UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ingenieria

PROYECTO

Sistema de gestión de ventas e inventario de Papelería

Grupo 5

Semestre 2020-2

BASES DE DATOS

Profesor: Ing. Fernando Arreola Franco

Integrantes:
Aguilera Ortiz, Alfredo
Barrientos Vaena, Luis Mauricio
Otero García, Christian
Vivanco Quintanar, Diego Armando

22 de mayo de 2020

1. PROBLEMA A RESOLVER Y REQUERIMIENTOS

1.1. Objetivo

El alumno analizará una serie de requerimientos y propondrá una solución que atienda a los mismos, aplicando los conceptos vistos en el curso.

2. Introducción

Este proyecto está diseñado bajo los parámetros requeridos para la gestión de ventas e inventario de una papelería utilizando el DBMS POSTGRESQL en el cual se permita a sus clientes realizar compras de articulos de papeleria, regalos, impresiones y recargas; además los empleados de la papelería puedan llevar un control de las ventas y registro de sus productos. Se ofrece además un sitio web para registrar a los clientes que ingresen a la página, donde pueden observar la variedad de productos que existen así como la descripción, precio y cantidad que se tiene de estos en el almacén, con el fin de dar a los clientes la mejor experiencia para que seleccionen sus artículos y realicen sus compras.

2.1. Descripción del problema

El problema se divide en dos partes:

2.1.1. Parte uno:

Consiste en el diseño de una base de datos. Una cadena de papelerías busca innovar la manera en que almacena su información, y los contratan para que desarrollen los sistemas informáticos para satisfacer los siguientes requerimientos: Se desea tener almacenados datos como la razón social, domicilio, nombre y teléfonos de los proveedores, razón social, nombre, domicilio y al menos un email de los clientes. Es necesario tener un inventario de los productos que se venden, en el que debe guardarse el código de barras, precio al que fue comprado el producto, fecha de compra y cantidad de ejemplares en la bodega (stock). Se desea guardar la marca, descripción y precio de los regalos, artículos de papelería, impresiones y recargas, siempre y cuando se tenga su correspondiente registro en el inventario. Debe también guardarse el número de venta, fecha de venta y la cantidad total a pagar de la venta, así como la cantidad de cada artículo y precio total a pagar por artículo. Además, se requiere que:

- Al recibir el código de barras de un producto, regrese la utilidad.
- Cada que haya la venta de un artículo, deberá decrementarse el stock por la cantidad vendida de ese artículo. Si el valor llega a cero, abortar la transacción. Si hay menos de 3, emitir un mensaje.
- Dada una fecha, o una fecha de inicio y fecha de fin, regresar la cantidad total que se vendió en esa fecha/periodo.
- Permitir obtener el nombre de aquellos productos de los cuales hay menos de 3 en stock.

- De manera automática se genere una vista que contenga información necesaria para asemejarse a una factura de una compra.
- Crear al menos, un índice, del tipo que se prefiera y donde se prefiera. Justificar el porqué de la elección en ambos aspectos.

Tomar en cuenta las siguientes consideraciones:

- Puede haber distintas soluciones al problema.
- Los requerimientos enlistados anteriormente, deberán ser realizados por medio de PLSQL, con los elementos que se consideren adecuados para resolverlos.
- El número de venta debe tener un formato similar a "VENT−001", prefijo VENT, seguido de un guión y un número secuencial.
- Donde este presente el atributo domicilio, está compuesto por estado, código postal, colonia, calle y número.
- El diseño debe satisfacer todos los principios de diseñoo, los requerimientos anteriores y un buen manejo de información.

2.1.2. Parte dos

Una vez diseñada y lista la base de datos, se debe crear una interfaz gráfica vía app móvil o web, que permita:

- Agregar la información de un cliente.
- Ingresar una venta, de hasta 3 artículos, los cuales podrán seleccionarse de una lista de opciones, permitir ingresar la cantidad, calcular el costo total de cada artículo y el costo total de toda la venta. Ingresar dicha información en la base de datos, respetando todas las restricciones de integridad.

