Proyecto Final: Bases de Datos

Diego Armenta Tezcucano

Mayo 22 de 2020

Introduccion:

El siguiente proyecto tiene como objetivo la creación de una base de datos para una tienda de abarrotes. A partir de los requerimientos presentados por el cliente se realizo un análisis para poder crear la base que mejor funcionara para el problema en concreto. Se determinó que el objetivo principal era el de realizar ventas y tener registro de lo que éstas involucraban; como el inventario y sus productos; los clientes y proveedores.

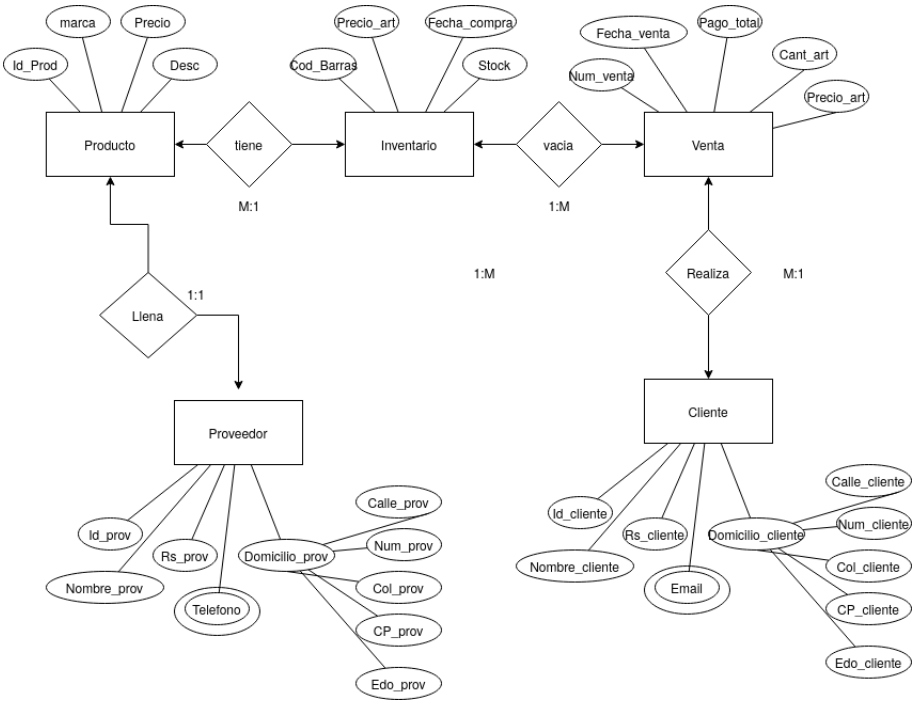
Plan de trabajo:

Se organizó el plan de trabajo en cuanto a la prioridad de los entregables. Los pasos tomados para realizar el proyecto fueron los siguientes:

1.- Creación de diagrama Entidad-Relación.
2.- Creación de Modelo Relacional.
3.- Creación de tablas y normalización.
4.- Programación de base.
5.- Creación de archivos CSV.
6.- Scripts para CSV.
7.- Programación de funciones y triggers de la base.
8.- Revisión del funcionamiento de las funciones.
9.- Diseño de la interfaz web.
10.- Relacionar interfaz con base de datos.
11.- Documentación.

Diseño:

1.- Diagrama Entidad-Relación:



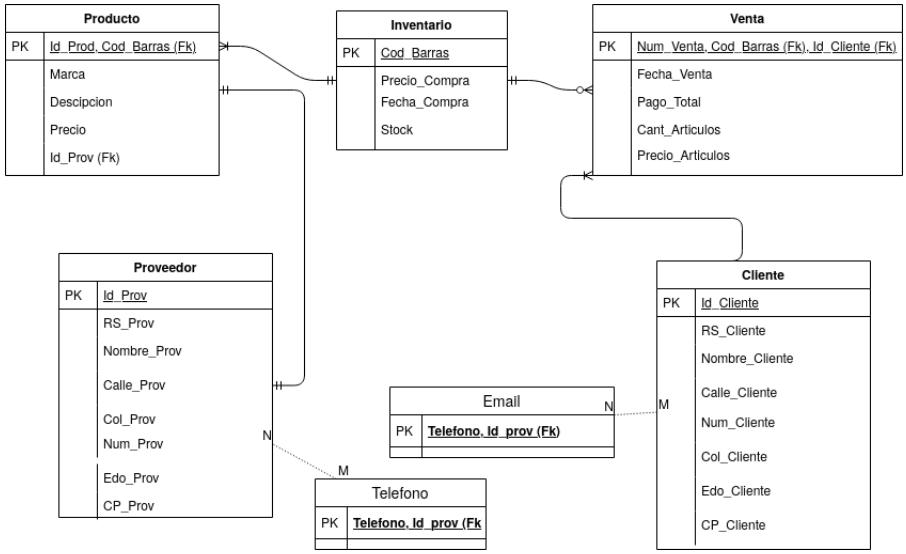
Una de las primeras propuestas que se tuvo del diagrama fue la de unir proveedor, producto y venta mediante inventario. Al principio esto parecía tener muchas ventajas ya que las entidades más importantes estarían unidas mediante el código de barras, además parecía tener sentido que un proveedor llenara el inventario, el inventario administrara los productos y las ventas. Sin embargo, esto hacia que la llave cod_barras se propagara hacia proveedor y en consecuencia hacia telefono ya que es un atributo multivaluado, por lo que se notó que esta configuración propagaba datos donde no se necesitaban.

La propuesta decisiva relaciona al proveedor con el producto, y el inventario conecta las ventas con el producto.

Se realizo el diagrama tomando en cuenta que si se relacionaba un inventario con muchos productos y muchas ventas se tendría la llave primaria del mismo (cod_barras) al alcance de los entidades más importantes, lo que permitiría relacionar facilmente producto con venta e inventario con cualquiera de las dos.

