Universidad Autónoma del Estado de Hidalgo

3.2 SQL Fragmentos

ALUMNO: Ian Abishai Ramírez Olvera
SEMESTRE Y GRUPO: 6to 2

1. Introducción

En la actualidad, la gestión eficiente de los recursos empresariales es un factor determinante para la competividad de las organizaciones. Dentro de este contexto, el control y administración de las flotillas vehiculares representa un componente esencial, especialmente para aquellas empresas que dependen del transporte, la logística y la movilidad de personal o mercancías. La Gestión de Flotilla de Autos permite registrar, monitorear y mantener la información relacionada con los vehículos, conductores, rutas, mantenimientos y consumo de combustible, con el propósito de optimizar los costos operativos y garantizar la disponibilidad de las unidades.
El presente proyecto tiene como objetivo el diseño e implementación de un sistema distribuido de bases de datos para la gestión de una flotilla de autos, utilizando MySQL como sistema de gestión de bases de datos relacional. Para ello, se establecen 3 nodos físicos independientes:

1. LCS1-Principal, encargado del control administrativo de la flotilla, los vehículos y la documentación asociada.

2. LCS2-Mantenimiento, responsable del registro y seguimiento de los servicios realizados a cada vehículo.

3. LCS3-Rutas, orientado a la gestión de conductores, rutas y transacciones de combustible.

Cada nodo maneja una porción especifica de la información total del sistema, aplicando fragmentación vertical y horizontal según el tipo de datos y el propósito operativo de cada unidad. Esta estructura permite mejorar la disponibilidad, seguridad y rendimiento del sistema, al distribuir la carga de trabajo y los datos entre distintos servidores o ubicaciones.
Además, se abordan los procesos ETL (Extracción, Transformación y Carga), mediante los cuales los datos son extraídos desde un nodo origen, transformados según las necesidades del análisis, y cargados en los nodos destino. También se implementan scripts SQL que demuestran la creación de nodos, la extracción y carga de información, y consultas distribuidas entre las distintas bases de datos.
En conjunto, esta practica busca reforzar los conocimientos sobre bases de datos distribuidas y su implementación en un entorno realista, fomentando el entendimiento de conceptos como la fragmentación, la replicación y la interoperabilidad entre sistemas. De esta manera, se contribuye al desarrollo de habilidades para diseñar arquitecturas de información robustas y escalables, aplicables a escenarios empresariales donde la integridad y la disponibilidad de los datos son fundamentales.

2. Marco teórico

- Fragmentación vertical

Técnica que consiste en dividir una tabla de una base de datos en varias tablas más pequeñas, conservando la clave principal en cada una. Esta división se basa en el tipo de datos, donde cada fragmento contiene un subconjunto de columnas de la tabla original, permitiendo que algunas aplicaciones accedan solo a un fragmento específico sin necesidad de procesar toda la tabla. Por ejemplo, un fragmento podría contener los datos de perfil de usuario y otro los datos de transacciones. En el sistema de gestión de flotilla se aplico fragmentación vertical en la tabla vehiculo, debido a que diferentes nodos requieren solo parte de sus atributos:

1. Nodo LCS1-Principal: mantiene los datos generales(marca, modelo, anio, placas, tipo, estado).
2. Nodo LCS2-Mantenimiento: conversa los datos relacionados con los servicios (idVehiculo, marca, modelo, anio, placas) y los vincula con mantenimiento.
3. Nodo LCS3-Rutas: guarda únicamente los datos necesarios para las rutas y transacciones de combustible (idVehiculo, marca, modelo, placas).

- Procesos ETL

Son un método para la integración de datos que consiste en extraer información de distintas fuentes, transformarla para limpiarla y organizarla según reglas de negocio, y cargarla en un destino final, como un almacén de datos, para su análisis. Se utilizan para consolidar datos de múltiples orígenes para obtener información de valor, mejorar la toma de decisiones y generar informes. Se diseñaron procesos ETL para sincronizar la información entre nodos:

1. Extract (Extracción): se obtienen registros de vehículos y flotilla del nodo principal mediante consultas SELECT.
2. Transform (Transformación): los datos se filtran y adaptan al formato requerido por cada nodo.
3. Load (Carga): los datos transformados se insertan en las tablas correspondientes de los otros nodos (vehiculo en LCS2 y LCS3).

