Universidad Nacional Autónoma De México
Facultad De Ciencias, 2025-II
Fundamentos De Bases De Datos

Practica05

Lenguaje para Definición de Datos. DDL, (Data definition Language)

Nombre del equipo:
NAMEisNULL

Integrantes:

Flores Mata Ricardo - 422127808

Matute Cantón Sara Lorena - 319331622

Martinez Leal Jose Maria - 317243970

Sánchez Cruz Norma Selene - 320198508

Suárez Ortiz Joshua Daniel - 320151260

Villegas Martínez Vania Victoria - 418003114

Restricciones del Modelo Relacional y consideraciones de diseño y traducción.

.

Entidades Fuertes y Atributos Multivaluados:

Bazar (IdBazar, NombreBazar, Calle, NúmeroInterior, NúmeroExterior, Colonia, Estado, Modalidad, FechaInicio, FechaFin)

1. Tiene como llave primaria IdBazar, el cual se considera como un entero positivo consecutivo.
2. Para los atributos NombreBazar, Calle, NúmeroInterior, NúmeroExterior, Colonia y Estado; su dominio es *varchar* y su longitud se considera en el modelo relacional.
3. En el atributo Modalidad guardamos si el bazar es al aire libre o en un lugar cerrado.
4. FechaInicio y FechaFin son de tipo *date* en un formato DD/MM/YYYY.

AmenidadBazar (IdBazar, AmenidadBazar)

1. Tabla del atributo Multivaluado Amenidades.
2. AmenidadBazar es de tipo *varchar*, para poder registrar todas las amenidades de un Bazar.
3. Llave primaria compuesta por IdBazar y AmenidadBazar.
4. Llave Foránea IdBazar la cual proviene como llave primaria de **Bazar**.

Negocio (IdNegocio, NombreNegocio, Descripción, PrecioMínimo, PrecioMáximo)

1. Tiene como llave primaria IdNegocio y lo consideramos como un entero positivo consecutivo.
2. Los atributos NombreNegocio y Descripción son de dominio varchar y su longitud se especifica en modelo relacional.
3. PrecioMínimo y PrecioMáximo se consideran como dominio numeric, el cual tiene una precisión de 2 decimales.

TelefonoNegocio (IdNegocio, Teléfono)

1. Tabla del atributo Multivaluado Teléfono de **Negocio**.
2. Télefono es de tipo *char* de longitud 10.
3. Llave primaria compuesta por IdNegocio y Teléfono.
4. Llave Foránea IdNegocio la cual proviene como llave primaria de **Negocio**.

CorreoNegocio (IdNegocio, Correo)

1. Tabla del atributo Multivaluado Correo **Negocio**.
2. Correo es de tipo *varchar* de longitud 50.
3. Llave primaria compuesta por IdNegocio y Correo.
4. Llave Foránea IdNegocio la cual proviene como llave primaria de **Negocio**.

RedSocialNegocio (IdNegocio, RedSocial)

1. Tabla del atributo Multivaluado RedSocial **Negocio**.
2. Correo es de tipo *varchar* de longitud 50.
3. Llave primaria compuesta por IdNegocio y RedSocial.
4. Llave Foránea IdNegocio la cual proviene como llave primaria de **Negocio**.

Estand (Número, Paquete, Precio)

1. Tiene como llave primaria Número y lo consideramos como int entre 1 y 20.
2. El atributo Paquete lo consideramos para guardar el Paquete que eligieran los emprendedores al momento de reservar el Estand para ocupar en el Bazar, por lo que su dominio es char(1), únicamente para elegir el número del paquete.
3. El atributo Precio es para guardar el precio del paquete que se haya elegido, por lo que su dominio es money.

AmenidadEstand (Número, AmenidadEstand)

1. Tabla del atributo Multivaluado Amenidad de **Estand**.
2. AmenidadEstand es de tipo *varchar* de longitud 100.
3. Llave primaria compuesta por Número y AmenidadEstand.
4. Llave Foránea Número la cual proviene como llave primaria de **Estand**.

