


Bases de Datos I

Cursada 2008

Clase 7: Recuperación de BD

Facultad de Ciencias Exactas
Universidad Nac. Centro de la Pcia. de Bs₁ As.




BASES DE DATOS I

Introducción a la Seguridad

Una base de datos es:
Un conjunto de datos integrados , adecuado a
varios usuarios y a diferentes usos

- o el uso concurrente de los datos plantea problemas de seguridad que deben ser paliadas con las facilidades que e proporciona el DBMS.
- o La protección de los datos deberá llevarse a cabo contra fallos físicos, fallos lógicos y fallos humanos (intencionados o no).

2



BASES DE DATOS I

Introducción a la Seguridad

El DBMS facilita mecanismos para

- o prevenir los fallos (subsistema de control),
- o detectarlos una vez que se han producido (subsistema de detección)
- o corregirlos después de haber sido detectados (subsistema de recuperación).

Aspectos fundamentales de la seguridad

- o Confidencialidad. No develar datos a usuarios no autorizados. Comprende también la privacidad (protección de datos personales).
- o Accesibilidad. La información debe estar disponible.
- o Integridad. Permite asegurar que los datos no han sido falseados.

3

Recuperación de Bases de Datos

Significa →

- Examinar las implicaciones de los fallos en las transacciones, sobre la base de datos.
- Examinar maneras de prevenir los problemas que pueden ocurrir como resultado de fallos en el sistema.

4

Transacciones

Transacción: conjunto de operaciones que forman una unidad conceptual de procesamiento.

Propiedades **ACID**

- **Atomicidad**: se ejecutan todas las operaciones de la transacción o no lo hace ninguna de ellas → si algo falla no deben quedar rastros de la ejecución inconclusa en la BD → **Recovery Manager + Transaction Manager**
- **Consistencia**: la ejecución de la transacción conserva la consistencia de la BD → **Concurrency Manager + restricciones**

5

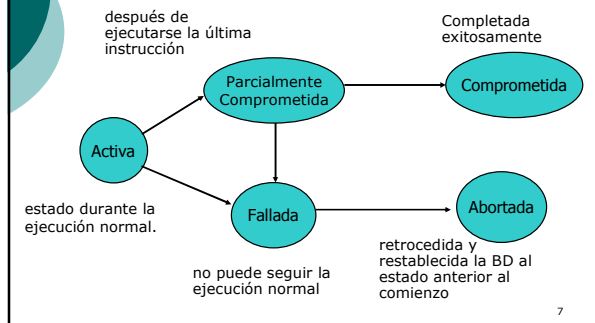
Transacciones

Propiedades **ACID**

- **Aislamiento** (isolation): cada transacción se ejecuta sin interferencia del resto de las transacciones que se ejecutan concurrentemente en el sistema → **Transaction Manager**
- **Durabilidad**: los cambios de una transacción finalizada exitosamente deben ser permanentes en la BD, incluso si hay fallos en el sistema → **Recovery Manager**

6

Diagrama de estado de una transacción



7

Transacciones

○ Transacción abortada →

● ACCIONES:

- Error no vinculado a lógica de la transacción (hardware o software) → **Reinicio**
- Error en la lógica del programa que ejecuta la transacción → **Cancelar**

8

Fallos con Pérdida de Información

Muchos **tipos diferentes de fallos** que pueden afectar al procesamiento de la base de datos

Cada uno debe ser tratado de forma distinta

- Paradas catastróficas del sistema debidas a errores del hardware o del software → **pérdida de contenido de la memoria principal**
- Fallos del soporte físico → **pérdida de la información guardada en almacenamiento secundario**

9

Fallos con Pérdida de Información

- Errores de software de las aplicaciones → fallo en una o más transacciones
- Desastres físicos naturales como incendios, inundaciones, terremotos y apagones
- Destrucción negligente o no intencionada de datos o instalaciones por operadores o usuarios
- Sabotaje o corrupción o destrucción intencionada de los datos, del hardware, del software o de las aplicaciones

10

Tratamientos ante Fallos

Eventos indeseables

- Esperados
 - Acciones ejecutadas durante el procesamiento normal de la transacción que permiten la recuperación ante fallos
- integran la lógica de la transacción

11

Tratamientos ante Fallos

Eventos indeseables

- Inesperados
 - Acciones ejecutadas después de ocurrir el fallo para restablecer el contenido de la BD
- garantizar las propiedades ACID
- Retornar al estado consistente previo a la ejecución de la transacción fallida.

