Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 2.1

Fragmentos BD Flotilla

ALUMNO:
Ariana García Melo

1. Introducción

En la actualidad, la gestión eficiente de flotillas de vehículos es un componente clave en la operación de empresas de transporte, logística y servicios. Una base de datos bien estructurada permite no solo el almacenamiento seguro de la información, sino también la optimización en la toma de decisiones estratégicas, el mantenimiento de los vehículos y el control del rendimiento operativo. Para lograr esto, es fundamental diseñar mecanismos de organización de los datos que faciliten su acceso, consulta y actualización de manera eficiente.

Dentro de las técnicas utilizadas para mejorar el rendimiento de las bases de datos se encuentran la fragmentación horizontal y la fragmentación vertical, las cuales permiten dividir la información en subconjuntos más manejables sin comprometer la integridad de los datos.

En esta práctica, se abordará la implementación de fragmentación horizontal y vertical dentro de un sistema de gestión de flotillas basado en MySQL, utilizando vistas para representar estos fragmentos de manera eficiente. Se restaurará una base de datos que almacena información clave sobre flotillas, vehículos, conductores, rutas, mantenimientos y transacciones de combustible. Posteriormente, se diseñarán y crearán cinco fragmentos horizontales y cinco fragmentos verticales, aplicando criterios de segmentación basados en las características y necesidades de la base de datos.

El objetivo principal de esta actividad es demostrar cómo la fragmentación de datos puede mejorar el rendimiento y la organización de la información dentro de un sistema de gestión de flotillas. A través de la implementación de estas técnicas, se podrá observar cómo la estructuración de la base de datos impacta en la eficiencia de las consultas, la reducción de redundancias y la optimización del acceso a los datos según las necesidades del usuario.

2. Marco teórico

La fragmentación en bases de datos es una técnica utilizada para mejorar el rendimiento y la administración de datos al dividir una base en fragmentos más pequeños, ya sea horizontalmente (por filas) o verticalmente (por columnas). Esta técnica permite optimizar el acceso a la información, mejorar la eficiencia de las consultas y reducir la carga de procesamiento sobre la base de datos centralizada [1].

La fragmentación horizontal se emplea para dividir los registros en subconjuntos con base en criterios específicos, mientras que la fragmentación vertical separa los atributos de una tabla en múltiples fragmentos más manejables. En el contexto de esta práctica, ambas estrategias se aplican utilizando vistas en MySQL, lo que facilita la segmentación lógica de los datos sin modificar la estructura física de las tablas [3].

Álgebra Relacional

El álgebra relacional es un conjunto de operaciones utilizadas para la manipulación de datos en bases relacionales. Estas operaciones incluyen selección, proyección, unión, intersección, diferencia, producto cartesiano y combinaciones [2].

En la fragmentación horizontal, la operación de selección juega un papel clave, ya que permite extraer subconjuntos de datos según condiciones específicas. De manera similar, en la fragmentación vertical, la proyección se usa para seleccionar columnas específicas.

MySQL y Sentencias Utilizadas

2.2.1 Creación de Vistas en MySQL

En MySQL, las vistas se utilizan para definir fragmentos sin modificar la estructura de la base de datos. Su sintaxis básica es:

```
1  CREATE VIEW nombre_vista AS
2  SELECT columnas
3  FROM tabla
4  WHERE condicion;
```

Listing 1: Creación de vistas.

3. Herramientas empleadas

Para la implementación de las consultas se utilizaron:

- MySQL como gestor de bases de datos.
- MySQL Workbench para la ejecución de consultas y visualización de resultados.
- LaTeX para la elaboración del reporte.
- Respaldo de base de datos: `https://github.com/idkAriana/Bases-de-Datos-Disribuidas/blob/72159c55cef163431820faba388a6cbdc5ba9c08/Respaldo_Flotilla_Views.sql`

4. Desarrollo

Restauración de Base de Datos

Restaurar una base de datos en MySQL Workbench es un proceso sencillo que permite recuperar una copia de seguridad previamente generada en formato .sql.Restaurar una base de datos en MySQL Workbench es un proceso sencillo que permite recuperar una copia de seguridad previamente generada en formato .sql. A continuación, se describen los pasos para llevar a cabo esta operación de manera efectiva:

