UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Proyecto final

Alumno:

Javier Alejandro Rivera Zavala - 311288876
Abraham Jimenez Reyes - 318230577

Profesor

Victor Manuel Corza Vargas

AYUDANTES

Alexis Hérnandez Castro
Diana Irma Canchola Hernández
Gibrán Aguilar Zuñiga
Rogelio Alcantar Arenas

ASIGNATURA

Fundamentos de bases de datos

12 de junio del 2023

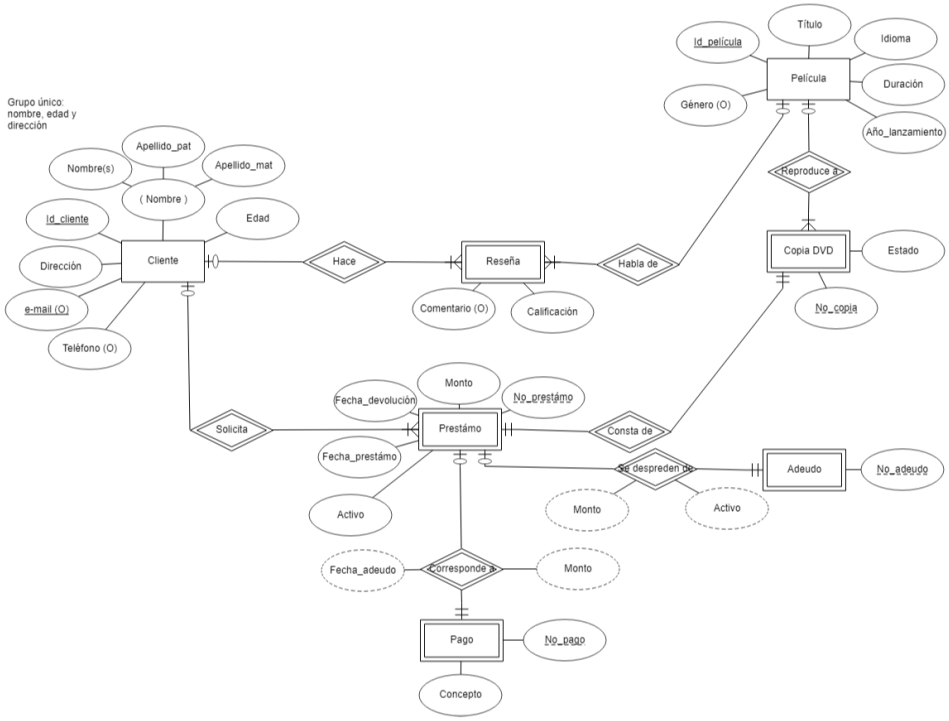
Ejercicios

1. Lista de requerimientos para la base de datos propuesta.
   La cadena "Ciencias Buster" quiere aprovechar el furor por los formatos físicos así que han decidido abrir algunos centros para rentar películas en formato DVD. Se le ha solicitado a nuestro equipo, desarrollar un ejemplo sobre un par de bases de datos que serán empleadas por los sistemas de la compañía, para poder tener una idea de su funcionamiento en interacción con dichos sistemas y así decidir si se nos contrata. La base que se abordará en el presente documento, simula la base que se empleará en las computadoras (en una versión más sencilla), desde las cuales se les dará servicio a los clientes. Se nos pide tomar en consideración que, parte de la información almacenada en esta base, será trasladada a otra base después de ser procesada adecuadamente por el sistema, dicha información tiene que ver con el dinero recaudado a lo largo de un periodo de operaciones a definir posteriormente y sus particularidades. Los requisitos que se nos pide cumplir, son los siguientes:

   - La base permite guardar información sobre las películas de forma simple, no nos extenderemos en los detalles aún, sólo necesitamos poder visualizar y manipular información concreta sobre películas. Es necesario poder guardar información de películas con las que aún no contamos físicamente aunque ya han sido terminadas de filmar y registradas para su lanzamiento. También se debe poder guardar información sobre aquellas con las que por algún motivo no contamos en el momento, pero pronto serán abastecidas. La información de las películas es aquella que usualmente se coloca en sus fichas informativas y debe de ser sólida (no es posible dejar alguna característica sin llenar).
   - Se debe guardar información general sobre los clientes, de igual forma guardaremos su información de forma condensada. Debe de incluir al menos una de 2 vías de contacto con el cliente, se puede aceptar sólo una pero no ninguna y se necesita la dirección de su domicilio. No se puede omitir información para identificar a los clientes, salvo la excepción ya mencionada.
   - Debe ser posible almacenar información de las copias en DVD con las que contamos en el momento, sólo pueden haber hasta 5 copias por sucursal y se debe de gestionar el estado en el cuál se encuentra cada una de nuestras copias. La información debe de estar completa para cada copia.
   - Se deben almacenar reseñas sobre las películas emitidas por nuestros clientes, le pueden asignar una calificación de 0 a 5 puntos a cada película y hacer un comentario al respecto, aunque pueden omitir este último. Se permite reseñar películas de entre aquellas en el repertorio de la tienda, aunque no la hayan rentado en la misma.
   - Se debe llevar registro de los préstamos con su costo base. No se pueden hacer nuevos préstamos a un usuario si este tiene algún adeudo activo, no se pueden prestar copias dañadas o extraviadas, no se puede prestar la misma película (en distintas copias) al mismo cliente al mismo tiempo, no se le pueden hacer más de 3 préstamos al mismo usuario al mismo tiempo y no se puede prestar una misma copia a más de un cliente al mismo tiempo. Debe de haber consistencia entre las fechas, la de termino debe de ser mayor o igual a la de inicio.
   - Se debe llevar registro de los adeudos, cada uno debe estar asociado a su respectivo préstamo y el monto debe de incluir los cargos (por retardo, daño ó extravío), mismos que el sistema gestionará eventualmente así como sus tarifas, pero no nosotros de momento. No puede registrarse un adeudo si el préstamo asociado está activo.
   - Se deben almacenar los pagos hechos por cada préstamo (con cargos de adeudo, si los hay), de momento se deja la gestión de la activación y desactivación de adeudos y préstamos mediante pago al personal que use el sistema, esto debido a eventuales ofertas y condonaciones que

puedan presentarse en el futuro. Las tarifas aún no se han decidido y no corren por nuestra cuenta.

