Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 2.4

3 nodos BDD Flotillas

ALUMNO:
Daniel Monroy Garnica

1. Introducción

Esta practica trata de como es la creación y el uso de los LCS para distintas tareas dentro de diferentes bases de datos pero que al tener la misma información se puede hacer consultas desde diferentes bases de datos, ademas del uso de procedimientos almacenados y triggers para que cuando se modifique en LCS principal se pueda modificar también en los otros 2 nodos.

2. Marco teórico

Fragmentación vertical

La fragmentación vertical es una técnica utilizada en el diseño de bases de datos distribuidas para mejorar el rendimiento y la eficiencia en el acceso a los datos. A diferencia de la fragmentación horizontal, que divide los datos en filas, la fragmentación vertical divide los datos en columnas. Esto significa que cada fragmento contiene un conjunto específico de columnas de una tabla. La complejidad de la fragmentación vertical radica en el gran número de posibles combinaciones de columnas que pueden formar un fragmento, lo que la hace más complicada que la fragmentación horizontal.
Para abordar esta complejidad, se han desarrollado métodos que utilizan matrices de atracción entre atributos, en lugar de matrices de afinidad, para determinar cómo agrupar las columnas de manera óptima. Estos métodos también pueden incluir técnicas de agrupamiento jerárquico y reglas de decisión basadas en la homogeneidad interna y la heterogeneidad externa de los grupos obtenidos. Además, estas técnicas pueden ser adaptadas para evaluar la calidad de las particiones resultantes.

Procesos ETL

Los procesos ETL (Extract, Transform, and Load) son fundamentales en el diseño y mantenimiento de almacenes de datos. Estos procesos se encargan de extraer datos de diversas fuentes, transformarlos en un formato adecuado para el análisis, y cargarlos en el almacén de datos. Debido a su importancia, es crucial evaluar la calidad de estos procesos desde las primeras etapas de desarrollo para evitar errores que puedan afectar la toma de decisiones basada en datos incorrectos.
Para evaluar la calidad de los procesos ETL, se han propuesto medidas que permiten medir la complejidad estructural de los modelos de procesos ETL a nivel conceptual. Estas medidas pueden ayudar a los diseñadores a predecir el esfuerzo necesario para mantener estos procesos. Además, se utilizan herramientas como los diagramas de actividades UML para modelar los procesos ETL y marcos como FMESP para validar las medidas propuestas.

SELECT + INTO OUTFILE

Escribe las filas resultantes en un archivo y permite el uso de terminadores de columna y fila para especificar un formato de salida específico. El valor predeterminado es terminar los campos con tabulaciones y las líneas con saltos de línea.
El archivo no debe existir. No se puede sobrescribir. El usuario necesita el privilegio FILE para ejecutar esta instrucción.

LOAD

La instrucción lee filas de un archivo de texto en una tabla a gran velocidad. El archivo puede leerse desde el host del servidor o del cliente, según se LOCAL indique el modificador. LOCAL Esto también afecta la interpretación de datos y la gestión de errores.
LOAD DATA Es el complemento de SELECT ... INTO OUTFILE. Para escribir datos de una tabla a un archivo, utilice SELECT ... INTO OUTFILE. Para volver a leer el archivo en una tabla, utilice LOAD DATA. La sintaxis de las cláusulas ‘ FIELDS“ y “ LINESes la misma para ambas instrucciones.

SELECT con tablas de dos bases de datos

Para realizar una consulta SELECT que involucre tablas de dos bases de datos diferentes, se pueden utilizar varias estrategias dependiendo del sistema de gestión de bases de datos que esté utilizando. Una forma común es utilizar consultas distribuidas o federadas, que permiten acceder a datos en diferentes bases de datos como si estuvieran en una sola base de datos.
Para realizar una consulta SELECT con INNER JOIN entre tablas de dos bases de datos diferentes, es necesario utilizar una referencia explícita a ambas bases de datos en la sintaxis SQL. Esto permite combinar datos de forma relacional, incluso si las tablas están almacenadas en esquemas o bases de datos separadas.

