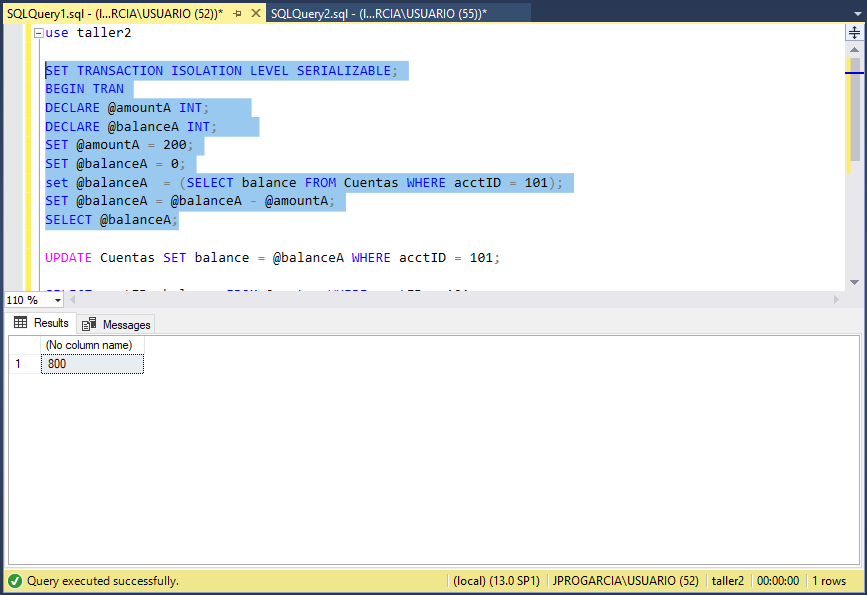
**Taller 2**

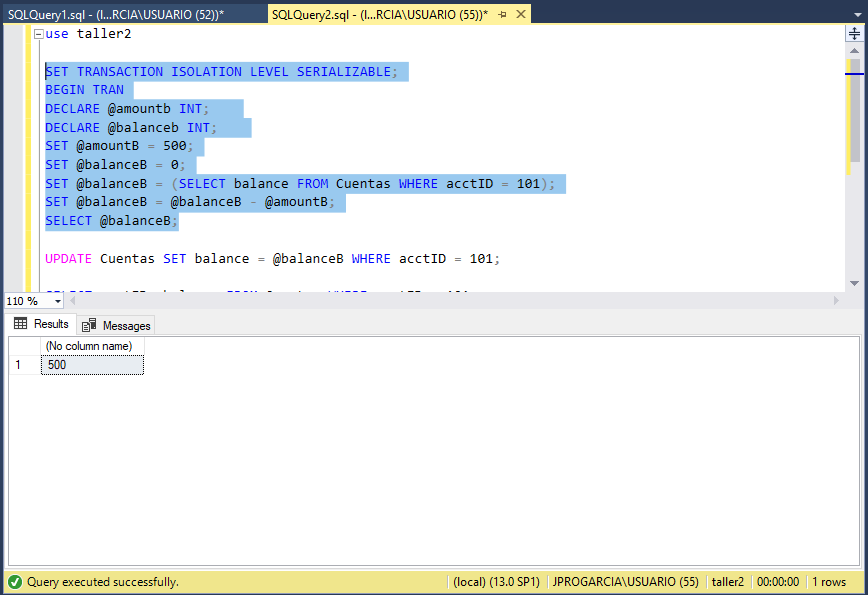
1. A
   1. **SERIALIZABLE**

Conexión 1



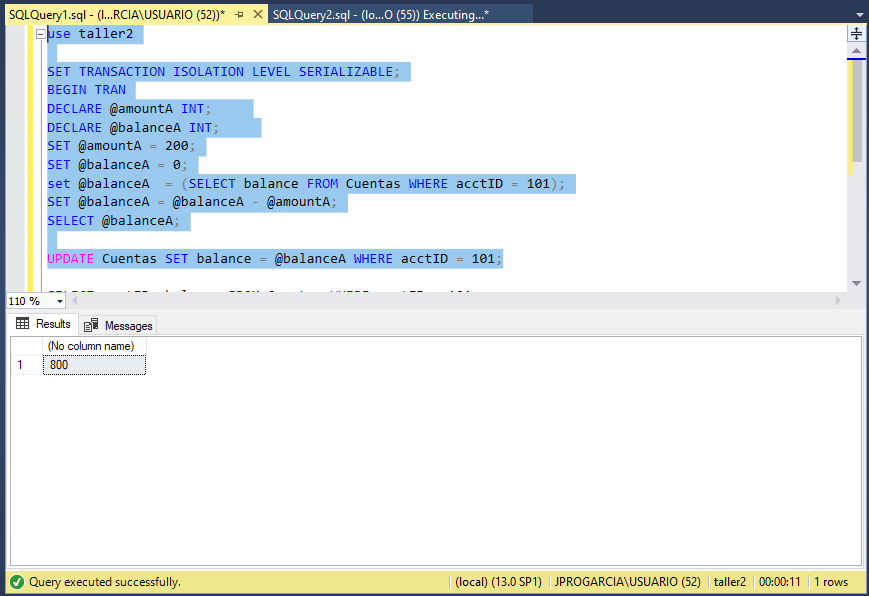
Este primer bloque que se encarga de iniciar la transaccion y hacer una operación con respecto al balance del id selecionado, como resultado nos muestra el valor

Conexión 2



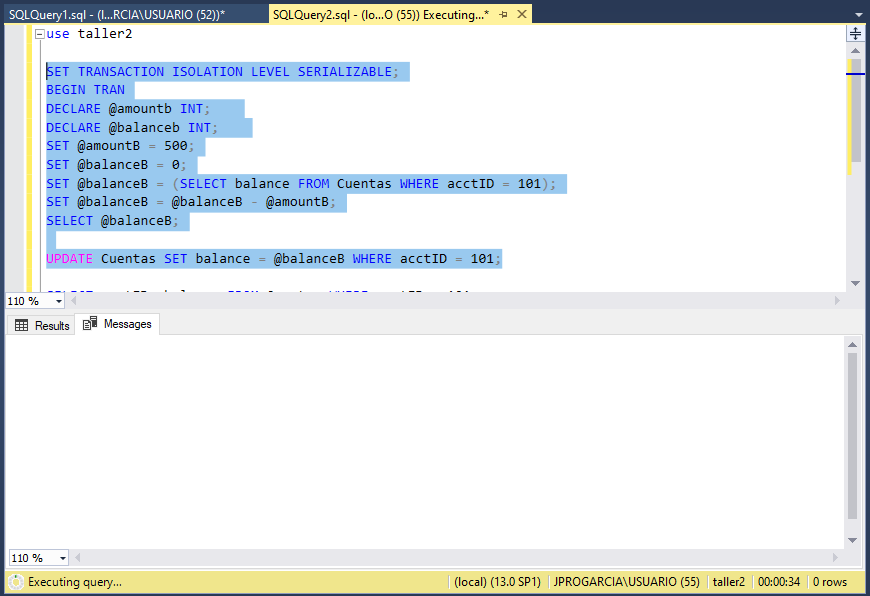
En este caso se encarga de hacer lo mismo que el anterior solo que se cambia la conexión.

Conexión 1



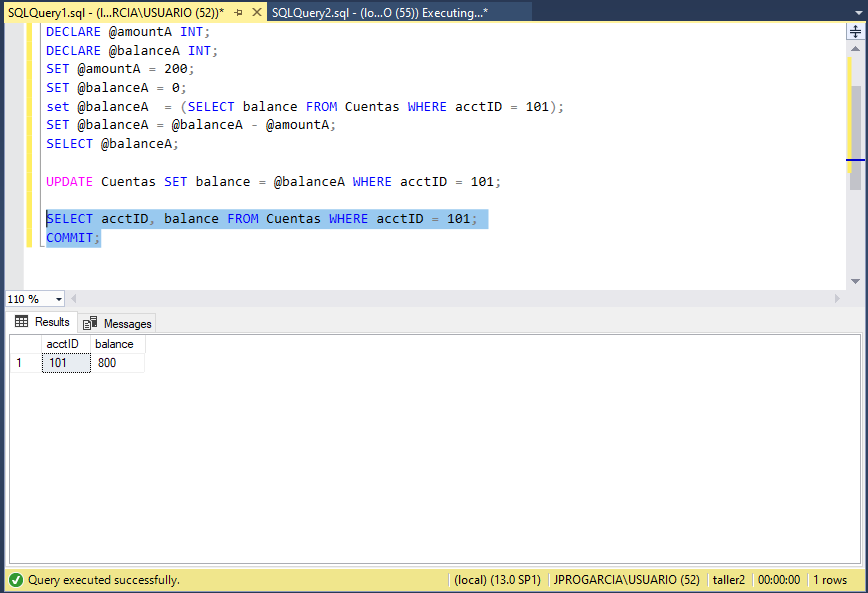
Al ejecutar el segundo bloque de la conexión 1 estamos actualizando el valor hayado anteriormente.

