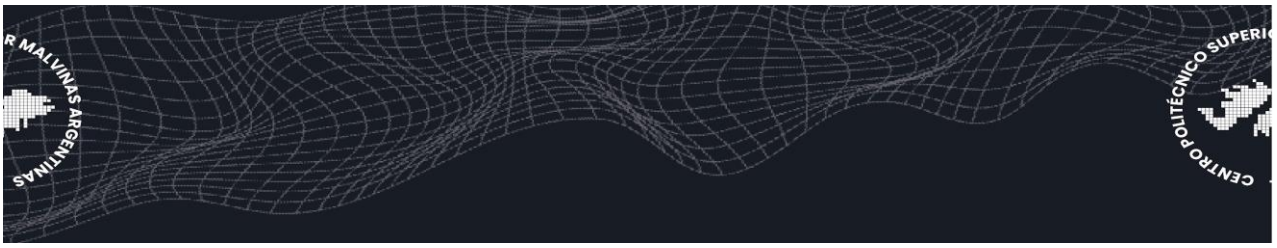


BASE DE DATOS

1° AÑO

Clase Nº 6: Optimización SQL

Contenido:Seguridad en base de datos,Optimizacion SQL y Indices SQL

En la clase de hoy trabajaremos los siguientes temas:

- Mejorar los tiempos de respuesta en un sistema de gestión de bases de datos relacional.
- Tipos de accesos a base de datos
- Controlar amenazas de base de datos
- Podrá identificar y crear Seguridad de base de datos
- Autentificación de diferentes Usuarios.
- Optimización de SQL
- Auditoria SQL

1. **Presentación:**

Bienvenidos a la clase N° 6 del espacio BASE DE DATOS

En esta clase sobre bases de datos debe ser, sin duda, La optimización de consultas en SQL es el proceso de mejorar el rendimiento de las consultas

en bases de datos. Esto se logra mediante la identificación y eliminación de cuellos de botella en las consultas, lo que permite que las consultas se ejecuten más rápido y de manera eficiente.

2. Desarrollo y Actividades:

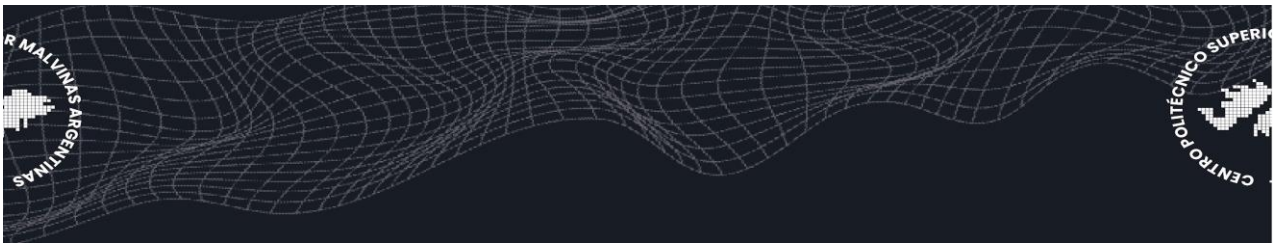
INTRODUCCIÓN A LA SEGURIDAD DE LAS BASES DE DATOS

Seguridad en las bases de datos significa proteger las bases de datos del acceso, modificación o destrucción no autorizados. Como la base de datos representa un recurso corporativo esencial, la seguridad es una meta importante. Además de la necesidad de preservar y proteger los datos para el funcionamiento continuo de la organización, los diseñadores de las bases de datos tienen la responsabilidad de proteger la privacidad de los individuos de quienes se guardan datos. Privacidad es el derecho que tienen los individuos de tener control sobre la información acerca de ellos. Muchos países tienen leyes diseñadas para proteger la privacidad, y toda organización que recabe y almacene información sobre las personas tiene la obligación legal de adoptar políticas que se ajusten a las leyes locales sobre la privacidad. El diseño de la base de datos debe reflejar el compromiso de la organización para proteger el derecho a la privacidad de los individuos, a través de la inclusión sólo de aquellos ítems que la organización tiene derecho a saber. Además, debe garantizarse la privacidad para proteger información almacenada cuya naturaleza sea delicada. Las amenazas a la seguridad ocurren en forma accidental o deliberada.

AMENAZAS ACCIDENTALES PARA LA SEGURIDAD

Los siguientes son algunos ejemplos de violaciones accidentales de la seguridad.

- El usuario solicita sin malicia un objeto u operación para los que no debería estar autorizado, y la solicitud se aprueba debido a un fallo en los procedimientos de autorización o porque hay un error en el sistema de administración de la base de datos o sistema operativo.
- Una persona envía por accidente un mensaje que debería enviarse a otro usuario, lo que resulta en el acceso no autorizado al contenido de la base de datos.



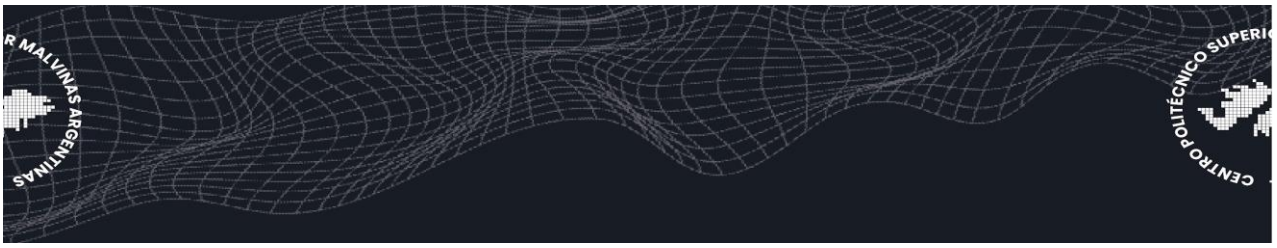
- Un error en el sistema de comunicaciones conecta a un usuario a una sesión que pertenece a otro que tiene diferentes privilegios de acceso.
- Por accidente, el sistema operativo sobrescribe, erróneamente, en archivos y destruye parte de la base de datos, busca en los archivos incorrectos, y luego los envía inadvertidamente al usuario, o no borra archivos que deben ser destruidos.