3. Herramientas empleadas

Describir qué herramientas se han utilizado...

1. MySQL Workbench es una herramienta visual y un entorno integrado de desarrollo diseñado para trabajar con bases de datos MySQL. Desarrollado por Oracle, este software se ha convertido en la elección preferida de desarrolladores y administradores de bases de datos gracias a su interfaz intuitiva y sus robustas funcionalidades.

2. CMD es una aplicación que se encuentra en los sistemas operativos Windows. En términos técnicos, el símbolo del sistema es un intérprete de línea de comandos, y su propósito es permitirle introducir comandos utilizando una sintaxis especial. Los comandos que se envían al símbolo del sistema, se introducen como líneas de texto, que son ejecutadas por el sistema operativo en cuanto se pulsa "Intro" en el teclado.

4. Desarrollo

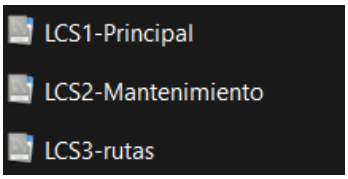
Respaldos de la GCS para crear los LCS

Lo primero que se va a hacer es un respaldo de las tablas que se vana a usar para cada uno de los nodos, desde el CMD se va a hacer estos respaldos para poder especificar que tablas se van a usar.

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqldump -u root -p sistemagestionflotillas flotilla vehiculo documento > C:\mysql\LCS1-Principal.sql
Enter password: ********
```

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqldump -u root -p sistemagestionflotillas vehiculo mantenimiento> C:\mysql\LCS2-Mantenimiento.sql
Enter password: ********
```

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqldump -u root -p sistemagestionflotillas vehiculo conductor ruta transac
cioncombustible > C:\mysql\LCS3-rutas.sql
```



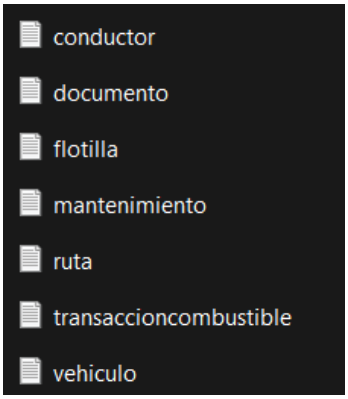
Después de eso se modifica en unas partes el script que salio para que se pueda crear el nodo de cada base de datos sin sus registros. Luego en el CMD se pone el comando para restaurar cada una de las bases de datos de los nodos.

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p si < C:\mysql\LCS1-Principal.sql
Enter password: ********

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p si < C:\mysql\LCS2-Mantenimiento.sql
Enter password: ********

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p si < C:\mysql\LCS3-Rutas.sql
Enter password: ********
```

Extracción de datos

Para poder extraer los datos del GCS se usa el select + into outfile para poder tener un archivo txt todos los registros dentro de la GCS.

```
use sistemagestionflotillas;
select * from flotilla into outfile '/mysql/flotilla.txt' FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
select * from conductor into outfile '/mysql/conductor.txt' FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
select * from documento into outfile '/mysql/documento.txt' FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
select * from mantenimiento into outfile '/mysql/mantenimiento.txt' FIELDS TERMINATED
BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
select * from ruta into outfile '/mysql/ruta.txt' FIELDS TERMINATED BY ',' OPTIONALLY
ENCLOSED BY '"' LINES TERMINATED BY '\n';
select * from transaccioncombustible into outfile '/mysql/transaccioncombustible.txt'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
select * from vehiculo into outfile '/mysql/vehiculo.txt' FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
```



