**MYSQL**

**POR:**

**JERICOTH CASTELLAR**

**Asignatura:**

**BASES DE DATOS**

**Semestre 5**

**Docente o Tutor:**

**JOHN CARLOS ARRIETA**

**FUNDACION UNIVERSITARIA COLOMBO INTERNACIONAL.**

**TECNOLOGIA EN DESARROLLO DE SOFTWARE**

**SEMESTRE V**

**CARTAGENA DE INDIAS DT y C.**

**2025.**

Contenido

[Introducción 3](#_Toc194960726)

[Objetivo general 3](#_Toc194960727)

[Objetivos específicos 3](#_Toc194960728)

[Justificación 4](#_Toc194960729)

[Desarrollo 5](#_Toc194960730)

[Identificación de entidades y atributos 5](#_Toc194960731)

[Relaciones 6](#_Toc194960732)

[Diagrama Entidad-Relacion (DER) 7](#_Toc194960733)

[Pasar el DER a Modelo Relacional (MR) 7](#_Toc194960734)

[Modelo Relacional (MR) 9](#_Toc194960735)

[IMPLEMENTACION DE BASE DE DATOS 10](#_Toc194960736)

Introducción

En la actualidad, la gestión eficiente de concesionarios de vehículos es clave para asegurar la competitividad en un mercado altamente dinámico. Los concesionarios deben manejar una variedad de información sobre los vehículos que comercializan, así como sobre los diferentes extras, precios y características técnicas que ofrecen a sus clientes. Para lograr una gestión adecuada, es fundamental contar con un sistema de gestión de concesionarios que permita almacenar y organizar toda esta información de manera efectiva.

Este sistema debe ser capaz de registrar y gestionar la información detallada sobre los vehículos, sus características estándar y opcionales, los detalles de las transacciones de ventas, los datos sobre los vendedores, así como los servicios oficiales asociados al concesionario. Este tipo de solución no solo facilita la venta de vehículos, sino que también mejora la atención al cliente y optimiza los procesos administrativos. Además, permite una trazabilidad clara de las transacciones, lo cual es esencial tanto para la gestión interna como para el cumplimiento de normativas legales y fiscales.

Objetivo general

Desarrollar un sistema de gestión dirigido a concesionarios de vehículos automotores que permita almacenar y gestionar de manera eficiente la información relacionada con los vehículos, sus características, los extras opcionales, los servicios oficiales asociados, las transacciones de venta y los vendedores.

Objetivos específicos

* Gestionar el inventario de vehículos disponibles en el concesionario y los servicios oficiales asociados, permitiendo el registro detallado de cada modelo, incluyendo marca, modelo, precio, potencia fiscal, cilindrada, equipamiento estándar y extras opcionales.
* Registrar y gestionar la información sobre los extras opcionales y sus precios, permitiendo definir qué extras son estándar para cada modelo y cuáles son opcionales.
* Implementar un sistema para registrar las transacciones de venta, que incluya el vendedor que realizó la venta, el precio de venta, el método de pago y los extras seleccionados por el cliente.
* Contabilizar las ventas realizadas por cada vendedor, facilitando un seguimiento detallado de sus transacciones y desempeño.
* Gestionar los servicios oficiales vinculados a cada concesionario, incluyendo su información básica, como nombre, dirección y NIF.
* Permitir la trazabilidad de cada vehículo vendido, incluyendo el número de bastidor, la matrícula, la fecha de entrega y si el vehículo estaba disponible en stock o fue solicitado a fábrica.

Justificación

El desarrollo de este sistema de gestión es necesario debido a la creciente demanda de soluciones tecnológicas que permitan mejorar la eficiencia de los concesionarios de vehículos en la gestión de sus operaciones. Las concesionarias deben gestionar una gran cantidad de información relacionada con los vehículos, los modelos, los extras, los precios, y las transacciones, lo que puede resultar complejo y propenso a errores si se realiza manualmente.

Un sistema informático bien estructurado puede reducir significativamente el margen de error, optimizar la gestión de inventarios, mejorar el seguimiento de las ventas y facilitar la administración de las relaciones con los clientes. Además, al permitir la trazabilidad de los vehículos y las transacciones, el sistema contribuiría a cumplir con las normativas legales y fiscales. Asimismo, la capacidad de registrar los detalles de las ventas y el rendimiento de los vendedores es un factor clave para mejorar la toma de decisiones y fomentar la mejora continua en la gestión comercial.

Desarrollo

**Mitto de Aires**

El desarrollo de este sistema de gestión es necesario debido a la creciente demanda de soluciones tecnológicas que permitan mejorar la eficiencia de los concesionarios de vehículos en la gestión de sus operaciones. Las concesionarias deben gestionar una gran cantidad de información relacionada con los vehículos, los modelos, los extras, los precios, y las transacciones, lo que puede resultar complejo y propenso a errores si se realiza manualmente.

Un sistema informático bien estructurado puede reducir significativamente el margen de error, optimizar la gestión de inventarios, mejorar el seguimiento de las ventas y facilitar la administración de las relaciones con los clientes. Además, al permitir la trazabilidad de los vehículos y las transacciones, el sistema contribuiría a cumplir con las normativas legales y fiscales. Asimismo, la capacidad de registrar los detalles de las ventas y el rendimiento de los vendedores es un factor clave para mejorar la toma de decisiones y fomentar la mejora continua en la gestión comercial.

