Universidad Autónoma del Estado de Hidalgo

REPORTE DE PRÁCTICA NO 2.4

3 nodos BDD Flotillas

ALUMNO: Esperilla Mendoza Luis Erick
Dr. Eduardo Cornejo-Velázquez

1. Introducción

Este trabajo se enfoca en la creación y gestión de tres nodos físicos en MySQL, cada uno con un esquema específico diseñado para manejar diferentes aspectos de la flotilla: el nodo LCS1-Principal, el nodo LCS2-Mantenimiento y el nodo LCS3-Rutas.

El objetivo principal de esta práctica es demostrar cómo se pueden utilizar técnicas de fragmentación vertical para distribuir la información de manera eficiente, así como implementar procesos ETL (Extract, Transform, Load) para la extracción, transformación y carga de datos entre los diferentes nodos.

Además, se explorarán comandos avanzados de MySQL como SELECT INTO OUTFILE y LOAD DATA INFILE para la manipulación de datos, y se realizarán consultas que involucren tablas de dos bases de datos diferentes.

2. Marco teórico

Fragmentación vertical

Consiste en dividir una tabla en subconjuntos de columnas (atributos), donde cada subconjunto se almacena en un nodo diferente. A diferencia de la fragmentación horizontal, que divide una tabla en filas, la fragmentación vertical se enfoca en la separación de atributos según su relevancia o uso en diferentes contextos.

Supongamos que tenemos una tabla vehiculo con los siguientes atributos:

Listing 1: Tabla de vehiculo

```
vehiculo (id_vehiculo, marca, modelo, anio, kilometraje,
    fecha_ultimo_mantenimiento, costo_mantenimiento)\\
```

Una posible fragmentación vertical podría ser:

Fragmento 1 (Nodo LCS1-Principal): id_vehiculo, marca, modelo, año

Fragmento 2 (Nodo LCS2-Mantenimiento): id_vehiculo, kilometraje, fecha_ultimo_mantenimiento, costo_mantenimiento

En este caso, el atributo id_vehiculo actúa como clave primaria y se mantiene en ambos fragmentos para permitir la unión de datos cuando sea necesario.

Procesos ETL

ETL (Extract, Transform, Load) es un proceso utilizado en la integración de datos que consiste en tres etapas principales:

1. Extracción (Extract): Recopilar datos de diversas fuentes, como bases de datos, archivos, APIs, etc.

2. Transformación (Transform): Limpiar, estructurar y convertir los datos en un formato adecuado para su almacenamiento y análisis.

3. Carga (Load): Almacenar los datos transformados en un sistema de destino, como un data warehouse, una base de datos o un repositorio centralizado.

SELECT + INTO OUTFILE

El comando SELECT + INTO OUTFILE en MySQL es una sentencia SQL que permite exportar el resultado de una consulta a un archivo en el servidor. Esta funcionalidad es útil cuando se necesita guardar datos en un formato legible o transferirlos a otros sistemas.

LOAD DATA INFILE

El comando LOAD DATA INFILE en MySQL es una sentencia SQL que permite cargar datos desde un archivo de texto plano o CSV directamente en una tabla de la base de datos. Esta funcionalidad es especialmente útil para importar grandes volúmenes de datos de manera eficiente, evitando la necesidad de insertar los datos manualmente o mediante scripts.

SELECT con tablas de dos bases de datos

En MySQL, es posible realizar consultas que involucren tablas de dos bases de datos diferentes utilizando la cláusula JOIN.

Esto se logra referenciando las tablas con el nombre de la base de datos seguido del nombre de la tabla, en el formato nombre_base_datos.nombre_tabla.

Esta funcionalidad es útil en entornos donde los datos están distribuidos en múltiples bases de datos pero necesitan ser consultados de manera conjunta.

3. Herramientas empleadas

1. DataGrip.

   DataGrip es un entorno de desarrollo integrado (IDE) para bases de datos desarrollado por JetBrains. Permite gestionar bases de datos de manera eficiente con una interfaz avanzada para la escritura y ejecución de consultas SQL.

   Uso en la práctica:

   - Conexión y gestión de la base de datos MySQL.
   - Creación y edición de vistas para la fragmentación.
   - Creación de nodos.
   - Scripts de extracción de datos.
   - Script de carga de datos.