Ticket (IdTicket, idCliente, idBazar, ComisiónBazar, idNegocio, RFC)

1. Su identificador es **id_Ticket** y lo consideramos como un entero positivo consecutivo.
2. El atributo **Precio Total** no se traduce porque es calculado (y depende del 20% de comisión para el bazar).
3. Se agrega idCliente por la traducción de la relación **Cliente - Comprar - Ticket**.
4. Se agregan idBazar y ComisiónBazar por la traducción de la relación **Bazar - Pagar - Ticket**.
5. Se agrega idNegocio por la traducción de la relación **Negocio - Imprimir - Ticket**.
6. Se agrega RFC por la traducción de la relación **Emprendedor - Cobrar - Ticket**.

PersonalOrganizador:

En la traducción del modelo entidad-relación al modelo relacional, siguiendo las reglas estándar, la entidad fuerte **Personal Organizador** se representaría como una superentidad de las entidades especializadas: **Seguridad, Limpieza y Médico**, lo que resultaría en la creación de 3 tablas separadas en la base de datos:

1. Seguridad
2. Limpieza
3. Médico

En este diseño o traducción en particular no estamos de acuerdo, puesto que lo que único que diferencia de Médico, Limpieza y Seguridad, únicamente es el título, pues estas entidades especializadas no tienen un atributo de más. Y adjunto a esto queremos cambiar el diseño por dos razones principales:

- Mejor organización del personal organizador: Mantener la información de los trabajadores en una única tabla facilita la administración y consulta de datos.
- Garantizar unicidad en la categoría del trabajador: Cada trabajador debe pertenecer únicamente a una categoría. Es decir, no puede ser simultáneamente un Médico y un Empleado de Limpieza, por lo que cada RFC debe aparecer solo una vez en la base de datos.

Para lograr esto, en lugar de dividir la información en múltiples tablas, optaremos por un diseño que mantenga una única tabla **PersonalOrganizador** con un atributo que indique a qué categoría pertenece (Seguridad, Limpieza o Médico). Así, cada trabajador tendrá un solo registro en la base de datos, asegurando que cada RFC esté asociado exclusivamente a un tipo de Trabajador.
Por lo que, siguiendo con la versión reducida del Modelo Relacional, tenemos que:

PersonalOrganizador (RFC, NombrePersonalOrganizador, APaternoPersonalOrganizador, AMaternoPersonalOrganizador, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Salario, esSeguridad, esLimpieza, esMédico)

1. Tiene como identificador **RFC**, considerado como un tipo de dato *char*: 4 letras, 6 dígitos, 2 letras y 1 dígito.

2. Los atributos NombreTrabajador, APaternoTrabajador, AMaterno, Calle, NúmeroExterior, NúmeroInterior, Colonia y Estado; son de tipo *varchar* y su longitud se define en el diagrama del modelo relacional.
3. FechaNacimiento es de tipo *date* en un formato DD/MM/YYYY.
4. Salario es de tipo *money*
5. esSeguridad, esLimpieza, esMédico; son de tipo *boolean* y solamente uno puede tener el valor *true*. Esto nos permite cumplir con la restricción de los casos de uso.

HorarioPersonalOrganizador (RFC, HorarioPersonalOrganizador)

1. Tabla del atributo Multivaluado Horario de **PersonalOrganizador**.
2. Horario es de tipo *varchar* de longitud 30.
3. Llave primaria compuesta por RFC y HorarioPersonalOrganizador.
4. Llave Foránea RFC la cual proviene como llave primaria de **PersonalOrganizador**.

TeléfonoPersonalOrganizador (RFC, TeléfonoPersonalOrganizador)

1. Tabla del atributo Multivaluado Teléfono de **PersonalOrganizador**.
2. TeléfonoPersonalOrganizador es de tipo *char* de longitud 10.
3. Llave primaria compuesta por RFC y TeléfonoPersonalOrganizador.
4. Llave Foránea RFC la cual proviene como llave primaria de **PersonalOrganizador**.

CorreoPersonalOrganizador (RFC, CorreoPersonalOrganizador)

1. Tabla del atributo Multivaluado Correo de **PersonalOrganizador**.
2. CorreoPersonalOrganizador es de tipo *varchar* de longitud 50.
3. Llave primaria compuesta por RFC y CorreoPersonalOrganizador.
4. Llave Foránea RFC la cual proviene como llave primaria de **PersonalOrganizador**.

