RECUPERACIÓN DE FALLAS

Màs asignaturas académicas
Publicaciones de Estudiantes
Diapositivas AIU
Acceso al Campus Virtual
Herramientas de Inteligencia Artificial
Revista Campus Mundi
Classes en Vivo
AIU Blog

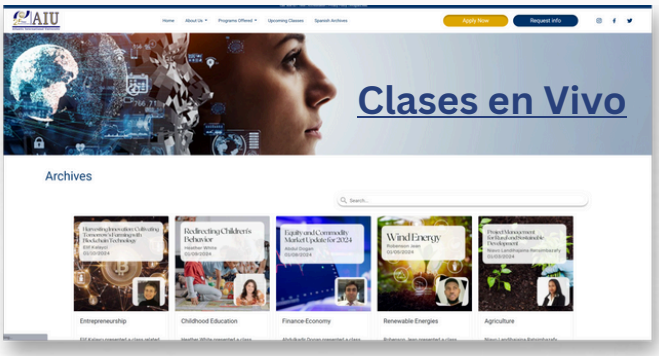


Tabla de contenido

Tabla de contenido

Clasificación de fallas

TIPOS DE FALLAS.

El sistema debe estar preparado para recuperarse no sólo de fallas puramente locales, como la aparición de una condición de desborde dentro de una transacción, sino también de fallas globales, como podría ser la interrupción del suministro eléctrico al CPU
Las fallas locales son las que afectan sólo a la transacción en donde ocurrió. Por el contrario las fallas globales, afectan a varias -y casi siempre a todas- las transacciones que se estaban efectuando en el momento de la falla, por lo cual tienen implicaciones importantes en el sistema.

Estas fallas pueden ser:

FALLA DEL SISTEMA

Por ejemplo interrupción del servicio eléctrico, estas afectan a todas Las transacciones que se estaban ejecutando pero no afectan a la base de datos. Las fallas de sistema se conocen también como caídas (crash) suaves. El problema aquí es que se pierda el contenido de memoria principal, en particular, las áreas de almacenamiento temporal o buffers. Si esto ocurre, no se conocerá el estado preciso de la transacción que se estaba ejecutando en el momento de la falla, esta transacción jamás se podrá completar con éxito por lo que será preciso anularla cuando se reinicie el sistema. Además, puede ocurrir que sea necesario volver a ejecutar algunas transacciones que sí se realizaron con éxito antes de la falla pero cuyas modificaciones no lograron efectuarse sobre la base de datos porque no lograron ser transferidas de los buffers de la base de datos a la base de Datos física (en disco).

FALLAS EN LOS MEDIOS DE ALMACENAMIENTO

Una falla de los medios de almacenamiento es un percance ene l cual se destruye físicamente alguna porción de la DB. La recuperación de una falla semejante implica en esencia cargar de nuevo la DB a partir de una copia de respaldo y utilizar después la bitácora para realizar de nuevo todas las transacciones terminadas desde que se hizo esa copia de respaldo.

No hay necesidad de anular las transacciones inconclusas en el momento de la falla, porque por definición todas las modificaciones de esas transacciones ya se anularon de todas maneras.

La parte de restauración de la utilería servirá entonces para recrear la DB después de una falla de los medios de almacenamiento a partir de una copia de respaldo especificada. Por ejemplo una falla en el controlador de disco o un aterrizaje de cabeza en el disco, estas fallas sí causan daños a la base de datos o a una porción de ella y afecta, al menos, a las transacciones que están haciendo uso de esa porción. Las fallas de los medios de almacenamiento se llaman caídas duras.

La Recuperación de una falla semejante implica, en esencia, cargar de nuevo la base de datos a partir de una copia de respaldo (database backup) y después utilizar la bitácora, o system log, para realizar de nuevo todas las transacciones terminadas desde que se hizo esa copia para respaldo. No hay necesidad de anular todas las transacciones inconclusas en el momento de la falla, porque por definición esas transacciones ya se anularon (se destruyeron) de todas maneras.

FALLAS POR CATÁSTROFES

Por ejemplo terremotos, incendios, inundaciones, etc. Su tratamiento es similar al de fallas de los medios. La principal técnica para manejar este tipo de fallas es la del database backup . Como se mencionó anteriormente, este es un respaldo periódico que se hace de la base de datos. Después de una caída de esta índole el BASES DE DATOS MIS 308 3 sistema se restaura recargando la base de datos con la copia del último respaldo y recreando la base de datos mediante la bitácora o system log.

ERRORES DEL SISTEMA

Como realizar operaciones que causen un overflow de un entero o la división por cero, así mismo puede ocurrir que se pasen valores erróneos a algún parámetro o que se detecte un error en la lógica de un programa, o que sencillamente no se encuentren los datos del programa. Además, en algunos ambientes de desarrollo el usuario puede explícitamente interrumpir una transacción durante su ejecución (por ejemplo: usando el control_C in VAX/VMS o en UNIX).