2. Github.

   GitHub es una plataforma de control de versiones que facilita la colaboración y el almacenamiento de archivos de proyectos. Se utilizó para alojar la evidencia de la actividad.

   Uso en la práctica:

   - Script finales de los nodos
   - Almacenamiento del reporte de la práctica.
   - Registro de la URL del repositorio en la Plataforma Garza.

4. Desarrollo

Esquema Conceptual Local de cada nodo

- DIAGRAMA LOCAL DEL NODO LCS1 - PRINCIPAL

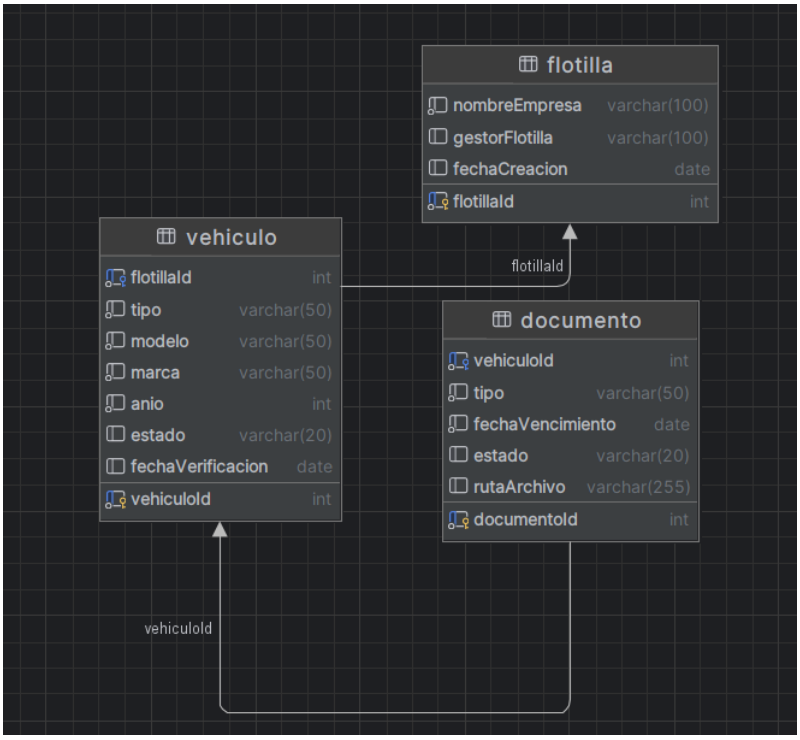

Figure 1: Diagrama nodo LCS1 - Principal

- DIAGRAMA LOCAL DEL NODO LCS2 - MANTENIMIENTO

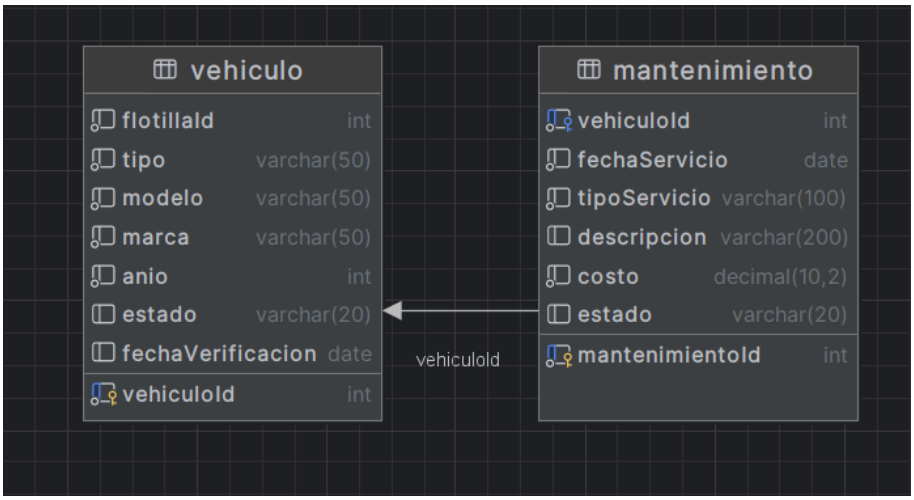

Figure 2: Diagrama nodo LCS2 - MANTENIMIENTO

- DIAGRAMA LOCAL DEL NODO LCS3 - RUTAS

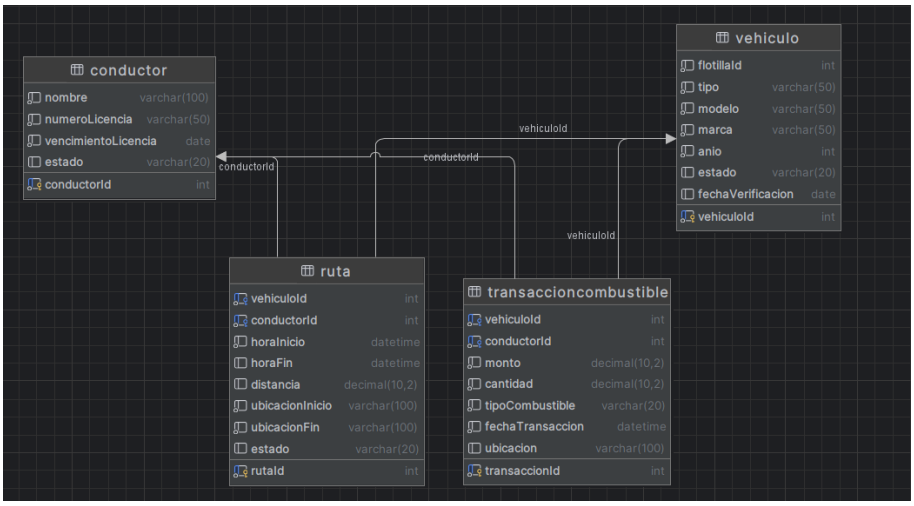

Figure 3: Diagrama nodo LCS3 - RUTAS

Script de creación de nodos

Listing 2: Script para la creación del nodo LCS1 - PRINCIPAL

```
CREATE DATABASE LCS1_PRINCIPAL;

USE lcs1_principal;

create table if not exists flotilla
(
    flotillaId     int auto_increment
        primary key,
    nombreEmpresa  varchar(100) not null,
    gestorFlotilla varchar(100) null,
    fechaCreacion  date         null
)
    auto_increment = 127;

create table if not exists vehiculo
(
    vehiculoId        int auto_increment
        primary key,
    flotillaId        int                              not null,
    tipo              varchar(50)                      not null,
    modelo            varchar(50)                      not null,
    marca             varchar(50)                      not null,
    anio              int                              not null,
    estado            varchar(20) default 'Activo'     null,
    fechaVerificacion date                             null,
    constraint vehiculo_ibfk_1
        foreign key (flotillaId) references flotilla (flotillaId)
            on delete cascade
)
    auto_increment = 127;