2. Modelo conceptual de la base de datos, respetando la nomenclatura de Peter Chen.

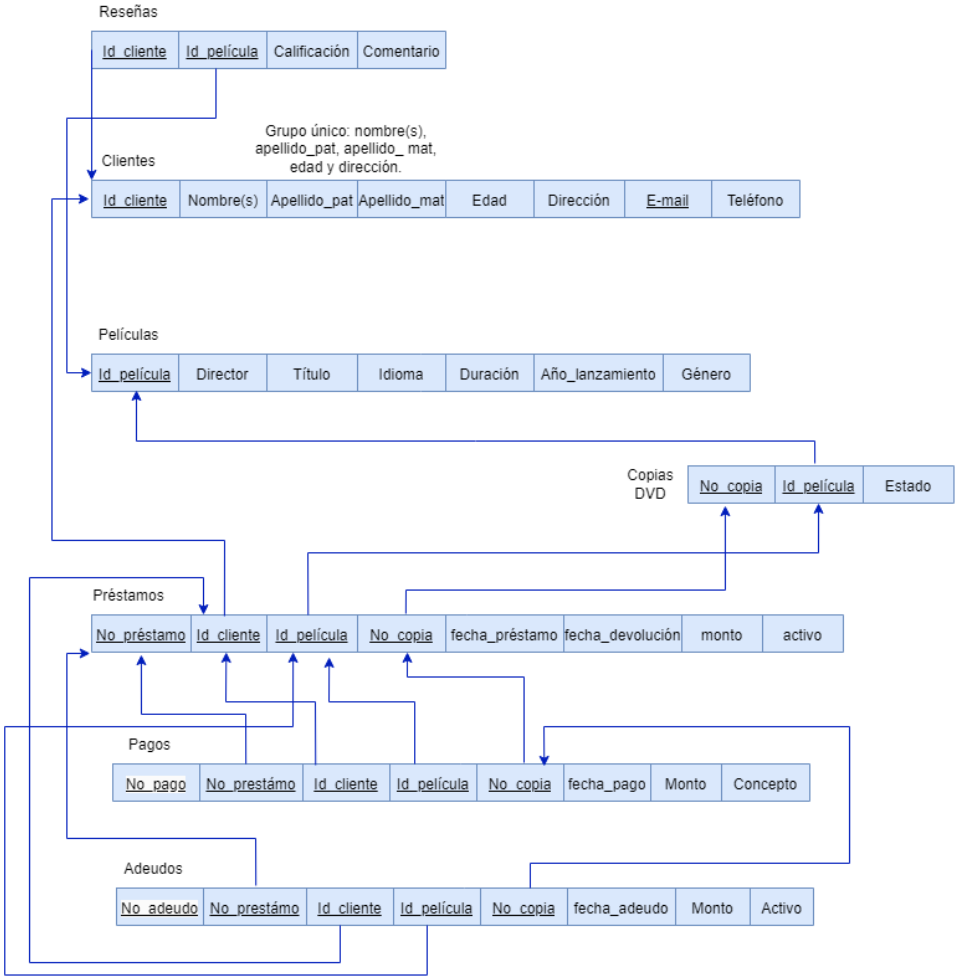


Notemos lo siguiente, sólo los clientes y películas son considerados entidades simples pues pueden existir independientemente en el esquema, por otro lado, las reseñas, los préstamos, adeudos, pagos y copias dependen de las películas y de los clientes directamente o bien de forma terciaria. No tiene sentido que las entidades que aquí modelamos como débiles, existan de forma independiente en el esquema, sin embargo, recordemos que se pretende usar la información desprendida de dichas entidades, para ser procesada y almacenar los resultados en otra base. El procesamiento de los datos correrá a cargo del sistema y no es nuestra tarea modelarlo, por el momento, pero será importante considerarlo cuanto indiquemos las restricciones adecuadas en nuestras instrucciones SQL.
Hay un conjunto de atributos que definen a las entidades tipo cliente, no por separado cada uno pero si juntas, se verán más a detalle cuando definamos las instrucciones SQL que crean la tabla en la base de datos. Respecto a las participaciones y su cardinalidad consideremos que:

- Un cliente puede hacer múltiples reseñas pero cada reseña sólo proviene de un cliente. Un cliente puede solicitar múltiples préstamos pero cada préstamo en particular, está asociado a un único cliente (los préstamos se registran de forma individualizada, no por fecha), un cliente puede estar registrado pero nunca requerir de nuestros servicios.
- Cada préstamo puede (o no) estar asociado a un sólo adeudo o pago y a su vez, cada pago corresponde a un sólo préstamo y cada adeudo también.
- Una película puede o no tener asociadas hasta 5 copias, pero cada copia corresponde a una sola película.

Hay atributos que dependen de la forma en la que las entidades débiles se relacionan con la entidades que las determinan, se verá con más claridad cuando construyamos las respectivas instrucciones SQL.

3. Modelo relacional.
   Se llevó a cabo el respectivo mapeo del diagrama E/R con la notación de Peter Chen hacia el modelo relacional. Notemos que se subrayan individualmente los campos que constituyen la llave primaria en cada relación (van juntos desde la esquina izquierda hacia la derecha), pero en realidad es el grupo en su totalidad el que constituye dicha llave. Para distinguir al grupo en "Clientes" que constituye una llave foránea sin dar lugar a una confusión, preferí usar un cuadro de texto encima de la relación o del conjunto de atributos en el ejercicio anterior.



4. Explicaremos la forma en que se construyeron las instrucciones SQL para crear la base de datos.

   Para crear la tabla "Clientes" empleamos como llave primaria al id del cliente, cuya unicidad será garantizada en el tramite de registro de un usuario. En el contexto de la base de datos, el id nos permite identificar unívocamente al cliente, sin embargo, para evitar redundancia en el registro, proponemos como llave candidata** la combinación de los campos nombres, apellido_paterno, apellido_materno, edad y dirección que permiten distinguir a una persona en el mundo físico, salvo que

sean gemelos con el mismo nombre y vivan donde mismo, pero de momento no seremos tan específicos. La dirección de e-mail de una persona en teoría es única, personal e intransferible, por lo qué podría identificar a cada persona de forma única, sin embargo, dado que no toda persona tiene una dirección de e-mail, no tomamos este campo como llave primaria. Ningún campo puede ser nulo, salvo el teléfono o el e-mail pero no ambos al mismo tiempo y usamos un dominio personalizado para las edades que va desde 18 hasta 130, pues sólo aceptamos clientes mayores de edad (y no hay hasta la fecha personas con edad mayor a 130).
** Las declaramos como CONSTRAINT ... UNIQUE ..., técnicamente son equivalentes en este contexto pero esta opción permitía disponer de ellas de forma más sencilla.

```
CREATE TABLE Clientes (
  id_cliente VARCHAR(11) PRIMARY KEY,
  nombres VARCHAR(30) NOT NULL,
  apellido_paterno VARCHAR(30) NOT NULL,
  apellido_materno VARCHAR(30) NOT NULL,
  direccion VARCHAR(70) NOT NULL,
  email VARCHAR(50) UNIQUE,
  telefono VARCHAR(10),
  edad rango_edad NOT NULL,
  CONSTRAINT chk_email_telefono CHECK ((email IS NOT NULL AND telefono IS NULL) OR
    (email IS NULL AND telefono IS NOT NULL) OR (email IS NOT NULL AND telefono IS
    NOT NULL)),
  CONSTRAINT chk_email_format CHECK (email IS NULL OR email LIKE '%@%'),
  CONSTRAINT chk_telefono_format CHECK (telefono IS NULL OR (telefono ~ '^55\d{8}$'
    )),
  CONSTRAINT chk_id_cliente_format CHECK (id_cliente ~ '^[A-Za-z]{5}\d{6}$'),
  CONSTRAINT unq_cliente_datos UNIQUE (nombres, apellido_paterno, apellido_materno,
     edad, direccion)
);
```

Para crear la tabla “Películas”, empleamos como llave primaria al id de la película (código ISAN en el mundo real o alguno similar) que debe identificar de manera única a cada película y por otro lado como llave candidata tenemos al conjunto de los campos titulo, año_lanzamiento y director que en general deberían bastar para identificar de manera única a una película, salvo en casos excepcionales. Restringimos el año de lanzamiento al rango 1900 y el presente año, no encontramos conveniente para esta base de prueba almacenar lanzamientos más allá del presente año, entre otras cosas por qué su código ISAN (o alguno equivalente) puede no haber sido asignado aún. No aceptamos campos nulos salvo el género, que puede ser indescifrable en el caso del cine experimental o de autor. Aunque nos referimos al ISAN, para esta simulación emplearemos un formato personalizado por cuestiones de practicidad. Cambiamos las ñ por ni en el código presentado aquí, por problemas con Latex.

```
CREATE TABLE Peliculas (
  id_pelicula VARCHAR(10) PRIMARY KEY CHECK (id_pelicula ~ '^[A-Za-z]{3}\d{3}[A-Za-
    z]{2}\d{2}$'),
  titulo VARCHAR(50) NOT NULL,
  anio_lanzamiento INT NOT NULL CHECK (anio_lanzamiento BETWEEN 1900 AND EXTRACT(
    YEAR FROM CURRENT_DATE)),
  director VARCHAR(90) NOT NULL,
  idioma VARCHAR(30) NOT NULL,
  duracion TIME NOT NULL,
  genero VARCHAR(30),
  CONSTRAINT unq_peli UNIQUE (titulo, anio_lanzamiento, director)
);
```