Identificación de entidades y atributos

Concesionario: (ID\_Concesionario, nombre, dirección)

Servicio oficial: (NIF, Nombre, dirección, precio de venta)

Venta y Transacción: (ID\_Transaccion, motivo de compra, matricula, garantía, fecha y hora, origen)

Inventario: (Numero de Bastidor, Dirección, Disponibilidad)

Vendedores: (NIF, nombre, dirección)

Historial de Ventas: (ID\_Venta, fecha, rendimiento, satisfacción del cliente)

Vehículos: (Modelo, precio, Descuentos, potencia fiscal, cilindrado, tipo, tamaño, capacidad, combustible, estado)

Extras opcionales: (ID\_Elemento, precio, disponibles, nombre, stock)

Clientes: (ID\_Cliente, extras seleccionados, precio individual, fecha de entrega, matricula, stock, método de pago, nombre, dirección, tipo de cliente)

Relaciones

Concesionario: Un Servicio Oficial tiene un concesionario, y un concesionario tiene un servicio oficial.

Servicio Oficial: Un concesionario tiene un servicio oficial, un inventario es administrado por un servicio oficial y una venta y transacción es registrada por un servicio oficial.

Venta y Transacción: Se registran múltiples Ventas y Transacciones en un servicio oficial, Un Cliente tiene una compra en una Venta Y Transacción, múltiples Vendedores tienen múltiples Ventas y Transacciones.

Inventario: Un servicio oficial administra múltiples inventarios. Un vehículo pertenece a un inventario.

Vendedores: Múltiples Ventas y Transacciones tienen múltiples Vendedores. Un historial de ventas pertenece a un solo Vendedor. Los clientes tienen un solo vendedor

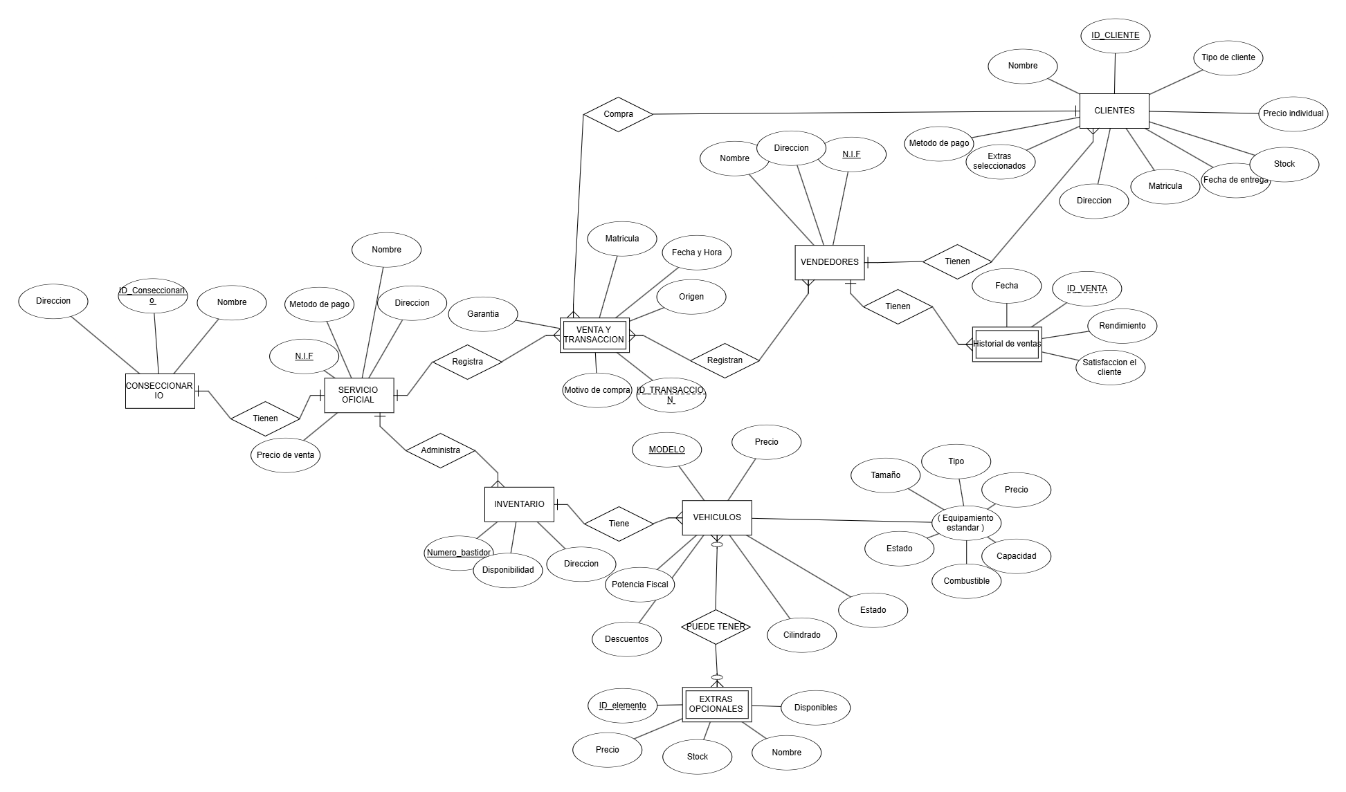
Historial de ventas: Múltiples historiales de venta pertenecen a un vendedor.