Por otro lado, la idea original entre la relación de producto y proveedor era que un proveedor pudiese surtir varios productos. Al análizar el problema se llego a la conclusión de que utilizando una relación uno a uno y pasando la clave del proveedor al producto generaba la misma funcionalidad, pero eliminaba de la llave primaria de producto el id_proveedor. Lo que soluciono conflictos a la hora de normalizar ya que no había necesidad de que los detalles del producto dependieran del id del proveedor.

2.- Modelo Relacional.



Producto							
Id_Prod	Cod_Barras (Fk)	Id_Prov (Fk)	Marca	Precio	Descripcion		
Inventario							
Cod_barras	Precio_Compra	Fecha_compra	Stock				
Venta							
Num_Venta	Cod_Barras (Fk)	Id_Cliente (Fk)	Fecha_Venta	Pago_Total	Cant_Articulos	Precio_Articulos	
Proveedor							
Id_Prov	RS_Prov	Nombre_Prov	Calle_Prov	Col_Prov	Edo_Prov	CP_Prov	Num_Prov
Cliente							
Id_Cliente	RS_Cliente	Nombre_Cliente	Calle_Cliente	Col_Cliente	Edo_Cliente	CP_Cliente	Num_Cliente
Telefono							
Telefono	Id_Prov (Fk)						
Email							
Email	Id_Cliente (Fk)						

3.- Normalización. Con las tablas siguientes se comenzó la normalización:

Producto							
Id_Prod	Cod_Barras (Fk)	Id_Prov (Fk)	Marca	Precio	Descripcion		
Inventario							
Cod_barras	Precio_Compra	Fecha_compra	Stock				
Venta							
Num_Venta	Cod_Barras (Fk)	Id_Cliente (Fk)	Fecha_Venta	Pago_Total	Cant_Articulos	Precio_Articulos	
Proveedor							
Id_Prov	RS_Prov	Nombre_Prov	Calle_Prov	Col_Prov	Edo_Prov	CP_Prov	Num_Prov
Cliente							
Id_Cliente	RS_Cliente	Nombre_Cliente	Calle_Cliente	Col_Cliente	Edo_Cliente	CP_Cliente	Num_Cliente
Telefono							
Telefono	Id_Prov (Fk)						
Email							
Email	Id_Cliente (Fk)						

Primera Forma Normal:

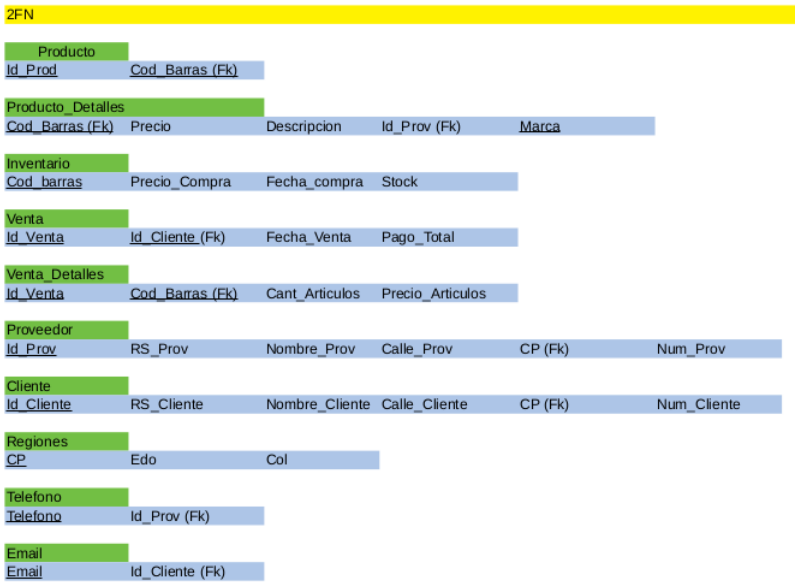
Se separarón los grupos de repetición de la tabla Proveedor, estos incluian el CP, el Num_Prov (numero de vivienda) y el Edo. Los atributos anteriores se pasaron a otra tabla, aquí al hacer lo mismo con la tabla cliente se notó que no había la necesidad de tener codigos postales separados para cliente y proveedor por lo que se utilizó una sola tabla para ambos.

También se separaron los grupos de repetición en la tabla Venta, que inlcuían el código de barras, la cantidad de articulos y el precio de los mismos. Esta información de pasó a otra tabla (venta_detalles) de manera que los datos generales de la venta estuviesen en una, y los detalles de la venta en otra.

1FN					
Producto					
Id_Prod	Cod_Barras (Fk)	Id_Prov (Fk)	Marca	Precio	Descripcion
Inventario					
Cod_barras	Precio_Compra	Fecha_compra	Stock		
Venta					
Id_Venta	Id_Cliente (Fk)	Fecha_Venta	Pago_Total		
Venta_Detalles					
Id_Venta	Cod_Barras (Fk)	Cant_Articulos	Precio_Articulos		
Proveedor					
Id_Prov	RS_Prov	Nombre_Prov	Calle_Prov	CP (Fk)	Num_Prov
Cliente					
Id_Cliente	RS_Cliente	Nombre_Cliente	Calle_Cliente	CP (Fk)	Num_Cliente
Regiones					
CP_	Edo	Col			
Telefono					
Telefono	Id_Prov (Fk)				
Email					
Email	Id_Cliente (Fk)				

