



UNIVERSIDAD NACIONAL SEDE REGIONAL CHOROTEGA,

CAMPUS NICOYA

ESCUELA DE INFORMÁTICA

Curso:

Diseño e Implementación de Base de Datos

Proyecto Final

Mi cine es tu cine

Profesor:

Francisco Coulon Ollivier

Estudiantes:

John Granados Rojas

Axel Castillo Zúñiga

Brenda Paola Espinoza Matarrita

I Ciclo, 2025

Hoja de Evaluación

Evaluación	Exposición
%	
15 ptos	Dominio del tema y fluidez (<i>debe demostrar que domina el contexto de la exposición</i>).
10 ptos	Presentación Personal (vestimenta Formal)
15 ptos	Desenvolvimiento, dominio del escenario, dicción y elocuencia
10 ptos	Uso <i>adecuado</i> de los medios audiovisuales (multimedia).
50 ptos	Cumplimiento del proyecto
100 puntos	TOTAL:
5%	Porcentaje Total:
Parte de desarrollo operativo o programada	
25 ptos	Diagrama Contextual
15 ptos	Diagrama relacional
15 ptos	Normalización de cada tabla diseñada
43 ptos	Implementación base datos, tablas, inserción de datos, consulta y procedimiento
2 ptos	Diccionario
100 puntos	TOTAL:
20%	Porcentaje total:
Parte Escrita	
2 ptos	Portada y hoja de evaluación

2 ptos	Tabla de contenido
4 ptos	Introducción
2 ptos	Planteamiento del problema:
4 ptos	Justificación
10 ptos	Objetivos (General y Específicos)
13 ptos	Resultados y su discusión
10 ptos	Conclusiones
13 ptos	Recomendaciones
3 ptos	Referencias (Normas APA)
10 ptos	Anexos
7 ptos	Bitácora

Aspectos de Estilo, Forma y Fondo

5 ptos	Redacción
5 ptos	Ortografía
10 ptos	Encadenamiento entre: Objetivos - Metodología - Resultados y Conclusiones
100 puntos	TOTAL
5%	Porcentaje total: 30%

Tabla de contenido

<i>Introducción</i>	5
<i>Planteamiento del Problema</i>	6
<i>Justificación</i>	7
<i>Objetivos</i>	8
<i>Descripción de su empresa (Nombre, que hacen y cómo visualizan este proyecto)</i>	9
<i>Resultados y su discusión</i>	10
<i>Modelo entidad relación</i>	12
<i>Modelo relacional</i>	13
<i>Normalización</i>	13
<i>Estrategia de auditoría a utilizar</i>	14
<i>Tecnologías utilizadas</i>	15
<i>Conclusiones</i>	15
<i>Recomendaciones</i>	16
<i>Referencia</i>	17
<i>Anexos</i>	17
<i>Consideraciones para una versión final/Siguientes pasos</i>	24
<i>Bitácoras Base de datos</i>	25

Introducción

Este proyecto tiene como objetivo crear e implementar una base de datos que facilite la gestión integral del cine “Mi Cine es Tu Cine”. Ya que contar con sistema de base de datos en la empresa Mi cine es Tu Cine, le facilitará organizar y almacenar información de manera más eficiente, facilitando su acceso, el uso y gestión, además ayudará a centralizar la información, evitando duplicaciones y mejorando la toma de decisiones esto basándose en datos actualizados y precisos.

Actualmente, “Mi Cine es Tu Cine” enfrenta diversas limitaciones al no contar con un sistema digital. La cartelera se administra de forma manual, las entradas se venden exclusivamente en boletería y los registros se anotan en libretas físicas. Este modelo genera ineficiencias, dificulta el control operativo y limita la capacidad de expansión y mejora continua, por ende, el desarrollo de este proyecto surge como respuesta a las dificultades que enfrenta actualmente la empresa “Mi Cine es Tu Cine”.

La idea es automatizar procesos clave como la cartelera, la venta de entradas y el registro de usuarios, en este contexto, se propone el desarrollo de un sistema de base de datos que permita centralizar, organizar y acceder a la información de forma precisa y segura. La digitalización de estos procesos permitirá reducir errores, evitar duplicaciones y mejorar la toma de decisiones a partir de datos actualizados.

Además, se busca que este sistema no solo cubra las necesidades actuales, sino que también sea escalable y adaptable a futuras mejoras tecnológicas, por ello, el sistema ha sido diseñado con enfoque escalable, adaptable a nuevas funcionalidades y conforme a principios de normalización y buenas prácticas en diseño relacional.

Para hacer posible este sistema, se utiliza el motor de base de datos SQL Server, además de la creación del modelo de entidad relación y el modelo relacional, así como procedimientos almacenados, validaciones y mecanismos de auditoría, entre otros.

Planteamiento del Problema

Actualmente, la empresa Mi Cine es Tu Cine administra la programación de funciones, la cartelera, la venta de entradas y el seguimiento de clientes mediante procesos completamente manuales. Las entradas se venden únicamente en la boletería física y se registran en libretas, lo que representa un alto riesgo operativo. Aunque se trata de una empresa nueva con múltiples sedes en Costa Rica, la falta de un sistema digital limita severamente su capacidad de expansión, automatización y adaptación a las nuevas exigencias del mercado, lo que la vuelve menos competitiva a comparación a otras empresas.

Este enfoque manual conlleva diversos problemas, entre ellos: errores en la asignación de asientos, duplicidad de funciones en horarios coincidentes, falta de trazabilidad sobre las ventas, imposibilidad de ofrecer múltiples formas de pago, y nula gestión de clientes o historial de compras. Además, la ausencia de una plataforma digital limita la posibilidad de vender entradas en línea, consultar la disponibilidad de asientos en tiempo real.

La empresa tampoco cuenta con un sistema capaz de registrar múltiples idiomas y subtítulos por película, asignar categorías dinámicas, asociar múltiples actores a una producción o administrar funciones simultáneas en distintas salas. Asimismo, la planificación de la cartelera se realiza mensualmente, sin posibilidad de hacer ajustes diarios.

También desde el punto de vista del cliente, la experiencia también se ve limitada, ya que no puede consultar la cartelera en tiempo real, seleccionar asientos, comprar entradas por internet. Esto representa una desventaja competitiva frente a cines que ya ofrecen estas funcionalidades mediante sistemas digitales.

En consecuencia, la empresa necesita con urgencia una solución técnica que permita centralizar la información, automatizar los procesos clave, mejorar la atención al cliente, normalizar y validar sus operaciones mediante una base de datos. El presente proyecto propone el diseño e implementación de dicho sistema, utilizando el motor de bases de datos SQL Server, con una estructura optimizada que contemple relaciones uno a muchos, muchos a muchos, claves primarias y foráneas, validaciones, y soporte para auditoría. Esta solución no solo resolverá los problemas actuales, sino que también servirá como base sólida para futuras integraciones con plataformas web.

