Universidad Autónoma del Estado de Hidalgo

Instituto de Ciencias Básicas e Ingeniería

Licenciatura en Ciencias Computacionales

1.2. Práctica. Gestión de flotilla de autos

Sexto semestre, Grupo dos

Catedrático: Dr. Eduardo Cornejo Velázquez

Base de datos distribuidas

Equipo:
Juan Carlos Montes Gonzalez
Jennifer Resendiz Isidro

1. Introducción

Analizar una flotilla de autos para el diseño e implementación de de su base de datos en MySQL que permita proponer una solución eficiente para su gestión.

Marco teórico

Flotillas de autos

Las flotillas de autos son utilizadas como transporte para mejorar la eficiencia de la movilidad y son un medio para que se lleve a cabo adecuadamente el trabajo. También ayudan al manejo de la logística porque cubren necesidades como el desplazamiento de productos, maquinarias, herramientas, equipos, insumos u ofrecer servicios [1].

Procedimientos Almacenados (*Stored Procedures*)

Los procedimientos almacenados son programas creados por el usuario y almacenados directamente en el servidor de la base de datos. Su objetivo principal es agrupar reglas u operaciones comerciales que deben ejecutarse de manera continua, con el fin de evitar la repetición de código en aplicaciones externas.

Entre sus ventajas destacan:

- **Mayor eficiencia:** Al ejecutarse en el mismo motor de la base de datos.
- **Facilitan el mantenimiento:** Permiten la preservación y reutilización del código.
- **Refuerzan la seguridad:** Los usuarios pueden ejecutar procesos sin necesidad de acceder directamente a las tablas subyacentes.

Funciones (*Functions*)

Las funciones en SQL son similares a los procedimientos, pero se distinguen por retornar siempre un valor o un conjunto de valores (como tablas). Pueden considerarse como *vistas paramétricas*, ya que permiten generar resultados dinámicos en función de los parámetros ingresados.

Las funciones se emplean frecuentemente para:

- Simplificar consultas complejas.
- Encapsular cálculos o transformaciones comunes.
- Aumentar la modularidad en el diseño de bases de datos.

En sistemas modernos, las funciones pueden devolver valores escalares (números, texto, fecha) o incluso conjuntos de registros completos.

Estructuras de Control: Condicionales y Repetitivas

Las estructuras de control son esenciales en la programación de bases de datos, ya que permiten dirigir el flujo de ejecución de un programa, realizar elecciones basadas en condiciones y repetir fragmentos de código de forma controlada.

2.4.1 Estructuras Condicionales

`IF-THEN-ELSIF-ELSE-END IF`**:** Permite el análisis secuencial de condiciones:

- `IF-THEN`: Evalúa una condición; si es verdadera (`TRUE`), ejecuta un bloque de instrucciones.
- `ELSIF`: Permite verificar condiciones adicionales si la anterior no se cumple.
- `ELSE`: Bloque opcional que se ejecuta cuando ninguna de las condiciones anteriores es verdadera.

`CASE-WHEN-THEN-ELSE-END CASE`**:** Evalúa una expresión con múltiples valores posibles.

2.4.2 Estructuras Repetitivas (Bucles)

`LOOP-EXIT WHEN-END LOOP:` Ejecuta un bloque de código al menos una vez; la salida del ciclo se controla mediante `EXIT` o `EXIT WHEN`.

`WHILE-LOOP-END LOOP:` Evalúa la condición antes de ejecutar el bloque. Si la condición es falsa inicialmente, el bloque no se ejecuta.

`FOR-IN-LOOP-END LOOP:` Repite un número específico de veces dentro de un rango definido. El contador se declara implícitamente y no requiere definición en `DECLARE`. Además, permite la iteración en orden inverso con `IN REVERSE`.

Disparadores (*Triggers*)

Los disparadores son programas que se activan automáticamente en respuesta a eventos específicos en la base de datos, como inserciones (`INSERT`), actualizaciones (`UPDATE`) o eliminaciones (`DELETE`) de datos.

Su objetivo es preservar la integridad y coherencia de los datos sin requerir intervención del usuario. Los disparadores son útiles para:

- Implementar reglas empresariales de forma automática.
- Auditar modificaciones en la base de datos.
- Ejecutar acciones en cascada, como añadir entradas en tablas relacionadas o verificar restricciones complejas.

