

Locking & Blocking Made Simple



Joe Webb
Microsoft SQL Server MVP
WebbTech Solutions, LLC
joew@webbtechsolutions.com



Our Agenda

- The Nature of Multi-User Databases
- The Basics of Locking and Blocking
- Techniques for Controlling Locking

A Bit of Background

Joe Webb

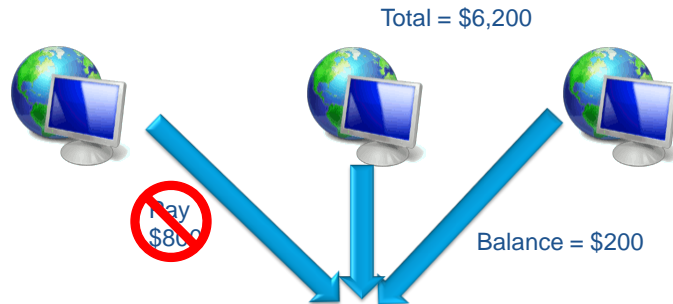
- Started in IT in 1993 after receiving a BSEE & MBA from Auburn University
- Founded WebbTech Solutions in 1996
 - Providing consulting, mentoring, & technical training
- Began working with SQL Server in 1996 with version 6.0
- 5-time recipient of the Microsoft MVP Award for SQL Server
- Author or Coauthor of 3 books
- Served 6 years on the Board of Directors for PASS (sqlpass.org)
 - Served 2 years on the Executive Committee as VP of Marketing and EVP of Finance
- Delivered more than 50 sessions at over 20 conferences in North America & Europe

The Nature of Multi-User Databases





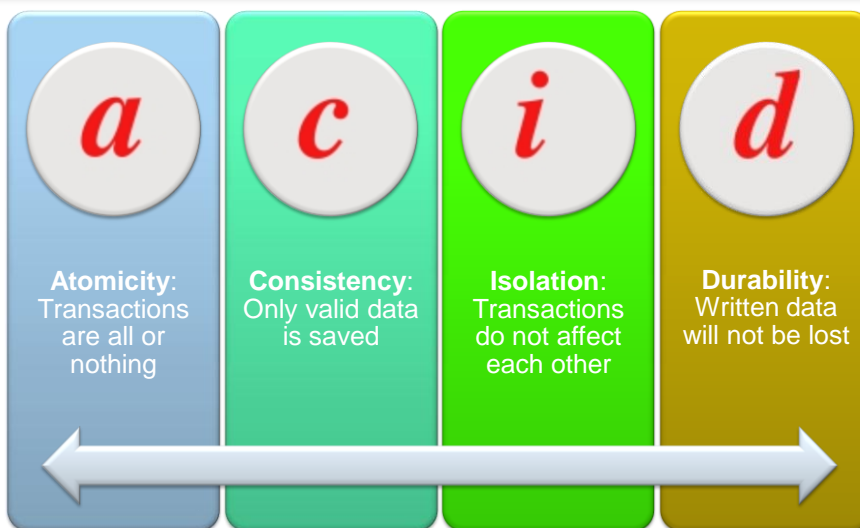
An Example



Customer_ID	Balance
123	\$200.00
124	\$1,500.00
125	\$2,000.00
126	\$2,500.00



The ACID Test



Locking & Blocking



Locking

- Ensures that resources are not being simultaneously used by different processes in conflicting ways
- Provides consistency
- Prevents conflict



Blocking

- Results when one process must wait for another process to release a resource
- Decreases transactional throughput, increases wait times, etc

Types of Locks

Lock Type	Description
Shared (S)	Indicates that the resource is being read by a process
Update (U)	Indicates that a process is going to update the resource. Only one Update lock can exist at a time for a resource. Before writing, it's converted to an Exclusive lock
Exclusive (X)	Used to prevent multiple processes from attempting the same resource at one time.
Intent (I)	Used to establish lock hierarchy. They are placed before lower level locks for efficiency and to protect the lower level locks
Schema (Sch)	Protects the definition of a resource during data manipulation or prevents data manipulation when a DDL command is executing
Bulk Update	Used when copying large amounts of data into a table
Key Range	Protects a range of key values for SERIALIZABLE operations

