

SQLskills Online Immersion Event

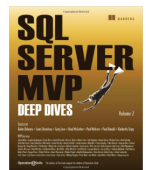
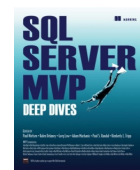
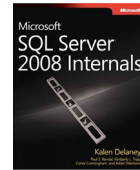
IETLB: Transactions, Locking, Blocking, Isolation, and Versioning

Kimberly L. Tripp
President / Founder, SQLskills.com
Kimberly@SQLskills.com
@KimberlyLTripp



Author/Instructor: Kimberly L. Tripp

- Consultant/Trainer/Speaker/Writer
- President/Founder, SYSolutions, Inc.
 - e-mail: Kimberly@SQLskills.com
 - blog: <http://www.SQLskills.com/blogs/Kimberly>
 - Twitter: @KimberlyLTripp
- Author/Instructor for SQL Server Immersion Events
- Instructor for multiple rotations of both the SQL MCM & Sharepoint MCM
- Author/Manager of SQL Server 2005 & 2008 Launch Content
- Author/Speaker at Microsoft TechEd, SQLPASS, ITForum, TechDays, SQLIntersection
- Author of several SQL Server Whitepapers on MSDN/TechNet: Partitioning, Snapshot Isolation, Manageability, SQLCLR for DBAs
- Author/Presenter for more than 25 online webcasts on MSDN and TechNet
- Author/Presenter for multiple online courses at Pluralsight
- Co-author MSPress Title: SQL Server 2008 Internals, the SQL Server MVP Project (1 & 2), and SQL Server 2000 High Availability
- Owner and Technical Content Manager of the SQLIntersection conference (with @PaulRandal)



PLURALSIGHT



Slide 2

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Daily Format – Tue, Wed, and Th

- **Lecture 1: 90 minutes from 10:00am until 11:30am PT**
 - Please use “CHAT” for questions *during* the lecture
- **Open Q&A 1: 30 minutes from 11:30am until 12:00 noon PT**
 - Will address unanswered questions from chat
 - May open up for “open mic” questions
- **Mandatory break: 30 minutes from 12:00 noon until 12:30pm PT**
 - Everyone needs a bit of a break!
- **Lecture 2: 90 minutes from 12:30pm until 2:00pm PT**
 - Please use “CHAT” for questions *during* the lecture
- **Open Q&A 1: up to 60 minutes from 2:00pm until 3:00pm PT**
 - Will address unanswered questions from chat
 - Will open up for “open mic” questions

Time Conversions

10:00am PT

= 1:00pm ET

= 6:00pm UTC



Slide 3

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Course Resources & Pluralsight Access

- **Paul (paul@sqlskills.com) will send you an email for a FREE 30-day access to ALL SQLskills online training classes on Pluralsight**
 - No strings attached, no credit-card required
 - If you don't receive it, send mail to Paul
- **Course resources will be sent upon completion (Friday+)**
 - I may want to update a slide and/or add some notes/resources based on questions and discussions
 - The resource zip will be password protected. Please do not distribute; this is for course attendees only. The password will be sent in email.
- **Email me (Kimberly@SQLskills.com) if you have problems finding it or opening it from Monday (Oct 15) forward; I plan to finalize the resources (complete the whiteboard, etc.) once class completes.**



PLURALSIGHT



Slide 4

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Course Resources

- **Course content (this deck)**
- **Whiteboard**
- **Demos**
 - All instructor-led demo scripts
 - All reference scripts
- **Reference links (in the “demo zip”)**
- **Online recordings for these sessions**

**Please do not redistribute: These resources are for IEVLT attendees only.
Please respect our IP and time in creating these resources and do not share/forward.**



Slide 5

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Module 1: Batches, Transactions, and Error Handling

- **Batches**
- **Transaction**
- **Transaction Mode**
- **Transaction Termination**
- **Transaction Flow / Logic**
- **Error Handling**



Slide 6

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Module 2: The Anatomy of a Data Modification

- Data
- Caching
- Logging
- Locking
- Durability



Slide 7

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Module 3: Locking / Isolation

- Statement Accuracy vs. Data Accessibility/Concurrency
- Read Committed
- Versioning
- Lock Hierarchy
- Row Locks
- Page Locks



Slide 8

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Module 4: Table Maintenance and Schema Locks

- Schema locks (object's schema)
- Schema locks (object)
- Relaxed FIFO
- Lock escalation

If time permits:

This module will wrap-up our 3rd session. We'll need all of the 3rd day+ for Module 5 and 6.

You will still receive all of the content!



Slide 9

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Module 5: Locking, Blocking, and Deadlocks (Intro)

- Blocking
- Lock Analysis
- Deadlocks
- Deadlock Analysis
- Deadlock Mitigation



Slide 10

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Module 6: Versioning

- **Understanding Isolation**
 - ❑ Read Uncommitted
 - ❑ Read Committed with Locking
 - ❑ Read Committed with Versioning
 - ❑ Repeatable Reads
 - ❑ Serializable
 - ❑ Snapshot
- **Controlling isolation levels**
- **Statement-level read consistency**
- **Transaction-level read consistency**
- **Overhead/monitoring**
- **Isolation summary**



Slide 11

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Module 1: Batches, Transactions, and Error Handling



Overview

- Batches
- Transaction
- Transaction Mode
- Transaction Termination
- Transaction Flow / Logic
- Error Handling



Slide 13

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Batches

- **Parsed as a unit so if any statement fails syntax then NO statements (in that batch) are executed**
 - SQL Server proceeds to next batch
 - **At execution, if a statement fails then that statement is rolled back and SQL Server exits the batch (or falls into catch block) and proceeds to the next batch**
 - Any statement following the failed statement is NOT executed
- ```
SELECT * FROM Adventureworks.person.person
SELECT * FROM Adventureworks.person.address
SELECT * FORM Adventureworks.person.emailaddress
go
```

Error: Server: Msg 170, Level 15, State 1, Line 3  
Line 3: Incorrect syntax near 'form'.



Slide 14

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

## Transactions

- **Controlled by transaction mode of session**
  - Defined explicitly
  - Defined at connection level
- **Allows multiple statements to be treated as a single logical unit of work**
- **Handled automatically across databases on the same server (i.e. instance)**
- **Handled by MSDTC explicitly with**
  - `BEGIN DISTRIBUTED TRANSACTION`
  - `COMMIT TRANSACTION`
- **Requires error handling logic**



Slide 15

<http://www.SQLskills.com>  
© SQLskills Online Immersion Event. All rights reserved.

## Transaction Mode

- **Auto-commit transaction (default)**
  - Statement-level implicit transaction
  - Each statement commits as a single unit
- **Explicit transaction (user-defined)**
  - `BEGIN TRANSACTION`
  - `COMMIT TRANSACTION`
- **Implicit transaction**
  - Session Level Setting
  - `SET IMPLICIT_TRANSACTIONS ON`
- **Batch scoped transactions (2005+)**
  - Only when MARS enabled on client



Slide 16

<http://www.SQLskills.com>  
© SQLskills Online Immersion Event. All rights reserved.



## Transaction Termination (1)

- **Resource error: automatically handled**
  - If the statement fails due to a SQL Server resource error (for example, the transaction log fills), SQL Server maintains data integrity automatically
- **User-defined error: programmatically handled**
  - Conditionally, when your business rules (i.e. constraint violation or a lock timeout (more on this coming up)) are violated, YOU must programmatically determine the fate of the transaction



Slide 17

<http://www.SQLskills.com>  
© SQLskills Online Immersion Event. All rights reserved.

## Transaction Termination (2)

- **SET ARITHABORT (on by default in SSMS/sqlcmd)**
  - ON – rollback statement/transaction when divide-by-zero or arithmetic overflow error occurs
  - OFF – return NULL
- **SET XACT\_ABORT (off by default)**
  - ON – rollback statement/transaction when runtime error occurs
- **Viewing session settings:**
  - DBCC USEROPTIONS
  - ```
SELECT *  
FROM sys.dm_exec_sessions  
WHERE session_id = @@spid (optionally)
```



Slide 18

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Understanding Transactions

- **BEGIN TRANSACTION**
 - Begins a new transaction
- **Supports a “name” but [VERY] limited uses**
- **Increments @@TRANCOUNT**
- **Supports special “marked transactions”**
 - BEGIN TRANSACTION** TransactionName
 - WITH MARK** ['important transaction']
 - This does NOT control rollback points (you’re always rolling back THE transaction)
 - Useful for restore/recovery operations
 - **STOPATMARK** [AFTER *datetime*]
 - Includes marked transaction
 - **STOPBEFOREMARK** [AFTER *datetime*]
 - Includes all operations up to – but not including the marked transaction



Slide 19

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Understanding Transactions

- **COMMIT TRANSACTION**
 - Finalizes the transaction
- **Decrements @@TRANCOUNT by 1**
- **Does NOT commit the transaction UNLESS the @@TRANCOUNT is 0**
- **Must correspond to a BEGIN TRANSACTION**
 - BEGIN TRANSACTION**
 - SQL statements*
 - COMMIT TRANSACTION**
- **Or, session must have implicit_transactions turned on. This is not recommended.**
 - SET IMPLICIT_TRANSACTIONS ON**
 - SQL statements...*
 - COMMIT TRANSACTION**
 - (open transaction until commit)



Slide 20

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Understanding Transactions

- **ROLLBACK TRANSACTION**
 - Aborts the transaction
- **Returns @@TRANCOUNT to 0**
- **Cancels all levels of “nested” transactions**
- **You can rollback to a savepoint and still be within the bounds of the transaction (see next slide)**
- **However, you cannot rollback to a named transaction**
Msg 6401, Level 16, State 1, Line 5
Cannot roll back TransactionName. No transaction or savepoint of that name was found.
- **There is ALWAYS only one transaction in the context of rollback**
- **Only savepoints can be used to undo state within a transaction**



Slide 21

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Understanding Transactions

- **SAVE TRANSACTION (a.k.a. savepoints)**
 - Are a “stack” (without any regard for “nesting” because that doesn’t exist) and you CAN reuse the same name
- **Does not affect @@TRANCOUNT**
- **When you ROLLBACK to a savepoint you must still commit or rollback the pending transaction**
- **Commonly used inside of procedural code to limit rollback effect**

```
BEGIN TRANSACTION
    SAVE TRANSACTION procname_tranname_statedesc
    Procedural code/logic
    If problem... ROLLBACK TRANSACTION procname_tranname_statedesc
COMMIT TRAN
RETURN [int]
```
- **A rollback will always rollback to the most recent savepoint (not necessarily the scoped savepoint, even in procedures)**



Slide 22

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Session Settings and Client Connectivity

From Pluralsight course:
Optimizing Procedures, Part 2
Section 2: Session Settings

SET Options	Required for Perf Features	Default Server Value	SSMS	SQLCMD	SQL Server Agent	Default OLE DB and ODBC Value	Default DB-Library Value	.NET / Your App
ANSI_DEFAULTS ²	NO	OFF	OFF	OFF	OFF	OFF	OFF	?
ANSI_NULL_DFLT_ON ²	NO	OFF	ON	ON	ON	ON	OFF	?
ANSI_NULLS ^{1,3}	YES = ON	OFF	ON	ON	ON	ON	OFF	?
ANSI_PADDING ^{2,3}	YES = ON	ON	ON	ON	ON	ON	OFF	?
ANSI_WARNINGS ²	YES = ON	OFF	ON	ON	ON	ON	OFF	?
ARITHABORT ²	YES = ON	ON	ON	OFF	OFF	OFF	OFF	?
CONCAT_NULL_YIELDS_NULL ^{2,3}	YES = ON	OFF	ON	ON	ON	ON	OFF	?
NUMERIC_ROUNDABORT ²	YES = OFF	OFF	OFF	OFF	OFF	OFF	OFF	?
QUOTED_IDENTIFIER ¹	YES = ON	OFF	ON	OFF	OFF	ON	OFF	?