En SQL, los disparadores pueden definirse para ejecutarse *antes* (`BEFORE`) o *después* (`AFTER`) del evento que los activa, y a nivel de fila o de tabla. Esta flexibilidad exige utilizarlos con precaución para evitar impactos negativos en el rendimiento.

Herramientas empleadas

1. **MySQL Server:** MySQL es un sistema de gestión de base de datos relacional (RDBMS) de código abierto. Se utiliza para almacenar, organizar y recuperar datos de manera eficiente, especialmente en aplicaciones web y sistemas de gestión de contenido (CMS). MySQL utiliza SQL (Structured Query Language) para interactuar con la base de datos, permitiendo a los usuarios realizar consultas, actualizaciones y otras operaciones sobre los datos [2].
2. **Overleaf:** Overleaf es un editor colaborativo de LaTeX en línea que permite a múltiples usuarios escribir, editar y publicar documentos científicos, técnicos y académicos de forma simultánea a través de un navegador web [3].

Desarrollo

Análisis de requisitos

- **Conductor:** Nombre, Teléfono, Fecha de nacimiento, Años de experiencia.
- **Vehículo:** Modelo, Año fabricación, Tipo de vehículo, Matrícula, Marca, Estado.
- **Combustible:** Consumo de gasolina, Fecha, Litros, Nivel de combustible, Precio total, Tipo de combustible.
- **Documentación:** Seguros, Tenencias, Tarjetas de regulación, Licencias de usuarios, Verificaciones, Multas.
- **Rastreo:** Monitoreo, Tiempo, Ubicación.

- **Mantenimiento:** Costo, Tipo de mantenimiento, Refracciones, Taller, Fecha, Tiempo de mantenimiento.
- **Rutas:** Inicio, Destino, Distancia, Tiempo Estimado.

Modelo Entidad-Relación

En la Figura 1 se presenta la propuesta de Modelo Entidad-Relación para la flotilla de autos.

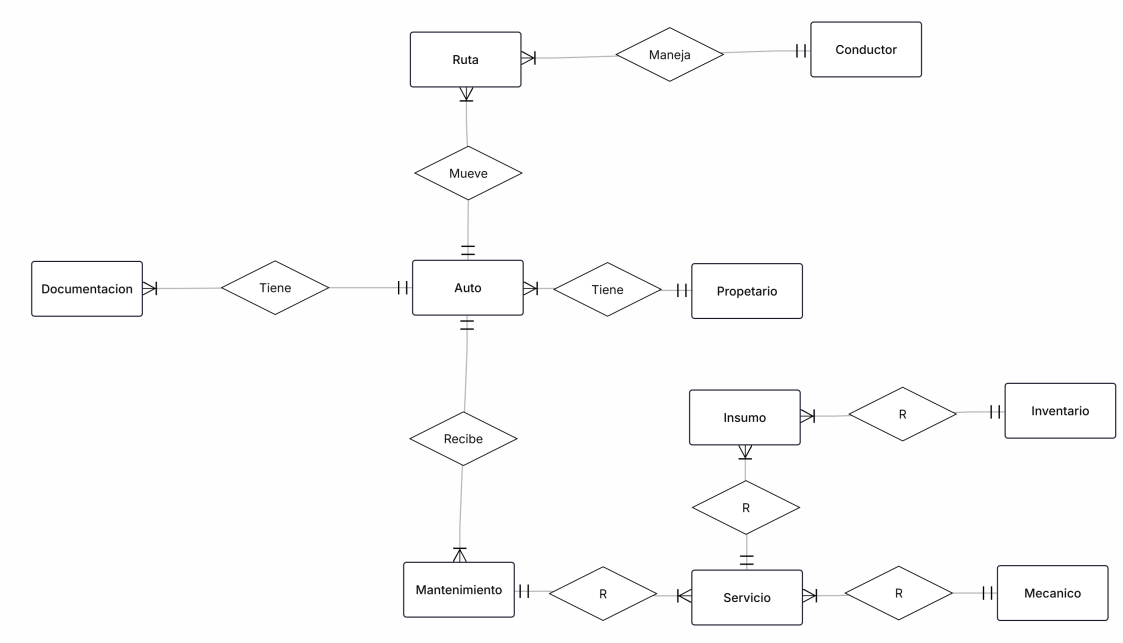

Figure 1: Modelo Entidad-Relación de la flotilla de autos

Table 1: Matriz de relaciones

Entidades	Conductor	Vehículo	Ruta	Rastreo	Mantenimiento	Documentación	Historial_Gasolina
Conductor		X	X				
Vehículo	X		X	X	X	X	X
Ruta	X	X					
Rastreo		X					
Mantenimiento		X					
Documentación		X					
Historial_Gasolina		X					

