

FUNDAMENTOS DE BASES DE DATOS

Semestre: 2024-1

Grupo 7081

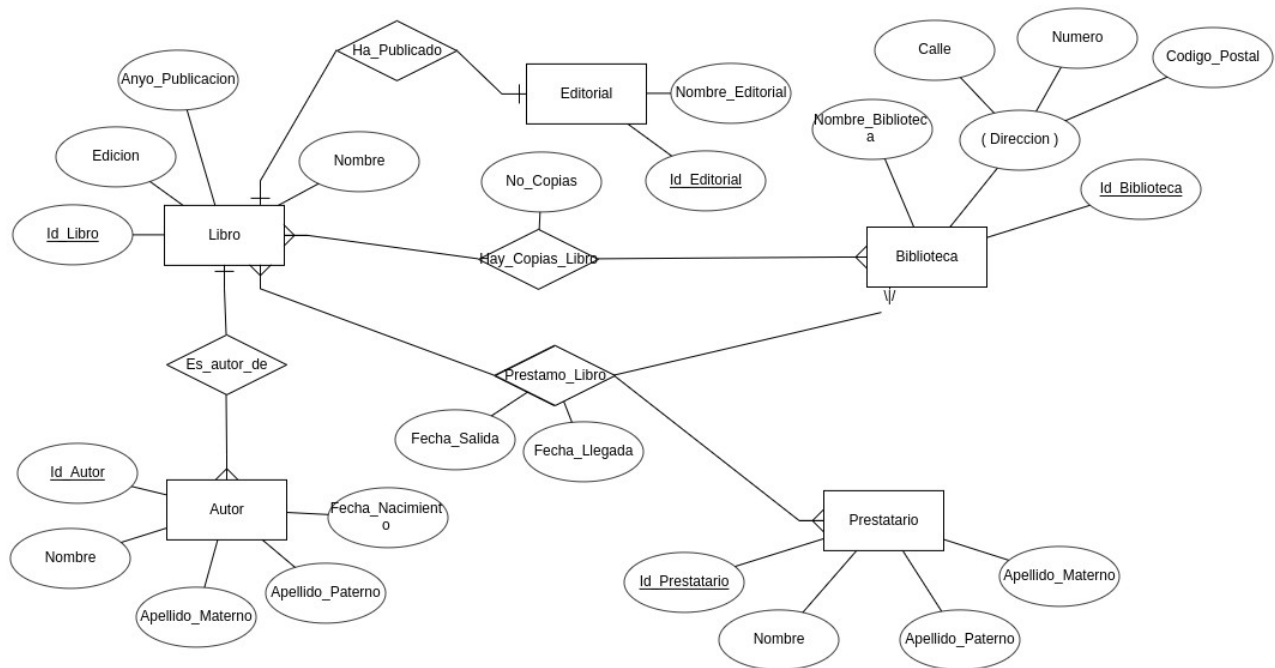
Profesor: Víctor Manuel Corza Vargas

Alumno : Rocha Salazar Mario Alberto

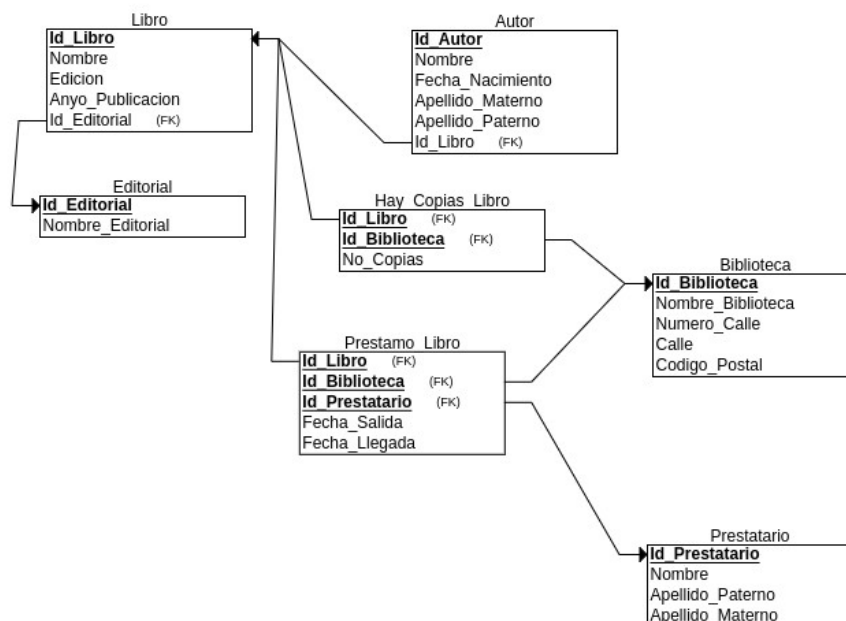
No de Cuenta: 319337112

Proyecto final: Base de Datos que admistra una cadena de Bibliotecas

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



2. Modelo relacional, deben incluirse: Llaves primarias y llaves foráneas



3. Script completo y sin errores para la creación de todos los elementos que conforman el esquema de la base de datos.

3.a El Script debe estar diseñado para la versión 14 de Postgres.

3.b Deben estar contempladas todas las llaves primarias, llaves candidatas y llaves foráneas; todas las llaves foráneas deben contar con un trigger de integridad referencial (SET NULL, CASCADE o SET DEFAULT).

```
CREATE TABLE Biblioteca
```

```
(  
  Nombre_Biblioteca VARCHAR(50) NOT NULL,  
  Numero_Calle INT NOT NULL,  
  Calle VARCHAR(50) NOT NULL,  
  Codigo_Postal INT NOT NULL,  
  Id_Biblioteca INT NOT NULL,  
  PRIMARY KEY (Id_Biblioteca)  
);
```

```
ALTER TABLE Biblioteca
```

```
ADD CONSTRAINT chk_Biblioteca_Id_Biblioteca_NonNegative  
CHECK (Id_Biblioteca >= 0);
```

```
ALTER TABLE Biblioteca
```

```
ADD CONSTRAINT Biblioteca_Numero_Calle_Positivo  
CHECK (Numero_Calle > 0);
```

```
-- Agrega la restricción corregida
```

```
ALTER TABLE Biblioteca
```

```
ADD CONSTRAINT Biblioteca_Codigo_Postal_5Digitos  
CHECK (CHAR_LENGTH(CAST(Codigo_Postal AS VARCHAR)) = 5);
