Sistemas transaccionales - Entrega 3

Grupo 4

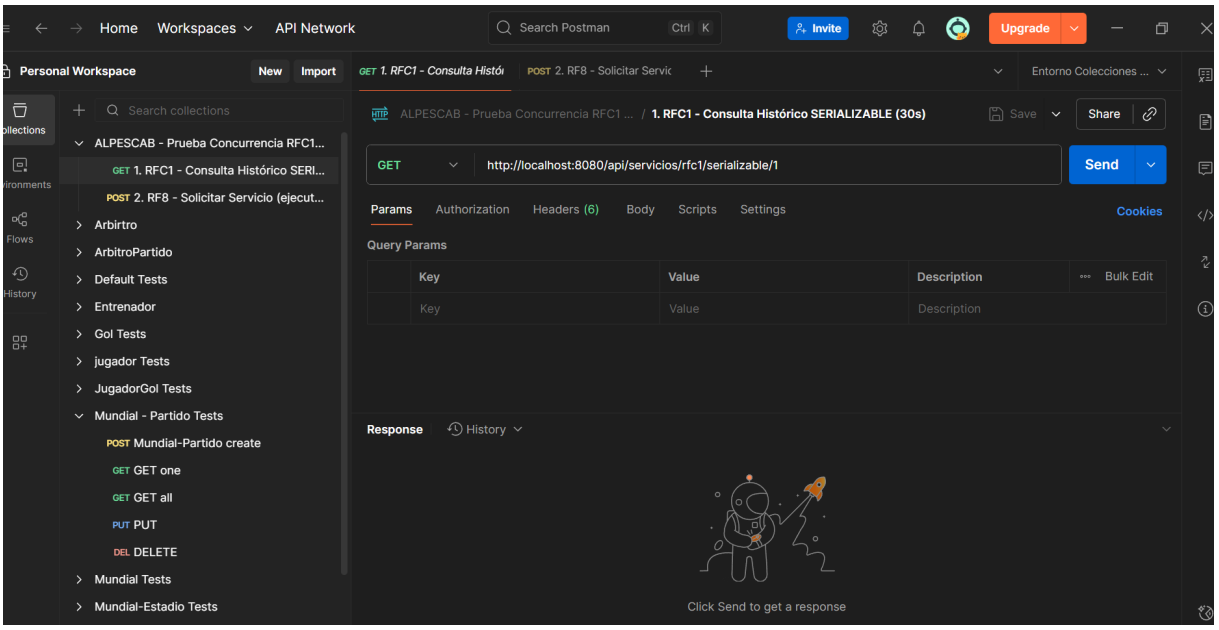
Aicardy Pacheco, Juan Angel
Castillo Alvarez, Mariana
Rodriguez Paez, Maria Camila