Modelo relacional

En la Figura 3 se presenta la propuesta de Modelo Relacional para la flotilla de autos.

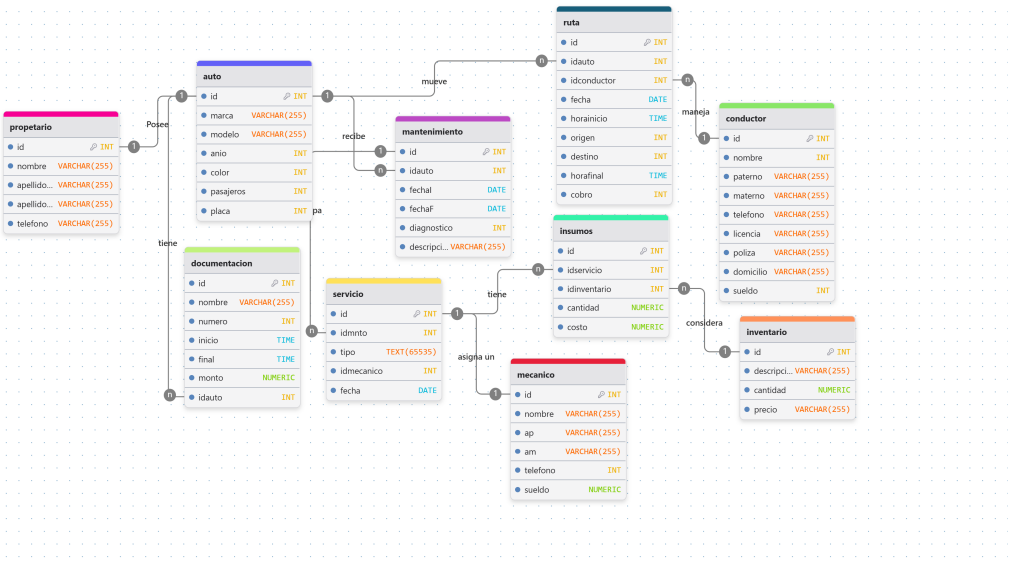

Figure 2: Modelo relacional de la flotilla de autos

Sentencias SQL

Listing 1: Crear base de datos FlotillaAutos.

```
CREATE DATABASE FlotillaAutos.
```

