Proyecto #2 Fase 1

Teoría de Base de Datos II

Ing. Isaac Orellana

Microsoft SQL Server Transaction Log

Andres Lopez – 21741019

10 – Septiembre

3er Periodo Año 2020

# Introducción

En el siguiente documento vamos a datar todo lo relacionado con el transaction log de Microsoft SQL Server y la manera en la que almacena los datos. A continuación, se presentarán fragmentos de mi investigación, donde se detalla la manera en la que funcionan ciertas características dentro del sistema y porque se hacen de esa manera y no otra. Se explican ciertas partes de la lógica dentro del sistema y la manera en la que se almacenan los datos ingresados y como se mantienen los registros de versiones pasadas de los datos archivados en la base de datos. Veremos la importancia de tener un transaction log y como este puede ser utilizado de diferentes maneras dentro de alguna base de datos SQL Server. Veremos también la manera en las que los datos almacenados dentro del transaction log nos ayudan a restaurar nuestra base de datos en caso de que haya alguna caída inesperada del sistema. Vemos los riesgos de cada una de las maneras de implementar el transaction log conllevan dentro del sistema y la seguridad de los datos que se almacenan.

Indices

[Introducción 2](#_Toc50648679)

[Objetivos 4](#_Toc50648680)

[¿Qué es un transaction log de Microsoft SQL? 5](#_Toc50648681)

[¿Que es lo que se almacena en el transaction log? 5](#_Toc50648682)

[¿Podría un servidor SQL funcionar sin el uso del transaction log? 6](#_Toc50648683)

[Modelo de Recuperación Simple 7](#_Toc50648684)

[Modelo de Recuperación de Registro Masivo 8](#_Toc50648685)

[Modelo de Recuperación Completa 8](#_Toc50648686)

[Lógica Detrás del Transaction Log 8](#_Toc50648687)

[¿Como se visualiza la información del transaction log? 10](#_Toc50648688)

[Bibliografía 13](#_Toc50648689)

# Objetivos

Como objetivo de la investigación tenemos principalmente ampliar el conocimiento que el lector pueda tener sobre lo que es el transaction log en el SQL Server de Microsoft.  
Otro de nuestros objetivos es informar al leyente la importancia e impacto que tiene la manera en que se aplique el uso del transaction log dentro de la base de datos.  
Considero importante también explicar un poco la lógica que descubrí durante mi investigación acerca del almacenamiento de datos y como las versiones y cambios que se hacen luego de cada acción realizada en la base de datos a través del transaction log.

# ¿Qué es un transaction log de Microsoft SQL?

Lo que conocemos como transaction log es un archivo esencial de cada base de datos de SQL Server. Este contiene los registros producidos durante cualquier proceso realizado en la base de datos. El transaction log es el componente mas importante de una base de datos SQL Server cuando se trata de recuperación después de una caída desastrosa, aunque debe ser un archivo no corrupto (sin ningún daño que pudiese comprometer los contenidos de el mismo). Luego de cada ocurrencia dentro de la base de datos, se hace un registro en el transaction log. Todos los cambios que ocurran dentro de este serán secuenciales.

## ¿Que es lo que se almacena en el transaction log?

El transaction log almacena toda transacción que se realice dentro de la base de datos de SQL Server a excepción de algunos que se registra mínimamente como ser el BULK IMPORT y SELECT INTO. Internamente estos registros se dividen en partes mas pequeñas llamadas Virtual Log Files (VLF) o Archivos Virtuales del Registro (AVR). Cuando un AVR (VLF) alcanza su capacidad máxima, los registros continúan siendo hechos en el siguiente AVR (VLF) disponible. Una representación visual de un AVR (VLF) se puede hacer a través de un archivo circular o bien, una lista doblemente enlazada. Con esto lo que se desea representar es que cuando se alcanza la mayor cantidad de información que el AVR (VLF) puede almacenar, comienza desde el inicio nuevamente, asumiendo claro que todos los requerimientos se hayan cumplido y que todas las partes inactivas se hayan truncado previamente. El proceso de truncado es necesario para que se logren marcar las partes inactivas para que se puedan reutilizar y sobrescribir luego.