Justificación

La solución propuesta para la empresa Mi Cine es Tu Cine, permitirá mejorar la gestión interna de funciones, disponibilidad de asientos, ventas, y trazabilidad mediante auditoría.

Asimismo, la implementación de un sistema de base de datos relacional en Mi Cine es Tu Cine es crucial para resolver las deficiencias actuales en la gestión de cartelera, funciones, boletos y clientes. La estructura actual, basada en registros físicos, imposibilita una administración eficiente, incrementa el margen de error humano y limita significativamente la escalabilidad de la empresa.

Por ende, la solución propuesta, diseñada e implementada en SQL Server, se compone de 20 tablas relacionales interconectadas, que permiten una representación fiel de las operaciones de la empresa. Incluye el manejo de películas, idiomas, subtítulos, actores, categorías, funciones, salas, asientos, clientes, métodos de pago, puntos de fidelidad, compras y tiquetes. Uno de los principales aportes del sistema es la automatización del ciclo de compra de boletos, integrando funciones como:

- Selección dinámica de asientos disponibles
- Registro y validación de métodos de pago por cliente

- Acumulación y canje de puntos
- Gestión flexible de carteleras y funciones por sala y fecha
- Control de entradas tanto para ventas físicas como en línea.

También, el modelo relacional incorpora relaciones de tipo uno a muchos y muchos a muchos (por ejemplo, películas con múltiples actores, categorías o funciones), optimizando la estructura para soportar consultas eficientes y evitar duplicidad de datos mediante normalización.

Este sistema no solo atiende las necesidades actuales de digitalización, sino que también se ha construido con una visión de futuro.

En resumen, la base de datos diseñada constituye el eje estructural sobre el cual Mi Cine es Tu Cine puede consolidar su operación actual, mejorar la toma de decisiones mediante datos trazables, y escalar su negocio hacia un modelo competitivo y moderno.

Objetivos

Objetivo General

El presente proyecto tiene como objetivo el diseño y desarrollo de una base de datos relacional en SQL Server, con el fin de automatizar el proceso de gestión de cartelera, venta de entradas y registro de usuarios de la empresa Mi Cine es Tu Cine.

Objetivos Específicos

- Realizar un modelo de entidad relación del proyecto mi cine es tu cine.
- Diseñar e implementar la base de datos relacional.
- Crear procedimientos almacenados que optimicen las operaciones.
- Establecer controles de integridad referencial y auditoría.
- Permitir consultas útiles para la operación y toma de decisiones.

- Realizar triggers y backup de la base de datos creada.

Descripción de su empresa (Nombre, que hacen y cómo visualizan este proyecto)

El nombre de la empresa es “Mi Cine es Tu Cine.

Mi Cine es Tu Cine es una cadena de cines costarricense con presencia en centros comerciales y malls es una empresa del sector entretenimiento, su actividad principal es la proyección de películas nacionales e internacionales mediante funciones programadas, y la venta de entradas directamente en boletería. Como parte de su visión a futuro, la empresa busca evolucionar hacia un modelo de operación digital que le permita competir en el mercado actual, mejorar la experiencia del cliente y optimizar la administración interna, mediante la creación de un sistema.

A pesar de ser una empresa nueva, Mi Cine es Tu Cine se ha identificado que la digitalización de sus procesos es un paso indispensable para el crecimiento y consolidación.

Actualmente, todos los registros de funciones, ventas, clientes y cartelera se manejan manualmente, lo cual genera problemas de trazabilidad, eficiencia y control en dicha empresa.

Por ende, la empresa ha impulsado la creación de un sistema de base de datos que centralice y automatice la gestión de funciones, salas, boletos, clientes, puntos, métodos de pago y auditoría. Este sistema ha sido diseñado e implementado utilizando SQL Server, haciendo uso de:

- Vistas que permiten observar en tiempo real datos como: asientos disponibles, películas en cartelera, películas más vendidas, puntos de clientes, y detalles completos de compras.

- Procedimientos almacenados que controlan operaciones críticas como registrar compras, gestionar películas y consultar disponibilidad.
- Triggers que automatizan tareas como reservas de asientos, registros en la auditoría y sumar puntos al cliente según el monto pagado en cada boleto, entre otros.
- Auditoría estructurada para asegurar transparencia y trazabilidad.
- Perfiles de usuario y permisos mediante logins creados con distintos niveles de acceso (administradores y usuarios normales).

En contexto, este proyecto se visualiza como la base estructural para una futura plataforma web o aplicación móvil, desde la cual los clientes podrán realizar sus compras en línea, acumular puntos, consultar funciones y recibir recomendaciones personalizadas.

Asimismo, se asume este proyecto como una prueba de concepto integral, abordando desde el diseño del modelo Entidad-Relación hasta la implementación de mecanismos de seguridad, control y acceso, bajo principios de escalabilidad, normalización y eficiencia en consultas.

Con esta solución, la empresa Mi Cine es Tu Cine no solo busca resolver sus limitaciones actuales, sino también transformarse en una empresa preparada para afrontar los desafíos tecnológicos del sector cinematográfico en el corto y mediano plazo y como una idea de negocio.

Resultados y su discusión

Se crearon las 20 tablas necesarias que cubren películas, funciones, boletos, clientes, salas, asientos, puntos, auditoría y más. Estas tablas incorporan claves primarias, foráneas y restricciones (CHECK, NOT NULL, UNIQUE) que aseguran la integridad y consistencia de los datos.

Asimismo, se implementaron 6 procedimientos almacenados claves:

- Registro de compra y asignación de asiento: permite registrar la compra de un asiento específico, vinculando cliente, método de pago, función y asiento.
- Listado de películas en cartelera: lista de películas activas según la fecha actual.
- Películas más vendidas: consulta de negocio para identificar títulos con mayor número de ventas.
- Asientos disponibles por función: devuelve solo los asientos disponibles por función.
- Bloqueo de asiento temporal: reserva asientos sin necesidad de compra inmediata.
- Crear o editar una película: gestiona alta y actualización de películas en un solo procedimiento.

