se espera que sean de 18 ejemplares por año.
Capacidad de almacenamiento requerida	83 bytes x 18 = 1.494 bytes = 1,494 kilo bytes/año.

(bytes/año):		
	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
1- *codigo	**Numeric (7)**	**principal**
2- precio_de_venta	**Numeric (12,2)**	
3- descuento	**Numeric (12,2)**	
4- descripcion	**Text**	
5- nombre	**CHARACTER** varying (20)	

Nombre de entidad	**almacen**
Descripción de entidad	Entidad que almacena los artículos (Las flores). Dicha entidad muestra sobre sus atributos datos referentes únicamente al almacén, como su número de localización y su dirección.
Tamaño registro (bytes/reg):	36 (bytes/reg)
Volumen estimado crecimiento (reg/año):	La proporción de aumento de los almacenes, si es que sucede es de a lo sumo 2 por año.
Capacidad de almacenamiento requerida (bytes/año):	36 bytes x 2 = 72 bytes/año.

	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
1- *dni	**Integer (**-2147483648 to +2147483647**)**	**principal**
2- estado	**CHARACTER** varying (12)	
3- municipio	**CHARACTER** varying (20)	

Nombre de entidad	**Se_ubica**
Descripción de entidad	Entidad sirve de enlace entre las entidades ‘articulo’ y ‘almacen’. Esta entidad específica que articulo y la cantidad de la misma que se ubica en los almacén.
Tamaño registro (bytes/reg):	36 (bytes/reg)
Volumen estimado crecimiento (reg/año):	La proporción de aumento de los almacenes, si es que

	sucede es de a lo sumo 2 por año.	
Capacidad de almacenamiento requerida (bytes/año):	36 bytes x 2 = 72 bytes/año.	
	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
1- *dni_almacen	**Integer (**-2147483648 to +2147483647**)**	**principal**
2- *dni_articulo	**Numeric (7)**	**secundaria**
3- cantidad	**Integer (**-2147483648 to +2147483647**)**	

Nombre de entidad	**Administra_almacen**	
Descripción de entidad	Entidad sirve de enlace entre las entidades 'almacen' y 'vendedor. Esta entidad relaciona el dni de un almacén con el dni de un vendedor, la relación especifica que almacenes el vendedor puede operar.	
Tamaño registro (bytes/reg):	8 (bytes/reg)	
Volumen estimado crecimiento (reg/año):	La proporción de aumento de esta tabla debe estar en proporción con la de la entidad vendedor y almacén, así que se espera que crezca alrededor de 20 registros por año.	
Capacidad de almacenamiento requerida (bytes/año):	8 bytes x 20 = 160 bytes/año.	
	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
3- *dni_almacen 1-	**Integer (**-2147483648 to +2147483647**)**	**principal**
2- *cedula_vendedor	**Integer (**-2147483648 to +2147483647**)**	**secundaria**

Nombre de entidad	vendedor	
Descripción de entidad	Usuario que facilita el proceso de venta. Esta entidad contiene atributos destinados a la identificación y localización de la persona asignada como un vendedor de la institución.	
Tamaño registro (bytes/reg):	87 (bytes/reg)	
Volumen estimado crecimiento (reg/año):	La proporción de aumento de esta tabla deber limitada, pues no se pretende contratar ilimitadamente nuevos vendedores, por lo que se espera de que sea de por lo menos 10 al año.	
Capacidad de almacenamiento requerida (bytes/año):	87 bytes x 10 = 870 bytes/año.	
	Atributos que contiene:	
Nombre atributo:	**Longitud atributo:**	**Clave** (principal, secundaria, Ajena).
1- Cedula	**Integer (**-2147483648 to +2147483647**)**	**principal**
2- Nombre	**CHARACTER** varying (10)	
3- Nombre2	**CHARACTER** varying (10)	
4- Apellido1	**CHARACTER** varying (15)	
5- Apellido2	**CHARACTER** varying (15)	
6- teléfono	**Numeric(12)**	
7- email	**CHARACTER** varying (30)	