```

```
CREATE TABLE Editorial
```

```
(  
  Nombre_Editorial VARCHAR(50) NOT NULL,  
  Id_Editorial INT NOT NULL,  
  PRIMARY KEY (Id_Editorial)  
);
```

```
CREATE TABLE Prestatario
```

```
(  
  Nombre VARCHAR(25) NOT NULL,  
  Apellido_Paterno VARCHAR(25) NOT NULL,  
  Apellido_Materno VARCHAR(25) NOT NULL,  
  Id_Prestatario INT NOT NULL,  
  PRIMARY KEY (Id_Prestatario)  
);
```

```
CREATE TABLE Libro
(
  Nombre VARCHAR(100) NOT NULL,
  Id_Libro INT NOT NULL,
  Edicion INT NOT NULL,
  Anyo_Publicacion DATE NOT NULL,
  Id_Editorial INT NOT NULL,
  PRIMARY KEY (Id_Libro),
  FOREIGN KEY (Id_Editorial) REFERENCES Editorial(Id_Editorial)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

```
ALTER TABLE Libro
ADD CONSTRAINT Libro_Id_Libro_NoNegative
CHECK (Id_Libro >= 0);
```

```
CREATE TABLE Autor
(
  Nombre VARCHAR(25) NOT NULL,
  Fecha_Nacimiento DATE NOT NULL,
  Apellido_Materno VARCHAR(25) NOT NULL,
  Apellido_Paterno VARCHAR(25) NOT NULL,
  Id_Autor INT NOT NULL,
  Id_Libro INT NOT NULL,
  PRIMARY KEY (Id_Autor),
  FOREIGN KEY (Id_Libro) REFERENCES Libro(Id_Libro)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

```
CREATE TABLE Hay_Copias_Libro
(
  No_Copias INT NOT NULL,
  Id_Libro INT NOT NULL,
  Id_Biblioteca INT NOT NULL,
  PRIMARY KEY (Id_Libro, Id_Biblioteca),
  FOREIGN KEY (Id_Libro) REFERENCES Libro(Id_Libro)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  FOREIGN KEY (Id_Biblioteca) REFERENCES Biblioteca(Id_Biblioteca)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

