

BASE DE DATOS 1º AÑO - 2025

Clase N° 6: Optimización SQL

Contenido: Optimización SQL.

Objetivos de la Clase:

En la clase de hoy, vamos a aprender a:

- Mejorar los tiempos de respuesta de un sistema de gestión de bases de datos relacional.
- Comprender los tipos de acceso a una base de datos.
- Identificar y controlar las amenazas a la seguridad de una base de datos.
- Implementar la seguridad de una base de datos.
- Entender la autenticación de usuarios.
- Aplicar técnicas de optimización de SQL.
- Conocer la auditoría de SQL.

Presentación:

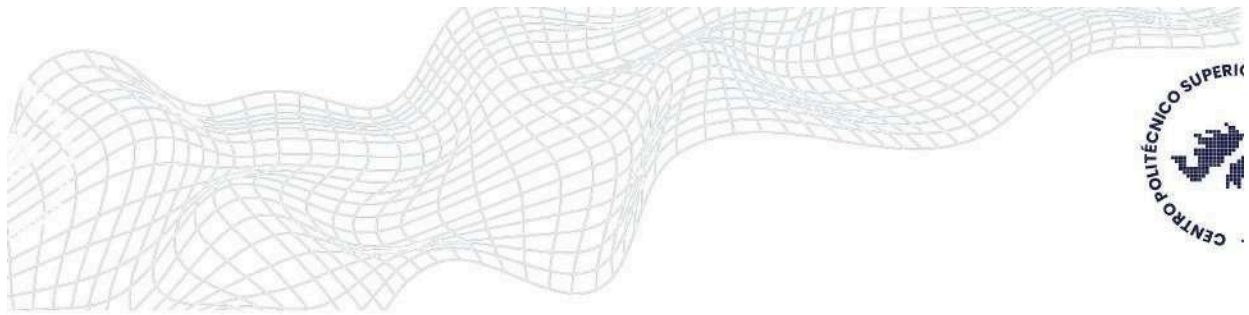
¡Bienvenidos a la clase N° 6 de Base de Datos!

En esta clase, nos centraremos en un tema fundamental: la optimización de consultas SQL. Optimizar las consultas SQL es el proceso de mejorar el rendimiento de las consultas a la base de datos. Esto se logra identificando y eliminando los "cuellos de botella" que hacen que las consultas sean lentas, permitiendo que se ejecuten de forma más rápida y eficiente. En resumen, se trata de hacer que la base de datos nos entregue la información que pedimos lo más rápido posible.

Desarrollo y Actividades:

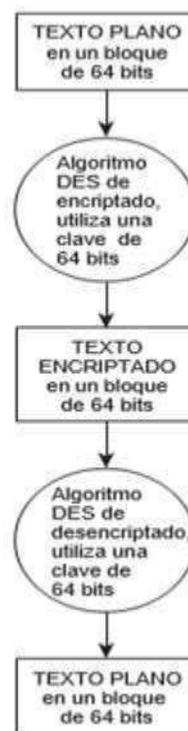
1. INTRODUCCIÓN A LA SEGURIDAD DE LAS BASES DE DATOS

La **seguridad en las bases de datos** se refiere a la protección de las bases de datos contra el acceso, la modificación o la destrucción no autorizados. Dado que las bases de datos son un recurso esencial para cualquier organización, la seguridad es un objetivo crítico.



Además de la necesidad de preservar y proteger los datos para la continuidad operativa de la organización, los diseñadores de bases de datos tienen la responsabilidad de proteger la **privacidad** de las personas cuyos datos se almacenan. La privacidad es el derecho de los individuos a tener control sobre su información personal. Muchas leyes protegen la privacidad, y las organizaciones deben cumplir con estas leyes al recopilar y almacenar información personal.

El diseño de la base de datos debe reflejar el compromiso de la organización con la privacidad, incluyendo solo la información necesaria y protegiendo los datos confidenciales.