(1) The only state that's important is how it's set when the stored procedure is CREATED; the runtime setting is irrelevant.

(2) If different than existing plan in cache, will be recompiled and added to the plan cache; performance may vary at execution.

(3) In a future version of SQL Server, these options will always be ON and any applications that explicitly set the option to OFF will produce an error. Avoid using this feature in new development work and plan to modify applications that currently use this feature.

pluralsight

Types of Errors

Categories of Errors Raised in T-SQL code

- Syntax
 - (SELEC * FROM SomeTable)
- Object resolution
 - (SELECT * FROM NonExistentTable)
- Statement termination errors
 - Execution resumes at next statement
- Scope/level termination errors
 - Execution resumes in calling procedure
- Batch termination errors
- Batch termination and rollback transaction errors
- Connection termination errors
- Server termination errors

Error Severities

■ Error Severities Are Grouped

- Severity 0-10 - Informational Messages
 - Informational messages do not jump to the TRY block of a TRY...CATCH construct
 - Severity 0 – T-SQL PRINT
 - Severity 1-9 – Informational messages / can convey state
 - Severity 10 - Converted to severity 0
- Severity 11-15 - Programming Errors / can be corrected by the user
 - Each has a primary purpose
- Severity 16: user defined errors, most commonly used
- Severity 17-25 - Resource Errors
 - Severity 17-19 – Errors that cannot be corrected by the user
 - Severity 19 and above - must be sysadmin to raise
 - Severity 20-25 - Terminates the connection
 - Cannot be caught by the CATCH block of a TRY...CATCH construct because execution is aborted when the connection is lost



Slide 25

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Handling Errors – The Old Way

- After each statement you need to test the state of @@ERROR
- To centralize error checking code, use labels with GOTO and RETURN statements
- Errors such as data type conversion errors causes batch to terminate; not every error can be trapped
- Makes for a lot of additional code

```
DECLARE @Error INT
SELECT @Error = @@Error
-- If you don't do this then you lose @@Error on the "success" of the IF
IF @@ERROR <> 0
BEGIN
    RAISERROR...
    ROLLBACK
    RETURN
END
```



Slide 26

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Handling Exceptions

- **SQL Server 2005 added exception handling**
 - Error handling in previous SQL Server versions tedious
 - @@ERROR set on each statement
 - Set variable with @@ERROR, then check value
 - BEGIN-END TRY BEGIN-END CATCH blocks in SQL Server 2005
 - Semantic equivalent of BEGIN-END blocks
 - Additional functions return error info
 - ERROR_NUMBER() – number of the error
 - ERROR_SEVERITY() – severity
 - ERROR_STATE() – error state number
 - ERROR_MESSAGE() – message text
 - ERROR_LINE() – line that generated the error
 - ERROR_PROCEDURE() – procedure



Slide 27

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Try/Catch Block Programming

```
BEGIN TRANSACTION
BEGIN TRY
    -- Generate a constraint violation error
    DELETE FROM Production.Product
    WHERE ProductID = 980;
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() AS ErrorNumber,
           ERROR_SEVERITY() AS ErrorSeverity,
           ERROR_STATE() as ErrorState,
           ERROR_MESSAGE() as ErrorMessage;

    -- no matter what – rollback?
    -- Or, conditionally?
    -- Could also rollback to a savepoint here
END CATCH

-- Anything else to do? (decrement @@trancount)
GO
```

<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/try-catch-transact-sql?view=sql-server-2017>



Slide 28

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Exception Handling: TRY/CATCH Semantics

- TRY/CATCH constructs may be nested
- Thrown exceptions are caught by the CATCH block associated with the closest compiled TRY/CATCH construct
- Error details are available at any time in catch block (not outside CATCH)
- Could result in an uncommittable transaction – best to check XACT_STATE() within CATCH block:

```
IF XACT_STATE() = -1
BEGIN
    RAISERROR (N'The transaction is in an uncommittable state. Rolling back
transaction.')
    ROLLBACK TRANSACTION;
END
ELSE
BEGIN
    ROLLBACK TRAN SavePoint1TerribleNotUniqueName -- Only controls state...
    COMMIT TRANSACTION -- must be used to reset @@trancount
END
```



Slide 29

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

TRY...CATCH, XACT_ABORT, and Uncommittable Transactions

- If XACT_ABORT is ON
 - And ANY error is caught
 - The transaction cannot commit (even partially, with a savepoint)
 - To be honest, they could allow it (if you rolled back to a point PRIOR to when the uncommittable statement occurred) BUT instead of trying to determine if that's the case, SQL disallows it and puts the transaction into an uncommittable state
 - One could argue about the overall logic of the process...
 - And, if "ROLLBACK" is the intention then ROLLBACK TRAN might be desirable
 - Breaks the overall logic of easily passing control but you CAN put XACT_ABORT in procedures to guarantee desired behavior
 - Long story short, **the transaction** becomes uncommittable and therefore doesn't allow savepoints
- If XACT_ABORT is OFF
 - *YOU* decide the fate of the transaction and the logic
 - You can do whatever you want AND you can use savepoints



Slide 30

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Exception Handling: (Re)throwing an Error

- **RAISERROR raises real errors within the scope of a TRY or CATCH block**
 - "Throws" within the scope of a TRY block
- **To "Rethrow"**
 - Capture error details within the catch block
 - Log error if desired
 - Pass error details to RAISERROR
 - Get procedure and line number from message
 - Can't throw a "system" error with RAISERROR
 - One workaround is to add 50000 to original error, then subtract 50000 on caller to get real number
 - Caller/client code that depends on error numbers will need to be rewritten
- **THROW statement in SQL 2012 accomplishes this**

 *hidden slide
w/extra details*



Slide 31

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

THROW and FORMATMESSAGE

- **Adds required functionality to TRY/CATCH**
 - You can (re)throw a system error
 - Always uses severity 16
- **Use FORMATMESSAGE in conjunction**
 - To format user-defined messages
 - Does not completely replace RAISERROR
 - Can't write to error log
 - Can't write error WITH NOWAIT

 *hidden slide
w/extra details*



Slide 32

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Review

- Batches
- Transaction
- Transaction Mode
- Transaction Termination
- Transaction Flow / Logic
- Error Handling



Slide 33

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Questions!



Module 2: The Anatomy of a Data Modification



Overview

- Data
- Caching
- Logging
- Locking
- Durability



Slide 36

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Anatomy of a Data Modification

- **User/application sends an UPDATE query**
 - The update is highly selective (only 5 rows)
- **Indexes exist to aid in finding these rows efficiently**
- **The update is a SINGLE statement batch therefore this is an IMPLICIT transaction**
 - Transactions can be 'explicit' or 'implicit'
 - Explicit transactions are controlled by the user
 - Started with BEGIN TRAN
 - Ended with COMMIT TRAN or ROLLBACK TRAN
 - Implicit transactions are created internally by SQL Server and committed automatically when the operations complete
 - And obviously rolled-back if something goes wrong



Slide 37

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Anatomy of a Data Modification

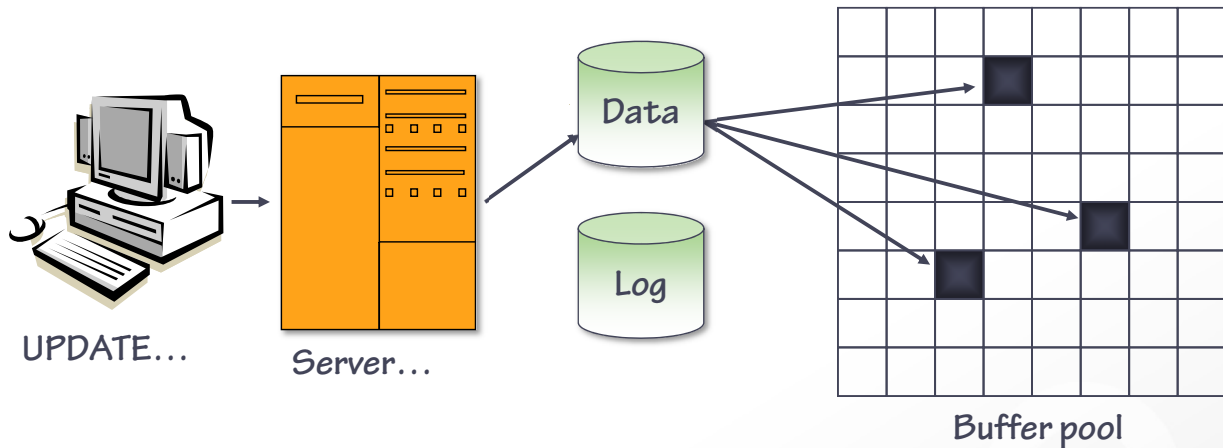
- **Server receives the request and locates the data in cache OR reads the data from disk into cache**
 - Since this is highly selective only the necessary pages are read into cache (maybe a few extra but that's not important here)
 - Let's use an example where the 5 rows being modified are located on 3 different data pages



Slide 38

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

What it looks like: Data Reading From Disk



Slide 39

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Anatomy of a Data Modification

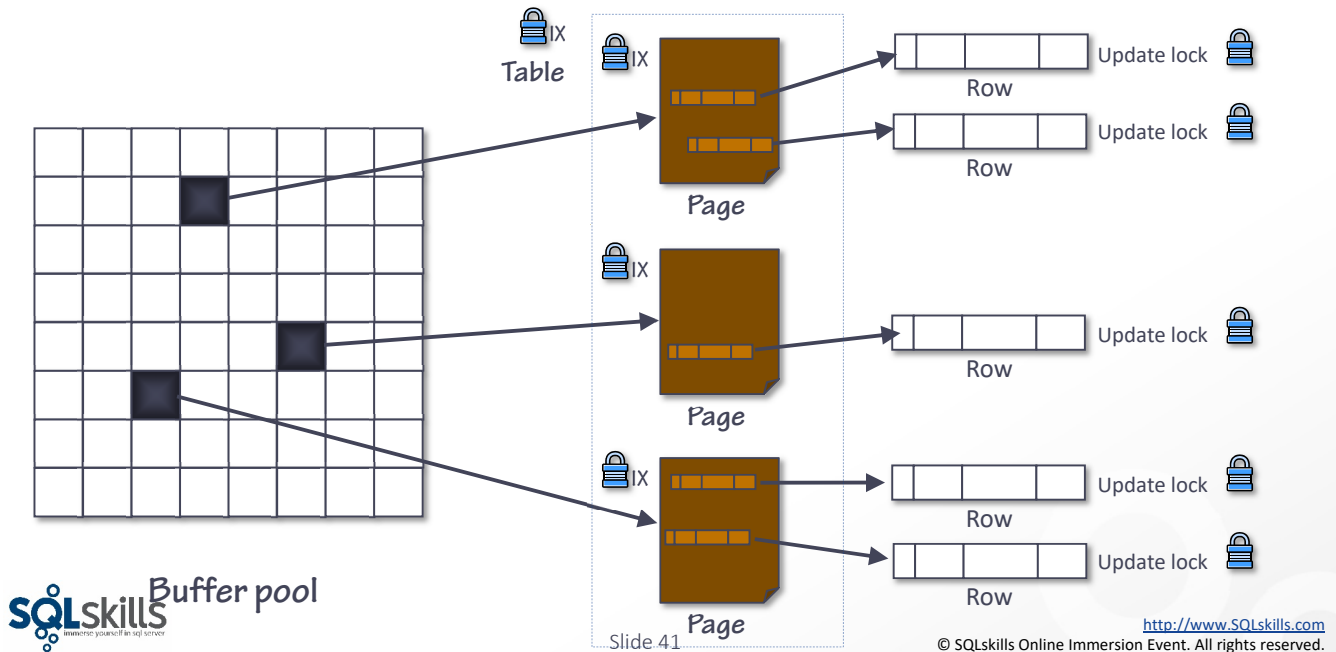
- **SQL Server proceeds to lock the necessary data**
 - Locks are necessary to give a consistent point FOR ALL rows from which to start
 - If any other transaction(s) have ANY of these rows locked we will wait until ALL locks have been acquired before we can proceed
 - Locks are initially taken to stabilize the rows and then upgraded to exclusive locks
 - In the case of this update (because it's highly selective and because indexes exist to make this possible) SQL Server will use row level locking



