



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

## Proyecto Final.

Bases de Datos.

Semestre 2023-2

PROFESOR

Fernando Arreola

## Grupo 1

EQUIPO PRSZ:

Ramírez Bartolo Ignacio  
Santiago Martinez Ricardo  
Pacheco Saavedra Angel Gael  
López Segundo Luis Ivan

10 de Junio 2023

# Índice

<b>1. Introducción.</b>	<b>1</b>
<b>2. Plan de Trabajo</b>	<b>2</b>
2.1. Semana 1: Modelo Entidad-Relación, . . . . .	3
2.1.1. Objetivos. . . . .	3
2.1.2. Plan de actividades: . . . . .	3
2.1.3. Indicadores. . . . .	3
2.1.4. Tiempo. . . . .	3
2.2. Semana 2: Modelo Relacional. . . . .	4
2.2.1. Objetivos. . . . .	4
2.2.2. Plan de actividades: . . . . .	4
2.2.3. Indicadores. . . . .	4
2.2.4. Tiempo. . . . .	4
2.3. Normalización del modelo relacional. . . . .	5
2.3.1. Objetivos. . . . .	5
2.3.2. Plan de actividades: . . . . .	5
2.3.3. Indicadores. . . . .	5
2.3.4. Tiempo. . . . .	5
2.4. Script de la creación de las tablas. . . . .	6
2.4.1. Objetivos. . . . .	6
2.4.2. Plan de actividades: . . . . .	6
2.4.3. Indicadores. . . . .	6
2.4.4. Tiempo. . . . .	6
2.5. Script para el agregado de información. . . . .	7
2.5.1. Objetivos. . . . .	7
2.5.2. Plan de actividades: . . . . .	7
2.5.3. Indicadores. . . . .	7
2.5.4. Tiempo. . . . .	7
2.6. Script de los códigos de la programación a nivel BD. . . . .	8
2.6.1. Objetivos. . . . .	8
2.6.2. Plan de actividades: . . . . .	8
2.6.3. Indicadores. . . . .	8
2.6.4. Tiempo. . . . .	8
2.7. Documento formal elaborado en LáTex . . . . .	9
2.7.1. Objetivos. . . . .	9
2.7.2. Plan de actividades: . . . . .	9
2.7.3. Indicadores. . . . .	9
2.7.4. Tiempo. . . . .	9
<b>3. Gestión de cambios del proyecto.</b>	<b>10</b>
<b>4. Gestión de las Comunicaciones.</b>	<b>11</b>
<b>5. Diseño.</b>	<b>12</b>
<b>6. Implementación.</b>	<b>19</b>
<b>7. Conclusiones.</b>	<b>35</b>
<b>8. Referencias.</b>	<b>37</b>

## 1. Introducción.

Mucho se ha dicho con respecto a la innovación de ciertos proyectos que buscan obtener nuevos alcances con los que se puedan obtener mejoras en las que se logre obtener la satisfacción de los clientes cuando quieren obtener un producto, del cual ya se ha vuelto costumbre que siempre ocurren imprevistos cuando un cliente se incentive a realizar la compra de un producto en algún sitio web. Debido a esto, se ha planteado en la resolución de alguna propuesta que permita reorganizar la información para plantear su solución y aplicarla en la práctica con una base de datos de una papelería que innove la información de sus datos donde se almacenen los datos de los proveedores y clientes, así como el inventario de los productos donde se maneje distintas sentencias en las que puedan realizar distintas acciones que innoven el almacenamiento de la información como generar una vista para asimilar una factura, mostrar la cantidad total que se vendió un producto con la ganancia en un cierto periodo de tiempo, generar alertas si solo quedan 3 productos (de los cuáles también se podrá consultar la lista de esos productos con esa característica) luego de realizar un pedido o abortar la transacción si el contador de productos está en 0, recibir la utilidad por medio del código de barras y utilizar un índice, entre otros requerimientos que ayudarán a la base de datos mantenerse innovadora para la papelería a la que se le quiere implementar. Este proyecto presenta por medio de los cuatro integrantes cumplir los siguientes objetivos, donde se remarca crear una base de datos que cumpla con los requerimientos pedidos por el profesor, del cual pone a prueba todos los conocimientos adquiridos sobre la asignatura de bases de datos que se asimilaron durante el semestre. Igualmente se puede demostrar que se tiene como objetivo analizar una serie de requerimientos para proponer una solución que atienda a los mismos requisitos. aplicando los conceptos vistos en el curso. La propuesta de solución que se planteó para realizar la base de datos consta de construir la base de datos de la misma manera en la que lo hemos hecho, pero la innovación radica en cómo se procede a agregar los requerimientos pedidos para este proyecto, por lo que llega a incluir los requisitos explicados en el párrafo anterior.

## 2. Plan de Trabajo

De manera general, durante la realización de este proyecto, se realizaron distintas actividades que abarcaban la creación de una base de datos, del cual abarca las siguientes actividades:

1. Creación del modelo Entidad-relación (diagrama para ilustrar las entidades, atributos y relaciones del problema).
2. Creación del modelo relacional (diagrama para comenzar a transcribir la creación de tablas con sus atributos y relaciones con restricciones).
3. Normalización del modelo relacional (normalizar el modelo relacional para reducir inconsistencias y redundancias).
4. Creación de las tablas por lenguaje de definición de datos (uso del comando CREATE para crear las tablas y secuencias con sus atributos y relaciones en la base de datos).
5. Inserción de la información por lenguaje de manipulación de datos (uso del comando INSERT INTO para insertar la información correspondiente en la base de datos a las tablas específicas).
6. Consulta de tablas e información por lenguaje de consulta de datos (uso del comando SELECT para consultar la información presente en las tablas de la base de datos).
7. Realización del documento formal que respalda el proyecto (Desarrollo del documento visto actualmente para describir cómo se fue realizando el proyecto).

En este plan de trabajo, se llevará a cabo la descripción de cada una de las actividades que abarcan el proyecto final para indicar que fue lo que se realizó y cómo se fue realizando para respaldar cómo se fue creando nuestra base de datos.

## **2.1. Semana 1: Modelo Entidad-Relación,**

### **2.1.1. Objetivos.**

Realizar el modelo entidad-relación que logre representar las relaciones que existen entre las distintas entidades que conforman la base de datos y asignar los atributos que caracterizan las dichas entidades.

### **2.1.2. Plan de actividades:**

- 29/mayo/2023: se realiza el modelo entidad-relación del problema que está descrito en la presentación del proyecto, del cual se evaluó encontrar las entidades presentes en el problema. Una vez ubicadas las entidades, se procedió a asignar sus atributos que ya venían indicados en el problema, pero por lo mientras se dejaban a un lado los atributos que no eran exclusivamente de una entidad. Finalmente se procedía a identificar las relaciones presentes que existían entre cada relación para indicar el grado entre cada relación. Se logró terminar el modelo entidad-relación en solo un día.
- 3/junio/2023: se revisa que el modelo entidad-relación no presente errores, del cual se logró llegar a la conclusión de que el modelo respectivo está realizado a la perfección.

### **2.1.3. Indicadores.**

Registro de los conceptos para identificar entidades, la relación de las características con ciertos conceptos para identificar atributos y de los verbos para identificar relaciones.

### **2.1.4. Tiempo.**

**Fecha de inicio de la actividad:** 29/mayo/2023

**Fecha de termino de la actividad:** 03/junio/2023

**Nombre de la persona que realizo la actividad:** Pacheco Saavedra Angel Gael, Santiago Martínez Ricardo

## **2.2. Semana 2: Modelo Relacional.**

### **2.2.1. Objetivos.**

Realizar el modelo relacional que logre representar el modelo relacional que se basa en el mismo modelo entidad-relación que fue realizado para este proyecto, de tal modo que se tenga transcrito el modelo intermedio relacional para escribirlo en la base de datos.

### **2.2.2. Plan de actividades:**

- 04/junio/2023: se realiza el modelo relacional del cual se basa en el modelo entidad-relación, del cual primeramente se procedió a escribir el modelo intermedio relacional de cada entidad con sus respectivos atributos y su tipo de dato que por lo menos no fueran atributos multivaluados. Si encontramos un atributo multivaluado, se crea una nueva relación el nombre del atributo multivaluado como llave primaria y recibe la llave primaria de su relación base como llave foránea. El siguiente paso consiste en crear el modelo intermedio de las relaciones en el que va a depender del tipo de relación que se tenga, del cual si es una relación m:m se le crea una nueva relación con las llaves primarias de las relaciones que une como llaves foráneas; o si es una relación 1:m o m:1 la llave primaria de la relación con cardinalidad 1 se le pasaría a la relación con cardinalidad “m” como llave foránea; y si es una relación con cardinalidad 1:1, la llave primaria de la relación que menos importancia tiene se pasa a la relación con mayor importancia como llave foránea. Una vez realizado el modelo relacional, se observa que esté realizado correctamente, del cual hay que tomar en cuenta las restricciones que tienen las relaciones, del cuál solo se tomaron en cuenta restricciones de llaves foráneas y llaves primarias.
- 05/junio/2023: se revisa que el modelo relacional no presente errores, del cual se llegó a la conclusión de que está preparado para pasarlo a la base de datos.