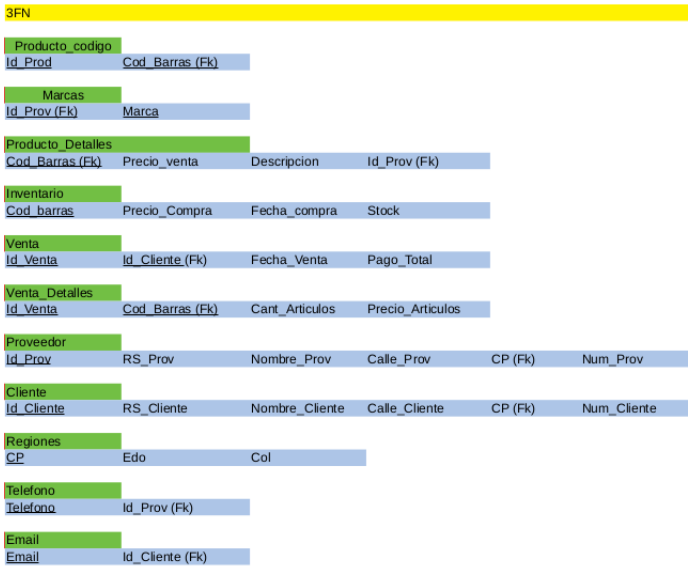
Segunda Forma Normal:

Para esta etapa de nomalización de identificó que la tabla producto contenía para todos los atributos dependencias parciales en Id_prod y Cod_Barras por lo que estos dos valores se pusieron en una tabla aparte y se dejo la tabla restante con Cod_Barras. La necesidad de un Id_prod en este momento podría parecer innecesaria. Pero muchas veces los codigos de barras de los productos no se encuentran, además de que son largos. Por lo que tener un Id_para el producto es necesario además de que originalmente es la llave primaria del producto.



Tercera Forma Normal:

Para la tercera forma normal se determino que en la tabla Producto_Detalles existía una relación transitiva del codigo de barras al id del proveedor, y de éste a la marca del producto, ya que si los proveedores son oficiales no venderan productos que no sean de su compañia, asi que se paso Id_prov y marca a otra tabla.



Implementación:

En la siguiente sección se listan unicamente las funciones, procedimientos y triggers que permiten realizar las actividades solicitadas en los procedimientos.

Requerimiento 1:

La siguiente función permite el cálculo de la utlidad a partir del precio de compra en el inventario y el precio de venta en el producto. El parámetro enviado es el codigo de barras del producto. En este caso los códigos de barra son muy simples para fines demostrativos.

```
create or replace function utilidad(cod integer) returns float AS $$
    declare compra float = 0;
    declare venta float = 0;
    declare utility float = 0;
    begin
      select e.precio_compra from inventario e where e.cod_barras = $1 into compra;
      select e.precio_venta from producto_detalles e where e.cod_barras = $1 into venta;
      utility = venta - compra;
      return utility;
    end;
$$ language plpgsql;
```

Requerimiento 2:

El primer punto del requerimiento que se trató fue el de darle numeración a las ventas de manera que el siguiente trigger forza que cada vez que se inserte una nueva venta se guarde como id, en el formato "Vent_000" de manera secuencial.

```
/*trigger para generar la numeracion secuencial del formato del nombre de la venta*/
create or replace function numeracion_venta() returns trigger as $$
    declare nombre varchar;
    declare numero int;
    begin
        select max(A::int) from (select substring(e.id_venta,6) as A from venta e) as B into numero;
        numero=numero+1;
        nombre =numero::varchar;
        new.id_venta = 'Vent-'||nombre;
        return new;
    end;
$$ language plpgsql;

create trigger numeracion_venta_trigger before insert on venta for each row execute procedure numeracion_venta();
```

De manera similar se realizó un trigger para tener los clientes de manera secuencial, ésto no se pidió explicitamente pero ayudó a la hora de realizar el agregado de la venta. Considerando que no todos los clientes que compraran iban a querer registrarse.

```
  /*trigger para generar la numeracion secuencial del id de cliente y agregar autmaticamente los nuevos codigos postales*/
create or replace function numeracion_cliente() returns trigger as $$
      declare numero int;
      begin
          select max(e.id_cliente) from cliente e into numero;
          numero=numero+1;
          new.id_cliente = numero;
          if not exists(select e.cp from regiones e where e.cp = new.cp)then
            insert into regiones (cp) values (new.cp);
          end if;
          return new;
      end;
  $$ language plpgsql;

create trigger numeracion_cliente_trigger before insert on cliente for each row execute procedure numeracion_cliente();
```

Lo siguiente fue crear una funcion para facilitar el agregado de una venta, se crearon dos funciones: una para crear una venta hecha por un cliente registrado y otra para clientes que no quisieran darse de alta. La segunda crea un cliente sin registro y lo asigna a la venta.

```
create or replace function crear_venta(cliente int) returns boolean as $$
    begin
          insert into venta (id_cliente, fecha_venta, pago_total) values ($1,current_date,0);
          return true;
    end;
$$ language plpgsql;

create or replace function crear_venta() returns boolean as $$
    declare max_cliente int;
    begin
          insert into cliente(cp) values (0);
          select max(id_cliente) from cliente into max_cliente;
          insert into venta (id_cliente,fecha_venta, pago_total) values (max_cliente,current_date,0);
          return true;
    end;
$$ language plpgsql;
```

La siguiente parte es un trigger que verifica que los productos cumplan los requisitos de stock, si no hay stock no permite la inserción y envia una alerta que diga que el stock es insuficiente. Tambien decrementa elstock al completar la venta.

Aqui se incluyeron dos funciones más para la generación de información, el calculo del subtotal por producto y el calculo del total de la venta.

```
/* trigger para chechar si hay stock, decrementar stock y actualizar subtotal y total de la compra*/

create or replace function check_stock() returns trigger as $$
    declare subtotal float=0;
    declare new_stock int=0;
    declare precio_anterior float=0;

    begin
      if (select e.stock from inventario e where e.cod_barras=new.cod_barras)<=3 then
        raise notice 'stock bajo';
      end if;
    if (select e.stock from inventario e where e.cod_barras=new.cod_barras)<=0 then
        return null;
    end if;
    if ((select e.stock from inventario e where e.cod_barras=new.cod_barras)-new.cant_articulos)<0 then
            return null;
    else

        select e.stock from inventario e where e.cod_barras=new.cod_barras into new_stock;
        select e.pago_total from venta e where e.id_venta=new.id_venta into precio_anterior;
        select e.precio_venta from producto_detalles e where e.cod_barras = new.cod_barras into subtotal;

        subtotal = subtotal *new.cant_articulos;
        new_stock = new_stock - new.cant_articulos;
        precio_anterior = precio_anterior + subtotal;

        update inventario set stock = new_stock where cod_barras = new.cod_barras;
        update venta set pago_total = precio_anterior where id_venta = new.id_venta;
        return new;
    end if;
    end;
$$ language plpgsql;

create trigger stock_price_trigger before insert on venta_detalles for each row execute procedure check_stock();
```