Slide 40

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

What It Looks Like: Acquiring Locks

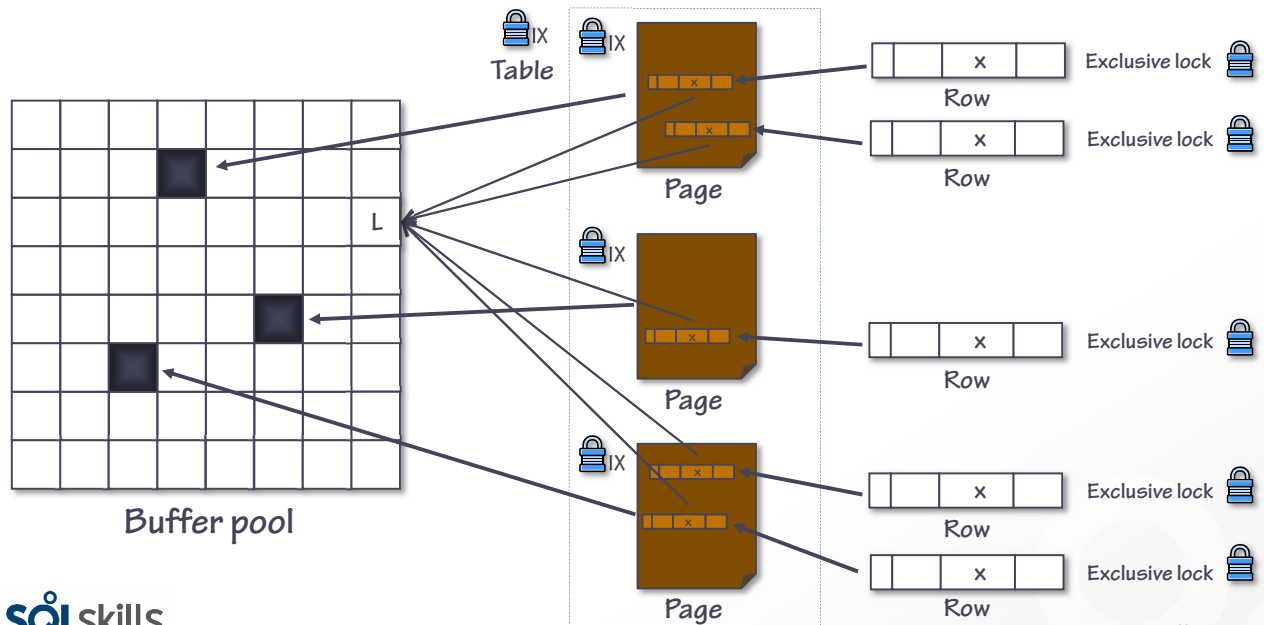


Anatomy of a Data Modification

*hidden slide
notes for prior slide*

- The rows are locked but there are also 'intent' locks at higher levels to make sure other larger locks (like other potentially conflicting page or table level locks) are not attempted and then fail
 - This transaction holds the following locks:
 - 5 update row-level locks
 - 3 intent-exclusive page-level locks
 - 1 intent-exclusive table-level lock
 - The connection also holds a shared database-level lock
- And if indexes are accessed/used then there might be additional locks required – to read the data – they are not significant here

What It Looks Like: Modifications



Slide 43

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Anatomy of a Data Modification

*hidden slide
notes for prior slide*

- SQL Server can now begin to make the modifications
- For EVERY row the process will include:
 - Change to a stricter lock (eXclusive lock)
 - An update lock helps to allow better concurrency by being compatible with other shared locks (readers). Readers can read the pre-modified data as it is transactionally consistent
 - The eXclusive lock is required to make the change because once modified no other reads should be able to see this un-committed change
 - Make the modification (in cache)
 - Log the modification to the transaction log pages (also in cache)



Slide 44

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Anatomy of a Data Modification

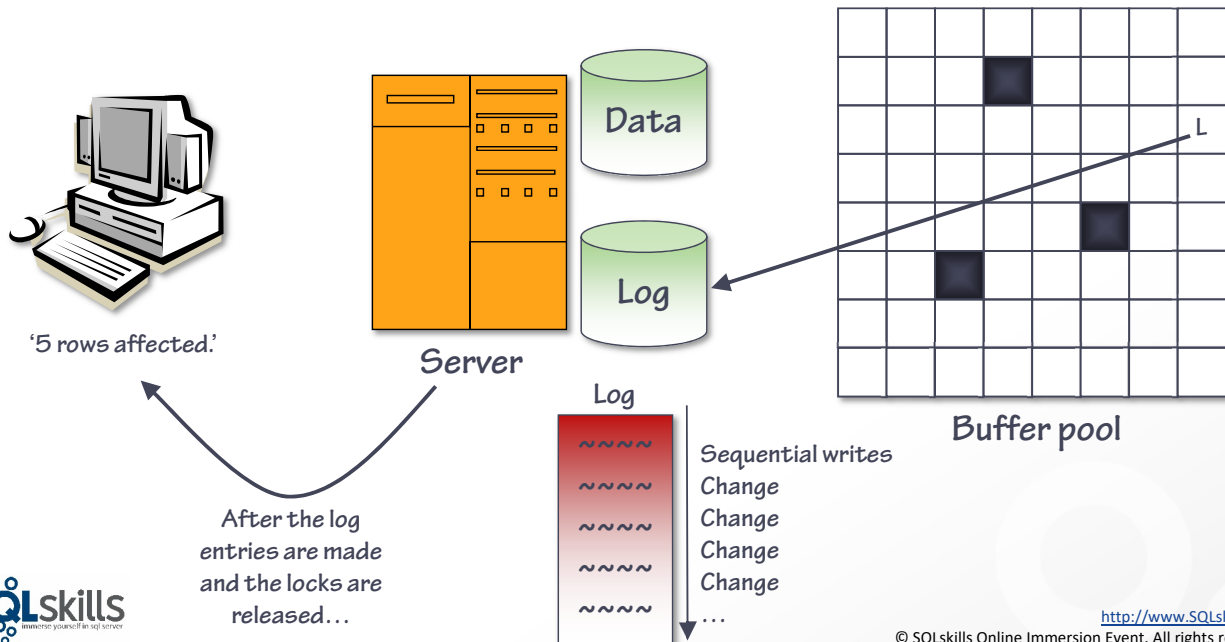
- Finally, the transaction is complete
- This is the **MOST** critical part (the key to Durability)
 - All rows have been modified
 - There are no other statements in this batch (because it's an implicit transaction)
- Steps are:
 - Write all log records for the transaction to the transaction log ON DISK (forced write-through to disk) = durable
 - This forces all of the transaction log up to the point of the COMMIT TRAN log record to be written to disk, regardless of which transaction it is for
 - Release all locks held by the transaction
 - Acknowledge the commit to the user/application:
 - A message may come back (e.g. 5 rows affected but not when NOCOUNT ON)
 - @@TRANCOUNT = 0 (this is what *really* defines the END of a transaction)



Slide 45

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

What It Looks Like: Committing



<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

So Now What?

- The transaction log ON DISK contains a record of the changes made to the database by the transaction
- The data pages in the buffer pool reflect the changes made to the database by the transaction
- When do the up-to-date data pages get written from buffer pool into the data files on disk?

Checkpoint



Slide 47

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Anatomy of a Data Modification: Where Are We At?

- **Optimization**
 - Data is very random
 - Log is sequential
- **Locks**
 - Granularity
 - Duration
 - Escalation
- **Transactions**
 - Can make a mess of things if you don't know what you're doing...



Slide 48

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Tips to Minimize Blocking

- **Write efficient transactions; keep them short and in one batch**
- **Do not allow interaction in the midst of the batch**
- **Use indexes to help SQL Server find, and lock, only the necessary data**
 - More details in Pluralsight course on indexing
- **Use a flavor of versioning or maybe even NOLOCK**
 - More details in the versioning module
- **Consider running decision-support queries and/or analysis on a secondary relational data warehouse / server**
- **In summary: locking becomes problematic “blocking” when long-running [inefficient] and conflicting transactions execute**
 - Shorter transaction times mean less potential blocking...



Slide 49

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Review

- **Data**
- **Caching**
- **Logging**
- **Locking**
- **Durability**



Slide 50

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Questions!



Module 3: Locking / Isolation



Overview

- Statement Accuracy vs. Data Accessibility/Concurrency
- Read Committed
- Versioning
- Lock Hierarchy
- Row Locks
- Page Locks



Slide 53

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Statement Accuracy vs. Data Accessibility/Concurrency

- **Read committed (with locking) does not provide a point in time to which a statement reconciles**
 - This will make sense as we discuss locks types and the length of time for which they're held
- **Sybase (originally) designed "read committed" using the ANSI/ISO standards**
 - Read committed also known as "Inconsistent Analysis"
 - Their design has both pros/cons:
 - Positive: concurrency (data that has not YET been modified, can still be read)
 - Negative: the only accuracy is that a row cannot be read if it's been modified; there's no relationship of that read to the overall statement
 - Leads to inaccuracies
 - Non-repeatable reads
 - Phantoms
- **Oracle is "read committed" by default but they do NOT adhere to the standards and do not have inconsistent analysis in read committed (we CAN achieve this)**



Slide 54

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Locking / Versioning

- SQL Server pre-2005 accomplishes isolation semantics through locking only
- SQL Server 2005 introduced versioning
 - 4 possible states of isolation for your database based on either:
 - Locking only
 - Versioning + locking
 - Locking is always used for writers
 - Versioning *can* be used for readers
 - With versioning enabled you still have locking, just more sparingly
 - Writers lock (for other writers)
 - Intent locks, metadata locks, other locks are same
- Default isolation level is ANSI/ISO “read committed” (using LOCKING)
 - EXCEPTION: Default isolation level for Windows Azure SQL Database is read committed using row versioning (RCSI/read_committed_snapshot)
- SQL Server implementation uses locking for all levels (for writers)



Slide 55

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Lock Hierarchy

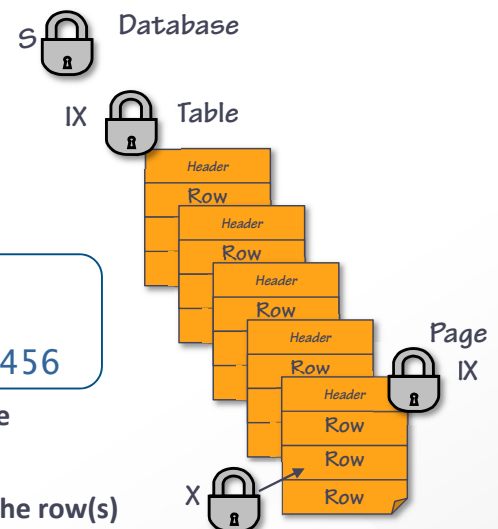
On connect, SQL Server allows access to the database and gives a Shared database lock to the connection.

