

SISTEMAS TRANSACCIONALES - ISIS2304

PROYECTO 1 ENTREGA 3

Laura Valentina Ceron Pulgarin *Código: 202214973*

l.ceronp@uniandes.edu.co

Franklin Smith Fernandez Romero *Código: 202215103*

f.fernandezr@uniandes.edu.co

Andres Mateo Chilito Avella *Código: 202214992*

a.chilitoa@uniandes.edu.co

Este documento presenta la solución para el proyecto número uno entrega tres de la materia *Sistemas Transaccionales*.

Universidad de los Andes

Bogotá - Colombia

30 de abril de 2024

BANCO DE LOS ALPES

1. En nuestra segunda iteración en la aplicación "Banco de los Andes", nuestro equipo se ha enfocado en la optimización y corrección de funcionalidades críticas para garantizar la integridad de las transacciones y la robustez operativa. Este proceso de mejora fue impulsado por la necesidad de fortalecer los mecanismos de seguridad y eficiencia en el manejo de transacciones financieras a gran escala, enfatizando en las propiedades ACID. Durante la fase de diagnóstico, identificamos fallos específicos en el manejo de las sentencias HTML que se utilizan para la generación dinámica de contenido en la interfaz de usuario. Estos fallos no solo representaban un riesgo potencial de seguridad, sino que también afectaban la fluidez y la experiencia del usuario. Para abordar estos problemas, implementamos una serie de correcciones que incluyen la limpieza de entradas y la validación de datos del lado del servidor, esto mejora mucho la fluidez y seguridad en la app, lo igual que garantiza ACID. Además, se revisaron las consultas SQL utilizadas para la interacción con la base de datos, optimizando índices y reescribiendo consultas para mejorar la eficiencia y reducir los tiempos de respuesta (esto como algo extra en nuestra iteración). Posteriormente, nuestro equipo llevó a cabo un conjunto comprehensivo de pruebas automatizadas y manuales, incluyendo pruebas de regresión y pruebas de carga para verificar la estabilidad y el rendimiento bajo condiciones de uso intensivo. Estas pruebas fueron diseñadas para cubrir todos los escenarios de uso posibles, asegurando que cada componente funcionara de acuerdo a las especificaciones técnicas y cumpliendo con los estándares de calidad más exigentes. Los resultados obtenidos tras esta fase de pruebas demostraron una mejora significativa en el rendimiento de la aplicación, con un rendimiento optimizado en superior del 90 %. Estos logros no solo reflejan el éxito de las intervenciones técnicas realizadas, sino que también reafirman nuestro compromiso con la mejora continua.

2. Nivel de aislamiento. READ COMMITTED:

- a) **Tiempo 1 Sesión 1:** Durante este tiempo se inicializó la sesión con nivel de aislamiento read committed".
- b) **Tiempo 2 Sesión 1:** Se eligió la cuenta "1002" para consignar un millón de pesos con su respectiva sentencia sql.

	SALDO		SALDO
1	82000	1	1082000

Figura 1: A la derecha el antes de la consignación y a la izquierda el después

- c) **Tiempo 3 Sesión 1:** Se realiza el registro en el log de la consignación.

ID	TIPO_OPERACION CUENTA	NUM CUENTA	OPERACION_BANCARIA
1	20002 CONSIGNACION	1002	10002
2	20006 CONSIGNACION	1002	10004

Figura 2: Se registra en el log con id 20006

- d) **Tiempo 4 Sesión 1 y 2:** Se realiza simultáneamente en sesión 1 con la cuenta "1004", un retiro de 50.000. En sesión 2, se define el nivel de aislamiento en Read committed".

	SALDO		SALDO
1	79500		
1	29500		

Transaction ISOLATION correcto.

Figura 3: Se observa el antes y después de la cuenta "1004", junto con el nivel de aislamiento respectivo en sesión 2

- e) **Tiempo 5 Sesión 1:** Se inserta en el log de operaciones el registro del retiro de los 50.000.

