Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas

LA RECUPERACIÓN DE LOS DATOS A PARTIR DE LOS FALLOS POR MEDIO DE LA BITÁCORA DE TRANSACCIONES DE LA BASE DE DATOS

Bryan David Velásquez Dávila
Asesorado por el Ing. Herman Veliz

Guatemala, noviembre de 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LA RECUPERACIÓN DE LOS DATOS A PARTIR DE LOS FALLOS POR MEDIO DE LA BITÁCORA DE TRANSACCIONES DE LA BASE DE DATOS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

BRYAN DAVID VELÁSQUEZ DÁVILA
ASESORADO POR EL ING. HERMAN VELIZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE DE 2018

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Oscar Humberto Galicia Nuñez
VOCAL V	Br. Carlos Enrique Gómez Donis
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Angel Roberto Sic Garía (a.i)
EXAMINADOR	Ing. Sergio Arnaldo Méndez Aguilar
EXAMINADOR	Ing. José Ricardo Morales Prado
EXAMINADOR	Ing. Edgar Estuardo Santos Sutuj
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

LA RECUPERACIÓN DE LOS DATOS A PARTIR DE LOS FALLOS POR MEDIO DE LA BITÁCORA DE TRANSACCIONES DE LA BASE DE DATOS

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 26 de febrero de 2016.

Bryan David Velásquez Dávila

ACTO QUE DEDICO A:

Dios	Por ser quien me ha dado la vida y la oportunidad de alcanzar mis sueños.
Mis padres	Por brindarme su apoyo y consejo incondicional y ser un ejemplo para mí en la vida.
Mis hermanos	Por apoyarme cada día con sus palabras y muestras de afecto.
Familia en general	Por el cariño y momentos compartidos.
La familia	Vela Minas, por su apoyo en mi vida personal y en especial a Perla Vela, por ser mi ayuda y apoyo en todo momento.
Mis amigos	Por estar en todo momento a mi lado.
Compañeros de trabajo	Por los momentos compartidos y apoyo en mi carrera.
Catedráticos	Por brindarme los conocimientos en el desarrollo de mi carrera.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por abrirme las puertas y poder iniciar esta carrera.
Facultad de Ingeniería	Por su enseñanza brindada en mi desarrollo profesional, que por medio de sus catedráticos me permitieron alcanzar este logro.
Asesor de tesis	Ingeniero Herman Veliz, por su paciencia y enseñanza en el desarrollo de este trabajo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Actualización Modificación sobre alguna cosa establecida que quede en estado actual.

Actualización diferida En base de datos, es un estado en el que no actualiza la información hasta que la transacción lleva a su punto de confirmación (*commit*). Si una transacción falla, no hay necesidad de deshacer (*rollback*) porque no se ha modificado la base de datos.

Anular Acción de dejar sin validez una cosa.

Archivo informático Es una unidad de datos o información almacenada en algún medio que puede ser utilizada por aplicaciones de la computadora.

Arquitectura de base de datos Explica cómo está estructurada una base de datos.

Auditoría Es el examen crítico y sistemático que realiza una persona calificada o grupo de personas independientes del sistema

Backup Se refiere a la copia y archivo de datos de la computadora de modo que se puede utilizar para

	restaurar la información original después de una eventual pérdida de datos.
Base de datos	Es un almacén que permite guardar grandes cantidades de información de forma organizada para que luego se encuentre fácilmente.
Binario	Es un sistema de numeración en el que los números se representan utilizando solamente dos cifras.
Bitácora	Un cuaderno o publicación que permite llevar un registro escrito de diversas acciones. Su organización es cronológica, lo que facilita la revisión de los contenidos anotados.
Bloque de disco	Es un conjunto de discos o sectores que realiza el sistema operativo.
Bloqueo	La base de datos no permite el ingreso de información debido a un problema de saturación.
Búcle	Conjunto de instrucciones que son ejecutadas de manera repetitiva, hasta que se cumpla una condición dada o se detenga manualmente el proceso.
Buffer	Memoria de almacenamiento temporal de información que permite transferir los datos entre

	unidades funcionales con características de transferencia diferentes.
Business intelligence	Es la habilidad para transformar los datos en información y la información en conocimiento, de forma que se pueda optimizar el proceso de toma de decisiones en los negocios.
Cláusula	En base de datos son sentencias del lenguaje de programación sql, por ejemplo, Select, Where, etc.
Comando	En informática es una instrucción u orden que el usuario proporciona a un sistema informático.
Commit	Es la acción de consignar un conjunto de cambios tentativos de forma permanente.
Comprimir	Hace referencia a la acción de presionar un objeto para que mantenga un tamaño mucho más pequeño del que tiene originalmente.
Concurrencia	La acción de permitir a varias transacciones acceder a una misma base de datos a la vez.
Conflicto	Es una manifestación de intereses opuestos, en forma de disputa.
Consistencia	Significa que los datos se mantienen idénticos durante cualquier operación, como transferencias.

Corrupción de datos	Es cuando la información ha sido modificada sin autorización legal.
Data warehouse	Es una base de datos corporativa que se caracteriza por integrar y depurar información de una o más fuentes distintas.
Dba	Es aquel profesional que administra las tecnologías de la información y la comunicación, responsable de los aspectos técnicos, tecnológicos, científicos, inteligencia de negocios y legales de bases de datos.
Dbms	Manejan de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.
Ddl	(Lenguaje de definición de datos) es un lenguaje que sirve para describir los datos y sus relaciones en una base de datos.
Directriz	Norma o conjunto de normas e instrucciones que se establecen o se tienen en cuenta al proyectar una acción o un plan.
Descartar	Rechazar o no tener en cuenta una posibilidad o circunstancia.
Disponibilidad	Posibilidad de una cosa o persona de estar presente cuando se le necesita.

Dispositivo	Es un aparato o mecanismo que desarrolla determinadas acciones.
Dml	Lenguaje de manipulación de datos.
Eficiente	La utilización correcta y con la menor cantidad de recursos para conseguir un objetivo o cuando se alcanza más objetivos con los mismos o menos recursos.
Extracción	Acción de obtener o sacar recursos para su uso posterior.
Factor	Es un elemento que actúa como condicionante para la obtención de un resultado.
Flexibilidad	Capacidad para adaptarse a diversas circunstancias.
Hardware	Conjunto de elementos físicos que constituyen una computadora o un sistema informático.
Indice	En base de datos es una estructura de datos que mejora la velocidad de las operaciones, por medio de identificador único de cada fila de una tabla.
Importación	En informática es el proceso de utilizar objetos externos entre sí.

Instancia	En informática es la iniciación de la ejecución de un programa en un computador,
Irreversible	Que no puede volver a un estado o situación anterior.
Inserción	Es la acción de agregar, ingresar alguna cosa o información.
Interfaz	Parte de un programa que permite el flujo de información entre un usuario y la aplicación.
Jerarquía	Es la forma de organización que se le asignará a diversos elementos de un mismo sistema.
Log	Es un registro de actividad de un sistema, que es guardado en un documento dentro del computador.
Masivo	Es aquello que se aplica en gran cantidad.
Memoria RAM	Memoria principal de la computadora donde residen programas y datos, sobre la que se pueden efectuar operaciones de lectura y escritura.
Minería de datos	Es el proceso de detectar la información procesable de los conjuntos grandes de datos.
Modificación	Es una acción que consiste en transformar, reformar, cambiar, alterar determinadas cosas.

Monitorear	Controlar el desarrollo de una acción o un susceso a través de uno o varios monitores.
Montar(base de datos)	Es la acción de poner en ejecución una base de datos.
Oracle	Es un sistema de manejo de bases de datos del modelo relacional desarrollado por la empresa Oracle Corporation.
Parámetro	Es el dato que se considera como imprescindible y orientativo para lograr evaluar o valorar una determinada situación.
Plataforma	Es un sistema que sirve como base para hacer funcionar determinados módulos de *hardware* o de *software*.
Redundancia	Definir o describir algo con palabras que no hacen más que volver a mencionar aquello que se explicó o mencionó.
Registro	En informática es un tipo o conjunto de datos almacenados en un sistema.
Rendimiento	Está vinculado a la proporción existente entre los recursos que se emplean para conseguir algo y el resultado que luego se obtiene.

Replicación	En informática es crear una copia exacta de un programa o archivo en un tiempo parcial.
Restauración	Volver al estado anterior a una cosa u objeto.
Reversión en cascada	Acción de regresar algo a un punto anterior en orden jerárquico (de menor importancia a mayor).
Rollback	Es una sentencia SQL que realiza la operación de devolver a la base de datos a algún estado previo.
Segmento de disco	Es un espacio de memoria de tamaño variable, compuesto por el identificador único del segmento (dentro del espacio de memoria del proceso) y el tamaño del segmento.
Sentencia	En programación una sentencia es una línea de código en algún lenguaje de programación.
Sincronización	Se refiere a que dos o más elementos, eventos u operaciones sean programadas para que ocurran en un momento predefinido de tiempo o lugar.
Software	Conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora.

SQL Server	Es un sistema de manejo de bases de datos del modelo relacional, desarrollado por la empresa Microsoft.
Tablespace	Una transacción es una secuencia de operaciones realizadas como una sola unidad lógica de trabajo.
Transacción	Una transacción es una secuencia de operaciones realizadas como una sola unidad lógica de trabajo.
Tipo de dato	En informática define un conjunto de valores y las operaciones sobre esos valores.
Tupla	En base de datos se refiere a un registro dentro de una tabla en una base de datos.
Truncar	En base de datos es la acción de eliminar la información de una tabla, dejando todos sus componentes.

RESUMEN

Las bases de datos se han convertido en parte fundamental para las empresas y los negocios generales. Es casi una obligación tener instalada una para llevar el control de todas las gestiones que se den, según el tipo de negocio. Como se sabe, puede que en algún momento presenten algún tipo problema; para lo cual es necesario estar preparados, para evitar la pérdida de la información de datos importantes.

En este trabajo de investigación se tiene la finalidad de presentar herramientas y soluciones para estos casos. Se ha investigado de tal forma que puedan ser aplicadas las soluciones de forma no muy complicada, aunque es necesario tener ciertos conocimientos técnicos en base de datos.

La recuperación de la información es un tema de mucha conveniencia, ya que en algún momento se pierda algo en los dispositivos que se utilizan diariamente. Por ejemplo, un teléfono, laptop, etc.; por esta razón, se debe saber que hacer al momento de que alguna causa suceda.

Oracle y SQL Server son dos administradores de base de datos de los más grandes y más usados en el mundo; por esa razón en este trabajo se investigó más a fondo sobre las herramientas para la recuperación de la información. También, se realizaron estudios y análisis de cómo un colapso en la base de datos afecta un sistema y a sus usuarios, por ejemplo, los bienes y tiempos dependiendo del tipo de negocio de la empresa y cómo se gestionen los procesos de negocio.

OBJETIVOS

General

Presentar una herramienta útil de recuperación de datos a partir fallos que está incluida dentro del gestor de la base de datos.

Específicos

1. Brindar un tema de solución a los conflictos y colapsos de las bases de datos que se dan de forma accidental o no controlada.

2. Presentar una herramienta para consulta de datos históricos dentro de la base de datos de los cuales no se tiene gestión y que, por lo tanto, se han dado por perdidos.

3. Estimar una vía de soluciones a los problemas comunes y diarios en las operaciones de los diferentes negocios que pueden ser irreversibles dentro de la gestión.

INTRODUCCIÓN

En los sistemas administradores de bases de datos la probabilidad de fallos siempre está presente. Aunque la instalación, los parámetros, las reglas, los estándares se hayan definido de una forma correcta, puede que no sean el motivo por el que una base de datos colapse. Estos errores comúnmente son nombrados errores no controlados. Para estos errores no puede darse alguna advertencia previa, simplemente hay que estar preparados con herramientas de respaldo para solucionar los fallos y recuperar la información perdida.

Una de estas herramientas es la bitácora de transacciones, un archivo con todas las operaciones que se realizan en la base de datos. Cada vez que se realiza una inserción, modificación o eliminación de datos, se almacena en la misma. Este proceso no se presenta al usuario, se maneja de forma transparente por el DBMS.

El motivo principal de la bitácora de transacciones es llevar el control de todo lo gestionado en la base de datos y, en sí, mantener la integridad y seguridad de la información almacenada. Este es un proceso complejo que no es muy común, por lo mismo no se le presta atención; pero si se realiza un análisis de los factores que pueden cubrir este archivo, se comprobará que es una herramienta poderosa para la recuperación de la información.

Uno de los puntos por los cuales esta herramienta es útil es el respaldo de la información al momento de colapso y fallo. Al almacenar todas las transacciones, se almacena en si un respaldo (no *backup*) de datos y puntos de restauración que pueden servir al momento de realizar una reparación formal.

1. RESTAURACIÓN DE LA BASE DE DATOS

El objetivo de una restauración completa de la base de datos es obtener la información que por algún motivo se ha perdido, por colapso o información errónea ingresada.

Durante el proceso de restauración, la base de datos completa se encuentra sin conexión. Antes de que ninguna parte de la base de datos esté en línea, se recuperan todos los datos a un punto coherente en el que todas las partes de la base de datos se encuentran en el mismo momento y en el que no existe ninguna transacción sin confirmar.

En el modelo de recuperación simple, no se puede restaurar la base de datos a un momento concreto de una copia de seguridad específica.

Se recomienda no adjuntar ni restaurar bases de datos de orígenes desconocidos o que no sean de confianza. Estas bases de datos pueden contener código malintencionado que podrían ejecutar código *transact-SQL* inesperado o provocar errores debido a la modificación del esquema o de la estructura de la base de datos física. Antes de usar una base de datos de origen desconocido o que no es de confianza, hay que ejecutar la acción ‘DBCC CHECKDB’ en la base de datos en un servidor y examinar también el código, como procedimientos almacenados u otro código definido por el usuario, de la base de datos.