APLICACIÓN DEL CONTROL DE CONCURRENCIA

Que ocurre por ejemplo cuando una transacción viola las reglas de serialización o cae en abrazo mortal o interbloqueo

Modelo de transacciones

Los sistemas que tratan el problema de control de concurrencia permiten que sus usuarios asuman que cada una de sus aplicaciones se ejecutan atómicamente, como si no existieran otras aplicaciones ejecutándose concurrentemente.

Esta abstracción de una ejecución atómica y confiable de una aplicación se conoce como una transacción. Un algoritmo de control de concurrencia asegura que las transacciones se ejecuten atómicamente controlando la intercalación de transacciones concurrentes, para dar la ilusión de que las transacciones se ejecutan serialmente, una después de la otra, sin ninguna intercalación. Las ejecuciones intercaladas cuyos efectos son los mismos que las ejecuciones seriales son denominadas serializables y son correctos ya que soportan la ilusión de la atomicidad de las transacciones.

El concepto principal es el de transacción. Informalmente, una transacción es la ejecución de ciertas instrucciones que accesan a una base de datos compartida. El objetivo del control de concurrencia y recuperación es asegurar que dichas transacciones se ejecuten atómicamente, es decir:

Cada transacción accede a información compartida sin interferir con otras transacciones, y si una transacción termina normalmente, todos sus efectos son permanentes, en caso contrario no tiene afecto alguno. Una base de datos está en un estado consistente si obedece todas las restricciones de integridad (significa que cuando un registro en una tabla haga referencia a un registro en otra tabla, el registro correspondientes debe existir) definidas sobre ella. Los cambios de estado ocurren debido a actualizaciones, inserciones y supresiones de información. Por supuesto, se quiere asegurar que la base de datos nunca entre en un estado de inconsistencia. Sin embargo, durante la ejecución de una transacción, la base de datos puede estar temporalmente en un estado inconsistente. El punto importante aquí es asegurar que la base de datos regresa a un estado consistente al fin de la ejecución de una transacción.

PROPIEDADES FUNDAMENTALES DE UNA TRANSACCIÓN:

1. Atomicidad Se refiere al hecho de que una transacción se trata como una unidad de operación.

2. Por lo tanto, o todas las acciones de la transacción se realizan o ninguna de ellas se lleva a cabo. La atomicidad requiere que si una transacción se interrumpe por una falla, sus resultados parciales sean anulados.

3. Consistencia La consistencia de una transacción es simplemente su correctitud. En otras palabras, una transacción es un programa correcto que lleva a la base de datos de un estado consistente a otro con la misma característica. Debido a esto, las transacciones no violan las restricciones de integridad de una base de datos.

4. Aislamiento Una transacción en ejecución no puede revelar sus resultados a otras transacciones concurrentes antes de finalizar.

5. Más aún, si varias transacciones se ejecutan concurrentemente, los resultados deben ser los mismos que si ellas se hubieran ejecutado de manera secuencial.

6. Permanencia Es la propiedad de las transacciones que asegura que una vez que una transacción finaliza exitosamente, sus resultados son permanentes y no pueden ser borrados de la base de datos por alguna falla posterior.

7. Por lo tanto, los sistemas manejadores de base de datos aseguran que los resultados de una transacción sobrevivirán a fallas del sistema. Esta propiedad motiva el aspecto de recuperación de base de datos, el cual trata sobre cómo recuperar la base de datos a un estado consistente donde todas las acciones que han finalizado con éxito queden reflejadas en la base.

8. En esencia, lo que se persigue con el procesamiento de transacciones es, por una parte obtener una transparencia adecuada de las acciones concurrentes a una base de datos y por otra, manejar adecuadamente las fallas que se puedan presentar en una base de datos.

9. La mayoría de medianas y grandes compañías modernas utilizan el procesamiento de transacciones para sus sistemas de producción, y es tan imprescindible que las organizaciones no pueden funcionar en ausencia de él. El procesamiento de transacciones representa una enorme y significativa porción del mercado de los sistemas informáticos (más de cincuenta billones de dólares al año) y es, probablemente, la aplicación simple más amplia de las computadoras. Además, se ha convertido en el elemento que facilita el comercio electrónico.

Como puede percibirse, el procesamiento de transacciones es una de las tareas más importantes dentro de un sistema de base de datos, pero a la vez, es una de las más difíciles de manejar debido a diversos aspectos, tales como:

10. Confiabilidad Puesto que los sistemas de base de datos en línea no pueden fallar

11. Disponibilidad Debido a que los sistemas de base de datos en línea deben estar actualizados correctamente todo el tiempo.