ID	TIPO_OPERACION CUENTA	NUM CUENTA	OPERACION_BANCARIA
1	20004 CONSIGNACION	1004	10004
2	20007 RETIRO	1004	10002

Figura 4: Se registra el retiro en el log con id 20007.

- f) **Tiempo 6 Sesión 2:** Para cuenta 1, es decir, para "1002", se intenta descontar 30.000 del saldo total. Sin embargo, sucede un bloqueo de recursos entre transacciones y sesión 2 no puede seguir avanzando hasta que sesión 1 libere los recursos.

The screenshot shows a software interface with a toolbar at the top. The central area displays a progress bar with the text "Tarea de ScriptRunner" and a status indicator showing a red circle with a white 'X' and a yellow double vertical bar. This indicates that a script or task is currently executing.

Figura 5: Bloqueo de recursos.

- g) **Tiempo 7 Sesión 1:** Sesión 1 realiza **commit** de sus cambios, liberando los recursos, por lo tanto, sesión 2 puede avanzar.



The screenshot shows a database interface with two rows of data. The first row is a header with columns labeled 'SALDO' and 'NUM CUENTA'. The second row contains the value '1 1052000'. Below this, a message 'Confirmación terminada.' is displayed.

SALDO	NUM CUENTA
1 1052000	

Figura 6: Commit sesión1, sesión2 avance.

- h) **Tiempo 8 Sesión 2:** Para cuenta 1 ("1002") se realiza registro en el log del retiro.

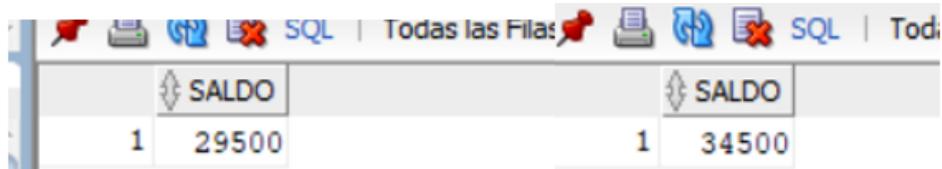


The screenshot shows a database interface with three rows of data in a table. The columns are labeled 'ID', 'TIPO_OPERACION CUENTA', 'NUM CUENTA', and 'OPERACION_BANCARIA'. The data rows are: 1 20002 CONSIGNACION, 1002, 10002; 2 20006 CONSIGNACION, 1002, 10004; and 3 20008 RETIRO, 1002, 10004.

ID	TIPO_OPERACION CUENTA	NUM CUENTA	OPERACION_BANCARIA
1	20002 CONSIGNACION	1002	10002
2	20006 CONSIGNACION	1002	10004
3	20008 RETIRO	1002	10004

Figura 7: Se registra retiro en el log con id 20008.

- i) **Tiempo 9 Sesión 2:** Para cuenta 2 ("1004") se realiza suma de intereses de 5.000.

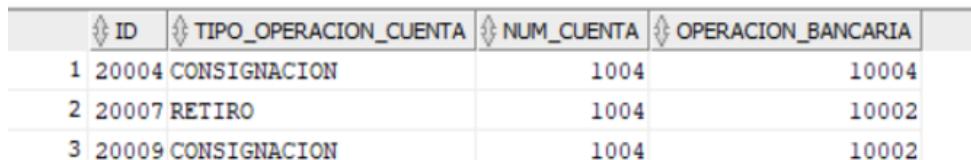


The screenshot shows a database interface with two rows of data. The first row is a header with columns labeled 'SALDO'. The second row contains the value '1 29500'. To the right, another row shows the value '1 34500' under the same 'SALDO' header.

SALDO
1 29500
1 34500

Figura 8: Antes de después de la suma de intereses.

- j) **Tiempo 10 Sesión 2:** Para cuenta 2 ("1004") se realiza registro en el log de la consignación.