Clientes: Venta y Transacciones tiene múltiples compras de un cliente. Los vendedores tienen múltiples clientes.

Vehiculos: Multiples vehiculos pertenecen a un inventario. Un Extra Opcional pertenece a un Vehiculo

Extras Opcionales: Multiples extras opcionales pertenece n a un solo vehiculo

Diagrama Entidad-Relacion (DER)



Pasar el DER a Modelo Relacional (MR)

Concesionario

* PK ID\_Concesionario
* Nombre
* Direccion

Servicio Oficial

* PK NIF
* Nombre
* Direccion
* Precio\_de\_Venta
* FK ID\_Concesionario

Venta Y Transaccion

* PK ID\_Transaccion
* Motivo\_de\_compra
* Matricula
* Garantía
* Fecha\_y\_hora
* Origen
* FK NIF
* FK ID\_CLIENTE

Inventario

* PK Numero\_de\_Bastidor
* Dirección
* Disponibilidad
* FK NIF

Vendedores

* PK NIF
* Nombre
* Dirección
* FK ID\_Transaccion

Historial de ventas

* PK ID\_Venta
* Fecha
* Rendimiento
* Satisfacción\_Del\_Cliente
* FK NIF

Vehiculo

* PK Modelo
* Precio
* Descuentos
* Potencia\_Fiscal
* Cilindrado
* Tipo
* Tamao
* Capacidad
* Combustible
* Estado
* FK Numero\_de\_Bastidor

Extras opcionales

* PK ID\_Elemento
* Precio
* Disponibles
* Stock
* FK Modelo

Modelo Relacional (MR)

Interfaz de usuario gráfica, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

IMPLEMENTACION DE BASE DE DATOS

1. Qué es y qué problema resuelve el uso del concepto de Vistas SQL en bases de datos Relacionales o SQL

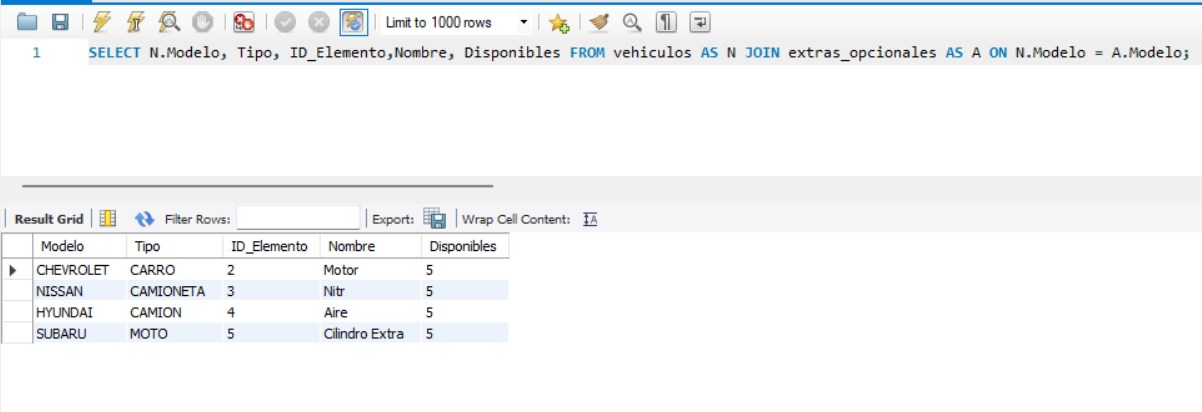
Una Vista SQL es una tabla virtual generada a partir de una consulta. Tiene de características que no almacena datos físicamente como una tabla tradicional si no que muestra los resultados de una consulta cada vez que se accede a ella. Por asi decirlo actúa como una ventana a los datos con el propósito de simplificar su administración y acceso.

Los problemas que resuelve el uso de una vista SQL son:

* **Simplificación de consultas complejas:** debido a que permite guardar consultas complejas, las vistas permiten un uso con mayor frecuencia de estas haciendo más fácil y rápido el acceso a los datos
* **Consistencia de los datos:** Las vistas SQL permiten y ayudan a mantener una consistencia debido a que centralizan consultas estándar es decir que evitan la duplicación o errores en el código de una consulta
* **Abstracción de datos:** La creación de las abstracciones de datos son más fáciles, se oculta la complejidad de la estructura subyacente a los usuarios finales.
* **Seguridad y control al acceso:** Las vistas SQL pueden limitar el acceso a ciertas colunas o filas de una tabla. Por ejemplo, se pueden crear vistas que muestren solo ciertos datos específicos a cierto grupo de usuarios o u usuario, protegiendo cierta información.
* **Compatibilidad con datos históricos o agregados:** Las vistas permiten mostrar ciertos datos históricos o calculados que son agregados por ciertos usuarios o que los usuarios necesitan con mayor frecuencia sin afectar el rendimiento del sistema

1. Analizar el Modelo Relacional del ejercicio que seleccionaron como Grupo de BD-1, diseñar una consulta multitabla y probarla (en MySQL, Oracle SQL, SQL Server y PostgreSQL).

