Cómo auditar cambios en una tabla MySQL o MariaDB

Domínguez Chávez, Jorge
Universidad Politécnica Territorial del estado Aragua
jodocha@upta.edu.ve

RESUMEN: En este trabajo, presentamos una propuesta interesante y practica para auditar una tabla en particular a partir de tres (3) campos básicos. Puede agregar dos campos adicionales, usuario e ip, para una auditoria completa. Los tres (3) o cinco (5) campos deberían estar en aquellas tablas que cambian continuamente o en todas las tablas de una base de datos. También, tenga en cuenta que la ejecución de un trigger y el procesamiento de los datos de auditoría sobrecarga el procesador y a la memoria, lo cual hace que el tiempo de respuesta de las transacciones disminuya.

Descriptores: Auditoría, base de datos, trigger, fecha, hora.

ABSTRACT: In this paper, we present an interesting proposal to audit a particular table from three (3) basic fields. Also, You can add two additional users and IP fields, for a full audit. The three (3) or five (5) fields in the tables should be continually changing or all tables in a database. Also, note that the execution of a trigger and the processing of the data overloads processor and memory, which makes the response time of transactions decreased.

Keywords: Audit, database, trigger, date, hour.

I. INTRODUCCIÓN

Como requisito en las operaciones de una base de datos es necesario realizar una auditoria de algunas o todas de las tablas de la base de datos; es decir, ver cambios que hacen, quien lo hizo, a que hora, el valor anterior y el valor nuevo.

Los eventos que modifican los datos de una tabla son las sentencias INSERT, DELETE, UPDATE. Sabemos que los trigger se pueden ejecutar antes (BEFORE) y/o después (AFTER) de que sean modificados los datos de un registro.

En esta primera entrega, trabajamos con tres (3) campos básicos en cada tabla a auditar.

II. METODOLOGÍA EMPLEADA

A. Fase de revisión

Los eventos que modifican los datos de una tabla son las sentencias INSERT, DELETE, UPDATE. Sabemos que los trigger se pueden ejecutar antes (BEFORE) y/o después (AFTER) de que sean modificados los datos de un registro.

En esta primera entrega, trabajamos con tres (3) campos básicos en cada tabla a auditar. El primer campo, Id, de tipo integer, largo 11, auto_increment y primary key. El segundo campo es creado, de tipo timestamp, valor por defecto current_timestamp y el tercer campo modificado es tipo timestamp, valor por defecto current_timestamp y extra on update current_timestamp. Ver ejemplo siguiente:

```
drop table p;

create table p(id int(11) auto_increment primary key, nombre varchar(32), creado timestamp default current_timestamp, modificado timestamp default current_timestamp on update current_timestamp) engine=innodb auto_increment=1;
```

El primer campo al ser auto_increment, lleva una secuencia de los registros ingresados en la tabla. Es un evento controlado por la sentencia INSERT y DELETE, cada vez que haga un inserción se incrementa su valor o un borrado en la tabla se crea un hueco y NO incrementa su valor. Ver la siguiente sentencia:

```
insert into `p` (nombre) values ('jaquelin');

insert into `p` (nombre) values ('carmen');

insert into `p` (nombre) values ('jorge');

insert into `p` (nombre) values ('antonio');
```

vea la figura 1.

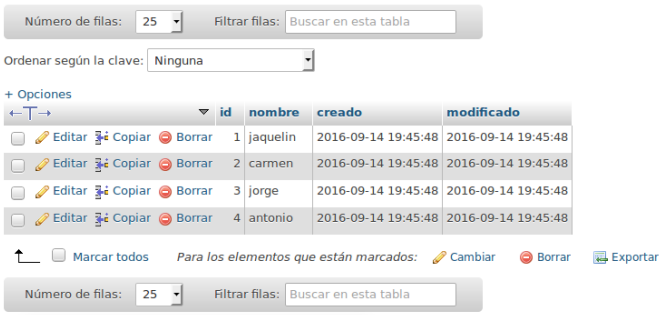

Figura 1: Listado de los primeros registros insertados en la tabla P.

Notar que el campo Id muestra valores correlativos. Los campos creado y modificado muestran la fecha y hora de la inserción; tiene valores iguales.

Ahora, al revisar la tabla encontramos huecos en la secuencia debemos investigar qué paso. Usemos la sentencia:

delete from p where id=2;

Si tenemos la tabla P con 4 registros, correlativo del 1 al 4...y borramos un registro, obtenemos:

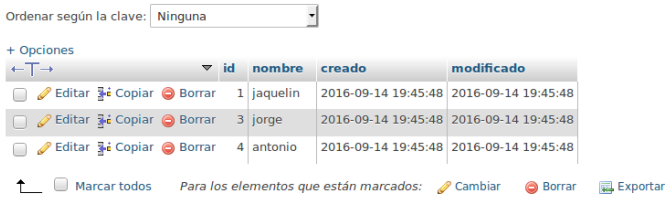

Figura 2: Listado de tabla P con el registro Id 2, eliminado.

e insertamos un nuevo registro con la sentencia:

insert into `p` (nombre) values ('david');

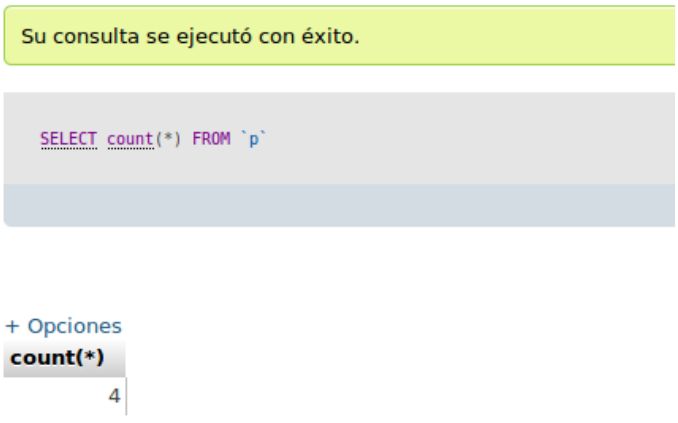

Figura 3: Resultado de SELECT count(*) FROM `p`.

