Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 2.1

FRAGMENTOS

ALUMNO:Israel Campos Vázquez
Dr. Eduardo Cornejo-Velázquez

1. Introducción

En bases de datos distribuidas, la fragmentación es una técnica utilizada para dividir y almacenar datos en múltiples nodos con el objetivo de mejorar el rendimiento, la disponibilidad y la escalabilidad del sistema. En lugar de almacenar toda la base de datos en un solo lugar, los datos se dividen en fragmentos más pequeños que pueden ser manejados de manera independiente.

Esta técnica es crucial en entornos distribuidos, ya que permite optimizar las consultas al almacenar los datos cerca de los usuarios que los necesitan con más frecuencia, reducir la latencia y minimizar la cantidad de datos que deben transferirse por la red.

Tipos de Fragmentación que abordaremos:

Fragmentación Horizontal: Divide las tablas en subconjuntos de filas, manteniendo las mismas columnas en cada fragmento. Cada fragmento contiene un conjunto específico de registros que cumplen ciertas condiciones, como la ubicación geográfica de los clientes. Ejemplo: Una tabla de clientes puede fragmentarse en función del país en el que residen.

Fragmentación Vertical: Divide las tablas en subconjuntos de columnas, manteniendo la clave primaria en cada fragmento. Es útil cuando diferentes aplicaciones acceden a distintos subconjuntos de datos. Ejemplo: Una tabla de empleados podría dividirse en un fragmento con información personal (nombre, dirección) y otro con datos laborales (salario, departamento).

2. Marco teórico

Fragmentación Vertical

La fragmentación vertical divide una tabla en subconjuntos de columnas, manteniendo en cada fragmento la clave primaria para garantizar la integridad de los datos. Se utiliza cuando diferentes aplicaciones o usuarios requieren acceder a distintas partes de la información de una misma entidad sin necesidad de cargar todas sus columnas.

Ventajas de la fragmentación vertical

1. Reduce el uso de memoria y mejora la eficiencia de consultas que solo necesitan ciertas columnas.
2. Optimiza el rendimiento al reducir la cantidad de datos transferidos por la red.
3. Permite almacenar fragmentos en distintos nodos según su uso, favoreciendo la escalabilidad.

Esta fragmentación evita cargar datos innecesarios en cada consulta, optimizando el rendimiento del sistema.

Como indican Elmasri y Navathe (2015)[1], "la fragmentación vertical mejora la eficiencia al reducir la cantidad de atributos accedidos en cada consulta, permitiendo una distribución óptima de la carga de trabajo en sistemas distribuidos".

Fragmentación Horizontal

La fragmentación horizontal divide una tabla en subconjuntos de registros (filas) con la misma estructura de columnas. Cada fragmento contiene un conjunto específico de filas que cumplen ciertas condiciones, como una región geográfica o un rango de valores. Esta técnica es especialmente útil cuando diferentes ubicaciones requieren acceder a subconjuntos específicos de datos sin necesidad de consultar toda la base de datos.

Ventajas de la fragmentación horizontal

1. Reduce la carga en cada nodo, ya que las consultas pueden ejecutarse en fragmentos más pequeños.
2. Optimiza el acceso a datos locales, disminuyendo la latencia de las consultas.
3. Mejora la disponibilidad, ya que si un nodo falla, otros fragmentos pueden seguir operando.

Según Özsu y Valduriez (2020)[2], "la fragmentación horizontal se basa en la distribución de filas a diferentes nodos de acuerdo con un criterio lógico, mejorando el rendimiento al reducir la cantidad de datos procesados por cada consulta".

3. Herramientas empleadas

1. DataGrip: Es un entorno de desarrollo integrado (IDE) para bases de datos creado por JetBrains. Está diseñado para ayudar a los desarrolladores y administradores de bases de datos a gestionar, consultar y optimizar sus bases de datos de manera eficiente.

2. MySQL Server: Es un sistema de gestión de bases de datos relacionales. Utiliza el lenguaje SQL para la administración y manipulación de datos. Se emplea para almacenar, organizar y gestionar información dependiendo el contexto en el que se utilice.

   Se suele utilizar para desarrollar aplicaciones web, gestionar datos de empresas, análisis de datos, entre otros.

