Universidad Autónoma del Estado de Hidalgo

ICBI

REPORTE DE PRÁCTICA NO.2.4

3 Nodos Bases de Datos Distribuidas

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

Introducción

En el ámbito de la gestión de datos, las bases de datos distribuidas (BDD) representan un enfoque fundamental para manejar grandes volúmenes de información provenientes de múltiples fuentes. Este tipo de sistemas permite almacenar y gestionar datos en varios nodos interconectados, facilitando la escalabilidad, la disponibilidad y la eficiencia en el acceso a la información. En este contexto, la extracción, importación y uso de tablas provenientes de distintas bases de datos se convierte en una tarea esencial para integrar y consolidar datos dispersos en un entorno unificado.
En esta practica se llevaron a cabo procesos de extracción, transformación y carga (ETL) para integrar tablas de tres bases de datos distintas. Estas bases de datos, que pueden diferir en estructura, formato y ubicación, representan un desafío en términos de interoperabilidad y consistencia de datos. A través de técnicas como la fragmentación vertical, así como el uso de herramientas y lenguajes de consulta especializados, se logró consolidar la información en un único entorno para su posterior análisis y explotación.
El objetivo principal de esta práctica es demostrar cómo se pueden integrar y gestionar datos heterogéneos en una base de datos distribuida.

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 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:
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 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.

Procesos ETL (Extract, Transform, Load)

Los procesos ETL (Extracción, Transformación y Carga) son fundamentales en el ámbito de la integración de datos y el almacenamiento en data warehouses. Estos procesos permiten mover datos desde múltiples fuentes heterogéneas a un repositorio centralizado, donde pueden ser utilizados para análisis y reporting.

- **Extracción (Extract):** En esta fase, los datos se recopilan de diversas fuentes, como bases de datos relacionales, archivos planos, APIs, o sistemas legacy. La extracción puede ser completa (todos los datos) o incremental (solo los cambios desde la última extracción).

- **Transformación (Transform):** Los datos extraídos se someten a una serie de operaciones para limpiarlos, normalizarlos y estructurarlos según los requisitos del destino. Esto incluye la eliminación de duplicados, la conversión de formatos, la aplicación de reglas de negocio y la agregación de datos.
- **Carga (Load):** Finalmente, los datos transformados se cargan en el sistema de destino, que puede ser un data warehouse, un data lake o una base de datos analítica. La carga puede ser en modo *full refresh* (sobrescribir todos los datos) o *incremental* (agregar solo los nuevos datos).

Los procesos ETL son esenciales para garantizar la calidad, consistencia y disponibilidad de los datos en entornos empresariales. **Referencia:** Kimball, R., Ross, M. (2013). The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling. Wiley.

SELECT + INTO FILE en Bases de Datos

La sentencia `SELECT + INTO FILE` es una operación común en bases de datos que permite exportar el resultado de una consulta a un archivo externo. Esta funcionalidad es útil para generar informes, compartir datos con otros sistemas o realizar copias de seguridad parciales.

- **Sintaxis básica:** La estructura típica de esta operación es:

```
SELECT columnas
INTO OUTFILE 'ruta_del_archivo'
FROM tabla
WHERE condiciones;
```

- **Formatos soportados:** Dependiendo del sistema de gestión de bases de datos (DBMS), el archivo generado puede estar en formatos como CSV, TXT o incluso binario.
- **Aplicaciones comunes:** Esta operación se utiliza en escenarios como la exportación de datos para análisis externos, la migración de datos entre sistemas o la creación de archivos de configuración.

Es importante tener en cuenta los permisos de escritura en el servidor y las restricciones de seguridad al utilizar esta operación. **Referencia:** Elmasri, R., Navathe, S. B. (2016). Fundamentals of Database Systems. Pearson.

LOAD en Bases de Datos

La operación `LOAD` se utiliza para cargar datos desde un archivo externo a una tabla en una base de datos. Esta operación es común en procesos de integración de datos, migraciones y actualizaciones masivas.