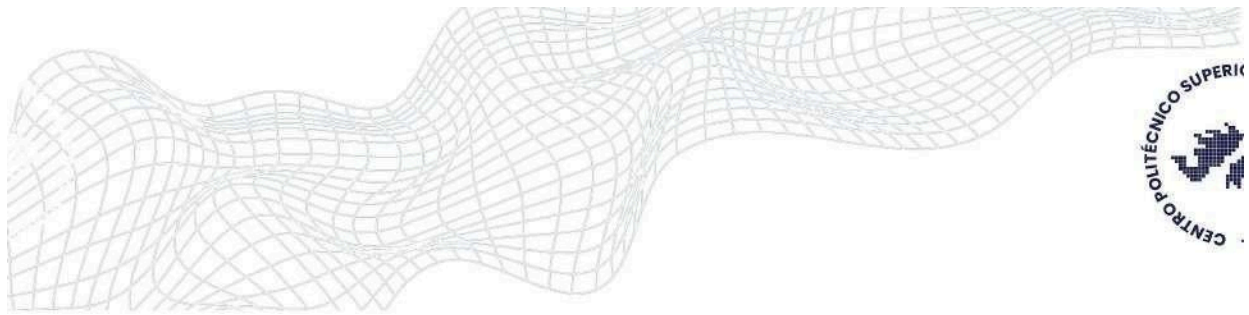


Las **amenazas a la seguridad** pueden ser accidentales o deliberadas:

1.1. Amenazas Accidentales a la Seguridad:

Son errores o incidentes no intencionales que pueden comprometer la seguridad de la base de datos. Ejemplos:

- Un usuario solicita acceso a datos a los que no está autorizado, y se le concede el acceso debido a un error en el sistema.
- Una persona envía accidentalmente información confidencial al usuario



equivocado.

- Un error en el sistema de comunicación conecta a un usuario a la sesión de otro, dándole acceso a información privilegiada.
- El sistema operativo sobrescribe archivos de la base de datos por error, borra información importante o envía los archivos incorrectos a los usuarios.

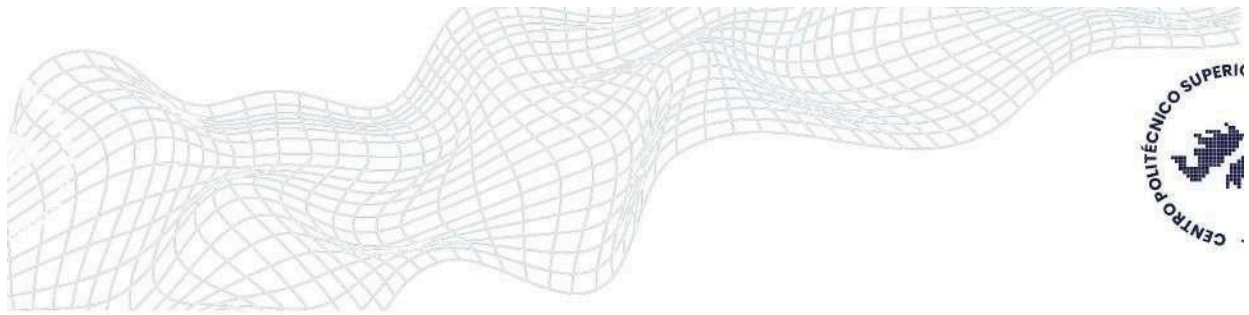
1.2. Amenazas Deliberadas a la Seguridad:

Son acciones intencionales realizadas por usuarios con el objetivo de obtener acceso no autorizado a la base de datos. Ejemplos:

- Interceptar las comunicaciones entre la base de datos y los usuarios para robar información.
- Utilizar dispositivos de espionaje electrónico para capturar información de las pantallas o impresoras.
- Leer o copiar información que los usuarios autorizados dejan desatendida.
- Hacerse pasar por un usuario autorizado para obtener acceso a la base de datos (suplantación de identidad).
- Escribir programas con código malicioso para evitar los controles de seguridad de la base de datos y acceder directamente a los datos.
- Escribir programas que realizan operaciones no autorizadas en la base de datos.
- Utilizar técnicas de consulta avanzadas para obtener información oculta en la base de datos.
- Robar dispositivos de almacenamiento físico que contienen datos de la base de datos.
- Hacer copias no autorizadas de los archivos de la base de datos, evitando los controles de seguridad.
- Sobornar, amenazar o manipular a usuarios autorizados para obtener información o dañar la base de datos.