1. Abrir MySQL Workbench y Conectarse al Servidor
2. Acceder a la Opción de Restauración.

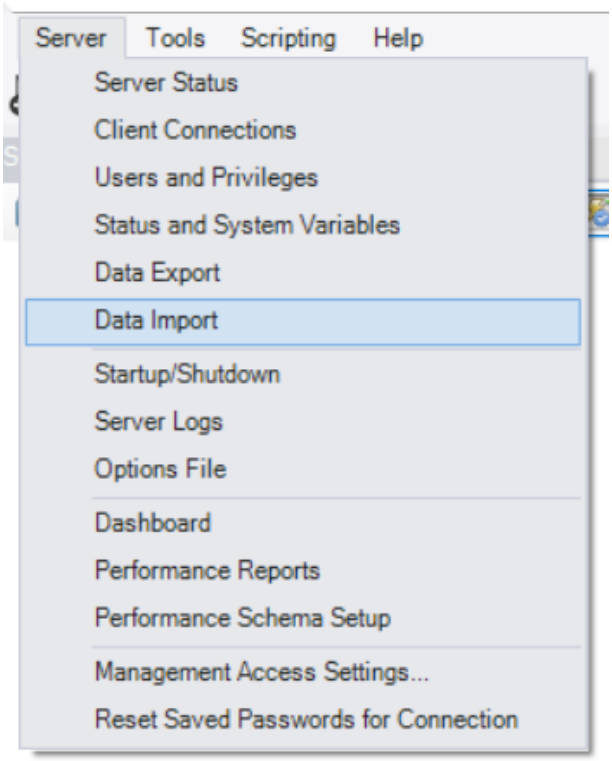

Figure 1: En la barra de menú, hacer clic en Server y seleccionar la opción Data Import.

3. Seleccionar el Archivo de RespaldO

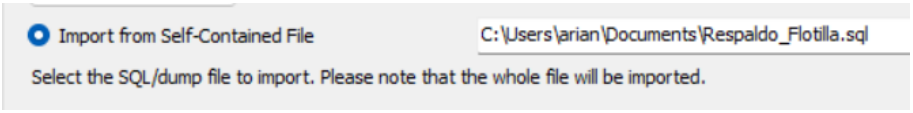

Figure 2: En la sección Import from Self-Contained File.

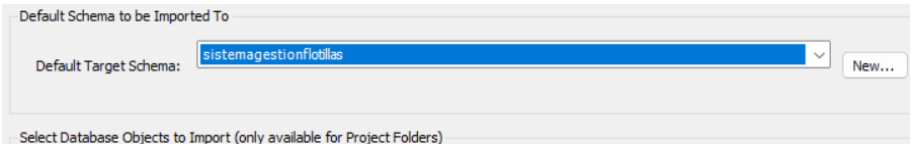

Figure 3: Elegir un esquema donde se restaurarán los datos.

4. Iniciar el Proceso de Restauración

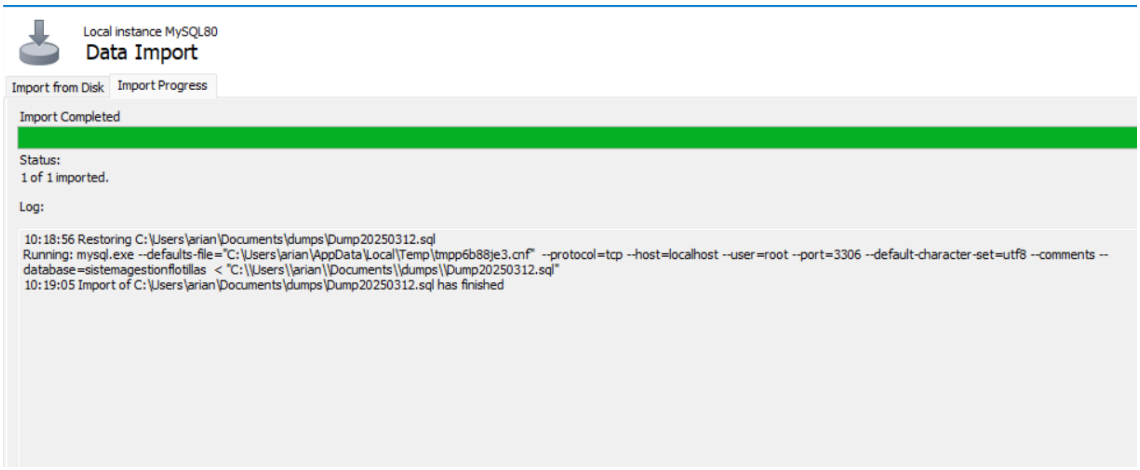

Figure 4: Hacer clic en Start Import para comenzar la restauración.

