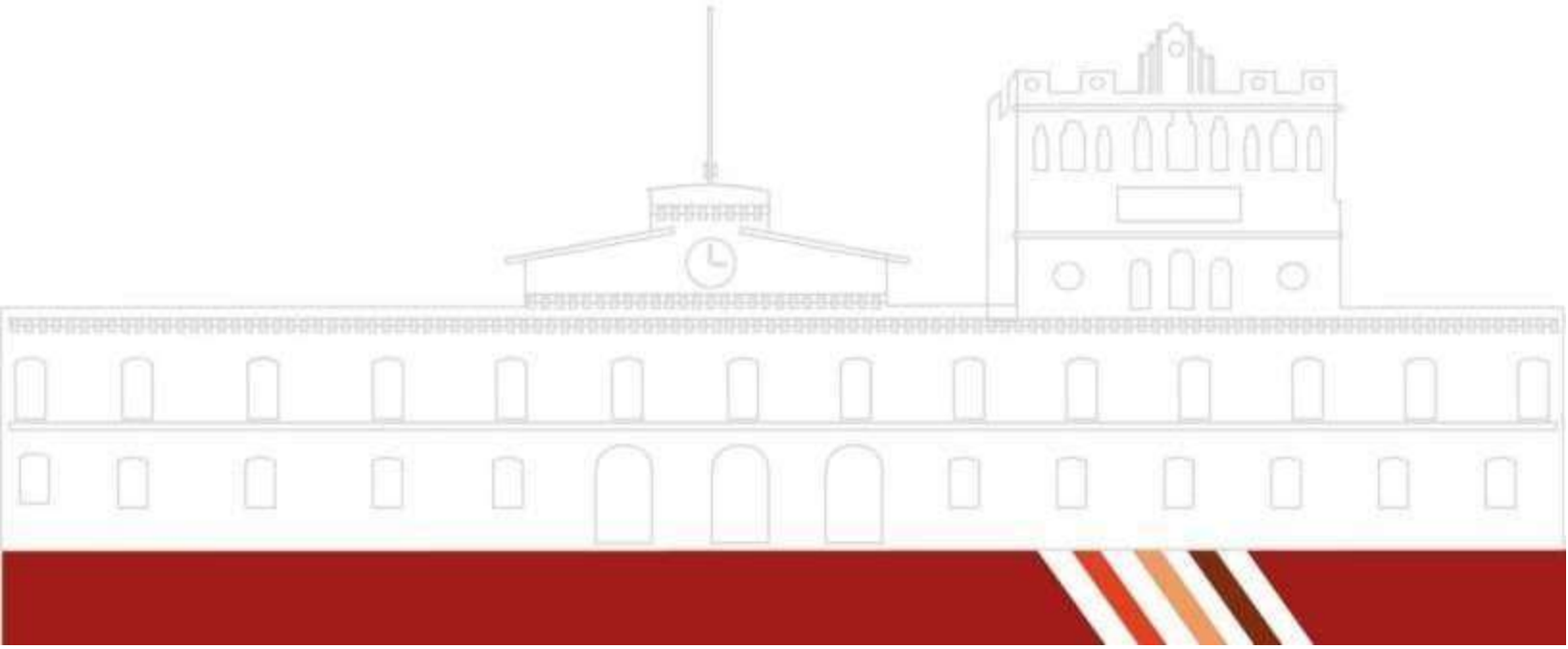
Reporte de Practica No. 2.1
Nombre de la Practica: Fragmentos
Alumno: Kevin Badillo Olmos
Dr. Eduardo Cornejo Velázquez



1. Introducción

La gestión eficiente de una flotilla vehicular constituye un elemento esencial para el éxito operativo y financiero de las organizaciones dedicadas al transporte, la logística y la distribución. Una administración ineficaz puede generar incrementos en los costos operativos, incumplimiento de normativas, y una disminución considerable en la productividad. Ante estos desafíos, la implementación de una base de datos sólida y centralizada se convierte en una herramienta estratégica que permite optimizar la toma de decisiones y el control de los recursos.