Carga de datos

En este caso tuve que usar el comando LOAD para cargar los datos en cada uno de los nodos según las tablas que tenían, ademas tuve que usar un comando para desactivar el uso de claves foráneas ya que al no existir en algunos nodos otras tablas no me dejaba cargar esa información por las llaves foráneas.

```
use lcs1_principal;
load data infile '/mysql/flotilla.txt' into table flotilla FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
load data infile '/mysql/vehiculo.txt' into table vehiculo FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
load data infile '/mysql/documento.txt' into table documento FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';

SET FOREIGN_KEY_CHECKS = 0;

use lcs2_mantenimiento;
load data infile '/mysql/vehiculo.txt' into table vehiculo FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
load data infile '/mysql/mantenimiento.txt' into table mantenimiento FIELDS TERMINATED
BY ',' OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';

use lcs3_rutas;
load data infile '/mysql/vehiculo.txt' into table vehiculo FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
load data infile '/mysql/conductor.txt' into table conductor FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
load data infile '/mysql/ruta.txt' into table ruta FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' LINES TERMINATED BY '\n';
load data infile '/mysql/transaccioncombustible.txt' into table
transaccioncombustible FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Consulta a dos BD

En esta parte hice la consulta por medio de inner join para poder juntar por tablas de diferentes bases de datos y así crear consultas eficientes.

1. En esta consulta use la LCS1-Principal y la LCS2-Mantenimiento para poder ver a que flotillas les pertenecia cada vehiculo.

```
select A.flotillaId ,A.nombreEmpresa, B.tipo, B.modelo from
lcs1_principal.flotilla A INNER JOIN lcs2_mantenimiento.vehiculo B ON
A.flotillaId = B.flotillaId order by flotillaId asc;
```

flotillaId	nombreEmpresa	tipo	modelo
1	Transportes Logística	Pickup	Freightliner Cascadia
3	Transportes Del Norte	SUV	Isuzu NPR
4	Transportes Del Este	Sedán	Peterbilt 579
4	Transportes Del Este	Tráiler	Kenworth T680
5	Transportes Pachuca	SUV	Ram 1500
5	Transportes Pachuca	Furgón	Mack Anthem
6	Transportes Carga	Pickup	Ram 1500
8	Transportes México	Pickup	Volvo VNL
8	Transportes México	Pickup	Chevrolet Silverado
10	Transportes Del Sur	Camión	Freightliner M2
12	Transportes Rápida	Pickup	Western Star 5700
13	Transportes Hidalgo	SUV	Chevrolet Silverado
14	Transportes Nacional	SUV	International LT
16	Transportes Nacional	Sedán	Chevrolet Silverado
16	Transportes Nacional	Tráiler	Chevrolet Silverado
17	Transportes Del Sur	Camión	Kenworth T680
17	Transportes Del Sur	Pickup	Peterbilt 579
20	Transportes Urbana	Sedán	Ford Transit
21	Transportes Águila	Camión	Mack Anthem
23	Transportes Del Oeste	Pickup	Volvo VNL
24	Transportes Pachuca	SUV	International LT
24	Transportes Pachuca	Pickup	Mack Anthem
26	Transportes Económica	Pickup	Chevrolet Silverado

2. En esta consulta use los nodos LCS1-Principal y LCS3-Rutas para ver que conductor usa cada auto y las rutas que ha tenido

