

Capítulo 15 – SISTEMA DE RECUPERACIÓN

Esquema de recuperación → responsable de la restauración de la bd al estado consistente previo al fallo.

15.1.- CLASIFICACIÓN DE LOS FALLOS

- ♦ **Fallo en la transacción:**
 - **Error lógico:** condición interna a la T (entrada incorrecta, datos no encontrados...)
 - **Error del sistema:** sistema en estado que no permite continuar la T (interbloqueo)
- ♦ **Caída del sistema** → pérdida memoria volátil (Fallo de Hardware, Software o SO)
supuesto de fallo-parada (se pierde la volátil, pero la no volátil no se corrompe)
- ♦ **Fallo de disco** → Pérdida de un bloque del disco o fallo en la transferencia de datos.

Los **algoritmos de recuperación** constan de **2 partes**:

1. Acciones durante el procesamiento normal de las transacciones.
2. Acciones después de ocurrir el fallo.

15.2.- ESTRUCTURA DE ALMACENAMIENTO

TIPOS DE ALMACENAMIENTO

- ♦ Almacenamiento **Volátil**: No sobrevive a la caída del sistema (memoria RAM, caché, etc.)
- ♦ Almacenamiento **No Volátil**: Sobrevive a la caída del sistema (Discos, cintas, etc.)
- ♦ Almacenamiento **Estable**: La información Nunca se pierde, aunque teóricamente es imposible puede obtenerse una aproximación.

IMPLEMENTACIÓN DEL ALMACENAMIENTO ESTABLE

Replicando la información en varios medios de almacenamiento no volátil con modos de fallos independientes. Es necesario que los dos discos no estén en el mismo lugar, en previsión de posibles incendios o desastres similares.

Sistemas RAID → disposición redundante de discos independientes.

- ♦ Es necesario garantizar que las 2 copias son exactamente iguales y ninguna de ellas contiene errores en la transferencia.
- ♦ Una **transferencia** entre **memoria y disco** puede finalizar:
 - Con **éxito**: información almacenada sin problemas
 - Con **fallo parcial**: el bloque de destino contiene información incorrecta
 - Con **fallo total**: No se llega a escribir el bloque de destino
- ♦ Se tienen **2 bloques físicos por cada** bloque **lógico**:
 - ♦ **Discos con imagen** → ambos en el mismo lugar.
 - ♦ **Copia de seguridad remota** → uno local y el otro remoto.
- ♦ Para **verificar** una **transferencia** correcta se realizan las siguientes **operaciones**:
 - 1.- Se escribe la información en el primer bloque físico
 - 2.- Al completarse la primera escritura con éxito, se escribe en el 2º disco
 - 3.- Operación completada siempre que la 2ª escritura haya tenido éxito.

Durante estas operaciones se examinan los 2 bloques físicos para confirmar que ambos coinciden y no existe error, en caso de haber error se vuelve a escribir dicho bloque.

ACCESO A LOS DATOS

Los datos se transfieren de los bloques de **memoria intermedia** a los **bloques físicos** (del disco) y a la inversa, mediante las dos operaciones siguientes:

- ◆ **Entrada:** Transfiere el bloque físico a la memoria.
- ◆ **Salida:** Transfiere el bloque de la memoria al disco, reemplazando el bloque físico.
- Cada transacción *T* posee un **área de trabajo privada** en la cual se guardan copias de los elementos de datos accedidos y actualizados por *T*, dicha área se crea al crearse *T* y se elimina al finalizar o abortarse *T*.
- Al realizar *T*, funciones de lectura o escritura se comprueba que el elemento exista en su área de trabajo o sinó se copia desde el disco, mediante la operación **Entrada**.
- La operación de **Salida** al disco puede hacerse de manera periódica, por necesidades de espacio, o al finalizar todas las operaciones de *T*.

15.3.- RECUPERACIÓN Y ATOMICIDAD

A fin de evitar estados inconsistentes de la Base de Datos **no** deben realizarse **escrituras** en el disco **hasta** que la **transacción** no haya finalizado **con éxito** todas sus operaciones, consiguiéndose de esta manera la **atomicidad** y la salida de todas las modificaciones realizadas por una *T* comprometida han de **realizarse aunque** se produzcan **fallos**.

