SQLskills Immersion Event

IEPTO2: Performance Tuning and Optimization

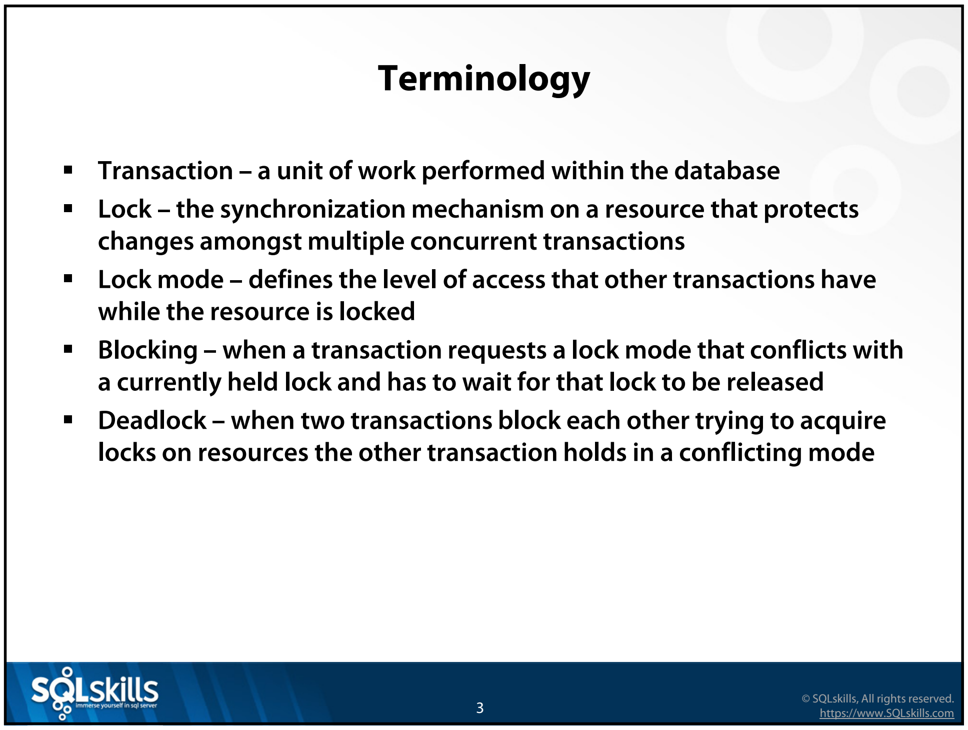
If time permits –

Module 11: Deadlock Analysis

Jonathan Kehayias
Jonathan@SQLskills.com

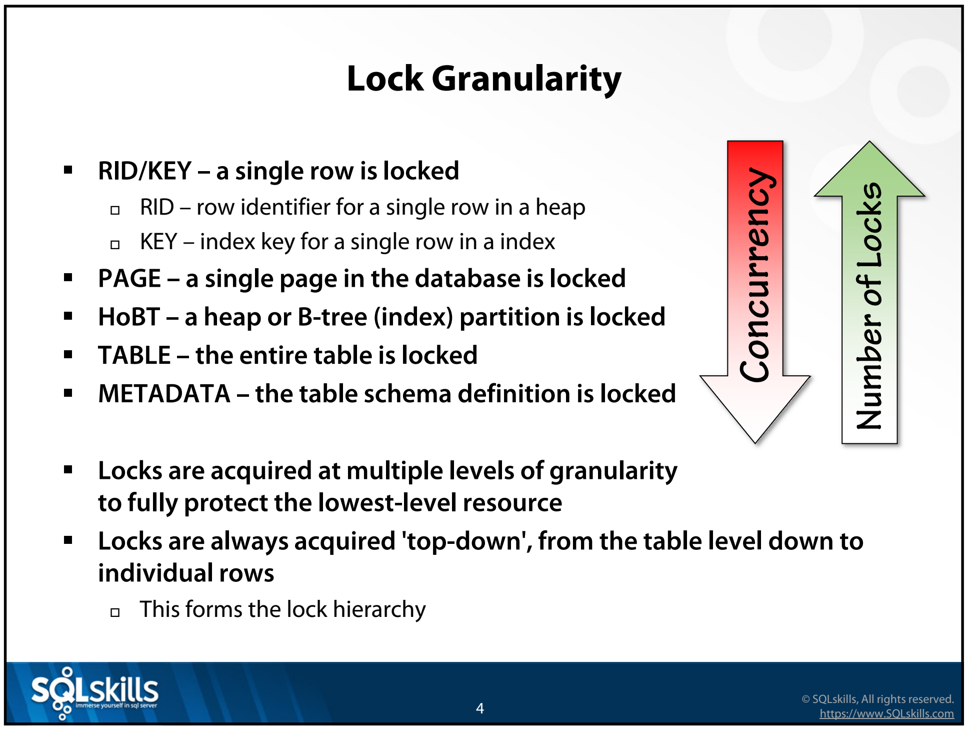
Overview

- Review of locking in SQL Server
- What is a deadlock (reminder)
- Collecting deadlock graphs
- Anatomy of a deadlock
- Reading deadlock graphs
- Resolving deadlocks



