Proyecto Final BD

DITTO

7 de enero de 2023

Integrantes

- De los Cobos García Carlos Alberto
- Herrera Carrillo Cristhian
- Martínez Hernández Gerardo

Introducción

Una base de datos es una colección de datos que se organizan de modo que se pueda acceder fácilmente a esta información, además de permitir un almacenamiento más preciso, confiable y de fácil acceso, ya que es una forma estructurada de almacenamiento, De hecho, en la actualidad, las bases de datos son esenciales, ya que se implementan desde un sftware bancario hasta una investigación científica o registros gubernamentales, pero sobre todo en los sitios web o aplicaciones móviles usadas de forma cotidiana.

Pero las bases de datos no necesariamente deben tener un uso masivo, también pueden ser utilizadas por pequeñas empresas. La principal razón para esto se debe a que las bases de datos facilitan mucho el acceso a la información o datos de todos los días, vale la pena entender qué son las bases de datos y como es que estas funcionan, razón por la cual a lo largo de este proyecto, se desarrolla una base de datos funcional y sencilla paso a paso para un buen desarrollo e implementación.

El término modelo de datos describe la estructura lógica de una base de datos, que determina las reglas sobre cómo se puede organizar y manipular la información que contiene. Razón por la cual se opta por usar bases de datos, ya que cuando los datos están estructurados, eso significa que están formateados y se pueden buscar. Mientras que, los datos no estructurados carecen tanto de formato como de accesibilidad y, por lo tanto, son mucho más difíciles de analizar.

Sea cual sea el tipo de datos con os que se está trabajando, estos deben organizarse de acuerdo con un modelo, que describe los detalles sobre cómo se desea implementar la base, como los tipos de datos requeridos u otras restricciones. Por último, necesitamos alguna forma de interactuar con la base de datos para realizar las acciones deseadas, para esto se hace uso de un sistema de administración de bases de datos que es el software que hace posible que los usuarios finales creen, modifiquen y administren bases de datos, así coo también definan, almacenen, manipulen y recuperen los datos.

Objetivo

Con la finalidad de resolver todos los requerimientos propuestos por el usuario, que en este caso se trata de una mueblería, se decide a proponer una solución que consta de un Modelo Entidad Relación, el cual cuenta con las entidades que, en conjunto, creemos ayudarán a dar una resolución adecuada al problema. Continuando con dar paso al Modelo Relacional el cual, utilizando las técnicas vistas en el curso, podrán acercarnos a un Modelo Final. Además de resolver los requerimientos propuestos, tenemos como objetivo poner en práctica aquellos conceptos que se revisaron a lo largo de la asignatura.

1. Planteamiento

Una mueblería quiere digitalizar su forma de operar para evitar tener sus registros de forma física y tener una vista completa de la información de todas sus sucursales, por lo que se plantea el siguiente requerimiento:

Se debe almacenar el código de barrar, nombre, precio de venta, precio de compra, stock y fotografía de cada artículo, clasificándolos teniendo en cuenta que un artículo sólo pertenece a una categoría. Un artículo puede ser surtido por mas de un proveedor, manteniendo registro de la fecha en la que se comenzó a surtir cada artículo y datos como el rfc, razón social, dirección, teléfono y cuenta para pago. De las ventas debe tenerse registro de su folio, fecha, monto total, monto por artículo, cantidad total de artículos, cantidad por artículo, quién concretó la venta y quién cobró. Para facturación y programas de lealtad debe tener registro de datos de clientes, tales como rfc, nombre, razón social, dirección, email y teléfono. Debe tenerse una visibilidad completa de los epleados y el rol que desempeñan, por lo que debe asignarse un número de empleado(irrepetible) y tener registro de datos como el supervisor directo, rfc, curp, nombre, teléfonos, dirección, email, su fecha de ingreso y tipo de empleado(debe ser cajero, vendedor, administrativo, seguridad o limpieza; sólo puede ser un tipo). Respecto a las sucursales, debe tenerse conocimiento de su ubicación, teléfono, año de fundación y debe consoderarse que un empleado sólo trabaja en una sucursal. Adicionalmente, deben cumplirse los siguientes puntos:

- Cada que se agregue un artículo a venta, debe actualizarse los totales(por artículo, venta y cantidad de artículos), así como validar que el artículo esté disponible.
- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el por qué de la elección en ambos aspectos.
- Lista de artículos no disponibles o con stock ¡3. Sí el artículo no está disponible, debe aparecer el mensaje "No disponble". Esta lista debe almacenarse en una tabla independiente, la cual debe actualizarse cada que se desee obtener esta lista.
- De manera automática se genere una vista que contenga información necesaria para asemejarse a un ticket de venta, incluyendo folio para facturación (caracteres aleatorios irrepetibles).
- Al iniciar una venta, debe validarse que el vendedor y el cajero, pertenezcan a la misma sucursal, en caso de que no sea así, debe mostrarse un mensaje de error.

- Dada una fecha o una fecha de inicio y fecha de fin, regresar el total vendido por sucursal y la cantidad de ventas efectuadas. Si no se especifica fecha, debe tomarse el día en que se hace la consulta.

Tomar en cuenta las siguientes consideraciones:

- Puede haber distintas soluciones al problema.
- Los requerimientos enlistados anteriormente, deberán ser realizados por medio de PgSQL, con los elementos que se consideren adecuados para resolverlos.
- Considerar que no necesariamente los clientes deben estar registrados para hacer una venta, pero la venta sí debe quedar registrada.
- el folio de la vente debe tener un formato similar a "MBL-001", prefijo MBL, seguido de un guión y un número secuencial.
- Donde este presente el atributos Domicilio, está compuesto por estado, código postal, colonia, calle y número.
- La razón social en el caso de los clientes es el nombre completo y debe ser un valor default.
- Donde este presente el atributo nombre, está compuesto por nombre, apellido paterno y materno.
- La propuesta de solución debe satisfacer todo los principios de diseño, los requerimientos anteriores y agregar las restricciones que se consideren necesarias para asegurar la consistencia e integridad de la información.

