Understanding Locking in SQL Server

<http://msdn.microsoft.com/en-us/library/aa213039(v=sql.80).aspx>

**SQL Server 2000**

63 out of 89 rated this helpful - [Rate this topic](http://msdn.microsoft.com/en-us/library/aa213039(v=sql.80).aspx#feedback)

Microsoft® SQL Server™ 2000 has multigranular locking that allows different types of resources to be locked by a transaction. To minimize the cost of locking, SQL Server locks resources automatically at a level appropriate to the task. Locking at a smaller granularity, such as rows, increases concurrency, but has a higher overhead because more locks must be held if many rows are locked. Locking at a larger granularity, such as tables, are expensive in terms of concurrency because locking an entire table restricts access to any part of the table by other transactions, but has a lower overhead because fewer locks are being maintained.

SQL Server can lock these resources (listed in order of increasing granularity).

|  |  |
| --- | --- |
| **Resource** | **Description** |
| RID | Row identifier. Used to lock a single row within a table. |
| Key | Row lock within an index. Used to protect key ranges in serializable transactions. |
| Page | 8 kilobyte –(KB) data page or index page. |
| Extent | Contiguous group of eight data pages or index pages. |
| Table | Entire table, including all data and indexes. |
| DB | Database. |

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

SQL Server uses these resource lock modes.

|  |  |
| --- | --- |
| **Lock mode** | **Description** |
| Shared (S) | Used for operations that do not change or update data (read-only operations), 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. |

Shared Locks

Shared (S) locks allow concurrent transactions to read (SELECT) a resource. 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 data has been read, 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

Update (U) locks prevent a common form of deadlock. A typical update pattern consists of a transaction reading a record, acquiring a shared (S) lock on the resource (page or row), and then modifying the row, 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. Otherwise, the lock is converted to a shared-mode lock.

Exclusive Locks

Exclusive (X) locks prevent access to a resource by concurrent transactions. No other transactions can read or modify data locked with an exclusive (X) lock.

Intent Locks

An intent lock indicates that SQL Server wants to acquire a shared (S) lock or exclusive (X) lock on some of the resources lower down in the hierarchy. For example, a shared intent lock placed at the table level means that a transaction intends on placing shared (S) locks 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 SQL Server 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) | Indicates the intention of a transaction to read some (but not all) resources lower in the hierarchy by placing S locks on those individual resources. |
| Intent exclusive (IX) | Indicates the intention of a transaction to modify some (but not all) resources lower in the hierarchy by placing X locks on those individual resources. IX is a superset of IS. |
| Shared with intent exclusive (SIX) | Indicates the intention of the transaction to read all of the resources lower in the hierarchy and modify some (but not all) resources lower in the hierarchy by placing IX locks on those individual resources. Concurrent IS locks at the top-level resource are allowed. For example, an SIX lock on a table places an SIX lock on the table (allowing concurrent IS locks), and IX locks on the pages being modified (and X 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. |

Schema Locks

Schema modification (Sch-M) locks are used when a table data definition language (DDL) operation (such as adding a column or dropping a table) is being performed.

Schema stability (Sch-S) locks are used when compiling queries. Schema stability (Sch-S) locks do not block any transactional locks, including exclusive (X) locks. Therefore, other transactions can continue to run while a query is being compiled, including transactions with exclusive (X) locks on a table. However, DDL operations cannot be performed on the table.

Bulk Update Locks

Bulk update (BU) locks are used 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 processes to bulk copy data concurrently into the same table while preventing other processes that are not bulk copying data from accessing the table.