```
ALTER TABLE Hay_Copias_Libro
ADD CONSTRAINT No_Copias_NoNegativo
CHECK (No_Copias >= 0);
```

```

CREATE TABLE Prestamo_Libro
(
    Fecha_Salida DATE NOT NULL,
    Fecha_Llegada DATE,
    Id_Libro INT NOT NULL,
    Id_Biblioteca INT NOT NULL,
    Id_Prestatario INT NOT NULL,
    PRIMARY KEY (Id_Libro, Id_Biblioteca, Id_Prestatario),
    FOREIGN KEY (Id_Libro) REFERENCES Libro(Id_Libro)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (Id_Biblioteca) REFERENCES Biblioteca(Id_Biblioteca)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    FOREIGN KEY (Id_Prestatario) REFERENCES Prestatario(Id_Prestatario)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

```

4. Script con las Instrucciones “Insert” que permitan poblar la base de datos (Este Script no se debe incluir en el reporte final, sólo en los entregables)

5. Evidencia del funcionamiento de al menos 4 restricciones de integridad referencial.

Evidencia 1:

- Tablas involucradas en la restricción.: Biblioteca y Hay_Copias_Libro
- Justificación del trigger de integridad referencial elegido.
Si modificamos el id de biblioteca queremos seguir manteniendo los libros que hay en esta biblioteca a pesar de que el id sea distinto.
- FK de la tabla que referencia y PK de la tabla referenciada:
FK: Hay_Copias_Libro.Id_Biblioteca
PK: Biblioteca.Id_Biblioteca
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```

1  UPDATE Biblioteca
2  SET Id_Biblioteca = 987654
3  WHERE Id_Biblioteca = 900001;
4

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 223 msec.

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

```

1 SELECT * FROM Hay_Copias_Libro
2 WHERE Id_Biblioteca = 987654;
3

```

Data Output Messages Notifications			
	no_copias integer	Id_libro [PK] integer	Id_biblioteca [PK] integer
1	5	300001	987654
2	8	300002	987654
3	12	300003	987654
4	6	300004	987654
5	10	300005	987654
6	7	300006	987654
7	15	300007	987654
8	9	300008	987654
9	11	300009	987654
10	14	300010	987654

Evidencia 2:

- Tablas involucradas en la restricción.:
Editorial y Libro
- Justificación del trigger de integridad referencial elegido.
Si modificamos el id de la editorial queremos seguir manteniendo su referencia pertinente en cada libro que haya publicado esa editorial
- FK de la tabla que referencia y PK de la tabla referenciada:
FK: Libro.Id_Editorial
PK: Editorial.Id_Editorial
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```

1 UPDATE Editorial
2 SET Id_Editorial = 234567
3 WHERE Id_Editorial = 200001;
4

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 216 msec.

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

```

1 SELECT * FROM Libro
2 WHERE Id_Editorial = 234567;
3
4
5

```

Data Output

Messages

Notifications

☰+

📄

▼

📋






▼

🗑️

🗄️

⬇️

📈

	<div>nombre</div> <div>character varying (100) </div>	<div>id_libro</div> <div>[PK] integer </div>	<div>edicion</div> <div>integer </div>	<div>anyo_publicacion</div> <div>date </div>	<div>id_editorial</div> <div>integer </div>
1	Echoes of Eternity	300110	3	2022-04-08	234567
2	The Secret Garden	300001	1	2012-05-15	234567

Evidencia 3:

- Tablas involucradas en la restricción.:
Libro y Autor
- Justificación del trigger de integridad referencial elegido.
Aunque modifiquemos el ID del libro, se deba seguir conociendo quien o quienes son los autores de tal libro
- FK de la tabla que referencia y PK de la tabla referenciada:
FK: Autor.Id_Libro
PK: Libro.Id_Libro
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```

1 UPDATE Libro
2 SET Id_Libro = 322222
3 WHERE Id_Libro = 300002;
4

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 223 msec.

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