SELECT N.Modelo, Tipo, ID\_Elemento,Nombre, Disponibles FROM vehiculos AS N JOIN extras\_opcionales AS A ON N.Modelo = A.Modelo;

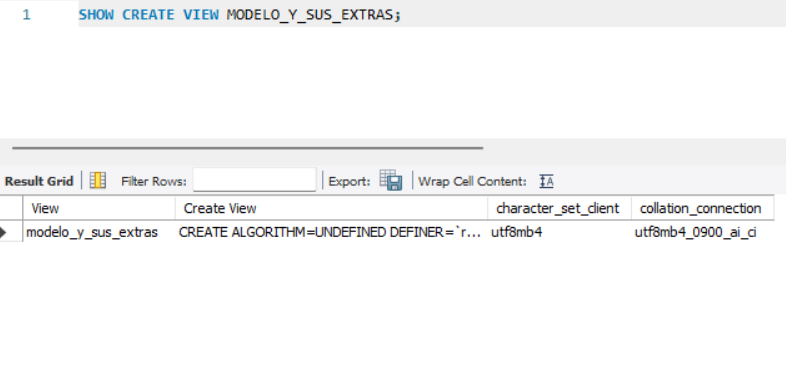


1. Usar la consulta anterior y crear con ella una Vista SQL

CREATE VIEW MODELO\_Y\_SUS\_EXTRAS AS SELECT N.Modelo, Tipo, ID\_Elemento,Nombre, Disponibles FROM vehiculos AS N JOIN extras\_opcionales AS A ON N.Modelo = A.Modelo;

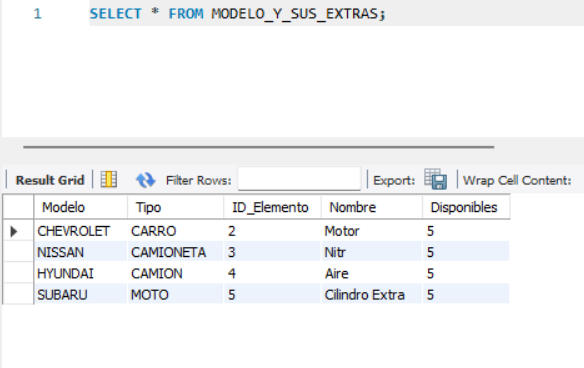
1. Mostrar la Vista SQL creada anteriormente

SHOW CREATE VIEW MODELO\_Y\_SUS\_EXTRAS;



1. Ejecutar la Vista SQL creada anteriormente

SELECT \* FROM MODELO\_Y\_SUS\_EXTRAS;

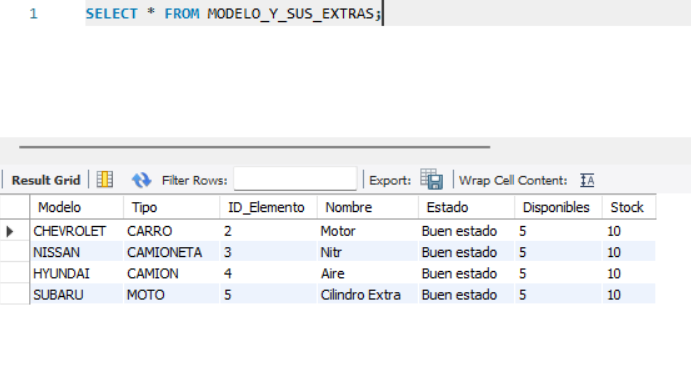


1. Abrir la Vista SQL y modificarla para cambiar la consulta por otra consulta o modificar la existente.

ALTER VIEW MODELO\_Y\_SUS\_EXTRAS AS SELECT N.Modelo, Tipo, ID\_Elemento,Nombre,Estado, Disponibles, Stock FROM vehiculos AS N JOIN extras\_opcionales AS A ON N.Modelo = A.Modelo;

1. Volver a ejecutar la Vista SQL

SELECT \* FROM MODELO\_Y\_SUS\_EXTRAS;



1. Las vistas tienen ventajas, desventajas y limitaciones, muchas de estas características dependen mucho del motor de bases de datos que estemos utilizando, en este punto usted debe:
   1. Describir las ventajas del uso de Vistas SQL para el X motor de bases de datos sobre el cual desarrolló la actividad

* Las visas permiten una simplificación de consultas complejas para reutilizarlas de manera más sencilla, facilitando el acceso a los datos
* Permite una mayor seguridad en MySQL las vistas permiten ocultar ciertas columnas o filas a los usuarios. Esto asegura que solo accedan a la información que necesitan y protege otros tipos de datos.
* Mejora la consistencia en los datos al centralizar las consultas en vistas ayudando a mantener una consistencia en los resultados cuando hay múltiples usuarios o aplicaciones consultando datos
  1. Describir las desventajas del uso de Vistas SQL para el X motor de bases de datos sobre el cual desarrolló la actividad.
* En MySQL las vistas pueden producir un rendimiento limitado debido a que las vistas no almacenan datos físicamente. Por lo tanto, cada vez que se consulta una vista, la consulta original se ejecuta nuevamente, lo cual puede impactar en el rendimiento si la consulta es muy compleja o las tablas tienen muchos registros.
* No se pueden crear índices directamente en la vista lo que dificulta la optiizacion del rendimiento de una consulta.
* En caso de que la estructura de las tablas cambie ya sea que se modifiquen, eliminen, o renombren las vistas asociadas pueden fallar, requiriendo que estas sean ajustadas manualmente.