Si todas las siguientes condiciones se cumplen, no es necesario un transaction log:

1. La transacción de la cual forma parte ya forma parte de un commit.
2. Las paginas de la base de datos que fueron alteradas ya fueron escritas en disco por medio de un punto de control.
3. El registro no es necesario al momento de hacer un respaldo (completo, diferencial, o registro).
4. El registro no es necesario para ninguna característica que pueda leer el registro.

## ¿Podría un servidor SQL funcionar sin el uso del transaction log?

Debido al diseño del SQL Server y ACID (atomicidad, consistencia, asolación, durabilidad) no es posible hacer que funcione sin el uso del transaction log. Toda transacción que se realice en el servidor debe cumplir las características:

1. Una transacción atómica debe terminar completamente, o no haber iniciado en general.
2. Una transacción cumple la consistencia en el sistema al asegurarse de que cuando una transacción termine, el sistema se encuentre en un estado valido.
3. Cuando una transacción se ejecuta en isolacion, se debe ejecutar como si fuera la única acción realizada por el sistema en ese momento dado.
4. Una transacción es durable una vez que ya haya sido realizada dicha acción, y una vez que los cambios hechos en el sistema debido a este ya sean permanentes.

En situaciones especificas es posible trabajar con 2 transaction logs en el ambiente de trabajo. Al tener múltiples transaction logs no hará mejor el rendimiento de la base de datos SQL Server de ninguna manera, y los registros de las transacciones solo se pueden hacer en un archivo a la vez así que no es posible el funcionamiento en paralelo de estos.  
Dicho esto, es conocimiento general que se recomienda utilizar dos o mas transaction logs en caso de que el o los primeros ya estén llenos o se estén quedando sin almacenamiento. De cualquier manera, estos problemas deben ser manejados previamente al crear nuevos respaldos y monitoreando los espacios en disco.

Para el rendimiento adecuado de una base de datos SQL Server es necesario que el transaction log crezca en términos del tamaño de nuestros datos almacenados, ya que esto significa que se están registrando las transacciones exitosamente.  
La plataforma de SQL cuenta con tres tipos de recuperación en caso de alguna caída, y dependiendo e ella, se le asigna un uso distinto al transaction log y esto define la manera en la que este se manifiesta dentro del sistema.  
Los tres modelos disponibles para SQL Server son: Modelo de Recuperación Simple, Modelo de Recuperación de Registro Masivo, Modelo de Recuperación Completa.

Modelo de Recuperación Simple  
El respaldo por parte del transaction log no es compatible. El proceso de truncado es automático y el espacio se reutiliza. Hay riesgo de perdida de datos porque los últimos cambios realizados a la base de datos están expuestos. En el modelo de recuperación simple hay muy poca probabilidad de que el transaction log crezca en tamaño; solo en situaciones especificas cuando haya una transacción desarrollándose o una transacción que requiera una gran cantidad de cambios.

Modelo de Recuperación de Registro Masivo  
Los respaldos por parte del transaction log son compatibles y necesarios en el día a día. No existe el proceso automático de truncado, y los registros del transaction log deben hacerse regularmente para marcar el espacio que no este en uso para luego asignarlo para sobreescritura. El modelo de recuperación de registro masivo reduce el espacio utilizado para el transaction log al hacer menos registros para las operaciones de mayor tamaño.

Modelo de Recuperación Completa  
Los respaldos del transaction log son compatibles y necesarios regularmente. No existe el riesgo de perdida de datos bajo circunstancias normales. No hay procesos automáticos en el truncado de registros de transacciones ya que los registros del transaction log deben ser realizados constantemente para de esta manera poder marcar el espacio inutilizado para ser utilizado en sobreescritura. En el modelo de recuperación completa se encuentra la mayor posibilidad de crecimiento del transaction log ya que todas las transacciones son registradas.

