**Proyecto de Base de Datos Grupal BD**

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

**Fecha de entrega: 5 febrero 2025**

**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.

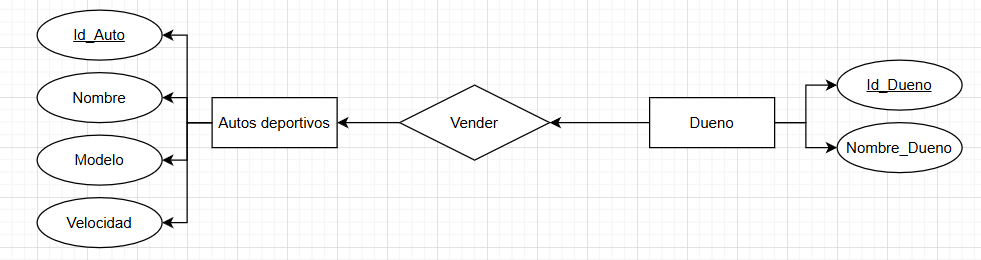
**Importancia del Conocimiento:** Diseñar bases de datos eficientes es crucial para:

* 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.

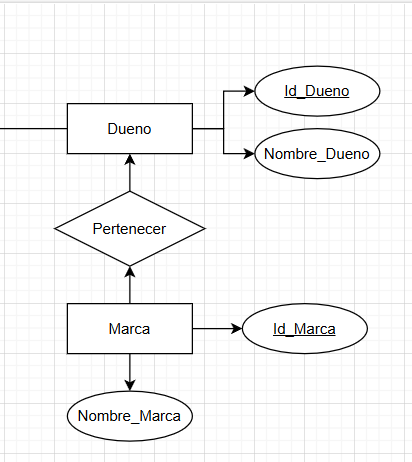
Para el presente proyecto hemos ideado un modelo de Autos deportivos como claves importantes tenemos a los nombres y modelos, Dueños de las marcas, cantidad de esas unidades en varios países, costo de los autos. Cada uno de estos datos estarán integrados en una base de datos fácil de entender al usuario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

**Diagrama entidad relación de Autos Deportivos**

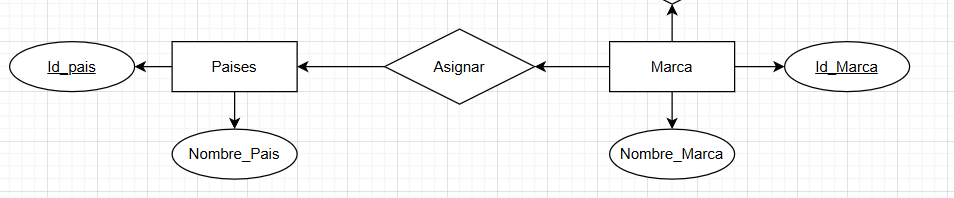
Para el presente proyecto hemos utilizado una herramienta gráfica para elaborar estos diagramas, Draw.io es una herramienta que integra todas las opciones gráficas de relación para generar nuestro modelo ER para el proyecto.



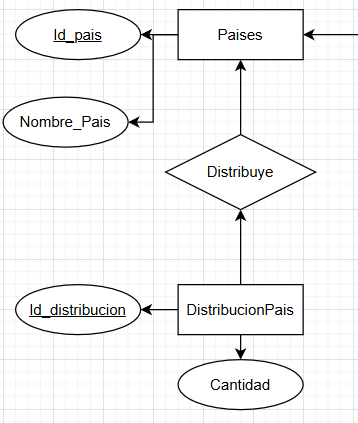
En cuanto a la primera relación obtenemos que es (M:1) ya que en primera instancia el dueño legítimo de la franquicia o la marca del carro es uno solo, pueden existir asociados, pero en este caso solo constaría como dueño legitimo uno solo, mientras que la relación M se refiere a que este único dueño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