Plan de trabajo

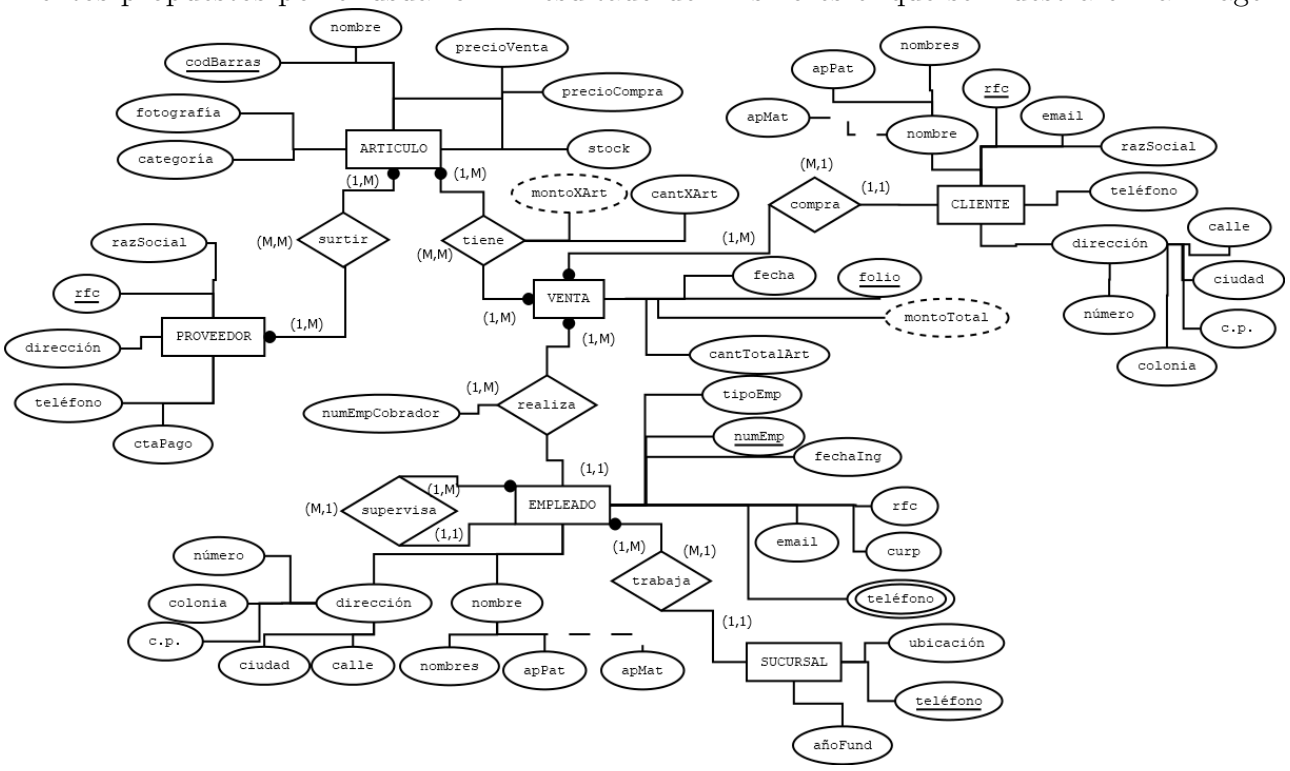
Debido a cuestiones academicas, el proyecto se llevo a cabo por solo 3 alumnos.
El proyecto se dividio en 6 secciones, de as cuales, las primeras 2 se trabajaron de manera cooperativa, mientras que las siguientes secciones se trabajaron de forma individual.

- ANÁLISIS Se hizo una junta para entender cual era nuestro punto de partida para el desarrollo del proyecto. Se analizó de que manera ibamos a trabajar y como se iba a dividir el trabajo. Fue de suma importancia ya que nos permitio realizar el proyecto de manera ordenada y concreta.
  Cabe aclarar que si surgía alguna duda entorno a lo que le correspondía a cada uno de los integrantes, nos reuniamos y se encontraba una solución.
- MODELADO Y DISEÑO. Analizamos la información o los datos que debemos almacenar, gracias a esto se comenzó el modelo entidad-relación. Esta seccion es fundamental para obtener una buena base de datos, que sea lo mas óptima posible, posteriormente se creó un modelo relacional, para observar si podíamos hacer mejoras, este fue el punto de partida para poder realizar los puntos a seguir para la manipulación de nuestros datos, de acuerdo a lo que se hizo en esta sección.
- REQUERIMIENTOS. Aqui pudimos observar las restricciones, operaciones y consultas necesarias para la manipulación de los datos. Fueron implementadas, aplicadas y corregidas dentro de nuestra base de datos por De los Cobos García Carlos Alberto.

