Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO. 2.4

Fragmentación Vertical y Procesos ETL en MySQL

ALUMNO: Jesus Eduardo Cornejo-Clavel
Dr. Eduardo Cornejo-Velázquez

1. Introducción

La práctica tiene como objetivo implementar una arquitectura distribuida de bases de datos mediante la fragmentación vertical, creando tres nodos (LCS1-Principal, LCS2-Mantenimiento, LCS3-Rutas) y realizando procesos ETL para extraer, transformar y cargar datos entre ellos. Además, se diseñarán consultas que involucren tablas de dos bases de datos nodded.

2. Marco teórico

Fragmentación Vertical

La fragmentación vertical consiste en dividir las tablas de una base de datos en columnas, creando tablas más especializadas. Esto improve la escalabilidad y el rendimiento al reducir la cantidad de datos leídos o escritos en cada operación.

Procesos ETL

Los procesos ETL (Extracción, Transformación y Carga) son esenciales para mover y transformar datos entre sistemas. En cette práctica, se usarán:

- **EXTRACT (Extracción):** Usar SELECT INTO OUTFILE para extraer datos de una base de datos.
- **TRANSFORM (Transformación):** Procesar los datos extraídos (si es necesario).
- **LOAD (Carga):** Usar LOAD DATA INFILE para cargar datos en otra base de datos.

SELECT + INTO FILE

El comando SELECT INTO OUTFILE se usa para exportar datos de una tabla a un archivo en formato plano, útil para la extracción en procesos ETL.

LOAD

El comando LOAD DATA INFILE permite cargar datos desde un archivo en formato plano a una tabla en MySQL, útil para la carga en procesos ETL.

SELECT con tablas de dos bases de datos

Se pueden realizar consultas que involucran tablas de diferentes bases de datos utilizando su nombre qualifications qualify (basePathName tableName).

3. Esquemas Conceptuales Locales

Nodo LCS1-Principal

Tablas incluidas:

- flotilla
- vehiculo
- documento

Diagrama ER:

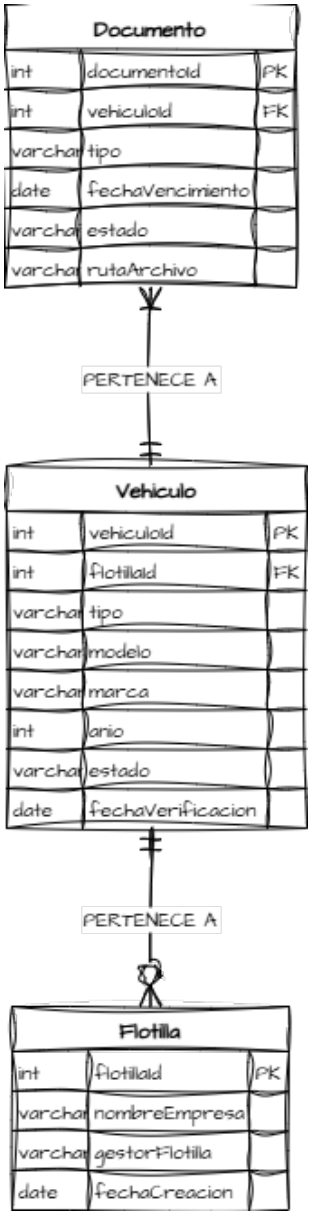

Figure 1: Diagrama ER del Nodo LCS1-Principal.

Nodo LCS2-Mantenimiento

Tablas incluidas:

- vehiculo
- mantenimiento

Diagrama ER:

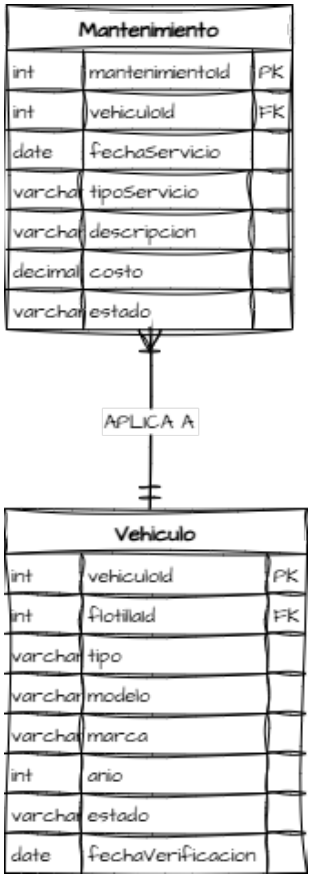

Figure 2: Diagrama ER del Nodo LCS2-Mantenimiento.