create table if not exists documento
(
    documentoId      int auto_increment
        primary key,
    vehiculoId       int                              not null,
    tipo             varchar(50)                      not null,
    fechaVencimiento dEate                             not null,
    estado           varchar(20) default 'Vigente'    null,
    rutaArchivo      varchar(255)                     null,
```

```
    constraint documento_ibfk_1
        foreign key (vehiculoId) references vehiculo (vehiculoId)
            on delete cascade
)
    auto_increment = 127;
```

//

Listing 3: Script para la creación del nodo LCS2 - MANTENIMIENTO

```
CREATE DATABASE LCS2_MANTENIMIENTO;

USE lcs2_mantenimiento;

create table if not exists vehiculo
(
    vehiculoId        int auto_increment
        primary key,
    flotillaId        int                             not null,
    tipo              varchar(50)                     not null,
    modelo            varchar(50)                     not null,
    marca             varchar(50)                     not null,
    anio              int                             not null,
    estado            varchar(20) default 'Activo'    null,
    fechaVerificacion date                            null
)
    auto_increment = 127;

create table if not exists mantenimiento
(
    mantenimientoId int auto_increment
        primary key,
    vehiculoId      int                               not null,
    fechaServicio   date                              not null,
    tipoServicio    varchar(100)                      not null,
    descripcion     varchar(200)                      null,
    costo           decimal(10, 2)                    not null,
    estado          varchar(20) default 'Completado'  null,
    constraint mantenimiento_ibfk_1
        foreign key (vehiculoId) references vehiculo (vehiculoId)
            on delete cascade
)
    auto_increment = 127;
```