Documentacion de atributo (modelo logico)

Nombre atributo	**Nombre, nombre2, apellido1, apellido2**
Descripción atributo	Estos atributos se utilizan en las entidades 'cliente' y 'vendedor', y sirven para identificar el nombre de las personas, las cuales se componen de cuatro partes, en el orden como esta ordenado. Además también se utiliza el atributo nombre en la entidad 'articulo' para representar el nombre por el cual se le conoce a un articulo.
Tipo atributo:	**1- Nombre (CHARACTER** varying (10)**)** **2- nombre2 (CHARACTER** varying (10)**)** **3- apellido1 (CHARACTER** varying (15)**)** **4- apellido2(CHARACTER** varying (15)**)**
Longitud atributo (bytes):	**1- Nombre (10 bytes)** **2- nombre2 (10 bytes)** **3- apellido1 (15 bytes)** 4- **apellido2(15 bytes)**
Restricciones (validaciones) requerida sobre el atributo:	**1- Nombre (not null, se debe especificar un valor)** **2- nombre2 (not null, se debe especificar un valor)** **3- apellido1 (not null, se debe especificar un valor)** 4- **apellido2(not null, se debe especificar un valor)**
Entidades que lo contienen:	**1-** cliente **2-** vendedor **3-** articulo

Nombre atributo	Dni, dni_almacen
Descripción atributo	Estos atributos identificadores seriales para las entidades pedido y almacén. Identifica según el orden de inserción los registros. Una lista de pedidos para una venta, está identificada según el orden de la realización de los pedidos. Los almacenes, precisamente por ser de naturaleza escasa, se pueden identificar según el orden de inserción.
Tipo atributo:	**1- Dni (Integer)** **2- dni_almacen(Integer)**
Longitud atributo (bytes):	**1- Dni (4 bytes)** 2- **dni_almacen(4 bytes)**
Restricciones (validaciones) requerida sobre el atributo:	**5- Dni (not null, se debe especificar un valor)** **6- dni_almacen (not null, se debe especificar un valor)**
Entidades que lo contienen:	**1-** pedido **2-** almacen **3-** administra_almacen

Nombre atributo	Cedula, email y direccion
Descripción atributo	Estos atributos son utilizados en las entidades 'cliente' y 'vendedor'. Sirven para identificar a la persona (en la base de datos) por medio de numero único de identificación único. Los atributos email y dirección sirven para identificar a las persona de la cedula tal. La entidad pedido identifica quien realizo el pedido de acuerdo a su numero de cedula.
Tipo atributo:	**1- Cedula (Integer)** **2- email (character varying)** **3- dirección(character varying)**
Longitud atributo (bytes):	**1- Cedula (4 bytes)** **2- email (30 bytes)** **3- dirección**(50 bytes)
Restricciones (validaciones) requerida sobre el atributo:	**7- Cedula (not null, se debe especificar un valor)** **8- email (not null, se debe especificar un valor)**
Entidades que lo contienen:	**1-** cliente **2-** vendedor **3-** pedido **4-** administra_almacen **5-** venta **6-** cuenta

Nombre atributo	Serial_comprobante
Descripción atributo	Este atributo identifica inequívocamente un proceso de venta. Cada venta contiene un identificador único. De esta manera se puede diferenciar todas las ventas.
Tipo atributo:	**serial_comprobante (numeric(12))**
Longitud atributo (bytes):	**serial_comprobante(3 bytes)**
Restricciones (validaciones) requerida sobre el atributo:	**serial_comprobante (check serial_comprobante > 99.999.999.999. and serial_comprobante < 1.000.000.000.000).**
Entidades que lo contienen:	**1-** pedido **2-** venta

