Transacciones

Es una unidad lógica de procesamiento de la Base de Datos que incluye una o más operaciones de acceso a la Base de datos, que pueden ser de inserción, eliminación, modificación o recuperación.

Una transacción se inicia por la ejecución de un programa de usuario escrito en un lenguaje de manipulación de datos de alto nivel o en un lenguaje de programación (SQL, Java) y está delimitado por instrucciones begin transaction y end transaction.

Un programa de aplicación puede contener más de una transacción si contiene varios límites de transacción. Si las operaciones de B. D. de una transacción no la actualizan, sino que únicamente recuperan datos, se dice que la transacción es de sólo lectura.

Propiedades ACID

***Atomicidad:** Una transacción es una unidad atómica de procesamiento; o se ejecuta en su totalidad o no se ejecuta en absoluto.

***Conservación de la consistencia:** Una transacción está conservando la consistencia si su ejecución completa lleva a la B. D. de un estado consistente a otro.

***Aislamiento (Isolation)**: Una transacción debe aparecer como si estuviera ejecutándose de forma aislada a las demás. Es decir, la ejecución de una transacción no debe interferir con la ejecución de ninguna otra transacción simultánea.

***Durabilidad:** Los cambios aplicados a la B. D. por una transacción confirmada deben persistir en la B. D. Estos cambios no deben perderse por culpa de un fallo.

Estados posibles

•**Activo:** Estado desde que comienza la ejecución de la transacción hasta completar la última instrucción, o hasta que se produce un fallo.

•**Parcialmente finalizado:** Se alcanza en el momento posterior a que la transacción ejecuta su última instrucción.

•**Finalizado:** Se alcanza cuando la transacción finalizó su ejecución con éxito, y sus acciones fueron almacenadas correctamente en almacenamiento secundario.

•**Fallado:** A este estado se arriba cuando la transacción no puede continuar con su ejecución normal.

•**Abortado:** Este estado garantiza que una transacción fallada no ha producido ningún cambio en la Base de Datos, manteniendo la integridad de la información allí contenida. En este punto hay 2 opciones para seguir:

- Reiniciar la transacción: Esta acción se puede llevar a cabo cuando el error se produjo por el entorno y no por la lógica interna de la transacción. Una transacción reiniciada se considera una nueva transacción

- Cancelar definitivamente la transacción: Esta acción se lleva a cabo cuando se detecta un error interno en la lógica de la transacción. Este error sólo puede ser salvado si la transacción es redefinida.

Planificacion serie

Una planificación S es serie si, por cada transacción T, que participa en la planificación, todas las operaciones de T se ejecutan consecutivamente en la planificación; en caso contrario, se dice que la planificación no es serie. Por consiguiente, en una planificación serie, sólo hay una transacción activa al mismo tiempo.

Planificación no serie o concurrente

Es intercalada, y es correcta solo cuando su resultado es igual a que si se hubiese utilizado una secuencial. Debe equivaler a una planificación en serie.

Planificación serializable

Una planificación S es de n transacciones serializable si es equivalente a alguna planificación en serie de las mismas n transacciones. ¡A partir de n transacciones es posible generar n! planificaciones en serie y muchas más planificaciones no serie.

*Se pueden formar dos grupos disjuntos con las planificaciones no serie:

Aquellas que son equivalentes a una (o más) planificaciones en serie y, por tanto, serializables.

Aquellas que no son equivalentes a ninguna planificación en serie y, por tanto, no son serializables.

Decir que una planificación no serie S es serializable es equivalente a decir que es correcta, porque es equivalente a una planificación en serie, que se considera correcta.

¿Cuándo se considera que dos planificaciones son equivalentes?

Analizaremos la **equivalencia por conflicto**. Dos planificaciones son equivalentes por conflicto si el orden de cualquier par de operaciones en conflicto es el mismo en las dos planificaciones.

Dos planificaciones P y P' se denominan equivalentes en cuanto a conflictos cuando P' se logra a partir del intercambio de instrucciones no conflictivas de P. Una planificación P es serializable en cuanto a conflictos si es equivalente en cuanto a conflictos a una planificación en serie.

Planificación recuperable

Una planificación recuperable es aquella en la que para todo par de transacciones Ti y Tj tales que Tj lee elementos de dato que ha escrito previamente Ti, la operación de finalizar Ti aparece antes que la de Tj.

Una planificación sin cascada es aquella para la que todo par de transacciones Ti y Tj tales Tj lee un elemento de datos que ha escrito previamente Ti, la operación de finalizar de Ti aparece antes que la operación de lectura de Tj.

Fallos

En un sistema pueden producirse varios tipos de fallos. Consideraremos los siguientes tipos de fallos:

1.Fallo en la transacción: Hay dos tipos de errores que pueden hacer que una transacción falle:

•Error lógico. La transacción no puede continuar con su ejecución normal a causa de alguna condición interna, como una entrada incorrecta, desbordamiento, etc.