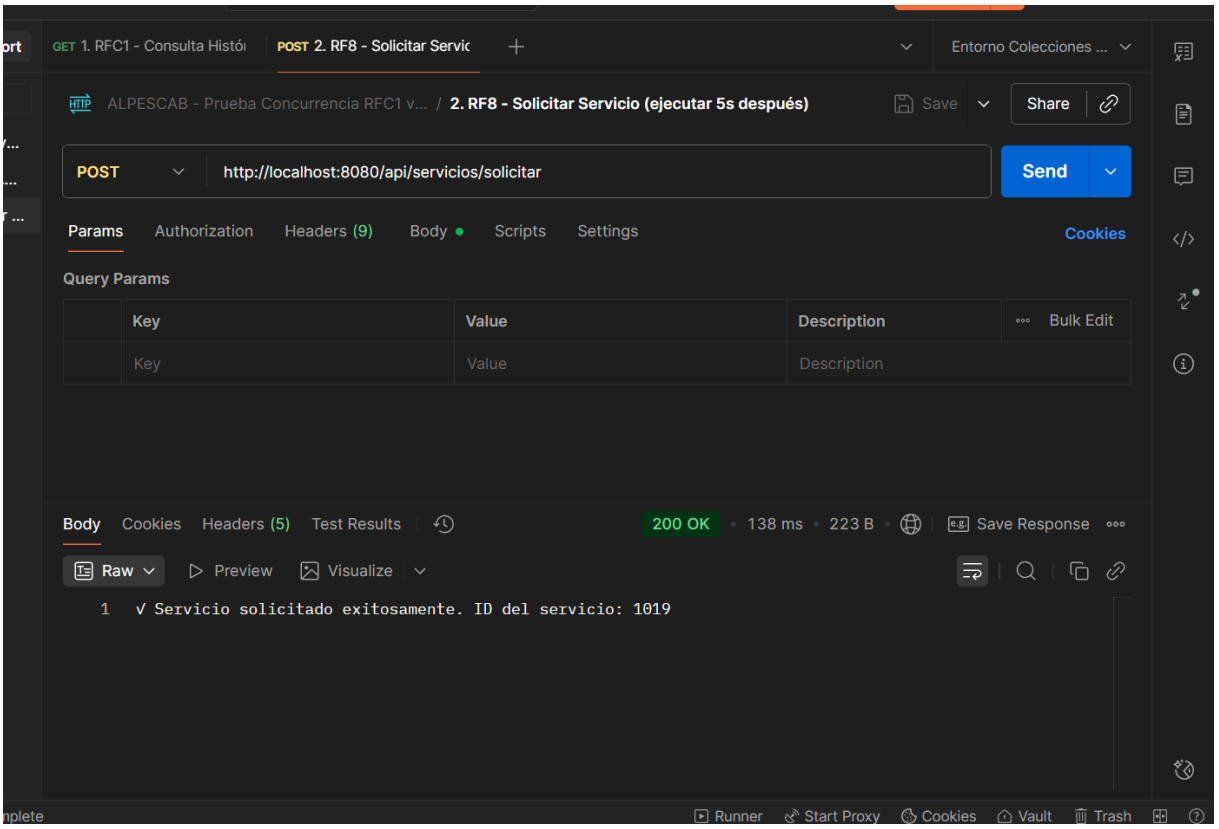
PUNTO 4

Escenario de concurrencia probado:

Se ejecutó RFC1 con nivel de aislamiento SERIALIZABLE, que realiza dos consultas separadas por 30 segundos. Durante ese intervalo, se ejecutó RF8 que insertó un nuevo servicio en la base de datos.

pruebas:





RFC1	Tiempo	RF8
Inicia transacción y realiza primera consulta	t1	
Espera (sleep de 30 segundos)	t2	
	t3	Inicia transacción y solicita nuevo servicio

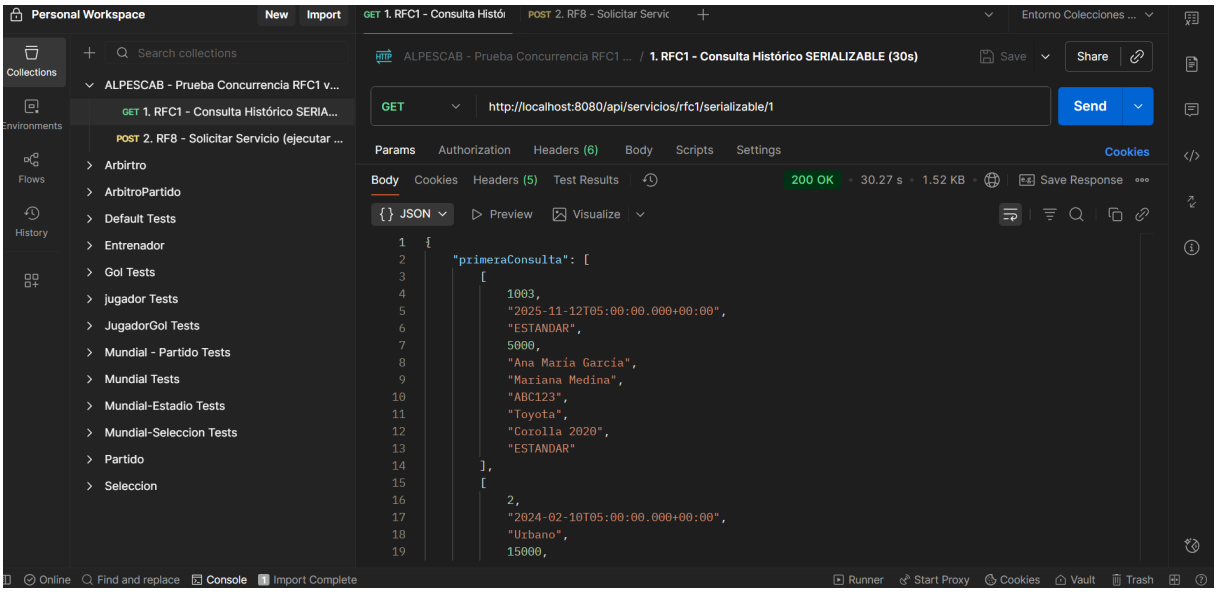
	t4	**Registra el servicio y finaliza transacción**
Realiza segunda consulta y reporta resultados	**t5**	