```
select A.conductorId, A.nombre, B.estado,B.UbicacionFin,C.modelo from
lcs3_rutas.ruta B inner join lcs1_principal.vehiculo C on
B.vehiculoId = C.vehiculoId inner join lcs3_rutas.conductor A on
A.conductorId = B.conductorId order by conductorId;
```

conductorId	nombre	estado	UbicacionFin	modelo
2	Patricia Vázquez	Pendiente	Cancún, QR	Mack Anthem
2	Patricia Vázquez	Completada	Guadalajara, Jalisco	Mack Anthem
2	Patricia Vázquez	En Curso	Querétaro, Qro	Isuzu NPR
4	Andrés Díaz	En Curso	Puebla, Pue	Ford Transit
5	Fernanda Díaz	Pendiente	Ciudad de México	Kenworth T680
6	Carmen López	En Curso	Guadalajara, Jalisco	Isuzu NPR
6	Carmen López	Pendiente	Querétaro, Qro	Volvo VNL
7	Francisco Martínez	En Curso	Tijuana, BC	Mack Anthem
8	Patricia Flores	Pendiente	Guadalajara, Jalisco	International LT
9	Laura Sánchez	En Curso	Puebla, Pue	Volvo VNL
10	Pedro Hernández	Completada	Pachuca, Hidalgo	Peterbilt 579
12	Carmen Castillo	En Curso	Tijuana, BC	Kenworth T680
12	Carmen Castillo	Completada	Querétaro, Qro	International LT
13	Miguel Vázquez	Completada	Ciudad de México	Volvo VNL
13	Miguel Vázquez	En Curso	Monterrey, NL	Kenworth T680
13	Miguel Vázquez	Completada	Cancún, QR	Peterbilt 579
14	Carlos Hernández	En Curso	Ciudad de México	Volvo VNL
14	Carlos Hernández	En Curso	Hermosillo, Son	Kenworth T680
14	Carlos Hernández	Pendiente	Pachuca, Hidalgo	Western Star 5...
15	Carmen Ortiz	En Curso	Hermosillo, Son	Peterbilt 579
16	Patricia Mendoza	Completada	Monterrey, NL	Isuzu NPR
17	Jorge García	Completada	Mérida, Yuc	Hino 338
17	Jorge García	En Curso	Guadalajara, Jalisco	Kenworth T680

Procedimientos almacenados y triggers

Para esta parte cree procedimientos y triggers los cuales se van a usar para cuando se vaya a insertar, actualizar o borrar un registro en la principal lo cual se hará primero con los procedimientos y después de que se haya realizado esa acción se va a activar alguno de los triggers para informar a los otros dos nodos.

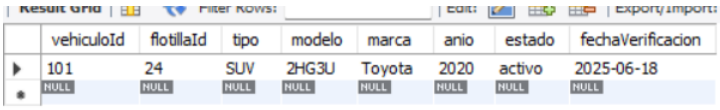
```
-- ////INSERT
DELIMITER $$
create procedure registroVehiculo(
        in vehid int,
    in floid int,
    in tip varchar(80),
    in model varchar(80),
    in mar varchar(80),
    in ani int,
    in est varchar(80),
    in feve date
)
begin
    insert into vehiculo(vehiculoid,flotillaid,tipo,modelo,marca,anio,estado,
    fechaVerificacion)
    values (vehid,floid,tip,model,mar,ani,est,feve);
end $$
DELIMITER ;

DELIMITER $$
create trigger insertLCS2 after insert on lcs1_principal.vehiculo for each row
begin
    insert into lcs2_mantenimiento.vehiculo(vehiculoid,flotillaid,tipo,modelo,marca,
    anio,estado,fechaVerificacion)
        values(new.vehiculoid,new.flotillaid,new.tipo,new.modelo,new.marca,new.anio,
    new.estado,new.fechaVerificacion);
end $$
DELIMITER ;

DELIMITER $$
create trigger insertLCS3 after insert on lcs1_principal.vehiculo for each row
begin
    insert into lcs3_rutas.vehiculo(vehiculoid,flotillaid,tipo,modelo,marca,anio,
    estado,fechaVerificacion)
        values(new.vehiculoid,new.flotillaid,new.tipo,new.modelo,new.marca,new.anio,
    new.estado,new.fechaVerificacion);
end $$
DELIMITER ;


-- /////UPDATE
DELIMITER $$
CREATE PROCEDURE actualizarFechaVeri(
    IN vehid INT,
    IN feve date
)
BEGIN
    UPDATE lcs1_principal.vehiculo
    SET
```