Por último se agrego una funcion para facilitar la inserción de un producto en la venta así como los datos de subtotal calculados anteriormente y la actualización del pago total de la venta a medida que entran productos.

```
create or replace function agregar_prod(registro varchar, codigo int, cantidad int) returns boolean as $$
    declare subtotal float=0;
    begin
        select e.precio_venta from producto_detalles e where e.cod_barras = codigo into subtotal;
        subtotal = subtotal *$3;
        insert into venta_detalles (id_venta, cod_barras, cant_articulos, precio_articulos) values ($1,$2,$3,subtotal);
        return true;
    end;
$$ language plpgsql;
```

Requerimiento 3:

Para este requerimiento se crearon dos funciones que regresen la cantidad de productos vendidos por producto, una lo realiza recibiendo como parámetro una fecha específica, la otra devuelve los resultados en un periodo delimitado por dos fechas.

```
create or replace function periodo_venta(varchar, varchar) returns TABLE (cod_barras int, cantidad_articulos bigint)as $$
    declare ini date;
    declare fin date;
    begin
      ini=date($1);
      fin=date($2);
      return query Select c.cod_barras, sum(c.cant_articulos) from (Select * from venta_detalles as B join (select e.id_venta from venta e where e.fecha_venta > ini
    end;
$$ language plpgsql;

create or replace function periodo_venta(varchar) returns TABLE (cod_barras int, cantidad_articulos bigint)as $$
    declare ini date;
    begin
      ini=date($1);
      return query Select c.cod_barras, sum(c.cant_articulos) from (Select * from venta_detalles as B join (select e.id_venta from venta e where e.fecha_venta = ini)
    end;
$$ language plpgsql;
```

Requerimientos 4:

Para el requerimiento 4 se creo la funcion productos_escasos() que permite obtener sin parámetros todos los productos que tengan menos de 3 en stock.

Requerimiento 5:

Se realizó una función que recolecte toda la informacion de las diferentes tablas (inventario, venta y productos) que pueda ser utilizada en una factura. La funcion se llama info_factura().

Requerimiento 6:

Se implemento un índice en el codigo de barras considerando que la tabla de productos es la más grande y que el atributo es compartido en multiples tablas.

```sql
/*requerimiento 4 obtener productos de los que haya menos de 3*/

create or replace function productos_escasos() returns TABLE (cod_barras int, cantidad_articulos int)as $$
    begin
      return query Select c.cod_barras, c.stock from inventario c where c.stock < 3;
    end;
$$ language plpgsql;

/*requerimiento 5 generar la informacion para la factura, Se completa este requerimiento con la interfaz web*/
create or replace function info_factura(vent varchar) returns table (id_cliente int, razon_social varchar, nombre varchar, cp int, venta varchar, fecha date,
  codigo_producto int, producto varchar, precio_individual float, cantidad int, subtotal float, total float) as $$

    begin
      return query select H.id_cliente, H.rs_cliente as razon_social, H.nombre_cliente as nombre, H.cp, H.venta, H.fecha_venta as fecha, H.cod_barras as
      codigo_producto, H.descripcion as producto, H.precio_venta as precio_individual, H.cant_articulos as cantidad, H.precio_articulos as subtotal,
       H.pago_total as total from (select * from cliente D join(select A.id_venta as venta, A.id_cliente as id_cliente2, A.fecha_venta, A.pago_total,
         B.* from venta A join (select * from  venta_detalles E join(Select precio_venta, cod_barras as cod_barras2, descripcion from producto_detalles) as F
         on E.cod_barras = F.cod_barras2) as B on A.id_venta=B.id_venta where A.id_venta = $1) as C on D.id_cliente = C.id_cliente2) H;
    end;
$$ language plpgsql;

/*requerimiento 6 indice , se eligio considerando que cod_barras es uno de los atributos más utilizados y compartidos entre otras tablas, ademas es de los que más

create index cod_index on inventario (cod_barras);
```

Pruebas:

Requisito 1.-

La funcion utlidad() regresa la ganancia de un producto dado su codigo de barras. En este caso los codigos de barras son simples para fines de ejemplificación.

```
armenta=# select utilidad (4);
 utilidad
----------
     14.8
(1 row)
```

Requisito 2.-

Con las funciones descritas para el requerimiento 2 en la etapa de implementación, se realizó la interfaz web que hace uso de las mismas para la inserción de una venta.

Este requerimiento se complementa con la interfaz web.

Ejemplo desde sql:

Se crea una venta nueva:

```
armenta=# select * from venta;
 id_venta | id_cliente | fecha_venta | pago_total
----------+------------+-------------+------------
 Vent-1   |          3 | 2020-02-02  |        838
 Vent-2   |          4 | 2020-02-22  |      176.2
 Vent-3   |          6 | 2020-03-12  |       88.1
 Vent-4   |          7 | 2021-04-17  |       1254
(4 rows)
```

```
armenta=# select crear_venta(4);
 crear_venta
-------------
 t
(1 row)
```