3. PLAN DE TRABAJO

3.1. Fechas Importantes

Actividad	Fecha
Entrega parcial del Proyecto	08/05/2020
Tercer Examen Parcial	12/05/2020
Fecha de Entrega Final	26/05/2020
Presentación del Proyecto	28-30/05/2020

Tabla 1: Fechas para las últimas actividades del curso en el semestre 2020-2.

3.2. Reuniones de Trabajo

Día	Horario
Lunes	19:00-21:00
Miercoles	15:00-17:00
Sábado	10:00-13:00

Tabla 2: Día y horario para las sesiones de trabajo acordadas por el equipo.

3.3. Roles de los integrantes del Equipo

Rol	Integrantes
Líder de Proyecto	Otero García Christian
Diseño web	Aguilera Ortiz Alfredo
Diseñador	Barrientos Veana Luis Mauricio
Analista	Vivanco Quintanar Diego Armando
Uso del repositorio Git	Barrientos Veana Luis Mauricio
Documentación del proyecto	Vivanco Quintanar Diego Armando
Implementación del diseño	Todos

Tabla 3: Roles de cada integrante del equipo.

3.4. Estimación de fechas para las actividades

Actividad	Fecha
Definición de Roles y Responsabilidades	28/04/2020
Diagramas MER/MR	29/04/2020
Script de creación de la base de datos y tablas	05/05/2020
Script para el agregado de información	05/05/2020
Script de toda la programación a nivel BD	08/05/2020
Códigos de lo implementado como parte de la etapa de presentación	10/05/2020
Documentación del proyecto	15/05/2020
La presentación a emplear a la hora de exponer	17/05/2020

Tabla 4: Fechas estimadas en las que se estiman que las actividades ya esten completadas.

3.5. Registro de reuniones y actividades

Reunión	Día	Fecha	Actividad	Horario
1	Martes	28/04/2020	Definición de Roles y Responsabilidades	21:00-22:00
2	Miércoles	29/04/2020	Análisis del Problema / Diagramas	15:00-18:00
3.1	Sábado	02/05/2020	Modelo Entidad-Relación	11:00-15:00
3.2	Sábado	02/05/2020	Modelo Relacional (DIA / ERWIN)	16:00-20:00
4	Martes	05/05/2020	Scripts DDL y DML	21:00-23:00
5	Miércoles	06/05/2020	Inserts sobre base de datos	12:00-15:00
6	Viernes	08/05/2020	Sesión con el profesor	20:00-21:30
7.1	Sábado	09/05/2020	Analisis de comentarios del profesor	13:00-15:00
7.2	Sábado	09/05/2020	Correcciones en el diseño del proyecto	16:00-18:00
8	Lunes	11/05/2020	Correcciones en DDL y DML	19:00-21:00
9.1	Viernes	15/05/2020	Programación a Nivel Base de Datos	16:00-18:00
9.2	Viernes	15/05/2020	Modificación a la página web	17:00-19:00
10	Domingo	17/05/2020	Conexión página web y Base de Datos	17:00-19:00
11.1	Lunes	18/05/2020	Programación Nivel Base de Datos	10:00-13:00
11.2	Lunes	18/05/2020	Conexión página web y Base de Datos	15:00-17:00
12	Martes	19/05/2020	Validar Stock	11:00-15:00
13	Miércoles	20/05/2020	Validar Stock (Continuación)	15:00-17:00
14	Jueves	21/05/2020	Triggers genericos	15:00-17:00
15	Viernes	22/05/2020	Seguimiento a conexión de la página	11:00-19:00
16	Sábado	23/05/2020	Desarrollo de Documentación	13:00-19:00
17	Domingo	24/05/2020	Pruebas de funcionamiento	13:00-19:00
18.1	Lunes	24/05/2020	Pruebas de funcionamiento	10:00-14:00
18.2	Lunes	24/05/2020	Detalles de la página web	16:00-12:00
19	Martes	24/05/2020	Finalización del proyecto	10:00-23:00

Tabla 5: Registro de las actividades realizadas por el equipo a lo largo de las sesiones tomadas.