### **2.2.3. Indicadores.**

Registro del modelo entidad-relación correctamente realizado y registro de los tipos de datos de cada atributo, las llaves primarias y foráneas de cada relación.

### **2.2.4. Tiempo.**

**Fecha de inicio de la actividad:** 04/junio/2023

**Fecha de termino de la actividad:** 05/junio/2023

**Nombre de la persona que realizo la actividad:** Pacheco Saavedra Angel Gael, Santiago Martínez Ricardo

## **2.3. Normalización del modelo relacional.**

### **2.3.1. Objetivos.**

Realizar la normalización del modelo relacional que permita implementar mejores diseños para que así se logre reducir las redundancias e inconsistencias que pueden presentar los datos.

### **2.3.2. Plan de actividades:**

- 05/junio/2023: una vez realizado correctamente el modelo relacional, se procedió a analizar si se debía llevar a cabo el proceso de la normalización por medio de la 1FN, 2FN y 3FN, llegando a una conclusión de tal vez no era una buena opción ya que se complica un poco más la implementación y complejidad de la base de datos para un usuario final de nuestra base de datos.

### **2.3.3. Indicadores.**

Registro del modelo relacional del problema planteado para el proyecto.

### **2.3.4. Tiempo.**

**Fecha de inicio y termino de la actividad:** 05/junio/2023

**Nombre de la persona que realizo la actividad:** López Segundo Luis Ivan, Pacheco Saavedra Angel Gael

## **2.4. Script de la creación de las tablas.**

### **2.4.1. Objetivos.**

Con base al modelo relacional normalizado, realizar la creación de las tablas en el manejador PostgreSQL de la base de datos para comenzar con la implementación de la base de datos en una plataforma de desarrollo llamado pgAdmin 4.

### **2.4.2. Plan de actividades:**

- 06/junio/2023: con ayuda del lenguaje de definición de datos (DDL) se crearon las distintas tablas que representan cada entidad de los modelos antes realizados, de tal modo que se procedería a únicamente escribir los comandos para la creación de tablas con CREATE, y de igual manera se asignan las restricciones para cada atributo (si es que tiene), de los cuáles solo se presentarán las restricciones de CASCADE, (NOT) NULL, FOREIGN KEY y PRIMARY KEY. De igual manera, también se crearían distintas secuencias para simular las ventas cada vez que alguien quiera comprar algo.
- Una vez que se realizó la transcripción de la creación de tablas, se procede a revisar en el manejador pgAdmin 4 la asignación de estas tablas, del cuál marca que si se crearon las tablas de manera satisfactoria.

### **2.4.3. Indicadores.**

Registro del modelo relacional con las restricciones planteadas del problema planteado para el proyecto.

### **2.4.4. Tiempo.**

**Fecha de inicio y termino de la actividad:** 06/junio/2023

**Nombre de la persona que realizo la actividad:** Ramírez Bartolo Ignacio, Santiago Martínez Ricardo



## **2.5. Script para el agregado de información.**

### **2.5.1. Objetivos.**

Con base al script que contiene la creación de las tablas, se procede a realizar la inserción de datos con base a los atributos que contiene cada tabla para comenzar con las pruebas de nuestra base de datos.

### **2.5.2. Plan de actividades:**

- 06/junio/2023: una vez hecho el script para la creación de las tablas, se procedió a insertar los datos para cada tabla con el comando INSERT y que así se puedan realizar las pruebas, del cual se agregan los datos con base a los atributos que presenta cada tabla.
- Ahora que se agregaron los datos para las tablas, se procede a probar con el comando SELECT si se agregaron los datos correctamente, del cual si aparecen todos los datos, se puede confirmar de que se logró realizar la inserción de información de manera satisfactoria.

### **2.5.3. Indicadores.**

Registro del modelo relacional y el registro de la creación de tablas en el manejador de la base de datos.

### **2.5.4. Tiempo.**

**Fecha de inicio y termino de la actividad:** 06/junio/2023

**Nombre de la persona que realizo la actividad:** López Segundo Luis Ivan, Pacheco Saavedra Angel Gael, Ramírez Bartolo Ignacio, Santiago Martínez Ricardo.

## **2.6. Script de los códigos de la programación a nivel BD.**

### **2.6.1. Objetivos.**

Con base al script de la creación de las tablas y de la inserción de información, realizar las funciones y triggers que permitan obtener la programación de los requisitos que pide el proyecto.

### **2.6.2. Plan de actividades:**

- 06/junio/2023: una vez hecho el script para la creación de las tablas, se procedió a insertar los datos para cada tabla con el comando INSERT y que así se puedan realizar las pruebas, del cual se agregan los datos con base a los atributos que presenta cada tabla.
- Ahora que se agregaron los datos para las tablas, se procede a probar con el comando SELECT si se agregaron los datos correctamente, del cual si aparecen todos los datos, se puede confirmar de que se logró realizar la inserción de información de manera satisfactoria.

### **2.6.3. Indicadores.**

Registro del modelo relacional y el registro de la creación de tablas en el manejador de la base de datos.

### **2.6.4. Tiempo.**

**Fecha de inicio y termino de la actividad:** 06/junio/2023

**Nombre de la persona que realizo la actividad:** López Segundo Luis Ivan, Pacheco Saavedra Angel Gael, Ramírez Bartolo Ignacio, Santiago Martínez Ricardo.

## **2.7. Documento formal elaborado en L<sup>A</sup>T<sub>E</sub>X**

### **2.7.1. Objetivos.**

Una vez terminada la base de datos, se tiene como objetivo documentar todo el trabajo que se realizó durante la realización del proyecto para que se explique con formalidad que se hizo en cada sección del proyecto para que se entienda de mejor manera como se llevó a cabo la creación de la base de datos.

### **2.7.2. Plan de actividades:**

- 07/junio/2023: Una vez que se terminó la base de datos, se procede a crear el documento que respalda la creación y funcionamiento de la base de datos, del cual es este mismo documento en el que se explica cómo se creó, de tal modo que abarca las siguientes secciones: Introducción (breve descripción del análisis del problema con su propuesta de solución y objetivos), plan de trabajo (describir las actividades que se realizaron con su plan de trabajo), diseño (descripción de cada fase de diseño), implementación (descripción del funcionamiento y código de triggers, funciones, stored procedures para que cumplan con los requerimientos del sistema) y conclusiones de cada integrante.
- 10/junio/2023: Una vez que se terminó el documento, se revisa para observar que cumpla con todos los requisitos pedidos y que logre explicar y justificar la creación de la base de datos con su funcionamiento.

### **2.7.3. Indicadores.**

Registro de los modelos entidad-relación y relacional, registros de la creación de tablas, inserción de información y códigos de programación.

### **2.7.4. Tiempo.**

**Fecha de inicio de la actividad:** 07/junio/2023

**Fecha de termino de la actividad:** 10/junio/2023

**Nombre de la persona que realizo la actividad:** López Segundo Luis Ivan, Ramírez Bartolo Ignacio.






### 3. Gestión de cambios del proyecto.

Peticiones de modificación  
Baja de integrante.

<b>Prioridad</b>	Normal.	
<b>Descripción del error / Adaptación</b>	Miembro del equipo se retira del proyecto antes del inicio del calendario.	
<b>Entorno de ocurrencia</b>	Periodo posterior a la entrega del proyecto, el integrante decidió dar de baja la materia; por lo que no formará parte del equipo de trabajo.	
<b>Aspectos afectados</b>	Distribución neta de trabajo por integrante.	
<b>Solicitado por</b>	Fernando Arreola Franco, cliente.	
<b>Diagnóstico y posibles soluciones</b>	Se deja de contemplar al miembro para la distribución de actividades así como desarrollo de la base de datos.	
<b>Información de apoyo</b>	La distribución de trabajo aún no se encontraba realizada, por lo que no hay redistribución documentable de trabajo.	
<b>Autorización</b>	<b>Revisado por</b>  <b>Responsable de Recursos Humanos</b>  <b>Ricardo Santiago</b>	<b>Aceptado por</b>  <b>Cliente</b>  <b>Fernando Arreola Franco</b>

## 4. Gestión de las Comunicaciones.

La comunicación es un componente fundamental en la gestión de proyectos y juega un papel clave en el éxito de los mismos. El 90 % de la planeación de un proyecto consiste en la comunicación, aunque esto puede variar dependiendo de varios factores, como la naturaleza y complejidad del proyecto, el tamaño del equipo y las necesidades específicas de comunicación. Para este proyecto en particular, se utilizaron los siguientes recursos.

