Universidad Nacional Autónoma de México

Facultad de Ciencias

Proyecto Final

Alumnos:
Karla Adriana Esquivel Guzmán
José Nieves Morán Silva
Sebastián Ávila Valdovinos

Profesores:
Gerardo Avilés Rosas
José Enrique Vargas Benítez
Gerardo Uriel Soto Miranda
Carlos Augusto Escalona Navarro

17 Diciembre 2018

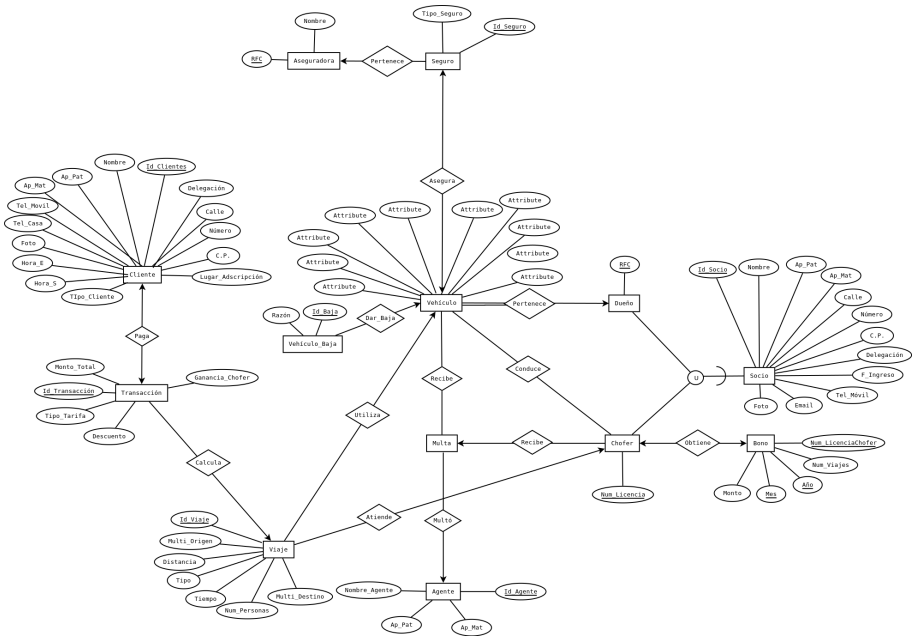


Figura 1: Modelo Entidad Relación

1. Modelo Entidad Relación

El modelo Entidad relación se apega totalmente a las especificaciones del PDF, desde un principio se intentó pensar de manera que cumpliera con la **3FN** y la **Forma Normal de Boyce codd**, así que lo hicimos con mucho cuidado.

2. Modelo Relacional

El modelo relacional ya se encuentra normalizado, por el diseño que hicimos desde un principio, las especificaciones de la normalización, están en la sección 3 del PDF. Las tablas que tenemos en el diagrama son:

- Socio: Esta tabla contiene los datos que comparten Choferes y Dueños.
- Multa: Contiene los datos de la multa, dirección(C.P., Delegación, Calle, Entre calles), Infracción, Monto y Fecha.
- Viaje: Contiene todos los datos del viaje incluido los identificadores del Chofer y el Vehículo, que participan en el viaje.
- Vehículo: Contiene las especificaciones del Vehículo incluido el identificador del dueño de este.
- Cliente: Contiene los datos del cliente.
- Transacción: Contiene los datos de cobro del viaje y ganancias del chofer.
- Vehículo_VehículoBaja: Contiene los datos de los vehículos dados de baja y la razón por la cual se dieron de baja.
- Dueño: Cuenta con el RFC de los dueños de los vehículos.
- Seguro: Contiene los datos del Seguro.
- Aseguradora: Contiene unicamente el RFC y Nombre de la aseguradora.
- Conduce: Datos del vehículo conducido y del chofer que lo conduce.
- Chofer: Cuenta con el Número de Licencia del chofer.
- Bono: Datos del bono económico que se les da a los choferes.
- Agente: Datos del agente que da una multa.

