REPORTE DE PRÁCTICA NO.2.1

Fragmentos

ALUMNO: Armando Oswaldo Pérez Romero
Dr. Eduardo Cornejo-Velázquez

1. Introducción

En el mundo de las bases de datos, manejar grandes cantidades de información de manera eficiente es un reto que todos enfrentamos en algún momento. Imagina tener una tabla enorme llena de datos, donde cada consulta se vuelve lenta o complicada porque hay demasiada información que procesar. Aquí es donde entra en juego una técnica llamada fragmentación, que básicamente consiste en dividir una tabla en partes más pequeñas y manejables para que todo funcione mejor.

Existen dos formas principales de hacer esta división: la fragmentación horizontal y la fragmentación vertical. La primera divide la tabla en grupos de filas, como si cortáramos una hoja de cálculo en pedazos más pequeños, cada uno con un conjunto específico de datos. La segunda, en cambio, divide la tabla por columnas, como si separáramos las hojas de un libro en capítulos independientes. Ambas técnicas tienen un objetivo común: hacer que los datos sean más fáciles de acceder, más rápidos de consultar y más sencillos de gestionar, especialmente cuando trabajamos con sistemas distribuidos, donde la información está repartida en varios servidores o nodos.

Este ejercicio no solo nos permite entender mejor cómo funcionan estas técnicas, sino también ver cómo pueden aplicarse en situaciones reales, donde el manejo de datos es clave para el éxito de cualquier sistema.

2. Marco teórico

SQL

El lenguaje de consulta estructurado (SQL, por sus siglas en inglés) es un lenguaje estándar empleado para administrar y manejar bases de datos relacionales. Mediante SQL, los usuarios tienen la posibilidad de realizar diversas operaciones en las bases de datos, tales como añadir, actualizar, eliminar y consultar información. Este lenguaje es fundamental para interactuar con sistemas de gestión de bases de datos (SGBD) relacionales, como MySQL, PostgreSQL, SQL Server y Oracle, entre otros.
SQL se compone de varios sublenguajes:

- **DDL (Data Definition Language)**: Permite definir y modificar la estructura de la base de datos, como la creación de tablas, índices y restricciones.
- **DML (Data Manipulation Language)**: Facilita la manipulación de datos, incluyendo operaciones como INSERT, UPDATE, DELETE y SELECT.
- **DCL (Data Control Language)**: Gestiona los permisos y el control de acceso a los datos.
- **TCL (Transaction Control Language)**: Controla las transacciones en la base de datos, como COMMIT y ROLLBACK.

SQL es un lenguaje declarativo, lo que significa que el usuario especifica qué datos desea obtener o modificar, y el SGBD se encarga de determinar la mejor manera de ejecutar la consulta.

Referencia: Escofet, C. M. (2002). El lenguaje SQL. UOC, la universidad virtual.

MySQL

MySQL es un sistema de gestión de bases de datos (SGBD) de código abierto ampliamente utilizado en el desarrollo de aplicaciones web, sistemas empresariales y diversas plataformas que requieren almacenamiento y manejo eficiente de datos. Se destaca por su rapidez, fiabilidad y facilidad de uso, además de ser compatible con múltiples sistemas operativos, como Windows, Linux y macOS.
Algunas características clave de MySQL incluyen:

- Soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).
- Escalabilidad y alto rendimiento, incluso con grandes volúmenes de datos.
- Amplia compatibilidad con lenguajes de programación como PHP, Python, Java y otros.
- Herramientas de administración gráfica, como MySQL Workbench, que facilitan la gestión de bases de datos.

MySQL es utilizado por empresas y organizaciones de todo el mundo debido a su flexibilidad y robustez. Además, cuenta con una comunidad activa que contribuye a su desarrollo y mejora continua.

Referencia: Santillán, L. A. C., Ginestà, M. G., Mora, Ó. P. (2014). Bases de datos en MySQL. Universitat oberta de Catalunya.

Fragmentación Horizontal en Bases de Datos Distribuidas

La fragmentación horizontal es una técnica utilizada en bases de datos distribuidas que consiste en dividir una tabla en subconjuntos de filas (tuplas) basándose en condiciones específicas. Cada fragmento contiene un subconjunto de filas que cumplen con un predicado definido. Esta técnica es útil para distribuir los datos en diferentes nodos de una red, mejorando así el rendimiento y la disponibilidad del sistema.
Existen dos tipos de fragmentación horizontal:

- **Fragmentación Horizontal Primaria**: Se realiza sobre una sola tabla, dividiéndola en fragmentos basados en condiciones aplicadas a sus atributos.
- **Fragmentación Horizontal Derivada**: Se basa en la fragmentación de una tabla relacionada con otra, utilizando claves foráneas para definir los fragmentos.