12

Algoritmos de Recuperación

Los algoritmos de Recuperación son técnicas que garantizan las propiedades A, C y D a pesar de las fallas.

- Tienen dos partes:
 1. Acciones durante el procesamiento normal de una transacción para asegurar que hay suficiente información de ayuda a la recuperación en caso de falla.
 2. Acciones ejecutadas después de una falla para recuperar el estado de consistencia de la base (propiedades A, C y D)

13

Recuperación

Un SGBD debe proporcionar las siguientes funcionalidades como ayuda a la recuperación:

- Mecanismo de copia de seguridad mediante el que se hagan copias de seguridad periódicas de la base de datos
- Facilidades de registro que mantengan el control del estado actual de las transacciones y de los cambios realizados en la base de datos
- Funcionalidad de puntos de comprobación que permita que las actualizaciones de la base de datos que estén llevándose a cabo se hagan permanentes
- Gestor de recuperación que permita al sistema restaurar la base de datos a un estado coherente después del fallo

14

Recuperación

Estrategias típicas

- Copias de respaldo en otros medios de almacenamiento (backups)
- Operaciones específicas: Redo, Undo
- Medio para registrar los movimientos realizados →
 - LOG de transacciones (diario o bitácora)
- Métodos de recuperación basadas en el contenido del log →
 - Actualización diferida (Deferred update)
 - Actualización inmediata (Immediate update)
- Técnicas para recuperación →
 - Vía reprocesamiento
 - Vía rollback/rollforward

15

Recuperación vía Reprocesamiento

- La base de datos se retorna a un estado correcto previo conocido y se reprocesan las transacciones ejecutadas hasta el momento del fallo
- Estrategia impráctica en algunos casos y no factible en otros →
 - El sistema tiene una carga de trabajo asincrónica, eventos concurrentes, etc.

16

Recuperación vía Rollback/Rollforward

- Salvar la BD periódicamente y mantener un archivo auxiliar con los cambios producidos (**LOG**) en orden cronológico.
- Write-Ahead Logging (WAL)
 - Fuerza las escrituras en el LOG antes de escribir en la base de datos (#1)
 - Fuerza el Commit de la transacción después de escribir en el LOG → escribe todos los registros del Log de la transacción antes del commit (#2)
- #1 garantiza Atomicidad.
- #2 garantiza Persistencia (durabilidad)

17

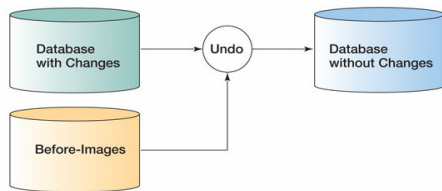
Recuperación vía Rollback/Rollforward

- Cuando ocurre un fallo →
 - Rollback: deshace los cambios erróneos y reprocesa las transacciones válidas
 - Rollforward: rehace los cambios en la base de datos usando datos salvados y transacciones válidas desde el último backup

18

Rollback

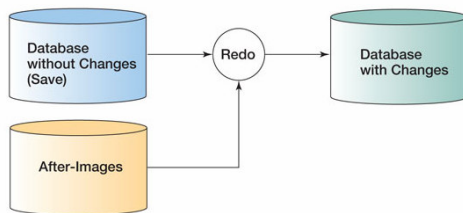
Imagen 'antes de': una copia de cada registro de la BD (o página) antes de ser modificada.



