

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA



Bases de datos

Proyecto Final

Profesor: Fernando Arreola Franco

Alumnos:

Carrasco Ruiz Alan Uriel
González López David
Jimenez Cervantes Angel Mauricio
Reyes Arroyo Pablo Antonio
Reyes Romero Luis Fernando
Vázquez Reyes Sebastián

GRUPO: 1

10 DE JUNIO DEL 2023

1 Introducción

Las bases de datos son una de las herramientas más eficientes actualmente, ya que podemos recopilar y organizar información de un tema o un contexto específico. Asimismo, con una base de datos somos capaces de almacenar una cantidad enorme de datos como sea necesaria, además de que podemos compartir la información de una forma más segura y podemos acceder rápidamente a los datos que necesitemos con las herramientas adecuadas.

Por otra parte, a lo largo del curso hemos aprendido a como desarrollar una base de datos partiendo de un análisis de requerimientos hasta llegar al modelo físico de la base de datos. De igual forma, logramos ver las ventajas que tiene el manejo de la información de cualquier tema cuando se encuentra de manera más organizada y detallada. Bajo este contexto, surge la idea de crear una base de datos como propuesta a una problemática dada, con el objetivo de poner nuestros conocimientos en práctica.

Dicho lo anterior, la situación que se nos presenta es la siguiente: una cadena de papelerías desea innovar su manera de trabajo, implementando una base de datos para llevar un mejor control del inventario disponible, la información y productos que suministran los proveedores, así como también los datos de los clientes y las ventas que se llevan a cabo. Entonces, nuestro equipo de trabajo se encargará de la creación de la base de datos cumpliendo con todos los requerimientos que nos proporcione el cliente. Además, la solución del problema anteriormente presentado se hará mediante un plan de trabajo conformado por 2 etapas: la etapa de diseño y la etapa de implementación, las cuales serán descritas a lo largo de este documento.

En la etapa de diseño nos ocuparemos del nivel conceptual, lógico y físico de la base de datos: El modelo Entidad-Relación propuesto, su respectivo Modelo-Relacional-Intermedio, después se hará el proceso de normalización para llegar al Modelo Relacional Final propuesto y la presentación del nivel físico del Modelo Relacional Final. En la etapa de implementación describiremos el funcionamiento y la creación de las funciones necesarias para que el usuario sea capaz de obtener la información de las ventas, compras, clientes y proveedores que necesite (es decir, los requerimientos solicitados cubiertos).

Por último, al final del documento se presentaran las conclusiones que formulará cada miembro de nuestro equipo de trabajo, donde se describirán las dificultades y aciertos que hubo en cada etapa del proyecto

2 Plan de Trabajo

El presente plan de trabajo busca describir el conjunto de esfuerzos necesarios para la realización y creación de la base de datos requerida, este mismo plan se dividirá de forma general en cinco fases distintas agrupadas en 2 etapas.

Durante la etapa de diseño, nos dedicamos, en primera instancia, al análisis de requerimientos necesario para interpretar las necesidades del cliente y proponer una solución, en esta etapa todos los miembros del equipo aportaron en base a sus conocimientos, ideas y propuestas de solución con respecto a los requerimientos planteados, por lo que hubo una participación total del equipo en la misma.

Posteriormente se aborda al diseño conceptual de la base que conlleva la formulación de un Modelo Entidad Relación e identificación de los atributos y entidades que jugaran un rol en la base. En esta etapa fueron los integrantes González López David, Jiménez Cervantes Ángel Mauricio y Reyes Arrollo Pablo Antonio quienes en primera instancia diseñaron el primer modelo MER de la base, el cual sirvió para que posteriormente los integrantes Carrasco Ruiz Alan Uriel, Vázquez Reyes Sebastián y Reyes Romero Luis Fernando, realizasen un refinamiento del diseño planteado.

Después, dimos inicio a la creación del diseño lógico, es decir, la creación de un Modelo Relacional, definición del dominio de los atributos y el proceso de normalización. Los integrantes Carrasco Ruiz Alan Uriel, Vázquez Reyes Sebastián y Reyes Romero Luis Fernando, fueron los encargados de formular el modelo Relacional Intermedio y Relacional Final, y también, de reajustar los modelos MER y MRI al momento de que en etapas posteriores se requirieron cambios menores en el diseño de la base.

Por ultimo, en la fase de diseño físico de la base de datos; (creación de tablas e inserción de registros), y la etapa de implementación, mostraremos la construcción de objetos y funciones necesarias para cubrir los requerimientos solicitados por el usuario. En estas ultimas fases del proyecto, fueron los integrantes Carrasco Ruiz Alan Uriel, González López David, Jiménez Cervantes Ángel Mauricio y Reyes Arrollo Pablo Antonio quienes diseñaron igualmente las diversas funcionalidades que resolviesen las necesidades definidas en los requisitos del proyecto, junto con un par de funcionalidades extra que, en base a nuestra lógica y entendimiento del problema, tenían sentido y facilitaban el manejo de la base.

Como extra, también nos dedicamos a crear una pequeña interfaz para conectar la base de datos con una pagina web.

A manera de resumen, se muestra en la siguiente tabla la fase en la que cada miembro trabajo:

Plan de Trabajo	
Etapas	Delegados
Análisis de Requerimientos	<ul style="list-style-type: none"> ■ González López David ■ Jiménez Cervantes Ángel Mauricio ■ Reyes Arroyo Pablo Antonio ■ Carrasco Ruiz Alan Uriel ■ Reyes Romero Luis Fernando ■ Vázquez Reyes Sebastián
Diseño Conceptual (MER y MRI)	<ul style="list-style-type: none"> ■ González López David (MER) ■ Jiménez Cervantes Ángel Mauricio (MER) ■ Reyes Arroyo Pablo Antonio (MER) ■ Carrasco Ruiz Alan Uriel (MIR) ■ Reyes Romero Luis Fernando (MIR) ■ Vázquez Reyes Sebastián (MIR)
Diseño Lógico (MRF)	<ul style="list-style-type: none"> ■ Carrasco Ruiz Alan Uriel ■ Reyes Romero Luis Fernando ■ Vázquez Reyes Sebastián
Diseño Físico	<ul style="list-style-type: none"> ■ González López David ■ Jiménez Cervantes Ángel Mauricio ■ Reyes Arroyo Pablo Antonio
Implementación	<ul style="list-style-type: none"> ■ González López David ■ Jiménez Cervantes Ángel Mauricio ■ Reyes Arroyo Pablo Antonio ■ Carrasco Ruiz Alan Uriel
Presentación	<ul style="list-style-type: none"> ■ Jiménez Cervantes Ángel Mauricio ■ Carrasco Ruiz Alan Uriel

3 Desarrollo

3.1. Diseño

Para esta etapa decidimos tomar la buena práctica de realizar tanto el diagrama Modelo Entidad Relación (MER), como el Modelo Relacional Intermedio (MIR) para conseguir un diseño mas sólido, el cual le generó una cierta consistencia a la propia base de datos, además de que facilitó el proceso de normalización entre las relaciones. A continuación, se presentará todo el proceso de la etapa de diseño de la base de datos planteada a groso modo. Dicho proceso fue el mismo que determinó al plan de trabajo, por lo que las dependencias previamente especificadas son las mismas que se remarcarán durante esta sección.

3.1.1. Análisis de Requerimientos

Como en cualquier proyecto que necesite de una solución TI, el análisis de requerimientos es fundamental para comprender y saber qué es lo que necesita y debe hacer el sistema propuesto. En primera instancia, identificamos los actores involucrados en el proyecto, y dadas las circunstancias, los actores únicamente engloban al usuario final. Por otro lado, también nos encargamos de definir las necesidades que el cliente desea cubrir con nuestro trabajo, o dicho de otra forma, la recopilación de requisitos. Dichos requisitos engloban no nada mas el almacenamiento de la información de la papelería y sus involucrados, si no también funciones que permitan al usuario:

- Consultar información acerca de las ganancias en un periodo dado.
- Actualización del stock de un producto en cada venta
- Obtener la utilidad de un producto con su código de barras
- Emitir una alerta cuando el stock de un producto sea menor a 3, y obtener los nombres de los productos con esta cantidad en inventario

Cabe mencionar que si bien se definieron en su gran mayoría todas las soluciones a las necesidades de los requerimientos planteados al inicio del proyecto en sí, a lo largo de la realización del proyecto y durante las etapas posteriores, se tuvieron que hacer ajustes sobre las soluciones dado a nuevas interpretaciones, concepciones o necesidades del equipo sobre la implementación de las soluciones decididas, por lo que esta etapa se

traslapó con las demás haciendo que la misma abarque más que sólo el inicio del proyecto, a pesar de haber marcado el punto inicial del propio proyecto en sí.