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

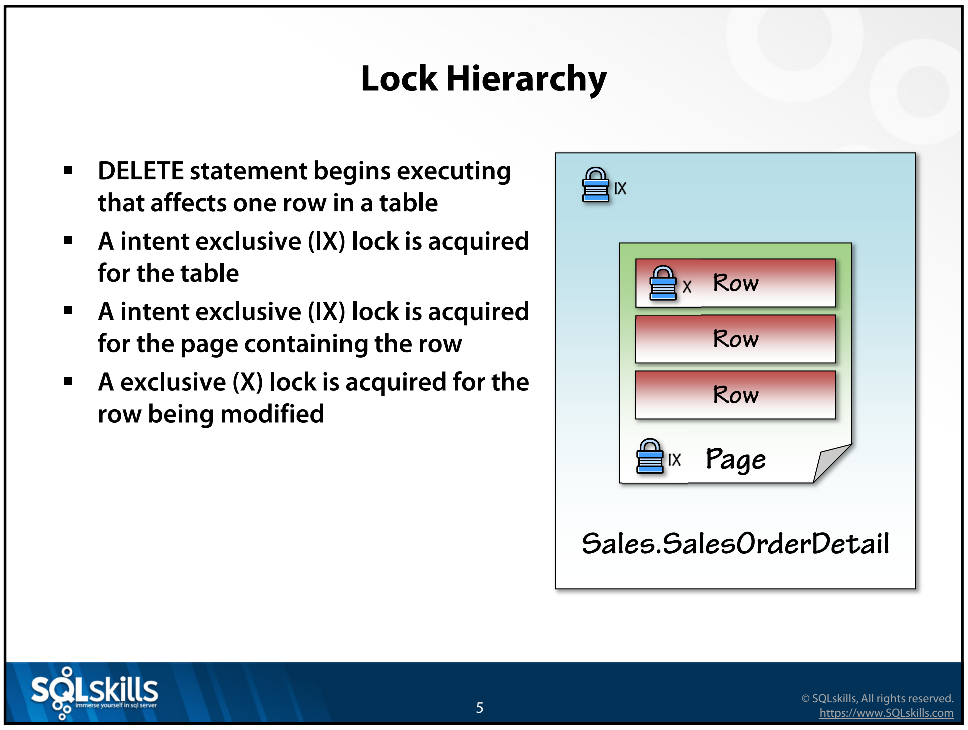


Lock Granularity

- 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

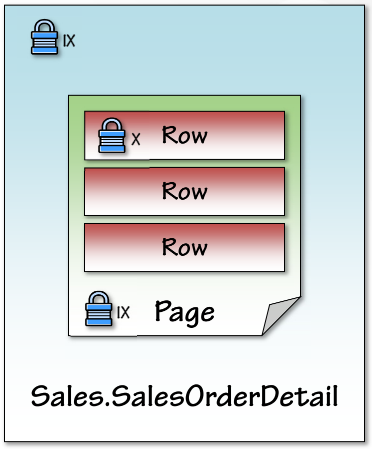
Concurrency
Number of Locks

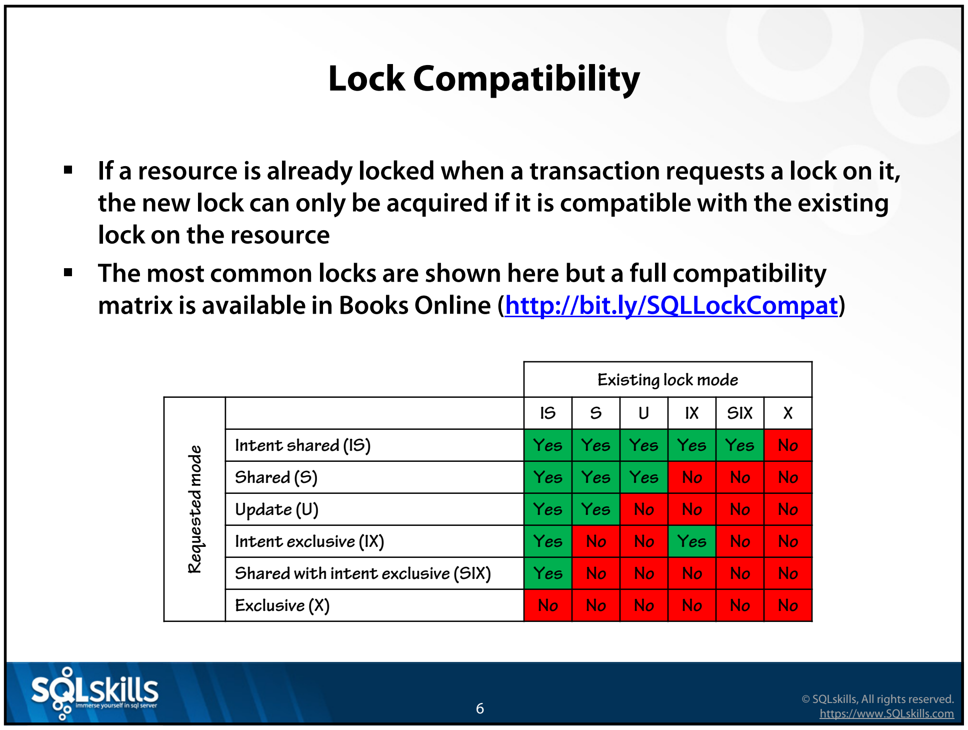
- Locks are acquired at multiple levels of granularity to fully protect the lowest-level resource
- Locks are always acquired 'top-down', from the table level down to individual rows
  - This forms the lock hierarchy



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





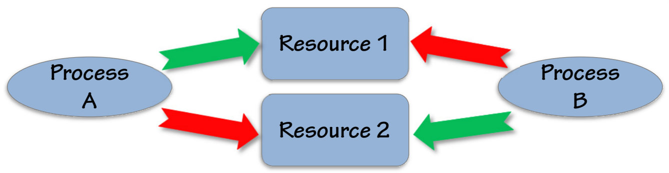
Lock Compatibility

- 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					
		IS	S	U	IX	SIX	X
Requested mode	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

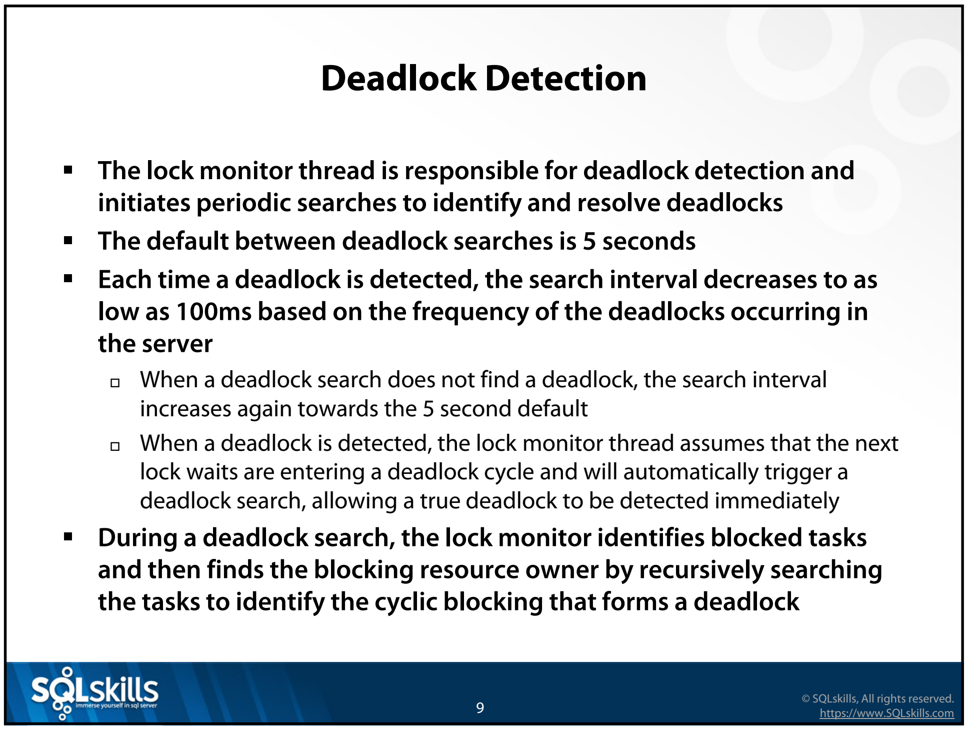
What is a Deadlock

- A condition when two or more processes are holding locks on resources and are each waiting for the other to release it's locks to progress, or where two or more processes are waiting for locks on resources in a circular chain



Misconceptions

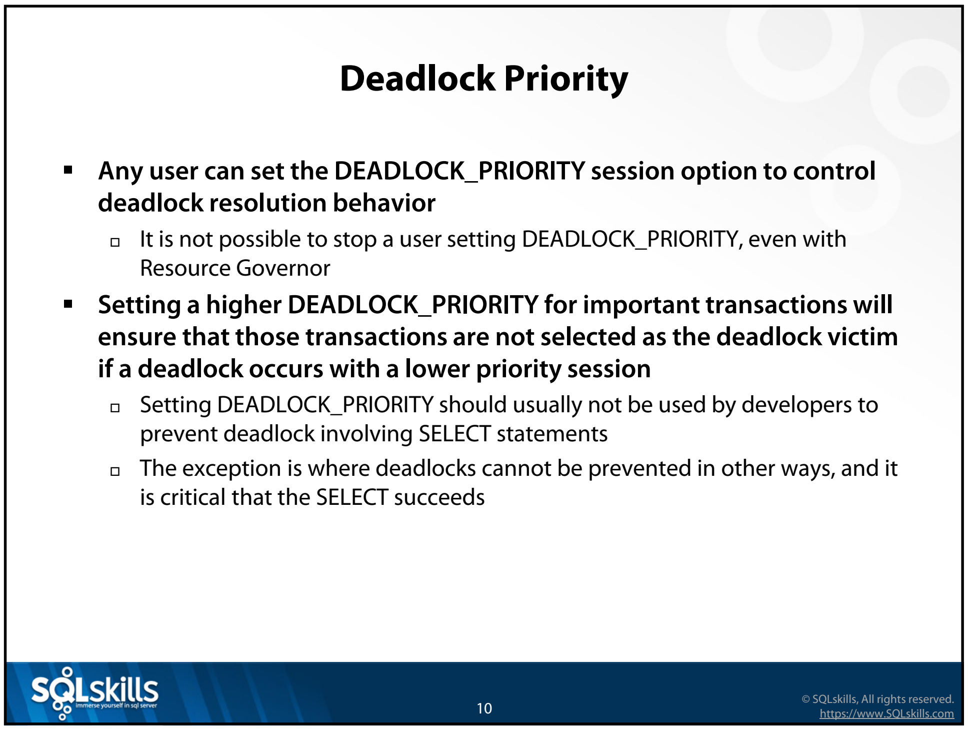
- Deadlocks are a bug in SQL Server
- Deadlocks cannot be prevented
- Using NOLOCK on all SELECT statements is the best way to prevent deadlocks from occurring
- Adding covering indexes for every type of query will prevent deadlocks from occurring
- Troubleshooting deadlocks is a complex task that requires an experienced SQL Server developer or administrator



Deadlock Detection

- The lock monitor thread is responsible for deadlock detection and initiates periodic searches to identify and resolve deadlocks
- The default between deadlock searches is 5 seconds
- Each time a deadlock is detected, the search interval decreases to as low as 100ms based on the frequency of the deadlocks occurring in the server
  - When a deadlock search does not find a deadlock, the search interval increases again towards the 5 second default
  - When a deadlock is detected, the lock monitor thread assumes that the next lock waits are entering a deadlock cycle and will automatically trigger a deadlock search, allowing a true deadlock to be detected immediately
- During a deadlock search, the lock monitor identifies blocked tasks and then finds the blocking resource owner by recursively searching the tasks to identify the cyclic blocking that forms a deadlock