19

Rollforward

Imagen 'después de': una copia de cada registro de la BR (o página) después de haber sido cambiada



20

Backups

- Backups regulares salvan el estado de la base de datos en un momento determinado

- Útil para restaurar la base de datos en caso de fallas catastróficas (destrucción física, sabotaje, etc.)
- Operación costosa en tiempo → ejecutada cuando el sistema está ocioso o con menor carga de trabajo
- Backup incremental → copia de respaldo frecuente de aquéllos registros que se han modificado
- Cómo asegurar persistencia de los cambios (**Durabilidad ACID**)

21

Recuperación basada en el LOG

Cada registro de cambio del LOG contiene →

- Identificador de la transacción (Id_T) que ejecuta el cambio
- Identificador del ítem de datos modificado (típicamente la ubicación en el disco)
- Valor viejo (que fue sobrescrito)
- Valor nuevo (luego del write)
- Otros registros del LOG contienen →
 - <Id_T, tiempo de comienzo> → transacción ACTIVA
 - <Id_T, tiempo de commit> → transacción confirmada
 - <Id_T, tiempo de aborto> → transacción abortada
- El LOG tiene los datos completos de los cambios desde el último backup
- Debe almacenarse en memoria estable !!

22

Recuperación basada en el LOG

- Operación de recuperación usa dos primitivas →
 - **redo**: rehace la actualización registrada en el LOG.
 - Escribe el valor nuevo sobre el ítem de datos modificado
 - **undo**: deshace la actualización registrada en el LOG
 - Escribe el valor viejo sobre el ítem de datos modificado

23

Recuperación basada en el LOG

- Ambas primitivas ignoran el estado del registro en la base de datos → sobrescriben directamente.
- La aplicación múltiple de estas operaciones es equivalente a la aplicación de la última vez → **idempotencia de undo y redo**

redo (redo (redo (... (redo(x)))))) ≡ redo (x)
 undo (undo (undo (... (undo(x)))))) ≡ undo (x)

24

Checkpoints

- Cuando se produce un fallo ...
 - Que acciones hay que rehacer (redo)?
 - Que acciones hay que deshacer (undo)?
- Inspección del LOG completo →
 - Costoso en tiempo
 - Muchas transacciones ya habrán confirmado sus cambios en la base de datos (ya son persistentes) → rehacerlas es inofensivo pero se pierde tiempo

25

Checkpoints

- Para reducir la carga de trabajo que implica la restauración → **checkpoints**.
 1. Grabar todos los registros del LOG actualmente en memoria principal en almacenamiento estable.
 2. Grabar en la BD todos los bloques de registros almacenados en buffers.
 3. Escribir en el LOG un registro < **checkpoint** >.

26

Checkpoints

Durante la recuperación → considerar solamente la transacción T_i más reciente comenzada antes del checkpoint, y las que comenzaron después de T_i .

1. Recorrer hacia atrás desde el final del LOG, hasta alcanzar el <**checkpoint**> más reciente.
2. Continuar recorriendo hasta encontrar < T_i **start**>.
3. Considerar la parte del LOG que sigue a este registro.

27

Checkpoints

1. Para todas las transacciones (comenzadas desde T_i o después) que no hayan registrado $\langle T_i \text{ commit} \rangle$, ejecutar $\text{undo}(T_i)$ → *tiene sentido en actualizaciones inmediatas o de registro desconocido!!*
2. Recorrer el LOG hacia adelante para todas las transacciones comenzadas desde T_i que *hayan* registrado $\langle T_i \text{ commit} \rangle$, ejecutando $\text{redo}(T_i)$.

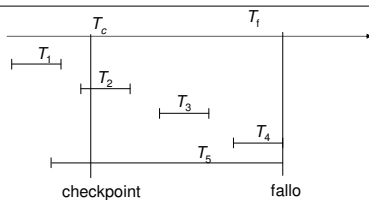
28

Checkpoints

- No esperar a que las transacciones activas finalicen
- No permitir que hagan modificaciones en los buffers o en el LOG mientras dura el checkpointing
- Debe insertarse un registro de checkpointing en el LOG, que incluya la lista de transacciones activas → $\langle \text{checkpoint}, L \rangle$
- Para la recuperación se necesita recorrer los registros correspondientes a las transacciones indicadas en L, para deshacer o rehacer cambios.

