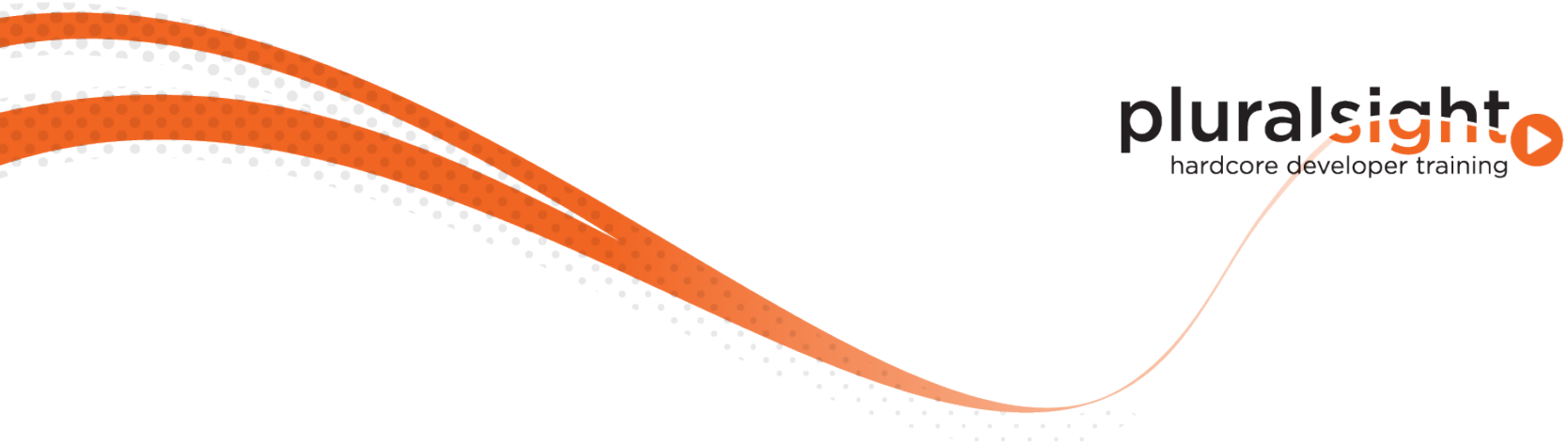
# SQL Server: Deadlock Analysis and Prevention

## Module 2: Locking Overview

Jonathan M. Kehayias
Jonathan@SQLskills.com



pluralsight
hardcore developer training

# Introduction

- **This module provides a brief overview of locking in SQL Server to provide the necessary background to troubleshoot deadlocks**
  - Full details are beyond the scope of this course, but will be covered in the upcoming course on SQL Server Transactions, Locking, and Isolation Levels
- **Locks are used by the Database Engine to synchronize access to a resource by multiple concurrent transactions**
- **Understanding how locking occurs in SQL Server is critical for being able to properly analyze and prevent deadlocks**

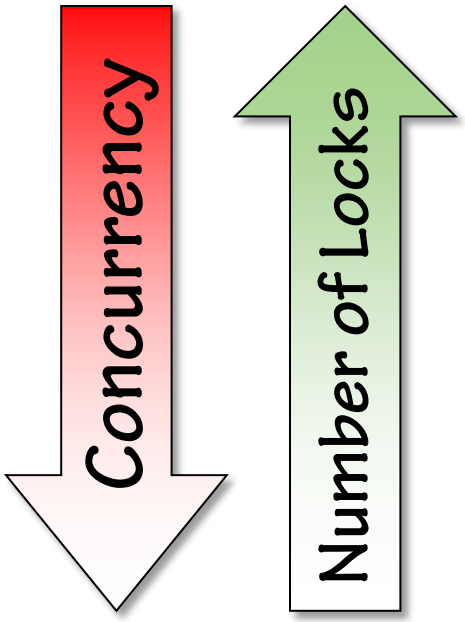
- **In this module we'll cover:**
  - Lock resource hierarchy
  - Lock modes
  - Lock compatibility
  - Isolation levels and locking hints
  - Lock escalation

