

# Proyecto de Base de Datos Grupal BD

**Integrantes:** Edwin Sarango, Josué Mejía, Santiago Cumbal, Richard Padilla

## Instrucciones Generales

### 1. Modelado de Base de Datos y Diccionario de Datos

**Objetivo:** Crear un diseño eficiente y bien documentado para la base de datos, utilizando el modelado ER y un diccionario de datos completo.

#### Actividades:

##### 1. Diseñar el modelo conceptual, lógico y físico.

**Práctica:** Se ha diseñado un modelo entidad-relación (ER) que refleja las entidades clave en el contexto de autos deportivos, incluyendo:

- Autos Deportivos (nombre, modelo, velocidad, etc.)
- Marcas (nombre de la marca, relación con dueños)
- Dueños (propietarios de las marcas)
- Distribución por País (cantidad de autos en cada país)
- Países (nombre del país)

El modelo garantiza la integridad de los datos y facilita la escalabilidad.

**Investigación:** Para mejorar la escalabilidad del modelo de bases de datos en sistemas de autos deportivos, se deben considerar las siguientes prácticas:

- Normalización: Reducir la redundancia y mejorar la consistencia de los datos.
- Índices: Implementar índices en campos clave para acelerar las consultas.
- Particionamiento: Distribuir los datos de autos deportivos por regiones o marcas para mejorar el rendimiento.
- Uso de SQL cuando sea necesario: Bases de datos como MySQL Workbench pueden ser útiles para almacenar información estructurada, como imágenes y especificaciones técnicas.

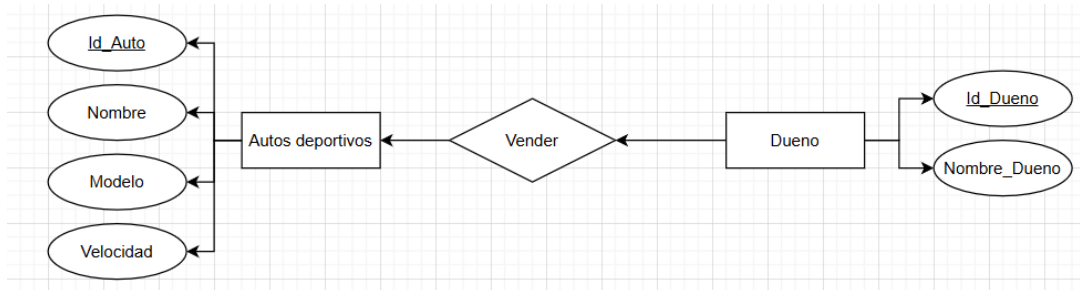
**Importancia del Conocimiento:** Diseñar bases de datos eficientes es crucial para:

- Optimizar consultas y tiempos de respuesta.
- Garantizar la consistencia y la integridad de los datos.
- Facilitar la escalabilidad y el mantenimiento a largo plazo.
- Mejorar la seguridad y el control de acceso a la información.

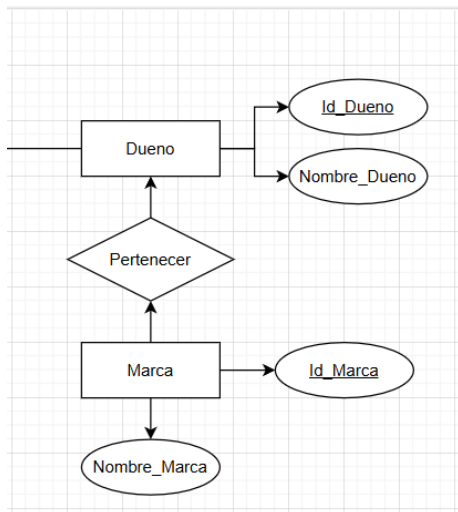
Para el presente proyecto hemos ideado un modelo de Autos deportivos como claves importantes tenemos a los nombres y modelos, Dueños de las marcas, cantidad de esas unidades en varios países, costo de los autos. Cada uno de estos datos estarán integrados en una base de datos fácil de entender al usuario y al público en general, resultantes de los conocimientos adquiridos por los estudiantes a lo largo del semestre.

## Diagrama entidad relación de Autos Deportivos

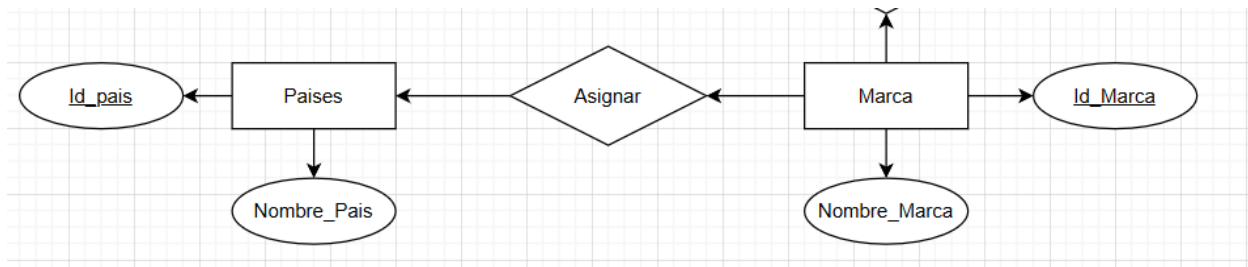
Para el presente proyecto hemos utilizado una herramienta gráfica para elaborar estos diagramas, Draw.io es una herramienta que integra todas las opciones gráficas de relación para generar nuestro modelo ER para el proyecto.



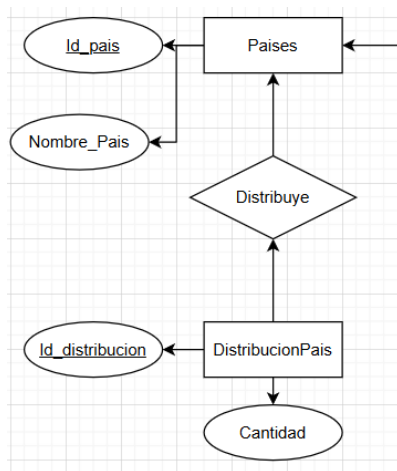
En cuanto a la primera relación obtenemos que es (M:1) ya que en primera instancia el dueño legítimo de la franquicia o la marca del carro es uno solo, pueden existir asociados, pero en este caso solo constaría como dueño legítimo uno solo, mientras que la relación M se refiere a que este único dueño de la marca puede fabricar varios carros, de varios modelos y así cumpliendo la relación.



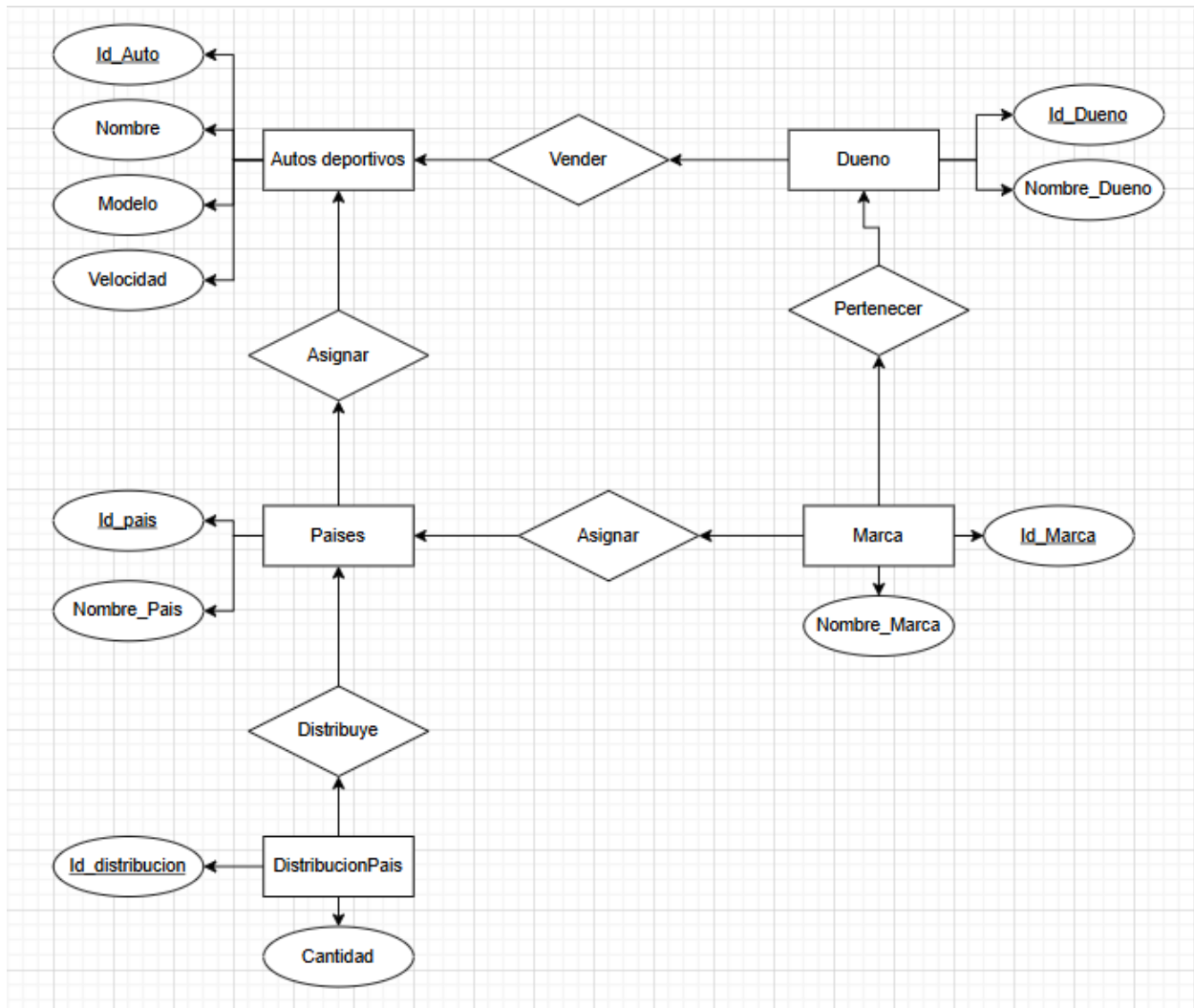
Relación Marca y Dueño es de (1:1) puesto que para el presente proyecto contamos solo con marcas reconocidas y mediante la investigación del proyecto se concluye que en general las marcas de autos reconocidas pertenecen a un solo dueño que en este caso para el proyecto es el dueño de la marca en sí, pero es importante realizar esta relación para futuras aplicaciones del código como consultas solo por marca y consultas solo por dueño por lo que se ha decidido quedar con ese modelo.



En la relación país y Marca tenemos el distintivo de ser (M:1) esto porque mediante investigaciones las marcas conocidas tienen su origen en los mismos países como puede ser Ferrari y Lamborghini siendo una marca italiana o BMW y Porsche siendo marcas alemanas en donde las fabrican, cumpliendo así la relación M que se tiene de los países de origen de estos autos.

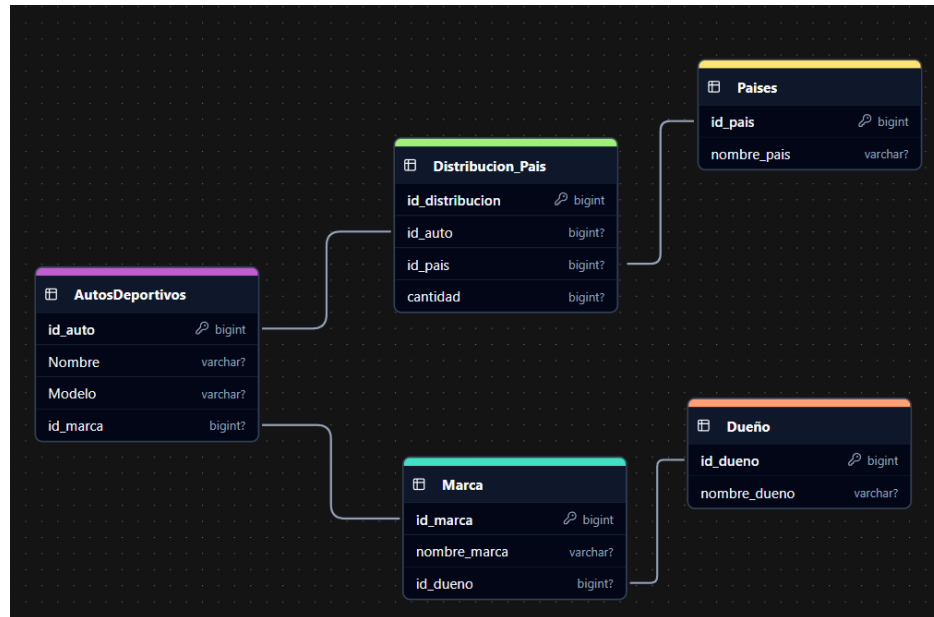


Como ultima relación del modelo ER del proyecto tenemos un agregado más como un estadístico en el que podemos conocer en qué países existe la mayor cantidad de autos o modelos vendidos por lo que la relación es de (M:M) suponiendo que mucha cantidad de autos es vendida en muchos países, al igual que muchos países distribuyen muchas cantidades de autos en donde más necesitan



De esta manera queda establecida el modelo entidad relación del proyecto en busca unificar datos estadísticos como la cantidad de autos, qué y cuantos autos existen y como un relevante conocer los dueños o marcas que están detrás de estas entidades.

## 1. Desarrollar un diccionario de datos detallado.



Mediante el uso de la plataforma ChartDB se organizó la estructura de los datos, como las tablas van a ir distribuidas y como los tipos de datos van a ser incorporados en el proyecto. Pues tenemos 5 tablas que nos ayudarán a realizar la organización mucho más detallada de nuestro modelo físico.

- Autos Deportivos ( id\_auto (PK), nombre, modelo ,costo ,id\_marca (FK) )
- Marcas (id\_marca (PK), nombre\_marca, id\_dueno (FK) )
- Dueños (id\_dueno (PK), nombre\_dueño)
- Distribución por País (id\_distribucion (PK), id\_auto (FK), id\_pais (FK), cantidad)
- Países (id\_pais (PK), nombre\_pais)

## Investigación sobre herramientas y métodos para generar diccionarios de datos

Algunas de las mejores herramientas para generar diccionarios de datos incluyen:

- MySQL Workbench - Permite documentar bases de datos y exportar diccionarios.
- DBDraw.io - Herramienta online para diagramar y documentar bases de datos.
- ChartBD – Herramienta para el modelado físico de la base de datos
- Generación manual en Excel o Google Sheets - Para proyectos pequeños, documentar los datos en una hoja de cálculo bien estructurada es útil.

## Importancia del Diccionario de Datos

- Consistencia: Define nombres y estructuras de datos de manera uniforme.
- Facilita la colaboración: Desarrolladores, analistas y administradores pueden entender la base de datos fácilmente.
- Evita errores: Asegura que los datos sean manejados correctamente.
- Optimiza el mantenimiento: Permite realizar cambios estructurados con menor impacto

## 2. Definir las restricciones de integridad referencial y eliminación - update.

### Práctica: Claves Primarias y Foráneas

Para garantizar la coherencia de los datos, se han definido claves primarias (PK) y claves foráneas (FK) en las tablas de la base de datos AutosDeportivosDB.