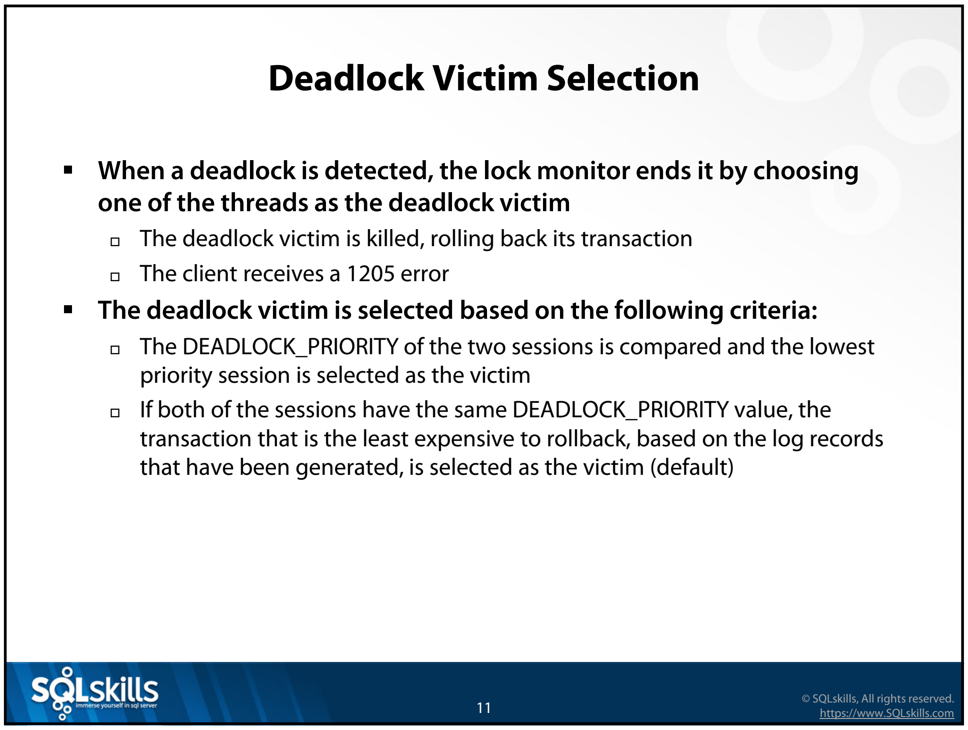
9



Deadlock Priority

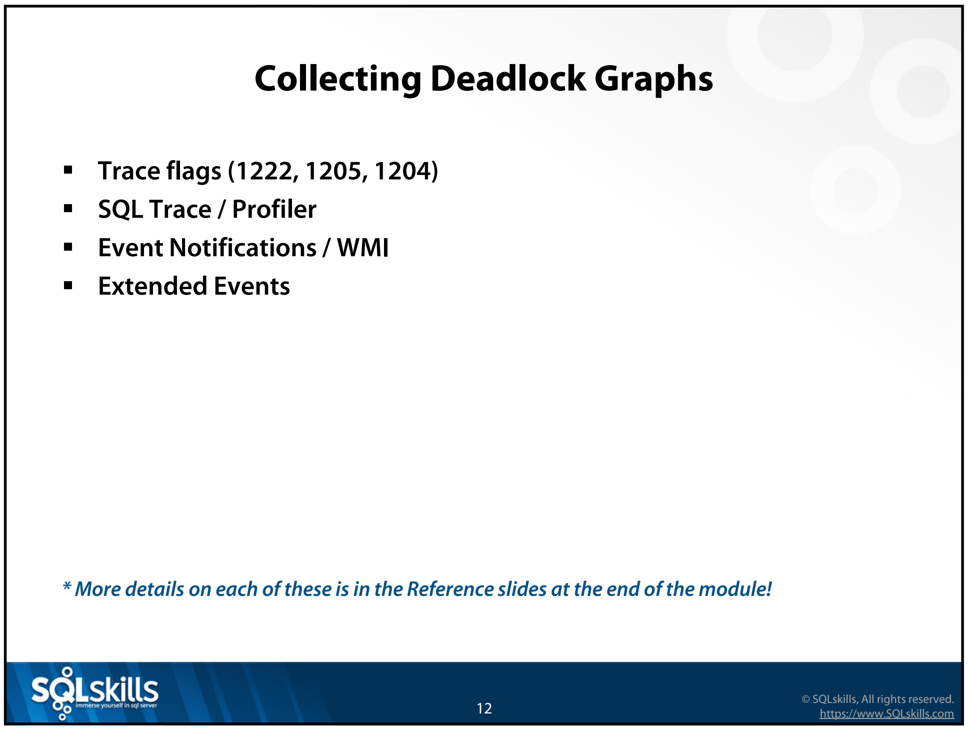
- Any user can set the DEADLOCK_PRIORITY session option to control deadlock resolution behavior
  - It is not possible to stop a user setting DEADLOCK_PRIORITY, even with Resource Governor
- Setting a higher DEADLOCK_PRIORITY for important transactions will ensure that those transactions are not selected as the deadlock victim if a deadlock occurs with a lower priority session
  - Setting DEADLOCK_PRIORITY should usually not be used by developers to prevent deadlock involving SELECT statements
  - The exception is where deadlocks cannot be prevented in other ways, and it is critical that the SELECT succeeds

10



Deadlock Victim Selection

- **When a deadlock is detected, the lock monitor ends it by choosing one of the threads as the deadlock victim**
  - The deadlock victim is killed, rolling back its transaction
  - The client receives a 1205 error
- **The deadlock victim is selected based on the following criteria:**
  - The DEADLOCK_PRIORITY of the two sessions is compared and the lowest priority session is selected as the victim
  - If both of the sessions have the same DEADLOCK_PRIORITY value, the transaction that is the least expensive to rollback, based on the log records that have been generated, is selected as the victim (default)



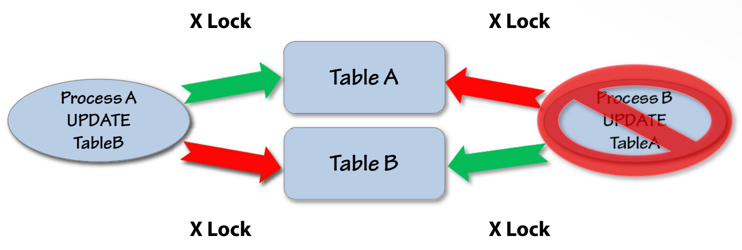
Collecting Deadlock Graphs

- **Trace flags (1222, 1205, 1204)**
- **SQL Trace / Profiler**
- **Event Notifications / WMI**
- **Extended Events**

**** More details on each of these is in the Reference slides at the end of the module!***

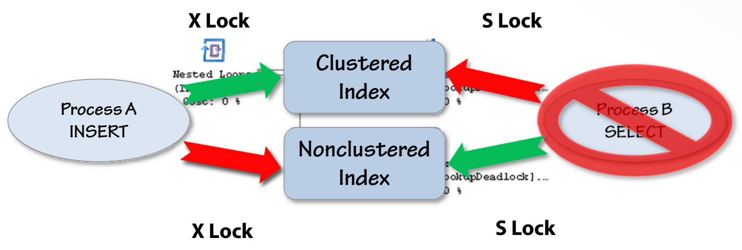
Reverse Object Order Deadlocks

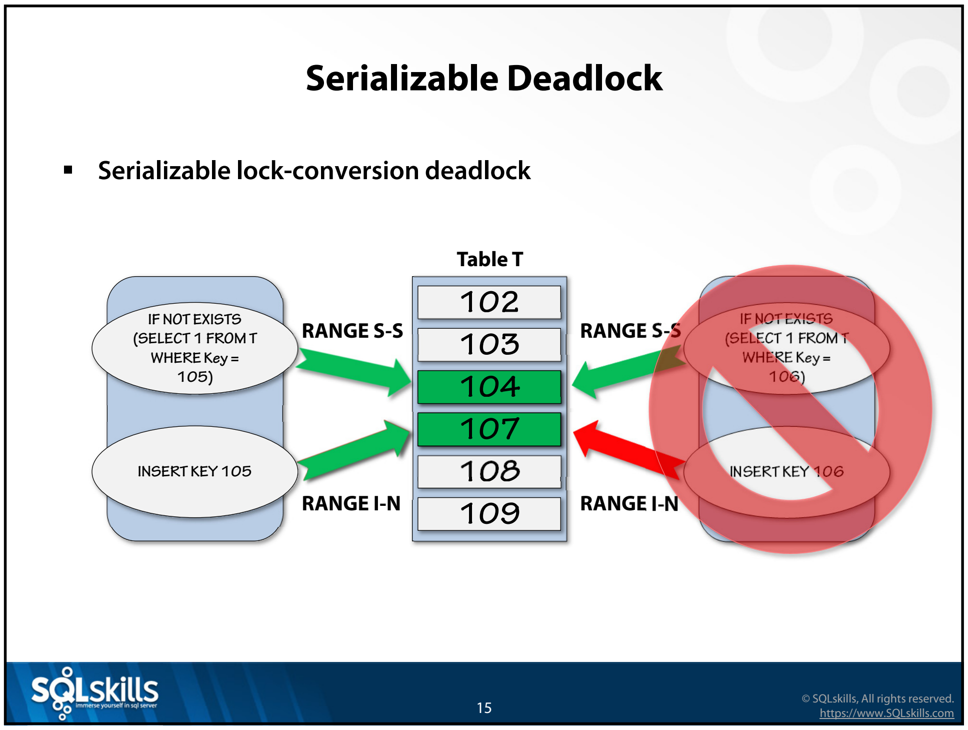
- Two processes access objects in reverse order