Conexión 2



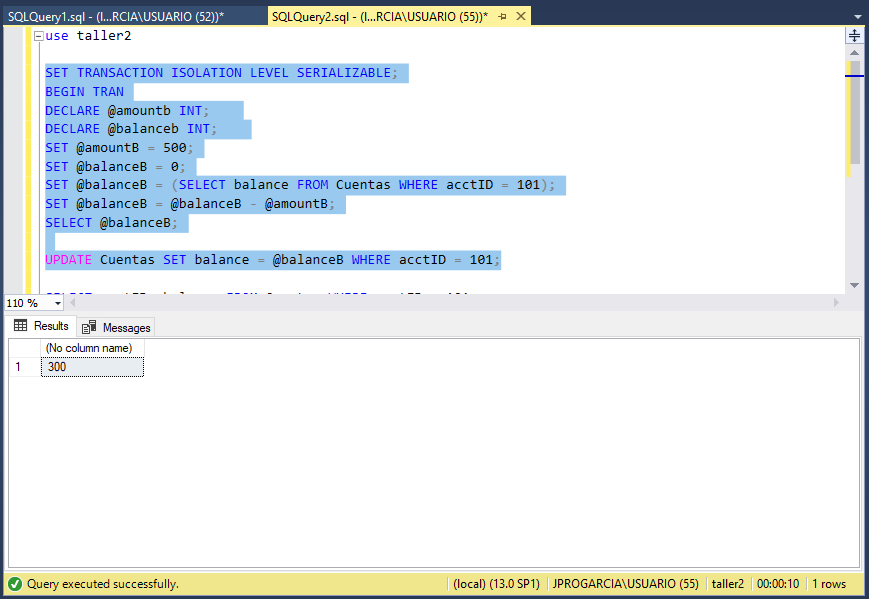
Lo Mismo que el anterio la diferencia es que al estar ejecutandose el cambio en la conexión 1 esta queda espereando el commit para poder ejecutarse.

Conexión 1

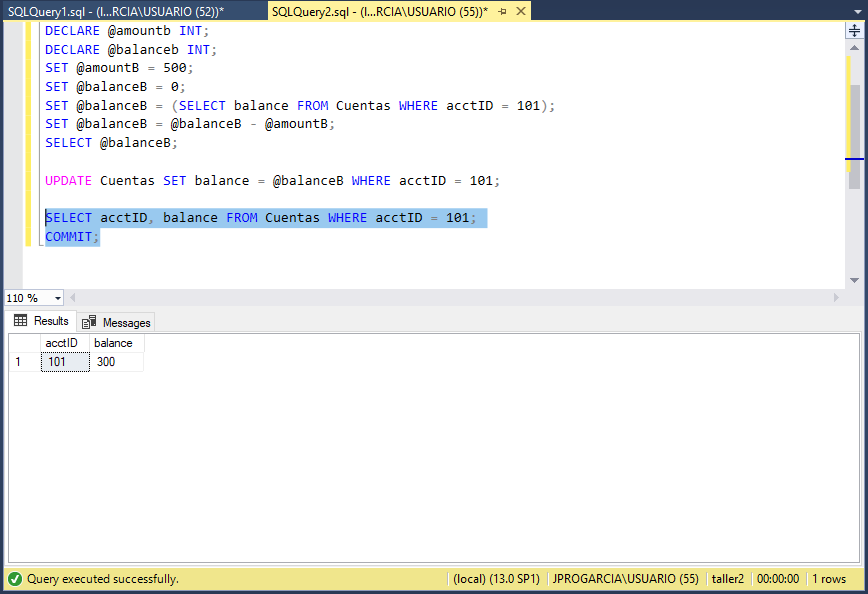


En el tercer bloque mostramos el dato que fue modificado y se le hace un commit

Conexión 2



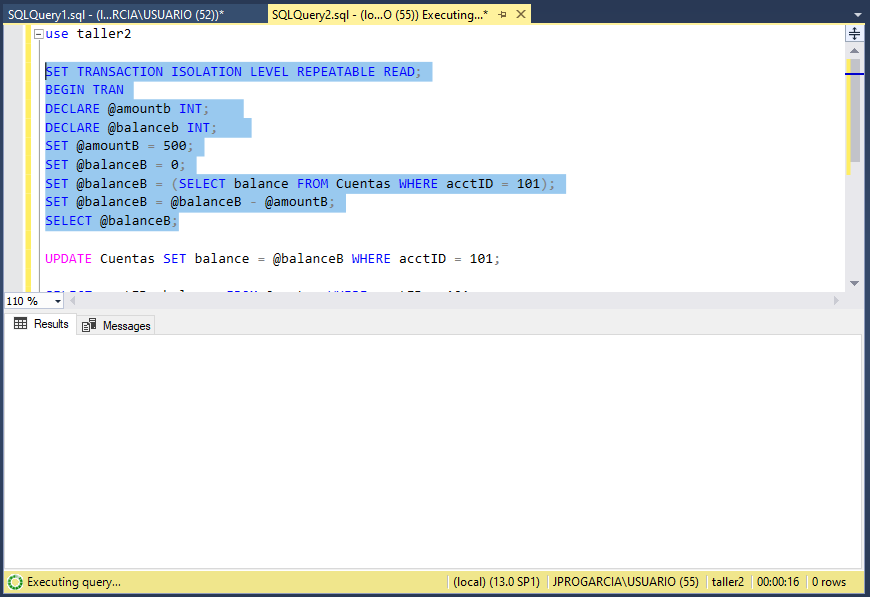
Al haber hecho commit en la primera conexión, la conexión dos pasa de estar en espera a ejecutar el codigo



En este ultimo bloque tambie muestra el dato que fue modificado y hace un commit

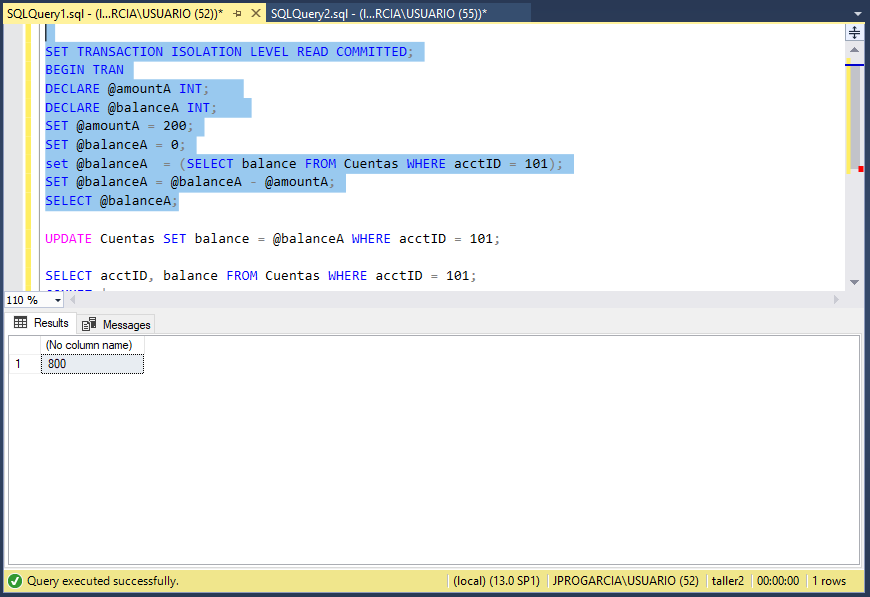
* 1. ¿Qué sucede si se reemplaza el nivel de aislamiento a 'REPEATABLE READ' en ambas conexiones?

**REPEATABLE READ**



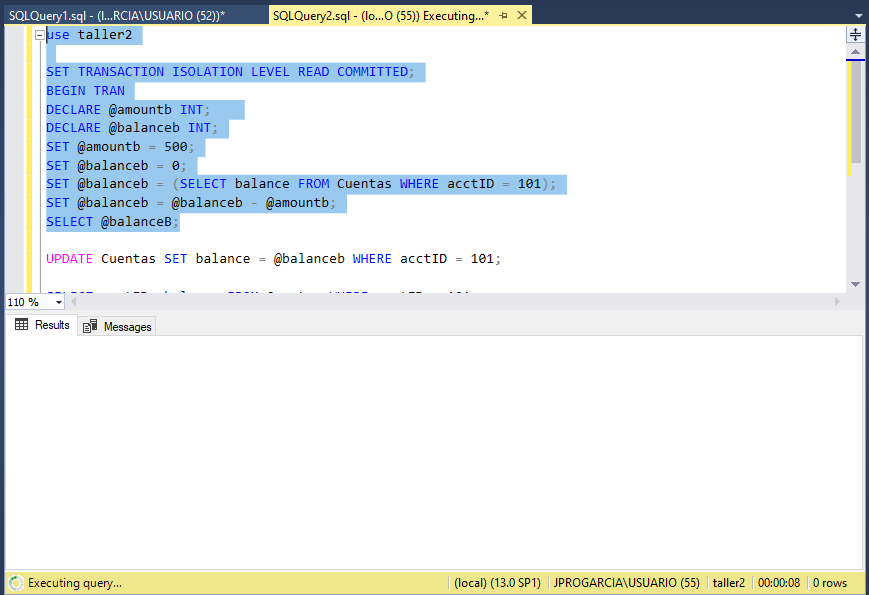
En este tipo transacción a diferencia de la anterior cuando se ejecuta el primer bloque de la conexión 2, este se queda esperando a que la conexión 1 termine como se muestra en la imagen.