12. Tiempos de Respuesta En sistemas de este tipo, el tiempo de respuesta de las transacciones no debe ser mayor a doce segundos.

13. Throughput Los sistemas de base de datos en línea requieren procesar miles de transacciones por segundo.

14. Atomicidad En el procesamiento de transacciones no se aceptan resultados parciales.

15. Permanencia No se permite la eliminación en la base de datos de los efectos de una transacción que ha culminado con éxito.

Control de concurrencia en Bases de Datos

El control de transacciones concurrentes en una base de datos brinda un eficiente desempeño del Sistema de Base de Datos, puesto que permite controlar la ejecución de transacciones que operan en paralelo, accesando a información compartida y, por lo tanto, interfiriendo potencialmente unas con otras.

El hecho de reservar un asiento en una avión mediante un sistema basado en aplicaciones web, cuando decenas de personas en el mundo pueden reservarlo también, nos da una idea de lo importante y crucial que es el control de concurrencia en un sistema de base de datos a mediana o gran escala. Otro ejemplo en el que podemos observar la incidencia del control de concurrencia en el siguiente: en una Base de Datos bancaria podría ocurrir que se paguen dos cheques en forma simultánea sobre una cuenta que no tiene saldo suficiente para cubrirlos en su totalidad, esto es posible evitarlo si se tiene un control de concurrencia. La Concurrencia en las Bases de Datos es de suprema importancia en los sistemas de información, ya que evita errores en el momento de ejecutar las diferentes transacciones.

PROCESAMIENTO DE LAS TRANSACCIONES

Así como el control de concurrencia, la recuperación de fallas está íntimamente ligada al procesamiento de las transacciones. Recordemos que una transacción tiene la cualidad de ser atómica a pesar de que puede estar compuesta de varias operaciones, la atomicidad se controla con la llegada al COMMIT. Si una transacción no sufrió ningún problema y se pudo ejecutar Completa, entonces el DBMS debe "comprometerse" a hacer permanentes los cambios que la transacción hizo sobre la base de datos ya que ésta debió quedar en un estado consistente. La atomicidad también ocurre en caso de que la transacción sufra algún problema que le impida ejecutarse. En este caso, la operación ROLLBACK señala el término no exitoso de la transacción, indicando al DBMS que "algo" debió salir mal, advierte que la base de datos puede estar en un estado inconsistente y que todas las modificaciones, que la transacción haya hecho Hasta el momento, deben "retroceder" o anularse

ROLLBACK DE LAS TRANSACCIONES

Si una transacción falla, por alguna razón, después de la actualización de la base de datos es necesario "anularla" (RollBack o UNDO). Cualquier valor del data ítem que haya sido cambiado por la transacción debe volver a su valor previo.

Si una transacción T es anulada (rollback), cualquier transacción S, que en el ínterin, haya leído un valor de un data ítem X modificado por T, entonces S también debe anularse (rollback).

De igual forma, si una transacción R ha leído un valor de un data ítem Y que ha sido modificado por S, entonces R también se debe anular y así sucesivamente.

Este fenómeno se conoce como rollback en cascada. El rollback en cascada puede ser muy costoso y es por ello que los mecanismos de recovery han catalogado a este fenómeno como indeseable o nunca requerido.

CONCEPTOS DE RECOVERY

La recuperación de las fallas de transacciones significa que la base de datos se debe restaurar desde algún estado considerado correcto del pasado - lo más cercano posible al momento de la falla. Para lograr esto el sistema debe mantener (externamente a la b/d) la información de todo lo que afecta a los data ítems de la base de datos . A esto se le llama Bitácora o system log como se mencionó anteriormente. Vamos a considerar las dos principales técnicas de recovery debido a fallas no catastróficas. A saber:

Deferred Update (o actualización diferida), también conocida como NO UNDO/REDO: esta técnica actualiza la base de datos sólo después de que la transacción ha llegado a su COMMIT. Antes de la llegada al COMMIT, todas las modificaciones son guardadas en un área de trabajo de la transacción local. Durante el COMMIT, las actualizaciones primero que nada son guardadas en el system log y luego se guardan en la base de datos. Si la transacción falla antes de llegar al commit no se guarda ningún cambio en la base de datos, de manera que no es necesario hacer ningún UNDO. Si la transacción llegó al commit grabó en el system log pero no grabó en la base de datos, es necesario hacer un REDO, es decir, grabar los efectos de la transacción que se encuentran en el log sobre la base de datos.

