

Escuela Politécnica Nacional
ESFOT
Proyecto de Base de Datos para Infraestructura de Inventarios

Integrantes:

- Catagña Esteban

-Paredes Mateo

-Mario Endara

-Jordy Navarro

1. Modelado de Base de Datos y Diccionario de Datos

Objetivo:

- Crear un diseño eficiente y bien documentado para la base de datos *inventario_deportivo_v2*, utilizando el modelado entidad-relación (MER) y un diccionario de datos completo.

Actividades:

- **Diseñar el modelo conceptual, lógico y físico:**
 - **Entidades clave:** Propietarios, Administradores, Ubicaciones, Infraestructura Deportiva, Mantenimiento, Auditoría.
- **Práctica:** Crear un modelo entidad-relación que refleje las relaciones entre las entidades mencionadas.
- **Investigación:** Identificar buenas prácticas para mejorar la escalabilidad de bases de datos enfocadas en sistemas de inventario.
- **Importancia:** Un diseño adecuado mejora la eficiencia y el mantenimiento del sistema.

Estructura de la Base de Datos

```

• CREATE DATABASE IF NOT EXISTS inventario_deportivo_v2;
• USE inventario_deportivo_v2;

-- Tabla de Propietarios
• CREATE TABLE propietarios (
    id_propietario INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL
);

-- Tabla de Administradores
• CREATE TABLE administradores (
    id_administrador INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARBINARY(255) -- Para almacenar correos cifrados
);

-- Tabla de Ubicación
• CREATE TABLE ubicaciones (
    id_ubicacion INT AUTO_INCREMENT PRIMARY KEY,
    provincia VARCHAR(50),
    canton VARCHAR(50),
    coordinacion_zonal VARCHAR(50)
);

-- Tabla de Infraestructura Deportiva (Optimizada para CSV)
• CREATE TABLE infraestructura_deportiva (
    id_infraestructura INT AUTO_INCREMENT PRIMARY KEY,
    fecha_actualizacion DATETIME,
    id_ubicacion INT,
    estado ENUM('Bueno', 'Regular', 'Malo'),
    características_cubierta VARCHAR(50),
    longitud DECIMAL(12,6),
    latitud DECIMAL(12,6),
    fotografia_principal TEXT,
    otras_fotografias TEXT,
    id_propietario INT,
    id_administrador INT,
    tipo_propiedad ENUM('Publico', 'Privado'),
    hora_apertura TIME,
    hora_cierre TIME,
    uso_escenario VARCHAR(50),
    costo_uso DECIMAL(10,2),
    FOREIGN KEY (id_propietario) REFERENCES propietarios(id_propietario),
    FOREIGN KEY (id_administrador) REFERENCES administradores(id_administrador),
    FOREIGN KEY (id_ubicacion) REFERENCES ubicaciones(id_ubicacion)
);

```

```

-- Tabla de Mantenimiento (Relacionada correctamente con la nueva estructura)
CREATE TABLE mantenimiento (
    id_mantenimiento INT AUTO_INCREMENT PRIMARY KEY,
    id_infraestructura INT,
    fecha DATE,
    descripcion TEXT,
    costo DECIMAL(10,2),
    FOREIGN KEY (id_infraestructura) REFERENCES infraestructura_deportiva(id_infraestructura)
);
-- Diccionarios

```

Desarrollar un Diccionario de Datos Detallado

```

-- 1.Consultar la estructura de las tablas en MySQL
SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, IS_NULLABLE, COLUMN_TYPE, COLUMN_KEY, EXTRA
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_SCHEMA = 'inventario_deportivo_v2';

-- 2. Consultas de claves foraneas
SELECT TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME, REFERENCED_TABLE_NAME, REFERENCED_COLUMN_NAME
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE TABLE_SCHEMA = 'inventario_deportivo_v2' AND REFERENCED_TABLE_NAME IS NOT NULL;

-- 3.Consultar las restricciones de integridad (ON DELETE / ON UPDATE)
SELECT TABLE_NAME, COLUMN_NAME, CONSTRAINT_NAME, UPDATE_RULE, DELETE_RULE
FROM INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS
WHERE CONSTRAINT_SCHEMA = 'inventario_deportivo_v2';

-- 4.Consultar todas las tablas con sus comentarios y descripciones
SELECT TABLE_NAME, TABLE_TYPE, ENGINE, TABLE_ROWS, CREATE_TIME, UPDATE_TIME, TABLE_COMMENT
FROM INFORMATION_SCHEMA.TABLES
WHERE TABLE_SCHEMA = 'inventario_deportivo_v2';

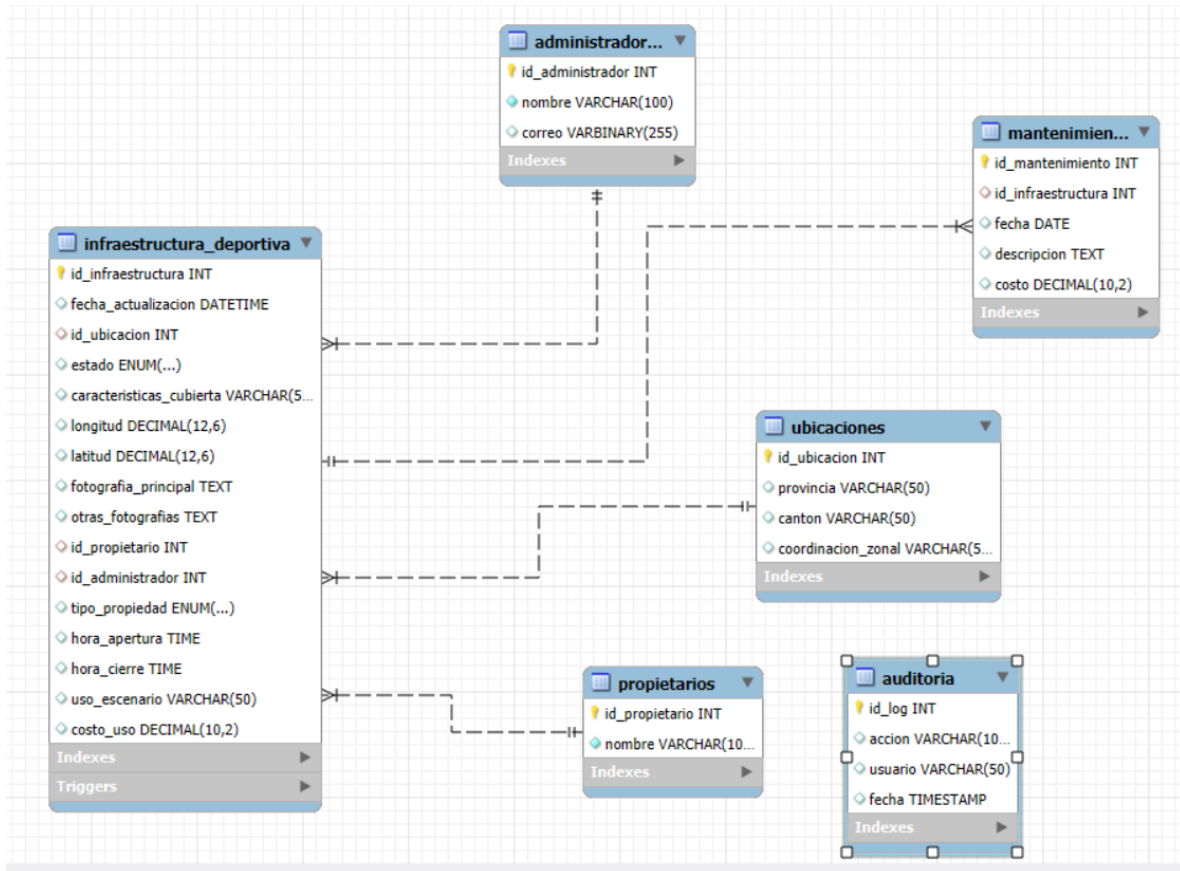
```