4. MODELO ENTIDAD-RELACION

En el modelo entidad relación se obtuvieron 4 entidades PROVEEDOR con clave primaria rs_prov, PRODUCTO con clave primaria cod_barras, VENTA con clave primaria num_venta y CLIENTE con clave primaria rs_cliente. Se crearon 3 relaciones, primero la relación SUMINISTRA con cardinalidad "muchos a muchos" que almacena la fecha de compra de los productos y el precio al que fueron comprados el cual sirve para compararlo con el precio al que son vendidos los productos y obtener la utilidad, la segunda relación es PRODUCTO_VENTA de cardinalidad "muchos a muchos" que almacena la cantidad de productos que seleccione el cliente y el precio por producto de cada uno de los artículos que elija, estos dos atributos son necesarios para calcular la cantidad a pagar en la tabla VENTA. Por último se tiene la relación VENTA_CLIENTE (cardinalidad "muchos a uno"), la cual sirve para identificar qué clientes realizaron las ventas. En la figura 1 se puede observar el diagrama Entidad-Relación implementado en DIA.

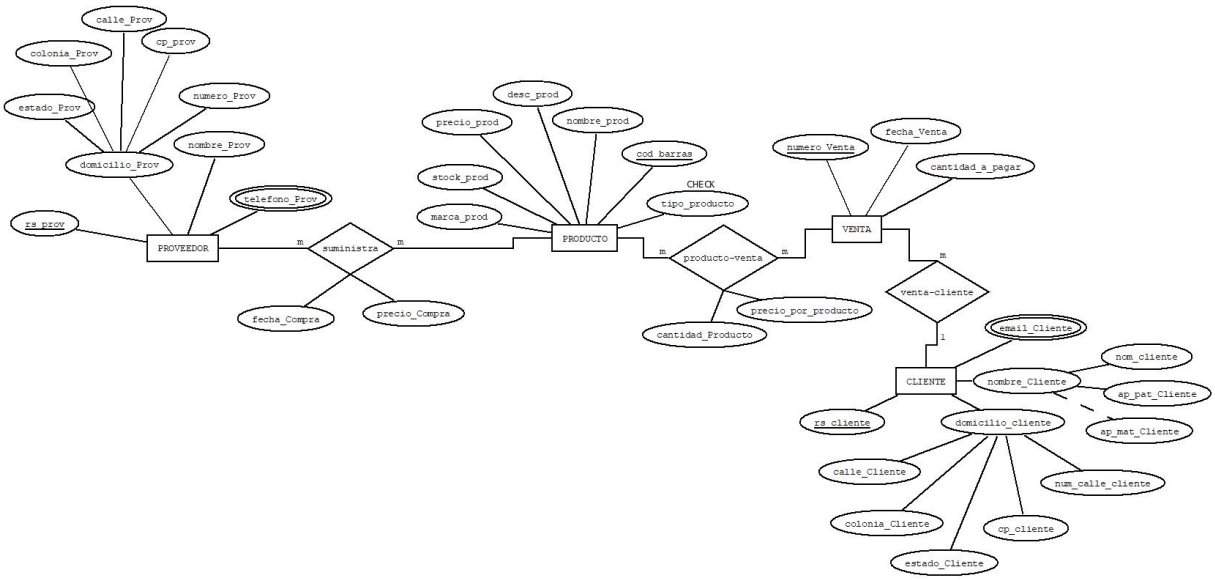

Figura 1: Modelo Entidad-Relación de la estructura de la base de datos del sistema.