- CREACIÓN DE TABLAS E INSERCIÓN DE DATOS. Fue una seccion sencilla, Martínez Hernández Gerardo y Herrera Carrillo Cristhian realizaron esta sección. A pesar de tener una cantidad considerable de tablas, se fue dando ayuda grupal para satisfacer esta sección.
- INTERFAZ GRÁFICA. Es importante esta sección por que nos permite observar de manera mas visual, mas amigable los datos de la base de datos. La interfaz gráfica se llevo a cabo por De los Cobos García Carlos Alberto.
- DOCUMENTACIÓN. la documentación fue una sección un poco extensa, pues se tenia que explicar paso por paso, parte por parte. La documentación fue realizada por Herrera Carrillo Cristhian. Esta seccion por permitio recopilar de forma concisa como fue la elaboración de cada etapa del proyecto. Ver detalladamente cada paso que se siguio para la creación de la base de datos es importante. De la forma mas resumida posible se presentó en este documento el desarrollo de este proyecto.
- HERRAMIENTAS.
  - Postgres-SQL: Esta fue la plataforma principal de este proyecto, hablamos de un gestos de base de datos, lo que nos permite almacenar, recuperar, definir y administrar los datos de nuestra base de datos.
  - Github: en está plataforma se presentó el trabajo final hacia el profesor.
  - ZOOM: en esta plataforma de juntas, nos reuníamos para ver avances, resolver dudas y ayudarnos si es que tuvimos problemas en algun punto.

Diseño

Para comenzar con esta sección, tenemos el MER que se realizó basándonos en los requerimientos propuestos por el usuario. El resultado del mismo es el que se muestra en la imagen:



Como se aprecia en el modelo, realizamos varias entidades:

1. Artículo: Representa a los artículos con los que cuenta la mueblería, tiene como atributo clave un código de barras, además incluye una fotografía del producto, su nombre, precio de venta y compra, la cantidad en almacén y la categoría a la que pertenece dicho artículo.

2. Proveedor: Esta entidad refiere a los proveedores de los artículos que ofrece la mueblería, como atributo clave cuenta con el rfc del mismo proveedor, además de tener razón social, la dirección (estado, código postal, colonia, calle y número), un teléfono de contacto y una cuenta para pagos.

3. Venta: Con esta entidad se manejarán las ventas del lugar, por ello cada venta tendrá su folio, la fecha en que se realizó, la cantidad total de artículos que tiene y, finalmente el monto total de dicha venta (valor calculado).

4. Cliente: Como se requiere almacenar información de algunos clientes, esta entidad cuenta con el rfc del cliente como atributo clave, el nombre completo del cliente (el apellido materno siendo opcional), la razón social, un teléfono de contacto y la dirección del mismo.

5. Empleado: La entidad tiene como atributo clave un número de empleado, el tipo de empleado (cajero, vendedor, administrativo, seguridad o limpieza), la fecha en la que ingresó a laborar, el rfc, curp, email, puede contar con más de un teléfono, el nombre completo y la dirección del empleado.

6. Sucursal: Esta entidad, se decidió colocar como atributo clave el teléfono de dicha sucursal, cuenta además con la ubicación de la sucursal y el año en que fue fundada.

Pasando a las relaciones entre entidades, destacamos:

1. Surtir: Esta relación se da entre Artículo y Proveedor, indica que un proveedor surte muchos artículos y que un artículo puede ser surtido por muchos proveedores, por ello la cardinalidad es de muchos a muchos.

2. Tiene: La relación "tiene"se da entre Artículo y Venta, nos dice que una venta puede tener varios artículos y, a su vez, un artículo puede pertenecer a varias ventas. Su cardinalidad será de muchos a muchos. Cabe resaltar que esta relación tiene algunos atributos propios, como lo es cantidad por artículo y el monto por artículo (que es un valor calculado).

3. Realiza: Con esta relación juntamos a Empleado con Venta, pues en una venta participa un empleado quien la logró, pero un empleado puede realizar muchas ventas. Por ello la cardinalidad es uno a muchos. Aquí el atributo de la relación será el número de empleado de aquél empleado que cobró la venta.

4. Compra: Con esta relación juntamos Cliente con Venta, cuya cardinalidad es uno a muchos, puesto que un cliente puede realizar muchas compras, pero una venta sólo se puede concretar por un cliente.

5. Trabaja: Esta relación se da entre Empleado y Sucursal, siendo su cardinalidad uno a muchos, puesto que en una sucursal pueden trabajar muchos empleados, pero un empleado sólo puede trabajar en una sucursal.

6. Supervisa: Dado que cada empleado tiene un supervisor, que a su vez es un empleado, se necesita crear una relación recursiva. La cual es de cardinalidad uno a muchos, siendo que un empleado sólo puede tener un supervisor, pero un supervisor puede tener muchos empleados a su cargo.

Coninuamos ahora con el Modelo Relacional o Modelo Intermedio, el cual está basado en el MER realizado en la parte previa. El cual queda de la siguiente manera:
Utilizando las diferentes formas de realizar el MR, sabiendo que al tener relaciones muchos a muchos se crea una tabla de detalle; mientras que al tener una relación uno a muchos, la llave primaria del 'uno' se propaga como llave foránea al 'muchos', apreciamos las tablas.

Para la tabla ARTICULO vemos que el campo del codigo de barras es la llave primaria, además de contar con los campos definidos en el MER:

- ARTICULO: codBarras varchar (12)(PK), fotografia BYTEA, categoría varchar(15), nombre varchar(15), precioVenta float, precioCompra float, stock smallint;

Continuamos con PROVEEDOR donde el rfc del mismo es la llave primaria y aparecen los campos definidos previamente:

- PROVEEDOR: rfcProv varchar(13)(PK), razSocialProv varhcar(150), estado varchar(50), calle varchar(50), colonia varchar(50), número smallint, cp varchar (5), teléfonoProv varchar(10), ctaPago varchar(20).