Modelo Entidad-Relación (MER):

- **Base de Datos:** inventario_deportivo_v2
- **Tablas y Relaciones:**
 - **Entidad Propietarios:** id_propietario (PK), nombre
 - **Entidad Administradores:** id_administrador (PK), nombre, correo (cifrado)
 - **Entidad Ubicaciones:** id_ubicacion (PK), provincia, cantón, coordinación_zonal
 - **Entidad Infraestructura Deportiva:** id_infraestructura (PK), fecha_actualizacion, id_ubicacion (FK), estado, características_cubierta, longitud, latitud, fotografia_principal, otras_fotografias, id_propietario (FK), id_administrador (FK), tipo_propiedad, hora_apertura, hora_cierre, uso_escenario, costo_uso

- **Entidad Mantenimiento:** id_mantenimiento (PK), id_infraestructura (FK), fecha, descripcion, costo
- **Entidad Auditoría:** id_log (PK), accion, usuario, fecha

Diagrama del Modelo Relacional:



(Falta el MER)

Desarrollo del Diccionario de Datos

- **Práctica:** Elaborar un diccionario detallado de todas las tablas, sus campos, tipos de datos y restricciones.
- **Investigación:** Herramientas para generar diccionarios de datos en MySQL y PostgreSQL.
- **Importancia:** Facilita la documentación y colaboración en el desarrollo del sistema.

Diccionario de Datos Completo:

Tabla	Campo	Tipo de Dato	Restricciones
Propietarios	id_propietario	INT	PRIMARY KEY, AUTO_INCREMENT
Propietarios	nombre	VARCHAR(100)	NOT NULL

Administradores	id_administrador	INT	PRIMARY KEY, AUTO_INCREMENT
Administradores	nombre	VARCHAR(100)	NOT NULL
Administradores	correo	VARBINARY(255)	Cifrado con AES
Ubicaciones	id_ubicacion	INT	PRIMARY KEY, AUTO_INCREMENT
Ubicaciones	provincia	VARCHAR(50)	NOT NULL
Ubicaciones	canton	VARCHAR(50)	NOT NULL
Ubicaciones	coordinacion_zonal	VARCHAR(50)	NOT NULL
Infraestructura Deportiva	id_infraestructura	INT	PRIMARY KEY, AUTO_INCREMENT
Infraestructura Deportiva	fecha_actualizacion	DATETIME	NOT NULL
Infraestructura Deportiva	id_ubicacion	INT	FOREIGN KEY
Infraestructura Deportiva	estado	ENUM('Bueno', 'Regular', 'Malo')	NOT NULL
Infraestructura Deportiva	caracteristicas_cubierta	VARCHAR(50)	NOT NULL
Infraestructura Deportiva	longitud	DECIMAL(12,6)	NOT NULL
Infraestructura Deportiva	latitud	DECIMAL(12,6)	NOT NULL
Infraestructura Deportiva	fotografia_principal	TEXT	NULL
Infraestructura Deportiva	otras_fotografias	TEXT	NULL
Infraestructura Deportiva	id_propietario	INT	FOREIGN KEY
Infraestructura Deportiva	id_administrador	INT	FOREIGN KEY
Infraestructura Deportiva	tipo_propiedad	ENUM('Publico', 'Privado')	NOT NULL
Infraestructura Deportiva	hora_apertura	TIME	NOT NULL
Infraestructura Deportiva	hora_cierre	TIME	NOT NULL
Infraestructura Deportiva	uso_escenario	VARCHAR(50)	NOT NULL
Infraestructura Deportiva	costo_uso	DECIMAL(10,2)	NOT NULL
Mantenimiento	id_mantenimiento	INT	PRIMARY KEY, AUTO_INCREMENT
Mantenimiento	id_infraestructura	INT	FOREIGN KEY

Mantenimiento	fecha	DATE	NOT NULL
Mantenimiento	descripcion	TEXT	NOT NULL
Mantenimiento	costo	DECIMAL(10,2)	NOT NULL
Auditoría	id_log	INT	PRIMARY KEY, AUTO_INCREMENT
Auditoría	accion	VARCHAR(100)	NOT NULL
Auditoría	usuario	VARCHAR(50)	NOT NULL
Auditoría	fecha	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

2. Seguridad, Auditoría y Control de Acceso

Objetivo:

Proteger los datos sensibles y controlar el acceso a la base de datos.

Actividades:

- **Implementación de Políticas de Acceso y Seguridad**

- **Práctica:** Creación de roles y permisos de usuario para Administradores y Auditores.

```
-- Crear usuario auditor con permisos de solo lectura
CREATE USER 'auditorEsteban'@'localhost' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
GRANT SELECT ON inventario_deportivo_v2.* TO 'auditorEsteban'@'localhost';
FLUSH PRIVILEGES;

-- Crear usuario administrador con permisos completos
CREATE USER 'adminMateo'@'localhost' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
GRANT ALL PRIVILEGES ON inventario_deportivo_v2.* TO 'adminMateo'@'localhost';
FLUSH PRIVILEGES;
```

- **Investigación:** Estrategias de seguridad en bases de datos de alta disponibilidad.
- **Importancia:** Protege la información sensible y previene accesos no autorizados.

