Características de las bases de datos #1

Hendrik López Dueñas

Mateo Flórez Bacca

Bases de datos I

Universidad Industrial de Santander

2018

**Contenido**

Introducción.

1. Seguridad.
2. Identificar sensibilidad.
3. Evaluar vulnerabilidad.
4. Endurecer.
5. Auditar.
6. Monitorear.
7. Pistas de auditoría.
8. Controlar Acceso.

II. Integridad.

1. De entidad.
2. De dominio.
3. De referencia.

III. Respaldo.

IV. Recuperación.

1. Recuperación simple.
2. Recuperación completa.
3. Recuperación de registros masivos.

Conclusiones.

Bibliografía

**Introducción**

El uso de las bases de datos es de vital importancia en el mundo contemporáneo, debido a que permiten almacenar la información de una manera ordenada y de fácil manejo para quienes tienen la autorización de administrarlas. Sin embargo, existen individuos y/u organizaciones que buscan constantemente acceder a esta información de manera ilícita. Es por eso que las bases de datos deben poseer ciertas características que les permita almacenar de manera adecuada y segura toda aquella data que se desee guardar. A continuación, se va a explicar cuatro características que toda base de datos debe poseer a la hora de manejar su información.

**Seguridad**

Hablar del concepto de seguridad en el contexto de las bases de datos resulta de suma importancia, ya que mucha de la información que se almacena en ellas suele ser vital para quien la administre. Ejemplos pueden ser las bases de datos de las entidades bancarias, que poseen los registros de sus clientes (números de cuenta, dinero almacenado, entre otros datos personales); dicho banco no puede administrarle esa información a cualquier individuo, ya que esto se puede prestar para realizar actividades ilegales, las cuales pueden comprometer al banco.

Es por eso que una base de datos debe cumplir con ciertos protocolos de seguridad, con el propósito de proteger la información que reside dentro. Se pueden considerar siete los principales protocolos de seguridad:

* Identificar sensibilidad: Las bases de datos pueden llegar a tener tablas que son sensibles a todo tipo de intrusiones no autorizadas, debido al contenido; ya sea por diversos tipos de malware hasta las inyecciones SQL. Se deben identificar qué tablas poseen mayor sensibilidad a estos ataques.
* Evaluación de la vulnerabilidad: Entendiendo qué tipo de agentes informáticos pueden acceder de manera ilegal a la base de datos, hay que examinar qué factores pueden permitir el acceso de ese tipo de software. Va desde analizar la estructura de la base de datos, hasta el sistema operativo con el que trabaja.
* Endurecer: Sabiendo los puntos vulnerables de la base de datos, en necesaria la intervención de los administradores para reparar o añadir algún sistema de seguridad que proteja la información. Existen diversas maneras de endurecer la seguridad de una base de datos; se puede encriptar la información, usar cortafuegos (como GreenSQL) o incluso mover la base de datos a un sitio físico diferente (cambiar de computadora).
* Auditar: Luego de que se haya hecho algún cambio en la configuración de la base de datos, se deben hacer auto-evaluaciones que reflejen que los cambios hechos afectan de manera positiva a la seguridad.
* Monitorear: Los administradores de la base de datos deben mantener vigilada en tiempo real las acciones que se realizan. Es gracias a esto que, al detectar comportamientos inusuales en el historial de movimientos de la base de datos, se pueden detectar las inyecciones SQL y el traspaso no autorizado de datos, ya sea por medio de algún malware o por algún administrador malintencionado que da excesiva autorización de acceso a terceros.
* Pistas de auditoría: Al aplicar pistas de auditoría, se puede tener un historial de los movimientos que pueden afectar la integridad de los datos (tablas incoherentes, datos no válidos, etc…) y también permite detectar datos sensibles al malware y a las inyecciones SQL.
* Controlar Acceso: En una base de datos, no todos los usuarios tienen el mismo nivel de acceso a la información. Es por eso que se deben mantener vigilados los privilegios que cada usuario posea, para detectar si hace algún movimiento no autorizado. También se puede cifrar o encriptar la información para que los individuos no autorizados no puedan acceder con facilidad a los valores que son almacenados.

**Integridad**

Se habla de integridad a la correcta relación y exactitud que tienen las tablas que conforman cualquier base de datos. Dicha integridad puede verse afectada de manera negativa si la seguridad de la base de datos no es del todo óptima, o si los administradores modificaron de manera errónea la información contenida. Existen varios comandos que permiten alterar la información de las bases de datos, pero depende del administrador si esto puede llegar a afectar o no a la integridad. Algunos comandos son INSERT, DELETE, UPDATE, entre otros. La integridad se puede ver dañada si existen tablas con datos incoherentes entre sí ó tipos de dato no válidos.

Además, existen ciertos tipos de integridad que toda base de datos debe cumplir para que sus datos no se vean comprometidos:

* De entidad: En cada tabla, las filas deben actuar como entidades únicas, además de que se debe exigir la integridad de las columnas (las columnas deben manejar un mismo tipo de dato) y la existencia de una clave principal. En el siguiente ejemplo, se mostrará una tabla que no cumple con este requisito.

|  |  |
| --- | --- |
| **Nombre del propietario** | **Edad** |
| Luis Correa | 54 |
| Alejandra Ruiz | Dieciocho |
| Ignacio Díaz | 34 |

En este ejemplo, la segunda columna carece de integridad, ya que define la edad con dos tipos de datos; esto puede generar errores a la larga, por lo que no posee integridad de entidad. Además, la tabla carece de alguna contraseña con la cual se pueda relacionar al nombre del propietario (ID, CC, etc…).