Conexión 1



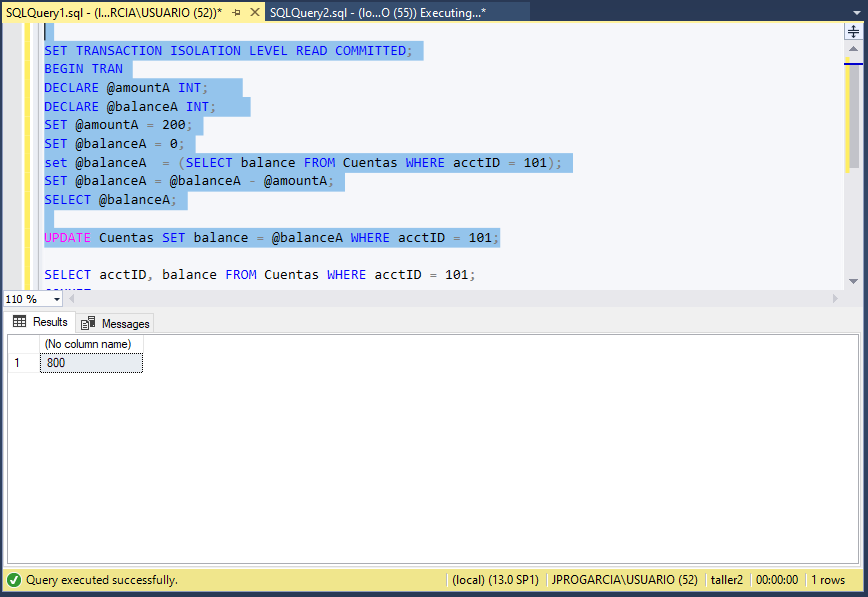
Este primer bloque que se encarga de iniciar la transacion y hacer una operación con respecto al balance del id selecionado, como resultado nos muestra el valor.

Conexión 2.



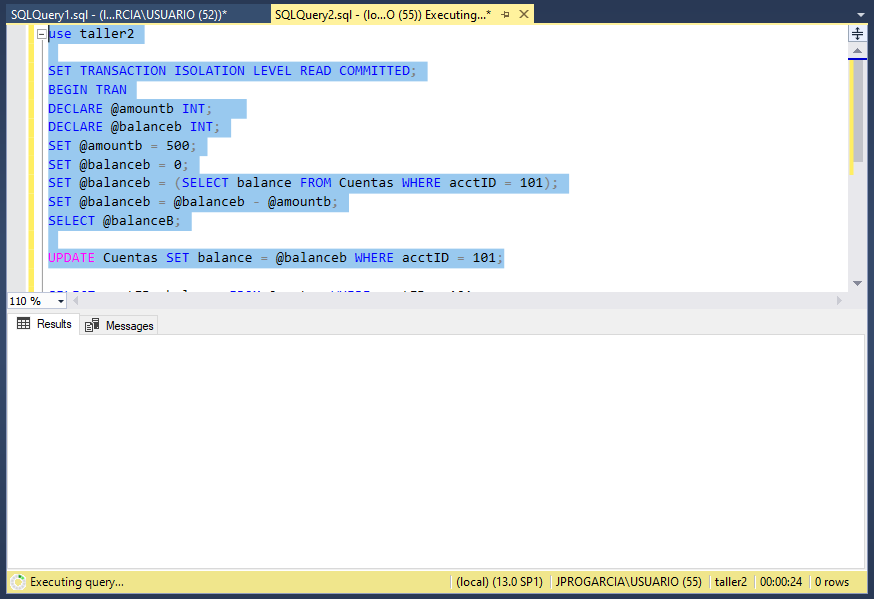
En este caso al ejecutar el bloque 1 de la conexión 2 inicia otra transacción y queda en espera a que finalice la conexión 1.

Conexión 1



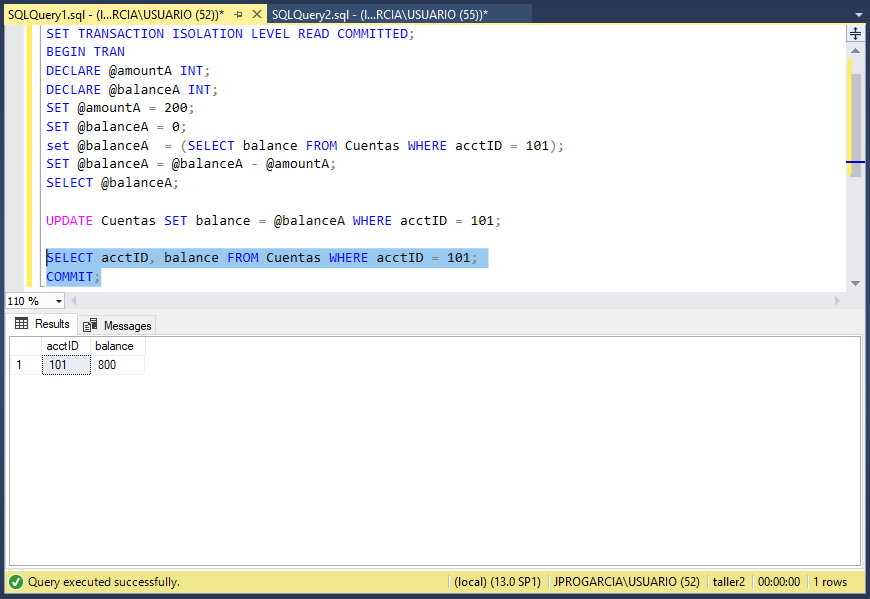
Al ejecutar el segundo bloque de la conexión 1 estamos actualizando el valor hayado anteriormente, en la base de datos

Conexión 2



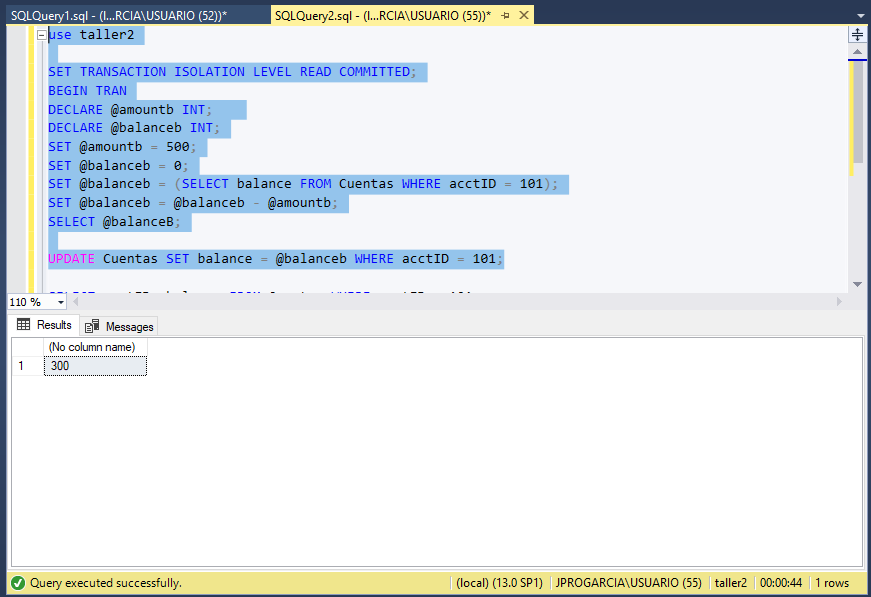
Al ejecutar el segundo bloque este sigue en espera para poder ejecutarse.