5. MODELO RELACIONAL

Derivado del modelo entidad relación (figura 1) se mapeo al siguiente modelo relacional donde resaltan los siguientes aspectos, al tener los atributos multivaluados (telefono y email) del PROVEEDOR y del CLIENTE respectivamente se transformaron en dos nuevas relaciones email_cliente y teléfono_proveedor. Se tiene una restricción de dominio en la tabla PRODUCTO en el atributo tipo_producto donde solo pueden ser impresiones, recargas, regalos, artículos de papelería, esto con el fin de tener una clasificación para los productos en la misma tabla ya que de otra manera habríamos tenido que recurrir a una especialización en la que obteniamos más tablas que nos podían complicar el manejo y la integridad de los datos; todos estos valores fueron representados en el DDL por las palabras 'ip','rc','rg','ap'. En las relaciones con cardinalidad "muchos a muchos" (SUMINISTRA y PRODUCTO_VENTA) las llaves primarias de las entidades a las que hacen referencia pasan como llaves foráneas.

El mapeo al modelo relacional queda formado por las siguientes relaciones:

- **PROVEEDOR:** {rs_prov(pk), nom_prov, calle_prov, colonia_prov, estado_prov, código_postal_prov, num_prov}
- **TELEFONO_PROVEEDOR:** {telefono(pk), rs_prov(fk)}
- **SUMINISTRA:** {[rs_prov(fk), cod_barras(fk)](pk), fecha_compra, precio_compra}
- **PRODUCTO:** {cod_barras(pk), nom_prod, desc_prod, precio_prod, stock_prod, marca_prod, tipo_producto(ck)}
- **PRODUCTO_VENTA:** {[cod_barras(fk), num_venta(fk)](pk), cantidad_producto, precio_por_producto}
- **VENTA:** {num_venta(pk), fecha_venta, cantidad_a_pagar, rs_cliente(fk)}
- **CLIENTE:** {rs_cliente(pk), calle_cliente, colonia_cliente, estado_cliente, cp_cliente, num_calle_cliente, ap_pat_cliente, ap_mat_cliente(N)}
- **EMAIL_CLIENTE:** {email(pk), rs_cliente(fk)}

En la figura 2 se observa el modelo relacional de nuestra base de datos implementada en ERWIN.

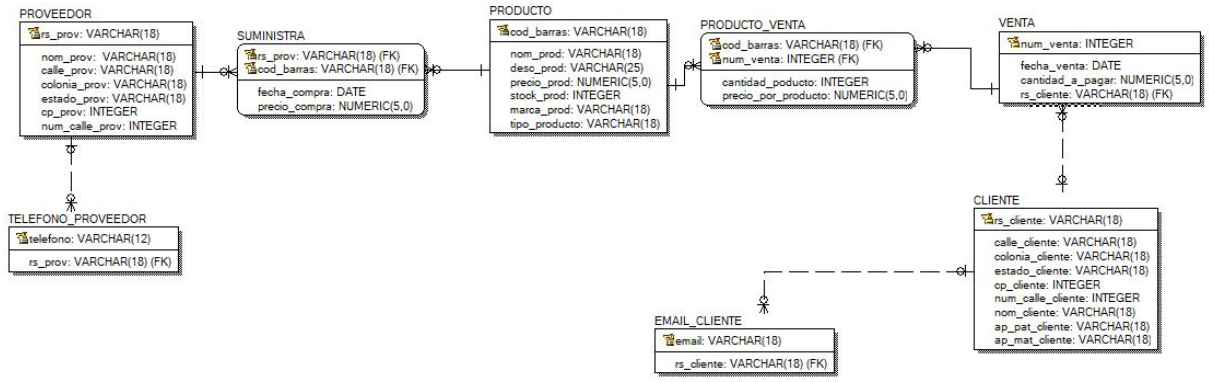

Figura 2: Modelo Relacional de la estructura de la base de datos del sistema.

6. NORMALIZACION

6.1. PROVEEDOR

A	B	C	D	E	F	G
rs_Prov	**nom_prov**	**calle_prov**	**colonia_prov**	**estado_prov**	**cp_prov**	**num_calle**

Tabla 6: Tabla sin normalizar de PROVEEDOR.