En esta tabla SURTIR se encuentra la relacion entre ARTICULO y PROVEEDOR, por ello se crea una nueva tabla cuya llave primaria es una llave compuesta por las llaves primarias de ambas tablas, que será llaves foráneas:

- SURTIR: [codBarras varchar(12)(FK), rfcProv varhcar(13)(FK)] (PK)

para VENTA tenemos que su llave primaria es el folio, el cual es una cadena de 7 caracteres, contando tambien con los demás campos definidos en el MER; destacando el rfc del cliente que será una llave foránea puesto que el cliente participa en una venta(su PK se propagó a VENTA), sin embargo, es NULL, debido a que no es necesario que el cliente esté registrado para que pueda comprar, vemos también que el monto totla de la venta será un valor calculado:

- VENTA: folio varchar(7)(PK), fecha timestamp, canTotalArt smallint, montoTotal float(C), rfcCliente varchar(13) (FK) NULL

Siguiendo. la tabla CLIENTE nos muestra el RFC del cliente como llave primaria, además de los campos que fueron definidos:

- CLIENTE: rfcCliente varchar(13) (PK), nombre(s) varchar(110), apPat varchar(50), apMat varchar(50) NULL, email varchar(100), razSocialCliente varchar(150), telefonoCliente varchar(12), estado varchar(50), calle varchar(50), colonia varchar(50), numero smallint, cp varchar(5)

La tabla TIENE es la relacion entre VENTA y ARTICULO, que tiene como PK la compuesta por folio y el código de barras, además de contar con los campos de la cantidfad de determinado articulo y el monto por cada uno de los artículos que contiene la venta:

- TIENE: [folio varchar(7) (FK), codBarras varchar(12)(FK)](PK), cantXARt smallint, montoXArt float(C)

Continuando con EMPLEADO, tiene como PK un número de empleado de 5 caracteres, teniendo los campos definidos en el MER, pero además cuenta con un campo para el supervisor del empleado, el cual puede ser nulo (en el caso de los supervisores) y es FK al ser una relacion recursiva. De igual manera tiene el teléfono de la sucursal en la que labora, que es una FK:

- EMPLEADO: numEmp varchar(5) (PK), fechaIng date, timpoEmp varchar(50),rfcEmp varchar(13), curp varchar(18), email varchar(100), nombres varchar(100), apPatEmp varchar(50), apMatEmp varchar(50) NULL, estadoEmp varchar(50), calleEmp varchar(50), coloniaEmp varchar(50), numeroEmp smallint, cpEmp varchar(50), numEmpSup varchar(5) (FK), telefonoSuc varchar(10)(FK)

En TELEFONO se tiene una PK compuesta por el número de empleado que es FK en conjunto con el teléfono propio del empleado, esto dado que un empleado puede tener varios teléfonos:

- TELEFONO: [numEmp varchar(5)(FK), telégono varhcar(10)](PK)

en REALIZA tenemos una PK compuesta por el folio de la venta y el número del empleado que realizó dicha venta, pero también cuenta con un campo designado para el número del empleado que la cobró, este campo como FK:

- REALIZA: [folio varchar(7)(FK), numEmp varchar(5) (FK)](PK), numEmpCobrador varchar(5) (FK)

Finalmente, la tala SUCURSAL tiene como PK el teléfono de la surcursal además de la dirección de la misma y el año en que fue fundada:

- SUCURSAL: telefonoSuc varchar(10)(PK), estadoSuc varchar(50), calleSuc varchar(50), coloniaSuc varchar(50), numeroSuc smallint, cpSuc varchar(5), añoFun smallint

Vamos ahora a llevar a cabo el proceso de normalización de las tablas que hemos propuesto. Obteniendo lo siguiente:
Para ARTICULO tenemos que ya está en 1FN. También cumple con 2FN, para 3FN separamos 'categoria' en otra tabla. Por lo que el id de categoria ahora sera FK en esta tabla. Resultando en dos tablas:

- ARTICULO: codBarras varchar(12)(PK), fotografia BYTEA, idCat smallint (FK), nombre varchar(15), precioVenta float, precioCompra float, stock smallint

- CATEGORIA: idCat smallint(PK), categoria varchar(15)

Para PROVEEDOR se tiene que está en 2FN. para la 3FN separamos 'estado' en otra tabla, de modo que id del estadp será FK:

- PROVEEDOR: rfcProv varchar(13)(PK), razSocialProv varhcar(150),idEdo smallint (FK), calle varchar(50), colonia varchar(50), número smallint, cp varchar (5), teléfonoProv varchar(10), ctaPago varchar(20)

- ESTADO: idEdo smallint (PK), estado varchar(50)

Para SURTIR se tiene que ya está en 3FN:

- SURTIR: [codBarras varchar(12) (FK), rfcprov varchar(13)(FK)](PK)

Con VENTA tenemos que igualmente ya se encuentra en la 3FN:

- VENTA: folio varchar(7)(PK), fecha timestamp, cantTotalArt smallint, montoTotal float (C), rfcCliente varchar (13) (FK) NULL.

Para CLIENTE tenemos que ya está en 1FN, igual en 2FN. Para la 3FN podemos separar estado en otra tabla:

- CLIENTE: rfcCliente varchar(13) (PK), nombres varchar(100), apPat varchar(50), apMat varchar(50) NULL, email varchar(100), razSocialCliente varchar (150), telefonoCliente varchar(12), idEdo smallint (FK), calleCliente varchar(50), coloniaCliente varchar(50), numeroCLiente smallint, cpCLiente varchar(50)

con TIENE vemos que cumple 1FN, para 2FN, aunque hay una llave compuesta, no existen dependencias funcionales, por lo que si cumple con 2FN. De igual manera cumple con la 3FN:

- TIENE: [folio varchar(7)(FK), codBarras varchar(12)(FK)](PK), cantXArt smallint, montoXArt float(C)

Para EMPLEADO vemos que cumple con 1FN, tambien con 2FN. Para la 3FN separamos 'estado' y 'tipo' en otras tablas. Así que el id de tipo para como FK a EMPLEADO. Quedando de la siguiente manera:

- EMPLEADO: numEmp varchar(5) (PK), fechaIng date, idTipo smallint (FK),rfcEmp varchar(13), curp varchar(18), email varchar(100), nombres varchar(100), apPatEmp varchar(50), apMatEmp varchar(50) NULL,idEdo smallint(FK), calleEmp varchar(50), coloniaEmp varchar(50), numeroEmp smallint, cpEmp varchar(50), numEmpSup varchar(5) (FK), telefonoSuc varchar(10)(FK).

