Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para iniciar una transacción:

Diagrama

Descripción generada automáticamente

El commit finaliza la transacción

Texto, Carta

Descripción generada automáticamente

Conservación de consistencia:

Si la base de datos estaba todo bien, cuando termina una transacción, no puede quedar inconsistente

Aislamiento: las transacciones se deberían ejecutar “en serie”, ninguna pisa a ninguna

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Recuperación: cuando se reinicia la base de datos se genera un proceso de recuperación, lo último que pasó y que no se haya perdido nada.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

¿Qué pasa si no se aplica ningún control de concurrencia?

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Diagrama

Descripción generada automáticamente con confianza media

Texto, Carta

Descripción generada automáticamente

Cómo resolver los conflictos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente (Semáforos)

Imagen que contiene Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

se pueden hacer una por commit

Transacciones:

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente con confianza baja

Como están en dos tablas distintas, no van a chocar las transacciones

Mismas tablas se bloquean

Interfaz de usuario gráfica, Aplicación, PowerPoint

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se hace a nivel fila el bloqueo, no a nivel tabla. Funciona igual: todo depende del where ya que lo que está ahí va a ser la clave.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación, Carta

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Son excluyentes esas condiciones, pero la base de datos no se da cuenta

Imagen de la pantalla de un celular

Descripción generada automáticamente con confianza media

Se traban mutuamente esperándose de forma infinita. Agarra una de las dos de manera aleatoria y mata a la otra

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Mató a T2 y perdió todo lo que hizo. Se hace un rollback

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

No hay forma de evitarlos, la bd no sabe que se van a leer mutuamente

Interfaz de usuario gráfica, Texto, Correo electrónico

Descripción generada automáticamente

Modificar nivel de aislamiento: se pueden evitar los conflictos en la bd

Dirty read: Una transacción modifica un dato, y la segunda lo lea y tal vez la 1 aborta. Entonces leyó algo sin sentido

Imagen que contiene Diagrama

Descripción generada automáticamente

Serializable: hace la bd más lenta

* A mayor nivel de control de concurrencia, menos paralelismo

Tabla

Descripción generada automáticamente

Tabla

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

En el segundo select de la tabla 1 se va a leer a ale cuando todavía no se confirmó transacción

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Como está uncommited la primera y la otra no, ahí sí no pegó, funcionó correctamente sin mostrar ningún valor innecesario

Imagen que contiene Diagrama

Descripción generada automáticamente

Y así con el resto.

El read uncommited se utiliza cuando sabemos que hay algo que no es riesgoso

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

La T1 se queda esperando hasta que comitee

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Se bloquea hasta que termina el select, ya que una vez que hice el select, bloquea las tablas, hasta que termine la transacción:

¿El select bloquea? Depende del nivel de aislamiento.

* Cuando definimos repeatable read, las filas **que cumplen** el mismo where, se bloquean

Cuando ya está serializable, de movida te bloquea toda la tabla de una:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Recovery manager: veo que las últimas op de las bd, por las dudas las revisa y se fija que queden bien hechas

Diagrama

Descripción generada automáticamente

Si se corta la luz, o algo así, el Recovery manager se va a fijar el último checkpoint y va a preguntar. ¿Están seguro que están hechos los últimos cambios?

Si no es así, se vuelca todo en Datafiles.

Cuando se inicializan a veces tarda, porque está chequeando el RM antes de mostrarte todo

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

* Es donde se guardan todos los datos, se vacían cada n semanas, sería lo ideal.
* Cuando una persona hace un select, primero se va a fijar en el log si hubo algún cambio. Si lo hubo, te muestra la info del log
* Se hacen backup de ambos, sino habría datos inconsistentes.

Se pueden usar vistas dentro de vistas, y Procedures dentro de procedures

Logs: cuando se hace un insert, otras transacciones, se hace como una realidad paralela. Si no hizo commit o rollback, la bd no sabe qué va a pasar:

* + Antes de pisar la bd, anota todo “en un papelito” (log) y luego cuando se commitea todo, después lo pase a la base de datos posta
  + Por eso el roll back es tan rápido. “Tira a la basura el papelito”