- **Sintaxis básica:** La estructura típica de esta operación es:

```
LOAD DATA INFILE 'ruta_del_archivo'
INTO TABLE nombre_tabla
FIELDS TERMINATED BY 'delimitador'
LINES TERMINATED BY 'delimitador_de_linea';
```

- **Formatos soportados:** Los archivos de entrada suelen estar en formato CSV o TXT, aunque algunos DBMS admiten otros formatos.
- **Aplicaciones comunes:** Esta operación es útil en escenarios como la carga inicial de datos en un sistema, la actualización masiva de registros o la integración de datos desde fuentes externas.

Es fundamental asegurarse de que el formato del archivo coincida con la estructura de la tabla y de que se manejen adecuadamente los errores durante la carga. **Referencia:** Date, C. J. (2004). An Introduction to Database Systems. Addison-Wesley.

3. Herramientas empleadas

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

Scripts de creación de nodos

```sql
CREATE DATABASE LCS1_Principal;
USE LCS1_Principal;

CREATE TABLE flotilla (
    flotillaId INT PRIMARY KEY,
    nombreEmpresa VARCHAR(100),
    gestorFlotilla VARCHAR(100),
    fechaCreacion DATE
);

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
);

CREATE TABLE documento (
    documentoId INT PRIMARY KEY,
    vehiculoId INT,
    tipo VARCHAR(50),
    fechaVencimiento DATE,
    estado VARCHAR(20),
    rutaArchivo VARCHAR(255)
);
/*-------------------------------------------------------------------------------*/

CREATE DATABASE LCS2_Mantenimiento;
USE LCS2_Mantenimiento;

CREATE TABLE vehiculo (
    vehiculoId INT PRIMARY KEY,
    estado VARCHAR(20),
    fechaVerificacion DATE
);

CREATE TABLE mantenimiento (
    mantenimientoId INT PRIMARY KEY,
    vehiculoId INT,
    fechaServicio DATE,
    tipoServicio VARCHAR(100),
    descripcion VARCHAR(200),
    costo DECIMAL(10, 2),
    estado VARCHAR(20)
);
/*-------------------------------------------------------------------------*/

CREATE DATABASE LCS3_Rutas;
USE LCS3_Rutas;

CREATE TABLE vehiculo (
    vehiculoId INT PRIMARY KEY,
    tipo VARCHAR(50),
    modelo VARCHAR(50),
    marca VARCHAR(50),
    anio INT
);

CREATE TABLE conductor (
```

```
    conductorId INT PRIMARY KEY,
    nombre VARCHAR(100),
    numeroLicencia VARCHAR(50),
    vencimientoLicencia DATE,
    estado VARCHAR(20)
);

CREATE TABLE ruta (
    rutaId INT PRIMARY KEY,
    vehiculoId INT,
    conductorId INT,
    horaInicio DATETIME,
    horaFin DATETIME,
    distancia DECIMAL(10, 2),
    ubicacionInicio VARCHAR(100),
    ubicacionFin VARCHAR(100),
    estado VARCHAR(20)
);

CREATE TABLE transaccionCombustible (
    transaccionId INT PRIMARY KEY,
    vehiculoId INT,
    conductorId INT,
    monto DECIMAL(10, 2),
    cantidad DECIMAL(10, 2),
    tipoCombustible VARCHAR(20),
    fechaTransaccion DATETIME,
    ubicacion VARCHAR(100)
);
```

Scripts de extracción de datos

```
USE sistemagestionflotillas;

SELECT * FROM conductor
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/conductor.txt'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT * FROM documento
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/documento.txt'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT * FROM flotilla
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/flotilla.txt'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT * FROM mantenimiento
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/mantenimiento.txt'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT * FROM ruta
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ruta.txt'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT * FROM transaccioncombustible
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transaccioncombustible.txt'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT * FROM vehiculo
INTO OUTFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/vehiculo.txt'
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

SELECT @@secure_file_priv;
```