Las restricciones establecidas incluyen:

- Claves Primarias (PK): Garantizan que cada fila en una tabla tenga un identificador único.
- Claves Foráneas (FK): Aseguran que los datos referenciados existan en la tabla padre.
- Acciones en Eliminación y Actualización: Se han definido reglas para manejar la eliminación y actualización de registros. **Eliminación – Update de cascada, set NULL, restrict, Constrains (NO ACTION)**

| Tabla            | Clave Primaria (PK) | Clave Foránea (FK) y Relación     | Restricción de Eliminación | Restricción de Actualización |
|------------------|---------------------|-----------------------------------|----------------------------|------------------------------|
| Duenos           | id_dueno            | -                                 | RESTRICT                   | CASCADE                      |
| Marcas           | id_marca            | id_dueno → Dueños.id_dueno        | SET NULL                   | CASCADE                      |
| AutosDeportivos  | id_auto             | id_marca → Marcas.id_marca        | SET NULL                   | CASCADE                      |
| Países           | id_pais             | -                                 | RESTRICT                   | CASCADE                      |
| DistribucionPais | id_distribucion     | id_auto → AutosDeportivos.id_auto | CASCADE                    | CASCADE                      |
|                  |                     | id_pais → Paises.id_pais          | CASCADE                    | CASCADE                      |

### Explicación de las Restricciones

#### CASCADE en Eliminación y Actualización:

Si se elimina un auto deportivo, también se eliminan sus registros en Distribución por País.

Si se actualiza un id\_auto, los cambios se reflejan en la tabla de distribución.

#### SET NULL en Eliminación:

Si una marca se elimina, los autos deportivos que pertenecían a esa marca quedan con id\_marca = NULL.

Si un dueño se elimina, sus marcas quedan con id\_dueno = NULL.

#### RESTRICT en Eliminación:

No se puede eliminar un país si tiene autos registrados en Distribución por País.

No se puede eliminar un dueño si tiene marcas activas.

#### CASCADE en Actualización:

Si cambia el id\_dueno, id\_marca, o id\_pais, la relación se actualiza automáticamente en todas las tablas dependientes.

**Investigación:** Investigar cómo la integridad referencial garantiza que las relaciones entre tablas sean consistentes, evitando errores como:

- Registros huérfanos: Evita que una tabla referencie a un valor inexistente en otra.
- Datos inconsistentes: Asegura que los cambios en una tabla se reflejen correctamente en otras.
- Errores de eliminación y actualización: Previene la eliminación de registros que aún tienen dependencias.

**Las mejores prácticas incluyen:**

1. Definir correctamente las claves primarias y foráneas.
2. Usar CASCADE solo cuando sea seguro eliminar datos relacionados.
3. Evitar SET NULL si el campo foráneo es obligatorio.
4. Utilizar RESTRICT o NO ACTION cuando se necesite evitar eliminaciones accidentales.
5. Auditar periódicamente las relaciones para prevenir datos inconsistentes.

**Importancia del Conocimiento:** Las restricciones de integridad aseguran que los datos no se corrompan.

- Evita la pérdida accidental de datos esenciales.
- Asegura que las relaciones entre entidades sean válidas en todo momento.
- Facilita el mantenimiento de la base de datos y mejora su fiabilidad.
- Garantiza que los datos sean consistentes y precisos en toda la aplicación.

## 2. Seguridad, Auditoría y Control de Acceso

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

**Actividades:**

```
-- Crear usuarios
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'Admin123';
CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'Usuario123';
CREATE USER 'auditor'@'localhost' IDENTIFIED BY 'Auditor123';

-- Crear rol de Administrador
CREATE ROLE 'Administrador';
GRANT ALL PRIVILEGES ON AutosDeportivosDB.* TO 'Administrador';

-- Crear rol de Usuario (solo consulta)
CREATE ROLE 'Usuario';
GRANT SELECT ON AutosDeportivosDB.AutosDeportivos TO 'Usuario';
GRANT SELECT ON AutosDeportivosDB.Paises TO 'Usuario';

-- Crear rol de Auditor (solo lectura y acceso a logs)
CREATE ROLE 'Auditor';
GRANT SELECT ON AutosDeportivosDB.* TO 'Auditor';

-- Asignar roles a usuarios
GRANT 'Administrador' TO 'admin'@'localhost';
GRANT 'Usuario' TO 'usuario'@'localhost';
GRANT 'Auditor' TO 'auditor'@'localhost';
```

```
-- Verificar roles y privilegios asignados
SHOW GRANTS FOR 'admin'@'localhost';
SHOW GRANTS FOR 'usuario'@'localhost';
SHOW GRANTS FOR 'auditor'@'localhost';

-- Asignar roles y privilegios a Usuario
GRANT USAGE ON *.* TO 'usuario'@'localhost';
GRANT 'Usuario' TO 'usuario'@'localhost';
CREATE ROLE 'Usuario';

-- Otorgar privilegios al rol 'Usuario'
GRANT SELECT ON AutosDeportivosDB.AutosDeportivos TO 'Usuario';
GRANT SELECT ON AutosDeportivosDB.Paises TO 'Usuario';
```

#### Explicación:

##### Crear usuarios:

- CREATE USER 'admin'@'localhost' IDENTIFIED BY 'Admin123';  
Crea un usuario llamado admin con la contraseña Admin123 que puede conectarse solo desde localhost.
- CREATE USER 'usuario'@'localhost' IDENTIFIED BY 'Usuario123';  
Crea un usuario llamado usuario con la contraseña Usuario123, igualmente limitado a conexiones desde localhost.
- CREATE USER 'auditor'@'localhost' IDENTIFIED BY 'Auditor123';  
Crea un usuario llamado auditor con la contraseña Auditor123, también limitado a localhost.

##### Crear roles:

- CREATE ROLE 'Administrador';  
Crea un rol denominado Administrador.
- CREATE ROLE 'Usuario';  
Crea un rol denominado Usuario.
- CREATE ROLE 'Auditor';  
Crea un rol denominado Auditor.

##### Asignar privilegios a los roles:

- GRANT ALL PRIVILEGES ON AutosDeportivosDB.\* TO 'Administrador';  
Asigna todos los privilegios sobre la base de datos AutosDeportivosDB al rol Administrador, lo que incluye permisos de lectura, escritura, creación, modificación, etc.
- GRANT SELECT ON AutosDeportivosDB.AutosDeportivos TO 'Usuario';  
Asigna el privilegio de solo lectura (SELECT) sobre la tabla AutosDeportivos de la base de datos AutosDeportivosDB al rol Usuario.
- GRANT SELECT ON AutosDeportivosDB.Paises TO 'Usuario';  
Asigna el privilegio de solo lectura (SELECT) sobre la tabla Paises de la base de datos AutosDeportivosDB al rol Usuario.
- GRANT SELECT ON AutosDeportivosDB.\* TO 'Auditor';  
Asigna el privilegio de solo lectura (SELECT) sobre todas las tablas de la base de datos AutosDeportivosDB al rol Auditor.



#### Asignar roles a usuarios:

- GRANT 'Administrador' TO 'admin'@'localhost';  
Asigna el rol Administrador al usuario admin.
- GRANT 'Usuario' TO 'usuario'@'localhost';  
Asigna el rol Usuario al usuario usuario.
- GRANT 'Auditor' TO 'auditor'@'localhost';  
Asigna el rol Auditor al usuario auditor.

#### Verificar roles y privilegios asignados:

- SHOW GRANTS FOR 'admin'@'localhost';  
Muestra los privilegios asignados al usuario admin.
- SHOW GRANTS FOR 'usuario'@'localhost';  
Muestra los privilegios asignados al usuario usuario.
- SHOW GRANTS FOR 'auditor'@'localhost';  
Muestra los privilegios asignados al usuario auditor.

#### Asignar roles y privilegios adicionales:

- GRANT USAGE ON \*.\* TO 'usuario'@'localhost';  
Este comando otorga permisos de uso de recursos de la base de datos al usuario usuario (aunque este es un comando redundante ya que el privilegio USAGE es generalmente implícito en los privilegios que se otorgan sobre bases de datos y tablas específicas).
- CREATE ROLE 'Usuario';  
Aquí se crea de nuevo el rol Usuario (esto parece innecesario ya que ya se creó antes).
- GRANT SELECT ON AutosDeportivosDB.AutosDeportivos TO 'Usuario';  
Asigna nuevamente el privilegio de lectura en la tabla AutosDeportivos al rol Usuario.
- GRANT SELECT ON AutosDeportivosDB.Paises TO 'Usuario';  
Asigna nuevamente el privilegio de lectura en la tabla Paises al rol Usuario.

## 1. Implementar políticas de acceso y seguridad.

**Práctica:** Roles de Usuario

- Administrador: Acceso total a la base de datos (puede leer, escribir, modificar y eliminar datos).
- Usuario: Acceso limitado para consultar autos deportivos y su distribución.
- Auditor: Solo puede leer registros y revisar logs de auditoría.

**Investigación:** Investigar sobre los mejores enfoques para la seguridad en bases de datos en entornos de alta disponibilidad.

Mejores Enfoques para Seguridad en Bases de Datos

- Principio de Menor Privilegio
  1. Asignar permisos estrictamente necesarios a cada usuario.
- Cifrado de Datos Sensibles
  2. Usar AES\_ENCRYPT() en MySQL para proteger información confidencial como contraseñas.
- Autenticación Segura
  3. Usar contraseñas fuertes y políticas de caducidad de contraseñas.
- Registros de Auditoría
  4. Monitorear actividades sospechosas con logs de acceso.
- Firewalls y Listas de Control de Acceso (ACLs)
  5. Restringir conexiones solo a direcciones IP de confianza.
- Respaldo de Datos Periódico
  6. Implementar backups automáticos para evitar pérdida de datos.

**Importancia del Conocimiento:** El control adecuado de acceso previene fugas de información y mejora la seguridad general.

**Importancia del Control de Acceso**

- Previene fugas de información y accesos no autorizados.
- Protege la integridad de los datos frente a modificaciones indebidas.
- Facilita la auditoría y rastreo de acciones sospechosas.
- Mejora la seguridad en sistemas de alta disponibilidad.

## 2. Cifrado de datos sensibles.

**Práctica:** El cifrado de datos sensibles es crucial para proteger información como contraseñas, detalles de pago, y otros datos privados. Para cifrar datos en MySQL, puedes usar funciones como AES\_ENCRYPT.

**Impacto en el rendimiento:**

- Rendimiento: El cifrado introduce una sobrecarga en las operaciones de la base de datos, ya que cada vez que se inserta o recupera información cifrada, la base de datos debe realizar operaciones adicionales de cifrado/descifrado. En bases de datos con grandes volúmenes de datos, esto puede tener un impacto significativo en el rendimiento.
- Optimización: Para mejorar el rendimiento, es recomendable cifrar únicamente los campos sensibles (como contraseñas), mientras que otros datos no deben ser cifrados. Además, el uso de índices puede ser limitado en campos cifrados, lo que puede afectar la rapidez de las consultas.

#### Investigación:

- Seguridad de la clave: La clave utilizada para el cifrado ('clave\_secreta' en el ejemplo) debe almacenarse de manera segura, preferiblemente fuera de la base de datos (por ejemplo, en un servicio de gestión de claves o un archivo seguro).
- Alternativas: Existen otras opciones como la función bcrypt para contraseñas, que se recomienda por su capacidad de hacer hashing en lugar de cifrado, y su resistencia a ataques de fuerza bruta.

### 3. Habilitar auditoría y registrar eventos de base de datos.

La auditoría es fundamental para el monitoreo de accesos y cambios en los datos de la base de datos. Permite detectar comportamientos sospechosos y mantener un registro detallado de las actividades realizadas por los usuarios.

#### Práctica:

**Activación de logs de auditoría:** Tanto en MySQL como en SQLServer, puedes habilitar el registro de eventos de base de datos para auditar actividades como inserciones, actualizaciones, eliminaciones, y accesos a la base de datos.

#### En MySQL:

MySQL ofrece un plugin de auditoría llamado MySQL Enterprise Audit Plugin, disponible en las ediciones empresariales.

```
INSTALL PLUGIN audit_log SONAME 'audit_log.so';  
SET GLOBAL general_log = 'ON';  
SET GLOBAL log_output = 'TABLE';
```

- Rendimiento: Los logs de auditoría pueden afectar el rendimiento, ya que cada acción de la base de datos será registrada. Para evitar sobrecargar el sistema, es recomendable establecer una política de retención de logs (por ejemplo, eliminar logs antiguos después de un cierto período).
- Optimización: El uso de herramientas especializadas de auditoría (como las mencionadas) permite un registro más eficiente y detallado, con menos impacto en el rendimiento.

#### Consideraciones adicionales:

- Acceso a logs: Es importante controlar quién tiene acceso a los logs de auditoría, ya que contienen información sensible sobre las actividades de los usuarios.
- Herramientas adicionales: Existen herramientas de auditoría adicionales que puedes integrar, como pgAudit en PostgreSQL o soluciones de terceros que ofrecen funcionalidades avanzadas de monitoreo y auditoría.

### 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.

```
C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqldump --version
mysqldump Ver 8.0.38 for Win64 on x86_64 (MySQL Community Server - GPL)

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 9.0.0 MySQL Community Server - GPL

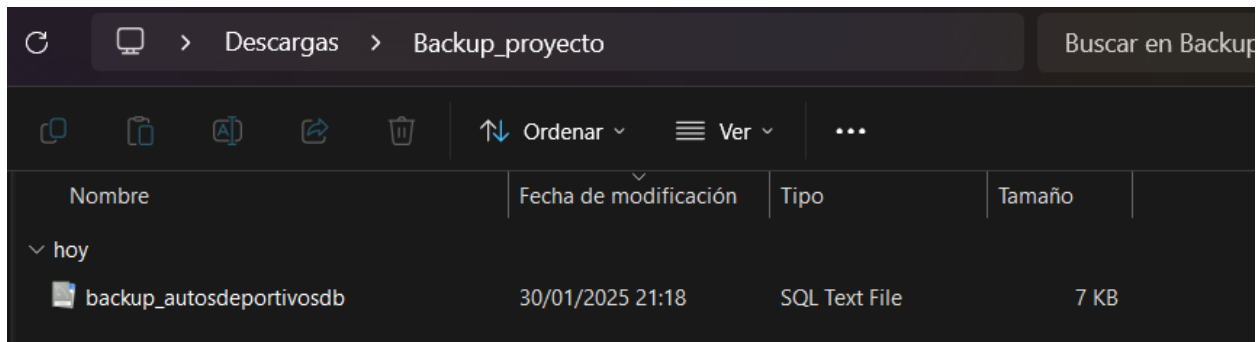
Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases
-> show databases
-> show databases;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
for the right syntax to use near 'show databases' at line 2
mysql> show databases;
+-----+
| Database |
+-----+
| autosdeportivosdb |
| basetrigger |
| concursococina |
+-----+
```

```
C:\Program Files\MySQL\MySQL Server 8.0\bin> mysqldump -u root -p autosdeportivosdb > C:\Users\santp\Downloads\Backup_pr
oyecto\backup_autosdeportivosdb.sql
Enter password: *****
```



| Descargas > Backup_proyecto |                       |               |        |  |
|-----------------------------|-----------------------|---------------|--------|--|
| Buscar en Backup            |                       |               |        |  |
| Ordenar Ver                 |                       |               |        |  |
| Nombre                      | Fecha de modificación | Tipo          | Tamaño |  |
| ▼ hoy                       |                       |               |        |  |
| backup_autosdeportivosdb    | 30/01/2025 21:18      | SQL Text File | 7 KB   |  |

Mysqldump es una herramienta que guarda los datos de una base de datos en un archivo. Para usarlo, escribes un comando en la terminal con tu usuario, la base de datos que quieres respaldar y el nombre del archivo donde se guardará la copia. La herramienta pide la contraseña para acceder a los datos y colocar la ruta donde quieres llevar el respaldo en este caso se llevó a una nueva carpeta.