3.1.2. Modelo Entidad Relación

Para la formulación del Modelo Entidad Relación, escogimos la opción de hacer un MER Extendido (MERE), con la finalidad de englobar todos los artículos que pueden estar involucrados en una papelería. Con esto en mente, definimos un total de 9 entidades, donde una entidad es un supertipo y 4 entidades son subtipos.

En primer instancia, tenemos a 5 entidades de las que se desea guardar y consultar información: el PROVEEDOR, el CLIENTE, la VENTA, el INVENTARIO y el PRODUCTO.

- **PROVEEDOR:** Para el proveedor necesitamos almacenar su dirección (desglosada en distintos atributos), su nombre, su o sus teléfonos de contacto y la razón social única de cada proveedor (con esta somos capaces de identificar a cada proveedor en específico).
- **INVENTARIO:** Para el inventario de un producto almacenaremos; el código de barras único para cada producto, la cantidad con la que cuenta la sucursal (stock), una foto para identificar el producto, la fecha de compra y el costo de compra al proveedor
- **CLIENTE:** Para el cliente los datos que debemos de aguardar son el RFC de cada cliente, su nombre completo el cuál se encuentra segmentado por su nombre de pila, apellido paterno y el apellido materno, donde el apellido materno es opcional, Asimismo, debemos de aguardar su dirección (también desglosado en diferentes atributos) y finalmente debemos de aguardar el correo de cada cliente el cual puede ser más de uno.
- **VENTA:** Los datos que guardaremos sobre una venta realizada únicamente son la fecha en que se realizo, el costo total a cobrar y el numero de venta (con este podremos identificar a cada venta)

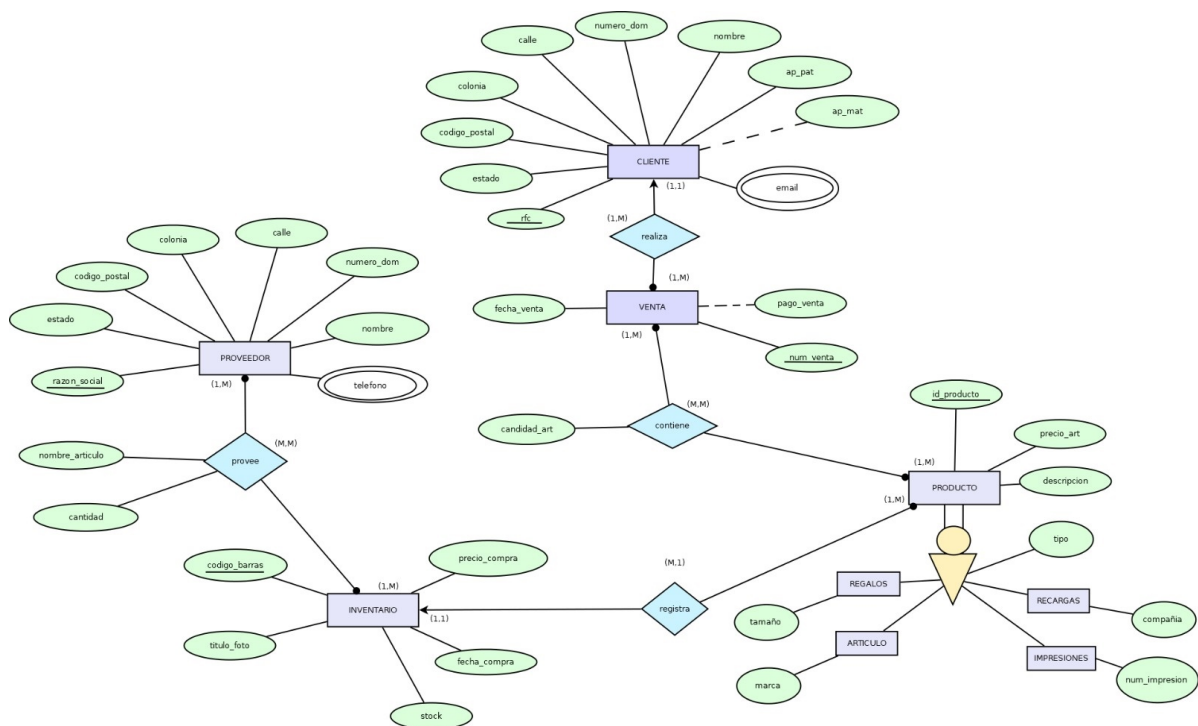
En la ultima entidad: PRODUCTO, decidimos colocarla en una relación con traslape y de tipo total, ya que el supertipo (PRODUCTO) puede ser más de un subtipo (Los distintos productos que ofrece la papelería) y decidimos que sea una relación total porque forzosamente debe pertenecer a un subtipo, o dicho de otra forma, un cliente puede elegir más

de un producto pero forzosamente debe ser alguno de los disponibles. Las entidades que funcionan como subtipos en este modelo son: REGALOS, ARTICULO, RECARGA e IMPRESIONES.

Aunado a las entidades, encontramos un total de 4 relaciones entre las entidades descritas anteriormente:

Relaciones		
Nombre	Involucrados	Descripción
provee	<ul style="list-style-type: none"> ■ PROVEEDOR ■ INVENTARIO 	Una relación con cardinalidad Muchos a Muchos (M,M) y de grado 2; representa la acción del proveedor de abastecer la sucursal con su producto en específico. En esta relación almacenamos el nombre del artículo que suministra el proveedor y la cantidad
registra	<ul style="list-style-type: none"> ■ INVENTARIO ■ PRODUCTO 	Una relación con cardinalidad Muchos a Uno (M,1) y de grado 2; representa la acción de llevar el conteo del producto en específico en el inventario
contiene	<ul style="list-style-type: none"> ■ VENTA ■ PRODUCTO 	Una relación con cardinalidad Muchos a Muchos (M,M) y de grado 2; representa los productos que son vendidos por cada venta. En esta relación almacenamos la cantidad de productos que tiene cada venta
realiza	<ul style="list-style-type: none"> ■ CLIENTE ■ VENTA 	Una relación con cardinalidad Uno a Muchos (1,M) y de grado 2; representa la acción de venderle un producto de la sucursal a un cliente

Con todo esto en mente, a continuación se muestra el Modelo Entidad Relación Extendido formulado:



3.1.3. Modelo Relacional

1. Modelo Relacional Intermedio: Con el modelo Entidad Relación construido, ahora nos concentramos en la formulación del Modelo Relacional Intermedio (MIR) correspondiente. En este mapeo se mostrará a cada entidad con sus respectivos atributos y sus debidas restricciones, por ejemplo si el atributo es llave primaria, si el atributo es opcional, si es derivado calculado, etc. Asimismo, cada atributo de cada relación deberá de tener un dominio, o dicho de otra forma, el tipo de dato del atributo; esto nos ayudará mas adelante en la construcción del modelo físico de la base de datos.

Para este mapeo es necesario cambiar la naturaleza de algunos atributos y relaciones mostrados anteriormente, ya que existen varias restricciones. Por ejemplo, cuando hay un atributo multivaluado se debe de crear una nueva relación y pasa como llave foránea la llave primaria de la entidad de donde viene el atributo multivaluado. También, cuando existe una relación muchos a muchos se debe de crear una nueva relación donde la llave primaria de esta nueva relación es una llave compuesta entre las llaves primarias de las entidades que están involucradas en esta relación, y a su vez, las llaves primarias de las relaciones involucradas pasan también como llaves foráneas. Por ultimo, recordemos que para las relaciones uno a muchos, la llave principal de la cardinalidad uno pasa como llave foránea a la entidad con car-

dinalidad muchos.