Descripción de lo sucedido:

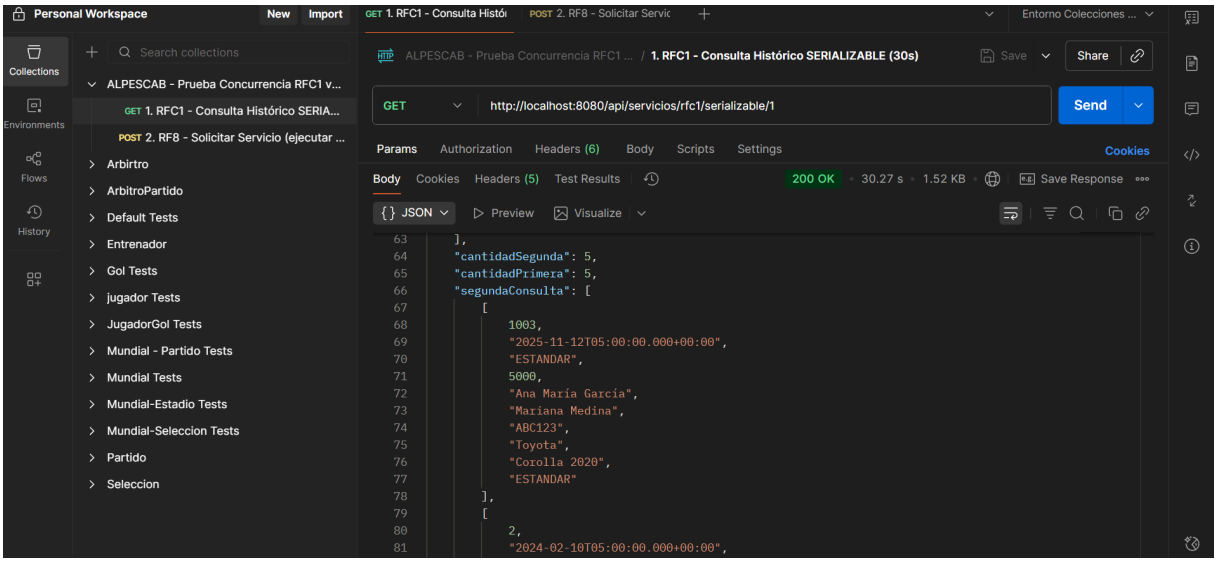
No, el componente que implementa RFC1 no necesita esperar a que termine la ejecución de RF8 para registrar la orden de servicio, ni RF8 necesita esperar a que termine RFC1 para registrar una nueva orden. Bajo el nivel de aislamiento SERIALIZABLE, cada transacción (RFC1 y RF8) opera como si fuera la única ejecutándose; es decir, cada una ve una "foto" de los datos confirmados antes de que comenzó su propia transacción.