//

Listing 4: Script para la creación del nodo LCS3 - RUTAS

```
CREATE DATABASE LCS3_RUTAS;

USE lcs3_rutas;

create table if not exists vehiculo
(
    vehiculoId        int auto_increment
        primary key,
    flotillaId        int                             not null,
    tipo              varchar(50)                     not null,
    modelo            varchar(50)                     not null,
    marca             varchar(50)                     not null,
    anio              int                             not null,
    estado            varchar(20) default 'Activo'    null,
    fechaVerificacion date                            null
)
    auto_increment = 127;

create table if not exists conductor
(
    conductorId         int auto_increment
        primary key,
```

```
    nombre              varchar(100)                 not null,
    numeroLicencia      varchar(50)                  not null,
    vencimientoLicencia date                         not null,
    estado              varchar(20) default 'Activo' null
)
    auto_increment = 127;

create table if not exists ruta
(
    rutaId          int auto_increment
        primary key,
    vehiculoId      int                              not null,
    conductorId     int                              not null,
    horaInicio      datetime                         not null,
    horaFin         datetime                         null,
    distancia       decimal(10, 2)                   null,
    ubicacionInicio varchar(100)                     not null,
    ubicacionFin    varchar(100)                     not null,
    estado          varchar(20) default 'Pendiente' null,
    constraint ruta_ibfk_1
        foreign key (vehiculoId) references vehiculo (vehiculoId)
            on delete cascade,
    constraint ruta_ibfk_2
        foreign key (conductorId) references conductor (conductorId)
            on delete cascade
)
    auto_increment = 127;

create table if not exists transaccioncombustible
(
    transaccionId    int auto_increment
        primary key,
    vehiculoId       int            not null,
    conductorId      int            not null,
    monto            decimal(10, 2) not null,
    cantidad         decimal(10, 2) not null,
    tipoCombustible  varchar(20)    not null,
    fechaTransaccion datetime       not null,
    ubicacion        varchar(100)   null,
    constraint transaccioncombustible_ibfk_1
        foreign key (vehiculoId) references vehiculo (vehiculoId)
            on delete cascade,
    constraint transaccioncombustible_ibfk_2
        foreign key (conductorId) references conductor (conductorId)
            on delete cascade
)
    auto_increment = 127;
```

Scripts de extracción de datos

Listing 5: Script para extraer los datos de la tabla conductor

```
    SELECT *
        INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
            conductor_data.csv'
        FIELDS TERMINATED BY ','
        ENCLOSED BY '"'
        LINES TERMINATED BY '\n'
        FROM conductor;
```

//

Listing 6: Script para extraer los datos de la tabla documento

```
SELECT *
    INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
        documento_data.csv'
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    FROM documento;
```

//

Listing 7: Script para extraer los datos de la tabla flotilla

```
SELECT *
    INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
        flotilla_data.csv'
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    FROM flotilla;
```

//

Listing 8: Script para extraer los datos de la tabla mantenimiento

```
SELECT *
    INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
        mantenimiento_data.csv'
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    FROM mantenimiento;
```

//

Listing 9: Script para extraer los datos de la tabla ruta

```
SELECT *
    INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\ruta_data.
        csv'
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    FROM Ruta;
```

//

Listing 10: Script para extraer los datos de la tabla transaccionCombustible

```
SELECT *
    INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
        combustible_data.csv'
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    FROM transaccioncombustible;
```

Listing 11: Script para extraer los datos de la tabla vehiculo

```
SELECT *
    INTO OUTFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
        vehiculo_data.csv'
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    FROM vehiculo;
```

Script de carga de datos

Listing 12: Script para cargar los datos de la tabla conductor

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
    conductor_data.csv'
INTO TABLE conductor
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Listing 13: Script para cargar los datos de la tabla documento

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
    documento_data.csv'
INTO TABLE documento
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Listing 14: Script para cargar los datos de la tabla flotilla

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
    flotilla_data.csv'
INTO TABLE flotilla
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Listing 15: Script para cargar los datos de la tabla mantenimiento