The screenshot shows a database interface with three rows of data in a table. The columns are labeled 'ID', 'TIPO_OPERACION CUENTA', 'NUM CUENTA', and 'OPERACION_BANCARIA'. The data rows are: 1 20004 CONSIGNACION, 1004, 10004; 2 20007 RETIRO, 1004, 10002; and 3 20009 CONSIGNACION, 1004, 10002.

ID	TIPO_OPERACION CUENTA	NUM CUENTA	OPERACION_BANCARIA
1	20004 CONSIGNACION	1004	10004
2	20007 RETIRO	1004	10002
3	20009 CONSIGNACION	1004	10002

Figura 9: Se registra consignación en el log con id 20009.

- k) **Tiempo 11 Sesión 1:** Se consulta el saldo de cuenta 1 ("1002") y cuenta 2 ("1004") en sesión 1.

	SALDO		SALDO
1	1082000	1	29500

Figura 10: Saldo cuenta 1 y saldo cuenta 2.

- l) **Tiempo 12 Sesión 2:** Sesión 2 realiza **commit** de sus cambios, guardando las operaciones realizadas.
- m) **Tiempo 13 Sesión 1 y 2:** Se consulta el saldo de cuenta 1 ("1002") y cuenta 2 ("1004") en sesión 1 y 2. En ambas cuentas quedan los mismos saldos en tiempo 13, pues al realizar commit la sesión 2, los cambios que realizó se hacen permanentes en la base de datos.

	SALDO		SALDO
1	1052000	1	34500

Figura 11: Saldo cuenta 1 y 2, en sesión 1 y 2.

- n) **SERIALIZABLE Tiempo 1 Sesión 1:** Durante este tiempo se inicializó la sesión con nivel de aislamiento "Serializable".
- ñ) **Tiempo 2 Sesión 1:** Se eligió la cuenta "1001" para consignar un millón de pesos con su respectiva sentencia sql.

	SALDO		SALDO
1	2005000	1	3005000

Figura 12: Antes y después de la consignación.

- o) **Tiempo 3 Sesión 1:** Se realiza el registro en el log de la consignación.

ID	TIPO_OPERACION CUENTA	NUM CUENTA	OPERACION_BANCARIA
1	20001 RETIRO	1001	10001
2	20005 CONSIGNACION	1001	10003
3	20015 CONSIGNACION	1001	10004

Figura 13: Se registra la consignación en el log con id 20015.

- p) **Tiempo 4 Sesión 1 y 2:** Se realiza simultáneamente en sesión 1 con la cuenta "1003", un retiro de 50.000. En sesión 2, se define el nivel de aislamiento en "Serializable".



Transaction ISOLATION correcto.

Figura 14: Se observa el saldo de cuenta 2 antes y después, junto con el nivel de aislamiento de sesión 2.

- q) **Tiempo 5 Sesión 1:** Se intenta realizar un registro en el log del retiro. Sin embargo, se observó un error de violación de restricción única, al ejecutar transacciones concurrentes en dos sesiones distintas. Inicialmente se pensó que el error se debía a la inserción de un dato duplicado en la tabla "operacionescuenta", esto no era así, posiblemente se debe a restricciones de integridad.

```
Error que empieza en la linea: 20 del comando :
INSERT INTO OPERACIONESCUENTA (ID, TIPO_OPERACION CUENTA, NUM CUENTA, OPERACION_BANCARIA)
VALUES (20005, 'RETIRO', 1003, 10005)
Informe de error -
ORA-00001: restricci n  nica (ISIS2304B15202410.SYS_C001259450) violada
```

Figura 15: Se observa el error (ORA-00001).

- r) **Tiempo 6 Sesión 2:** Se realiza el descuento de 30.000 en sesión 2 cuenta 1.



Figura 16: Saldo con 30.000 de retiro.