Imagine the user submitting a query to modify a single row:

```
UPDATE dbo.Products
SET Amount = 40.45
WHERE ProductId = 112456
```

To perform the update SQL Server needs to first access the table and lock it with an IX (Intent eXclusive) lock.

Second, SQL Server needs to find the page on which the row(s) exist and lock them with IX locks. If an index exists this can speed the request and limit the number of pages accessed.



Finally, an eXclusive lock is acquired on the row(s) that needs to be modified.



Slide 56

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Row Locks: Shared, Update, and eXclusive

- **Shared:** many readers
- **Update:** reading with the intent to modify but has NOT yet modified
 - Allows better concurrency
- **eXclusive:** has made a modification and the transaction is still pending

R e q u e s t e d		Lock HELD		
		<u>S</u> hared	<u>U</u> pdate	e <u>X</u> clusive
	<u>S</u> hared	Granted	Granted	WAIT
	<u>U</u> pdate	Granted	WAIT	WAIT
	e <u>X</u> clusive	WAIT	WAIT	WAIT



Slide 57

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Shared, Update, and eXclusive

- **By DEFAULT (read committed [2] locking)**
 - Shared locks: read locks held until the resource has been read and processed
 - These locks are released almost immediately.
 - eXclusive locks: locks held for writers from just before the transaction modifies the data
 - These locks are not released until the owner (the transaction) has completed (i.e. committed).
 - Update locks: acquired at the beginning of the statement to isolate all of the needed rows for an update to guarantee a consistent starting point!
- **In other isolation levels, the behavior of locks can be different; we'll cover this in the versioning module**
 - Read uncommitted [1] – row locks are not used
 - Repeatable reads [3] – shared locks are used and held
 - Serializable [4] – key range locks and other resource (table) locks are used/held
 - Read committed versioning [2] and Snapshot Isolation [5] use versioning

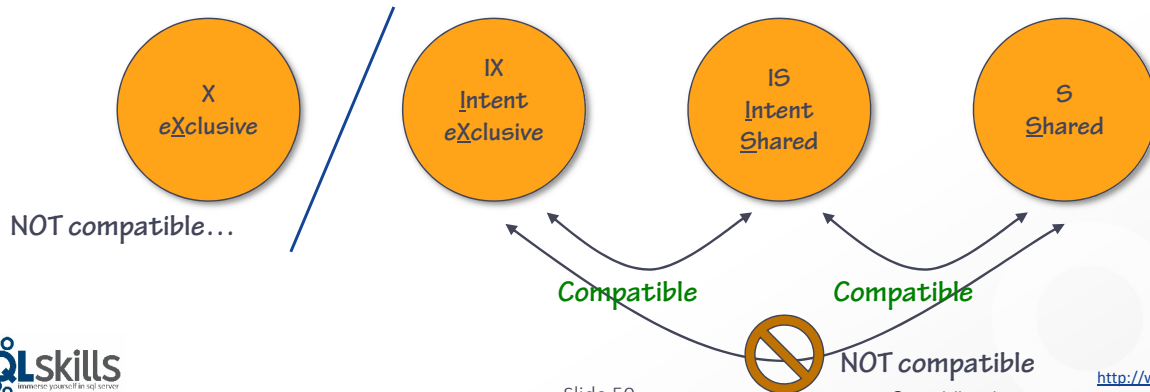


Slide 58

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Page and Table Level Locks: Intent Locks

- Shared: only readers on the table
- Intent Shared: reader with a shared page or row level lock at a lower granularity
- Intent eXclusive: writer with an exclusive page or row level lock at a lower granularity
- eXclusive: only ONE writer on the table



Slide 59

© SQLskills Online Immersion Event. All rights reserved. <http://www.SQLskills.com>

Review

- Statement Accuracy vs. Data Accessibility/Concurrency
- Read Committed
- Versioning
- Lock Hierarchy
- Row Locks
- Page Locks



Slide 60

© SQLskills Online Immersion Event. All rights reserved. <http://www.SQLskills.com>

Questions!



Module 4: Table Maintenance and Schema Locks



Overview

- Schema locks (object's schema)
- Schema locks (object)
- Relaxed FIFO
- Lock escalation

If time permits:

This module will wrap-up our 3rd session. We'll need all of the 3rd day+ for Module 5 and 6.

You will still receive all of the content!



Slide 63

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Schema Locks

- Two “types” of “schema” locks (confusing)
 - Schema locks at the object's schema level (meaning table-level)
 - Schema locks at the schema container level (meaning schema.object)
 - Transferring an object from one schema to another, locks the entire destination schema (see <http://bit.ly/260rWol> for an example)
- Schema-stability (SCH-S)
 - Prevents the schema changing or IAM pages being added to the IAM chain
 - Sometimes used when performing allocation order scan
- Schema-modification (SCH-M)
 - Used when the an object's schema must change
 - Changing the definition of the object
 - Maintenance: creating an index / rebuilding an index / partition switching
 - `sp_recompile tablename` (requires a SCH-M on the table) *NOTE: `sp_recompile procedurename` does NOT*
 - Not compatible with any other lock type – think of it as a super-exclusive lock
 - SQL Server 2014 has a low priority lock wait option for partition switching or index rebuilds; this can DRAMATICALLY reduce long blocking chains: <http://tinyurl.com/kc4g7dq>



Slide 64

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Plan Invalidation: A Strange and Weird History

▪ Versions prior to 2012

- If database option: **auto_update_stats** is ON
 - Updating statistics causes plan invalidation
- If database option: **auto_update_stats** is OFF
 - Updating statistics does NOT cause plan invalidation
- If you manually update statistics and have set **auto_update_statistics** to OFF, add **sp_recompile @tname** to your stats scripts (remembering SCH_M problems)
- For more info: Erin Stellato's links about auto update stats/plan invalidation:
 - Statistics and Recompilations: <http://erinstellato.com/2012/01/statistics-recompilations/>
 - Statistics and Recompilations, Part II: <http://erinstellato.com/2012/02/statistics-recompilations-part-ii/>

 *hidden slide
w/extra details*

▪ SQL Server 2012+

- Plan invalidation is NOT affected by the setting of auto update stats
- Plan invalidation does NOT occur if data has not changed
 - Only for UPDATE STATISTICS
 - An index rebuild will update statistics as well as cause plan invalidation, even if no data has changed thus increasing the importance for “only when data has changed” logic in maintenance routines.



Slide 65

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

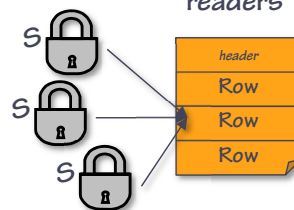
Blocking – Is it Really a Problem? It depends...

- Locks guarantee consistency
- First incompatible lock request waits
- Once an incompatible lock request is made then pending locks will wait even if they're compatible with locks currently held
 - Special case WHEN the requested lock IS compatible WITH ALL pending requests...
- Primary reason for the locks to WAIT
 - Why? To prevent lock starvation

Even compatible
shared locks must
WAIT (or queue)
behind the
incompatible lock



Imagine 3
compatible
readers



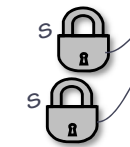
Time

Relaxed FIFO To Reduce Really Long Blocking Chains

- <http://blogs.msdn.com/b/psssql/archive/2009/06/02/sql-server-lock-manager-and-relaxed-fifo.aspx> (<http://bit.ly/NWAYVC>)
- An update is incompatible with another update
- The requested (second) update will wait behind the current update...
- What about shared locks if ONLY update locks? This is OK!

Compatible locks ARE allowed to reduce long queuing problems.
REQUIREMENT: Requested locks must be compatible with ALL currently held AND all pending requests.

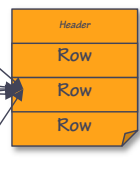
Then, more readers attempt to read...



Then, an UPDATE lock is requested



Same starting point
3 compatible readers



Time

BUG / NOTE: In 2008 R2, a SCH_S lock request was granted despite it being incompatible with the union of granted and waiting requests (when a SCH_M was waiting), which might lead to lock starvation (of the SCH_M). In 2012, the SCH_S lock request is blocked

Nasty Blocking CHAINS

Relaxed FIFO isn't Always Enough

- Imagine someone's running a NOLOCK query against a very large table (the query takes 4 minutes to run [ok, not that nasty])
- This query's running off-hours but the standard is to use NOLOCK
- An **sp_recompile tname** is requested

Unfortunately now...
everybody waits.....

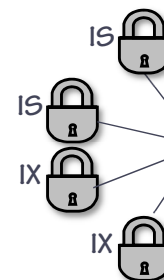
Relaxed FIFO only let's through the folks that are compatible with ALL pending locks (not just those that are currently held).

This can create terribly long blocking chains.



Then an
sp_recompile tname
is executed

Other readers AND
writers are allowed



A NOLOCK
query is running
on the table

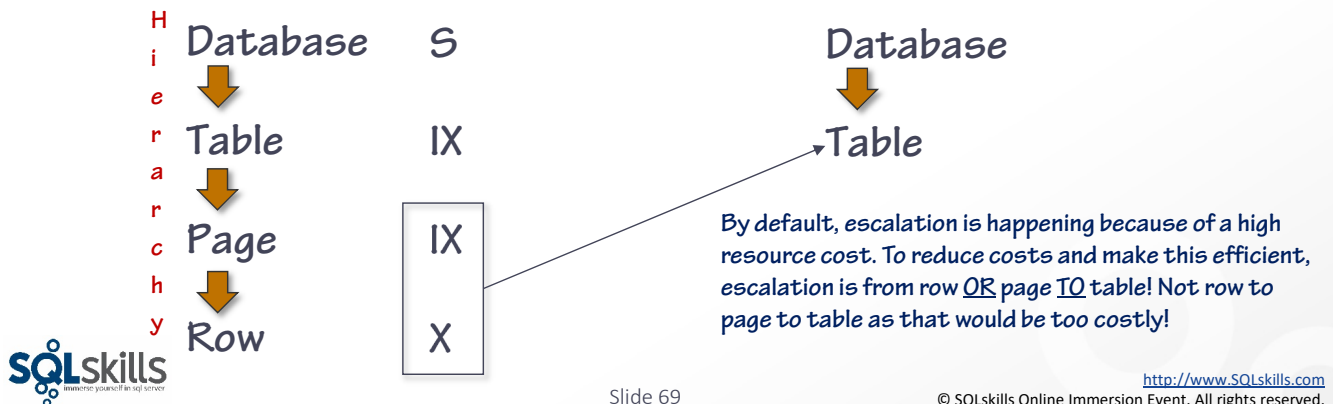


Time

Lock Escalation

- What is 'lock escalation'?