5. Verificar la Restauración

Fragmentación Horizontal

La fragmentación horizontal se basa en dividir los datos en subconjuntos según condiciones específicas.

4.2.1 Ejemplo 1

Vehículos Activos e Inactivos.

$$\text{Vehiculos\_Activos} = \sigma_{\text{estado}='Activo'}(\text{Vehiculo})$$

$$\text{Vehiculos\_Inactivos} = \sigma_{\text{estado}\neq'Activo'}(\text{Vehiculo})$$

```
CREATE VIEW Vehiculos_Activos AS
SELECT * FROM Vehiculo
WHERE estado = 'Activo';

CREATE VIEW Vehiculos_Inactivos AS
SELECT * FROM Vehiculo
WHERE estado <> 'Activo';
```

Listing 2: Sentencia SQL

4.2.2 Ejemplo 2

Conductores con Licencia Vigente y Vencida.

$$Conductores\_Vigentes = \sigma_{\text{vencimientoLicencia}\geq\text{CURDATE()}}(\text{Conductor})$$

$$Conductores\_Vencidos = \sigma_{\text{vencimientoLicencia}<\text{CURDATE()}}(\text{Conductor})$$

```sql
CREATE VIEW Conductores_Vigentes AS
SELECT * FROM Conductor
WHERE vencimientoLicencia >= CURDATE();

CREATE VIEW Conductores_Vencidos AS
SELECT * FROM Conductor
WHERE vencimientoLicencia < CURDATE();
```

Listing 3: Sentencia SQL

4.2.3 Ejemplo 3

Rutas en Curso y Completadas.

$$Rutas\_En\_Curso = \sigma_{\text{estado}='Pendiente'}(\text{Ruta})$$

$$Rutas\_Completadas = \sigma_{\text{estado}='Completado'}(\text{Ruta})$$

```sql
CREATE VIEW Rutas_En_Curso AS
SELECT * FROM Ruta
WHERE estado = 'Pendiente';

CREATE VIEW Rutas_Completadas AS
SELECT * FROM Ruta
WHERE estado = 'Completado';
```

Listing 4: Sentencia SQL

4.2.4 Ejemplo 4

Mantenimientos Realizados en 2025.

$$\pi_{\text{mantenimientoId, vehiculoId, tipoServicio, fechaServicio, costo}}(\sigma_{\text{fechaServicio}\geq'2025-01-01'\wedge\text{fechaServicio}\leq'2025-12-31'}(\text{Mantenimiento}))$$

```sql
CREATE VIEW Mantenimientos_2025 AS
SELECT * FROM Mantenimiento
WHERE fechaServicio BETWEEN '2025-01-01' AND '2025-12-31';
```

Listing 5: Sentencia SQL

4.2.5 Ejemplo 5

Transacciones de Combustible por encima de 100 pesos.

$$\pi_{\text{transaccionId, vehiculoId, conductorId, monto, cantidad, tipoCombustible, fechaTransaccion}}(\sigma_{\text{monto}>100}(\text{TransaccionCombustible}))$$

```sql
CREATE VIEW Transacciones_Mayores_100 AS
SELECT * FROM TransaccionCombustible
WHERE monto > 100;
```

Listing 6: Sentencia SQL

Fragmentación Vertical

La fragmentación vertical implica seleccionar columnas específicas para optimizar consultas.

4.3.1 Ejemplo 1

Información Básica de Vehículos.

$$\pi_{\text{vehiculoId, tipo, marca, modelo, anio}}(\text{Vehiculo})$$

```
CREATE VIEW Vehiculo_Basico AS
SELECT vehiculoId, tipo, marca, modelo, anio FROM Vehiculo;
```

Listing 7: Sentencia SQL

4.3.2 Ejemplo 2

Información de Mantenimiento con Costo.

$$\pi_{\text{mantenimientoId, vehiculoId, tipoServicio, costo}}(\text{Mantenimiento})$$

```
CREATE VIEW Mantenimiento_Costos AS
SELECT mantenimientoId, vehiculoId, tipoServicio, costo FROM Mantenimiento;
```

Listing 8: Sentencia SQL

4.3.3 Ejemplo 3

Conductores con Información de Licencia.

$$\pi_{\text{conductorId, nombre, numeroLicencia, vencimientoLicencia}}(\text{Conductor})$$

```
CREATE VIEW Conductor_Licencia AS
SELECT conductorId, nombre, numeroLicencia, vencimientoLicencia FROM Conductor;
```

Listing 9: Sentencia SQL

4.3.4 Ejemplo 4

Documentos de Vehículos sin Ruta de Archivo.