Cifrado de Datos Sensibles

- **Práctica:** Aplicar cifrado en MySQL a los correos de los administradores.
- **Investigación:** Evaluación de algoritmos de cifrado como AES_ENCRYPT y su impacto en el rendimiento.
- **Importancia:** Protege información confidencial de los usuarios.

```
-- Encriontacion de datos sensibles
ALTER TABLE administradores MODIFY correo VARBINARY(255);
UPDATE administradores SET correo = AES_ENCRYPT('correo@example.com', 'llave_secreta') WHERE id_administrador = 1;
```

Habilitación de Auditoría y Registro de Eventos

- **Práctica:** Configurar logs de acceso y auditoría en MySQL.
- **Investigación:** Herramientas de auditoría en MySQL y PostgreSQL.
- **Importancia:** Permite el rastreo de cambios y la detección de actividades sospechosas.

```
-- Tabla de Auditoría
CREATE TABLE auditoria (
    id_log INT AUTO_INCREMENT PRIMARY KEY,
    accion VARCHAR(100),
    usuario VARCHAR(50),
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

- Creacion de triggers para un correcto funcionamiento

```
-- Trigger para Auditoría en Infraestructura
CREATE TRIGGER log_infraestructura_insert
AFTER INSERT ON infraestructura_deportiva
FOR EACH ROW
INSERT INTO auditoria (accion, usuario)
VALUES ('Se agregó un nuevo escenario', CURRENT_USER());
```

3. Respaldos y Recuperación de Datos

Objetivo: Asegurar la integridad y disponibilidad de los datos mediante técnicas de respaldo confiables.

Actividades:

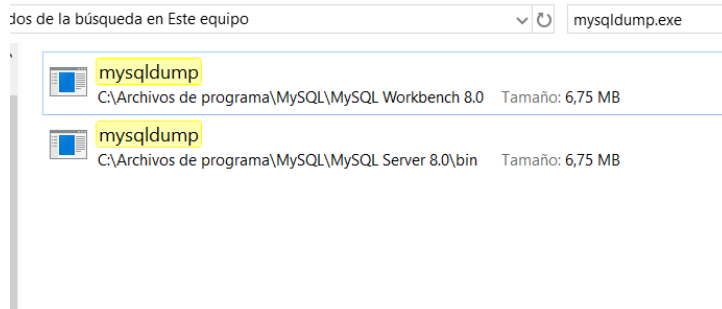
1. Crear respaldos completos (full backups).

Práctica: Utilizar mysqldump o herramientas similares para hacer respaldos completos de la base de datos.

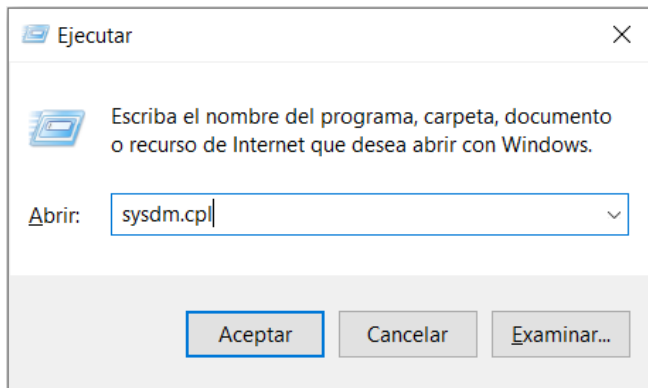
Investigación: Buscar estrategias de respaldo para bases de datos de gran tamaño y la mejor manera de gestionarlas.

Importancia del Conocimiento: Los respaldos completos permiten restaurar toda la base de datos ante una falla.

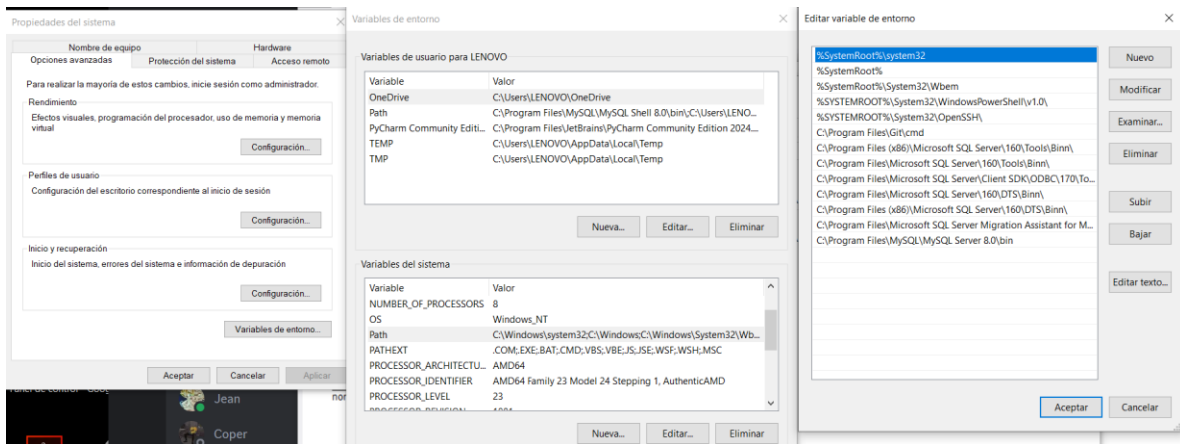
Para la instalación de mysqldump en caso de no tener instalado, debemos ir a nuestro equipo y buscar lo siguiente mysqldump.exe



Una vez encontrado los archivos nos vamos al archivo **/bin** y copiamos la ruta del archivo, una vez realizado lo anterior ingresamos a Windows + R y ejecutamos **sysdm.cpl**



Se nos apertura la siguiente ventana de propiedades del sistema, nos vamos a opciones avanzadas y variables de entorno seguimos en variables del sistema, buscamos Path, ponemos en editar, se nos nuevamente otra ventana ponemos en nuevo y pegamos la ruta del archivo **/bin** y lo demás es aceptar todos los cambios.



Una vez culminado los anteriores pasos se verifica la versión instalada en el cmd.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.5371]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\LENOVO>mysqldump
Usage: mysqldump [OPTIONS] database [tables]
OR      mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]
OR      mysqldump [OPTIONS] --all-databases [OPTIONS]
For more options, use mysqldump --help

C:\Users\LENOVO>
```

Para gestionar un respaldo con mysqldump nos ayudamos con la siguiente línea de código desde cmd, en donde -u es el usuario donde estamos gestionando la base de datos, -p es la contraseña que tiene el usuario y por último el nombre de la base de datos junto con el nombre del archivo donde vamos a guardar la base de datos.

```
C:\Users\LENOVO>mysqldump -u root -p ProyectoInfraEstrucutra > RespaldoDump.sql
Enter password: *****
```

Para verificar la ruta donde creo el archivo podemos guiarnos de **dir** la cual realizaremos en la siguiente línea

```
C:\Users\LENOVO>dir RespaldoDump.sql
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: F683-FFD8

Directorio de C:\Users\LENOVO

02/02/2025  21:17                4.395 RespaldoDump.sql
               1 archivos                4.395 bytes
               0 dirs 276.213.899.264 bytes libres
```

Por último para verificar el contenido de nuestro respaldo usamos el comando **type**.