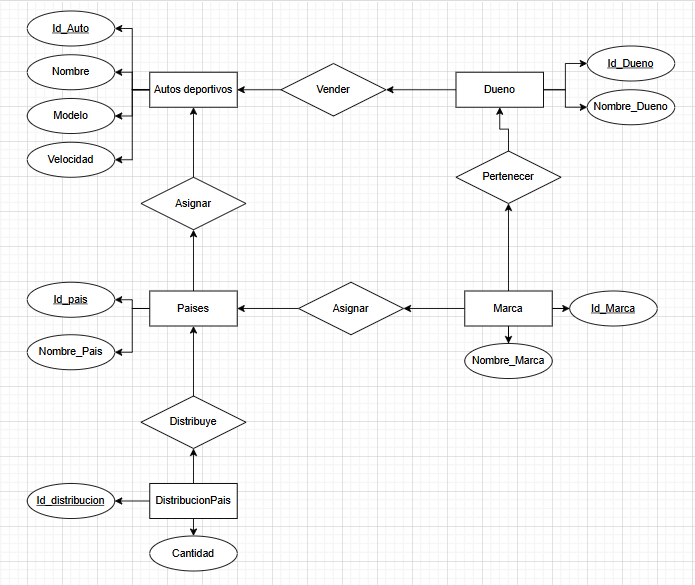
Relación Marca y Dueño es de (1:1) puesto que para el presente proyecto contamos solo con marcas reconocidas y mediante la investigación del proyecto se concluye que en general las marcas de autos reconocidas pertenecen a un solo dueño que en este caso para el proyecto es el dueño de la marca en sí, 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.

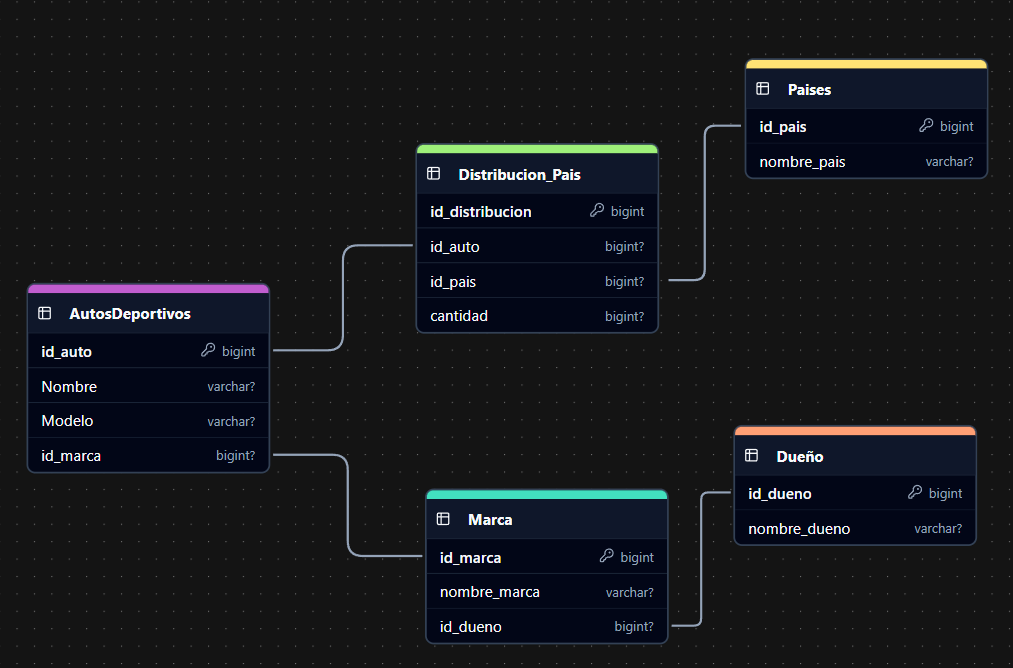


Como ultima relación del modelo ER del proyecto tenemos un agregado más como un estadístico en el que podemos conocer en qué países existe la mayor cantidad de autos o modelos vendidos por lo que la relación es de (M:M) suponiendo que mucha cantidad de autos es vendida en muchos países, al igual que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

1. **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 |
| **Paises** | 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

1. Usar AES\_ENCRYPT() en MySQL para proteger información confidencial como contraseñas.

* Autenticación Segura

1. Usar contraseñas fuertes y políticas de caducidad de contraseñas.

* Registros de Auditoría

1. Monitorear actividades sospechosas con logs de acceso.

* Firewalls y Listas de Control de Acceso (ACLs)

1. Restringir conexiones solo a direcciones IP de confianza.

* Respaldo de Datos Periódico

1. 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.