La fragmentación horizontal permite optimizar las consultas al reducir la cantidad de datos que deben ser accedidos en cada nodo, lo que mejora la eficiencia del sistema.

Referencia: Özsu, M. T., Valduriez, P. (2020). Principles of Distributed Database Systems. Springer.

Fragmentación Vertical en Bases de Datos Distribuidas

La fragmentación vertical es otra técnica utilizada en bases de datos distribuidas, que consiste en dividir una tabla en subconjuntos de columnas (atributos). Cada fragmento contiene un subconjunto de columnas de la tabla original, lo que permite distribuir los datos de manera más eficiente según las necesidades de acceso y consulta.
Algunas ventajas de la fragmentación vertical incluyen:

- Reducción del ancho de las tablas, lo que puede mejorar el rendimiento de las consultas que acceden a un subconjunto específico de columnas.
- Mayor control sobre la privacidad y seguridad de los datos, ya que se pueden almacenar columnas sensibles en nodos separados.
- Optimización del almacenamiento al evitar la redundancia de datos no necesarios en ciertos nodos.

La fragmentación vertical es especialmente útil en entornos donde diferentes aplicaciones o usuarios necesitan acceder a diferentes subconjuntos de atributos de una tabla.

Referencia: Özsu, M. T., Valduriez, P. (2020). Principles of Distributed Database Systems. Springer.

3. Herramientas Utilizadas

MySQL Workbench

MySQL Workbench es una herramienta visual de diseño y administración de bases de datos que facilita la creación, modificación y gestión de bases de datos MySQL. Ofrece un entorno integrado para diseñar esquemas de bases de datos, escribir consultas SQL, administrar usuarios y realizar tareas de mantenimiento. En esta práctica, se utilizó MySQL Workbench para implementar y probar la fragmentación vertical y horizontal, así como para ejecutar consultas y visualizar los resultados de manera eficiente.

Overleaf (Editor LaTeX Online)

Overleaf es un editor en línea basado en LaTeX que permite crear, editar y compartir documentos académicos, técnicos y científicos de alta calidad. Su interfaz intuitiva y su capacidad para compilar documentos en tiempo real lo convierten en una herramienta ideal para la redacción de informes, artículos y presentaciones. En este reporte, se utilizó Overleaf para la elaboración del documento, aprovechando sus plantillas y funcionalidades para dar formato al contenido de manera profesional.

4. Desarrollo

1. Conductores Activos e Inactivos

Esta sección crea vistas para separar a los conductores en dos grupos: activos e inactivos.

- Conductores Activos: $\sigma_{\text{estado}'Activo'}(\text{Conductor})$
- Conductores Inactivos: $\sigma_{\text{estado}'Inactivo'}(\text{Conductor})$

```

-- Vista para Conductores Activos
CREATE VIEW ConductoresActivos AS
SELECT * FROM Conductor WHERE estado = 'Activo';

-- Vista para Conductores Inactivos
CREATE VIEW ConductoresInactivos AS
SELECT * FROM Conductor WHERE estado = 'Inactivo';

-- Consulta para Conductores Activos
SELECT * FROM ConductoresActivos;

-- Consulta para Conductores Inactivos
SELECT * FROM ConductoresInactivos;
```

2. Vehículos en Ruta y en Mantenimiento

Esta sección crea vistas para separar los vehículos en dos grupos: aquellos en ruta y aquellos en mantenimiento.

- Vehículos en Ruta: $\sigma_{\text{estado}='EnRuta'}(\text{Vehiculo})$
- Vehículos en Mantenimiento: $\sigma_{\text{estado}='Mantenimiento'}(\text{Vehiculo})$

```
-- Vista para Veh culos en Ruta
CREATE VIEW VehiculosEnRuta AS
SELECT * FROM Vehiculo WHERE estado = 'En Ruta';

-- Vista para Veh culos en Mantenimiento
CREATE VIEW VehiculosEnMantenimiento AS
SELECT * FROM Vehiculo WHERE estado = 'Mantenimiento';

-- Consulta para Vehiculos en Ruta
SELECT * FROM VehiculosEnRuta;

-- Consulta para Vehiculos en Mantenimiento
SELECT * FROM VehiculosEnMantenimiento;
```

3. Documentos Vigentes y Vencidos

Esta sección crea vistas para separar los documentos en dos grupos: vigentes y vencidos.