Listing 2: Crear tablas de la base de datos FlotillaAutos.

```
  create table auto (
  id_auto int not null,
  marca varchar(255) not null,
  modelo varchar(255) not null,
  anio int not null,
  color varchar(255) not null,
  pasajeros int not null,
  placa varchar(255) not null,
  primary key(id_auto)
);

create table conductor (
  id_conductor int not null,
  nombre varchar(255) not null,
  apellidopaterno varchar(255) not null,
  apellidomaterno varchar(255) not null,
  telefono varchar(255) not null,
  licencia varchar(255) not null,
  poliza varchar(255) not null,
  domicilio varchar(255) not null,
  sueldo decimal(10,2) not null,
  primary key(id_conductor)
);

create table documento (
  id_documento int not null,
  id_auto int not null,
  numero varchar(255) not null,
  nombre varchar(255) not null,
  inicio date not null,
  final date not null,
  monto decimal(10,2) not null,
  primary key(id_documento),
  foreign key (id_auto) references auto(id_auto)
);

create table ruta (
  id_ruta int not null,
  id_auto int not null,
  id_conductor int not null,
  fecha date not null,
  horainicio time not null,
  origen varchar(255) not null,
  destino varchar(255) not null,
  horallegada time not null,
  cobro decimal(10,2) not null,
  primary key(id_ruta),
```

```
  foreign key (id_auto) references auto(id_auto),
  foreign key (id_conductor) references conductor(id_conductor)
);

create table mantenimiento (
  id_mantenimiento int not null,
  id_auto int not null,
  fechainicio date not null,
  fechafinal date not null,
  diagnostico varchar(255) not null,
  descripcion varchar(500) not null,
  primary key(id_mantenimiento),
  foreign key (id_auto) references auto(id_auto)
);

create table mecanico (
  id_mecanico int not null,
  nombre varchar(100) not null,
  apellidopaterno varchar(255) not null,
  apellidomaterno varchar(255) not null,
  telefono varchar(255) not null,
  sueldo decimal(10,2) not null,
  primary key(id_mecanico)
);

create table servicio (
  id_servicio int not null,
  id_mantenimiento int not null,
  id_mecanico int not null,
  tipo varchar(255) not null,
  fecha date not null,
  primary key(id_servicio),
  foreign key (id_mantenimiento) references mantenimiento(id_mantenimiento),
  foreign key (id_mecanico) references mecanico(id_mecanico)
);

create table inventario (
  id_inventario int not null,
  descripcion varchar(255) not null,
  cantidad int not null,
  precio decimal(10,2) not null,
  primary key(id_inventario)
);

create table insumo (
  id_insumo int not null,
  id_servicio int not null,
  id_inventario int not null,
  cantidad int not null,
  costo decimal(10,2) not null,
  primary key(id_insumo),
  foreign key (id_servicio) references servicio(id_servicio),
  foreign key (id_inventario) references inventario(id_inventario)
);
```

```
CREATE TABLE propietario (
   id_propietario INT NOT NULL,
   nombre VARCHAR(255) NOT NULL,
   apellidopaterno VARCHAR(255) NOT NULL,
   apellidomaterno VARCHAR(255) NOT NULL,
   telefono VARCHAR(255) NOT NULL,
   id_auto INT NOT NULL,
  c PRIMARY KEY(id_propietario),
   FOREIGN KEY (id_auto) REFERENCES auto(id_auto)
);
```

