Nivel 2- Iteración 3 - Documentación Sistema de Producción ProdAndes

Jairo Emilio Bautista Mora, Nelson Andrés Sánchez Otálora

Manejo Transaccional de la Información

Universidad de los Ande, Bogotá, Colombia

{je.bautista10, na.sanchez162}@uniandes.edu.co

Fecha de presentación: Abril 15 de 2015

**Diseño de esquema de respaldo de Datos**

Para un esquema de respaldo de la información, se planea implementar un servidor que se encargue de la administración y almacenamiento de los Logs que está generando la aplicación con respecto a su comunicación con la base de Datos. Para el correcto funcionamiento de este, se tendrá en cuenta los siguientes aspectos:

* El servidor de Logs será una máquina de alta disponibilidad (99,9999% disponible al año).
* El servidor de la aplicación le enviará cada cierto tiempo un Heartbeat al servidor de Logs, con el fin de que este último valide la disponibilidad del servicio.
* Cada transacción que se realice en el servidor de aplicación tendrá un identificador único.
* Por cada sentencia ejecutada en una transacción por el servidor de la aplicación, este a su vez enviará por un canal seguro al servidor de Logs la información del ID de la transacción y la sentencia SQL.
* El servidor de Logs almacenará en una base de datos local la información del ID de transacción y sentencia SQL de forma secuencial, según sea enviado por el servidor de la aplicación.
* Cada cierto tiempo de periodo determinado, el servidor de aplicación enviará un checkpoint al servidor de Logs. Validando que todas las instrucciones SQL que ha ejecutado han sido exitosas.
* El servidor de Logs manejará una copia de la base de datos de la aplicación. Y cada cierto periodo de tiempo, actualizará la información de esta con los Logs que no haya ejecutado aún.

De esta forma, una vez establecido el protocolo entre servidor de aplicación y Logs, en caso de haber un fallo de disponibilidad a nivel de base de datos. El servidor de aplicación enviará una señal al servidor Logs para entrar en modo recuperación. O en caso de que no llegue el Heartbeat del servidor de aplicación (lo cual indica que no está disponible), entrará en modo de recuperación. En el modo de recuperación, dependiendo si es el servidor de la aplicación o la base de datos principal, se comportará de la siguiente forma:

* En caso de que el servidor de aplicación sea el que no esté disponible, el servidor Log realizará las ejecuciones que tenga en los Logs después del último CheckPoint; haciendo rollback a todas las transacciones que no hayan finalizado en commit.
* En caso de ser la base de datos la que no esté disponible, el servidor de la aplicación seguirá ejecutándose ordinariamente. La diferencia radicará en que el servidor de Logs adicional al almacenamiento de las sentencias SQL, marcará un CheckPoint especial y ejecutará los cambios que lleguen del servidor de la aplicación en la base de datos local. Una vez que el servidor de Logs detecte que la base de datos esté bien, realizará las actualizaciones correspondientes desde el CheckPoint especial marcado anteriormente.

De esta forma, se puede recuperar los datos en caso de un fallo de disponibilidad.

**Documentación nuevos requerimientos:**

* Requerimiento funcional 17:

Este requerimiento funcional recibe como única información la que trabajar el código de la estación de producción a cambiar el estado. Se verifica cual estado tenía antes del cambio y con eso empieza a trabajar manejando dos casos diferentes y excluyentes, la estación estaba activada o desactivada.

En el primer caso, si la estación estaba activada, busca las etapas que estaban en espera de esa estación o era la que estaba activa. Luego cuenta las etapas en espera y asignadas a las estaciones que están activas que sean diferentes a esta. Luego a las etapas de producción que buscamos anteriormente, las vamos asignando con un método de balanceo a las otras estaciones. Finalmente desactivamos la estación y le indicamos que ya no tiene ninguna etapa asignada.

En el segundo caso, si la estación estaba desactivada, busca las estaciones activadas contando cuantas etapas estaban asignadas a ella. Luego va balanceando la carga desasignado alguno de los que estaban asignados a ellos. Luego pone uno de los que desasigno de las estaciones activas en activo para la que vamos a activar, y las demás en espera de esa, y cambiamos el estado de la que vamos a activar por Activo.

El método de balanceo de carga que implementamos fue ir aumentando a los que tengan menor carga en el caso de desactivación, y vamos disminuyendo la carga de los activos hasta que sea menor a un promedio.

En caso de que se generen abrazos mortales u alguna otra anomalía de concurrencia durante las actualizaciones, hace rollback deshaciendo todos los cambios. Las consultas para verificar la información necesaria para trabajar se hacen siguiendo un nivel de aislamiento de read committed.

* Requerimiento funcional de consulta 5:

Este requerimiento funcional de consulta solo era necesario consultar varias tablas al tiempo, mediante joins. Las tablas a consultar eran: pedidos, producto, materiales, usuario y cliente. Para evitar lecturas malas, se usó nivel de aislamiento de read committed.

* Requerimiento funcional de consulta 6:

Este requerimiento funcional de consulta solo era necesario consultar varias tablas al tiempo, mediante joins. Las tablas a consultar eran: pedidas, usuario y cliente. Para evitar lecturas malas, se usó nivel de aislamiento de read committed.

* Requerimiento funcional de consulta 7:

Este requerimiento funcional de consulta solo era necesario consultar varias tablas al tiempo, mediante joins. Las tablas a consultar eran: pedidos, producto, materiales, usuario y cliente. Para evitar lecturas malas, se usó nivel de aislamiento de read committed.

**Justificacion transaccionalidad:**

Se uso un nivel de isolación de read commited, para no bajar el desempeño, y que los clientes solo vean la información correctamente actualizada. Para algunos caso se crearon savepoints para poder hacer rollback.