4. Desarrollo

Importación de datos

Listing 1: Import data

```
mysql -u root -p fragmentacion < "Flotillas.sql"
```

Fragmentación Horizontal

Listing 2: Rutas.

```
CREATE VIEW frag1Rutas AS
SELECT * FROM Ruta
WHERE distancia > 500;

CREATE VIEW frag2Rutas AS
SELECT * FROM Ruta
WHERE distancia <= 500;

SELECT * FROM frag1Rutas
UNION
SELECT * FROM frag2Rutas ORDER BY rutaId ASC;
```

Listing 3: Conductor.

```
CREATE VIEW frag1Conductor AS
SELECT * FROM Conductor
WHERE estado = 'Activo';

CREATE VIEW frag2Conductor AS
SELECT * FROM Conductor
WHERE estado = 'Inactivo';

SELECT * FROM frag1Conductor
UNION
SELECT * FROM frag2Conductor ORDER BY nombre ASC;
```

Listing 4: Vehiculo.

```
CREATE VIEW frag1Vehiculo AS
SELECT * FROM Vehiculo
WHERE anio > 2020;

CREATE VIEW frag2Vehiculo AS
SELECT * FROM Vehiculo
WHERE anio <= 2020;

SELECT * FROM frag1Vehiculo
UNION
SELECT * FROM frag2Vehiculo;
```

Listing 5: Documento.

```
CREATE VIEW frag1Documento AS
SELECT * FROM Documento
WHERE estado = 'Vigente';

CREATE VIEW frag2Documento AS
```

```
SELECT * FROM Documento
WHERE estado = 'Vencido';

CREATE VIEW frag3Documento AS
SELECT * FROM Documento
WHERE estado = 'Por Vencer';

SELECT * FROM frag1Documento
UNION
SELECT * FROM frag2Documento
UNION
SELECT * FROM frag3Documento ORDER BY fechaVencimiento ASC;
```

Listing 6: Mantenimiento.

```
CREATE VIEW frag1Mantenimiento AS
SELECT * FROM Mantenimiento
WHERE costo < 1500;

CREATE VIEW frag2Mantenimiento AS
SELECT * FROM Mantenimiento
WHERE costo >= 1500;

SELECT * FROM frag1Mantenimiento
UNION
SELECT * FROM frag2Mantenimiento ORDER BY fechaServicio ASC;
```

Fragmentación Vertical

Listing 7: RutaVehiculo.

```
SELECT distinct r.ubicacionInicio,r.ubicacionFin,r.estado,r.distancia,v.modelo FROM
JOIN Vehiculo AS v on v.vehiculoId = r.vehiculoiD;

select * from rutaVehiculo;
```

Listing 8: RutaConductor.

```
SELECT distinct r.ubicacionInicio,r.ubicacionFin,r.estado,r.distancia,c.conductorId
JOIN Conductor AS c on c.conductorId = r.conductorId;

select * from rutaConductor;
```

Listing 9: VehiculoMantenimiento.

```
SELECT distinct m.fechaServicio,m.descripcion,m.costo,v.modelo FROM Mantenimiento AS
JOIN Vehiculo AS v on v.vehiculoId = m.vehiculoId;

select * from vehiculoMantenimiento;
```

Listing 10: VehiculoDocumento.

```
SELECT distinct d.fechaVencimiento,d.tipo,v.modelo FROM Documento AS d
JOIN Vehiculo AS v on v.vehiculoId = d.vehiculoId;

select * from vehiculoDocumento;
```

Listing 11: VehiculoCombustible.

```
SELECT distinct c.tipoCombustible,c.cantidad,c.monto,v.modelo FROM TransaccionCombus
JOIN Vehiculo AS v on v.vehiculoId= c.vehiculoId;

select * from vehiculoCombustible;
```

5. Conclusiones

Al realizar esta práctica, aprendí a ver la información fragmentada de manera vertical y de manera horizonta, aunque fue en forma de vistas, pero me sirvió para darme una idea de en lo que consiste la fragmentación.

Además, puse en practica el crear vistas y haces consultas de tipo select utilizando union o join para presentar la información fragmentada de manera conjunta.

Referencias Bibliográficas

References

[1] Elmasri, R., Navathe, S. B. (2015). Fundamentals of Database Systems (7th ed.). Pearson.

[2] Özsu, M. T., Valduriez, P. (2020). Principles of Distributed Database Systems (4th ed.). Springer.