En los detalles siguientes se muestra la venta y como no hay ningun producto registrado en ella.

```
armenta=# select * from venta detalles;
 id_venta | id_cliente | fecha_venta | pago_total
----------+------------+-------------+------------
 Vent-1   |          3 | 2020-02-02  |        838
 Vent-2   |          4 | 2020-02-22  |      176.2
 Vent-3   |          6 | 2020-03-12  |       88.1
 Vent-4   |          7 | 2021-04-17  |       1254
 Vent-5   |          4 | 2020-05-21  |          0
(5 rows)

armenta=# select * from venta_detalles;
 id_venta | cod_barras | cant_articulos | precio_articulos
----------+------------+----------------+------------------
 Vent-1   |         14 |              4 |            352.4
 Vent-1   |         23 |              4 |            485.6
 Vent-1   |         22 |              1 |            117.7
 Vent-2   |          6 |              1 |             58.5
 Vent-2   |          5 |              2 |            109.6
 Vent-3   |         14 |              1 |             88.1
 Vent-4   |         23 |              4 |            485.6
 Vent-4   |         20 |              2 |            220.6
 Vent-4   |         21 |              4 |              456
 Vent-4   |         15 |              1 |             91.8
(10 rows)
```

Se llama el registro del inventario antes de agregar el producto, fije su atencion en el producto con codigo de barras 23.

```
armenta=# select * from inventario;
 cod_barras | precio_compra | fecha_compra | stock
------------+---------------+--------------+-------
          1 |          32.1 | 2016-11-28   |    27
          2 |          33.5 | 2017-11-20   |    32
          3 |          34.9 | 2019-10-17   |     8
          4 |          36.3 | 2019-10-20   |    15
          5 |          37.7 | 2018-10-12   |    23
          6 |          39.1 | 2018-10-24   |    17
          7 |          40.5 | 2016-10-12   |    15
          8 |          41.9 | 2018-10-29   |    38
          9 |          43.3 | 2016-11-15   |    25
         10 |          44.7 | 2018-12-12   |    47
         11 |          46.1 | 2017-11-16   |    39
         12 |          47.5 | 2016-10-17   |     6
         13 |          48.9 | 2019-10-20   |    12
         14 |          50.3 | 2016-10-10   |    24
         15 |          51.7 | 2019-12-24   |    35
         16 |          53.1 | 2019-10-11   |    14
         17 |          54.5 | 2017-12-19   |    12
         18 |          55.9 | 2018-12-25   |    32
         19 |          57.3 | 2016-12-14   |    28
         20 |          58.7 | 2019-12-28   |    41
         21 |          60.1 | 2018-11-30   |    20
         22 |          61.5 | 2019-10-22   |    19
         23 |          62.9 | 2017-12-28   |     4
         24 |          64.3 | 2017-12-20   |    34
(24 rows)
```

Se agrega el producto.

```
armenta=# select agregar_prod('Vent-5', 23,4);
 agregar_prod
--------------
 t
(1 row)
```

Ahora el trigger se llama automaticmante y el stock de ese producto baja.

```
armenta=# select * from inventario;
 cod_barras | precio_compra | fecha_compra | stock
------------+---------------+--------------+-------
          1 |          32.1 | 2016-11-28   |    27
          2 |          33.5 | 2017-11-20   |    32
          3 |          34.9 | 2019-10-17   |     8
          4 |          36.3 | 2019-10-20   |    15
          5 |          37.7 | 2018-10-12   |    23
          6 |          39.1 | 2018-10-24   |    17
          7 |          40.5 | 2016-10-12   |    15
          8 |          41.9 | 2018-10-29   |    38
          9 |          43.3 | 2016-11-15   |    25
         10 |          44.7 | 2018-12-12   |    47
         11 |          46.1 | 2017-11-16   |    39
         12 |          47.5 | 2016-10-17   |     6
         13 |          48.9 | 2019-10-20   |    12
         14 |          50.3 | 2016-10-10   |    24
         15 |          51.7 | 2019-12-24   |    35
         16 |          53.1 | 2019-10-11   |    14
         17 |          54.5 | 2017-12-19   |    12
         18 |          55.9 | 2018-12-25   |    32
         19 |          57.3 | 2016-12-14   |    28
         20 |          58.7 | 2019-12-28   |    41
         21 |          60.1 | 2018-11-30   |    20
         22 |          61.5 | 2019-10-22   |    19
         24 |          64.3 | 2017-12-20   |    34
         23 |          62.9 | 2017-12-28   |     0
(24 rows)
```

Si se intenta agregar la cantidad de un producto que supere el stock disponible, esta no se agregara a la venta y enviara una alerta. Preste atencion al producto 3

```
armenta=# select agregar_prod('Vent-5', 3,10);
 agregar_prod
--------------
 t
(1 row)
```

Efectivamente el stock permanece sin cambio.

```
armenta=# select * from inventario
armenta-# ;
 cod_barras | precio_compra | fecha_compra | stock
------------+---------------+--------------+-------
          1 |          32.1 | 2016-11-28   |    27
          2 |          33.5 | 2017-11-20   |    32
          3 |          34.9 | 2019-10-17   |     8
          4 |          36.3 | 2019-10-20   |    15
          5 |          37.7 | 2018-10-12   |    23
          6 |          39.1 | 2018-10-24   |    17
          7 |          40.5 | 2016-10-12   |    15
          8 |          41.9 | 2018-10-29   |    38
          9 |          43.3 | 2016-11-15   |    25
         10 |          44.7 | 2018-12-12   |    47
         11 |          46.1 | 2017-11-16   |    39
         12 |          47.5 | 2016-10-17   |     6
         13 |          48.9 | 2019-10-20   |    12
         14 |          50.3 | 2016-10-10   |    24
         15 |          51.7 | 2019-12-24   |    35
         16 |          53.1 | 2019-10-11   |    14
         17 |          54.5 | 2017-12-19   |    12
         18 |          55.9 | 2018-12-25   |    32
         19 |          57.3 | 2016-12-14   |    28
         20 |          58.7 | 2019-12-28   |    41
         21 |          60.1 | 2018-11-30   |    20
         22 |          61.5 | 2019-10-22   |    19
         24 |          64.3 | 2017-12-20   |    34
         23 |          62.9 | 2017-12-28   |     0
(24 rows)
```