2. SEGURIDAD FÍSICA Y AUTENTICACIÓN DEL USUARIO

La seguridad de las bases de datos debe ser parte de un plan de seguridad más amplio que incluya medidas para proteger las instalaciones físicas donde se almacenan los datos.



La **seguridad física** implica proteger el edificio y las salas de servidores. Esto puede incluir:

- Controlar el acceso a las instalaciones mediante guardias de seguridad, tarjetas de identificación, escáneres de huellas dactilares, etc.
- Proteger las copias de seguridad de la base de datos almacenándolas en lugares seguros.

La **autenticación del usuario** es el proceso de verificar la identidad de un usuario que intenta acceder a la base de datos. Se implementa comúnmente a nivel del sistema operativo y puede incluir:

- Solicitar un nombre de usuario (ID) y una contraseña.
- Utilizar tarjetas de identificación, llaves, escáneres de voz, huellas dactilares, retina, etc.
- Implementar procedimientos de autenticación más complejos, como preguntas de seguridad.

Es importante destacar que las contraseñas deben mantenerse en secreto, cambiarse con frecuencia y almacenarse de forma segura (por ejemplo, encriptadas).

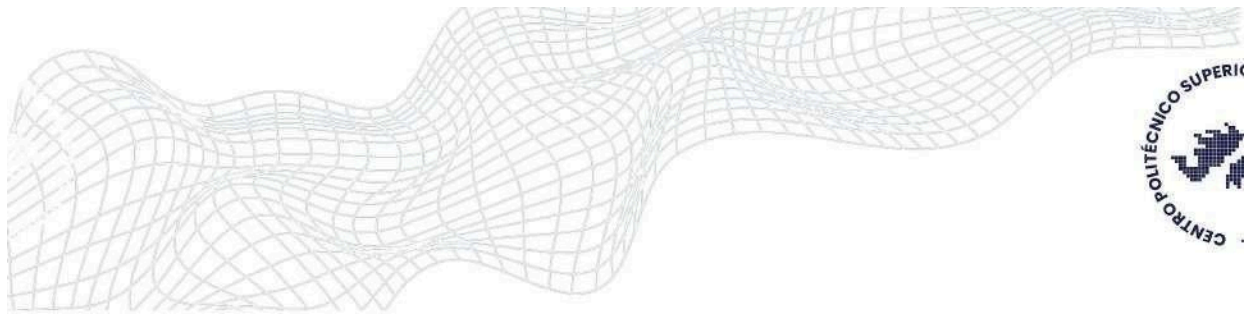
Además de la autenticación a nivel del sistema operativo, algunos sistemas de gestión de bases de datos (DBMS) pueden requerir una autenticación adicional al nivel de la base de datos.

3. AUTORIZACIÓN

La **autorización** es el proceso de asignar a los usuarios derechos o privilegios para acceder y utilizar los objetos de la base de datos. La mayoría de los DBMS multiusuario tienen subsistemas de seguridad que permiten al Administrador de la Base de Datos (ABD) especificar qué usuarios pueden acceder a qué datos y qué operaciones pueden realizar.

El ABD utiliza un **lenguaje de autorización** (como los comandos GRANT y REVOKE en SQL) para definir **reglas de autorización**. Estas reglas especifican quién tiene acceso a qué información y qué acciones pueden realizar (por ejemplo, leer, insertar, actualizar o eliminar datos).

El mecanismo de autorización protege la base de datos al prevenir el acceso



no autorizado a los datos. Es fundamental que los diseñadores de bases de datos implementen estos mecanismos de seguridad para proteger la información valiosa.

4. CONTROL DEL ACCESO

El **control del acceso** es la forma en que se implementan las autorizaciones. Se trata de garantizar que los datos y otros recursos se accedan solo de las maneras permitidas.

El ABD puede utilizar una **matriz de control de acceso** para planificar y gestionar el acceso a la base de datos.

- Las **columnas** de la matriz representan los objetos de la base de datos (tablas, vistas, datos, etc.).
- Las **filas** representan a los usuarios, roles, grupos de usuarios o aplicaciones.
- Las **celdas** de la matriz especifican el tipo de acceso permitido (por ejemplo, READ, INSERT, UPDATE, DELETE).