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
    mantenimiento_data.csv'
INTO TABLE mantenimiento
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Listing 16: Script para cargar los datos de la tabla ruta

```
    LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
        ruta_data.csv'
    INTO TABLE Ruta
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n'
    IGNORE 1 ROWS;
```

Listing 17: Script para cargar los datos de la tabla combustible

```
LOAD DATA INFILE 'C:\\ProgramData\\MySQL\\MySQL␣Server␣8.0\\Uploads\\
    combustible_data.csv'
INTO TABLE transaccioncombustible
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Listing 18: Script para cargar los datos de la tabla vehiculo

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

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

Consulta 1: Obtener información de vehículos que estén activos con sus mantenimientos y rutas

Listing 19: Script Obtener información de vehículos que estén activos con sus mantenimientos y rutas

```
SELECT
    p.vehiculoId AS id_vehiculo,
    p.marca,
    p.modelo,
    p.anio,
    m.fechaServicio,
    m.tipoServicio,
    r.ubicacionInicio,
    r.ubicacionFin,
    r.horaInicio,
    r.horaFin
FROM
    lcs1_principal.vehiculo p
LEFT JOIN
    lcs2_mantenimiento.mantenimiento m ON p.vehiculoId = m.vehiculoId
LEFT JOIN
    lcs3_rutas.ruta r ON p.vehiculoId = r.vehiculoId
WHERE
    p.estado = 'Activo'
ORDER BY
    p.vehiculoId, m.fechaServicio DESC;
```

	id_vehiculo	marca	modelo	anio	fechaServicio	tipoServicio	ubicacionInicio	ubicacionFin	hor
1	6	Ram	Ford F-150	2023	<null>	<null>	<null>	<null>	<null>
2	7	Ram	Volvo VNL	2017	<null>	<null>	<null>	<null>	<null>
3	8	Volvo	Mack Anthem	2011	<null>	<null>	Querétaro, Qro	Querétaro, Qro	2023-0
4	15	Kenworth	Ram 1500	2018	<null>	<null>	Guadalajara, Jalisco	Tijuana, BC	2023-0
5	23	Kenworth	Kenworth T680	2010	2023-12-05	Reparación motor	Hermosillo, Son	Mérida, Yuc	2023-0
6	27	Kenworth	International LT	2012	<null>	<null>	Mérida, Yuc	Puebla, Pue	2023-0
7	27	Kenworth	International LT	2012	<null>	<null>	Guadalajara, Jalisco	Tijuana, BC	2023-0
8	28	Chevrolet	Hino 338	2010	<null>	<null>	<null>	<null>	<null>
9	33	Ram	Freightliner Cascadia	2023	2023-01-10	Alineación	Tijuana, BC	Monterrey, NL	2023-0
10	34	Kenworth	Peterbilt 579	2021	2021-04-19	Alineación	Hermosillo, Son	Cancún, QR	2023-0
11	34	Kenworth	Peterbilt 579	2021	2021-04-19	Alineación	Ciudad de México	Pachuca, Hidalgo	2023-0
12	41	Mercedes-Benz	Mack Anthem	2018	2021-02-27	Cambio de aceite	<null>	<null>	<null>
13	44	Ram	Chevrolet Silverado	2017	2023-01-28	Rotación de llantas	Tijuana, BC	Hermosillo, So	-0

31 rows

Figure 4: Ejecución de código. 31 filas

Consulta 2: Obtener transacciones de combustible con información del conductor y vehículo

Listing 20: Script para obtener transacciones de combustible que se hicieron en un intervalo de dias con información del conductor y vehículo