- Documentos Vigentes: $\sigma_{\text{estado}='Vigente'}(\text{Documento})$
- Documentos Vencidos: $\sigma_{\text{estado}='Vencido'}(\text{Documento})$

```sql
-- Vista para Documentos Vigentes
CREATE VIEW DocumentosVigentes AS
SELECT * FROM Documento WHERE estado = 'Vigente';

-- Vista para Documentos Vencidos
CREATE VIEW DocumentosVencidos AS
SELECT * FROM Documento WHERE estado = 'Vencido';

-- Consulta para Documentos Vigentes
SELECT * FROM DocumentosVigentes;

-- Consulta para Documentos Vencidos
SELECT * FROM DocumentosVencidos;
```

4. Mantenimientos Completados y Pendientes

Esta sección crea vistas para separar los mantenimientos en dos grupos: completados y pendientes.

- Mantenimientos Completados: $\sigma_{\text{estado}='Completado'}(\text{Mantenimiento})$
- Mantenimientos Pendientes: $\sigma_{\text{estado}='Pendiente'}(\text{Mantenimiento})$

```sql
-- Vista para Mantenimientos Completados
CREATE VIEW MantenimientosCompletados AS
SELECT * FROM Mantenimiento WHERE estado = 'Completado';

-- Vista para Mantenimientos Pendientes
CREATE VIEW MantenimientosPendientes AS
SELECT * FROM Mantenimiento WHERE estado = 'Pendiente';

-- Consulta para Mantenimientos Completados
SELECT * FROM MantenimientosCompletados;

-- Consulta para Mantenimientos Pendientes
SELECT * FROM MantenimientosPendientes;
```

5. Rutas Completadas y en Curso

Esta sección crea vistas para separar las rutas en dos grupos: completadas y en curso.

- Rutas Completadas: $\sigma_{\text{estado}='Completada'}(\text{Ruta})$
- Rutas en Curso: $\sigma_{\text{estado}='EnCurso'}(\text{Ruta})$

```sql
-- Vista para Rutas Completadas
CREATE VIEW RutasCompletadas AS
SELECT * FROM Ruta WHERE estado = 'Completada';

-- Vista para Rutas en Curso
CREATE VIEW RutasEnCurso AS
SELECT * FROM Ruta WHERE estado = 'En Curso';

-- Consulta para Rutas Completadas
SELECT * FROM RutasCompletadas;

-- Consulta para Rutas en Curso
SELECT * FROM RutasEnCurso;
```

6. Conductores con sus Vehículos Asignados

Esta sección crea una vista que combina información de conductores y vehículos asignados.

$$\pi_{\text{Conductor.conductorId, Conductor.nombre, Conductor.numeroLicencia, Vehiculo.vehiculoId, Vehiculo.marca, Vehiculo.modelo}}$$
$$(\text{Conductor} \bowtie_{\text{Conductor.conductorId} = \text{Ruta.conductorId}} \text{Ruta} \bowtie_{\text{Ruta.vehiculoId} = \text{Vehiculo.vehiculoId}} \text{Vehiculo})$$

```sql
-- Crear la vista
CREATE VIEW ConductoresConVehiculos AS
SELECT
    c.conductorId,
    c.nombre AS nombreConductor,
    c.numeroLicencia,
    v.vehiculoId,
    v.marca,
    v.modelo
FROM Conductor c
JOIN Ruta r ON c.conductorId = r.conductorId
JOIN Vehiculo v ON r.vehiculoId = v.vehiculoId;

-- Consultar la vista
SELECT * FROM ConductoresConVehiculos;
```

7. Flotillas con sus Vehículos Asociados

Esta sección crea una vista que combina información de flotillas y vehículos asociados.

$$\pi_{\text{Flotilla.flotillaId, Flotilla.nombreEmpresa, Vehiculo.vehiculoId, Vehiculo.marca, Vehiculo.modelo, Vehiculo.anio}}$$
$$(\text{Flotilla} \bowtie_{\text{Flotilla.flotillaId} = \text{Vehiculo.flotillaId}} \text{Vehiculo})$$

```sql
-- Crear la vista
CREATE VIEW FlotillasConVehiculos AS
SELECT
    f.flotillaId,
    f.nombreEmpresa,
    v.vehiculoId,
    v.marca,
    v.modelo,
    v.anio
FROM Flotilla f
JOIN Vehiculo v ON f.flotillaId = v.flotillaId;

-- Consultar la vista
SELECT * FROM FlotillasConVehiculos;
```

8. Vehículos con sus Documentos Asociados

Esta sección crea una vista que combina información de vehículos y documentos asociados.