AMENAZAS DELIBERADAS PARA LA SEGURIDAD

Las violaciones deliberadas a la seguridad ocurren cuando un usuario logra de manera intencional acceso no autorizado a la base de datos. Un empleado disgustado que conozca el sistema de cómputo de la organización representa una amenaza enorme a la seguridad. Los espías industriales que buscan información para la competencia también amenazan la seguridad. Hay muchos modos de hacer violaciones deliberadas en la seguridad, entre las que están las siguientes:

Intervención de líneas de comunicación para interceptar mensajes hacia y desde la base de datos

- Escucha furtiva electrónica para obtener señales de las estaciones de trabajo, impresoras y otros dispositivos dentro de un edificio
- Lectura o copiado de pantallas e impresiones dejadas con descuido por usuarios autorizados
- Suplantación de un usuario autorizado, o con mayor acceso, por medio del uso de su registro y clave.
- Escritura de programas con código ilegal para esquivar el sistema de administración de la base de datos y su mecanismo de autorización, con el fin de acceder a los datos directamente desde el sistema operativo
- Escribir programas de aplicaciones con código que efectúa operaciones no autorizadas
- Obtención de información sobre datos ocultos, al consultarlos con inteligencia a la base de datos
- Retirar dispositivos de almacenamiento físico de los equipos de cómputo.



- Hacer copias físicas de archivos almacenados sin pasar por el sistema de administración de la base de datos, con lo que se esquivan sus mecanismos de seguridad
- Sobornar, amenazar o influir en los usuarios autorizados, a fin de utilizarlos como agentes para obtener información o dañar la base de datos.

SEGURIDAD FÍSICA Y AUTENTIFICACIÓN DEL USUARIO

La seguridad de las bases de datos se implementa mejor si sólo es una parte de un plan más amplio para controlar la seguridad. Este plan debe comenzar con medidas físicas para proteger el edificio, con precauciones especiales para las instalaciones de cómputo. El diseño de un edificio seguro en su planta física está claramente fuera del dominio del diseñador de la base de datos. Sin embargo, el ABD o administrador de base de datos debe tener la capacidad de sugerir medidas para controlar el acceso a las instalaciones de la base de datos. Es frecuente que éstas comiencen en la entrada principal, donde todos los empleados deben ser identificados visualmente por guardias, o con el uso de gafetes, huellas digitales, firmas u otros mecanismos. Para ingresar a las instalaciones de cómputo debe requerirse una identificación adicional. Las medidas físicas de la seguridad deben ampliarse para que cubran cualquier ubicación en la que datos fuera de línea, como los respaldos, también estén almacenados en forma segura.

Como la seguridad física de las estaciones de trabajo puede ser difícil de implementar, los controles de la seguridad de aquéllas requieren la autenticación de los usuarios. Autenticación significa la verificación de la identidad del usuario (hacer una comprobación para garantizar que el usuario real es quien dice ser). Por lo general se implementa al nivel del sistema operativo. Cuando el usuario se registra, él o ella ingresa una ID de usuario, cuya validez se revisa. El sistema tiene un perfil del usuario para esa ID, lo que da información sobre el usuario. El perfil normalmente incluye una clave (password) que se supone sólo conoce el usuario. Las claves deben conservarse en secreto y cambiadas con frecuencia. Una precaución elemental de seguridad es que el sistema requiere que las claves se cambien cada mes. Es obvio que el sistema nunca debe desplegar las claves al registrarse, y los perfiles almacenados deben mantenerse seguros, tal vez en forma encriptada. Aunque las claves son el método de autentificación que se usa con más amplitud, no son muy seguras, ya que los

usuarios en ocasiones las escriben, escogen palabras que son fáciles de adivinar o las comparten con otros usuarios. En ciertas organizaciones, los usuarios deben insertar tarjetas o llaves cuando se registran. En otras, se examina la voz, huellas digitales, retina, u otras características físicas del usuario. Algunas utilizan un procedimiento de autentificación más allá de una sola clave. Un procedimiento puede consistir en responder una serie de pregun- tas y requerir de más tiempo y ser más difícil de reproducir que una clave. Aunque la autenticación puede hacerse sólo a nivel del sistema operativo, es posible que se pida otra vez al nivel de la base de datos. Como mínimo, debe pedirse al usuario que proporcione una clave adicional para acceder a la base de datos**.**