Conexión 1

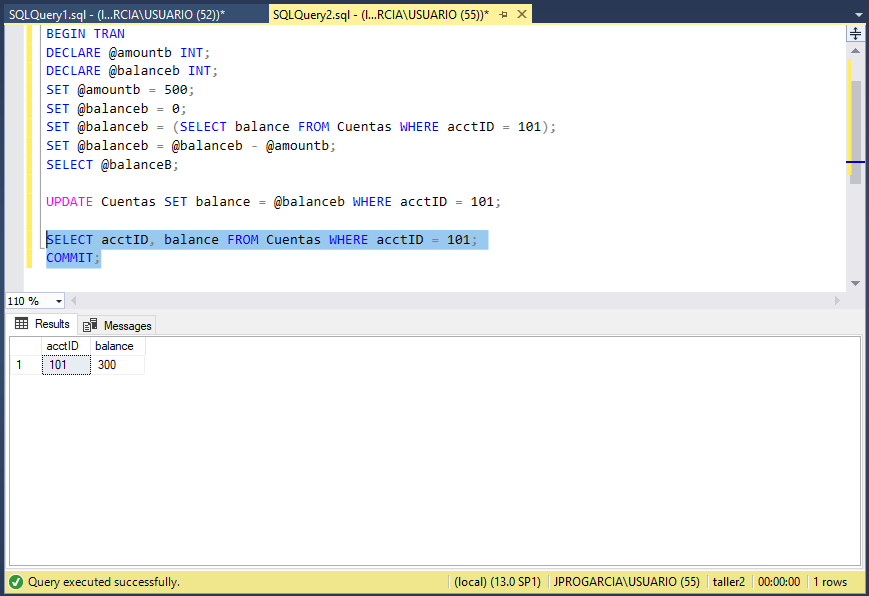


Al ejecutar el ultimo bloque de la conexión 1, vemos que nos muestra el valor que fue modificado con u respectivo id y después hace commit

Conexión 2



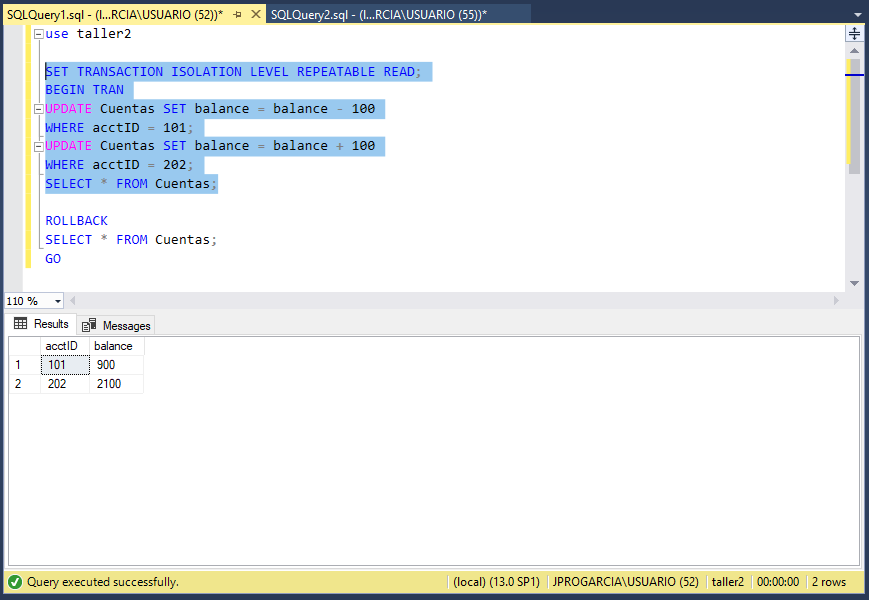
Al hacer commit la conexión 1, la conexión empezó a ejecutarse y nos muestra el valor y actualiza el dato elegido



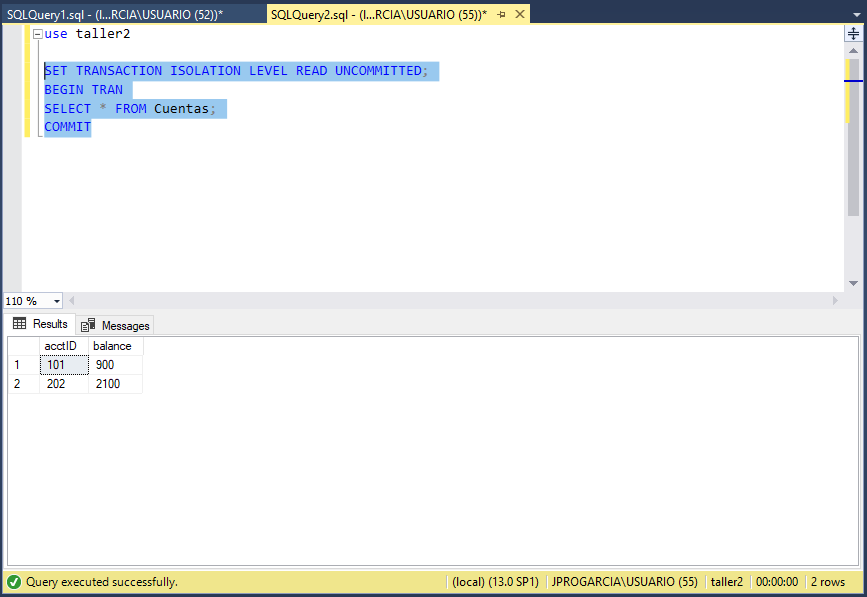
Y por último al ejecutar el ultimo bloque de la conexión 2, nos muestre el valor que fue modificado con su respectivo id

**¿Cómo se comporta el sistema?**

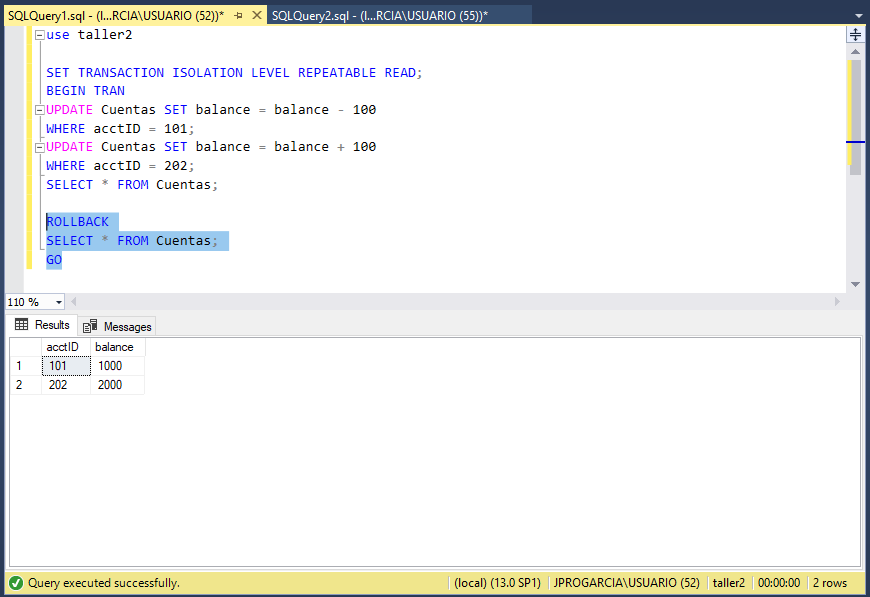
El sistema espera a que la conexión 1 termine su proceso y haga commit para después poder empezar a ejecutar la conexión 2, y dicha conexión toma en cuenta los cambios del commit realizado



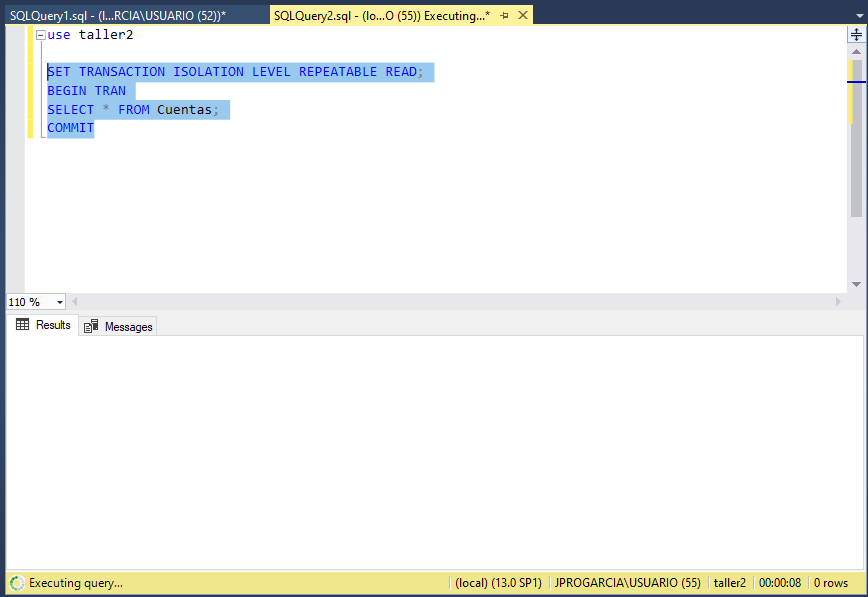
En este bloque de la conexión 1 se encarga de iniciar la transacion y modifica los valores del balance de cada registro, a uno le incrementa 100 y al otro decrementa 100

