**19.2 Stable-Storage Implementation**

**Donde un papel fundamental en los algoritmos de recuperación**

**Implementacion de un almacenamiento estable**

* **Necesita replicar información necesaria en varios discos no volátiles con modos de fallos independientes y actualizar la información de forma controlada para garantizar que un fallo durante la transferencia de datos no dañe la información**
* ***Recordemos que los sistemas RAID garantizan que el fallo no provoque la pérdida de datos. La forma más sencilla y rápida de RAID es el disco en espejo, que mantiene dos copias de cada bloque en discos separados. Otras formas de RAID ofrecen menores costes, pero a costa de un menor rendimiento.***

**La transferencia de bloques entre la memoria y el almacenamiento en disco puede resultar en:**

**-Finalización con éxito. La información transferida ha llegado sin problemas a su destino.**

**- Fallo parcial. Se ha producido un fallo en medio de la transferencia y el bloque de destino tiene información incorrecta.**

**- Fallo total. El fallo se produjo lo suficientemente pronto durante la transferencia como para que el bloque de destino permanece intacto**

si **se produce un fallo en la transferencia de datos, el sistema lo detecte e invoque un procedimiento de recuperación para restaurar el bloque a un estado consistente. Para ello, el sistema debe mantener dos bloques físicos para cada bloque lógico de la base de datos**

**Una operación de salida se ejecuta de la siguiente manera:**

**1. Escribir la información en el primer bloque físico.**

**2. Cuando la primera escritura se completa con éxito, se escribe la misma información en el segundo bloque físico.**

**3. La salida se completa sólo después de que la segunda escritura se complete con éxito.**

**19.2.2 Acceso a los datos**

**La base de datos está dividida en unidades de almacenamiento de longitud fija llamadas bloques**. **Los bloques son las unidades de transferencia de datos** **hacia y desde el disco** y pueden contener varios elementos de datos. Suponemos que **ningún elemento de datos** abarca dos o más bloques. Esta suposición es realista para la mayoría de las aplicaciones de procesamiento de datos, como un banco o una universidad. **Las transacciones introducen información del disco en la memoria principal y luego devuelven la información al disco. Las operaciones de entrada y salida se realizan en unidades de bloques.**

**-Los bloques que residen en el disco se denominan bloques físicos**

**-Los bloques que residen temporalmente en la memoria principal se denominan bloques de memoria intermedia**.

Los movimientos de bloques entre el disco y la memoria principal se inician mediante las dos operaciones siguientes:

1. **input(B) transfiere el bloque físico B a la memoria principal.**
2. **output(B) transfiere el bloque del buffer B al disco y reemplaza el bloque físico bloque físico correspondiente.**

***Conceptualmente, cada transacción Ti tiene un área de trabajo privada en la que se guardan copias de los datos* a los que Ti accede y actualiza**.

**El sistema crea esta área de trabajo cuando se inicia la transacción; el sistema la elimina cuando la transacción se compromete o aborta. Cada dato X guardado en el área de trabajo de la transacción Ti se denomina xi.**

La transacción Ti interactúa con el sistema de base de datos transfiriendo datos hacia y desde su área de trabajo al buffer del sistema. área de trabajo al buffer del sistema. **La transferencia de datos se realiza mediante estas dos operaciones**

1. **read(X) asigna el valor del dato X a la variable local xi** . Esta operación se ejecuta operación como sigue:

2**. write(X) asigna el valor de la variable local xi al dato X del bloque buffer**. Esta operación se ejecuta de la siguiente manera

Nótese que ambas operaciones pueden requerir la transferencia de un bloque del disco a la memoria principal. Sin embargo, no requieren específicamente la transferencia de un bloque de la memoria principal al disco.

Un bloque de la memoria intermedia se escribe finalmente en el disco porque el gestor de la memoria intermedia necesita el espacio de memoria para otros fines o porque el sistema de la base de datos desea

para reflejar el cambio a B en el disco. Diremos que el sistema de base de datos realiza una salida forzada del buffer B si emite una salida(B).

19.8 Early Lock Release and Logical Undo Operations

Liberación anticipada de bloqueos y operaciones lógicas de deshacer

**Cualquier índice utilizado en el procesamiento de una transacción, como un árbol B+, puede ser tratado como datos normales, pero para aumentar la concurrencia, podemos utilizar el algoritmo de control de concurrencia del árbol B+ para permitir que los bloqueos se liberen antes**, de una manera que no sea de dos fases.