- TIPO: idTipo smallint(PK), tipo varchar(50)

con TELEFONO se cumple 3FN:

- TELEFONO: [numEmp varchar (5)(FK), telefono varchar(10)](PK)

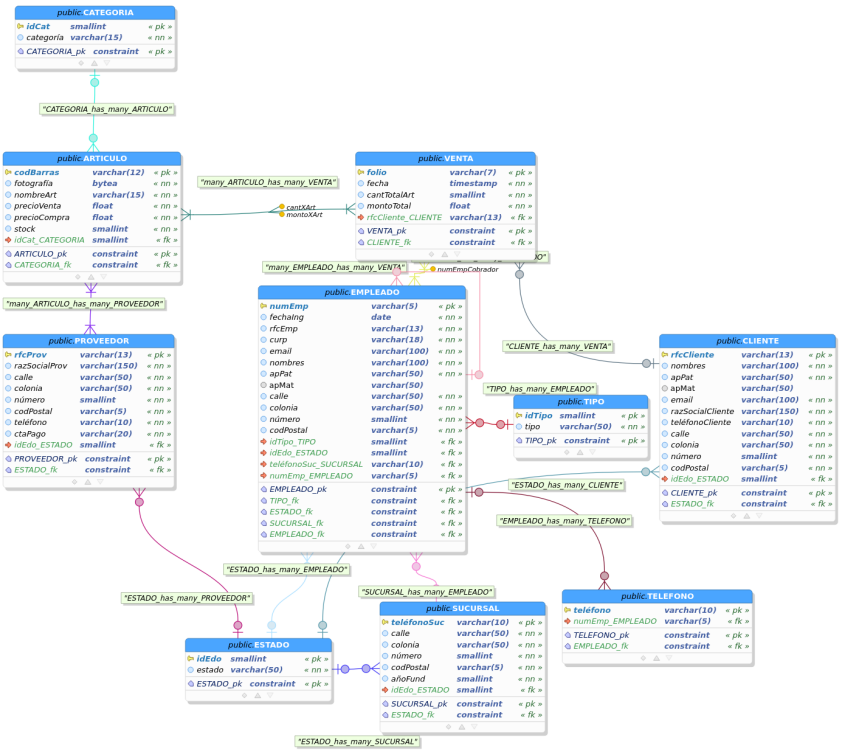
Con REALIZA tenemos que a pesar de tener llave compuesta cumple con 2FN. Tambien con la 3FN:

- REALIZA: [folio varchar(7)(FK), numEmp varchar(5)(FK)](PK), numEmpCobrador varchar(5)(FK)

Finalmente para SUCURSAL se cumple 2FN. Para la 3FN separamos 'estado' en otra tabla:

- SUCURSAL: telefonoSuc varchar(10)(PK), idEdo smallint(FK), calleSuc varchar(50), coloniaSuc varchar(50), numeroSuc smallint, cpSuc varchar (5), añoFun smallint

Utilizando PGModeler, MR Final nos quedaría de la siguiente manera:



Como se ve PgModeler en automático coloca los respectivos CONSTRAINT´S para las PK's y las FK's correspondientes. De igual forma es posible observar el código SQL para la creación de las tablas el cual utilizamos como guía para llevar a cabo dicho proceso.