* 1. Describir cuales son las limitaciones que presentan las Vistas SQL en el motor de BD que está utilizando, por ejemplo: Cuáles son las instrucciones SQL que no pueden ser utilizadas dentro de una Vista SQL. etc.
* No puedes incluir sentencias que modifiquen los datos como “INSERT”, “UPDATE”, “DELETE”, entre otros
* No puedes usar variables locales ni subconsultas que utilicen variables que se definen dentro de la vista.
* No se admite “ORDER BY” dentro de una vista al menos no si no esta acompañado de una cláusula “LIMIT”.
* En MySQL no admite vistas materializadas nativamente es decir que no almacenan datos físicamente esto significa que las vistas siempre generan sus datos en tiempo real, lo que puede ser costoso en términos de rendimiento.
* No se permiten vistas que involucren funciones almacenadas con afectos secundarios
* Las vistas no pueden contener consultas dinámicas ni procedimientos almacenados.

7. Para el desarrollo de los siguientes puntos referentes al tema de Transacciones, ustedes como grupo deben realizar las siguientes tareas sobre OracleSql, SQLServer y PostgreSQL y MySQL:

7.1. Crear una base de datos llamada prueba\_transacciones\_grupoXYZ, siendo XYZ el nombre del ejercicio escogido como equipo.

CREATE DATABASE prueba\_transacciones\_grupo29;

7.2. Crear tablas Deudores(cc, clave, nombre, apellido, email). email es único, todos son obligatorios

CREATE TABLE Deudores(

cc INT NOT NULL PRIMARY KEY,

clave VARCHAR(50) NOT NULL,

nombre VARCHAR(100) NOT NULL,

apellido VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL

);

7.3. Crear tablas Creditos(id, fecha, valor, cuotas, interes, estado, deudor\_id), todos son obligatorios, estado por defecto es Activo

CREATE TABLE Creditos(

id INT NOT NULL PRIMARY KEY,

fecha DATE NOT NULL,

valor INT NOT NULL,

cuotas INT NOT NULL,

interes VARCHAR(10) NOT NULL,

estado VARCHAR(20) NOT NULL DEFAULT "Activo",

deudor\_id INT NOT NULL,

FOREIGN KEY (deudor\_id) REFERENCES Deudores (cc)

);

7.4. Crear tablas Pagos(id, fecha, valor, credito\_id). Todos son o bligatorios. Use el comando check para evitar que la columna valor solo puede aceptar valor positivos mayores que 0 y la fecha del pago no puede ser mayor que la fecha actual

DEBIDO A LIMITACIONES DE MySQL es necesario crear un “TRIGGER” para poder realizar el check en fecha.

CREATE TABLE Pagos (

id INT NOT NULL PRIMARY KEY,

fecha DATE NOT NULL,

valor INT NOT NULL CHECK (valor > 0),

credito\_id INT NOT NULL,

FOREIGN KEY (credito\_id) REFERENCES Creditos (id) );

DELIMITER //

CREATE TRIGGER before\_pagos\_insert

BEFORE INSERT ON Pagos

FOR EACH ROW BEGIN

IF NEW.fecha > CURDATE() THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'La fecha del pago DEBE SER MENOR O IGUAL A LA ACTUAL'; END IF; END;

//

CREATE TRIGGER before\_pagos\_update

BEFORE UPDATE ON Pagos

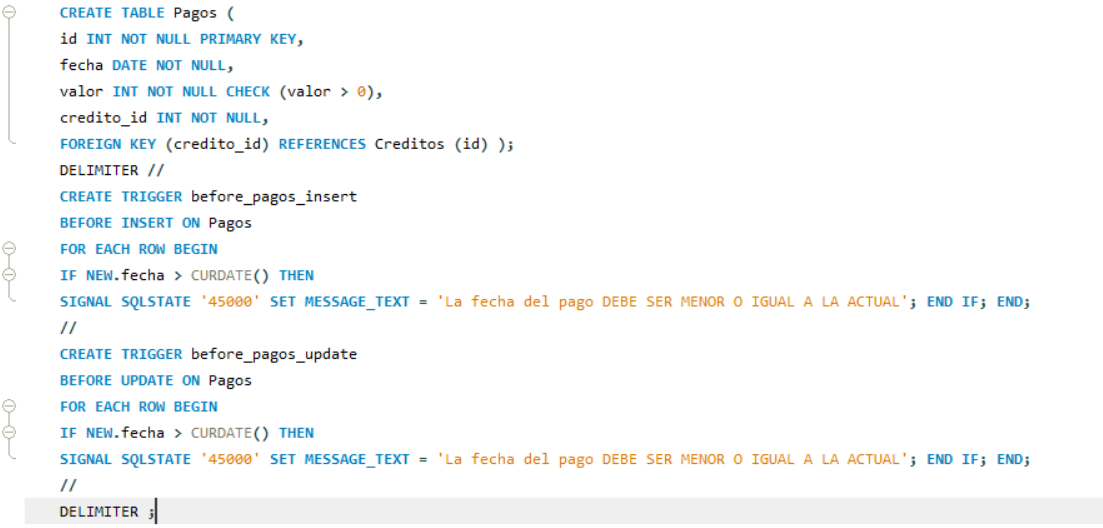
FOR EACH ROW BEGIN

IF NEW.fecha > CURDATE() THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'La fecha del pago DEBE SER MENOR O IGUAL A LA ACTUAL'; END IF; END;