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

- Si tu base de datos soporta binarios, herramientas como Percona XtraBackup permiten respaldar datos en un formato binario sin dividir ni afectar el rendimiento.
- Especifica un tamaño de paquete mayor para manejar mejor las consultas de bases de datos grandes en mysqldump en este caso se usaría lo siguiente: `mysqldump --max-allowed-packet=1G -u root -p nombre_base_datos > respaldo.sql`

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

- Su importancia radica en que permiten restaurar toda la base de datos en caso de fallos, errores humanos o ataques cibernéticos. Al tener una copia exacta de todos los datos, se puede recuperar el sistema rápidamente, minimizando el impacto en las operaciones y evitando pérdidas significativas de información.

## 2. Configurar respaldos incrementales.

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

- Para el momento mysqldump no cuenta con la capacidad de poder hacer respaldos de forma incremental debido a las limitaciones que se tiene, por lo tanto, la mejor herramienta para poder trabajar de esta forma es con **Percona XtraBackup** que es una herramienta gratuita para hacer respaldos rápidos y seguros de bases de datos MySQL sin detener su funcionamiento la cual se usa en sistemas operativos Linux lo cual hace complicado su descarga pero con una funcionalidad bastante elevada, el código que se necesita para poder generar un backup de esta forma es:

```
xtrabackup --backup --target-dir=/home/usuario/backup_incr1 --incremental-  
basedir=/home/usuario/backup_full --user=root --password=tu_contraseña
```

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

- Un respaldo incremental guarda solo los datos modificados desde el último respaldo completo o incremental, ahorrando tiempo y espacio.  
En los casos que se pueden utilizar:
- Cuando la base de datos es grande y los cambios diarios son menores.
- Para sistemas donde el tiempo de respaldo debe ser mínimo.
- En entornos que requieren frecuentes puntos de restauración.
- 

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

- Los respaldos incrementales son fundamentales para la eficiencia en la gestión de bases de datos, especialmente cuando su tamaño es considerable y los cambios realizados son frecuentes, pero no masivos. A diferencia de los respaldos completos, que almacenan toda la información, los incrementales registran únicamente los datos modificados desde el último respaldo, lo que permite ahorrar espacio de almacenamiento y tiempo durante

el proceso. Además, son especialmente útiles en entornos empresariales o críticos donde es necesario realizar copias de seguridad de forma constante sin afectar el rendimiento del sistema. Al reducir el volumen de datos a respaldar, no solo se optimizan los recursos disponibles, sino que también se acelera el tiempo necesario para recuperar la información en caso de una eventual falla.

### 3. Implementar respaldos en caliente (Hot Backups).

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

De igual forma Percona XtraBackup se encarga de hacer los **Hot Backups** que normalmente no puede generar el programa de mysqldump ya que no permite realizar respaldos en caliente sin bloquear la base de datos. Cuando usas mysqldump, especialmente sin opciones como --single-transaction, puede haber bloqueos en las tablas, lo que puede afectar el rendimiento, especialmente en bases de datos grandes. Se puede hacer mediante el siguiente código:

```
mysqldump -u root -p --single-transaction autosdeportivosdb > backup_caliente.sql
```

El problema será el hecho del rendimiento para la propia base de datos lo cual no pasará en Percona XtraBackup aquí podemos generarlo de la siguiente forma:

```
xtrabackup --backup --target-dir=/path/to/backup --user=root --password=your_password
```

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

- Los respaldos en caliente son posibles debido a las operaciones transaccionales y el manejo de archivos binarios en bases de datos.
- Se recomiendan para bases de datos críticas, como en comercio electrónico o aplicaciones empresariales.
- Percona XtraBackup mantiene la consistencia de los datos mediante el registro de transacciones y no requiere tiempos de inactividad.

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

- Esto resulta especialmente relevante para plataformas en línea, servicios financieros, aplicaciones empresariales y cualquier sistema que funcione en tiempo real, donde una interrupción podría generar pérdidas económicas, malestar en los clientes o incluso riesgos para la seguridad de la información. Implementar respaldos en caliente garantiza la continuidad del negocio, facilita la recuperación rápida ante posibles fallos y contribuye a mantener la integridad de los datos, todo mientras el sistema sigue operando de manera normal.

#### 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

```
98 -- ////////////////////////////////// PARTE 4 //////////////////////////////////
99 • CREATE INDEX idx_marca ON AutosDeportivos(id_marca);
100
101 • CREATE INDEX idx_pais ON DistribucionPais(id_pais);
102
103
```

Output

| #   | Time     | Action  | Message  |
|-----|----------|---|--|
| ✓ 2 | 01:02:44 | USE AutosDeportivosDB                               | 0 row(s) affected                                      |
| ✓ 3 | 01:02:49 | CREATE INDEX idx_marca ON AutosDeportivos(id_marca) | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 |
| ✓ 4 | 01:02:49 | CREATE INDEX idx_pais ON DistribucionPais(id_pais)  | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 |

```
CREATE TABLE `autosdeportivos` (
  `id_auto` int NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL,
  `modelo` varchar(100) NOT NULL,
  `costo` decimal(10,2) NOT NULL,
  `id_marca` int DEFAULT NULL,
  PRIMARY KEY (`id_auto`),
  KEY `idx_marca` (`id_marca`),
  CONSTRAINT `autosdeportivos_ibfk_1` FOREIGN KEY (`id_marca`) REFERENCES `marcas` (`id_marca`) ON DELETE SET NULL ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci

1 CREATE TABLE `distribucionpais` (
2   `id_distribucion` int NOT NULL AUTO_INCREMENT,
3   `id_auto` int DEFAULT NULL,
4   `id_pais` int DEFAULT NULL,
5   `cantidad` int NOT NULL,
6   PRIMARY KEY (`id_distribucion`),
7   KEY `id_auto` (`id_auto`),
8   KEY `idx_pais` (`id_pais`),
9   CONSTRAINT `distribucionpais_ibfk_1` FOREIGN KEY (`id_auto`) REFERENCES `autosdeportivos` (`id_auto`) ON DELETE CASCADE ON UPDATE CASCADE,
10  CONSTRAINT `distribucionpais_ibfk_2` FOREIGN KEY (`id_pais`) REFERENCES `paises` (`id_pais`) ON DELETE CASCADE ON UPDATE CASCADE
11 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

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

- Los índices más comunes en bases de datos transaccionales incluyen:

**Índice B-tree:** El más usado para búsquedas rápidas en bases de datos. Ideal para columnas con un rango de valores.

**Índice Hash:** Usado para búsquedas exactas, pero no soporta rangos.

**Índice Full-Text:** Utilizado para buscar dentro de cadenas de texto grandes

**Importancia del Conocimiento:** Los índices son cruciales para acelerar las consultas y mejorar el rendimiento general de la base de datos. Mejoran significativamente la velocidad de las consultas, especialmente en bases de datos con grandes volúmenes de datos. Si no se gestionan correctamente, sin embargo, pueden afectar el rendimiento de las inserciones y actualizaciones.

2. Optimizar consultas SQL.

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

```
EXPLAIN SELECT AutosDeportivos.nombre, Países.nombre_pais
FROM AutosDeportivos
JOIN DistribucionPais ON AutosDeportivos.id_auto = DistribucionPais.id_auto
JOIN Países ON DistribucionPais.id_pais = Países.id_pais
WHERE DistribucionPais.cantidad > 100;
```

|   | id | select_type | table            | partitions          | type   | possible_keys    | key                 | key_len             | ref  | rows | filtered | Extra               |
|---|----|-------------|------------------|---------------------|--------|------------------|---------------------|---------------------|--|------|----------|---------------------|
| ▶ | 1  | SIMPLE      | DistribucionPais | <small>NULL</small> | ALL    | id_auto,idx_pais | <small>NULL</small> | <small>NULL</small> | <small>NULL</small>                        | 3    | 33.33    | Using where         |
|   | 1  | SIMPLE      | AutosDeportivos  | <small>NULL</small> | eq_ref | PRIMARY          | PRIMARY             | 4                   | autosdeportivosdb.DistribucionPais.id_auto | 1    | 100.00   | <small>NULL</small> |
|   | 1  | SIMPLE      | Países           | <small>NULL</small> | eq_ref | PRIMARY          | PRIMARY             | 4                   | autosdeportivosdb.DistribucionPais.id_pais | 1    | 100.00   | <small>NULL</small> |

El código realiza una consulta que selecciona el nombre del automóvil (AutosDeportivos.nombre) y el nombre del país (Países.nombre\_pais) de las tablas AutosDeportivos y Países, respectivamente. Utiliza dos uniones (JOIN): la primera une la tabla AutosDeportivos con DistribucionPais mediante la columna id\_auto, y la segunda une DistribucionPais con Países usando la columna id\_pais. La cláusula WHERE filtra los registros para que solo se muestren aquellos donde la cantidad de autos distribuidos en un país sea mayor a 100. El uso de EXPLAIN permite analizar cómo se ejecutará esta consulta y sus optimizaciones.

Practica: Aplicación de 3 join

136 •  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150

```
SELECT
    AutosDeportivos.nombre,
    Marcas.nombre_marca,
    Países.nombre_pais
FROM
    AutosDeportivos
JOIN
    Marcas ON AutosDeportivos.id_marca = Marcas.id_marca
JOIN
    DistribucionPais ON AutosDeportivos.id_auto = DistribucionPais.id_auto
JOIN
    Países ON DistribucionPais.id_pais = Países.id_pais
WHERE
    DistribucionPais.cantidad > 50;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

|   | nombre              | nombre_marca | nombre_pais    |
|---|---------------------|--------------|----------------|
| ▶ | Ferrari 488         | Ferrari      | Estados Unidos |
|   | Lamborghini Huracan | Lamborghini  | Alemania       |

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



JOIN los diferentes tipos que tenemos:

**INNER JOIN:** Se utiliza cuando solo se necesitan los registros coincidentes.

**LEFT JOIN:** Para obtener todos los registros de la tabla izquierda y los coincidentes de la derecha.

**Subconsultas:** Son útiles para evitar múltiples consultas, pero deben usarse con precaución debido a su posible impacto en el rendimiento.

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

- Las consultas optimizadas garantizan que el sistema sea rápido y eficiente. Especialmente en bases de datos grandes o con alta carga, la optimización de las consultas puede reducir significativamente el tiempo de respuesta.

### 3. Utilizar particionamiento de tablas.

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

```
-- 3. Particionamiento de tablas
CREATE TABLE Reservas (
    id_reserva INT AUTO_INCREMENT,
    id_cliente INT,
    id_auto INT,
    fecha_reserva DATE,
    PRIMARY KEY (id_reserva, fecha_reserva) -- Incluimos fecha_reserva en la clave primaria
)
PARTITION BY RANGE (YEAR(fecha_reserva)) (
    PARTITION p0 VALUES LESS THAN (2022),
    PARTITION p1 VALUES LESS THAN (2023),
    PARTITION p2 VALUES LESS THAN (2024)
);
```

#### DDL for autosdeportivosdb.reservas

```
1 CREATE TABLE `reservas` (
2     `id_reserva` int NOT NULL AUTO_INCREMENT,
3     `id_cliente` int DEFAULT NULL,
4     `id_auto` int DEFAULT NULL,
5     `fecha_reserva` date NOT NULL,
6     PRIMARY KEY (`id_reserva`,`fecha_reserva`)
7 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
8 /*!50100 PARTITION BY RANGE (year(`fecha_reserva`))
9 (PARTITION p0 VALUES LESS THAN (2022) ENGINE = InnoDB,
10 PARTITION p1 VALUES LESS THAN (2023) ENGINE = InnoDB,
11 PARTITION p2 VALUES LESS THAN (2024) ENGINE = InnoDB) */
```

Este código crea una tabla llamada Reservas con cuatro columnas: id\_reserva, id\_cliente, id\_auto y fecha\_reserva. La tabla se configura para tener particionamiento por rango en la columna fecha\_reserva, dividiendo los registros en particiones según el año de la fecha de reserva.

La partición por rango (RANGE) se realiza en función del año de la columna fecha\_reserva utilizando la función YEAR(fecha\_reserva). El particionamiento se realiza en tres particiones:

Partición p0: Incluye registros con una fecha de reserva hasta el año 2021 (excluyendo el 2022).

Partición p1: Incluye registros con una fecha de reserva hasta el año 2022 (excluyendo el 2023).

Partición p2: Incluye registros con una fecha de reserva hasta el año 2023 (excluyendo el 2024)

**Investigación:** Investigar sobre los beneficios del particionamiento y cómo implementarlo en sistemas de bases de datos grandes.

- El particionamiento mejora la escalabilidad y el rendimiento al permitir que las consultas solo trabajen con las particiones relevantes, reduciendo el número de filas que necesitan ser procesadas. También mejora el rendimiento de las operaciones de mantenimiento, como la eliminación de datos antiguos.

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

- El particionamiento es una técnica poderosa para gestionar grandes volúmenes de datos, ya que ayuda a mejorar la velocidad de consulta y la gestión de datos históricos, lo que es esencial para bases de datos de gran escala o bases de datos con datos que crecen rápidamente.

## 5. Procedimientos Almacenados, Vistas y Triggers, Funciones (prácticas de cada uno)

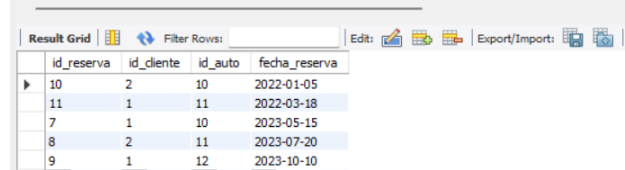
**Objetivo:** Mejorar la eficiencia y automatizar tareas mediante el uso de procedimientos almacenados, vistas y triggers.