29

Esquema de Situaciones con Checkpoint



- T_1 puede ignorarse → cambios registrados en la BD por el checkpoint
- T_2 y T_3 → rehacer → **Durabilidad**
- T_4 y T_5 → deshacer → **Atomicidad**

30

Recuperación de Bases de Datos

T18: **start**.
T83: **start**
T18: 1982374, Balance, \$900.00, \$950.00
T29: **start**
T83: 3874843, Balance, \$20.00, \$70.00
T29: 4948543, Balance, \$1350.00, \$350.00
T18: 9384945, Balance, \$200.00, \$250.00
T18: **commit**
T29: 3984554, Balance, \$900.00, \$1900.00
T29: **commit**
Falla

Recuperación de Bases de Datos

T18: **start**.
T83: **start**
T18: 1982374, Balance, \$900.00, \$950.00
T29: **start**
T83: 3874843, Balance, \$20.00, \$70.00
T29: 4948543, Balance, \$1350.00, \$350.00
T18: 9384945, Balance, \$200.00, \$250.00
T18: **commit**
T29: 3984554, Balance, \$900.00, \$1900.00
T29: **commit**
Falla

Mecanismo de Undo

- 33

Mecanismo de Redo

- (1) T = conjunto de transacciones con $\langle T_i, \text{commit} \rangle$ (y sin $\langle T_i, \text{end} \rangle$) en el LOG
- (2) Para cada $\langle T_i, X, X_{\text{nuevo}} \rangle$ en el LOG, avanzando hacia el final del LOG hacer:
 - Si $T_i \in T \rightarrow \text{write}(X, X_{\text{nuevo}})$ (buffer)
output(X) (copia en BD)
- (3) Para cada $T_i \in T \rightarrow \text{write} \langle T_i, \text{end} \rangle$

34

Rollback de Transacciones

- Cascading rollback
 - S lee un ítem de datos que T escribe
 - Si T es abortada, S leyo un valor 'inexistente' \rightarrow S debe ser abortada
- Métodos prácticos de recuperación
 - \rightarrow garantizan que no habrá rollback en cascada
 - \rightarrow tópico relativo a problemas de concurrencia

35

Recuperación basada en Act. Diferidas

- Las transacciones no cambian la BD hasta que alcanzan el punto de confirmación
 - No lo alcanzan hasta que el LOG no ha sido actualizado
- \rightarrow Estrategia **NoUndo/Redo**

36

Recuperación basada en Act. Inmediatas

- Actualiza "inmediatamente" la BD usando WAL
- Dos estrategias
 - **Undo/NoRedo** → actualiza la BD en el disco antes del commit
 - **Undo/Redo** → puede hacer commit antes de que los cambios se hayan grabado efectivamente en la BD (puede haber cambios en buffers)

37

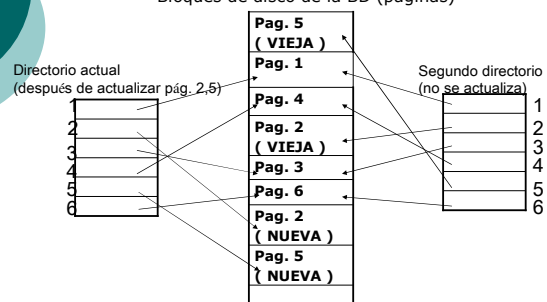
Doble Paginación (Shadow Paging)

- Considera la BD constituida por páginas de disco de tamaño fijo
- El segundo directorio no cambia y el corriente apunta a la página más reciente
- Estrategia NoUndo/NoRedo

38

Doble Paginación (Shadow Paging)

Bloques de disco de la BD (páginas)



39

Doble Paginación

- Para confirmar una transacción
 - Volcar el buffer a la BD
 - Volcar el directorio corriente a disco
 - no sobrescribir el segundo directorio
 - Copiar la dirección del directorio corriente a la del segundo directorio → la transacción alcanza el commit
- No es necesario recuperar luego de un fallo
→ las transacciones pueden continuar a partir del segundo directorio.

40

Doble Paginación: Desventajas

- Fragmentación de los datos → páginas relacionadas están separadas en el disco
- Sobrecarga para los Commit (si el directorio es extenso) → necesidad de volcar cada página modificada y el directorio
- Copiar el directorio es costoso →
 - Puede reducirse usando una estructura de árbol B⁺ → copiar solamente los caminos que llevan a hojas modificadas
- Colector de Basura → páginas que dejan de ser apuntadas por los directorios deberían ser liberadas.

41