Plataforma utilizada	Uso
<b>WhatsApp</b> 	WhatsApp se implementó como un canal de comunicación ágil y conveniente, permitiendo el intercambio de información en tiempo real y la rápida toma de decisiones. Se crearon grupos de chat específicos para diferentes equipos y roles dentro del proyecto, lo que facilitó la organización y la comunicación eficiente.
<b>Discord</b> 	La utilización de Discord permitió crear canales de comunicación temáticos, los cuales se organizaron de manera lógica y estructurada para abordar distintos aspectos del proyecto. Se crearon canales para discutir y compartir información relacionada con las tendencias identificadas, compartir recursos relevantes, debatir ideas y tomar decisiones.
<b>Google Drive</b> 	Para comenzar, se creó una carpeta principal en Google Drive específicamente para este proyecto. Dentro de esta carpeta, se crearon subcarpetas para cada sección relevante del documento, como la introducción, objetivos, metodología, resultados, conclusiones, entre otras. Esta estructura de carpetas permite mantener una organización clara y sistemática de todos los archivos relacionados con el proyecto.
<b>Google Docs</b> 	Se utilizó Google Docs como una herramienta colaborativa y de edición en tiempo real para la creación y gestión del documento principal. Google Docs permitió a los miembros del equipo trabajar de manera simultánea en el mismo documento, facilitando la comunicación y la colaboración efectiva.
<b>Google Presentaciones</b>  Google Slides	Google Presentaciones proporcionó una interfaz intuitiva y fácil de usar, lo que permitió a los miembros del equipo del proyecto diseñar y estructurar las diapositivas de forma rápida y sencilla. Se seleccionaron plantillas profesionales y adecuadas al tema de la exposición, lo que proporcionó una apariencia visualmente coherente y atractiva en toda la presentación.

## 5. Diseño.

Descripción de lo realizado en las correspondientes fases de diseño de las bases de datos, agregando los resultados de cada una de ellas.

### ■ Análisis de requerimientos.

- Para realizar esta etapa del diseño de la base de datos, únicamente se procedió a analizar el problema propuesto en el PDF para pensar cómo sería la resolución del problema. Primeramente, se comenzó a analizar la descripción del problema, de tal modo que se revisarán los requerimientos de ese problema para comenzar a crear el diseño conceptual, que igualmente se tomarían en cuenta sobre cómo es que se van a realizar las funciones propuestas para nuestra base de datos. Una vez que se observaron los requerimientos que necesita la base de datos, se comenzaría a plantear el resto de las etapas de diseño.

### ■ Diseño conceptual

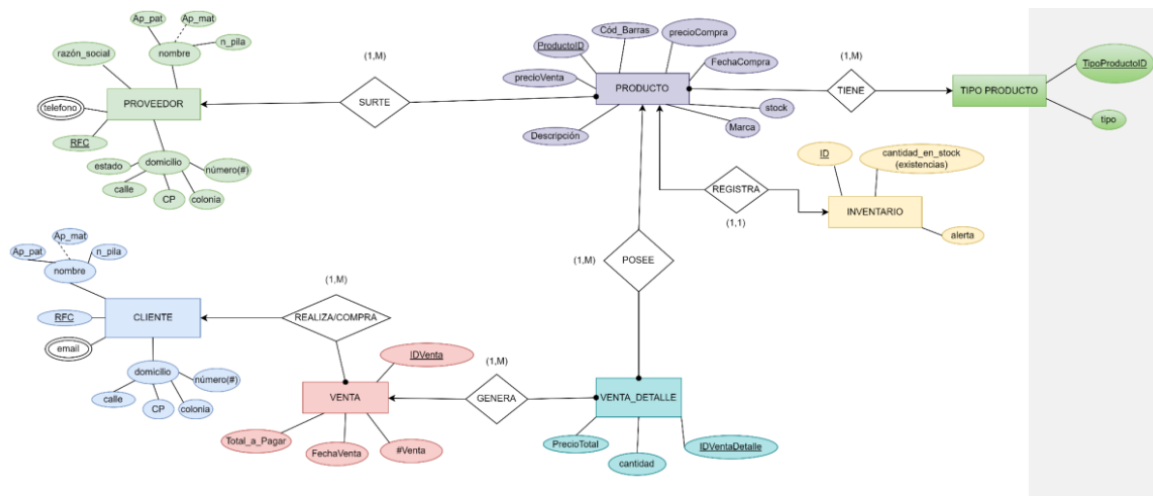
- En un término más simple, aquí se realiza el modelo entidad-relación, del cual se realizó de la misma manera en la que se realizaba en clase. Se procedió a identificar los sustantivos que se presentaban en el problema, los cuales se terminaron encontrando los siguientes sustantivos:
  - PROVEEDOR
  - PRODUCTO
  - CLIENTE
  - VENTA
  - VENTA\_DETALLE
  - INVENTARIO
  - TIPO\_PRODUCTO
- Una vez que se encontraron los sustantivos, se encontraron los siguientes atributos que se relacionaban con su atributo específico:
  - PROVEEDOR: RFC (clave primaria), razón social, domicilio\*, nombre\*\* y teléfonos (atributo multivaluado).
  - PRODUCTO: ID del producto (clave primaria), código de barras, precio al que fue comprado el producto, fecha de compra y cantidad de ejemplares en la bodega (stock). marca, descripción y precio de los regalos.
  - CLIENTE: rfc (clave primaria), nombre\*\*, domicilio\* y al menos un email de los clientes (atributo multivaluado).
  - VENTA: el ID de la venta (clave primaria), el número de venta, fecha de venta y la cantidad total a pagar de la venta.
  - VENTA\_DETALLE: el ID de ese detalle de la venta (clave primaria), la cantidad de cada artículo y precio total a pagar por artículo.
  - INVENTARIO: el ID de cierto artículo en el inventario (clave primaria), la alerta cuando solo quedan 3 o 0 artículos en stock y la cantidad en el inventario (stock).
  - TIPO\_PRODUCTO: el ID del tipo del producto (clave primaria) y el tipo del producto.

\*\* El nombre es un atributo compuesto por nombre de pila, apellido paterno y apellido materno.

\*El domicilio es un atributo compuesto por “estado, calle, código postal, colonia y número”.

- • Una vez que se tienen las entidades con sus atributos, se buscan los verbos que se encargaría de relacionar las entidades. En nuestro caso, solamente encontramos relaciones de tipo 1:1 y 1:M, por lo que se consideró una ausencia de relaciones de tipo M:M. Los verbos que se encontraron junto con las entidades que relaciona y su tipo de relación son:
  - Surte: relaciona PROVEEDOR (cardinalidad 1) y PRODUCTO (cardinalidad M).
  - Tiene: relaciona PRODUCTO (cardinalidad 1) y TIPO\_PRODUCTO (cardinalidad M).
  - Registra: relaciona PRODUCTO (cardinalidad 1) e INVENTARIO (cardinalidad 1).
  - Posee: relaciona PRODUCTO (cardinalidad 1) y VENTA\_DETALLE (cardinalidad M).