Podemos observar que la tabla 6 se encuentra en 1FN.La clave primaria es la razón social del proveedor (rs_prov), en este caso se presenta una dependencia parcial ya que no podemos acceder a los campos calle_proc y y num_calle solo con rs_prov, para eliminar esta dependencia parcial se establece que debe existir una clave compuesta por rs_prov y cp_prov, por otra parte creamos una tabla con clave primaria (PK) que es cp_prov con la cual accedemos a estado_prov y colonia_prov evitando asi una dependencia parcial.
Por lo tanto la 2FN queda de la siguiente manera:

- A,F $->$ {B,C,D,E,G} => 2FN
- F $->$

La tabla se encuentra en 2FN pero observamos que hay transitividad entre los atributos de la tabla. Para obtener la 3FN obtenemos las dependencias transitivas:

- A,F$->${C,G} $----$Calle_Num_Provedor => 3FN
- F$->${D,E} $----$Estado_Colonia_Proveedor
- A$->$B $----$Proveedor

Por lo tanto las tablas que obtenemos son las siguientes:

- **Calle_Num_Proveedor**

rs_Prov	cp_prov	calle_prov	num_calle

Tabla 7: Tabla para acceder a los campos calle y número de calle del domicilio del proveedor.

- **Estado_Colonia_Proveedor**

cp_prov	colonia_prov	estado_prov

Tabla 8: Tabla para acceder a los campos colonia y estado de del proveedor.

- **Proveedor**

rs_Prov	nom_prov

Tabla 9: Tabla para acceder al campo nombre del proveedor.

6.2. PRODUCTO

La tabla 10 de inicio esta en 1FN, no tiene dependencias parciales ni transitivas por lo que esta en 3FN.

cod_Barras	nom_Prod	desc_Prod	precio_Prod	stock_Prod	marca_Prod	tP

Tabla 10: Tabla para acceder a los campos del producto.

6.3. VENTA

num_venta	fecha_venta	cantidad_pagar

Tabla 11: Tabla para acceder a los campos del producto.

La tabla 11 de inicio esta en 1FN, pero observando podemos darnos cuenta que no hay dependencias parciales ni transitivas por lo tanto esta en 3FN.

6.4. CLIENTE

A	B	C	D	E	F	G	H	I
rs_c	nom_c	apP_c	apM_c	est_c	col_c	cp_c	calle_c	ncalle_c

Tabla 12: Tabla sin normalizar de CLIENTE.

La tabla 12 en principio se encuentra en 1FN, la clave primaria (PK) es la razón social del cliente que en nuestro contexto es la RFC. Una vez que identificamos que la tabla esta en 1FN observamos que se presenta una dependencia parcial, ya que si observamos para acceder al nombre de la calle y al número de la calle no lo podemos hacer unicamente con la razón social del cliente sino que se accede a esta información mediante una clave compuesta por rs_prov y cp_prov. Con lo anterior evitamos una dependencia parcial, por su parte con el código postal accedemos al estado y la colonia donde vive el cliente en donde claramente no hay dependencia parcial

Por lo tanto la 2FN queda de acuerdo con las siguiente dependencias funcionales:

- A,G$->$ B,C,D, H,I
- G$->$ E,F

Ahora que hemos obtenido la 2FN podemos ver que hay transitividad con la razón social del cliente ya que mediante esta clave podemos acceder a las columnas nombre, apellido paterno y materno del cliente. Por este motivo estos datos deben pasarse a otra tabla, dejando asi tablas libres de transitividad. Las dependencias quedan de la siguiente forma:

- A− > B,C,D —> CLIENTE
- G− > E,F ——->ESTADO_COLONIA_CLIENTE
- A,G − > H, I —>CALLE_NUM_CLIENTE

La 3FN queda organizada en 3 tablas, las cuales son las siguientes:

- **CALLE_NUM_CLIENTE**

rs_cliente	cp_cliente	calle_cliente	num_calle_cliente

Tabla 13: Tabla para acceder a los campos calle y número de calle del domicilio del cliente.