Nombre atributo	Precio_del_pedido, total_monto, monto
Descripción atributo	Estos atributos aparecen respectivamente en las entidades 'pedido', 'venta' y 'cuenta'. Estos atributos representan cantidades monetarias. El atributo 'precio del pedido' muestra el costo para la cantidad de una pedido realizado. El atributo 'total_monto' muestra el valor total para una venta (el cual es la suma de todos los precios de los pedidos). El atributo 'monto' es la cantidad 'el saldo' que el cliente posee en su cuenta bancaria.
Tipo atributo:	**1- Precio_del_pedido (Numeric (12,2)** **2- total_monto(Numeric (12,2))** **3- monto(Numeric (12,2))**
Longitud atributo (bytes):	**1- Precio_del_pedido (5 bytes)** **2- total_monto(5 bytes)** **3- monto(5 bytes))**
Restricciones (validaciones) requerida sobre el atributo:	**1- Precio_del_pedido (check Precio_del_pedido > 0)** **2- total_monto(check total_monto > 0)** **3- monto(check monto > 0)**
Entidades que lo contienen:	**1-** pedido **2-** venta **3-** cuenta

Nombre atributo	Código, código articulo

Descripción atributo	Este atributo identifica inequívocamente a un determinado articulo.
Tipo atributo:	**1- Código (Numeric (7))** **2- código articulo(Numeric (7))**
Longitud atributo (bytes):	**1- Código (3 bytes)** 2- **código articulo(3 bytes)**
Restricciones (validaciones) requerida sobre el atributo:	**1- Código (check Código > 999.999 and Código < 10.000.000).** **2- código articulo(check Código > 999.999 and Código < 10.000.000).**
Entidades que lo contienen:	**1-** pedido **2-** articulo **3-** se_ubica

Nombre atributo	**Estado, municipio**
Descripción atributo	Estos atributos muestran la ubicación geográfica del almacén.
Tipo atributo:	**1- Estado (CHARACTER** varying (12)) **2- municipio (CHARACTER** varying (20))
Longitud atributo (bytes):	**1- Estado (12 bytes)** 2- **municipio (20 bytes)**
Restricciones (validaciones) requerida sobre el atributo:	**1- Estado (check Estado = 'merida' or Estado = 'aragua' or Estado = 'miranda' or Estado = 'trujillo' or Estado = 'bolivar' or Estado = 'potuguesa' or Estado = 'distrito capital' or Estado = 'monagas') not null.** **2- Municipio(not null, se debe especificar un valor)**
Entidades que lo contienen:	**1-** almacen

Nombre atributo	**Numero bancario**
Descripción atributo	Este atributo identifica inequívocamente la cuenta bancaria de un cliente.
Tipo atributo:	**Numero bancario(numeric (20))**
Longitud atributo (bytes):	**Numero bancario(9 bytes)**
Restricciones (validaciones) requerida sobre el atributo:	**Numero bancario (check serial_comprobante > 9.999.999.999.999.999.999 and serial_comprobante < 100.000.000.000.000.000.000).**
Entidades que lo contienen:	**1-** cuenta **2-** venta

Nombre atributo	Descuento, impuesto
Descripción atributo	Estos atributos especifican cantidades que se descuenta o suman a las cantidades de compra. El atributo descuento se encuentra en la entidad 'articulo' y sirve para rebajar el precio del pedido. El atributo impuesto se encuentra en la entidad venta, y sirve para aumentar el precio de una venta.
Tipo atributo:	**1- Descuento(integer)** **2- Impuesto(numeric (12,2))**
Longitud atributo (bytes):	**1- Descuento(4 bytes)** 2- **Impuesto (3 bytes)**
Restricciones (validaciones) requerida sobre el atributo:	**1- Descuento(check Descuento > 0)** **2- impuesto(check impuesto > 0)**
Entidades que lo contienen:	**1-** articulo **2-** venta

Nombre atributo	Estado
Descripción atributo	Este atributo se ubica en la entidad venta, y sirve para especificar el estado de una venta. Si el estado es igual a 'false', entonces se denota que la venta para el registro x esta aun sin completar, si el estado es igual a 'true' entonces quiere decir que la venta ya está realizada.
Tipo atributo:	**Estado (boolean)**
Longitud atributo (bytes):	**Estado (1 byte)**
Restricciones (validaciones) requerida sobre el atributo:	**Estado (default Estado = false);**
Entidades que lo contienen:	**1-** venta