1.1. Tipos de fallos

Existen diversas causas por las que el sistema de bases de datos puede fallar. Algunos tipos de fallos son los siguientes:

- Fallos locales previstos por la aplicación (*rollback* explícito o programado): durante la ejecución de una transacción pueden presentarse condiciones que requieran su cancelación de la misma (por ejemplo, un saldo insuficiente en una cuenta bancaria implica la cancelación de una transacción de reintegro sobre dicha cuenta).

- Fallos locales no previstos (sobrecarga, división por cero): errores lógicos de programación, o interrupciones provocadas. Un fallo local solo afecta a la transacción que se está ejecutando donde ha ocurrido el fallo. Normalmente supone la pérdida de los datos en memoria principal y en los búfers de entrada/salida.

- Fallos por imposición del control de concurrencia: el subsistema de control de concurrencia puede decidir abortar una transacción T (para reiniciarla posteriormente) bien porque viola la seriabilidad o porque varias transacciones están en un estado de bloqueo mortal y se ha seleccionado T como víctima.

- Fallos del sistema (caídas no críticas): consisten en un mal funcionamiento del hardware, errores software (del sistema gestor de base de datos (SGBD) o del sistema operativo) o de red, que afectan a todas las transacciones en progreso de ejecución en el sistema de base de datos. No dañan físicamente el disco (el contenido de la base de

datos permanece intacto y no se corrompe), pero se pierden los datos en la memoria principal en los búferes de entrada/salida.

- Fallos catastróficos o físicos: algunos ejemplos son el corte del suministro eléctrico, robo del disco, incendio, sabotaje, sobre escritura en discos o cintas por error, etc. Estos tipos de fallos suceden con menos frecuencia que el resto y su recuperación es más complicada.

- Fallos en la Transacción: por error lógico: la transacción no puede continuar su ejecución a causa de alguna condición interna; y por error del sistema: el sistema alcanza un estado no correcto. La transacción puede volver a ejecutarse más tarde.

- Caída del sistema: por ejemplo, errores del hardware o software de base de datos o software del sistema operativo.

- Fallo de disco: daño físico en el medio de almacenamiento masivo. Estos fallos ocurren cuando el disco se deteriora y se estropea, impidiendo escribir en este o bien cuando datos críticos de la base de datos se corrompen por el mal estado del disco que provocan un mal funcionamiento de la base de datos e incluso la pérdida de datos. Por ejemplo, si ocurre una rotura o un aterrizaje de alguna cabeza lectora-escritora en el disco, si funciona mal la lectura o la escritura o si ocurre un fallo durante una transferencia de datos). Afectan a todas las transacciones en curso y suponen la pérdida de información en disco (es decir, algunos bloques del disco pueden perder sus datos). Por ello es necesario la utilización de discos raid que permitan duplicar los datos; para el caso de que falle un disco la base de datos continúe trabajando

sin problema y simplemente al reemplazar el disco deteriorado, se vuelve a tener la base de datos asegurada.

1.2. Tipos de errores

- Errores de usuario: la mayoría de los problemas de pérdida de datos tienen que ver con el propio usuario, es decir, ocurren porque el propio usuario elimina la información. Las instrucciones DML (lenguaje de manipulación de datos) (*insert*, *delete* o *update*) de manipulación de datos son reversibles en las bases de datos transaccionales (especialmente si cumplen la norma *ACID* (atomicidad, consistencia, aislamiento, durabilidad)), pero no lo son las instrucciones DDL (*drop table* por ejemplo). Además, cuando la transacción se acepta (con un *commit*, por ejemplo), entonces pasa a ser definitiva. Si la transacción había borrado datos, estos se habrán perdido.

Los errores de usuario son los fallos más complicados de arreglar porque responden a una voluntad de borrar definitivamente los datos, la única posibilidad de recuperar los datos estará en la existencia de al menos una copia de los mismos.

- Errores en la ejecución de instrucciones: ocurren cuando una instrucción no pudo ejecutarse correctamente porque no cumple una determinada restricción y no se lleva a cabo y los datos no llegan a grabarse. Si la instrucción era compleja y había provocado empezar a volcar los datos hasta llegar a uno que no cumple la condición, entonces la instrucción se detiene y puede dejar datos incoherentes al no finalizar la instrucción.

- Errores en la programación de aplicaciones: en realidad es un problema parecido al anterior. Un caso habitual ocurre cuando resulta que una aplicación permite a varios usuarios trabajar con los mismos datos concurrentemente con la posibilidad de alguna interrupción (dos sesiones que se bloquean mutuamente) y los datos que intentan grabar no se graben jamás y, por lo tanto, se pierdan.

Hay otros numerosos ejemplos de estos problemas, bucles infinitos que escriben y escriben datos sin control hasta llenar los *tablespace* (almacén lógico de los archivos de la base de datos), problemas con los permisos de usuario al escribir, mal control de tablas mutantes en 'PL/SQL'.

- Errores de red: ocurre cuando la red pierde la conexión por problemas de hardware (el cable se desconecta, un aparato de red deja de funcionar) o de software (el *listener* (obtiene información en red) se bloquea). Se puede evitar el problema disponiendo de elementos redundantes de red en el servidor: dos interfaces de red conectadas a dos estructuras distintas de red.

- Errores críticos: son fallos que provocan la caída de la instancia de la base de datos, ya sea por un mal funcionamiento del software del sistema operativo (a veces también causado por fallos en disco) o por problemas en el hardware (desde problemas en la memoria ram hasta que se queme la placa base del servidor) que causan el detenimiento de la base de datos y, probablemente, pérdidas de datos durante el acceso a la misma.

En Oracle se puede simular este fallo mediante el uso del comando 'SHUTDOWN ABORT' que hace que se apague la base de datos de forma

inmediata. Como la información primero se almacena en memoria *RAM* y luego en disco, se perderán los datos en memoria, además, podemos haber empezado a almacenar transacciones y dejarlas a medio camino que provocan una base de datos corrupta (con datos incoherentes) que podría dejar de funcionar.

1.3. Modelos de recuperación en Sql Server

Los modelos de recuperación realizan el mantenimiento del registro de transacciones. Se listan los tipos de modelo de recuperación.

1.3.1. Modelo de recuperación simple

En modo de recuperación simple o sencillo (*simple recovery model*) las operaciones de registro mínimo realizarán un registro mínimo en el archivo log de SQL Server, que minimizan las escrituras y maximizan el rendimiento.

Todas las transacciones serán registradas en archivo log, sin embargo, una vez que la transacción finaliza, el espacio utilizado por dicha transacción podrá ser reutilizado para el registro de otras transacciones. No es necesario realizar copias de seguridad para reutilizar su espacio.

Sólo se permiten copias de seguridad completas o diferenciales. No se puede recuperar (*restore*) a un momento en el tiempo (*stopat*).

Recupera automáticamente el espacio de registro para mantener al mínimo los requisitos de espacio; en esencia, la necesidad de administrar el espacio del registro de transacciones. Para obtener información acerca de las

copias de seguridad de base de datos en el modelo de recuperación simple; vea copias de seguridad completas de bases de datos (SQL Server).

Las operaciones que requieren copias de seguridad del registro de transacciones no son compatibles con el modelo de recuperación simple. Las siguientes características no se pueden utilizar en modo de recuperación simple:

- *AlwaysOn* o creación de reflejo de la base de datos
- Recuperación de medios sin pérdida de datos
- Restauraciones a un momento dado

Los cambios realizados después de la copia de seguridad más reciente no están protegidos. En caso de desastre, es necesario volver a realizar dichos cambios. Solo se puede recuperar hasta el final de una copia de seguridad.

En el modelo de recuperación simple no se puede restaurar la base de datos a un momento concreto de una copia de seguridad específica.

Se recomienda no adjuntar ni restaurar bases de datos de orígenes desconocidos o que no sean de confianza. Estas bases de datos pueden contener código malintencionado que podría ejecutar código *transact-SQL* inesperado o provocar errores debido a la modificación del esquema o de la estructura de la base de datos física. Antes de usar una base de datos de origen desconocido o que no es de confianza se ejecuten ‘DBCC CHECKDB’ en la base de datos en un servidor que no sea de producción y se examine también el código, como procedimientos almacenados u otro código definido por el usuario de la base de datos.

Una restauración completa de base de datos con el modelo de recuperación simple implica una o dos instrucciones *restore*, en función de si desea restaurar una copia de seguridad diferencial de la base de datos. Si solo se usa copias de seguridad completas de la base de datos, se debe restaurar solo la copia de seguridad más reciente.

Si también se usa una copia de seguridad diferencial de la base de datos se restaura la copia de seguridad completa más reciente de la base de datos sin recuperar la base de datos; a continuación, se restaura la copia de seguridad diferencial más reciente de la base de datos y se recupera la base de datos.

1.3.2. Modelo de recuperación completa

En modo de recuperación completo (*full recovery model*) las operaciones de registro mínimo no se comportarán como tal, realizándose siempre un registro completo en el archivo log de SQL Server.

Todas las transacciones serán registradas en el archivo log. Solo la ejecución de una sentencia *backup log* permitirá reutilizar el espacio del archivo log ocupado por transacciones antiguas almacenadas en el *backup* del archivo log. Es decir, hasta que una transacción del archivo log de SQL Server no se guarda en un *backup*, no se permite su eliminación del archivo log, con la excepción de la *sentencia* 'BACKUP LOG WITH TRUNCATE_ONLY' *(*o 'BACKUP LOG WITH NO_LOG'), la cual permite vaciar el archivo log sin realizar físicamente una copia de seguridad (solo se debe usar en caso de emergencia). Si no se realizan copias de seguridad del archivo log periódicamente, el archivo log de SQL Server crecerá indefinidamente.

Se permite cualquier tipo de copia de seguridad (*backup*): completas, diferenciales, de archivos, de grupo de archivos (*filegroup*), y del archivo log. Es posible recuperar (*restore*) a un momento del tiempo (*stopat*).

El tema de restauración de archivo (modelo de restauración completa) solo es pertinente para las bases de datos con varios archivos o grupos de archivos con el modelo de recuperación completa o de carga masiva.

El objetivo de una restauración de archivos consiste en restaurar uno o varios archivos dañados sin necesidad de restaurar la totalidad de la base de datos. Un escenario de restauración de archivos consiste en una única secuencia de restauración que copia, pone al día y recupera los datos apropiados.

Si el grupo de archivos que se restaura es de lectura/escritura, es necesario aplicar una cadena ininterrumpida de copias de seguridad de registros después de que se restaure la última copia de seguridad de datos o diferencial. De esta forma, el grupo de archivos se actualiza a los registros existentes en los registros activos actuales del archivo de registro. Normalmente, el punto de recuperación está cerca del final del registro, aunque no necesariamente.

Si el grupo de archivos que se restaura es de solo lectura, por lo general, la aplicación de las copias de seguridad de registros no es necesaria y se omitirá. Si se hizo la copia de seguridad después de que el archivo pasará a ser de solo lectura, será la última copia de seguridad que se restaurará. La puesta al día se detiene en el punto de destino.

El objetivo de una restauración completa de la base de datos es restaurar toda la base de datos. Durante el proceso de restauración, la base de datos completa se encuentra sin conexión. Antes de que ninguna parte de la base de datos esté en línea, se recuperan todos los datos a un punto coherente en el que todas las partes de la base de datos se encuentran en el mismo momento y en el que no existe ninguna transacción sin confirmar.

En el modelo de recuperación completa, después de restaurar la copia o copias de seguridad de los datos, debe restaurar todas las copias de seguridad de registros de transacciones posteriores y, a continuación, recuperar la base de datos. Puede restaurar una base de datos a un punto de recuperación específico en una de estas copias de seguridad de registros. El punto de recuperación puede ser una fecha y hora específica, una transacción marcada o un número de secuencia de registro.

Al restaurar una base de datos, especialmente en el modelo de recuperación completa o el modelo de recuperación optimizado para cargas masivas de registros, debe usar una única secuencia de restauración. Una secuencia de restauración consta de dos o más operaciones de restauración que mueven datos en una o varias fases de restauración.

En general, la recuperación de una base de datos hasta el momento del error incluye los siguientes pasos básicos:

Realizar una copia de seguridad del registro de transacciones activo (denominado el final del registro). De esta forma se crea una copia del final del registro. Si el registro de transacciones activo no está disponible, todas las transacciones de esa parte del registro se pierden.

Restaurar la copia de seguridad completa más reciente sin recuperar la base de datos (*restore database* (nombre de base de datos) *from* (dispositivo de *backup*) *with norecovery*).

Si existen copias de seguridad diferenciales, restaurar la más reciente sin recuperar la base de datos (*restore database* (nombre de base de datos) *from* (dispositivo de *backup* diferencial) *with norecovery*).

Al restaurar la copia de seguridad diferencial más reciente se reduce el número de copias de seguridad de registros que se deben restaurar.

Restaurar los registros secuencialmente con la opción *norecovery*, comenzando por la primera copia de seguridad de registros de transacciones creada después de la copia de seguridad que se acaba de restaurar.

Recuperar la base de datos (*restore database* (nombre de base de datos) *with recovery*). Como alternativa, este paso se puede combinar con la restauración de la última copia de seguridad de registros.

1.3.3. Modelo de recuperación *bulk-logged* (registro masivo)

El modo de recuperación de registro masivo solo debe ser utilizado de forma intermitente o eventual para mejorar el rendimiento de las operaciones de registro mínimo.