RESULTADO PRESENTADO POR RFC1

Primera consulta



Segunda Consulta

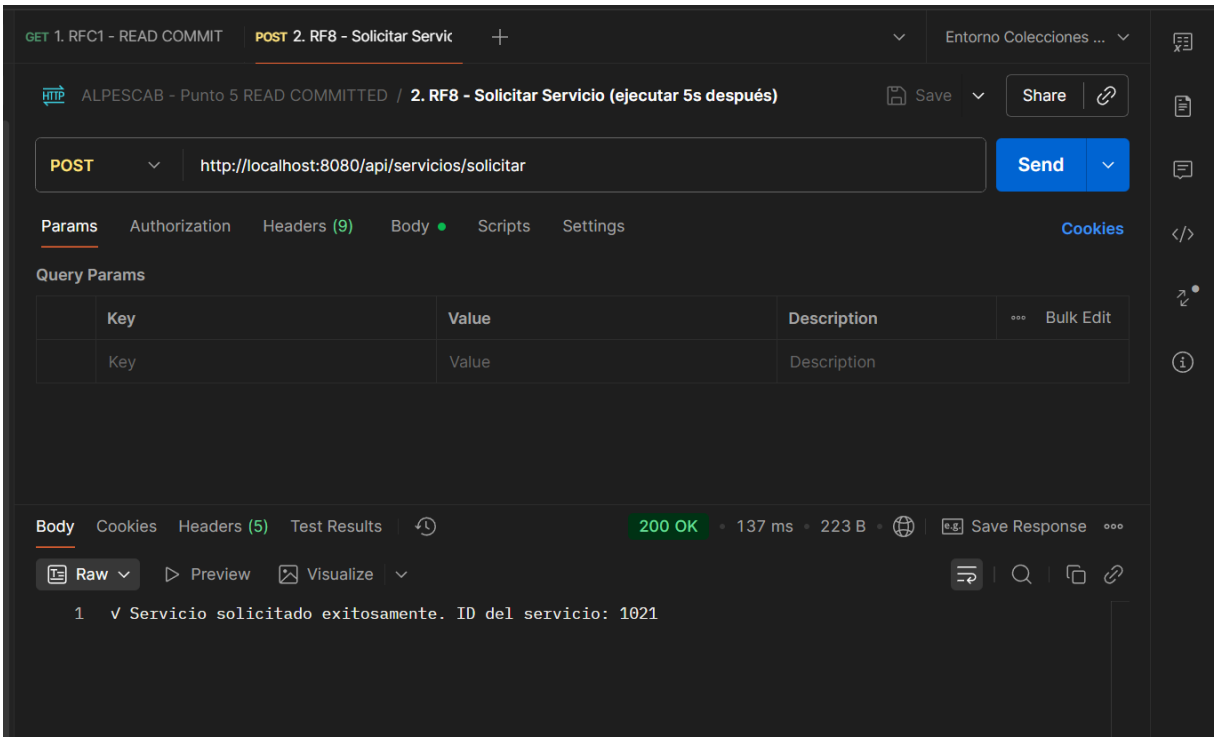


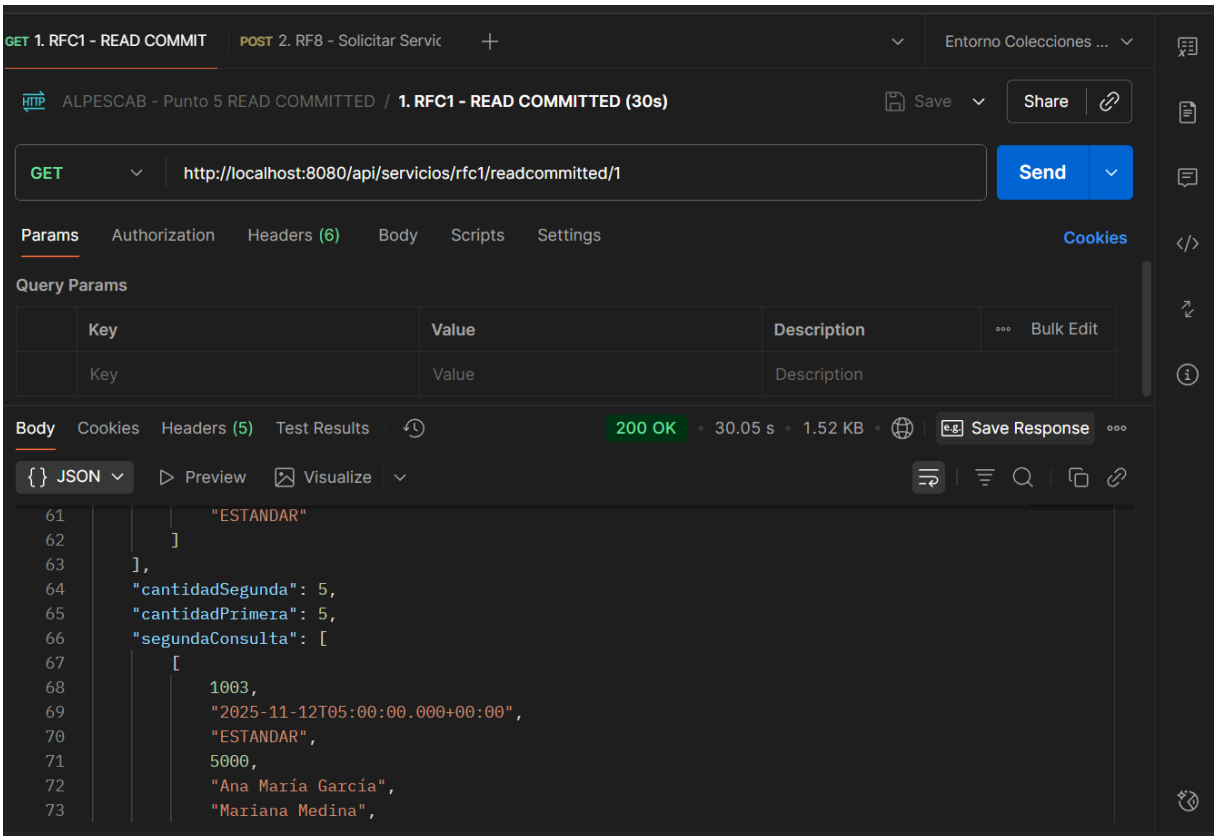
Conclusión:

El nivel de aislamiento SERIALIZABLE garantiza que una transacción vea una "fotografía" consistente de los datos desde el inicio hasta el fin de la transacción, aislando completamente los cambios concurrentes realizados por otras transacciones. Esto previene anomalías como lecturas no repetibles y lecturas fantasma.

PUNTO 5

pruebas





DESDE POSTMAN NO FUNCIONA PQ SON LA MISMA SESIÓN, TOCA HACERLO DESDE VS CODE COMO EN EL TALLER CON DOS CONEXIONES DIFERENTES

LÍNEA DEL TIEMPO - PUNTO 5 (READ COMMITTED)

RFC1 (Serializable)	Tiempo	RF8 (Read Committed)
Inicia transacción y realiza primera consulta	**t1**	
Espera (sleep de 30 segundos)	**t2**	
	t3	**Inicia transacción y solicita nuevo servicio**
	t4	**Registra el servicio y finaliza transacción**
Realiza segunda consulta y reporta resultados	**t5**	

DESCRIPCIÓN DE LO SUCEDIDO

¿RFC1 debió esperar a que terminara RF8?