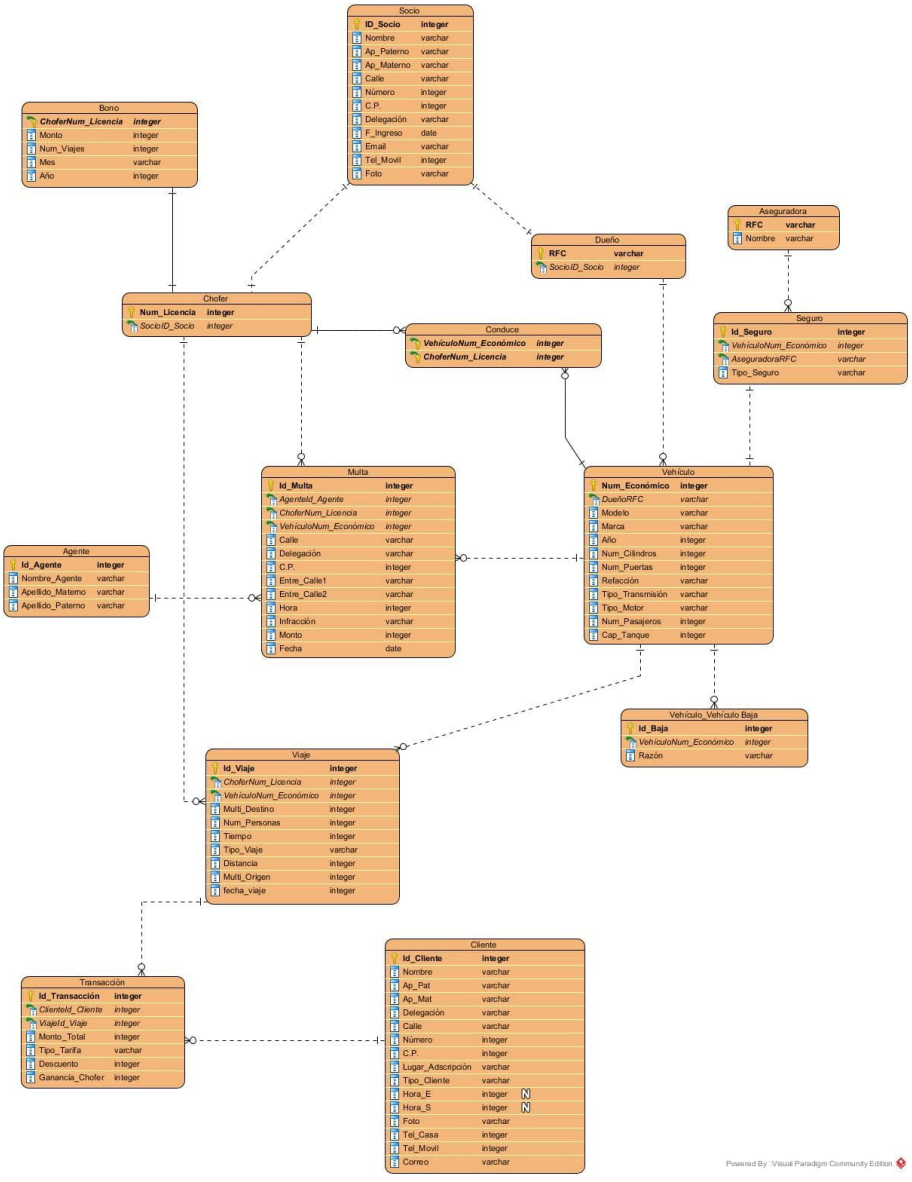


Figura 2: Modelo relacional

3. Normalización

Dependencias Funcionales:

- SOCIO
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: ID_Socio → Nombre,Ap_Materno,Ap_Materno,Calle, Numero, C.P. , Delegacion, F_Ingreso, Email, Tel_Movil, Foto.
- CHOFER
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Num_Licencia → Socio_ID_Socio,
- DUEÑO
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: FRC → Socio_ID_Socio.
- VEHICULO
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Num_Economico → DueñoRFC, Modelo, Marca, Año, Num_Cilindros, Num_Puertas, Relacion, Tipo_Transmision, Tipo_Motor, Num_Pasajeros, Cap_Tanque.
- VEHICULO_BAJA
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Id_Baja → VehiculoNum_Economico, Razon.
- ASEGURADORA
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: RFC → Nombre.
- SEGURO
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Id_Seguro → VehiculoNum_Economico, AseguradoraRFC, Tipo_Seguro
- AGENTE
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Id_Agente → Nombre_Agente, Apellido_Pat, Apellito_Mat.
- MULTA
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Id_Multa → ChoferNum_Licencia, AgenteId_Agente, VehiculoNum_Economico, Calle, Delegacion, C.P. , EntreCalle1, EntreCalle2, Hora, Infraccion, Monto, Fecha.
- VIAJE
  La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Id_Viaje → ChoferNum_Licencia, VehiculoNum_Economico, Multi_Destino, Num_Personas, Tiempo, Tipo_Viaje, Distancia, Multi_Origen.
- CLIENTE La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Id_Cliente → Nombre, Ap_Pat, Ap_Mat, Delegacion, Calle, Numero, C.P. , Lugar_Adscripcion, Tipo_Cliente, Hora_E, Hora_S, Foto, Tel_Casa, Tel_Movil.

- TRANSACCION La unicas dependencias funcionales existentes en esta tabla por diseño son las de: Id_Transaccion → ViajeId_Viaje, ClienteId_Cliente, Monto_Total, Tipo_Tarifa, Descuento, Ganancia_Chofer.
- BONO La unicas dependencias funcionales existentes en esta tabla por diseño son las de: ChoferNum_Licencia, Mes, Año → Monto, Num_Viajes.
- CONDUCE Esta es una tabla sin dependencias funcionales.

NORMALIZAR LA BASE DE DATOS

Primera Forma Normal

Una relacion esta 1FN si:

- No hay orden de arriba-a-abajo en las filas.
- No hay orden de izquierda-a-derecha en las columnas.
- No hay filas duplicadas.
- Cada intersección de fila-y-columna contiene exactamente un valor del dominio aplicable (y nada más).
- Todas las columnas son regulares [es decir, las filas no tienen componentes como IDs de fila, IDs de objeto, o timestamps ocultos].

Nuestras tablas cumplen con estas condiciones por lo que estan en **1FN**.

Segunda Forma Normal

Una relación está en **2FN** si está en **1FN** y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir que no existen dependencias parciales. (Todos los atributos que no son clave principal deben depender únicamente de la clave principal). Nuestras tablas cumplen estar en **1FN** y como todas las relaciones solo dependende la clave principal, entonces no existen dependencias parciales por lo que estan en **2FN**.

Tercera Forma Normal

Una tabla está en 3NF si y solo si las tres condiciones siguientes se cumplen:

- La tabla está en la segunda forma normal (2NF)
- Ningún atributo no-primario de la tabla es dependiente transitivamente de una clave primaria
- Es una relación que no incluye ningún atributo clave

Nuestras tablas estan en 2FN y nuevamente como todas las relaciones solo dependende la clave principal, no hay dependencias transitivas por lo que esta en 3FN.

Forma Normal Boyce-Codd (FNBC)

Se dice que una tabla está en FNBC si y solo si está en 3FN y cada dependencia funcional no trivial tiene una clave candidata como determinante. En términos menos formales, una tabla está en FNBC si está en **3FN** y los únicos determinantes son claves candidatas.
Como en nuestras tablas todas las relaciones solo dependende la clave principal entonces cumplen con esta condicion por lo que estan FNBC.

4. Definición de tablas en PostgreSQL

- Tabla para Vehículo, incluye los datos tal y como se especifica en la sección 2.

```
CREATE TABLE vehiculo (
   num_economico int,
   modelo varchar(30),
   marca varchar(30),
   ano int ,
   num_cilindros int ,
   num_puertas int,
   refaccion char(1),
   tipo_transmicion varchar(15),
   num_pasajeros int,
   tipo_motor varchar(15),
   cap_tanque real,
   PRIMARY KEY (num_economico)
);
```