Scripts de carga de datos

```
USE LCS1Principal;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/vehiculo.txt'
INTO TABLE vehiculo
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/documento.txt'
INTO TABLE documento
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/flotilla.txt'
INTO TABLE flotilla
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

-- ---------------------------

USE LCS2Mantenimiento;
SET FOREIGN_KEY_CHECKS=0;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/vehiculo.txt'
INTO TABLE vehiculo
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/mantenimiento.txt'
INTO TABLE mantenimiento
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

-- -------------------------------------

USE LCS3Rutas;

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/vehiculo.txt'
INTO TABLE vehiculo
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/conductor.txt'
INTO TABLE conductor
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/transaccioncombustible.txt'
INTO TABLE transaccioncombustible
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';

LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/ruta.txt'
INTO TABLE ruta
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Triggers

```
USE LCS1Principal;

DELIMITER //

CREATE TRIGGER after_vehiculo_insert
AFTER INSERT ON LCS1Principal.vehiculo
FOR EACH ROW
BEGIN
    -- Insertar en LCS2Mantenimiento.vehiculo
    INSERT INTO LCS2Mantenimiento.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);

    -- Insertar en LCS3Rutas.vehiculo
    INSERT INTO LCS3Rutas.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 after_vehiculo_update
AFTER UPDATE ON LCS1Principal.vehiculo
FOR EACH ROW
BEGIN
    -- Actualizar en LCS2Mantenimiento.vehiculo
    UPDATE LCS2Mantenimiento.vehiculo
    SET
        flotillaId = NEW.flotillaId,
        tipo = NEW.tipo,
        modelo = NEW.modelo,
        marca = NEW.marca,
        anio = NEW.anio,
        estado = NEW.estado,
        fechaVerificacion = NEW.fechaVerificacion
    WHERE vehiculoId = NEW.vehiculoId;

    -- Actualizar en LCS3Rutas.vehiculo
    UPDATE LCS3Rutas.vehiculo
    SET
        flotillaId = NEW.flotillaId,
        tipo = NEW.tipo,
        modelo = NEW.modelo,
        marca = NEW.marca,
        anio = NEW.anio,
        estado = NEW.estado,
        fechaVerificacion = NEW.fechaVerificacion
    WHERE vehiculoId = NEW.vehiculoId;
END //

DELIMITER ;

DELIMITER //

CREATE TRIGGER after_vehiculo_delete
AFTER DELETE ON LCS1Principal.vehiculo
FOR EACH ROW
BEGIN
    -- Eliminar en LCS2Mantenimiento.vehiculo
    DELETE FROM LCS2Mantenimiento.vehiculo
    WHERE vehiculoId = OLD.vehiculoId;

```

```
    -- Eliminar en LCS3Rutas.vehiculo
    DELETE FROM LCS3Rutas.vehiculo
    WHERE vehiculoId = OLD.vehiculoId;
END //

DELIMITER ;
```

Scripts consultas

```sql
SELECT
    v.vehiculoId,
    v.marca,
    v.modelo,
    r.rutaId,
    r.horaInicio,
    r.horaFin,
    r.ubicacionInicio,
    r.ubicacionFin
FROM
    LCS1Principal.vehiculo v
JOIN
    LCS3Rutas.ruta r
ON
    v.vehiculoId = r.vehiculoId;