```

1  SELECT * FROM Autor
2  WHERE Id_Libro = 322222;
3
4
5

```

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑️

📦

⬇️

📈

	nombre character varying (25)	fecha_nacimiento date	apellido_materno character varying (25)	apellido_paterno character varying (25)	id_autor [PK] integer	id_libro integer
1	Isabel Rodriguez	1990-05-20	Santos	Ramirez	400002	322222
2	Isabel Nunez	1979-12-03	Cabrera	Salazar	400102	322222

Evidencia 4:

- Tablas involucradas en la restricción.: Prestatario y Prestamo_Libro
- Justificación del trigger de integridad referencial elegido.
Debemos de conocer quien fue el Prestatario de cada prestamo aunque el ID del Prestatario se haya modificado despues
- FK de la tabla que referencia y PK de la tabla referenciada:
FK: Prestamo_Libro.Id_Prestatario
PK: Prestatario.Id_Prestatario
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```

1  UPDATE Prestatario
2  SET Id_Prestatario = 111111
3  WHERE Id_Prestatario = 100001;
4

```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 215 msec.

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

```

1  SELECT * FROM Prestamo_Libro
2  WHERE Id_Prestatario = 111111;
3
4

```

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑

🗄

⬇

📈

	fecha_salida date	fecha_llegada date	id_libro [PK] integer	id_biblioteca [PK] integer	id_prestatario [PK] integer
1	2024-03-12	2024-03-31	300001	987654	111111

6. Evidencia del funcionamiento de al menos 3 restricciones check para “atributos” de varias tablas.

Evidencia 1:

- Tabla elegida
Hay_Copias_Libro
- Atributo elegido
No_Copias
- Breve descripción de la restricción
No debe permitirse un numero de copias negaitvo, debe ser mayor a 0
- Instrucción para la creación de la restricción.

```
ALTER TABLE Hay_Copias_Libro
ADD CONSTRAINT No_Copias_NoNegativo
CHECK (No_Copias >= 0);
```

- Instrucción que permita evidenciar que la restricción esta funcionando.

```
INSERT INTO Hay_Copias_Libro (No_Copias,
Id_Libro, Id_Biblioteca)
VALUES (-5, 1, 1);
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

Data Output	Messages	Notifications
ERROR: Failing row contains (-5, 1, 1).new row for relation "hay_copias_libro" violates check constraint "no_copias_nonnegativo"		
ERROR: new row for relation "hay_copias_libro" violates check constraint "no_copias_nonnegativo" SQL state: 23514 Detail: Failing row contains (-5, 1, 1).		

Evidencia 2:

- Tabla elegida
Biblioteca
- Atributo elegido
Codigo_Postal
- Breve descripción de la restricción

El código postal solo puede tener una longitud de 5 dígitos.

- Instrucción para la creación de la restricción.

```
ALTER TABLE Biblioteca
ADD CONSTRAINT Biblioteca_Codigo_Postal_5Digitos
CHECK (CHAR_LENGTH(CAST(Codigo_Postal AS VARCHAR)) = 5);
```

- Instrucción que permita evidenciar que la restricción está funcionando.

```
INSERT INTO Biblioteca (Nombre_Biblioteca, Numero_Calle, Calle, Codigo_Postal,
Id_Biblioteca)
VALUES ('Biblioteca Ejemplo', 123, 'Calle Principal', 123456, 1);
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

Data Output Messages Notifications

ERROR: Failing row contains (Biblioteca Ejemplo, 123, Calle Principal, 123456, 1).new row for relation "biblioteca" violates check constraint "biblioteca_codigo_postal_5digitos"

ERROR: new row for relation "biblioteca" violates check constraint "biblioteca_codigo_postal_5digitos"

SQL state: 23514

Detail: Failing row contains (Biblioteca Ejemplo, 123, Calle Principal, 123456, 1).