- **ESTADO_COLONIA_CLIENTE**

cp_cliente	estado_cliente	colonia_cliente

Tabla 14: Tabla para acceder a los campos colonia y estado de del cliente.

- **CLIENTE**

rs_cliente	nom_cliente	apPat_cliente	apMat_cliente

Tabla 15: Tabla para acceder al campo nombre, apellido paterno y apellido materno del cliente.

7. Implementación

Partiendo de las tablas que resultaron del proceso de la normalización en el punto anterior procederemos a crear las respectivas tablas, definiendo sus tipos de datos, las llaves primarias y foráneas asi como las funciones, triggers, vistas y lo que se necesite programar a nivel bases de datos para cumplir los requerimientos solicitados y de forma indirecta nos ayude a tener una mejor presentación y manejo del sistema.

7.1. DDL

Como ya se mencionó la primera parte de la implementación de la base de datos es la creación de las tablas. Por acuerdo del equipo y por comodidad al visualizar los cambios que se hagan en la base de datos optamos por trabajar sobre PGADMIN.

Query Editor

```
10  create database Papeleria;
11  \c Papeleria;
12
13  CREATE TABLE PROVEEDOR(
14  rs_prov  varchar(30) NOT NULL,
15  nom_prov varchar(30)  NOT NULL
16  );
17
18  CREATE TABLE ESTADO_COLONIA_PROVEEDOR(
19  cp_prov int NOT NULL,
20  estado_prov varchar(30) NOT NULL,
21  colonia_prov varchar(30) NOT NULL
22  );
```

Figura 3: Creación de la base de datos, la tabla PROVEEDOR y la tabla ESTADO_COLONIA_PROVEEDOR.

En la figura 3 se observa la estructura de las tablas PROVEEDOR que tiene por columnas una razón social del provedor y su nombre, siendo que la primera columna es la llave primaria de dicha tabla, ambas columnas son un varchar(30) y no pueden ser nulos los campos; por su parte ESTADO_COLONIA_PROVEEDOR tiene por columnas el código postal de tipo de tipo int, el estado y la colonia del domicilio de tipo varchar. Sin excepción todos los campos no pueden ser nulos. La llave primaria es el código postal.

En la figura 4 se observa la creación de las tablas CALLE_NUM_PROVEEDOR que lleva por columnas la razón social del proveedor, el código postal, calle y el numero de la calle. En donde el numero de la calle y el código postal son de tipo int, mientras que los demás son de tipo varchar.

```
24  CREATE TABLE CALLE_NUM_PROVEEDOR(
25  rs_prov varchar(30) NOT NULL,
26  cp_prov int NOT NULL,
27  calle_prov varchar(30) NOT NULL,
28  calle_num_prov int NOT NULL
29  );
30
31  CREATE TABLE TELEFONO_PROVEEDOR(
32  rs_prov varchar(30) NOT NULL,
33  telefono varchar(12) NOT NULL
34  );
```

Figura 4: Creación de la tabla CALLE_NUM_PROVEEDOR y la tabla TELEFONO_PROVEEDOR.

7.2. DML

7.3. Programación a nivel Bases de Datos

7.3.1. Funciones

- **Utilidad del producto**
  Función que devuelve la utilidad de un producto, recibe como parámetro el código de barras de un producto, devuelve una cadena que indica la utilidad del producto. Se declaran cuatro variables (prov, venta, resultado y cadena) de tipo numeric(5,0) para las tres primera y de tipo varchar para la última variable.. La variable "prov" va a almacenar el precio al que fue comprado el producto del proveedor mientras que la variable venta almacena el valor del precio al que se vende a los clientes. Resultado tendrá por valor la diferencia entre venta y prov, siendo esta la utilidad del producto, finalmente en cadena asignaremos un mensaje que indique al usuario la utilidad del producto, concatenando así el valor de "resultado" pero para ello castearemos la variable ya que es de tipo numeric y no es compatible concatenar con caracteres. Con el casteo el mensaje queda completo y se puede retornar el mensaje para ser visto por el usuario.