- Genera: relaciona VENTA (cardinalidad 1) y VENTA\_DETALLE (cardinalidad M).
- Realiza/compra: relaciona CLIENTE (cardinalidad 1) y VENTA (cardinalidad M).
- Por lo tanto, el modelo entidad-relación quedaría de la siguiente manera:



#### ■ Diseño lógico.

- En un término más simple, aquí se realiza el modelo relacional, del cual se basa en el modelo entidad-relación en el que aquí se procede a transformar el modelo entidad-relación al modelo relacional, de tal modo que se van a transformar entidades fuertes, atributos (simples, compuestos, multivaluados, llaves primarias, llaves foráneas, y restricciones como el (not) null, check, cascade, unique, etc) y relaciones (que en nuestro caso solo serán de tipo 1:M y M:M).
- Con el uso de buenas prácticas al momento de realizar el diseño del modelo entidad relación se logra una estructura de base de datos eficiente y escalable. Esto implica identificar y definir correctamente las entidades, atributos y relaciones, así como establecer las restricciones y reglas necesarias para garantizar la integridad de los datos.
- Mediante el uso y modificación de la base de datos se tienen algunas consideraciones que quedan por parte de los requerimientos del cliente, ya que se debe modificar para que el sistema opere bajo sus necesidades y proyección de crecimiento, por ello es importante partir de una base sólida de trabajo, donde se muestre un comportamiento adecuado para poder evolucionar a soluciones mejores.
- Se deben tener en cuenta las necesidades y proyecciones de crecimiento del cliente, así como las posibles actualizaciones y mejoras futuras lo cual garantiza una mayor satisfacción del cliente y un sistema más robusto y escalable en el largo plazo.
- Se hizo de la manera en cómo se realizó en clase, donde se escribe el nombre de la entidad y se agregan sus atributos con su tipo de dato específico y las restricciones que requiera; y en el caso de las relaciones, si se tiene una relación 1:M se pasa la llave primaria de la relación con cardinalidad 1 a la relación con cardinalidad M como llave foránea, y con la relación de la cardinalidad 1:1 la llave primaria de la relación que menos importancia tiene se pasa a la relación con mayor importancia como llave foránea, de tal modo que el modelo relacional quedaría de la siguiente manera quedaría de la siguiente manera.

PROVEEDOR
<p><b>PROVEEDOR:</b> {ProveedorRFC VARCHAR(13) (PK)</p> <p>ApellidoPaternoProveedor VARCHAR(100),</p> <p>ApellidoMaternoProveedor VARCHAR(100),</p> <p>NombreProveedor VARCHAR(100),</p> <p>RazonSocial VARCHAR(255),</p> <p>EstadoProveedor VARCHAR(100) ,</p> <p>CodigoPostalProveedor VARCHAR(5) ,</p> <p>ColoniaProveedor VARCHAR(100) ,</p> <p>CalleProveedor VARCHAR(100) ,</p> <p>NumeroProveedor VARCHAR(10)}</p> <p><b>TELEFONOPROVEEDOR:</b> { Telefono VARCHAR(15) (PK),</p> <p>RFC VARCHAR(13) (FK)}</p>



CLIENTE
<p><b>CLIENTE:</b> {ClienteRFC VARCHAR(13) (PK),</p> <p>ApellidoPaternoCliente VARCHAR(100),</p> <p>ApellidoMaternoCliente VARCHAR(100),</p> <p>NombreCliente VARCHAR(100),</p> <p>EstadoCliente VARCHAR(100) NOT NULL,</p> <p>CodigoPostalCliente VARCHAR(5) NOT NULL,</p> <p>ColoniaCliente VARCHAR(100) NOT NULL,</p> <p>CalleCliente VARCHAR(100) NOT NULL,</p> <p>NumeroCliente VARCHAR(10) NOT NULL}</p> <p><b>EMAILSCLIENTE:</b> { EmailCliente VARCHAR(255) (PK),</p> <p>ClienteRFC VARCHAR(13) (FK)}</p>

TIPO_PRODUCTO
<p><b>TIPOPRODUCTO:</b> { TipoProductoID INTEGER,</p> <p>Tipo VARCHAR(50) NOT NULL UNIQUE}</p>

PRODUCTO
<p><b>PRODUCTO:</b> { ProductID INTEGER (PK),</p> <p>CodigoBarras VARCHAR(13) NOT NULL UNIQUE,</p> <p>PrecioCompra NUMERIC(8, 2) NOT NULL,</p> <p>Foto TEXT,</p> <p>FechaCompra DATE NOT NULL,</p> <p>Stock INTEGER NOT NULL,</p> <p>Marca VARCHAR(100) UNIQUE,</p> <p>Descripcion TEXT,</p> <p>PrecioVenta NUMERIC(8, 2) NOT NULL,</p> <p>TipoProductID INTEGER (FK),</p> <p>ProveedorRFC VARCHAR(13) (FK)}</p>

VENTA
<p><b>VENTA:</b> { VentaID INTEGER(PK)</p> <p>NumeroVenta VARCHAR(50) NOT NULL</p> <p>FechaVenta DATE NOT NULL</p> <p>ClienteRFC VARCHAR(13) (FK)</p> <p>TotalPagar NUMERIC(10, 2)}</p>

DETALLEVENTA
<b>DETALLEVENTA:</b> { VentaDetalleID INTEGER(PK) VentalD INTEGER (FK) ProductoID INTEGER (FK) Cantidad INTEGER NOT NULL PrecioTotal NUMERIC(10, 2)}

INVENTARIO
<b>INVENTARIO:</b> { InventarioID INTEGER(PK) ProductoID INTEGER (FK) CantidadEnStock INTEGER NOT NULL Alerta BOOLEAN NOT NULL}

Algunas consideraciones que se tuvieron al momento de realizar el mapeo fue hacerlo de una manera similar a como se implementa dentro de SQL, con la finalidad de hacer la creación de las tablas de una manera eficaz y práctica. De esta manera se definieron los tipos de datos con precisión para su posterior declaración dentro de postgres, debido a ello algunos datos, por ejemplo los que hacen referencia a una cantidad monetaria “PrecioVenta NUMERIC(8, 2) NOT NULL” incluyen una definición con uso de precisión, donde son ocho dígitos decimales y los otros dos dígitos fraccionarios. Con la intención de representar centavos en la denominación del tipo de moneda correspondiente, para este caso pesos mexicanos. La definición de entidades, atributos y relaciones se realizó proyectando una estructura de base de datos coherente y consistente con los requerimientos y objetivos del sistema. Se llevó a cabo un análisis detallado de las necesidades del negocio, y se tradujeron en entidades que representan los objetos principales del dominio dentro de un contexto enfocado hacia la administración de una papelería.

- Diseño físico

- En un término más simple, aquí se realiza la implementación de la base de datos, del cual se basa en el modelo entidad-relación y en el modelo relacional. Aquí se basa principalmente en realizar las 3 etapas pedidas en el proyecto, la creación de tablas, inserción de información y programación a nivel de la base de datos.
- Esta etapa puede definirse a continuación en el siguiente bloque, con ayuda de algunas pruebas y descripción de la base de datos, así como su aplicación.

## 6. Implementación.

Justo antes de comenzar el desarrollo del proyecto el primer paso entonces fue investigar el uso, aplicación y modificación de postgresql, específicamente para la creación de la base de datos con el fin de poder integrar cada una de las funciones, procedimientos, triggers, sentencias, o los algoritmos necesarios para poder cubrir los requerimientos de la base de datos planteada y así poder otorgar al cliente de manera efectiva su petición.

En este punto es importante mencionar que la normalización no fue aplicada en su totalidad debido a la naturaleza y complejidad de los datos involucrados en el proyecto. Además, se identificó que la normalización completa podría haber resultado en una estructura compleja y poco intuitiva para los usuarios finales.

Si bien la base de datos no está completamente normalizada, se implementaron medidas en la construcción de la misma para mantener la consistencia y la calidad de los datos almacenados.

Esta decisión fue tomada después de un análisis de los requerimientos y las limitaciones del proyecto, considerando las necesidades específicas de los usuarios y los objetivos a cumplir. Se busca proporcionar una base de datos que sea funcional, comprensible y que satisfaga las necesidades del cliente en términos de acceso eficiente y facilidad de uso.

- *Creación de tablas:* una vez que se obtuvo el modelo relacional, se procede a modificarlo para programarlo con lenguaje de definición de datos (DDL), de tal modo que así se utilicen los comandos de ese lenguaje que son CREATE, ALTER y DROP., del cual como solo se quieren crear las tablas y secuencias se usará únicamente CREATE con las restricciones que se piden que son (NOT) NULL, FOREIGN KEY, CASCADE, PRIMARY KEY, UNIQUE, de tal modo que quedaría de la siguiente manera:
- De igual manera se establecen las condiciones para que se pueda generar dentro del número de venta se tenga un formato similar a "VENT-001", con el prefijo incorporado VENT descrito dentro de la tabla de ventas, utilizando el valor por default asignado, seguido de las condiciones para que mantenga siendo un número compuesto por tres dígitos.

```

CREATE TABLE Proveedores (
    RFC VARCHAR(13) PRIMARY KEY,
    ApellidoPaterno VARCHAR(100),
    ApellidoMaterno VARCHAR(100),
    Nombres VARCHAR(100),
    RazonSocial VARCHAR(255) NOT NULL,
    Estado VARCHAR(100) NOT NULL,
   CodigoPostal VARCHAR(5) NOT NULL,
    Colonia VARCHAR(100) NOT NULL,
    Calle VARCHAR(100) NOT NULL,
    Numero VARCHAR(10) NOT NULL
);

```

Figura 1: Tabla para proveedores.

```

-- Tabla TelefonosProveedor
CREATE TABLE TelefonosProveedor (
    Telefono VARCHAR(15) PRIMARY KEY,
    ProveedorRFC VARCHAR(13),
    FOREIGN KEY (ProveedorRFC) REFERENCES Proveedores(RFC) ON DELETE CASCADE ON UPDATE CASCADE
);

```

Figura 2: Tabla de teléfonos del proveedor.

```
-- Tabla Clientes
CREATE TABLE Clientes (
    RFC VARCHAR(13) PRIMARY KEY,
    ApellidoPaterno VARCHAR(100),
    ApellidoMaterno VARCHAR(100),
    Nombres VARCHAR(100),
    Estado VARCHAR(100) NOT NULL,
    CodigoPostal VARCHAR(5) NOT NULL,
    Colonia VARCHAR(100) NOT NULL,
    Calle VARCHAR(100) NOT NULL,
    Numero VARCHAR(10) NOT NULL
);
```

Figura 3: Tabla para los clientes.

```
-- Tabla EmailsCliente
CREATE TABLE EmailsCliente (
    Email VARCHAR(255) PRIMARY KEY,
    ClienteRFC VARCHAR(13),
    FOREIGN KEY (ClienteRFC) REFERENCES Clientes(RFC) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Figura 4: Tabla para los correos del cliente.