Bookmark Lookup Deadlock

- Bookmark lookup deadlock





Serializable Deadlock

- Serializable lock-conversion deadlock

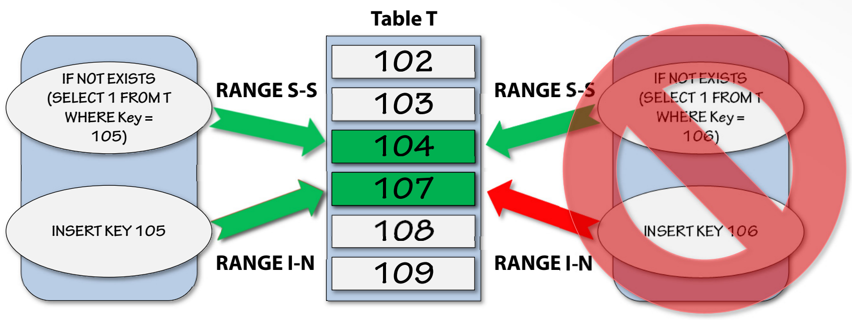
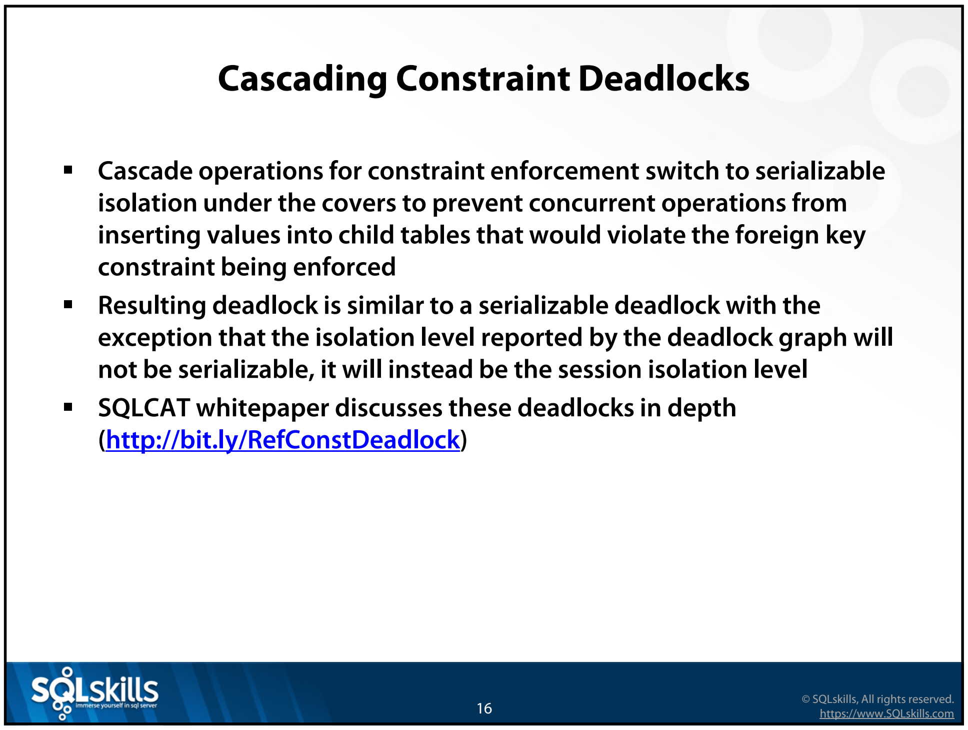


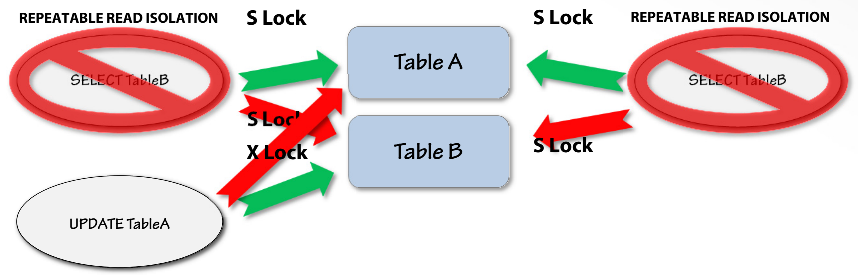
Table T
IF NOT EXISTS (SELECT 1 FROM T WHERE Key = 105)
INSERT KEY 105
RANGE S-S
RANGE I-N
102
103
104
107
108
109
RANGE S-S
RANGE I-N
IF NOT EXISTS (SELECT 1 FROM T WHERE Key = 106)
INSERT KEY 106



Cascading Constraint Deadlocks

- Cascade operations for constraint enforcement switch to serializable isolation under the covers to prevent concurrent operations from inserting values into child tables that would violate the foreign key constraint being enforced
- Resulting deadlock is similar to a serializable deadlock with the exception that the isolation level reported by the deadlock graph will not be serializable, it will instead be the session isolation level
- SQLCAT whitepaper discusses these deadlocks in depth (http://bit.ly/RefConstDeadlock)