Nombre atributo	fecha
Descripción atributo	Registra el momento del sistema en el cual se introdujo o comenzó una registro de venta en la entidad 'venta'
Tipo atributo:	**fecha (TIMESTAMP (with time zone))**
Longitud atributo (bytes):	**fecha (8 bytes)**
Restricciones (validaciones) requerida sobre el atributo:	**fecha (default fecha = now(), "que sería la hora actual del sistema");**
Entidades que lo contienen:	**1-** venta

Nombre atributo	descripcion
Descripción atributo	Atributo que guarda una descripción en forma de texto sobre un determinado articulo.
Tipo atributo:	**descripcion (TEXT)**
Longitud atributo (bytes):	**descripcion (**variable unlimited length**)**
Restricciones (validaciones) requerida sobre el atributo:	**descripcion (default descripción = null;**
Entidades que lo contienen:	**1-** articulo

Documentación de ficha técnica-SGBDR

<table>
<tr><th colspan="2">Ficha técnica de SGBDR:</th></tr>
<tr><th>Nombre:</th><th>Microsoft SQL Server</th></tr>
<tr><th colspan="2">Requerimientos:</th></tr>
<tr><td colspan="2">Requerimientos de Hardware: Recomendados; 1gb ram, procesador qualcore para soporte en paralelos de tareas de la base de datos.

Requerimientos de software: Esta disponibles, tanto para Docker, Windows y los derivados de GNU/Linux.</td></tr>
<tr><td>Descripción:</td><td>Es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft. Utiliza un lenguaje de desarrollo denominado ‘Transact-SQL (TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL). Dicho lenguaje se puede utilizar en línea de comando ó mediante la interfaz grafica de Management Studio.</td></tr>
<tr><td>Características:</td><td>• Soporte de transacciones.
• Soporta procedimientos almacenados.
• Incluye también un entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente.
• Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información.
• Además permite administrar información de otros servidores de datos.
• En el manejo de SQL mediante líneas de comando se utiliza el SQLCMD, osql, o PowerShell.
• Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos.</td></tr>
</table>

<table>
<tr><th colspan="2">Ficha técnica de SGBDR:</th></tr>
<tr><td>Nombre:</td><td>MySQL</td></tr>
<tr><td colspan="2">Requerimientos:</td></tr>
<tr><td colspan="2">Requerimientos de Hardware: Recomendados; 1gb ram, procesador qualcore para soporte en paralelos de tareas de la base de datos.

Requerimientos de software: Esta disponibles, Windows y los derivados de GNU/Linux.</td></tr>
<tr><td>Descripción:</td><td>MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base de datos de código abierto más popular del mundo. Fue programado en C, C++.</td></tr>
<tr><td>Características:</td><td>• MySQL Workbench es el entorno integrado oficial de MySQL. Desarrollado por MySQL AB, y permite a los usuarios administrar gráficamente las bases de datos MySQL y diseñar visualmente las estructuras de las bases de datos.
• Percona Toolkit es un kit de herramientas multiplataforma para MySQL, desarrollado en Perl.[21] Percona Toolkit puede ser usado para probar que la replicación funciona correctamente, arreglar datos corruptos, automatizar tareas repetitivas y acelerar los servidores.
• Usa GNU Automake, Autoconf, y Libtool para portabilidad
• Uso de multihilos mediante hilos del kernel.
• Usa tablas en disco b-tree para búsquedas rápidas con compresión de índice
• Tablas hash en memoria temporales
• El código MySQL se prueba con Purify (un detector de memoria perdida comercial) así como con Valgrind, una herramienta GPL.
• Seguridad: ofrece un sistema de contraseñas y privilegios seguro mediante verificación basada en el host y el tráfico de contraseñas está cifrado al conectarse a un servidor.
• Soporta gran cantidad de datos. MySQL Server tiene bases de datos de hasta 50 millones de registros.</td></tr>
</table>