•Error del sistema: El sistema se encuentra en un estado no deseado como consecuencia del cual una transacción no puede continuar con su ejecución normal; sin embargo, se puede volver a ejecutar más tarde.

2.Caída del sistema: Un mal funcionamiento del hardware o un error en el software de la base de datos o del sistema operativo causa la pérdida del contenido de la memoria volátil y aborta el procesamiento de una transacción.

3.Fallo del disco: Algunos bloques de disco pueden perder sus datos por un mal funcionamiento de lectura o escritura, o por un fallo de una cabeza de lectura/escritura.

Fallos: Ya que reejecutar o no una transacción fallida NO asegura la consistencia de la Base de Datos. Para garantizar la consistencia de la base de datos y la atomicidad de las transacciones a pesar de los fallos, existen algoritmos de recuperación que constan de dos partes:

1. Acciones que se ejecutan durante el procesamiento normal de las transacciones, que realizan acciones que permiten recuperar el estado de consistencia de la Base de Datos ante un fallo.

2. Acciones que se activan luego de detectarse un error en el procesamiento de las transacciones, que permiten recuperar el estado de consistencia ante un fallo.

Control de concurrencia

· **Bloqueo compartido:** Debe ser realizado por una transacción para leer un dato que no modificará. Mientras dure este bloqueo, se impide que otra transacción pueda escribir el mismo dato. Un bloqueo compartido puede coexistir con otro bloqueo compartido sobre el mismo dato.

· **Bloqueo exclusivo:** Se genera cuando una transacción necesita escribir un dato. Este bloqueo garantiza que otra transacción no pueda utilizar ese dato (ni lectura ni escritura). Un bloqueo exclusivo es único.

Protocolos de bloqueo: conjunto de reglas que indican el momento en que una transacción puede bloquear y desbloquear cada uno de los elementos de datos. Restringen el número de planificaciones posibles. Hay varias alternativas, analizaremos solamente 2:

• **Bloqueo de dos fases:** cada transacción realiza las peticiones de bloqueo y desbloqueo en dos fases:

1. Fase crecimiento: Una transacción puede obtener bloqueos pero no puede liberarlos.

2. Fase de decrecimiento: Una transacción puede liberar bloqueos pero no puede obtener ninguno nuevo.

• **Basado en marcas temporales:** Asegura que todas las operaciones LEER y ESCRIBIR conflictivas se ejecuten en el orden de las marcas temporales. Esto asegura secuenciabilidad en cuanto a conflictos.

Granularidad

Se llama granularidad de los elementos de datos al tamaño de dichos elementos. Lo que se necesita es un mecanismo que permita definir múltiples niveles de granularidad. Se puede construir uno permitiendo que los elementos de datos sean de varios tamaños y definiendo una jerarquía de granularidades de los datos, en la cual las granularidades pequeñas están anidadas en otras más grandes.

El término granularidad fina indica elementos de tamaño pequeño, en tanto que granularidad gruesa designa elementos grandes.

Deben considerarse ventajas y desventajas a la hora de elegir el tamaño de los elemento de datos. Cuanto mayor sea el tamaño del elemento de datos, menor será el grado de concurrencia permitido. Por otro lado, cuanto menor sea el tamaño del elemento, más elementos habrá en la Base de Datos. Cada elemento estará asociado a un bloqueo, el sistema tendrá un mayor número de bloqueos activos que deberá manejar el gestor de bloqueos. Se efectuarán más operaciones de bloquear y desbloquear, dando lugar a una mayor sobrecarga. Además, se requerirá más espacio de almacenamiento para la tabla de bloqueos.

¿Cuál es el mejor tamaño para los elementos?

Depende del tipo de las transacciones implicadas. a la hora de elegir el tamaño de los elemento de datos.

Interbloqueo

Existe interbloqueo cuando existe un conjunto de transacciones, tal que toda transacción del conjunto esta esperando un elemento de datos bloqueado por otra transaccion del conjunto.

Los interbloqueos son absolutamente preferibles a los estados inconsistentes, ya que se pueden tratar haciendo retroceder las transacciones; mientras que los estados inconsistentes producen problemas en el mundo real que el sistema de Base de Datos no los puede manejar.

Prevencion de interbloqueos

Una forma de evitar el interbloqueo es utilizando un protocolo de prevención de interbloqueo, que se utiliza en el bloqueo de dos fases. Requiere que cada transacción bloquee con antelación todos los elementos que necesita (algo que normalmente no es práctico); si alguno de los elementos no puede obtenerse, ninguno de los elementos se bloquea. La transacción espera y después intenta de nuevo bloquear todos los elementos que necesita. Esta solución limita la concurrencia.