- SELECT + INTO FILE

Es una sentencia SQL que exporta los resultados de una consulta directamente a un archivo. A diferencia de SELECT INTO, que crea una nueva tabla en la base de datos, la versión INTO FILE envía los datos a un archivo en el sistema de almacenamiento, lo que la hace útil para exportar datos para otros propósitos, como análisis o respaldo.

```
SELECT * FROM vehiculo
INTO OUTFILE '/tmp/vehiculos.cvs'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\'
```

- LOAD

Se refiere principalmente a la instrucción LOAD DATA INFILE, que se utiliza para cargar datos de un archivo de texto a una tabla de manera muy eficiente. También puede referirse a mysqlslap, una herramienta para simular cargas de trabajo y probar el rendimiento del servidor. Los archivos genreados con SELECT INTO OUTFILE se cargan en las tablas de los nodos secundarios con el siguiente comando:

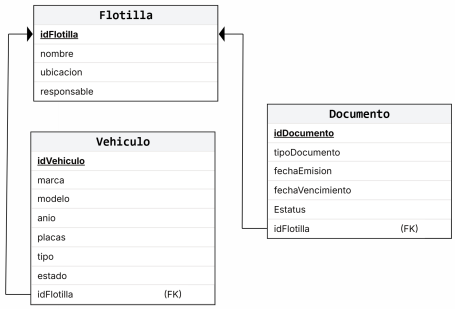
```
SELECT * FROM vehiculo
INTO OUTFILE '/tmp/vehiculos.cvs'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\'
```

- SELECT con tablas de dos bases de datos

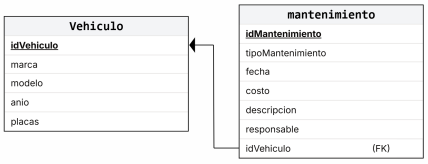
El Modelo Entidad-Relación (MER) es una herramienta gráfica para diseñar bases de datos, creada por Peter Chen en 1976, que representa conceptos como entidades, sus atributos y las relaciones entre ellas. Las entidades son objetos del mundo real (como un "estudiante" o un "curso"), los atributos son sus características (como "nombre" o "carrera"), y las relaciones describen cómo interactúan las entidades (como un "estudiante" que se inscribe en un "curso").

+ Esquema Conceptual Local de cada nodo

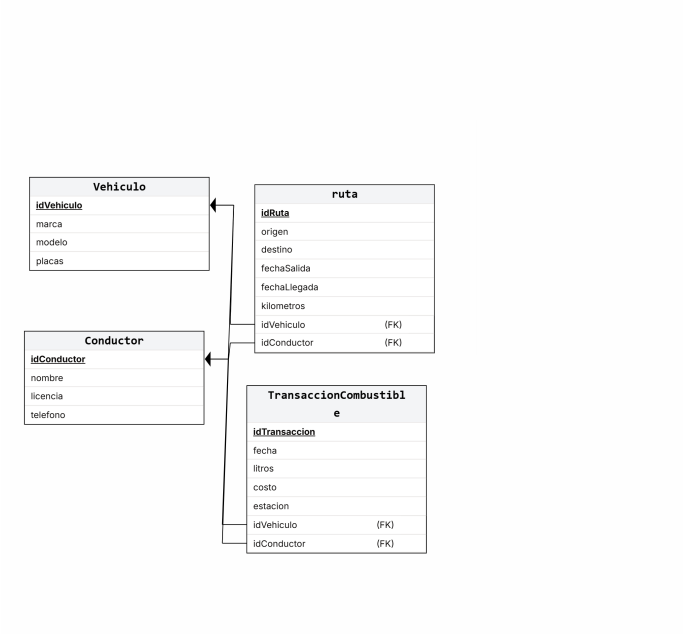
1. Nodo LCS1 - Principal: flotilla, vehiculo, documento.



2. Nodo LCS2 - Mantenimiento: vehiculo, mantenimiento.



3. Nodo LCS3 - Rutas: Vehiculo, conductor, ruta, transaccionCombustible.



Cada esquema local define las entidades y relaciones que cada area gestiona de forma independiente.