El presente reporte aborda el diseño e implementación de una base de datos relacional en **MySQL**, orientada a la **gestión integral de una flotilla de vehículos**. El propósito principal es desarrollar una solución centralizada que no solo facilite el almacenamiento y la consulta de información, sino que también asegure la **integridad, seguridad y disponibilidad de los datos** para los diferentes actores involucrados: administradores, operadores, talleres mecánicos y propietarios.

Para ello, se ha seguido una **metodología estructurada de diseño de bases de datos**, que comprende desde el levantamiento de requerimientos y el análisis del dominio del problema, hasta la elaboración de un **modelo Entidad–Relación** y su conversión a un **modelo lógico normalizado**, finalizando con la **implementación física** en MySQL. Asimismo, se incorporó un **esquema de seguridad basado en vistas SQL**, lo que permite controlar el acceso a la información de acuerdo con los roles definidos en el sistema.

2.1. Bases de Datos Relacionales

Una **base de datos relacional** es un sistema de almacenamiento estructurado que organiza la información en **tablas** compuestas por **filas** (tuplas) y **columnas** (atributos). Este modelo, propuesto por **Edgar F. Codd** en la década de 1970, se fundamenta en la **teoría de conjuntos** y la **lógica de predicados**.
Entre sus principales ventajas destacan la **integridad de los datos**, la **flexibilidad para ejecutar consultas complejas** mediante el **Lenguaje Estructurado de Consultas (SQL)**, y la **reducción de la redundancia** a través del proceso de **normalización**. Estos aspectos permiten mantener una estructura coherente, escalable y fácilmente mantenible para el manejo de grandes volúmenes de información.

2.2. MySQL

MySQL es uno de los **Sistemas de Gestión de Bases de Datos Relacionales (RDBMS)** de código abierto más utilizados a nivel mundial. Se distingue por su **fiabilidad**, **rendimiento elevado** y **facilidad de uso**, lo que lo convierte en una opción ampliamente adoptada tanto en entornos empresariales como académicos.
Al ser compatible con el estándar **SQL**, MySQL permite la implementación de **modelos relacionales robustos** que garantizan las propiedades **ACID** (Atomicidad,

Consistencia, Aislamiento y Durabilidad), esenciales para la correcta gestión de transacciones. Además, su arquitectura modular y su amplia compatibilidad con diversos sistemas operativos lo convierten en una herramienta versátil para el desarrollo de aplicaciones que requieren una gestión eficiente de datos.

2.3. Metodología de Diseño de Bases de Datos

El **diseño de bases de datos** es un proceso estructurado que permite definir la organización y las relaciones de los datos de manera lógica y eficiente. Este proceso suele dividirse en tres fases fundamentales:

1. **Diseño Conceptual:**
   En esta etapa se identifican las entidades, sus atributos y las relaciones que existen entre ellas. El resultado es un **Modelo Entidad–Relación (ERD)**, que describe de forma abstracta la estructura de los datos, sin depender del sistema de gestión que se utilizará.
2. **Diseño Lógico:**
   Se traduce el modelo conceptual en un **esquema relacional**, definiendo las **tablas**, **claves primarias (PK)** y **claves foráneas (FK)**. En esta fase se aplica el proceso de **normalización**, con el fin de garantizar la consistencia y eliminar redundancias.
3. **Diseño Físico:**
   En esta última etapa se implementan decisiones específicas del RDBMS elegido, como la **creación de índices**, la **asignación de tipos de datos** y la **configuración del almacenamiento**, optimizando el rendimiento y la eficiencia del sistema.

2.4. Normalización

La **normalización** es un procedimiento que tiene como propósito **organizar las columnas y tablas** de una base de datos para **reducir la redundancia** y **mejorar la integridad de los datos**.
Este proceso implica dividir tablas grandes en otras más pequeñas y coherentes, manteniendo relaciones adecuadas entre ellas. Las **formas normales** (1NF, 2NF, 3NF, etc.) constituyen un conjunto de reglas que orientan esta estructuración.
En el presente proyecto, el diseño se ajusta a la **Tercera Forma Normal (3NF)**, garantizando que cada atributo dependa únicamente de la clave primaria, lo cual mejora la consistencia de la información y facilita el mantenimiento del sistema.