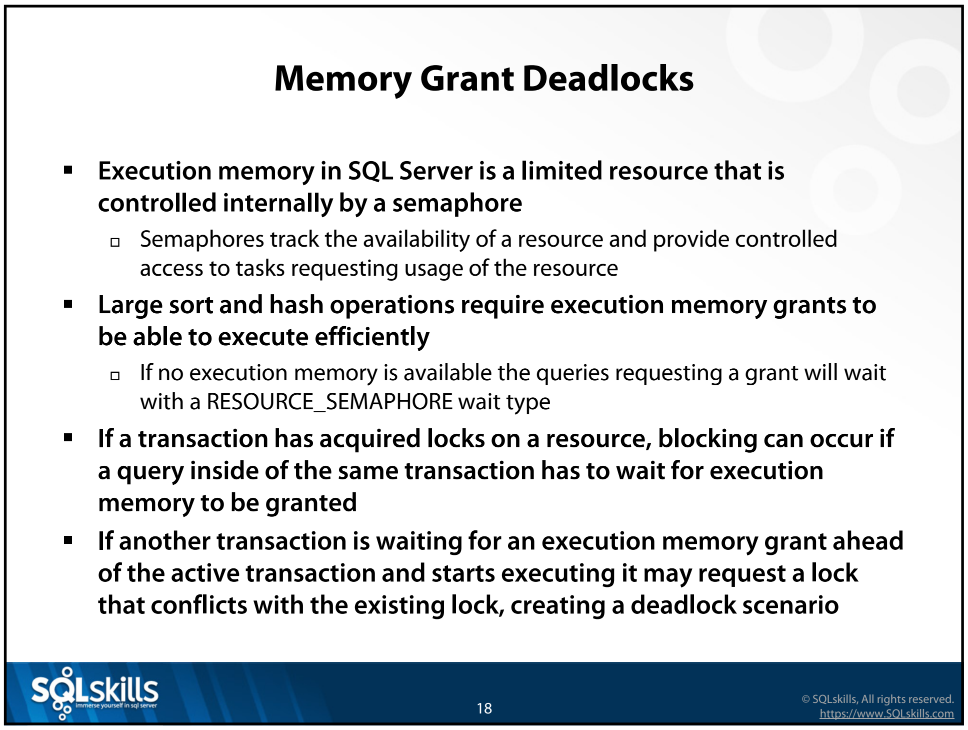
Multi-victim Deadlocks



SQLskills immerse yourself in sql server

17

© SQLskills, All rights reserved.
https://www.SQLskills.com



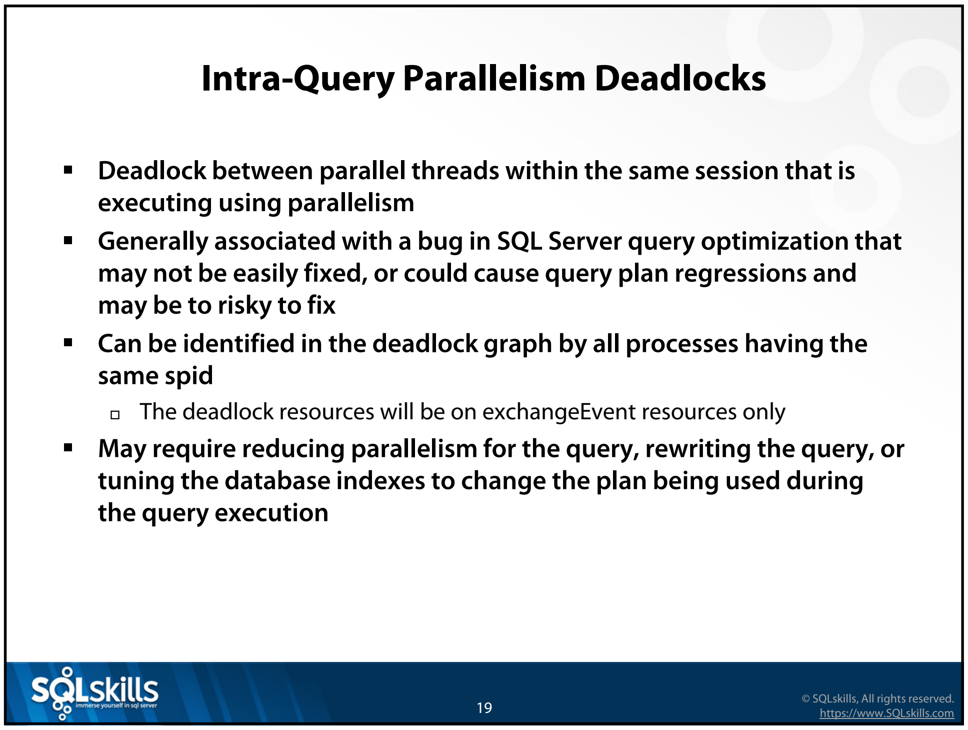
Memory Grant Deadlocks

- **Execution memory in SQL Server is a limited resource that is controlled internally by a semaphore**
  - Semaphores track the availability of a resource and provide controlled access to tasks requesting usage of the resource
- **Large sort and hash operations require execution memory grants to be able to execute efficiently**
  - If no execution memory is available the queries requesting a grant will wait with a RESOURCE_SEMAPHORE wait type
- **If a transaction has acquired locks on a resource, blocking can occur if a query inside of the same transaction has to wait for execution memory to be granted**
- **If another transaction is waiting for an execution memory grant ahead of the active transaction and starts executing it may request a lock that conflicts with the existing lock, creating a deadlock scenario**

SQLskills immerse yourself in sql server

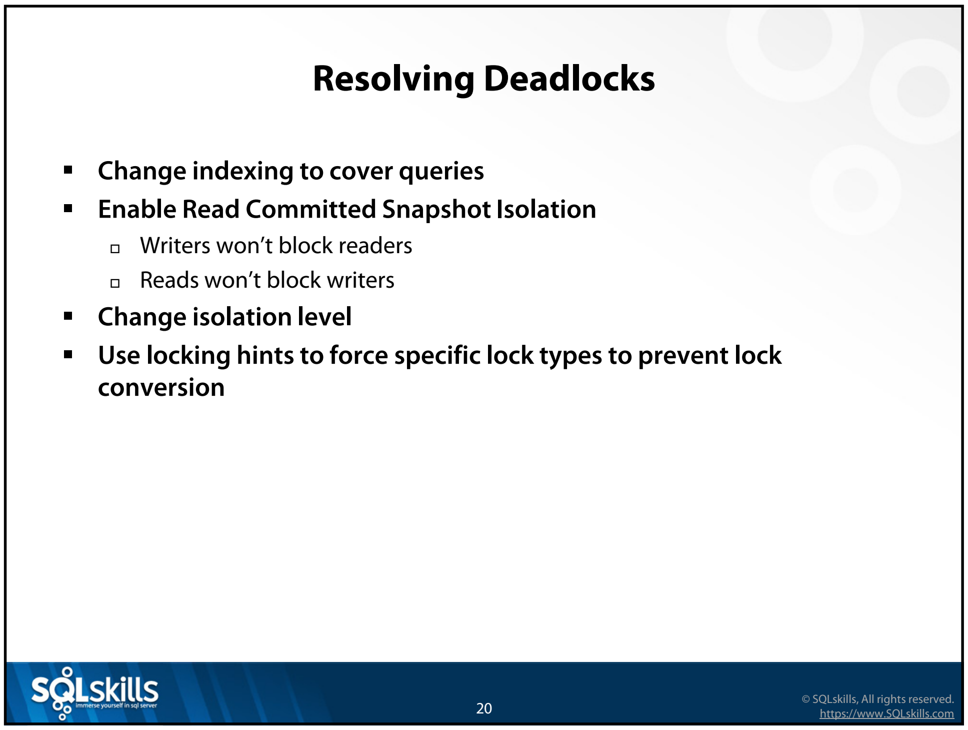
18

© SQLskills, All rights reserved.
https://www.SQLskills.com



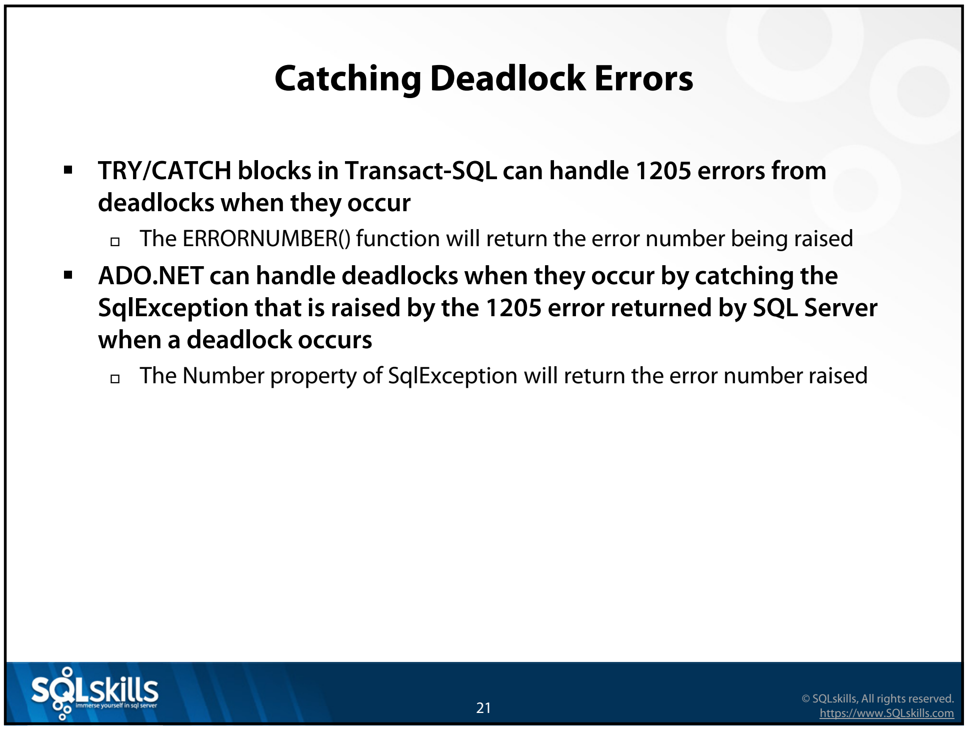
Intra-Query Parallelism Deadlocks

- Deadlock between parallel threads within the same session that is executing using parallelism
- Generally associated with a bug in SQL Server query optimization that may not be easily fixed, or could cause query plan regressions and may be to risky to fix
- Can be identified in the deadlock graph by all processes having the same spid
  - The deadlock resources will be on exchangeEvent resources only
- May require reducing parallelism for the query, rewriting the query, or tuning the database indexes to change the plan being used during the query execution



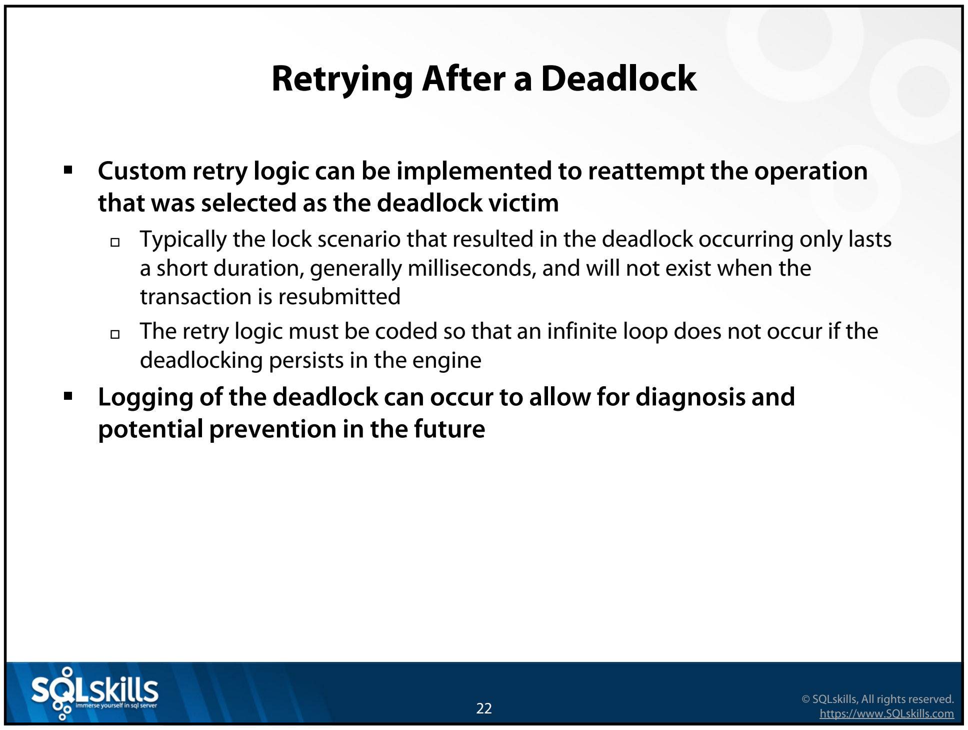
Resolving Deadlocks

- Change indexing to cover queries
- Enable Read Committed Snapshot Isolation
  - Writers won't block readers
  - Reads won't block writers
- Change isolation level
- Use locking hints to force specific lock types to prevent lock conversion