**Actividades:**

### 1. Crear procedimientos almacenados.

**Práctica:** Crear un procedimiento para calcular el precio total de una reserva, aplicando descuentos y cargos adicionales, aplicar 2 ejercicios y explicar comprensión al 100%.

```
178
179 -- Insertar datos en la tabla Reservas
180 • insert into Reservas (id_cliente, id_auto, fecha_reserva)
181 values
182 (1, 10, '2023-05-15'),
183 (2, 11, '2023-07-20'),
184 (1, 12, '2023-10-10'),
185 (2, 10, '2022-01-05'),
186 (1, 11, '2022-03-18');
187
188 -- EJERCICIO 1/2
```



| id_reserva | id_cliente | id_auto | fecha_reserva |
|------------|------------|---------|---------------|
| 10         | 2          | 10      | 2022-01-05    |
| 11         | 1          | 11      | 2022-03-18    |
| 7          | 1          | 10      | 2023-05-15    |
| 8          | 2          | 11      | 2023-07-20    |
| 9          | 1          | 12      | 2023-10-10    |

-- EJERCICIO 1/2

Delimiter //

-- Procedimiento para calcular el precio total de la reserva

create procedure CalcularPrecioReserva(in id\_reserva int)

BEGIN

declare precio\_base decimal(10,2);

declare descuento decimal(10,2);

declare cargo\_adicional decimal(10,2);

declare precio\_final decimal(10,2);

-- Obtener el precio base del auto

select costo

into precio\_base

from AutosDeportivos

join Reservas on AutosDeportivos.id\_auto = Reservas.id\_auto

where Reservas.id\_reserva = id\_reserva

limit 1; -- Esto sera para que solo muestre una fila en la consulta

-- Aplicar descuento dependiendo del año del modelo del auto

if exists(

select 1

from AutosDeportivos

where id\_auto = (select id\_auto from Reservas where id\_reserva = id\_reserva limit 1)

and modelo = '2021'

) then

set descuento = precio\_base \* 0.10;

else

set descuento = 0;

end if;

-- Aplicar cargo adicional dependiendo del país

```
if exists (
    select 1
    from DistribucionPais
    where id_auto = (select id_auto from Reservas where id_reserva = id_reserva limit 1)
    and id_pais = 1
) then
    set cargo_adicional = precio_base * 0.05;
else
    set cargo_adicional = 0;
end if;

-- Calcular el precio final aplicando descuento y cargo adicional
set precio_final = precio_base - descuento + cargo_adicional;

-- Mostrar el precio final
select
    precio_base as 'Precio Base',
    descuento as 'Descuento',
    cargo_adicional as 'Cargo Adicional',
    precio_final as 'Precio Final';
end //

Delimiter ;
```

```

187
188 -- EJERCICIO 1/2
189 Delimiter //
190
191 • -- Procedimiento para calcular el precio total de la reserva
192 create procedure CalcularPrecioReserva(in id_reserva int)
193 BEGIN
194     declare precio_base decimal(10,2);
195     declare descuento decimal(10,2);
196     declare cargo_adicional decimal(10,2);
197     declare precio_final decimal(10,2);
198
199     -- Obtener el precio base del auto
200     select costo
201     into precio_base
202     from AutosDeportivos
203     join Reservas on AutosDeportivos.id_auto = Reservas.id_auto
204     where Reservas.id_reserva = id_reserva
205     limit 1; -- Esto sera para que solo muestre una fila en la consulta
206
207     -- Aplicar descuento dependiendo del año del modelo del auto
208     if exists(
209         select 1
210
211         -- Aplicar descuento dependiendo del año del modelo del auto
212         if exists(
213             select 1
214             from AutosDeportivos
215             where id_auto = (select id_auto from Reservas where id_reserva = id_reserva limit 1)
216             and modelo = '2021'
217         ) then
218             set descuento = precio_base * 0.10;
219         else
220             set descuento = 0;
221         end if;
222
223     -- Aplicar cargo adicional dependiendo del país
224     if exists (
225         select 1
226         from DistribucionPais
227         where id_auto = (select id_auto from Reservas where id_reserva = id_reserva limit 1)
228         and id_pais = 1
229     ) then
230         set cargo_adicional = precio_base * 0.05;
231     else

```

```

220 if exists (
221     select 1
222     from DistribucionPais
223     where id_auto = (select id_auto from Reservas where id_reserva = id_reserva limit 1)
224     and id_pais = 1
225 ) then
226     set cargo_adicional = precio_base * 0.05;
227 else
228     set cargo_adicional = 0;
229 end if;
230
231 -- Calcular el precio final aplicando descuento y cargo adicional
232 set precio_final = precio_base - descuento + cargo_adicional;
233
234 -- Mostrar el precio final
235 select
236     precio_base as 'Precio Base',
237     descuento as 'Descuento',
238     cargo_adicional as 'Cargo Adicional',
239     precio_final as 'Precio Final';
240 end //
241
242 Delimiter ;

```

```

243
244 • -- Calcular el precio de una reserva mediante el id_reserva
245 select*from Reservas;
246 • call CalcularPrecioReserva(10);
247
248 -- EJERCICIO 2/2

```

| Result Grid | Filter Rows: | Export:         | Wrap Cell Content: |
|-------------|--------------|-----------------|--------------------|
|             |              |                 |                    |
| Precio Base | Descuento    | Cargo Adicional | Precio Final       |
| 280000.00   | 28000.00     | 14000.00        | 266000.00          |

Result 46 x

```

247
248 -- EJERCICIO 2/2
249 -- Sera el mismo ejercicio de arriba ya que solo tenemos costos en la tabla de autos deportivos
250 • select*from Reservas;
251 • call CalcularPrecioReserva(11);
252
253
254

```

| Result Grid | Filter Rows: | Export:         | Wrap Cell Content: |
|-------------|--------------|-----------------|--------------------|
|             |              |                 |                    |
| Precio Base | Descuento    | Cargo Adicional | Precio Final       |
| 300000.00   | 30000.00     | 15000.00        | 285000.00          |

Result 55 x

**Investigación:** Explorar cómo los procedimientos almacenados pueden mejorar la reutilización de código y la eficiencia.

#### **Reutilización de Código:**

- 1.- Encapsula la lógica de negocio, evitando la repetición de código.
- 2.- Cambios en la lógica se realizan en un solo lugar, facilitando mantenimiento.

#### **Eficiencia:**

1. El servidor de base de datos optimiza el plan de ejecución, mejorando la velocidad.
2. Realiza operaciones complejas solo una vez en la base de datos.

#### **Seguridad:**

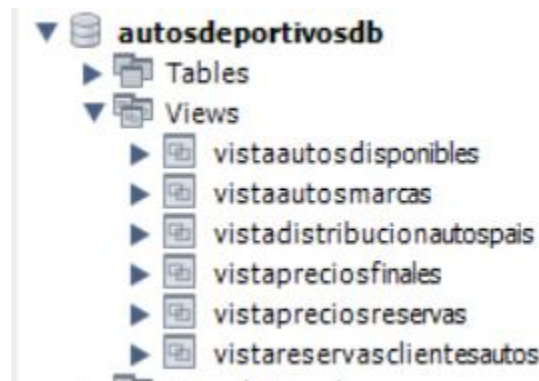
1. Restringe el acceso directo a datos, ejecutando solo procedimientos definidos.
2. Reduce el riesgo de errores al centralizar la lógica en procedimientos.

En pocas, los procedimientos almacenados optimizan la reutilización de código, mejoran el rendimiento, aumentan la seguridad y facilitan el mantenimiento y la escalabilidad. Son ideales para sistemas que requieren eficiencia y gestión centralizada de la lógica de negocio.

**Importancia del Conocimiento:** Los procedimientos almacenados centralizan la lógica y pueden mejorar el rendimiento al ejecutarse directamente en el servidor.

2. **Crear vistas para simplificar consultas complejas.**

**Práctica:** Crear vistas que presenten información de varias tablas de manera unificada (por ejemplo, una vista que combine datos de Vuelos, Clientes y Reservas).



-- Vista de Autos deportivos y marcas

create view VistaAutosMarcas as

select

AutosDeportivos.id\_auto,

AutosDeportivos.nombre as 'Nombre del Auto',

AutosDeportivos.modelo,

```

    AutosDeportivos.costo as 'Precio del Auto',
    Marcas.nombre_marca as 'Marca',
    Duenos.nombre_dueño as 'Dueño de la Marca'
from
    AutosDeportivos
join
    Marcas on AutosDeportivos.id_marca = Marcas.id_marca
join
    Duenos on Marcas.id_dueno = Duenos.id_dueno;

-- Vista de distribucion de autos por pais
create view VistaDistribucionAutosPais as
select
    AutosDeportivos.nombre as 'Nombre del Auto',
    Paises.nombre_pais as 'País',
    DistribucionPais.cantidad as 'Cantidad Distribuida'
from
    DistribucionPais
join
    AutosDeportivos on DistribucionPais.id_auto = AutosDeportivos.id_auto
join
    Paises on DistribucionPais.id_pais = Paises.id_pais;

-- Vistas de informacion de clientes y autos
create view VistaReservasClientesAutos as
select
    Reservas.id_reserva,
    Reservas.fecha_reserva,
    Usuarios.nombre as 'Nombre del Cliente',
    Usuarios.email as 'Correo del Cliente',
    AutosDeportivos.nombre as 'Auto Reservado',

```



```

    AutosDeportivos.modelo as 'Modelo del Auto'
from
    Reservas
join
    Usuarios on Reservas.id_cliente = Usuarios.id_usuario
join
    AutosDeportivos on Reservas.id_auto = AutosDeportivos.id_auto;

-- Vistas de precios de reservas
create view VistaPreciosReservas as
select
    Reservas.id_reserva,
    AutosDeportivos.nombre as 'Auto Reservado',
    AutosDeportivos.modelo as 'Modelo del Auto',
    Reservas.fecha_reserva,
    (AutosDeportivos.costo -
        (case
            when AutosDeportivos.modelo = '2021' then AutosDeportivos.costo * 0.10
            else 0
        end) +
        (case
            when DistribucionPais.id_pais = 1 then AutosDeportivos.costo * 0.05
            else 0
        end)) as 'Precio Final'
from
    Reservas
join
    AutosDeportivos on Reservas.id_auto = AutosDeportivos.id_auto
join
    DistribucionPais on AutosDeportivos.id_auto = DistribucionPais.id_auto;

```

```

-- Autos disponibles para reserva
create view VistaAutosDisponibles as
select
    AutosDeportivos.id_auto,
    AutosDeportivos.nombre as 'Nombre del Auto',
    AutosDeportivos.modelo,
    AutosDeportivos.costos as 'Precio',
    Paises.nombre_pais as 'País'
from
    AutosDeportivos
join
    DistribucionPais on AutosDeportivos.id_auto = DistribucionPais.id_auto
join
    Paises on DistribucionPais.id_pais = Paises.id_pais
where
    DistribucionPais.cantidad > 0;

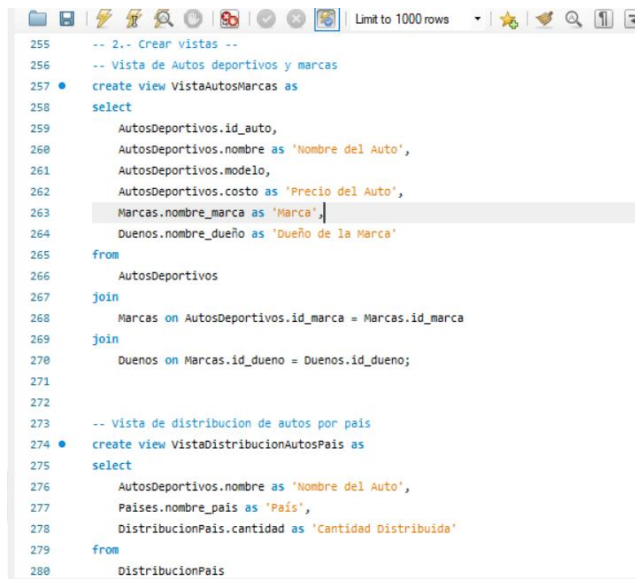
-- Vista de Autos y Precios Finales con Descuento y Cargo Adicional
create view VistaPreciosFinales as
select
    AutosDeportivos.nombre as 'Nombre del Auto',
    AutosDeportivos.modelo,
    AutosDeportivos.costos as 'Precio Base',
    (case
        when AutosDeportivos.modelo = '2021' then AutosDeportivos.costos * 0.10
        else 0
    end) as 'Descuento',
    (case
        when DistribucionPais.id_pais = 1 then AutosDeportivos.costos * 0.05
        else 0
    end) as 'Cargo Adicional',
    (AutosDeportivos.costos -

```

```

(case
    when AutosDeportivos.modelo = '2021' then AutosDeportivos.costo * 0.10
    else 0
end) +
(case
    when DistribucionPais.id_pais = 1 then AutosDeportivos.costo * 0.05
    else 0
end)) as 'Precio Final'
from
    AutosDeportivos
join
    DistribucionPais on AutosDeportivos.id_auto = DistribucionPais.id_auto;

```



```

255 -- 2.- Crear Vistas --
256 -- Vista de Autos deportivos y marcas
257 • create view VistaAutosMarcas as
258 select
259     AutosDeportivos.id_auto,
260     AutosDeportivos.nombre as 'Nombre del Auto',
261     AutosDeportivos.modelo,
262     AutosDeportivos.costo as 'Precio del Auto',
263     Marcas.nombre_marca as 'Marca',
264     Duenos.nombre_dueño as 'Dueño de la Marca'
265 from
266     AutosDeportivos
267 join
268     Marcas on AutosDeportivos.id_marca = Marcas.id_marca
269 join
270     Duenos on Marcas.id_dueno = Duenos.id_dueno;
271
272
273 -- vista de distribucion de autos por pais
274 • create view VistaDistribucionAutosPais as
275 select
276     AutosDeportivos.nombre as 'Nombre del Auto',
277     Países.nombre_pais as 'País',
278     DistribucionPais.cantidad as 'Cantidad Distribuida'
279 from
280     DistribucionPais

```

```

273 -- Vista de distribucion de autos por pais
274 • create view VistaDistribucionAutosPais as
275 select
276     AutosDeportivos.nombre as 'Nombre del Auto',
277     Países.nombre_pais as 'País',
278     DistribucionPais.cantidad as 'Cantidad Distribuida'
279 from
280     DistribucionPais
281 join
282     AutosDeportivos on DistribucionPais.id_auto = AutosDeportivos.id_auto
283 join
284     Países on DistribucionPais.id_pais = Países.id_pais;
285
286 -- Vistas de informacion de clientes y autos
287 • create view VistaReservasClientesAutos as
288 select
289     Reservas.id_reserva,
290     Reservas.fecha_reserva,
291     Usuarios.nombre as 'Nombre del Cliente',
292     Usuarios.email as 'Correo del Cliente',
293     AutosDeportivos.nombre as 'Auto Reservado',
294     AutosDeportivos.modelo as 'Modelo del Auto'
295 from
296     Reservas
297 join
298     Usuarios on Reservas.id_cliente = Usuarios.id_usuario