- Tabla para Socio, especifica datos compartidos entre Chofer y Dueño, su llave primaría es un identificador para socio.

```
CREATE TABLE socio (
   id_socio int,
   nombre varchar(30),
   ap_paterno varchar(30),
   ap_materno varchar(30),
   calle varchar(80),
   numero int ,
   cp int,
   delegacion varchar(50),
   f_ingreso date,
   email varchar(80),
   tel_movil char(10),
   foto varchar(100),
   PRIMARY KEY(id_socio)
);
```

- Tabla para Cliente, especifica datos del cliente.

```
CREATE TABLE cliente (
  id_cliente int,
  nombre varchar(30),
  ap_paterno varchar(30),
  ap_materno varchar(30),
  delegacion varchar(50),
  calle varchar(80),
  numero int,
  cp int,
  lugar_adscripcion varchar(150),
  --a academico , t trabajador , e estudiante
  tipo_cliente char(1),
  hora_e time,
  hora_s time,
  foto varchar(100),
  tel_casa char(8),
  tel_movil char(10),
  mail varchar(50),
  PRIMARY KEY(id_cliente)

);
```

- Tabla para Dueño, Especifica el RFC del dueño de algún vehículo.

```
CREATE TABLE dueno(
   id_socio int,
   rfc char(14),
   PRIMARY KEY (rfc)
);
```

- Tabla para Chofer, Utiliza como llave primaria, el número de licencia del Chofer.

```
CREATE TABLE chofer(
   id_socio int,
   numero_licencia char(8),
   PRIMARY KEY (numero_licencia)
);
```

- Tabla para Multa, Contiene los datos sobre la multa.

```
CREATE TABLE multa(
   id_multa int,
   id_agente int,
   numero_licencia char(8),
   num_economico int,
   calle varchar(80),
   delegacion varchar(50),
   cp char(5),
   entre_calle1 varchar(80),
   entre_calle2 varchar(80),
   hora time,
   fecha date,
   infraccion varchar(120),
   monto real,
   PRIMARY KEY (id_multa)
);
```

- Tabla Conduce, muestra el id del vehículo y id del chofer que lo conduce .

```
CREATE TABLE conduce(
  num_economico int,
  numero_licencia char(8)
);
```

- Tabla Aseguradora, tiene como atributos nombre y rfc de la aseguradora.

```
CREATE TABLE aseguradora(
   rfc char(14),
   nombre varchar(150),
   PRIMARY KEY(rfc)
);
```

- Tabla para Seguro, tabla que contiene el tipo de seguro.

```
CREATE TABLE seguro(
   id_seguro int,
   rfc char(14),
   num_economico int,
   tipo_seguro varchar(50),
   PRIMARY KEY (id_seguro)
);
```

- Tabla Viaje, Contiene los datos del viaje.

```
CREATE TABLE viaje(
   id_viaje int,
   numero_licencia char(8),
   num_economico int,
   multi_destino char(1) DEFAULT 'N',
   num_personas int DEFAULT 0,
   --tiempo en minutos
   tiempo int DEFAULT 0,
   tipo char(7),
   distancia real DEFAULT 0,
   multi_origen char(1) DEFAULT 'N',
   fecha  date,
   PRIMARY KEY (id_viaje)
);
```

- Tabla para Transacción, tabla que guarda costos de viaje y ganancias del chofer.

```
CREATE TABLE transaccion(
   id_transaccion int,
   id_viaje int,
   id_cliente int,
   monto_total real,
   --promocional, normal
   tipo_tarifa varchar(80) ,
   descuento real,
   ganancia_chofer real,
   PRIMARY KEY(id_transaccion)
);
```

- Tabla para Baja, Contiene los datos de vehículos dados de baja.

```
CREATE TABLE baja(
   id_baja int,
   num_economico int,
   fecha date,
   razon varchar(100),
   PRIMARY KEY(id_baja)
);
```

- Tabla para bono, Contiene los datos del bono económico que obtendrá el chofer.

```
CREATE TABLE bono(
   numero_licencia char(8),
   ano int,
   mes int,
   num_viajes int DEFAULT 0,
```

```
    monto real DEFAULT 0,
    PRIMARY KEY (numero_licencia,ano,mes)
);
```

- Tabla para Agente, Contiene los datos del agente.

```
CREATE TABLE agente(
    id_agente int,
    nombre varchar(30),
    ap_pat varchar(30),
    ap_mat varchar(30),
    PRIMARY KEY(id_agente)
);
```

5. Procedimientos Almacenados y Disparadores

Implementamos dos procedimientos almacenados cuya definicion ,de ambos , esta en procedimientos.sql

- El primero idealmente tendría que estar asociado un temporizador y se tendría que ejecutar cada cierto tiempo sin embargo no supimos como hacer esto.Este procedimiento es para calcular los bonos de los taxistas y se llama poner_bono y se llama sin argumentos.