1. **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.

1. **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.

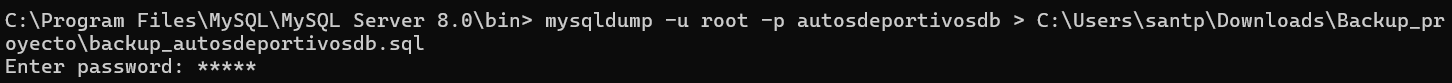
**Actividades:**

1. **Crear respaldos completos (full backups).**

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

Texto

Descripción generada automáticamente



Captura de pantalla de un celular

Descripción generada automáticamente

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.

1. **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.

1. **Implementar respaldos en caliente (Hot Backups).**

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

De igual forma Percona XtraBackup se encargade hacer 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 sera el hecho del rendimiento para la propia base de datos lo cual no pasara 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

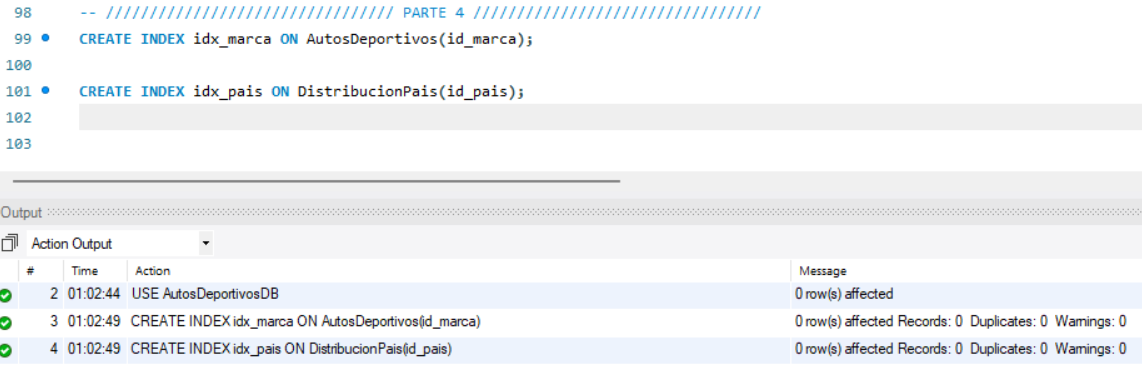
**Objetivo:** Mejorar la eficiencia en la recuperación de datos mediante la optimización de consultas y el uso adecuado de índices.

**Actividades:**

1. **Crear y gestionar índices.**

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

Crear dos índices importantes





Texto

Descripción generada automáticamente con confianza media

**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.

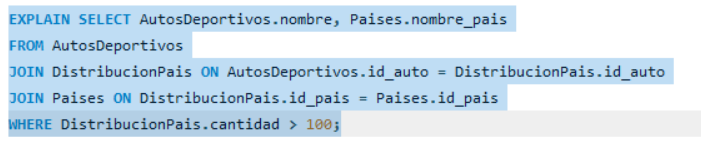
**Índice Full-Text**: Utilizado para buscar dentro de cadenas de texto grandes

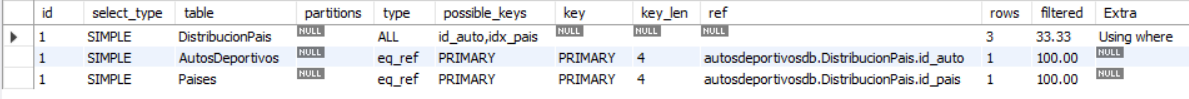
**Importancia del Conocimiento:** Los índices son cruciales para acelerar las consultas y mejorar el rendimiento general de la base de datos.