Emprendedor (RFC, NombreEmprendedor, APaternoEmprendedor, AMaternoEmprendedor, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, Género, idNegocio)

1. Tiene como identificador **RFC**, considerado como un tipo de dato *char*: 4 letras, 6 dígitos, 2 letras y 1 dígito.
2. Los atributos NombreEmprendedor, APaternoEmprendedor, AMaternoEmprendedor, Calle, NúmeroExterior, NúmeroInterior, Colonia y Estado; son de tipo *varchar* y su longitud se define en el diagrama del modelo relacional.
3. Género es de tipo *char* de longitud 1.
4. se le agrega la llave foránea *idNegocio* de Negocio por la relación emprender.

TeléfonoEmprendedor (RFC, TeléfonoEmprendedor)

1. Tabla del atributo Multivaluado Teléfono de **Emprendedor**.
2. TeléfonoEmprendedor es de tipo *char* de longitud 10.
3. Llave primaria compuesta por RFC y TeléfonoEmprendedor.
4. Llave Foránea RFC la cual proviene como llave primaria de **Emprendedor**.

CorreoEmprendedor (RFC, CorreoEmprendedor)

1. Tabla del atributo Multivaluado Correo de **Emprendedor**.
2. CorreoEmprendedor es de tipo *varchar* de longitud 50.
3. Llave primaria compuesta por RFC y CorreoEmprendedor.
4. Llave Foránea RFC la cual proviene como llave primaria de **Emprendedor**.

Cliente (idCliente, NombreCliente, APaternoCliente, AMaternoCliente, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado, esFísico, esVirtual)

1. Es superentidad de: **Físico y Virtual**, consideramos una especialización total con traslape. Por lo que se genera una única tabla Cliente con todos los atributos
2. Su identificador es **id_Cliente** y lo consideramos como entero positivo consecutivo.
3. NombreCliente, APaternoCliente, AMaternoCliente, Calle, NúmeroExterior, NúmeroInterior, Colonia, Estado son atributos de tipo *varchar* y su longitud se especifica en el modelo relacional.
4. esFísico y esVirtual son atributos de tipo *boolean* y no hay problema si ambos están en modo *true*.

MétodoPago (idCliente, MétodoPago)

1. Tabla del atributo multivaluado MétodoPago de **Cliente**.
2. MétodoPago es de tipo *varchar* para elegir entre efectivo o tarjeta.
3. Llave primaria compuesta por idCliente y MétodoPago.
4. Llave foránea idCliente que proviene de **Cliente** que es llave primaria.

Tarjeta (NúmeroTarjeta, Vencimiento, CVV, idCliente)

1. NúmeroTarjeta es la llave primaria y lo consideramos como tipo de dato *char* y su longitud es de 16 caracteres.
2. Vencimiento es de tipo *date* en un formato DD/MM/YYYY.
3. CVV es de tipo *varchar* y tiene una longitud de 4 caracteres (porque existen tarjetas con un cvv de longitud 3 y otras de 4).
4. Se agrega idCliente por la traducción de la relación **Cliente - Domiciliar - Tarjeta**.

Entidades débiles:

.

Producto (idNegocio, idProducto, NombreProducto, Descripción, Tipo, Precio, Presentación, Stock)

1. idNegocio es heredado de Negocio porque **Producto** es una entidad débil.
2. idProducto es de tipo *int* como entero positivo consecutivo.
3. NombreProducto, Descripción, Tipo, Presentación son de tipo *varchar* y su longitud se especifica en el modelo relacional.
4. PrecioProducto es de tipo money.
5. Stock se agrega por la traducción de la relación **Negocio - Vender - Producto**.

Servicio (idNegocio, idServicio, NombreServicio, Descripción, Tipo, PrecioServicio, Duración)

1. idNegocio es heredado de Negocio porque **Producto** es una entidad débil.
2. idServicio es de tipo *int* como entero positivo consecutivo.
3. NombreServicio, Descripción, Tipo son de tipo *varchar* y su longitud se especifica en el modelo relacional.
4. PrecioServicio es de tipo *money*.
5. Duración es de tipo *time* (dado que solamente necesitamos las horas y/o minutos).

Perecedero (idNegocio, idProducto, FechaPreparación, FechaCaducidad)

1. Hereda idNegocio yidProducto de la tabla **Producto**.
2. FechaPreparación y FechaCaducidad son de tipo *date* en un formato DD/MM/YYYY.