También se desarrollaron más de 10 vistas (views) que permiten generar reportes compuestos y visualizar información relevante sin exponer directamente las tablas. Algunas de las vistas más importantes son:

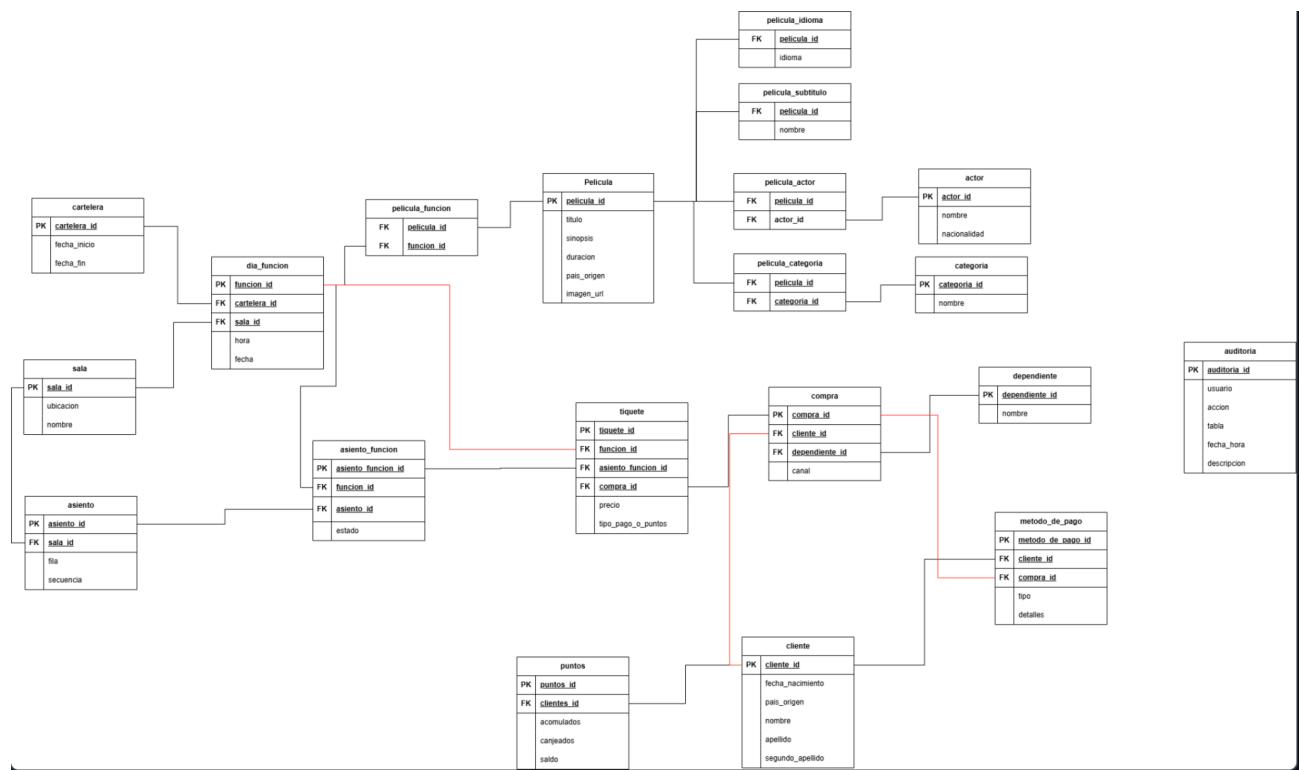
- vw_peliculas_cartelera: muestra cartelera con salas, horarios e información de películas.
- vw_asiento_disponibles: filtra y devuelve asientos marcados como “libres”.
- vw_peliculas_mas_vendidas: útil para marketing y decisiones comerciales.
- vw_compra_todo: consulta consolidada que muestra la trazabilidad completa de una compra (cliente, asiento, función, película, idioma, subtítulo, método de pago, dependiente, etc.).

Para reforzar la automatización y trazabilidad, se integraron triggers que:

- Reservan automáticamente un asiento al registrar un tiquete (trg_reservar_asiento).
- Registran eventos de auditoría al insertar una nueva película o eliminar funciones (trg_auditoria_pelicula_insert, trg_auditoria_funcion_delete).

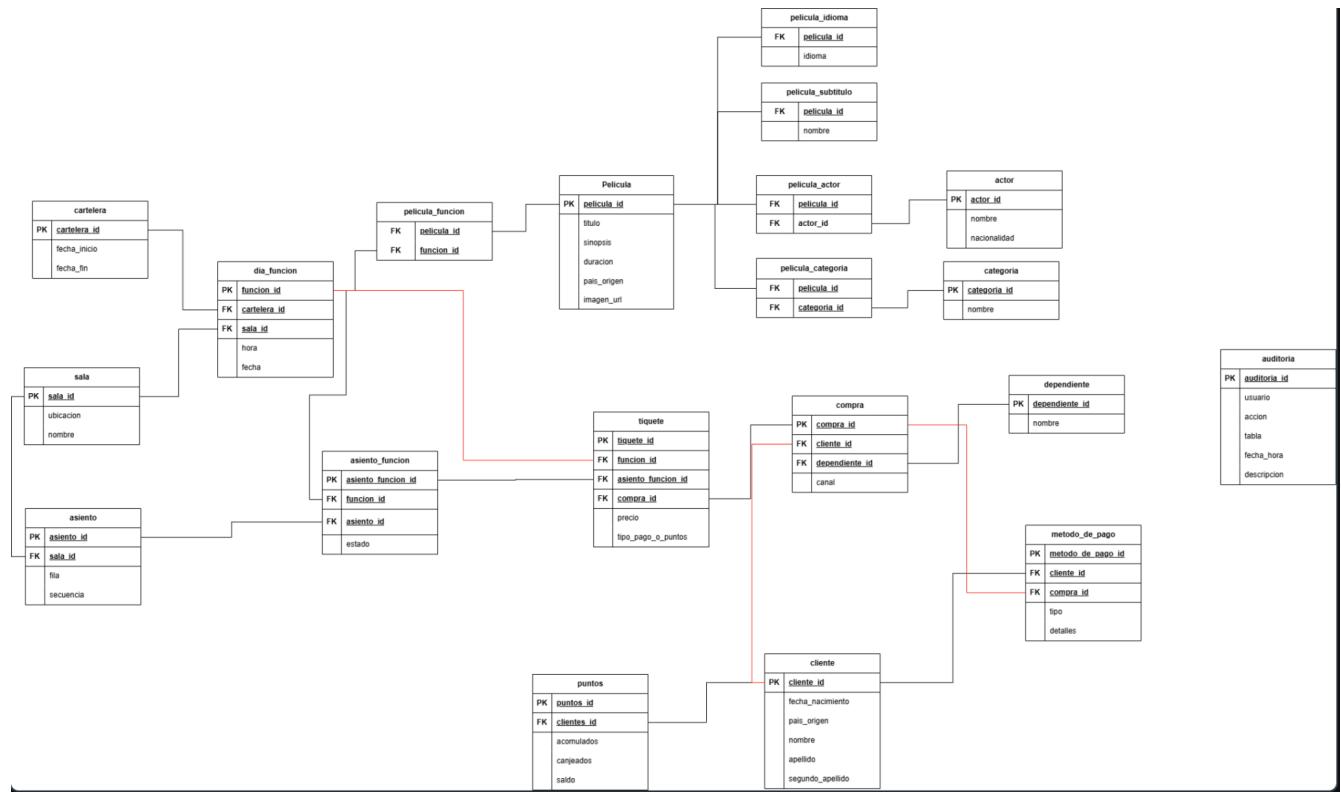
Además, se implementaron roles de usuarios y control de permisos en SQL Server, lo que permitió asignar distintos niveles de acceso según el perfil del usuario. Por ejemplo, usuario_normal puede visualizar ciertas vistas mediante GRANT SELECT, mientras que los usuarios administradores (martillo, francisco_coulon) tienen control total (db_owner). Y por último se implementa un backup, que es una copia de seguridad de la base de datos, esto para proteger en caso de pérdida o corrupción.