Las operaciones de registro mínimo realizarán un registro mínimo en el archivo log de SQL Server, minimizando las escrituras y maximizando el rendimiento de SQL Server.

Todas las transacciones serán registradas en archivo log. Solo la ejecución de una sentencia *backup log* permitirá reutilizar el espacio del archivo log ocupado por transacciones antiguas almacenadas en el *backup* del log. Es decir, hasta que una transacción del archivo log de SQL Server no se guarda en un *backup*, no se permite su eliminación del archivo log, con la excepción de la sentencia 'BACKUP LOG WITH TRUNCATE_ONLY' ó 'BACKUP LOG WITH NO_LOG', la cual permite vaciar el archivo log sin realizar físicamente una copia de seguridad (solo usar en caso de emergencia). Si no se realizan copias de seguridad del archivo log periódicamente, crecerá indefinidamente.

Debe tenerse en cuenta, que al realizar una copia del archivo log durante un periodo de tiempo en el que han ocurrido operaciones de registro mínimo, la copia del archivo log almacenará tanto el contenido del *transaction log*, como el contenido de las páginas de datos afectadas (que se leerá de los correspondientes archivos de datos). Es decir, se tendrá un tamaño del archivo log relativamente pequeño, sin embargo, la copia podría tener un tamaño considerablemente grande.

Una conocida estrategia de configuración de bases de datos SQL Server es mantener un modo de recuperación completo durante toda la jornada, y en periodos de carga masiva o de operaciones de mantenimiento, utilizar un modo de recuperación de registro masivo (*bulk logged*), con el objetivo de mejorar el rendimiento de operaciones de registro mínimo ('SELECT INTO','BULK INSERT', bcp.exe, operaciones *DDL* como 'CREATE INDEX' ó 'DROP INDEX', etc. La utilización de una estrategia mixta de modo de registro completo y registro masivo probablemente requiera una actualización del plan de contingencias o políticas de *backups* y *restores* de SQL Server.

2. GESTIÓN DE RECUPERACIÓN DE LOS DATOS EN UNA BASE DE DATOS

Un sistema gestor de base de datos es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos.

Estos sistemas, también, proporcionan métodos para mantener la integridad de los datos, para administrar el acceso de usuarios a los datos y para recuperar la información si el sistema se corrompe. Permiten presentar la información de la base de datos en variados formatos. La mayoría incluye un generador de informes. También, pueden incluir un módulo gráfico que permita presentar la información con gráficos y tablas.

Hay muchos tipos distintos según cómo manejen los datos y muchos tamaños distintos de acuerdo a si operan en computadoras personales y con poca memoria o grandes sistemas que funcionan en *mainframes* con sistemas de almacenamiento especiales.

Generalmente, se accede a los datos mediante lenguajes de interrogación, lenguajes de alto nivel que simplifican la tarea de construir las aplicaciones.

También, simplifican la interrogación y la presentación de la información. El sistema gestor de base de datos permite controlar el acceso a los datos, asegurar su integridad, gestionar el acceso concurrente a ellos, recuperar los datos tras un fallo del sistema y hacer copias de seguridad. Las bases de datos

y los sistemas para su gestión son esenciales para cualquier área de negocio y deben ser gestionados con esmero.

2.1. Gestionar la recuperación de la información en Sql Server

Normalmente, todos los datos de una base de datos de SQL Server se restauran antes de que se recupere la base de datos. Sin embargo, una operación de restauración puede recuperar una base de datos sin restaurar realmente una copia de seguridad; por ejemplo, al recuperar un archivo de solo lectura que es coherente con la base de datos. Esto se conoce como restauración de solo recuperación. Cuando los datos sin conexión ya son coherentes con la base de datos y solo es necesario lograr que estén disponibles, una operación de solo restauración completa la recuperación de la base de datos y pone los datos en línea.

Una restauración de solo recuperación se puede realizar para una base de datos completa o para uno o varios archivos o grupos de archivos.
Una restauración de solo recuperación de base de datos puede resultar útil en las siguientes situaciones:

No se recuperó la base de datos al restaurar la última copia de seguridad en una secuencia de restauración y ahora se desea recuperar la base de datos para ponerla en línea.

La base de datos está en modo de espera y desea que se pueda actualizarla sin aplicar otra copia de seguridad de registros.

Una base de datos se restaura por etapas. Una vez finalizada la restauración del grupo de archivos principal, uno o varios de los archivos no

restaurados son coherentes con el nuevo estado de la base de datos; esto puede deberse a que la base de datos ha sido de solo lectura durante algún tiempo. Estos archivos solo necesitan recuperarse, no es necesario copiar los datos.

En una operación de restauración de solo recuperación los datos del grupo de archivos sin conexión pasan a estar en línea; no se produce ninguna fase de copia de datos, puesta al día ni reversión. Para obtener información acerca de las fases de la restauración, debe verse Información general sobre restauración y recuperación (SQL Server).

2.2. Gestión de recuperación de la información en Oracle

- Archivos *redo log*: los propósitos de los archivos de redo log proveen información de las transacciones ante un evento de falla de la base de datos. Cada transacción es escrita sincrónicamente en los archivos de redo log con el propósito de proveer un mecanismo de recuperación en caso de falla (con excepciones como cargas o lecturas hechas con la opción *nologgin*). Esto incluye transacciones que no han sido confirmadas (*commit*), información del segmento de *undo* y sentencias de administración de esquemas y objetos. Los archivos de redo log son usados en situaciones como falla de una instancia para recuperar los datos confirmados (*commit*) que no han sido escritos a los *datafiles*. Los archivos redo log son usados solo para recuperación.

Los archivos *redo log* registran todos los cambios hechos en los datos y proveen mecanismos de recuperación de un sistema ante una falla.

- Los archivos *redo log* están organizados dentro de grupos.

- Una base de datos Oracle requiere al menos de 2 grupos.

- Cada *redo log* está dentro de un grupo llamado *member* (asociado físicamente a un archivo en disco).

En cuanto a la estructura de archivos de *redo log* el *DBA* puede configurar la opción de la base de datos Oracle para mantener copias de archivos de *redo log* y así evitar tener un único punto de falla en una base de datos.

Grupos de *redo log online* es un conjunto de copias idénticas de archivos de *redo log* online.

El proceso *LGWR* (encargado de escribir los registros en los archivos *redo log*) escribe concurrentemente la misma información en todos los miembros de redo log del grupo. Oracle Server necesita un mínimo de 2 grupos de archivos de *redo log online* para la normal operación de la base de datos.

Oracle Server secuencialmente registra todos los cambios hechos en la base de datos al búfer de *redo log*. Las entradas a los *redo log* son escritas desde el búfer de los grupos llamado grupo actual de *redo log online* por el proceso *LGWR*. El *LGWR* escribe bajo las siguientes situaciones:

- Cuando se hace *commit* a transacciones
- El búfer de *redo log* se llena (*become one-third full*)
- Hay más de 1 MB de registros modificados en el búfer de *redo log*

Antes que el proceso *DBWR* (proceso que se encarga de escribir a disco. es el único con permiso de escritura en disco en la base de datos), escriba los bloques modificados en el búfer de la base de datos en cache a los *datafiles*.

Los *redo logs* son usados de manera cíclica. Cada grupo de archivos de *redo log* está identificado por un número de secuencia que es sobrescrito cada vez que el archivo log es reusado.

Un sistema de recuperación consiste en restaurar la base de datos a un estado que se sepa correcto, tras cualquier fallo que la haya dejado en un estado incorrecto.

2.2.1. Retornar la base de datos a un estado consistente

En Oracle hay recuperación automática ante fallos, el proceso varía dependiendo del tipo de fallo y las estructuras afectadas. Existen varias estructuras de recuperación, las cuales se mencionan a continuación.

- Redo log son archivos de almacenamiento de cambios en la base de datos. Se almacenan los cambios confirmados y no confirmados (en área global de memoria) actualiza la base de datos a partir del fallo. Contiene dos partes importantes.

- *Online redo log*: cada registro contiene el valor antiguo y el nuevo.

- *Archived redo log*: almacenaje de *redo log online* en archivos de reutilización (‘ARCHIVELOG ‘o ‘NOARCHIVELOG’).

Se puede recuperar de la base de datos desde un fallo de una o varias transacciones deshaciendo o rehaciendo las operaciones individualmente, transacción por transacción, a partir de archivos log.

- Archivos de control: un archivo de control es un archivo binario pequeño que forma parte de una base de datos Oracle. El archivo de control se utiliza para hacer un seguimiento del estado de la base de datos y la estructura física.

Cada dase de datos Oracle debe tener al menos un archivo de control; sin embargo, se recomienda crear más de uno. Cada copia de un archivo de control debe ser almacenado en una unidad de disco diferente multiplexada para hacer el registro en línea. Se utilizan para minimizar el riesgo de que todos los archivos de control sean borrados o estén dañados. El archivo de control contiene información como:

- Nombre de la base de datos
- Marca de la hora de creación de la base de datos
- Nombres y ubicaciones de archivos de datos
- Los nombres y ubicaciones de los redo log
- El actual número de secuencia de registro
- Información de *checkpoint*

La arquitectura física global de una base de datos se mantiene por medio de sus archivos de control, donde se registra la información de control referente a todos los archivos de la base de datos. Se utilizan para conservar la coherencia interna y guiar las operaciones de recuperación.

Estos archivos son fundamentales para la base de datos, por ello se almacenan varias copias en línea. Estos archivos deben almacenarse en discos físicos separados. Estos archivos almacenan la información en forma binaria para afirmar la integridad y seguridad de la base de datos.

El archivo de control debe estar disponible para la escritura por el servidor de base de datos Oracle siempre que la base de datos está abierta. Sin el archivo de control, la base de datos no puede ser montado y la recuperación es difícil.

El archivo de control de una base de datos Oracle se crea en el mismo tiempo que la base de datos. De forma predeterminada, por lo menos una copia del archivo de control se crea durante la creación de bases de datos. En algunos sistemas operativos de forma predeterminada es para crear varias copias. Debe crear dos o más copias del archivo de control durante la creación de bases de datos. También, puede crear archivos de control posteriormente.

- Directrices para los archivos de control

Pautas para administrar los archivos de control de una base de datos:

- Proporcionar los nombres de archivo para los archivos de control
- Control de archivos en discos diferentes *multiplex*
- Copia de seguridad archivos de control
- Administrar el tamaño de los archivos de control
- Creación de control archivos

Pasos que seguir para los archivos de control

- Creación de archivos de control inicial.

- Creación de copias adicionales, cambio de nombre y la reubicación de los archivos de control.

- Recuperar archivos de control y bases de datos por completo: en el caso de que se haya perdido o dañado un archivo de control, se deberá cerrar la base de datos y recuperar los archivos de control antes de recuperar la base de datos.

Tomar en cuenta que el proceso de recuperación restaurará el archivo de control original. Se debe renombrar la versión restaurada del archivo de control con el nombre de archivo original. Iniciar y montar la base de datos y comience el proceso de recuperación:

Montar los archivos redo log: para realizar el montaje de los archivos redo log, se abre el símbolo del sistema SQL* Plus, y se introduce lo siguiente:

‘CONNECT SYS/SYS_PASSWORD AS SYSDBA
STARTUP MOUNT;’

Oracle solicitará que introduzcan los nombres de archivo de registro. En primer lugar, Oracle buscará los archivos archive log y proporcionará de modo automático los nombres correctos correspondientes a los existentes. Si Oracle no puede encontrar los archivos archive log necesarios, será necesario aplicar de forma manual los registros de redo log con conexión necesarios.

Cuando se aplican los registros redo log con conexión de forma manual, deberá proporcionar la ruta completa y el nombre de archivo. Si introduce un registro incorrecto, vuelva a introducir el comando:

Para volver a poner en línea la base de datos y restablecer los registros, introducir el siguiente comando en el símbolo del sistema SQL*Plus:

‘ALTER DATABASE OPEN RESETLOGS;’

En los directorios donde se encuentran almacenados los registros redo log archivados, suprimir todos los archivos de registro.

- Segmentos *rollback: e*n cada base de datos Oracle se tiene uno o más segmentos de *rollback* donde se almacena la información que ha sido cambiada por las transacciones. Estas transacciones pueden ser definitivas, es decir, se ha realizado ya el *commit*, o puede que aún no se haya hecho dicho *commit*. Este tipo especial de segmento se utiliza principalmente para realizar una lectura consistente de la base de datos Oracle mientras se están modificando los datos y para llevar a cabo las recuperaciones de la base cuando esta cae por algún motivo.

La información de un segmento se almacena en las llamadas entradas de *rollback*. En estas entradas se detalla en qué *datafile* estaba el dato modificado, en qué bloque y también el dato anterior que se ha modificado en la transacción. Además, todas las entradas de *rollback* de una misma transacción están encadenadas unas con otras para que así, si se deben deshacer los cambios llevados a cabo en esa transacción, resulte más fácil de encontrarlos.

Las vistas más importantes de las que se puede extraer información sobre los segmentos de rollback son: V$ROLLNAME, DBA_ROLLBACK_SEGS y V$ROLLSTAT.

En una base de datos Oracle, se guardan todos los cambios en los bloques modificados en la estructura archivos redo; cuando se crea una nueva entrada de *rollback* en un segmento, realmente se está modificando un bloque

que se encuentra en dicho segmento de *rollback*, con la información de dicha entrada; por lo que este cambio también se almacena en los archivos *redo log*. Este doble almacenamiento de la información en los segmentos de *rollback* y en los archivos *redo log* es fundamental para realizar un buen proceso de recuperación de la base de datos en caso de que se produzca en fallo. Cuando se produce una caída de la base de datos, en el momento de la recuperación, se recupera el estado de los segmentos de *rollback* tanto con las transacciones que ya se habían terminado como con aquellas transacciones cuyo *commit* aún no se había realizado. El siguiente paso que se produce es el *rollback* de todas aquellas transacciones que se encuentran en los segmentos y para las cuales no se había completado el *commit*.