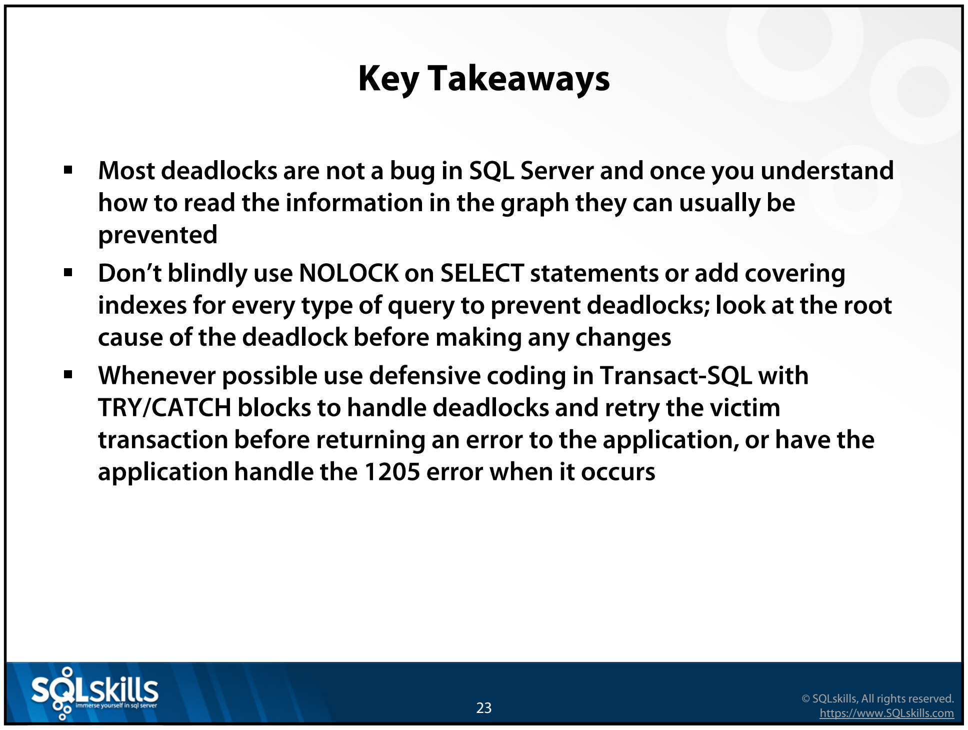
Catching Deadlock Errors

- **TRY/CATCH blocks in Transact-SQL can handle 1205 errors from deadlocks when they occur**
  - The ERRORNUMBER() function will return the error number being raised
- **ADO.NET can handle deadlocks when they occur by catching the SqlException that is raised by the 1205 error returned by SQL Server when a deadlock occurs**
  - The Number property of SqlException will return the error number raised



Retrying After a Deadlock

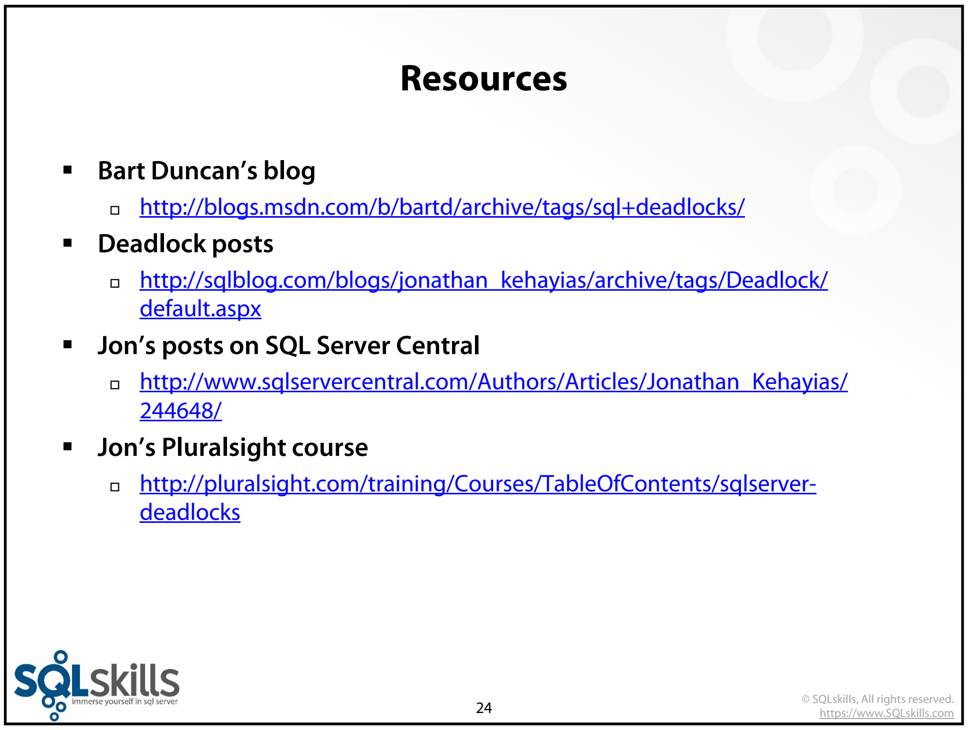
- **Custom retry logic can be implemented to reattempt the operation that was selected as the deadlock victim**
  - Typically the lock scenario that resulted in the deadlock occurring only lasts a short duration, generally milliseconds, and will not exist when the transaction is resubmitted
  - The retry logic must be coded so that an infinite loop does not occur if the deadlocking persists in the engine
- **Logging of the deadlock can occur to allow for diagnosis and potential prevention in the future**



Key Takeaways

- Most deadlocks are not a bug in SQL Server and once you understand how to read the information in the graph they can usually be prevented
- Don't blindly use NOLOCK on SELECT statements or add covering indexes for every type of query to prevent deadlocks; look at the root cause of the deadlock before making any changes
- Whenever possible use defensive coding in Transact-SQL with TRY/CATCH blocks to handle deadlocks and retry the victim transaction before returning an error to the application, or have the application handle the 1205 error when it occurs

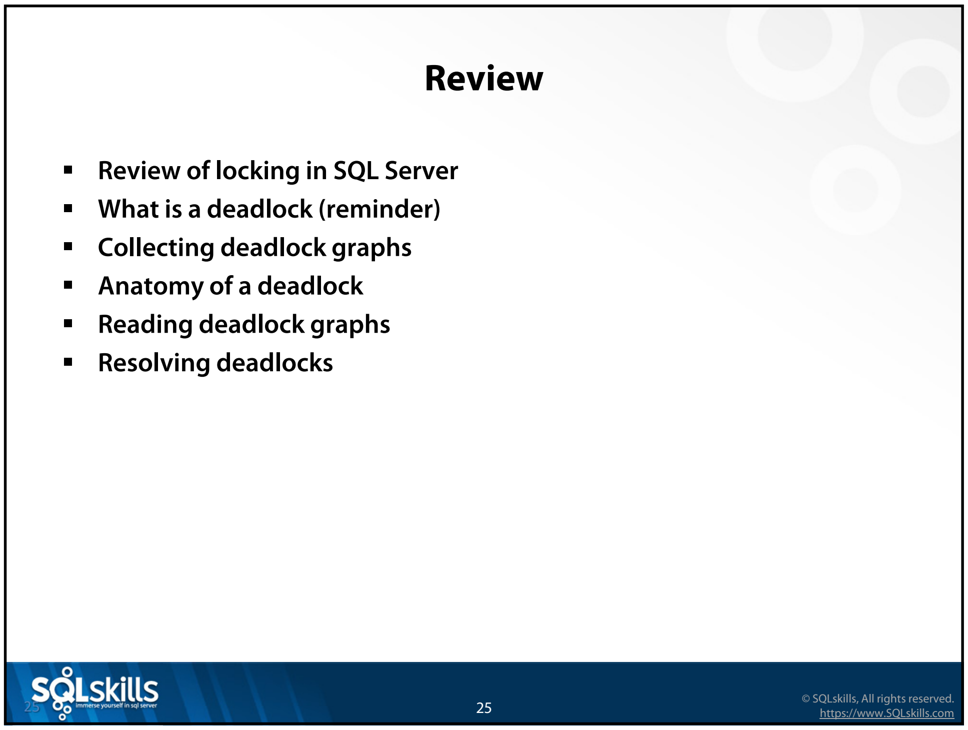
23



Resources

- Bart Duncan's blog
  - http://blogs.msdn.com/b/bartd/archive/tags/sql+deadlocks/
- Deadlock posts
  - http://sqlblog.com/blogs/jonathan_kehayias/archive/tags/Deadlock/default.aspx
- Jon's posts on SQL Server Central
  - http://www.sqlservercentral.com/Authors/Articles/Jonathan_Kehayias/244648/
- Jon's Pluralsight course
  - http://pluralsight.com/training/Courses/TableOfContents/sqlserver-deadlocks