15.4.- RECUPERACIÓN BASADA EN EL REGISTRO HISTÓRICO

- El registro histórico mantiene todas las actividades de actualización de la Base de Datos, un *registro de actualización del registro histórico* describe una única escritura con los siguientes campos:
 - **Identificador de la Transacción.**
 - **Identificador del elemento de datos** que se escribe.
 - **Valor anterior** del elemento antes de la escritura.
 - **Valor nuevo** del elemento una vez escrito.
- Existen otros *registros* que mantienen otros sucesos significativos relativos a las transacciones:
 - ***T* iniciada.**
 - ***T* comprometida.**
 - ***T* abortada.**
- Los registros del registro histórico deben ser actualizados antes de que las operaciones de escritura se graben sobre la Base de Datos.
- El registro histórico permite además *deshacer* una operación recurriendo al *Valor Anterior*
- Para que el *registro histórico* sea útil para recuperar errores del sistema debe estar grabado sobre almacenamiento estable

Modificación diferida de la Base de Datos

Las operaciones de escritura de una transacción T son almacenadas en el *registro histórico*, pero se retrasa su escritura sobre la Base de Datos hasta que T se compromete parcialmente (cuando se ha ejecutado la instrucción final de T).

- En caso de caer el sistema antes de que la transacción complete su ejecución, se ignora la información del *registro*.
- La utilización de esta técnica tan solo necesita en el *registro histórico* del *Valor Nuevo*, simplificando la estructura de este.
- El sistema puede manejar cualquier fallo mediante la instrucción *rehacer(T)*, que fija el *Valor Nuevo* (contenido en el *registro histórico*) en todos los elementos de datos actualizados por T .
- Al ocurrir un fallo el sistema de recuperación consulta el *registro histórico*, rehaciendo todas las transacciones de las cuales el *registro* contenga T iniciada y T comprometida.

Modificación inmediata de la Base de Datos

Realiza las *salidas* de la base de datos escribiéndolas en la propia Base de Datos mientras la transacción esté todavía activa, estas modificaciones se denominan *modificaciones no comprometidas*.

- En caso de caer el sistema o de un fallo de la propia transacción el sistema utiliza el *Valor Anterior* del *registro histórico* para restaurar la integridad de la Base de Datos
- La operación de restauración se realiza mediante la instrucción *deshacer(T)*.
- Todas las operaciones *escribir* son precedidas por la actualización del *registro histórico*. De manera que toda transacción T debe deshacerse si el *registro* contiene T iniciada pero no contiene T comprometida.

Puntos de revisión

Para evitar la sobrecarga que supone el recorrido completo del registro al ocurrir un fallo en el sistema, se crean los *puntos de revisión*.

- Los *puntos* se crean en forma de registros del *registro histórico*, de manera que todas las transacciones que se encuentren comprometidas antes del *punto de revisión* estarán completadas y escritas en la Base de Datos.
- La búsqueda en el *registro histórico* se realiza hacia atrás comenzando por el último registro hasta encontrar el primer *punto de revisión*, rehaciendo y deshaciendo todas las transacciones que finalizaron después del *punto de revisión*.

PAGINACIÓN EN LA SOMBRA

Es una técnica de recuperación alternativa a las basadas en *registro histórico*, en ella la Base de Datos se divide en *páginas* de longitud fija.

- Se utiliza una *tabla de páginas* para localizar la posición en el disco de una página determinada.
- La *paginación en la sombra* se basa en el hecho de que existan 2 *tablas de páginas*, idénticas cuando comienza una transacción, pero la tabla en la *sombra* no se actualiza durante la transacción de manera que mantiene los datos originales.
- La *tabla de páginas en la sombra* se actualiza únicamente cuando la transacción se ha comprometido y acabado completamente su ejecución.
- La *tabla en la sombra* está almacenada en memoria estable y al caer el sistema simplemente hay que convertirla en la tabla activa para recuperar los datos, no haciendo falta las operaciones *deshacer* ni *rehacer*.
- Frente al *registro histórico*, pese a eliminar la sobrecarga que supone la escritura de registros y la rapidez en la recuperación, tiene los siguientes **inconvenientes**:
 - **Sobrecarga en el compromiso**: que supone la escritura de los bloques correspondientes a la tabla en la sombra, la tabla actual y el bloque de los datos en el disco.
 - **Fragmentación de datos**: Se obliga a que cada página de datos cambie su localización en cada escritura (para mantener la copia antigua)
 - **Recogida de basura**: Al comprometerse una transacción y actualizarse la tabla en la sombra hay que liberar el bloque de la copia antigua mediante algún algoritmo de recogida de basura.
 - **Dificulta la ejecución concurrente**: más que el *registro histórico*

15.6.- TRANSACCIONES CONCURRENTES Y RECUPERACIÓN

Independientemente del nº de T's en ejecución **un único registro histórico**.

Interacción con el control de concurrencia → Es necesario garantizar que una transacción no modifica un valor modificado por otra T hasta que la 1ª se haya comprometido (se puede conseguir mediante bloqueo estricto de dos fases).

Retroceso de transacciones → En el caso de que dos transacciones hayan actualizado el mismo elemento de datos, al recorrer el registro histórico hacia atrás, situará el valor inicial de los datos al retroceder las 2 transacciones.