Para facilitar estas tareas, Oracle guarda por cada bloque una tabla de las transacciones que en cada momento se están ejecutando. Además, por cada transacción, por cada nuevo cambio que se realiza en los bloques de datos, se crea una entrada de *rollback* que se encadena a las anteriores entradas que están asignadas a esa misma transacción de forma ordenada.

Gracias a este sistema, cada vez que se desea restaurar el estado de una transacción al realizar su *rollback*, simplemente se debe detectar en qué bloque del segmento de *rollback* se están almacenando los cambios producidos por dicha transacción, observando en las tablas de transacciones los bloques del segmento de *rollback*; una vez detectado el bloque, se deben seguir una a una las entradas de la transacción para ir restaurando los valores antiguos en los bloques de datos de forma ordenada.

- Asignación de las transacciones a los segmentos *rollback: c*ada vez que comienza una nueva transacción, se asigna a un determinado segmento de dos formas diferentes. Se asigna la transacción al siguiente segmento

de *rollback* que se encuentre libre en ese momento de manera automática. Solamente se asigna una transacción cuando se realiza una instrucción de *DDL* o de *DML* que no sea un *select*.

También, se puede asignar una transacción a un segmento de *rollback* en concreto, de forma manual. De esta forma, se puede asignar a un segmento de *rollback* grande una transacción que se conocen de antemano que modifica un gran volumen de datos. Una vez finalizada la transacción, Oracle vuelve a asignar la siguiente de manera automática al primer *rollback* que encuentra libre. La instrucción es la siguiente:

```
'SET TRANSACTION
USE ROLLBACK SEGMENT NOMBRE_SEGMENTO;'
```

Cuando se finaliza una transacción, Oracle libera la información, aunque no la destruye, para soportar la lectura consistente de consultas que han comenzado antes de que se realizara el *commit*. Para asegurarse que la información se encuentra en los segmentos de *rollback* el mayor tiempo posible para estas consultas sin borrarla; Oracle va asignando las extensiones a los segmentos de *rollback* de manera secuencial y, cuando se ha llenado el segmento, se reutilizan las extensiones empezando nuevamente por la primera como si fuera un segmento circular.

Un segmento de *rollback* puede tener asignado solamente un número fijo de transacciones como máximo. Oracle se encarga de asignar las transacciones de una instancia de manera que todos los segmentos tengan el mismo número de transacciones, sin tener en cuenta el tamaño de las mismas ya que de antemano no lo puede conocer. El número de transacciones que puede contener cada segmento de *rollback* depende del tamaño del bloque.

- Asignación de extensiones: como se ha indicado anteriormente, un segmento de *rollback* debe tener al menos dos extensiones y cada una de sus extensiones está formada por un número determinado de bloques. A continuación, se explicará cómo se organizan las transacciones en los segmentos de rollback.

En un segmento de *rollback* pueden estar realizándose a la vez varias transacciones. Estas transacciones pueden estar escribiendo en distintas extensiones o incluso en la misma. Sin embargo, en un bloque de una extensión solamente puede contener información de una transacción; es decir, no pueden escribir dos transacciones en el mismo bloque de la misma extensión a la vez. Además, como se ha indicado que la escritura de transacciones es secuencial, en cada momento una transacción escribe en una sola extensión. Cuando una transacción se queda sin espacio para escribir en la extensión donde estaba, puede hacer dos cosas: bien reutilizar una extensión que ya estaba asignada al segmento o bien requerir una nueva extensión para el segmento de *rollback*.

La primera transacción que necesita más espacio chequea la siguiente extensión del segmento de *rollback*, y si no contiene información de transacciones activas, la adquiere. A partir de ese momento, todas las demás transacciones que necesiten espacio utilizarán esta extensión. Si nuevamente se llena esta extensión y alguna transacción sigue necesitando espacio libre, Oracle vuelve a comprobar si en la siguiente extensión que le toca ocupar, siguiendo el orden secuencial y circular de asignación de extensiones, no se están realizando transacciones activas.

Como se ve, Oracle mantiene un anillo formado por las extensiones que ha ido adquiriendo este segmento de rollback y siempre intenta reusar una de

las extensiones que lo forman antes que adquirir una nueva del sistema. Si en algún momento Oracle necesita utilizar una extensión y, en todas las que forman parte del anillo se están escribiendo transacciones activas, se ve obligado a adquirir una nueva extensión del sistema y a añadirla al anillo del segmento de *rollback* para seguir escribiendo. El número máximo de extensiones que pueden formar parte de un segmento de *rollback* viene determinado por un parámetro definido en el archivo de la base de datos *initSID.ora*, que es *MAXEXTENTS*.

- Desasignación de extensiones a un segmento *rollback: a*l borrar un segmento de rollback, todas las extensiones que tenía asignadas el segmento se devuelven al *tablespace* y pueden ser utilizadas por el resto de objetos que pertenecen al *tablespace*. Existe otra manera de devolver el espacio utilizado por un segmento sin tener que eliminarlo. Al momento de crear un segmento de *rollback* se puede indicar un valor en el parámetro *optimal* de la cláusula *storage*, que representa el tamaño óptimo de dicho segmento en *bytes*. Cada vez que Oracle necesita una nueva extensión para el segmento de *rollback*, comparta el tamaño que tiene dicho segmento con el valor del parámetro *optimal*, y si lo ha sobrepasado, irá devolviendo al *tablespace* las extensiones más antiguas que se van encontrando en las que ya no quedan transacciones activas.

El valor del parámetro *optimal* nunca podrá ser menor que el espacio necesario para la creación del segmento, en el que participan los parámetros *initial_extent, next_extent*, y *min_extents*. Para consultar los valores de estos parámetros se utiliza la vista DBA_ROLLBACK_SEGS de la siguiente forma:

‘SELECT segment_name, initial_extent, next_extent, min_extents, max_extents FROM DBA_ROLLBACK_SEGS;’

Y para conocer si los segmentos *rollback* tienen asignado un tamaño óptimo:

‘SELECT name, optsize FROM v$rollname, v$rollstat
WHERE v$rollname.usn = v$rollstat.usn ORDER BY 1;’

- Estados de un segmento *rollback: u*n segmento de *rollback* puede encontrarse en un determinado estado dependiendo de cada momento. Los estados en los que puede encontrarse son:

    - *Offline*: no ha sido adquirido por ninguna instancia de la base de datos.

    - *Online*: ha sido adquirido por alguna de las instancias y puede contener datos de transacciones activas.

    - *Needs recovery*: contiene datos de transacciones que no se pueden hacer *rollback* porque sus *datafiles* están inaccesibles o corruptos.

    - *Partly available*: contiene información de una transacción ‘en duda’ que son transacciones en entornos de base de datos distribuidas de las que aún no se ha recibido respuesta.

    - *Invalid*: ha sido borrado.

Las causas por las que un segmento puede pasar de un estado a otro son las siguientes:

Tabla I. **Estados de los segmentos rollback**

Estado inicial	Estado final	Motivo
Online	Offline	Se pone el segmento de '*rollback* offline' con la instrucción: 'ALTER ROLLBACK SEGMENT nombre_segmento OFFLINE'.
Offline	Online	Se pone el segmento de '*rollback* online' con la instrucción: 'ALTER ROLLBACK SEGMENT nombre_segmento OFFLINE'.
Offline	Invalid	Al borrar el segmento *rollback.*
Online	Partly available	Cuando falla la red y hace que una transacción distribuida quede en duda.
Partly available	Online	Se resuelve la transacción en duda.
Partly avalable	Offline	No se resuelve la transacción en duda.
Partly available	Needs recovery	Cuando un fallo del medio hace inaccesible la transacción en duda.

Continuación de tabla I.

Online	**Needs recovery**	**Cuando un fallo del medio hace inaccesibles los *datafiles* o cuando el semgmento está corrupto.**
Needs recovery	Offline	Si se soluciona satisfactoriamente la recuperación del medio.
Needs recovery	Invalid	Si se borra el segmento de *rollback.*

Fuente: elaboración propia.

Los estados de *partly available* y *needs recovery* son prácticamente iguales. En ambos el segmento contiene información de transacciones que no han sido resueltas. En un estado *partly available*, las transacciones distribuidas no han sido resueltas por culpa de un fallo en la red, mientras que en el estado de n*eeds recovery*, las transacciones no han sido resueltas por un fallo del medio o por estar corrupto el propio segmento.

Una diferencia entre ambos estados es que un administrador o el propio Oracle puede poner un segmento partly available en estado online, mientras que un segmento que necesita recovery, primero debe pasar a estado offline y luego online. Además, un segmento en estado needs recovery puede ser borrado por el administrador para eliminarlo si estaba corrupto; sin embargo, no se puede borrar un segmento *partly available*, primero se debe resolver la transacción en duda.

2.3. Backups de la base de datos

Existen varios tipos de backups de base de datos, se mencionan a continuación.

2.3.1. Backup a la nube para servidores de base de datos

El sistema de *backup* a la nube puede hacer copias en tiempo real de bases de datos SQL, así como de Exchange Server[1] sin ningún problema y sin afectar el funcionamiento de los sistemas. Este es un servicio especializado y es específicamente para servidores de bases de datos. El sistema corre tanto en Windows como en Linux.

- *Backup* para Exchange Server
- *Backup* de base de datos

2.3.2. Backup en Microsoft Sql Server

Respaldar un servidor SQL es muy importante. El sistema soporta SQL Server versiones 2005, 2008 y 2012. También, es capaz de sacar *backups* en tiempo real de las bases de datos sin interrumpir su operación.

3. CONTENIDO DE LA BITÁCORA DE TRANSACCIONES

La bitácora es una herramienta que permite registrar, analizar, detectar y notificar eventos que sucedan en cualquier sistema de información utilizado en las organizaciones. La estructura más ampliamente usada para grabar las modificaciones de la base de datos.

La estructura es ampliamente usada para grabar las modificaciones de la base de datos con respecto a las diferentes transacciones que se ejecutan.

Cada registro de la bitácora escribe una única escritura de base de datos y tiene lo siguiente:

- Nombre de la transacción: nombre de la transacción que realizó la operación de escritura.
- Nombre del dato: el nombre único del dato escrito.
- Valor antiguo: el valor del dato antes de la escritura.
- Valor nuevo: el valor que tendrá el dato después de la escritura.
- Es fundamental que siempre se cree un registro en la bitácora cuando se realice una escritura antes de que se modifique la base de datos.

También, se tiene la posibilidad de deshacer una modificación que ya se ha escrito en la base de datos; esto se realizará usando el campo del valor antiguo de los registros de la bitácora.

Los registros de la bitácora deben residir en memoria estable, como resultado el volumen de datos en la bitácora puede ser exageradamente grande.

3.1. Estructura del servidor de registro de transacciones de Sql

El registro de transacciones de SQL Server es un archivo único que por lo general tiene una extensión de archivo .LDF. Aunque es posible tener varios archivos de registro de una base de datos, el registro de transacciones siempre se escribe de forma secuencial y varios archivos de registro físico se vuelven como un archivo circular continuo.

SQL Server utiliza el registro de transacciones para asegurar que todas las transacciones mantengan su estado, incluso en caso de un fallo en el servidor o base de datos. Todas las transacciones se escriben en el registro de transacciones antes de que se escriba en los archivos de datos. Esto se conoce como la regla de escritura anticipada a la bitácora de transacciones.

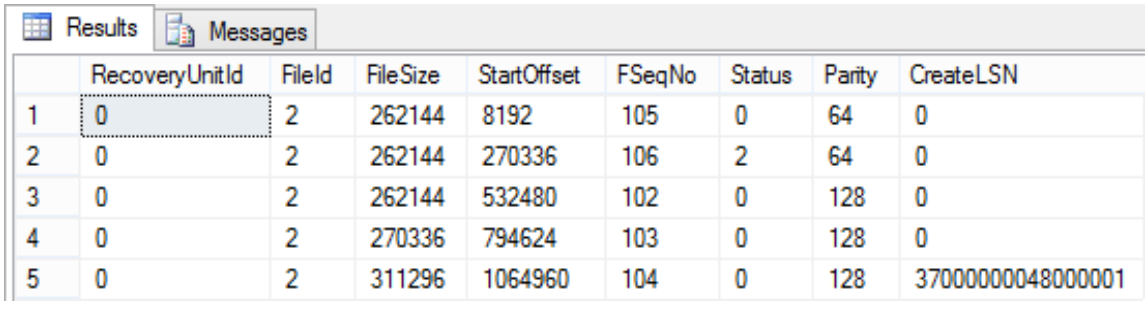
Cada acción realizada en SQL Server se registra en el registro de transacciones de SQL Server; múltiples entradas pueden ser creadas para una transacción, así como todos los bloqueos que se tomaron durante la operación. Cada entrada del registro tiene un número único conocido como el LSN (número de secuencia del archivo log).

Lógicamente el registro de transacciones de SQL Server se divide en varias secciones conocidas como archivos de registro virtuales o VLF. El registro de transacciones lógico se trunca y se expande en unidades de VLF. Si un VLF ya no contiene una transacción activa, el VLF puede ser marcado para su reutilización. Si el registro necesita más espacio, se asigna espacio en incremento de VLF. El número y tamaño de los archivos VLF se decide por el motor de base de datos y se esforzará para asignar el menor número posible de VLF. Aunque el tamaño y el número de VLF no se pueden configurar, se ve afectado por el tamaño inicial y el incremento de crecimiento de registro de transacciones.