```
SELECT
    t.transaccionId AS id_transaccion,
    t.fechaTransaccion,
    t.monto,
    c.nombre AS nombre_conductor,
    c.numeroLicencia,
    v.marca,
    v.modelo
FROM
    lcs3_rutas.transaccioncombustible t
JOIN
    lcs3_rutas.conductor c ON t.conductorId = c.conductorId
JOIN
    lcs1_principal.vehiculo v ON t.vehiculoId = v.vehiculoId
WHERE
    t.fechaTransaccion BETWEEN '2023-02-01' AND '2023-03-31'
ORDER BY
    t.fechaTransaccion DESC;
```

	id_transaccion	fechaTransaccion	monto	nombre_conductor	numeroLicencia	marca	modelo
1	3	2023-02-03 04:00:00	374.93	Patricia Morales	LIC-000078	Ram	Volvo VNL
2	52	2023-02-03 01:00:00	348.37	Lucía López	LIC-000051	Kenworth	International LT
3	51	2023-02-02 17:00:00	81.06	Carmen López	LIC-000006	Ram	Ford F-150
4	44	2023-02-02 14:00:00	42.98	Andrés Díaz	LIC-000004	Peterbilt	Ford F-150
5	65	2023-02-02 10:00:00	312.17	Daniela Ortiz	LIC-000086	Chevrolet	Mack Anthem
6	81	2023-02-02 06:00:00	162.24	Luis Morales	LIC-000046	Ford	Peterbilt 579
7	85	2023-02-01 21:00:00	299.32	Miguel Flores	LIC-000025	Mercedes-Benz	Chevrolet Silverado
8	43	2023-02-01 20:00:00	367.58	Luis Gómez	LIC-000099	Kenworth	Hino 338
9	99	2023-02-01 13:00:00	181.78	Jorge Pérez	LIC-000021	Isuzu	Ram 1500
10	31	2023-02-01 12:00:00	494.23	Francisco Jiménez	LIC-000054	Ram	Freightliner Cascadia
11	83	2023-02-01 07:00:00	197.55	Ricardo Pérez	LIC-000070	Peterbilt	Ram 1500
12	53	2023-02-01 01:00:00	57.50	Jorge García	LIC-000042	Ford	Peterbilt 579

12 rows

Figure 5: Ejecución de código. 12 filas

Procedimientos Almacenados

Listing 21: Script para obtener vehículos con su último mantenimiento

```
DELIMITER //

CREATE PROCEDURE ObtenerVehiculosConUltimoMantenimiento()
BEGIN
    SELECT
        v.vehiculoId,
        v.marca,
        v.modelo,
        v.anio,
        v.estado,
        m.fechaServicio,
        m.tipoServicio,
        m.costo
    FROM
        lcs1_principal.vehiculo v
    LEFT JOIN
        lcs2_mantenimiento.mantenimiento m ON v.vehiculoId = m.vehiculoId
    WHERE
        m.fechaServicio = (
            SELECT MAX(fechaServicio)
            FROM lcs2_mantenimiento.mantenimiento
            WHERE vehiculoId = v.vehiculoId
        )
    ORDER BY
        v.vehiculoId;
END //

DELIMITER ;

CALL ObtenerVehiculosConUltimoMantenimiento();
```

	vehiculoId	marca	modelo	anio	estado	fechaServicio	tipoServicio	costo
1	2	Freightliner	Isuzu NPR	2015	Inactivo	2023-12-15	Alineación	2870.
2	3	Mercedes-Benz	Kenworth T680	2021	Mantenimiento	2022-11-03	Rotación de llantas	590.
3	4	Isuzu	Chevrolet Silverado	2018	En Ruta	2023-12-01	Revisión de frenos	2712.
4	5	Chevrolet	Ram 1500	2017	Inactivo	2020-08-12	Balanceo	568.
5	12	Freightliner	Ford F-150	2015	Inactivo	2023-01-28	Rotación de llantas	2921.
6	17	Ford	Mack Anthem	2014	En Ruta	2024-01-13	Cambio de aceite	3093.
7	19	Ram	Peterbilt 579	2013	En Ruta	2022-07-29	Alineación	2538.
8	20	Isuzu	Hino 338	2017	Inactivo	2023-04-23	Rotación de llantas	3398.
9	21	Volvo	Chevrolet Silverado	2017	En Ruta	2023-03-12	Cambio de aceite	3011.
10	22	Mercedes-Benz	Isuzu NPR	2013	Inactivo	2023-06-30	Revisión de frenos	1967.
11	23	Kenworth	Kenworth T680	2010	Activo	2023-12-05	Reparación motor	2250.
12	24	Ram	Freightliner Cascadia	2020	Mantenimiento	2023-10-27	Cambio de filtros	2336.
13	25	Peterbilt	International LT	2018	En Ruta	2022-08-24	Revisión de frenos	835.
14	26	Kenworth	International LT	2010	Mantenimiento	2022-02-23	Cambio de filtros	3256.
15	29	Ford	Peterbilt 579	2012	En Ruta	2021-03-15	Balanceo	1228.
16	30	Kenworth	International LT	2011	En Ruta	2024-05-06	Cambio de aceite	3087.
17	31	Mercedes-Benz	Ford F-150	2014	En Ruta	2024-02-29	Reparación motor	675.
18	33	Ram	Freightliner Cascadia	2023	Activo	2023-01-10	Alineación	8.

63 rows

Figure 6: Ejecución de código. 63 filas

Triggers

Listing 22: Script para que el trigger se active después de una operación INSERT en la tabla mantenimiento. Este trigger actualizará el campo fechaVerificacion en la tabla vehiculo de la base de datos principal.

