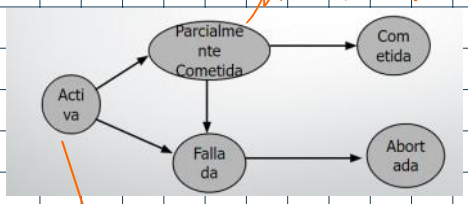
Transacciones

jueves, 19 de octubre de 2023 14:41

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



- 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.

Que hacer luego de un fallo?

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

Soluciones

- Registro Histórico
- Doble paginación

- <T iniciada>
- <T, E, Va, Vn>
  - Identificador de la transacción
  - Identificador del elemento de datos
  - Valor anterior
  - Valor nuevo
- <T Commit>
- <T Abort>

Dos técnicas de bitácora

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

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.

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.

<T0 B, 2000, 2050>
<T0 commit>

Transacción:

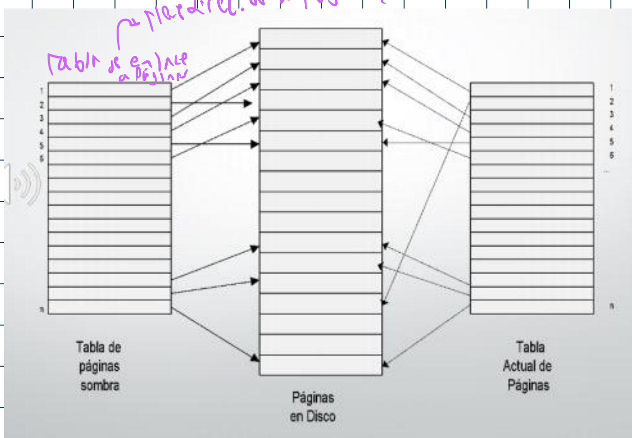
- Condición de idempotencia. → No importa cuántas veces hago una transacción, va a dar lo = siempre.

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. → si tengo 1000 transacciones bien y 1 mal, tengo q hacerlas de nuevo sí o sí.
  - Lleva mucho tiempo.
- Checkpoints (monousario) → marca cada x transacciones
  - Se agregan periódicamente indicando desde allí hacia atrás todo OK.
  - Periodicidad?

Doble paginación:
Dividir BD en nodos/páginas → datos.
Tablas q' apuntan a páginas

Se hacen 2 tablas en disco y cada una apunta a páginas generales



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.

Transferencia, Concurrencia.

Entorno centralizado

- Varias transacciones ejecutándose simultáneamente compartiendo recursos.
- Deben evitarse los mismos problemas de consistencia de datos
- Transacciones correctas, en ambientes concurrente pueden llevar a fallos

Serializabilidad → se va a hacer 1 transacción.
¿concurrente?
(capaz mezclar op. de transacciones distintas, pero nunca a la vez)?

Planificación: secuencia de ejecución de transacciones

- Involucra todas las instrucciones de las transacciones
- Conservan el orden de ejecución de las mismas
- Un conjunto de m transacciones generan m! planificaciones en serie
- La ejecución concurrente no necesita una planificación en serie.

Conclusiones

- El programa debe conservar la consistencia
- La inconsistencia temporal puede ser causa de inconsistencia en planificaciones en paralelo
- Una planificación concurrente debe equivaler a una planificación en serie
- Solo las instrucciones READ y WRITE son importantes y deben considerarse.