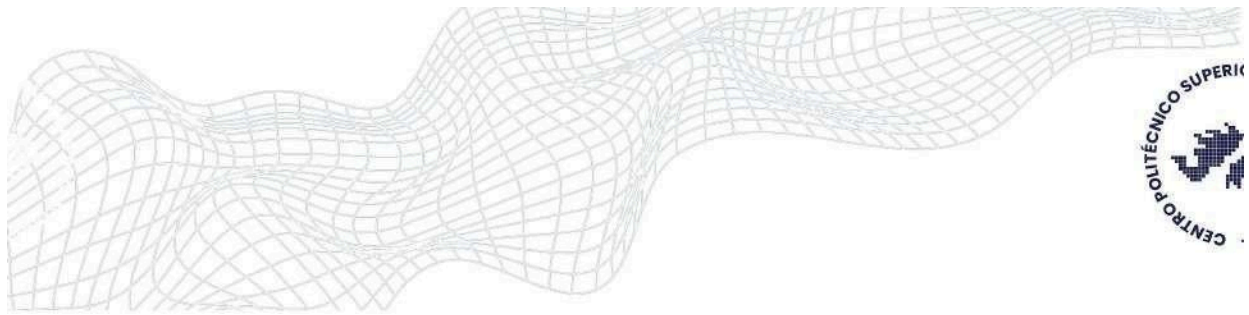
Una vez que se ha definido la matriz de control de acceso, el ABD utiliza el lenguaje de autorización para implementarla.

El ABD tiene el poder de crear y modificar la estructura de la base de datos, así como de conceder o revocar el acceso a otros usuarios. En algunos casos, el ABD puede delegar parte de su poder de autorización a otros usuarios. Sin embargo, esto puede ser peligroso si no se gestiona adecuadamente, ya que puede llevar a una situación en la que el control del acceso se vuelve difícil de administrar.

5. USO DE VISTAS PARA EL CONTROL DEL ACCESO

Una **vista** es una tabla virtual que se crea a partir de una consulta a una o más tablas base. Las vistas son una herramienta útil para el control del acceso porque:

- Simplifican el modelo de la base de datos para los usuarios, ocultando la complejidad de las tablas subyacentes.
- Actúan como una medida de seguridad, ocultando datos que los



usuarios no deberían ver.

Las vistas pueden ser:

- **Dependientes del valor:** Se crean utilizando una cláusula **WHERE** en la consulta **SELECT** para mostrar solo ciertas filas de una tabla (por ejemplo, mostrar solo los estudiantes de informática).
- **Independientes del valor:** Se crean seleccionando solo ciertas columnas de una tabla para mostrar solo información específica (por ejemplo, mostrar solo el ID, nombre y especialidad de los estudiantes).

6. REGISTROS DE SEGURIDAD Y PROCEDIMIENTOS DE AUDITORÍA

Un **registro de seguridad** (o bitácora) es un registro de todos los intentos de violar la seguridad de la base de datos. Esto permite detectar y responder a posibles ataques.

La **auditoría** es un proceso que registra todas las acciones realizadas en la base de datos, incluyendo:

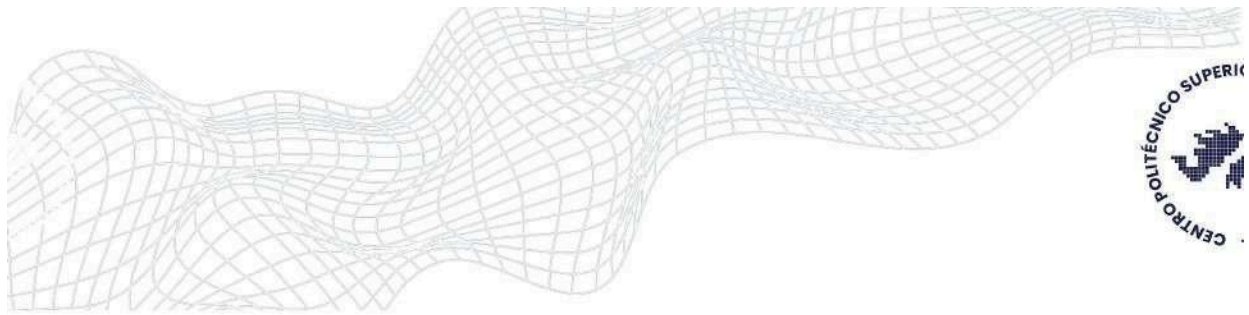
- Quién accedió a la base de datos.
- Qué operaciones se realizaron.
- Desde qué estación de trabajo se accedió.
- Cuándo ocurrió el acceso.
- Qué datos se accedieron y cuáles fueron sus valores antes y después de cualquier modificación.