Evidencia 3:

- Tabla elegida
Libro
- Atributo elegido
Id_Libro
- Breve descripción de la restricción
El Id de un libro solo puede ser una secuencia de números positivos
- Instrucción para la creación de la restricción.

```
ALTER TABLE Libro
ADD CONSTRAINT Libro_Id_Libro_NoNegative
CHECK (Id_Libro >= 0);
```

- Instrucción que permita evidenciar que la restricción está funcionando.

```
INSERT INTO Libro (Nombre, Id_Libro, Edicion, Anyo_Publicacion, Id_Editorial)
VALUES ('Libro Ejemplo', -1, 1, '2023-01-01', 1);
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

ERROR: Failing row contains (Libro Ejemplo, -1, 1, 2023-01-01, 1).new row for relation "libro" violates check constraint "libro_id_libro_nonnegative"

ERROR: new row for relation "libro" violates check constraint "libro_id_libro_nonnegative"

SQL state: 23514

Detail: Failing row contains (Libro Ejemplo, -1, 1, 2023-01-01, 1).

7. Evidencia de la creación de al menos tres dominios personalizados

Evidencia 1:

- Tabla elegida
Biblioteca
- Atributo elegido
Nombre_Biblioteca
- Breve descripción del dominio y de la restricción check propuesta.
El nombre de la biblioteca debe de ser no nulo y debe de ser menor a 100 caracteres
- Instrucción para la creación del dominio personalizado.

```
CREATE DOMAIN Nombre_Biblioteca_Type AS VARCHAR(50)
CHECK (LENGTH(VALUE) <= 50 AND VALUE IS NOT NULL);
```

```
ALTER TABLE Biblioteca
```

```
ALTER COLUMN Nombre_Biblioteca TYPE Nombre_Biblioteca_Type;
```

- Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.

```
1 INSERT INTO Biblioteca (Nombre_Biblioteca, Numero_Calle, Calle,Codigo_Postal, Id_Biblioteca)
2 VALUES (NULL, 101, 'Calle Ejemplo', 12345, 1);
3
```

ERROR: value for domain nombre_biblioteca_type violates check constraint "nombre_biblioteca_type_check"

SQL state: 23514

Evidencia 2:

- Tabla elegida
Hay_Copias_Libro
- Atributo elegido
No_Copias
- Breve descripción del dominio y de la restricción check propuesta.
El numero de copias debe ser mayor a 0.
- Instrucción para la creación del dominio personalizado.

```
CREATE DOMAIN No_Copias_Type AS INT
CHECK (VALUE >= 0);
```

```
ALTER TABLE Hay_Copias_Libro
```

```
ALTER COLUMN No_Copias TYPE No_Copias_Type;
```

- Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.

```
1 INSERT INTO Hay_Copias_Libro (No_Copias, Id_Libro, Id_Biblioteca)
2 VALUES (-5, 1, 1);
3
```

Data Output Messages Notifications

ERROR: value for domain no_copias_type violates check constraint "no_copias_type_check"

SQL state: 23514

Evidencia 3:

- Tabla elegida
Libro
- Atributo elegido
Nombre_Libro
- Breve descripción del dominio y de la restricción check propuesta.
El nombre del libro no puede ser nulo ni tener mas de 100 caracteres
- Instrucción para la creación del dominio personalizado.

```
CREATE DOMAIN Nombre_Libro_Type AS VARCHAR(100)
CHECK (LENGTH(VALUE) <= 100 AND VALUE IS NOT NULL);
```

```
ALTER TABLE Libro
ALTER COLUMN Nombre TYPE Nombre_Libro_Type;
```

- Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.

```
1 INSERT INTO Libro (Nombre, Id_Libro, Edicion, Anyo_Publicacion, Id_Editorial)
2 VALUES (NULL, 1, 1, '2023-01-01', 1);
3
```

ERROR: value for domain nombre_libro_type violates check constraint "nombre_libro_type_check"

SQL state: 23514

8. Evidencia del funcionamiento de al menos 2 restricciones para “tuplas” en diferentes tablas

Evidencia 1:

- Tabla elegida
Biblioteca
- Breve descripción de la restricción.
El nombre de ninguna Biblioteca debe exceder los 50 caracteres.
- Instrucción para la creación de la restricción

```
CREATE TABLE Biblioteca
(
  Nombre_Biblioteca VARCHAR(50) NOT NULL,
  Numero_Calle INT NOT NULL,
  Calle VARCHAR(50) NOT NULL,
  Codigo_Postal INT NOT NULL,
  Id_Biblioteca INT NOT NULL,
  PRIMARY KEY (Id_Biblioteca)
);
```

- Instrucción “Insert” o “Update” que permita evidenciar que la restricción esta funcionando.