```
DELIMITER //

CREATE TRIGGER after_insert_mantenimiento
AFTER INSERT ON lcs2_mantenimiento.mantenimiento
FOR EACH ROW
BEGIN
    UPDATE lcs1_principal.vehiculo
    SET fechaVerificacion = NEW.fechaServicio
    WHERE vehiculoId = NEW.vehiculoId;
END //

DELIMITER ;

INSERT INTO lcs2_mantenimiento.mantenimiento (vehiculoId, fechaServicio,
    tipoServicio, descripcion, costo, estado)
VALUES (1, '2023-10-25', 'Cambio de aceite', 'Cambio de aceite y filtro',
    150.00, 'Completado');
```

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	1	8	Pickup	Volvo VNL	International	2023	Mantenimiento	2023-10-25

Figure 7: Ejecución de código

Replicacion del nodo principal a los nodos Mantenimiento y rutas (AFTER INSERT, UPDATE AND DELETE

Listing 23: Script para que al insertar un nuevo vehiculo en el nodo principal este lo hago en los otros dos nodos

```
-- INSERT -- INSERT -- INSERT -- INSERT

DELIMITER //

CREATE TRIGGER after_insert_vehiculo
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);

    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 ;

INSERT INTO lcs1_principal.vehiculo (vehiculoId, flotillaId, tipo, modelo, marca,
    anio, estado, fechaVerificacion)
    VALUES (1, 11, 'Sedan', 'Corolla', 'Toyota', 2020, 'Activo', '2023-10-25');
```

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	1	11	Sedán	Corolla	Toyota	2020	Activo	2023-10-25

Figure 8: Nodo LCS1 - PRINCIPAL

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	1	11	Sedán	Corolla	Toyota	2020	Activo	2023-10-25

Figure 9: Nodo LCS2 - MANTENIMIENTO

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	1	11	Sedán	Corolla	Toyota	2020	Activo	2023-10-25

Figure 10: Nodo LCS3 - RUTAS

Listing 24: Script para que al actualizar un nuevo vehiculo en el nodo principal este lo hago en los otros dos nodos

```
DELIMITER //

CREATE TRIGGER after_update_vehiculo
AFTER UPDATE 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)
    ON DUPLICATE KEY UPDATE
        flotillaId = NEW.flotillaId,
        tipo = NEW.tipo,
        modelo = NEW.modelo,
        marca = NEW.marca,
        anio = NEW.anio,
        estado = NEW.estado,
        fechaVerificacion = NEW.fechaVerificacion;

    -- Actualizar o insertar en el nodo rutas
    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)
    ON DUPLICATE KEY UPDATE
        flotillaId = NEW.flotillaId,
        tipo = NEW.tipo,
        modelo = NEW.modelo,
        marca = NEW.marca,
        anio = NEW.anio,
        estado = NEW.estado,
        fechaVerificacion = NEW.fechaVerificacion;
END //

DELIMITER ;

UPDATE lcs1_principal.vehiculo
    SET estado = 'Mantenimiento'
    WHERE vehiculoId = 1;
```

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	1	11	Sedán	Corolla	Toyota	2020	Mantenimiento	2023-10-25

Figure 11: Nodo LCS1 - PRINCIPAL

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	1	11	Sedán	Corolla	Toyota	2020	Mantenimiento	2023-10-25

Figure 12: Nodo LCS2 - MANTENIMIENTO

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	1	11	Sedán	Corolla	Toyota	2020	Mantenimiento	2023-10-25

Figure 13: Nodo LCS3 - RUTAS

Listing 25: Script para que al eliminar un nuevo vehiculo en el nodo principal este lo hago en los otros dos nodos