**Como resultado de la liberación temprana de bloqueos, es posible que un valor en un nodo del árbol B+ sea actualizado por una transacción T1, que inserta una entrada (V 1, R1), y posteriormente por otra transacción T2, que inserta una entrada (V 2, R2) en el mismo nodo, desplazando la entrada (V 1, R1) incluso antes de que T1 finalice su ejecución*. En este punto, no podemos deshacer transacción T1 sustituyendo el contenido del nodo por el valor antiguo antes de que T1 realizara su inserción, ya que eso también desharía la inserción realizada por T2***;

-----------------------------

19.8.1 Operaciones lógicas

1. **Las operaciones como las inserciones y eliminaciones del árbol B+ liberan los bloqueos antes de tiempo.**

**- No pueden deshacerse restaurando los valores antiguos (deshacer físicamente), ya que una vez que se libera un bloqueo, otras transacciones pueden haber actualizado el árbol B+.**

**- En cambio, las inserciones (o eliminaciones) se deshacen ejecutando una operación de eliminación (o inserción) lo que se conoce como deshacer lógico.**

**-------------------------------**

**19.8.2 Registros de deshacer lógicos**

**1. Cuando se inicia la operación, se registra <Ti, Oj, operation-begin>. Aquí Oj es un identificador único de la instancia de la operación.**

**2. Mientras se ejecuta la operación, los registros normales con información física de rehacer y deshacer se registran.**

**3. Cuando la operación se completa, <Ti, Oj, operation-end, U> es se registra, donde U contiene la información necesaria para realizar una información de deshacer lógica**.

----------------------------

**19.8.3 Reversión de transacciones con deshacer lógicamente**

La reversión de la transacción Ti se realiza de la siguiente manera:

Examinar el registro hacia atrás

1. Si se encuentra un registro <Ti , X, V1, V2>, se realiza el deshacer y se registra un al <Ti , X, V1>.

2. Si a <Ti , Oj , operación-fin, U> se encuentra un registro

Deshacer la operación lógicamente utilizando la información de deshacer U.

- Las actualizaciones realizadas durante el roll back se registran igual que

durante la ejecución normal de la operación.

- Al final del rollback de la operación, en lugar de registrar un registro de

registro de fin de operación, se genera un registro<Ti, Oj, operación-aborto>.

Omite todos los registros de registro precedentes para Ti hasta el registro

---------------------

3. Si se encuentra un registro de sólo rehacer ignorarlo

4. Si un <Ti , Oj , operación-aborto> es encontrado: omitir todos los registros de registro precedentes para Ti hasta el registro <Ti , Oj , operación-inicio> se encuentre.

5. Detener la exploración cuando se encuentre el registro <Ti , inicio> se encuentra

6. Añadir un registro <Ti , abortado> al registro

-----------------------------------

19.8.4 Algoritmo de recuperación con Deshacer Lógico

1. (Fase de Rehacer): Se escanea el registro hacia adelante desde el último registro < punto de control L> hasta final del registro

1. Repetir la historia rehaciendo físicamente todas las actualizaciones de todas transacciones

2. Crear una lista de deshacer durante la exploración de la siguiente manera

- La lista de deshacer se establece en L inicialmente

- Siempre que se encuentre <Ti start> se añade Ti a la lista de deshacer

- Siempre que se encuentre <Ti commit> o <Ti abort>, Ti se elimina de la lista de deshacer

Esto lleva a la base de datos al estado en el que se encontraba antes de la caída, con transacciones comprometidas y no comprometidas que se han rehecho.

Ahora la lista de deshacer contiene transacciones que están incompletas, es decir, no han sido comprometidas ni revertidas completamente.

-------------------------

2. (Fase de deshacer): Escanear el registro hacia atrás, realizando el deshacer en los registros del registro de las transacciones encontradas en la lista de deshacer.

- Los registros de las transacciones que se están deshaciendo se procesan como descrito anteriormente, a medida que se encuentran

-Escaneo único compartido para todas las transacciones que se están deshaciendo

- Cuando se encuentra <Ti start> para una transacción Ti en la lista de deshacer, se escribe un <Ti abort> registro de registro.

- Detener el escaneo cuando se hayan encontrado registros <Ti start> para todas las Ti en la lista deshacer

Esto deshace los efectos de las transacciones incompletas (aquellas que no tienen ni registros de registro de commit ni de abortado). La recuperación se ha completado.