Nota: Separamos Producto y Servicio en dos entidades débiles separadas debido a que pensamos que un negocio puede ofrecer servicios y vender productos al mismo tiempo sin que una afecte la otra.

Relaciones que generan tabla:

.

Trabajar (idBazar, RFC, FechaAsistencia)

- idBazar, RFC son llaves foráneas por la cardinalidad n:m de la relación Trabajar entre Bazar y Personal Organizador.
- Fecha asistencia se considera de tipo date.

RegistrarProducto (idTicket,idNegocio, idProducto, Cantidad)

- idTicket y (idNegocio,idProducto) son llaves foráneas por la cardinalidad n:m de la relación Registrar entre Ticket y Producto.
- Cantidad es un dato de tipo *int*

RegistrarServicio (idTicket, idNegocio, idServicio, Duración)

- idTicket y (idNegocio, idServicio) son llaves foráneas por la cardinalidad n:m de la relación Registrar entre Ticket y Servicio.
- Duración es un atributo de tipo *time* (dado que solamente necesitamos las horas y/o minutos que dura el servicio).

Agendar (idBazar, idNegocio, NúmeroEstand, FechaAsistencia)

- idBazar, idNegocio, NúmeroEstand son llaves que se agregan a la tabla porque es una relación n-aria.
- Tiene como discriminante el atributo **FechaAsistencia** para garantizar que un Negocio no registre dos veces que asistirá el mismo día al mismo Bazar.
- Por lo que, como consideración de diseño, para esta tabla se define la siguiente llave compuesta: idNegocio, FechaAsistencia.

Relaciones que modifican tabla:

.

Comprar

- Relación entre Cliente y Ticket.
- Como tiene cardinalidad n:1, modifica la tabla **Ticket** mandando la llave primaria de Cliente, es decir, idCliente se agrega a la tabla Ticket.

Domiciliar

- Relación entre ClienteVirtual (que por la traducción, la tabla se llama Cliente) y Tarjeta.
- Por su cardinalidad 1:n, modifica la tabla **Tarjeta**, mandando la llave primaria de **Cliente**, es decir, idCliente se agrega a la tabla Tarjeta.

Pagar

- Relación entre Bazar y Ticket.
- Por su cardinalidad 1:n, modifica la tabla Ticket mandando la llave primaria de Bazar, es decir, idBazar se agrega a la tabla Ticket.
- El atributo de la relación pagar, **ComisiónBazar**, también se agrega a la tabla Ticket.

Imprimir

- Relación entre Negocio y Ticket.
- Por su cardinalidad 1:n modifica la tabla Ticket, agregando la llave primaria de Negocio, es decir, idNegocio se agrega a la tabla Ticket.

Cobrar

- Relación entre Emprendedor y Ticket.
- Por su cardinalidad 1:n, agrega la llave primaria de Emprendedor a la tabla Ticket, es decir, el RFC del Emprendedor se agrega a Ticket.

Emprender

- Relación entre Negocio y Emprendedor.
- Por su cardinalidad 1:n, modifica la tabla Emprendedor, agregando la llave primaria de Negocio.
- Como su único atributo es calculado, no se agrega al modelo relacional.

Vender

- Relación débil entre Negocio y Producto.
- Por su debilidad, no se genera tabla (porque ya está considerada en la tabla **Producto**). Sin embargo, al tener atributos, se agregan a la tabla **Producto**, en este caso, solamente se agrega Stock.

Ofrecer

- Relación entre Negocio y Servicio.

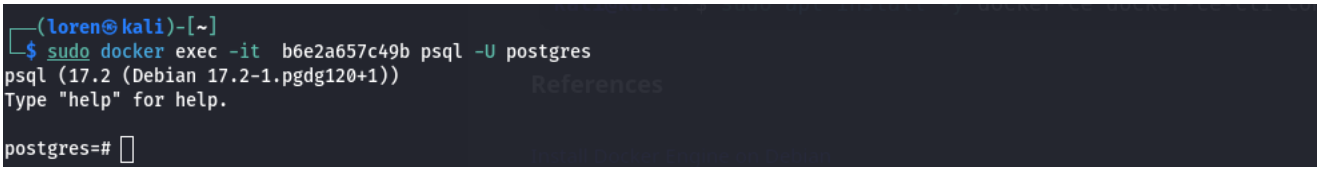
- Como es una relación débil, no se traduce porque ya está considerada en la tabla **Servicio** y como no tiene atributos, tampoco modifica ninguna tabla.