Dicho esto, a continuación se muestra el mapeo del Modelo Relacional Intermedio:

- **PROVEEDOR**
 - provRazonSocial varchar(50) (PK)
 - provNombre varchar(40)
 - provEstado varchar(40)
 - provCP smallint
 - provColonia varchar(40)
 - provCalle varchar(40)
 - provNumero smallint
- **TELEFONO**
 - telTelefono bigint (PK)
 - telCodigoTel varchar(4)
 - provRazonSocial varchar(50) (FK)
- **INVENTARIO**
 - invCodigoBarras bigint (PK)
 - invPrecioCompra MONEY
 - invFotoUrl TEXT
 - invFechaCompra DATE
 - invStock smallint
- **PROVEE**
 - provRazonSocial varchar(50) (PK, FK)
 - invCodigoBarras bigint (PK, FK)
 - cantidad smallint
- **CLIENTE**
 - cliRFC char(13) (PK)
 - cliNombre varchar(60)
 - cliApPat varchar(40)
 - cliApMat varchar(40) (N)
 - cliEstado varchar(40)
 - cliCP smallint
 - cliColonia varchar(40)

- cliCalle varchar(40)
- cliNumero smallint
- EMAIL
 - emaEmail varchar(80) (PK)
 - cliRFC char(13) (FK)
- VENTA
 - venNumVenta varchar(15) (PK)
 - venFechaVenta DATE
 - venMontoTotal MONEY (N)
 - cliRFC char(13) (FK)
- PRODUCTO
 - prodIdProducto SERIAL (PK)
 - prodDescripcion varchar(150)
 - prodPrecioVenta MONEY
 - invCodigoBarras bigint (FK)
- CONTENER
 - venNumVenta varchar(15) (PK, FK)
 - prodIdProducto SERIAL (PK, FK)
 - contCantidadProducto smallint
- REGALO
 - prodIdProducto SERIAL (PK, FK)
 - regMarcaRegalo varchar(50)
 - regCategoriaRegalo varchar(50)
- ARTICULO
 - prodIdProducto SERIAL (PK, FK)
 - artMarcaArticulo varchar(50)
- IMPRESION
 - prodIdProducto SERIAL (PK, FK)
 - impTamaño varchar(50)
 - impFormato varchar(50)
- RECARGA
 - prodIdProducto SERIAL (PK, FK)
 - recCompania varchar(50)

2. Proceso de Normalización:

■ 1^º Forma Normal:

Durante la construcción del Modelo Entidad Relación y el Modelo Relacional Intermedio eliminamos los atributos multivaluados y los convertimos en una tabla mas, además, cada tabla, además de las relaciones del MER, cuenta con una sola clave única. Por lo tanto, nuestro diseño ya cumple con la 1^º Forma Normal.

■ 2^º Forma Normal:

Nuestro modelo cumple con la 2FN porque ninguna de las tablas cuenta con dependencias parciales, debido a las mismas razones descritas arriba

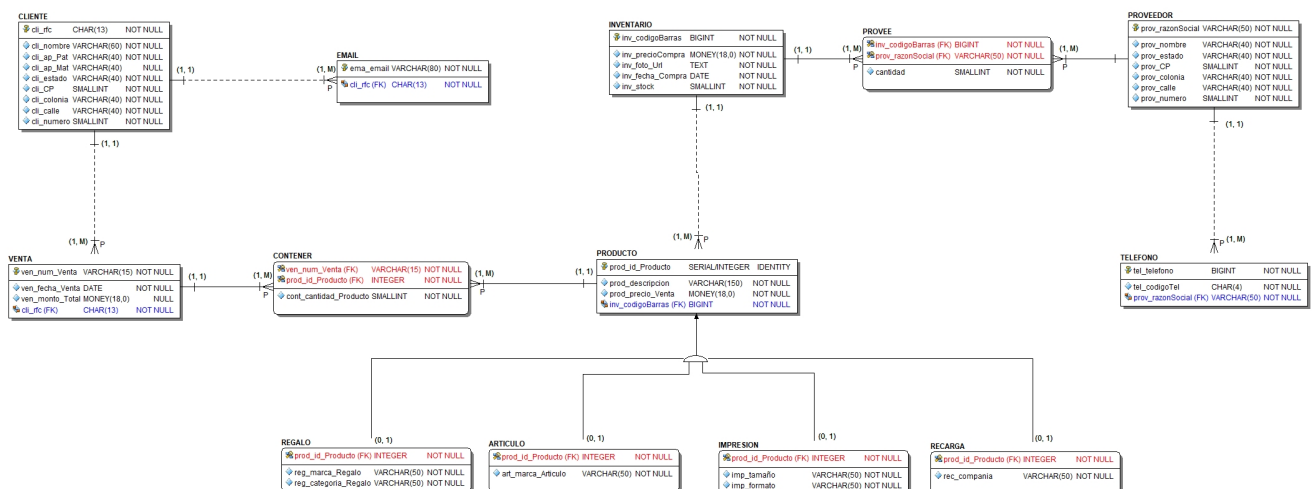
■ 3^º Forma Normal:

Las dependencias transitivas no se presentan en ninguna de las tablas creadas en el MRI, por lo tanto, nuestro diseño ya cumple con la 3FN

3. Modelo Relacional Final:

Con ayuda de un software especializado, terminamos la creación del Modelo Relacional Final, donde se muestran todas las relaciones identificando las llaves primarias y foráneas de cada una de ellas, así como sus atributos correspondientes y su interacción con las demás relaciones.

Dicho esto, a continuación se muestra la creación del Modelo Relacional Final:



3.1.4. Diseño Físico

En base al Modelo Relacional Final, damos inicio a la creación de nuestra base de datos en un sistema. El manejador que usamos para esta tarea es PostgreSQL.

Empezaremos mostrando la creación de las tablas de las entidades:

Creación de la tabla PROVEEDOR:

```
CREATE TABLE PROVEEDOR
(prov_razon_Social varchar(50) NOT NULL,
prov_nombre varchar (40),
prov_estado varchar(40) NOT NULL,
prov_CP smallint not null,
prov_colonia varchar(40) not null,
prov_calle varchar (40) NOT NULL,
prov_numero smallint NOT NULL,
CONSTRAINT "pk_proveedor" PRIMARY KEY (prov_razon_Social)
);
```

Creación de la tabla INVENTARIO:

```
CREATE TABLE INVENTARIO
(inv_codigo_Barras bigint NOT NULL,
inv_precio_Compra money not null,
inv_foto_Url text NOT NULL,
inv_fecha_Compra date NOT NULL,
inv_stock smallint NOT NULL,
inv_modificacion_stock date not null default now(),
CONSTRAINT "pk_inventario" PRIMARY KEY (inv_codigo_Barras)
);
```

Creación de la tabla CLIENTE:

```
CREATE TABLE CLIENTE
(cli_rfc char(13) NOT NULL,
cli_nombre varchar (60) NOT NULL,
cli_ap_Pat varchar (40) NOT NULL,
cli_ap_Mat varchar (40),
cli_estado varchar(40) NOT NULL,
cli_CP smallint not null,
cli_colonia varchar(40) not null,
cli_calle varchar (40) NOT NULL,
cli_numero smallint NOT NULL,
CONSTRAINT "pk_cliente" PRIMARY KEY (cli_rfc)
);
```

Creación de la tabla VENTA:

```
CREATE TABLE VENTA
(ven_num_Venta varchar(15) NOT null CHECK (ven_num_venta ~ '^VENT-[0-9]+$'),
ven_fecha_Venta date NOT null,
ven_monto_Total money default 0,
ven_rfc_Cliente char(13) not null,
CONSTRAINT "pk_venta" PRIMARY KEY (ven_num_Venta),
CONSTRAINT "fk_venta" FOREIGN KEY (ven_rfc_Cliente) REFERENCES CLIENTE(cli_rfc) on update cascade on delete restrict
);
```

Creación de la tabla PRODUCTO:

```
CREATE TABLE PRODUCTO
(prod_id_Producto SERIAL not null,
prod_codigo_Barras bigint NOT null,
prod_descripcion varchar (150) not null,
prod_precio_Venta money NOT null check(prod_precio_venta > cast(0 as money)),
CONSTRAINT "pk_producto" PRIMARY KEY (prod_id_Producto),
CONSTRAINT "fk_producto" FOREIGN KEY (prod_codigo_Barras) REFERENCES inventario(inv_codigo_Barras) on update cascade on delete restrict
);
```