- s) **Tiempo 7 Sesión 1:** Se realiza **commit** exitoso en sesión 1.
t) **Tiempo 8 Sesión 2:** Se realiza el registro del retiro en el log de cuenta 1.

ID	TIPO_OPERACION CUENTA	NUM CUENTA	OPERACION_BANCARIA
1	20001 RETIRO	1001	10001
2	20010 RETIRO	1001	10002
3	20005 CONSIGNACION	1001	10003
4	20015 CONSIGNACION	1001	10004

Figura 17: Se observa el registro en el log con id 20010.

- u) **Tiempo 9 Sesión 2:** Se realiza la suma de los intereses de 5.000.

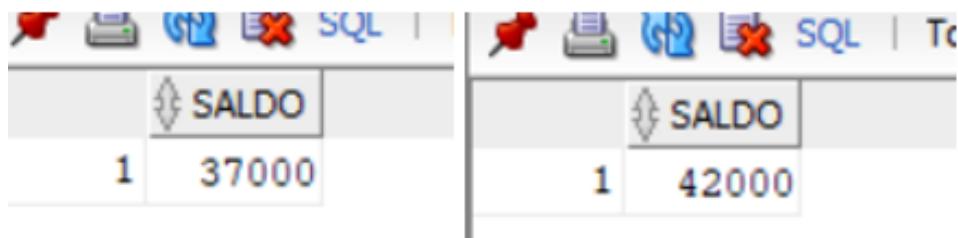


Figura 18: Se observa el antes y después de los intereses.

- v) **Tiempo 10 Sesión 2:** Se realiza el registro en el log de la consignación.

ID	TIPO_OPERACION CUENTA	NUM CUENTA	OPERACION_BANCARIA
1	20003 RETIRO	1003	10003
2	20011 CONSIGNACION	1003	10003

Figura 19: Se observa el registro de la consignación con id 20011.

- w) **Tiempo 11 Sesión 1:** Se realiza la consulta del saldo respectivo en cuentas 1 y 2.



Figura 20: Se observa el saldo actual de las dos cuentas en sesión 1.

- x) **Tiempo 12 Sesión 2:** Se realiza **commit** exitoso en sesión 2.
- y) **Tiempo 13 Sesión 1 y 2:** Se realiza la consulta del saldo respectivo en cuentas 1 y 2 en ambas sesiones. Podemos notar que en ambas sesiones el saldo de las dos cuentas es exactamente el mismo. Esto se debe a que sesión 1 realizó commit primero, después sesión 2 trabajo sobre los cambios confirmados en la base de datos por sesión 1 y cuando sesión 2 realizó su respectivo commit, los cambios quedaron permanentemente guardados en la base de datos.



Figura 21: Se observa el saldo actual de las dos cuentas en sesión 1 y 2.

3. Con la reciente iteración de mejoras y correcciones para la aplicación "Banco de los Andes", se han introducido nuevos requerimientos funcionales que fortalecen la gestión transaccional y la integridad de la información. A continuación, detallaremos la utilidad de estos requerimientos. El primer requerimiento de funcional de consulta permite a los usuarios consultar todas las operaciones realizadas en una cuenta específica durante los últimos 30 días, garantizando un nivel de aislamiento transaccional 'SERIALIZABLE'. Este nivel de aislamiento es el más estricto y se implementa para prevenir anomalías como lecturas sucias, no repetibles y fantasmas, asegurando así la consistencia de los datos durante la consulta. La consulta se ha diseñado para ejecutar un rollback automático si no se puede completar, informando adecuadamente al usuario sobre cualquier fallo. Además, se ha establecido un temporizador de 30 segundos para las pruebas de concurrencia, asegurando que el sistema pueda manejar adecuadamente múltiples solicitudes simultáneas. Con respecto al segundo requerimiento funcional permite a los usuarios consultar operaciones recientes en sus cuentas. Sin embargo, este requiere un nivel de aislamiento 'READ COMMITTED', que aunque previene lecturas sucias, permite ciertas anomalías como lecturas no repetibles y fantasmas. Este nivel es generalmente más flexible y adecuado para entornos donde la consistencia absoluta no es crítica, pero la eficiencia es prioritaria. Finalmente, el ultimo requerimiento funcional implica la modificación de los componentes de la aplicación que gestionan las operaciones de retiro, consignación y transferencia entre cuentas. Cada operación debe ser manejada como una transacción independiente, asegurando que todas las actividades sean atómicas, consistentes, aisladas y duraderas (ACID). Este enfoque transaccional es vital para mantener la integridad de los datos, especialmente en un entorno bancario donde la exactitud y la seguridad son primordiales. Se ha especificado que cada transacción realice un commit solo si ambas sentencias (registro en el log y actualización del saldo) se ejecutan exitosamente; de lo contrario, se realiza un rollback. La implementación de estos requerimientos no solo mejora la funcionalidad y la seguridad de la aplicación, sino que también asegura