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.

1. **Optimizar consultas SQL.**

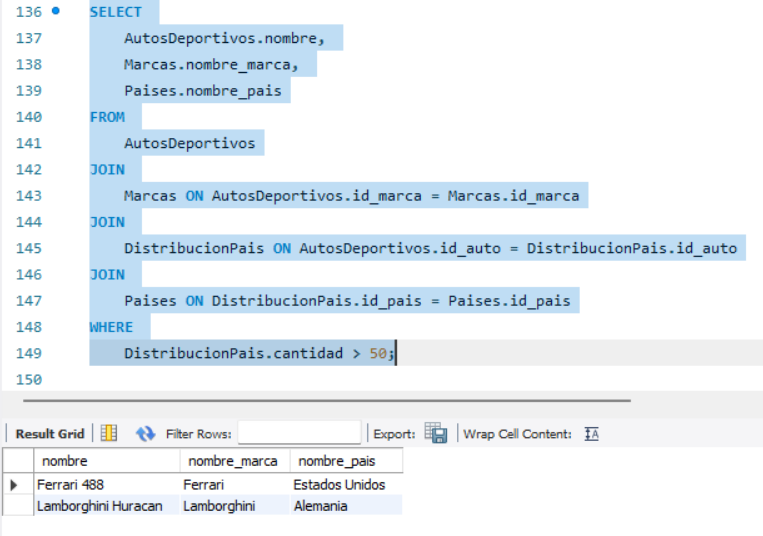
**Práctica:** Utilizar herramientas como EXPLAIN para identificar cuellos de botella en las consultas y optimizarlas.





El código realiza una consulta que selecciona el nombre del automóvil (AutosDeportivos.nombre) y el nombre del país (Paises.nombre\_pais) de las tablas AutosDeportivos y Paises, respectivamente. Utiliza dos uniones (JOIN): la primera une la tabla AutosDeportivos con DistribucionPais mediante la columna id\_auto, y la segunda une DistribucionPais con Paises 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**

****

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

JOIN los diferetntestipos 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.

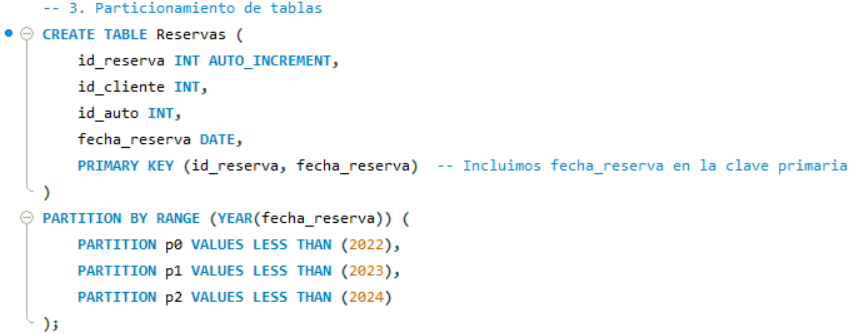
**Subconsultas**: Son útiles para evitar múltiples consultas, pero deben usarse con precaución debido a su posible impacto en el rendimiento.

**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.

1. **Utilizar particionamiento de tablas.**

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



Texto

Descripción generada automáticamente

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%.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

-- 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 ;

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

**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.

1. **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).

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

-- 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.costo 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.costo as 'Precio Base',