Creación de la tabla REGALO:

```
create table REGALO
(reg_id_Producto SERIAL not null,
reg_marca_Regalo varchar(50) default 'GENERIC',
reg_categoria_Regalo varchar(50),
CONSTRAINT "pk_regalo" PRIMARY key (reg_id_Producto),
CONSTRAINT "fk_regalo" FOREIGN KEY (reg_id_Producto) REFERENCES PRODUCTO(prod_id_Producto) on update restrict on delete restrict
);
```

Creación de la tabla ARTICULO:

```
create table ARTICULO
(art_id_Producto SERIAL,
art_marca_Articulo varchar(50) default 'GENERIC',
CONSTRAINT "pk_articulo" PRIMARY KEY (art_id_Producto),
CONSTRAINT "fk_articulo" FOREIGN KEY (art_id_Producto) REFERENCES PRODUCTO(prod_id_Producto) on update restrict on delete restrict
);
```

Creación de la tabla IMPRESION:

```
create table IMPRESION
(imp_id_Producto SERIAL,
imp_tamaño varchar(50),
imp_formato varchar(50),
CONSTRAINT "pk_impresion" PRIMARY KEY (imp_id_Producto),
CONSTRAINT "fk_impresion" FOREIGN KEY (imp_id_Producto) REFERENCES PRODUCTO(prod_id_Producto) on update restrict on delete restrict
);
```

Creación de la tabla RECARGA:

```
create table RECARGA
(rec_id_Producto SERIAL,
rec_compania varchar(50),
CONSTRAINT "pk_recarga" PRIMARY KEY (rec_id_Producto),
CONSTRAINT "fk_recarga" FOREIGN KEY (rec_id_Producto) REFERENCES PRODUCTO(prod_id_Producto) on update restrict on delete restrict
);
```

Creación de las tablas (M,M) de las relaciones del MER:

Creación de la tabla PROVEER:

```
CREATE TABLE PROVEER
(prov_razon_Social varchar(50) NOT NULL,
inv_codigo_Barras bigint NOT NULL,
cantidad smallint not null check (cantidad > 0),
CONSTRAINT "pk_provee" PRIMARY KEY (prov_razon_Social, inv_codigo_Barras),
CONSTRAINT "fk_proveedor" FOREIGN KEY (prov_razon_Social) REFERENCES PROVEEDOR(prov_razon_Social) on update cascade on delete restrict,
CONSTRAINT "fk_inventario" FOREIGN KEY (inv_codigo_Barras) REFERENCES INVENTARIO(inv_codigo_Barras) on update cascade on delete restrict
);
```

Creación de la tabla CONTENER:

```
CREATE TABLE CONTENER
(ven_num_Venta varchar(15) NOT NULL,
prod_id_Producto serial NOT NULL,
cont_cantidad_Producto smallint NOT NULL,
cons_monto_parcial money,
CONSTRAINT "pk_contiene" PRIMARY KEY (ven_num_Venta, prod_id_Producto),
CONSTRAINT "fk_venta" FOREIGN KEY (ven_num_Venta) REFERENCES VENTA(ven_num_Venta) on update restrict on delete restrict,
CONSTRAINT "fk_producto" FOREIGN KEY (prod_id_Producto) REFERENCES PRODUCTO(prod_id_Producto) on update restrict on delete restrict
);
```

Creación de las tablas de los atributos multivaluados:

Creación de la tabla TELEFONO:

```
CREATE TABLE TELEFONO
(tel_telefono BIGINT NOT NULL, -- mejor en espacio y velocidad
tel_proveedor varchar(50) NOT NULL,
tel_codigoTel varchar(4) not null default '+52',
CONSTRAINT "pk_telefono" PRIMARY KEY (tel_telefono),
CONSTRAINT "fk_telefono" FOREIGN KEY (tel_proveedor) REFERENCES PROVEEDOR(prov_razon_Social) on update cascade on delete cascade
);
```

Creación de la tabla EMAIL:

```
CREATE TABLE EMAIL
(ema_email varchar(80) NOT NULL,
ema_cliente char(13) NOT NULL,
CONSTRAINT "pk_email" PRIMARY KEY (ema_email),
CONSTRAINT "fk_email" FOREIGN KEY (ema_cliente) REFERENCES CLIENTE(cli_rfc) on update cascade on delete cascade
);
```


Con las tablas creadas, procedemos a insertar los registros en las tablas para llenar la base de datos:

Inserción de Registros a la tabla CLIENTE:

```
insert into cliente values
('lesm8806304FA', 'Matías', 'Leyva', 'Suarez', 'Estado de Mexico', '04050', 'Tlalnepantla De Baz', 'Convento De Santa Ines', '38'),
('EATA920901JA4', 'Alejandra', 'Torres', 'Estrada', 'CDMX', '03100', 'Benito Juarez', 'Parroquia', '705'),
('MOLV970411Q17', 'Valeria', 'Lopez', 'moreno', 'Estado de Mexico', '07900', 'Nezahualcoyotl', 'calle 17', '168'),
('RUSF901215bg2', 'Francisco', 'Ruiz', 'Sanchez', 'CDMX', '03020', 'alvaro obregon', 'Doctor Jose Maria Vertiz', '800'),
('FETL7805208P9', 'Luis', 'Fernandez', 'Torres', 'CDMX', '03100', 'coyoacan', 'Av Insurgentes Sur', '1384'),
('GAA900227RU9', 'jose', 'Garcia', 'Arreola', 'CDMX', '02834', 'roma', 'Rio Panuco', '16');
select * from cliente
```

Inserción de Registros a la tabla CONTENER:

```
insert into contener
values ('VENT-0009', 1, 3),
('VENT-0009', 3, 1),
('VENT-0010', 13, 2),
('VENT-0010', 14, 2),
('VENT-0010', 18, 3),
('VENT-0011', 14, 1),
('VENT-0011', 15, 2),
('VENT-0011', 16, 2),
('VENT-0012', 9, 1),
('VENT-0012', 11, 1),
('VENT-0013', 10, 1),
('VENT-0014', 7, 1),
('VENT-0015', 4, 1);
select * from contener
```

Inserción de Registros a la tabla EMAIL:

```
insert into email
values ('matias_s@gmail.com', 'LESM8806304FA'),
('ale_torres12@outlook.com', 'EATA920901JA4'),
('ale_torres@gmail.com', 'EATA920901JA4'),
('vale_lopez_23@gmail.com', 'MOLV970411Q17'),
('vale_moreno@hotmail.com', 'MOLV970411Q17'),
('fran_suarez@gmail.com', 'RUSF901215BG2'),
('ltorres@gmail.com', 'FETL7805208P9'),
('jose_arreola@gmail.com', 'GAA900227RU9 ');
select * from email e
```

Inserción de Registros a la tabla INVENTARIO:

```
insert into inventario (inv_codigo_barras, inv_precio_compra, inv_foto_url, inv_fecha_compra, inv_stock)
values (394789821972, 45.00, 'https://s3.amazonaws.com/productos.papeleria/AT%26T.png', '2023-06-01', 100),
(394789821973, 90.00, 'https://s3.amazonaws.com/productos.papeleria/AT%26T.png', '2023-04-01', 100),
(600724784868, 275.00, 'https://s3.amazonaws.com/productos.papeleria/Mochila-Hombre.png', '2023-02-10', 7),
(293191272802, 40.00, 'https://s3.amazonaws.com/productos.papeleria/Movistar.png', '2023-05-01', 100),
(293191272803, 80.00, 'https://s3.amazonaws.com/productos.papeleria/Movistar.png', '2023-05-01', 100),
(447860646475, 350.00, 'https://s3.amazonaws.com/productos.papeleria/Oso-Gigante-Beige.jpg', '2023-03-15', 2),
(741231014278, 45.00, 'https://s3.amazonaws.com/productos.papeleria/PilloFon.png', '2023-06-01', 100),
(741231014279, 90.00, 'https://s3.amazonaws.com/productos.papeleria/PilloFon.png', '2023-06-01', 100),
(457917919598, 60.00, 'https://s3.amazonaws.com/productos.papeleria/Taza.jpg', '2023-05-20', 10),
(114056211990, 45.00, 'https://s3.amazonaws.com/productos.papeleria/Telcel.png', '2023-06-05', 100),
(114056211991, 90.00, 'https://s3.amazonaws.com/productos.papeleria/Telcel.png', '2023-06-05', 100),
(399590618155, 50.00, 'https://s3.amazonaws.com/productos.papeleria/Cauderno-Jeans.png', '2023-01-25', 15),
(219016431214, 30.00, 'https://s3.amazonaws.com/productos.papeleria/Cuaderno-Cuadro-Chico.png', '2023-01-25', 13),
(226380482565, 30.00, 'https://s3.amazonaws.com/productos.papeleria/Cuaderno-Raya.png', '2023-01-25', 20),
(687499736256, 15.00, 'https://s3.amazonaws.com/productos.papeleria/Goma.png', '2023-02-15', 37),
(502798828568, 1.50, 'https://s3.amazonaws.com/productos.papeleria/Lapiz.png', '2023-04-01', 47),
(484139753236, 17.00, 'https://s3.amazonaws.com/productos.papeleria/Pritt.png', '2023-05-27', 24),
(715150864290, 10.00, 'https://s3.amazonaws.com/productos.papeleria/Pluma.png', '2023-04-01', 15);
select * from inventario
```