Listing 3: Poblado de tablas

```
insert into auto values
(1, 'Toyota', 'Corolla', 2018, 'Blanco', 5, 'ABC-123'),
(2, 'Nissan', 'Versa', 2020, 'Negro', 5, 'XYZ-456'),
(3, 'Honda', 'Civic', 2019, 'Rojo', 5, 'LMN-789'),
(4, 'Ford', 'Transit', 2021, 'Gris', 12, 'JKL-321'),
(5, 'Chevrolet', 'Aveo', 2017, 'Azul', 5, 'QWE-654');

insert into conductor values
(1, 'Juan', 'P rez', 'L pez', '5512345678', 'LIC12345', 'POL123',
'Calle Falsa 123', 8500.50),
(2, 'Mar a', 'G mez', 'Ram rez', '3323456789', 'LIC67890', 'POL456',
'Av. Central 45', 9200.00),
(3, 'Carlos', 'Hern ndez', 'Torres', '8123456789', 'LIC11223', 'POL789',
'Col. Reforma 76', 8800.75),
(4, 'Ana', 'Mart nez', 'Flores', '2223456789', 'LIC33445', 'POL321',
'Blvd. Ju rez 22', 9100.00),
(5, 'Luis', 'S nchez', 'Morales', '4423456789', 'LIC55667', 'POL654',
'Av. Hidalgo 11', 8700.30);

insert into documento values
(1, 1, 'DOC001', 'Tarjeta de Circulaci n', '2024-01-01', '2026-01-01', 1200.00),
(2, 2, 'DOC002', 'Seguro Vehicular', '2024-05-01', '2025-05-01', 4500.50),
(3, 3, 'DOC003', 'Verificaci n Ambiental', '2024-03-15', '2025-03-15', 800.00),
(4, 4, 'DOC004', 'Licencia de Transporte', '2024-07-10', '2026-07-10', 2000.25),
(5, 5, 'DOC005', 'Seguro contra Da os', '2024-02-20', '2025-02-20', 3800.00);

insert into ruta values
(1, 1, 1, '2025-08-01', '08:00:00', 'Queretaro',
'Hidalgo', '10:45:00', 250.00),
(2, 2, 2, '2025-08-02', '09:30:00', 'Pachuca',
'CDMX', '10:15:00', 380.50),
(3, 3, 3, '2025-08-03', '07:15:00', 'Estado de Hidalgo',
'Guadalajara', '10:55:00', 520.00),
(4, 4, 4, '2025-08-04', '10:00:00', 'Saltillo',
'Monterrey', '12:00:00', 300.00),
(5, 5, 5, '2025-08-05', '11:20:00', 'Merida',
'Xalapa', '01:50:00', 670.75);

insert into mantenimiento values
(1, 1, '2025-07-01', '2025-07-02', 'Cambio de aceite',
'Se cambi aceite y filtro'),
(2, 2, '2025-06-10', '2025-06-11', 'Frenos desgastados', '
Se cambiaron balatas delanteras'),
(3, 3, '2025-05-05', '2025-05-06', 'Problemas el ctricos',
'Revisi n de sistema el ctrico'),
(4, 4, '2025-04-12', '2025-04-14', 'Neum ticos en mal estado',
'Reemplazo de llantas'),
(5, 5, '2025-03-18', '2025-03-19', 'Bateria descargada',
'Se cambio la bateria');

insert into mecanico values
(1, 'Pedro', 'Luna', 'Garcia', '5511122233', 9500.00),
(2, 'Jorge', 'Rios', 'Martinez', '3333344455', 9800.75),
```

```
(3, 'Sofia', 'Castro', 'Mendoza', '8112233445', 9100.00),
(4, 'Andres', 'Vega', 'Santos', '2223344556', 9700.50),
(5, 'Laura', 'Jimenez', 'Ortiz', '4425566778', 9400.25);

insert into servicio values
(1, 1, 1, 'Cambio-de-Aceite', '2025-07-01'),
(2, 2, 2, 'Cambio-de-Frenos', '2025-06-10'),
(3, 3, 3, 'Revision-Electrica', '2025-05-05'),
(4, 4, 4, 'Cambio-de-Llantas', '2025-04-12'),
(5, 5, 5, 'Cambio-de-Bateria', '2025-03-18');

insert into inventario values
(1, 'Aceite-5W30', 50, 300.00),
(2, 'Filtro-de-Aceite', 40, 120.50),
(3, 'Balatas', 30, 600.00),
(4, 'Llantas-185/65R15', 20, 1500.00),
(5, 'Bateria-12V', 15, 2200.75);

insert into insumo values
(1, 1, 1, 4, 1200.00),
(2, 1, 2, 1, 120.50),
(3, 2, 3, 1, 600.00),
(4, 4, 4, 4, 6000.00),
(5, 5, 5, 1, 2200.75);

INSERT INTO propietario VALUES
(1, 'Roberto', 'Garcia', 'Mendez', '5545234307', 1),
(2, 'Sandra', 'Lopez', 'Ruiz', '3331665830', 2),
(3, 'Miguel', 'Torres', 'Vargas', '8146331677', 3),
(4, 'Elena', 'Ramirez', 'Castro', '2221359439', 4),
(5, 'Javier', 'Ortega', 'Silva', '4431034961', 5);
```