Modelo entidad relación



Fuente: Elaboración Propia

Modelo relacional



Fuente: Elaboración Propia

Normalización

La base de datos se encuentra normalizada al menos hasta la Tercera Forma Normal (3FN), garantizando la eliminación de redundancias y dependencias funcionales innecesarias.

Primera Forma Normal (1FN):

Todas las tablas contienen valores atómicos (no repetitivos ni multivaluados). Por ejemplo, en la tabla **pelicula_subtitulo**, cada registro contiene un solo subtítulo por película.

Segunda Forma Normal (2FN):

En tablas con claves primarias compuestas como **pelicula_actor** y **pelicula_categoria**, todos los atributos dependen por completo de la clave compuesta y no parcialmente de uno solo de los campos.

Tercera Forma Normal (3FN):

Se eliminaron dependencias transitivas. Por ejemplo, en la tabla cliente, los atributos nombre_cliente, apellido y segundo_apellido dependen directamente del cliente_id y no de otros campos intermedios.

Estrategia de auditoría a utilizar

Se implementó la tabla auditoria con los siguientes campos: usuario, acción, tabla, fecha/hora y descripción. Cualquier acción relevante puede registrarse mediante triggers o procedimientos futuros, permitiendo trazabilidad total.

Esta tabla almacena registros detallados sobre las acciones realizadas por los usuarios en el sistema y tiene los siguientes campos:

- usuario: nombre del usuario que ejecuta la acción (usando SYSTEM_USER).
- acción: tipo de operación realizada (por ejemplo: INSERT, DELETE, UPDATE).
- tabla: nombre de la tabla afectada.
- fecha_hora: marca temporal del momento exacto en que se ejecutó la acción.
- descripción: mensaje descriptivo personalizado que detalla la operación registrada.

Mecanismos implementados

Se crearon triggers (AFTER) que permiten el registro automático de acciones relevantes:

- trg_auditoria_pelicula_insert: registra la inserción de nuevas películas.
- trg_auditoria_funcion_delete: registra la eliminación de funciones.
- trg_reservar_asiento: actualiza el estado del asiento en la tabla asiento_funcion y se puede vincular a una auditoría futura.

También se implementó una vista especializada vw_auditoria que permite consultar el historial completo, y vw_auditoria_reciente, que muestra sólo los eventos del último mes.

Tecnologías utilizadas

- SQL Server 2019 – Motor de base de datos relacional.
- Lenguaje SQL – Para consultas y procedimientos.
- GitHub – Para control de versiones.
- Azure Data Studio – Herramienta para consultas y gestión. (utilizado por Brenda)
- Docker – Para entorno aislado de pruebas. (utilizado por Brenda, utiliza una imagen de sql server en docker).
- SQL Server Management Studio (SSMS)

Conclusiones

El desarrollo de este proyecto no solo significó crear una base de datos funcional, sino también entender cómo una solución tecnológica puede transformar por completo la operación de una empresa como Mi Cine es Tu Cine. A lo largo del proceso, fuimos más allá del simple cumplimiento de requerimientos técnicos: analizamos la realidad de una empresa emergente, sus necesidades, limitaciones y su proyección a futuro.

Construir esta base de datos nos permitió aplicar conceptos clave como normalización, integridad referencial, procedimientos almacenados, vistas, triggers y auditoría, con el objetivo de ofrecer una solución sólida, segura y escalable. Cada tabla y cada relación fueron diseñadas con propósito, pensando en la eficiencia del sistema y en la experiencia del usuario final, tanto interno como externo.

Más allá del código y las consultas, este proyecto refleja el poder de la tecnología para generar orden, facilitar el trabajo diario y abrir nuevas posibilidades. Un sistema como este puede marcar la diferencia entre una empresa que sobrevive y una que evoluciona.

Recomendaciones

Seguir construyendo este proyecto:

Esta base de datos ya organiza y controla los aspectos esenciales del cine, pero su verdadero potencial se alcanzará cuando se conecte con una interfaz web o móvil que acerque toda esta funcionalidad al público. La recomendación es continuar el desarrollo con un frontend intuitivo y accesible.

Implementar auditoría más profunda y proactiva:

Aunque ya existe un sistema de auditoría, sería crucial registrar más eventos críticos y generar alertas automatizadas.

Pensar en el usuario siempre:

Más allá de las relaciones entre tablas, nunca hay que olvidar que detrás de cada compra hay una persona que quiere disfrutar de una película sin complicaciones. Cada nueva función que se implemente debe mejorar esa experiencia.

Referencia

Cliente. (2025). Comunicación directa.

Anexos

Scripts generados

Tablas:

```

CREATE DATABASE mi_cine_es_tu_cine;
GO

USE mi_cine_es_tu_cine;
GO

-- Tabla #1: Película
CREATE TABLE pelicula (
    pelicula_id INT PRIMARY KEY IDENTITY(1,1),
    titulo VARCHAR(50) NOT NULL,
    sinopsis VARCHAR(250) NOT NULL,
    duracion INT NOT NULL,
    pais_origen VARCHAR(50) NOT NULL,
    imagen_url VARCHAR(250) NOT NULL
);
GO

-- Tabla #2: Película Idioma
CREATE TABLE pelicula_idioma (
    pelicula_id INT NOT NULL,
    idioma VARCHAR(50) NOT NULL,
    CONSTRAINT fk_pelicula_pi FOREIGN KEY (pelicula_id) REFERENCES pelicula(pelicula_id)
);
GO

-- Tabla #3: Película Subtítulo
CREATE TABLE pelicula_subtitulo (
    pelicula_id INT NOT NULL,
    subtítulo VARCHAR(50) NOT NULL,
    CONSTRAINT fk_pelicula_ps FOREIGN KEY (pelicula_id) REFERENCES pelicula(pelicula_id)
);
GO

-- Tabla #4: Actor
CREATE TABLE actor (
    actor_id INT PRIMARY KEY IDENTITY(1,1),
    nombre_actor VARCHAR(100) NOT NULL,
    nacionalidad VARCHAR(50) NOT NULL
);
GO

-- Tabla #5: Categoría
CREATE TABLE categoría (
    categoría_id INT PRIMARY KEY IDENTITY(1,1),
    nombre_categoria VARCHAR(50) NOT NULL
);
GO

-- Tabla #6: Película-Actor (Relación N:M)
CREATE TABLE película_actor (
    película_id INT NOT NULL,
    actor_id INT NOT NULL,
    CONSTRAINT fk_película_p FOREIGN KEY (película_id) REFERENCES película(película_id),
    CONSTRAINT fk_actor_a FOREIGN KEY (actor_id) REFERENCES actor(actor_id),
    CONSTRAINT pk_película_actor PRIMARY KEY (película_id, actor_id)
);
GO

-- Tabla #7: Película-Categoría (Relación N:M)
CREATE TABLE película_categoría (
    categoría_id INT NOT NULL,
    película_id INT NOT NULL,
    CONSTRAINT fk_categoria_a FOREIGN KEY (categoría_id) REFERENCES categoría(categoría_id),
    CONSTRAINT fk_película_pc FOREIGN KEY (película_id) REFERENCES película(película_id),
    CONSTRAINT pk_película_categoria PRIMARY KEY (película_id, categoría_id)
);
GO

-- Tabla #8: Cartelera
CREATE TABLE cartelera (
    cartelera_id INT PRIMARY KEY IDENTITY(1,1),
    fecha_inicio DATE NOT NULL,
    fecha_fin DATE NOT NULL
);
GO