Inserción de Registros a la tabla PRODUCTO:

```
insert into producto (prod_codigo_barras, prod_descripcion, prod_precio_venta)
values (394789821972, 'Recarga Telefonica AT&T 50', 50),
(394789821973, 'Recarga Telefonica AT&T 100', 100),
(600724784868, 'Mochila Hombre Color Negro', 350),
(293191272802, 'Recarga Telefonica Movistar 50', 50),
(293191272803, 'Recarga Telefonica Movistar 100', 100),
(447860646475, 'Peluche Oso Gigante', 500),
(741231014278, 'Recarga Telefonica PilloFon 50', 50),
(741231014279, 'Recarga Telefonica PilloFon 100', 100),
(457917919598, 'Taza Decorada Mario Bros', 80),
(114056211990, 'Recarga Telefonica Telcel 50', 50),
(114056211991, 'Recarga Telefonica Telcel 100', 100),
(399590618155, 'Cuaderno Jean Book Cuadro', 65),
(219016431214, 'Cuaderno Cuadro Chico', 35),
(226380482565, 'Cuaderno Raya', 35),
(687499736256, 'Goma', 18),
(502798828568, 'Lapiz', 2),
(484139753236, 'Pritt', 25),
(715150864290, 'Pluma', 15);
select * from producto
```

Inserción de Registros a la tabla PROVEEDOR:

```
insert into proveedor (prov_razon_social, prov_nombre, prov_estado, prov_CP, prov_colonia, prov_calle, prov_numero)
values ('cartones para todo', 'CARTONES Y ALGO MAS', 'Tamaulipas', '07329', 'alguna', 'vicente guerrero', 263),
('cartones 2', 'CARTONES Y ALGO MA', 'Guerrero', '08267', 'alguna_1', 'vicente', 267),
('cartones 3', 'CARTONES Y ALGO M', 'Chiapas', '18273', 'alguna_2', 'vicente av', 264),
('cartones 4', 'CARTONES Y ALGO', 'Sonora', '13652', 'alguna_3', 'vicente avenida', 269),
('cartones 5', 'CARTONES', 'Michoacan', '27201', 'alguna_4', 'avenida vicente guerrero', 213);
select * from proveedor p
```


Inserción de Registros a la tabla PROVEER:

```
insert into proveer
values ('cartones para todo', 600724784868, 2),
('cartones 4', 457917919598, 3),
('cartones 4', 399590618155, 2),
('cartones 4', 219016431214, 6),
('cartones 4', 226380482565, 4),
('cartones 4', 687499736256, 10),
('cartones 4', 502798828568, 16);
select * from proveer
```

Inserción de Registros a la tabla TELEFONO:

```
insert into telefono
values (5544332211, 'cartones para todo'),
(5528273612, 'cartones para todo'),
(7339283818, 'cartones 2'),
(4448927237, 'cartones 3'),
(5529383920, 'cartones 2'),
(2227834749, 'cartones 4'),
(5593474882, 'cartones 5'),
(5567838920, 'cartones 5');
select * from telefono t
```

Inserción de Registros a la tabla VENTA:

```
insert into venta (ven_fecha_venta, ven_rfc_cliente)
values ('2023-05-20', 'MOLV970411Q17'),
('2023-05-20', 'lesm8806304FA'),
('2023-05-25', 'EATA920901JA4'),
('2023-05-25', 'RUSF901215bg2'),
('2023-05-25', 'FETL7805208P9'),
('2023-05-30', 'GAA900227RU9'),
('2023-05-30', 'MOLV970411Q17');
select * from venta
```

3.2. Implementación

Para dar por finalizado el proceso de creación de la base de datos, ahora abarcaremos la etapa de implementación de las funciones y objetos necesarios para la consulta y actualización de los datos en la base, haciendo uso así de todo lo que conocemos de DML y DDL en SQL:

- Aumento de Stock y validación del mínimo del mínimo Stock

Para la resolución de este requerimiento se definieron 2 funciones y 1 trigger.

La primera función es propiamente ejecutada por medio de un trigger el cual se activa en cuanto la tabla “inventario” tiene una actualización, más específicamente, cuando en el inventario se reduce el stock tras una venta. Así pues, la función en sí primeramente declara una variable a la cual por medio de una consulta se le asignará el valor del stock solicitado gracias al id del producto del cual se realizó la venta. Tras lo anterior se especifican dos casos, el primero siendo el que, si la resta del stock con la cantidad de producto es mayor a cero, se realiza la venta y se “valida” la transacción, por lo cual el stock se ve reducido, por otro lado, el segundo caso es referido a si el valor de la resta es menor a cero, la transacción se invalida y se cancela la venta.

```
CREATE OR REPLACE FUNCTION actualizar_stock()
RETURNS TRIGGER AS $$
declare stock smallint;
BEGIN
    SELECT i.inv_stock INTO stock FROM
    inventario i
    where i.inv_codigo_barras = (select p.prod_codigo_barras from producto p
    where p.prod_id_producto = new.prod_id_producto);
    if (stock - new.cont_cantidad_producto) > 0 then
        UPDATE inventario
        SET inv_stock = stock - NEW.cont_cantidad_producto,
        inv_modificacion_stock = now()
        WHERE inv_codigo_barras = (select p.prod_codigo_barras from producto p
        where p.prod_id_producto = new.prod_id_producto);
    else
        raise notice 'No se puede generar compra';
        return null;
    end if;
    return new;
END;
$$ LANGUAGE plpgsql;
```

```
create or replace TRIGGER insercion_contener
BEFORE INSERT ON contener
FOR EACH ROW
EXECUTE FUNCTION actualizar_stock();

--insert into contener values ('VENT-0007', 2, 3)
--select * from contener
```

La segunda función lo único que hace es validar si el valor del stock es menor a 3, siendo este el caso, la función alertará al usuario sobre el estado del stock.

```
CREATE OR REPLACE FUNCTION revisar_stock_suficiente()
RETURNS TRIGGER AS $$
BEGIN
    if new.inv_stock < 3 then
        raise notice 'Queda poco inventario del producto con codigo de barras: %', new.inv_codigo_barras;
    end if;
    return new;
END;
$$ LANGUAGE plpgsql;

create or replace TRIGGER actualizacion_stock
after update ON inventario
FOR EACH ROW
EXECUTE FUNCTION revisar_stock_suficiente();

--select * from inventario i
|
```

- Vista con un stock mínima de 3

Para la solución de este requerimiento se decidió crear una vista la cual conjunta por medio de un “inner join” a la tabla “producto” y a la tabla “inventario”, esto para que directamente analizar y mostrar los productos que justamente tengan un stock menor a 3.

```
CREATE OR REPLACE VIEW min_stock as
SELECT p.prod_descripcion , i.inv_stock FROM producto p
INNER JOIN inventario i ON p.prod_codigo_barras = i.inv_codigo_barras
WHERE i.inv_stock<3;

select * from min_stock
```

- Obtención de la utilidad del producto y la validación de la existencia de la utilidad

Para la resolución de este requerimiento se definieron 2 funciones y un trigger. La primera tiene como funcionalidad el calcular la utilidad por medio de un código de barras de tipo “big int”, así pues, en la función

se definen dos variables denominadas “costoProducto” y “precioVentaProducto” ambas de tipo “money”, a las cuales posteriormente se les asignan valores por medio de dos consultas, la primera de “costoProducto” se le es asignado el valor per se del producto almacenado en el inventario, siendo el valor determinante de dicha consulta el código de barras del producto. Por otro lado, la segunda asignación referida a “precioVentaProducto” es realizada mediante una consulta a la tabla producto en la cual por medio del código de barras del producto se le asigna el valor a la venta del propio producto.