- When the cost to create / maintain the locks is too high (roughly 5000 locks for a single statement / request)
- SQL Server chooses to lock at a higher level in the hierarchy to save lock memory / resources

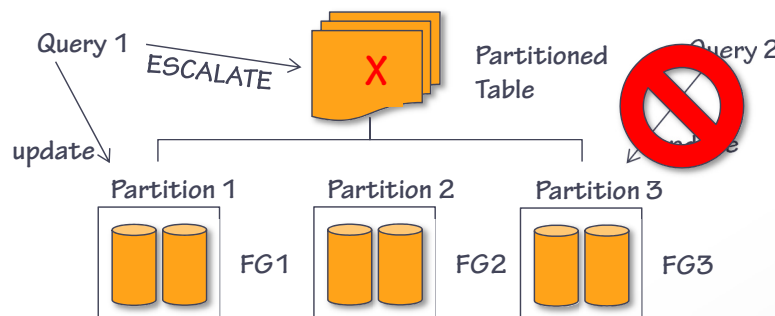


Slide 69

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Lock Escalation: The Problem

- Lock escalation on partitioned tables reduces concurrency as the table lock locks **ALL** partitions



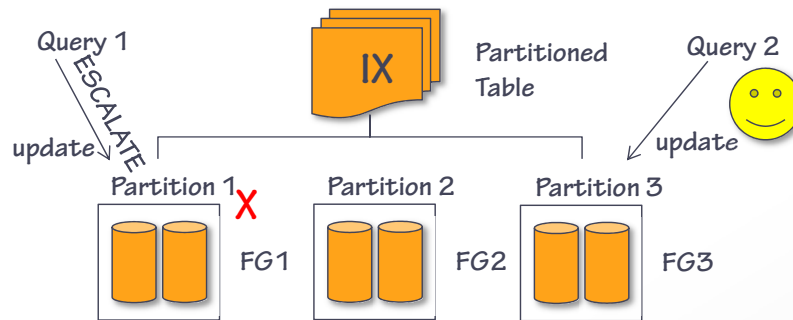
- Only way to solve this before 2008 is to disable escalation

Slide 70

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Lock Escalation: The Solution

- SQL Server 2008+ allows lock escalation to the partition level, allowing concurrent access to other partitions



- Escalation to partition level does not block queries on other partitions



Slide 71

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Options and Syntax

- Escalation setting is per-table, set with ALTER TABLE:
 - `ALTER TABLE mytable SET (LOCK_ESCALATION = {AUTO | TABLE | DISABLE})`
- AUTO:** Partition-level escalation if the table is partitioned
- TABLE:** Always table-level escalation
- DISABLE:** Don't escalate unless absolutely necessary
- How to tell what setting a table already has?
 - `SELECT lock_escalation_desc FROM sys.tables WHERE name = 'mytablename'`
- There is no tool (SSMS) support for ALTER TABLE



Slide 72

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Lock Escalation Summary

 hidden slide
w/extra details

- What is 'lock escalation'?

- When there are too many locks and SQL Server takes a lock higher in the hierarchy to save lock memory

Hierarchy

Database



Table



Page



Row

S

IX

IX

X

SQL Server 2005

At compilation/optimization
chooses row or page

At execution escalates:

Row -> table

Page -> table

Never Row->page->table

Why? Lock conversion too
expensive and they're
already out of resources...

SQL Server 2008+

At compilation/optimization
chooses row or page

At execution escalates:

Row -> partition/table

Page -> partition/table

By default, partition-level
lock escalation is off
(principle of least surprise)



Slide 73

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Lock Escalation: Monitoring

 hidden slide
w/extra details

- Locking can be monitored using the DMV `sys.dm_tran_locks`

- `SELECT *`
`FROM sys.dm_tran_locks`
`WHERE [resource_type] <> 'DATABASE'`

- Table level escalation will show:

	resource_type	resource_associated_entity_id	request_mode	request_type	request_status
1	OBJECT	2105058535	X	LOCK	GRANT

- Partition level escalation will show:

	resource_type	resource_associated_entity_id	request_mode	request_type	request_status
1	HLOBT	72057594039042048	X	LOCK	GRANT
2	OBJECT	2105058535	IX	LOCK	GRANT



Slide 74

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Locks – Granularity/Escalation



- **Granularity chosen at compilation**
 - Row/page or partition or table
- **Escalated at execution time if resources (to handle all of the smaller locks) are not available**
 - Row-to-table or page-to-table (or to partition if enabled on 2008+)
 - Locks do NOT escalate row-to-page-to-partition/table
 - Generally, lock escalation is desired as it results in less resources being needed to handle a query/transaction and often the query can complete faster, but at the expense of concurrency
 - SQL Server 2005+ has two trace flags, one allows lock escalation under memory pressure (1224), one does not (1211)
 - SQL Server 2008+ ONLY supports partition-level lock escalation when set at the table-level through ALTER TABLE



Slide 75

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Review

- Schema locks (object's schema)
- Schema locks (object)
- Relaxed FIFO
- Lock escalation



Slide 76

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Questions!



Module 5: Locking, Blocking, and Deadlocks (Intro)



Overview

- Blocking
- Lock Analysis
- Deadlocks
- Deadlock Analysis
- Deadlock Mitigation



Slide 79

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Availability, What About [B]locking?

- **ACID transaction design requirements**
 - Atomicity Consistency **Isolation** Durability
- **Isolation levels (session setting shown in sys.dm_exec_sessions)**
 - Read uncommitted (1)
 - Read committed (2)
 - Uses locking (when read_committed_snapshot is not set)
 - Uses versioning (when read_committed_snapshot is set)
 - Repeatable reads (3)
 - Serializable (4)
 - Snapshot (5)



Slide 80

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Locking Prevents Conflicts

- **No other transactions can invalidate the data set through modifications**
- **Is this always what you want or need?**
 - Queuing applications typically want locks, if someone is modifying that row – we want to go to another not see last transactional state
 - Very volatile prices – don't want to give them the last price if it's currently being updated, so wait... (but the update's fast)
- **What about long running transactions or cases where you don't need absolute current...**



Slide 81

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Tools for Troubleshooting Blocking

- **Dynamic Management Views**
- **System stored procedures**
 - `sp_lock [@@spid]`
- **SQL Profiler**
- **Performance Monitor counters**
- **PSS Script**
 - `sp_blocker_pss08` (KB#271509)
- **Blocked Process Report**
- **Pssdiag/SQLdiag (<http://diagmanager.codeplex.com/>)**
- **SQL Server 2008+ adds**
 - Performance Data Collection
 - Extended Events



Slide 82

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Detecting Blocking

- **SSMS Reports – Top Transactions by Blocking**
- **Using DMVs**
 - sys.dm_tran_locks and sys.dm_os_waiting_tasks
 - sys.dm_os_wait_stats
 - sys.dm_exec_requests
 - CROSS APPLY sys.dm_exec_sql_text(sql_handle)
 - Shows the last command submitted
- **Using system procedures**
 - sp_lock/sp_lock @@spid
 - sp_who2 (DMV-based rewrite of sp_who)
 - Shows whether blocked and by who (BlkBy column)
 - LastBatch to show you when the last command was submitted from the client
 - CPUTime and DiskIO to show relative activity for transactions blocked/blocking
- **Adam Machanic's sp_whoisactive**
 - Very comprehensive and free monitoring tool



Slide 83

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Detecting Blocking

- **SQL Server Blocked Process Threshold**
- **Set at an instance level**
 - `sp_configure 'blocked process threshold', n (secs)`
- **Each spid blocked for n seconds triggers an event that can be tracked using:**
 - Trace
 - Event Notification
 - WMI – SQL Agent Alert
 - See Jonathan Kehayias' blog post: Using the Block Process Report in SQL Server 2005/2008
 - <http://www.sqlservercentral.com/articles/Blocking/64474/>

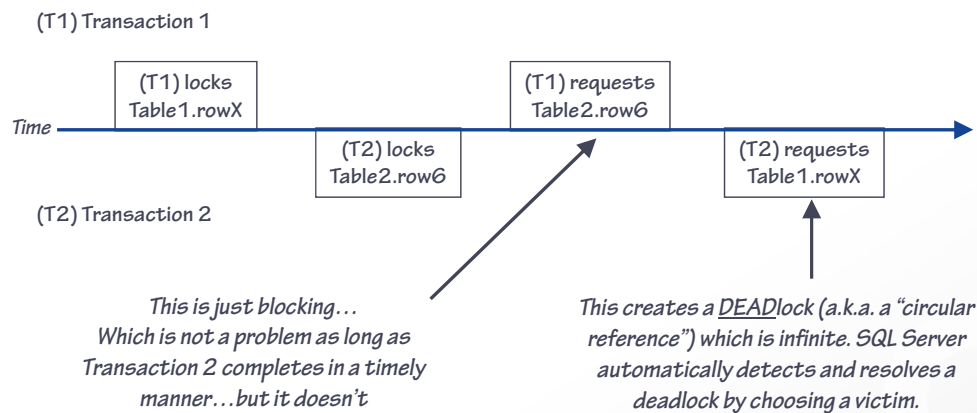


Slide 84

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

When Blocking Becomes Deadly ☺

- Two (or more) transactions request mutually desired resources in an undesirable order



Slide 85

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Deadlocks

- All resources have similar deadlock handling
- Deadlocks can be caused by
 - Data locks: locks of rows, pages, partitions, tables, and database metadata
 - Worker threads: queued tasks waiting for worker threads can cause deadlock ('intra-query parallelism deadlock')
 - Memory: two tasks waiting for memory
 - Parallel query execution resources
 - MARS interleaving



Slide 86

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Deadlock Monitor

- Background process in the lock manager
- Checks every 5 seconds usually, but time between checks throttles up and down based on how frequently deadlocks are being discovered
- If something enters the wait queue right after a deadlock is detected, a deadlock search will immediately begin
 - I.e. it is a pessimistic process
- If a deadlock is found, one of the participants is rolled back



Slide 87

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Deadlock Resolution

- By default, SQL Server chooses the least expensive victim to rollback (based on the amount of transaction log generated)
- If desired, you can impact this choice:
 - **SET DEADLOCK_PRIORITY**
 - LOW (equivalent of -5)
 - NORMAL (equivalent of 0)
 - HIGH (equivalent of 5)
 - Things like shrink, reorganize are set to low automatically
 - SQL Server 2005+ allows numeric value (-10 through 10)
 - Can also set with a variable (lookup in a table)
- **The deadlock victim receives error 1205:**
 - Your transaction (<tran id>) was deadlocked with another process and has been chosen the deadlock victim. Rerun your transaction.



Slide 88

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Detecting Deadlocks

- **Perfmon Counter: Deadlocks/Sec**
- **SQL Profiler events**
 - Select Locks/Deadlock Graph
 - Save all deadlock graphs to single file or one file per-deadlock
 - Saved in XML format with XDL extension
 - Displayed graphically in profiler
 - Or can be analyzed using XML APIs
- **Extended Events**
 - Writes deadlock graph
 - System health session tracks deadlocks
- **Trace flag 1222 (1204 and 1205)**
 - Writes deadlock info to SQL Server error log



Slide 89

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Homework: Resources

- **Online Training:**
 - See Jonathan Kehayias' course on Pluralsight:
[SQL Server: Deadlock Analysis and Prevention](#)
- **Online blog posts from Bart Duncan**
 - Search: "bart duncan deadlocks"
 - Deadlock Troubleshooting, Parts 1, 2 and 3
 - <http://tinyurl.com/clnwyyl> (Part 1)



Avoiding Deadlocks

- **Minimize blocking**
 - More effective queries
 - More effective indexes
 - Add or remove lock hints
- **Access resources in the same order**
- **SET DEADLOCK_PRIORITY** to lower value
- **Add retry logic in application if error 1205 returned**
- **Consider versioning; either statement-level or transaction-level**



Slide 91

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Review

- **Blocking**
- **Lock Analysis**
- **Deadlocks**
- **Deadlock Analysis**
- **Deadlock Mitigation**



Slide 92

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Questions!