+ Script de creación de nodos

Scripts SQL que crean las tablas de cada nodo (como los que ya estan para estan para LCS1, LCS2 y LCS3). Se incluyen las sentencias CREATE TABLE con las llaves foraneas y relaciones internas.

```sql
CREATE DATABASE LCS1_Principal;
USE LCS1_Principal;

CREATE TABLE flotilla(
idFlotilla INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100) NOT NULL,
ubicación VARCHAR(100),
responsable VARCHAR(100)
);

CREATE TABLE vehiculo(
idVehiculo INT PRIMARY KEY AUTO_INCREMENT,
idFlotilla INT,
marca VARCHAR(50),
modelo VARCHAR(50),
anio INT,
placas VARCHAR(15) UNIQUE,
tipo VARCHAR(50),
estado VARCHAR(50),
FOREING KEY (idFlotilla) REFERENCES flotilla(idFlotilla)
);

CREATE TABLE documento(
idDocumento INT PRIMARY KEY AUTO_INCREMENT,
idVehiculo INT,
tipoDocumento VARCHAR(50),
fechaEmision DATE,
fechaVencimiento DATE,
estatus VARCHAR(30),
FOREIGN KEY (idVehiculo) REFERENCES vehiculo(idVehiculo)
);
```

```sql
CREATE DATABASE LCS2_Mantenimiento;
USE LCS2_Mantenimiento;

CREATE TABLE vehiculo(
idVehiculo INT PRIMARY KEY,
marca VARCHAR(50),
modelo VARCHAR(50),
anio INT,
placas VARCHAR(15)
);

CREATE TABLE mantenimiento (
idMantenimiento INT PRIMARY KEY AUTO_INCREMENT,
idVehiculo INT,
tipoMantenimiento VARCHAR(100),
fecha DATE,
costo DECIMAL(10,2),
descripcion TEXT,
responsable VARCHAR(100),
FOREIGN KEY (idVehiculo) REFERENCES vehiculo(idVehiculo)
);
```

```sql
CREATE DATABASE LCS3_Rutas;
USE LCS3_Rutas;

CREATE TABLE vehiculo(
idVehiculo INT PRIMARY KEY,
marca VARCHAR(50),
modelo VARCHAR(50),
anio INT,
placas VARCHAR(15)
);

CREATE TABLE conductor(
idConductor INT PRIMARY KEY AUTO_INCREMENT,
nombre VARCHAR(100),
licencia VARCHAR(50) UNIQUE,
telefono VARCHAR(20)
);

CREATE TABLE ruta(
idRuta INT PRIMARY KEY AUTO_INCREMENT,
idVehiculo INT,
idConductor INT,
origen VARCHAR(100),
destino VARCHAR(100),
fechaSalida DATETIME,
fechaLlegada DATETIME,
kilometros DECIMAL(8,2),
FOREIGN KEY (idVehiculo) REFERENCES vehiculo(idVehiculo),
FOREIGN KEY (idConductor) REFERENCES conductor(idConductor)
);

CREATE TABLE transaccionCombustible(
idTransaccion INT PRIMARY KEY AUTO_INCREMENT,
idVehiculo INT,
idRuta INT,
fecha DATETIME,
litros DECIMAL(8,2),
costo DECIMAL(10,2),
estación VARCHAR(100),
FOREIGN KEY (idVehiculo) REFERENCES vehiculo(idVehiculo),
FOREIGN KEY (idRuta) REFERENCES ruta(idRuta)
);
```

+ Scripts de extracción de datos

Aqui son consultas que obtienen los datos del nodo principal

```sql
SELECT idVehiculo, marca, modelo, anio, placas FROM vehiculo;
SELECT idFlotilla, nombre FROM flotilla;
```

+ Script de carga de datos

Inserciones o cargas desde archivo en los otros nodos:

```sql
INSERT INTO vehiculo (idVehiculo, marca,modelo, anio, placas) VALUES (1, 'Nissan', 'Versa', 2024, 'ABC123');
```

+ Script de consulta de datos a dos tablas en al menos dos de los nodos

Se utiliza la cláusula JOIN especificando las tablas en la cláusula FROM y las condiciones de unión en la cláusula ON, usando la clave primaria de una tabla y la clave foránea de la otra. La consulta básica es SELECT columnas FROM tabla1 JOIN tabla2 ON tabla1.id = tabla2.id, donde las tablas tabla1 y tabla2 se unen por sus respectivas columnas id.

```sql
SELECT v.marca, v.modelo, m.tipoMantenimiento, r.origen, r.destino
FROM LCS2_Mantenimiento.vehiculo v
JOIN LCS2_Mantenimiento.mantenimiento m ON v.idVehiculo = m.idVehiculo
JOIN LCS3_Rutas.ruta r ON v.idVehiculo = r.idVehiculo;
```

3. Conclusiones

La implementación de un sistema distribuido para la Gestión de Flotilla de Autos permitió comprender de manera práctica como se puede estructurar y administrar la información de una organización a través de distintos nodos físicos y lógicos, manteniendo la coherencia e integridad de los datos.
La distribución de nodos demostró la importancia de la fragmentación vertical y horizontal para optimizar el acceso a la información, evitando la duplicación innecesaria de datos y reduciendo la carga de procesamiento en un solo servidor.
A nivel conceptual, el ejercicio reforzo los conocimientos sobre el diseño de sistemas de bases de datos distribuidas, la definición de esquemas conceptuales locales, la aplicación de claves foráneas para mantener integridad referencial y el manejo de consultas multibase. A nivel practico, se evidencio como una arquitectura distribuida puede mejorar la disponibilidad, seguridad y escalabilidad del sistema de información.
En conclusión, este trabajo representa una aplicación efectiva de los principios de bases de datos distribuidas dentro de un entorno realista. Permite visualizar cómo la segmentación lógica de la información y la correcta definición de procesos ETL contribuyen a una gestión más eficiente de los recursos, lo cual resulta fundamental para cualquier organización que busque optimizar la administración de su flotilla y mejorar su toma de decisiones operativas.