Nodo LCS3-Rutas

Tablas incluidas:

- vehiculo
- conductor
- ruta
- transaccionCombustible

Diagrama ER:

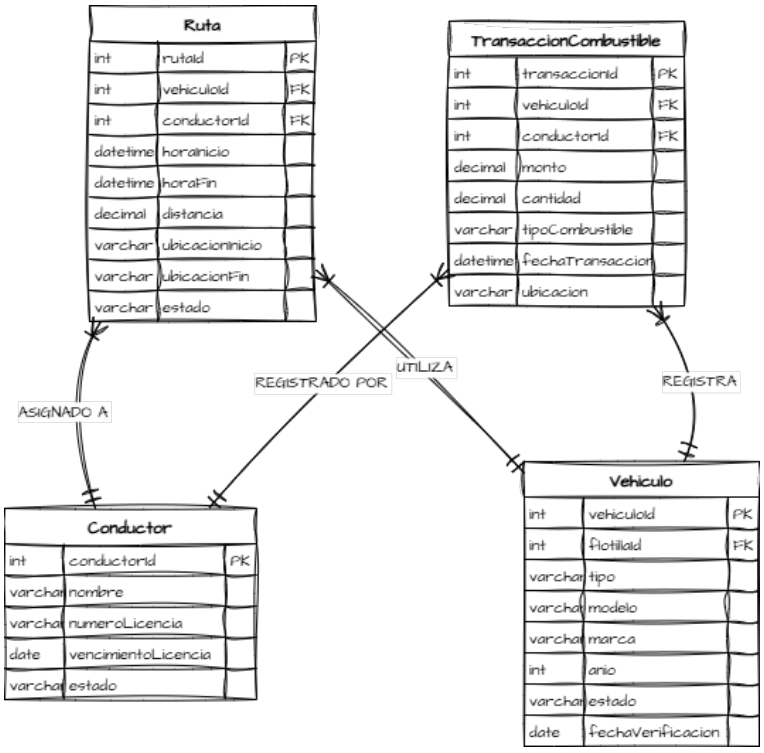

Figure 3: Diagrama ER del Nodo LCS3-Rutas.

4. Script de Creación de Nodos

Creación de Bases de Datos

Listing 1: Creación de las bases de datos

```
-- Crear nodos
CREATE DATABASE LCS1_Principal;
CREATE DATABASE LCS2_Mantenimiento;
CREATE DATABASE LCS3_Rutas;
```

Creación de Tablas en cada Nodo

4.2.1 Nodo LCS1-Principal (LCS1_Principal)

Listing 2: Creación de tablas en LCS1-Principal

```
USE LCS1_Principal;

CREATE TABLE Flotilla (
    flotillaId INT AUTO_INCREMENT PRIMARY KEY,
    nombreEmpresa VARCHAR(100) NOT NULL,
    gestorFlotilla VARCHAR(100),
    fechaCreacion DATE
);

CREATE TABLE Vehiculo (
```

```
    vehiculoId INT AUTO_INCREMENT PRIMARY KEY,
    flotillaId INT,
    tipo VARCHAR(50) NOT NULL,
    modelo VARCHAR(50) NOT NULL,
    marca VARCHAR(50) NOT NULL,
    anio INT NOT NULL,
    estado VARCHAR(20),
    fechaVerificacion DATE,
    FOREIGN KEY (flotillaId) REFERENCES Flotilla(flotillaId)
);

CREATE TABLE Documento (
    documentoId INT AUTO_INCREMENT PRIMARY KEY,
    vehiculoId INT,
    tipo VARCHAR(50) NOT NULL,
    fechaVencimiento DATE NOT NULL,
    estado VARCHAR(20),
    rutaArchivo VARCHAR(255),
    FOREIGN KEY (vehiculoId) REFERENCES Vehiculo(vehiculoId)
);
```

4.2.2 Nodo LCS2-Mantenimiento (LCS2_Mantenimiento)

Listing 3: Creación de tablas en LCS2-Mantenimiento

```
USE LCS2_Mantenimiento;

CREATE TABLE Vehiculo (
    vehiculoId INT PRIMARY KEY,
    flotillaId INT,
    tipo VARCHAR(50),
    modelo VARCHAR(50),
    marca VARCHAR(50),
    anio INT,
    estado VARCHAR(20),
    fechaVerificacion DATE,
    FOREIGN KEY (flotillaId) REFERENCES LCS1_Principal.Flotilla(flotillaId)
);

CREATE TABLE Mantenimiento (
    mantenimientoId INT AUTO_INCREMENT PRIMARY KEY,
    vehiculoId INT,
    fechaServicio DATE NOT NULL,
    tipoServicio VARCHAR(100) NOT NULL,
    descripcion VARCHAR(200),
    costo DECIMAL(10,2) NOT NULL,
    estado VARCHAR(20),
    FOREIGN KEY (vehiculoId) REFERENCES Vehiculo(vehiculoId)
);
```