Al ejecutar la sentencia select count(*) from tabla, seguimos teniendo los mismos 4 registros.

Al ejecutar la sentencia:

delete from p where id=2;

Obtenemos un (1) hueco, en el registro 2; y un nuevo registro con Id 5.

Figura 4: Hueco dejado por el borrado del registro 2 de la tabla.

Sin embargo, los campos creado y modificado continúan mostrando el mismo contenido. Ahora, si actualizamos el registro 4 con la sentencia:

update `p` set nombre='guadalupe' WHERE id=4;

Figura 5: Listado de tabla P, luego de modificar registro 4.

Notar que en el registro 4, el campo modificado muestra la fecha y hora de la modificación sobre el registro. El evento UPDATE controla el campo modificado, de ahí el cambio de valor.

Con esta información, hacemos reportes de una manera sencilla, tales como:

- Qué tablas han sufridos cambios recientemente.
- Qué tablas no sufrieron cambios el pasado año.
- Qué tablas NO han sufrido cambios.
- Mostrar los cambios a las tablas por un periodo especifico.

- Mostrar las tablas más activas en un determinado periodo.

Con estas herramientas es posible analizar el desempeño y actividad sobre la tablas, volver al estado anterior de la tabla y/o en un punto especifico.

Para la siguiente entrega, escribiremos un trigger para auditar los cambios (Inserciones, Actualizaciones y eliminaciones) a una tabla. Recordar que será un trigger asociado a una tabla en específico. Hay que tener en cuenta tres cosas en un trigger:

1. Se debe crear una tabla de log para almacenar la información de auditoría.
2. En el script hay que cambiar el nombre de la tabla sobre la cual se esta creando el trigger.
3. Sólo trabaja con BEFORE o AFTER en la tabla requerida.

III Resultados

Cualquier empresa, administrador u usuario de base de datos que aplique nuestra propuesta metodológica aquí presentada, podrá tener:

1. Seguridad en la integridad de sus datos.
2. Seguridad de NO arriesgarse al robo de contraseñas y/o de los datos almacenados en su equipo.
3. Seguridad en la NO suplantación de registros.
4. Conocimiento de quién y en qué momento hubo un cambio de algún datos.
5. Campos de auditoría, automáticos, fueran del acceso de los usuarios.
6. Velocidad en procesos según necesidades.
7. Cuidado, a pesar de todo, su seguridad no es infalible.

IV Discusión

Una tarea de un administrador o un técnico o un usuario de una base de datos es identificar quién, cómo, cuando, donde, en qué medida o cantidad un usuarios tiene acceso a insertar, actualizar y/o eliminar datos, con qué, cómo y cuándo. Verificar que los datos están completos, son confiables y disponibles es importante para una empresa. Una auditoría NO sólo sería necesaria por un tiempo limitado, para usuarios o datos específicos, sino que puede ser requerida 24 horas por 7 días, para todos los datos que se introduzcan o estén en el DBMS[1].

Además, la seguridad informática está sustentada en la protección de la información. El valor de ambos es incalculable, por lo que debemos tomar conciencia de los riesgos a los que están expuestos los SI, su ubicación y almacenamiento, distribución y uso de la información.

Existen herramientas informáticas que hacen tareas de auditoria, algunas de ellas son privativas, de alto costo. Otras son software de código abierto. Sin embargo, a veces, conocer nuestro software nos permite utilizar sentencias y/o comandos que realizan una tarea aceptable. Para iniciar una cultura informática en nuestra organización.

La recuperabilidad significa que, si se da algún error en los datos, hay un error de programa o de hardware, el administrador recupere la base de datos al tiempo y estado en que se encontraba en estado consistente antes de que el daño se causara. Las actividades de recuperación incluyen el hacer respaldos de la base de datos y almacenar esos respaldos de manera que se minimice el riesgo de daño o pérdida de los mismos, tales como hacer diversas copias en medios de almacenamiento removibles y almacenarlos fuera del área en antelación a un desastre anticipado. La recuperación es una de las tareas más importantes de un administrador.

Nuestra propuesta, demuestra que los métodos simples generan excelentes resultados, además, es un proceso práctico, simple y elegante para verificar la calidad de los datos almacenados en nuestra base de datos.

Referencias

[1] Date and time functions. Available: http://stackoverflow.com/questions/168736/how-do-you-set-a-default-value-for-a-mysql-datetime-column
[2] Date and timestamp functions. Avaable: http://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html

Autor
Dr. Jorge Domínguez Chávez
Profesor Universidad Politécnica Territorial del estado Aragua, Venezuela
Tutor Designado Universidad Nacional Autónoma de México, México
Profesor Visitante Universidade Federal do Rio Grande do Sul, Brasil

[1]Data Base Management System. En Inglés para Sistema de Gestión de Bases de Datos (SGBD).