Clase del viernes 22 de setiembre del 2023

ACID corresponde a Atomicity, Consistency, Isolation, Durability.
Una operación atómica a nivel de hardware es una operación que se ejecuta y no puede ser interrumpida, tarda un ciclo de reloj y es indivisible.

¿cómo se logra que una serie de statements se comporten como

una operación atómica del sistema operativo?

-Con atomicidad.

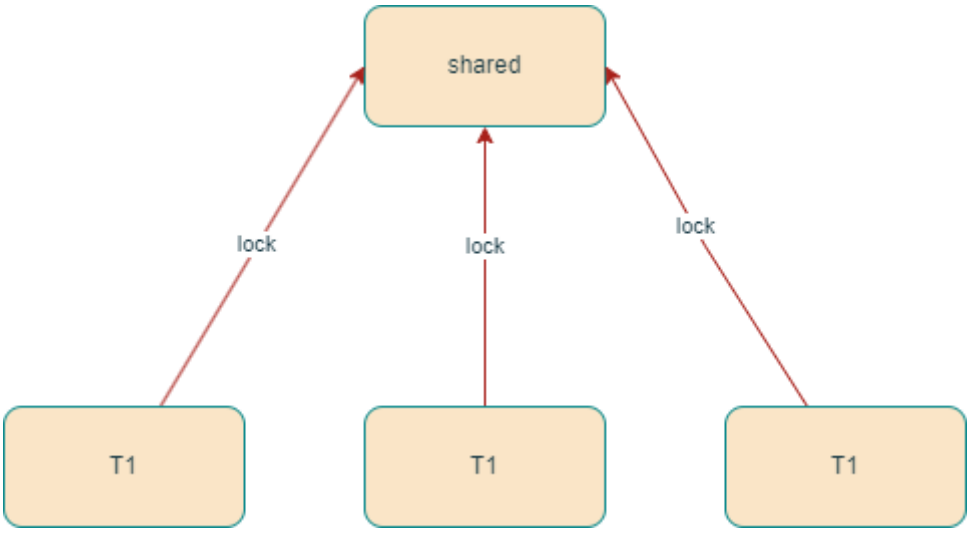
Atomicidad. (Atomicity)

La atomicidad permite lograrlo con statements, con problemas de poder como cuando se apaga la computadora, con errores y transacciones.

¿Cómo se asegura atomicidad en una base de datos?

-Con el concepto de locks o candados.

Cuando se tiene un recurso compartido se debe de proteger. Cuando varios usuarios intentan hacer lock al mismo tiempo el lock se serializa. El CPU se le entrega a cada proceso por pequeños periodos de tiempo llamados quantum. Cuando se ejecuta un lock no existe forma de que se expropie el CPU por otra operación en media ejecución. Si todas las transacciones llegan simultáneamente una entra de primero, obtiene el recurso compartido, cuando finaliza las otras transacciones no pueden tomar el control del recurso, hasta que se indique el release con un free.



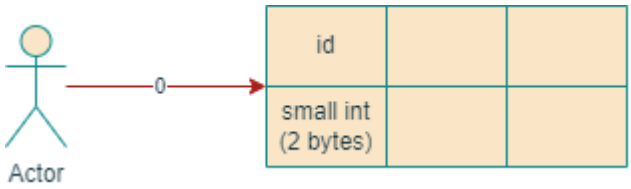
Si se quiere lograr un muy buen rendimiento manteniendo una base de datos SQL manejando transacciones con locks, se debe hacer una inversión muy grande en términos de hardware.

En ocasiones se ignora la atomicidad, lo que causa que en bases de datos no SQL se quiten los locks, y se puede dar el caso de se escriban valores incorrectos o se sobrescriban. Pero se elige cuando para el use case no es tan necesaria la atomicidad en un nivel de costo beneficio.

Consistencia. (Consistency)

Se mueve de un estado válido a otro estado válido, por ejemplo al iniciar una transacción, hacer un lock del recurso, si se va la luz en medio del proceso, si uno de los procesos era de update o insert a la base de datos, al volver la luz el recurso compartido estaría en un estado inconsistente porque no llegó al final, es por esto que la consistencia trata de ir desde un estado válido a otro estado válido. El estado válido puede ser el mismo donde empecé, lo que también es conocido como un rollback, es deshacer todos los estados de una transacción. La consistencia también asegura que cuando se escriben datos deben ser correctos en base a los tipos de datos y la integridad referencial.