2.5. Fragmentación y Vistas (Views)

La **fragmentación** consiste en dividir una base de datos en partes más pequeñas denominadas **fragmentos**, con el fin de mejorar su rendimiento, seguridad o accesibilidad. Puede ser de dos tipos: **horizontal** (división por filas) y **vertical** (división por columnas).
En sistemas centralizados, es posible aplicar una **fragmentación virtual** mediante el uso de **vistas SQL**. Una **vista** es una consulta almacenada que actúa como una **tabla virtual**, mostrando únicamente la información relevante para cada usuario.
Además de facilitar la organización de los datos, las vistas son un **mecanismo de**

seguridad eficaz, ya que permiten **restringir el acceso** a información sensible sin otorgar permisos directos sobre las tablas base.

2. Metodología de diseño e implementación

2.1. Levantamiento de Requisitos

Basado en el análisis del artículo "Flotilla de autos: ¿cómo administrarla de forma eficiente?" de Edenred México, se identificaron los siguientes requisitos funcionales y de datos:

- **Entidades Clave:** Vehículo, Conductor, Propietario, Taller Mecánico, Mantenimiento, Documento, Gasto y Ruta.
- **Procesos a Gestionar:**
  - Control de gastos de combustible y peajes.
  - Gestión y seguimiento de vencimientos de documentos (seguros, tenencias, etc.).
  - Planificación y registro de mantenimientos preventivos y correctivos.
  - Asignación y seguimiento de rutas a los conductores.
- **Roles y Requisitos de Acceso:**
  - **Administrador de Flotilla:** Requiere una visión global de la operación, incluyendo el estado de todos los vehículos, rutas activas, costos totales y alertas de mantenimiento o vencimientos.

  Taller Mecánico: Solo necesita acceso a los datos técnicos de los vehículos que atiende y su historial de mantenimiento, sin ver información financiera o de rutas.
  - **Operador (Chofer):** Necesita consultar las rutas que tiene asignadas, los vehículos que opera y registrar los gastos asociados.
  - **Propietario del Auto:** Requiere visibilidad sobre el estado de sus vehículos, los costos de mantenimiento y los documentos asociados.

2.2. Modelo Conceptual

Se diseñó un Modelo Entidad-Relación que representa las entidades y sus interconexiones.

[Imagen de un Diagrama Entidad-Relación para un sistema de flotillas]

- Un Propietario puede poseer uno o muchos Vehiculos.
- Un Vehiculo es asignado a una o muchas Rutas a lo largo del tiempo.
- Un Conductor es asignado a una o muchas Rutas.
- Un Vehiculo recibe uno o muchos Mantenimientos en un TallerMecanico.
- Un Vehiculo tiene asociados muchos Documentos y Gastos.

2.3. Modelo Lógico y Normalización

El modelo conceptual se tradujo en el siguiente esquema relacional, aplicando la 3NF para evitar redundancias. Por ejemplo, los datos del conductor no se repiten en cada ruta, sino que se enlazan mediante una clave foránea (idConductor).

- **Propietario** (idPropietario, nombre, rfc, direccion, telefono)
- **Vehiculo** (idVehiculo, placa, marca, modelo, anio, numeroSerie, idPropietario)
- **Conductor** (idConductor, nombre, numeroLicencia, fechaVencimientoLicencia)
- **TallerMecanico** (idTaller, nombre, direccion, telefono)
- **Mantenimiento** (idMantenimiento, idVehiculo, idTaller, fechaServicio, descripcion, costo)
- **Documento** (idDocumento, idVehiculo, tipoDocumento, fechaVencimiento)
- **Gasto** (idGasto, idVehiculo, idConductor, concepto, monto, fecha)
- **Ruta** (idRuta, idVehiculo, idConductor, origen, destino, fechaSalida, estatus)

2.4. Seguridad y Operación

La estrategia de seguridad se basa en el **principio de mínimo privilegio**. El acceso a los datos no se otorga directamente sobre las tablas base. En su lugar, se crearon vistas SQL específicas para cada rol. Los permisos de la base de datos (GRANT SELECT) se asignarán únicamente sobre estas vistas, garantizando que los usuarios solo puedan consultar la información estrictamente necesaria para sus funciones.