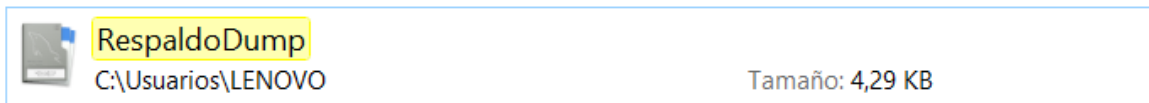
```
C:\Users\LENOVO>
C:\Users\LENOVO>type RespaldoDump.sql
-- MySQL dump 10.13  Distrib 8.0.41, for Win64 (x86_64)
--
-- Host: localhost    Database: ProyectoInfraEstrucutra
--
-- Server version      8.0.41

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

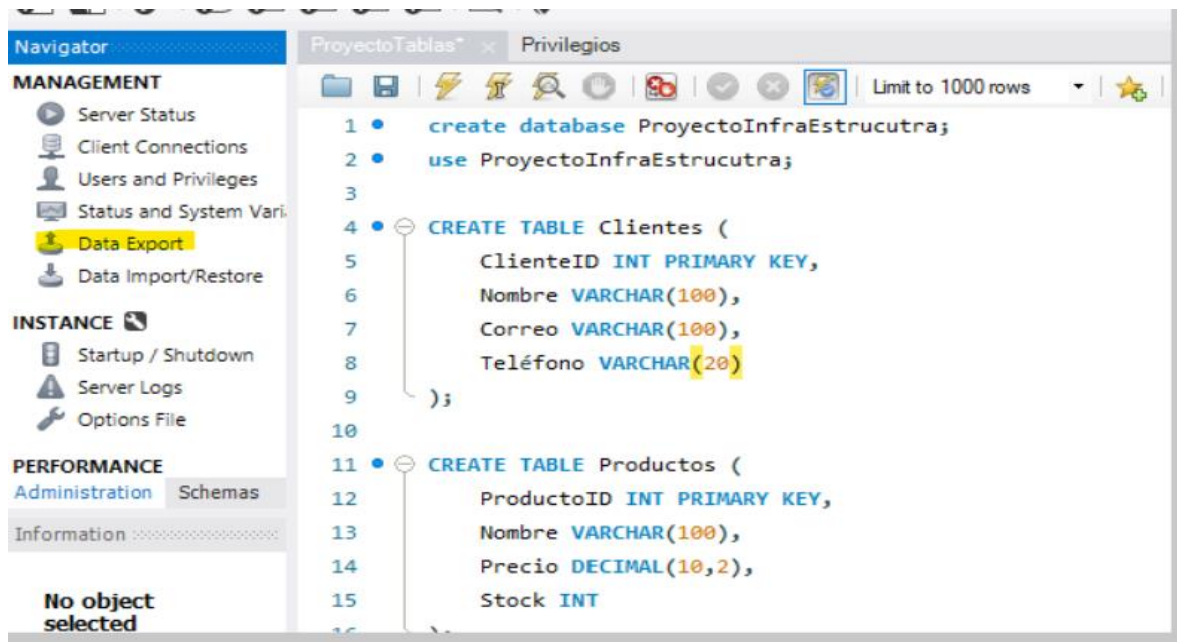
--
-- Table structure for table `auditoria`
--

DROP TABLE IF EXISTS `auditoria`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `auditoria` (
  `ID` int NOT NULL AUTO INCREMENT,
```

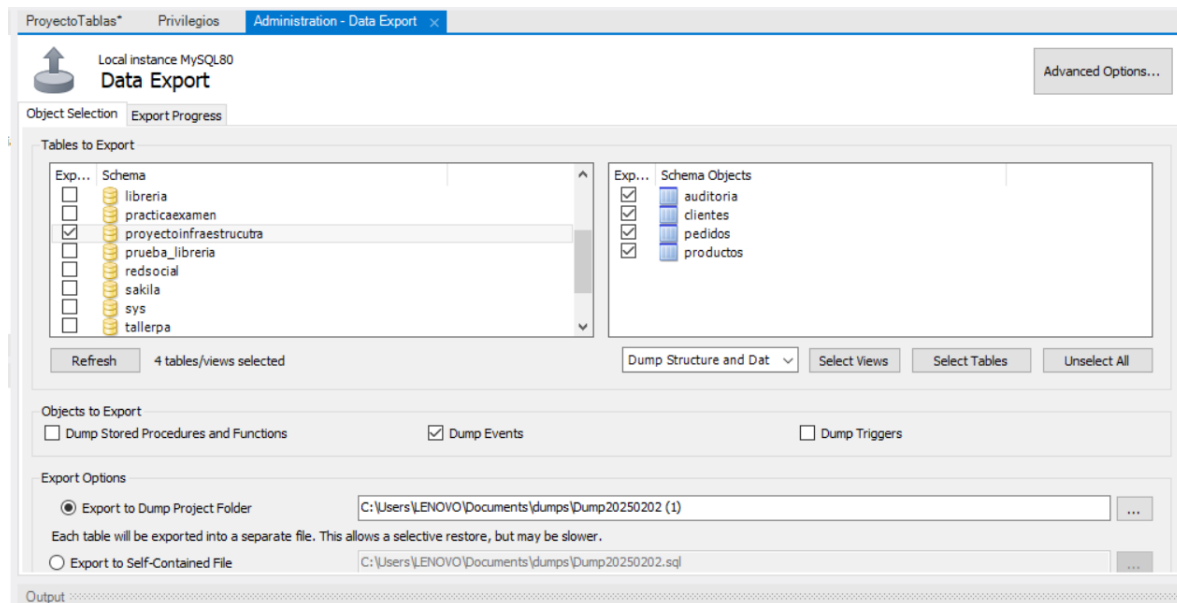
De igual manera podemos verificar el respaldo creado, en caso de no estipular una ruta de creación, el archivo se creará en la ruta desde donde los estamos gestionando



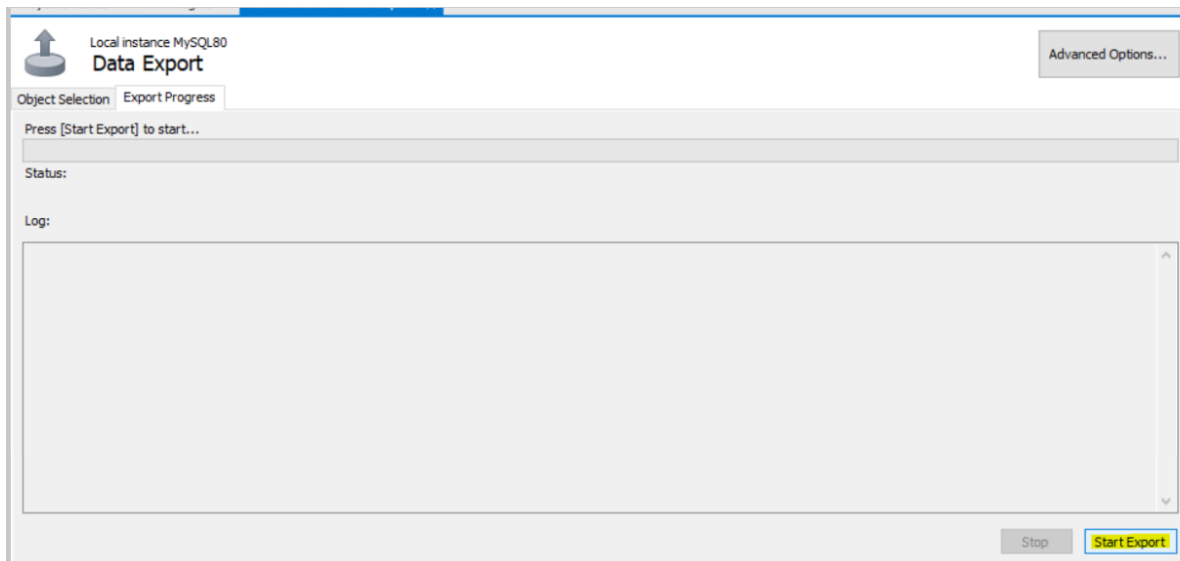
En caso de que se nos haga imposible por la línea de comandos, también podemos verificar el respaldo desde el entorno visual de Workbench, como primer paso nos dirigimos a Data Export