```
CREATE SEQUENCE tipoproductos_seq START 1;
-- Tabla TipoProductos
CREATE TABLE TipoProductos (
    TipoProductoID INTEGER DEFAULT NEXTVAL('tipoproductos_seq') PRIMARY KEY,
    Tipo VARCHAR(50) NOT NULL,
    CONSTRAINT uk_tipo_producto UNIQUE (Tipo)
);
```

Figura 5: Tabla para tipo de productos.

```
-- Tabla Productos
CREATE TABLE Productos (
    ProductoID INTEGER DEFAULT NEXTVAL('productos_seq') PRIMARY KEY,
    CodigoBarras VARCHAR(13) NOT NULL,
    PrecioCompra NUMERIC(8, 2) NOT NULL,
    Foto TEXT,
    FechaCompra DATE NOT NULL,
    Stock INTEGER NOT NULL,
    Marca VARCHAR(100),
    Descripcion TEXT,
    PrecioVenta NUMERIC(8, 2) NOT NULL,
    TipoProductoID INTEGER,
    ProveedorRFC VARCHAR(13),
    FOREIGN KEY (TipoProductoID) REFERENCES TipoProductos(TipoProductoID),
    FOREIGN KEY (ProveedorRFC) REFERENCES Proveedores(RFC)
    CONSTRAINT uk_codigo_barras UNIQUE (CodigoBarras)
);
CREATE SEQUENCE venta_id_seq START 1;
CREATE SEQUENCE numero_venta_seq START 1;
```

```
CREATE INDEX idx_productos_productoid ON Productos (ProductoID);
```

Figura 6: Tabla productos.

```
Tabla Ventas
CREATE TABLE Ventas (
    VentaID INTEGER DEFAULT NEXTVAL('venta_id_seq') PRIMARY KEY,
    NumeroVenta VARCHAR(50) NOT NULL DEFAULT 'VENT-' || LPAD(CAST(NEXTVAL('numero_venta_seq') AS TEXT), 3, '0'),
    FechaVenta DATE NOT NULL,
    ClienteRFC VARCHAR(13),
    TotalPagar NUMERIC(10, 2),
    FOREIGN KEY (ClienteRFC) REFERENCES clientes(RFC)
);
```

Figura 7: Tabla ventas



```
-- Tabla Venta_Detalles
CREATE SEQUENCE venta_detalle_seq START 1;
CREATE TABLE Venta_Detalles (
    VentaDetalleID INTEGER DEFAULT NEXTVAL('venta_detalle_seq') PRIMARY KEY,
    VentaID INTEGER NOT NULL,
    ProductoID INTEGER NOT NULL,
    Cantidad INTEGER NOT NULL,
    PrecioTotal NUMERIC(10, 2),
    FOREIGN KEY (VentaID) REFERENCES Ventas(VentaID) ON UPDATE CASCADE,
    FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID) ON UPDATE CASCADE
);
```

Figura 8: Tabla detalle de ventas.

```
-- Tabla Inventarios
CREATE SEQUENCE inventario_id_seq START 1;
CREATE TABLE Inventarios (
    InventarioID INTEGER DEFAULT nextval('inventario_id_seq'),
    ProductoID INTEGER NOT NULL,
    CantidadEnStock INTEGER NOT NULL,
    Alerta BOOLEAN NOT NULL DEFAULT FALSE,
    FOREIGN KEY (ProductoID) REFERENCES Productos(ProductoID) ON UPDATE CASCADE
);
```

Figura 9: Tabla detalle de inventario.

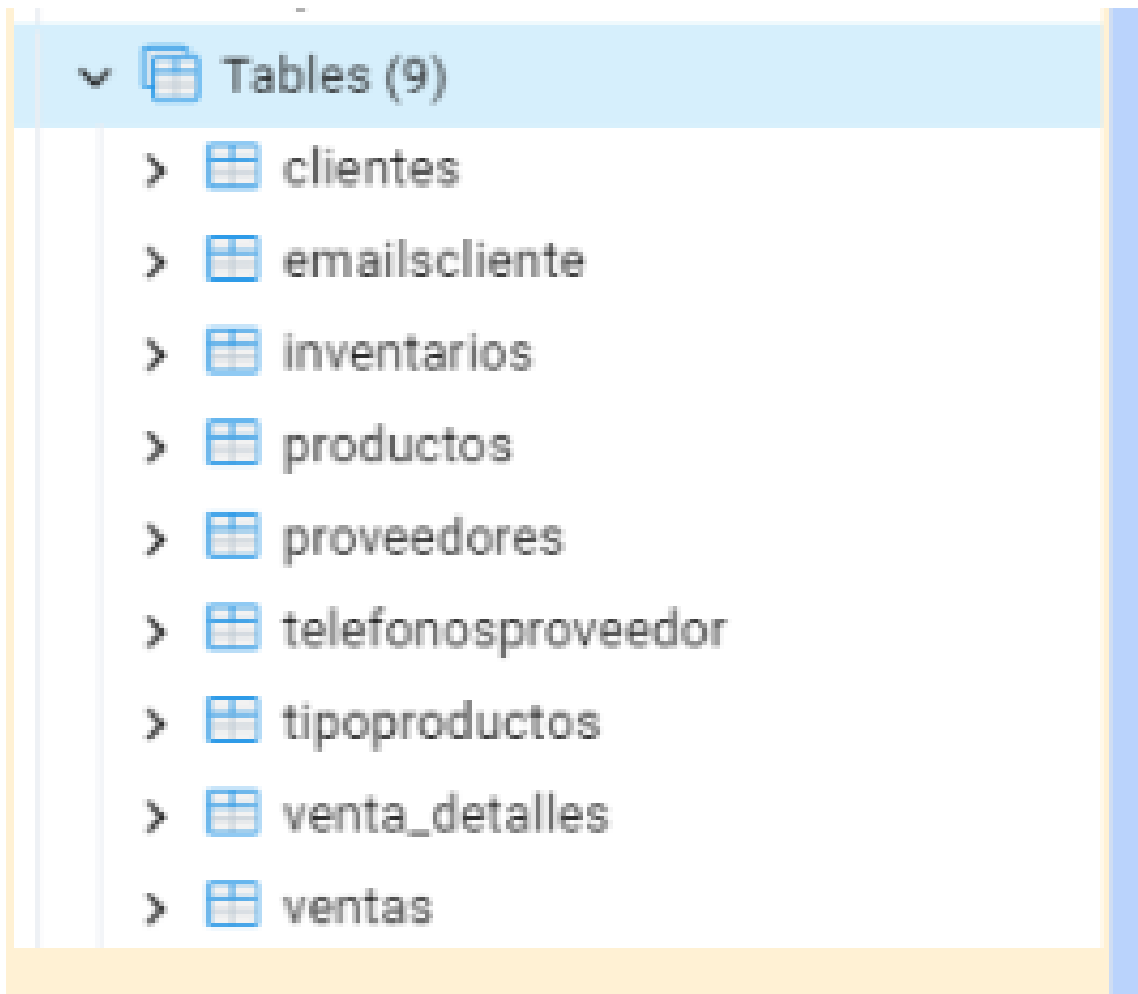


Figura 10: Comprobando la existencia de las tablas dentro de la base.

- Inserción de información: con las tablas creadas, se utilizará el lenguaje de manipulación de datos (DML) para insertar, actualizar y borrar datos, de tal modo que como solo se van a insertar los datos, sólo se utilizará el comando INSERT INTO “nombre\_tabla” VALUES, de tal modo que para nuestra base de datos, ya vendrían incluidos mínimo 10 tuplas de información que abarcan los atributos de cada relación, de tal modo que quedarían de la siguiente manera, tomando en cuenta los atributos que fueron descritos en la sintaxis:

Algunos ejemplos de inserción de datos para las tablas.