//

DELIMITER ;



7.5. Insertar un al menor un Deudor:

El primero con cc=123, calve=Abc, nombre = Fulanito, apellido=De tal, [email=fulanito1@gmail.com](mailto:email=fulanito1@gmail.com)

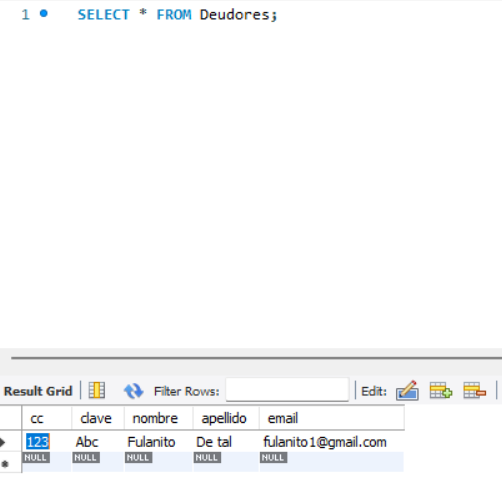
INSERT INTO Deudores (cc, clave, nombre, apellido, email) VALUES (123, "Abc", "Fulanito", "De tal", "fulanito1@gmail.com");

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

7.6. Mostrar todos los Deudores insertados en la BD

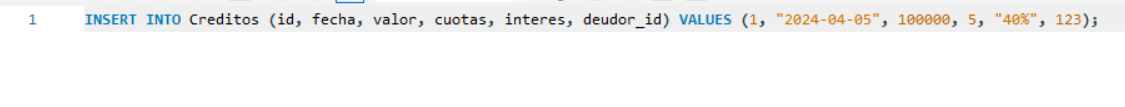
SELECT \* FROM Deudores;



7.7. Insertar al menos un Crédito:

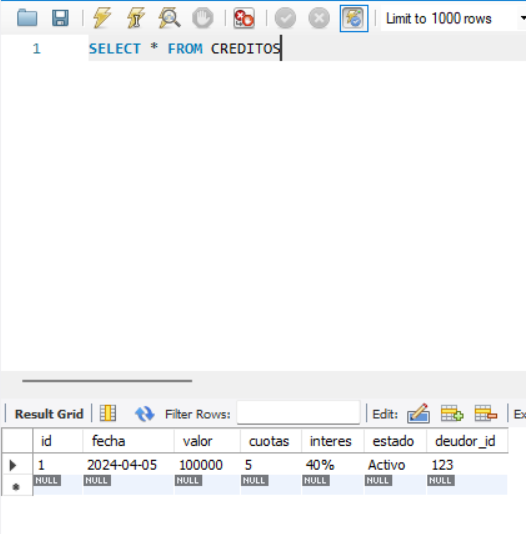
id=1, fecha=hace un año, valor=100000, cuotas=5, interes\_mes=40%, deudor\_id=123

INSERT INTO Creditos (id, fecha, valor, cuotas, interes, deudor\_id) VALUES (1, "2024-04-05", 100000, 5, "40%", 123);



7.8. Mostrar todos los Créditos insertados en la BD

o SELECT \* FROM Creditos;



7.9. Insertar 5 pagos así (no modifique ningún valor del ejemplo, colocar las fechas correspondientes, el formato de fecha correspondiente para bd):

7.9.1. id=1, fecha=un mes después del crédito, valor=28000, credito\_id=1

7.9.2. id=2, fecha=un mes después de la fecha actual, valor=28000, credito\_id=1

7.9.3. id=3, fecha=un mes después del primer pago, valor=28000, credito\_id=1

7.9.4. id=4, fecha=un mes después del primer pago, valor=28000, credito\_id=1

7.9.5. id=5, fecha=un mes después del primer pago, valor=-28000, credito\_id=1

INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (1, "2024-05-05", 28000, 1);

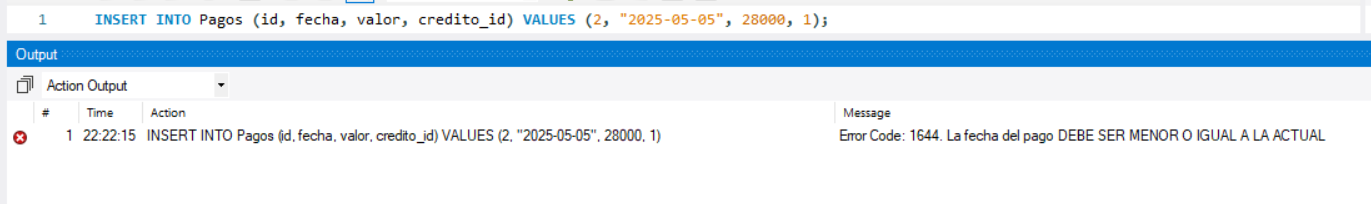
INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (2, "2025-05-05", 28000, 1);

INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (3, "2024-06-05", 28000, 1);

INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (4, "2024-06-05", 28000, 1);

INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (5, "2024-06-05", -28000, 1);





Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

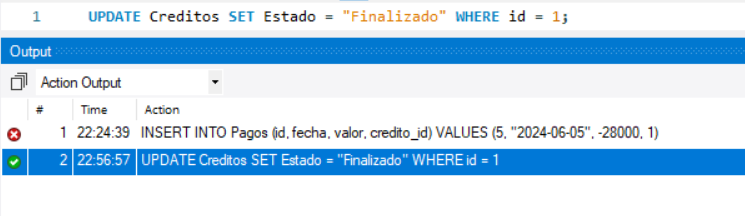
El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación, Sitio web

El contenido generado por IA puede ser incorrecto.