```
INSERT INTO Biblioteca (Nombre_Biblioteca, Numero_Calle, Calle, Codigo_Postal,
Id_Biblioteca)
VALUES ('Biblioteca con un nombre extremadamente largo que excede los 50 caracteres
permitidos', 123, 'Calle Ejemplo', 12345, 1);
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

ERROR: value too long for type character varying(50)

SQL state: 22001

Evidencia 2:

- Tabla elegida
Libro
- Breve descripción de la restricción.
Ningun dato de un Libro puede ser NULL

- Instrucción para la creación de la restricción

```
CREATE TABLE Libro
(
  Nombre VARCHAR(100) NOT NULL,
  Id_Libro INT NOT NULL,
  Edicion INT NOT NULL,
  Anyo_Publicacion DATE NOT NULL,
  Id_Editorial INT NOT NULL,
  PRIMARY KEY (Id_Libro),
  FOREIGN KEY (Id_Editorial) REFERENCES
  Editorial(Id_Editorial)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
```

- Instrucción “Insert” o “Update” que permita evidenciar que la restricción esta funcionando.

```
INSERT INTO Libro (Nombre, Id_Libro, Edicion, Anyo_Publicacion, Id_Editorial)
VALUES (NULL, 1, 1, '2023-01-01', 1);
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

Data Output Messages Notifications

ERROR: Failing row contains (null, 1, 1, 2023-01-01, 1).null value in column "nombre" of relation "libro" violates not-null constraint

ERROR: null value in column "nombre" of relation "libro" violates not-null constraint
SQL state: 23502

Detail: Failing row contains (null, 1, 1, 2023-01-01, 1).

9. Consultas: Plantea 3 consultas que consideres relevantes para la base de datos propuesta.

Evidencia 1:

- Redacción clara de la consulta.
Recuperar el nombre del libro y la fecha de salida de todos los libros prestados en la biblioteca ubicada en ‘Calle del Saber’, además se requiere recuperar el nombre completo del Prestatario que realizo el prestamo
- Código en lenguaje SQL de la consulta.Código en lenguaje SQL de la consulta.

```
SELECT Libro.Nombre AS Nombre_Libro,
       Prestamo_Libro.Fecha_Salida,
       CONCAT(Prestatario.Nombre, ' ', Prestatario.Apellido_Paterno, ' ',
       Prestatario.Apellido_Materno) AS Nombre_Prestatario
FROM Prestamo_Libro
JOIN Libro ON Prestamo_Libro.Id_Libro = Libro.Id_Libro
JOIN Biblioteca ON Prestamo_Libro.Id_Biblioteca = Biblioteca.Id_Biblioteca
JOIN Prestatario ON Prestamo_Libro.Id_Prestatario = Prestatario.Id_Prestatario
WHERE Biblioteca.Calle = 'Calle del Saber';
```

- Ejecutar la consulta en Postgres e incluir una captura de pantalla con el resultado de la consulta.

Data Output

Messages

Notifications

≡+

▼

▼

	<div>nombre_libro</div> <div>character varying (100)</div> <div>🔒</div>	<div>fecha_salida</div> <div>date</div> <div>🔒</div>	<div>nombre_prestatario</div> <div>text</div> <div>🔒</div>
1	Susurros del Alma	2024-05-01	Lucia Gutierrez Molina
2	Crónicas Mágicas	2024-05-02	Gustavo Silva Garcia
3	Llamarada Eterna	2024-05-03	Ana Martinez Lopez
4	Sueños Desvelados	2024-05-04	Hugo Nunez Ramirez
5	Ecos de Otro Mundo	2024-05-05	Elena Jimenez Gomez
6	Viaje a lo Desconocido	2024-05-06	Andres Rodriguez Herrera
7	Cuentos de Encantamie...	2024-05-07	Karina Lopez Santana
8	Reinos Encantados	2024-05-08	Juan Flores Gutierrez
9	Maravillas de los Sueños	2024-05-09	Marcelo Nunez Vargas
10	Aventuras Místicas	2024-05-10	Renata Rojas Cruz

- Evidencia 2:**
- Redacción clara de la consulta.
Numero de copias del libro titulado ‘Whispers of the Stars’ que pertenecen a la biblioteca cuyo nombre es ‘Biblioteca Central’
 - Código en lenguaje SQL de la consulta.Código en lenguaje SQL de la consulta.