$$\pi_{\text{documentoId, vehiculoId, tipo, fechaVencimiento, estado}}(\text{Documento})$$

```
CREATE VIEW Documento_Sin_Ruta AS
SELECT documentoId, vehiculoId, tipo, fechaVencimiento, estado FROM Documento;
```

Listing 10: Sentencia SQL

4.3.5 Ejemplo 5

Información de Transacciones sin Ubicación.

$$\pi_{\text{transaccionId, vehiculoId, conductorId, monto, cantidad, tipoCombustible, fechaTransaccion}}(\text{TransaccionCombustible})$$

```
CREATE VIEW Transaccion_Simple AS
SELECT transaccionId, vehiculoId, conductorId, monto, cantidad, tipoCombustible,
    fechaTransaccion FROM TransaccionCombustible;
```

Listing 11: Sentencia SQL

5. Conclusiones

La fragmentación en bases de datos relacionales es una técnica fundamental para optimizar la administración y el rendimiento de los sistemas de información. En el caso de la gestión de flotillas, donde se manejan grandes volúmenes de datos sobre vehículos, conductores, rutas y transacciones, la implementación de fragmentación horizontal y vertical permite mejorar la eficiencia en el acceso y consulta de la información.

Desde el punto de vista teórico, el álgebra relacional proporciona una base sólida para comprender y formalizar las operaciones utilizadas en la fragmentación de bases de datos. Operaciones como la selección y la proyección permiten definir fragmentos lógicos de manera precisa antes de ser traducidos en consultas SQL. La aplicación de estos conceptos en MySQL, mediante la creación de vistas, demuestra cómo los principios teóricos pueden implementarse en un entorno práctico para optimizar el almacenamiento y la manipulación de datos.

Además, la fragmentación contribuye significativamente a la optimización de consultas en bases de datos de gran tamaño. Al dividir la información en fragmentos específicos, se reduce la cantidad de datos procesados en cada consulta, lo que disminuye la carga sobre el sistema y mejora los tiempos de respuesta. Esto es particularmente útil en sistemas distribuidos, donde la fragmentación puede utilizarse para mejorar la distribución de carga y la tolerancia a fallos.

El uso de fragmentación horizontal y vertical en bases de datos relacionales es una estrategia eficaz para mejorar la gestión de información en sistemas complejos como la administración de flotillas. La correcta aplicación de estas técnicas no solo optimiza el rendimiento de las consultas y reduce la redundancia de datos, sino que también permite una organización más eficiente de la información. La combinación de teoría, mediante el uso de álgebra relacional, y práctica, a través de la implementación en MySQL, demuestra cómo la fragmentación puede ser utilizada como una herramienta poderosa para la optimización de bases de datos en distintos contextos empresariales y tecnológicos.

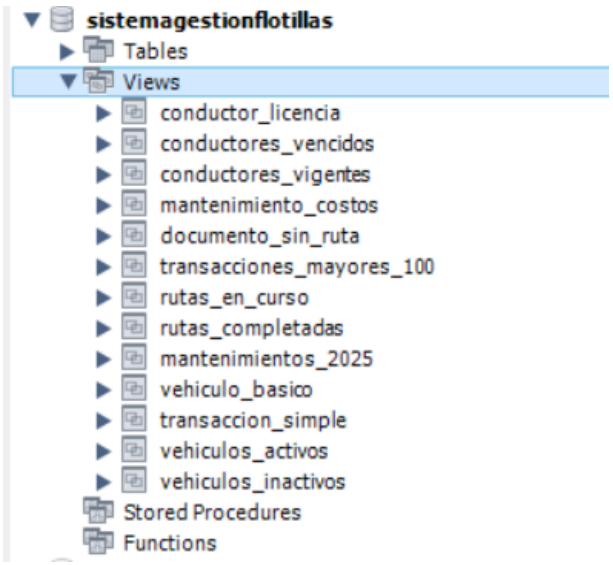

Figure 5: Resultado final

Referencias Bibliográficas

References

[1] Silberschatz, A., Korth, H. F., Sudarshan, S. (**2020**). Database System Concepts. (7th ed.). *McGraw-Hill*

[2] Elmasri, R., Navathe, S. (**2021**). Fundamentals of Database Systems. (7th ed.). *Pearson*

[3] Elmasri, R., Navathe, S. (**2015**). Database Systems: A Practical Approach to Design, Implementation, and Management. (6th ed.). *Pearson*

[4] Documentación oficial de MySQL: `https://dev.mysql.com/doc/`