4.2.3 Nodo LCS3-Rutas (LCS3_Rutas)

Listing 4: Creación de tablas en LCS3-Rutas

```
USE LCS3_Rutas;

CREATE TABLE Vehiculo (
    vehiculoId INT PRIMARY KEY,
    flotillaId INT,
    tipo VARCHAR(50),
    modelo VARCHAR(50),
    marca VARCHAR(50),
    anio INT,
    estado VARCHAR(20),
    fechaVerificacion DATE,
    FOREIGN KEY (flotillaId) REFERENCES LCS1_Principal.Flotilla(flotillaId)
);

CREATE TABLE Conductor (
    conductorId INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    numeroLicencia VARCHAR(50) NOT NULL,
    vencimientoLicencia DATE NOT NULL,
    estado VARCHAR(20)
);

CREATE TABLE Ruta (
    rutaId INT AUTO_INCREMENT PRIMARY KEY,
    vehiculoId INT,
    conductorId INT,
    horaInicio DATETIME NOT NULL,
    horaFin DATETIME,
    distancia DECIMAL(10,2),
    ubicacionInicio VARCHAR(100) NOT NULL,
    ubicacionFin VARCHAR(100) NOT NULL,
    estado VARCHAR(20),
    FOREIGN KEY (vehiculoId) REFERENCES Vehiculo(vehiculoId),
    FOREIGN KEY (conductorId) REFERENCES Conductor(conductorId)
);

CREATE TABLE TransaccionCombustible (
    transaccionId INT AUTO_INCREMENT PRIMARY KEY,
    vehiculoId INT,
    conductorId INT,
    monto DECIMAL(10,2) NOT NULL,
    cantidad DECIMAL(10,2) NOT NULL,
    tipoCombustible VARCHAR(20) NOT NULL,
    fechaTransaccion DATETIME NOT NULL,
    ubicacion VARCHAR(100),
    FOREIGN KEY (vehiculoId) REFERENCES Vehiculo(vehiculoId),
    FOREIGN KEY (conductorId) REFERENCES Conductor(conductorId)
);
```

5. Scripts de Extracción y Carga de Datos

Extracción de Datos (SELECT INTO OUTFILE)

Listing 5: Ejemplo de extracción de datos

```
-- Ejecutar en LCS2_Mantenimiento
SELECT mantenimientoId, vehiculoId, fechaServicio, tipoServicio, descripcion, costo, estado
INTO OUTFILE '/tmp/mantenimiento_data.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Carga de Datos (LOAD DATA INFILE)

Listing 6: Ejemplo de carga de datos

```
-- Ejecutar en LCS2_Mantenimiento
LOAD DATA INFILE '/tmp/mantenimiento_data.csv' INTO TABLE Mantenimiento
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

6. Script de Consulta de Datos

Consulta entre LCS1-Principal y LCS3-Rutas

Listing 7: Consulta entre bases LCS1-Principal y LCS3-Rutas

```
SELECT
    Vehiculo.vehiculoId,
    Vehiculo.modelo,
    Ruta.rutaId,
    Ruta.horaInicio,
    Ruta.ubicacionInicio
FROM
    LCS1_Principal.Vehiculo
JOIN
    LCS3_Rutas.Ruta ON Vehiculo.vehiculoId = Ruta.vehiculoId;
```

Consulta entre LCS2-Mantenimiento y LCS1-Principal

Listing 8: Consulta entre bases LCS2-Mantenimiento y LCS1-Principal

```
SELECT
    Mantenimiento.mantenimientoId,
    Vehiculo.modelo,
    Mantenimiento.fechaServicio,
    Mantenimiento.tipoServicio
FROM
    LCS2_Mantenimiento.Mantenimiento
JOIN
    LCS1_Principal.Vehiculo ON Mantenimiento.vehiculoId = Vehiculo.vehiculoId;
```

7. Conclusiones

En esta práctica se logró:

- Crear una arquitectura distribuida con tres nodos: LCS1-Principal, LCS2-Mantenimiento y LCS3-Rutas.
- Implementar fragmentación vertical para distribuir las tablas entre los diferentes nodos.
- Establecer las relaciones entre tablas de diferentes bases de datos.
- Desarrollar procesos ETL para la extracción y carga de datos entre nodos.
- Crear consultas que involucran tablas de diferentes bases de datos.

La implementación de esta arquitectura distribuida permite mejorar el rendimiento y la escalabilidad del sistema, ya que cada nodo puede especializarse en un tipo de operación específica.

Referencias Bibliográficas

References

[1] Sommerville, I. (2011). *Software Engineering*. Pearson.

[2] Elmasri, R.; Navathe, S. (2016). *Fundamentos de Sistemas de Bases de Datos*. Pearson Educación.

[3] Date, C. J. (2011). *SQL y las bases de datos*. McGraw-Hill.