```

```
-- Tabla #9: Sala
CREATE TABLE sala (
    sala_id INT PRIMARY KEY IDENTITY(1,1),
    ubicacion VARCHAR(100) NOT NULL,
    nombre_sala VARCHAR(50) NOT NULL
);
GO

-- Tabla #10: Función
CREATE TABLE funcion (
    funcion_id INT PRIMARY KEY IDENTITY(1,1),
    cartelera_id INT NOT NULL,
    sala_id INT NOT NULL,
    hora VARCHAR(50) NOT NULL,
    fecha DATE NOT NULL,
    CONSTRAINT fk_cartelera_f FOREIGN KEY (cartelera_id) REFERENCES cartelera(cartelera_id),
    CONSTRAINT fk_sala_f FOREIGN KEY (sala_id) REFERENCES sala(sala_id)
);
GO

-- Tabla #11: Película-Función (Relación N:M)
CREATE TABLE pelicula_funcion (
    pelicula_id INT NOT NULL,
    funcion_id INT NOT NULL,
    CONSTRAINT fk_pelicula_f FOREIGN KEY (pelicula_id) REFERENCES pelicula(pelicula_id),
    CONSTRAINT fk_funcion_f FOREIGN KEY (funcion_id) REFERENCES funcion(funcion_id),
    CONSTRAINT pk_pelicula_funcion PRIMARY KEY (pelicula_id, funcion_id)
);

-- Tabla #12: Asiento
CREATE TABLE asiento (
    asiento_id INT PRIMARY KEY IDENTITY(1,1),
    sala_id INT NOT NULL,
    fila VARCHAR(25) NOT NULL,
    secuencia VARCHAR(50) NOT NULL,
    CONSTRAINT fk_sala_a FOREIGN KEY (sala_id) REFERENCES sala(sala_id)
);
GO

-- Tabla #13: Asiento-Función
CREATE TABLE asiento_funcion (
    asiento_funcion_id INT PRIMARY KEY IDENTITY(1,1),
    funcion_id INT NOT NULL,
    asiento_id INT NOT NULL,
    estado VARCHAR(30) NOT NULL, -- Libre o Reservado
    CONSTRAINT fk_funcion_af FOREIGN KEY (funcion_id) REFERENCES funcion(funcion_id),
    CONSTRAINT fk_asiento_af FOREIGN KEY (asiento_id) REFERENCES asiento(asiento_id),
    CONSTRAINT chk_estado_asiento CHECK (estado IN ('Libre', 'Reservado'))
);
GO

-- Tabla #14: Dependiente
CREATE TABLE dependiente (
    dependiente_id INT PRIMARY KEY IDENTITY(1,1),
    nombre_dependiente VARCHAR(100) NOT NULL
);
GO

-- Tabla #15: Cliente
CREATE TABLE cliente (
    cliente_id INT PRIMARY KEY IDENTITY(1,1),
    fecha_nacimiento DATE NULL,
    pais_origen VARCHAR(50) NULL,
    nombre_cliente VARCHAR(50) NOT NULL,
    apellido VARCHAR(50) NULL,
    segundo_apellido VARCHAR(50) NULL
);
GO

DROP TABLE compra
-- Tabla #16: Compra
CREATE TABLE compra (
    compra_id INT PRIMARY KEY IDENTITY(1,1),
    cliente_id INT NOT NULL,
    dependiente_id INT NOT NULL,
    metodo_pago_id INT NOT NULL,
    canal VARCHAR(50) NOT NULL,
    CONSTRAINT fk_cliente_c FOREIGN KEY (cliente_id) REFERENCES cliente(cliente_id),
    CONSTRAINT fk_dependiente_c FOREIGN KEY (dependiente_id) REFERENCES dependiente(dependiente_id),
    CONSTRAINT fk_metodo_pago FOREIGN KEY (metodo_pago_id) REFERENCES metodo_de_pago(metodo_pago_id),
    CONSTRAINT chk_canal_compra CHECK (canal IN ('boleteria', 'web'))
);
GO

-- Tabla #17: Tiquete
CREATE TABLE tiquete (
    tiquete_id INT PRIMARY KEY IDENTITY(1,1),
    funcion_id INT NOT NULL,
    asiento_funcion_id INT NOT NULL,
    compra_id INT NOT NULL,
    precio INT NOT NULL,
    tipo_pago_o_puntos VARCHAR(75) NOT NULL,
    CONSTRAINT fk_funcion_t FOREIGN KEY (funcion_id) REFERENCES funcion(funcion_id),
    CONSTRAINT fk_asiento_funcion_t FOREIGN KEY (asiento_funcion_id) REFERENCES asiento_funcion(asiento_funcion_id),
    CONSTRAINT fk_compra_t FOREIGN KEY (compra_id) REFERENCES compra(compra_id)
);
GO

-- Tabla #18: Puntos
CREATE TABLE puntos (
    puntos_id INT PRIMARY KEY IDENTITY(1,1),
    cliente_id INT NOT NULL,
    acumulado INT NOT NULL,
    canjeado INT NULL,
    saldo INT NULL,
    CONSTRAINT fk_cliente_p FOREIGN KEY (cliente_id) REFERENCES cliente(cliente_id)
);
GO
```

```
-- Tabla #19: Método de Pago
CREATE TABLE metodo_de_pago (
    metodo_pago_id INT PRIMARY KEY IDENTITY(1,1),
    cliente_id INT NOT NULL,
    tipo VARCHAR(50) NOT NULL,
    detalles VARCHAR(75) NOT NULL,
    CONSTRAINT fk_cliente_mp FOREIGN KEY (cliente_id) REFERENCES cliente(cliente_id)
);
GO

-- Tabla #20: Auditoria
CREATE TABLE auditoria (
    auditoria_id INT PRIMARY KEY IDENTITY(1,1),
    usuario VARCHAR(50) NOT NULL,
    accion VARCHAR(20) NOT NULL,
    tabla VARCHAR(50) NOT NULL,
    fecha_hora DATETIME NOT NULL DEFAULT GETDATE(),
    descripcion VARCHAR(100) NULL
);
GO
```