Inmediate Update (actualización inmediata) conocida también como UNDO/REDO: esta técnica propone que algunas operaciones actualicen la base de datos antes de que la transacción llegue al commit.
Estas actualizaciones son guardadas en el system log a nivel de disco antes de ser aplicadas a la base de datos. Si una transacción falla después de grabar algunos cambios en la base de datos y antes de llegar al commit, se debe aplicar un UNDO para anular los efectos sobre la base de datos, es decir, la transacción debe aplicar un ROLLBACK y hacer un REDO para grabar los valores anteriores a la falla.
Una variación de este algoritmo UNDO/REDO consiste en permitir que todos los cambios de hace la transacción antes de llegar al commit afecten a la base de datos y si ocurre una falla sólo se requiere un UNDO de manera que esta técnica también se le conoce como UNDO/NO REDO.

TÉCNICAS BASADAS EN ACTUALIZACIÓN DIFERIDA

La idea que manejan estas técnicas consiste en postergar o diferir cualquier actualización a la base de datos hasta que la transacción complete su ejecución exitosamente y llegue a su COMMIT. Durante la ejecución de la transacción las actualizaciones son grabadas en el log y en el área de trabajo de la transacción.

Cuando la transacción llega a su COMMIT, el log es forzado a escribirse en disco y luego las actualizaciones se reflejan en la base de datos. Si la transacción falla antes de llegar a su COMMIT no es necesario aplicar el UNDO a ninguna operación pues la transacción no ha afectado a la base de datos de ninguna manera. El protocolo típico de la actualización diferida se define como sigue:

1. Una transacción no puede hacer cambios a la base de datos hasta que llegue a su COMMIT.

2. Una transacción no llega a su COMMIT hasta que todas las operaciones de actualización hayan sido registradas en el log y el log haya sido forzado a escribirse en disco. Si la transacción falla después de llegar a su COMMIT pero antes de que los cambios sean reflejados en la base de datos, basta con aplicar el algoritmo REDO según las operaciones y los valores de los data ítems que aparecen en el log. Por eso a esta técnica, a veces, se le conoce como NO UNDO/REDO.

RECOVERY UTILIZANDO ACTUALIZACIÓN DIFERIDA EN AMBIENTES MONOUSUARIO.

En un ambiente como este el algoritmo de recovery es relativamente sencillo. El algoritmo conocido como RDU_S (recovery using referred update in a single-user environment) usa un procedimiento REDO para rehacer ciertas operaciones de escritura (write-ítem) sobre data ítems de la base de datos.

PROCEDURE RDU_S usa dos listas de transacciones: las transacciones que han llegado a su COMMIT desde el último checkpoint y las transacciones activas (por lo menos, una transacción cae en esta categoría pues el ambiente es monousuario).

Aplica el algoritmo REDO a todas las operaciones que han escrito (writeítem) sobre los ítems, de las transacciones que han llegado a su COMMIT según el orden en el que ellas aparecen en el log. Luego reinicia todas las transacciones activas.

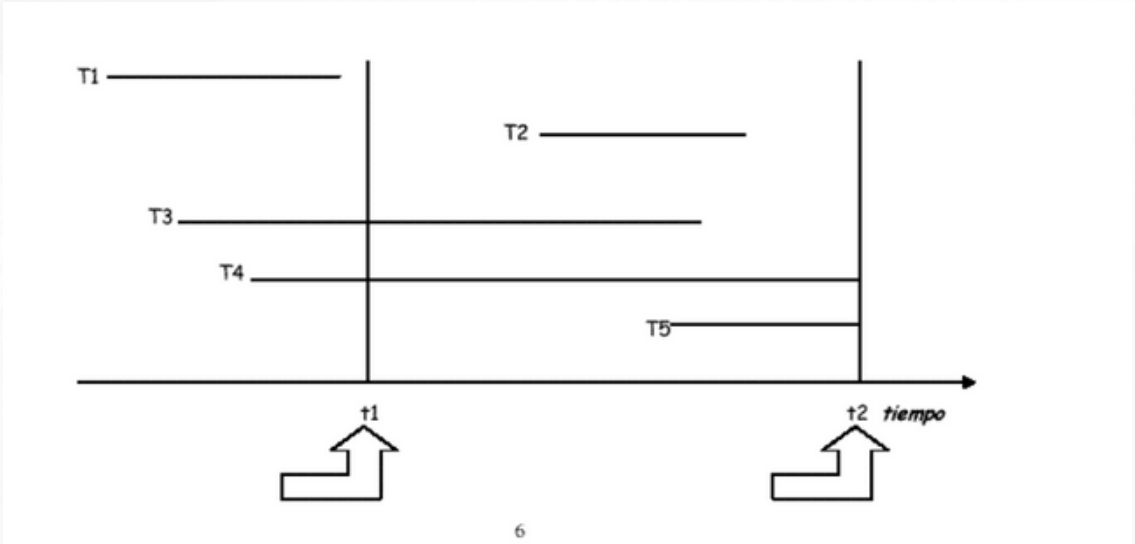
El REDO es definido de la siguiente manera: REDO(WRITE_OP) rehacer una operación de escritura sobre un data ítem WRITE_OP consiste en examinar sus entradas al log [write_ítem,T,X,new_value] y colocar como valor del data ítem X el valor que aparece en new_value, considerado el AFIM (valor del data ítem después de la actualización, AFter Image). La operación REDO se aplica para buscar lo idéntico. Esto es que, si ella se ejecuta una y otra vez, el resultado debe ser equivalente a que si se ejecutara una sola vez.