Para la tabla que registra las copias, el campo no_copia está restringido según lo convenido y será dicho campo junto con el id de la película la combinación que constituye la llave primaria para

cada copia. Como llave foránea tenemos al id de la película y se emplean los trigger de integridad referencial ON DELETE RESTRICT y ON UPDATE CASCADE, dado que buscamos asegurar que se vacíen en otra base los datos comerciales antes de borrar información de la base local, ponemos la primera restricción para asegurar la visualización de dicha información (y su debida manipulación) antes de borrar usuarios, dicha manipulación escapa del alcance de esta base de muestra, pero las restricciones indicadas nos ayudan a simular el comportamiento esperado del sistema. En el caso de UPDATE no permitimos que los cambios sobre el id se propaguen a las copias, así sin más, pues puede haber algunas de ellas en circulación y eso generaría problemas además de que el id de una película no puede cambiar así por que sí en el mundo real.

```
CREATE TABLE Copias (
  no_copia INT NOT NULL CHECK (no_copia BETWEEN 1 AND 5),
  id_pelicula VARCHAR(10) NOT NULL,
  estado estado_copia NOT NULL,
  PRIMARY KEY (no_copia, id_pelicula),
  FOREIGN KEY (id_pelicula) REFERENCES Peliculas(id_pelicula) ON DELETE RESTRICT ON
    UPDATE RESTRICT
);
```

En el caso siguiente, las reseñas podrían ser gestionadas y registradas como los datos comerciales, por ello podríamos restringir su eliminación si están asociadas a un cliente o una película en particular que se desean eliminar. Con fines pedagógicos implementaremos los triggers ON DELETE CASCADE y ON UPDATE CASCADE para tener un poco de variedad en cuánto a los mismos y por qué en una aplicación real, la extracción y depuración de datos sobre las reseñas puede ser gestionada por el sistema, además, sería conveniente no guardar reseñas de películas o clientes inexistentes. No aceptamos campos nulos salvo el comentario, que el usuario se puede ahorrar.

```
CREATE TABLE Resenias (
  id_pelicula VARCHAR(10) NOT NULL,
  id_cliente VARCHAR(11) NOT NULL,
  calificacion INT NOT NULL CHECK (calificacion BETWEEN 0 AND 5),
  comentario TEXT,
  PRIMARY KEY (id_pelicula, id_cliente),
  FOREIGN KEY (id_pelicula) REFERENCES Peliculas(id_pelicula) ON DELETE CASCADE ON
    UPDATE CASCADE,
  FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente) ON DELETE CASCADE ON
    UPDATE CASCADE,
);
```

Para la siguiente tabla, la llave primaria estará constituida por los campos no_prestamo, id_cliente, id_pelicula y no_copia, el número de préstamo se genera de forma seriada (para tener los préstamos enumerados) y aunque en principio es único, dada la interacción de está tabla con otras 2 dependientes de la misma (que cuentan con un campo serial cada una), para asegurar la unicidad de registros es que empleamos a tal conjunto como llave primaria (y para mantener la dependencia entre entidades). Las llaves foráneas serán el id del cliente que solicita el préstamo así como el número de la copia solicitada con su respectivo id de la película. No se admiten campos nulos y empleamos triggers ON DELETE RESTRICT y ON UPDATE RESTRICT para garantizar la consistencia de nuestros datos comerciales ante la modificación de los respectivos campos en las tablas “Clientes” y “Copias”, de lo contrario podríamos tener conflictos sobre el registro de nuestras operaciones comerciales. Notemos que aún así, no se impide que los registros puedan ser modificados individualmente en otros aspectos, tales modificaciones deben de ser gestionadas a nivel de permisos o bien, por el sistema en general y escapan del alcance de esta práctica. Evitamos la redundancia de registros al exigir unicidad de la combinación de los campos id_cliente, id_pelicula, fecha_prestamo

y fecha_devolucion. Por el momento no añadimos restricción para la fecha de inicio, esto para poder ingresar préstamos viejos en la tabla y poder trabajar con ella en la presente practica, es tarea del usuario cuidar ese detalle aunque en una implementación final, tendría que cuidarse ese aspecto.

```
CREATE TABLE Prestamos (
  no_prestamo SERIAL,
  id_cliente VARCHAR(11) NOT NULL,
  id_pelicula VARCHAR(10) NOT NULL,
  no_copia INT NOT NULL,
  fecha_prestamo DATE NOT NULL,
  fecha_devolucion DATE NOT NULL,
  monto DECIMAL(11, 5) NOT NULL,
  activo BOOLEAN NOT NULL DEFAULT false CHECK ((activo = (fecha_devolucion >=
    CURRENT_DATE)) OR (activo = false AND fecha_devolucion = CURRENT_DATE)),
  PRIMARY KEY (no_prestamo, id_cliente, id_pelicula, no_copia),
  FOREIGN KEY (id_cliente) REFERENCES Clientes(id_cliente) ON DELETE RESTRICT ON
    UPDATE RESTRICT,
  FOREIGN KEY (id_pelicula, no_copia) REFERENCES Copias(id_pelicula,no_copia) ON
    DELETE RESTRICT ON UPDATE RESTRICT,
  CONSTRAINT check_fechas CHECK (fecha_devolucion >= fecha_prestamo),
  CONSTRAINT unq_copia_activo EXCLUDE (no_copia WITH =) WHERE (activo = true),
  CONSTRAINT unq_peli_simul EXCLUDE (id_cliente WITH =, id_pelicula WITH =) WHERE (
    activo = true),
  CONSTRAINT unq_fecha_prestamo UNIQUE (id_cliente, id_pelicula, fecha_prestamo,
    fecha_devolucion)
);
```

En la tabla “Adeudos” tenemos una llave primaria constituida por los campos no_adeudo, no_prestamo, id_cliente, id_pelicula y no_copia que nos permite identificar de manera única a cada adeudo, dada la interacción de esta tabla con la tabla “Prestamos” es que recurrimos a esta llave, pues aunque el número de adeudo es seriado, en su interacción con la tabla préstamos no nos garantiza identificar a cada registro adecuadamente. Como mencionamos en el caso de la tabla anterior, para asegurar la consistencia de los datos comerciales, se restringe la eliminación y modificación en las tablas “Clientes” y “Copias”. Esta restricción nos obliga a revisar el registro de los adeudos de un cliente antes de efectuar su respectiva eliminación. Si bien la característica antes mencionada no necesariamente debe de ser gestionada a nivel de base de datos, en esta base necesitamos simular el comportamiento final de nuestro sistema en su conjunto, dados los requerimientos del cliente. Evitamos la redundancia de registros al exigir la unicidad de la combinación de los campos id_cliente, id_pelicula y fecha_adeudo.

```
CREATE TABLE Adeudos (
  no_adeudo SERIAL,
  no_prestamo INT NOT NULL,
  id_cliente VARCHAR(11) NOT NULL,
  id_pelicula VARCHAR(10) NOT NULL,
  no_copia INT NOT NULL,
  fecha_adeudo DATE NOT NULL,
  monto DECIMAL(11, 5) NOT NULL,
  activo BOOLEAN NOT NULL,
  PRIMARY KEY (no_adeudo, no_prestamo, id_cliente, id_pelicula, no_copia),
  FOREIGN KEY (no_prestamo, id_cliente, id_pelicula, no_copia) REFERENCES Prestamos
    (no_prestamo, id_cliente, id_pelicula, no_copia) ON DELETE RESTRICT ON UPDATE
    RESTRICT,
  CONSTRAINT unq_fecha_adeudo UNIQUE (id_cliente, id_pelicula, fecha_adeudo)
);
```