Ficha técnica de SGBDR:	
Nombre:	**PostgreSQL**
Requerimientos:	
Requerimientos de Hardware: Recomendados; 1gb ram, procesador qualcore para soporte en paralelos de tareas de la base de datos. **Requerimientos de software:** Esta disponibles, Windows y los derivados de GNU/Linux.	
Descripción:	PostgreSQL, también llamado Postgres, es un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto, publicado bajo la licencia PostgreSQL,[1] similar a la BSD o la MIT. Fue programado y desarrollado en lenguaje C.
Características:	- Números de precisión arbitraria. - Texto de largo ilimitado. - Figuras geométricas (con una variedad de funciones asociadas). - Direcciones IP (IPv4 e IPv6). - Bloques de direcciones estilo CIDR. - los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS. - Integridad transaccional. - Herencia de tablas. - Tipos de datos y operaciones geométricas.

Tabla Comparativa de SGBDR

	Criterios Técnicos	Criterios económicos	Criterios formación de usuario	Criterio de servicio técnico	Criterios de rendimiento	Criterios de política de la organización
Microsoft SQL Server	Posee un sólido desarrollo, por ser desarrollado principalmente para el sector empresarial, y de paga.	Pensado principalmente como una versión paga, pero hay una versión libre que ofrece funcionalidades limitadas.	Posee una excelente documentacion de todas sun funcionalidades y herramientas.	Posee herramientas verificadas para control y administración de la base de datos	Posee un rendimiento excepcional para bases de datos de hasta 5 millones de registros, con capacidad de aprovechar todos los núcleos e hilos de los procesadores modernos.	Política empresariales estándar, uso correcto de las licencias públicas, normas y leyes legales de por medio, y referencias créditos de la base de datos.
MySQL	Posee la mayor comunidad activad, y es el SGBDR mas popular, por lo que se pueden conseguir montones de funcionalidad extras y configuraciones personalizadas.	Pensado principalmente como un sistema de software libre, pero que además posee una versión paga, con funcionalidades cercanas a la versión gratis.	Al poseer la mayor comunidad activa, se encuentran la mayor cantidad de documentos explicativos e tutoriales.	Posee la mayor variedad de herramientas para el control y administración de la base de datos.	Posee una gran rendimiento para el manejo de base de datos grandes de hasta 5 millones de registros, con capacidad aprovechar el máximo potencial de los procesadores modernos.	Política empresariales estándar, uso correcto de las licencias públicas, normas y leyes legales de por medio, y referencias créditos de la base de datos.

PostgreSQL	**Es desarrollado por usuarios altruista, se sabe que es muy probable que presenten fallos a medida crece la base de datos.**	**Pensado para la comunidad libre, es gratis en todas sus funcionalidades.**	**Al ser 100% openSoftware y ser unas de las menos usadas, es un poco mas difícil conseguir la requerida documentacion**	**Posee un conjunto de herramientas útiles para el control y mantenimiento de la base de datos.**	**Posee un excelente rendimiento en bases de datos pequeñas, optimisable para bases de datos medianas, pero no está diseñada para bases de datos inmensas de múltiples capas.**	**Libre uso de software, para fines indistintos, capacidad de modificar el código original y lograr una base de datos original y estructura de acuerdo a los fines requeridos.**

Referencias

1- Fundamentos de Sistemas de Bases de Datos 5ta Edicion Ramez Elmasri LIBRO

2- Postgre-SQL-Notes-For-Professionals-ElSaber21.com

3- Fundamentos de SQL, 3ra Edición - Andy Oppel-FREELIBROS.ORG

4- http://www.mysql.com/ (Wikipedia)

5- http://www.postgresql.org/ (Wikipedia)

6- http://www.microsoft.com/sql/ (Wikipedia)