- El segundo procedimiento almacenado que implementamos fue para mantener la consistencia entre los viajes y las transacciones asi como para aplicar los descuentos especificados en el proyecto lo asociamos antes de la inserción de una transacción.Lo que hace este procedimiento es en resulmen lo siguente:
  asigna la tarifa por default que es salida en 15 y cada kilometro 8 pesos.Luego si es aplica los descuentos dependiendo si esta dentro de C.U. o fuera.Despúes les hace el descuento dependiendo su condicion de estudiante trabajador o academico.Y al final realiza los descuentos por las personas extra en el viaje.

6. Consultas

–01 La aseguradora con mas asegurados

```
select rfc from (select rfc, count(rfc) as cuenta from seguro group by rfc order
by cuenta desc) as data LIMIT 1;
```

–02 El vehiculo mas multado

```
select num_economico from (select num_economico, count(num_economico) as cuenta
from multa group by num_economico order by cuenta desc) as data LIMIT 1;
```

–03 El condutor mas multado

```
select numero_licencia from (select numero_licencia, count(numero_licencia)
as cuenta from multa group by numero_licencia order by cuenta desc) as data
LIMIT 1;
```

–04 El agente con mas multas generadas

```
select id_agente from (select id_agente, count(id_agente) as cuenta from multa
group by id_agente order by cuenta desc) as data LIMIT 1;
```

–05 El chofer que mas ha ganado con bonos

```
select numero_licencia from (select numero_licencia, sum(monto) as total  from
bono group by numero_licencia order by total desc) as data LIMIT 1;
```

–06 El chofer que mas bonos a tenido

```
select numero_licencia from (select numero_licencia, count(numero_licencia)
as num_bonos  from bono group by numero_licencia order by num_bonos desc) as
data LIMIT 1;
```

–07 El numero de socios por C.P.

```
select cp from (select cp, count(cp) as cuenta from socio group by cp order
by cuenta desc) as data LIMIT 1;
```

–08 El numero de chofers que manejas mas de 1 vehiculo

```
select count(numero_licencia) from (select numero_licencia, count(numero_licencia)
as num_carros from conduce group by numero_licencia order by num_carros) as
data where num_carros > 1;
```

–09 El numero de dueños con mas de 1 vehiculo

```
select count(rfc) from (select rfc, count(rfc) as num_carros from vehiculo group
by rfc order by num_carros) as data where num_carros > 1;
```

–10 El C.P. donde mas multas se levantan

```
select cp from (select cp, count(cp) as cuenta from multa group by cp order
by cuenta desc) as data LIMIT 1;
```

–11 El chofer con mas kilometros recorridos

```
select numero_licencia from (select numero_licencia, sum(distancia) as dist
from viaje group by numero_licencia order by dist desc) as data LIMIT 1;
```

–12 El cliente con mas kilometros recorridos

```
select id_cliente from (select id_cliente, sum(distancia) as dist from (select
transaccion.id_cliente as id_cliente, viaje.* from viaje INNER JOIN transaccion
ON viaje.id_viaje = transaccion.id_viaje) as data group by id_cliente order
by dist desc) as data LIMIT 1;
```

–13 El vehiculo con mas kilometros recorridos

```
select num_economico from (select num_economico, sum(distancia) as dist from
viaje group by num_economico order by dist desc) as data LIMIT 1;
```

–14 El chofer con mayores ganancias por viajes

```
select numero_licencia from (select numero_licencia, sum(ganancia_chofer) as
total_ganancia from (select transaccion.ganancia_chofer as ganancia_chofer,
viaje.* from viaje INNER JOIN transaccion ON viaje.id_viaje = transaccion.id_viaje)
as data group by numero_licencia order by total_ganancia desc) as data LIMIT
1;
```