La auditoría permite rastrear las acciones de los usuarios, incluso los usuarios autorizados, y puede ayudar a identificar la causa de problemas o accesos no autorizados.

Los **disparadores** se pueden utilizar para iniciar automáticamente un procedimiento de auditoría cuando se realizan cambios en una tabla específica.

7. ENCRIPTADO

El **encriptado** es el proceso de codificar los datos para que solo puedan ser leídos por personas autorizadas. Los datos encriptados se ven como texto sin sentido para cualquiera que no tenga la clave de desencriptado.



El encriptado se utiliza para proteger los datos almacenados en la base de datos y los datos que se transmiten entre sistemas.

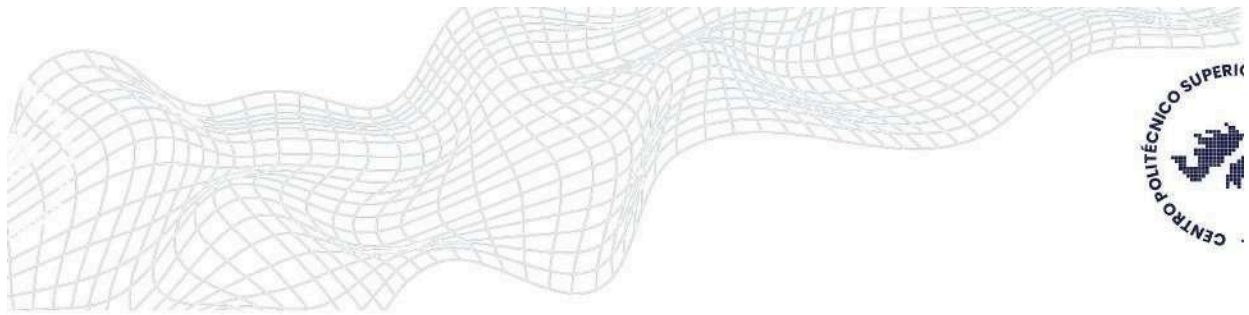
Un **sistema de encriptado** consta de:

- Un **algoritmo de encriptado**: Un conjunto de reglas para transformar el texto normal (texto plano) en texto encriptado (texto cifrado).
- Una **clave de encriptado**: Una pieza de información secreta que se utiliza con el algoritmo de encriptado.
- Un **algoritmo de desencriptado**: Un conjunto de reglas para transformar el texto encriptado de nuevo en texto normal.
- Una **clave de desencriptado**: Una pieza de información secreta que se utiliza con el algoritmo de desencriptado.



Existen diferentes tipos de algoritmos de encriptado, incluyendo:

- **Encriptado simétrico**: Utiliza la misma clave para encriptar y desencriptar los datos (por ejemplo, el Estándar de Encriptado de Datos o DES, y el Estándar de Encriptado Avanzado o AES).
- **Encriptado de clave pública**: Utiliza un par de claves, una clave pública para encriptar los datos y una clave privada para desencriptarlos¹ (por ejemplo, el algoritmo RSA).
- www.itdo.com
- www.itdo.com



8. OPTIMIZACIÓN DE SQL

Las **malas prácticas** al escribir consultas SQL pueden resultar en consultas lentas que se vuelven aún más lentas a medida que aumenta la cantidad de datos en la base de datos. Por ejemplo, una consulta que inicialmente tarda 200 ms puede tardar 4 segundos o más cuando la tabla contiene cientos de miles o millones de registros.