```

SELECT Hay_Copias_Libro.No_Copias
FROM Hay_Copias_Libro
JOIN Libro ON Hay_Copias_Libro.Id_Libro = Libro.Id_Libro
JOIN Biblioteca ON Hay_Copias_Libro.Id_Biblioteca = Biblioteca.Id_Biblioteca
WHERE Libro.Nombre = 'Whispers of the Stars'
AND Biblioteca.Nombre_Biblioteca = 'Biblioteca Central';

```

- Ejecutar la consulta en Postgres e incluir una captura de pantalla con el resultado de la consulta.

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

no_copias

integer

🔒

1

6

Evidencia 3:

- Redacción clara de la consulta.
Recuperar el numero de copias disponibles del libro llamado ‘Ecos del Pasado’, así como la edicion del libro y la direccion de las bibliotecas que lo tengan disponible.
- Código en lenguaje SQL de la consulta.Código en lenguaje SQL de la consulta.

```
SELECT
Hay_Copias_Libro.No_Copias AS Numero_de_Copias,
Libro.Edicion AS Numero_de_Edicion,
Biblioteca.Nombre_Biblioteca,
Biblioteca.Numero_Calle,
Biblioteca.Calle,
Biblioteca.Codigo_Postal
FROM
Libro
JOIN
Hay_Copias_Libro ON Libro.Id_Libro = Hay_Copias_Libro.Id_Libro
JOIN
Biblioteca ON Hay_Copias_Libro.Id_Biblioteca = Biblioteca.Id_Biblioteca
WHERE
Libro.Nombre = 'Ecos del Pasado';
```

- Ejecutar la consulta en Postgres e incluir una captura de pantalla con el resultado de la consulta.

Data Output

Messages

Notifications

	numero_de_copias integer	numero_de_edicion integer	nombre_biblioteca character varying (50)	numero_calle integer	calle character varying (50)	codigo_postal integer
1	9	1	El Refugio del Libro	101	Calle Imaginaria	56789
2	8	1	Biblioteca Infinita	606	Avenida de los Libros	24680

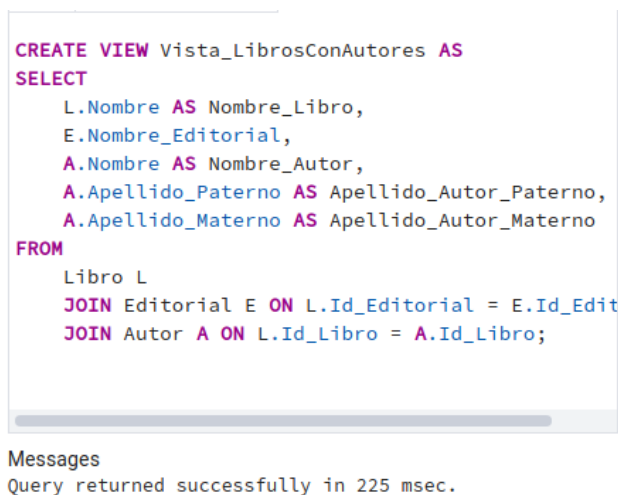
10. Vistas: Plantea 3 vistas que consideres relevantes para la base de datos propuesta.

Evidencia 1:

- Redacción clara de la vista planteada.
Basado en el Nombre del libro, obtener todos los datos relacionados con el que no se almacenan directamente en Libro, como son el nombre de la Editorial/es y el nombre del autor/es.
- Código en lenguaje SQL que permita crear la vista solicitada.

```
CREATE VIEW Vista_LibrosConAutores AS
SELECT
    L.Nombre AS Nombre_Libro,
    E.Nombre_Editorial,
    A.Nombre AS Nombre_Autor,
    A.Apellido_Paterno AS Apellido_Autor_Paterno,
    A.Apellido_Materno AS Apellido_Autor_Materno
FROM
    Libro L
    JOIN Editorial E ON L.Id_Editorial = E.Id_Editorial
    JOIN Autor A ON L.Id_Libro = A.Id_Libro;
```

- Ejecutar el código para la creación de la vista en Postgres e incluir una captura de pantalla con la vista creada satisfactoriamente.