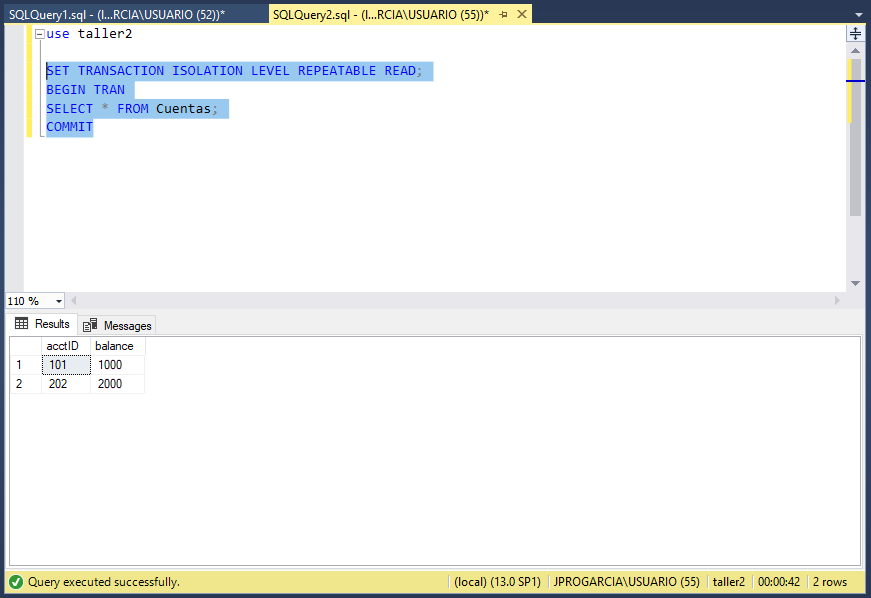


En este bloque empieza la transacion hace una consulta y por ultimo hace un commit



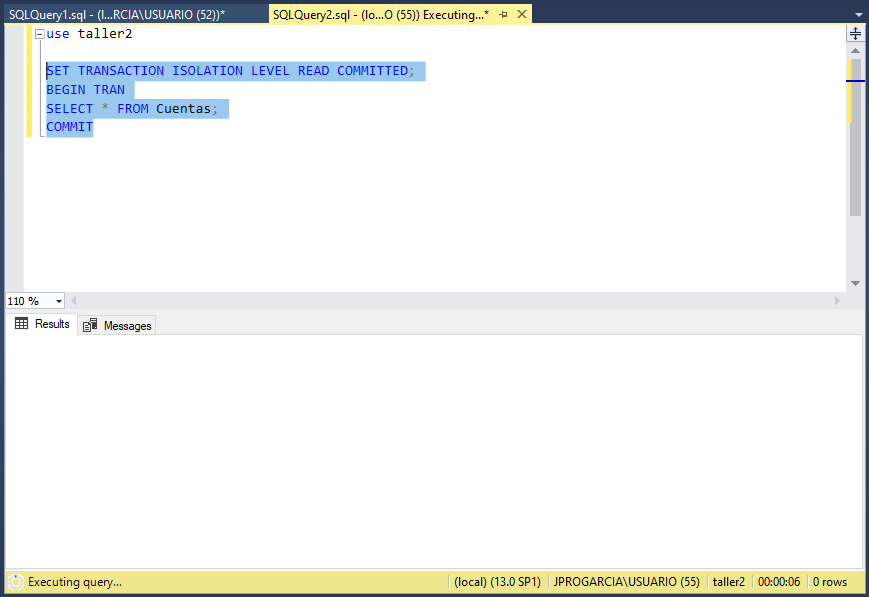
En este bloque se retrocede todo lo que se hizo en el primer bloque y hace una consulta donde muestra los valores iniciales.

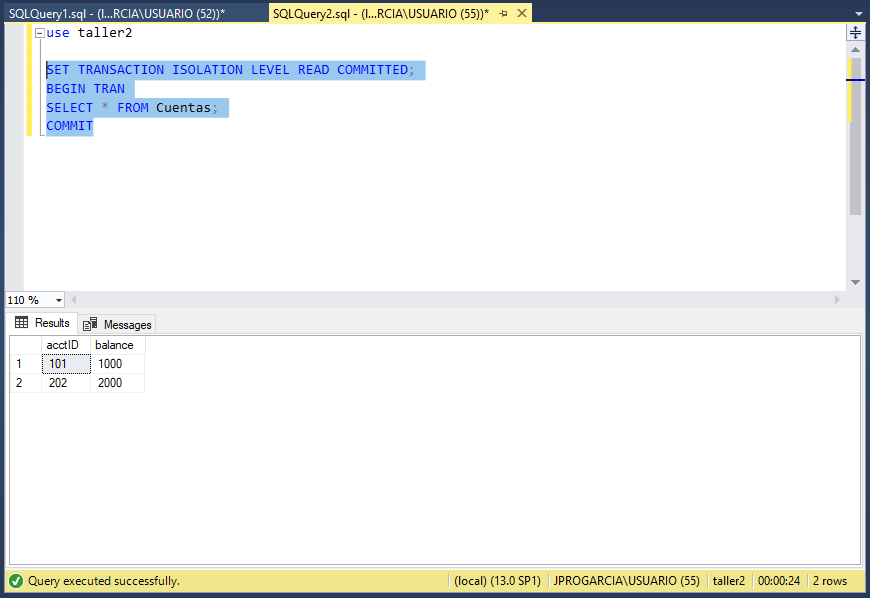
**¿Qué sucede si se reemplaza el nivel de aislamiento a 'REPEATABLE READ' en la conexión 2?** 



Al cambiar la transacción por **REPEATABLE READ** se ve que la conexión 2 entra en modo espera y no pasa a ejecutarse hasta que acaba la transacción de la conexión 1 y los valores quedan iguales que al iniciar.

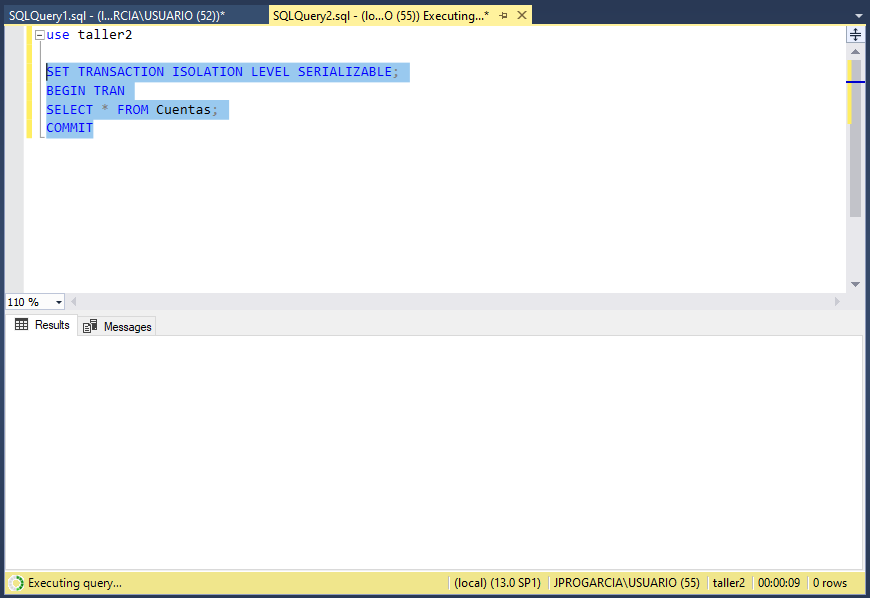
**¿Qué sucede si se reemplaza el nivel de aislamiento a 'READ COMMITTED' en la conexión 2?**

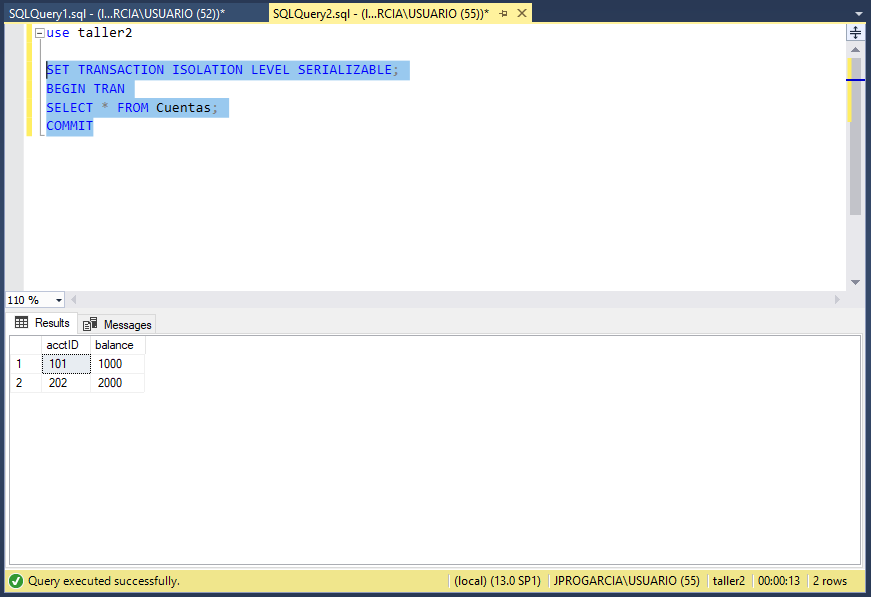




Al cambiar la transacción por **READ COMMITTED** se ve que la conexión 2 entra en modo espera y no pasa a ejecutarse hasta que acaba la transacción de la conexión 1 y los valores quedan iguales que al iniciar.