(case

when AutosDeportivos.modelo = '2021' then AutosDeportivos.costo \* 0.10

else 0

end) as 'Descuento',

(case

when DistribucionPais.id\_pais = 1 then AutosDeportivos.costo \* 0.05

else 0

end) as 'Cargo Adicional',

(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

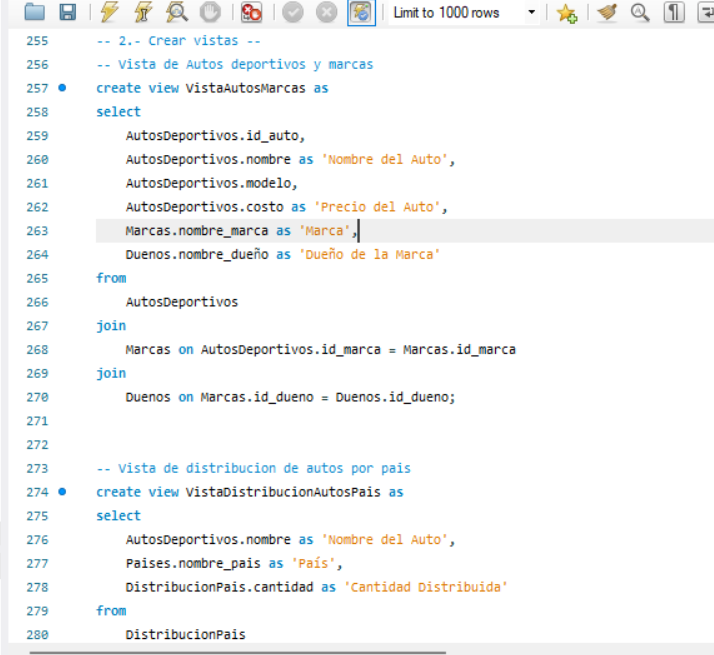
end)) as 'Precio Final'

from

AutosDeportivos

join

DistribucionPais on AutosDeportivos.id\_auto = DistribucionPais.id\_auto;



Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Imagen que contiene Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**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.

1. **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 loos 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';

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

**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.

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

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

1. **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.

1. **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.

# 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.

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

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

1. **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.

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

1. **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.

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

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

**CONSIDERACIONES**

**Sugerencia para mejorar el trabajo en equipo y habilidades blandas:**

Para optimizar la colaboración, sugiero crear una **tabla de responsabilidades y capacitación**. Esta tabla permitirá monitorear quién es responsable de cada tema, qué actividades se han realizado para capacitar a los compañeros y cuándo se realizaron. Esto fomenta la responsabilidad individual y la transparencia en el equipo.

### Ejemplo de tabla de seguimiento:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Responsable** | **Tema Asignado** | **Fecha de asigación** | **Fecha de culminación** | **Fecha de Capacitación** | **Capacitación a Compañeros** | **Observaciones** |
| Guissela Franco | Consultas |  |  | 01/12/2025 | Índices y optimización | Uso de EXPLAIN |
| Danna López | Seguridad |  |  | 03/12/2025 | Cifrado y control de acceso | Implementación |

**Mejoras en habilidades blandas:**

**Comunicación efectiva:** Promover reuniones de retroalimentación para que todos los miembros intercambien ideas y soluciones. Fomentar la participación activa en las discusiones y evitar el trabajo mecánico.

**Investigación y curiosidad:** Incentivar a los miembros a investigar profundamente sobre los temas, identificando problemas no documentados y buscando soluciones innovadoras.

**Colaboración activa:** Fomentar un ambiente de colaboración, promoviendo sesiones de brainstorming y revisiones entre compañeros, y asegurando que todos estén alineados con el progreso del proyecto.

**Responsabilidad colectiva:** Asegurar que los miembros del equipo no solo sean responsables de sus tareas individuales, sino también del éxito global del proyecto. Esto incluye apoyar a los compañeros en su aprendizaje.

**Temáticas Disponibles**

* **El grupo es libre de elegir una temática sin repetirse con los demás grupos, seguir el ejemplo indicado**

**Entregables**

Texto

Descripción generada automáticamente

**EXPLICACION**

**Presentación (PPT)**:

Crear una carpeta llamada Presentaciones donde se suba el archivo .ppt o .pptx correspondiente a la explicación del proyecto, los objetivos, las actividades, y los resultados alcanzados.

**Documento Informe**:

* Subir el informe detallado del proyecto en formato .docx o .pdf, incluyendo:

Resumen ejecutivo

Descripción de cada fase del proyecto

Resultados obtenidos

Conclusiones

**Modelados (ER)**:

Crear una carpeta llamada Modelados para almacenar los diagramas de modelado ER. Estos pueden estar en formatos como .png, .jpg, .pdf.