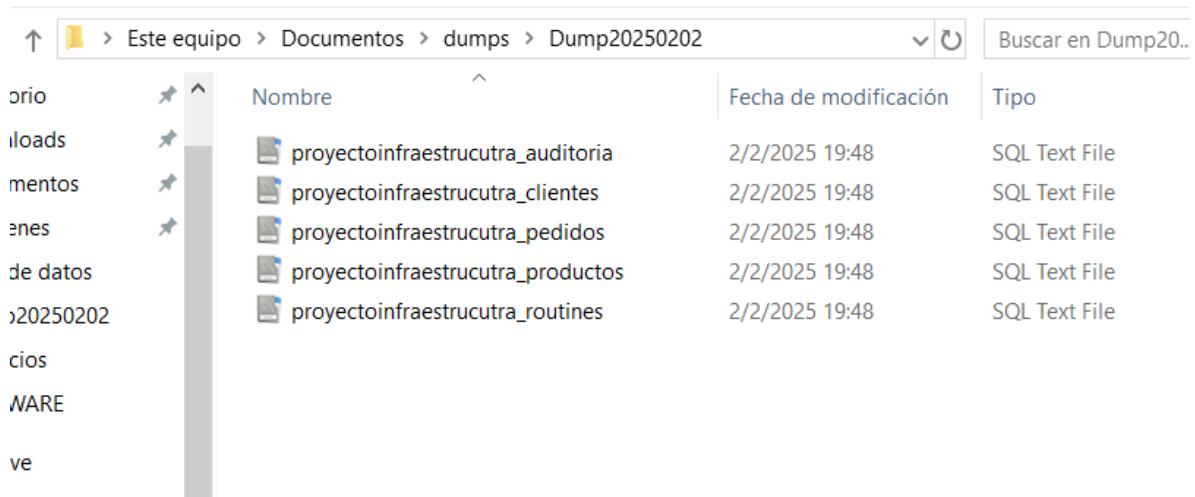
Elegimos la base de datos que queremos gestionar el respaldo con sus respectivas tablas, vistas o funciones.



Por último, damos en el comienzo de exportación



El respaldo se encuentra creado en una carpeta llamado dumps dentro de Documentos.



2. Configurar respaldos incrementales.

Práctica: Realizar respaldos incrementales para reducir el tiempo y espacio de almacenamiento.

Investigación: Investigar cómo realizar respaldos incrementales y cuándo es más conveniente utilizarlos.

Importancia del Conocimiento: Los respaldos incrementales permiten optimizar los recursos y acelerar los tiempos de recuperación.

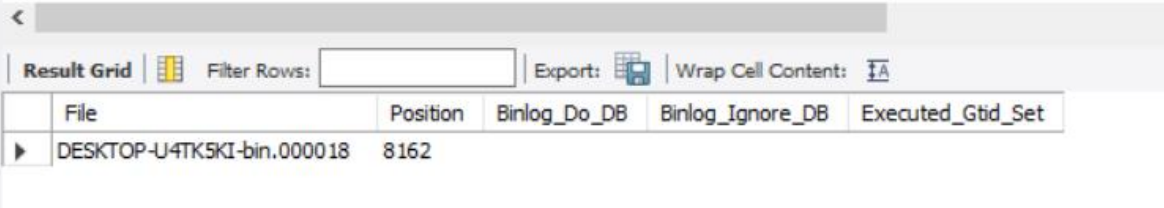
Los respaldos incrementales permiten reducir el tiempo de respaldo y el espacio requerido, ya que solo se copian los cambios. Si el espacio de almacenamiento es limitado, los respaldos incrementales son una buena opción, ya que ocupan menos espacio que los respaldos completos. Si tu base de datos tiene un volumen de cambios pequeño (por ejemplo, pocas inserciones, actualizaciones o eliminaciones), los respaldos incrementales son ideales, ya que solo se respaldan esos cambios. Si necesitas la capacidad de restaurar

la base de datos a un momento específico, los respaldos incrementales son esenciales. Combinados con un respaldo completo y los binlogs, permiten una recuperación precisa.

Desventaja del respaldo incremental:

Los respaldos incrementales dependen del último respaldo completo. Si el respaldo completo está corrupto o no está disponible, los respaldos incrementales no serán útiles.

```
89
90 • SHOW MASTER STATUS;
```



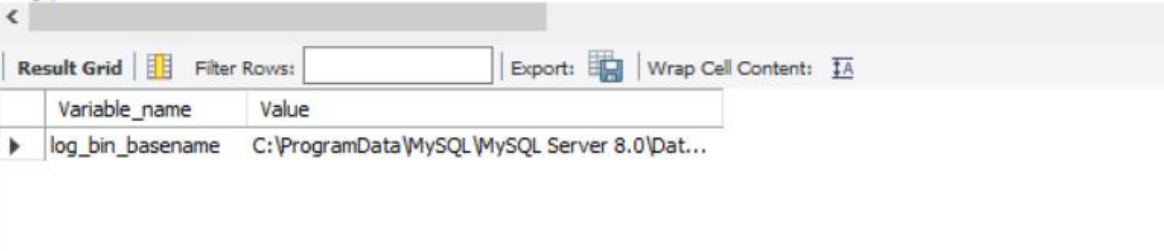
File	Position	Binlog_Do_DB	Binlog_Ignore_DB	Executed_Gtid_Set
DESKTOP-U4TK5KI-bin.000018	8162			

```
mysqlbinlog --start-position=45678 --stop-never /ruta/a/mysql-bin.000023 > /ruta/del/respaldo/incremental_$(date +%F).sql
```

Un ejemplo de uso es con mysqlbinlog, el cual utiliza los archivos bin que se crean en la base de datos, para realizar la anterior practica debemos conocer la posición y la ruta del archivo que queremos realizar el respaldo incremental.

```
C:\Users\LENOVO>mysqlbinlog --start-position=12299 --stop-never "C:/ProgramData/MySQL/MySQL Server 8.0/Data/DESKTOP-U4TK5KI-bin.000018" > "C:/Users/Lenovo/Documents/respaldo_incremental_$(date +%F).sql"
```

```
87 -- verificacion de archivos bin
88 • SHOW VARIABLES LIKE 'log_bin_basename';
89 -- verificacion de posicion archivos bin
90 • SHOW MASTER STATUS;
91 • SHOW BINARY LOGS;
92 • SHOW BINLOG EVENTS IN 'DESKTOP-U4TK5KI-bin.000018' LIMIT 10;
93
94
```