Puntos de revisión → Los registros correspondientes a *puntos de revisión* deben contener una lista con las transacciones activas en ese momento. Es requisito que ninguna transacción escriba mientras se crea un punto de revisión (las que no lo cumplen se llaman punto de revisión difuso).

Recuperación al reiniciar → El sistema **contruye dos listas** cuando se recupera de una caída: *Lista-deshacer* y *Lista-rehacer*, generadas al recorrer el *registro histórico* hacia atrás en busca de un *punto de revisión*:

- Para cada transacción *T* encontrada comprometida: *T* se añade a *lista-rehacer*
- Para cada transacción *T* encontrada como iniciada: si *T* no está en *lista-rehacer*, entonces incluir *T* en *lista-deshacer*.
- Una vez examinado el registro histórico se analiza la *lista* del *punto de revisión*, para cada transacción *T* que no esté en *lista-rehacer*, *T* se añade a *lista-deshacer*.

El **proceso de restauración** una vez determinadas las transacciones a *deshacer* y *rehacer* sigue así:

- Se recorre el *registro histórico* hacia atrás deshaciendo todas las operaciones de las transacciones de la *lista-deshacer*.
- Se localiza el *punto de revisión*.
- Se recorre el *registro histórico* hacia delante rehaciendo las operaciones de las transacciones de la *lista-rehacer*.

Es importante el orden de las operaciones para garantizar la integridad de los datos, así siempre se debe ejecutar en primer lugar **deshacer hacia atrás** y luego **rehacer hacia delante**.

15.7.- GESTIÓN DE LA MEMORIA INTERMEDIA

Registro histórico con memoria intermedia

La escritura de cada registro del *registro histórico* en memoria estable (disco) en el mismo momento de su creación supone una sobrecarga muy alta para el sistema. Así pues los registros se encuentran acumulados en memoria hasta que son escritos en disco, pasado un tiempo. Una caída del sistema supondrá la pérdida de varios registros, luego las operaciones para la recuperación deben adecuarse a esta situación:

- La transacción *T* no pasa a estar comprometida efectivamente hasta que el registro correspondiente no se encuentra en el disco.
- Antes de escribirse en disco el registro *T comprometida*, todos los registros de las operaciones de *T*, deben de estar escritos en disco.
- Antes de que un bloque de datos pueda escribirse en el disco, todos los registros del *registro histórico* pertenecientes a operaciones en ese bloque, deben de estar escritos en disco.(REA-Regla de registro de Escritura Anticipada)

Base de Datos con memoria intermedia

Debido a las restricciones que exige el *registro histórico* se limita la libertad de escritura de bloques de datos a través de la memoria virtual, la secuencia de acciones a llevar a cabo es:

- 1) Escritura en almacenamiento estable de todos los registros del *registro histórico* con operaciones del bloque *B* a guardar.
- 2) Escritura de memoria al disco del bloque *B*.
- 3) Lectura de un nuevo bloque *B'* desde el disco a memoria.

Durante estas operaciones no puede escribirse sobre el bloque *B*, esto se garantiza mediante el bloqueo exclusivo del bloque datos, dicho bloqueo de corta duración se denomina *pestillo*. Los *pestillos* pueden liberarse sin necesidad de ningún protocolo de concurrencia (no necesita bloqueo en 2 fases)

El papel del Sistema Operativo en la gestión de la memoria intermedia

La **memoria intermedia** de la Base de Datos puede **gestionarse de 2 maneras**:

- ❖ El **sistema de la Base de Datos** reserva parte de la memoria principal para usarla como intermedia y **se encarga** de la gestión.
 - **Inconveniente** → Limita la utilización flexible de la memoria, además de que la Base de Datos no puede acceder a toda la memoria.
- ❖ El sistema de la Base de Datos implementa la memoria intermedia dentro de la **memoria virtual del sistema operativo**, aunque el sistema operativo no debería realizar la escritura debido a las restricciones del *registro histórico*, debería de ser la propia Base de Datos la que se encargara de la escritura.
 - **Inconveniente** → Los sistemas operativos actuales controlan completamente la memoria virtual, por lo que la Base de Datos no tiene control sobre la escritura de un bloque en memoria virtual.

15.8.- FALLO CON PÉRDIDA DE ALMACENAMIENTO NO VOLATIL

- Para afrontar una pérdida de registros en el disco se puede realizar el **volcado** de la base de datos entera (copias de seguridad -cintas u otros discos-) periódicamente (1 al día).
- **Para recuperarse** se utiliza el volcado más reciente y sobre él utilizando el registro histórico para recuperar el estado anterior al fallo (no es necesaria ninguna acción deshacer).