Metodo bitacora

Registro histórico que consiste en registrar, en un archivo auxiliar y externo a la Base de Datos, todos los movimientos producidos por las transacciones antes de producir los cambios sobre la Base de Datos misma. La estructura del archivo es simple, cada transacción debe indicar su comienzo, su finalización y cada una de las acciones que modifiquen la BD (operaciones de escrituras), también debe agregarse cuando una transacción es abortada.

En caso de producirse un error, se tiene constancia de todos los movimientos efectuados. El gestor del SGBD es el encargado de administrar el archivo de bitácora. Para esto identifica cada transacción asignándole un número correlativo único.

Se debe garantizar que primero se graben los buffers correspondientes al archivo de bitácora en memoria secundaria, antes de grabar en disco los buffers de la Base de Datos.

Cuando en la bitácora se registra un comienzo, algunos cambios en la Base de Datos, y luego no se registra finaliza u aborta, la transacción debe abortarse.

Para todas las transacciones que alcanzan el estado de finalizada, el trabajo registrado en bitácora es innecesario. Sólo cuando se produce un error, se deben activar las acciones que subsanan dicho error, utilizando para ello el registro en bitácora.

Presenta dos alternativas para implementar el método de recuperación:
Modificación diferida de la Base de Datos: Todas las escrituras en disco de las actualizaciones de la Base de Datos se demoran hasta que la transacción alcance el estado de finalizada en bitácora. Ventaja: Si una transacción activa falla, se puede asegurar que los cambios producidos por la transacción no tuvieron impacto sobre la Base de Datos. El fallo es ignorado y la BD permanece íntegra. Este método utiliza un procedimiento de recuperación, denominado REHACER. Después de ocurrir un fallo, el subsistema de recuperación consulta la bitácora para determinar las transacciones que deben rehacerse. Las transacciones tienen una condición de idempotencia que asegura que aunque se reejecute n veces una transacción, se genera siempre el mismo resultado sobre la BD.

Modificación inmediata de la Base de Datos: Ante un cambio, se guarda en la Bitácora y luego inmediatamente se lleva a la Base de Datos. Los cambios en la Base de Datos se realizan a medida que se producen en la transacción que se está ejecutando, siempre bajo la consigna que indica que un cambio se registra primero en bitácora para luego grabarse en la Base de Datos. Existe un procedimiento DESHACER, cuyo efecto es retrotraer la Base de Datos al estado que tenía antes de comenzar la transacción. DESHACER vuelve a la Base de Datos los valores viejos de cada dato modificado por Ti registrados en bitácora. En resumen, si se produce un error, se debe REHACER toda transacción de la bitácora que haya finalizado, y se debe DESHACER toda transacción iniciada que no haya finalizado.

Paginación sombra

Considera que la B. D. está compuesta de un número n de páginas de disco de tamaño fijo (o bloques de disco) para la recuperación.

Se construye un directorio con n entradas, donde la entrada i apunta a la página i de la B. D. en disco.

El directorio se guarda en la memoria principal si no es demasiado grande, y todas las referencias (lecturas y escrituras) a las páginas de la B. D. en disco pasan por él.

Cuando empieza la ejecución de una transacción, el directorio actual (cuyas entradas apuntan a las páginas más recientes o últimas de la B. D. en disco) se copia en un directorio sombra. El directorio sombra se guarda en el disco, mientras que la transacción utiliza el directorio actual. Durante la ejecución de la transacción, el directorio sombra nunca se modifica. Cuando se efectúa una operación write_item, se crea una nueva copia de la página de B. D. modificada, pero no se sobrescribe la copia antigua. En vez de ello, la nueva página se escribe en algún bloque de disco que no se ha utilizado anteriormente. La entrada del directorio actual se modifica de modo que apunte al nuevo bloque de disco, mientras que el directorio sombra no se modifica y sigue apuntando al antiguo bloque de disco, no modificado. Se guardan dos versiones de las páginas actualizadas.

La versión antigua es referenciado por el directorio sombra y la versión nueva por el directorio actual. Para recuperarse ante un fallo durante la ejecución de una transacción, es suficiente con liberar las páginas modificadas de la B. D. y descartar el directorio actual.

El estado de la B. D. anterior a la ejecución de la transacción está disponible a través del directorio sombra, y ese estado se recupera restableciendo el directorio sombra.

Desventajas:

•La tarea de dividir la Base de Datos en páginas puede resultar compleja.

•Las páginas actualizadas de la BD cambian de lugar en el disco.

•El costo de escribir directorios sombra en disco cada vez que se confirman las transacciones es significativo.

•Es necesario definir algoritmos denominados recolectores de basura que detecten nodos no referenciados y liberen el espacio ocupado no utilizado.

•La operación de migración entre los directorios actuales y sombra debe ser implementada como una operación atómica.