7.10. Cambiar o actualizar el estado del Crédito con id=1, para que ahora sea Finalizado

UPDATE Creditos SET Estado = "Finalizado" WHERE id = 1;



7.11. Mostrar los datos del Crédito con id=1

SELECT \* FROM Creditos WHERE id=1;

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

7.12. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

SELECT \* FROM Pagos WHERE credito\_id=1;

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

7.13. Totalizar el total de valores pagados para el crédito con id=1, para que aparezca así: TOTAL PAGOS: $140000

En el enunciado se dice que debe aparecer “TOTAL PAGOS: $140000”sin embargo al realizar la suma de los pagos, solo algunos si fueron permitidos en la base de datos fue $84000, faltan un total de $56000 esto porque faltan los de la inserción a pagos en la cual la fecha era mayor a la actual y la otra del valor menor a 0.

SELECT CONCAT("TOTAL PAGOS: $", SUM(valor)) AS TOTAL\_PAGOS FROM Pagos WHERE credito\_id=1;

Interfaz de usuario gráfica, Texto, Aplicación

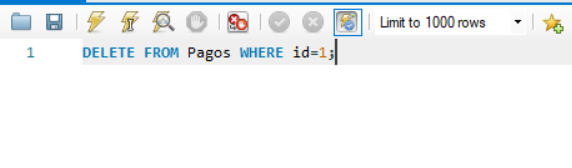
El contenido generado por IA puede ser incorrecto.

7.14. Analice el resultado de cada una de las operaciones anteriores, comente y explique el porqué de cada uno.

* Insertando datos en la tabla de pagos solo 3 de las 5 fueron realizadas con éxito, las otras no debido a que la fecha establecida era mayor que la actual, como condición en el “TRIGGER”. Por lo tanto, no pudo realizar todas las inserciones correctamente, además de que en la 2da inserción el CHECK evito que se ingresara un valor menor a 0 .

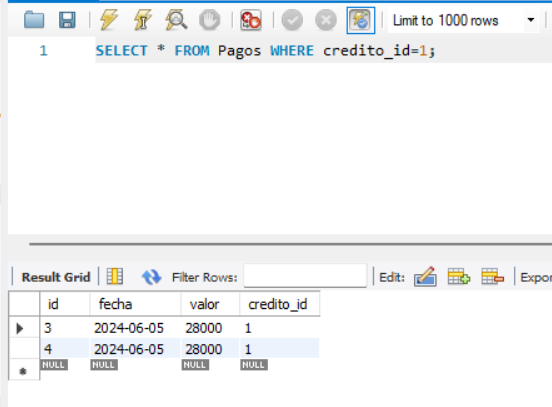
7.15. Elimine todos los pagos de crédito con ID: 1

DELETE FROM Pagos WHERE id=1;



7.16. Mostrar todos los pagos que existen para el crédito con id=1

SELECT \* FROM Pagos WHERE crédito\_id=1;



7.17. inicie una TRANSACCION

7.18. dentro de la transacción volver a insertar los mismo pagos anteriores:

7.18.1. id=1, fecha=un mes después del crédito, valor=28000, credito\_id=1

7.18.2. id=2, fecha=un mes después de la fecha actual, valor=28000, credito\_id=1

7.18.3. id=3, fecha=un mes después del primer pago, valor=28000, credito\_id=1

7.18.4. id=4, fecha=un mes después del primer pago, valor=28000, credito\_id=1

7.18.5. id=5, fecha=un mes después del primer pago, valor=-28000, credito\_id=1

START TRANSACTION;