AUTORIZACIÓN

Además de la autentificación, la mayor parte de los sistemas de administración de bases de datos para multiusuarios tiene sus propios subsistemas de seguridad, los cuales proveen la autorización de usuario, método por el que se asigna a los usuarios derechos para usar objetos de la base de datos. La mayoría de los sistemas multiusuarios tiene un lenguaje de autorización que forma parte del sublenguaje de datos. Por ejemplo, SQL provee comandos de autorización estándar para asegurar privilegios a los usuarios, como se ve en la sección
9.8. El ABD usa el lenguaje de autorización para especificar los derechos del usuario por medio de reglas de autorización, que son enunciados que especifican cuáles usuarios tienen acceso a cuál información, y cuáles operaciones les está permitido utilizar sobre qué datos. El mecanismo de autorización está diseñado para proteger la base de datos al impedir a las personas la lectura, actualización o eliminación no autorizada del contenido de la base de datos. Estas restricciones se agregan a los mecanismos de seguridad que provee el sistema operativo. Sin embargo, en un número sorprendentemente grande de casos, los subsistemas de seguridad de la base de datos son mínimos o no se utilizan en forma completa. El diseñador debe reconocer que los datos constituyen un recurso corporativo valioso, e incluir los mecanismos de seguridad disponibles como factor importante en la evaluación alternativa de sistemas de administración de bases de datos, así como desarrollar políticas de seguridad efectivas que utilicen los controles de que dispone el sistema elegido.

CONTROL DEL ACCESO

El control del acceso es el medio por el que se implementan las autorizaciones. Controlar el acceso significa asegurarse de que a los datos u otros recursos sólo se accede en las formas autorizadas. Al planear el acceso, el ABD puede usar una matriz de control del acceso a la base de datos, como se ilustra en la figura 9.1. Los encabezados de las columnas representan objetos de la base de datos, que pueden ser los nombres de las tablas, vistas, ítems de datos, objetos, módulos u otras categorías, en función del modelo de la base de datos y sistema de administración que se utilice. Las etiquetas de las filas representan individuos, roles, grupos de usuarios o aplicaciones.

El contenido de las celdas especifica el tipo de acceso permitido. Los valores de las entradas también dependen del sistema particular que se utilice, pero la elección por lo general incluye READ, INSERT, UPDATE, DELETE y sus combinaciones. Una vez que la matriz de control de acceso está completa, el ABD debe usar el lenguaje de autorización apropiado para implementarla. Por supuesto, al ABD le está permitido crear y cambiar la estructura de la base de datos y usar el lenguaje de autorización para garantizar o revocar el acceso a otros. Algunos sistemas permiten que el ABD delegue parte de su poder de autorización, en cuyo caso, ciertos usuarios tienen permiso de modificar las estructuras existentes de la base de datos o crear estructuras nuevas y actualizar el estado de los datos. En un ambiente multiusuario, tales cambios tienen consecuencias para los demás usuarios. Como el ABD es con frecuencia el único que tiene un punto de vista amplio de las necesidades de todos los usuarios, es frecuente que no sea conveniente dar esa clase de autorización. En ocasiones, el ABD da a ciertos usuarios el poder de autorizar a otros la ejecución de operaciones sobre la base de datos. Sin embargo, tener muchos de tales “autorizadores” es peligroso en extremo, pues los autorizadores generan a otros que también autorizan, lo que lleva a que la situación se salga de control muy rápido y que el ABD tenga dificultades para revocar las autorizaciones.

OBJECT						
SUBJECT	S t uden t table	S t uVi ew1	Wr apUp Procedure	Facu lt y table	En r o ll table	. . .
User U101	read, update	read	execute	read		. . .
User U102		read				. . .
Role Adviso r	read	read			read, insert, update, delete	. . .
. . .	. . .	. . .	. . .	. . .	. . .	. . .

USO DE LAS VISTAS PARA EL CONTROL DEL ACCESO

La vista es un método muy utilizado para implementar el control del acceso. El mecanismo de la vista tiene dos propósitos principales. Es una facilidad para el usuario, pues simplifica y personaliza el modelo externo con el que éste maneja la base de datos, lo que lo libera de las complejidades del modelo subyacente. También es una medida de seguridad, ya que oculta estructuras y datos que el usuario no debería ver. El modelo relacional es un modelo que consiste por completo en vistas o alguna combinación de tablas básicas y vistas relacionales. Una vista relacional se obtiene de tablas básicas por medio de la operación SELECT con el fin de obtener columnas o filas, o mediante el uso de otras operaciones para obtener datos calculados o materializados. Al especificar restricciones en la línea WHERE del enunciado SELECT que se usa para crear vistas, éstas se hacen dependientes del valor. La figura 9.2(a) da un ejemplo de vista creada a partir de la tabla S t ud e n t incluyendo sólo los datos de los estudiantes cuya especialidad es CSC. Las vistas independientes del valor se crean mediante la especificación de columnas de las tablas básicas y omitir la línea WHERE del enunciado SELECT. En la figura 9.2(b) se presenta un ejemplo de vista de la tabla S t ud e n t que sólo muestra las columnas s t u ID, s t uName y ma j o r .

REGISTROS DE SEGURIDAD Y PROCEDIMIENTOS DE AUDITORIA

Otra herramienta de seguridad importante es el registro de seguridad (bitácora), que es un diario que registra todos los intentos de violar la

seguridad. La violación puede sólo registrarse o enviar un mensaje inmediato al operador o al ABD. Saber de la existencia del registro es un disuasivo en sí mismo. Si el ABD sospecha que los datos están siendo atacados sin que se disparen los registros de la seguridad, es posible que ordene un procedimiento.