Finalmente, en la tabla “Pagos” tenemos una llave primaria constituida por los campos no_pago, no_prestamo, id_cliente, id_pelicula y no_copia que nos permite identificar de manera única a cada adeudo ya que aún cuando el número de préstamo se genera de forma seriada, dada la dependencia de esta tabla respecto a la tabla de préstamos, para asegurar la identificación clara de cada registro es que recurrimos a esta llave primaria. Como mencionamos en el caso de la tabla anterior, para asegurar la consistencia de los datos comerciales, se restringe la eliminación y modificación en las tablas “Clientes” y “Copias”. Esta restricción nos obliga a revisar el registro de los adeudos de un cliente antes de efectuar su respectiva eliminación. Como en el caso de la tabla anterior, esta característica no necesariamente debe de ser gestionada a nivel de base de datos, en esta base necesitamos simular el comportamiento final de nuestro sistema en su conjunto, dados los requerimientos del cliente. Evitamos la redundancia de registros al exigir la unicidad de la combinación de los campos id_cliente, id_pelicula y fecha_pago.

```
CREATE TABLE Pagos (
  no_pago SERIAL,
  no_prestamo INT NOT NULL,
  id_cliente VARCHAR(11) NOT NULL,
  id_pelicula VARCHAR(10) NOT NULL,
  no_copia INT NOT NULL,
  fecha_pago DATE NOT NULL,
  monto DECIMAL(11, 5) NOT NULL,
  concepto concepto_pago NOT NULL,
  PRIMARY KEY (no_pago, no_prestamo, id_cliente, id_pelicula, no_copia),
  FOREIGN KEY (no_prestamo, id_cliente, id_pelicula, no_copia) REFERENCES Prestamos
    (no_prestamo, id_cliente, id_pelicula, no_copia) ON DELETE RESTRICT ON UPDATE
    RESTRICT,
  CONSTRAINT unq_fecha_pago UNIQUE (id_cliente, id_pelicula, fecha_pago)
);
```

Notemos que las llaves primarias compuestas para las últimas 2 tablas, muestran también el hecho de que ambas entidades dependen de la existencia de un préstamo. También es importante señalar que según la bibliografía consultada e incluso al preguntar a ChatGPT al respecto, nos encontramos con que los campos de tipo SERIAL pueden dar lugar a anomalías al momento de interactuar entre si dentro de una tabla y puede haber problemas al identificar un registro, por este motivo (ante la duda) y para seguir las normas planteadas sobre la dependencia de entidades (en el esquema del cuál partimos), empleamos las llaves primarias compuestas que se exponen en el código, de otra forma, hubiésemos optado por el número de pago, de adeudo ó de préstamo como llaves primarias, queda expuesto el caso por si se cuestionan algo respecto de la minimalidad de la llave.

5. Se crearon 100 instrucciones insert para cada tabla, en algunos casos, unos cuantos más para tener variedad de opciones. Se adjunta el archivo con las instrucciones.

6. *a*) La restricción de integridad referencial presente en la instrucción FOREIGN KEY (id_pelicula) REFERENCES Peliculas(id_pelicula) ON DELETE RESTRICT ON UPDATE RESTRICT involucra a las tablas “Copias” y “Peliculas”, toma la llave primaria id_pelicula de la segunda tabla y es referenciada desde id_pelicula en la primera. Se emplean estos triggers pues una alteración en el registro de películas sobre alguna película en concreto (se borre o se cambie), mientras alguna copia de la misma se encuentra en circulación, puede dar lugar a anomalías e imprecisiones en nuestro registro comercial y eso no es aceptable. Las restricciones forzarán a verificar primero en el registro de copias si es posible efectuar cambios, esto nos conecta con la siguiente restricción que expondremos.

   b) El siguiente trigger para llave foránea se encuentra presente en la expresión FOREIGN KEY

(id_cliente) REFERENCES Clientes(id_cliente) ON DELETE RESTRICT ON UPDATE RESTRICT, e involucra a las tablas "Prestamos" y "Clientes". Lo que hacemos con este trigger es restringir el borrado/actualización de clientes, siempre que estos estén asociados a algún préstamo, de modo que aseguramos la consistencia en nuestros datos comerciales. Lo que intentamos hacer es forzar al operador de la base, a verificar primero el estado de los préstamos de un cliente antes de modificar o borrar su información, esto con miras a vaciar la información contenida en esta base dentro de una base general que llevará registro de nuestras operaciones. El sistema que hace uso de la base se encargara de procesar la información adecuadamente.

c) El trigger representado por la expresión FOREIGN KEY (no_prestamo, id_cliente, id_pelicula, no_copia) REFERENCES Prestamos(no_prestamo, id_cliente, id_pelicula, no_copia) ON DELETE RESTRICT ON UPDATE RESTRICT involucra a las tablas "Adeudos" y "Prestamos". El motivo por el cuál seleccionamos este tipo de trigger es básicamente el mismo que en los 2 casos anteriores, nos aseguramos de que haya consistencia entre los registros de nuestras operaciones comerciales y de que se haga un vaciado/análisis adecuado antes de borrar préstamos asociados a adeudos.

d) En la tabla "Reseñas" tenemos la siguiente restricción FOREIGN KEY (id_pelicula) REFERENCES Peliculas(id_pelicula) ON DELETE CASCADE ON UPDATE CASCADE e involucra además de la tabla ya mencionada, a la tabla "Peliculas". Las reseñas no son información realmente crucial sobre las operaciones de la empresa, además, su existencia está ligada a la existencia de un cliente y una película cuyo borrado ya se encuentra restringido, esto de acuerdo a los préstamos en los que se encuentren registrados, eso ya nos obliga a verificar antes con que entidades están ligados y hacer el análisis y/ó vaciado pertinente. Dado lo anterior, sólo existe un riesgo real de borrado de las reseñas que corresponden a películas tales que el cliente no ha rentado en nuestro establecimiento, pero conoce de algún otro lado. La información antes mencionada puede servirnos en el momento (para hacer recomendaciones, por ejemplo), pero realmente no es susceptible de un análisis en este nivel, por ello permitimos que las reseñas sean borradas en cascada, cuando el cliente o la película a los que están ligadas son eliminados. Por último es importante mencionar que también se opto por esta restricción un poco para darle variedad a la práctica ya que las demás restricciones son prácticamente iguales, quizás pudimos optar por lo mismo en este caso, pero había buenos motivos para no hacerlo como ya se explicó antes.

Incluimos al final del documento las capturas correspondientes que reflejan el funcionamiento de cada trigger de integridad referencial.

7. Evidencia del funcionamiento de al menos 3 restricciones check para "atributos" de varias tablas.

a) La primer restricción check está presente en la tabla "Peliculas" en la declaración del id de la película y verifica que el id siga un determinado formato. La instrucción dentro de la construcción de la tabla, es

```
CHECK (id_pelicula ~ '^[A-Za-z]{3}\d{3}[A-Za-z]{2}\d{2}$')
```

y verifica que id_pelicula tenga 3 letras al inicio, seguidas de 3 dígitos, después 2 letras y finalmente 2 dígitos.

b) La segunda restricción check está presente en la tabla "Copias" y verifica que el número de copia se encuentre en un rango determinado. La instrucción dentro de la construcción de la tabla, es

```
CHECK (no_copia BETWEEN 1 AND 5)
```

y verifica que no_copia tenga un valor entre 1 y 5.

c) La tercer restricción check se encuentra en la tabla "Reseñas" y verifica que la calificación se encuentre en un determinado rango. La instrucción dentro de la construcción de la tabla, es

```
CHECK (calificacion BETWEEN 0 AND 5)
```

y verifica que calificación tenga un valor entre 0 y 5.

Se adjuntan las capturas correspondientes al final del documento.