24

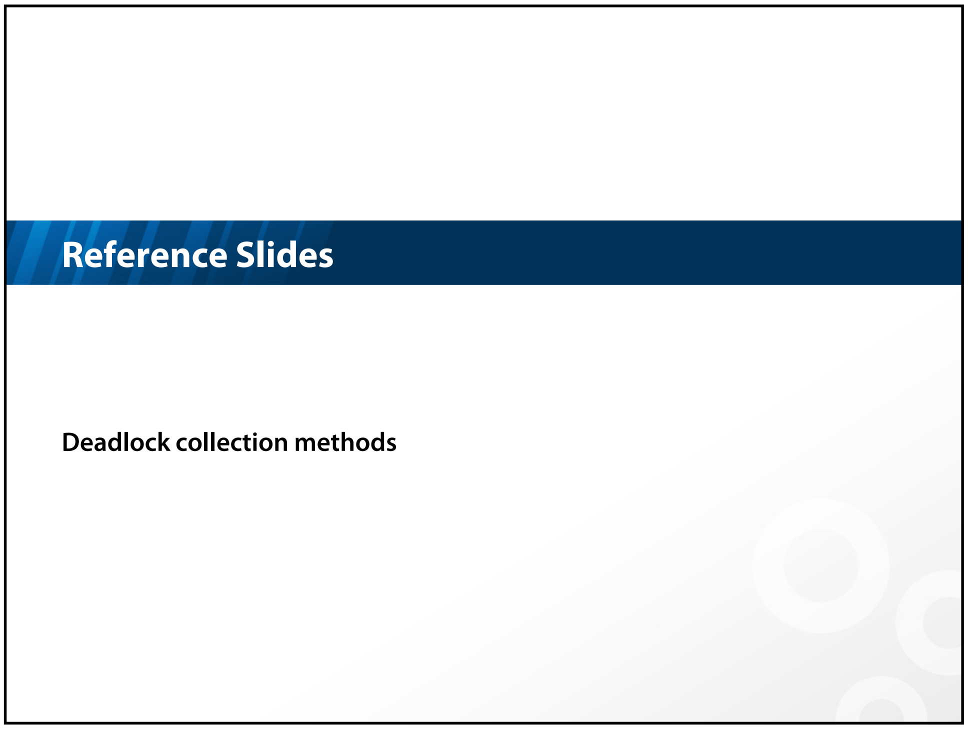


Review

- Review of locking in SQL Server
- What is a deadlock (reminder)
- Collecting deadlock graphs
- Anatomy of a deadlock
- Reading deadlock graphs
- Resolving deadlocks

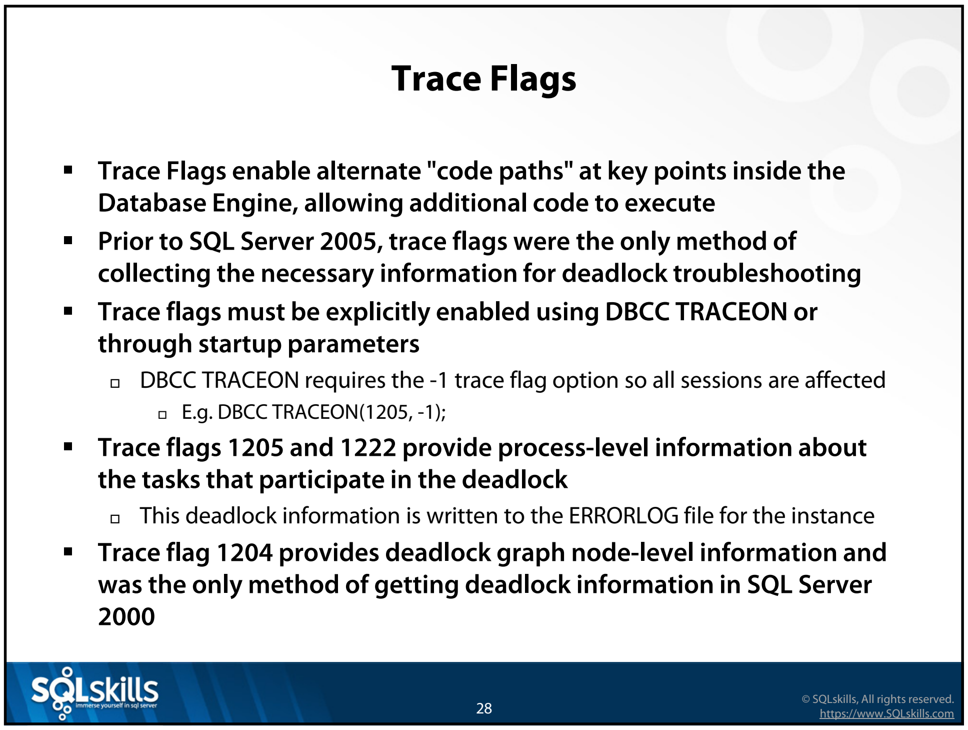
25

Questions?