Vista dependiente del valor

```
CREATE VIEW CSCMAJ AS
  SELECT stuId, lastName, firstName,
  credits FROM Student
  WHERE major ='CSC';
```

Vista independiente del valor

```
CREATE VIEW StuView1 AS
  SELECT stuId, lastName, firstName,
  major FROM Student;
```

```
CREATE OR REPLACE TRIGGER
EnrollAuditTrail BEFORE UPDATE OF
    grade ON Enroll
FOR EACH ROW BEGIN
INSERT INTO EnrollAudit
VALUES(SYSDATE, USER, :OLD.stuId, :OLD.courseNo,
:OLD.grade, :NEW.grade);
END;
```

Auditoría. Este sistema de auditoría registra todos los accesos a la base de datos, mantiene información acerca del usuario que solicitó el acceso, la operación que ejecutó, la estación de trabajo desde la que lo hizo, la hora exacta en que ocurrió, el tipo de datos, su valor anterior y su valor actual, si lo hubiera. De esa manera, la auditoría tiene la capacidad de descubrir el origen de las operaciones sospechosas ejecutadas sobre la base de datos, aun si las hubieran realizado usuarios autorizados, por ejemplo empleados a disgusto. También es posible usar **disparadores** para que inicien un procedimiento de auditoría para una tabla, con el registro de todos los cambios, la hora en que se hicieron, y la identidad del usuario que los ejecutó. Por ejemplo, en Oracle, si se desea vigilar los cambios realizados a las calificaciones en la tabla En roll , primero se debe hacer una tabla que mantenga los registros de auditoría. Este esquema sería el siguiente para dicha tabla.

Encriptado

Como réplica a la posibilidad de que haya archivos a los que se acceda directamente desde el sistema operativo, o que sean robados, es posible guardar los datos de la base en forma encriptada. Sólo el sistema de gestión de bases de datos (DBMS) puede descifrarlos, de modo que cualquier persona que los obtenga por otros medios recibirá datos sin sentido. Cuando los usuarios autorizados acceden a la información de manera adecuada, el DBMS recupera los datos y los decodifica en forma automática. El encriptado también debe usarse siempre que los datos se envíen a otros sitios, a fin de que quien intervenga las líneas tam- bién los reciba en forma

cifrada. El encriptado requiere un sistema de encriptado, que con- siste en los componentes que se mencionan a continuación:

■ Un algoritmo de encriptado, que toma al texto normal (texto plano) como entrada, hace algunas operaciones con éste y produce como salida al texto en clave (texto encriptado)

■ Una clave de encriptado, que es parte de entrada para el algoritmo de encriptado, y se elige de un conjunto muy grande de claves posibles

■ Un algoritmo de desencriptado, que opera sobre el texto encriptado como entrada y produce al texto plano como salida

■ Una clave de desencriptado, que es parte de entrada para el algoritmo de encriptado y se escoge de un conjunto muy grande de posibles claves.

Un esquema muy usado para encriptar datos es el Data Encryption Standard (DES, Estándar de encriptado de datos), inventado por la National Bureau of Standards y adoptado en 1977. En el esquema DES, el algoritmo en sí es público, mientras que la clave es privada. Usa encriptado simétrico, en el que la clave de encriptado es la misma que la de desencriptado, y el algoritmo de desencriptado es el inverso del de encriptado. La figura 9.4 presenta la panorámica del proceso DES. Como el algoritmo es un estándar, es posible implementar el hardware para él en un solo chip, de modo que el encriptado y desencriptado sean muy rápidos y baratos, en comparación con la implementación en software del algoritmo. El algoritmo DES utiliza una clave de 56 bits en bloques de 64 bits sobre el texto plano, y produce bloques de 64 bits de texto encriptado. Cuando los datos se codifican, se separan en bloques de 64 bits. Dentro de cada bloque los caracteres se sustituyen y acomodan de acuerdo con el valor de la clave. El algoritmo de decodificación usa la misma clave para volver a los caracteres originales y restaurarlos en sus posiciones originales en cada bloque. Hay dos desafíos importantes con el sistema DES, la seguridad de la clave y la facilidad de violar el código. La clave debe mantenerse segura o el encriptado no servirá de nada, ya que cualquiera que poseyera la clave tendría acceso a los datos. Por tanto, la seguridad depende del secreto con que se guarde la clave, pero ésta debe comunicarse a todos los usuarios autorizados. Entre más personas la conozcan, más probable es que llegue a usuarios no autorizados. Asimismo, es frecuente que sea necesario distribuir la clave a los receptores de los

mensajes encriptados. Si se emplean líneas de telecomunicación, la transmisión de la clave permitirá a quienes intervengan las líneas acceder con facilidad a los mensajes encriptados. Con frecuencia se utilizan líneas más seguras para enviar la clave, o se hace por correo o mensajería. Aunque el estándar DES aún se usa con amplitud, no es muy seguro porque puede violarse en una cantidad de tiempo razonable. En el año 2000 se desarrolló y adoptó como estándar nuevo una versión mejorada llamada Advanced Encryption Standard (AES, Estándar de encriptado avanzado). Utiliza un esquema simétrico más sofisticado que el DES, y soporta tres tamaños posibles de clave: 128, 192 o 256 bits, lo que depende del nivel de seguridad necesario. Entre más grande sea el tamaño de la clave, más difícil es vio- lar el esquema.