Los detalles de la venta (Vent_5) muestran como sólo se agrego el producto en existencia a la misma.

```
armenta=# select * from venta_detalles;
 id_venta | cod_barras | cant_articulos | precio_articulos
----------+------------+----------------+------------------
 Vent-1   |         14 |              4 |            352.4
 Vent-1   |         23 |              4 |            485.6
 Vent-1   |         22 |              1 |            117.7
 Vent-2   |          6 |              1 |             58.5
 Vent-2   |          5 |              2 |            109.6
 Vent-3   |         14 |              1 |             88.1
 Vent-4   |         23 |              4 |            485.6
 Vent-4   |         20 |              2 |            220.6
 Vent-4   |         21 |              4 |              456
 Vent-4   |         15 |              1 |             91.8
 Vent-5   |         23 |              4 |            485.6
(11 rows)
```

Requerimiento 3:

Se muestran los productos vendidos en una fecha y en un periodo entre fechas.

```
armenta=# select * from venta;
 id_venta | id_cliente | fecha_venta | pago_total
----------+------------+-------------+------------
 Vent-1   |          3 | 2020-02-02  |        838
 Vent-2   |          4 | 2020-02-22  |      176.2
 Vent-3   |          6 | 2020-03-12  |       88.1
 Vent-4   |          7 | 2021-04-17  |       1254
 Vent-5   |          4 | 2020-05-21  |      808.4
(5 rows)

armenta=# select * from periodo_venta('2020-02-02');
 cod_barras | cantidad_articulos
------------+--------------------
         14 |                  4
         22 |                  1
         23 |                  4
(3 rows)

armenta=# select * from periodo_venta('2016-11-12','2021-02-02');
 cod_barras | cantidad_articulos
------------+--------------------
          5 |                  2
          6 |                  1
         12 |                  4
         14 |                  5
         22 |                  1
         23 |                  8
(6 rows)
```

Requerimiento 4:

Se muestran los productos donde hay stock bajo (menor a 3)

```
armenta=# select * from inventario;
 cod_barras | precio_compra | fecha_compra | stock
------------+---------------+--------------+-------
          1 |          32.1 | 2016-11-28   |    27
          2 |          33.5 | 2017-11-20   |    32
          3 |          34.9 | 2019-10-17   |     8
          4 |          36.3 | 2019-10-20   |    15
          5 |          37.7 | 2018-10-12   |    23
          6 |          39.1 | 2018-10-24   |    17
          7 |          40.5 | 2016-10-12   |    15
          8 |          41.9 | 2018-10-29   |    38
          9 |          43.3 | 2016-11-15   |    25
         10 |          44.7 | 2018-12-12   |    47
         11 |          46.1 | 2017-11-16   |    39
         13 |          48.9 | 2019-10-20   |    12
         14 |          50.3 | 2016-10-10   |    24
         15 |          51.7 | 2019-12-24   |    35
         16 |          53.1 | 2019-10-11   |    14
         17 |          54.5 | 2017-12-19   |    12
         18 |          55.9 | 2018-12-25   |    32
         19 |          57.3 | 2016-12-14   |    28
         20 |          58.7 | 2019-12-28   |    41
         21 |          60.1 | 2018-11-30   |    20
         22 |          61.5 | 2019-10-22   |    19
         24 |          64.3 | 2017-12-20   |    34
         23 |          62.9 | 2017-12-28   |     0
         12 |          47.5 | 2016-10-17   |     2
(24 rows)
```

```
armenta=# select * from productos_escasos();
 cod_barras | cantidad_articulos
------------+--------------------
         23 |                  0
         12 |                  2
(2 rows)
```

Requerimiento 5:

Se muestran la informacion necesaria para una factura, este requerimiento se complementa con la interfaz web.

```
armenta=# select * from info_factura('Vent-1');
armenta=#
```

```
 id_cliente | razon_social | nombre |  cp   | venta |   fecha    | codigo_producto |    producto    | precio_individual | cantidad | subtotal | total
------------+--------------+--------+-------+-------+------------+-----------------+----------------+-------------------+----------+----------+------
-
          3 | ninguna      | mario  | 88708 | Vent-1 | 2020-02-02 |              14 | 19 descripcion |              88.1 |        4 |    352.4 |   838
          3 | ninguna      | mario  | 88708 | Vent-1 | 2020-02-02 |              22 | 30 descripcion |             117.7 |        1 |    117.7 |   838
          3 | ninguna      | mario  | 88708 | Vent-1 | 2020-02-02 |              23 | 28 descripcion |             121.4 |        4 |    485.6 |   838
(3 rows)
```

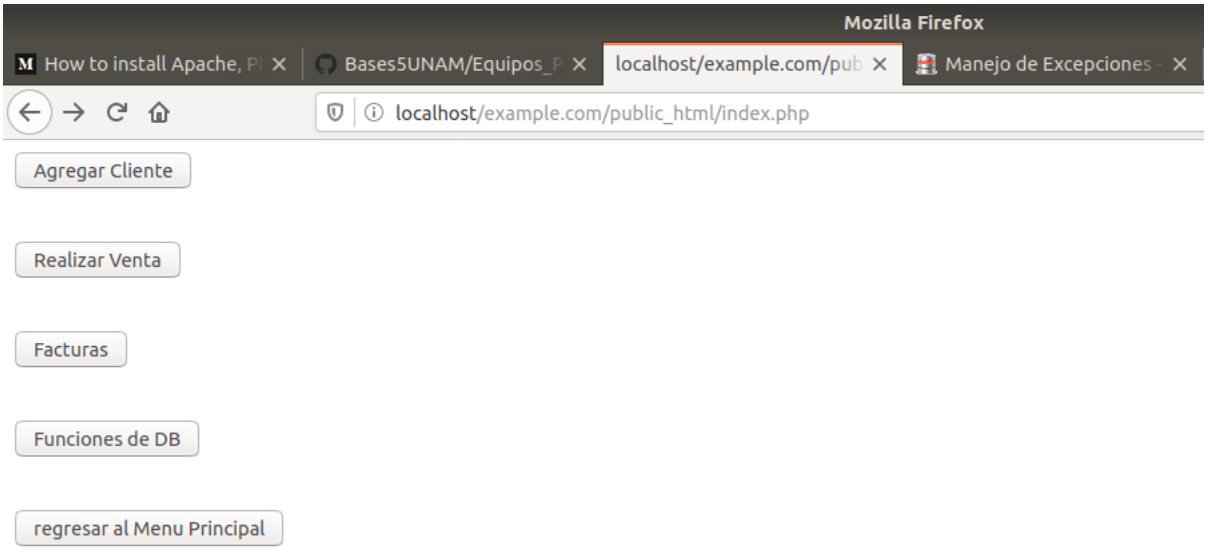
Pruebas en interfaz web:

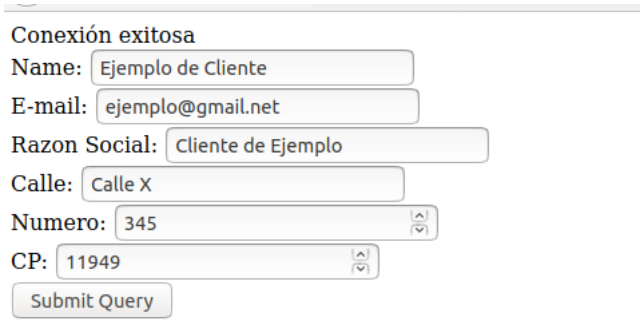
Pruebas para la insercion de cliente:
Se verifican los clientes antes de comenzar el proceso:

```
armenta=# delete from cliente where cp =0;
DELETE 3
armenta=# select * from cliente;
 id_cliente |      rs_cliente      | nombre_cliente | calle_cliente | num_cliente |  cp
------------+----------------------+----------------+---------------+-------------+-------
          1 | empleado             | luisa          | Calle 2       |         320 | 11949
          2 | empresa x            | sancho         | Calle 3       |         165 | 18576
          3 | ninguna              | mario          | Calle 13      |          57 | 88708
          4 |                      | samir          | Avenida 8     |          60 | 29970
          5 | asalariado           | jesus          | Cerrada 5     |         304 | 75130
          6 | cliente              | jaime          | bardas        |         336 | 58157
          7 | razon social ejemplo | jaime          | arboles       |         351 | 41522
          8 | asalariado           | fernanda       | lluvia        |         104 | 42432
          9 | cliente              | fernando       | nube          |         221 | 15337
         10 | razon social ejemplo | alejandro      | fuego         |         352 | 96616
(10 rows)

armenta=#
```

Se realiza el proceso de agregar cliente.





Se verifica en la base que se haya agregado el cliente:

```
armenta=# select * from cliente;
 id_cliente |      rs_cliente      |   nombre_cliente   | calle_cliente | num_cliente |  cp
------------+----------------------+--------------------+---------------+-------------+-------
          1 | empleado             | luisa              | Calle 2       |         320 | 11949
          2 | empresa x            | sancho             | Calle 3       |         165 | 18576
          3 | ninguna              | mario              | Calle 13      |          57 | 88708
          4 |                      | samir              | Avenida 8     |          60 | 29970
          5 | asalariado           | jesus              | Cerrada 5     |         304 | 75130
          6 | cliente              | jaime              | bardas        |         336 | 58157
          7 | razon social ejemplo | jaime              | arboles       |         351 | 41522
          8 | asalariado           | fernanda           | lluvia        |         104 | 42432
          9 | cliente              | fernando           | nube          |         221 | 15337
         10 | razon social ejemplo | alejandro          | fuego         |         352 | 96616
         11 | Cliente de Ejemplo   | Ejemplo de Cliente | Calle X       |         345 | 11949
(11 rows)

armenta=#
```

Prueba para la venta y factura:
Se verifican los datos de ventas antes de iniciar el proceso:

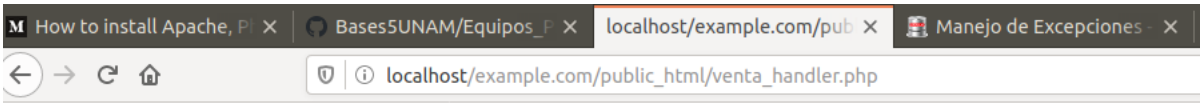
```
 id_venta | id_cliente | fecha_venta | pago_total
----------+------------+-------------+------------
 Vent-1   |          3 | 2020-02-02  |        838
 Vent-2   |          4 | 2020-02-22  |      176.2
 Vent-3   |          6 | 2020-03-12  |       88.1
 Vent-4   |          7 | 2021-04-17  |       1254
 Vent-20  |          5 | 2020-05-22  |          0
 Vent-21  |          5 | 2020-05-22  |          0
 Vent-5   |          4 | 2020-05-21  |      808.4
 Vent-22  |            | 2020-05-22  |      517.5
 Vent-23  |          4 | 2020-05-22  |      517.5
 Vent-6   |          5 | 2020-05-22  |      164.4
 Vent-7   |          5 | 2020-05-22  |      164.4
 Vent-8   |          5 | 2020-05-22  |      164.4
 Vent-24  |          4 | 2020-05-22  |      517.5
 Vent-10  |          5 | 2020-05-22  |      164.4
 Vent-11  |          5 | 2020-05-22  |      266.6
```

```
 Vent-5   |          4 | 2020-05-21  |      808.4
 Vent-22  |            | 2020-05-22  |      517.5
 Vent-23  |          4 | 2020-05-22  |      517.5
 Vent-6   |          5 | 2020-05-22  |      164.4
 Vent-7   |          5 | 2020-05-22  |      164.4
 Vent-8   |          5 | 2020-05-22  |      164.4
 Vent-24  |          4 | 2020-05-22  |      517.5
 Vent-10  |          5 | 2020-05-22  |      164.4
 Vent-11  |          5 | 2020-05-22  |      266.6
 Vent-12  |          5 | 2020-05-22  |      266.6
 Vent-13  |          5 | 2020-05-22  |      266.6
 Vent-16  |          5 | 2020-05-22  |      102.2
 Vent-17  |          5 | 2020-05-22  |      102.2
 Vent-18  |          5 | 2020-05-22  |      102.2
 Vent-19  |          5 | 2020-05-22  |      102.2
(21 rows)
```

Se realiza el agregado desde la interfaz.