-

7.3.2. TRIGGERS

1. **Trigger para verificar stock**
   Se creó la función verifica_stock junto con el trigger tg_comprueba_stock que la dispara antes de que se inserte un registro en PRODUCTO_VENTA. Dentro de la función se creó la variable 'stock' que guarda el stock de los productos al momento de que se solicita un artículo y va comparando, primero si el stock es igual a cero no se permite

la venta lanzando el mensaje 'No hay productos en almacén', después se valida si hay la cantidad de productos que se solicitan es menor al stock disponible pero además hay menos de tres productos en stock, se permita realizar la compra y se mande el mensaje 'Contamos con menos de 3 productos en stock'. La siguiente validación es para cuando se compran todo el stock mostrando el mensaje 'Compraste los últimos productos en almacen'. Y por último cuando el stock es menor a la cantidad que solicita el cliente y además el stock es menor a tres se aborta la venta y se lanzan dos mensajes 'Contamos con menos de 3 productos en almacén'y 'No hay suficientes unidades en almacén para completar su compra';

2. **Actualiza monto total y fecha**
   Se creó la función que calcula el monto total y la fecha al momento de insertar en la tabla PRODUCTO_VENTA, se declararon las variables cantidad, precio, monto total y fecha, que almacenan la cantidad de productos cuando se ingresan en la tabla PRODUCTO_VENTA, además se almacena el precio por producto de la tabla PRODUCTO cuando se ingresa un nuevo código de barras y número de venta. Teniendo estos datos, para calcular el monto total se multiplicó el precio por la cantidad y se almacenó en la variable monto total. Por último para actualizar la tabla venta en el atributo cantidad a pagar se sumó la cantidad a pagar que tiene una venta más la variable monto_total. Para poner la fecha actual de la venta se utilizó now() en la variable fecha.

3. **Obtiene el precio por producto**
   Esta función obtiene el precio por producto que es disparado en la tabla PRODUCTO_VENTA cuando es insertado una nuevo pedido, se creó una variable que almacena el precio por producto de la tabla PRODUCTO cuando detecta que hay un nuevo código de barras solicitado, posteriormente se actualiza la tabla PRODUCTO_VENTA colocando el precio que le corresponde al producto seleccionado por el cliente.

8. Conclusiones

- **Aguilera Ortiz Alfredo**
  La elaboración de este proyecto fue de muchos aprendizajes , creo que fue una manera efectiva de lograr consolidar lo visto en clases de teoría y de laboratorio , al hacer un ejemplo más apegado a la vida real elaborando una página web y vinculando está a una base , esto es muy cercano a lo que nos podemos encontrar en la vida laboral , gracias a esto logramos observar como se elaboran estos procesos en un ambiente real y no solo en manejadores del laboratorio. Dentro de ese proyecto hubo muchas dificultades , empezando con el diseño de la base de datos la cual nos tomó más tiempo de lo esperado , la mayor parte del tiempo se ocupó en la programación de la base de datos así como triggers además de esto teníamos errores normales en la programación , sin embargo de una manera más personal tuve varias dificultades al vincular la base con nuestra nuestra página , ya que se tuvo que aprender algunos aspectos nuevos en html y de manera mas concreta de php que no había visto jamás. Creo que al hacer una base de datos desde el diseño , hasta la vinculación con nuestra página web se cierra el círculo de aprendizaje , tal vez no logramos observar de manera real al 100

- **BARRIENTOS VEANA LUIS MAURICIO**
  Si se analizan eficazmente los requerimientos para plantear una solución empezando por el modelo entidad relación las siguientes etapas se gestionan más fácilmente. En un principio al hacer el análisis se planteó una solución que incluía hacer una especialización, pero después de ir avanzando en el proyecto era evidente que esta propuesta complicaba las relaciones entre las tablas más usadas, por lo tanto, fue rediseñado el diagrama para dar una solución más adecuada al problema. En la programación se tuvieron dificultades con la sintaxis de postgres, hubo muchos puntos en los que se tenía la idea de cómo resolverlos, pero para ejecutarlos había que corregir la sintaxis y estructura del código para que funcionaran, por lo que se hizo mucha investigación para aprender la sintaxis y usarla correctamente. Un aspecto valioso que ayudó a desarrollar el trabajo consistentemente fue que desde un principio se dimensionó el alcance del proyecto, lo que nos sirvió para delegar responsabilidades e ir llevando un avance en todas las fases.