El siguiente comando se puede ejecutar para ver cuántos archivos de registro virtuales existen, cuántos se ha utilizado y cuáles son sus tamaños. Esto se utiliza para determinar cuál debe ser el tamaño y el incremento correcto.

- *Dbcc loginfo*: el resultado es el siguiente:

Figura 1. **Respuesta de consulta de base de datos con el proceso dbcc loginfo**


Results Messages

	RecoveryUnitId	FileId	FileSize	StartOffset	FSeqNo	Status	Parity	CreateLSN
1	0	2	262144	8192	105	0	64	0
2	0	2	262144	270336	106	2	64	0
3	0	2	262144	532480	102	0	128	0
4	0	2	270336	794624	103	0	128	0
5	0	2	311296	1064960	104	0	128	37000000048000001

Fuente: elaboración propia.

Tabla II. **Descripción de campos del proceso dbcc loginfo**

Columna	Descripción
FieldId	Este es el número de identificación del archivo de registro físico. Sólo se aplica si se tiene más de un archivo de registro físico.
FileSize	El tamaño del archivo en bytes.
StartOffset	Este es el desplazamiento donde el VLF comienza en bytes. La salida se clasifica en esta columna.
FSeqNo	Este es el orden en el que se utilizará el VLF. El número más grande es el que se está utilizando actualmente.
Status	Hay 2 valores posibles, 0 y 2; 2 significa que el VLF no puede ser reutilizado y 0 significa que está listo para su reutilización.
Parity	Hay 2 posibles valores 64 y 128.
CreateLSN	Este es el LSN cuando se creó el VLF. Si el CreateLSN es 0, significa que se ha creado cuando se creó el archivo de registro de transacción física.

Fuente: elaboración propia.

Hay cuatro razones principales por las que uno podría estar interesado en leer el registro de transacciones.

- Auditoría forense

SQL Server ofrece una gran variedad de métodos que pueden ser implementados como medidas preventivas para evitar la necesidad de usar el registro de transacciones de SQL Server para auditar una base de datos. Esto incluye auditoría de SQL Server, trazas y eventos prolongados, captura de datos modificados, por nombrar solo unos pocos. La mayoría de estos, con la excepción de la traza predeterminada, requiere la aplicación antes de cualquier evento que ocurre.

El registro de transacciones de SQL Server está siempre presente y como tal puede ofrecer información valiosa después de un suceso que se produjo a pesar del hecho de que ninguna configuración especial avanzada se ha hecho.

En ausencia de todas las medidas preventivas, ser capaz de leer el registro de transacciones de SQL Server ofrece la posibilidad de descubrir quién lleva a cabo una transacción específica después de los hechos, así como la capacidad de obtener los valores modificados.

- Recuperación

Debido a que cada acción se registra en la bitácora de transacciones, de tal manera que una transacción puede ser puesta al día o deshacerse, la bitácora de transacciones de SQL Server puede ser utilizado para recuperar los datos perdidos.

Dado que solo los cambios son registrados, los datos se almacenan en un formato no entendible, obtener y leer esta información del registro no es fácil.

- Solución de problemas

En ciertos casos, ser capaz de ver exactamente lo que ocurrió en una base de datos por un período específico de tiempo es necesario. Por ejemplo, que el tamaño de la base de datos ha crecido inexplicablemente durante el transcurso de la noche. Sería imposible decir con solo observar los datos, esta podría ser la razón por la que ha crecido. Sin embargo, la lectura del registro de transacciones proporciona la información exacta acerca de lo que ocurrió durante el período específico que podría haber dado como resultado el rápido crecimiento de la base de datos.

- Identificación de un punto de restauración

Cuando se trata de restaurar una copia de seguridad, si se obtiene el punto exacto en el tiempo durante el que la base de datos está en un punto estable, se puede realizar el proceso de restauración para obtener los datos perdidos o dañados. En lugar de restaurar varias copias de seguridad hasta encontrarse uno que contenga los datos pertinentes de forma intacta, se puede leer el registro de transacciones para determinar cuándo se produjo el suceso y restaurar el preciso momento justo antes de que se produjo el evento.

4. PUNTOS DE RESTAURACIÓN EN LA BITÁCORA DE TRANSACCIONES

4.1. Creación de los puntos de restauración

Suponer que se está ejecutando una transacción T que ha realizado modificaciones sobre elementos de la base de datos, algunas de las cuales pueden haberse llevado a disco.

Es posible que ocurra un fallo de T en un momento en el que la transacción está en curso (ejecutándose), o bien cuando ya haya terminado.

Está claro que T deberá ser anulada si ha quedado a medio, puesto que no alcanzó su punto de confirmación (no llegó a anotar su entrada <*commit*, T> en la bitácora).

Nótese que para que sea posible deshacer cambios realizados sobre elementos de base de datos, es necesario que en el archivo de bitácora existan las entradas correspondientes a dichos cambios. ¿Y qué sucede si la transacción había finalizado cuando ocurre el fallo?

Se podría pensar algo como: esta transacción sí ha llegado a su punto de confirmación, luego se puede dar por terminada con éxito, y olvidarse de esta.

Sin embargo, no es seguro que esta transacción esté confirmada realmente, pues es posible que algún cambio sobre un elemento de la base de datos todavía no se hubiera llevado a disco.

Esto ocurre si el fallo tiene lugar mientras se están grabando en disco los elementos de la base de datos modificados: algunos cambios pueden quedar grabados en la base de datos en disco, pero otros no.

Corresponde con el estado confirmada de T revertida, *rollback*. Por esto, si la transacción había terminado con éxito (sí llegó a escribir <*commit*, T> en la bitácora), es necesario rehacer sus modificaciones para asegurar que quedan realmente consolidadas en disco.

Nótese que para que sea posible rehacer cambios, es necesario que en el archivo de bitácora, existan sus entradas correspondientes.

4.2. Escritura anticipada a la bitácora de transacciones

Se debe tomar en cuenta que la bitácora es un archivo en disco. Así que, como ya se sabe, actualizar la bitácora en disco implica:

- Copiar el bloque adecuado de la bitácora en disco principal
- Actualizar el bloque en memoria (insertar una nueva entrada)
- Copiar el bloque desde la memoria al archivo de bitácora en disco

Esto supondría una escritura en disco cada vez que se inserta una nueva entrada en la bitácora (es decir, cada vez que se anota la ejecución de una operación de transacción).

Para conseguir una única escritura por bloque, en la práctica:

- Se mantiene un bloque completo de bitácora en memoria: el búfer de bitácora.

- Cuando se llena de entradas, el bloque completo se escribe en disco (en el archivo de bitácora).

Con esto se evita escribir varias veces en el disco un mismo bloque de la bitácora, pero provoca otros problemas, que se mostrarán a continuación con un ejemplo.

Suponer que una transacción T en ejecución y que en el archivo de bitácora en disco ya está almacenada la correspondiente entrada <INICIAR,T>.

Tras modificar los elementos X e Y de la base de datos, se anotan las siguientes entradas sólo en el búfer de bitácora en memoria:

<ESCRIBIR,T,X,7,15>
<ESCRIBIR,T,Y,2,5>

Suponer que también algunos de estos cambios se consolidan en la base de datos en disco, por ejemplo, estos dos anteriores: en la base de datos se tiene por tanto X=15 e Y=5.

T realiza dos modificaciones más sobre los elementos Z y V, y después finaliza con éxito, así que se anotan, todavía solo en el búfer de bitácora, las entradas:

<ESCRIBIR,T,Z,6,3>, <ESCRIBIR,T,V,8,1> , <*COMMIT*,T>

El *commit* de T indica al sistema que puede confirmar en disco los cambios (hechos por T) aún no llevados a la V, es decir, los que modifican los valores Z de 6 a 3 y V de 8 a 1.

Suponer ahora que el sistema comienza a escribir en la base de datos en disco los elementos Z y V modificados por T; pero, recordar que, la realización de estos cambios todavía no se ha anotado en la bitácora en disco (sino que las entradas correspondientes siguen en el búfer de bitácora, que no está lleno aún).

Si el fallo ocurre justo durante la consolidación de los cambios de la base de datos en el disco (transferencia), entonces, es posible que algunos cambios de T sí se graben en el disco (en la base de datos queda Z=3), pero otros no (en la base de datos sigue V=8).

Lo más normal es que el fallo haya provocado la pérdida del contenido de la memoria principal (es decir, el área local a T, el búfer de bitácora y el búfer de base de datos). El proceso de recuperación accede al archivo de bitácora en disco donde hallará la entrada <INICIAR,T>, pero no encontrará ninguna de las entradas del búfer de bitácora que todavía no se habían volcado en disco cuando ocurrió el fallo.

En particular no encontrará la entrada <*COMMIT*,T>, así que intentará anular T. Sin embargo, T no debe ser anulada, pues se realizó totalmente y con éxito. Lo único que ocurre es que no se completó la confirmación en el disco de sus cambios. Tan solo se debería rehacer cada una de sus operaciones.

Este error se hubiera evitado si, antes de comenzar la consolidación de cambios de base de datos en el disco, se hubiera grabado en disco (y con éxito) la porción de bitácora que sólo estaba en el búfer de bitácora: el sistema hubiera detectado la confirmación de T (pues en la bitácora en disco estaría la entrada <*COMMIT*,T>); además, sí sería posible rehacer las modificaciones de T, porque estarían en bitácora (en disco) todas las entradas <ESCRIBIR,T,...>.

Pero aún existe un problema adicional. Se decía que el sistema intentará anular T. Dos de las entradas perdidas a causa del fallo son:

<ESCRIBIR,T,X,7,15>, <ESCRIBIR,T,Y,2,5>

Así que no se tiene anotado en bitácora en disco ni el valor anterior ni el valor nuevo para X ni para Y.

Pero ambos cambios fueron llevados a la base de datos en disco, de forma que X e Y ya contienen sus nuevos valores 15 y 5. No es posible deshacer estas operaciones de forma correcta.

Otra entrada perdida es <ESCRIBIR,T,Z,6,3>, que corresponde a otro cambio de T que sí ha dado tiempo a consolidar en disco antes del fallo. Tampoco en este caso se puede deshacer esta operación, por la misma razón que antes. Se ve que es imposible recuperar la base de datos a un estado de consistencia anterior.

Todo esto también se hubiera evitado si, antes de comenzar la grabación de cambios de base de datos en el disco, se hubiera grabado en disco la porción de bitácora que solo estaba en el búfer de bitácora: sí sería posible deshacer las modificaciones de T (si ello fuera necesario, claro), porque estarían en bitácora (en disco) todas las entradas <ESCRIBIR,T,...> necesarias.

En realidad, lo que se ha hecho es razonar que, para evitar estos problemas, es necesario seguir un protocolo de bitácora adelantada, puesto que asegurará que la base de datos podrá ser recuperada de forma correcta.

4.3. Protocolo de la escritura anticipada

- No se puede grabar en disco los cambios realizados por una transacción T, hasta que se haya forzado la escritura en disco de toda entrada de bitácora para T, hasta el momento actual.

- La operación confirmar una transacción no se puede completar hasta que se haya forzado la escritura en disco de cualquier entrada de bitácora para esa transacción.

Es decir, fuerza la escritura de la bitácora en disco antes de consolidar cualquier cambio realizado por T. Ya sea para consolidar cambios en disco antes de que T alcance su punto de confirmación o para consolidarlos después de que T lo haya alcanzado. Así que nunca va a ocurrir que, primero, se lleven a disco los cambios de elementos de base de datos realizados por T y, segundo, el sistema falle al ir a grabar en disco la parte de bitácora.

Pero sí puede ocurrir que, primero, se grabe físicamente la parte de la bitácora de una modificación y, segundo, el sistema falle cuando va a aplicar el cambio a la base de datos en disco.

En este último caso, al recuperar T, el gestor de recuperación verá si cada modificación pudo hacerse o no físicamente, pues está registrada en bitácora. Podrá deshacerla o rehacerla usando la entrada correspondiente en bitácora.

Con el uso del protocolo de bitácora adelantada, siempre se está seguro de que se puede llevar la base de datos a un estado consistente, aunque ocurra un fallo mientras se graban en disco los cambios de elementos de base de datos, pues será posible rehacer y/o deshacer los cambios.

4.4. Puntos de validación y sincronización del sistema de base de datos

Suponer que ocurre una caída en un sistema de base de datos donde se ejecutaban varias transacciones de forma concurrente. Al reiniciar el sistema, el gestor de recuperación accede al archivo de bitácora (en disco). Donde están anotadas todas las entradas correspondientes a modificaciones sobre elementos de la base de datos, realizadas por las transacciones. Para aquellas que alcanzaron su punto de confirmación, también estarán anotadas las entradas *commit* correspondientes. ¿Cómo sabe el sistema qué transacciones estaban 'activas' (en ejecución) cuando ocurrió el fallo y, por tanto, debe deshacer? Y, ¿cómo sabe el sistema cuáles transacciones debe rehacer? Pues el sistema debe examinar la bitácora desde el principio y rastrearla hacia adelante para determinar que:

- T estaba activa en el momento del fallo si ha escrito una entrada <INICIAR,T>, pero no ha escrito ninguna entrada <*COMMIT*,T> ni <*ROLLBACK*,T>.

- T alcanzó su punto de confirmación si ha escrito <*COMMIT*,T> en la bitácora.

Es posible evitar tener que recorrer toda la bitácora, en busca de las transacciones activas (para anularlas (deshacer sus operaciones) o descartarlas para reiniciarlas posteriormente). También, se puede evitar tener que rehacer todas las transacciones confirmadas.

