Diseño de Bases de Dato

Seguridad e Integridad de datos: Agenda

Transacciones	• Propidades • Estados
Transacciones monusuarias	• Atomicidad • Protocolos
Transacciones centralizadas	• Aislamiento • Consistencia • Durabilidad

Transacciones

Transacción: colección de operaciones que forman una única unidad lógica de trabajo.

- Propiedades **ACID**
  - Atomicidad: todas las operaciones de la transacción se ejecutan o no lo hacen ninguna de ellas
  - Consistencia: la ejecución aislada de la transacción conserva la consistencia de la BD
  - Aislamiento (isolation): cada transacción ignora el resto de las transacciones que se ejecutan concurrentemente en el sistema, actúa c/u como única.
  - Durabilidad: una transacción terminada con éxito realiza cambios permanentes en la BD, incluso si hay fallos en el sistema

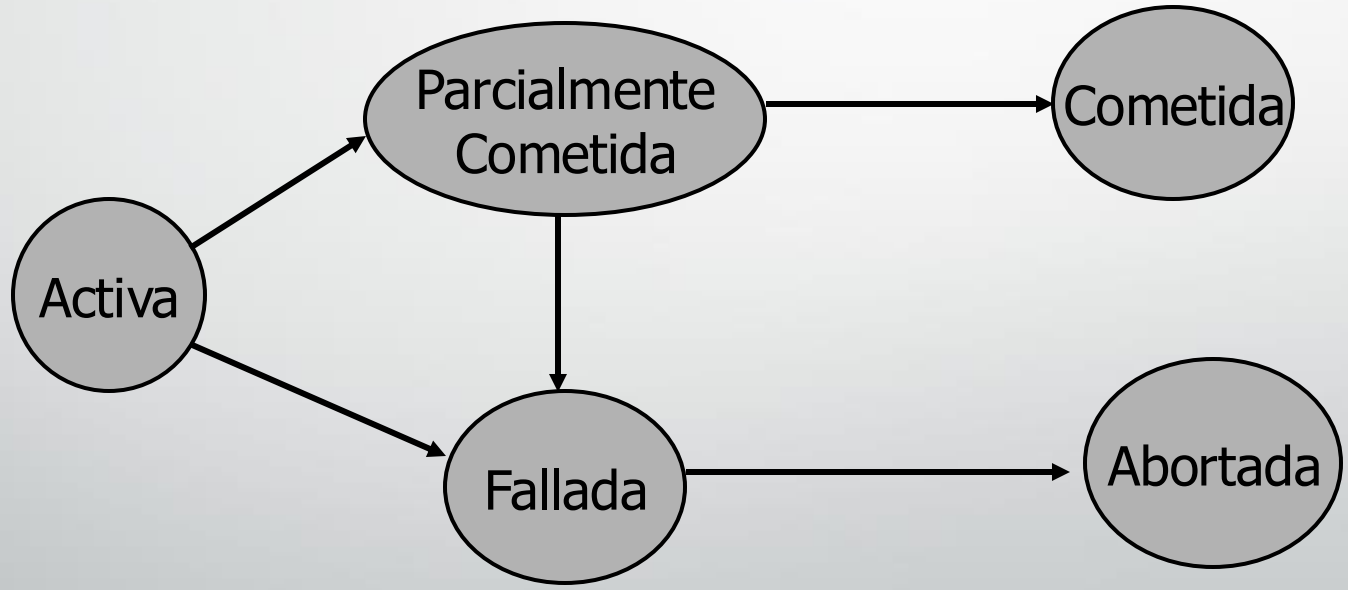
Transacciones

Estados de una transacción

- Activa: estado inicial, estado normal durante la ejecución.
- Parcialmente Cometida: después de ejecutarse la última instrucción
- Fallada: luego de descubrir que no puede seguir la ejecución normal
- Abortada: después de haber retrocedido la transacción y restablecido la BD al estado anterior al comienzo de la transacción.
- Cometida: tras completarse con éxito.

Transacciones

Diagrama de estado de una transacción



Transacciones

Modelo de transacción

- READ (A, a1)
- a1 := a1 – 100;
- WRITE(A, a1)
- READ (B, b1)
- b1 := b1 + 100;
- WRITE(B, b1)

Diferencia entre READ, WRITE y INPUT, OUTPUT.

Uso de transacciones:

- En sistemas monousuario
- En sistemas concurrentes
- En sistemas distribuidos

Transacciones

1.READ (A, a1)
2.a1 := a1 – 100;
3.WRITE(A, a1)
4.READ (B, b1)
5.b1 := b1 + 100;
6.WRITE(B, b1)

BD A = 1000 900 (3)
B = 2000 2100(6)

Memoria local
A = 1000 (1) 900 (2)
B = 2000 (4) 2100 (5)

FALLO LUEGO DE 3 Y ANTES DE 6 ? QUE PASA?

