Lock Modes

<http://msdn.microsoft.com/en-us/library/ms175519(v=sql.105).aspx>

**SQL Server 2008 R2**

[Other Versions](javascript:;)

11 out of 16 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/ms175519(v=sql.105).aspx#feedback)

The Microsoft SQL Server Database Engine locks resources using different lock modes that determine how the resources can be accessed by concurrent transactions.

**Applies to**: SQL Server 2008 R2 and higher versions.

The following table shows the resource lock modes that the Database Engine uses.

|  |  |
| --- | --- |
| **Lock mode** | **Description** |
| Shared (S) | Used for read operations that do not change or update data, such as a SELECT statement. |
| Update (U) | Used on resources that can be updated. Prevents a common form of deadlock that occurs when multiple sessions are reading, locking, and potentially updating resources later. |
| Exclusive (X) | Used for data-modification operations, such as INSERT, UPDATE, or DELETE. Ensures that multiple updates cannot be made to the same resource at the same time. |
| Intent | Used to establish a lock hierarchy. The types of intent locks are: intent shared (IS), intent exclusive (IX), and shared with intent exclusive (SIX). |
| Schema | Used when an operation dependent on the schema of a table is executing. The types of schema locks are: schema modification (Sch-M) and schema stability (Sch-S). |
| Bulk Update (BU) | Used when bulk copying data into a table and the **TABLOCK** hint is specified. |
| Key-range | Protects the range of rows read by a query when using the serializable transaction isolation level. Ensures that other transactions cannot insert rows that would qualify for the queries of the serializable transaction if the queries were run again. |

[Shared Locks](javascript:void(0))

Shared (S) locks allow concurrent transactions to read (SELECT) a resource under pessimistic concurrency control. For more information, see [Types of Concurrency Control](http://msdn.microsoft.com/en-us/library/ms189132(v=sql.105).aspx). No other transactions can modify the data while shared (S) locks exist on the resource. Shared (S) locks on a resource are released as soon as the read operation completes, unless the transaction isolation level is set to repeatable read or higher, or a locking hint is used to retain the shared (S) locks for the duration of the transaction.

[Update Locks](javascript:void(0))

Update (U) locks prevent a common form of deadlock. In a repeatable read or serializable transaction, the transaction reads data, acquiring a shared (S) lock on the resource (page or row), and then modifies the data, which requires lock conversion to an exclusive (X) lock. If two transactions acquire shared-mode locks on a resource and then attempt to update data concurrently, one transaction attempts the lock conversion to an exclusive (X) lock. The shared-mode-to-exclusive lock conversion must wait because the exclusive lock for one transaction is not compatible with the shared-mode lock of the other transaction; a lock wait occurs. The second transaction attempts to acquire an exclusive (X) lock for its update. Because both transactions are converting to exclusive (X) locks, and they are each waiting for the other transaction to release its shared-mode lock, a deadlock occurs.

To avoid this potential deadlock problem, update (U) locks are used. Only one transaction can obtain an update (U) lock to a resource at a time. If a transaction modifies a resource, the update (U) lock is converted to an exclusive (X) lock.

[Exclusive Locks](javascript:void(0))

Exclusive (X) locks prevent access to a resource by concurrent transactions. With an exclusive (X) lock, no other transactions can modify data; read operations can take place only with the use of the NOLOCK hint or read uncommitted isolation level.

Data modification statements, such as INSERT, UPDATE, and DELETE combine both modification and read operations. The statement first performs read operations to acquire data before performing the required modification operations. Data modification statements, therefore, typically request both shared locks and exclusive locks. For example, an UPDATE statement might modify rows in one table based on a join with another table. In this case, the UPDATE statement requests shared locks on the rows read in the join table in addition to requesting exclusive locks on the updated rows.

[Intent Locks](javascript:void(0))

The Database Engine uses intent locks to protect placing a shared (S) lock or exclusive (X) lock on a resource lower in the lock hierarchy. Intent locks are named intent locks because they are acquired before a lock at the lower level, and therefore signal intent to place locks at a lower level.

Intent locks serve two purposes:

* To prevent other transactions from modifying the higher-level resource in a way that would invalidate the lock at the lower level.
* To improve the efficiency of the Database Engine in detecting lock conflicts at the higher level of granularity.