```
50          fechaVerificacion = feve
51      WHERE vehiculoid = vehid;
52  END $$
53  DELIMITER ;
54
55  DELIMITER $$
56  CREATE TRIGGER updateLCS2
57  AFTER UPDATE ON lcs1_principal.vehiculo
58  FOR EACH ROW
59  BEGIN
60      UPDATE lcs2_mantenimiento.vehiculo
61      SET
62          fechaVerificacion = NEW.fechaVerificacion
63      WHERE vehiculoid = NEW.vehiculoid;
64  END $$
65  DELIMITER ;
66
67  DELIMITER $$
68  CREATE TRIGGER updateLCS3
69  AFTER UPDATE ON lcs1_principal.vehiculo
70  FOR EACH ROW
71  BEGIN
72      UPDATE lcs3_rutas.vehiculo
73      SET
74          fechaVerificacion = NEW.fechaVerificacion
75      WHERE vehiculoid = NEW.vehiculoid;
76  END $$
77  DELIMITER ;
78
79
80  -- /////DELETE
81  DELIMITER $$
82  CREATE PROCEDURE eliminarVehiculo(
83      IN vehid INT
84  )
85  BEGIN
86      DELETE FROM lcs1_principal.vehiculo WHERE vehiculoid = vehid;
87  END $$
88  DELIMITER ;
89
90  DELIMITER $$
91  CREATE TRIGGER deleteLCS2
92  AFTER DELETE ON lcs1_principal.vehiculo
93  FOR EACH ROW
94  BEGIN
95      DELETE FROM lcs2_mantenimiento.vehiculo WHERE vehiculoid = OLD.vehiculoid;
96  END $$
97  DELIMITER ;
98
99  DELIMITER $$
100 CREATE TRIGGER deleteLCS3
101 AFTER DELETE ON lcs1_principal.vehiculo
102 FOR EACH ROW
103 BEGIN
```

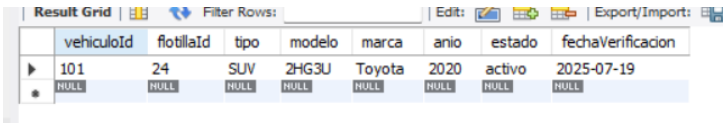
```
    DELETE FROM lcs3_rutas.vehiculo WHERE vehiculoid = OLD.vehiculoid;
END $$
DELIMITER ;
```

Teniendo todo eso, se llama cada uno de los procedimientos dentro del LCS1 para poder probar que si funciona, y para saber si funciona o no los triggers se puede usar una consulta especifica para encontrar el registro en cada uno de los nodos.

```
call registroVehiculo(101,24,'SUV','2HG3U','Toyota',2020,'activo','2025-06-18');
call actualizarFechaVeri(101,'2025-07-19');
call eliminarVehiculo(101);

-- Consulta para comprobar en los otros nodos lo que pasa con este registro
select * from vehiculo where vehiculoId = 101;
```



vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
101	24	SUV	2HG3U	Toyota	2020	activo	2025-06-18
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



Result Grid | Filter Rows: | Edit: | Export/Import:

vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
101	24	SUV	2HG3U	Toyota	2020	activo	2025-07-19
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Filter Rows: | Edit: | Export/Import:

vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

5. Conclusiones

En conclusión, el uso de los LCS es muy practico si lo aplicamos para cuando queremos usar BDD y se quiere simular como es que se pueden usar, ademas aprendí que el proceso ETL es muy eficiente para cuando se quiere traspasar la información de una base de datos a otra.

Referencias Bibliográficas

References

[1] MySQL:: MySQL 8.4 Reference Manual.(**s.f.**). *https://dev.mysql.com/doc/refman/8.4/en/*

[2] Morffi, A.R., & Ortiz, Y.V.(**2012**). Un método para la fragmentación vertical de bases de datos y su variante como evaluador de particiones.

[3] Muñnoz, L., Pardillo, J., Mazón, J., & Trujillo, J.(**2009**). Definición y validación de medidas para procesos ETL en almacenes de datos.