15.9.- TÉCNICAS AVANZADAS DE RECUPERACIÓN

La utilización del *bloqueo estricto en dos fases* reduce significativamente la concurrencia en determinadas estructuras como páginas indexadas con *árboles B+*, si se utilizan técnicas para la liberación rápida de los bloqueos, entonces no pueden aplicarse las técnicas de recuperación vistas anteriormente (aptdo. 6) Existen técnicas alternativas de recuperación diseñadas para ser utilizadas con *bloqueos de liberación rápida*.

Registro de deshacer lógico

- El control de concurrencia para *árboles B+*, permite liberar el bloqueo de algunas hojas antes de que la transacción se comprometa (libera algunos bloqueos rápidamente), permitiendo a otras transacciones modificar el árbol, al retroceder transacciones podría llegarse a un estado inconsistente.
- Se crea entonces el **registro histórico lógico**, donde se guarda información acerca de las operaciones (no sólo de los valores como en el registro histórico físico); Por ejemplo una operación de inserción debe deshacerse de manera “lógica” mediante una operación de borrado.
- Cada transacción *T* al finalizar una operación y antes de liberar ningún bloqueo escribe en el **registro histórico lógico** un registro $\langle T, O, \text{fin-operación}, D \rangle$ que contiene un identificador único de la Operación *O* y la **información para Deshacer** (*D*).
- **Operaciones lógicas** → requieren operaciones deshacer lógicas (inserción y borrado).
- Antes de empezar una operación lógica se escribe en el **registro histórico** $\langle T, O, \text{inicio_operación} \rangle$. Durante su ejecución se registran todas sus **modificaciones** de forma normal (val.anterior–val.nuevo), al finalizar se escribe un registro fin_operación.

Retroceso de transacciones

- El **retroceso** durante el modo de operación normal (no durante una recuperación) recorre el registro histórico hacia atrás y devuelve los datos a sus valores anteriores.
- Se escriben **registros especiales** sólo para la operación rehacer de la forma $\langle T, X, V \rangle$ que contiene el valor V con el que se ha restaurado el elemento de datos X durante el retroceso. Estos registros se denominan *registros de compensación del registro histórico*.
- Se toman **acciones especiales** cuando se encuentra un registro del registro histórico de la forma $\langle T, O, \text{fin-operación}, D \rangle$:
 - 1) Se retrocede la operación mediante la información deshacer D. Las modificaciones se registran como cualquier otra operación (incluido generar inicio-operación y fin-operación)
 - 2) Cuando continúa el recorrido atrás, se ignoran todos los registros de la T hasta que se encuentra el registro $\langle T, O, \text{inicio-operación} \rangle$, a partir del cual se procesan de nuevo de manera normal los registros referentes a la transacción.
- Cuando se ha retrocedido la T, se añade un registro $\langle T \text{ abortada} \rangle$.
- Si los fallos ocurren durante una operación lógica (no hay registro fin-operación) los registros físicos del registro histórico se utilizan para retroceder la operación incompleta.
- Si se utilizan bloqueos para la concurrencia han de liberarse sólo una vez la T se ha retrocedido.

Puntos de revisión → El protocolo de los puntos de revisión no se modifica.

Recuperación al reiniciar → Las acciones de recuperación al reiniciar después de un fallo se realizan en **dos fases**:

- 1) En la **fase rehacer** se vuelven a realizar modificaciones de todas las transacciones mediante la exploración hacia delante del *registro histórico* desde la última revisión.
 - * Repite todas las acciones de modificación que fueron ejecutadas después del punto de revisión y cuyos registros alcanzaron un registro histórico estable.
 - * Se determina todas las T's que o bien se encuentran en la lista del pto de revisión, o bien comenzaron más tarde, pero no se tiene el registro de abortada o comprometida (se estaban ejecutando), todas estas T's deben retrocederse por lo que se ponen en la lista-deshacer.
- 2) En la **fase deshacer** se retroceden todas las transacciones de la *lista-deshacer*, recorriendo el *registro histórico* hacia atrás, ignorándose los registros de la transacción a deshacer que se encuentran entre *fin-operación* e *inicio-operación*.

Revisión difusa → La revisión requiere que mientras se efectúa el punto de revisión se suspendan temporalmente todas las modificaciones de la Base de Datos. Dicho protocolo se modifica para permitir modificaciones después de haber escrito en el *registro histórico* el registro *revisión*, pero antes de escribir en disco los bloques de la memoria intermedia que han sufrido modificaciones, el punto creado recibe el nombre de *punto de revisión difuso*.

La idea reside en guardar en una posición fija del disco la dirección del registro que contiene el último *punto de revisión*, (con lo cual no hará falta recorrer hacia atrás el registro histórico para buscarlo) llamada *última-revisión*. Esta información no se actualiza al crear un nuevo *punto de revisión*, sino después de haber escrito en disco todos los bloques de la memoria intermedia, siguiendo la técnica de *registro histórico de escritura anticipada*.