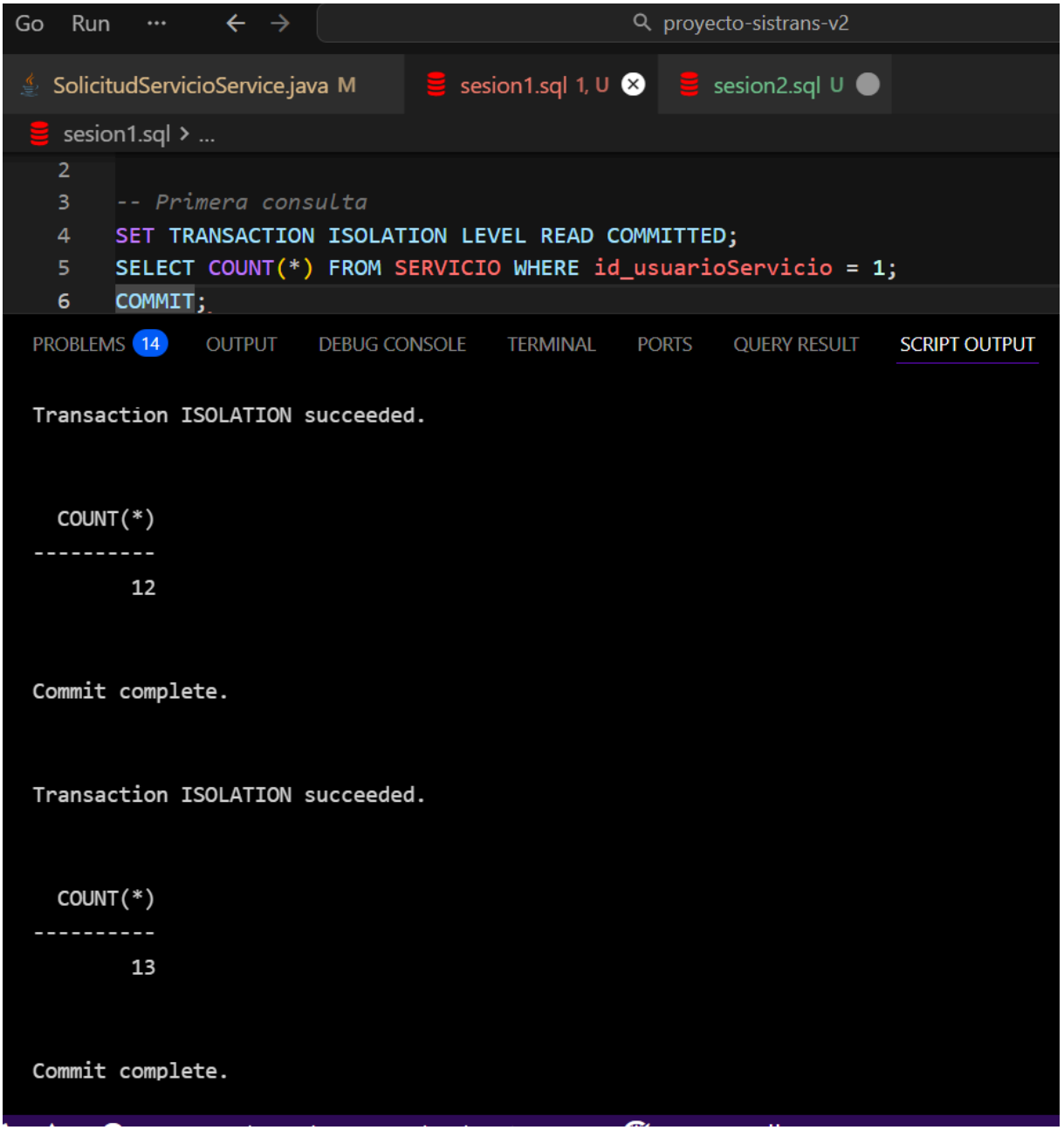
-NO. RFC1 NO tuvo que esperar a que terminara RF8. Ambas transacciones se ejecutaron de manera concurrente e independiente.

En el nivel de aislamiento READ COMMITTED:

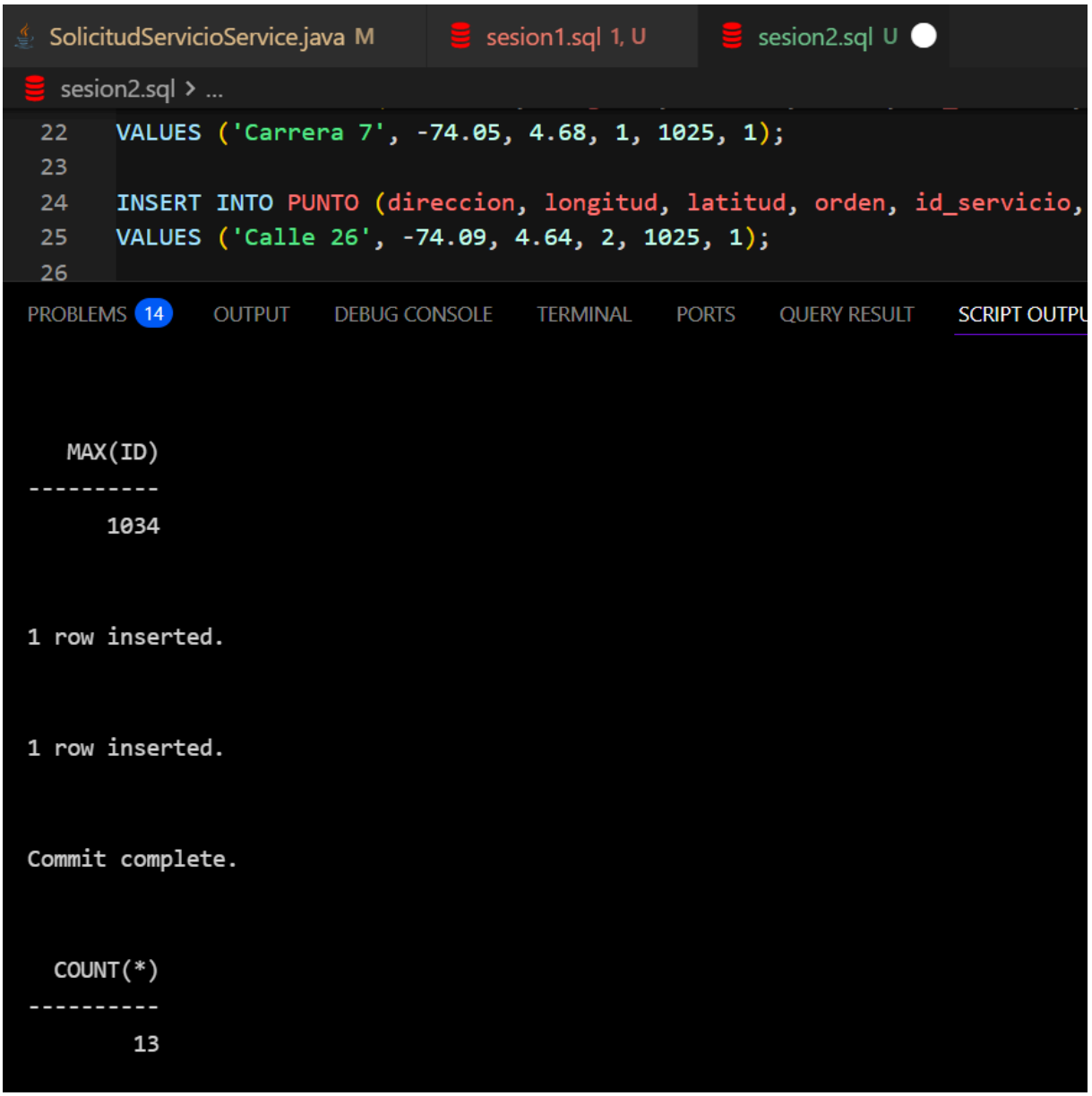
RFC1 y RF8 pueden ejecutarse al mismo tiempo sin bloquearse
Cada transacción lee los datos confirmados (COMMITTED) más recientes
No hay bloqueos entre lecturas (SELECT) e inserciones (INSERT)
RFC1 hizo su primera lectura, luego RF8 insertó y confirmó el servicio, y finalmente
RFC1 hizo su segunda lectura en una nueva transacción, por eso pudo ver el cambio.

RESULTADO PRESENTADO POR RFC1

RESULTADOS PRIMERA CONSULTA



RESULTADOS SEGUNDA CONSULTA



CONCLUSIÓN - DIFERENCIA CON PUNTO 4

Aspecto	SERIALIZABLE	READ COMMITTED
Primera lectura	5 servicios	5 servicios
RF8 ejecuta	Inserta servicio	Inserta servicio
Segunda lectura	5 servicios (NO ve RF8)	6 servicios (SÍ ve RF8)

Fenómeno	Previene lecturas no repetibles	Permite lecturas no repetibles

READ COMMITTED permite que se vean los cambios confirmados por otras transacciones concurrentes, causando lecturas no repetibles (non-repeatable reads). Esto demuestra menor aislamiento que SERIALIZABLE.