# Terminology

- Transaction – a unit of work performed within the database
- Lock - the synchronization mechanism on a resource that protects changes amongst multiple concurrent transactions
- Lock mode - defines the level of access that other transactions have while the resource is locked
- Blocking - when a transaction requests a lock mode that conflicts with a currently held lock and has to wait for that lock to be released
- Deadlock - when two transactions block each other trying to acquire locks on resources the other transaction holds in a conflicting mode

# Lock Granularity

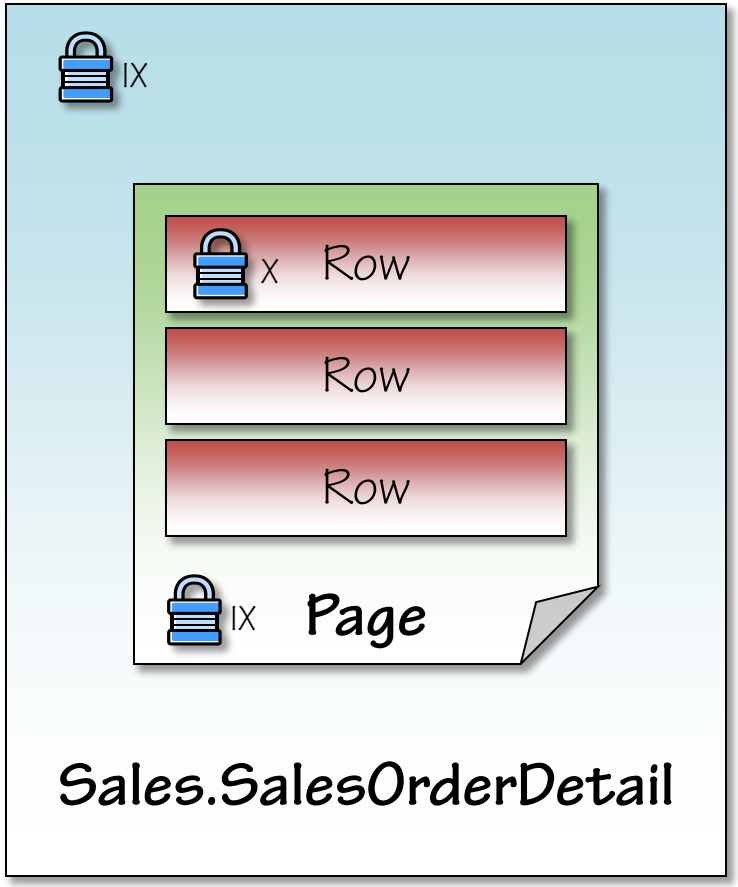
- **RID/KEY – a single row is locked**
  - RID – row identifier for a single row in a heap
  - KEY – index key for a single row in a index
- **PAGE - a single page in the database is locked**
- **HoBT – a heap or B-tree (index) partition is locked**
- **TABLE – the entire table is locked**
- **METADATA - the table schema definition is locked**




- **Locks are acquired at multiple levels of granularity to fully protect the lowest-level resource**
- **Locks are always acquired 'top-down', from the table level down to individual rows**
  - This forms the lock hierarchy

# Lock Hierarchy

- DELETE statement begins executing that affects one row in a table
- A intent exclusive (IX) lock is acquired for the table
- A intent exclusive (IX) lock is acquired for the page containing the row
- A exclusive (X) lock is acquired for the row being modified



# Shared and Update Locks

- **Shared (S) – acquired for read operations that do not modify data**
  - No transaction can modify data while the shared lock exists
  - Concurrent SELECT operations can read the data
  - Locks are released as soon as the read operation completes unless the isolation level is repeatable read or higher or hints are used
- **Update (U) – acquired for resources that will be updated**
  - Only one transaction can acquire an update (U) lock at a time
  - Prevents deadlocks caused by lock conversions from a shared (S) lock to an exclusive (X) lock
  - The update (U) lock is converted to an exclusive (X) lock to modify the data
  - Concurrent shared (S) locks are allowed