$$\pi_{\text{Vehiculo.vehiculoId, Vehiculo.marca, Vehiculo.modelo, Documento.documentoId, Documento.tipo, Documento.fechaVencimiento}}$$

$$(\text{Vehiculo} \bowtie_{\text{Vehiculo.vehiculoId} = \text{Documento.vehiculoId}} \text{Documento})$$

```sql
-- Crear la vista
CREATE VIEW VehiculosConDocumentos AS
SELECT
    v.vehiculoId,
    v.marca,
    v.modelo,
    d.documentoId,
    d.tipo AS tipoDocumento,
    d.fechaVencimiento
FROM Vehiculo v
JOIN Documento d ON v.vehiculoId = d.vehiculoId;

-- Consultar la vista
SELECT * FROM VehiculosConDocumentos;
```

9. Mantenimientos con Detalles del Vehículo

Esta sección crea una vista que combina información de mantenimientos y detalles del vehículo.

$$\pi_{\text{Mantenimiento.mantenimientoId, Mantenimiento.fechaServicio, Mantenimiento.tipoServicio, Mantenimiento.costo, Vehiculo.vehiculoId, Vehiculo.marca, V}}$$

$$(\text{Mantenimiento} \bowtie_{\text{Mantenimiento.vehiculoId} = \text{Vehiculo.vehiculoId}} \text{Vehiculo})$$

```sql
-- Crear la vista
CREATE VIEW MantenimientosConVehiculos AS
SELECT
    m.mantenimientoId,
    m.fechaServicio,
    m.tipoServicio,
    m.costo,
    v.vehiculoId,
    v.marca,
    v.modelo
FROM Mantenimiento m
JOIN Vehiculo v ON m.vehiculoId = v.vehiculoId;

-- Consultar la vista
SELECT * FROM MantenimientosConVehiculos;
```

10. Transacciones de Combustible con Detalles del Conductor y Vehículo

Esta sección crea una vista que combina información de transacciones de combustible, conductores y vehículos.

$$\pi_{\text{TransaccionCombustible.transaccionId, TransaccionCombustible.fechaTransaccion, TransaccionCombustible.cantidad, TransaccionCombustible.tipoCombu}}$$

$$(\text{TransaccionCombustible} \bowtie_{\text{TransaccionCombustible.conductorId} = \text{Conductor.conductorId}} \text{Conductor} \bowtie_{\text{TransaccionCombustible.vehiculoId} = \text{V}}$$

```sql
-- Crear la vista
CREATE VIEW TransaccionesCombustibleConDetalles AS
SELECT
    t.transaccionId,
    t.fechaTransaccion,
    t.cantidad,
    t.tipoCombustible,
    c.conductorId,
    c.nombre AS nombreConductor,
    v.vehiculoId,
    v.marca,
    v.modelo
FROM TransaccionCombustible t
JOIN Conductor c ON t.conductorId = c.conductorId
JOIN Vehiculo v ON t.vehiculoId = v.vehiculoId;

-- Consultar la vista
SELECT * FROM TransaccionesCombustibleConDetalles;
```

5. Conclusiones

En esta práctica, se exploraron dos conceptos clave en el diseño y manejo de bases de datos: la fragmentación horizontal y la fragmentación vertical. A través de la implementación de vistas en SQL, se logró dividir y organizar los datos de manera eficiente, lo que permite una mejor gestión y acceso a la información almacenada. La fragmentación horizontal permitió dividir los datos en subconjuntos basados en condiciones específicas, como el estado de los conductores (activos e inactivos), el estado de los vehículos (en ruta o en mantenimiento) o el estado de los documentos (vigentes o vencidos), optimizando así las consultas y mejorando el rendimiento de la base de datos.
Respaldo: https://github.com/Osw-do/Bases-de-Datos-Distribuidas/blob/56e74307f2e5f5649c293207167a8c3e278cd72f/Siste

Referencias Bibliográficas

References

[1] Grabowska, S.; Saniuk, S. (**2022**). Business models in the industry 4.0 environment—results of web of science bibliometric analysis. *J. Open Innov. Technol. Mark. Complex*, *8(1)*, 19.

[2] Elmasri, R.; Navathe, S. B. (**2016**). *Fundamentals of Database Systems* (7th ed.). Pearson.

[3] Silberschatz, A.; Korth, H. F.; Sudarshan, S. (**2019**). *Database System Concepts* (7th ed.). McGraw-Hill Education.

[4] Codd, E. F. (**1970**). A relational model of data for large shared data banks. *Communications of the ACM*, *13(6)*, 377–387. https://doi.org/10.1145/362384.362685

[5] Date, C. J. (**2003**). *An Introduction to Database Systems* (8th ed.). Addison-Wesley.

[6] Garcia-Molina, H.; Ullman, J. D.; Widom, J. (**2008**). *Database Systems: The Complete Book* (2nd ed.). Pearson.

[7] Ramakrishnan, R.; Gehrke, J. (**2003**). *Database Management Systems* (3rd ed.). McGraw-Hill.