Views

```
USE mi_cine_es_tu_cine
GO

--#1 Para ver películas que están en cartelera
CREATE VIEW vw_peliculas_cartelera AS
SELECT
    p.pelicula_id,
    p.titulo,
    p.sinopsis,
    p.duracion,
    p.pais_origen,
    p.imagen_url,
    f.fecha,
    f.hora,
    s.nombre_sala,
    s.ubicacion
FROM pelicula p
INNER JOIN pelicula_funcion pf ON p.pelicula_id = pf.pelicula_id
INNER JOIN funcion f ON pf.funcion_id = f.funcion_id
INNER JOIN sala s ON f.sala_id = s.sala_id
INNER JOIN cartelera c ON f.cartelera_id = c.cartelera_id
WHERE GETDATE() BETWEEN c.fecha_inicio AND c.fecha_fin;
GO

--#2 Para visualizar actores que participan en esas películas
CREATE VIEW vw_pelicula_actor AS
SELECT
    p.pelicula_id,
    p.titulo,
    p.sinopsis,
    p.duracion,
    p.pais_origen,
    p.imagen_url,
    a.nombre_actor,
    a.nacionalidad
FROM pelicula p
INNER JOIN pelicula_actor pa ON p.pelicula_id = pa.pelicula_id
INNER JOIN actor a ON pa.actor_id = a.actor_id
GO

--#3 Para visualizar los asientos que tiene cada sala
CREATE VIEW vw_sala_asientos AS
SELECT
    s.sala_id,
    s.ubicacion,
    s.nombre_sala,
    a.asiento_id,
    a.fila,
    a.secuencia
FROM sala s
LEFT JOIN asiento a ON s.sala_id = a.sala_id
GO
```

```
--#4 Para poder ver los asientos disponibles solo si están en el estado de libre
CREATE VIEW vw_asiento_disponibles AS
SELECT
    af.asiento_funcion_id,
    a.fila,
    a.secuencia,
    s.nombre_sala,
    f.fecha,
    f.hora
FROM asiento_funcion af
INNER JOIN asiento a ON af.asiento_id = a.asiento_id
INNER JOIN funcion f ON af.funcion_id = f.funcion_id
INNER JOIN sala s ON f.sala_id = s.sala_id
WHERE af.estado = 'Libre';
GO

--#5 Realiza un conteo de cuales películas son las más vendidas
CREATE VIEW vw_peliculas_mas_vendidas AS
SELECT
    p.titulo,
    COUNT(t.tiquete_id) AS cantidad_ventas
FROM pelicula p
INNER JOIN pelicula_funcion pf ON p.pelicula_id = pf.pelicula_id
INNER JOIN tiquete t ON pf.funcion_id = t.funcion_id
GROUP BY p.titulo
GO

--#Para ver las vistas en el orden deseado
SELECT *
FROM vw_peliculas_mas_vendidas
ORDER BY cantidad_ventas DESC;
GO
```

```
--#6 Ver subtítulo e idioma de una película
CREATE VIEW vw_pelicula_sub_idioma AS
SELECT
    p.pelicula_id,
    p.titulo,
    pi.idioma,
    ps.subtitle
FROM pelicula p
LEFT JOIN pelicula_idioma pi ON p.pelicula_id = pi.pelicula_id
LEFT JOIN pelicula_subtitle ps ON p.pelicula_id = ps.pelicula_id
GO

--#7 Para ver los puntos que tiene el cliente
CREATE VIEW vw_cliente_puntos AS
SELECT
    c.cliente_id,
    CONCAT(c.nombre_cliente, ' ', ISNULL(c.apellido,''), ' ', ISNULL(c.segundo_apellido, '')) AS nombre_completo,
    p.acumulado,
    p.canjeado,
    p.saldo
FROM cliente c
INNER JOIN puntos p ON c.cliente_id = p.cliente_id;
GO
```

```
--#8 para ver métodos de pago en compra
CREATE VIEW vw_metodo_pago_compra AS
SELECT
    c.compra_id,
    CONCAT_WS(' ', cl.nombre_cliente, cl.apellido, cl.segundo_apellido) AS nombre_cliente,
    mp.tipo AS tipo_pago,
    mp.detalles AS detalles_pago,
    c.canal
FROM compra c
LEFT JOIN cliente cl ON c.cliente_id = cl.cliente_id
INNER JOIN metodo_de_pago mp ON c.metodo_pago_id = mp.metodo_pago_id
GO
```

```
--#9 Para visualizar compra mas a detalle
CREATE VIEW vw_compra_tiquete_dependiente AS
SELECT
    c.compra_id,
    CONCAT_WS(' ', cl.nombre_cliente, cl.apellido, cl.segundo_apellido) AS nombre_cliente,
    c.nombre_dependiente,
    c.canal,
    mp.tipo AS tipo_pago,
    mp.detalles AS detalles_pago,
    t.precio,
    t.tipo_pago_o_puntos
FROM compra c
LEFT JOIN cliente cl ON c.cliente_id = cl.cliente_id
INNER JOIN metodo_de_pago mp ON c.metodo_pago_id = mp.metodo_pago_id
INNER JOIN dependiente d ON c.dependiente_id = d.dependiente_id
INNER JOIN tiquete t ON c.compra_id = t.compra_id
GO
```

```
--#10 Vista tabla auditoria
CREATE VIEW vw_auditoria AS
SELECT
    a.auditoria_id,
    a.usuario,
    a.accion,
    a.tabla AS tabla_afectada,
    a.fecha_hora,
    a.descripcion
FROM auditoria a
GO
```

```
--#11 Ver auditoria de menos de un mes
CREATE VIEW vw_auditoria_reciente AS
SELECT
    a.auditoria_id,
    a.usuario,
    a.accion,
    a.tabla AS tabla_afectada,
    a.fecha_hora,
    a.descripcion
FROM auditoria a
WHERE fecha_hora >= DATEADD(MONTH, -1, GETDATE());
GO
```

```
--#12 Para ver todo desde compra
CREATE VIEW vw_compra_todo AS
SELECT
    c.compra_id,
    CONCAT_WS(' ', cl.nombre_cliente, cl.apellido, cl.segundo_apellido) AS nombre_cliente,
    d.nombre_dependiente,
    c.canal,
    mp.tipo AS tipo_pago,
    mp.detalles AS detalles_pago,
    t.precio,
    t.tipo_pago_o_puntos,
    df.fecha,
    s.nombre_sala,
    s.ubicacion,
    ass.filas,
    ass.seccuencia,
    p.titulo,
    p.descripcion,
    p.duracion,
    p.pais_origen,
    p.imagen_url,
    pl.idioma,
    ps.subtitle,
    ps.actor,
    a.nacionalidad,
    pct.nombre_categoria
FROM compra c
LEFT JOIN cliente cl ON c.cliente_id = cl.cliente_id
INNER JOIN metodo_de_pago mp ON c.metodo_pago_id = mp.metodo_pago_id
INNER JOIN dependiente d ON c.dependiente_id = d.dependiente_id
INNER JOIN tiquete t ON c.compra_id = t.compra_id
INNER JOIN funcion f ON t.funcion_id = df.funcion_id
INNER JOIN sala s ON f.sala_id = s.sala_id
INNER JOIN asiento ass ON ass.sala_id = ass.sala_id
INNER JOIN pelicula_funcion pf ON df.funcion_id = pf.funcion_id
INNER JOIN pelicula p ON pf.pelicula_id = p.pelicula_id
INNER JOIN pelicula_idioma pi ON p.pelicula_id = pi.pelicula_id
INNER JOIN idioma i ON pi.idioma_id = i.idioma_id
INNER JOIN pelicula_actor pa ON p.pelicula_id = pa.pelicula_id
INNER JOIN actor a ON pa.actor_id = a.actor_id
INNER JOIN pelicula_categoria pc ON p.pelicula_id = pc.pelicula_id
INNER JOIN categoria pcs ON pc.pelicula_id = pcs.categoria_id
GO
```

```
--13 Peliculas en cartelera mas sus actores e idiomas
CREATE VIEW vw_pelicula_cartelera AS
SELECT
    p.pelicula_id,
    p.titulo,
    p.sinopsis,
    p.duracion,
    p.pais_origen,
    p.imagen_url,
    pl.idioma,
    ps.subtitle,
    a.nombre_actor,
    a.nacionalidad,
    cs.nombre_categoria,
    f.fecha,
    f.hora,
    s.nombre_sala,
    s.ubicacion
FROM pelicula p
INNER JOIN pelicula_funcion pf ON p.pelicula_id = pf.pelicula_id
INNER JOIN funcion f ON pf.funcion_id = f.funcion_id
INNER JOIN pelicula_idioma pl ON p.pelicula_id = pl.pelicula_id
INNER JOIN pelicula_subtitle ps ON p.pelicula_id = ps.pelicula_id
INNER JOIN pelicula_actor pa ON p.pelicula_id = pa.pelicula_id
INNER JOIN actor a ON pa.actor_id = a.actor_id
INNER JOIN pelicula_categoria pc ON p.pelicula_id = pc.pelicula_id
INNER JOIN categoria cs ON pc.categoria_id = cs.categoria_id
INNER JOIN sala s ON f.sala_id = s.sala_id
INNER JOIN cartelera c ON f.cartelera_id = c.cartelera_id
WHERE GETDATE() BETWEEN c.fecha_inicio AND c.fecha_fin;
GO
```

```
--#14 Pelicula con sus asientos funcion
CREATE VIEW vw_pelicula_sala_asiento_funcion AS
SELECT
    p.pelicula_id,
    p.titulo,
    p.sinopsis,
    p.duracion,
    p.pais_origen,
    p.imagen_url,
    f.fecha,
    f.hora,
    s.nombre_sala,
    s.ubicacion,
    a.fila,
    a.secuencia,
    af.estado
FROM pelicula p
INNER JOIN pelicula_funcion pf ON p.pelicula_id = pf.pelicula_id
INNER JOIN funcion f ON pf.funcion_id = f.funcion_id
INNER JOIN sala s ON f.sala_id = s.sala_id
INNER JOIN asiento a ON s.sala_id = a.sala_id
INNER JOIN asiento_funcion af ON f.funcion_id = af.funcion_id
GO
```