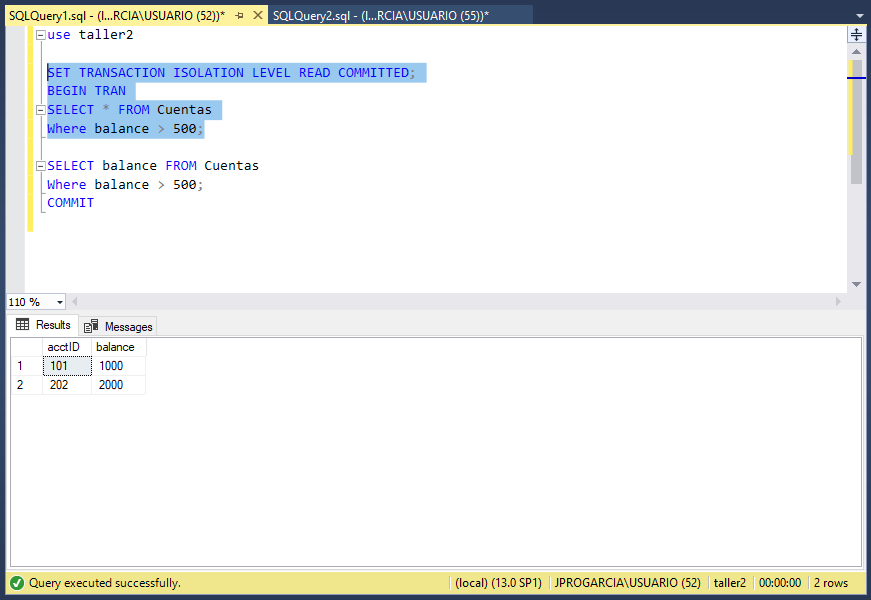
**¿Qué sucede si se reemplaza el nivel de aislamiento a 'SERIALIZABLE' en la conexion 2?**





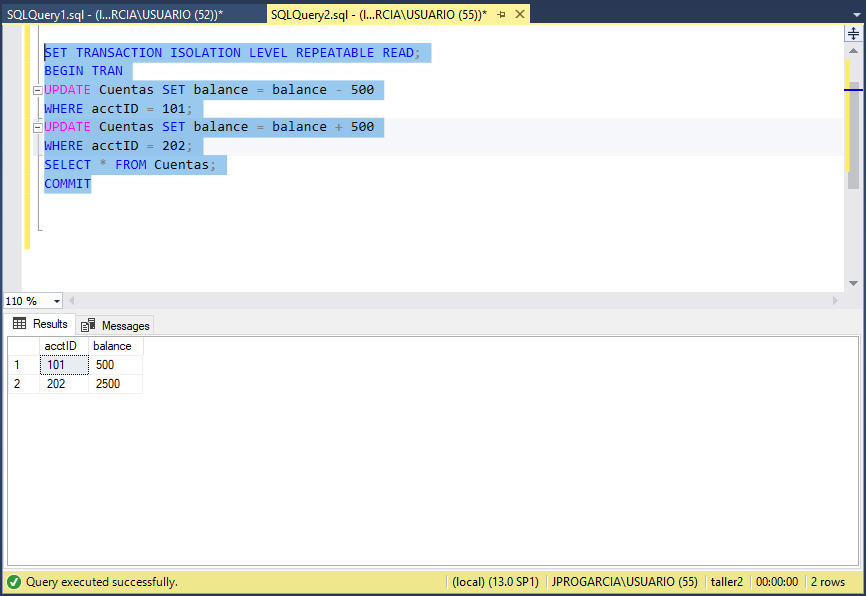
Del mismo modo que las anteriores al cambiar la transaccion por **SERIALIZABLE** la conexión 2 queda en espera y no pasa a ejecutarse hasta que termine la transacción de la conexión 1.

Conexión 1



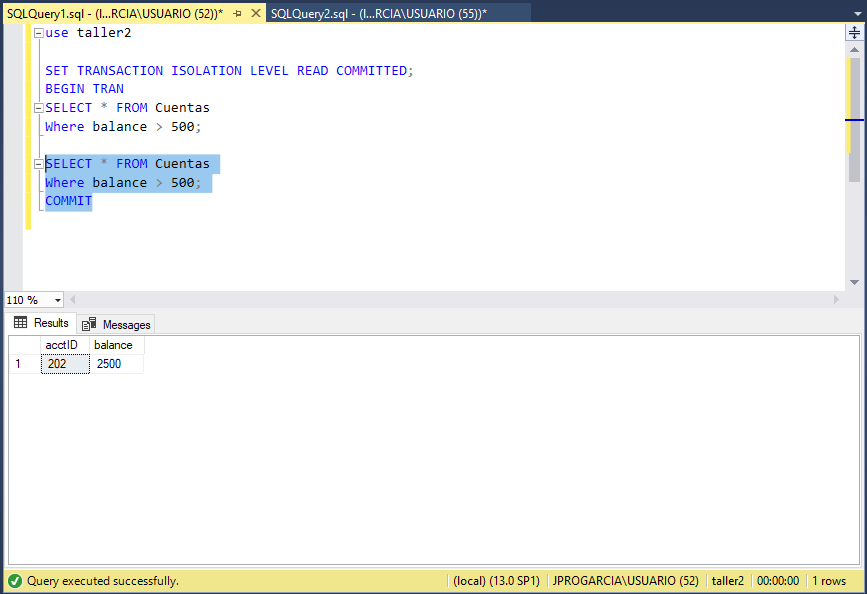
Al ejecutar el primer bloque de la conexión 1 vemos que inicia la transaccion y hace una busqueda de los balances mayores a 500.

Conexión 2



En este bloque se inicia la transaccion y modifica los dos datos de la tabla a uno le aumenta 500 y al otro le resta 500 y despues hace una consulta donde nos muestra los nuevos datos modificados y por ultimo hace un commit.

Conexión 1

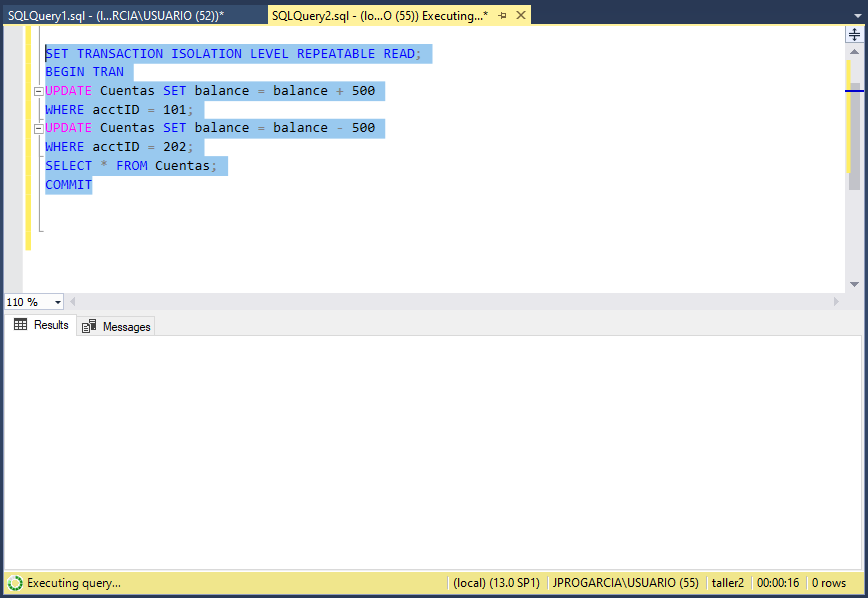


En este bloque volvemos a hacer la consulta donde el balance es mayor que 500.

**¿Cómo se comporta el sistema?**

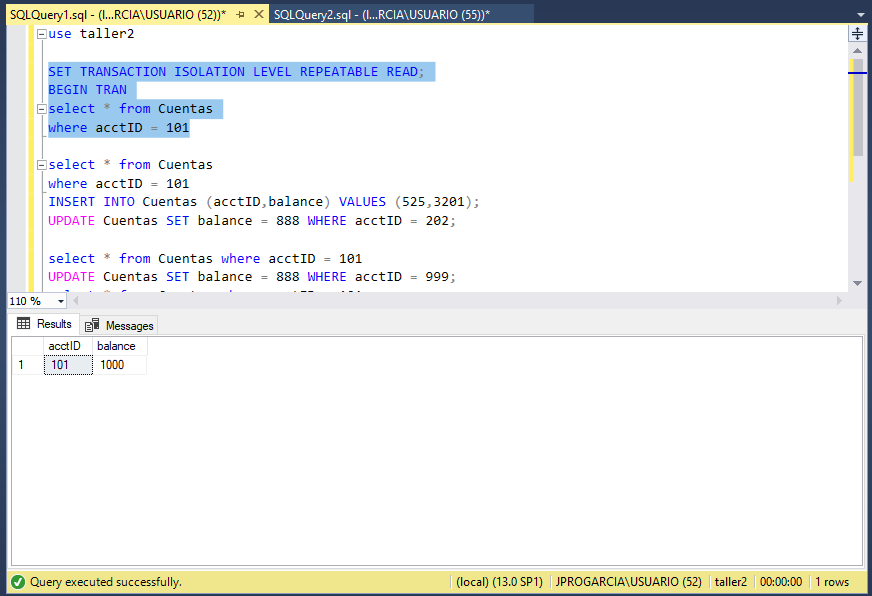
El sistema en este caso que usamos Read Committed de nivel de transacción en ambas Conexiones se puede ver que para poder actualizar un dato de la conexión 2 no es necesario esperar que se haga commit en la conexión 1.

