**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.