**NIVELES DE AISLAMIENTOS EN SQL SERVER**

El comportamiento de SQL Server en operaciones de lectura o de escritura, se detallan los diferentes niveles de aislamiento basados en bloqueos (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE) y los niveles de aislamiento basados en versionado de filas (SNAPSHOT), se explican los males de la concurrencia (lecturas sucias, lecturas no repetibles, lecturas fantasma, y conflictos de actualización), como establecer el nivel de aislamiento deseado (SET TRANSACTION ISOLATION LEVEL y las opciones de base de datos READ\_COMMITTED\_SNAPSHOT y ALLOW\_SNAPSHOT\_ISOLATION), cómo conocer el tiempo máximo de bloqueo (@@LOCK\_TIMEOUT) y como establecer el tiempo máximo de bloqueo (SET LOCK\_TIMEOUT), etc.

El nivel de aislamiento de una transacción (transaction isolation level) **define el grado en que se aísla una transacción de las modificaciones de recursos o datos realizadas por otras transacciones**. Esto dicho así, quizás no permita entender fácilmente su significado (excepto para DBAs o programadores muy avanzados). El nivel de aislamiento de una transacción es una característica **de vital importancia** en el desarrollo de aplicaciones de base de datos, ya que **afecta a los tipos y duración de bloqueos** que se producen en nuestra infraestructura de base de datos, lo cual, tiene efectos directos en el rendimiento y tiempo de respuesta de nuestras consultas y transacciones. Además, el nivel de aislamiento nos permite indicar cómo deseamos que se comporte nuestro motor de base de datos frente a situaciones como las lecturas sucias (dirty reads), las lecturas no repetibles o las lecturas fantasma, comportamientos que pueden impactar consistentemente en la eficacia de nuestros desarrollos en entornos de base de datos con concurrencia. Evidentemente, la elección del modo de aislamiento es más importante cuanto mayor es la concurrencia de la base de datos (sin concurrencia, ¿a quién le interesan los modos de aislamiento?).

**READ UNCOMMITTED**

**READ UNCOMMITTED** puede recuperar datos modificados pero no confirmados por otras transacciones (**lecturas sucias - dirty reads**). En este nivel se pueden producir todos los efectos secundarios de simultaneidad (lecturas sucias, lecturas no repetibles y lecturas fantasmas - ej: entre dos lecturas de un mismo registro en una transacción A, otra transacción B puede modificar dicho registro), pero no hay bloqueos ni versiones de lectura, por lo que se minimiza la sobrecarga. **Una operación de lectura (SELECT) no establecerá bloqueos compartidos (shared locks)** sobre los datos que está leyendo, por lo que no será bloqueada por otra transacción que tenga establecido un bloqueo exclusivo por motivo de una operación de escritura. Este nivel de aislamiento ofrece grandes beneficios de rendimiento, pero sólo deberemos utilizarlo en aquellos casos en que la ocurrencia de lecturas sucias (dirty reads) no sea un problema.

**READ COMMITTED**

**READ COMMITTED** permite que entre dos lecturas de un mismo registro en una transacción A, otra transacción B pueda modificar dicho registro, obteniéndose diferentes resultados de la misma lectura. Evita las lecturas sucias (dirty reads), pero por el contrario, permite lecturas no repetibles. Es la **opción por defecto en SQL Server 2000 y SQL Server 2005**. Con este nivel de aislamiento, **una operación de lectura (SELECT) establecerá bloqueos compartidos (shared locks) sobre los datos que está leyendo**. Sin embargo, dichos bloqueos compartidos finalizarán junto con la propia operación de lectura, de tal modo que entre dos lecturas cabe la posibilidad de que otra transacción realice una operación de escritura (ej: UPDATE), en cuyo caso, la segunda lectura obtendrá datos distintos a la primera lectura (lecturas no repetibles).

**REPEATABLE READ**

**REPEATABLE READ** evita que entre dos lecturas de un mismo registro en una transacción A, otra transacción B pueda modificar dicho registro, con el efecto de que en la segunda lectura de la transacción A se obtuviera un dato diferente. De este modo, ambas lecturas serían iguales (lecturas repetidas). Para ello, **una operación de lectura (SELECT) establecerá bloqueos compartidos (shared locks) sobre los datos que está leyendo, y los mantendrá hasta el final de la transacción**, garantizando así que no se produce **lecturas no repetibles (non repeatable reads)**. Mayor consistencia en la transacción, mediante mayores recursos y bloqueos (**se evitan los problemas de las lecturas sucias y de las lecturas no repetibles**, pagando el precio de necesidad de mayores recursos). Sin embargo, este modo de aislamiento **no evita las lecturas fantasma**, es decir, una transacción podría ejecutar una consulta sobre un rango de filas (ej: 100 filas) y de forma simultánea otra transacción podría realizar un inserción de una o varias filas sobre el mismo rango.

**SNAPSHOT**

**SNAPSHOT**. Evita los problemas de las lecturas sucias (dirty reads), de las lecturas no repetibles (non repeatable reads), y de las lecturas fantasma (phantom reads). Sin embargo, en vez de realizar un uso intensivo de bloqueos, su funcionamiento se basa en el almacenamiento de versiones de filas en TEMPDB. Si una transacción empieza en un momento del tiempo determinado, sólo será capaz de acceder a la información de la base de datos que era real en aquel momento del tiempo. Por ello, es posible que existan múltiples versiones de la misma fila en TEMPDB, con el fin de servir a múltiples transacciones. Debido al impacto que tiene este modo de aislamiento sobre la base de datos del sistema TEMPDB, sólo es recomendable en bases de datos intensivas en lecturas o en bases de datos con grandes problemas de bloqueos. Una situación de concurrencia particular de éste modo de aislamiento es el caso de los **conflictos de actualización (update conflicts)**: Si una fila es leída en una transacción, y posteriormente como parte de la misma transacción es modificada, PERO entre la lectura inicial y la posterior modificación ha sido alterada por otra transacción, se produce un conflicto de actualización (update conflict), que implica un mensaje de error 3960 (Msg 3960, Level 16, State 4 Line 1. Snapshot isolation transaction aborted due to update conflict. You cannot use snapshot isolation to access table 'myTable' directly or indirectly in database 'myDB' to update, delete or insert the row that has been modified or deleted by another transaction. Retry the transaction or change de isolation level for the update/delete statement), y además también implica que la transacción se deshace (Rollback).

**SERIALIZABLE**

**SERIALIZABLE** garantiza que una transacción recuperará exactamente los mismos datos cada vez que repita una operación de lectura (es decir, la misma sentencia SELECT con la misma cláusula WHERE devolverá el mismo número de filas, luego no se podrán insertar filas nuevas en el rango cubierto por la WHERE, etc. - se evitarán las **lecturas fantasma**), aunque para ello aplicará un nivel de bloqueo que puede afectar a los demás usuarios en los sistemas multiusuario (realizará un bloqueo de un rango de índice - conforme a la cláusula WHERE - y si no es posible bloqueará toda la tabla). Evita los problemas de las lecturas sucias (dirty reads), de las lecturas no repetibles (non repeatable reads), y de las lecturas fantasma (phantom reads).