**¿Qué sucede si se reemplaza el nivel de aislamiento a 'REPEATABLE READ' en la conexión 1?**



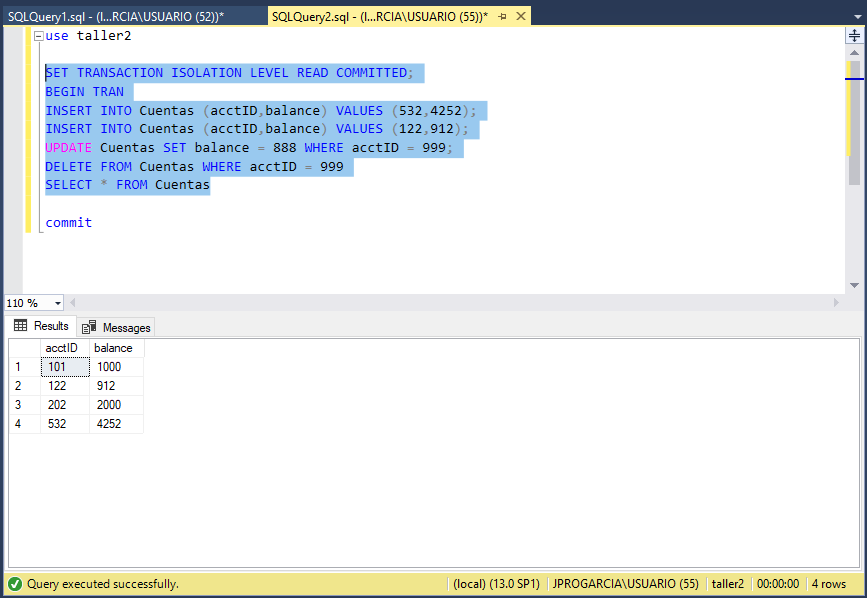
Al cambiar la transaccion a REPEATABLE READ podemos observar que la conexión 2 queda esperando a que se realice un commit en la conexión 1, y así poder ejecutarse.

Conexión 1



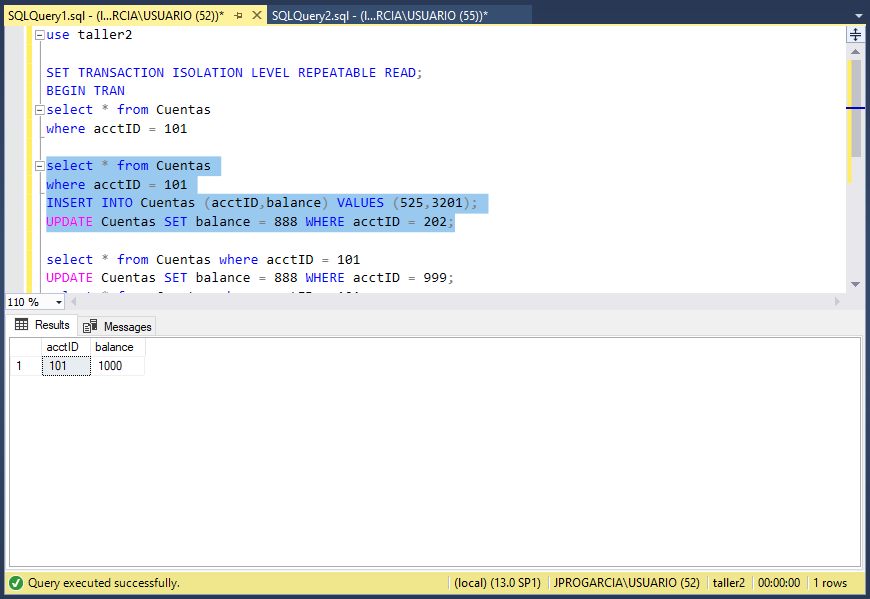
En este primer bloque ejecutamos la transacion y se muestra el balance del id selecionado

Conexión 2



En este bloque se inicia la transaccion y despues de esto se insertan dos datos aleatorios, se actualiza el balance del id 999, se elimina el dato del id 999 y por ultimo se muestra la tabla con los datos restantes.

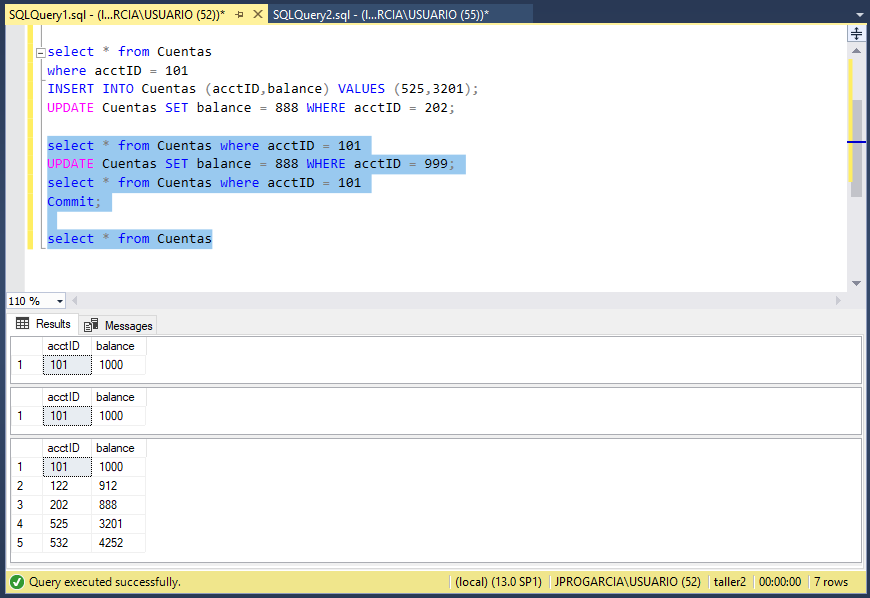
Conexión 1



En este bloque se muestra el dato seleccionado, se inserta un nuevo dato y actualiza el balance a 888 del id 202

Conexión 2: Se hace commit

Conexión 1:



En este bloque se muestra el id selecionado, se actualiza el balance en el id 999, vuelve a mostra el balance del id 101 y hcae comitt y por ultimo mostramos la tabla

**¿Cómo se comporta el sistema?**

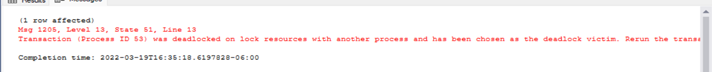
Podemos ver que al insertar y modificar datos en las dos conexiones al haber iniciado las transacciones no pone ningún tipo de problema esto debido a la transacción seleccionada

**Problemas de recurrencia Encontrados**

Algunos de los errores de recurrencia que encontramos es que, al ejecutar los distintos códigos, la consola de SQL mostraba el mismo tipo de error, el cual era que se debía definir una variable escalar.



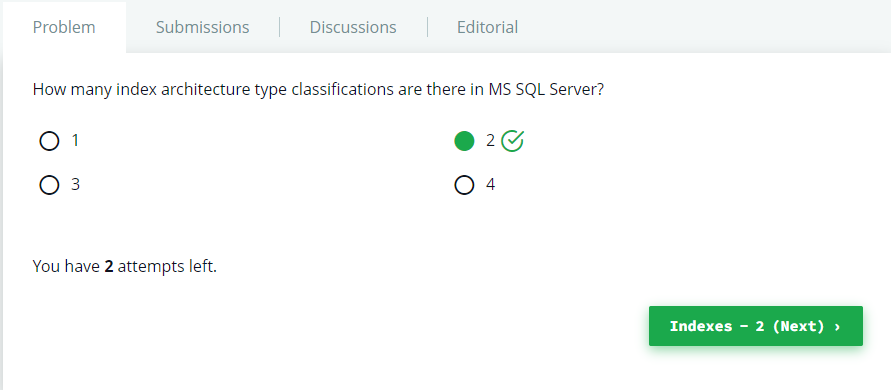
Este fue otro de los errores que tuvimos en ciertas ocasiones, este lo tuvimos en 1 de las 2 conexiones donde estábamos corriendo las pruebas.