Reference Slides

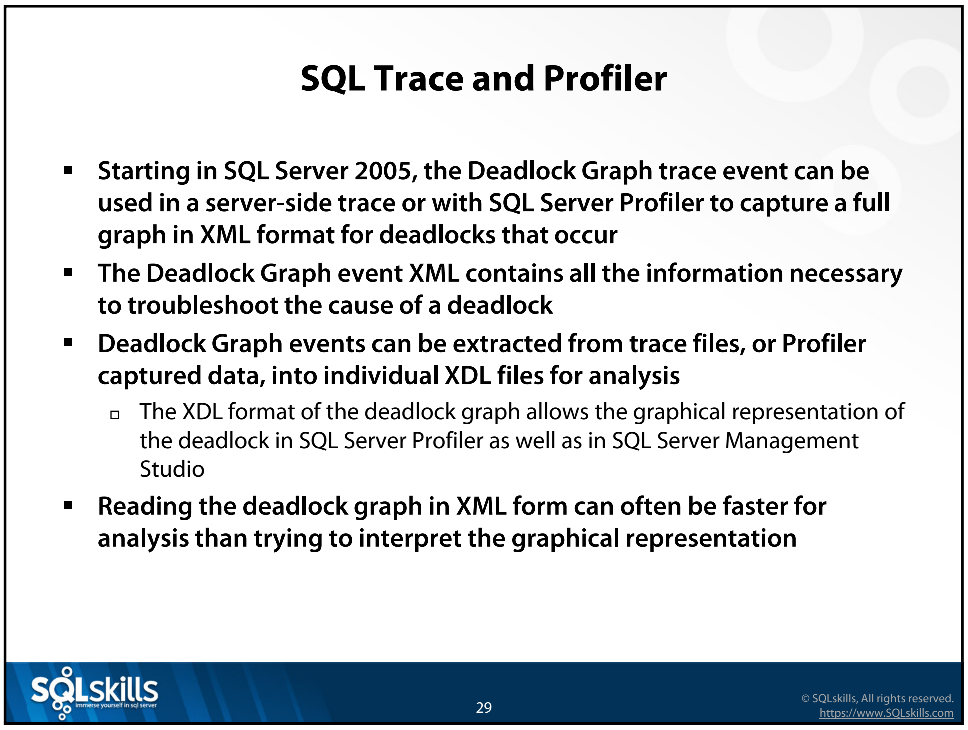
Deadlock collection methods



Trace Flags

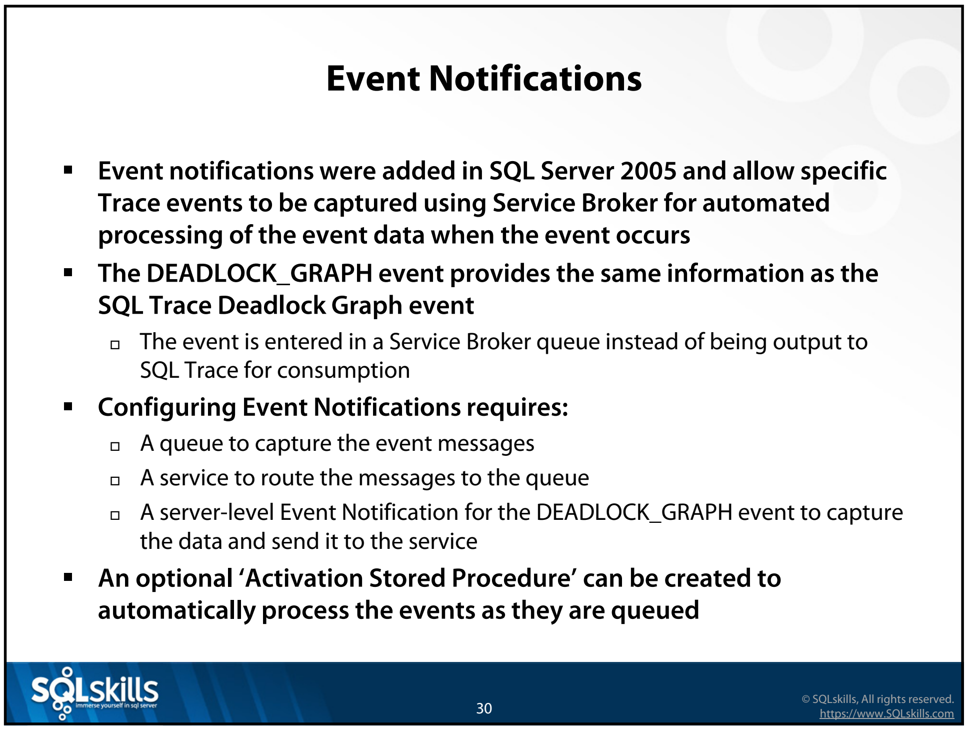
- **Trace Flags enable alternate "code paths" at key points inside the Database Engine, allowing additional code to execute**
- **Prior to SQL Server 2005, trace flags were the only method of collecting the necessary information for deadlock troubleshooting**
- **Trace flags must be explicitly enabled using DBCC TRACEON or through startup parameters**
  - DBCC TRACEON requires the -1 trace flag option so all sessions are affected
    - E.g. DBCC TRACEON(1205, -1);
- **Trace flags 1205 and 1222 provide process-level information about the tasks that participate in the deadlock**
  - This deadlock information is written to the ERRORLOG file for the instance
- **Trace flag 1204 provides deadlock graph node-level information and was the only method of getting deadlock information in SQL Server 2000**

28



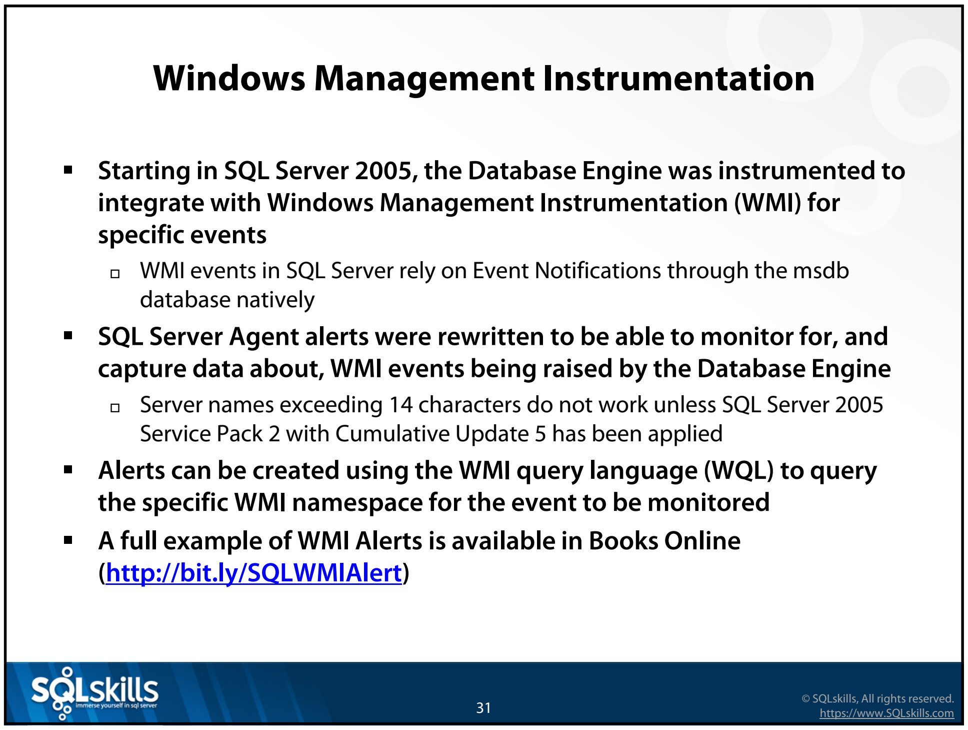
SQL Trace and Profiler

- Starting in SQL Server 2005, the Deadlock Graph trace event can be used in a server-side trace or with SQL Server Profiler to capture a full graph in XML format for deadlocks that occur
- The Deadlock Graph event XML contains all the information necessary to troubleshoot the cause of a deadlock
- Deadlock Graph events can be extracted from trace files, or Profiler captured data, into individual XDL files for analysis
  - The XDL format of the deadlock graph allows the graphical representation of the deadlock in SQL Server Profiler as well as in SQL Server Management Studio
- Reading the deadlock graph in XML form can often be faster for analysis than trying to interpret the graphical representation



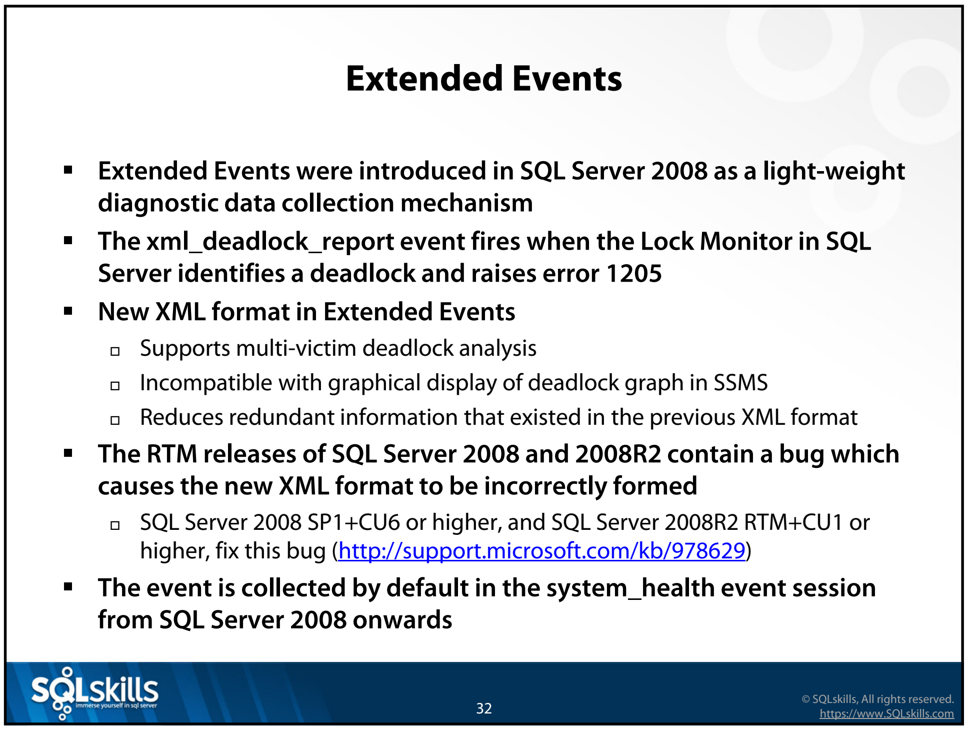
Event Notifications

- Event notifications were added in SQL Server 2005 and allow specific Trace events to be captured using Service Broker for automated processing of the event data when the event occurs
- The DEADLOCK_GRAPH event provides the same information as the SQL Trace Deadlock Graph event
  - The event is entered in a Service Broker queue instead of being output to SQL Trace for consumption
- Configuring Event Notifications requires:
  - A queue to capture the event messages
  - A service to route the messages to the queue
  - A server-level Event Notification for the DEADLOCK_GRAPH event to capture the data and send it to the service
- An optional 'Activation Stored Procedure' can be created to automatically process the events as they are queued



Windows Management Instrumentation

- **Starting in SQL Server 2005, the Database Engine was instrumented to integrate with Windows Management Instrumentation (WMI) for specific events**
  - WMI events in SQL Server rely on Event Notifications through the msdb database natively
- **SQL Server Agent alerts were rewritten to be able to monitor for, and capture data about, WMI events being raised by the Database Engine**
  - Server names exceeding 14 characters do not work unless SQL Server 2005 Service Pack 2 with Cumulative Update 5 has been applied
- **Alerts can be created using the WMI query language (WQL) to query the specific WMI namespace for the event to be monitored**
- **A full example of WMI Alerts is available in Books Online (http://bit.ly/SQLWMIAlert)**



Extended Events

- **Extended Events were introduced in SQL Server 2008 as a light-weight diagnostic data collection mechanism**
- **The xml_deadlock_report event fires when the Lock Monitor in SQL Server identifies a deadlock and raises error 1205**
- **New XML format in Extended Events**
  - Supports multi-victim deadlock analysis
  - Incompatible with graphical display of deadlock graph in SSMS
  - Reduces redundant information that existed in the previous XML format
- **The RTM releases of SQL Server 2008 and 2008R2 contain a bug which causes the new XML format to be incorrectly formed**
  - SQL Server 2008 SP1+CU6 or higher, and SQL Server 2008R2 RTM+CU1 or higher, fix this bug (http://support.microsoft.com/kb/978629)
- **The event is collected by default in the system_health event session from SQL Server 2008 onwards**