Si una tabla tiene un id de tipo small int de tamaño 2 bytes, si un usuario intenta guardar una O, sin consistencia se permite guardar el archivo, pero la consistencia no lo permite porque causaría inconsistencia.



Se podría obviar la consistencia, en un sistema de facturación por ejemplo, en el cual se tiene una tabla de productos, una tabla de factura, y una tabla que une a ambas, al comprar productos se genera una factura donde se guardan los id's y la cantidad de lo comprado, para acceder a la información se realiza un select con varios joins, lo cual genera varios problemas cuando existen muchos queries. En una compañía grande se podría cambiar el modelo con un documento donde se guarde id de factura, el total, productos guardando el id, nombre y precio de cada uno, como

los datos están juntos el usuario tiene menos retrasos y se ahorra cruces entre tablas, relaciones y filtros, lo que requiere mucho poder computacional.

Se sacrifica la consistencia para reducir el costo de la infraestructura y acceder a la información más rápido, con el mismo hardware de una base SQL, se puede aumentar el rendimiento transaccional.

Aislamiento. (Isolation).

Garantiza que dos o más transacciones corran simultáneamente sin que haya interferencia entre ellas, por lo que se debe realizar un control de concurrencia. Dicho control indica hasta qué momento el sistema soporta compartir recursos.

A nivel de bases de datos se tienen 2 tipos de lock, uno para lectura y otro para escritura. Requieren acceso exclusivo o compartido, se le llaman normalmente lock exclusivo o compartido. Se utilizan cuando requiero acceso exclusivo a un recurso compartido, pero solo un elemento puede estar leyendo dicho recurso a la vez, bloquea todos los demás procesos hasta que finalice.

Lock Type	Read	Write
Read	X	.
Write	.	.

Permite manejar control de concurrencia y transacciones paralelas. Esto le ingresa a la base de datos relacional, overhead porque se están revisando los log constantemente y blocking.

A nivel de blocking se tienen las fases:

- Two Phase(2PL)
- Expanding Phase
- Shrinking Phase

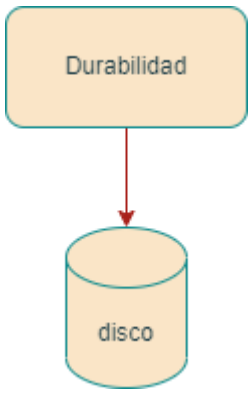
La multiversión indica que se pueden tener múltiples versiones de los datos almacenados.

Durabilidad. (Durability).

Una vez se logra correr una transacción atómicamente y se respeta la consistencia mientras se ejecuta todo de manera concurrente, se garantiza el aislamiento. En el momento en el que se llega al final de la transacción, normalmente se realiza un commit, este consiste en tomar el estado de la base de datos y lo guarde. El estado se guarda en disco. Se debe garantizar que para cada dato almacenado, este dato debe irse hasta disco, por que si solo se queda en memoria y por una situación como que se fue la luza, el dato se borra y es como si el dato nunca hubiera existido, por lo que la base de datos no va a ser consistente. En ese caso al recuperar la base de datos, esta indica que se aplicó una transacción pero en la base de datos desapareció. El estado de la base de datos debe ser persistente, una vez se ha hecho commit los debes deben ir a disco y seguir ahí. La forma de lograr que el estado de la base de datos persistente es metiendo los datos a discos.

Algunas bases de datos relacionales manejan el concepto de checkpoint, el cual escribe a disco para garantizar al durabilidad.

En una base de datos como un ndb cluster, cuyo funcionamiento consiste en mantener los datos en memoria, es una línea muy delgada en donde se garantiza que los datos van a ser durables y se garantiza la escritura de los datos a disco, la probabilidad de que se apaguen todas las replicas y se pierdan datos es muy baja.



De aquí nacen las base de datos NoSQL.