Otro de los errores era que el código se quedaba ejecutando durante un tiempo sin dar una respuesta o mensaje de lo que ocurría.

Otro de los errores es que toca darle constantemente al comitt para que monte todo lo que hay actualizado.

1. Evidencia de Realizacion ejecicios HackerRank

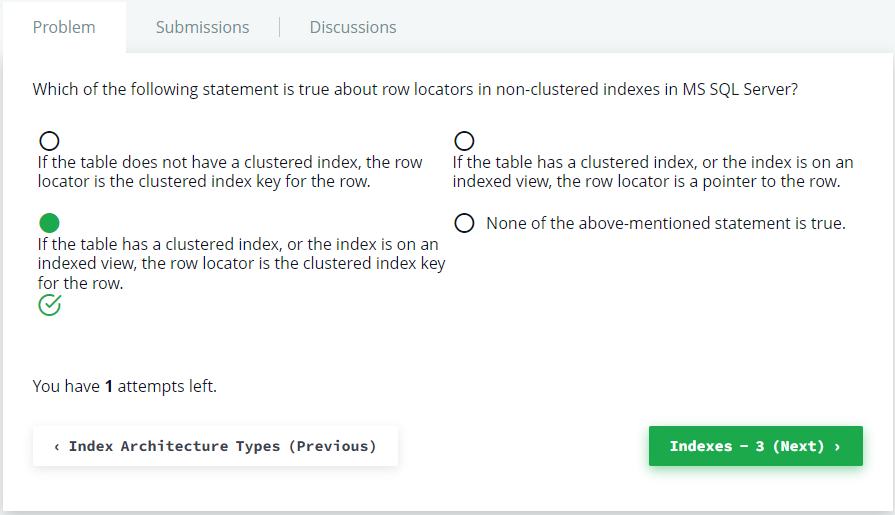


* 1. ¿Cuántas clasificaciones de tipo de arquitectura de índice hay en MS SQL Server?

serian dos tipos de clasificaciones agrupado y no agrupado que la diferencia de estas dos

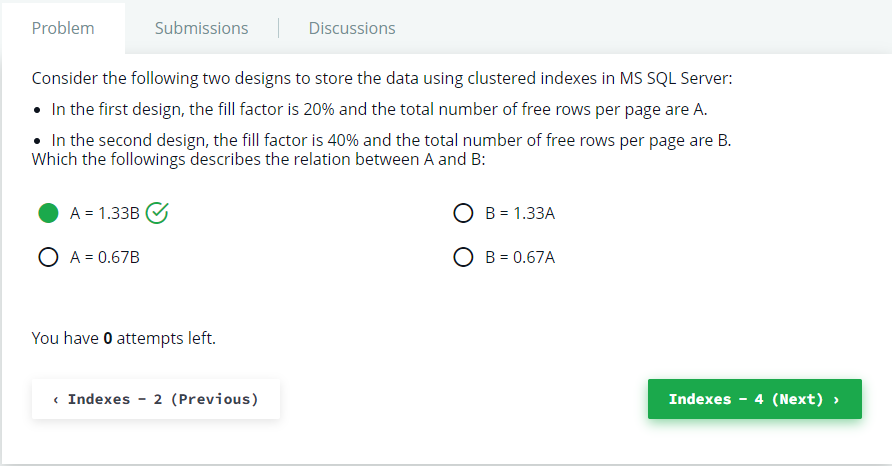
son que la clasificación no agrupada las filas de datos de la tabla subyacente no se clasifican ni almacenan en orden según sus claves no agrupadas.

Cabe recalcar que esta pregunta es un poco desactualizada ya que en MS SQL Server contamos ya con muchos más índices.



* 1. ¿Cuál de las siguientes afirmaciones es verdadera sobre los localizadores de filas en índices no agrupados en MS SQL Server?

Los localizadores de fila en filas de índice no agrupadas son un puntero a una fila o son una clave de índice agrupado para una fila.



* 1. Considere los siguientes dos diseños para almacenar los datos usando índices agrupados en MS SQL Server:

En el primer diseño, el factor de relleno es del 20 % y el número total de filas libres por página es A.

En el segundo diseño, el factor de relleno es del 40 % y el número total de filas libres por página es B.

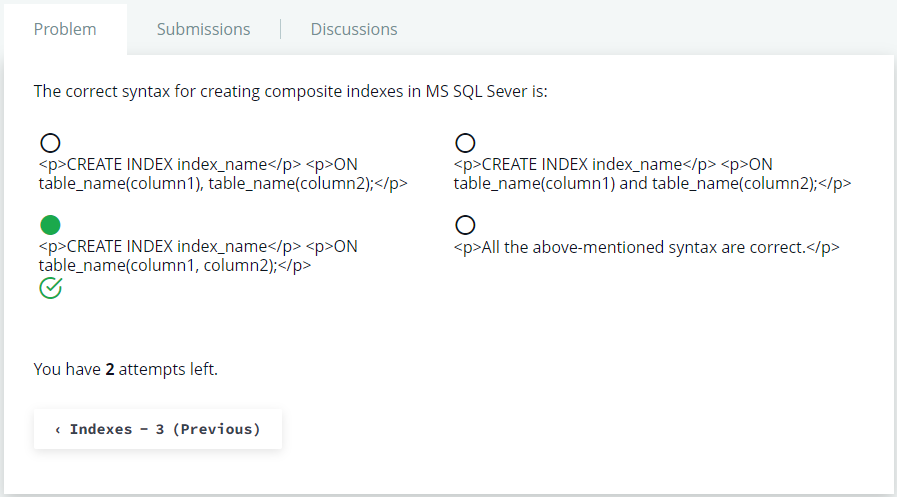
¿Que de lo siguiente describe la relación entre A y B?

La respuesta es A=1.33B.

En el primer diseño el espacio lleno es 20% y el espacio restante es 80%

Mientras en el segundo diseño el espacio lleno es 40% y el espacio restante es 60%

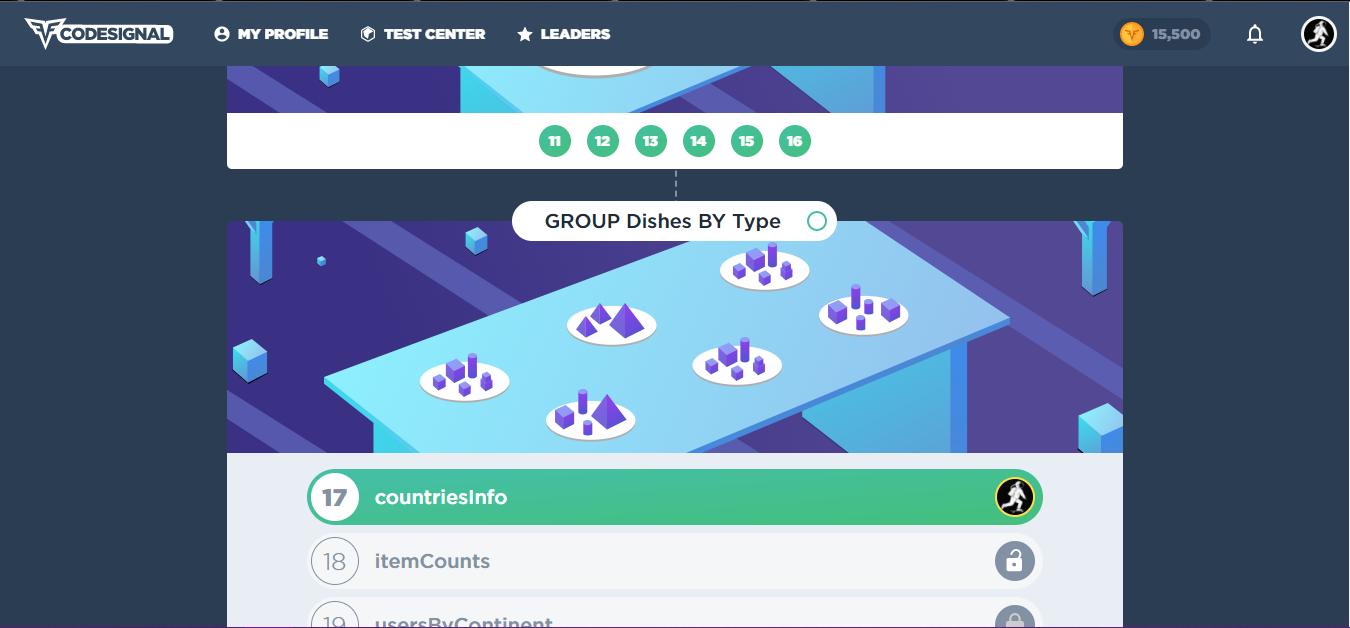
Entonces al replazar los valores del espacio restante en las opciones el unico que da una equivalencia es 80%=1.33(60%)



* 1. La sintaxis correcta para crear índices compuestos en MS SQL Server es:

La sintaxis es correcta ya que es lo que se a ensaeñeda en clase

1. Evidencia de Realizacion ejercicios CodeSignal



FELIZ DIA