Implementación

```
-- Database generated with pgModeler (PostgreSQL Database Modeler).
-- pgModeler version: 0.9.4
-- PostgreSQL version: 14.0
-- Project Site: pgmodeler.io
-- Model Author: ---

-- Database creation must be performed outside a multi lined SQL file.
-- These commands were put in this file only as a convenience.
--
-- object: new_database | type: DATABASE --
-- DROP DATABASE IF EXISTS new_database;
CREATE DATABASE new_database;
-- ddl-end --

-- object: public."CATEGORIA" | type: TABLE --
-- DROP TABLE IF EXISTS public."CATEGORIA" CASCADE;
CREATE TABLE public."CATEGORIA" (
        "idCat" smallint NOT NULL,
        "categoría" varchar(15) NOT NULL,
        CONSTRAINT "CATEGORIA_pk" PRIMARY KEY ("idCat")
);
-- ddl-end --
ALTER TABLE public."CATEGORIA" OWNER TO postgres;
-- ddl-end --

-- object: public."ARTICULO" | type: TABLE --
-- DROP TABLE IF EXISTS public."ARTICULO" CASCADE;
CREATE TABLE public."ARTICULO" (
        "codBarras" varchar(12) NOT NULL,
        "fotografía" bytea NOT NULL,
        "nombreArt" varchar(15) NOT NULL,
        "precioVenta" float NOT NULL,
        "precioCompra" float NOT NULL,
        stock smallint NOT NULL,
        "idCat_CATEGORIA" smallint,
        CONSTRAINT "ARTICULO_pk" PRIMARY KEY ("codBarras")
);
-- ddl-end --
ALTER TABLE public."ARTICULO" OWNER TO postgres;
-- ddl-end --

CREATE TABLE public."PROVEEDOR" (
        "rfcProv" varchar(13) NOT NULL,
        "razSocialProv" varchar(150) NOT NULL,
        calle varchar(50) NOT NULL,
        colonia varchar(50) NOT NULL,
        "número" smallint NOT NULL,
        "codPostal" varchar(5) NOT NULL,
        "teléfono" varchar(10) NOT NULL,
        "ctaPago" varchar(20) NOT NULL,
        "idEdo_ESTADO" smallint,
        CONSTRAINT "PROVEEDOR_pk" PRIMARY KEY ("rfcProv")
);
-- ddl-end --
ALTER TABLE public."PROVEEDOR" OWNER TO postgres;
-- ddl-end --

-- object: public."ESTADO" | type: TABLE --
-- DROP TABLE IF EXISTS public."ESTADO" CASCADE;
CREATE TABLE public."ESTADO" (
        "idEdo" smallint NOT NULL,
        estado varchar(50) NOT NULL,
        CONSTRAINT "ESTADO_pk" PRIMARY KEY ("idEdo")
);
-- ddl-end --
ALTER TABLE public."ESTADO" OWNER TO postgres;
-- ddl-end --

-- object: public."VENTA" | type: TABLE --
-- DROP TABLE IF EXISTS public."VENTA" CASCADE;
CREATE TABLE public."VENTA" (
        folio varchar(7) NOT NULL,
        fecha timestamp NOT NULL,
        "cantTotalArt" smallint NOT NULL,
        "montoTotal" float NOT NULL,
        "rfcCliente_CLIENTE" varchar(13),
        CONSTRAINT "VENTA_pk" PRIMARY KEY (folio)
);
-- ddl-end --
ALTER TABLE public."VENTA" OWNER TO postgres;
-- ddl-end --

-- object: public."EMPLEADO" | type: TABLE --
-- DROP TABLE IF EXISTS public."EMPLEADO" CASCADE;
CREATE TABLE public."EMPLEADO" (
        "numEmp" varchar(5) NOT NULL,
        "fechaIng" date NOT NULL,
        "rfcEmp" varchar(13) NOT NULL,
        curp varchar(18) NOT NULL,
        email varchar(100) NOT NULL,
```

```
        nombres varchar(100) NOT NULL,
        "apPat" varchar(50) NOT NULL,
        "apMat" varchar(50),
        calle varchar(50) NOT NULL,
        colonia varchar(50) NOT NULL,
        "número" smallint NOT NULL,
        "codPostal" varchar(5) NOT NULL,
        "idTipo_TIPO" smallint,
        "idEdo_ESTADO" smallint,
        "teléfonoSuc_SUCURSAL" varchar(10),
        CONSTRAINT "EMPLEADO_pk" PRIMARY KEY ("numEmp")
);
-- ddl-end --
ALTER TABLE public."EMPLEADO" OWNER TO postgres;
-- ddl-end --

-- object: public."SUCURSAL" | type: TABLE --
-- DROP TABLE IF EXISTS public."SUCURSAL" CASCADE;
CREATE TABLE public."SUCURSAL" (
        "teléfonoSuc" varchar(10) NOT NULL,
        calle varchar(50) NOT NULL,
        colonia varchar(50) NOT NULL,
        "número" smallint NOT NULL,
        "codPostal" varchar(5) NOT NULL,
        "añoFund" smallint NOT NULL,
        "idEdo_ESTADO" smallint,
        CONSTRAINT "SUCURSAL_pk" PRIMARY KEY ("teléfonoSuc")
);
-- ddl-end --
ALTER TABLE public."SUCURSAL" OWNER TO postgres;
-- ddl-end --

-- object: public."TIPO" | type: TABLE --
-- DROP TABLE IF EXISTS public."TIPO" CASCADE;
CREATE TABLE public."TIPO" (
        "idTipo" smallint NOT NULL,
        tipo varchar(50) NOT NULL,
        CONSTRAINT "TIPO_pk" PRIMARY KEY ("idTipo")
);
-- ddl-end --
ALTER TABLE public."TIPO" OWNER TO postgres;
-- ddl-end --

-- object: public."CLIENTE" | type: TABLE --
-- DROP TABLE IF EXISTS public."CLIENTE" CASCADE;
CREATE TABLE public."CLIENTE" (
        "rfcCliente" varchar(13) NOT NULL,
        nombres varchar(100) NOT NULL,
        "apPat" varchar(50) NOT NULL,
        "apMat" varchar(50),
        email varchar(100) NOT NULL,
        "razSocialCliente" varchar(150) NOT NULL,
        "teléfonoCliente" varchar(10) NOT NULL,
        calle varchar(50) NOT NULL,
        colonia varchar(50) NOT NULL,
        "número" smallint NOT NULL,
        "codPostal" varchar(5) NOT NULL,
        "idEdo_ESTADO" smallint,
        CONSTRAINT "CLIENTE_pk" PRIMARY KEY ("rfcCliente")
);
-- ddl-end --
ALTER TABLE public."CLIENTE" OWNER TO postgres;
-- ddl-end --

-- object: public."TELEFONO" | type: TABLE --
-- DROP TABLE IF EXISTS public."TELEFONO" CASCADE;
CREATE TABLE public."TELEFONO" (
        "teléfono" varchar(10) NOT NULL,
        "numEmp_EMPLEADO" varchar(5),
        CONSTRAINT "TELEFONO_pk" PRIMARY KEY ("teléfono")
);
-- ddl-end --
ALTER TABLE public."TELEFONO" OWNER TO postgres;
-- ddl-end --

-- object: "CATEGORIA_fk" | type: CONSTRAINT --
-- ALTER TABLE public."ARTICULO" DROP CONSTRAINT IF EXISTS "CATEGORIA_fk" CASCADE;
ALTER TABLE public."ARTICULO" ADD CONSTRAINT "CATEGORIA_fk" FOREIGN KEY ("idCat_CATEGORIA")
REFERENCES public."CATEGORIA" ("idCat") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: "ESTADO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."PROVEEDOR" DROP CONSTRAINT IF EXISTS "ESTADO_fk" CASCADE;
ALTER TABLE public."PROVEEDOR" ADD CONSTRAINT "ESTADO_fk" FOREIGN KEY ("idEdo_ESTADO")
REFERENCES public."ESTADO" ("idEdo") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: "CLIENTE_fk" | type: CONSTRAINT --
-- ALTER TABLE public."VENTA" DROP CONSTRAINT IF EXISTS "CLIENTE_fk" CASCADE;