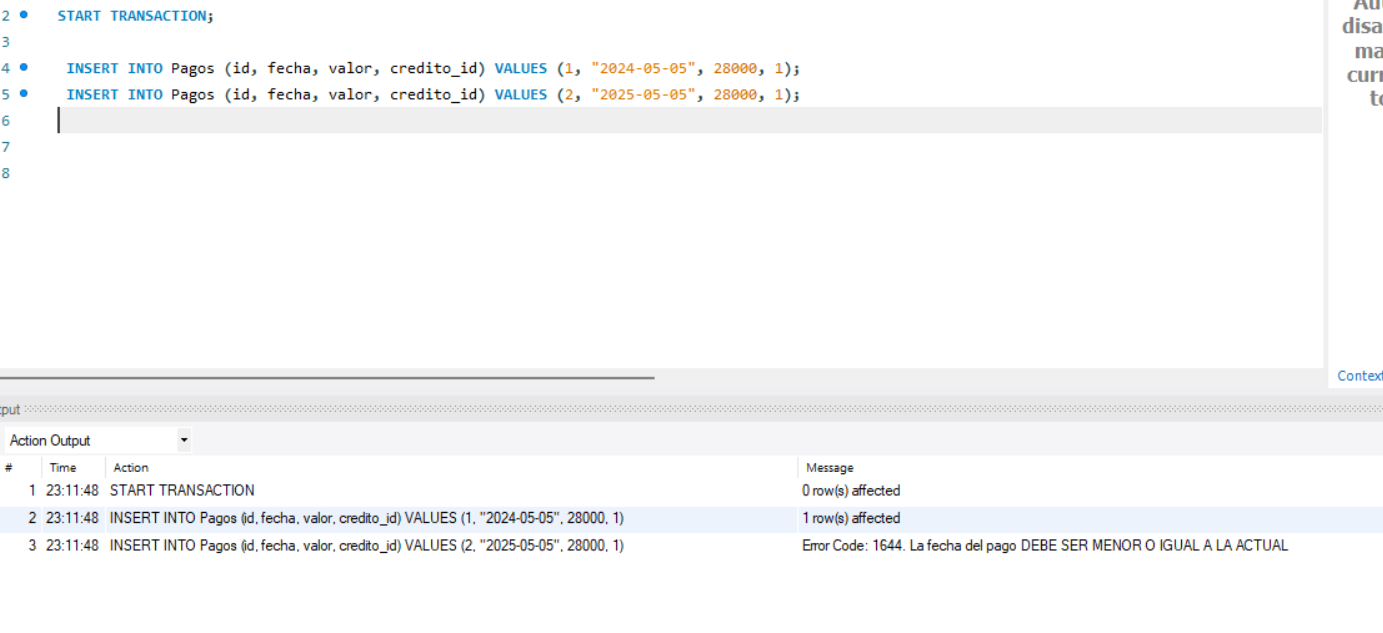
INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (1, "2024-05-05", 28000, 1);

INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (2, "2025-05-05", 28000, 1);

INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (3, "2024-06-05", 28000, 1);

INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (4, "2024-06-05", 28000, 1);

INSERT INTO Pagos (id, fecha, valor, credito\_id) VALUES (5, "2024-06-05", -28000, 1);



7.19. Cambiar o actualizar el estado del Crédito con id=1, para que ahora sea Finalizado

UPDATE Creditos SET Estado = "Finalizado" WHERE id = 1;

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

7.20. Mostrar los datos del Crédito con id=1

SELECT \* FROM Creditos WHERE id=1;

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

7.21. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

SELECT \* FROM Pagos WHERE credito\_id=1;

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

7.22. Totalizar el total de valores pagados para el crédito con id=1, para que aparezca así: TOTAL PAGOS: $140000

Debido a que como se menciono anteriormente en el punto 7.14 el total de pagos no es el que pide el enunciado.

SELECT CONCAT("TOTAL PAGOS: $", SUM(valor)) AS TOTAL\_PAGOS FROM Pagos WHERE credito\_id=1;

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

7.23. Cancelar la transaccion (rollblack)

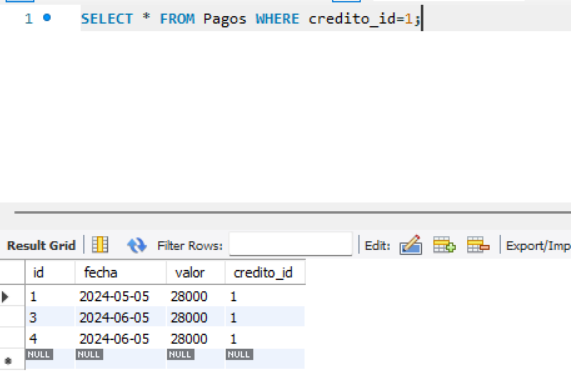
ROLLBACK;

Interfaz de usuario gráfica, Texto, Correo electrónico

El contenido generado por IA puede ser incorrecto.

7.24. Mostrar todos los Pagos insertados al Crédito con ID=1 insertados en la BD

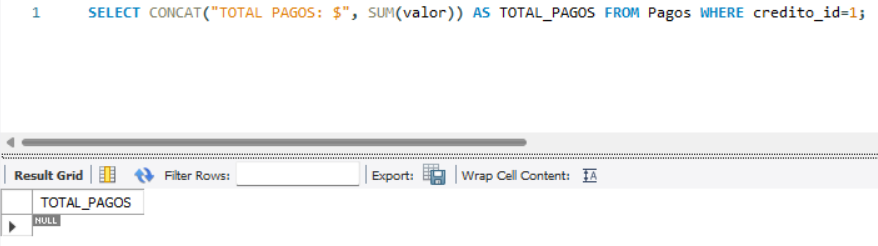
SELECT \* FROM Pagos WHERE credito\_id=1;



7.25. Totalizar el total de valores pagados para el crédito con id=1, para que aparezca así: TOTAL PAGOS: $140000

No se pudo realizar este enunciado correctamente debido a que la suma de todos los pagos fue cancelada en el punto 7.33 cancelando la transacción.

SELECT CONCAT (“TOTAL PAGOS: ”, SUM (valor)) AS TOTAL\_PAGOS FROM pagos WHERE credito\_id=1;



7.26. Analice el resultado de cada una de las operaciones anteriores, comente y explique el porqué de cada uno.

Se realizaron 5 inserciones de las cuales solo 3 se ejecutraron, las otras dos fueron canceladas por el “TRIGGER” y “CHECK”

No se pudo realizar este enunciado correctamente debido a que la suma de todos los pagos fue cancelado en el punto 7.33 cancelando la transacción.

Durante el proceso el “ROLLBACK” cancelo todas las insersiones regresando a cuando se iniciaron las transacciones.