Un segundo enfoque es el encriptado de clave pública, que utiliza parejas de números primos. En la figura 9.5 se presenta la panorámica del encriptado de clave pública. Se escoge para cada usuario una pareja de números primos

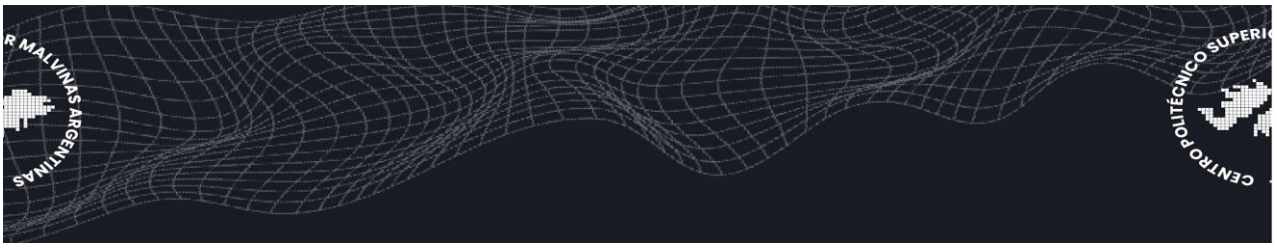
muy grandes (p, q), como clave privada del usuario, y el producto de la pareja p*q que se convierte en la clave pública de ese usuario. Las claves públicas se comparten con libertad, de modo que cualquier persona que desee enviar un mensaje al usuario encuentre su clave pública con facilidad. Después, la clave pública se emplea como entrada de un algoritmo de encriptado, que produce el texto encriptado para ese usuario. Cuando éste recibe un mensaje encriptado debe producir los factores primos de la clave pública para decodificarlo. Como no existe un método fácil o rápido de encontrar los factores primos de un número grande, es difícil en extremo que un intruso los obtenga. Sin embargo, un intruso con la determinación de violar la clave lo hará, siempre y cuando esté decidido a dedicar recursos sustanciales a la tarea. Este método sólo es tan seguro como la clave privada, por lo que debe darse a los usuarios sus claves privadas en alguna forma segura, y deben protegerse éstas contra su divulgación.

Un método bien conocido de encriptado de clave pública es el RSA, llamado

así en honor a sus desarrolladores, Rivest, Shamir y Adleman.

Panorámica del encriptado de clave pública

OPTIMIZACION DE SQL



Las malas prácticas en la formulación de tu consulta pueden traducirse en consultas lentas cuyo rendimiento no hace sino empeorar conforme la cantidad de datos que almacenas crece, por poner un ejemplo, lo que era una consulta de 200 ms, se convierte en una de 4 seg cuando la tabla registra cientos de miles de datos.

En este artículo voy a darte algunos tips que pueden ayudarte a reducir el tiempo de ejecución para tus consultas SQL, sobre todo si la escala de tu aplicación está en los cientos de miles o millones de registros en las tablas de la base de datos.

Muchas personas asumen que una consulta lenta es una señal de que es necesario introducir tecnologías adicionales a SQL, especialmente aquellas enfocadas en el manejo de Big Data. La realidad es que son pocos y muy específicos los casos en que la información supera las capacidades de un buen motor de base de datos, en la gran mayoría de los casos es más práctico y de mayor beneficioso solucionar el problema en SQL, revisar la estructura de tu información, entre otras cosas que cubriremos a continuación.

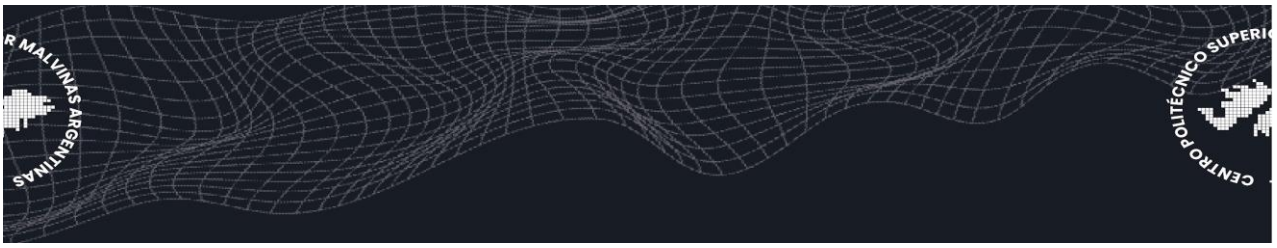
MEDICIÓN

En el trabajo de optimización de rendimiento no hay nada más importante que la medición de los resultados, medir es el primer paso para mejorar el rendimiento de cualquier cosa. Sin medición no podemos decir si algún cambio produjo una mejora, o al contrario, empeoró los resultados; más importante aún, sin medición no podemos identificar los problemas antes de que se presenten frente a nuestros usuarios.

LA REGLA 80/20

Nate Berkopec, en su libro The Complete Guide To Rails Performance hace mención del principio de Pareto, 80% of the output will come from 20% of the input, y cómo este mismo principio aplica al código, asumiendo que 80% de los beneficios de rendimiento están en optimizar una pequeña porción de todo nuestro código.

Esto quiere decir que en lugar de tratar de optimizar cada línea de código que nos topamos, primero debemos identificar los casos más graves de rendimiento y los que al modificarse o corregirse van a dar cuenta del mayor beneficio para la aplicación en general en velocidad y performance.



Sin medición, es imposible encontrar estos pequeños cambios que generarán el 80% de las mejoras, lo que puede significar que terminemos enfrascados en trabajo de micro optimizaciones, sin atacar los problemas más grandes de una aplicación, o en este caso, de una consulta.