–15 La tiempo acumulado de todos los viajes multi destino y multi origen

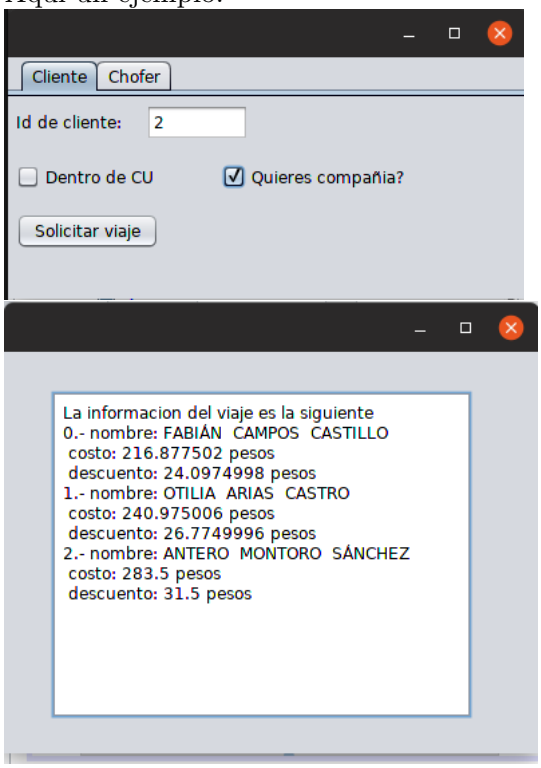
```
select sum(tiempo) from viaje where multi_destino = 'S' and multi_origen = 'S';
```

7. Aplicación

Nuestra aplicacion se describe de la siguiente manera:
Tiene dos pestañas una para los clientes y una para los choferes.

7.1. Clientes

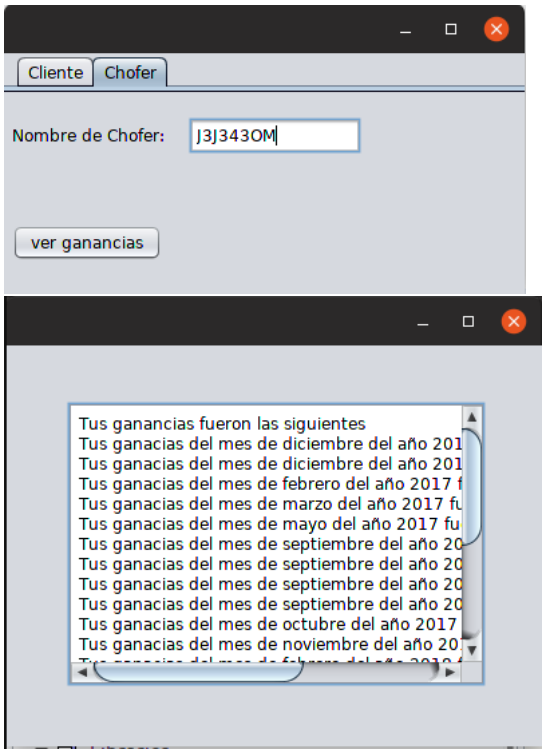
En el caso de los clientes se les pide que den su id , si es dentro o fuera de C.U. y si quiere ir acompañado.Esto para aplicar los descuentos correspondientes.En caso de que quiera ir acompañado el programa tiene una pequeña rutina para generar usuarios y asignarlos a su viaje. Toma en cuenta que no manejamos los erroes y que si das un identificador que no existe en la base de datos posiblemente tenga un comportamiento inesperado.
Aqui un ejemplo:



La salida solo son los nombres de los clientes con los que se le asigno viajar ademas de mostrar el el costo y los descuentos.

7.2. Choferes

En el caso de los choferes solo se provee un servicio para ver sus ganancias ordenados por mes y por año.Unicamente debes ingresar el numero de licencia para obtener estos.Tampoco maneja los errores.
Ejemplo



7.3. Consideraciones

Toma en cuenta que debes cambiar los parametros como sesion y usuario para poder realizar la conexión con postgres.