Que hacer luego de un fallo?

- Re-ejecutar la transacción fallada → no sirve
- Dejar el estado de la BD como está → no sirve

Problema: modificar la BD sin seguridad que la transacción se va a cometer.

- Solución: indicar las modificaciones

Soluciones

- Registro Historico
- Doble paginación

Registro Histórico

Bitácora

- secuencia de actividades realizadas sobre la BD.
- Contenido de la bitácora
  - <T iniciada>
  - <T, E, Va, Vn>
    - Identificador de la transacción
    - Identificador del elemento de datos
    - Valor anterior
    - Valor nuevo
  - <T Commit>
  - <T Abort>

Registro Historico

Las operaciones sobre la BD deben almacenarse luego de guardar en disco el contenido de la Bitácora

Dos técnicas de bitácora

- Modificación diferida de la BD
- Modificación inmediata de la BD

Registro Histórico

Modificación diferida

- Las operaciones write se aplazan hasta que la transacción esté parcialmente cometida, en ese momento se actualiza la bitácora y la BD

Registro Historico

1. READ (A, a1)
2. a1 := a1 – 100;
3. READ (B, b1)
4. b1 := b1 + 100;
5. WRITE(A, a1)
6. WRITE(B, b1)

Bitacora

1. <T start> 2 no produce efecto
5. <T, A, 900 > 3, 4 sin efecto
6. <T, B, 2100>
7. <T commit>

Memoria RAM

1. A = 1000
2. A = 900
3. B = 2000
4. B = 2100

Plan1 1, 2, 3, 4, 5, 6, 7 , 8 , 9 fallo
Recupera del Fallo y no habria que hacer nada???

Plan2 1,2,3,4,5,6,7, 8 fallo
Recupera del Fallo y que hacemos???

Plan3 1,2,3, 4, 5, 6, 7 fallo
Recupera del fallo y no habria que hacer nada???

Planotro fallo antes de 7
Base de datos bien.

Base de Datos

A = 1000
900 (8)
B = 2000
2100 (9)

Base de Datos

A = 1000
900 (8)
B = 2000

Base de Datos

A = 1000

B = 2000

Registro Histórico

Dada la siguiente transacción

- < To Start >
- < To, A, 900 >
- < To, B, 2100 >
- < To Commit >

Recién con To parcialmente cometida, entonces se actualiza la BD.

- No se necesita valor viejo, se modifica la BD al final de la transacción o no se modifica.

Ante un fallo, y luego de recuperarse:

- REDO (Ti), para todo Ti que tenga un Start y un Commit en la Bitácora.
- Si no tiene Commit entonces se ignora, dado que no llegó a hacer algo en la BD.

Registro Histórico

Modificación inmediata:

- La actualización de la BD se realiza mientras la transacción está activa y se va ejecutando.
- Se necesita el valor viejo, pues los cambios se fueron efectuando.
- Ante un fallo, y luego de recuperarse:
  - REDO(Ti), para todo Ti que tenga un Start y un Commit en la Bitácora.
  - UNDO(Ti), para todo Ti que tenga un Start y no un Commit.

Registro Histórico

Transacción:

- Condición de idempotencia.

Buffers de Bitácora

- Grabar en disco c/registro de bitácora insume gran costo de tiempo → se utilizan buffer, como proceder?
  - Transacción está parcialmente cometida después de grabar en memoria no volátil el Commit en la Bitácora.
  - Un Commit en la bitácora en memoria no volátil, implica que todos los registros anteriores de esa transacción ya están en memoria no volátil.
  - **Siempre** graba primero la Bitácora y luego la BD.

Registro Histórico

Puntos de verificación:

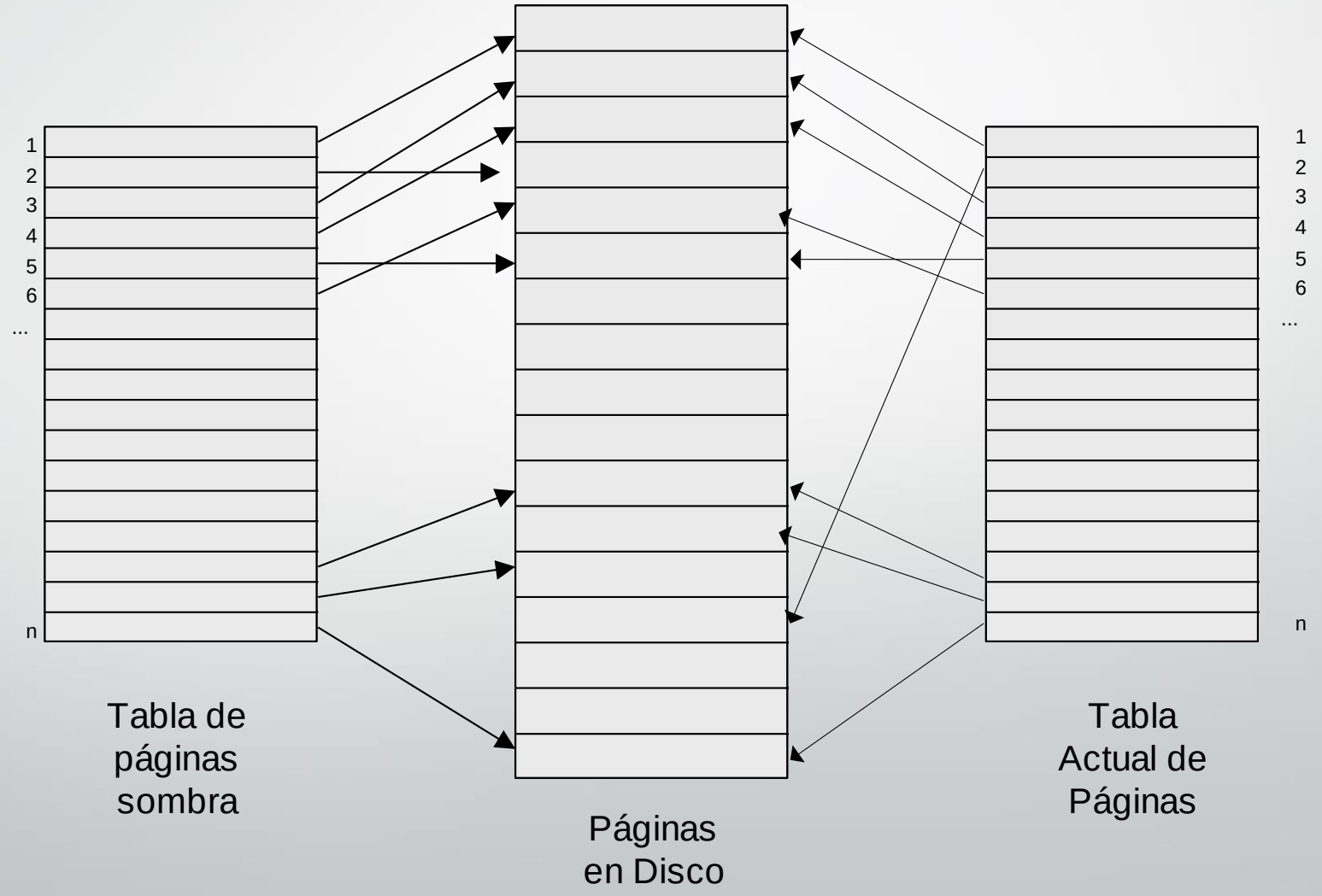
- Ante un fallo, que hacer
  - REDO, UNDO: según el caso
- Revisar la bitácora:
  - Desde el comienzo?: probablemente gran porcentaje esté correcto y terminado.
  - Lleva mucho tiempo.
- Checkpoints (monousario)
  - Se agregan periódicamente indicando desde allí hacia atrás todo OK.
  - Periodicidad?

Doble Paginación

Paginación en la sombra:

- Ventaja: menos accesos a disco
- Desventaja: complicada en un ambiente concurrente/distribuido.
- N páginas equivalente a páginas del SO.
  - Tabla de páginas actual
  - Tabla de páginas sombra

Doble Paginación



Doble Paginación

Ejecución de la operación *escribir*

- Ejecutar **entrada**(X) si página i-ésima no está todavía en memoria principal.
- Si es la primer escritura sobre la página i-ésima, modificar la tabla actual de páginas así:
  - Encontrar una página en el disco no utilizada
  - Indicar que a partir de ahora está ocupada
  - Modificar la tabla actual de página indicando que la i-ésima entrada ahora apunta a la nueva página

Doble Paginación

En caso de fallo y luego de la recuperación

- Copia la tabla de páginas sombra en memoria principal.
- Abort automáticos, se tienen la dirección de la página anterior sin las modificaciones.

Recuperación en caso de Fallo

Ventajas:

- Elimina la sobrecarga de escrituras del log
- Recuperación más rápida (no existe el REDO o UNDO).

Desventajas:

- Sobrecarga en el compromiso: la técnica de paginación es por cada transacción.
- Fragmentación de datos: cambia la ubicación de los datos continuamente.
- Garbage Collector: ante un fallo queda una página que no es mas referenciada.