La **optimización de SQL** es el proceso de mejorar el rendimiento de las consultas SQL para que se ejecuten más rápido y utilicen los recursos de la base de datos de manera más eficiente.

Es importante recordar que, en la mayoría de los casos, los problemas de rendimiento de SQL se pueden resolver optimizando las consultas y la estructura de la base de datos, en lugar de recurrir a tecnologías de Big Data.

8.1. Medición

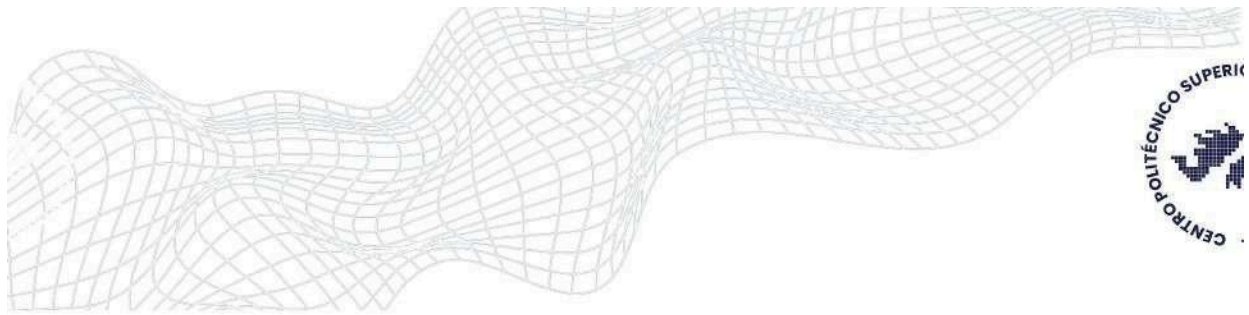
La **medición** es fundamental para la optimización del rendimiento. Debemos medir el tiempo que tardan las consultas en ejecutarse antes y después de realizar cambios para determinar si las optimizaciones son efectivas.

- Los sistemas de gestión de bases de datos (DBMS) proporcionan herramientas para medir el tiempo de ejecución de las consultas.
- Las librerías de acceso a datos en los lenguajes de programación también pueden proporcionar información sobre el rendimiento de las consultas.
- Existen herramientas de monitoreo de bases de datos (como Newrelic) que ofrecen información detallada sobre el rendimiento de la base de datos.

8.2. La Regla 80/20 (Principio de Pareto)

El **Principio de Pareto** establece que el 80% de los resultados provienen del 20% del esfuerzo. En el contexto de la optimización de SQL, esto significa que el 80% de las mejoras de rendimiento se pueden lograr optimizando el 20% de las consultas.

Es más eficiente enfocarse en optimizar las consultas que tienen el mayor impacto en el rendimiento general, en lugar de intentar optimizar cada



consulta individual.

8.3. Mediciones en el Gestor de Base de Datos

Los DBMS proporcionan herramientas para medir el tiempo de ejecución de las consultas. Por ejemplo, en MySQL:

- El tiempo de ejecución de una consulta se muestra en la terminal después de ejecutar la consulta.
- Se puede activar el "profiling" para registrar información detallada sobre el tiempo de ejecución de cada parte de una consulta.

8.4. Mediciones en la Aplicación

Las librerías de acceso a datos (como ActiveRecord en Rails o Eloquent en Laravel) pueden proporcionar información sobre el tiempo de ejecución de las consultas. Es importante configurar estas librerías para que registren esta información.

8.5. Herramientas Independientes del Stack Tecnológico

Existen herramientas de monitoreo de bases de datos que son independientes de la tecnología utilizada para desarrollar la aplicación (por ejemplo, Newrelic). Estas herramientas proporcionan información valiosa sobre el rendimiento de la base de datos, incluyendo qué consultas son más lentas o se ejecutan con mayor frecuencia.