5. Fragmentos

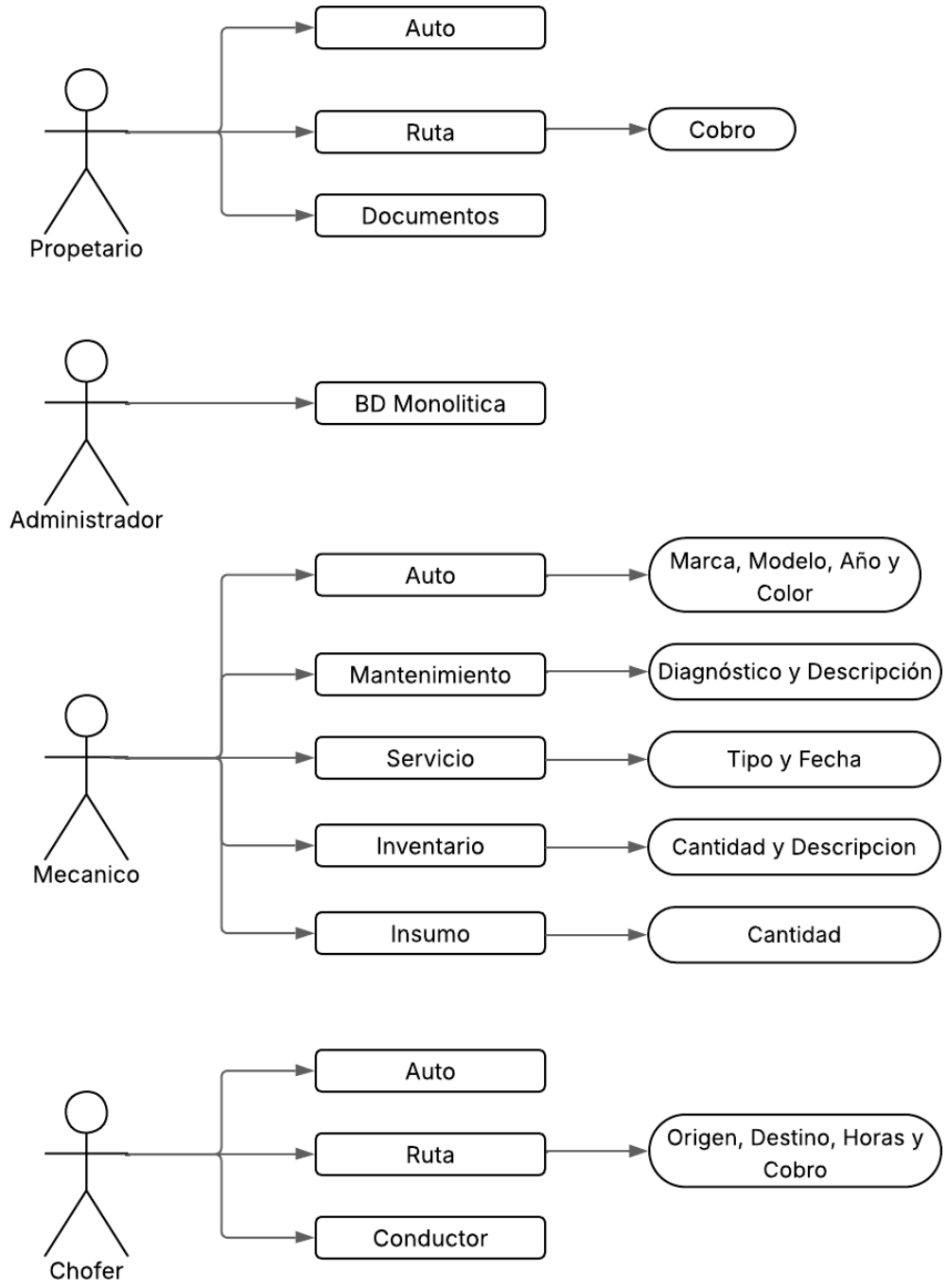

Figure 3: Esquema de los fragmentos

6. Fragmentos como BD Monolitica

- Propietario: ve solo info administrativa y financiera.
- Mecánico: ve solo info técnica y de inventario.
- Chofer: ve solo info operativa de rutas.

Listing 4: Fragmento Propietario

```
CREATE DATABASE fragmento_propietario;
USE fragmento_propietario;

CREATE TABLE auto (
    id_auto INT NOT NULL,
    marca VARCHAR(255) NOT NULL,
    modelo VARCHAR(255) NOT NULL,
    placa VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_auto)
);

CREATE TABLE documento (
    id_documento INT NOT NULL,
    id_auto INT NOT NULL,
    nombre VARCHAR(255) NOT NULL,
    inicio DATE NOT NULL,
    final DATE NOT NULL,
    monto DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_documento),
    FOREIGN KEY(id_auto) REFERENCES auto(id_auto)
);

CREATE TABLE ruta (
    id_ruta INT NOT NULL,
    id_auto INT NOT NULL,
    origen VARCHAR(255) NOT NULL,
    destino VARCHAR(255) NOT NULL,
    cobro DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_ruta),
    FOREIGN KEY(id_auto) REFERENCES auto(id_auto)
);
```