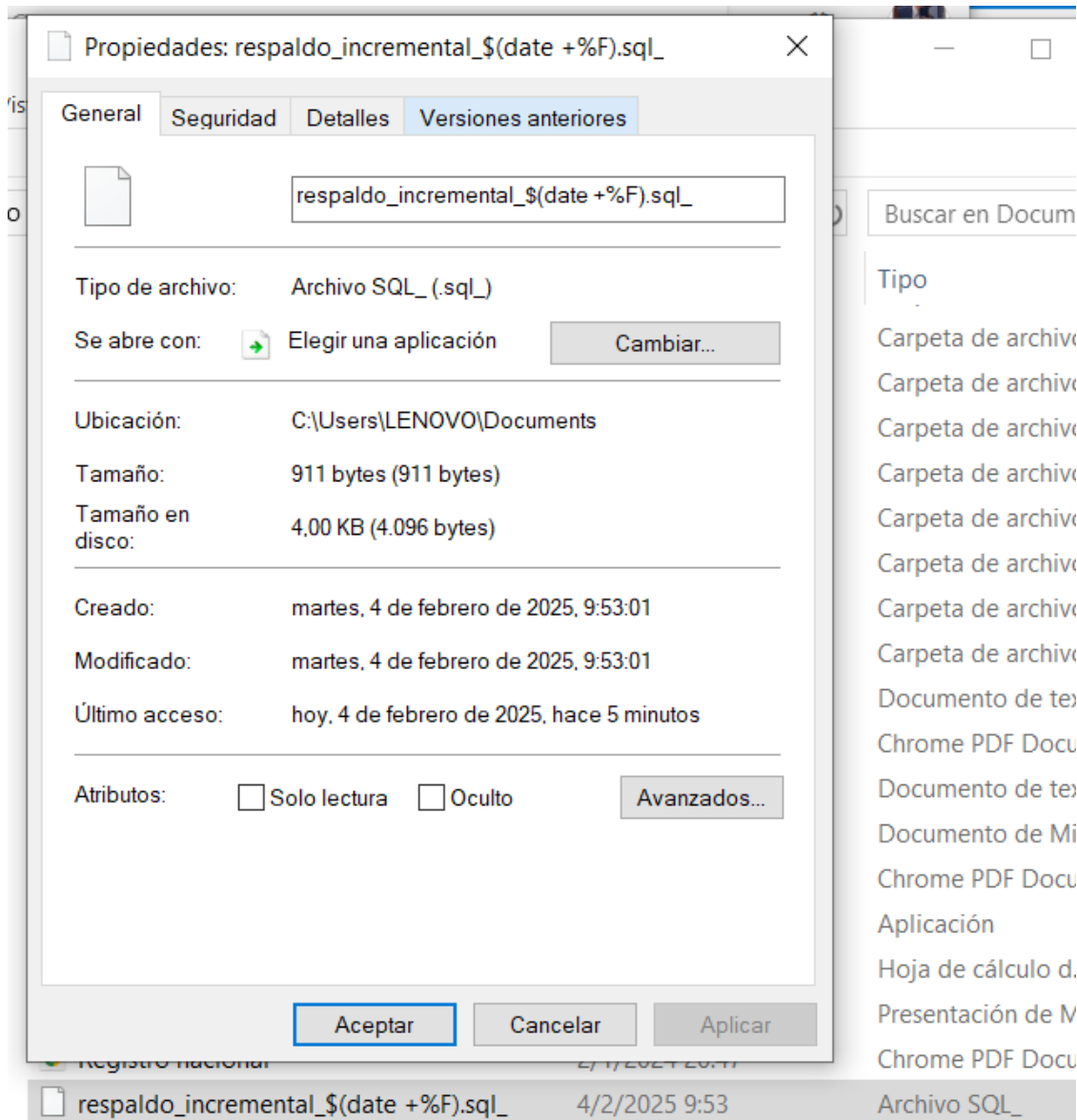
Variable_name	Value
log_bin_basename	C:\ProgramData\MySQL\MySQL Server 8.0\Dat...

Para obtener la información correspondiente a la anterior línea de código tenemos que ejecutar los siguientes códigos en el MySQL, para conocer, la ruta de los archivos bin, la posición del archivo, el nombre del archivo, con todo para poder cambiar la ruta de guardado podemos modificar solo la siguiente parte

```
> "C:/Users/Lenovo/Documents/respaldo_incremental_$(date +%F).sql"
```

De este modo el archivo se guardará de manera automática en documentos o la carpeta que

desean cambiar la ruta



3. Implementar respaldos en caliente (Hot Backups).

Práctica: Hacer respaldos sin interrumpir el servicio (por ejemplo, usando Percona XtraBackup).

Investigación: Investigar cómo hacer respaldos sin detener la base de datos.

Importancia del Conocimiento: Los respaldos en caliente son esenciales para bases de datos de producción que no pueden permitirse inactividad.

Es fundamental para bases de datos de producción que no pueden permitirse tiempos de inactividad, ya que el proceso de respaldo se realiza mientras las aplicaciones siguen accediendo a la base de datos.

XtraBackup: es una herramienta de respaldo para MySQL que realiza copias consistentes de bases de datos en vivo sin necesidad de detener la base de datos, lo cual es muy diferente

al uso de mysqldump que en ocasiones nos puede bloquear la base de datos mientras realiza el proceso de respaldo. **XtraBackup** crea copias físicas, lo que significa que se copian los archivos MySQL, incluyendo archivos innoDB, los archivos .frm de las tablas y por ultimo los archivos binarios.

Podemos mencionar que la gran ventaja de **XtraBackup** es que no necesita detener la base de datos para poder correrse, **XtraBackup** es una herramienta que está basada para Unix y Linux.

```
xtrabackup --backup --target-dir=/var/backups/mysql --user=root --password=tu_contraseña
```

Para poder gestionar un respaldo completo nos podemos ayudar de la siguiente linea:

-backup: Indica que se realizará un respaldo.

-target-dir=/var/backups/mysql: Directorio donde se guardará el respaldo.

-user=root --password=tu_contraseña: Credenciales de acceso.

```
xtrabackup --backup --target-dir=/var/backups/incremental_1 --incremental-basedir=/var/backups/mysql --user=root --password=tu_contraseña
```

Para poder realizar un respaldo incrementable nos guiaremos de la siguiente linea, el cual guardara los datos desde el ultimo respaldo

4. Optimización y Rendimiento de Consultas

Objetivo: Mejorar la eficiencia en la recuperación de datos mediante la optimización de consultas y el uso adecuado de índices.

Actividades:

1. Crear y gestionar índices.

Práctica: Implementar índices en las columnas más consultadas, como VueloID, ClienteID, etc.

Crear dos índices importantes

Investigación: Investigar sobre los tipos de índices más adecuados para bases de datos transaccionales y cómo afectan el rendimiento.

Importancia del Conocimiento: Los índices son cruciales para acelerar las consultas y mejorar el rendimiento general de la base de datos.

```
-- Parte 4 optimizacion
```

- CREATE INDEX idx_infraestructura_propietario ON infraestructura_deportiva(id_propietario);
- CREATE INDEX idx_infraestructura_ubicacion ON infraestructura_deportiva(id_ubicacion);

Estos dos índices serían los más importantes para reducir el tiempo de búsqueda en información relevante, ya que id_propietario y id_ubicacion nos pueden servir para filtrar o hacer uniones.