Restaurar backup de una BD en Docker

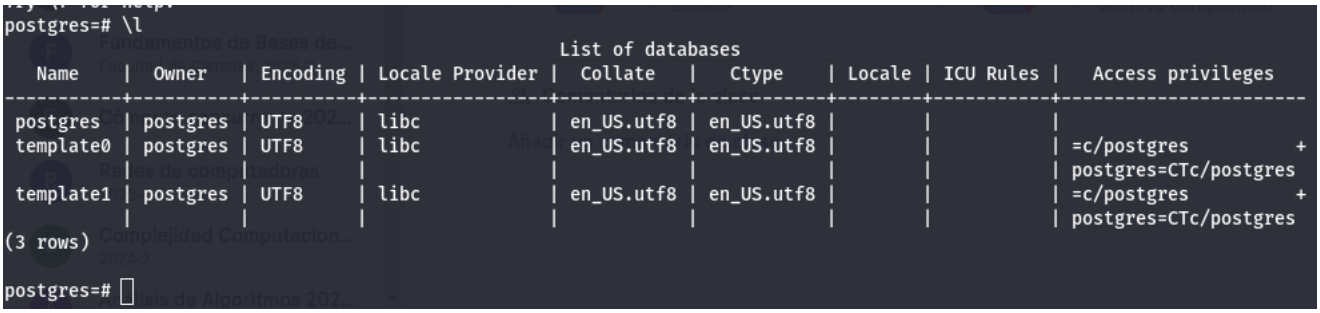
Para este ejercicio seguimos los pasos mostrados en el archivo "Restaurar backup de una BD en Docker." proporcionados por el ayudante. En particular se seguirán los pasos para el proceso en linux.

Mediante el comando sudo docker ps -a listamos nuestros contenedores de docker para buscar el id de postgres. Observamos que en este caso el id es b6e2a657c49b.

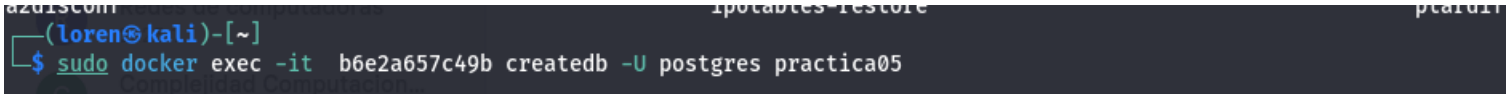
Una vez que obtuvimos el id accedemos a su psql.



Una vez ahí vimos nuestras bases de datos. En este caso observamos que solo teníamos las necesarias para el funcionamiento de postgres.



Posteriormente creamos una nueva base de datos vacía para poder realizar el backup. A esta nueva base de datos le llamamos practica05.



Una vez creada accedimos a su psql.

Al ser una nueva base de datos observamos que no había alguna relación, por lo cual procedimos con el backup.

