**PUNTO 6**

Para realizar un escenario de prueba que implicaba la ejecución concurrente de la consulta de operaciones realizadas sobre una cuenta (RFC4) y el registro de una operación sobre la misma cuenta (RF6), ambas con el nivel de aislamiento SERIALIZABLE, se consideró que este nivel de aislamiento es el más restrictivo y pretende que cada transacción se ejecute de manera aislada de las demás. Aquí se describe cómo se estructuró el escenario:

**Pasos para la Ejecución Concurrente de RFC4 y RF6**

1. Inicio de RFC4: Se inició una consulta para obtener las operaciones realizadas en una cuenta específica en los últimos 30 días. La transacción se estableció con un nivel de aislamiento SERIALIZABLE, lo que significaba que una vez comenzada, no vería los cambios que otras transacciones realizaran después de este punto.
2. Simulación de retardo en RFC4: Durante la ejecución de RFC4, se incluyó un retardo artificial (por ejemplo, Thread.sleep(30000) en Java) para simular una operación de larga duración y permitir la concurrencia.
3. Ejecución concurrente de RF6: Mientras RFC4 aún estaba en ejecución (durante los 30 segundos de retardo), se intentó realizar una operación de consignación en la misma cuenta desde otro hilo o sesión de usuario. Esta transacción también era SERIALIZABLE.
4. Finalización de RFC4 y RF6: RFC4 completó su ejecución y devolvió los resultados. Posteriormente, RF6 intentó completar la consignación.

**Descripción de lo Sucedido**

*Bloqueo debido a SERIALIZABLE:* Dado que ambas transacciones eran SERIALIZABLE, RF6 (transacción de consignación) probablemente quedó bloqueada hasta que RFC4 (consulta de operaciones) completó su ejecución. Esto se debió a que el nivel SERIALIZABLE intenta prevenir la condición de lectura fantasma, lo que implica que ninguna transacción puede ver los cambios en los datos que están siendo leídos por otra transacción hasta que esta última concluya.

**Comportamiento esperado:** RF6 esperó a que RFC4 se completara antes de poder ejecutar la consignación.

**Resultado Presentado por RFC4**

*Resultado de RFC4:* Mostró todas las operaciones realizadas en los últimos 30 días hasta el punto de inicio de la transacción. No incluyó la consignación realizada por RF6 si esta fue iniciada después de que comenzó RFC4, ya que el nivel SERIALIZABLE asegura que las transacciones son completamente aisladas.

*Visibilidad de la consignación en RFC4:* La consignación realizada durante RF6 no apareció en los resultados de RFC4 debido al aislamiento de la transacción.

**Conclusión**

En este escenario, RFC4 completó su ejecución y mostró los datos que existían antes de su inicio, sin incluir la nueva consignación realizada por RF6. Posteriormente, una vez que RFC4 finalizó, RF6 pudo proceder, actualizando el saldo y registrando la nueva operación en la base de datos. Esta operación fue visible para futuras consultas una vez completada.

Este tipo de prueba fue crucial para entender y validar el comportamiento de las aplicaciones en entornos de alta concurrencia y resultó esencial para sistemas que requieren integridad y consistencia de datos, como los sistemas bancarios.

**PUNTO 7**

Para el escenario de prueba de concurrencia donde se ejecutó primero el RFC5 – CONSULTA DE OPERACIONES REALIZADAS SOBRE UNA CUENTA – bajo el nivel de aislamiento READ COMMITTED, y casi simultáneamente, se ejecutó RF6 – REGISTRAR OPERACIÓN SOBRE CUENTA, para realizar una consignación, se siguieron los siguientes pasos y se observaron ciertos comportamientos:

**Pasos para la Ejecución Concurrente de RFC5 y RF6**

1. **Inicio de RFC5:** Se inició una consulta de operaciones realizadas sobre una cuenta en los últimos 30 días con el nivel de aislamiento READ COMMITTED. Esto permitió que la consulta viera solo las operaciones que habían sido confirmadas hasta el momento de su ejecución.
2. **Simulación de retardo en RFC5:** Para permitir la concurrencia, se simuló un retardo en la consulta de RFC5, manteniendo la consulta activa por un período extendido.
3. **Inicio de RF6 durante RFC5:** Mientras RFC5 aún estaba en ejecución, se inició RF6 para registrar una consignación en la misma cuenta. Dado que RF6 estaba configurada con un nivel de aislamiento SERIALIZABLE, intentó realizar cambios que requerían un alto grado de aislamiento.
4. **Finalización de ambas transacciones:** RF6 completó la transacción de consignación mientras RFC5 aún estaba en proceso.

**Descripción de lo Sucedido**

* **Independencia de RF6 respecto a RFC5:** RF6 no tuvo que esperar a que RFC5 terminara gracias al nivel de aislamiento READ COMMITTED de RFC5, que no bloquea lecturas de datos confirmados por otras transacciones. RF6 pudo proceder con la inserción de la nueva operación sin demoras.
* **Visibilidad de la nueva consignación en RFC5:** Debido al aislamiento READ COMMITTED, es posible que RFC5 no viera la consignación realizada por RF6 si la consignación se confirmó después de que RFC5 comenzara su ejecución.

**Resultado Presentado por RFC5**

**Datos mostrados por RFC5:** La consulta mostró las operaciones realizadas sobre la cuenta en los últimos 30 días hasta el momento de su inicio. La nueva consignación realizada por RF6 probablemente no fue visible en los resultados de RFC5, ya que esta transacción se confirmó después del inicio de la consulta de RFC5.

**Conclusión**

El escenario demostró que bajo el nivel de aislamiento READ COMMITTED, las consultas pueden continuar sin interrupciones mientras otras transacciones realizan cambios en los datos concurrentemente. Además, mostró que RFC5 no necesariamente reflejará los cambios hechos por transacciones concurrentes que se confirmen después de su inicio. Esto es crucial en entornos donde la frescura de los datos es menos crítica que la disponibilidad y el rendimiento, como en informes de operaciones no críticos o análisis secundarios.