MEDICIONES EN EL GESTOR

Medir el tiempo de ejecución de una consulta dependerá del gestor que uses, en mysql el tiempo de ejecución de una consulta aparece justo después de los resultados si usas la terminal. Adicionalmente puedes medir la duración de una consulta de la siguiente manera:

Cambia el valor de la variable profilling, para que las consultas registren su tiempo de ejecución

mysql> set profiling=1;

Ejecuta la consulta

mysql> SELECT * from tabla gigante;

Revisa el rendimiento con show profiles;

mysql> show profiles;

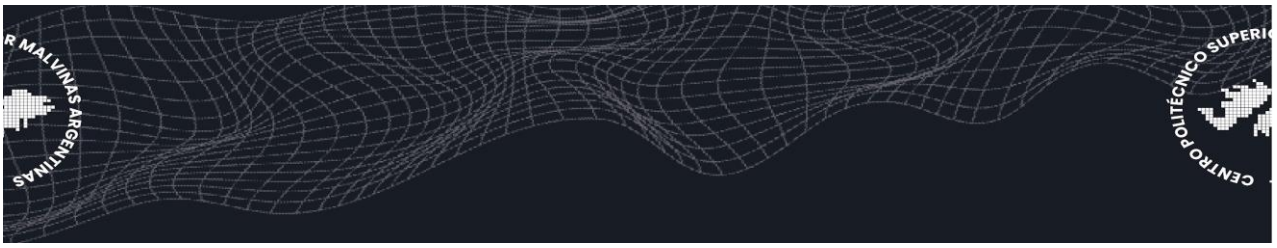
Mediciones en la aplicación

En muchas ocasiones, tus consultas SQL serán generadas por una librería en tu aplicación, si usas Rails será el ActiveRecord, en Laravel Eloquent, etc.

En estos casos, será tu elección de librerías la que te ayude a definir las herramientas para medir tus consultas, por ejemplo: el ActiveRecord de Rails imprime en log el tiempo de ejecución de tu consulta en modo de desarrollo.

Lo importante es configurar tu librería de conexión a la base de datos para que esta te pueda informar del tiempo de ejecución de tus consultas.

HERRAMIENTAS INDEPENDIENTES DE TU STACK



Así como cada framework y herramienta tiene librerías que te ayudarán a registrar tus mediciones, existen herramientas en la nube, que son agnósticas de la tecnología que uses, y que puedes configurar en tu aplicación para registrar información de tu base de datos.

Una de ellas, es Newrelic, una solución basada en la nube que configuras en cualquier framework o lenguaje: Rails, Django, Laravel, Node.js, Go, entre otros.

Newrelic ofrece una versión gratuita que te ayuda a visualizar qué tipo de consultas consumen más tiempo o se ejecutan más frecuentemente en tu aplicación. Además, puedes visualizar cuáles de tus rutas están demorando más en responder, para que puedas revisarlas en detalle.

TIPOS DE OPTIMIZACIÓN

El primer tip de optimización de una base de datos SQL, es invertir el tiempo en aprender SQL a detalle. En muchas ocasiones, un mismo resultado puede obtenerse con distintos tipos de consultas, en ocasiones, 3 consultas se pueden ejecutar en una sola.

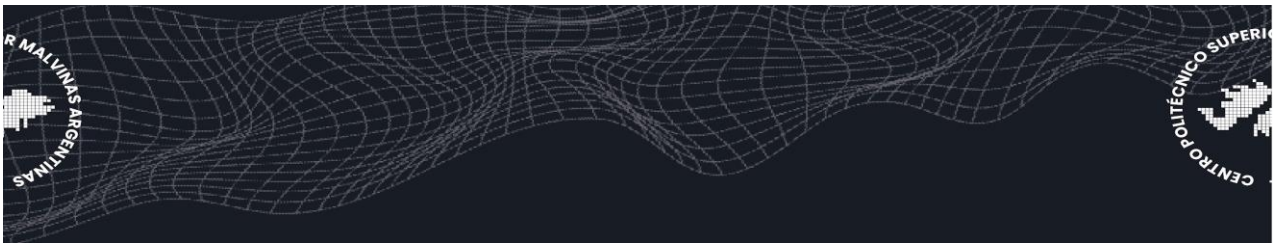
Para que tú puedas analizar y concluir cuál es la mejor forma de obtener un resultado complejo, debes de conocer las distintas posibilidades y el impacto que tiene cada una de ellas. En ese sentido, te recomiendo ampliamente el Curso Profesional de Bases de Datos SQL, impartido aquí en CódigoFacilito.

Dicho esto, a continuación, una serie de revisiones generales que puedes aplicar para muchos tipos de problemas de rendimiento en la ejecución de una consulta.

ÍNDICES

Los motores de bases de datos SQL pueden agregar índices a ciertas columnas que aceleran las queries.

Este proceso se actualiza en inserción por lo que hay un trade off al agregar un index, inserciones ligeramente más lentas y consultas más rápidas.



El uso correcto de índices puede acelerar consultas en tablas con muchos registros, pero sobre todo, el mal uso de ellos puedes traducirse en consultas muy lentas que afectan el rendimiento de la página

Una de las principales por las que una consulta se ejecuta muy lento, es que hagas operaciones sobre un campo que no tiene un index. En algunos casos observados en nuestra plataforma, agregar un índice a un campo usado para ordenamiento o filtrado de la información, ha resultado en consultas que toman la mitad del tiempo.

En otros escenarios los indexes están pero no se están usando, un claro ejemplo es el campo id que Rails usa para identificar de manera única cada registro en una tabla, por defecto este campo se indexa en la base de datos.