-- ------------------------------------ --

SELECT
    d.documentoId,
    d.tipo,
    d.fechaVencimiento,
    c.conductorId,
    c.nombre,
    c.numeroLicencia
FROM
    LCS1Principal.documento d
JOIN
    LCS3Rutas.conductor c
ON
    d.vehiculoId = c.conductorId;
```

vehiculoId	marca	modelo	rutaId	horaInicio	horaFin	ubicacionInicio	ubicacionFin
54	Chevrolet	Ford Transit	1	2023-01-30 12:00:00	2023-01-18 18:00:00	Guadalajara, Jalisco	Hermosillo, Son
61	Volvo	International LT	2	2023-01-23 13:00:00	2023-01-24 02:00:00	Pachuca, Hidalgo	Hermosillo, Son
17	Ford	Mack Anthem	3	2023-01-20 01:00:00	2023-01-28 11:00:00	Puebla, Pue	Guadalajara, Jalisco
95	Freightliner	Isuzu NPR	4	2023-02-01 04:00:00	2023-01-08 02:00:00	Puebla, Pue	Monterrey, NL
80	Peterbilt	Volvo VNL	5	2023-01-09 23:00:00	2023-02-01 15:00:00	Mérida, Yuc	Ciudad de México
92	Ford	Kenworth T680	6	2023-01-31 21:00:00	2023-01-08 11:00:00	Hermosillo, Son	Hermosillo, Son
29	Ford	Peterbilt 579	7	2023-01-09 23:00:00	2023-01-16 17:00:00	Monterrey, NL	Tijuana, BC
1	International	Volvo VNL	8	2023-01-20 21:00:00	2023-01-21 15:00:00	Cancún, QR	Ciudad de México
4	Isuzu	Chevrolet Silverado	9	2023-02-01 01:00:00	2023-01-08 23:00:00	Monterrey, NL	Hermosillo, Son
37	Mercedes-Benz	Mack Anthem	10	2023-02-02 06:00:00	2023-01-27 23:00:00	Monterrey, NL	Cancún, QR
9	International	Ford F-150	11	2023-01-12 23:00:00	2023-01-02 00:00:00	Monterrey, NL	Tijuana, BC

documentoId	tipo	fechaVencimiento	conductorId	nombre	numeroLicencia
1	Factura	2025-04-27	52	Pedro Martínez	LIC-000052
2	Factura	2025-04-18	62	Jorge García	LIC-000062
3	Factura	2025-05-18	24	María Gómez	LIC-000024
4	Tarjeta Circulación	2025-10-16	20	Roberto Ortiz	LIC-000020
5	Permiso Carga	2025-10-29	45	Ana Díaz	LIC-000045
6	Permiso Carga	2026-02-16	90	Ricardo Torres	LIC-000090
7	Tarjeta Circulación	2025-05-25	54	Francisco Jiménez	LIC-000054
8	Seguro	2025-09-16	94	Carlos Pérez	LIC-000094
9	Verificación	2025-11-12	20	Roberto Ortiz	LIC-000020
10	Factura	2025-12-26	86	Daniela Ortiz	LIC-000086
11	Verificación	2025-10-08	17	Jorge García	LIC-000017

Respaldos de Scripts

https://github.com/Osw-do/Bases-de-Datos-Distribuidas
/blob/a642ba220047eafdefd41c5c1efa3f2dc27d1f38/Respaldos

5. Conclusiones

Esta práctica permitió comprender la importancia de la gestión de bases de datos distribuidas y cómo mantener la consistencia de los datos entre ellas. Los triggers, las consultas entre bases de datos y las operaciones de importación/exportación son herramientas esenciales para garantizar la integridad y disponibilidad de la información en un sistema distribuido.
En conclusión, el trabajo realizado en esta práctica proporcionó una base sólida para el diseño, implementación y gestión de sistemas de bases de datos distribuidos, lo cual es fundamental en entornos empresariales donde la información está segmentada en múltiples sistemas.

Referencias Bibliográficas

References

[1] Kimball, R., & Ross, M. (**2013**). *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*. Wiley.

[2] Elmasri, R., & Navathe, S. B. (**2016**). *Fundamentals of Database Systems*. Pearson.

[3] Date, C. J. (**2004**). *An Introduction to Database Systems*. Addison-Wesley.

[4] Inmon, W. H. (**2005**). *Building the Data Warehouse*. Wiley.

[5] 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.