* Incluir versiones del modelo conceptual, lógico y físico.

**Diccionario de Datos**:

Subir un archivo en formato .xlsx o .csv que contenga el diccionario de datos. Este debe incluir detalles como:

Nombre de la tabla

Descripción de la tabla

Campos (nombre, tipo de datos, restricciones, etc.)

Relación con otras tablas

**Responsabilidades**:

Subir un archivo que detalle las responsabilidades de cada miembro del equipo, indicando qué tareas corresponden a cada uno. Este archivo puede ser una tabla en formato .xlsx o .docx.

**Script Actividades a Realizar**:

Subir los scripts de las actividades, como la creación de la base de datos, la implementación de procedimientos almacenados, vistas, triggers, etc. Estos archivos pueden ser .sql o .sh (si son scripts de shell para automatizar tareas).

Estos scripts deben estar organizados en carpetas según la actividad, por ejemplo, Scripts/Modelado, Scripts/Seguridad, Scripts/Auditoría, etc.

**RUBRICA**

|  |  |  |
| --- | --- | --- |
| **Criterio** | **Descripción** | **Puntos** |
| **1. Modelado de Base de Datos y Diccionario de Datos** | | **8** |
| **Diseño del Modelo Conceptual, Lógico y Físico** | El modelo ER refleja las entidades y relaciones correctamente. | 4 |
| **Diccionario de Datos** | El diccionario de datos es detallado, claro y bien estructurado, incluye tablas, campos, relaciones y restricciones. | 2 |
| **Restricciones de Integridad Referencial** | Se definen correctamente las claves primarias y foráneas entre las tablas. | 1 |
| **Escalabilidad y Mejores Prácticas** | El modelo incluye prácticas recomendadas para la escalabilidad y la integración de sistemas de reservas. | 1 |
| **2. Seguridad, Auditoría y Control de Acceso** | | **8** |
| **Políticas de Acceso y Seguridad** | Roles y permisos bien definidos para controlar el acceso a las tablas y vistas. | 3 |
| **Cifrado de Datos Sensibles** | Se implementa correctamente el cifrado de datos sensibles, como contraseñas y detalles de pago. | 2 |
| **Auditoría y Registro de Eventos** | Se habilitan herramientas de auditoría para monitorear el acceso y las actividades de los usuarios en la base de datos. | 3 |
| **3. Respaldos y Recuperación de Datos** | | **5** |
| **Respaldos Completos e Incrementales** | Los respaldos completos e incrementales están implementados y explicados correctamente. | 3 |
| **Respaldos en Caliente** | Se implementa correctamente el respaldo sin interrumpir el servicio (hot backups). | 2 |
| **4. Optimización y Rendimiento de Consultas** | | **5** |
| **Índices y Optimización de Consultas** | Se implementan índices apropiados y se optimizan consultas SQL con herramientas como EXPLAIN. | 3 |
| **Particionamiento de Tablas** | El particionamiento de tablas está correctamente implementado y explicado. | 2 |
| **5. Procedimientos Almacenados, Vistas y Triggers** | | **5** |
| **Procedimientos Almacenados** | Se crean procedimientos almacenados para cálculos y tareas recurrentes. | 2 |
| **Vistas y Simplificación de Consultas** | Se crean vistas para simplificar consultas complejas y mejorar el acceso a datos. | 2 |
| **Triggers para Auditoría y Control de Cambios** | Se implementan triggers para mantener un historial de cambios y automatizar tareas. | 1 |
| **6. Monitoreo y Optimización de Recursos** | | **2** |
| **7. Git y Control de Versiones** | | **2** |
| **Uso de Git y Control de Versiones** | El repositorio está correctamente configurado, con commits claros y frecuentes. | 1 |
| **Colaboración y Flujo de Trabajo en Git** | Se siguen buenas prácticas en el flujo de trabajo (uso de ramas, fusión, etc.). | 1 |
| **Total** |  | **35** |