Module 6: Versioning



Overview

- **Understanding Isolation**
 - Read Uncommitted
 - Read Committed with Locking
 - Read Committed with Versioning
 - Repeatable Reads
 - Serializable
 - Snapshot
- **Controlling isolation levels**
- **Statement-level read consistency**
- **Transaction-level read consistency**
- **Overhead/monitoring**
- **Isolation summary**



Slide 95

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Read Uncommitted

Phenomenon: Dirty Reads

- **A read transaction can read another transaction's uncommitted (or in-flight) changes – resulting in “dirty reads”**
- **DML statements always use exclusive locking**
- **SQL Server implementation:**
 - Row locks are not used (SCH_S locks are used) for the dirty read transaction and locks against data being accessed are not honored
- **Resulting phenomenon:**
 - Statements execute with the possibility of inaccurate data since the “in-flight” data read may continue to change or even be invalidated (e.g. rolled back)

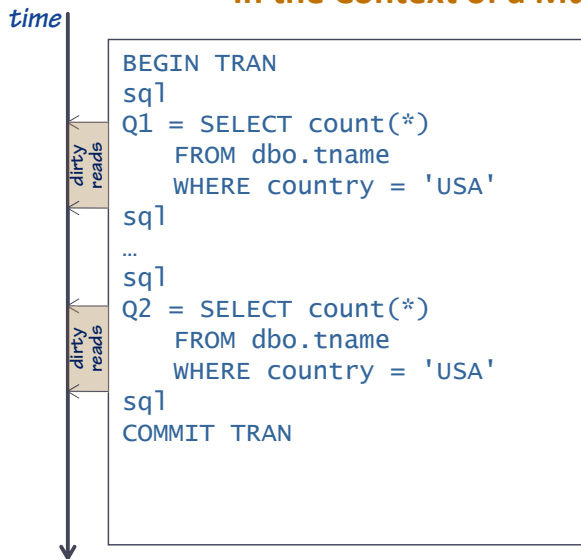


Slide 96

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Read Uncommitted

In the Context of a Multi-Statement Transaction



Read Uncommitted

- Q1 > Q2
- Q1 < Q2
- Q1 = Q2
- The data accessed for Q1 and Q2 is not guaranteed to be committed at the time the row is read
- Locks are neither acquired nor honored by Q1 or Q2 and therefore the data may change between the two as well as be in-flight (or dirty) during either read
- Because locks are not honored, the reader does not have to wait to read data that's in-flight
- Trading off accuracy for concurrency



Slide 97

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Read Committed (using Locking)

Phenomenon: Inconsistent Analysis

- Read committed “using locking” is the default behavior in ALL releases
- In-flight transaction’s data cannot be read by a read committed transaction – only committed changes are visible
- DML statements always use exclusive locking
- In implementation:
 - Locks are released (for readers – not writers) as resources are read, a row may be read more than once in some scenarios
- Resulting phenomenon:
 - Reads are not repeatable through the life of a transaction – as a result a row may not be read consistently during the life of a transaction
 - Don’t have a DEFINABLE point in time to which a query reconciles...

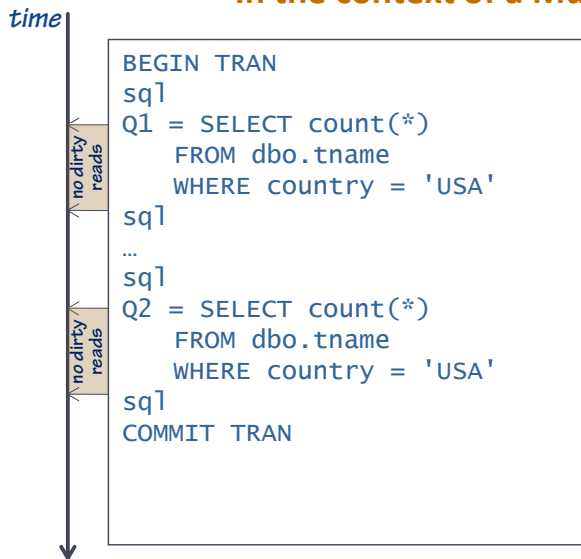


Slide 98

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Read Committed

In the context of a Multi-Statement Transaction



Read Committed

- Q1 > Q2
- Q1 < Q2
- Q1 = Q2
- The data accessed for Q1 and Q2 is guaranteed to be **ONLY** committed data
- Rows accessed during Q1 are *not* locked and therefore may be read inconsistently even in Q1 (non-repeatable reads)
- Because only committed data can be read, the reader may have to wait if a writer is modifying rows
- Allowing *some* accuracy while allowing *some* concurrency



Slide 99

<http://www.SQLskills.com>
 © SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Repeatable Read

Phenomenon: Phantoms

- In-flight transaction's data cannot be read and data modified is accessible only to the repeatable read transaction
- Data read, but not modified is accessible to other transactions for reads, but not DML
- In implementation:
 - Locks are held for the life of a transaction, rows which are read are locked and can be repeatably read during the life of a transaction
- Resulting phenomena:
 - Rows which were not present at the beginning of the transaction can appear – in the result

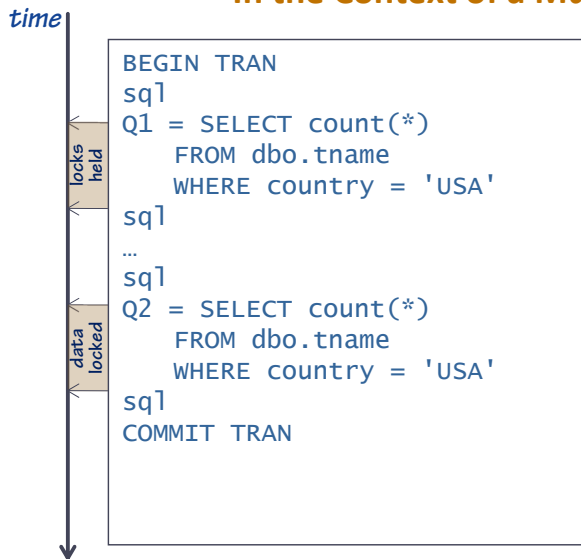


Slide 100

<http://www.SQLskills.com>
 © SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Repeatable Read

In the Context of a Multi-Statement Transaction



Repeatable Read

- $Q1 < Q2$
- $Q1 = Q2$
- As the rows in Q1 are read, the shared locks remain
- Only rows accessed by Q1 are locked; this does not prevent new rows from entering set (phantoms)
- Q2 will have at least the same number of rows as Q1
- Locked rows are not accessible to other transactions
- Increasing accuracy while reducing concurrency



Slide 101

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Serializable

Phenomenon: None

- In-flight transaction's data cannot be read and data modified is accessible only to the serializable transaction
- Data read, but not modified is accessible to other transactions for reads, but not DML
- In implementation:
 - Locks are held for the life of a transaction and held at higher levels within indexes to prevent rows from entering the "set"
- Implementation side effect:
 - To prevent rows from entering the "set" of data, the "set" of data needs to be locked
 - If appropriate indexes do not exist then higher levels of locking might be necessary (i.e. table-level locking)

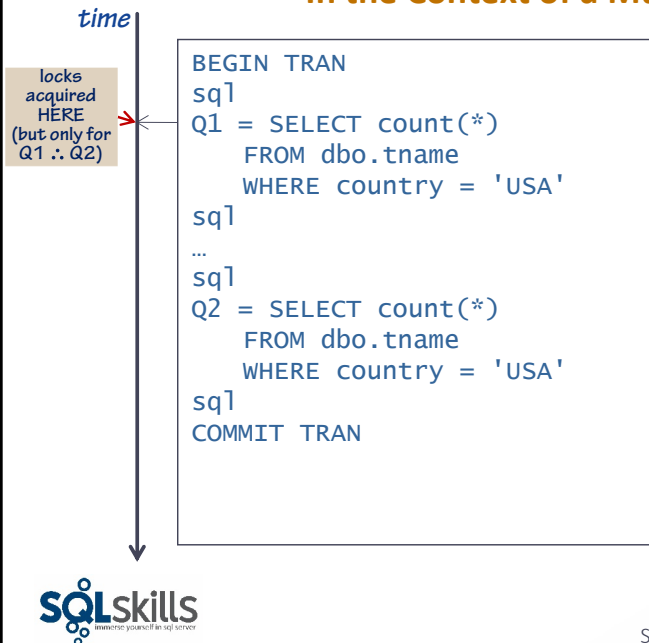


Slide 102

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Understanding Isolation Levels

In the Context of a Multi-Statement Transaction



Serializable

- Q1 = Q2
- Rows accessed by Q1 are locked and cannot change between Q1 and Q2
- Serializable protects the entire set and does not allow the data returned from Q1 to change by any other process – guaranteeing the state of the data
- Uses locks (indexes/tables) to guarantee consistency
- Locked rows are not accessible for modifications to other transactions – this creates the most blocking
- Absolute accuracy while reducing concurrency the most

Slide 103

<http://www.SQLskills.com>
 © SQLskills Online Immersion Event. All rights reserved.

Isolation in Implementation

- SQL Server 2005 added an optional row versioning-based isolation, in combination with the locking implementation, and now you can control isolation in essentially four different configurations by defining the point in time to which you want your statements to reconcile
- Two end results (and database options) for implementing row-level versioning:
 - Statement-level read consistency
 - “Read Committed Isolation Using Row Versioning” (often referred to as RCSI)
 - Database option: `read_committed_snapshot`
 - Transaction-level read consistency
 - “Snapshot Isolation”
 - Database option: `allow_snapshot_isolation`

Slide 104

<http://www.SQLskills.com>
 © SQLskills Online Immersion Event. All rights reserved.