RECOVERY UTILIZANDO ACTUALIZACIÓN DIFERIDA EN AMBIENTES MULTIUSUARIOS

Para ambientes multiusuarios con control de concurrencia, el recovery puede hacerse más complejo dependiendo del protocolo de control de concurrencia usado. En general, a mayor grado de concurrencia que se desea llegar más difícil se vuelve el proceso de recovery. Supongamos que el control de concurrencia se logra con el protocolo de dos fases asignando cerrojos (locks) a los data ítem requeridos por una transacción antes de que ella comience a ejecutarse. Para combinar el método de actualización diferida para recovery con la técnica para el control de la concurrencia, es necesario mantener todos los cerrojos sobre los ítems activos hasta que la transacción llegue a su COMMIT. Después de ello, los cerrojos se remueven.

A continuación se da el algoritmo de recovery llamado RDU_M (recovery deferred update in a multiuser environment) asumiendo que las entradas del checkpoint se encuentran en el log.

PROCEDURE RDU_M Usa dos listas de transacciones mantenidas por el sistema: las transacciones que llegaron a su COMMIT (T) desde el último checkpoint y las transacciones activas (T´). Aplica el algoritmo REDO a todas las operaciones que han escrito (write-ítem) sobre los ítems, de las transacciones que han llegado a su COMMIT según el orden en el que ellas aparecen en el log. Las transacciones que están activas y que no han llegado a su COMMIT deben ser canceladas y sometidas nuevamente a ejecución.



6

En este ejemplo, cuando el checkpoint ocurre en el tiempo t1, T1 ha llegado a su COMMIT mientras que T3 y T4 no. T2 y T3 llegan a su COMMIT antes de que en t2 ocurra la caída del sistema, sin embargo ni T4 ni T5 logran terminar.

Según el método de RDU_M no es necesario hacer REDO a la transacción T1 pues esto ocurrió antes del último checkpoint. Las operaciones de escritura de T2 y T3 deben ser rehechas, debido a que estas llegaron a su COMMIT después del último checkpoint.
Tanto T4 como T5 son anuladas (no se toman en cuenta) debido a que ninguna de sus operaciones de escritura afectaron a la base de datos bajo el protocolo de actualización diferida.

OPERACIONES QUE NO AFECTAN A LA BASE DE DATOS.

En general, no todas las operaciones de las transacciones afectan a la base de datos: generación e impresión de mensajes o reportes son ejemplos de ello.
Si una transacción falla antes de completarse, el reporte no debe ser tomado en cuenta, esto debe ocurrir sólo después de que la transacción haya llegado a su COMMIT.
Esto suele controlarse dejando este tipo de acciones en una cola de batch jobs. Los batch jobs son ejecutados sólo si la transacción es exitosa si no lo es se cancelan.

TÉCNICAS BASADAS EN ACTUALIZACIÓN INMEDIATA.

UNDO/REDO RECOVERY UTILIZANDO ACTUALIZACIÓN INMEDIATA EN AMBIENTES MONOUSUARIO.

PROCEDURE RIU_S usa dos listas de transacciones: las transacciones que han llegado a su COMMIT desde el último checkpoint y las transacciones activas (por lo menos, una transacción cae en esta categoría ges el ambiente es monousuario). Aplica el algoritmo UNDO a Todas las operaciones que han escrito (write-ítem) sobre los ítems, de las transacciones activas que aparecen en el log. Aplica el algoritmo REDO a todas las transacciones que han llegado a su COMMITsegún el orden en el que ellas aparecen en el log. El UNDO se define como: UNDO(WRITE_OP) deshacer una operación de escritura sobre un data ítem WRITE_OP consiste en examinar sus entradas al log [write_ítem,T,X,old_value,new_value] y colocar como valor del data ítem X el valor que aparece en old_value, considerado el BFIM (valor del data ítem antes de la actualización, Before Image). Deshacer un número de operaciones de escritura de una o más transacciones desde el log implica proceder en orden inverso a como estas operaciones fueron grabadas en el log.