Lockable Resources



Resource

- Database
- File
- Object
- Page
- Key
- Extent
- RID
- Application
- Metadata
- HOBT
- Allocation unit

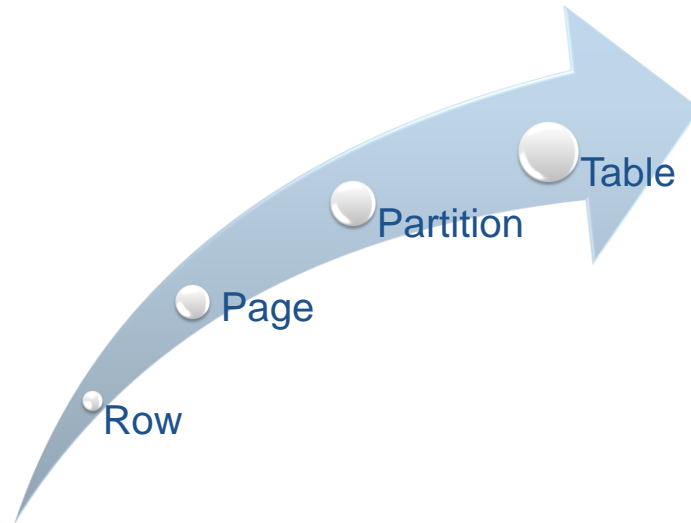
Lock Compatibility

	Shared	Update	Exclusive
Shared			
Update			
Exclusive			

= Ok

= Wait

Lock Granularity



Lock Escalation

- SQL Server attempts to be as granular as possible to increase concurrency
- Lock Escalation is the process of converting many lower-level locks to fewer higher level locks
- Escalations occur when:
 - The number of locks for a table scan is greater than approximately 5,000
 - The amount of non-AWE memory used for locks exceeds 24% and the locks setting is zero
 - The amount of non-AWE memory used for locks exceed 40% and the locks setting is non-zero
- Escalation decreases the amount of server resources required to keep track of locks while increasing the possibility of concurrency conflicts



Techniques for Controlling Locks

- Using Granularity Hints
- Using Transaction Isolation Levels



Granularity Hints

Hint	Description
NOLOCK	Instructs SQL Server to disregard locks placed on resources, allowing "dirty reads"
PAGLOCK	Places locks at the page level, rather than row or table level
ROWLOCK	Places locks on rows rather than pages or tables
TABLOCK	Places locks on a table rather than at a lower level
TABLOCKX	Places an exclusive lock on a table
UPDLOCK	Places an update lock on a resource
XLOCK	Indicates that an exclusive lock is to be placed on the specified resource; maybe used with PAGLOCK, ROWLOCK, and TABLOCK

Transaction Isolation Level



Transaction
Isolation Levels

- Read Uncommitted
- Read Committed
- Repeatable Read
- Serializable
- Snapshot

Transaction Isolation Levels

Read Uncommitted

- Instructs SQL Server to completely ignore normal locking activities
 - Does not place read locks on resources when reading
 - Does not observe write locks on resources when reading
 - Allows “dirty reads” – reading uncommitted data from other transactions
- Most beneficial for lookup tables that seldom change or for Data Warehouse applications
- Can also be invoked using the WITH (NOLOCK) hint



Transaction Isolation Levels

Read Committed

- Reads only data that has been committed to the database
- If resources are exclusively locked, the process will wait indefinitely for the lock to be released
- Prevents “dirty reads” but allows non-repeatable reads and “phantom reads”
- Places a shared lock on resources during reads
- Default transaction isolation level for SQL Server
- Can control how long a transaction will wait for a resource using the LOCK_TIMEOUT option for the connection
 - When using the LOCK_TIMEOUT option, client applications must catch the error raised when a query times out



Transaction Isolation Levels

Repeatable Read

- Designed to ensure that read data is not changed by other processes during a transaction
- Places a shared lock on all read resource; the lock persists until the transaction is completed or rolled back
- Your transaction can see its own changes to data
- Increases the duration of read locks and hence has an adverse affect on concurrency
- Allows “phantom reads”