Controlling Isolation Behavior

- Default behavior
- Statement-level read consistency OR
“Read Committed Isolation Using Row Versioning”

```
ALTER DATABASE <database_name>  
SET READ_COMMITTED_SNAPSHOT ON  
WITH ROLLBACK AFTER 5
```

- Transaction-level read consistency OR “Snapshot Isolation”

```
ALTER DATABASE <database_name>  
SET ALLOW_SNAPSHOT_ISOLATION ON
```

- Both database options can be turned on as well:

```
ALTER DATABASE <database_name> SET READ_COM...  
ALTER DATABASE <database_name> SET ALLOW_SNA...
```



Slide 105

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Versioning: Impact and Overhead

- Versioning overhead is the same in ALL three configurations:
 - Read_committed_snapshot
 - Allow_snapshot_isolation
 - Or, with both turned on (the versioning is the same with one or both on)
- Overhead in tempdb can vary
 - Always tied to the transactions that require the versions
 - Possible for “transaction-level” to require the versions to stick around longer
- Monitor with sys.dm_db_file_space_usage
 - Prior to SQL Server 2012 this DMV only worked for tempdb
 - RETURNS: database_id, file_id, filegroup_id, total_page_count, allocated_extent_page_count, unallocated_extent_page_count, version_store_reserved_page_count, user_object_reserved_page_count, internal_object_reserved_page_count, mixed_extent_page_count
 - In SQL Server 2012, this can be used in any database and returns nulls for version_store_reserved_page_count, user_object_reserved_page_count, and internal_object_reserved_page_count



Slide 106

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Read Committed (using Locking)

Default Behavior

- All phenomena, except dirty reads, are possible, even in the bounds of a single select query
- In volatile databases a long-running query may produce inconsistent results
 - Can increase isolation to remove phenomena
 - Increasing isolation requires locks to be held longer
 - This can create blocking



Slide 107

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Read Committed (using Versioning)

Database Changed to READ_COMMITTED_SNAPSHOT

- No phenomena are possible in the bounds of a single read committed statement
- Only used by statements that are in READ COMMITTED isolation
 - Statements cannot have lock hints
 - Statement-level lock / isolation hints override this
 - If your query uses NOLOCK then you do NOT use versions; you must remove lock hints
- In volatile databases, a multi-statement transaction may yield different results for subsequent access of the same data
- Each statement is consistent but only for the execution of that statement, not for the life of the transaction (if the transaction has multiple statements)
- Each time data is read by a new statement the latest version is used

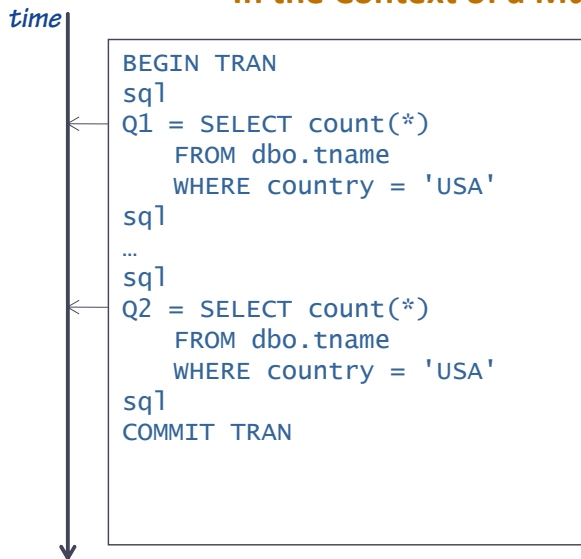


Slide 108

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Read Committed (using Versioning)

In the Context of a Multi-Statement Transaction



Statement-Level Read Consistency or Read Committed Versioning or RCSI = RC Snapshot Isolation

- Q1 > Q2
- Q1 < Q2
- Q1 = Q2
- Q1 and Q2 are both guaranteed to be accurate as of the beginning of the *statement* and the “version” of the row cannot change for the life of the *statement*
- RCSI guarantees the accuracy of the statement but the data can change and read differently by Q2



Slide 109

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Snapshot Isolation

Database Changed to ALLOW_SNAPSHOT_ISOLATION

- Setting **ALLOW**s users to ask for versioning by requesting snapshot isolation
 - NOT on by default
- **ALL** phenomena and **ALL** default locking behaviors are **EXACTLY** the same unless you explicitly ask for versioning through snapshot isolation
- Once requested, no phenomena are possible in the bounds of a transaction running under snapshot isolation
- In volatile databases, a multi-statement transaction will always see the transactionally accurate version which existed when the transaction started
- Versions must stick around longer
- Multi-statement transactions may have conflicts

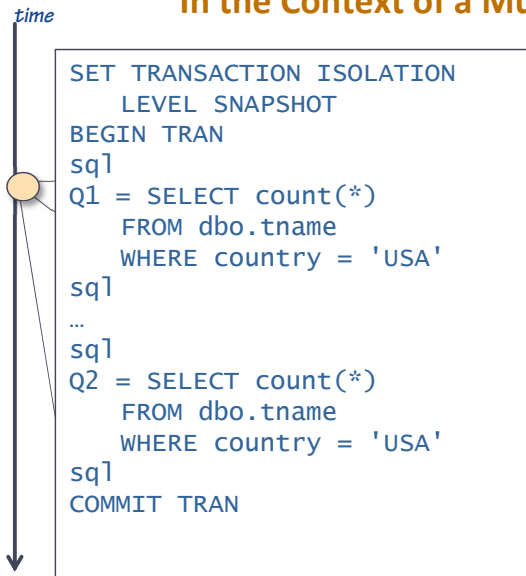


Slide 110

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Level: Snapshot Isolation

In the Context of a Multi-Statement Transaction



Transaction-Level Read Consistency or Snapshot Isolation

- Q1 = Q2
- Rows accessed by Q1 are consistent at start of transaction
- Data isolated at start of transaction and ensures that the transaction sees the same data at Q2, even if changed by other transactions
- Uses the row version store in tempdb
- Rows are accessible to other transactions



Slide 111

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Read Consistency for Multiple Statements

- Imagine running 5 queries (5 large sales reports)
- To what point in time do you want all of the reports to reconcile?
 - The point in time the statement starts
 - The point in time the transaction starts

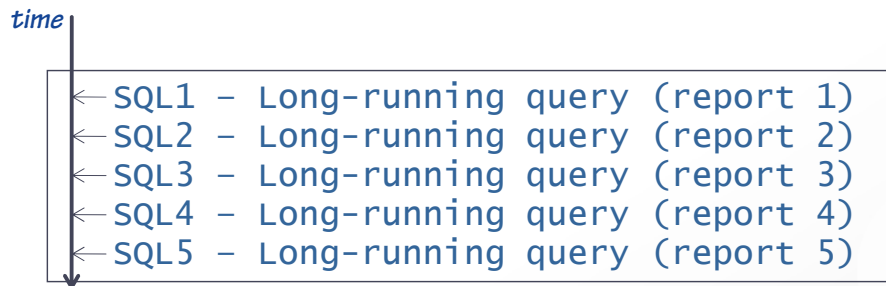


Slide 112

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Statement-Level Read Consistency

- Set the `read_committed_snapshot` database option
- Do nothing else...
- Each query reconciles to the point in time at which it starts



Slide 113

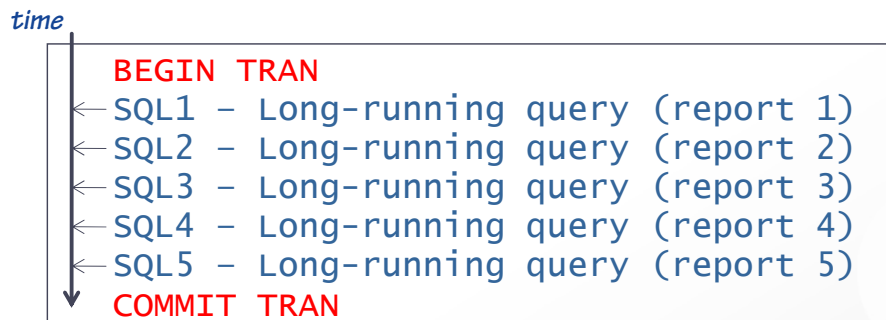
<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Statement-Level Read Consistency

- Set the `read_committed_snapshot` database option
- Do nothing else...
- Each query reconciles to the point in time at which it starts

EVEN IN A TRANSACTION

All statements reconcile to the point in time that they started



Slide 114

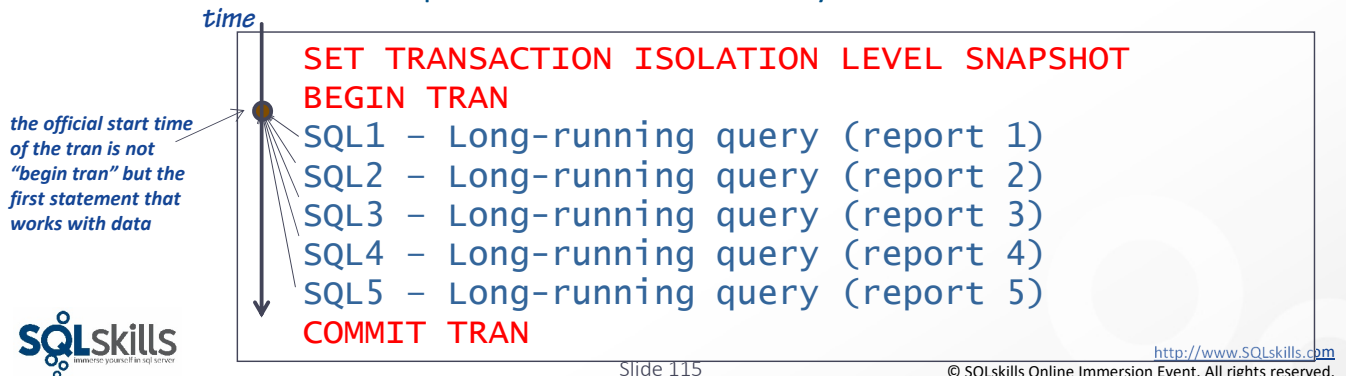
<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Transaction-Level Read Consistency (1 of 2)

- Set the allow_snapshot_isolation database option
- Request snapshot isolation (set transaction isolation...)
- All queries reconcile to the point in time the transaction starts (the first real statement)

REQUIRES THE EXPLICIT TRANSACTION (BEGIN/COMMIT) DEFINITION

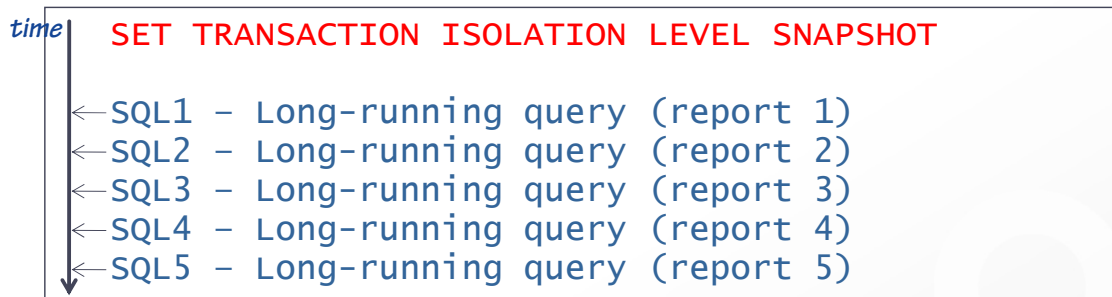
All statements reconcile to the point in time that the transaction officially starts



Transaction-Level Read Consistency (2 of 2) NEW SLIDE

- Set the allow_snapshot_isolation database option
- Request snapshot isolation (set transaction isolation...)
- All queries reconcile to the point in time the transaction starts
- But... you didn't say BEGIN TRAN

All statements reconcile to the point in time that the STATEMENT starts!



Read Consistency

Without the Request for Snapshot Isolation

- Set the `allow_snapshot_isolation` database option ON
- **WHAT IF YOU FORGET** to request snapshot isolation
 - If the database option: `read_committed_snapshot` is ON then, all queries reconcile to the point in time the statement starts
 - If the database option: `read_committed_snapshot` is OFF and you FORGET to ask for snapshot isolation then, then all queries use **read committed with locking**

time

```

SET TRANSACTION ISOLATION LEVEL SNAPSHOT
BEGIN TRAN
SQL1 - Long-running query (report 1)
SQL2 - Long-running query (report 2)
SQL3 - Long-running query (report 3)
SQL4 - Long-running query (report 4)
SQL5 - Long-running query (report 5)
COMMIT TRAN
  
```



Slide 117

<http://www.SQLskills.com>
 © SQLskills Online Immersion Event. All rights reserved.