ALTER TABLE public."VENTA" ADD CONSTRAINT "CLIENTE_fk" FOREIGN KEY ("rfcCliente_CLIENTE")
REFERENCES public."CLIENTE" ("rfcCliente") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: "TIPO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."EMPLEADO" DROP CONSTRAINT IF EXISTS "TIPO_fk" CASCADE;
ALTER TABLE public."EMPLEADO" ADD CONSTRAINT "TIPO_fk" FOREIGN KEY ("idTipo_TIPO")
REFERENCES public."TIPO" ("idTipo") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --
```

```
-- object: "ESTADO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."EMPLEADO" DROP CONSTRAINT IF EXISTS "ESTADO_fk" CASCADE;
ALTER TABLE public."EMPLEADO" ADD CONSTRAINT "ESTADO_fk" FOREIGN KEY ("idEdo_ESTADO")
REFERENCES public."ESTADO" ("idEdo") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: "SUCURSAL_fk" | type: CONSTRAINT --
-- ALTER TABLE public."EMPLEADO" DROP CONSTRAINT IF EXISTS "SUCURSAL_fk" CASCADE;
ALTER TABLE public."EMPLEADO" ADD CONSTRAINT "SUCURSAL_fk" FOREIGN KEY ("teléfonoSuc_SUCURSAL")
REFERENCES public."SUCURSAL" ("teléfonoSuc") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: "ESTADO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."SUCURSAL" DROP CONSTRAINT IF EXISTS "ESTADO_fk" CASCADE;
ALTER TABLE public."SUCURSAL" ADD CONSTRAINT "ESTADO_fk" FOREIGN KEY ("idEdo_ESTADO")
REFERENCES public."ESTADO" ("idEdo") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: "ESTADO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."CLIENTE" DROP CONSTRAINT IF EXISTS "ESTADO_fk" CASCADE;
ALTER TABLE public."CLIENTE" ADD CONSTRAINT "ESTADO_fk" FOREIGN KEY ("idEdo_ESTADO")
REFERENCES public."ESTADO" ("idEdo") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: "EMPLEADO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."TELEFONO" DROP CONSTRAINT IF EXISTS "EMPLEADO_fk" CASCADE;
ALTER TABLE public."TELEFONO" ADD CONSTRAINT "EMPLEADO_fk" FOREIGN KEY ("numEmp_EMPLEADO")
REFERENCES public."EMPLEADO" ("numEmp") MATCH FULL
ON DELETE SET NULL ON UPDATE CASCADE;
-- ddl-end --

-- object: public."many_ARTICULO_has_many_PROVEEDOR" | type: TABLE --
-- DROP TABLE IF EXISTS public."many_ARTICULO_has_many_PROVEEDOR" CASCADE;
CREATE TABLE public."many_ARTICULO_has_many_PROVEEDOR" (
        "codBarras_ARTICULO" varchar(12) NOT NULL,
        "rfcProv_PROVEEDOR" varchar(13) NOT NULL,
        CONSTRAINT "many_ARTICULO_has_many_PROVEEDOR_pk" PRIMARY KEY ("codBarras_ARTICULO","rfcProv_PROVEEDOR")
);
-- ddl-end --

-- object: "ARTICULO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."many_ARTICULO_has_many_PROVEEDOR" DROP CONSTRAINT IF EXISTS "ARTICULO_fk" CASCADE;
ALTER TABLE public."many_ARTICULO_has_many_PROVEEDOR" ADD CONSTRAINT "ARTICULO_fk" FOREIGN KEY ("codBarras_ARTICULO")
REFERENCES public."ARTICULO" ("codBarras") MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: "PROVEEDOR_fk" | type: CONSTRAINT --
-- ALTER TABLE public."many_ARTICULO_has_many_PROVEEDOR" DROP CONSTRAINT IF EXISTS "PROVEEDOR_fk" CASCADE;
ALTER TABLE public."many_ARTICULO_has_many_PROVEEDOR" ADD CONSTRAINT "PROVEEDOR_fk" FOREIGN KEY ("rfcProv_PROVEEDOR")
REFERENCES public."PROVEEDOR" ("rfcProv") MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: public."many_ARTICULO_has_many_VENTA" | type: TABLE --
-- DROP TABLE IF EXISTS public."many_ARTICULO_has_many_VENTA" CASCADE;
CREATE TABLE public."many_ARTICULO_has_many_VENTA" (
        "codBarras_ARTICULO" varchar(12) NOT NULL,
        "folio_VENTA" varchar(7) NOT NULL,
        "cantXArt" smallint NOT NULL,
        "montoXArt" float NOT NULL,
        CONSTRAINT "many_ARTICULO_has_many_VENTA_pk" PRIMARY KEY ("codBarras_ARTICULO","folio_VENTA")
);
-- ddl-end --

-- object: "ARTICULO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."many_ARTICULO_has_many_VENTA" DROP CONSTRAINT IF EXISTS "ARTICULO_fk" CASCADE;
ALTER TABLE public."many_ARTICULO_has_many_VENTA" ADD CONSTRAINT "ARTICULO_fk" FOREIGN KEY ("codBarras_ARTICULO")
REFERENCES public."ARTICULO" ("codBarras") MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: "VENTA_fk" | type: CONSTRAINT --
-- ALTER TABLE public."many_ARTICULO_has_many_VENTA" DROP CONSTRAINT IF EXISTS "VENTA_fk" CASCADE;
ALTER TABLE public."many_ARTICULO_has_many_VENTA" ADD CONSTRAINT "VENTA_fk" FOREIGN KEY ("folio_VENTA")
REFERENCES public."VENTA" ("folio") MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: "VENTA_fk" | type: CONSTRAINT --
-- ALTER TABLE public."many_ARTICULO_has_many_VENTA" DROP CONSTRAINT IF EXISTS "VENTA_fk" CASCADE;
ALTER TABLE public."many_ARTICULO_has_many_VENTA" ADD CONSTRAINT "VENTA_fk" FOREIGN KEY ("cantXArt")
REFERENCES public."VENTA" ("cantTotalArt") MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: "VENTA_fk" | type: CONSTRAINT --
-- ALTER TABLE public."many_ARTICULO_has_many_VENTA" DROP CONSTRAINT IF EXISTS "VENTA_fk" CASCADE;
ALTER TABLE public."many_ARTICULO_has_many_VENTA" ADD CONSTRAINT "VENTA_fk" FOREIGN KEY ("montoXArt")
REFERENCES public."VENTA" ("montoTotal") MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: public."many_EMPLEADO_has_many_VENTA" | type: TABLE --
-- DROP TABLE IF EXISTS public."many_EMPLEADO_has_many_VENTA" CASCADE;
CREATE TABLE public."many_EMPLEADO_has_many_VENTA" (
        "numEmp_EMPLEADO" varchar(5) NOT NULL,
        "folio_VENTA" varchar(7) NOT NULL,
        "numEmpCobrador" varchar(5) NOT NULL,
```

```
        CONSTRAINT "many_EMPLEADO_has_many_VENTA_pk" PRIMARY KEY ("numEmp_EMPLEADO","folio_VENTA")
);
-- ddl-end --

-- object: "EMPLEADO_fk" | type: CONSTRAINT --
-- ALTER TABLE public."many_EMPLEADO_has_many_VENTA" DROP CONSTRAINT IF EXISTS "EMPLEADO_fk" CASCADE;
ALTER TABLE public."many_EMPLEADO_has_many_VENTA" ADD CONSTRAINT "EMPLEADO_fk" FOREIGN KEY ("numEmp_EMPLEADO")
REFERENCES public."EMPLEADO" ("numEmp") MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --

-- object: "VENTA_fk" | type: CONSTRAINT --
-- ALTER TABLE public."many_EMPLEADO_has_many_VENTA" DROP CONSTRAINT IF EXISTS "VENTA_fk" CASCADE;
ALTER TABLE public."many_EMPLEADO_has_many_VENTA" ADD CONSTRAINT "VENTA_fk" FOREIGN KEY ("folio_VENTA")
REFERENCES public."VENTA" (folio) MATCH FULL
ON DELETE RESTRICT ON UPDATE CASCADE;
-- ddl-end --
```