El sistema gestor de base de datos, en particular, establece un punto de revisión de forma automática y periódica: por ejemplo cada M minutos, o

cuando se han grabado en el búfer de bitácora un número T de entradas del tipo <*COMMIT*,T> desde el último punto de validación.

El uso de puntos de validación hace necesario un nuevo tipo de entrada en la bitácora, denominado registro de validación, el cual contiene la siguiente información:

- Lista de identificadores de las transacciones activas (en el instante del punto de validación).

- Dirección dentro del archivo de la bitácora, de la primera y última entradas correspondientes a cada transacción activa (para rapidez de acceso si es necesario anular una transacción).

Marcar un punto de revisión/sincronización implica lo siguiente:

- Suspender temporalmente la ejecución de todas las transacciones en curso. Escribir en disco todas las entradas de bitácora que residan en ese momento en el búfer de bitácora (en memoria).

- Forzar la escritura en disco de todos los bloques del búfer de base de datos que se hayan modificado (es decir, se llevan a disco todas las actualizaciones realizadas).

- Escribir una entrada (registro de validación) en la bitácora en disco. Escribir en un archivo especial de arranque del sistema, la dirección física del registro de validación dentro del archivo bitácora. El gestor de recuperación accede a este archivo especial de arranque cuando el sistema es reiniciado tras el fallo.

- Reanudar la ejecución de transacciones. Así pues, cuando ocurra un fallo solo será necesario examinar la bitácora desde el último punto de revisión hacia adelante; toda transacción cuya entrada <*COMMIT*,T> esté en la bitácora antes de dicho punto de validación no deberá rehacer sus operaciones de escritura, pues ahora se está en la seguridad de que fueron consolidadas por completo (ya que en el punto de revisión se forzó la escritura en disco de tales actualizaciones).

Nota: el que marcar un punto de revisión implique la escritura en disco, no quiere decir que no existan otros momentos cuando también se consoliden cambios en disco, aparte de en estos puntos de validación.

Por otro lado, si una transacción T1 aborta (y, por tanto, es revertida), cualquier otra T2 (no confirmada) que haya leído elementos escritos por T1; también, debería ser anulada.

Esto es la reversión en cascada y puede ocurrir si el protocolo de recuperación garantiza planes recuperables, pero no planes sin cascada o planes estrictos.

Como ya se ha indicado antes, la anulación en cascada puede consumir mucho tiempo, y por esto los mecanismos de recuperación se suelen diseñar de modo que nunca sea necesaria la reversión en cascada.

Para deshacer transacciones es necesario emplear las entradas de escritura, y nótese que las entradas de lectura se anotan en bitácora para determinar si es necesaria la reversión en cascada.

Por tanto, si el método de recuperación garantiza que nunca ocurrirá anulación en cascada, no será necesario anotar en bitácora entradas correspondientes a lecturas de elementos.

Nota: como se verá más adelante, dependiendo de la técnica de recuperación que utilice el sistema gestor de base de datos, las modificaciones llevadas a la base de datos desde el búfer de base de datos en un punto de validación serán solo las realizadas por transacciones T ya confirmadas (actualización diferida) o todas, independientemente de si la T que las hizo está o no confirmada (actualización inmediata).

4.5. Técnicas de recuperación de fallos del sistema

A continuación, se describirán algunas técnicas de recuperación. No son descripciones de cómo un sistema específico implementa la recuperación; la intención es describir conceptualmente varias estrategias distintas para la recuperación de caídas del sistema de base de datos.

Con frecuencia las técnicas de recuperación están ligadas con los mecanismos de control de la concurrencia. Es decir, algunas técnicas funcionan mejor con unos métodos de control de concurrencia que con otros. Con todo, se analizarán, en primer lugar, los conceptos de recuperación sin tener en cuenta los mecanismos de control de la concurrencia; es decir, considerando un sistema monousuario; después se indicarán las circunstancias en las que, si se emplea un protocolo de control de concurrencia determinado, conviene usar un mecanismo específico de recuperación.

Una estrategia de recuperación representativa podría resumirse informalmente de la siguiente manera:

- Tras un fallo en el sistema de tipo 5 o 6 que ha producido daños en una parte considerable de la base de datos, el método de recuperación consistirá en restaurar una copia de seguridad anterior de la base de datos y reconstruir un estado más actualizado, rehaciendo las operaciones de las transacciones confirmadas anotadas en la bitácora hasta el momento de la caída.

- Si el fallo en el sistema es de los tipos 1 a 4 la estrategia consistirá en invertir los cambios que provocaron la inconsistencia, es decir, deshacer algunas operaciones. También, puede ser necesario rehacer algunas operaciones para restaurar un estado consistente de la base de datos. En este caso, solo se necesitan consultar las entradas anotadas en la bitácora.

En ambos casos, se considera que el archivo de la bitácora en disco no sufre ningún daño.

4.6. Técnicas de actualizaciones diferidas

Si el sistema sigue esta técnica, una transacción T no modifica la base de datos antes de completar con éxito su ejecución y llegar a su punto de confirmación; es decir, se difiere la grabación en la base de datos de los cambios realizados por T hasta después de su confirmación.

Durante la ejecución de T, todas sus actualizaciones se realizan en su área de trabajo privada y en el búfer de base de datos, y son anotadas en el búfer de bitácora (todo ello en memoria). Una vez que T finaliza con éxito, se fuerza la escritura en disco de la bitácora, momento en el que T alcanza su

punto de confirmación. Después se consolidan los cambios en la base de datos en disco.

Si ocurre el fallo antes de que T haya llegado a su punto de confirmación, no habrá modificado la base de datos en absoluto, por lo que no es necesario deshacer ninguna operación.

La información de bitácora asociada a esta transacción se utiliza para la ejecución de las escrituras diferidas. Si el fallo sucede una vez que T ha alcanzado su punto de confirmación, habrá que rehacer T (partir de las entradas de bitácora) porque no es seguro que todos sus cambios se hayan llevado a disco.

Si el sistema sigue esta técnica, una vez reiniciado, el gestor de recuperación ejecutará el algoritmo de recuperación no deshacer / rehacer.

- Crear dos listas de transacciones, llamadas activas y confirmadas, ambas vacías.

- Inicializar activas a la lista de transacciones activas que aparecen en el registro de validación (correspondiente al último punto de revisión) escrito en la bitácora.

- Examinar la bitácora a partir del último punto de revisión hacia adelante.

- Si se encuentra una entrada <INICIAR,T>, se añade T a la lista activas.

- Si se encuentra una entrada <*COMMIT*,T>, mover T de la lista activas a la lista confirmadas.

- Al terminar de examinar la bitácora, rehacer las operaciones de escritura de las transacciones de la lista confirmadas, en el mismo orden en que se escribieron en bitácora, reiniciar las transacciones de la lista activas.

El sistema está ahora preparado para aceptar nuevos trabajos (la base de datos está en un estado consistente).

Rehacer una operación de escritura realizada por T consiste en examinar su correspondiente entrada en bitácora <ESCRIBIR,T,X,valor_nuevo> y asignar valor_nuevo al elemento X de la base de datos. La operación rehacer debe ser idempotente (es decir, ejecutarla varias veces debe equivaler a ejecutarla una sola vez). Nótese que en las entradas de bitácora tipo escritura solo es necesario almacenar los valores nuevos y no los anteriores, puesto que nunca se requiere deshacer operaciones, pero sí rehacerlas. Reiniciar una transacción significa reintroducirla en el sistema como si fuera nueva. Puede hacerse el sistema de forma automática o el usuario manualmente.

Nota: es importante observar que las operaciones se rehacen en el orden en que aparecen anotadas en la bitácora, aunque pertenezcan a transacciones diferentes. Esto es, no se rehace cada transacción en aislado, sino que se van rehaciendo a la vez todas las transacciones confirmadas, operación a operación. Si el sistema es monousuario, en la lista activas habrá como máximo una transacción. Si el sistema es multiusuario, dependiendo de los protocolos de control de la concurrencia empleados, el proceso de recuperación puede ser más complejo. Cuando mayor sea el grado de concurrencia, más complicada será la tarea de recuperación:

Se considera un sistema donde la bitácora incluye puntos de validación y el control de la concurrencia utiliza bloqueo en dos fases, por lo que todos los

bloqueos de los elementos son mantenidos hasta que T llegue a su punto de confirmación. Después, los bloqueos pueden liberarse. Esto garantiza planes estrictos y seriables.

Una desventaja de este método es que limita la ejecución concurrente de las transacciones, pues los elementos permanecen bloqueados hasta que T llega a su punto de confirmación.

Está claro que dichas listas no almacenarán transacciones en sí, sino los identificadores de dichas transacciones, representados aquí con la letra T.

En realidad, todo el proceso de recuperación ha de ser idempotente; si el sistema falla durante dicho proceso, el segundo intento de recuperación podría rehacer operaciones de escritura que ya hubieran sido restauradas durante el primer intento. Pero su principal ventaja es que nunca es necesario deshacer operaciones, porque:

- Una transacción no modifica la base de datos hasta que llega a su punto de confirmación; luego, ninguna transacción se revierte por haber fallado durante su ejecución.

- Una transacción T1 nunca lee un elemento modificado por otra transacción T2 no confirmada, porque los elementos permanecen bloqueados por T2 hasta que esta alcanza su punto de confirmación; luego, nunca hay reversión en cascada.

5. BITÁCORA DE TRANSACCIONES EN LOS DIFERENTES DBMS

5.1. Funciones del sistema gestor de base de datos

- Administración de diccionario de datos: los *DBMS* almacenan las definiciones de los elementos de datos y de sus relaciones en un diccionario de datos. En concreto todos los programas acceden a los datos a través del *DBMS*.

El *DBMS* usa el *data dictionary* para ver las estructuras de datos requeridos y sus relaciones, librando de estar programando complejas relaciones en cada programa.

Administración en el almacenamiento de los datos: los *DBMS* crean y administran estructuras complejas requeridas para el almacenamiento de los datos, librando de estar programando y definiendo las características físicas de los datos. Ayuda con las validaciones y tipos que se usan; también, los DBMS guardan la base de datos en múltiples archivos físicos por lo que se puede acceder a estos en el disco concurrentemente.

- Presentación y transformación de los datos: los *DBMS* transforman los datos ingresados en la estructura requerida para ser almacenados dichos datos, o sea que hacen distinción entre el formato lógico y el físico de los datos, manteniendo independencia en los datos.

- Administración de seguridad: consiste básicamente en los permisos que se las da a diferentes usuarios que manipulan la base de datos para mantener la integridad de los datos.

- Control de acceso multiusuario: el sistema gestor de base de datos crea estructuras complejas que permiten a varios usuarios acceder a los datos. Para que los datos no sean perjudicados el *DBMA* usa algoritmos complejos para mantener la integridad de dichos datos y permitir el acceso concurrente de varios usuarios a la base de datos.

- Administración de recuperación y respaldos: provee procedimientos que aseguran la integridad de los datos a través de respaldos y recuperación de datos por cualquier falla en el sistema o el hardware.

- Administración de integridad de datos: promueve y refuerza reglas de integridad de datos y elimina dichos problemas de integridad y redundancia de datos maximizando su consistencia.

- Lenguajes de acceso a la base de datos e interfaces de aplicaciones programadas: existe un lenguaje no procedimental que es un lenguaje de consultas uno son los *DDL* (lenguaje de definición de datos) y el otro *DML*(lenguaje de manipulación de datos). El *DDL* define cómo se almacenarán los datos y el *DML* permite al usuario extraer datos. Además, el sistema gestor de base de datos permite interactuar a las aplicaciones creadas en lenguajes de alto nivel como Visual Basic, con la base de datos.

- Interfaces de comunicación de base de datos: provee de interfaces intermediarias entre la base de datos y el usuario que pueden ser implementadas.

- Establecer y mantener las trayectorias de acceso a la base de datos de tal forma que los datos puedan ser accesados rápidamente.

- Manejar los datos de acuerdo con las peticiones de los usuarios.

- Registrar el uso de las bases de datos.

- Interacción con el manejador de archivos. Esto a través de las sentencias en DML al comando del sistema de archivos. Así el manejador de base de datos es el responsable del verdadero almacenamiento de los datos.

- Respaldo y recuperación. Consiste en contar con mecanismos implantados que permitan la recuperación fácilmente de los datos en caso de ocurrir fallas en el sistema de base de datos.

- Control de concurrencia. Consiste en controlar la interacción entre los usuarios concurrentes para no afectar la inconsistencia de los datos.

- Seguridad e integridad. Consiste en contar con mecanismos que permitan el control de la consistencia de los datos para evitar que sean perjudicados por cambios no autorizados o previstos.

- El objetivo primordial de un sistema manejador base de datos es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer, almacenar y manipular información de la base de

datos. Todas las peticiones de acceso a la base se manejan centralizadamente por medio del DBMS, por lo que este paquete funciona como interfaz entre los usuarios y la base de datos.

5.2. Bitácora de transacciones en Sql Server

Cada base de datos en SQL Server tiene un *transaction log* (bitácora de transacciones) asociado. Este es un componente esencial de SQL Server, el cual la utiliza para registrar un historial de cada modificación que sufre la base de datos como resultado de las transacciones. Dicho registro es de vital importancia para mantener la integridad de los datos y deshacer los cambios resultantes de transacciones incompletas ya sea por error del sistema o por la cancelación por parte de los usuarios.

Durante la operación de la base de datos la escritura a la bitácora tiene prioridad, es decir, todos los cambios primero se escriben a la bitácora y luego se aplican a la base de datos.

Debido a su importancia, es imperativo respaldar la bitácora regularmente ya que, de no hacerlo, será imposible recuperar la base de datos en caso de falla.