Por lo general si no utilizamos un índice la consulta debería ir fila por fila lo cual sería una pérdida de tiempo si la tabla cuenta con muchas filas

Tipos de índices:

-Índice B-Tree: es una estructura de datos en forma de árbol que permite búsquedas, inserciones y eliminaciones eficientes. Es el tipo de índice más común en bases de datos relacionales.

Es ideal para consultas de igualdad (=) y rangos (>, <, BETWEEN).

- Índice de texto (FULLTEXT): índice especializado para búsquedas de texto completo. Permite realizar búsquedas avanzadas en columnas de tipo TEXT o VARCHAR.

Operadores booleanos (AND, OR, NOT).

- Índice compuesto: Un índice compuesto (también llamado índice múltiple o índice compuesto) es un índice que se crea sobre varias columnas de una tabla.

2. Optimizar consultas SQL.

Práctica: Utilizar herramientas como EXPLAIN para identificar cuellos de botella en las consultas y optimizarlas.

Práctica: Aplicación de 3 join

Investigación: Investigar cómo hacer uso eficiente de las uniones (JOIN), subconsultas, y optimizar las consultas complejas.

Importancia del Conocimiento: Las consultas optimizadas aseguran un sistema rápido y eficiente, especialmente en sistemas con alta demanda.

```
97 • EXPLAIN SELECT * FROM infraestructura_deportiva
98 WHERE id_ubicacion = 1 AND estado = 'Bueno';
99
100
```

Result Grid							
Filter Rows:		Export:		Wrap Cell Content:			
	id	select_type	table	partitions	type	possible_keys	key
▶	1	SIMPLE	infraestructura_deportiva	NULL	ref	idx_infraestructura_ubicacion	idx_infraes

Con la ayuda de explain la consulta nos permite identificar posibles problemas como escaneos completos de tabla que pueden ralentizar las consultas.

```

-- uso de join con index
• SELECT
    i.id_infraestructura,
    i.estado,
    i.caracteristicas_cubierta,
    p.nombre AS propietario,
    u.provincia AS ubicacion,
    m.fecha AS mantenimiento_fecha
FROM
    infraestructura_deportiva i
JOIN
    propietarios p ON i.id_propietario = p.id_propietario
JOIN
    ubicaciones u ON i.id_ubicacion = u.id_ubicacion
LEFT JOIN
    mantenimiento m ON i.id_infraestructura = m.id_infraestructura
WHERE
    i.estado = 'Bueno' AND m.fecha >= CURDATE() - INTERVAL 30 DAY;

```

Para poder realizar una consulta centrada en los 3 índices generados nos ayudamos de join y left join, el cual nos ayudara filtrando el mantenimiento más actualizado que se ha realizado

3. Utilizar particionamiento de tablas.

Práctica: Dividir tablas grandes, como Reservas, en particiones según una clave (por ejemplo, por fecha).

Investigación: Investigar sobre l

os beneficios del particionamiento y cómo implementarlo en sistemas de bases de datos grandes.

Importancia del Conocimiento: El particionamiento de tablas mejora la escalabilidad y el rendimiento en bases de datos con gran volumen de datos.

Para poder realizar la participación en una tabla que ya fue creada debemos realizar lo siguiente.

Primero debemos de crear una nueva tabla con la misma estructura de la tabla que nos vamos a guiar.

```
> CREATE TABLE mantenimiento_new (  
    id_mantenimiento INT NOT NULL,  
    id_infraestructura INT NOT NULL,  
    fecha DATE NOT NULL,  
    descripcion TEXT,  
    costo DECIMAL(10,2),  
    PRIMARY KEY (id_mantenimiento, fecha),  
    INDEX idx_infraestructura (id_infraestructura)  
) PARTITION BY RANGE (TO_DAYS(fecha))(  
    PARTITION p2021_q1 VALUES LESS THAN (TO_DAYS('2021-02-01')),  
    PARTITION p2021_q2 VALUES LESS THAN (TO_DAYS('2021-04-01')),  
    PARTITION p2021_q3 VALUES LESS THAN (TO_DAYS('2021-06-01')),  
    PARTITION p2021_q4 VALUES LESS THAN (TO_DAYS('2022-08-01'))  
- );  
|
```

Hay que recordar que al momento de realizar la partición a la tabla esta debe estar incluida dentro de la tabla nueva para que no haya errores, de igual manera para poder particionar por fechas se recomienda utilizar **TO_DAYS**, el cual nos ayudara particionando la tabla por las fechas estipuladas.

```
-- Migracion de datos  
INSERT INTO mantenimiento_new SELECT * FROM mantenimiento;
```

Luego de realizar la creación de la nueva tabla y sus particiones comenzamos con la migración de datos, en todo caso para que esto funcione las tablas deben tener la misma estructura para que no haya inconvenientes.

```
DROP TABLE mantenimiento;  
-- renombre de la tabla nueva  
RENAME TABLE mantenimiento_new TO mantenimiento;
```

Como ultimo para poder mantener la anterior estructura, realizamos una eliminación de la tabla y un renombre, de esta forma estamos manteniendo la estructura correspondiente con la que se empezó.

5. Procedimientos almacenados, Vistas y Triggers

1. Práctica: Crear el procedimiento para calcular el total de una reserva

```
CREATE TABLE reservas (  
    id_reserva INT AUTO_INCREMENT PRIMARY KEY,  
    id_infraestructura INT,  
    fecha_reserva DATE,  
    precio_base DECIMAL(10,2),  
    descuento DECIMAL(5,2),  
    cargo_adicional DECIMAL(10,2),  
    FOREIGN KEY (id_infraestructura) REFERENCES infraestructura_deportiva(id_infraestructura)  
);  
DELIMITER //
```

Se procede a la creación de una tabla para poder obtener el total de una reserva

Creación del procedimiento almacenado para calcular el total de la reserva

```
CREATE PROCEDURE calcular_precio_total_reserva(  
    IN id_reserva INT,  
    OUT precio_total DECIMAL(10,2)  
)  
BEGIN  
    DECLARE precio_base DECIMAL(10,2);  
    DECLARE descuento DECIMAL(5,2);  
    DECLARE cargo_adicional DECIMAL(10,2);  
    -- Valor de la reserva  
    SELECT r.precio_base, r.descuento, r.cargo_adicional  
    INTO precio_base, descuento, cargo_adicional  
    FROM reservas r  
    WHERE r.id_reserva = id_reserva;  
    -- Calcular el precio total  
    SET precio_total = precio_base - (precio_base * descuento / 100) + cargo_adicional;  
END //  
DELIMITER ;
```

Al crear el procedimiento almacenado se puede observar que se realiza un porcentaje de descuento y un cargo por mantenimiento adicional, además se calcula el precio total.