```
-- Tabla Proveedores
INSERT INTO Proveedores (RFC, ApellidoPaterno, ApellidoMaterno, Nombres, RazonSocial, Estado, CodigoPostal, Colonia, Calle, Numero)
VALUES ('PROV001', 'Pérez', 'Gómez', 'Juan', 'Proveedor 1', 'Estado 1', 'CP1', 'Colonia 1', 'Calle 1', '1');

-- Tabla TelefonosProveedor
INSERT INTO TelefonosProveedor (Telefono, ProveedorRFC)
VALUES ('1234567890', 'PROV001');

-- Tabla Clientes
INSERT INTO Clientes (RFC, ApellidoPaterno, ApellidoMaterno, Nombres, Estado, CodigoPostal, Colonia, Calle, Numero)
VALUES ('CLI001', 'López', 'García', 'María', 'Estado 2', 'CP2', 'Colonia 2', 'Calle 2', '2');

-- Tabla EmailsCliente
INSERT INTO EmailsCliente (Email, clienteRFC)
VALUES ('cliente@example.com', 'CLI001');

-- Tabla TipoProductos
INSERT INTO TipoProductos (Tipo)
VALUES ('Papel'), ('Oficina'), ('Maqueta');

-- Tabla Productos
INSERT INTO Productos (CodigoBarras, PrecioCompra, Foto, FechaCompra, Stock, Marca, Descripción, PrecioVenta, TipoProductoID)
VALUES ('1234567890123', 10.50, 'imagen1.jpg', '2023-01-01', 100, 'Marca 1', 'Descripción 1', 15.99, 1),
('9876543210987', 8.75, 'imagen2.jpg', '2023-02-01', 50, 'Marca 2', 'Descripción 2', 12.99, 2);

-- Tabla Ventas
INSERT INTO Ventas (FechaVenta, clienteRFC)
VALUES ('2023-05-01', 'CLI001'), ('2023-05-02', 'CLI001');

-- Tabla Venta_Detalles
INSERT INTO Venta_Detalles (VentaID, ProductoID, Cantidad)
VALUES (1, 1, 3), (1, 2, 2), (2, 1, 2), (2, 2, 1);
```

Corroboramos que se guarden los datos:

Query

Query History

Scratch Pad

1

2

SELECT \* FROM Productos;

Data Output

Messages

Notifications

	productoid [PK] integer	codigobarras character varying (13)	preciocompra numeric (8,2)	foto text	fechacompra date	stock integer	marca character varying (100)	descripcion text
1	1	1234567890123	10.50	imagen1.jpg	2023-01-01	100	Marca 1	Descripción 1
2	2	9876543210987	8.75	Imagen2.jpg	2023-02-01	50	Marca 2	Descripción 2

- *Programación a nivel base de datos*: una vez que las tablas ya contienen la información guardada, se procede a finalmente programar las funciones, triggers, vistas, transacciones y stored procedures. Igualmente se aplicaría el lenguaje de consulta de datos con el comando SELECT que permite realizar las consultas que se piden en el proyecto, de tal modo que de esta forma se realizarán los requerimientos con su sintaxis:
- Para ello se separan cada uno de los problemas para resolver de manera individual a través del caso de uso que se requiera resolver con la finalidad de poder mantener un orden en la descripción del proceso de solución.

- Generar una vista que se asemeje a una factura: *SELECT \* FROM Factura WHERE NumeroVenta = 'VENT-001'*;

```

7  -- Vista Factura
8  CREATE OR REPLACE VIEW Factura AS
9  SELECT v.NumeroVenta, v.FechaVenta, c.Nombres, c.RFC, c.Estado, c.CodigoPostal,
10         c.Colonia, c.Calle, c.Numero,
11         vd.Cantidad, p.Marca, p.Descripcion, p.PrecioVenta, vd.PrecioTotal
12  FROM Ventas v
13  JOIN Clientes c ON v.ClienteRFC = c.RFC
14  JOIN Venta_Detalles vd ON v.VentaID = vd.VentaID
15  JOIN Productos p ON vd.ProductoID = p.ProductoID;

```

Query

Query History

Scratch Pad

1

SELECT \* FROM Factura WHERE NumeroVenta = 'VENT-001';

Data Output

Messages

Notifications

	numeroventa character varying (50)	fechaventa date	nombres character varying (100)	rfc character varying (18)	estado character varying (100)	codigo postal character varying (5)
1	VENT-001	2023-05-01	María	CL1001	Estado 2	CP2
2	VENT-001	2023-05-01	María	CL1001	Estado 2	CP2

	colonia character varying (100)	calle character varying (100)	numero character varying (10)	cantidad integer	marca character varying (100)	descripcion text	precioventa numeric (8,2)
1	Colonia 2	Calle 2	2	3	Marca 1	Descripción 1	15.99
2	Colonia 2	Calle 2	2	2	Marca 2	Descripción 2	12.99

precioventa	preciototal
numeric (8,2)	numeric (10,2)
15.99	47.97
12.99	25.98

Vista se crea utilizando la sentencia CREATE OR REPLACE VIEW y combina datos de varias tablas: Ventas, Clientes, Venta.Detalles y Productos. La vista muestra información relacionada con las facturas, incluyendo el número de venta, la fecha de venta, los detalles del cliente y los detalles de los productos vendidos. La vista "Factura" selecciona varias columnas de las tablas relacionadas y las combina en una única tabla con la finalidad de obtener toda la información disponible del cliente para poder emitir un recibo funcional.

#### **Función ventas\_ganancia:**

- Basándose en una fecha, o una fecha de inicio y fecha de fin, se regresa la cantidad total que se vendió y la ganancia correspondiente en esa fecha/periodo: *SELECT \* FROM ventas\_ganancia('2023-06-01', '2023-06-01')*;

```
-- Función ventas_ganancia
CREATE OR REPLACE FUNCTION ventas_ganancia(p_fecha_inicio DATE, p_fecha_fin DATE)
RETURNS TABLE(total_vendido NUMERIC, ganancia NUMERIC) AS $$
BEGIN
    RETURN QUERY
        SELECT CAST(SUM(vd.Cantidad) AS NUMERIC) AS total_vendido, CAST(SUM(vd.PrecioTotal - (vd.Cantidad * p.PrecioCompra)) AS NUMERIC) AS ganancia
        FROM Ventas v
        INNER JOIN Venta_Detalles vd ON v.VentaID = vd.VentaID
        INNER JOIN Productos p ON vd.ProductoID = p.ProductoID
        WHERE v.FechaVenta BETWEEN p_fecha_inicio AND p_fecha_fin;
END; $$ LANGUAGE plpgsql;
```

Query

Query History

1

SELECT \* FROM ventas\_ganancia('2023-05-01', '2023-05-021');

Data Output

Messages

Notifications

	total_vendido numeric	ganancia numeric
1	8	40.17

Esta función recibe dos parámetros de tipo DATE: p\_fecha\_inicio y p\_fecha\_fin, que representan el rango de fechas dentro del cual se calcularán las ventas y la ganancia. Dentro del bloque BEGIN, la función utiliza la cláusula RETURN QUERY para devolver los resultados de la consulta SELECT. La consulta se encarga de calcular la suma de la cantidad vendida y la ganancia generada durante el rango de fechas especificado. Finalmente, la función devuelve una tabla con dos columnas: total\_vendido y ganancia, que representan el total vendido y la ganancia generada durante el rango de fechas especificado.

- Cuando se compre el artículo, el stock se decrementa por cada artículo vendido. Mandará alerta si quedan menos 3 artículos en stock al completar un pedido, y si el stock llega a 0, se aborta la transacción. Se tiene que actualizar el total a pagar por artículo y el total a pagar por la venta:

```

-- Modificar la función actualizar_stock_producto
CREATE OR REPLACE FUNCTION actualizar_stock_producto() RETURNS TRIGGER AS $$
BEGIN
    UPDATE Productos
    SET Stock = Stock - NEW.Cantidad
    WHERE ProductoID = NEW.ProductoID;

    UPDATE Inventarios
    SET CantidadEnStock = CantidadEnStock - NEW.Cantidad
    WHERE ProductoID = NEW.ProductoID;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
--

```

Esta función es un trigger function que se ejecuta después de insertar registros en la tabla Venta.Detalles. Su objetivo es actualizar el stock de los productos en la tabla Productos y en la tabla Inventarios después de una venta. Resta la cantidad vendida del stock actual y actualiza los valores correspondientes en ambas tablas. También actualiza el campo TotalPagar en la tabla Ventas con el nuevo total de la venta.