```

```

303 • create view VistaPreciosReservas as
304 select
305     Reservas.id_reserva,
306     AutosDeportivos.nombre as 'Auto Reservado',
307     AutosDeportivos.modelo as 'Modelo del Auto',
308     Reservas.fecha_reserva,
309     (AutosDeportivos.costo -
310         (case
311             when AutosDeportivos.modelo = '2021' then AutosDeportivos.costo * 0.10
312             else 0
313         end) +
314         (case
315             when DistribucionPais.id_pais = 1 then AutosDeportivos.costo * 0.05
316             else 0
317         end)) as 'Precio Final'
318 from
319     Reservas
320 join
321     AutosDeportivos on Reservas.id_auto = AutosDeportivos.id_auto
322 join
323     DistribucionPais on AutosDeportivos.id_auto = DistribucionPais.id_auto;
324
325 -- Autos disponibles para reserva
326 • create view VistaAutosDisponibles as
327 select
328     AutosDeportivos.id_auto,

```

```

339 where
340     DistribucionPais.cantidad > 0;
341
342 -- Vista de Autos y Precios Finales con Descuento y Cargo Adicional
343 • create view VistaPreciosFinales as
344 select
345     AutosDeportivos.nombre as 'Nombre del Auto',
346     AutosDeportivos.modelo,
347     AutosDeportivos.costo as 'Precio Base',
348     (case
349         when AutosDeportivos.modelo = '2021' then AutosDeportivos.costo * 0.10
350         else 0
351     end) as 'Descuento',
352     (case
353         when DistribucionPais.id_pais = 1 then AutosDeportivos.costo * 0.05
354         else 0
355     end) as 'Cargo Adicional',
356     (AutosDeportivos.costo -
357         (case
358             when AutosDeportivos.modelo = '2021' then AutosDeportivos.costo * 0.10
359             else 0
360         end) +
361         (case
362             when DistribucionPais.id_pais = 1 then AutosDeportivos.costo * 0.05
363             else 0
364         end)) as 'Precio Final'

```

```

351     end) as 'Descuento',
352     (case
353     when DistribucionPais.id_pais = 1 then AutosDeportivos.costo * 0.05
354     else 0
355     end) as 'Cargo Adicional',
356     (AutosDeportivos.costo -
357     (case
358     when AutosDeportivos.modelo = '2021' then AutosDeportivos.costo * 0.10
359     else 0
360     end) +
361     (case
362     when DistribucionPais.id_pais = 1 then AutosDeportivos.costo * 0.05
363     else 0
364     end)) as 'Precio Final'
365 from
366     AutosDeportivos
367 join
368     DistribucionPais on AutosDeportivos.id_auto = DistribucionPais.id_auto;
369
370 -- Ver las vistas creadas
371 • select*from VistaAutosMarcas;
372 • select*from VistaDistribucionAutosPais;
373 • select*from VistaReservasClientesAutos;
374 • select*from VistaPreciosReservas;
375 • select*from VistaAutosDisponibles;
376 • select*from VistaPreciosFinales;

```

**Investigación:** Investigar las ventajas de usar vistas en lugar de consultas complejas repetitivas.

### Ventajas de Usar Vistas en SQL

- Evita repetir consultas complejas.
- Mejora la eficiencia al almacenar resultados preprocesados.
- Restringe acceso a ciertos datos sin modificar permisos de tablas.
- Facilita la actualización de consultas sin afectar código existente.
- Simplifica la lectura y escritura de consultas SQL.

**Importancia del Conocimiento:** Las vistas ayudan a simplificar el acceso a datos complejos y pueden mejorar la seguridad al limitar el acceso directo a las tablas.

### 3. Implementar triggers para auditoría y control de cambios.

**Práctica:** Crear triggers que registren cambios en las tablas de Reservas y Pagos cada vez que un registro se actualiza o elimina., 2 ejercicios conocimiento al 100%

-- 3.- Crear Triggers y control de cambios

-- En esta tabla se guardarán los cambios

create table HistorialCambios (

id\_cambio INT AUTO\_INCREMENT PRIMARY KEY,

tabla\_afectada VARCHAR(50),

tipo\_cambio ENUM('UPDATE', 'DELETE'),

id\_registro INT,

```
detalles TEXT,  
fecha_cambio TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```
-- trigger para registrar cambios en Reservas 1/2
```

```
DELIMITER //
```

```
CREATE TRIGGER before_update_reservas  
BEFORE UPDATE ON Reservas  
FOR EACH ROW  
BEGIN  
    INSERT INTO HistorialCambios (tabla_afectada, tipo_cambio, id_registro, detalles)  
    VALUES ('Reservas', 'UPDATE', OLD.id_reserva,  
        CONCAT('Reserva cambiada: ID ', OLD.id_reserva,  
            ', Cliente: ', OLD.id_cliente,  
            ', Auto: ', OLD.id_auto,  
            ', Fecha Anterior: ', OLD.fecha_reserva,  
            ' -> Nueva Fecha: ', NEW.fecha_reserva));  
END //
```

```
DELIMITER ;
```

```
-- trigger para registrar eliminaciones en autosdeportivos 2/2
```

```
DELIMITER //
```

```
CREATE TRIGGER before_delete_autosdeportivos  
BEFORE DELETE ON autosdeportivos  
FOR EACH ROW  
BEGIN  
    INSERT INTO HistorialCambios (tabla_afectada, tipo_cambio, id_registro, detalles)
```

```

VALUES ('autosdeportivos', 'DELETE', OLD.id_auto,
        CONCAT('Auto eliminado: ID ', OLD.id_auto,
               ', Marca: ', OLD.id_marca,
               ', Modelo: ', OLD.modelo));
END //

DELIMITER ;

-- actualizar una reserva

update Reservas set fecha_reserva = '2019-07-22' where id_reserva = 10; -- aqui cambiamos la fecha y/o la id_reserva
para modifica

-- eliminar un auto

delete from autosdeportivos where id_auto = 7; -- aqui cambiamos solo la id_auto para eliminar
select*from autosdeportivos;

-- ver los cambios

select*from HistorialCambios;

-- Ver solo los cambios relacionados con las eliminaciones en autosdeportivos

select*from HistorialCambios where tabla_afectada = 'autosdeportivos' and tipo_cambio = 'DELETE';

-- Ver solo los cambios relacionados con las actualizaciones en reservas

select*from HistorialCambios where tabla_afectada = 'Reservas' and tipo_cambio = 'UPDATE';

```

```

370
379 -- 3.- Crear Triggers y control de cambios
380
381 -- En esta tabla se guardaran los cambios
382 CREATE TABLE HistorialCambios (
383     id_cambio INT AUTO_INCREMENT PRIMARY KEY,
384     tabla_afectada VARCHAR(50),
385     tipo_cambio ENUM('UPDATE', 'DELETE'),
386     id_registro INT,
387     detalles TEXT,
388     fecha_cambio TIMESTAMP DEFAULT CURRENT_TIMESTAMP
389 );
390
391 -- trigger para registrar cambios en Reservas 1/2
392 DELIMITER //
393
394 CREATE TRIGGER before_update_reservas
395 BEFORE UPDATE ON Reservas
396 FOR EACH ROW
397 BEGIN
398     INSERT INTO HistorialCambios (tabla_afectada, tipo_cambio, id_registro, detalles)
399     VALUES ('Reservas', 'UPDATE', OLD.id_reserva,
400         CONCAT('Reserva cambiada: ID ', OLD.id_reserva,
401             ', Cliente: ', OLD.id_cliente,
402             ', Auto: ', OLD.id_auto,
403             ', Fecha: ', OLD.fecha_reserva));
404 END //
405
406 DELIMITER ;
407
408
409 -- trigger para registrar eliminaciones en autosdeportivos 2/2
410
411 DELIMITER //
412
413 CREATE TRIGGER before_delete_autosdeportivos
414 BEFORE DELETE ON autosdeportivos
415 FOR EACH ROW
416 BEGIN
417     INSERT INTO HistorialCambios (tabla_afectada, tipo_cambio, id_registro, detalles)
418     VALUES ('autosdeportivos', 'DELETE', OLD.id_auto,
419         CONCAT('Auto eliminado: ID ', OLD.id_auto,
420             ', Marca: ', OLD.id_marca,
421             ', Modelo: ', OLD.modelo));
422 END //
423
424 DELIMITER ;
425
426 -- actualizar una reserva
427 update Reservas set fecha_reserva = '2019-07-22' where id_reserva = 10; -- aqui cambiamos la fecha y/o la id_reserva para modifica
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```



```

432
433 -- ver loos cambios
434 • select*from HistorialCambios;
435
436 -- Ver solo los cambios relacionados con las eliminaciones en autosdeportivos

```

Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:

|   | id_cambio | tabla_afectada  | tipo_cambio | id_registro | detalles   | fecha_cambio        |
|---|-----------|-----------------|-------------|-------------|--|---------------------|
| ▶ | 1         | Reservas        | UPDATE      | 10          | Reserva cambiada: ID 10, Cliente: 2, Auto: 10, Fecha Anterior: 2022-01-01 -> Nueva Fecha: 2022-01-01 | 2025-01-31 13:52:30 |
|   | 2         | Reservas        | UPDATE      | 10          | Reserva cambiada: ID 10, Cliente: 2, Auto: 10, Fecha Anterior: 2022-01-01 -> Nueva Fecha: 2019-07-22 | 2025-01-31 13:53:02 |
|   | 3         | autosdeportivos | DELETE      | 7           | Auto eliminado: ID 7, Marca: 4, Modelo: 2021   | 2025-01-31 13:54:08 |
| ✱ | NULL      | NULL            | NULL        | NULL        | NULL   | NULL                |

HistorialCambios 73
Apply

**Investigación:** Investigar cómo utilizar triggers para mantener un historial de cambios en la base de datos.

Los triggers son procedimientos almacenados en bases de datos que se ejecutan automáticamente cuando ocurre un evento específico (como INSERT, UPDATE o DELETE) en una tabla. Para mantener un historial de cambios, se utilizan triggers de tipo BEFORE o AFTER en las tablas que se desean monitorear.

**Ventajas:**

- Mantener un registro de cambios para asegurar trazabilidad.
- Permite revisar el historial de las modificaciones hechas en la base de datos.

**Importancia del Conocimiento:** Los triggers permiten automatizar tareas como la auditoría y validación de datos.

**Acciones para realizar de forma automática, es decir si desea aplicar un cálculo de descuento y cambio del IVA que se debe hacer donde se pone esos valores y como se automatiza**

## 6. Monitoreo y Optimización de Recursos

**Objetivo:** Controlar el rendimiento de la base de datos, identificando y solucionando problemas de recursos.

### Actividades:

#### 1. Monitorear el rendimiento de consultas.

**Práctica:** Usar herramientas como SHOW PROCESSLIST para detectar consultas lentas y optimizarlas.

| Result Grid                |    |                 |                 |                   |         |      |                        |                  |
|----------------------------|----|-----------------|-----------------|-------------------|---------|------|------------------------|------------------|
| Filter Rows:               |    |                 |                 |                   |         |      |                        |                  |
| Export: Wrap Cell Content: |    |                 |                 |                   |         |      |                        |                  |
|                            | Id | User            | Host            | db                | Command | Time | State                  | Info             |
| ▶                          | 7  | event_scheduler | localhost       | NULL              | Daemon  | 1623 | Waiting on empty queue | NULL             |
|                            | 10 | root            | localhost:56847 | NULL              | Sleep   | 184  |                        | NULL             |
|                            | 11 | root            | localhost:56848 | autosdeportivosdb | Query   | 0    | init                   | SHOW PROCESSLIST |

- Este comando mostrará una lista de todas las consultas que se están ejecutando en ese momento. Hay que prestar atención a las consultas que están en estado "Sending data" o "Sorting result" por mucho tiempo, ya que pueden ser lentas.

### Análisis de las consultas

#### Consulta 1:

|   |   |                 |           |      |        |      |                        |      |
|---|---|-----------------|-----------|------|--------|------|------------------------|------|
| ▶ | 7 | event_scheduler | localhost | NULL | Daemon | 1623 | Waiting on empty queue | NULL |
|---|---|-----------------|-----------|------|--------|------|------------------------|------|

**Análisis:** Esta no es una consulta de usuario, sino el **event scheduler** de MySQL, que está esperando eventos programados para ejecutar y no es necesario optimizar esto, ya que es un proceso interno del servidor MySQL.

**Acción:** No se requiere ninguna acción.

#### Consulta 2:

|  |    |      |                 |      |       |     |  |      |
|--|----|------|-----------------|------|-------|-----|--|------|
|  | 10 | root | localhost:56847 | NULL | Sleep | 184 |  | NULL |
|--|----|------|-----------------|------|-------|-----|--|------|

**Análisis:** Esta es una conexión inactiva (Sleep). El usuario root está conectado, pero no está ejecutando ninguna consulta. Las conexiones en estado Sleep no consumen muchos recursos, pero si hay muchas conexiones inactivas, pueden agotar los recursos del servidor.

**Acción:** Si hay muchas conexiones inactivas, se considera reducir el tiempo de espera (wait\_timeout) en la configuración de MySQL para cerrar automáticamente las conexiones inactivas.

### Ejemplo de configuración en my.cnf o my.ini:

Se ingresan los siguientes comandos dentro de un CMD de Windows

```
Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.26100.3037]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Windows\System32>echo %PROGRAMDATA%\MySQL\MySQL Server 5.6\my.ini
C:\ProgramData\MySQL\MySQL Server 5.6\my.ini

C:\Windows\System32>[mysqld]wait_timeout = 300
```

[mysql]

wait\_timeout = 300

Cierra conexiones inactivas después de 300 segundos (5 minutos)

### Consulta 3:

|    |      |                 |                   |       |   |      |                  |
|----|------|-----------------|-------------------|-------|---|------|------------------|
| 11 | root | localhost:56848 | autosdeportivosdb | Query | 0 | init | SHOW PROCESSLIST |
|----|------|-----------------|-------------------|-------|---|------|------------------|

**Análisis:** Esta es la consulta que se está ejecutando actualmente (SHOW PROCESSLIST) y no es una consulta que necesite optimización, ya que es un comando administrativo para ver las consultas en ejecución.

**Acción:** No se requiere ninguna acción.

**Investigación:** Investigar las mejores prácticas para monitorear el rendimiento de las consultas en producción.

### Optimizar consultas con EXPLAIN:

Si se identifica una consulta lenta, se usa EXPLAIN para analizarla. Por ejemplo:

```
EXPLAIN SELECT * FROM AutosDeportivos WHERE id_marca = 1;
```

EXPLAIN mostrará cómo MySQL ejecuta la consulta, incluyendo los índices utilizados y el orden de las operaciones. Si no se están utilizando índices, se considera agregarlos. Esto mejorará el rendimiento de las consultas que filtran por id\_marca.

**Importancia del Conocimiento:** El monitoreo proactivo puede identificar cuellos de botella antes de que afecten el rendimiento del sistema.

## 2. Realizar pruebas de carga.

**Práctica:** Simular múltiples usuarios concurrentes usando herramientas como Apache JMeter para ver cómo responde la base de datos bajo alta carga.

**Investigación:** Investigar cómo realizar pruebas de estrés y carga en bases de datos de alto rendimiento.

**Importancia del Conocimiento:** Las pruebas de carga aseguran que el sistema sea capaz de manejar tráfico alto y crecimiento de datos.

## 3. Optimizar el uso de recursos y gestionar índices.

**Práctica:** Identificar índices no utilizados y eliminarlos para liberar recursos y mejorar la velocidad de las operaciones de escritura.

**Investigación:** Investigar cómo ajustar el número de índices según el tipo de consulta (lectura/escritura).

**Importancia del Conocimiento:** La optimización de los recursos asegura un uso eficiente del hardware y mejora la escalabilidad.

### 1. Identificar cuellos de botella:

- Si el tiempo de respuesta es alto o hay muchos errores, es posible que la base de datos esté bajo estrés.

### 2. Ajustar la configuración de MySQL:

- Si la base de datos no puede manejar la carga, se aconseja ajustar parámetros como max\_connections, innodb\_buffer\_pool\_size, o query\_cache\_size.

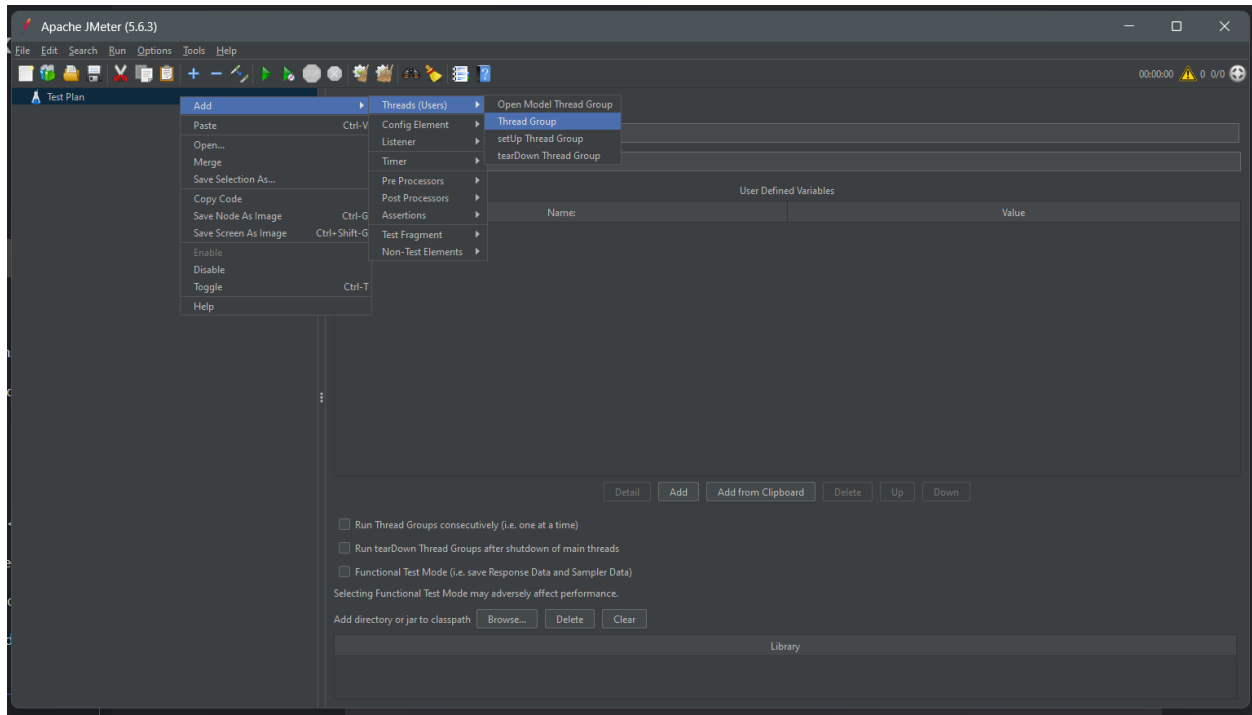
### 3. Repetir las pruebas:

- Después de realizar optimizaciones, se ejecuta nuevamente las pruebas de carga para verificar si el rendimiento ha mejorado.

## Pruebas

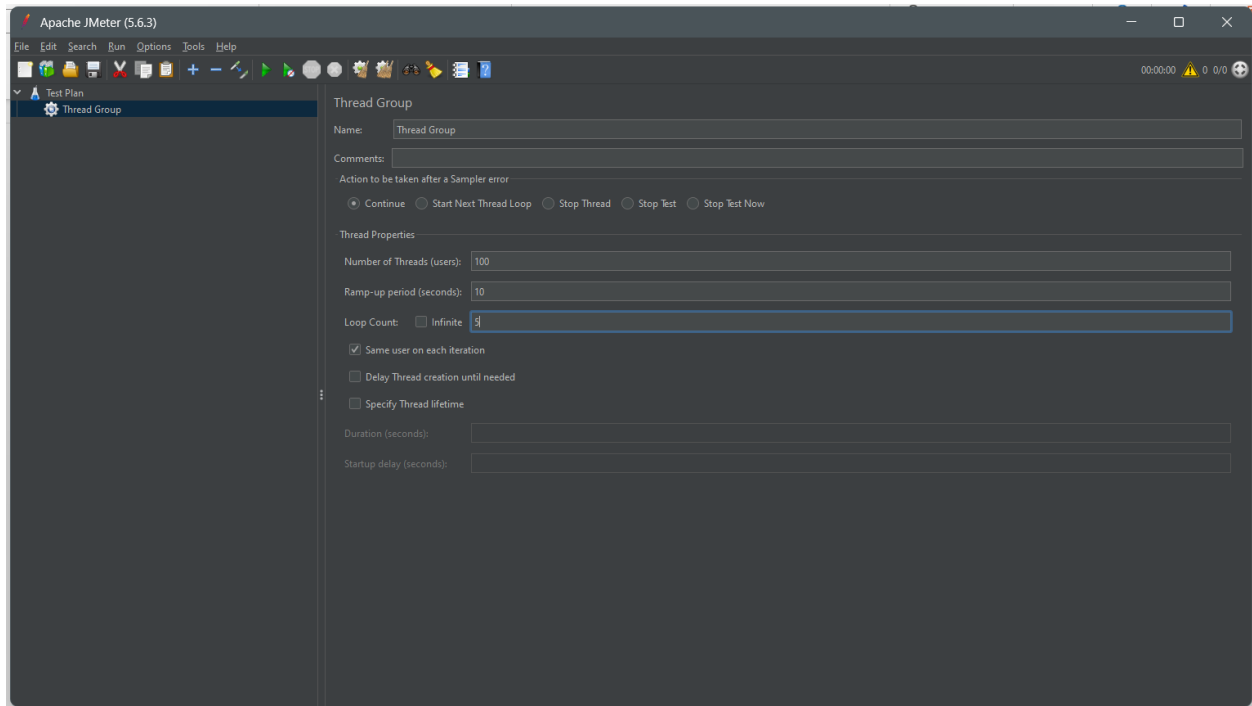
En la ventana principal de JMeter, en "Test Plan", se agrega un Threads (Users) > Thread Group.

- Un **Thread Group** representa un grupo de usuarios virtuales que ejecutarán las pruebas.



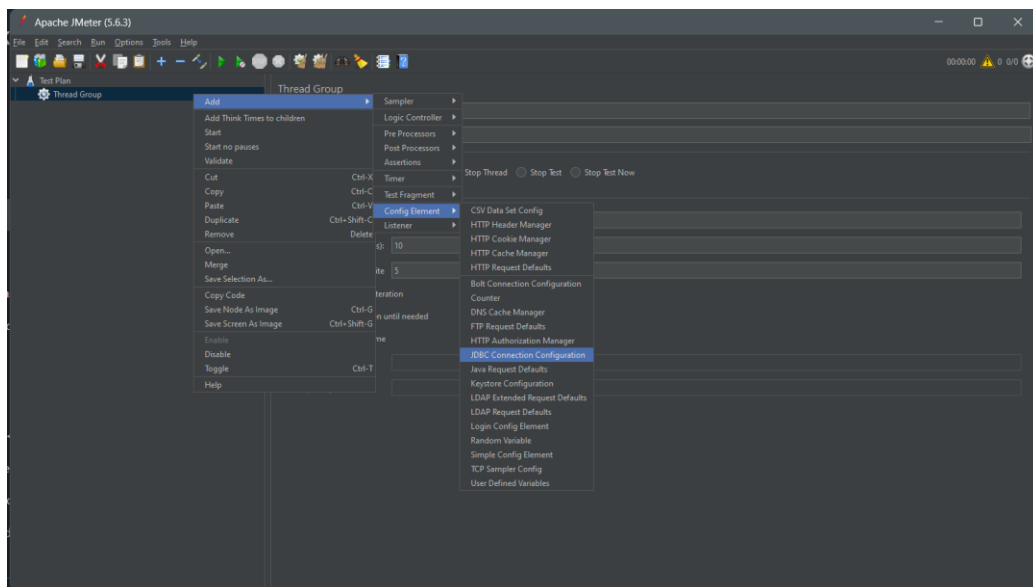
En el Thread Group, se configura los siguientes parámetros:

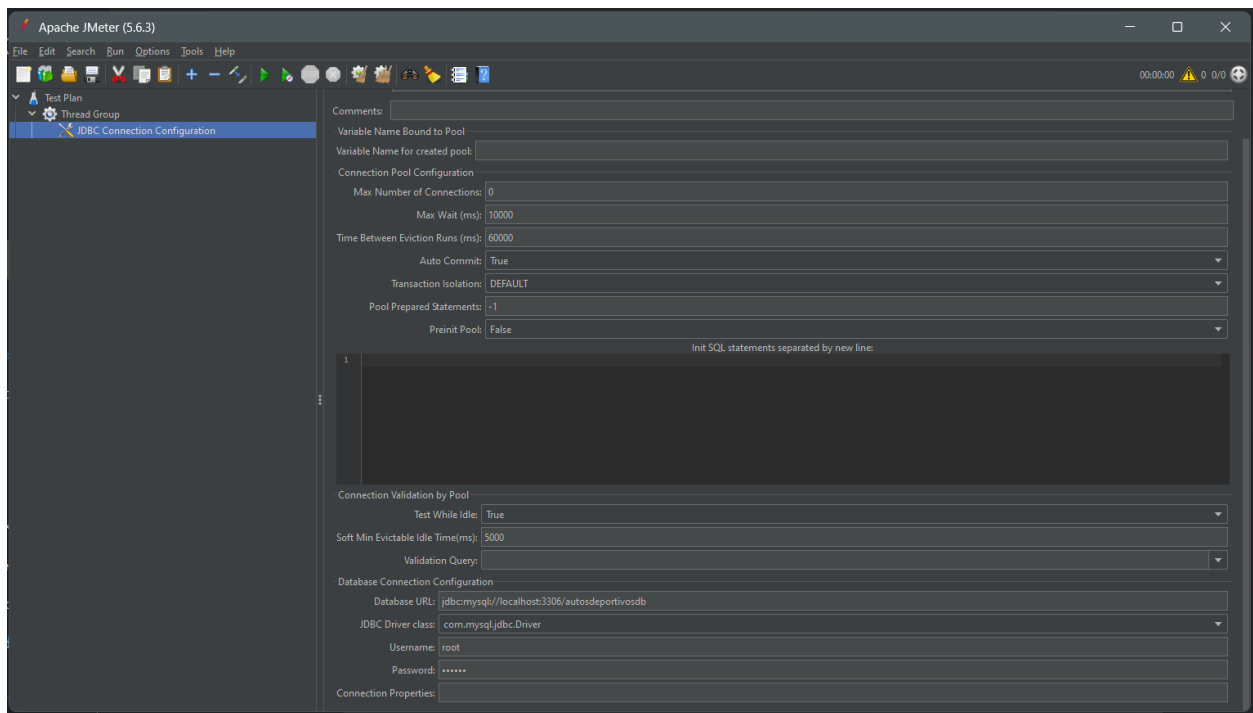
- **Number of Threads (users):** Número de usuarios concurrentes que se simularán (por ejemplo, 100).
- **Ramp-Up Period (in seconds):** Tiempo en segundos para que todos los usuarios se activen (por ejemplo, 10 segundos).
- **Loop Count:** Número de veces que cada usuario ejecutará la prueba (por ejemplo, 5).



Se agrega un **JDBC Connection Configuration**, y se configuran los siguientes parámetros:

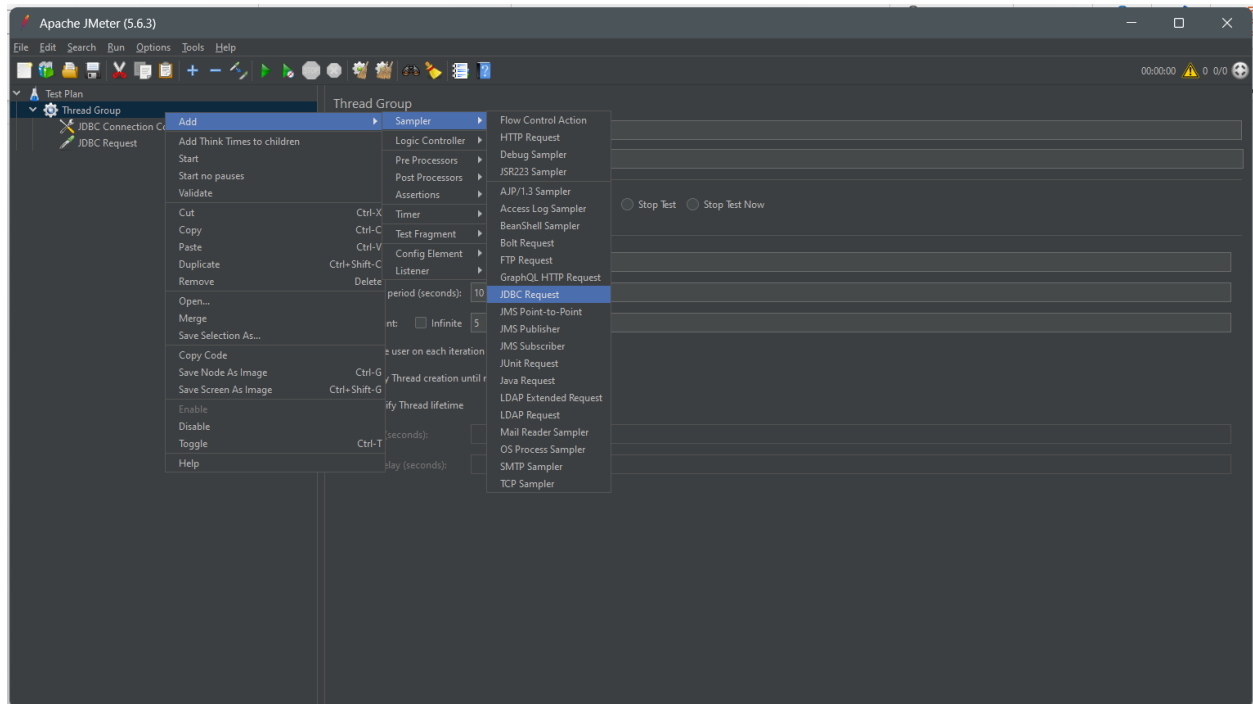
- **Variable Name:** Un nombre para la conexión (por ejemplo, mysql\_connection).
- **Database URL:** La URL de la base de datos MySQL (por ejemplo: jdbc:mysql://localhost:3306/autosdeportivosdb).
- **JDBC Driver Class:** Se selecciona com.mysql.jdbc.Driver.
- **Username:** El usuario de MySQL (por ejemplo, root).
- **Password:** La contraseña de MySQL.

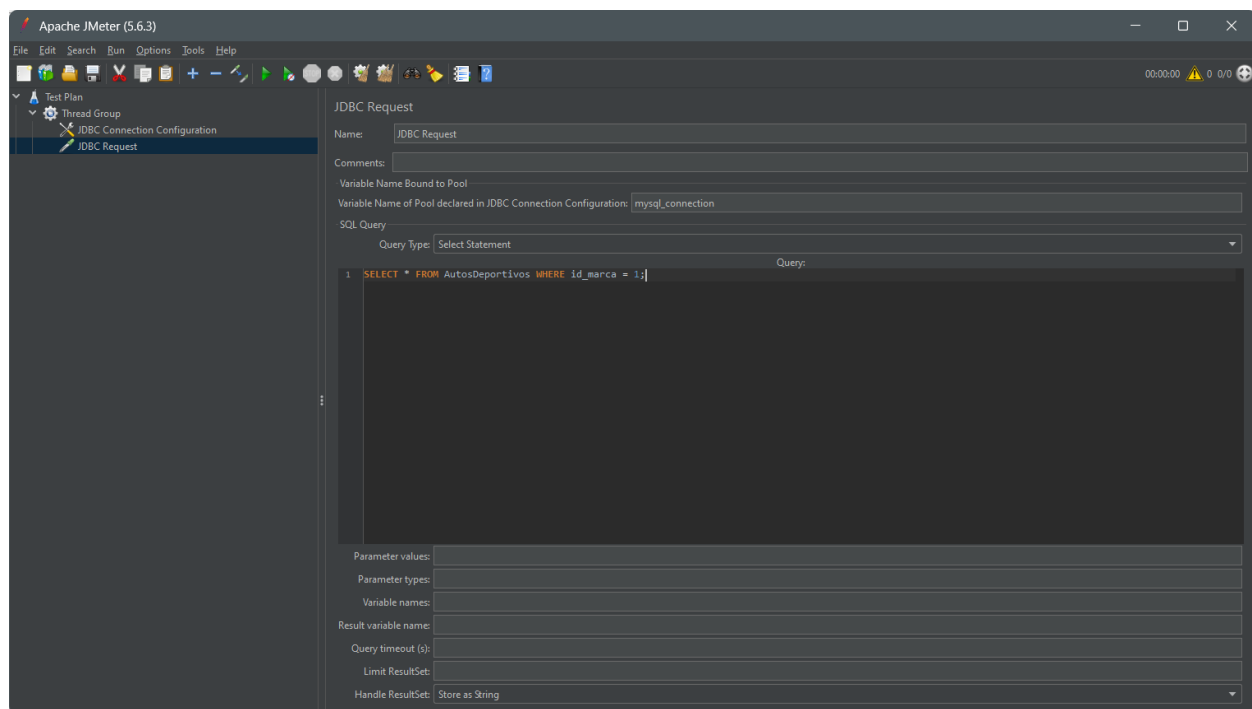




Se Agrega un JDBC Request y se configuran los siguientes parámetros:

- **Variable Name:** El mismo nombre que se usó en el JDBC Connection Configuration (por ejemplo, mysql\_connection).
- **SQL Query:** Se escribe la consulta SQL que se desea probar. Por ejemplo:  
SELECT \* FROM AutosDeportivos WHERE id\_marca = 1;





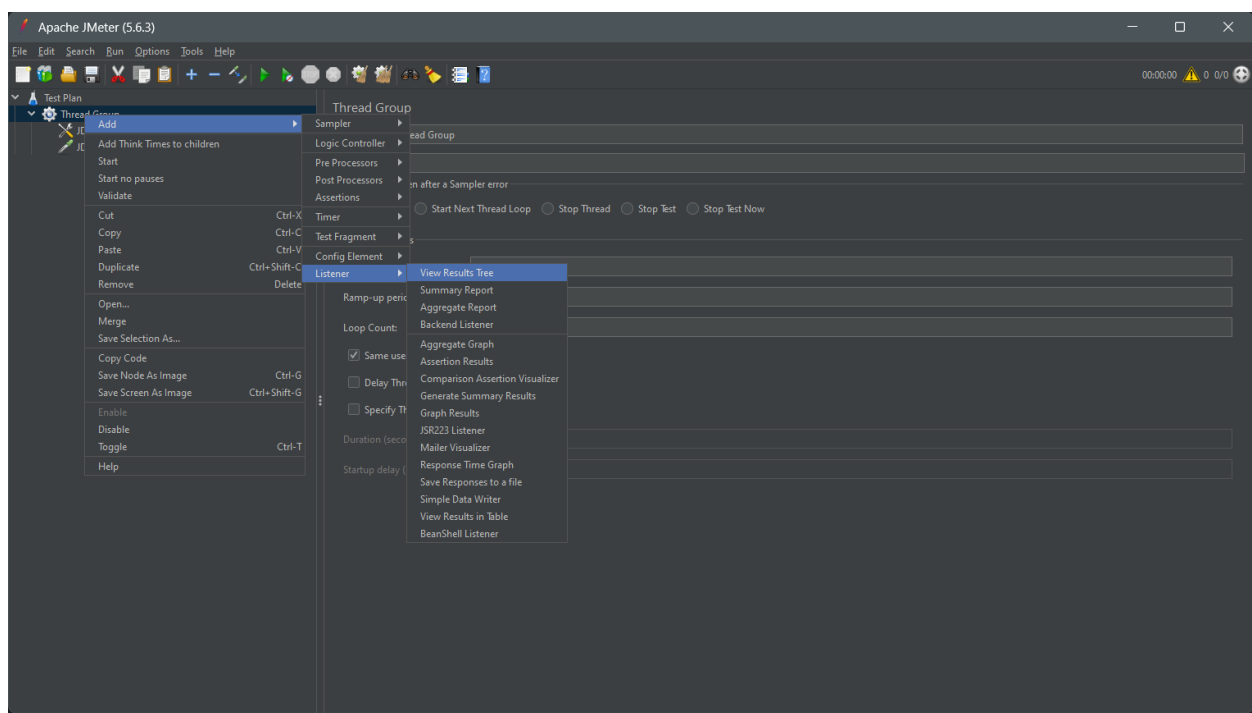
Ahora se agregan los Listeners para Ver los Resultados

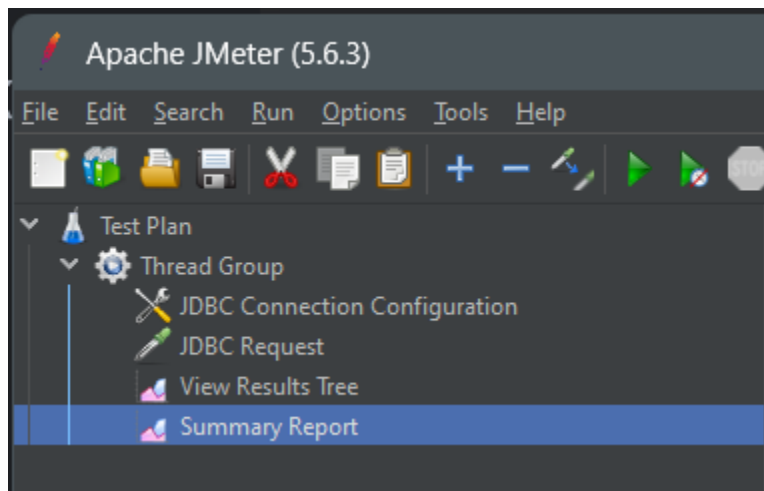
### 1. View Results Tree:

- Este listener permitirá ver los detalles de cada consulta ejecutada, incluyendo el tiempo de respuesta y si hubo errores.

### 2. Summary Report:

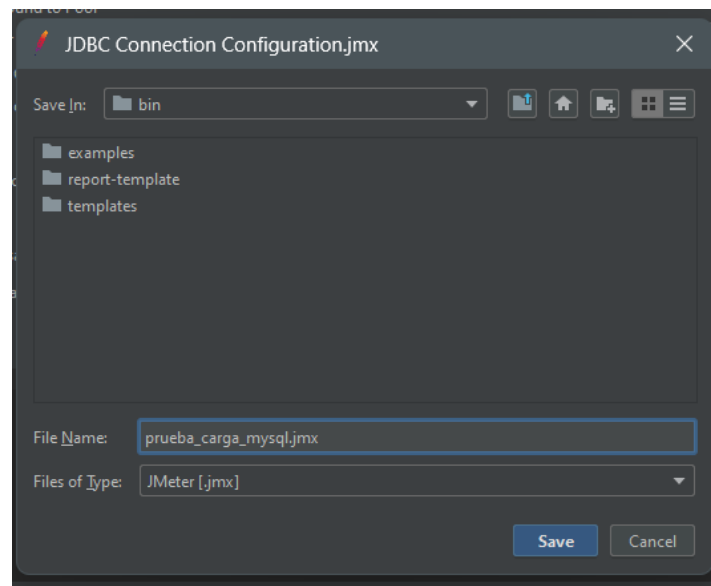
- Este listener dará un resumen estadístico de las pruebas, incluyendo el tiempo promedio de respuesta, el número de solicitudes por segundo y los errores.





### Ahora se ejecutan las Pruebas de Carga

Se Guarda la configuración en un archivo.jmx (por ejemplo, prueba\_carga\_mysql.jmx).



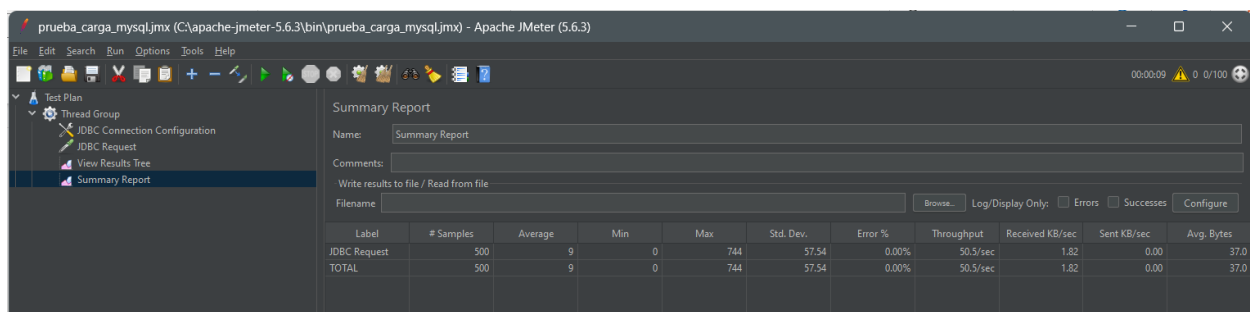
#### 4. Ejecución:

- JMeter simulará los usuarios concurrentes y ejecutará las consultas SQL en tu base de datos.

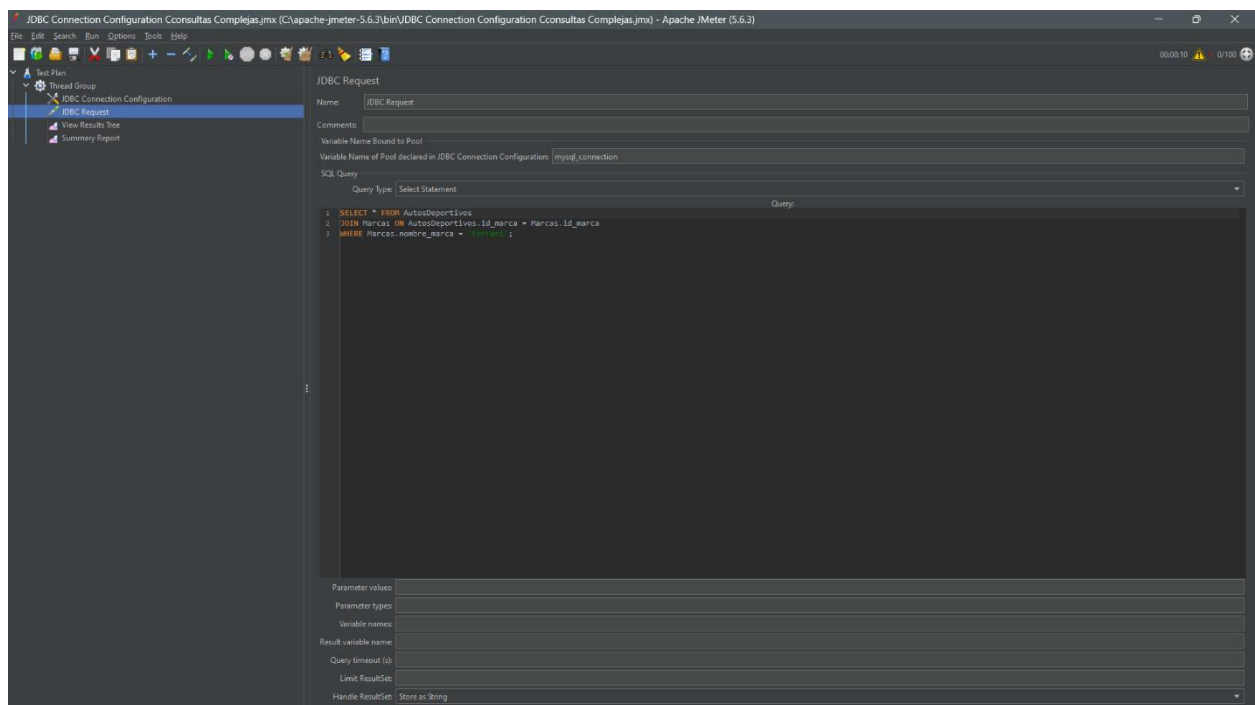
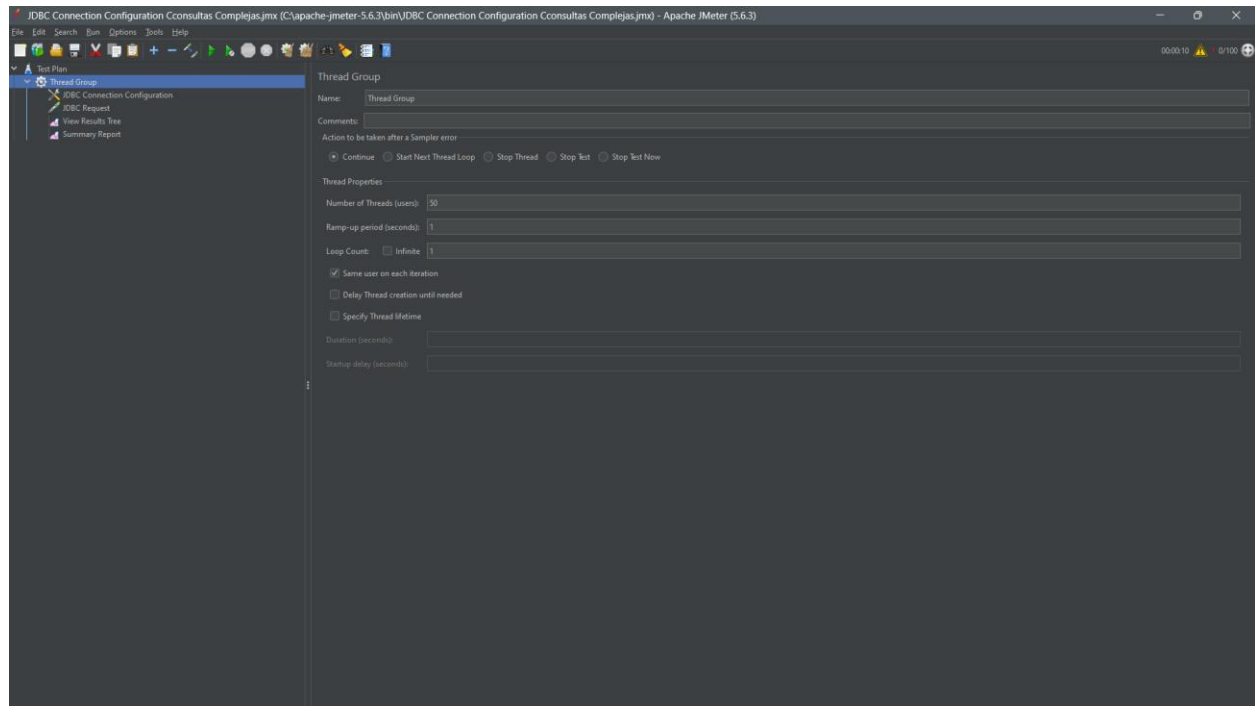
#### 5. Monitorear los resultados:

- Observa los resultados en el **View Results Tree** y el **Summary Report**.
- Prestar atención a:
  - **Tiempo de respuesta:** Cuánto tarda cada consulta en ejecutarse.
  - **Errores:** Si hay consultas que fallan debido a problemas de conexión o tiempo de espera.
  - **Throughput:** Número de consultas por segundo que la base de datos puede manejar.





| Parámetro         | Valor Recomendado          | Explicación  |
|-------------------|----------------------------|--|
| Number of Threads | 50 a 200                   | Número de usuarios concurrentes. Comienza con 50 y aumenta gradualmente. |
| Ramp-Up Period    | 1 a 2 segundos por usuario | Tiempo para activar todos los usuarios.                                  |
| Loop Count        | 1 a 2                      | Número de veces que cada usuario ejecuta la prueba                       |



Se escribe una consulta compleja, por ejemplo:

```
SELECT * FROM AutosDeportivos
```

```
JOIN Marcas ON AutosDeportivos.id_marca = Marcas.id_marca
```

```
WHERE Marcas.nombre_marca = 'Ferrari';
```

### 1. Resultados con parámetros

- **Number of Threads:** 50
- **Ramp-Up Period:** 1s
- **Loop Count:** 1

Summary Report

Name:Summary Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only:

Errors

Successes

Configure

| Label        | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|--------------|-----------|---------|-----|-----|-----------|---------|------------|-----------------|-------------|------------|
| JDBC Request | 50        | 13      | 10  | 23  | 1.75      | 0.00%   | 50.4/sec   | 5.51            | 0.00        | 112.0      |
| TOTAL        | 50        | 13      | 10  | 23  | 1.75      | 0.00%   | 50.4/sec   | 5.51            | 0.00        | 112.0      |

### 2. Resultados con parámetros

- **Number of Threads:** 100
- **Ramp-Up Period:** 1s
- **Loop Count:** 1

Summary Report

Name:

Summary Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only:

☐ Errors

☐ Successes

Configure

| Label        | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|--------------|-----------|---------|-----|-----|-----------|---------|------------|-----------------|-------------|------------|
| JDBC Request | 100       | 13      | 10  | 20  | 2.22      | 0.00%   | 99.5/sec   | 10.88           | 0.00        | 112.0      |
| TOTAL        | 100       | 13      | 10  | 20  | 2.22      | 0.00%   | 99.5/sec   | 10.88           | 0.00        | 112.0      |

### 3. Resultados con parámetros

- **Number of Threads:** 150
- **Ramp-Up Period:** 1s
- **Loop Count:** 1

Summary Report

Name:

Summary Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only:

☐ Errors

☐ Successes

Configure

| Label        | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|--------------|-----------|---------|-----|-----|-----------|---------|------------|-----------------|-------------|------------|
| JDBC Request | 150       | 14      | 10  | 27  | 3.34      | 0.00%   | 141.9/sec  | 15.52           | 0.00        | 112.0      |
| TOTAL        | 150       | 14      | 10  | 27  | 3.34      | 0.00%   | 141.9/sec  | 15.52           | 0.00        | 112.0      |

### 4. Resultados con parámetros

- **Number of Threads:** 200
- **Ramp-Up Period:** 1s
- **Loop Count:** 1

Summary Report

Name:

Summary Report

Comments:

Write results to file / Read from file

Filename

Browse...

Log/Display Only:

☐ Errors

☐ Successes

Configure

| Label        | # Samples | Average | Min | Max | Std. Dev. | Error % | Throughput | Received KB/sec | Sent KB/sec | Avg. Bytes |
|--------------|-----------|---------|-----|-----|-----------|---------|------------|-----------------|-------------|------------|
| JDBC Request | 200       | 12      | 0   | 57  | 8.63      | 24.00%  | 196.3/sec  | 23.60           | 0.00        | 123.1      |
| TOTAL        | 200       | 12      | 0   | 57  | 8.63      | 24.00%  | 196.3/sec  | 23.60           | 0.00        | 123.1      |

The screenshot shows the JMeter 'Text' sampler result view. On the left, a list of 20 'JDBC Request' items is shown, with the 15th item selected. The right pane displays the 'Response data' for the selected request. The 'Response header' section shows a 'shared' attribute with a value of 'false, driver=com.mysql.jdbc.Driver, url=jdbc:mysql://localhost:3306/autosdeportivosdb, user=root'. The 'Additional field' section shows 'Type Result' as 'SampleResult', 'ContentType' as 'text/plain', and 'DataEncoding' as 'UTF-8'.

La causa de los errores en las solicitudes JDBC es la **sobrecarga del servidor**, es posible que MySQL no esté manejando correctamente la cantidad de conexiones simultáneas. Aquí se analiza en más detalle.

MySQL tiene un límite de conexiones simultáneas definido en la configuración (my.cnf o my.ini).

Si el número de conexiones activas supera este límite, MySQL rechaza nuevas solicitudes con un error como:

The screenshot shows the 'Response message' field in the JMeter 'Text' sampler result view. The message is: 'java.sql.SQLException: Cannot create PoolableConnectionFactory (Connection exception, SQL-server rejected establishment of SQL-connection, message from server: "Too many connections")'. The 'Response code' is 'null 0'.

“Too many connections”

The screenshot shows a MySQL command-line interface window titled 'Scrip de Funciones de Usuario...'. The command 'SHOW VARIABLES LIKE 'max\_connections';' has been executed. The result is displayed in a table with two columns: 'Variable\_name' and 'Value'. The value for 'max\_connections' is 151.

| Variable_name   | Value |
|-----------------|-------|
| max_connections | 151   |

```
SET GLOBAL max_connections = 500;
```

| #   | Time     | Action                                | Message           | Duration / Fetch      |
|-----|----------|---------------------------------------|-------------------|-----------------------|
| ✓ 1 | 17:05:12 | SHOW VARIABLES LIKE 'max_connections' | 1 row(s) returned | 0.016 sec / 0.000 sec |
| ✓ 2 | 17:06:20 | SET GLOBAL max_connections = 500      | 0 row(s) affected | 0.000 sec             |

| Result Grid |                 | Filter Rows: |
|-------------|-----------------|--------------|
|             | Variable_name   | Value        |
| ▶           | max_connections | 500          |

[illegible]

Se puede concluir que MySQL ha alcanzado el límite de conexiones simultáneas y está rechazando nuevas conexiones, pero acepta las conexiones hasta el límite establecido anteriormente y algunas pruebas después del error pasaron exitosamente, pero muchas fallaron después de eso lo que confirma que el servidor está sobrecargado.

## 7. Git y Control de Versiones

**Objetivo:** Asegurar que el código relacionado con la base de datos esté versionado y que el equipo pueda colaborar de manera eficiente.

### Actividades:

#### 1. Configurar un repositorio de Git para el proyecto.

**Práctica:** Inicializar un repositorio en Git y subir los archivos de definición de la base de datos, scripts de SQL y procedimientos almacenados.

Link repositorio: <https://github.com/EdwinSarango12/Proyecto-BasesDeDatos>

#### • Inicializar un repositorio en Git:

1. Se abre una terminal en la carpeta raíz del proyecto.
2. Se ejecuta el comando `git init` para inicializar un nuevo repositorio de Git.
3. Se crea un archivo `.gitignore` para excluir archivos que no deben ser versionados (por ejemplo, archivos de configuración local, archivos binarios, etc.).
4. Se agrega los archivos de definición de la base de datos (esquemas, scripts SQL, procedimientos almacenados) al repositorio usando `git add`.
5. Se realiza el primer commit con `git commit -m "Initial commit: Added database schema and SQL scripts"`.

**Investigación:** Investigar buenas prácticas de flujo de trabajo en Git (por ejemplo, uso de ramas, `git merge`).

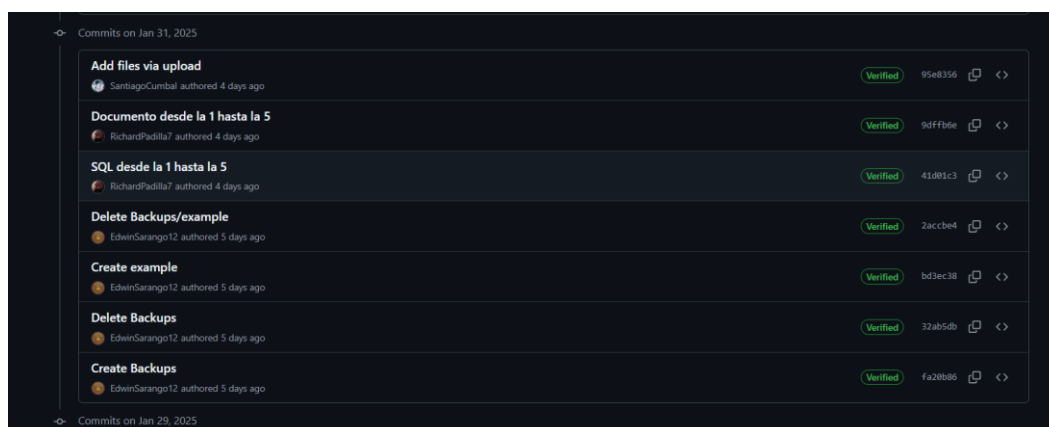
#### • Buenas prácticas de flujo de trabajo en Git:

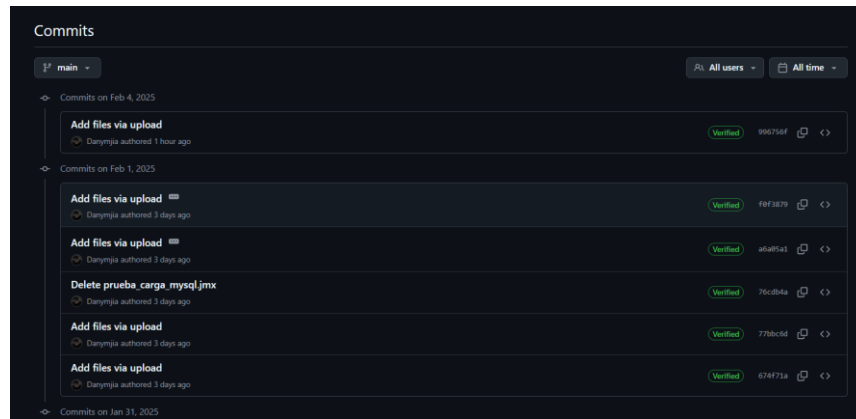
- **Uso de ramas:** Se utilizan ramas para desarrollar nuevas características o corregir errores. Por ejemplo, `git checkout -b feature/nueva-funcionalidad`.
- **Git merge vs. Git rebase:** Se investiga cuándo es apropiado usar `git merge` (para combinar ramas) y `git rebase` (para reescribir el historial de commits).
- **Flujos de trabajo comunes:** Se investiga sobre modelos como Git Flow, GitHub Flow o GitLab Flow, y se elige el que mejor se adapte al proyecto.

**Importancia del Conocimiento:** Git permite la colaboración y el manejo eficiente de cambios en el código, especialmente cuando se trabaja en equipo.

#### 2. Realizar commits frecuentes y con mensajes claros.

**Práctica:** Hacer commits regularmente, describiendo claramente los cambios realizados en los scripts SQL y la estructura de la base de datos.





**Investigación:** Investigar cómo utilizar git rebase y git pull para evitar conflictos.

- Investiga cómo usar git rebase para mantener un historial de commits limpio y lineal.
- Aprende a usar git pull --rebase para evitar conflictos al integrar cambios desde la rama principal.

**Importancia del Conocimiento:** Un flujo de trabajo claro en Git mejora la colaboración y la gestión de versiones.

### 3. Automatizar pruebas con GitHub Actions.

**Práctica:** Crear flujos de trabajo de CI/CD que automaticen las pruebas de las consultas SQL y otros scripts relacionados con la base de datos.

1. Se crea un archivo de configuración de GitHub Actions en la carpeta .github/workflows/ (por ejemplo, sql-tests.yml).
2. Se define un flujo de trabajo que se active en cada push o pull request.
3. Se configura el flujo para ejecutar pruebas automatizadas en los scripts SQL y consultas de la base de datos.

4. Ejemplo de configuración básica:

name: SQL Tests

on: [push, pull\_request]

jobs:

test:

runs-on: ubuntu-latest

steps:

- name: Checkout code

uses: actions/checkout@v2

- name: Set up database

run: |

docker run -d --name test-db -e MYSQL\_ROOT\_PASSWORD=root -p 3306:3306

mysql:latest

# Esperar a que la base de datos esté lista

sleep 10

- name: Run SQL tests

run: |

mysql -h 127.0.0.1 -u root -proot < scripts/test.sql

**Investigación:** Investigar sobre integración continua y cómo aplicarla en bases de datos con GitHub Actions.

- Se Investiga cómo integrar pruebas automatizadas en bases de datos, incluyendo la ejecución de scripts SQL en entornos aislados (por ejemplo, usando Docker).
- Se exploran herramientas como dbunit o tSQLt para pruebas de bases de datos.

**Importancia del Conocimiento:** Las pruebas automáticas aseguran que las bases de datos se mantengan consistentes y funcionales a lo largo del tiempo.

- Git es esencial para la colaboración en equipo, permitiendo un control preciso de los cambios en el código y la base de datos.
- Commits frecuentes y bien documentados mejoran la trazabilidad y facilitan la resolución de conflictos.
- Automatización de pruebas con herramientas como GitHub Actions garantiza la consistencia y funcionalidad de la base de datos en cada cambio.

Al implementar estas prácticas, el equipo trabaja de manera más eficiente, reduce errores y mantiene un código de base de datos robusto y bien documentado.