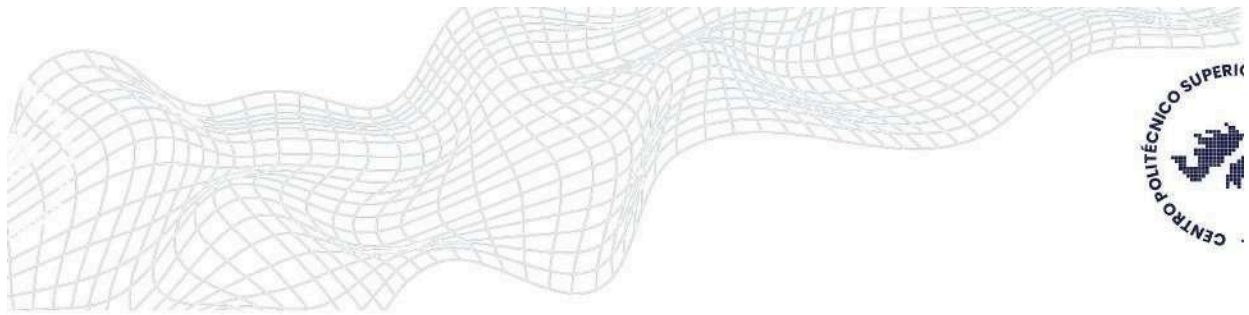
8.6. Tipos de Optimización

El primer paso para optimizar una base de datos SQL es **aprender SQL en profundidad**. A menudo, el mismo resultado se puede obtener con diferentes tipos de consultas, y algunas consultas se pueden combinar para mejorar el rendimiento.

Además de un buen conocimiento de SQL, existen técnicas generales que se pueden aplicar para mejorar el rendimiento de las consultas:

8.7. Índices

Los **índices** son estructuras de datos que permiten al DBMS encontrar filas específicas en una tabla de forma rápida. Agregar un índice a una columna



puede acelerar significativamente las consultas que filtran u ordenan por esa columna.

Sin embargo, los índices tienen un costo: ralentizan las operaciones de inserción, actualización y eliminación de datos. Por lo tanto, es importante utilizar los índices de forma adecuada.

- Las consultas son lentas si filtran u ordenan por una columna que no tiene un índice.
- A veces, los índices existen pero no se utilizan correctamente. Por ejemplo, si una consulta ordena por un campo que no es el índice principal, puede ser lenta.

Es importante revisar periódicamente los índices de una tabla y eliminar los índices innecesarios para acelerar las operaciones de escritura en la base de datos.

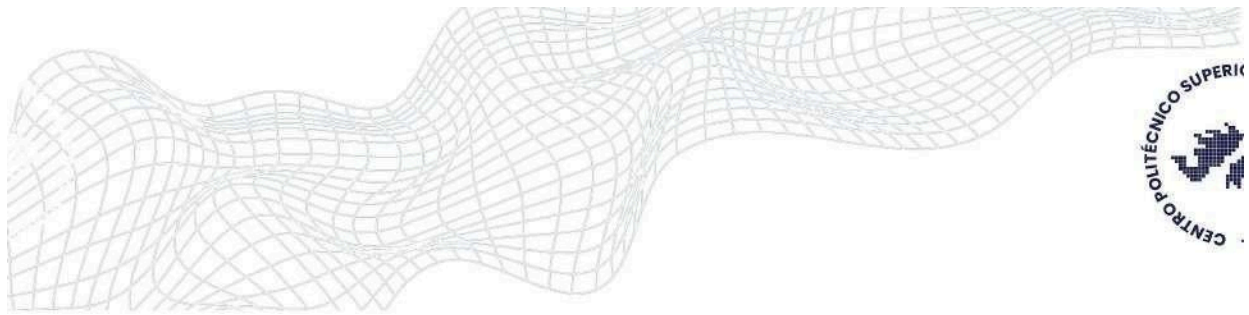
8.8. El Problema de N+1 Consultas

El **problema de N+1 consultas** ocurre cuando una aplicación ejecuta una consulta a la base de datos por cada elemento de una lista de resultados. Esto puede resultar en un gran número de consultas y un rendimiento deficiente.

Este problema suele ser el resultado del mal uso de las librerías de acceso a datos en la aplicación.

La solución es utilizar la **carga anticipada** de los registros relacionados utilizando operaciones **JOIN** en las consultas. Esto permite obtener todos los datos necesarios en una sola consulta, en lugar de realizar múltiples consultas individuales.