3. Desarrollo (MySQL)

3.1. Esquema

El siguiente código SQL DDL (Data Definition Language) crea la estructura completa de la base de datos `gestion_flotilla`.

```
CREATE DATABASE IF NOT EXISTS gestion_flotilla;
USE gestion_flotilla;

CREATE TABLE Propietario (
  idPropietario INT PRIMARY KEY AUTO_INCREMENT,
  nombre VARCHAR(100) NOT NULL,
  rfc VARCHAR(13) UNIQUE
);

CREATE TABLE Vehiculo (
  idVehiculo INT PRIMARY KEY AUTO_INCREMENT,
  placa VARCHAR(10) UNIQUE NOT NULL,
  marca VARCHAR(50),
  modelo VARCHAR(50),
  anio INT,
  numeroSerie VARCHAR(20) UNIQUE,
  idPropietario INT,
  FOREIGN KEY (idPropietario) REFERENCES Propietario(idPropietario)
);
```

3.2. Datos de Ejemplo y Consultas (Fragmentos)

Se poblaron las tablas con datos de ejemplo utilizando sentencias `INSERT INTO`. Posteriormente, se crearon las vistas para cada rol.

Fragmento del Administrador

Esta vista consolida la información operativa clave.

```
CREATE OR REPLACE VIEW Vista_Administrador AS
SELECT
    v.placa, v.marca, v.modelo,
    c.nombre AS nombreConductor,
    r.origen, r.destino, r.estatus AS estatusRuta,
    m.fechaServicio, m.tipoMantenimiento
FROM Vehiculo v
LEFT JOIN Ruta r ON v.idVehiculo = r.idVehiculo
LEFT JOIN Conductor c ON r.idConductor = c.idConductor
LEFT JOIN Mantenimiento m ON v.idVehiculo = m.idVehiculo;

-- Consulta de ejemplo:
SELECT * FROM Vista_Administrador WHERE estatusRuta = 'En Curso';
```

Fragmento del Taller Mecánico

Esta vista muestra solo la información técnica y de mantenimiento de los vehículos atendidos por un taller específico (ej. Taller con ID 1).

```
CREATE OR REPLACE VIEW Vista_TallerMecanico_1 AS
SELECT v.placa, v.marca, v.modelo, v.anio,
v.numeroSerie, m.fechaServicio, m.descripcion,
m.costo
FROM Mantenimiento m
JOIN Vehiculo v ON m.idVehiculo = v.idVehiculo
WHERE m.idTaller = 1;

-- Consulta de ejemplo:
SELECT * FROM Vista_TallerMecanico_1 ORDER BY fechaServicio DESC;
```

4. Conclusiones

Se ha **diseñado e implementado exitosamente** una **base de datos relacional en MySQL** que cumple con los requerimientos necesarios para la **gestión integral de una flotilla vehicular**. El modelo de datos, cuidadosamente **normalizado**, asegura la **integridad de la información** y **reduce la redundancia**, mientras que la **arquitectura de seguridad basada en vistas SQL** garantiza un **acceso controlado y segmentado** según los distintos perfiles de usuario.

La solución propuesta representa una **plataforma sólida, eficiente y escalable**, capaz de **centralizar la información operativa crítica**. De esta manera, contribuye a una **toma de decisiones más precisa** por parte de los administradores, **optimiza los procesos de mantenimiento y control** para conductores y talleres, y **proporciona transparencia y trazabilidad** para los propietarios de los vehículos.

Como línea de desarrollo futuro, se recomienda la **implementación de una aplicación web o móvil** que actúe como **interfaz gráfica de usuario (GUI)** para la base de datos. Esta extensión facilitaría la captura y actualización de información en tiempo real, además de permitir la **generación de dashboards interactivos y reportes visuales**, promoviendo un **análisis más dinámico e intuitivo** de los datos de la flotilla.

Bibliografía

- Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, 13(6), 377-387.

- Edenred México. (2023). *Flotilla de autos: ¿cómo administrarla de forma eficiente?*. Recuperado de https://www.edenred.mx/blog/flotilla-de-autos-como-administrarla

- MySQL Documentation. (2025). *MySQL 8.0 Reference Manual*. Oracle Corporation.