Listing 5: Fragmento Mecanico

```
CREATE DATABASE fragmento_mecanico;
USE fragmento_mecanico;

CREATE TABLE auto (
    id_auto INT NOT NULL,
    marca VARCHAR(255) NOT NULL,
    modelo VARCHAR(255) NOT NULL,
    anio INT NOT NULL,
    color VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_auto)
);

CREATE TABLE mantenimiento (
    id_mantenimiento INT NOT NULL,
    id_auto INT NOT NULL,
    diagnostico VARCHAR(255) NOT NULL,
    descripcion VARCHAR(500) NOT NULL,
    PRIMARY KEY(id_mantenimiento),
    FOREIGN KEY(id_auto) REFERENCES auto(id_auto)
);

CREATE TABLE servicio (
    id_servicio INT NOT NULL,
    id_mantenimiento INT NOT NULL,
    tipo VARCHAR(255) NOT NULL,
    fecha DATE NOT NULL,
    PRIMARY KEY(id_servicio),
    FOREIGN KEY(id_mantenimiento) REFERENCES mantenimiento(id_mantenimiento)
);

CREATE TABLE inventario (
    id_inventario INT NOT NULL,
    descripcion VARCHAR(255) NOT NULL,
    cantidad INT NOT NULL,
    precio DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_inventario)
);

CREATE TABLE insumo (
    id_insumo INT NOT NULL,
    id_servicio INT NOT NULL,
    id_inventario INT NOT NULL,
    cantidad INT NOT NULL,
    costo DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_insumo),
    FOREIGN KEY(id_servicio) REFERENCES servicio(id_servicio),
    FOREIGN KEY(id_inventario) REFERENCES inventario(id_inventario)
);
```

Listing 6: Fragmento Chofer

```
CREATE DATABASE fragmento_chofer;
USE fragmento_chofer;

CREATE TABLE auto (
    id_auto INT NOT NULL,
    marca VARCHAR(255) NOT NULL,
    modelo VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_auto)
);

CREATE TABLE conductor (
    id_conductor INT NOT NULL,
    nombre VARCHAR(255) NOT NULL,
    apellido_paterno VARCHAR(255) NOT NULL,
    apellido_materno VARCHAR(255) NOT NULL,
    PRIMARY KEY(id_conductor)
);

CREATE TABLE ruta (
    id_ruta INT NOT NULL,
    id_auto INT NOT NULL,
    id_conductor INT NOT NULL,
    origen VARCHAR(255) NOT NULL,
    destino VARCHAR(255) NOT NULL,
    hora_inicio TIME NOT NULL,
    hora_llegada TIME NOT NULL,
    cobro DECIMAL(10,2) NOT NULL,
    PRIMARY KEY(id_ruta),
    FOREIGN KEY(id_auto) REFERENCES auto(id_auto),
    FOREIGN KEY(id_conductor) REFERENCES conductor(id_conductor)
);
```

7. ETL

- Extract: leer los datos de la BD monolítica.
- Transform: Solo utilizar las columnas que necesita cada fragmento. Filtramos o seleccionamos solo las columnas y tablas wue interesan para cada fragmento.
- Load: insertar los datos y transformados en las tablas del fragmento correspondiente.

8. Conclusiones

Esta práctica permitió poner a prueba nuestras habilidades para analizar un problema aplicado a la gestión de una flotilla de autos, un tema en el que inicialmente no contábamos con conocimiento previo. Diseñamos una base de datos en MySQL para proponer una solución que facilite su administración.

La fragmentación permitió identificar cómo diferentes usuarios (propietario, mecánico, chofer) requieren subconjuntos distintos de datos, lo que confirma la utilidad de la fragmentación vertical y horizontal en sistemas distribuidos.

Referencias Bibliográficas

References

[1] Edenred, E. (2022, diciembre 2). Flotilla de autos: cómo administrarla. Edenred.mx. https://www.edenred.mx/blog/flotilla-de-autos-como-administrarla

[2] Erickson, J. (2024, August 29). MySQL: Understanding what it is and how it's used. https://www.oracle.com/mx/mysql/what-is-mysql/#: :text=MySQL

[3] Universidad, U. (2025, August 4). Overleaf: Herramienta definitiva para escribir en LaTeX online. Universidad Americana De Europa. https://unade.edu.mx/overleaf-herramienta-definitiva-para-escribir-en-latex-online/::text=Overleaf

[4] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2006). *Fundamentos de bases de datos* (5a ed., F. Sáenz Pérez, A. García Cordero & J. Correas Fernández, Trads.). McGraw-Hill/Interamericana de España.