UNDO/REDO RECOVERY UTILIZANDO ACTUALIZACIÓN INMEDIATA EN AMBIENTES MULTIUSUARIOS.

PROCEDURE RIU_M usa dos listas de transacciones: las transacciones que han llegado a su COMMIT desde el último checkpoint y las transacciones. Activa el algoritmo UNDO a todas las operaciones que han escrito (write-ítem) sobre los ítems, de las transacciones activas que aparecen en el log. Las operaciones deben ser deshechas en orden inverso a como aparecen en el log. Aplica el algoritmo REDO a todas las transacciones que han llegado a su COMMIT según el orden en el que ellas aparecen en el log.

RECOVERY EN TRANSACCIONES MULTI-BASES DE DATOS.

Hasta ahora hemos asumido que las transacciones que se ejecutan acceden una sola base de datos.
Existen casos de transacciones que trabajan con varias bases de datos, estas son llamadas transacciones multi-base de datos. Incluso los datos pueden estar almacenados y manejados por DBMS, no sólo relacionales, sino también de redes o jerárquicos.
En este caso cada DBMS envuelto en una transacción multi-base de datos tendrá su propia técnica de recovery y su propio controlador de transacciones separado de los otros DBMS.
Para mantener la atomicidad de una transacción multi-base de datos es necesario tener un mecanismo de recovery de dos niveles y un manejador global de recovery, llamado a veces, el coordinador en adición a los manejadores locales.

El coordinador sigue un protocolo llamado protocolo del commit en dos fases.

Definido como: Fase 1: Cuando todas las bases de datos participantes en la transacción han concluido su parte señalan esto al coordinador. Entonces el coordinador envía un mensaje de "prepararse para el commit" a todos los participantes (DBMS´s). Cada participante que recibe el mensaje se ve forzado a escribir su log en el disco y enviar un "listo para el commit" o un "OK" al coordinador. Si la escritura en disco falla o un participante falla, este envía un "no puedo hacer commit" o un "no OK" al coordinador. Si el coordinado no recibe réplica de ningún tipo después de un intervalo de tiempo, asume un "no OK" como respuesta.

Fase 2: Si todos los participantes responden OK la transacción es exitosa y el coordinador envía el commit de la transacción a todos los participantes. Si algún participante respondió "no OK" el coordinador manda un rollback o un UNDO a todos los participantes de la transacción, esto se hace haciendo el UNDO según como lo estipule el log.

Recuperación por bitácora

El sistema mantiene una bitácora o system log en cinta, por lo general, o en disco en donde se detallan todas las operaciones de actualización y los valores iniciales y finales de cada objeto. Por lo tanto, si resulta necesario anular alguna modificación específica, el sistema puede utilizar la entrada correspondiente de la bitácora para restaurar el valor original del objeto modificado.

SYSTEM LOG O BITÁCORA.

Para recuperarse de las fallas de las transacciones, el sistema mantiene un log (llamado journal o periódico) que mantiene el curso de todas las transacciones que afectan los datos ítems de la base de datos. Al conjunto de log se le conoce como system log.

Esta información es mantenida en disco (memoria secundaria) de manera que sólo puede estar afectada, eventualmente, por fallas en medios de almacenamiento. Periódicamente, los log son respaldados en cintas para protegerlos contra fallas por catástrofes. Los log tienen una serie de entradas que se detallan a continuación. Las T se refieren a un identificador único de cada transacción. 1. [start-transaction,T]: registra que la transacción ha comenzado. 2. [write-ítem,T,X,old,new-value]: registra que la transacción T ha cambiado el valor del data ítem X del valor viejo old al valor nuevo newvalue. 3. [read-ítem,T,X]: registra que la transacción T ha leído el valor del dato ítem X de la base de datos. 4. [commit,T]: registra que la transacción T ha terminado exitosamente y que los efectos pueden ser grabados permanentemente (committed, comprometidos) en la base de datos. 5. [abort,T]: registra que la transacción T ha sido abortada.

Puntos de verificación

Un registro checkpoint se graba periódicamente en el log justo en el punto en donde se guardó, en la base de datos en el disco, el efecto de todas las operaciones de escritura hechas por las transacciones que terminaron exitosamente (llegaron al commit). Si hay una falla, las transacciones que se completaron antes del checkpoint no deben ser rehechas en el recovery (no debe aplicarse un REDO).

El DBMS debe decidir cada cuando hace un checkpoint. Esto puede ser cada m minutos o cada N transacciones completadas exitosamente, estos son parámetros del sistema.

Hacer un checkpoint implica:

1. Suspender temporalmente la ejecución de todas las transacciones.
2. Forzar la escritura de todas las actualizaciones de las operaciones de las transacciones que llegaron al commit que están en los buffers de memoria principal al disco.
3. Escribir un registro checkpoint en el log y forzar la escritura del log en el disco.
4. Reanudar la ejecución de las transacciones.

