Instituto Tecnológico de Costa Rica

Base de Datos II - IC4302

Clase Virtual - Viernes 1 de setiembre 2023

Conceptos vistos en clase

- **Snapshot de la base datos:** Se puede ver como una versión de los datos que están en memoria.
- **Switchover:** Persona encargada de hacer el cambio manualmente de primario a secundario.
- **Connection string de una base de datos:** El URI, más credenciales para entrar a una base datos, se puede enviar una lista de servidores.

Repaso de Write Ahead Log

Si se tienen dos sistemas de base de datos, estos sistemas escriben en un archivo secuencial los cambios que entran a la base de datos. Cuando el cambio entra a la base de datos, Postgresql lo hace persistente en el storage de la base de datos.

- **Ventajas**:
    - Escribe la transacción al final de un archivo. Además, la escritura es más rápida
    - Cuando se hacen escrituras de forma secuencial se pueden utilizar discos de menor costo.

Ejemplo: Si la base de datos sufre alguna caída, y luego se restaura, revisa donde quedó el puntero y sigue leyendo los datos. Además, si en ese momento se estaba procesando, esa parte del log, la base de datos revisa que cambios se hicieron.

Una transacción en SQL no es atómica a nivel de hardware (No se ejecuta en un ciclo de reloj) y lo que puede suceder es que los datos sean inconsistentes y se realiza un snapshot data, en forma de ejemplo se podría ver que, en el transcurso de una transacción, la base de datos se cae y el sistema se percata de esto, se utilizan los datos del snapshot que son consistentes y recupera los datos perdidos en el proceso mencionado.

En temas de replicación, se transfiere desde un primario las transacciones que se ingresan en el Write Ahead Log, y pasa todas las transacciones. Cuando se hace replicación los datos se deben de sacar lo más rápido posible del servidor primario y llevarlo al servidor secundario.

Replicación en cascada: Cuando se posee varios servidores. ya sea uno primario y varios secundarios, la replicación fluye del primario al secundario y así seguidamente pasarlo a otro servidor secundario.

Locking

Se da algo que es conocido como **race condition**, que se da sobre un recurso compartido, cuando hay dos transacciones (Ejemplo, actualización en una base de datos).

Cuando se tiene un recurso compartido entre dos transacciones, el proceso de **Locking** lo que hace es cuando una transacción utiliza el recurso, da un acceso restringido únicamente a esa transacción y la otra transacción no podrá usar el recurso hasta que la primera transacción termine, y se aplica de nuevo lo que se mencionó anteriormente, la primera transacción se bloquea hasta que la segunda transacción termine.

Movimiento de Log entre servidores:

- **Asíncrono:** File based, se define un tamaño de log y conforme se llena, hace la transmisión del log. Una desventaja es que puede haber perdida de datos.
- **Síncrono:** Transmisión en tiempo real, tiene como problema que se gastan más recursos, ya que las transacciones se empaquetan.

Read Replica: Se define un servidor primario que recibe lecturas y escrituras, se replican y el read replica sirve información, el riesgo es que los datos pueden estar de sincronizados y afecta consistencia.

Logical Replication

Se tiene un servidor primario y varios servidores secundarios, el primario recibe actualizaciones y se aplica replicación lógica a nivel de tabla, el servidor primario publica un endpoint(Canal de información), que ayuda a conectar(suscripción) a ese endpoint varias réplicas, y esto genera snapshot con su última versión. Hay snapshot full (trae toda la tabla de la base de datos), y el incremental calcula diferencias que hay con el otro snapshot.

Cuando se agrega una nueva replica solicita los snapshots existentes. El problema es que se deben de procesar los datos antes de moverlos y causa perdida de datos. Se utiliza el Write Ahead Log para mover los datos de una forma muy efectiva.

Trigger Based Primary-Standby Replication

Forma en que la que podemos definir las transacciones de un servidor a otro. Se utiliza un trigger que se dispara cuando se necesita hacer la sincronización de replicación de datos, esto se usa cuando se trabaja con gran cantidad de datos, y si se tiene un proceso que corre por mucho tiempo en una réplica, uno se espera que termine para que replique, sino afecta el job.

SQL-Based Replication Middleware

Se tienen dos servidores, pero cada servidor no conoce la existencia del otro, por lo tanto, no existe replicación entre los dos servidores, en este ámbito no existe el concepto de replicación lógica y los ya mencionados.

Como lo dice su nombre Middleware, se tiene un actor en medio llamado Proxy, cuando el usuario envía un comando SQL, ese comando llega al proxy y se encarga de enviarle una copia del comando a los servidores y lo aplican en cada uno de sus servidores.

Synchronous Multimaster Replication

Se tienen dos servidores que tienen un recurso compartido, de los cuales el primer servidor recibirá dos transacciones que entran al servidor de base de datos, si se ejecutaron dos programas simultáneamente y llegan a memoria simultáneamente, los dos programas entran a la lista de programas listos para recibir el procesador simultáneamente, y al entrar de esa forma siempre uno va a quedar primero que el otro, si por ejemplo, si llegó primero la transacción dos, le coloca un lock al recurso compartido, y cuando a la transacción 1 le dan el procesador la transacción se bloquea, hasta que termine la transacción dos y se libere el recurso compartido, cuando esto sucede se va a serializar.

Si se poseen multimaster, la transacción 1 entro al servidor 1 y de igual forma la transacción 2 con el servidor 2, llega a un punto donde el servidor 1 le dice al servidor 2 que necesita el recurso compartido, y el servidor 2 también puede solicitar lo mismo simultáneamente. Por lo tanto de deben definir reglas, si ambos servidores necesitan el recurso compartido, se define que primero se le da al servidor 1, y estos mensajes se dan por la red, de cómo se trabajará con el sistema compartido, por medio de la red se consulta entre los dos servidores las reglas definidas, según todo lo mencionado con la sincronización requiere mucho tiempo, si no es por lo lineamentos ya dichos puede haber inconsistencia de datos, además genera overhead y delay, y aun así pone en muchos tiempo de espera a los usuarios.

Asynchronous Multimaster Replication

Si por ejemplo se tienen tres transacciones y las tres desean trabajar con el recurso compartido a la vez, se le permite que usen el mismo recurso y lo modifiquen al mismo tiempo, y esto genera inconsistencia en la base de datos, se definen de nuevo reglas para definir cómo resolver los conflictos y cuando no se resuelven, se bloquea la base de datos y notifica para que se resuelva el conflicto manualmente.