Allowing Read Committed Using Statement-Level Snapshot

- Database option


```
ALTER DATABASE <database_name>
SET READ_COMMITTED_SNAPSHOT ON
WITH ROLLBACK AFTER 5
```
- No other changes necessary...
- If you use **READ COMMITTED** – no changes to your queries or your applications:
 - If you “hint” with lock hints (`NOLOCK`, `READUNCOMMITTED`, `SERIALIZABLE`, etc.) then you must remove these hints
 - If you depend on locking – re: queues, readers to wait for change then you might need to use `READCOMMITTEDLOCK`
- Changes to blocking... might be extreme!
- However, if this is **NOT** your performance problem (meaning concurrency isn’t your bottleneck) then you may hinder performance
- Expect this change in behavior at a cost (10-15%)



Slide 118

<http://www.SQLskills.com>
 © SQLskills Online Immersion Event. All rights reserved.

Allowing Snapshot Isolation

- **Database option**
`ALTER DATABASE <database_name>
SET ALLOW_SNAPSHOT_ISOLATION ON`
- **Session setting**
`SET TRANSACTION ISOLATION LEVEL SNAPSHOT`
- **Changes to applications:**
 - Request snapshot isolation
 - Make sure your transactions have error handling (use TRY...CATCH)
 - Test for conflict detection (mandatory = you don't have a choice, SQL will error/kill the 2nd updater)
- **Expect this change in behavior at a higher cost**



Slide 119

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Row Length Changes

- **Cost in row overhead**
 - When version is needed, 14 bytes added to row
 - When indexes are rebuilt, **versions are removed** (may need FILLFACTOR to reduce possible fragmentation)
- **If versioning, do you depend on locking?**
 - OK, most of you will say no but if you use queues...
 - Tip: Use `READCOMMITTEDLOCK` hint for queues that require locks
- **If transaction-level is needed and you have multiple writers (with snapshot isolation), you can have conflicts?**
 - Be sure to have error handling/conflict detection, see Snapshot Isolation whitepaper for details and examples



Slide 120

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Management/Monitoring

- Version store in tempdb
- Versions removed when no longer needed
- Read committed using statement-level snapshot won't hold versions as long, in theory, because only statement-level
- Snapshot isolation may have more impact on tempdb as versions held for life of transaction
- Lots of long-running transactions may stress tempdb
- See whitepaper for examples of queries that can help you monitor the version store through DMVs



Slide 121

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation: Key Points (1 of 2)

- Consider turning on `allow_snapshot_isolation` to determine OLTP overhead and monitor version-store activity (without changing any other behaviors)
- Most common to use statement-level read consistency / RCSI / “Read Committed Isolation Using Row Versioning”
 - No problems with update conflicts because every statement uses versions that are applicable to that statement's starting time
 - Little to no code changes required for statements currently using read committed with locking; this database option seamlessly makes those statements switch to using versioning (only statements with hints need to be changed)
- Often, `allow_snapshot_isolation` is also ON but more limited in use
 - Only use “snapshot isolation” for multi-statement reporting “transactions”
 - `SET TRANSACTION ISOLATION LEVEL SNAPSHOT` must be added to the code...
 - NO update conflicts if you're not setting transaction isolation level snapshot with transactions that make modifications



Slide 122

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation: Key Points (2 of 2)

- **Using locking**
 - You are always trading off between accuracy and concurrency
- **Using versioning**
 - You increase concurrency and accuracy with overhead in tempdb
- **If you are trying to do real-time reporting in an OLTP environment**
 - You will get inconsistencies/inaccuracies/anomalies in your long running reads unless you increase locks (which creates more and more blocking)
 - You should consider versioning
 - You'll want to make sure you optimize tempdb
 - Whitepaper: Working with tempdb in SQL Server 2005
 - <http://technet.microsoft.com/en-us/library/cc966545.aspx>
 - You'll want to make sure you understand versioning
 - Whitepaper: SQL Server 2005 Row Versioning-Based Transaction Isolation
 - <http://msdn.microsoft.com/en-us/library/ms345124.aspx>



Slide 123

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Review

- **Understanding Isolation**
 - Read Uncommitted
 - Read Committed with Locking
 - Read Committed with Versioning
 - Repeatable Reads
 - Serializable
 - Snapshot
- **Controlling isolation levels**
- **Statement-level read consistency**
- **Transaction-level read consistency**
- **Overhead/monitoring**
- **Isolation summary**



Slide 124

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Levels: Quick Review (1 of 2)



- **(1): Read uncommitted**
 - “Dirty reads” – an option ONLY for readers
 - Any data (even that which is in-flight/locked) can be viewed
- **(2): Read committed using locking (default)**
 - Only committed changes are visible
 - Data in an intermediate state cannot be accessed
 - The point-in-time to which your statement reconciles is not guaranteed, only the state of each row as it’s accessed is guaranteed (committed rows only)
- **(2): Read committed using versioning (read_committed_snapshot)**
 - Statement-level read consistency
 - New non-blocking, non-locking (i.e. SCH_S), version-based statements (in read committed ONLY)
 - The point-in-time to which your statement reconciles is the time it began



Slide 125

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Isolation Levels: Quick Review (2 of 2)



- **(3): Repeatable reads (uses locking)**
 - All reads are consistent for the life of a transaction
 - Shared locks are NOT released after the data is processed – does not allow writers (does allow other readers)
 - Does not protect entire set (phantoms may occur)
- **(4): Serializable (uses locking)**
 - All reads are consistent (from the time the resource is locked) for the life of the transaction
 - Avoids phantoms – no new records
 - The point-in-time to which your transaction reconciles is not guaranteed, only the state of each set (from when it’s first accessed) is guaranteed
- **(5): Snapshot isolation uses versioning (db: allow_snapshot_isolation)**
 - Transaction-level consistency using versioning
 - Non-blocking, non-locking, version-based transactions
 - The point-in-time to which your transaction reconciles is the time the transaction began (this must be requested with SET TRANSACTION ISOLATION SNAPSHOT)



Slide 126

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

“What If” Scenarios



- **Scenario 1: Code changes for transaction-level read consistency made but the database option `allow_snapshot_isolation` gets turned off**
 - Client application code:

```
SET TRANSACTION ISOLATION LEVEL SNAPSHOT
<executes without error>
BEGIN TRAN
<executes without error>
SELECT * FROM [dbo].[member];
```

Msg 3952, Level 16, State 1, Line 5
Snapshot isolation transaction failed accessing database 'Credit'
because snapshot isolation is not allowed in this database. Use ALTER
DATABASE to allow snapshot isolation.



Slide 127

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

“What If” Scenarios



- **Scenario 2: Cross-database transactions with databases of different versioning configurations**
 - DB1 – `read_committed_snapshot` + `allow_snapshot_isolation`
 - DB2 – neither is enabled
 - Client application code:

```
USE [DB1];
GO
SET TRANSACTION ISOLATION LEVEL SNAPSHOT;
GO
BEGIN TRAN
SELECT * FROM [DB1].[schema].[table]; <no problem>
SELECT / UPDATE / ANYTHING FROM [DB2].[schema].[table];
```

Msg 3952, Level 16, State 1, Line 5
Snapshot isolation transaction failed accessing database 'Credit'
because snapshot isolation is not allowed in this database. Use ALTER
DATABASE to allow snapshot isolation.



Slide 128

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

“What If” Scenarios



■ Scenario 3: Cross-database transactions with databases of different versioning configurations

- DB1 – read_committed_snapshot + allow_snapshot_isolation
- DB2 – neither is enabled
- Client application code:

```
USE [DB1];  
GO  
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
GO  
BEGIN TRAN  
SELECT * FROM [DB1].[schema].[table]; <no problem>  
SELECT / UPDATE / ANYTHING FROM [DB2].[schema].[table];
```
- NO PROBLEMS!
 - Reads in DB1 are read-committed using versions (no waits)
 - Reads in DB2 are read-committed using locking (possible [indefinite] waits)



Slide 129

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Questions!



Module 7: Review



What to do next?

- **Lots of content...**
 - Reviewing sooner rather than later will help!
 - Using a mix of mediums will help you see things in different ways and likely for some of you one will be better than another – multiple will be the way to make it stick!
- **A possibly review strategy?**
 - Print the slides + the whiteboard
 - Watch the course more interactively while taking notes and playing with demos
 - Small chunks – a module every day or two for a couple of weeks
 - Go through the recommended videos
 - Offer some “short courses” or “brown bag” lunch sessions where you deliver 30-45 minute topics to team members

If you want to learn something, read about it. If you want to understand something, write about it. If you want to master something, teach it.
– Yogi Bhajan
 - Always keep learning!



Slide 132

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

What to do next?

■ Where do you start?

- Find where your server is “waiting most” (see Paul’s Wait Statistics resources)
 - Blog post: Wait statistics, or please tell me where it hurts (<https://bit.ly/2wsQHQE>)
 - Pluralsight course: SQL Server: Performance Troubleshooting Using Wait Statistics (<https://bit.ly/2CzpnV2>)
- Do some “root cause analysis” to see why / where you’re waiting
 - Is it poorly written code?
 - Can you see any solutions to make the transactions more optimized / shorter
 - Is it poorly optimized indexes / structures?
 - Can you work on performance / tuning
 - Is it just the nature of the workload (meaning optimized code / fast + efficient transactions AND well designed / optimized indexes)?
 - Can you consider versioning
 - How optimal is tempdb?



Slide 133

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Self-paced Study and Exercises (1 of 2)

■ Some additional locking videos

- Video 01 - Locking + 02 Locking demo (MCM Readiness)
 - Lots of overlap but you might hear something differently
- Video 03 - Snapshot Isolation
 - Recorded when we were still using SI instead of Versioning but the content is still useful
- Video 04 - Paul’s Logging and Recovery course on Pluralsight
 - This will give you greater depth into logging and recovery and when log writes actually occur

■ Videos for filtered statistics and indexing

- Video 05 - Skewed Data, Poor Cardinality Estimates, and Plans Gone Bad (PASS 2013)
 - VLTs and filtered statistics solutions (a lot of helpful code for automation)
- Video 06 - SQL Server Indexing for Performance (PASS 2015)
 - Good overview of indexing, versions, and some tips/tricks (cool demo)
- Video 06a - SQL Server Indexing for Performance (Pluralsight)
 - 7+ hours on index internals and indexing strategies



Slide 134

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Self-paced Study and Exercises (2 of 2)

- **Statement Execution and the Plan Cache**
 - Pluralsight: SQL Server: Optimizing Ad Hoc Statement Performance
 - <https://www.pluralsight.com/courses/sqlserver-optimizing-adhoc-statement-performance>
- **Stored Procedure Performance**
 - Start with the blog post: BLOG Building High Performance Stored Procedures
 - Video 07 - Building High Performance Stored Procedures (PASS 2014)
 - Pluralsight: SQL Server: Optimizing Stored Procedure Performance
 - <https://www.pluralsight.com/courses/sqlserver-optimizing-stored-procedure-performance>
 - Pluralsight: SQL Server: Optimizing Stored Procedure Performance - Part 2
 - <https://www.pluralsight.com/courses/sqlserver-optimizing-stored-procedure-performance-part2>



Slide 135

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

THANK YOU!