```

-- El disparador actualizar_stock_producto_trigger
CREATE TRIGGER actualizar_stock_producto_trigger
AFTER INSERT ON Venta_Detalles
FOR EACH ROW
EXECUTE FUNCTION actualizar_stock_producto();
CREATE OR REPLACE FUNCTION actualizar_stock_producto() RETURNS TRIGGER AS $$
DECLARE
    nuevo_stock INTEGER;
BEGIN
    SELECT Stock - NEW.Cantidad INTO nuevo_stock
    FROM Productos
    WHERE ProductoID = NEW.ProductoID;
    IF nuevo_stock < 0 THEN
        RAISE EXCEPTION 'Stock insuficiente para completar la transacción';
    ELSIF nuevo_stock = 0 THEN
        RAISE EXCEPTION 'El stock ha llegado a cero. Abortando la transacción';
    ELSIF nuevo_stock < 3 THEN
        RAISE NOTICE 'Quedan menos de 3 productos en stock. Se ha generado una alerta';
    END IF;
    UPDATE Productos
    SET Stock = nuevo_stock
    WHERE ProductoID = NEW.ProductoID;
    UPDATE Inventarios
    SET CantidadEnStock = nuevo_stock
    WHERE ProductoID = NEW.ProductoID;
    UPDATE Ventas
    SET TotalPagar = (
        SELECT SUM(PrecioTotal)
        FROM Venta_Detalles
        WHERE VentaID = NEW.VentaID
    )
    WHERE VentaID = NEW.VentaID;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

```

Este trigger se dispara después de insertar registros en la tabla Venta\_Detalles. Ejecuta la función actualizar\_stock\_producto definida anteriormente, lo que actualiza el stock de los productos y el campo TotalPagar en la tabla Ventas.

## Pruebas.

```
1 INSERT INTO Venta_Detalles (VentaID, ProductoID, Cantidad)
2 VALUES (3, 2, 3);
```

Data Output Messages Notifications

ERROR: Stock insuficiente para completar la transacción  
CONTEXT: función PL/pgSQL actualizar\_stock\_producto() en la línea 10 en RAISE  
SQL state: P0001

```
1 INSERT INTO Venta_Detalles (VentaID, ProductoID, Cantidad)
2 VALUES (3, 2, 45);
```

Data Output Messages Notifications

NOTICE: Quedan menos de 3 productos en stock. Se ha generado una alerta  
INSERT 0 1

Query returned successfully in 132 msec.



## Función de control: Precio Total

```
-- Función calcular_precio_total
CREATE OR REPLACE FUNCTION calcular_precio_total()
RETURNS TRIGGER AS $$
BEGIN
    NEW.PrecioTotal := (
        SELECT PrecioVenta * NEW.Cantidad
        FROM Productos
        WHERE ProductoID = NEW.ProductoID
    );
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Disparador calcular_precio_total_trigger
CREATE TRIGGER calcular_precio_total_trigger
BEFORE INSERT OR UPDATE ON Venta_Detalles
FOR EACH ROW
EXECUTE FUNCTION calcular_precio_total();
```

La función `calcular_precio_total` y el disparador `calcular_precio_total_trigger` están relacionados y se utilizan para calcular y actualizar el precio total de un detalle de venta en la tabla `Venta_Detalles`. La función `calcular_precio_total` se ejecuta antes de insertar o actualizar un registro en la tabla `Venta_Detalles`. Su objetivo principal es calcular el precio total del detalle de venta en función del precio de venta unitario y la cantidad de productos vendidos.

Cuando se dispara el evento de inserción o actualización en la tabla `Venta_Detalles`, el disparador asociado invocará automáticamente esta función. La función asigna el valor del precio total calculado a la columna "PrecioTotal" del nuevo registro (NEW). Para calcular el precio total, la función realiza una consulta a la tabla `Productos`, donde obtiene el precio de venta unitario y lo multiplica por la cantidad de productos vendidos (NEW.Cantidad) para obtener el precio total. Luego, la función devuelve el nuevo registro (NEW).

De manera más resumida o platicada, la función `calcular_precio_total` se encarga de calcular el precio total de un detalle de venta en función del precio de venta unitario y la cantidad de productos vendidos. El disparador `calcular_precio_total_trigger` se asocia a la tabla `Venta_Detalles` y llama a la función `calcular_precio_total` antes de insertar o actualizar un registro en la tabla. De esta manera, el precio total se calcula automáticamente y se actualiza en el registro correspondiente de `Venta_Detalles`.

- Lista de productos con menos de 3 en stock: `SELECT Descripcion FROM Productos WHERE Stock < 3;`

Query Query History

```
1 select descripcion
2 from productos where stock<3;
3
4
```

Data Output Messages Notifications

	descripcion text
1	Descripción 2

```
1 SELECT Descripción
2 FROM Productos
3 WHERE Stock < 3;
4
```

- Regresar utilidad por medio del código de barras: *SELECT calcular\_utilidad('1234567890123');*

```
--Calcular utilidad por el código de barras
CREATE OR REPLACE FUNCTION calcular_utilidad(p_codigo_barras VARCHAR(13))
RETURNS NUMERIC AS $$
DECLARE
    v_precio_compra NUMERIC;
    v_precio_venta NUMERIC;
BEGIN
    SELECT PrecioCompra INTO v_precio_compra
    FROM Productos
    WHERE CodigoBarras = p_codigo_barras;

    SELECT PrecioVenta INTO v_precio_venta
    FROM Productos
    WHERE CodigoBarras = p_codigo_barras;

    RETURN v_precio_venta - v_precio_compra;
END; $$ LANGUAGE plpgsql;
```

La función toma un parámetro p\_codigo\_barras, que es un VARCHAR(13). Este parámetro será el código de barras del producto del cual queremos calcular la utilidad.

Declaramos dos variables v\_precio\_compra y v\_precio\_venta que son de tipo NUMERIC. Estas variables almacenarán el precio de compra y el precio de venta del producto, respectivamente.

Luego, realizamos una consulta SQL SELECT para obtener el PrecioCompra del producto que tiene el CodigoBarras igual al parámetro p\_codigo\_barras. Guardamos el resultado en la variable v\_precio\_compra.

Realizamos una segunda consulta SQL SELECT para obtener el PrecioVenta del mismo producto. Guardamos el resultado en la variable v\_precio\_venta.

Finalmente, devolvemos la diferencia entre el PrecioVenta y el PrecioCompra. Esto será la utilidad que obtenemos de la venta de ese producto.

Query	Query History	Scratch
1	SELECT calcular_utilidad('1234567890123');	
2		
3		

Data Output	Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🔍</div> <div>⬇️</div> <div>📈</div> </div>		
	<div>calcular_utilidad</div> <div>numeric</div> <div>🔒</div>	
1	5.49	

- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la

elección en ambos aspectos: *CREATE INDEX idx\_productos\_productoid ON Productos (ProductoID);*

El índice creado para este proyecto es de tipo “Clustered”, por lo que las tuplas se van a ir almacenando con su índice respectivo conforme se vayan agregando. Se eligió aplicar el índice en el ID del producto, puesto que nuestra entidad “PRODUCTO” es la que más se relaciona con el resto de entidades, por lo que vendría útil una organización de los datos con respecto a los productos que se tienen almacenados en la papelería, por lo que sí llega a ayudar ordenarlos por índices para que se tenga un orden con la información proporcionada.