8. Evidencia de la creación de al menos tres dominios personalizados. Se deben utilizar restricciones check en la creación de los tres dominios.

   a) El primer dominio personalizado lo empleamos en la tabla "Clientes" y nos sirve para tener un rango de edades realista. Se emplea en el atributo edad de la tabla ya mencionada y se define así:

   ```
   CREATE DOMAIN rango_edad AS INT
     CHECK(VALUE >= 18 AND VALUE <= 130);
   ```

   Este domino nos sirve para registrar únicamente clientes mayores de edad, tan mayores como es posible según los registros históricos. Para lograr lo anterior, la instrucción check propuesta verifica que el valor de la edad esté entre 18 (edad que define la mayoría de edad en gran parte de los países del mundo) y 130 años (La persona más longeva de la que se tiene registro tuvo 122 años).

   b) El segundo dominio personalizado se emplea en la tabla "Copias" y nos sirve para definir 3 posibles estados para las copias almacenadas en cada sucursal. Se hace uso de este dominio en el atributo estado de la tabla ya mencionada y se define como sigue:

   ```
   CREATE DOMAIN estado_copia AS VARCHAR(8)
     CHECK(VALUE IN ('bien', 'dañada','perdida'));
   ```

   Este dominio es útil en tanto que de momento no tiene sentido crear toda una tabla para gestionar los distintos tipos de estados, la instrucción check verifica que la cadena ingresada para definir el estado, sea una de 3 posibles: bien, dañada ó perdida.

   c) El tercer dominio propuesto se emplea en la tabla "Pagos" y sirve para definir el tipo de pago efectuado por un cliente. La instrucción check empleada dentro de la construcción de la tabla, es la siguiente:

   ```
   CREATE DOMAIN concepto_pago AS VARCHAR(7)
     CHECK (VALUE IN ('normal', 'adeudo'));
   ```

   Este domino es útil de forma similar que el dominio anterior, asegura que la cadena ingresada como estado sea una de 2 posibles opciones de pago: adeudo y normal.

   Adjuntamos las capturas correspondientes a cada dominio al final del documento.

9. Evidencia del funcionamiento de al menos 2 restricciones para tuplas en diferentes tablas.

   a) La primer restricción para tuplas empleada en nuestra base se encuentra en la tabla "Clientes", dicha restricción se asegura de que cada cliente registrado cuente con al menos una vía de contacto. La instrucción empleada dentro de la construcción de la tabla, es la siguiente:

```
CONSTRAINT chk_email_telefono
   CHECK ((email IS NOT NULL AND telefono IS NULL) OR
   (email IS NULL AND telefono IS NOT NULL) OR
   (email IS NOT NULL AND telefono IS NOT NULL))
```

Se asegura de que en cada tupla sólo uno de los campos para guardar una vía de contacto (teléfono ó e-mail) este vació, pero no ambos al mismo tiempo. La instrucción

```
INSERT INTO Clientes (id_cliente, nombres, apellido_paterno, apellido_materno,
    direccion, email, telefono, edad)
    VALUES ('ABCDE123456', 'Juan', 'García', 'Pérez',
    'Calle 123 No. 1, Ciudad de Mexico', NULL, NULL, 23);
```

generará un error si se intenta ingresarla, pues no cumple con la restricción indicada.

b) La segunda restricción para tuplas de la que hacemos uso se encuentra presente en la tabla "Prestamos" y se asegura de que no haya préstamos activos para la misma copia de una película. La instrucción empleada dentro de la construcción de la tabla, es la siguiente:

```
CONSTRAINT unq_copia_activo EXCLUDE (no_copia WITH =, id_pelicula WITH =)
    WHERE (activo = true)
```

que hace uso de EXCLUDE y se cerciora de que no pueda haber 2 préstamos activos de la misma copia de una película. Si intentamos ingresar las instrucciones

```
INSERT INTO Prestamos (id_pelicula, id_cliente, no_copia, fecha_prestamo,
    fecha_devolucion, monto, activo)
    VALUES ('HMH052us01', 'ABCDE123456', 1, '2023-06-11',
    '2023-06-18', 20.99, true);

INSERT INTO Prestamos (id_pelicula, id_cliente, no_copia, fecha_prestamo,
    fecha_devolucion, monto, activo)
    VALUES ('HMH052us01', 'FGHIJ789012', 1, '2023-06-11',
    '2023-06-18', 20.99, true);
```

Se generará un error al tratar de efectuar el segundo insert.

Una de las restricciones opera al nivel de cada tupla individualmente, mientras que la otra opera en el total de tuplas. Tenemos otras tales como

```
CHECK ((activo = (fecha_devolucion >= CURRENT_DATE)) OR
(activo = false AND fecha_devolucion = CURRENT_DATE))
```

en la tabla préstamos que se asegura de que las fechas en el registro del préstamo sean consistentes.

10. Plantea 3 consultas que consideres relevantes para la base de datos propuesta.

a) La primer consulta nos permite conocer los ingresos percibidos por la tienda durante el mes de agosto de 2022 (podría ser algún otro si ajustamos las fechas).

```
SELECT SUM(monto) AS monto_total_agosto_2022
FROM Pagos
WHERE fecha_pago >= '2022-08-01' AND fecha_pago < '2022-09-01';
```

b) La segunda consulta nos permite conocer que películas han obtenido alguna vez la puntuación máxima por parte de los usuarios, sirve para darnos una idea inmediata sobre una posible recomendación a los usuarios. La consulta en cuestión es

```
SELECT DISTINCT titulo, director
FROM (Peliculas
JOIN Reseñas ON Peliculas.id_pelicula = Reseñas.id_pelicula) AS pelis_puntuadas
WHERE calificacion = 5;
```

Sólo nos muestra cada película una sola vez con información acotada a sus campos titulo y director.

c) Nuestra última consulta nos deja conocer la edad promedio de aquellos clientes que ven películas de terror (podría ser otro género), con tales datos podemos mejorar la experiencia del cliente y también nuestra estrategia de negocios.

```
SELECT TRUNC(AVG(edad), 0) AS promedio_edad_gustan_terror
FROM Clientes
WHERE id_cliente IN (
    SELECT DISTINCT id_cliente
    FROM (Prestamos
    JOIN Peliculas ON Prestamos.id_pelicula = Peliculas.id_pelicula)
    AS clientes_terror
    WHERE genero = 'Terror'
);
```

Como es usual, las capturas se colocan al final del documento.

11. Plantea 3 vistas que consideres relevantes para la base de datos propuesta.

a) La primer vista nos permite visualizar una tabla con el registro de aquellos clientes que no han tenido buenas experiencias (han puntuado más de una película con una calificación inferior a 3) y una vía de contacto. Es útil para dar asesoría a los clientes y así tener un mayor índice de retención, ya sea que se presenten en la tienda o se les contacte en caso de que no hayan vuelto en un tiempo.

```
CREATE VIEW clientes_insatisfechos AS
SELECT c.id_cliente, c.nombres, c.apellido_paterno, c.apellido_materno,
       COALESCE(c.email, c.telefono) AS contacto
FROM clientes c
JOIN reseñas r ON c.id_cliente = r.id_cliente
WHERE r.calificacion <= 2
GROUP BY c.id_cliente, c.nombres, c.apellido_paterno, c.apellido_materno,
c.email, c.telefono
HAVING COUNT(*) > 1;
```

b) Crearemos una vista que nos permita visualizar el historial de pagos de cada usuario en forma condensada(cadena con el registro de fechas, montos y conceptos). Será útil para imprimir facturas y similares.

```
CREATE VIEW historial_pagos AS
SELECT c.id_cliente, c.nombres || ' ' || c.apellido_paterno
       || ' ' || c.apellido_materno AS cliente,
```

```
        STRING_AGG(p.fecha_pago || ' - ' || p.monto || ' - ' || p.concepto, '; ')
        AS historial_pagos
FROM Clientes c
JOIN Pagos p ON c.id_cliente = p.id_cliente
GROUP BY c.id_cliente, c.nombres, c.apellido_paterno, c.apellido_materno;
```

c) Nuestra última vista consta de un registro conforme a la popularidad de nuestras películas, las ordenamos conforme a la cantidad de reseñas recibidas y adjuntamos su puntuación promedio.

```
CREATE VIEW peliculas_populares AS
SELECT
    p.id_pelicula,
    p.titulo,
    p.director,
    p.año_lanzamiento,
    COUNT(*) AS cantidad_reseñas,
    AVG(r.calificacion) AS puntuacion_promedio
FROM
    Peliculas p
LEFT JOIN
    Reseñas r ON p.id_pelicula = r.id_pelicula
GROUP BY
    p.id_pelicula,
    p.titulo,
    p.director,
    p.año_lanzamiento
ORDER BY
    cantidad_reseñas DESC;
```

Se adjuntan los archivos solicitados con las instrucciones para la creación de la base, para llenar la base y con las instrucciones para consultas y vistas, también se adjunta un "backup" de la base generado desde PGAdmin si lo encuentran más conveniente. Los archivos contienen comentarios para hacer más claro el código en caso de que requieran saber algo más. También como extra se crearon triggers que se activan cuanto tratamos de ingresar registros que alteran la consistencia de nuestra información (y para cumplir los requisitos planteados al inicio), vienen justo después de la tabla en la que se activan y son: trg_verificar_estado_copia, trg_verificar_adeudo_activo, trigger_verificar_cantidad_copias, trg_verificar_monto_adeudo, trg_verificar_fecha_adeudo, trg_verificar_fecha_pago y trg_verificar_actividad_prestamo. En los comentarios del código se explica la función de cada uno.