Algunos Scripts de procedimiento almacenados

```
-- 1. Registrar la compra de un asiento para una pelicula
ALTER PROCEDURE sp_registrar_compra_asiento
@cliente_id INT,
@dependiente_id INT,
@metodo_pago_id INT,
@funcion_id INT,
@asiento_id INT,
@precio INT,
@tipo_pago_o_puntos VARCHAR(75)
AS
BEGIN TRY
    BEGIN TRY
        DECLARE @asiento_funcion_id INT;
        INSERT INTO asiento_funcion (funcion_id, asiento_id, estado)
        VALUES (@funcion_id, @asiento_id, 'Reservado');
        SET @asiento_funcion_id = SCOPE_IDENTITY();
        INSERT INTO compra (cliente_id, dependiente_id, metodo_pago_id, canal)
        VALUES (@cliente_id, @dependiente_id, @metodo_pago_id, 'web');
        DECLARE @compra_id INT = SCOPE_IDENTITY();
        INSERT INTO ticket (funcion_id, asiento_funcion_id, compra_id, precio, tipo_pago_o_puntos)
        VALUES (@funcion_id, @asiento_funcion_id, @compra_id, @precio, @tipo_pago_o_puntos);
    END TRY
    BEGIN CATCH
        RAISERROR('Error al registrar la compra del asiento.', 16, 1);
    END CATCH
END TRY
END CATCH
END;
GO
```

```
-- 2. Peliculas en cartelera
ALTER PROCEDURE sp_peliculas_en_cartelera
AS
BEGIN
    BEGIN TRY
        SELECT DISTINCT p.titulo, c.fecha_inicio, c.fecha_fin
        FROM pelicula p
        JOIN pelicula_funcion pf ON p.pelicula_id = pf.pelicula_id
        JOIN funcion f ON pf.funcion_id = f.funcion_id
        JOIN cartelera c ON f.cartelera_id = c.cartelera_id
        WHERE GETDATE() BETWEEN c.fecha_inicio AND c.fecha_fin;
    END TRY
    BEGIN CATCH
        RAISERROR('Error al obtener las películas en cartelera.', 16, 1);
    END CATCH
END;
GO
```

```
-- 3. Asientos disponibles para una función
ALTER PROCEDURE sp_asientos_disponibles @funcion_id INT
AS
BEGIN
    BEGIN TRY
        SELECT a.asiento_id, a.fila, a.secuencia
        FROM asiento a
        JOIN sala s ON a.sala_id = s.sala_id
        JOIN funcion f ON f.sala_id = s.sala_id
        WHERE f.funcion_id = @funcion_id
        AND a.asiento_id NOT IN (
            SELECT asiento_id FROM asiento_funcion WHERE funcion_id = @funcion_id AND estado = 'Reservado'
        );
    END TRY
    BEGIN CATCH
        RAISERROR('Error al obtener los asientos disponibles.', 16, 1);
    END CATCH
END;
GO
```