que los procesos cumplan con los estándares modernos de aplicaciones financieras. Cada uno de estos nuevos requerimientos ha sido cuidadosamente documentado en el código, proporcionando una referencia clara para futuros desarrollos y mantenimiento. Estos cambios reflejan nuestro compromiso con la mejora continua y la adaptación a las necesidades cambiantes de nuestros usuarios, asegurando que "Banco de los Andes" se mantenga a la vanguardia en tecnología bancaria.

4. Escenario de prueba de concurrencia 1 "SERIALIZABLE".

- ◊ **Pasos para la ejecución:** Inicialmente, un usuario realiza la ejecución del RFC4, es decir, realiza una consulta de las operaciones realizadas sobre una cuenta, eligiendo la cuenta deseada. Posterior a ello, antes de que pasen los 30 segundos de la ejecución de RFC4, el usuario en otra instancia podrá ejecutar el componente de RF6, es decir, registrar una operación sobre una cuenta para realizar una consignación en la misma cuenta en la que se está trabajando.
- ◊ **Descripción de lo sucedido:** Cuando el usuario realiza la consulta (RFC4) obtiene las operaciones realizadas sobre la cuenta en los últimos 30 días. Es importante mencionar que, mientras se está realizando esta consulta, cualquier operación realizada en la cuenta, como la consignación, se verá reflejada en los resultados de la consulta solo después de que RFC4 se haya completado. Al mismo tiempo, el usuario realiza una consignación en la misma cuenta (RF6). Aunque esta operación se está llevando a cabo al mismo tiempo que la consulta del RFC4, el nivel de aislamiento garantiza que ambas transacciones se ejecuten consistentemente y que los resultados sean coherentes y sin ningún fantasma.
- ◊ **Ejecución:**

5. Escenario de prueba de concurrencia 2 READ COMMITTED".

- ◊ **Pasos para la ejecución:** Inicialmente, un usuario realiza la ejecución del RFC5, es decir, realiza una consulta de las operaciones realizadas sobre una cuenta, eligiendo la cuenta deseada, teniendo en cuenta que se encuentra en un nivel de aislamiento read committed". Posterior a ello, antes de que pasen los 30 segundos de la ejecución de las consultas de las operaciones sobre la cuenta, el usuario en otra instancia podrá ejecutar el componente de RF6, es decir, registrar una operación sobre una cuenta para realizar una consignación en la misma cuenta en la que se está trabajando.
- ◊ **Descripción de lo sucedido:**

El resultado del RFC5 muestra las operaciones realizadas en la cuenta durante el último mes, incluyendo la consignación efectuada concurrentemente al ejecutar RF6. Esto puede reflejar operaciones anteriores a la finalización de la consignación, según el momento exacto de la consulta y la confirmación de la consignación.