IMPORTANTE. Hacemos notar que creamos 2 bases toda vez que el profesor nos dijo que podíamos entregar 2 bases si estas tenían al menos 7 tablas (como las entregadas), si es el profesor quien revisa este proyecto, entonces él ya sabe, si no, por favor consulte con él al respecto, saludos.

Capturas

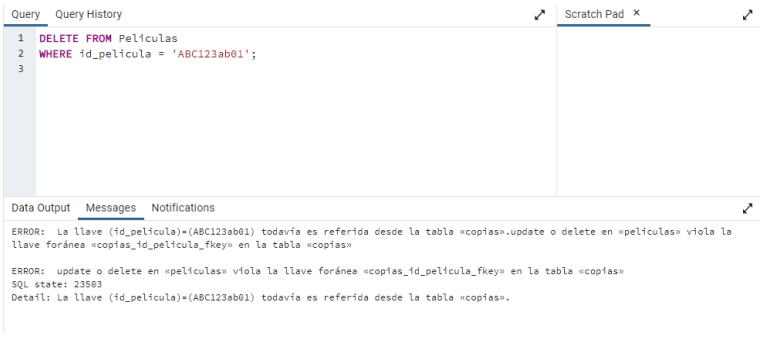


Figura 1: Primer trigger de integridad referencial

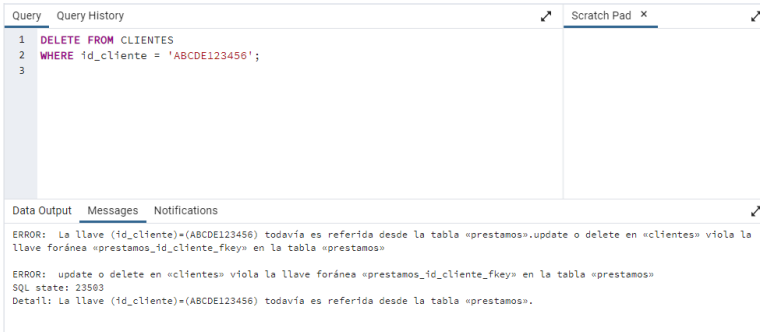


Figura 2: Segundo trigger de integridad referencial

Figura 3: Tercer trigger de integridad referencial

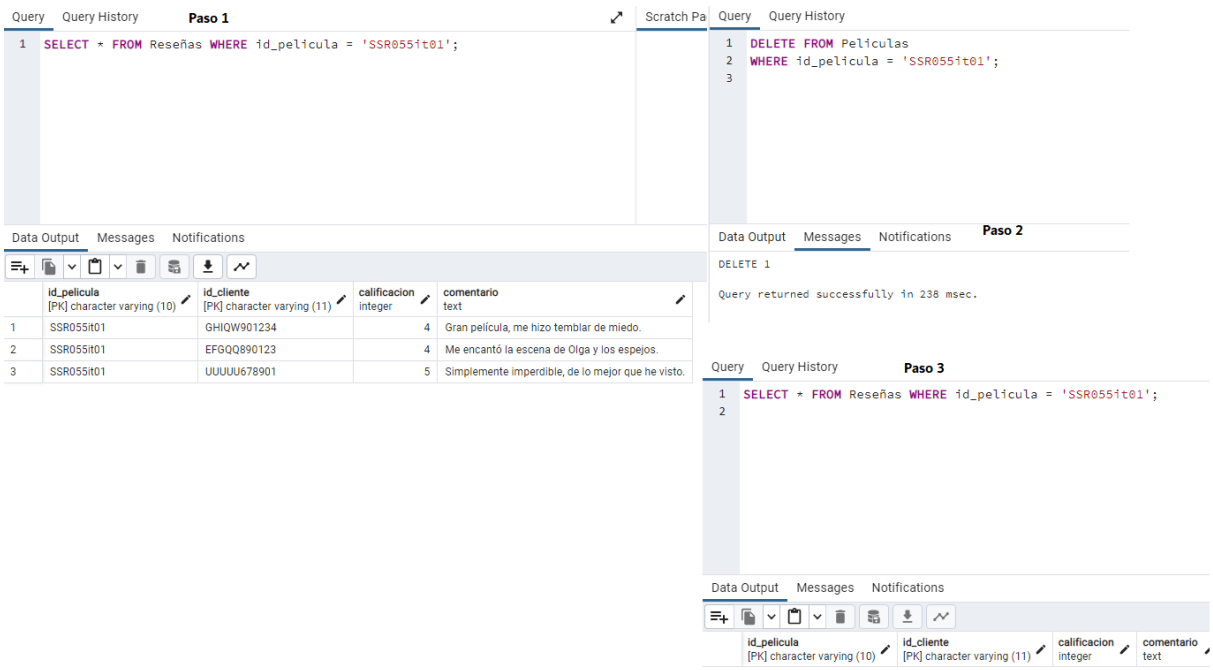


Figura 4: Cuarto trigger de integridad referencial

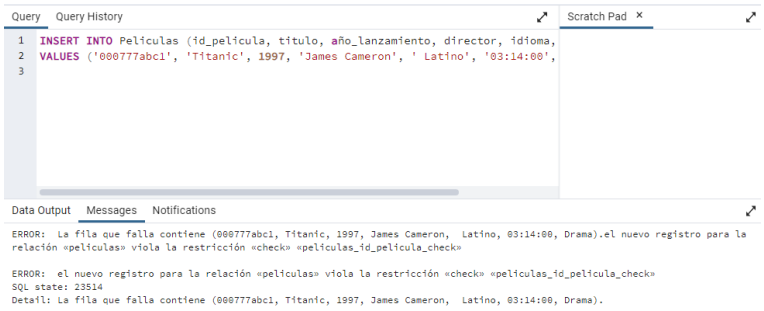


Figura 5: Primer check para atributos

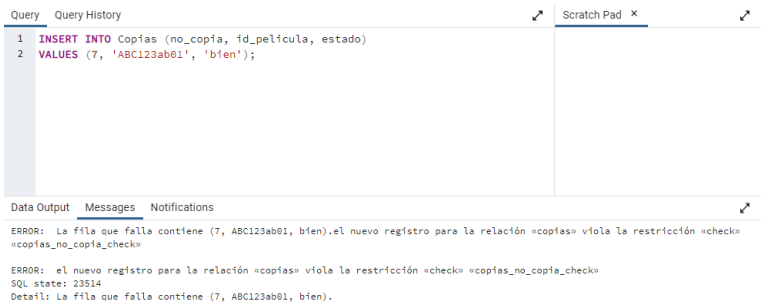


Figura 6: Segundo check para atributos