- **CHRISTIAN OTERO GARCIA**
  La elaboración de este proyecto representó un gran desafío para nosotros en distintos ámbitos, la más importante es que debido a las emergencias sanitarias no fue posible colaborar de manera presencial con mis compañeros y el proyecto se llevó a cabo completamente de manera virtual, esto trajo muchas ventajas y lamentablemente algunas desventajas.

  Como Gerente de Proyecto tuve muchos retos; desde un inicio dar un diagnóstico para la definición de objetivos, flujos de trabajo, etapas y fechas clave de este proyecto para lograr un plan de proyecto efectivo. Otro punto es la constante supervisión que había que tener con todos los aspectos del proyecto, adelantarse a posibles problemáticas como el tiempo y avance esperado respecto a fechas definidas. Y no menos importante la comunicación constante y la unificación del equipo.

  Aplicar todos los conocimientos y habilidades de las Bases de Datos a una problemática real fue un gran competencia, ya que hubo que retomar algunos de los temas más difíciles, como utilización de Triggers, Funciones y además entender un poco de HTML, y PHP.

  Como aciertos puedo mencionar la gran cohesión entre los miembros del equipo, ya que cada uno se desempeñó en lo que consideró tener mejores aptitudes y esto a su vez impulsó la cooperación, el avance constante y el cumplimiento de los objetivos.

- **VIVANCO QUINTANAR DIEGO ARMANDO**
  El objetivo planteado para la realización de este proyecto se cumplió ya que se logro desarrollar una base de datos que gestiona las ventas y los productos en almacén de una papelería, durante el proceso del proyecto fuimos utilizando todos los temas del curso de Bases de Datos, desde el diseño conceptual, pasando por el diseño lógico, la normalización de algunas tablas, su respectiva implementación en el manejador de postgres llevando consigo el desarrollo del DDL, DML. Así pues, también la parte de la programación a nivel base de datos en donde creamos triggers, funciones y vistas. Finalmente acoplamos la base de datos a una interfaz, que en este caso y por acuerdo del equipo fue una página web, ya que uno de los miembros ya tenia un poco de

experiencia en ese campo. Personalmente la parte más difícil del proyecto fue la parte de la programación a nivel base de datos y la conexión de la base con la página.

Al principio estaba algo desconcertado con los requerimientos del proyecto ya que no tenía experiencia con diseño web pero fue bastante interesante abordar esta parte ya que aprendí un poco sobre PHP y la manera de hacer una conexión con alguna base. En general me gusto trabajar con este equipo, me acople muy bien al trabajar con ellos y considero que aunque nos dividimos las responsabilidades de alguna manera todos nos involucramos en todo el proyecto desde el diseño, la implementación y las pruebas.

9. Referencias

- Lockhart, T. (1996). Guia del Programador de PostgreSQL. University of California, Berkeley CA: Postgres Global Development Group.
  $http://index-of.co.uk/SERVIDORES/Postgres-Programmer.pdf$
- The PostgreSQL Global Development Group. (1996). TRIGGERS PROCEDURES. Mayo 10, 2020, de The PostgreSQL Global Development Group. Sitio web:
  $https://www.postgresql.org/docs/9.4/pltcl-trigger.$
- Lockhart, T. (1996). Guia del Programador de PostgreSQL. University of California, Berkeley CA: Postgres Global Development Group.
  $http://index-of.co.uk/SERVIDORES/Postgres-Programmer.pdf$