Un registro checkpoint puede incluir una lista de las transacciones activas (aquellas que no se han terminado de ejecutar). El punto crítico de una falla del sistema es que se pierde el contenido de la memoria principal. Por tanto, ya no se conocerá el estado preciso de la transacción que se estuviera realizando en el momento de la falla; esa transacción jamás se podrá completar con éxito, por lo cuál será preciso anularla cuando se reinicie el sistema.

Cada cierto intervalo previamente establecido el sistema establece un “punto de revisión” de manera automática. El establecimiento de un punto de revisión implica:

•a) Grabar físicamente el contenido de los buffers en la DB física;

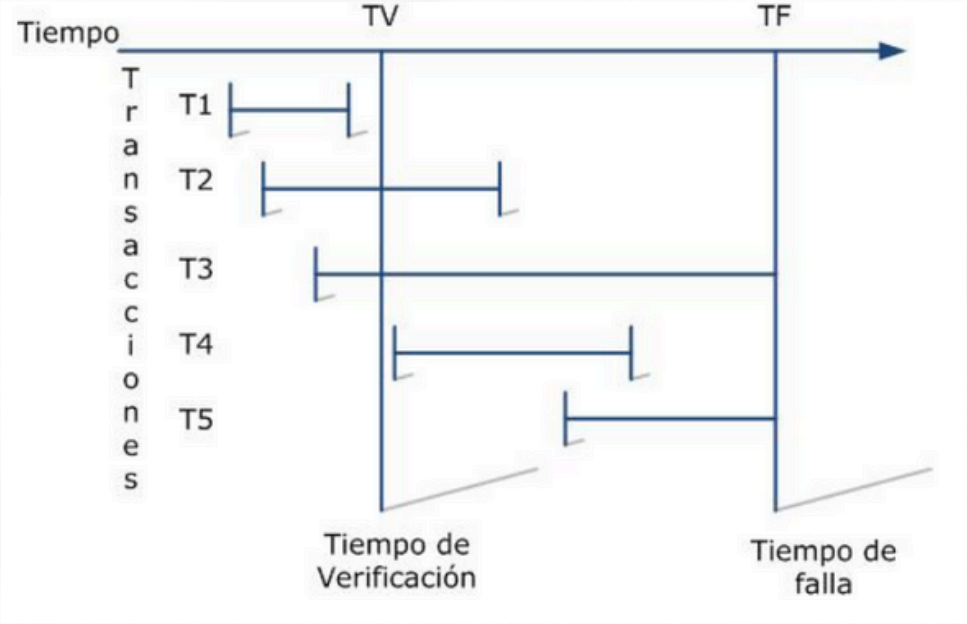
•b) Y grabar físicamente un registro de punto de revisión especial en la bitácora física.

El registro de punto de revisión incluye una lista de todas las transacciones que se estaban realizando en el momento de establecerse el punto de revisión. Para comprender la forma como se utiliza esta información deberá leerse de la siguiente manera:

Introduction to computer organization and architecture by Antreas Naziris

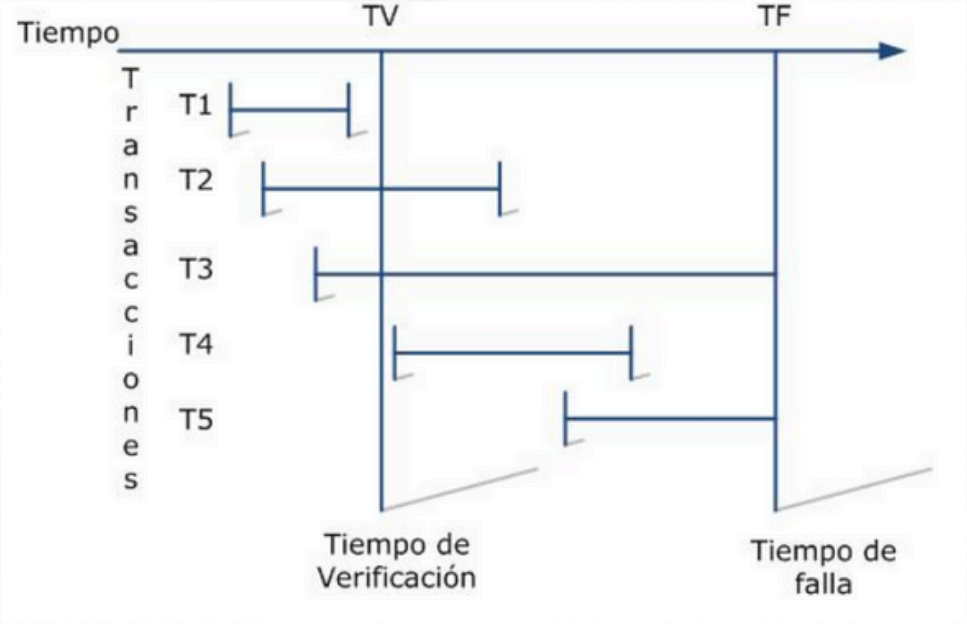
Introduction to computer engineering by Priyanka Israni

General Basics of Cloud technology with azure by Jay Vijayasimha



Se presentó una falla en el sistema en el momento tf.