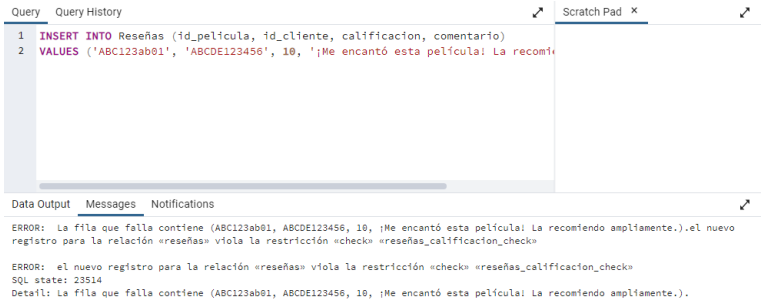


Figura 7: Tercer check para atributos

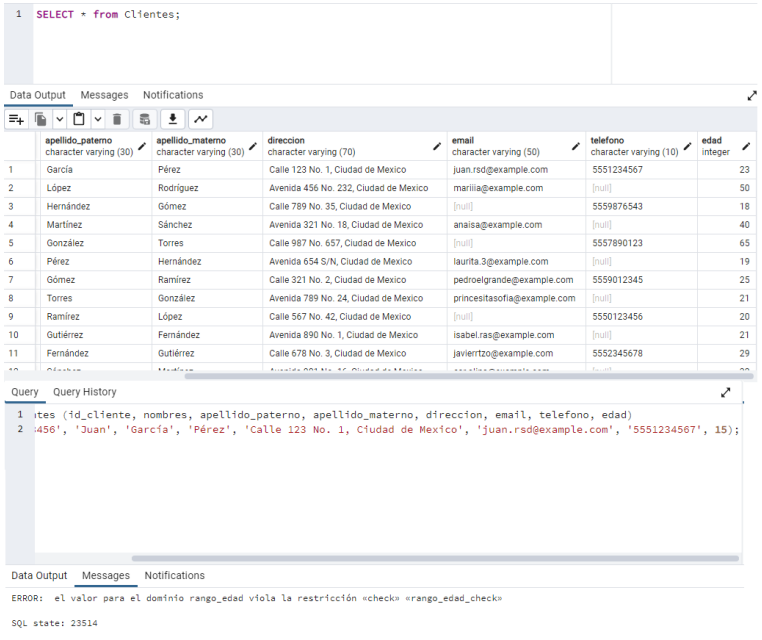


Figura 8: Primer dominio personalizado

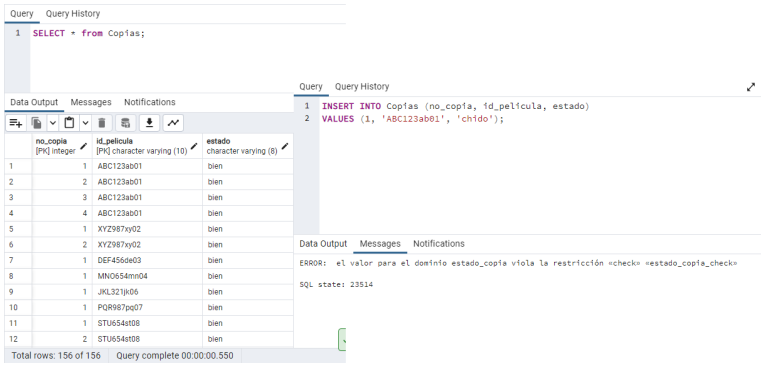


Figura 9: Segundo dominio personalizado

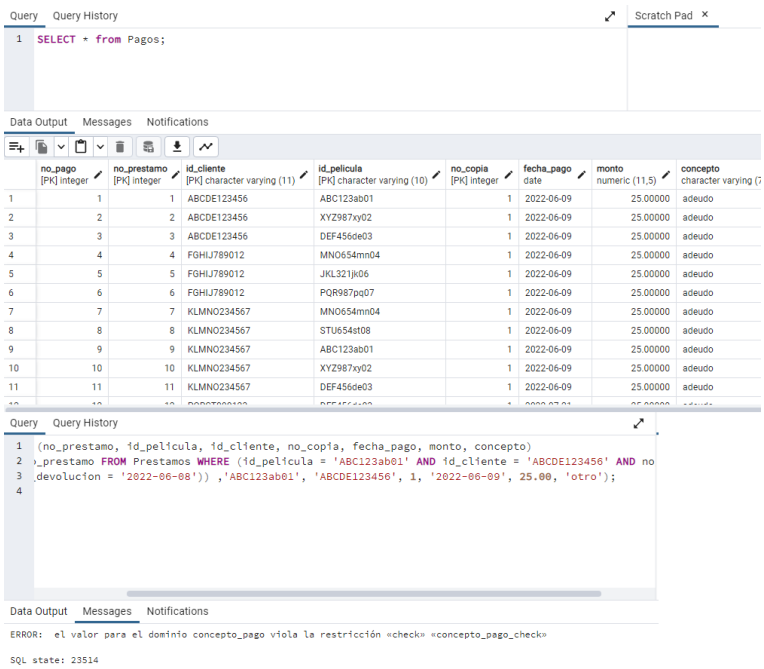


Figura 10: Tercer dominio personalizado

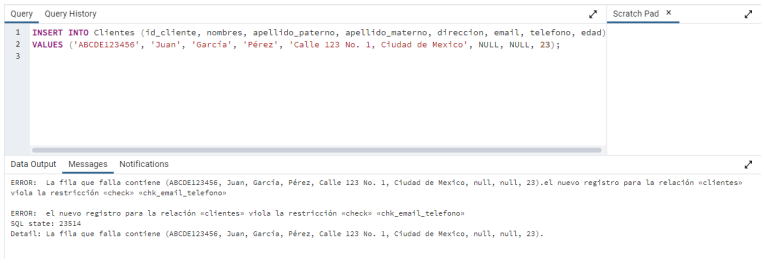


Figura 11: Primer check para tuplas

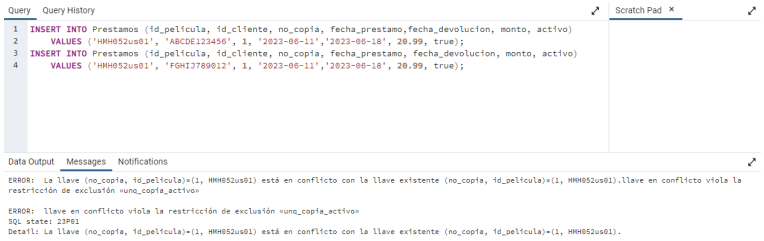


Figura 12: Segundo check para tuplas

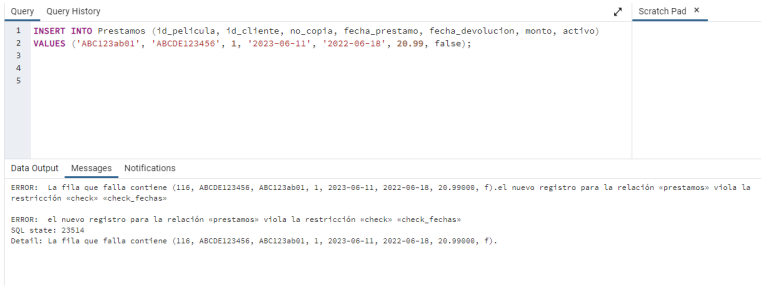


Figura 13: Tercer check para tuplas

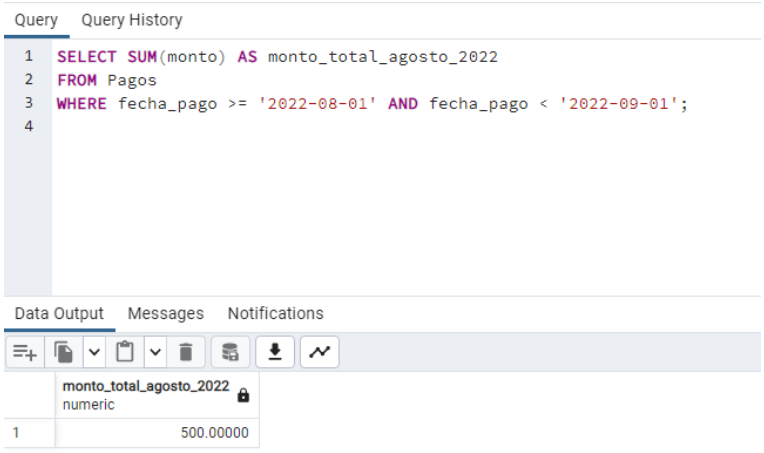


Figura 14: Primer consulta

Query Query History

```
SELECT DISTINCT titulo, director
FROM (Peliculas
JOIN Reseñas ON Peliculas.id_pelicula = Reseñas.id_pelicula) AS pelis_puntuadas
WHERE calificacion = 5;
```