```
CREATE INDEX idx_productos_productoid ON Productos (ProductoID);
```

## 7. Conclusiones.

### **Pacheco Saavedra Angel Gael:**

Se logró cumplir con un conjunto de objetivos clave, a la vez que se enfrentaron desafíos que permitieron un crecimiento significativo y una comprensión más profunda del diseño y la gestión de bases de datos.

Dificultades: Uno de los obstáculos principales fue la gestión de las relaciones entre las tablas y la implementación de claves foráneas para garantizar la integridad referencial. Esto se vio particularmente en el manejo del stock de productos, donde el desafío residía en actualizar correctamente los niveles de stock después de cada venta. Sin embargo, a través de un análisis cuidadoso y una comprensión sólida de las relaciones entre las tablas, se superó este desafío.

Retos: El desafío principal en este proyecto fue el uso de disparadores y funciones para automatizar tareas específicas, como el cálculo del precio total de cada detalle de venta y la actualización del stock en las tablas de productos e inventarios. Asegurar que estos disparadores funcionaran de manera eficiente y sin errores fue crucial para mantener la precisión y la actualización de los datos. Además, el diseño de la base de datos necesitaba ser lo suficientemente flexible para admitir cambios y adiciones futuras.

Aciertos: A pesar de los desafíos, se lograron varios aciertos significativos. El diseño de la base de datos con un alto nivel de normalización ayudó a evitar la redundancia de datos y a asegurar la coherencia de los datos. Además, los disparadores y funciones implementados ayudaron a automatizar ciertas tareas y a garantizar la precisión y la actualización de los datos. Finalmente, la creación de índices para mejorar el rendimiento y la implementación de secuencias para la creación de IDs únicos demostraron ser exitosos en la mejora de la eficiencia de la base de datos.

En conclusión, este proyecto final de la base de datos ha sido una prueba exhaustiva de las habilidades y conocimientos adquiridos en los últimos seis semestres de la carrera de Ingeniería en Computación. La capacidad de superar los desafíos presentados y de lograr con éxito el diseño e implementación de una base de datos robusta y eficiente es un testimonio de las habilidades técnicas desarrolladas y de la capacidad para aplicarlas en situaciones prácticas y complejas. Este proyecto sirve como una base sólida para futuras exploraciones y desarrollos en el campo de las bases de datos y la ingeniería en computación.

### **Santiago Martínez Ricardo.**

Cuando se inició el proyecto la incertidumbre, así como el riesgo eran demasiado grandes, realmente la implementación de una base de datos consideraba un reto abrumador, ya que este proyecto en particular unifica todos los temas desarrollados a través del semestre.

El mayor reto sin duda fue aprender el lenguaje de consultas, sobre todo en su parte DML, ya que esto implicaba un reto de lógica mental para poder estructurar cada una de las consultas y generar las funciones, así como los triggers correspondientes o la generación de las vistas, esto me supuso un reto ya que en lo particular me es un poco complicado aprender a programar, pero a pesar de ello lo disfruto, ya que me llena de felicidad superar un reto. La parte del diseño de la base de datos fue todo un reto al momento de definir algunos parámetros y el simple hecho de construirla desde cero generó un alto impacto a la responsabilidad del diseño, si bien se partieron de varias ideas, solo hubo una única que cumplió con todos los alcances requeridos por el proyecto.

A pesar de ser solo una simulación, este proyecto ayuda a comprender cómo sería un escenario real donde se tienen requerimientos que cumplir en tiempos definidos y responsabilidades establecidas, considero que un acierto al momento de diseñar la base de datos fue no complicar más las cosas utilizando normalización cuando no parece que fuera necesario utilizarlo. Me gustaría mejorar mis conocimientos dentro del campo de las bases de datos y tal vez en un futuro dedicarme a ese rubro.

De igual forma, el aprender a utilizar latex fue complicado y poco intuitivo, ya que considera un esfuerzo adicional e invita a salir de la zona de confort para evolucionar a otro nivel en cuanto a formalidad en los textos redactados.

A través de la elaboración del proyecto aprendí que sé muchas más cosas de las que yo esperaba conocer, superé mis expectativas y a pesar de entender casi lo básico puedo decir que este proyecto en particular puso a prueba la confianza en mí mismo y mientras se resolvía pude superar ese miedo a progresar.

### **Ramírez Bartolo Ignacio,**

Luego de realizar este proyecto, se logró cumplir con ciertos objetivos que se tenían planteados, de tal

modo que se concluyó con ciertas especificaciones:

- Primeramente, se llegó a tener una gran cantidad de dificultades que presentaban ciertos obstáculos para comenzar con el proyecto por la complejidad que se tenía para crear una base de datos, de tal modo que al inicio se pensó que no se lograría concluir satisfactoriamente las actividades del proyecto. Conforme pasó el tiempo mientras se iban realizando las actividades, se comenzó a tener mejor motivación mientras íbamos concluyendo el proyecto de manera rápida, de tal modo que se logró concluir cada actividad de la creación de la base de datos de manera satisfactoria. Una de las dificultades principales que se tuvo en la realización del proyecto se basó en la creación de las funciones para complementar el funcionamiento de la base de datos, de tal modo que mientras se iba repasando el tema de programación se iba desarrollando de manera satisfactoria la programación de los triggers y funciones de nuestra base de datos.
- Los principales retos que se presentaron en el proyecto se basó en el aprendizaje de la programación de la base de datos con todos los requisitos que pedía, de tal modo que sí representaba un reto por no tener una idea sobre cómo implementar funciones y triggers, lo cual se tuvo que aprender didácticamente sobre ese tema, del cual se iba logrando conforme aplicamos los conocimientos de ese tema mientras se iban creando las funciones y triggers. Igualmente se tuvo como reto completar con la realización del proyecto a pesar de todas las dificultades que se presentaban durante la realización del proyecto, del cual pudo completarse de manera satisfactoria.
- Este proyecto logró darme ciertos aciertos cuando realicé el proyecto, del cuál me ayudó a repasar todos los temas que se vieron en el curso para que me ayude a aprender y memorizar todo para el examen final que se realizaría de la materia, por lo que de igual manera, este proyecto hizo que recordará y pusiera en buenas prácticas todo lo relacionado con respecto a lo que tengo que saber sobre las bases de datos.

Por ende, se logró llegar a la conclusión de que se logró proponer una solución sobre el problema propuesto en este proyecto, de tal modo que esto me permitió repasar todos los temas de las bases de datos, desde las fases de diseño hasta la implementación en una base de datos, por lo que siento que este proyecto embarca todo lo que se vió durante el curso, y ayudó a que se remarcaron todos los errores que provocaba cuando aplicaba los conocimientos a la práctica, por lo que se puede decir que el proyecto fue útil y se desarrolló completamente correcto.

### **López Segundo Luis Ivan.**

A lo largo del proyecto se presentaron varias dificultades así como retos, sobre todo la implicación de aplicar todos los temas y conocimientos adquiridos en clase tanto teóricos como prácticos y la habilidad adquirida en el laboratorio de la materia.

Una de las partes que más se pudieron complicar es el uso y el buen manejo de llaves tanto primarias como foráneas para poder relacionar a cada uno de las tablas, Asimismo también algo que se dificulta fue la creación de diferentes estructuras como lo puede ser los joins e índices para posibles consultas que requiera el usuario al momento de utilizar la base de datos y que se requiera información.

Algo que también se complicó fue la creación tanto de funciones así como de triggers ya que estos fueron los últimos elementos que se vieron a lo largo del curso, por ende, son más complejos tanto para su creación y el uso de los mismos, ya que estos requieren de un nivel de abstracción mayor así como un manejo de lógica para su correcto funcionamiento.

También a lo largo del proyecto fue fundamental el buen aprendizaje del lenguaje de datos SQL y cómo es que se aplica específicamente en Postgres ya que esta interfaz tiene sus limitantes así como ventajas.

Un gran reto propuesto en el presente proyecto fue la densidad y los alcances del mismo ya que es más complejo en comparación de los que se realizó a lo largo del semestre como lo fueron las prácticas presentadas en el laboratorio, por esta razón, fue necesario aplicar de buena forma las habilidades obtenidas y el manejo de el lenguaje SQL.

Finalmente se concluye que el proyecto fue fundamental para la aplicación de los conocimientos en un entorno más real de trabajo y mismo que nos sirve para introducir a ejemplos que se puedan presentar en un futuro campo laboral donde se piden funcionalidades específicas y limitantes, se lograron aplicar de buena manera los conocimientos a lo largo del curso y sentar mejor las bases de los mismos, así como reafirmar el manejo de las bases de datos como para su creación, uso, seguridad y manejo de las mismas.

## 8. Referencias.

- PostgreSQL Global Development Group. (2021). Documentación de PostgreSQL 13.4. Recuperado de <https://www.postgresql.org/docs/13/>
- Arreola, F. (2023). Presentación del tema VI: [Diseño físico de una base de datos]. Consultado en: [https://github.com/FernandoArreolaF/Bases1UNAM/blob/master/Presentaciones/Tema\\_VI.pdf](https://github.com/FernandoArreolaF/Bases1UNAM/blob/master/Presentaciones/Tema_VI.pdf)
- Arreola, F. (2023). Presentación del tema VII: [Lenguaje de consulta de datos]. Consultado en: [https://github.com/FernandoArreolaF/Bases1UNAM/blob/master/Presentaciones/Tema\\_VII.pdf](https://github.com/FernandoArreolaF/Bases1UNAM/blob/master/Presentaciones/Tema_VII.pdf)