Es importante tener en cuenta que, en algunos casos, una única consulta compleja puede ser más lenta que varias consultas sencillas. Es crucial utilizar las herramientas de medición para evaluar el impacto de las optimizaciones y asegurarse de que realmente mejoran el rendimiento.



Actividad 1:

1. ¿Qué es la seguridad en bases de datos?
2. ¿Qué son las amenazas a la seguridad en bases de datos? Da 3 ejemplos.
3. ¿Qué son los controles de acceso en bases de datos?
4. ¿Qué es la optimización en SQL? Cita 3 ejemplos de técnicas de optimización.
5. ¿Qué son los índices en bases de datos?

Actividad Integradora de Cierre:

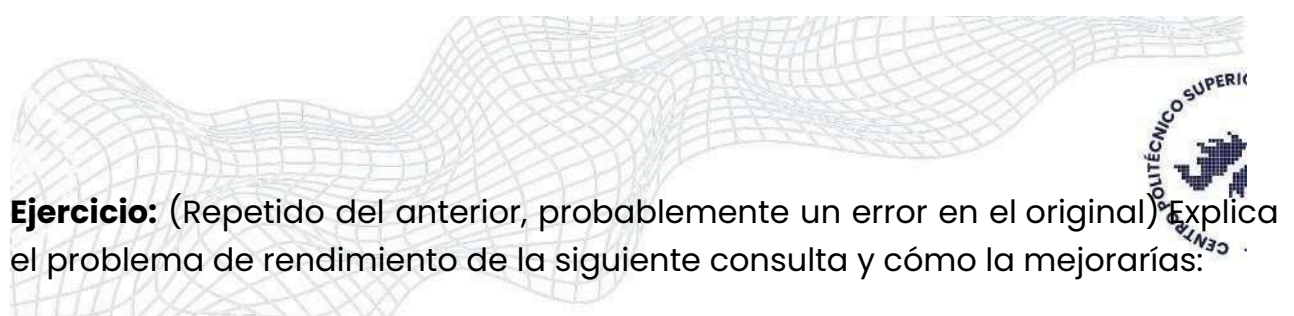
La tabla **clientes** puede tener las siguientes columnas:

- **id_cliente**: Identificador único del cliente.
- **nombre**: Nombre del cliente.
- **apellido**: Apellido del cliente.
- **direccion**: Dirección del cliente.
- **ciudad**: Ciudad donde reside el cliente.
- **estado**: Estado donde reside el cliente.
- **codigo_postal**: Código postal del cliente.
- **telefono**: Número de teléfono del cliente.
- **email**: Correo electrónico del cliente.

Ejercicio: Explica el problema de rendimiento de la siguiente consulta y cómo la mejorarías:

SQL
SELECT * FROM clientes;

1. (Pista: Considera el impacto de esta consulta si la tabla **clientes** contiene millones de registros).



Ejercicio: (Repetido del anterior, probablemente un error en el original) Explica el problema de rendimiento de la siguiente consulta y cómo la mejorarías:

SQL
SELECT * FROM clientes;

2. (Pista: Considera el impacto de esta consulta si la tabla **clientes** contiene millones de registros).

9. Cierre:

Al trabajar con SQL y bases de datos, es fundamental desarrollar habilidades y comprender conceptos clave, incluyendo:

- **Utilizar índices:** Los índices mejoran el rendimiento de las consultas al permitir búsquedas más rápidas y eficientes de los datos.
- **Limitar el número de filas devueltas:** Devolver un gran número de filas puede ralentizar las consultas. Utilizar cláusulas como **LIMIT** (en MySQL) mejora el rendimiento.
- **Evitar subconsultas innecesarias:** Las subconsultas pueden ser útiles pero también pueden ralentizar las consultas. Es importante asegurarse de que sean necesarias y se utilicen eficientemente.
- **Utilizar cláusulas **WHERE** y **JOIN** de forma efectiva:** Estas cláusulas mejoran el rendimiento.



Bibliografía Obligatoria:

- Marquez, M. (2011) : Base de datos. Editorial Universitat Jaume.
- Catheren M. R. (2009): Base de Datos. Edición McGraw Hill
- LINK DE APOYO: <https://codigofacilito.com/articulos/tips-rendimiento-sql>