Lógica Detrás del Transaction Log  
Los registros de las transacciones de SQL Server operan, en términos de logica, como una cadena de registros. Cada registro se identifica por un numero secuencial de registro (NSR). Cada vez que se escribe un nuevo registro en la parte logica del registro con un NSR mayor al NSR del registro anterior; en otras palabras, el NSR es secuencial en términos de la asignación de su valor numérico. Los registros son almacenados en secuencia serial al momento de su creación, entonces NSR2 debe, en teoría, ser mayor que NSR1, ya que el cambio relacionado con NSR2 fue hecho dentro del sistema luego de que el cambio de NSR1 fuera hecho. Cada registro contiene el identificador de la transacción a la cual pertenece. Para cada una de las transacciones, los registros son asociados con la transacción a la cual están individualmente conectadas en una cadena que utiliza punteros para apresurar la velocidad de rollback de cada transacción.

Los registros de modificación de datos son una operación logica hecha o son un registro de imagen de antes o después de los datos modificados. Las imágenes previas son una copia de los datos antes de que la operación que los modificara fuera ejecutada; la imagen posterior es una copia de los datos luego de que la modificación de los datos fuera ejecutada.

# ¿Como se visualiza la información del transaction log?

Ya visto como se manejan los datos dentro del transaction log de SQL Server podemos tomarnos un momento para hablar sobre como se presenta y que procesos se realizan para que nosotros los administradores podamos visualizar los datos almacenados.  
Aun siendo un tema complicado, es un tipo de conocimiento necesario si se intenta trabajar con la herramienta de lector de registros. Debido a su misma complejidad es que se vuelve casi esencial el uso de una herramienta, para simplificar su uso para el o los administradores del sistema.

La secuencia de operaciones se dicta por el numero de secuencia de registro (NSR), pero dado que varias transacciones pueden ser realizadas en el mismo momento, todas las entradas relacionadas a una transacción en especifico podrían no aparecer en secuencia; debido a esto es necesario e importante en ver el ID de transacción para saber que registros pertenecen a que transacciones.

Al momento de ingresar algún dato nuevo en alguna de las tablas podemos observar la presencia de ciertos atributos llamados “RowLog Contents X” (siendo ‘X’ el numero secuencial que indica el numero de RowLog Content). Al momento de ver las tablas afectadas por el ingreso de datos nuevos, podemos notar que se encuentran un par de operaciones con distintos nombres. Entre ellos esta LOP\_BEGIN\_XACT la cual indica el inicio de cualquier operación y la hora a la que comenzó. Similarmente hay una llamada LOP\_COMMIT\_XACT que almacena la hora en la que se termino completamente la transacción. Otra operación es LOP\_LOCK\_XACT la cual se encarga de almacenar los bloqueos que se pudieran realizar durante la transacción. Todo esto nos lleva a la 4ta de las tablas, la cual es la encargada de almacenar los datos que se ingresaron; es decir, la operación que va a datar los cambios que se harán a la versión anterior a la transacción (antes de la transacción). Todos estos datos se encuentran en los RowLog Columns, los cuales son 6 en total, y para saber cuáles columnas son relevantes a la transacción que se este llevando a cabo se puede revisar dentro de la columna “Num Elements”.

Ahora, ya con el conocimiento de donde se almacena cada cosa no es tan simple como ver lo almacenado para poder ver los cambios que hicimos porque lo que se almacena dentro de estas tablas. ¿Por qué es esto? Porque lo que se almacena dentro de nuestro RowLog Column es un dato converso a un formato particular para hacer mas eficiente su almacenamiento.