Presentación

La función 'conexion()' almacena en varias variables los datos necesarios para conectarse a la base de datos utilizada. Después se guarda la base junto con la función "pg:connectçuyos parámetros son estos datos que se requieren para la conexión. Se ha puesto dentro de comentarios una sentencia para probar si se lograba conectar a la base. Finalmente la çonexion()regresa la variable que contiene la base de datos, lista para ser utilizada por la interfaz.

```php
<?php
3 references
function conexion(){
    $host = "host=localhost";
    $port = "port=5432";
    $dbname = "dbname=postgres";
    $user = "user=postgres";
    $password = "password=123";

    $db = pg_connect("$host $port $dbname $user $password");

    /*if($db){
        echo "exito";
    }else{
        echo "error";
    }*/

    return $db;
}
//echo conexion();
?>
```

Conclusiones

- De los Cobos García Carlos Alberto: Al momento de realizar este trabajo, se tuvo que reforzar mucho los conocimientos vistos a lo largo del curso, además de comprender algunos nuevos. No fue nada sencillo trasladar a un caso real los conceptos vistos. Sin embargo creo que es una gran experiencia este proyecto. Considero que se cumplieron los objetivos.

- Herrera Carrillo Cristhian: este proyecto sirvió para implementar todos los conceptos vistos en clase para la creación de una buena base de datos, desde el modelos entidad-relación, el modelo relacional, la representación intermedia, la creación de tablas e inserción de datos en el manejadores hasta la creación de una interfaz graficas para que el usuario final observe como esta trabajando la base de datos.

Sin duda alguna es un proyecto bastante completo y complejo pues si abarca todo lo visto en clase, y podemos concluir que se nos proporcionaron las bases y conocimientos adecuado para una buena creación y sobre todo para la implementación de una base de datos utilizando herramientas útiles para el desarrollo.

- Martínez Hernández Gerardo: Se dificultaron varias cosas que al ser vistas teoricamente ya al momento de implementarlas fue mas complejo, de igual manera yo tuve el error con mis servidores ya que despues de hacer algunos cambios, que pude levantar una base de datos que ya tenia, y eso me consumió bastante tiempo el detectar errores, también me ayude de varios videos de youtube.