• El punto de verificación más reciente antes de tf se tomó en el momento tv.
• Las transacciones de tipo T1 se completaron antes del tiempo tv.
BASES DE DATOS MIS 308
18
• Las transacciones de tipo T2 se iniciaron antes del tiempo TV y se completaron después del tiempo TV y antes del tiempo tf.
• Las transacciones de tipo T3 también se iniciaron antes del tiempo TV pero no se completaron antes del tiempo tf.
• Las transacciones de tipo T4 se iniciaron después del tiempo TV y se completaron antes del tiempo TV.
• Por último, las transacciones de tipo T5 también se iniciaron después del tiempo TV pero no se completaron antes del tiempo tf.
Por tanto, ene l momento del reinicio el sistema efectúa el siguiente procedimiento a fin de identificar las transacciones de los tipos T2-T5:



Se presentó una falla en el sistema en el momento tf.

• El punto de verificación más reciente antes de tf se tomó en el momento tv.
• Las transacciones de tipo T1 se completaron antes del tiempo tv.
BASES DE DATOS MIS 308
18
• Las transacciones de tipo T2 se iniciaron antes del tiempo TV y se completaron después del tiempo TV y antes del tiempo tf.
• Las transacciones de tipo T3 también se iniciaron antes del tiempo TV pero no se completaron antes del tiempo tf.
• Las transacciones de tipo T4 se iniciaron después del tiempo TV y se completaron antes del tiempo TV.
• Por último, las transacciones de tipo T5 también se iniciaron después del tiempo TV pero no se completaron antes del tiempo tf.
Por tanto, ene l momento del reinicio el sistema efectúa el siguiente procedimiento a fin de identificar las transacciones de los tipos T2-T5:

•1. Comenzar con 2 listas de transacciones, la lista ANULAR y la lista REPETIR. Igualar la lista ANULAR a la lista de todas las transacciones incluidas en el registro de punto de revisión. Dejar vacía la lista REPETIR.
•2. Examinar la bitácora hacia delante a partir del registro de punto de revisión.
•3. Si se encuentra una entrada de bitácora de “iniciar transacción” para la transacción T, añadir T a la lista ANULAR.
•4. Si se encuentra una entrada de bitácora de “comprender” para la transacción T, pasar esa transacción de la lista ANULAR a la lista REPETIR.
•5. Cuando se llegue al final de la bitácora, las listas ANULAR y REPETIR identificarán, respectivamente, las transacciones de los tipos T3 y T5 y la de los tipos T2 y T4.

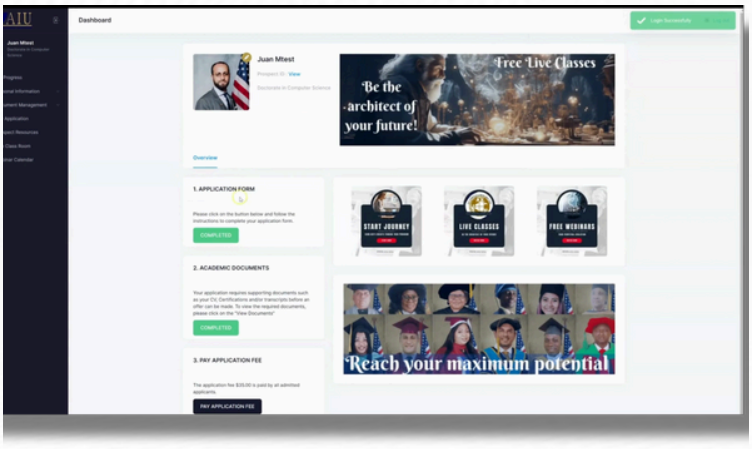
Enseguida el sistema revisará la bitácora hacia atrás, anulando todas las transacciones de la lista ANULAR. A continuación la revisará otra vez hacia delante, realizando de nuevo todas las transacciones de la lista REPETIR. Por último, una vez terminada toda esa actividad de recuperación, el sistema estará listo para aceptar trabajos nuevos.

¿Disfrutaste esta lectura?

Contáctanos

Solicitar Información

Demo del Campus Virtual



Galería de Graduados