Data Output Messages Notifications

	titulo character varying (50)	director character varying (90)
1	Rocky	John G. Avildsen
2	Terminator	James Cameron
3	Hable con ella	Pedro Almodóvar
4	El secreto de sus ojos	Juan José Campanella
5	La comunidad	Álex de la Iglesia
6	La double vie de Véronique	Krzysztof Kieślowski
7	El Sexto Sentido	M. Night Shyamalan
8	Nueve reinas	Fabián Bielinsky

Total rows: 21 of 21 Query complete 00:00:02.602

Figura 15: Segunda consulta

Query Query History

```
SELECT TRUNC(AVG(edad), 0) AS promedio_edad_gustan_terror
FROM Clientes
WHERE id_cliente IN (
    SELECT DISTINCT id_cliente
    FROM (Prestamos
    JOIN Peliculas ON Prestamos.id_pelicula = Peliculas.id_pelicula) AS clientes_terror
    WHERE genero = 'Terror'
);
```

Data Output Messages Notifications

	promedio_edad_gustan_terror numeric
1	25

Figura 16: Tercer consulta

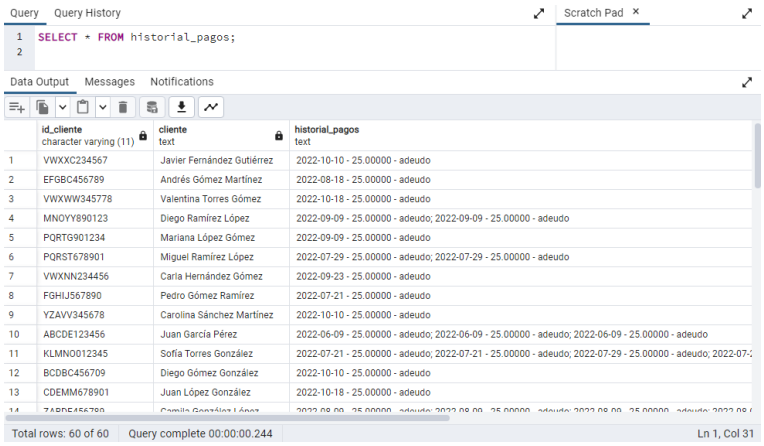
Query Query History Scratch Pad

```
SELECT * FROM clientes_insatisfechos;
```

Data Output Messages Notifications

	id_cliente character varying (11)	nombres character varying (30)	apellido_paterno character varying (30)	apellido_materno character varying (30)	contacto character varying
1	PQREA235567	Luisa	Gómez	Ramírez	luisa.gr@example.com
2	FGHIJ789012	María	López	Rodríguez	mariiia@example.com
3	IJKCC901234	Mariana	López	Gómez	marian_asp@example.com
4	GHIII567890	Sara	Martínez	López	5555678901
5	KLMNO012345	Sofía	Torres	González	princesitasofia@example.com
6	ZABDE456789	Camila	González	López	5554567890
7	UVWXY345678	Luis	González	Torres	5557890123
8	KLMHH901234	Valeria	Fernández	López	valeria.lop.es@example.com
9	KLMNO234567	Carlos	Hernández	Gómez	5559876543

Figura 17: Primer vista

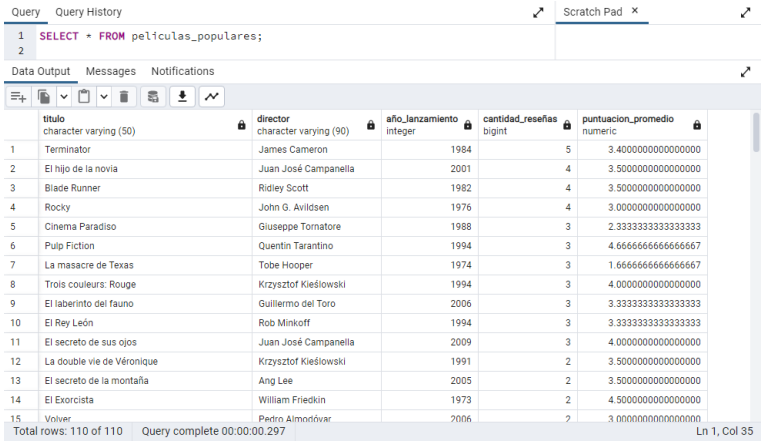


```
SELECT * FROM historial_pagos;
```

	id_cliente character varying (11)	cliente text	historial_pagos text
1	VWXXC234567	Javier Fernández Gutiérrez	2022-10-10 - 25.00000 - adeudo
2	EFGBC456789	Andrés Gómez Martínez	2022-08-18 - 25.00000 - adeudo
3	VWXWW345778	Valentina Torres Gómez	2022-10-18 - 25.00000 - adeudo
4	MNOYY890123	Diego Ramírez López	2022-09-09 - 25.00000 - adeudo; 2022-09-09 - 25.00000 - adeudo
5	PQRTG901234	Mariana López Gómez	2022-09-09 - 25.00000 - adeudo
6	PQRST678901	Miguel Ramírez López	2022-07-29 - 25.00000 - adeudo; 2022-07-29 - 25.00000 - adeudo
7	VWXNN234456	Carla Hernández Gómez	2022-09-23 - 25.00000 - adeudo
8	FGHIJ567890	Pedro Gómez Ramírez	2022-07-21 - 25.00000 - adeudo
9	YZAVV345678	Carolina Sánchez Martínez	2022-10-10 - 25.00000 - adeudo
10	ABCDE123456	Juan García Pérez	2022-06-09 - 25.00000 - adeudo; 2022-06-09 - 25.00000 - adeudo; 2022-06-09 - 25.00000 - adeudo
11	KLMNO012345	Sofía Torres González	2022-07-21 - 25.00000 - adeudo; 2022-07-21 - 25.00000 - adeudo; 2022-07-29 - 25.00000 - adeudo; 2022-07-
12	BCDBC456709	Diego Gómez González	2022-10-10 - 25.00000 - adeudo
13	CDEMM678901	Juan López González	2022-10-18 - 25.00000 - adeudo

Total rows: 60 of 60 Query complete 00:00:00.244 Ln 1, Col 31

Figura 18: Segunda vista



```
SELECT * FROM peliculas_populares;
```

	titulo character varying (50)	director character varying (90)	año_lanzamiento integer	cantidad_reseñas bigint	puntuacion_promedio numeric
1	Terminator	James Cameron	1984	5	3.4000000000000000
2	El hijo de la novia	Juan José Campanella	2001	4	3.5000000000000000
3	Blade Runner	Ridley Scott	1982	4	3.5000000000000000
4	Rocky	John G. Avildsen	1976	4	3.0000000000000000
5	Cinema Paradiso	Giuseppe Tornatore	1988	3	2.3333333333333333
6	Pulp Fiction	Quentin Tarantino	1994	3	4.6666666666666667
7	La masacre de Texas	Tobe Hooper	1974	3	1.6666666666666667
8	Trois couleurs: Rouge	Krzysztof Kieślowski	1994	3	4.0000000000000000
9	El laberinto del fauno	Guillermo del Toro	2006	3	3.3333333333333333
10	El Rey León	Rob Minkoff	1994	3	3.3333333333333333
11	El secreto de sus ojos	Juan José Campanella	2009	3	4.0000000000000000
12	La double vie de Véronique	Krzysztof Kieślowski	1991	2	3.5000000000000000
13	El secreto de la montaña	Ang Lee	2005	2	3.5000000000000000
14	El Exorcista	William Friedkin	1973	2	4.5000000000000000
15	Volver	Pedro Almodóvar	2006	2	3.0000000000000000

Total rows: 110 of 110 Query complete 00:00:00.297 Ln 1, Col 35

Figura 19: Tercer vista