```
CREATE VIEW Vista_LibrosConAutores AS
SELECT
    L.Nombre AS Nombre_Libro,
    E.Nombre_Editorial,
    A.Nombre AS Nombre_Autor,
    A.Apellido_Paterno AS Apellido_Autor_Paterno,
    A.Apellido_Materno AS Apellido_Autor_Materno
FROM
    Libro L
    JOIN Editorial E ON L.Id_Editorial = E.Id_Edit
    JOIN Autor A ON L.Id_Libro = A.Id_Libro;
```

Messages
Query returned successfully in 225 msec.

- Incluir un ejemplo que los evaluadores puedan ejecutar para verificar el funcionamiento de las vistas.

```
SELECT * FROM Vista_LibrosConAutores
WHERE Nombre_Libro = 'Legends of the Sea';
```

Evidencia 2:

- Redacción clara de la vista planteada.
Dirección de la biblioteca de un libro o varios en específico donde se encuentre disponible para préstamo.
- Código en lenguaje SQL que permita crear la vista solicitada.

```
CREATE VIEW Vista_LibrosDisponibles AS
SELECT L.Nombre AS Nombre_Libro, B.Nombre_Biblioteca, B.Calle, B.Codigo_Postal
FROM Libro L
    JOIN Hay_Copias_Libro HCL ON L.Id_Libro = HCL.Id_Libro
    JOIN Biblioteca B ON HCL.Id_Biblioteca = B.Id_Biblioteca
WHERE HCL.No_Copias > 0;
```

- Ejecutar el código para la creación de la vista en Postgres e incluir una captura de pantalla

con la vista creada satisfactoriamente.

```
CREATE VIEW Vista_LibrosDisponibles AS
SELECT L.Nombre AS Nombre_Libro, B.Nombre_Biblioteca
FROM Libro L
JOIN Hay_Copias_Libro HCL ON L.Id_Libro = HCL.Id_Lib
JOIN Biblioteca B ON HCL.Id_Biblioteca = B.Id_Biblioteca
WHERE HCL.No_Copias > 0;
```

Messages
Query returned successfully in 232 msec.

- Incluir un ejemplo que los evaluadores puedan ejecutar para verificar el funcionamiento de las vistas.

```
SELECT * FROM Vista_LibrosDisponibles
WHERE Nombre_Libro = 'Whispers in the Wind';
```

- **Evidencia 3:**

- Redacción clara de la vista planteada.

Vista que maneje los préstamos realizados en fechas específicas, esta contendrá el nombre del libro, el nombre de la biblioteca y el nombre del prestatario y las fechas.

- Código en lenguaje SQL que permita crear la vista solicitada.

```
CREATE VIEW Vista_PrestamosPrestatario AS
SELECT
    PL.Fecha_Salida,
    PL.Fecha_Llegada,
    L.Nombre AS Nombre_Libro,
    B.Nombre_Biblioteca,
    P.Nombre AS Nombre_Prestatario
FROM Prestamo_Libro PL
JOIN Libro L ON PL.Id_Libro = L.Id_Libro
JOIN Biblioteca B ON PL.Id_Biblioteca = B.Id_Biblioteca
JOIN Prestatario P ON PL.Id_Prestatario = P.Id_Prestatario;
```

- Ejecutar el código para la creación de la vista en Postgres e incluir una captura de pantalla con la vista creada satisfactoriamente.

```
SELECT *
FROM Vista_PrestamosPrestatario
WHERE Fecha_Salida >= '2024-01-01' AND Fecha_Salida
```

Messages
Successfully run. Total query runtime: 894 msec. 20 rows affected.

- Incluir un ejemplo que los evaluadores puedan ejecutar para verificar el funcionamiento de las vistas.

```
SELECT *
FROM Vista_PrestamosPrestatario
WHERE Fecha_Salida >= '2024-01-01' AND Fecha_Salida <= '2024-03-31';
```