El campo id, además de identificar nuestros registros, también suele usarse para saber en qué orden fueron creados. Alternativamente, el campo created at que guarda la fecha de creación también nos puede ayudar a ordenar los elementos según fueron agregados a la tabla.

Considerando una tabla con aprox unos 3 millones de registros, estas dos consultas entregan el mismo resultado, este es un ejemplo visto en nuestra plataforma:

SELECT * from test ORDER BY created_at desc LIMIT 100;

SELECT * from test ORDER BY id desc LIMIT 100;

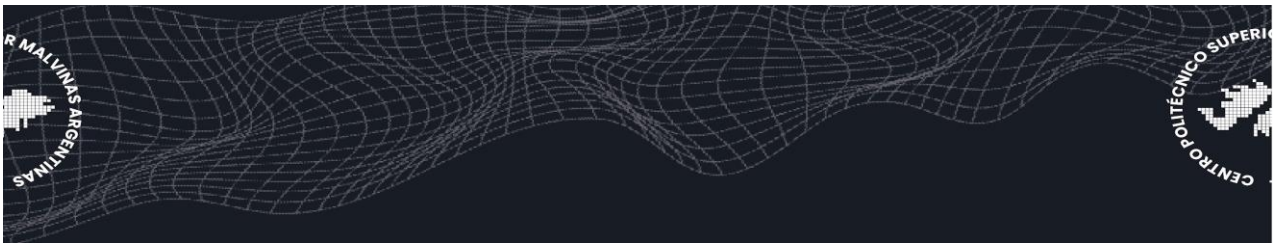
La única diferencia es que una tarda 0s y la otra casi 3s,

Al filtrar u ordenar, procura siempre hacerlo con el campo que tiene un index, y si no tiene, considera agregarlo.

Adicionalmente, algunos desarrolladores recomiendan mantener en revisión los indexes de una tabla para que sean eliminados en caso de que ya no sean utilizados, para acelerar las operaciones de inserción en la base de datos. Recuerda que el uso de índices no es gratuito, tiene una repercusión en el tiempo de inserción, por lo que eliminar los índices no usados, puede ser una buena idea.

EL PROBLEMA DE N+1 CONSULTAS

El problema de N+1 queries describe los escenarios en los que por cada elemento de una lista con la que trabajamos, como en el ejemplo la lista de



En los cursos se introduce un query nuevo, es solo una referencia, ya que en el mundo real es fácil introducir problemas de N+2, N+3 etc. donde cada elemento en la lista introduce múltiples queries.

En los escenarios donde las consultas son escritas manualmente, es prácticamente imposible encontrar uno de estos casos, normalmente los encontramos como resultado del mal uso de una librería para el manejo de la base de datos en nuestro código.

Para solucionar estos escenarios, la recomendación general es;hacer una carga anticipada de los registros usando alguno de los distintos mecanismos que nuestras librerías de base de datos ofrecen. Esto usualmente se traduce en el uso correcto de operaciones JOIN en nuestras consultas.

En este escenario, no importa si tus consultas tardan 10ms si estás ejecutando 100 de ellas. Reducir la cantidad de consultas que requiere obtener un resultado, también es optimización.

¡Importante! En algunos escenarios, una consulta individual que trae consigo todos los datos necesarios, puede ser más lenta que muchas consultas pequeñas, en algunos escenarios que hemos observado aquí, hemos reducido la cantidad de consultas de una operación de cientos a un par, solo para ver cómo las nuevas consultas, aunque menos, tardan muchísimo en ejecutarse, de nuevo, es importante tener a la mano tus herramientas de medición, para no hacer modificaciones engañosas.

Actividad

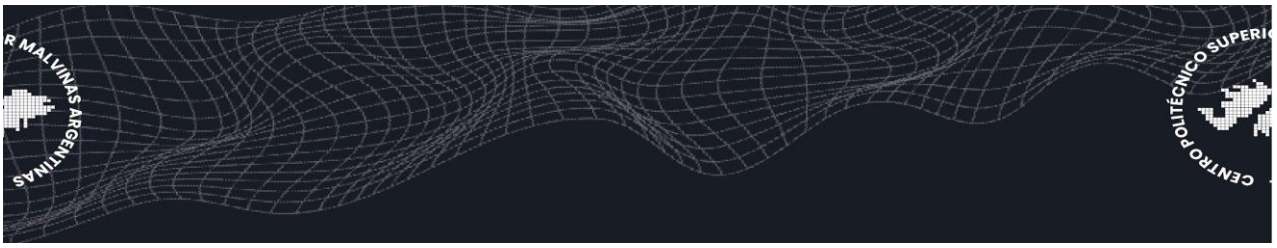
1.-¿Que es Seguridad en base de datos?

2.-¿Que son amenazas de Seguridad en base de datos?, dar 3 ejemplos

3.- ¿Qué son los controles de accesos en base de datos?

5.- ¿Qué es optimización en SQL? , citar 3 ejemplos

6.-¿Qué es Índices en base de datos?



3.-Actividad Integradora de Cierre:

La tabla `clientes` puede tener diferentes columnas, pero a continuación se presentan algunas posibles columnas que puede tener la tabla `clientes`:

- `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.

1.-Ejercicio: Utilizar la cláusula WHERE de manera efectiva para limitar el número de filas devueltas.