Entonces, con todo lo anterior la función realiza la diferencia entre el costo y precio del producto para así guardarlo dentro de una variable que es la utilidad en sí la cual retorna la misma función y marca el final de la misma, resolviendo así el requerimiento tal cual se determinó.

```
CREATE OR REPLACE FUNCTION obtener_utilidad_por_codigo_de_barras(codigo_barras bigint)
RETURNS money
AS $$
DECLARE
    costo_producto money;
    precio_venta_producto money;
    utilidad money;
BEGIN
    SELECT i.inv_precio_compra into costo_producto FROM inventario i WHERE i.inv_codigo_barras =codigo_barras;
    SELECT p.prod_precio_venta into precio_venta_producto FROM producto p where p.prod_codigo_barras = codigo_barras;
    RAISE NOTICE 'Costo Producto: %',costo_producto;
    RAISE NOTICE 'Precio a la Venta Producto: %',precio_venta_producto;
    utilidad = precio_venta_producto-costo_producto;
    RAISE NOTICE 'Utilidad resultante: %', precio_venta_producto-costo_producto;

    RETURN utilidad;
END;
$$
LANGUAGE PLPGSQL;

--select obtener_utilidad_por_codigo_de_barras(600724784868)
```

La segunda función fue un agregado extra que refina y beneficia al propietario de la base, esto debido a que la función explícitamente nos dice que lo que busca es comprobar la utilidad, esto tras ser llamada por un trigger antes de insertar un producto en la propia tabla “producto”. La función realiza un análisis en el cual el precio del producto a la venta debe de ser mayor al valor del producto almacenado en el “inventario”, denotando así dos casos en el que la función realiza un aviso de si existe o no una utilidad.


```

CREATE OR REPLACE FUNCTION comprobar_utilidad()
RETURNS TRIGGER AS $$
declare costo_producto money;
begin
    SELECT i.inv_precio_compra into costo_producto FROM inventario i
    WHERE i.inv_codigo_barras = new.prod_codigo_barras;
    if new.prod_precio_venta > costo_producto then
        raise notice 'Existe una utilidad de: %', new.prod_precio_venta - costo_producto;
    else
        raise notice 'NO Existe una utilidad para el producto con codigo de barras: %', new.prod_codigo_barras;
    end if;
    return new;
END;
$$ LANGUAGE plpgsql;

create or replace TRIGGER comprobar_utilidad_producto
before insert ON producto
FOR EACH ROW
EXECUTE FUNCTION comprobar_utilidad();

```

■ Obtención de la factura

```

begin;
    select p., t.tel_codigo, t.tel_telefono, c2., now(), v., c., pr.*, i.inv_codigo_barras
    from proveedor p
    inner join telefono t on p.prov_razon_social = t.tel_proveedor
    inner join proveer p2 on p.prov_razon_social = p2.prov_razon_social
    inner join inventario i on p2.inv_codigo_barras = i.inv_codigo_barras
    inner join producto pr on i.inv_codigo_barras = pr.prod_codigo_barras
    inner join contener c on pr.prod_id_producto = c.prod_id_producto
    inner join venta v on c.ven_num_venta = v.ven_num_venta
    inner join cliente c2 on v.ven_rfc_cliente = c2.cli_rfc
    where v.ven_num_venta = 'VENT-0007';
rollback

```

```

CREATE FUNCTION obtener_factura_cursor(num_venta varchar(15)) RETURNS setof refcursor AS
$$
DECLARE c_cliente_info refcursor;
DECLARE c_producto refcursor;
declare c_venta_info refcursor;
BEGIN
    OPEN c_cliente_info FOR
        select * from cliente c
        inner join venta v on c.cli_rfc = v.ven_rfc_cliente
        where v.ven_num_venta = num_venta ;
    RETURN NEXT c_cliente_info;
    OPEN c_producto FOR
        select i.inv_codigo_barras as codigo_barras, pr.prod_descripcion, c.cont_cantidad_producto, pr.prod_precio_venta , c.cont_monto_parcia
        from contener c
        inner join producto pr on c.prod_id_producto = pr.prod_id_producto
        inner join inventario i on pr.prod_codigo_barras = i.inv_codigo_barras
        where c.ven_num_venta = num_venta;
    RETURN NEXT c_producto;
    OPEN c_venta_info FOR
        select v.ven_monto_total, (v.ven_monto_total * .16) as iva, (v.ven_monto_total +(v.ven_monto_total * .16)) as monto_final
        from venta v
        where v.ven_num_venta = num_venta;
    RETURN NEXT c_venta_info;
END;
$$ LANGUAGE plpgsql;

```

```

BEGIN;
SELECT obtener_factura_cursor('VENT-0007');
FETCH ALL IN "<unnamed portal 1>";
FETCH ALL IN "<unnamed portal 2>";
FETCH ALL IN "<unnamed portal 3>";
COMMIT

```

■ Creación del índice

El índice creador se decidió en primera instancia que fuese de tipo “int” en el código de barras en la tabla de producto, esto debido a que, a lo largo de la creación de la base de datos, del proceso de todas las inserciones y del funcionamiento de la base en sí, notamos que las consultas a la tabla productos eran bastante continuas y variadas en cuanto a registro, para aclarar lo anterior podemos contrastar con otras dos tabla que se tuvieron en cuenta para la asignación del índice.

La primera tabla tomada en cuenta fue la de inventario, esto dado a los accesos al stock, la cantidad de consultas que requerían a la tabla, y el uso del código de barras per se. La razón por la cual no nos decantamos por la definición del índice en esta tabla fue en primera instancia que el código de barras está definido como “pk”, además de que los registros de la tabla son “latentes” por lo que no varían y no son de interés funcional. La otra tabla fue la de venta, esto dado igualmente a su cantidad de consultas que la requieren, pero la descartamos rápidamente dado a lo poco funcional y de interés que era el colocar el índice en su número de venta.

Por esto mismo, decidimos colocar el índice en la tabla y atributo ya mencionado ya que a nuestro parecer conjunta todas las ventajas de las tablas anteriores y descarta principalmente sus desventajas funcionales en cuanto a implementación y relevancia de consultas.

```
CREATE INDEX idx_producto ON PRODUCTO("prod_codigo_Barras");
```

Para conseguir el funcionamiento del anterior código primero definimos al tipo de dato del atributo “numVenta” como “serial”, esto para posteriormente alterarlo por medio de una secuencia en la cual por medio de una función se determina que el tras cierto número de caracteres de formato, se siga una secuencia numérica de forma automática.

```

create SEQUENCE numVenta_Seq;
SELECT setval('numVenta_Seq',001);

create or replace function num_venta() returns varchar as
$$
select right('000' || nextval('numVenta_Seq')::varchar,4) as id;
$$
language sql;

alter table venta
ALTER COLUMN ven_num_Venta SET DEFAULT (('VENT-'::text || num_venta()::text))

```

■ Aumento del Stock

```

CREATE OR REPLACE FUNCTION calcular_aumento_stock()
RETURNS TRIGGER AS $$
declare stock smallint;
BEGIN
    SELECT i.inv_stock INTO stock
    from inventario i
    where i.inv_codigo_barras = new.inv_codigo_barras;
    update inventario
    set inv_stock = stock + new.cantidad
    where inv_codigo_barras = new.inv_codigo_barras;
    return new;
END;
$$ LANGUAGE plpgsql;

create or replace TRIGGER reabastecimiento_stock
after insert ON proveer
FOR EACH ROW
EXECUTE FUNCTION calcular_aumento_stock();

--select * from inventario i
--insert into proveer values ('cartones 5', 715150864290, 30)
--select * from proveer p
--select * from inventario i