Mostrar el procedimiento almacenado creado

```
-- Mostrar el procedimiento almacenado creado
CALL calcular_precio_total_reserva(1, @precio_total);
```

2. Crear vistas para simplificar consultas complejas

Práctica: Crear vistas que presenten información de varias tablas (ubicaciones, infraestructura y mantenimiento)

```
-- Creacion de vistas (ubicaciones,infraestructura_deportiva,mantenimiento)
DELIMITER //
CREATE VIEW vista_infraestructura_mantenimiento_ubicacion AS
SELECT
    u.provincia AS ubicacion_provincia,
    u.canton AS ubicacion_canton,
    u.coordinacion_zonal AS ubicacion_coordinacion_zonal,
    i.estado AS infraestructura_estado,
    i.caracteristicas_cubierta AS infraestructura_caracteristicas,
    i.costo_uso AS infraestructura_costo_uso,
    m.fecha AS mantenimiento_fecha,
    m.descripcion AS mantenimiento_descripcion,
    m.costo AS mantenimiento_costo
FROM
    infraestructura_deportiva i
JOIN
    ubicaciones u ON i.id_ubicacion = u.id_ubicacion
LEFT JOIN
    mantenimiento m ON i.id_infraestructura = m.id_infraestructura;
DELIMITER ;
```

A través de la vista se puede simplificar el acceso a datos complejos y optimizar consultas además de mejorar la seguridad de la base de datos.

```
DELIMITER ;
-- Verificación de ejecución de vista
SHOW FULL TABLES LIKE 'vista_infraestructura_mantenimiento_ubicacion';
```

Triggers

1. Creación de tablas para el uso de triggers

```
-- Triggers
CREATE TABLE pagos (
    id_pago INT AUTO_INCREMENT PRIMARY KEY,
    id_reserva INT,
    monto DECIMAL(10,2),
    fecha_pago DATE,
    FOREIGN KEY (id_reserva) REFERENCES reservas(id_reserva)
);

CREATE TABLE auditoria_reservas (
    id_auditoria INT AUTO_INCREMENT PRIMARY KEY,
    id_reserva INT,
    accion VARCHAR(50),
    fecha_accion DATETIME,
    campo_modificado VARCHAR(50),
    valor_anterior VARCHAR(255),
    valor_nuevo VARCHAR(255)
);

CREATE TABLE auditoria_pagos (
    id_auditoria INT AUTO_INCREMENT PRIMARY KEY,
    id_pago INT,
    accion VARCHAR(50),
    fecha_accion DATETIME,
    campo_modificado VARCHAR(50),
    valor_anterior VARCHAR(255),
    valor_nuevo VARCHAR(255)
);

DELIMITER //
```

Creación de Triggers

```
CREATE TRIGGER auditoria_reservas_update
AFTER UPDATE ON reservas
FOR EACH ROW
BEGIN
    DECLARE campo VARCHAR(50);
    DECLARE valor_anterior VARCHAR(255);
    DECLARE valor_nuevo VARCHAR(255);
    -- Verificar cada campo actualizado
    IF OLD.precio_base != NEW.precio_base THEN
        SET campo = 'precio_base';
        SET valor_anterior = OLD.precio_base;
        SET valor_nuevo = NEW.precio_base;
        INSERT INTO auditoria_reservas (id_reserva, accion, fecha_accion, campo_modificado, valor_ant
        VALUES (NEW.id_reserva, 'UPDATE', NOW(), campo, valor_anterior, valor_nuevo);
    END IF;
```

```

IF OLD.descuento != NEW.descuento THEN
    SET campo = 'descuento';
    SET valor_anterior = OLD.descuento;
    SET valor_nuevo = NEW.descuento;
    INSERT INTO auditoria_reservas (id_reserva, accion, fecha_accion, campo_modificado, valor_a
    VALUES (NEW.id_reserva, 'UPDATE', NOW(), campo, valor_anterior, valor_nuevo);
END IF;

IF OLD.cargo_adicional != NEW.cargo_adicional THEN
    SET campo = 'cargo_adicional';
    SET valor_anterior = OLD.cargo_adicional;
    SET valor_nuevo = NEW.cargo_adicional;
    INSERT INTO auditoria_reservas (id_reserva, accion, fecha_accion, campo_modificado, valor_a
    VALUES (NEW.id_reserva, 'UPDATE', NOW(), campo, valor_anterior, valor_nuevo);
END IF;

END //

DELIMITER ;
-- Triggers cuando se elimine un registro
DELIMITER //

CREATE TRIGGER auditoria_reservas_delete
AFTER DELETE ON reservas
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_reservas (id_reserva, accion, fecha_accion)
    VALUES (OLD.id_reserva, 'DELETE', NOW());
END //

DELIMITER ;
-- Trigger para auditoria de pago
DELIMITER //

CREATE TRIGGER auditoria_pagos_update
AFTER UPDATE ON pagos
FOR EACH ROW
BEGIN

```



```

DECLARE campo VARCHAR(50);
DECLARE valor_anterior VARCHAR(255);
DECLARE valor_nuevo VARCHAR(255);

-- Verificar cada campo actualizado
IF OLD.monto != NEW.monto THEN
    SET campo = 'monto';
    SET valor_anterior = OLD.monto;
    SET valor_nuevo = NEW.monto;
    INSERT INTO auditoria_pagos (id_pago, accion, fecha_accion, campo_modificado, valor_anterior,
    VALUES (NEW.id_pago, 'UPDATE', NOW(), campo, valor_anterior, valor_nuevo);
END IF;

IF OLD.fecha_pago != NEW.fecha_pago THEN
    SET campo = 'fecha_pago';
    SET valor_anterior = OLD.fecha_pago;
    SET valor_nuevo = NEW.fecha_pago;
    INSERT INTO auditoria_pagos (id_pago, accion, fecha_accion, campo_modificado, valor_anterior,
    VALUES (NEW.id_pago, 'UPDATE', NOW(), campo, valor_anterior, valor_nuevo);
END IF;
END //

DELIMITER ;
-- Trigger eliminacion
DELIMITER //

CREATE TRIGGER auditoria_pagos_delete
AFTER DELETE ON pagos
FOR EACH ROW
BEGIN
    INSERT INTO auditoria_pagos (id_pago, accion, fecha_accion)
    VALUES (OLD.id_pago, 'DELETE', NOW());
END //

DELIMITER ;

```

Resultados Obtenidos

- Con la base de datos bien estructurada se diseñó un modelo claro y organizado que nos ayudó con la facilitación en la gestión de la infraestructura sin complicaciones
- Con la seguridad se implementó cifrados en datos sensibles y se estableció permisos para la protección de datos
- Gracias a los respaldos confiables pudimos obtener unas copias seguras e incrementables para evitar pérdidas en caso de fallos

- También se optimizaron búsquedas con índices y particionamiento de tablas, logrando así un mejor rendimiento en el sistema

Conclusiones

Pudimos concluir que la base de datos para la infraestructura deportiva no solo permitió organizar y gestionar la información de manera eficiente, sino que también garantizó seguridad, rapidez y confiabilidad en el sistema. La implementación de buenas prácticas, como el cifrado de datos, los respaldos periódicos y la optimización de consultas, asegura que la base pueda crecer y adaptarse a nuevas necesidades sin comprometer su rendimiento. Además, la documentación detallada facilita el mantenimiento y futuras mejoras. En definitiva, este proyecto sienta las bases para un sistema robusto, seguro y escalable, que optimiza la administración y uso de los recursos deportivos.