* De dominio: En cada columna de las tablas, la integridad de los datos debe estar asegurado mediante la restricción del tipo de dato. Existe el comando CHECK, que permite corroborar esta integridad. A continuación se mostrará un ejemplo que no cumple con el siguiente requisito:

|  |  |  |
| --- | --- | --- |
| **Documento** | **Tipo de Sangre** | **RH** |
| 1005773482 | AB | Negativo |
| 1057285430 | B | Positivo |
| 1001827432 | 0 | Positivo |

En la segunda columna, existe un valor numérico en lo que se supone son cadenas de texto. Se colocó un cero en lugar de una O para referirse al tipo de sangre *O+*. Debido a esto, no se considera que la tabla tenga integridad de dominio.

* De referencia: Las tablas que se relacionan entre sí deben tener coherencia en la forma en la que están relacionadas. Además, a la hora de eliminar y/o modificar filas, la coherencia debe seguir intacta. En el siguiente ejemplo, se mostrarán dos tablas relacionadas entre sí, pero que carecen de este tipo de integridad:

|  |  |  |
| --- | --- | --- |
| **Código comprador** | **Producto comprado** | **Unidades compradas** |
| 47239576 | 132C | 2 |
| 87387384 | 854R | 1 |
| 29458384 | 190C | 1 |

|  |  |  |
| --- | --- | --- |
| **Código producto** | **Nombre producto** | **Unidades en Stock** |
| 132C | Papas BBQ | 15 |
| 190C | Lomo de Cerdo (500g) | 10 |

En este par de tablas se observa que un comprador solicitó el producto con código 854R. Sin embargo, dicho producto no existe en la base de datos de la tienda, por lo que no existe total coherencia en las tablas relacionadas. Por ende, carece de integridad de referencia.

**Respaldo**

En el contexto de bases de datos, una copia de seguridad o respaldo es una copia que contiene información importante de la fuente original. Esta copia puede ser completa o parcial, y existen diferentes estrategias para decidir a qué información vale la pena generarle un duplicado. El propósito de una copia de seguridad es recuperar información en caso de una pérdida no deseada de datos. Los causantes de este problema pueden ser de diversa índole, desde accidentes por parte de los usuarios, hasta ataques malintencionados contra la base de datos.

Teniendo en cuenta que los servidores que almacenan una base de datos cualquiera están localizados necesariamente en una ubicación real, tangible, cabe la posibilidad de que ciertas alteraciones físicas comprometan la funcionalidad de la base de datos. Por ejemplo, un mal manejo del sistema de refrigeración podría resultar en daños al servidor por culpa de altas temperaturas; un pico anormal de voltaje podría resultar en cables quemados; una catástrofe natural, como un terremoto o un tornado podrían destruir completamente las instalaciones físicas de la base de datos. Es por esto que se recomienda generar múltiples respaldos en diferentes ubicaciones, distantes entre sí.

**Recuperación**

Llegados al punto en el que, lamentablemente, ocurre una pérdida de información valiosa, es necesario poner en marcha los mecanismos de recuperación de datos. Dependiendo del tipo de respaldo que se haya generado la recuperación se efectuará de una u otra manera. SQL Server ofrece, por ejemplo, tres modelos de recuperación diferentes:

* Recuperación simple: El modelo de recuperación simple consiste en una copia de los registros de la base de datos. Este respaldo es generado manualmente y carece de opciones de automatización o respaldo del historial de cambios (log). Si no se mantiene actualizado, se podrían perder cambios importantes.
* Recuperación completa: El modelo de recuperación completa es el más usado generalmente. Consiste de una copia de los registros y del historial de cambios de la base de datos. Este respaldo puede ser generado automáticamente, y permite elegir una recuperación de una fecha específica, por lo que el riesgo de pérdida de información es mínimo. No obstante, una mala implementación de este modelo podría terminar en datos incoherentes tras una recuperación.
* Recuperación de registros masivos: Este modelo de recuperación es aplicado en una situación específica: cuando es necesario realizar cambios masivos en los registros, este respaldo puede ser generado rápida y eficazmente, de modo que no afecta el proceso de transacción de datos en masa y, en caso de que este último tenga algún inconveniente, el respaldo ya fue generado con éxito.

**Conclusiones**

Las bases de datos requieren que sus administradores mantengan a salvo la información que se desea almacenar, pues existen muchas formas en las que la data puede verse comprometida. Es por eso que, al final, muchos de los esfuerzos que realizan los administradores de las bases de datos se concentran en mantenerla segura de ataques por parte de terceros, los cuales buscan robar y/o eliminar la información que se almacena ahí. Sin embargo, también es menester mantener copias en caso de que la información se pierda de manera accidental.

**Bibliografía**

* <https://revista.seguridad.unam.mx/numero-12/principios-basicos-de-seguridad-en-bases-de-datos>
* <https://github.com/larskanis/greensql-fw>
* <http://php.net/manual/es/security.database.sql-injection.php>
* <https://es.slideshare.net/guby31/integridad-en-las-bases-de-datos>
* <https://www.youtube.com/watch?v=7nH7sPUg2BQ>
* https://docs.microsoft.com/es-es/sql/relational-databases/backup-restore/back-up-and-restore-of-sql-server-databases?view=sql-server-2017