# Exclusive and Intent Locks

- **Exclusive (X) – acquired for data modifications**
  - Prevents access to a resource from concurrent transactions
  - Ensures that multiple changes cannot be made to the same resource at the same time
  - Reads can only occur using NOLOCK or read uncommitted, read committed snapshot, or snapshot isolation levels
- **Intent (I) – acquired to establish the lock hierarchy**
  - Acquired on higher-level resources to protect locks on lower-level resources
    - E.g. a table-level intent exclusive (IX) lock is required before a page-level exclusive (X) or intent exclusive (IX) lock can be acquired
  - Prevents transactions from modifying/locking the higher-level resource with an incompatible lock for the lower-level lock being acquired
  - Intent shared (IS), intent exclusive (IX), shared with intent exclusive (SIX), intent update (IU), shared intent update (SIU) and update intent exclusive (UIX)

# Schema and Key-Range Locks

- **Schema (Sch) – acquired to protect or modify an object's definition**
  - Schema modification (Sch-M) locks are acquired during DDL operations to prevent access to the object while its definition is changed
  - Schema stability (Sch-S) locks are acquired when compiling and executing queries to prevent modification of the object definition by DDL operations
- **Key-range - acquired to protect a range of rows when using the serializable isolation level**
  - Prevents other transactions from inserting rows into the range to ensure that the data returned by a query would remain the same if it were run again within the transaction

# Lock Compatibility

- If a resource is already locked when a transaction requests a lock on it, the new lock can only be acquired if it is compatible with the existing lock on the resource
- The most common locks are shown here but a full compatibility matrix is available in Books Online (http://bit.ly/SQLLockCompat)

| | | Existing lock mode | | | | | |
|---|---|---|---|---|---|---|---|
| Requested mode | | IS | S | U | IX | SIX | X |
| | Intent shared (IS) | Yes | Yes | Yes | Yes | Yes | No |
| | Shared (S) | Yes | Yes | Yes | No | No | No |
| | Update (U) | Yes | Yes | No | No | No | No |
| | Intent exclusive (IX) | Yes | No | No | Yes | No | No |
| | Shared with intent exclusive (SIX) | Yes | No | No | No | No | No |
| | Exclusive (X) | No | No | No | No | No | No |

# Isolation Levels and Locking Hints

- **SQL Server provides five different transaction isolation levels**
  - Read Uncommitted, Read Committed, Repeatable Read, Snapshot, Serializable
- **Locking hints can be specified for individual tables referenced in DML statements**
  - HOLDLOCK, NOLOCK, NOWAIT, PAGLOCK, READCOMMITTED, READCOMMITTEDLOCK, READPAST, READUNCOMMITTED, REPEATABLEREAD, ROWLOCK, SERIALIZABLE, TABLOCK, TABLOCKX, UPDLOCK, XLOCK
- **The isolation levels and locking hints change the locking behavior of transactions**
  - Full details are beyond the scope of this course, but will be covered in the upcoming course on SQL Server Transactions, Locking, and Isolation Levels
  - More details can be found in Books Online: (http://bit.ly/TranIsoLevs)

# Lock Escalation

- **The lock granularity is chosen during query compilation**
  - Row, page, partition, or table
- **During query execution, the lock granularity may be escalated if the resources necessary for the lower-level locks are not available**
  - Lock escalation can occur from row-to-table or page-to-table
    - If partition-level escalation is enabled in SQL Server 2008 onwards, escalation can be row-to-partition or page-to-partition
  - Locks NEVER escalate row-to-page-to-partition/table
- **Lock escalation can be desirable under some circumstances, but the higher-level locking reduces concurrency and can lead to blocking**

# Summary

- **Locking is an integral part of relational database management systems and protects data during concurrent activity in the database**
- **Locks can be acquired at different levels of granularity and in different modes to control concurrent data access**
- **Blocking occurs when a transaction attempts to acquire a lock with a mode that conflicts with a lock that is held by another transaction**
- **Deadlocks occur when two transactions block each other trying to acquire locks on resources the other transaction holds in a conflicting mode**

- **The next module will look at:**
  - Deadlock detection