Finalmente llegamos al paso más importante. Utilizando el comando pg_restore restauramos el archivo .backup. A continuación solo se muestra un fragmento del resultado del comando.

```
┌──(loren㉿kali)-[~]
└─$ sudo docker exec -i b6e2a657c49b pg_restore --verbose --clean --no-acl --no-owner -U  postgres -d  practica05 < /home/loren/Documents/ejemplo.backup
pg_restore: connecting to database for restore
pg_restore: dropping FK CONSTRAINT trolebus trolebus_fkey
pg_restore: while PROCESSING TOC:
pg_restore: from TOC entry 5056; 2606 225711 FK CONSTRAINT trolebus trolebus_fkey postgres
pg_restore: error: could not execute query: ERROR:  relation "public.trolebus" does not exist
Command was: ALTER TABLE ONLY public.trolebus DROP CONSTRAINT trolebus_fkey;
pg_restore: dropping FK CONSTRAINT trenligero trenligero_fkey
pg_restore: from TOC entry 5052; 2606 225655 FK CONSTRAINT trenligero trenligero_fkey postgres
pg_restore: error: could not execute query: ERROR:  relation "public.trenligero" does not exist
Command was: ALTER TABLE ONLY public.trenligero DROP CONSTRAINT trenligero_fkey;
pg_restore: dropping FK CONSTRAINT tipotransporte tipotransporte_fkey
pg_restore: from TOC entry 5050; 2606 225627 FK CONSTRAINT tipotransporte tipotransporte_fkey postgres
pg_restore: error: could not execute query: ERROR:  relation "public.tipotransporte" does not exist
Command was: ALTER TABLE ONLY public.tipotransporte DROP CONSTRAINT tipotransporte_fkey;
pg_restore: dropping FK CONSTRAINT telefono telefono_fkey
pg_restore: from TOC entry 5045; 2606 225570 FK CONSTRAINT telefono telefono_fkey postgres
pg_restore: error: could not execute query: ERROR:  relation "public.telefono" does not exist
Command was: ALTER TABLE ONLY public.telefono DROP CONSTRAINT telefono_fkey;
pg_restore: dropping FK CONSTRAINT taxi taxi_fkey
pg_restore: from TOC entry 5057; 2606 225725 FK CONSTRAINT taxi taxi_fkey postgres
pg_restore: error: could not execute query: ERROR:  relation "public.taxi" does not exist
Command was: ALTER TABLE ONLY public.taxi DROP CONSTRAINT taxi_fkey;
pg_restore: dropping FK CONSTRAINT tarjeta tarjeta_fkey
pg_restore: from TOC entry 5058; 2606 225735 FK CONSTRAINT tarjeta tarjeta_fkey postgres
pg_restore: error: could not execute query: ERROR:  relation "public.tarjeta" does not exist
Command was: ALTER TABLE ONLY public.tarjeta DROP CONSTRAINT tarjeta_fkey;
pg_restore: dropping FK CONSTRAINT rtp rtp_fkey
pg_restore: from TOC entry 5055; 2606 225697 FK CONSTRAINT rtp rtp_fkey postgres
pg_restore: error: could not execute query: ERROR:  relation "public.rtp" does not exist
Command was: ALTER TABLE ONLY public.rtp DROP CONSTRAINT rtp_fkey;
pg_restore: dropping FK CONSTRAINT reparartrolebus reparartrolebus_fkey2
pg_restore: from TOC entry 5098; 2606 226059 FK CONSTRAINT reparartrolebus reparartrolebus_fkey2 postgres
pg_restore: error: could not execute query: ERROR:  relation "public.reparartrolebus" does not exist
```

Para comprobar que este proceso haya sido exitoso entramos una vez más al psql de practica05 y observamos sus relaciones. Comprobamos que el procedimiento fue exitoso al aparecer en pantalla las siguientes relaciones.

```
                 List of relations
 Schema |          Name          | Type  |  Owner
--------+------------------------+-------+----------
 public | componerse             | table | postgres
 public | contar                 | table | postgres
 public | deshabilitar           | table | postgres
 public | dia                    | table | postgres
 public | especialidad           | table | postgres
 public | estacion               | table | postgres
 public | examenmedico           | table | postgres
 public | incidente              | table | postgres
 public | licencia               | table | postgres
 public | linea                  | table | postgres
 public | manejarmetro           | table | postgres
 public | manejarmetrobus        | table | postgres
 public | manejarmicrobus        | table | postgres
 public | manejarrtp             | table | postgres
 public | manejartaxi            | table | postgres
 public | manejartrenligero      | table | postgres
 public | manejartrolebus        | table | postgres
 public | metro                  | table | postgres
 public | metrobus               | table | postgres
 public | microbus               | table | postgres
 public | operador               | table | postgres
 public | pagarmetro             | table | postgres
 public | pagarmetrobus          | table | postgres
 public | pagarmicrobus          | table | postgres
 public | pagarrtp               | table | postgres
 public | pagartaxi              | table | postgres
 public | pagartrenligero        | table | postgres
 public | pagartrolebus          | table | postgres
 public | reparametro            | table | postgres
 public | reparametrobus         | table | postgres
 public | reparamicrobus         | table | postgres
 public | reparartp              | table | postgres
 public | reparartaxi            | table | postgres
 public | reparartrenligero      | table | postgres
```

Finalmente revisamos que aparecieran en el dbeaver.