La interfaz nos muestra que se completó la venta y muestra los datos de la factura haciendo uso de las funciones implementadas en la base.

No se agregan los productos para los que no haya stock suficiente, en la parte hay un aviso de stock insuficiente para un producto, pero si se agregan los productos que existen.



Conexión exitosa 11105'Vent-25'105
El articulo 14 tiene stock insuficiente

Datos de FACTURA:

Numero de cliente: 11

Nombre: Ejemplo de Cliente

razon social: Cliente de Ejemplo

CP: 11949

Numero de venta: Vent-25

Fecha: 2020-05-22

----------Total de la compra: 905.5

----------Desgloce:

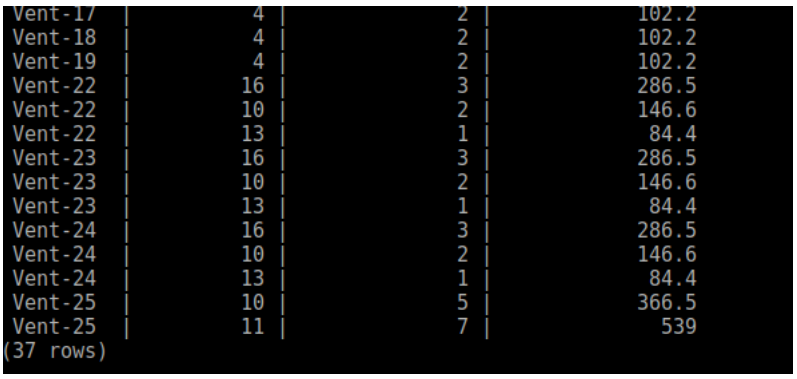
----------Articulo: 10 Descripcion: 7 descripcion Cantidad: 73.3 Subtotal: 5

----------Articulo: 11 Descripcion: 16 descripcion Cantidad: 77 Subtotal: 7

----------Articulo: Descripcion: Cantidad: Subtotal:

Se muestra en la base que la venta se agregó, asi como los detalles de cada producto en la misma.

```
 Vent-5   |           4 | 2020-05-21 |      808.4
 Vent-22  |             | 2020-05-22 |      517.5
 Vent-23  |           4 | 2020-05-22 |      517.5
 Vent-6   |           5 | 2020-05-22 |      164.4
 Vent-7   |           5 | 2020-05-22 |      164.4
 Vent-8   |           5 | 2020-05-22 |      164.4
 Vent-24  |           4 | 2020-05-22 |      517.5
 Vent-10  |           5 | 2020-05-22 |      164.4
 Vent-11  |           5 | 2020-05-22 |      266.6
 Vent-12  |           5 | 2020-05-22 |      266.6
 Vent-13  |           5 | 2020-05-22 |      266.6
 Vent-16  |           5 | 2020-05-22 |      102.2
 Vent-17  |           5 | 2020-05-22 |      102.2
 Vent-18  |           5 | 2020-05-22 |      102.2
 Vent-19  |           5 | 2020-05-22 |      102.2
 Vent-25  |          11 | 2020-05-22 |      905.5
(22 rows)
```



Vent-17	4	2	102.2
Vent-18	4	2	102.2
Vent-19	4	2	102.2
Vent-22	16	3	286.5
Vent-22	10	2	146.6
Vent-22	13	1	84.4
Vent-23	16	3	286.5
Vent-23	10	2	146.6
Vent-23	13	1	84.4
Vent-24	16	3	286.5
Vent-24	10	2	146.6
Vent-24	13	1	84.4
Vent-25	10	5	366.5
Vent-25	11	7	539

(37 rows)

Conclusiones:

Este proyecto requirio una cantidad fuerte de esfuerzo y dedicación, cada seccion del mismo no fué trivial y realizarla exigía prestar atención a lo que se estaba haciendo, así como pensar cual era la mejor solución al problema.

La etapa de diseño me permitió ver lo importante que es el estructurar bien la base desde el inicio, ya que si no se hace de forma correcta a lo largo de las demás fases irán surgiendo problemas que será dificil arreglar y requerirá soluciones menos elegantes. Al inicio del documento se menciono que existian diferentes propuestas de resolución, al intentar trabajar con estas propuestas se generaron el tipo de problemas mencionados anteriormente, por lo que la propuesta seleccionada se hizo en base no solo al análisis de la misma sino también al verificar que implementandola se trabajaría en las demás fases de una manera más fluida.

La etapa de nomalización de igual manera represento un ejercicio de análisis para determinar el propósito del proceso. De igual manera que en la fase anterior, el intentar utilizar propuestas descartadas para la nomalización generaba tablas que poco ayudaban al entendimiento de los datos y a la resolución del problema. Las tablas generadas a partir de las propuestas descartadas relacionaban datos en sitios que no los necesitaban o donde incluso eran más un impedimento para manejar la información.

La etapa de implementación fue la que personalmente más disfruté pues me permitio hacer uso de los conocimientos clásicos de programación para gestionar características de la base, en esta fase fue que me dí cuenta de cuál poderosas son las bases de datos y del porque la importancia de la programación dentro de la base. La programación PL/SQL permite crear límites

dentro de la base en cuanto a seguridad, pero también permite extenderla a manera que la gestión de datos sea mucho más sencilla e incluso procese información acerca de quien maneja la base.

En general creo que fue un ejercicio muy interesante que permitio poner a prueba lo aprendido a lo largo del curso. Fue una oportuinidad de adentrarme en un campo que conocía de una manera muy superficial y que ahora a generado gran interes en mi.