Transaction Isolation Levels

Serializable

- Places a shared lock on all resources read during a transaction; the lock persists until the transaction is committed or rolled back
- Other transactions cannot insert rows into the table that would change the result of this transaction
- Implemented with Key Range locks
- Prevents “phantom reads” but has a significantly adverse affect on concurrency
- Can also be invoked using the WITH (HOLDLOCK) hint



Transaction Isolation Levels

Read Committed Snapshot

- Implemented by setting the READ_COMMITTED_SNAPSHOT ON database setting, causes READ COMMITTED isolation level to use row versioning
- Provides statement level snapshots using row versioning; versioning information is stored in data pages within tempdb
- Read operations retrieve the latest copy of data from the version store when the statement is started
- Write operations perform like READ COMMITTED using update and exclusive locks



Transaction Isolation Levels

Snapshot Isolation

- Implemented with the `ALLOW_SNAPSHOT ISOLATION ON` database option is set, allows transaction isolation level to be set to `SNAPSHOT`
- Provides transaction level isolation using row versioning
- Row version information is kept in `tempdb` and data is read as of the start of the transaction using the current transaction id
- Write operations use an optimistic locking (ie no update locks are placed) which can cause update conflicts to occur
 - The likelihood of conflicts can be reduced by using the `WITH (UPDLOCK)` hint



Avoiding Excessive Locking

Database Design

- Normalize the database
- Consider column partitioning in cases where diverse access methods are used
- Consider horizontal partitioning to decrease blocking exposure and provide multiple objects to be accessed

Indexing

- Use narrow keys for indexes
- Create adequate indexes for reading data to reduce the number of rows to be read
- Consider including additional columns in nonclustered indexes to convert them to covering indexes



Avoiding Excessive Locking

Optimization

- Optimize queries, resolving the long running queries that consume resources and keep locks open
- Update statistics
- Review query plans to ensure the optimal plans are being used
- Keep transactions as short as possible
- Increase non-AWE memory using the 3GB switch
- Reduce the occurrences of Key Lookups
- Avoid using resource intensive programming techniques such as cursors, etc.



Avoiding Excessive Locking

Transaction Isolation Levels

- Select the least restrictive possible for the given set of tasks
 - For example, choose READ UNCOMMITTED for Business Intelligence scenarios
- Consider implementing row versioning isolation
 - Readers do not block writer; writers do not block readers
 - Overall number of locks is reduced thus reducing system overhead



Tools to Monitor Locking & Blocking

DMVs & System Views

- `sys.dm_tran_locks` - returns lock manager resources and current requests for these resources for the SQL Server instance
- `sys.partitions` – contains information about system resources that may be locked
- `sys.dm_exec_sessions` – returns information about the active sessions

Windows Performance Monitor

- The SQL Server: Locks object provides information such as Average Wait time (ms), Lock Requests / sec, and Lock Waits / sec



Tools to Monitor Locking & Blocking

SQL Profiler

- Use the Blocked Process Report event class
- Must define how long a process is blocked to be considered "blocked" using the 'blocked process threshold' configuration option

sp_blocker_pss08 Procedure

- Stored procedure available from KB 271509 that queries underlying system tables and views to report blocking. Available for SQL Server 2000, 2005, and 2008

Data Collection in SQL Server 2008

- The data collector collects information from the DMVs and preserves that information in a database that can be queried to look for trends over time



Additional Resources

- <http://www.sql-server-performance.com>
- <http://msdn.microsoft.com/en-us/library/ms191168.aspx>
- <http://www.simple-talk.com/content/article.aspx?article=671>
- <http://support.microsoft.com/kb/271509>
- <http://support.microsoft.com/kb/271509>
- “Inside SQL Server” books
- <http://www.webbtechsolutions.com/blog>
- <http://www.twitter.com/joewebb>



Questions?



Thank you!

joew@webbtechsolutions.com

<http://www.webbtechsolutions.com/blog>

<http://www.twitter.com/joewebb>

<http://speakerrate.com/joewebb>