```

■ Cálculo del monto parcial y del monto total de la venta

Estás dos funciones fueron creadas con la finalidad de que al momento de registrar una venta, el campo monto parcial y monto total de la venta se calculará de forma automática.

```

CREATE OR REPLACE FUNCTION calcular_monto_parcial()
RETURNS TRIGGER AS $$
declare precio money;
begin
    select p.prod_precio_venta into precio from producto p
    where p.prod_id_producto = new.prod_id_producto;
    update contener
    set cont_monto_parcial = new.cont_cantidad_producto * precio
    where prod_id_producto = new.prod_id_producto and ven_num_venta = new.ven_num_venta;
    return new;
END;
$$ LANGUAGE plpgsql;

create or replace TRIGGER calculo_insercion_contener
after insert ON contener
FOR EACH ROW
EXECUTE FUNCTION calcular_monto_parcial();

--insert into contener values ('VENT-0007', 1, 3)
--select * from contener
--select * from producto

```

```

CREATE OR REPLACE FUNCTION calcular_monto_total()
RETURNS TRIGGER AS $$
declare monto_total money;
begin
    select sum(c.cont_monto_parcial) into monto_total from contener c
    where c.ven_num_venta = new.ven_num_venta;

    update venta
    set ven_monto_total = monto_total
    where ven_num_venta = new.ven_num_venta;
    return new;
END;
$$ LANGUAGE plpgsql;

create or replace TRIGGER calculo_monto_total_venta
after insert ON contener
FOR EACH ROW
EXECUTE FUNCTION calcular_monto_total();

--insert into venta (ven_fecha_venta, ven_rfc_cliente) values ('2023-05-30', 'GAA900227RU9 ')
--select * from venta
--insert into contener values ('VENT-0008', 18, 4)
--select * from contener
--select * from producto

```

■ Validación de las mayúsculas

Esta función sólo se creo con la finalidad de que al momento de hacer una consulta usando el rfc del cliente o que al momento de generar un nuevo cliente no tengamos errores al momento de registrarlo.


```

CREATE OR REPLACE FUNCTION uppercase_function()
RETURNS TRIGGER AS $$
BEGIN
    NEW.cli_rfc := UPPER(NEW.cli_rfc);
    NEW.cli_nombre := UPPER(NEW.cli_nombre);
    NEW.cli_ap_Pat := UPPER(NEW.cli_ap_Pat);
    NEW.cli_ap_Mat := UPPER(NEW.cli_ap_Mat);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER uppercase_cli_rfc
BEFORE INSERT ON cliente
FOR EACH ROW
EXECUTE FUNCTION uppercase_function();

```

- Ganancias en un rango de fechas dado

La primera función contempla el caso de la utilidad dada una sola fecha de inicio, en la cual, la propia función realiza un par de subconsultas y de inner joins para así poder obtener en base a una cierta fecha, la diferencia del valor del producto comprado y sobre cuanto se vendió, es decir, su utilidad.

```

CREATE OR REPLACE FUNCTION ganancias_a_partir_de(fecha date)
RETURNS money AS $$
declare ganancia money;
BEGIN
    select sum(g.ganancia) into ganancia
    from (
        select c_info.id_producto, c_info.suma_cant, c_info.suma_vendida, i.inv_precio_compra,
        (c_info.suma_cant*i.inv_precio_compra) as suma_precop_venta,
        c_info.suma_vendida - (c_info.suma_cant*i.inv_precio_compra) as ganancia
        from producto p
        inner join (
            select c.prod_id_producto as id_producto, sum(c.cont_cantidad_producto) as suma_cant,
            sum(c.cont_monto_parcial) as suma_vendida
            from contener c
            inner join venta v on c.ven_num_venta = v.ven_num_venta
            where v.ven_fecha_venta >= fecha
            group by c.prod_id_producto
        ) c_info on p.prod_id_producto = c_info.id_producto
        inner join inventario i on p.prod_codigo_barras = i.inv_codigo_barras
        ) g;
    return ganancia;
END;
$$ LANGUAGE plpgsql;

select ganancias_a_partir_de('2023-05-27')

```

La segunda función realiza igualmente 3 subconsultas para la obtención de la utilidad, pero a diferencia de la función anterior, su funcionalidad se basa en un “between” dada una fecha de inicio y de fin las cuales fueron mandadas como parámetros de la propia función.

```

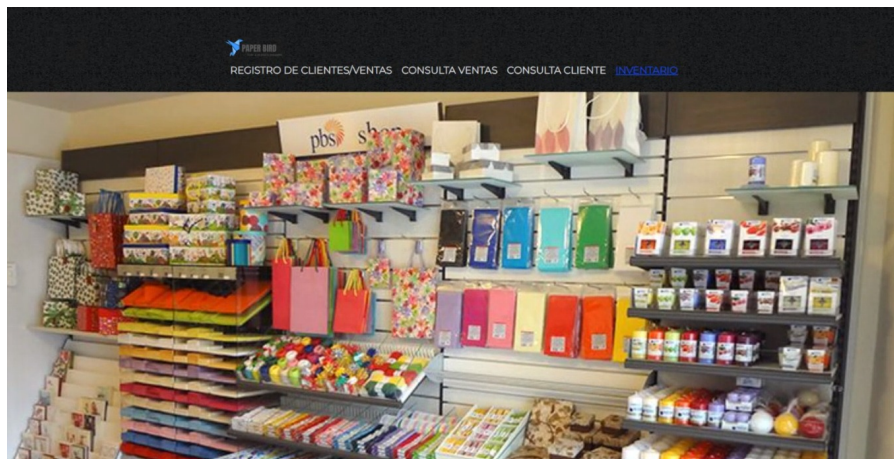
CREATE OR REPLACE FUNCTION ganancias_por_rango_fechas(fecha_inicio date, fecha_fin date)
RETURNS money AS $$
declare ganancia money;
BEGIN
    select sum(g.ganancia) into ganancia
    from (
        select c_info.id_producto, c_info.suma_cant, c_info.suma_vendida, i.inv_precio_compra,
            (c_info.suma_cant*i.inv_precio_compra) as suma_precop_venta,
            c_info.suma_vendida - (c_info.suma_cant*i.inv_precio_compra) as ganancia
        from producto p
        inner join (
            select c.prod_id_producto as id_producto, sum(c.cont_cantidad_producto) as suma_cant,
                sum(c.cont_monto_parcial) as suma_vendida
            from contener c
            inner join venta v on c.ven_num_venta = v.ven_num_venta
            where v.ven_fecha_venta between fecha_inicio and fecha_fin
            group by c.prod_id_producto
        ) c_info on p.prod_id_producto = c_info.id_producto
        inner join inventario i on p.prod_codigo_barras = i.inv_codigo_barras
    ) g;
    return ganancia;
END;
$$ LANGUAGE plpgsql;
select ganancias_por_rango_fechas('2023-05-27', '2023-06-01'

```

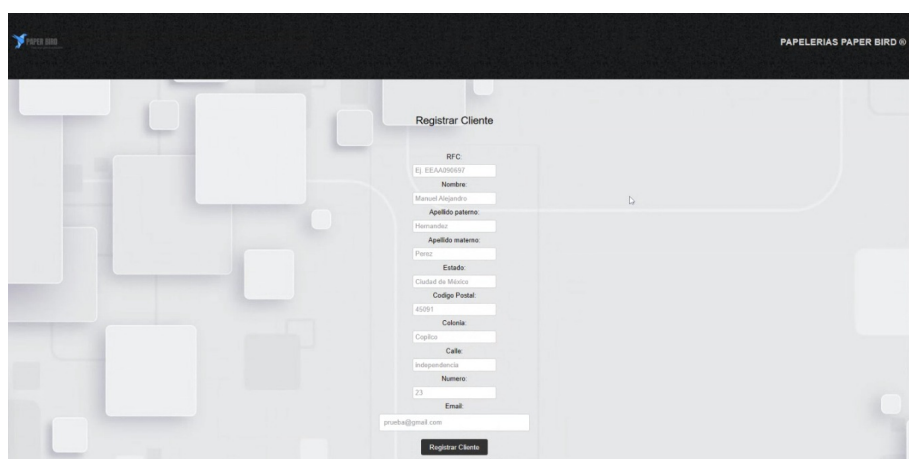
4 Presentación

La modalidad seleccionada como forma de conexión a la base de datos fue por medio de una página web realizada con HTML y PHP.