Con el nivel de aislamiento READ COMMITTED, RF6 no necesita esperar a que RFC5 termine, ya que puede visualizar cambios de otras transacciones una vez confirmados.

RFC5 presenta filas adicionales que aparecen debido a "fantasmas", resultando de inserciones o actualizaciones concurrentes. Las consignaciones de RF6 podrían no reflejarse de inmediato debido a estos "fantasmas".
- ◊ **Ejecución:**

Inserción

Inserción

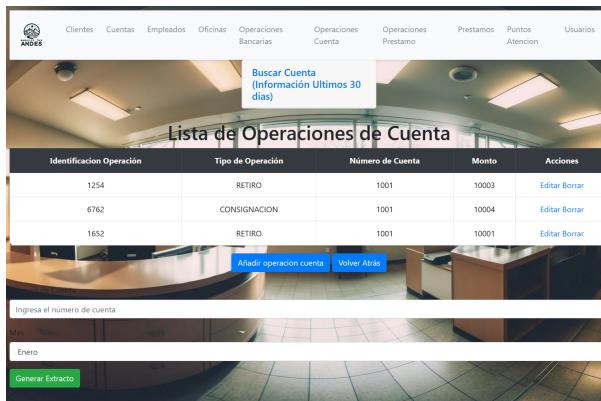


Figura 22: Estado inicial (Serializable)

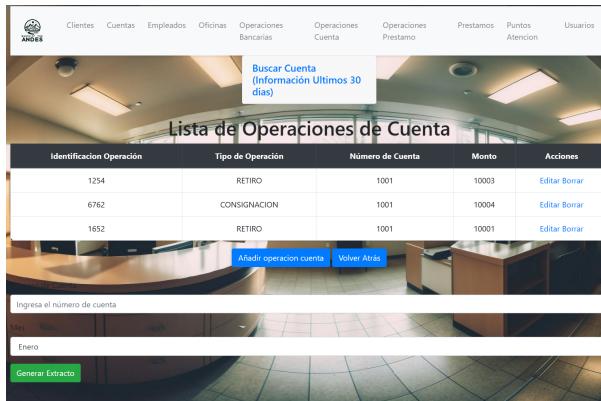


Figura 23: Estado Final (Serializable)

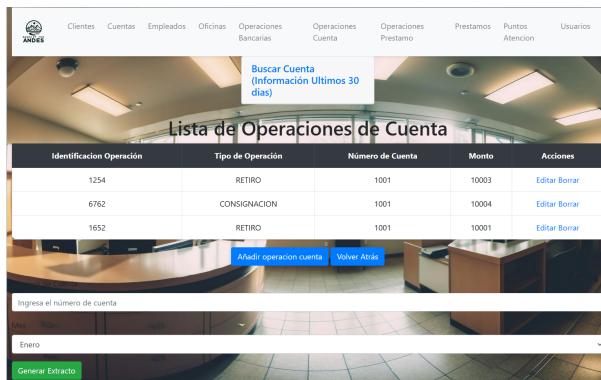


Figura 24: Estado Inicial (Read Committed)

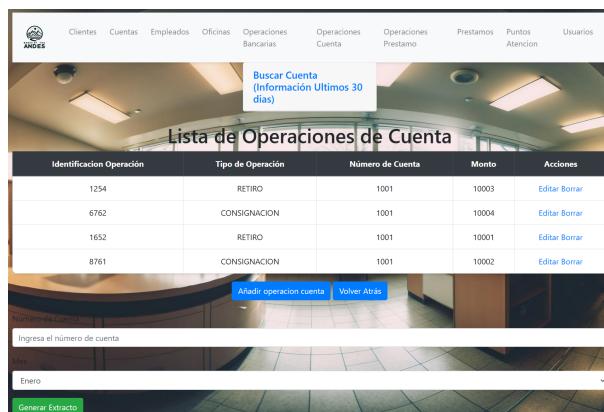


Figura 25: Estado Final (Read Committed)