For example, a shared intent lock is requested at the table level before shared (S) locks are requested on pages or rows within that table. Setting an intent lock at the table level prevents another transaction from subsequently acquiring an exclusive (X) lock on the table containing that page. Intent locks improve performance because the Database Engine examines intent locks only at the table level to determine if a transaction can safely acquire a lock on that table. This removes the requirement to examine every row or page lock on the table to determine if a transaction can lock the entire table.

Intent locks include intent shared (IS), intent exclusive (IX), and shared with intent exclusive (SIX).

|  |  |
| --- | --- |
| **Lock mode** | **Description** |
| Intent shared (IS) | Protects requested or acquired shared locks on some (but not all) resources lower in the hierarchy. |
| Intent exclusive (IX) | Protects requested or acquired exclusive locks on some (but not all) resources lower in the hierarchy. IX is a superset of IS, and it also protects requesting shared locks on lower level resources. |
| Shared with intent exclusive (SIX) | Protects requested or acquired shared locks on all resources lower in the hierarchy and intent exclusive locks on some (but not all) of the lower level resources. Concurrent IS locks at the top-level resource are allowed. For example, acquiring a SIX lock on a table also acquires intent exclusive locks on the pages being modified and exclusive locks on the modified rows. There can be only one SIX lock per resource at one time, preventing updates to the resource made by other transactions, although other transactions can read resources lower in the hierarchy by obtaining IS locks at the table level. |
| Intent update (IU) | Protects requested or acquired update locks on all resources lower in the hierachy. IU locks are used only on page resources. IU locks are converted to IX locks if an update operation takes place. |
| Shared intent update (SIU) | A combination of S and IU locks, as a result of acquiring these locks separately and simultaneously holding both locks. For example, a transaction executes a query with the PAGLOCK hint and then executes an update operation. The query with the PAGLOCK hint acquires the S lock, and the update operation acquires the IU lock. |
| Update intent exclusive (UIX) | A combination of U and IX locks, as a result of acquiring these locks separately and simultaneously holding both locks. |

[Schema Locks](javascript:void(0))

The Database Engine uses schema modification (Sch-M) locks during a table data definition language (DDL) operation, such as adding a column or dropping a table. During the time that it is held, the Sch-M lock prevents concurrent access to the table. This means the Sch-M lock blocks all outside operations until the lock is released.

Some data manipulation language (DML) operations, such as table truncation, use Sch-M locks to prevent access to affected tables by concurrent operations.

The Database Engine uses schema stability (Sch-S) locks when compiling and executing queries. Sch-S locks do not block any transactional locks, including exclusive (X) locks. Therefore, other transactions, including those with X locks on a table, continue to run while a query is being compiled. However, concurrent DDL operations, and concurrent DML operations that acquire Sch-M locks, cannot be performed on the table.

[Bulk Update Locks](javascript:void(0))

The Database Engine uses bulk update (BU) locks when bulk copying data into a table, and either the **TABLOCK** hint is specified or the **table lock on bulk load**table option is set using **sp\_tableoption**. Bulk update (BU) locks allow multiple threads to bulk load data concurrently into the same table while preventing other processes that are not bulk loading data from accessing the table.

[Key-Range Locks](javascript:void(0))

Key-range locks protect a range of rows implicitly included in a record set being read by a Transact-SQL statement while using the serializable transaction isolation level. Key-range locking prevents phantom reads. By protecting the ranges of keys between rows, it also prevents phantom insertions or deletions into a record set accessed by a transaction.

[See Also](javascript:void(0))

Concepts

[Lock Granularity and Hierarchies](http://msdn.microsoft.com/en-us/library/ms189849(v=sql.105).aspx)

[Lock Compatibility (Database Engine)](http://msdn.microsoft.com/en-us/library/ms186396(v=sql.105).aspx)

[Deadlocking](http://msdn.microsoft.com/en-us/library/ms177433(v=sql.105).aspx)

[Cursor Locking](http://msdn.microsoft.com/en-us/library/ms190211(v=sql.105).aspx)

[Key-Range Locking](http://msdn.microsoft.com/en-us/library/ms191272(v=sql.105).aspx)

[Locking Hints](http://msdn.microsoft.com/en-us/library/ms189857(v=sql.105).aspx)