4. Referencias Bibliográficas

References

[1] Chaubey, R. (2023, 9 agosto). Understanding fragmentation in distributed databases. https://www-c–sharpcorner-com.translate.goog/article/understanding-fragmentation-in-distributed-databases/?$_{xt}r_sl = en_{xt}r_tl = es_{xt}r_hl = es_{xt}r_pto = sge : : text = vertical$

[2] campusMVP.es. (2021, 3 noviembre). TUTORIAL SQL 3: Consultas multi-tabla [Vídeo]. YouTube. https://www.youtube.com/watch?v=h4d9KsyoSQI

[3] Claytonsiemens. (s. f.). Extraer, transformar, cargar (ETL) - Azure Architecture Center. Microsoft Learn. https://learn.microsoft.com/es-es/azure/architecture/data-guide/relational-data/etl

[4] colaboradores de Wikipedia. (2025h, julio 30). Modelo entidad-relación. Wikipedia, la Enciclopedia Libre. https://es.wikipedia.org/wiki/Modelo$_{e}ntidad - relaci$

[5] Espino, V. A. P. (2019, 2 junio). ¿Cómo hacer consulta SQL de 2 tablas relacionadas? Stack Overflow En Español. https://es.stackoverflow.com/questions/268965/c

[6] Ibm. (2025, 11 junio). ETL. ¿Qué es ETL (extracción, transformación, carga)? https://www.ibm.com/es-es/think/topics/etl: :text=ETL

[7] NullSafe Architect. (2018, 3 septiembre). Ciencia de datos desde 0: Proceso de ETL(Extract-Transform-Load) con Logstash [Vídeo]. YouTube. https://www.youtube.com/watch?v=$_{x}26AMwAetQ$

[8] Oracle. (2021b, junio 18). What is ETL? Your Guide to Data Integration Essentials. https://www.oracle.com/mx/integration/what-is-etl/: :text=Extracci

[9] PyNet Labs. (2025, 12 junio). Top Difference Between OSI and TCP/IP Model (2025). PyNet Labs. https://www-pynetlabs-com.translate.goog/difference-between-osi-and-tcp-ip-model/?$_{xt}r_{s}l = en_{xt}r_{t}l = es_{xt}r_{h}l = es_{xt}r_{p}to = sge : : text = Similitudes$

[10] ¿Qué es ETL? - Explicación de extracción, transformación y carga (ETL) - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/etl/: :text=en

[11] ¿Qué es ETL? Google Cloud. (s. f.-a). Google Cloud. https://cloud.google.com/learn/what-is-etl?hl=es

[12] Rozo, J. (s. f.). Fragmentación vertical. Scribd. https://es.scribd.com/document/483776882/Fragmentacion-vertical: :text=La

[13] SELECT INTO OUTFILE - Apache Doris. (s. f.). https://doris.apache.org/docs/3.x/data-operate/export/outfile?$_{xt}r_{s}l = en_{xt}r_{t}l = es_{xt}r_{h}l = es_{xt}r_{p}to = sge$

[14] Sharma, D., Sharma, D. (2024, 6 octubre). Allocation Fragmentation and Replication In Distributed Databases: A Quick Start Guide. Learn — Hevo. https://hevodata-com.translate.goog/learn/fragmentation-and-replication-in-distributed-database/?$_{xt}r_{s}l = en_{xt}r_{t}l = es_{xt}r_{h}l = es_{xt}r_{p}to = sge : : text = Fragmentaci$

[15] SQL SELECT INTO - Syntax, Use Cases, and Examples. (2023, 29 diciembre). Hightouch. https://hightouch-com.translate.goog/sql-dictionary/sql-select-into?$_{xt}r_{s}l = en_{xt}r_{t}l = es_{xt}r_{h}l = es_{xt}r_{p}to = sge_{xt}r_{h}ist = true$

[16] SQL SELECT INTO – cómo copiar conjuntos de datos. (2025, 16 enero). IONOS Digital Guide. https://www.ionos.mx/digitalguide/servidores/configuracion/sql-select-into/: :text=tabla

[17] Wdzieczna, D. (2022, 14 julio). Cómo unir dos tablas en SQL. LearnSQL.es. https://learnsql.es/blog/como-unir-dos-tablas-en-sql/: :text=Usar

[18] Vide, O. (s. f.). Procesos ETL: transformando datos en información - blog incentro. Incentro. https://www.incentro.com/es-ES/blog/proceso-etl