¿Qué tan frecuentemente hay que hacer los respaldos? La respuesta está en función de dos factores principales:

- ¿Qué tan frecuentemente cambian los datos almacenados en la base de datos?

- ¿Qué tan sensible es la organización a la pérdida de información?

Por ejemplo, si la base de datos almacena datos clínicos y la tasa de cambios es bastante alta, es decir, se agrega y se cambia información constantemente, entonces los respaldos tendrán que ocurrir con mayor frecuencia que cuando se trata de una base de datos que casi no cambia o cuya información no es crítica para el negocio. Entre más frecuentes sean los respaldos, se estará más protegido contra la pérdida de datos.

Otro beneficio de realizar respaldos frecuentes es que cuando la base de datos opera en modo de recuperación completa, el contenido de la bitácora se almacena hasta que se realice un respaldo, lo cual significa que el archivo de la bitácora crecerá y crecerá mientras no se respalde. Una vez respaldado, SQL Server reutilizará el espacio dentro del archivo lo cual pondrá bajo control su crecimiento.

5.2.1. Como evitar problemas con la bitácora de transacciones

- Respaldar la bitácora frecuentemente.

- No poner límite en el tamaño de la bitácora. En lugar de limitar el tamaño del archivo en sí, mejor colocar la bitácora sola en un espacio dedicado y monitorear el espacio libre constantemente.

- Si se realizan cambios masivos de datos, utilizar comandos que no hagan uso de la bitácora.

Es importante que cada vez que un usuario ingrese o modifique datos se grabe un campo de auditoría en un registro identificándolo; debe asegurarse de que los registros no puedan ser modificados ni borrados.

Los registros de auditoría están diseñados para permitir el rastreo de cualquier entrada o proceso llevado a cabo en un sistema. Los detalles de cada transacción se registran en un archivo de transacciones. El estudio de transacciones puede proporcionar información de cómo se modificó el archivo.

El almacenamiento de estos detalles es automático e invisible para el usuario; también, se debe almacenar la información relativa al usuario, de forma que sea claro saber quién llevó a cabo la transacción. Si el sistema tiene un reloj interno, también, se marca cada transacción con la hora exacta para saber cuándo ocurrió. Si surge la necesidad de revisar un registro particular en un archivo, es relativamente fácil determinar quién hizo la transacción, cuándo ocurrió, cuáles datos contenía la transacción y cómo se modificó la base de datos o el registro del archivo maestro.

En caso de las aplicaciones integradas que corren en redes, es casi imprescindible el establecimiento de autorizaciones individuales para cada consulta de datos y para cada tipo de modificación, pues de otro modo, se haría difícil evitar que cualquier persona consulte o modifique datos sin autorización de la gerencia. Para esto:

Debería existir una tabla que indique para cada potencial usuario: su contraseña, a que aplicaciones puede acceder; qué actividades de consulta o modificación de datos está autorizado a realizar.

El sistema de control interno debería prever: quién tendrá la responsabilidad de manejar dicha tabla, quién puede autorizar modificaciones a ella, qué constancias escritas deben dejarse de cada modificación.

5.2.2. Configuración de la bitácora de transacciones en Sql Server

Las bases de datos de SQL Server tienen un registro de transacciones que almacena todas las operaciones realizadas en la base de datos.

El registro de transacciones se debe truncar periódicamente para evitar que se llene. Sin embargo, algunos factores pueden retrasar el truncamiento del registro, por lo que es importante supervisar el tamaño del registro. Algunas operaciones se pueden registrar mínimamente para reducir su impacto sobre el tamaño del registro de transacciones.

El registro de transacciones es un componente esencial de la base de datos y, si se produce un error del sistema, podría ser requerido para volver a poner la base de datos en un estado coherente. El registro de transacciones nunca se debe eliminar o mover, a menos que se conozcan totalmente las implicaciones de esas acciones.

Los puntos de comprobación crean puntos buenos conocidos a partir de los cuales se empiezan a aplicar los registros de transacciones durante la recuperación de base de datos.

5.2.3. Operaciones de registro mínimo en Sql Server

En SQL Server existen algunas operaciones denominadas de registro mínimo debido a que realizan un registro mínimo de información en el archivo log de SQL Server (minimizando las escrituras en el archivo log de SQL Server, así como el tamaño y crecimiento del archivo log de SQL Server), bajo ciertas condiciones que ahora se verán. Este tipo de operaciones minimizan

sensiblemente la cantidad de escrituras realizadas en el archivo log de SQL Server, lo cual, puede resultar beneficioso desde el punto de vista del rendimiento de SQL Server.

Un ejemplo típico son las sentencias *delete* y *truncate*. La sentencia *delete* registrará en el archivo log la información correspondiente al borrado de cada fila afectada (realizará un borrado fila a fila). Por el contrario, la sentencia *truncate* tan solo registrará las páginas de datos afectadas por su ejecución (es decir, registrará solo qué páginas de datos han dejado de estar asignadas a qué tabla, sin registrar el contenido de dichas páginas). También, es cierto que en una sentencia *delete* es posible utilizar una cláusula *where*, mientras que en una sentencia *truncate* no es posible utilizar una cláusula *where*. La conclusión es clara: la sentencia *truncate* es inmediata incluso con tablas con cientos de millones de filas, mientras que la operación *delete* es muy costosa y requerirá escribir en el archivo log de SQL Server una gran cantidad de información (elevado tiempo de ejecución y riesgo de llenar el archivo del archivo log e incluso el disco duro). Existen más detalles que diferencian a *delete* y *truncate*, como la de asignación de páginas.

El objetivo de las operaciones de registro mínimo en SQL Server es muy claro: permitir recuperar la transacción, sin permitir la recuperación a un momento dado, pero con grandes ventajas de rendimiento y de necesidades de almacenamiento del archivo log de SQL Server. Es decir, si a la mitad de la ejecución de una operación de registro mínimo se produce una caída de la instancia de SQL Server, en el siguiente arranque de SQL Server, se podrá volver a un estado consistente de la base de datos, ya que SQL Server será capaz de deshacer (*rollback*) o rehacer (*redo*) las transacciones necesarias (gracias a la información almacenada en el archivo log de SQL Server) para garantizar la consistencia de la base de datos. Sin embargo, al realizar un

registro mínimo no se podrá recuperar una base de datos a un momento del tiempo, es decir, una sentencia *restore database stopat* o *restore log stopat*.

Operaciones de importación masiva (BCP.EXE, *openrowset, bulk y bulk insert*), siempre y cuando se especifique la opción *tablock*, la tabla no se esté replicando; dependiendo, también, de la definición de índices de la tabla y de la existencia o no de datos en la tabla, en particular:

- Si la tabla no tiene índices, el registro de las páginas de datos será mínimo. Si la tabla no tiene índices agrupados sino uno o más índices no agrupados, el registro de las páginas de datos siempre será mínimo. Sin embargo, el modo como se registran las páginas de índice será mínimo solo si la tabla está vacía (si la tabla no está vacía, el registro de las páginas de índice será completo).

- Si la tabla tiene un índice agrupado y está vacía, los datos y las páginas de índice se registrarán de forma mínima. En cambio, si la tabla tiene un índice agrupado, pero no está vacía, las páginas de datos y las de índice se registrarán de forma completa, independientemente del modelo de recuperación utilizado.

- Operaciones ‘SELECT INTO’. Este tipo de operaciones es vital al trabajar con grandes cantidades de datos, por ejemplo, ocurre en entornos de *Data warehouse* y *Business intelligence*, donde resulta ser muy querido: la manera más rápida de cargar una tabla, es a través de *select into* en un modo de recuperación simple o de registro masivo; especialmente, si la base de datos está correctamente dimensionada.

- Operaciones DDL como *create index, alter index rebuild, dbcc reindex, drop index*. El hecho de que estas operaciones puedan ser tratadas como operaciones de registro mínimo, implica una sensible mejora de rendimiento en la ejecución de planes de mantenimiento y otras tareas de administración de base de datos que impliquen operaciones DDL; además, de minimizar el espacio ocupado en los archivos de log de SQL Server**.**

Las operaciones de copias de seguridad y restauración de SQL Server se producen dentro del contexto del modelo de recuperación de la base de datos. Los modelos de recuperación se han diseñado para controlar el mantenimiento del registro de transacciones. Un modelo de recuperación es una propiedad de base de datos que controla la forma como se registran las transacciones: si el registro de transacciones requiere que se realice la copia de seguridad y si lo permite y qué tipos de operaciones de restauración hay disponibles. Existen tres modelos de recuperación: simple, completa y por medio de registros de operaciones masivas. Normalmente, en las bases de datos se usa el modelo de recuperación completa o el modelo de recuperación simple. Una base de datos se puede cambiar a otro modelo de recuperación en cualquier momento.

5.3. Bitácora de transacciones en Oracle – Transacciones en Oracle

El servidor de Oracle garantiza la consistencia de los datos con base en transacciones. Las transacciones proporcionan mayor flexibilidad y control cuando los datos cambian y ello asegura la consistencia de los datos en el caso de un fallo en el proceso del usuario o del sistema. Las transacciones consisten en sentencias DML que componen un cambio consistente en los datos.

Muchos sistemas operativos modernos manejan el acceso a archivos bastante bien; se puede usar ASM para la administración de almacenamiento de bases de datos Oracle, ya que administrar dispositivos de la base de datos requiere un administrador de sistemas experimentado y, además es tardado, por lo que casi siempre la velocidad extra ganada con estos dispositivos no vale la pena para la agilidad y esfuerzo extra requeridos para mantenerlos.

6. LECTURA Y OBTENCIÓN DE DATOS DE LA BITÁCORA DE TRANSACCIONES

6.1. Lectura de la bitácora de transacciones en Sql Server

A pesar de que Microsoft no proporciona una herramienta, hay algunas funciones incluidas que se pueden usar para ver el registro de transacciones. Debido a que estos procedimientos son indocumentados, tampoco están soportados por Microsoft y, como tal, viene con la advertencia de 'uso en su propio riesgo'.

6.1.1. FN_DBLOG*()*

Esta función con valores de tabla (que era *dbcc log* antes de SQL Server 2005) permite ver las entradas de registro de transacciones en línea. Este procedimiento acepta dos parámetros; el inicio *LSN* y el final *LSN*. Para ver todas las entradas disponibles se pueden pasar un valor nulo para ambos parámetros; se mostrarán todas las entradas en la parte activa del registro en línea.

'SELECT * FROM FN_DBLOG (NULL,NULL)'

6.1.2. FN_DUMP_DBLOG*()*

Esta función lee tanto el registro en línea y copias de seguridad de registro y acepta 68 parámetros. Todos los parámetros deben especificarse con el fin de ejecutar la instrucción.

‘SELECT * FROM fn_dump_dblog(NULL,NULL,'DISK',1 ,
'D:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\
Backup\NombreBaseDeDatos.bak'
,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL, NULL,NULL,NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL, NULL,NULL,NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL, NULL,NULL,NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL, NULL,NULL,NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL, NULL,NULL,NULL,NULL,NULL,NULL,NULL
,NULL,NULL,NULL,NULL);’

La respuesta de la consulta es la siguiente:

Figura 2. **Respuesta de consulta de base de datos con el proceso *FN_DUMP_DBLOG***

Fuente: elaboración propia.

6.1.3. Interpretación de los datos en FN_DBLOG y FN_DUMP_DBLOG

Este es un tema muy complicado, pero con el fin de comprender el valor de utilizar una herramienta de lector de registro, es necesario tener una idea básica de lo que sería necesario para interpretar los datos en el registro sin la ayuda de una herramienta.

La secuencia de operaciones se indica mediante el *LSN* (número de secuencia del archivo log); múltiples operaciones pueden ocurrir al mismo tiempo, todas las entradas vinculadas a transacciones específicas pueden no aparecer en secuencia; por lo que es importante observar también en el ID de la transacción para las entradas de registro, que corresponden a las transacciones.

Ejemplo, se realizará una inserción de una fila en una tabla simple denominada [dbo.attribute] y ver lo que sucede en el registro.

‘INSERT INTO dbo.attribute VALUES ('Red')’

En esta consulta solo se está buscando en las entradas con el mismo *ID* de transacción que el *INSERT*.

‘SELECT [Transaction ID], [Current LSN], [Transaction Name], [Operation], [Context],[AllocUnitName],[Begin Time],[End Time], [Transaction SID],
[Num Elements] , [RowLog Contents 0], [RowLog Contents 1],
[RowLog Contents 2], [RowLog Contents 3]
FROM fn_dblog (NULL, NULL)

WHERE [Transaction ID] = (SELECT [Transaction ID] FROM fn_dblog (null,null) WHERE [Transaction Name] = 'INSERT')’

Figura 3. **Respuesta de consulta de base de datos con el proceso FN_DBLOG**

Results | Messages

	Transaction ID	Current LSN	Transaction Name	Operation	Context	AllocUnitName	Begin Time	End Time
1	0000:00000e36	0000006a:00000158:0002	INSERT	LOP_BEGIN_XACT	LCX_NULL	NULL	2014/09/19 11:07:49:480	NULL
2	0000:00000e36	0000006a:00000158:0003	NULL	LOP_LOCK_XACT	LCX_NULL	NULL	NULL	NULL
3	0000:00000e36	0000006a:00000158:0017	NULL	LOP_INSERT_ROWS	LCX_CLUSTERED	dbo.attribute.PK_...	NULL	NULL
4	0000:00000e36	0000006a:00000158:0018	NULL	LOP_COMMIT_XACT	LCX_NULL	NULL	NULL	2014/09/19 11:07:49

Fuente: elaboración propia.