Consulta sin cláusula WHERE:

```
SELECT * FROM clientes;
```

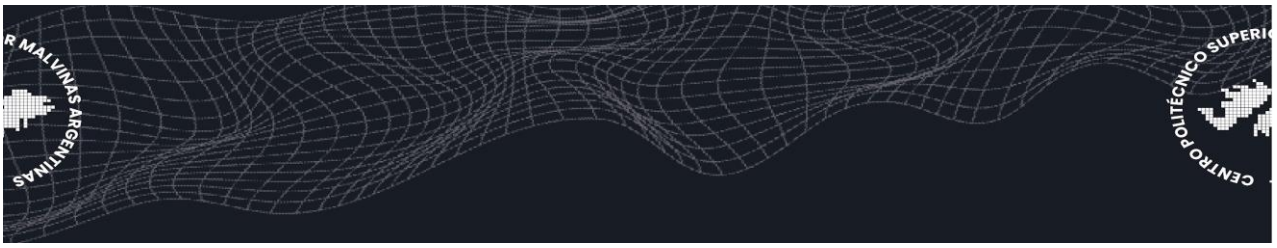
2.-Ejercicio: Utilizar la cláusula WHERE de manera efectiva para limitar el número de filas devueltas.

Consulta sin cláusula WHERE:

```
SELECT * FROM clientes;
```

4.-Cierre:

Al utilizar SQL en una base de datos, se pueden aprender diversas habilidades y conceptos importantes, entre ellos:



1. Utilizar índices: Los índices pueden mejorar significativamente el rendimiento de una consulta al permitir que se realice una búsqueda más rápida y eficiente de los datos en la base de datos.

2. Limitar el número de filas devueltas: Si una consulta devuelve un gran número de filas, esto puede ralentizar el rendimiento de la consulta. Limitar el número de filas devueltas (por ejemplo, con la cláusula LIMIT en MySQL) puede mejorar el rendimiento.

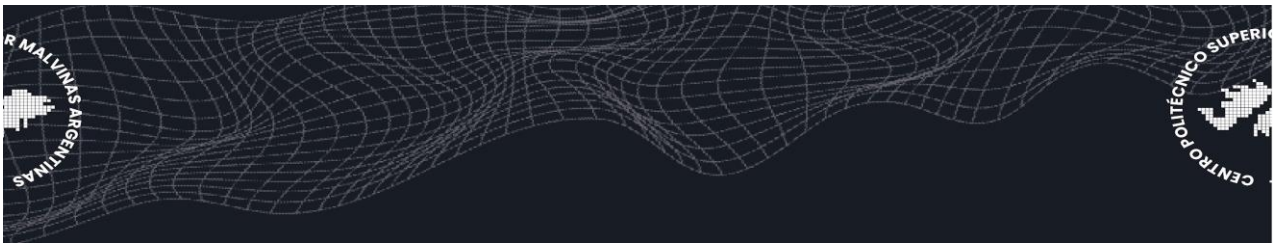
3. Evitar subconsultas innecesarias: Las subconsultas pueden ser útiles para realizar operaciones avanzadas con datos, pero también pueden ralentizar el rendimiento de una consulta. Es importante asegurarse de que las subconsultas son necesarias y se están utilizando de la manera más eficiente posible.

4. Utilizar cláusulas WHERE y JOIN de manera efectiva: Las cláusulas WHERE y JOIN pueden mejorar el rendimiento de una consulta al limitar el número de filas que se deben buscar. Es importante utilizar estas cláusulas de manera efectiva para garantizar que la consulta sea lo más eficiente posible.

5. Optimizar la estructura de la base de datos: La estructura de la base de datos puede tener un impacto significativo en el rendimiento de las consultas. Es importante asegurarse de que la base de datos esté estructurada de la manera más eficiente posible para que las consultas se ejecuten rápidamente.

Tomamos un momento para repensar lo que hemos aprendido, En esta clase

La optimización de consultas en SQL puede mejorar significativamente el rendimiento de las consultas en bases de datos. Al utilizar técnicas como el



6. Utilizar índices: Los índices pueden mejorar significativamente el rendimiento de una consulta al permitir que se realice una búsqueda más rápida y eficiente de los datos en la base de datos.

7. Limitar el número de filas devueltas: Si una consulta devuelve un gran número de filas, esto puede ralentizar el rendimiento de la consulta. Limitar el número de filas devueltas (por ejemplo, con la cláusula LIMIT en MySQL) puede mejorar el rendimiento.

8. Evitar subconsultas innecesarias: Las subconsultas pueden ser útiles para realizar operaciones avanzadas con datos, pero también pueden ralentizar el rendimiento de una consulta. Es importante asegurarse de que las subconsultas son necesarias y se están utilizando de la manera más eficiente posible.

9. Utilizar cláusulas WHERE y JOIN de manera efectiva: Las cláusulas WHERE y JOIN pueden mejorar el rendimiento de una consulta al limitar el número de filas que se deben buscar. Es importante utilizar estas cláusulas de manera efectiva para garantizar que la consulta sea lo más eficiente posible.

10. Optimizar la estructura de la base de datos: La estructura de la base de datos puede tener un impacto significativo en el rendimiento de las consultas. Es importante asegurarse de que la base de datos esté estructurada de la manera más eficiente posible para que las consultas se ejecuten rápidamente.

Tomamos un momento para repensar lo que hemos aprendido, En esta clase

La optimización de consultas en SQL puede mejorar significativamente el rendimiento de las consultas en bases de datos. Al utilizar técnicas como el

uso de índices, limitar el número de filas devueltas y evitar subconsultas innecesarias, se puede mejorar el rendimiento de las consultas y garantizar que la base de datos funcione de manera eficiente.

3. 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