Para demostrar esto presentare un ejemplo:

0x300008000100000002000001001200526564

30 – Estado del bit A  
00 – Estado del bit B  
0800 – Desplazamiento para encontrar el numero de columnas dentro de la fila  
01000000 – Datos de tamaño fijo de col=1  
0200 – Numero de columnas  
00 – Bitmap nulo  
0100 – Numero de columnas de tamaño de variable  
1200 – Posición donde el primer tamaño de columna termina, lo cual es un byte que se intercambia a 0x0012 que se convierte en 18  
526564 – Datos visibles de tamaño de columnas variables

# Conclusiones

Luego de mi investigación, pude llegar a un par de conclusiones con respecto a lo que, previo a esta, pensaba sobre las bases de datos.  
Siendo esta mi segunda clase de Teoría de Base de Datos, creo que puedo decir que poseo un poco de experiencia utilizando bases de datos, ya que en pasadas clases fue necesario el uso de las mismas para desarrollar pruebas, tareas y proyectos.  
Algo que yo pensaba, era que los datos eran almacenados de la misma manera en la que se almacenarían en una computadora normal a la hora de almacenar algún tipo de dato (siendo esto, del dato original a su equivalencia binaria). Luego de la investigación noto que es un proceso mucho mas complejo que eso, y por eso mismo toma mucho mas trabajo y conocimiento. El hecho de que se convierta a un formato mas complicado que el simple binario hace mucho más complicada su conversión de vuelta a su estado original, pero demuestra ser una manera mas eficiente y segura de almacenar datos. Siento que la característica de no dejar posible la vista normal en caso de algún error en términos de seguridad los datos originales es muy importante, pero el desarrollo y los complejos algoritmos necesarios para almacenar todos los datos necesarios para poder representar los datos de manera que la persona que los necesite los pueda entender me parece una maravilla porque fue definitivamente algo muy elaborado que nos ayuda en el día a día aun si no somos nosotros quienes interactúan de manera directa con el sistema de base de datos.

Otra conclusión a la que llegue fue el hecho de toda la elaboración que se logra a través de las herramientas desarrolladas al momento de crear una base de datos es extremadamente importante.  
Previamente mi concepto de crear una base de datos solo rodeaba aspectos como ser mantener un ambiente flexible en el cual se pudieran hacer los cambios necesarios y deseados sin crear conflictos dentro de nuestro sistema, pero cuando investigue sobre todos los procesos que corren dentro de la base de datos solo como un por si acaso llega a pasar algo malo me di cuenta de lo equivocado que estaba. Es normal equivocarse, pero la magnitud de mi error siento que fue grande. Solo el proceso detrás de crear un sistema por medio del cual se pueda restaurar todo lo que tenia almacenado en caso de una caída inesperada abarca en términos de trabajo lo que pensé que tomaba hacer una base de datos. La manera en la que se desenvuelven las características de la base de datos dependiendo de que es lo que se planee hacer lo vuelve muy flexible al momento de personalizar una base para que resuelva un problema que otras bases encuentren difícil de resolver. La maniobrabilidad de lo que se puede lograr a través de el estudio de lo que se busca resolver, veo, que abre muchas puertas para crear algo nuevo y eficiente; sobre todo, innovador.

# Recomendaciones

Luego de la investigación puedo recomendar lo siguiente para alguien que busque comprender el SQL Server de Microsoft:

1. Considero la lectura sobre el tema muy importante, pero también considero que buscar la explicación de las lógicas detrás de cada característica o función son una excelente ayuda para poder comprender de mejor manera como trabajar al momento de resolver algún proceso propio
2. Dentro de mi investigación logre encontrarme con un blog que describía la importancia de utilizar el dato correcto para cada columna, de esta manera se pueden evitar un numero de problema que normalmente podría presentarse a la hora de el ingreso de datos. El asignar datos correctos para

# Bibliografía

<https://www.sqlshack.com/beginners-guide-sql-server-transaction-logs/>

<https://docs.microsoft.com/en-us/sql/relational-databases/sql-server-transaction-log-architecture-and-management-guide?view=sql-server-ver15>

<https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-type-conversion-database-engine?view=sql-server-ver15>

Microsoft SQL Server 2000 Bible (Paul Nielsen, 2000)

Fundamentos de SQL, 3ra Edición (Andy Oppel, Robert Sheldon, 2006)