Puede observarse que la primera entrada es una operación ‘LOP_BEGIN_XACT’. Esto indica el comienzo de una transacción y muestra la fecha y hora de inicio. Del mismo modo, la última entrada es una operación ‘LOP_COMMIT_XACT’ que muestra la fecha y hora de finalización de la transacción.

La segunda fila indica que una operación de bloqueo fue hecha (‘LOP_LOCK_XACT’). Si se desea, puede obtenerse información sobre el bloqueo con la columna *lock Information*.

La tercera fila es donde empieza a ser realmente interesante. Esta es la entrada donde se encontrarán los datos reales que se insertaron en la tabla. Los valores de los datos se almacenan en el contenido de las columnas *RowLog*. Hay 6 columnas *RowLog* de contenido.

Para saber cuales de estas columnas son relevantes para la operación que ocupa, se puede comprobar la columna *Num Elements.*

Figura 4. **Respuesta de consulta de base de datos con el proceso FN_DBLOG**

	Num Elements	RowLog Contents 0	RowLog Contents 1	RowLog Contents 2	RowLog Contents 3	RowLog Contents 4	RowLog Contents 5
1	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	3	0x30000800010000000200000010012005265 64	0x	0x0101000C0000CF4D9E230000010200040204000A819444...	0x	0x	0x
4	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fuente: elaboración propia.

En este caso particular, hay 3 elementos, lo que significa que solo se deben observar a los valores presentes en [*RowLog Contents* 0], [*RowLog Contents* 1] y [*RowLog Contents* 2]. La información de los valores que han sido ingresados en la transacción se encuentran en forma hexadecimal, por lo que no puede obtenerse a simple vista el valor. Para obtener el valor se debe reconstruir la entrada.

Los datos de la *RowLog Contents* 0 son:

0x30000800010000000200000010012005265 64

Que puede ser desglosado de la siguiente manera.

La consulta devuelve lo siguiente.

Tabla III. **Explicación de contenido devuelto por la base de datos**

Valor hexadecimal	Posible valor
30	Estado de bit.
00	Bit de estado B.
0800	Offset para encontrar el número de columnas de la fila.
01000000	Los datos de longitud fija col = 1.
0200	Número de columnas.
00	Mapa de bits null.
0100	Número de columnas de longitud variable.
1200	Posición en la que termina la primera columna de longitud variable, esto es byte intercambiando que es 0x0012 que se traduce en 18.

Fuente: elaboración propia.

El valor insertado real se puede extraer desde el registro. Ahora se realizará un ejemplo de actualización de una entrada. Se cambiará el valor ‘rojo’ a ‘rad’.

‘UPDATE dbo.attribute SET name_e = 'Rad' WHERE id = 1’

El cambio en la consulta al registro de transacciones se realiza en [Transaction Name] = update, queda de la siguiente manera.

‘SELECT [Transaction ID], [Current LSN], [Transaction Name], [Operation], [Context], [RowLog Contents 0], [RowLog Contents 1]
FROM fn_dblog (NULL, NULL)
WHERE [Transaction ID] = (Select [Transaction ID] FROM fn_dblog(null,null) WHERE [Transaction Name] = 'UPDATE')’

La consulta retorna lo siguiente:

Figura 5. **Respuesta de consulta de base de datos con el proceso FN_DBLOG**

	Transaction ID	Current LSN	Transaction Name	Operation	Context	RowLog Contents 0	RowLog Contents 1
1	0000:00000e39	0000006a:00000160:0001	UPDATE	LOP_BEGIN_XACT	LCX_NULL	NULL	NULL
2	0000:00000e39	0000006a:00000160:0002	NULL	LOP_MODIFY_ROW	LCX_CLUSTERED	0x65	0x61
3	0000:00000e39	0000006a:00000160:0003	NULL	LOP_COMMIT_XACT	LCX_NULL	NULL	NULL

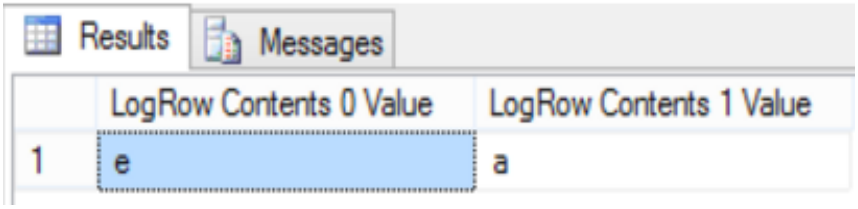
Fuente: elaboración propia.

Observe los valores en *RowLog Contents* 0 y 1. Si se convierte el valor a un tipo de dato *varchar* se observa que SQL Server solo registra el cambio real. *RowLog Contents* 0 contiene el valor antes y *RowLog Contents* 1 contiene el valor después.

‘SELECT CAST(0x65 AS VARCHAR)’

‘SELECT CAST(0x61 AS VARCHAR)’

Figura 6. **Respuesta de consulta de base de datos con el proceso CAST**



Fuente: elaboración propia.

6.2. Lectura de la bitácora de transacciones en Oracle

Para visualizar el contenido de la bitácora de transacciones en Oracle, la base de datos debe contener las siguientes configuraciones:

Se ejecuta el comando siguiente y muestra las configuraciones estándar.

‘SQL> ARCHIVE LOG LIST;
DATABASE LOG MODE: ARCHIVE MODE
AUTOMATIC ARCHIVAL: ENABLED
ARCHIVE DESTINATION: /ARCHIVE1/EXPL1
OLDEST ONLINE LOG SEQUENCE: 2029
NEXT LOG SEQUENCE TO ARCHIVE: 2031
CURRENT LOG SEQUENCE: 2031’

Al realizar la siguiente consulta en la línea de comando de Oracle:

‘SELECT * FROM v$log’

6.3. Oracle Logminer

Es una herramienta de Oracle que permite consultar a la base de datos en línea y recuperar información de los archivos de registro a través de una interfaz SQL.

Puede identificarse cuando un daño lógico de una base de datos se ha dado, errores a nivel de aplicación. Estos podrían incluir errores como aquellos en los que las filas erróneas se han eliminado debido a valores incorrectos en una cláusula *where* por ejemplo, las filas se actualizan con valores incorrectos, el índice incorrecto se pierde y así sucesivamente. Por ejemplo, una aplicación de usuario puede actualizar por error una base de datos para dar a todos los empleados el 100 % en aumentos salariales en lugar del 10 %; un administrador de la base (DBA) podría eliminar accidentalmente una tabla importante del sistema. Es importante saber exactamente cuándo se cometió un error para saber cuándo iniciar la recuperación. Esto permite restaurar la base de datos al estado como se encontraba justo antes de la corrupción.

6.3.1. Configuración de logminer

Hay cuatro objetos básicos en una configuración *logminer* con lo que se debe estar familiarizado: la base de datos fuente, la base de datos de la minería, el diccionario *logminer* y los archivos de registro redo log con los datos de interés.

La base de datos fuente produce todos los archivos de registro de redo log que *logminer* puede analizar.

La base de datos de la minería es la que utiliza *logminer* cuando se realiza el análisis.

El diccionario *logminer* es utilizado para proporcionar los nombres de tablas y columnas, en lugar de ID de objetos internos, en la presentación de los datos de registro de redo log que solicite.

Logminer utiliza el diccionario para traducir identificadores de objetos internos y tipos de datos de objetos y los formatos de datos externos. Sin un diccionario, *Logminer* devuelve los ID de objeto internos y presenta los datos como datos binarios.

Por ejemplo, observar la siguiente instrucción SQL:

‘INSERT INTO HR.JOBS(JOB_ID, JOB_TITLE, MIN_SALARY, MAX_SALARY) VALUES('IT_WT','Technical Writer', 4000, 11000);’

Sin el diccionario *logminer* muestra lo siguiente:

‘INSERT INTO "UNKNOWN"."OBJ# 45522"("COL 1","COL 2","COL 3","COL 4") VALUES
(HEXTORAW('45465f4748'),HEXTORAW('546563686e6963616c20577269746572'), HEXTORAW('c229'),HEXTORAW('c3020b'));’

6.3.2. Requisitos para el uso de logminer

Fuente y minería de base de datos: la base de datos de origen y la base de datos de minería deben estar corriendo en la misma plataforma de hardware.

La base de datos de minería puede ser la misma que la base de datos fuente o pueden estar completamente separadas. También, debe ejecutar la misma versión o una versión posterior del software de base de datos Oracle como base de datos de origen; debe usar el mismo juego de caracteres (o un súper conjunto del conjunto de caracteres) utilizado por la base de datos de origen.

6.3.3. Diccionario de logminer

El diccionario debe ser producido por la misma base de datos fuente que genera los archivos de registro *redo log* que *logminer* analizará.

Todos los archivos de registro de *redo log: d*ebe ser producido por la misma base de datos fuente. Debe ser de una versión 8.0 o posterior de base de datos Oracle. Sin embargo, varias de las características introducidas *logminer* partir de la versión 9.0.1 de trabajo solo con archivos de registro de rehacer producidos en una Oracle9 i base de datos o posterior.

Logminer no permite mezclar archivos de registro *redo log* de diferentes bases de datos o el uso de un diccionario de una base de datos diferente a la que generó los archivos de registro *redo log* para ser analizadas.

6.3.4. Consulta a la bitácora de transacciones con logminer

Logminer permite realizar consultas basadas en los valores de columna. Por ejemplo, puede realizar una consulta para mostrar todos los cambios a la tabla “EMPLEADOS“ que tienen un aumento de sueldo más de una cierta cantidad. Datos como este se pueden utilizar para analizar el comportamiento del sistema y realizar tareas de auditoría.

Para la extracción de datos con *logminer* hacia los archivos redo log se utilizan dos funciones: ‘DBMS_LOGMNR.MINE_VALUE’ y ‘DBMS_LOGMNR.COLUMN_PRESENT’**.** Como apoyo a estas funciones se proporcionan ‘REDO_VALUE’ y ‘UNDO_VALUE que son columnas en la vista ‘’V$LOGMNR_CONTENTS’.

El siguiente es un ejemplo de cómo se puede utilizar la función *MINE_VALUE* para seleccionar todos los cambios a la tabla ‘empleados’ que aumentaron el sueldo en más del doble de su valor original:

‘SELECT SQL_REDO FROM V$LOGMNR_CONTENTS
WHERE
SEG_NAME = 'EMPLOYEES' AND
SEG_OWNER = 'HR' AND
OPERATION = 'UPDATE' AND
DBMS_LOGMNR.MINE_VALUE(REDO_VALUE, 'HR.EMPLOYEES.SALARY')
>
2*DBMS_LOGMNR.MINE_VALUE(UNDO_VALUE,
'HR.EMPLOYEES.SALARY');’

Como se muestra en este ejemplo, la función *MINE_VALUE* toma dos argumentos:

El primero especifica si se debe extraer el rehacer (*redo value*) o deshacer (*undo value*) parte de los datos. La porción de rehacer de los datos son los datos que se encuentran en la columna después de una inserción, actualización o eliminación.

El segundo argumento es una cadena que especifica el nombre completo de la columna para ser extraído (en este caso, ‘empleados’.salario). La función *MINE_VALUE* siempre devuelve una cadena que se puede convertir de nuevo al tipo de datos originales.

CONCLUSIONES

1. El respaldo en las bases de datos es un tema de alta importancia. Todo lo referido al negocio se encuentra almacenada en esto; por lo tanto, se deben tener las prevenciones y precauciones debidas para el momento cuando estas tengan un fallo técnico o lógico.

2. Conocer recuperadores de bases de datos representa ser precavidos. Si se tienen los conocimientos básicos en estos temas, la reacción ante un error o colapso en la base de datos puede ser inmediata y se puede hacer el uso de las herramientas técnicas adecuadas.

3. Tener presentes los posibles tipos de errores y colapsos de una base de datos es importante para reaccionar de buena manera cuando esto pase, y así atacar al problema específico.

RECOMENDACIONES

1. Realizar *backup* de base de datos en un lapso de tiempo considerable. Estos tiempos puede que sean asignados por el jefe de área para cuando se hagan transacciones de alta importancia.

2. Contemplar un espacio específico para el almacenamiento de *backup* de base de datos para el momento de fallo o pérdida de información para tener el respaldo para proceder a la solucionar.

3. Tener programas de diagnósticos de base de datos instalados en servidores adecuados para la prevención de los daños inesperados. Esto para que siempre esté en revisión el estado de la base de datos y prevenir fallos que provoquen pérdida de información.

BIBLIOGRAFÍA

1. ÁRIAS, Angel. *Fundamentos de progamación y base de datos*. 2a ed. Estados Unidos: IT Campus Academy, 2016. 286 p.

2. CARDOSO, Lucía. *Sistemas de bases de datos II*. 8a ed. Caracas, Venezuela: Universidad Católica Andrés, 2006. 206 p.

3. CHAO, Lee. *Database development and management*. Estados Unidos: Auerbach Publications, 2005. 607 p.

4. DATE, CJ., *Introducción a los sistemas de bases de datos*. 7a ed. México: Pearson Education, 2011. 936 p.

5. MORRIS, Coronel. *Database systems design implementation and management*. 12a ed. Estados Unidos: Cengage Learning, 2016. 784 p.

ANEXOS

Anexo 1. **Colapso en base de datos mundial de pasaportes y visas estadounidenses**

Fuente: Mail Online. http://www.dailymail.co.uk/news/article-2703554/Glitch-crashes-global-US-passport-visa-operations.html. Consulta: 16 de julio de 2016.

Anexo 2. **Colapso de base de datos de registro de automóviles**

Fuente: Tampa Bay Times.
http://www.tampabay.com/news/politics/stateroundup/state-database-crashes-infuriate-florida-motorists-and-tax-collectors/2300073, Consulta: 14 de octubre de 2016.