```
1
2                   DELIMITER //
3
4                   CREATE TRIGGER before_delete_vehiculo
5                   BEFORE DELETE ON lcs1_principal.vehiculo
6                   FOR EACH ROW
7                   BEGIN
8                       DELETE FROM lcs2_mantenimiento.mantenimiento
9                       WHERE vehiculoId = OLD.vehiculoId;
10
11                      DELETE FROM lcs3_rutas.ruta
12                      WHERE vehiculoId = OLD.vehiculoId;
13
14                      DELETE FROM lcs3_rutas.transaccionCombustible
15                      WHERE vehiculoId = OLD.vehiculoId;
16
17                      DELETE FROM lcs2_mantenimiento.vehiculo
18                      WHERE vehiculoId = OLD.vehiculoId;
19
20                      DELETE FROM lcs3_rutas.vehiculo
21                      WHERE vehiculoId = OLD.vehiculoId;
22                  END //
23
24                  DELIMITER ;
25
26                  DELETE FROM lcs1_principal.vehiculo
27                  WHERE vehiculoId = 1;
```

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	2	74	SUV	Isuzu NPR	Freightliner	2015	Inactivo	<null>
2	3	36	Pickup	Kenworth T680	Mercedes-Benz	2021	Mantenimiento	<null>

Figure 14: Nodo LCS1 - PRINCIPAL

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	2	74	SUV	Isuzu NPR	Freightliner	2015	Inactivo	<null>
2	3	36	Pickup	Kenworth T680	Mercedes-Benz	2021	Mantenimiento	<null>

Figure 15: Nodo LCS2 - MANTENIMIENTO

	vehiculoId	flotillaId	tipo	modelo	marca	anio	estado	fechaVerificacion
1	2	74	SUV	Isuzu NPR	Freightliner	2015	Inactivo	<null>
2	3	36	Pickup	Kenworth T680	Mercedes-Benz	2021	Mantenimiento	<null>

Figure 16: Nodo LCS3 - RUTAS

5. Conclusiones

En este trabajo, se creó un sistema para gestionar flotillas usando tres bases de datos interconectadas. Se usaron herramientas como triggers para asegurar que los cambios en una base se reflejen en las otras, manteniendo la información actualizada y consistente. También se manejaron cuidadosamente las relaciones entre tablas para evitar errores.

Esto permitió realizar consultas combinadas y operaciones seguras, asegurando que el sistema funcione de manera eficiente y confiable.

En resumen, se logró un diseño práctico para administrar flotillas de manera distribuida.

Referencias Bibliográficas

References

[1] MySQL. (s.f.). *Documentación oficial de MySQL*. `https://dev.mysql.com/doc/`

[2] Celko, J. (2018). *SQL para Smarties: Programación avanzada en SQL*. Madrid, España: Anaya Multimedia.

[3] García-Molina, H., Ullman, J. D., & Widom, J. (2008). *Sistemas de bases de datos: El libro completo*. México: Pearson Educación.

[4] Elmasri, R., & Navathe, S. B. (2016). *Fundamentos de sistemas de bases de datos*. Madrid, España: Pearson.

[5] MySQL Tutorial. (s.f.). *Triggers en MySQL*. + `https://www.mysqltutorial.org/mysql-triggers/`

[6] Oracle. (s.f.). *Bases de datos distribuidas: Conceptos y diseño*. `https://docs.oracle.com/en/database/`

[7] Gómez, A. (2020). *Gestión de flotillas vehiculares: Herramientas y tecnologías*. Bogotá, Colombia: Editorial Limusa.

[8] W3Schools. (s.f.). *Tutorial de SQL*. `https://www.w3schools.com/sql/`

[9] Hernández, L. (2019). *Diseño de bases de datos relacionales*. Ciudad de México, México: Alfaomega.

[10] Pérez, J. (2021). *Buenas prácticas en MySQL: Optimización y seguridad*. Barcelona, España: Marcombo.