La siguiente imagen muestra la ventana principal de la interfaz de la base de datos, en este apartado se gestionan los accesos a los registros dentro de la base por medio de una barra de menú principal. A su vez, se permite personalizar esta misma ventana a gusto del cliente, conteniendo así la información y elementos que deseen.



En la siguiente imagen, se nos muestra la parte de la interfaz que nos permite la inserción de clientes dentro de la base de datos, todo esto en base al formato mostrado el cual es justamente el de los atributos de la tabla cliente definida dentro de la base en sí.

A screenshot of a web application interface showing a form titled 'Registrar Cliente'. The form is set against a light gray background with a pattern of overlapping squares. The form fields are as follows: 'RFC' with the value 'E6A000000', 'Nombre' with the value 'Manuel Alejandro', 'Apellido paterno' with the value 'Hernandez', 'Apellido materno' with the value 'Perez', 'Estado' with the value 'Ciudad de Mexico', 'Codigo Postal' with the value '45001', 'Colonia' with the value 'Capitza', 'Calle' with the value 'Independencia', 'Numero' with the value '23', and 'Email' with the value 'prueba@gmail.com'. At the bottom of the form is a button labeled 'Registrar Cliente'.

La siguiente ventana nos permite la inserción de ventas en función a los artículos definidos dentro de la base de datos, esto mismo se realiza tras validar la existencia de un cierto cliente ya previamente registrado

dentro de la propia base. Su formato de inserción contempla un “serial” ya definido en la sección de DML dentro de la base.

Por ultimo, la siguiente ventana muestra los registros de los clientes insertados en la base de datos, esto permite gestionar de una forma óptima a la propia base dado a la visibilidad de información que ofrece.

cli_rfc	cli_nombre	cli_ap_pat	cli_ap_mat	cli_estado	cli_cp	cli_colonia	cli_calle	cli_numero
AECD000612HR	DAVID	ARREOLA	CODD	Ciudad de México	9800	Coyoacan	desconocida	4
LESM8806304FA	MATÍAS	LEYVA	SUAREZ	Estado de México	4050	Tlalnepantla De Baz	Convento De Santa Ines	38
EATA920901JA4	ALEJANDRA	TORRES	ESTRADA	CDMX	3100	Benito Juarez	Parroquia	705
MOLV970411Q17	VALERIA	LOPEZ	MORENO	Estado de México	7900	Nezahualcoyotl	calle 17	168
RUSF901215BG2	FRANCISCO	RUIZ	SANCHEZ	CDMX	3020	alvaro obregon	Doctor Jose Maria Vertiz	800
FETL7805208P9	LUIS	FERNANDEZ	TORRES	CDMX	3100	coyoacan	Av Insurgentes Sur	1384
GAA900227RU9	JOSE	GARCIA	ARREOLA	CDMX	2834	roma	Rio Panuco	16

5 Conclusiones

- Carrasco Ruiz Alan Uriel

En general este proyecto me gustó bastante ya que fue muy desafiante en algunos aspectos, principalmente en la parte de implementar todos los conocimientos que vimos en clase desde el tema 1 del curso, por otra parte investigando por cuenta propia muchos temas que a lo mejor tenía noción pero no manejaba al 100%. Así como el trabajar en equipo te ayuda a resolver algunos aspectos que a veces no vez por estar tan concentrado en algo, la parte que mas me agrado fue el cómo partimos de un problema dado, y de ahí fuimos implementando todas las fases, teniendo que pasar de fase en fase con un plan de trabajo, que también nos hizo aplicar conocimientos de otras materias, principalmente para la organización.

- González López David:

Tras todo lo presentado en este trabajo me gustaría concluir comentando que si bien el proyecto nos permitió profundizar en varios aspectos y conceptos ya no sólo de SQL sino también del manejo de bases y de postgresql per se, también me resultó en lo personal bastante pesado y retador, esto debido a que hubieron varios conceptos que no dominaba del todo y que tuve que comprender a la mala, además, al tratarse de un proyecto final el cual fue dejado propiamente en las últimas semanas de clase, se me traslaparon las tareas con las de otras materias, por lo cual se me dificultó el realizar las actividades de las cuales yo era responsable de una forma plena, a pesar de ello considero que este fue un proyecto interesante y bastante retador.

- Jimenez Cervantes Angel Mauricio

El desarrollo de este proyecto me permitió poner en práctica los conocimientos adquiridos en clases de teoría y laboratorio. Las diferentes fases de la elaboración de una base de datos son de gran importancia para asegurar la calidad del resultado final. Considero que la implementación del modelo lógico y el diseño del modelo físico son etapas clave, ya que sientan las bases para las fases posteriores.

Aunque el proyecto final coincidió con las últimas semanas de clase y se superpuso con otras tareas de otras materias, considero que fue un desafío interesante que me permitió aplicar lo aprendido y enfrentar nuevos retos.

En resumen, este proyecto me brindó la oportunidad de profundizar en SQL, PostgreSQL y el diseño de bases de datos, además de permitirme adquirir nuevos conocimientos y habilidades. Aunque fue exigente, considero que valió la pena y contribuyó a mi desarrollo académico.

- Reyes Arroyo Pablo Antonio

El desarrollo del presente proyecto me permitió poner en práctica los conocimientos adquiridos en clases de teoría y laboratorio. Las fases de la elaboración de una base de datos son de gran importancia para prevenir correcciones en la elaboración de fases posteriores, ya que es un proceso secuencial el que se sigue. A mi parecer, las fases de las que depende que se mantenga un correcto enfoque, es la implementación del modelo lógico y todo lo que corresponde al modelo físico, ya que en el modelo lógico si se lleva a cabo de manera correcta con los requerimientos obtenidos, se podrá brindar una buena base para todas las fases siguientes. Por otro lado, el modelo físico, que incluye toda la parte de la programación, es de gran ayuda, ya que se pueden facilitar tareas con el uso de las diversas herramientas. La parte que corresponde a la programación en la base de datos, fue mi favorita, ya que me permitió adquirir nuevos conocimientos y reafirmar los obtenidos.

- Reyes Romero Luis Fernando:

Una vez concluido este proyecto puedo decir que se ha cumplido el objetivo de este mismo, ya que gracias a lo que vimos a lo largo del curso en nuestra clase de teoría y en nuestra clase de laboratorio logramos implementar una base de datos partiendo desde los requerimientos dados por un cliente satisfaciendo sus necesidades. Asimismo, gracias a este proyecto nos dimos cuenta de la importancia que tiene las etapas de la elaboración de una base de datos, ya que a lo largo de la etapa de desarrollo e implementación al momento de manipular los datos teníamos errores o se nos hacía difícil la manipulación de los datos y esto debido al tipo de dato que habíamos puesto desde el mapeo del Modelo Relacional Intermedio, entonces se tenía que modificar todo desde ese punto. Esa fue una de las dificultades que tuvimos otra dificultad que tuvimos a lo largo del desarrollo de la implementación fue que al momento de cumplir los requerimientos necesitábamos ayuda de otro atributo que no se había creado, entonces se tenía que modificar todo desde el Modelo Entidad Relación con estas dificultades que tuvimos nos dimos cuenta de la importancia de las buenas practicas

y de tener todo bien desde un inicio para que no ocurra esto.

■ Vázquez Reyes Sebastián :

Tras la finalización del proyecto, he de admitir que fue todo un reto para mi realizar este trabajo. Repase por completo lo que es el modelo Relacional y Entidad relación, tanto básico como extendido. Y también me ayudo a comprender de mejor manera el proceso de normalización de una base de datos, y el manejo de restricciones en la creación de los objetos en el diseño físico. Sin embargo, la creación de funciones y objetos en SQL fue mi punto débil.

La creación de vistas, disparadores y subconsultas anidadas son cosas que me costaron entender. Esto me ayudo a darme cuenta de lo que necesito estudiar para poder considerar mi aprendizaje completo en el curso.

Por otro lado, un proyecto de esta índole me ayudo a entender la importancia de llevar un proceso ordenado para la creación de una solución TI, la obtención de requisitos y la división de actividades fue algo que nos ayudo muchísimo para acabar de la mejor manera posible nuestro proyecto

Por ultimo, me gustaría mencionar lo importante que fue la corrección de errores desde nuestro modelo ER, pues un error desde ese punto cambiaba toda nuestra base de datos y fue de gran ayuda darnos cuenta desde el inicio.