Triggers

```
-- 1 Evita insertar un tiquete si el asiento ya está reservado.
CREATE TRIGGER trg_prevenir_asiento_reservado
ON tiquete
INSTEAD OF INSERT
AS
BEGIN
    IF EXISTS (
        SELECT 1
        FROM inserted i
        JOIN asiento_funcion af ON i.asiento_funcion_id = af.asiento_funcion_id
        WHERE af.estado = 'Reservado'
    )
    BEGIN
        RAISERROR('El asiento ya está reservado.', 16, 1);
        ROLLBACK;
    END
    ELSE
    BEGIN
        INSERT INTO tiquete (funcion_id, asiento_funcion_id, compra_id, precio, tipo_pago_o_puntos)
        SELECT funcion_id, asiento_funcion_id, compra_id, precio, tipo_pago_o_puntos
        FROM inserted;
    END
END;
GO
```

--2 Sumar puntos al cliente según el monto pagado en cada tiquete.

```
CREATE TRIGGER trg_sumar_puntos
ON tiquete
AFTER INSERT
AS
BEGIN
    UPDATE p
    SET acumulado = p.acumulado + pt.puntos_sumados,
        saldo = p.saldo + pt.puntos_sumados
    FROM puntos p
    JOIN (
        SELECT c.cliente_id, SUM(i.precio / 100) AS puntos_sumados
        FROM inserted i
        JOIN compra c ON c.compra_id = i.compra_id
        GROUP BY c.cliente_id
    ) pt ON p.cliente_id = pt.cliente_id;
END;
GO
```

```
--3 Para auditorias de película
CREATE TRIGGER trg_auditoria_pelicula
ON película
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @usuario NVARCHAR(50) = SUSER_SNAME();
    DECLARE @accion NVARCHAR(20);
    DECLARE @descripcion NVARCHAR(100);

    IF EXISTS (SELECT * FROM inserted) AND EXISTS (SELECT * FROM deleted)
    BEGIN
        SET @accion = 'UPDATE';
        SET @descripcion = 'Actualización en película';
    END
    ELSE IF EXISTS (SELECT * FROM inserted)
    BEGIN
        SET @accion = 'INSERT';
        SET @descripcion = 'Inserción en película';
    END
    ELSE IF EXISTS (SELECT * FROM deleted)
    BEGIN
        SET @accion = 'DELETE';
        SET @descripcion = 'Eliminación en película';
    END;

    INSERT INTO auditoria (usuario, accion, tabla, fecha_hora, descripcion)
    VALUES (@usuario, @accion, 'pelicula', GETDATE(), @descripcion);
END;
GO
```

Login

```
--Usamos la master para poder crear un login
USE master
GO

--Creamos los login deseados con respectivas con sus contraseñas
CREATE LOGIN martillo WITH PASSWORD = '123456';
CREATE LOGIN francisco_coulon WITH PASSWORD = 'coulon123';
CREATE LOGIN usuario_normal WITH PASSWORD = 'usuario123';
GO

--Luego elegimos la local para poder asignar los permisos de cada uno
USE mi_cine_es_tu_cine
GO

--Creamos el usuario con su respectivo login y sus permisos
CREATE USER martillo FOR LOGIN martillo;
ALTER ROLE db_datareader ADD MEMBER martillo;
ALTER ROLE db_datawriter ADD MEMBER martillo;
ALTER ROLE db_owner ADD MEMBER martillo;

CREATE USER francisco_coulon FOR LOGIN francisco_coulon;
ALTER ROLE db_datareader ADD MEMBER francisco_coulon;
ALTER ROLE db_datawriter ADD MEMBER francisco_coulon;
ALTER ROLE db_owner ADD MEMBER francisco_coulon;

--Se le asigna manual a que puede ver
GRANT SELECT ON dbo.vw_asiento_disponibles TO usuario_normal;
GRANT SELECT ON dbo.vw_pelicula_actor TO usuario_normal;
GRANT SELECT ON dbo.vw_peliculas_mas_vendidas TO usuario_normal;
GRANT SELECT ON dbo.vw_pelicula_sub_idioma TO usuario_normal;
GRANT SELECT ON dbo.vw_sala_asientos TO usuario_normal;
GRANT SELECT ON dbo.vw_pelicula_cartelera_isa TO usuario_normal;
GRANT SELECT ON dbo.vw_pelicula_sala_asiento_funcion TO usuario_normal;

GO
```

Consideraciones para una versión final/Siguientes pasos

Optimización del modelo relacional:

Revisar y validar las relaciones entre las tablas para asegurar integridad referencial, evitar redundancias y garantizar el correcto uso de claves primarias y foráneas.

Implementación en un SGBD

Migrar el diseño lógico a un sistema gestor de base de datos (SQL Server) y probar su funcionalidad mediante scripts de inserción, actualización y consulta.

Seguridad y control de acceso

Definir roles y permisos para diferentes tipos de usuarios, asegurando la protección de datos sensibles y evitando accesos no autorizados.

Pruebas funcionales y de rendimiento

Realizar pruebas exhaustivas para verificar que el sistema cumple con los requerimientos planteados, que las consultas son eficientes y que el rendimiento es óptimo bajo carga.

Documentación del sistema

Crear una documentación técnica completa que incluya el modelo de datos, la lógica de negocio, las funciones implementadas y los procedimientos de mantenimiento.

Bitácoras Base de datos

Bitácora 1

Fecha	Lugar	Estudiantes	Observaciones
24 mayo, 2025	Virtual, Discord	Axel Castillo John Granos Brenda Espinoza	-Se realizó el modelo entidad-relación. -Se empezó con la documentación, creando un drive, haciendo la portada, introducción y objetivos.
24 mayo, 2025	Virtual, Discord	Axel Castillo John Granos Brenda Espinoza	-Se pasó del modelo entidad relación al modelo relacional.

Bitácora 2

Fecha	Lugar	Estudiantes	Observaciones
-------	-------	-------------	---------------

28 mayo, 2025	Virtual, meet	Axel Castillo John Granos Brenda Espinoza	-Nos reunimos con el Profesor Coulon, para que nos brindara retroalimentación y revisara el modelo entidad-relación. -Asimismo, se corrigieron algunos atributos de la entidad relación y se agregaron otros en película sugeridos por el profesor.
28 mayo, 2025	Virtual, meet	Axel Castillo John Granos Brenda Espinoza	-Se agregó parte del modelo entidad relación al modelo relacional.

Bitácora 3

Fecha	Lugar	Estudiantes	Observaciones
30 mayo, 2025	Virtual, discord	John Granos Brenda Espinoza Axel Castillo	-Se finalizó el modelo relacional con sus respectivos cambios, en draw.io

Bitácora 4

Fecha	Lugar	Estudiantes	Observaciones
07 junio, 2025	Virtual, discord	John Granos Brenda Espinoza Axel Castillo	-Se reparten los temas para realizar la base de datos John (tablas) Axel (sps, triggers) Brenda (triggers, backup)

Bitácora 5

Fecha	Lugar	Estudiantes	Observaciones
08 junio, 2025	Virtual, discord	John Granos Brenda Espinoza Axel Castillo	-Se crean las ramas del repositorio. -Se realizaron cambios en las tablas, además se hicieron 8 vistas.