A Basic Overview of Locks in SQL Server

We have all heard of locks and can probably even name a few; shared or exclusive locks, for example. We notice them more when something goes wrong and we run into blocking or other performance problems, but what are locks and how do they work in SQL Server?

In this blog, I aim to give a basic answer to that question and provide you with an overview of the different lock modes in SQL Server and how it all works…

Let us make it clear from the off; locks are an **essential** part of SQL Server. In a multi-user system, there will be many users who wish to access the same resources at the same time. This means that SQL Server must have measures in place to handle concurrency and prevent adverse side effects. Locking is one of those measures.

When the database engine processes a statement, the query processor decides which resources need to be accessed and how they will be used. Based on this information, it then determines what types of locks are required to protect each resource. The locks acquired also depend upon the transaction isolation level setting (more on that in a future blog). The query processor then requests the locks from the lock manager, which grants them if no conflicting locks are held by other transactions. If a conflicting lock is held by another transaction, the query will be made to wait until the other lock is released. In short, locks are there to protect resources.

Now that we know that locks are essential for the basic functioning of a healthy server, let us have a look at the different resources that can be locked. The following table shows the resources that SQL Server can place locks on:

Resource	Description
RID (Row Identifier)	A single row within a heap
Key	A row lock within an index
Page	An 8Kb page in a database
Extent	A contiguous group of 8 pages
HoBT	A heap or B-tree
Table	The whole table including indexes

File	A lock on a database file
Application	A lock on an application-specific resource
Metadata	A lock on metadata
Allocation_Unit	A lock on an allocation unit
Database	A lock on the whole database
Object	A lock on anything that has an entry in sys.all_objects. E.g. a stored procedure or view

SQL Server makes use of **multigranular locking** to try and minimize the cost of locking by allowing different types of resources to be locked by a transaction. Locking at a lower granularity, such as rows, increases concurrency, as access is only restricted to those rows rather than the whole table. Still, it has a higher overhead because a lock must be held on each row.

This overhead comes in the form of increased memory usage. A lock as an in-memory structure is 96 bytes in size, so locking millions of rows could have a high overhead compared to gaining a singular lock on the table.

SQL Server uses **lock escalation** to manage the locking granularity. Lock escalation is internally managed and decides at which point to move a set of locks to a higher granularity. This means that SQL Server dynamically manages locking without any input needed from the user.

The Database Engine typically acquires locks at multiple levels of granularity to fully protect a resource; this process is described as the lock hierarchy. Take, for example, a read of an index. To fully protect this read, SQL Server needs shared locks on rows and intent shared locks on the pages and table. If it did not do this, then something might gain a lock at a higher level, which would be problematic.

Lock Modes

Talking of the different lock modes, below is a quick summary:

Lock Mode	Description
Shared (S)	Used for read operations that won't need to update or modify data
Update (U)	Used for resources that might be updated
Exclusive (X)	Used for data modification operations
Intent (I)	Used to establish lock hierarchy (IS), (IX), (IU), (SIU), (UIX) and (SIX)
Schema (Sch)	Used when the schema of a table is updated
Bulk Update (BU)	Used to bulk copy data into a table
Key-range	Protects range of rows

Shared locks allow concurrent transactions to read a resource, but no other transaction can modify the resource while the lock is held.

Update locks are used by transactions that might need to update the resource. Only one transaction can gain an update lock at a time. If data is to be modified, then the lock is converted to an exclusive lock. This means you can still acquire shared locks to read data while the update lock is held.

Exclusive locks are used to ensure that only one transaction can update data at one time and that nothing else can access that data unless the read uncommitted isolation level is set. (More on isolation levels in a coming blog).

Intent locks are used to protect the lock hierarchy and are acquired before locks are requested at lower levels. They protect the lower-level resources by stopping transactions modifying higher-level resources. They come in several different forms: Intent shared (IS), Intent exclusive (IX), shared with intent exclusive (SIX), intent update (IU), shared with intent update (SIU) and Update with intent exclusive (UIX).

Schema modification Locks (sch-M) are used to prevent concurrent access to a table while data definition/modification language operations, such as adding a column, truncating a table or dropping a table are happening.

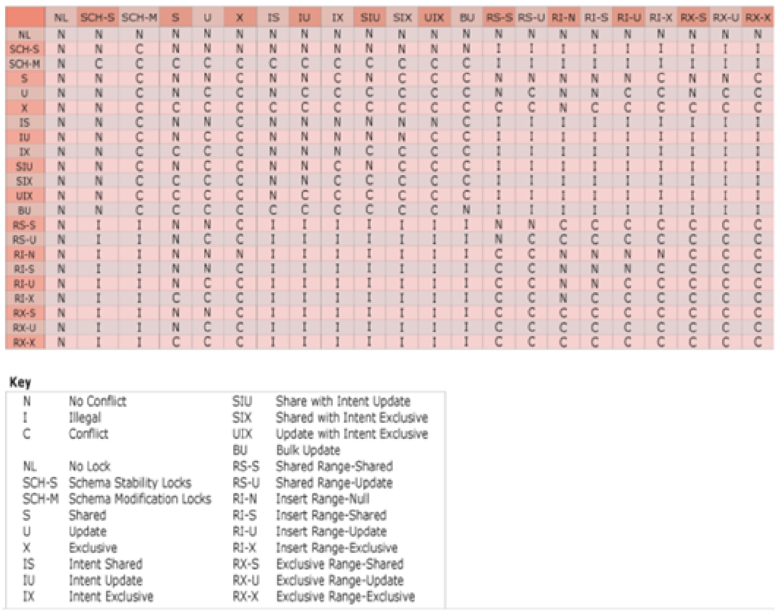
Schema stability Locks (sch-S) are used when the database engine compiles and executes a query and only block sch-M locks.

Bulk Update Locks allow multiple threads to bulk load data concurrently into a table, while blocking access for processes that are not bulk loading data**.**

Key-Range Locks are used by the serializable isolation to protect against phantom reads by locking all value in a key range.

Lock Compatibility

An introduction to locking would not be complete without the lock compatibility table (https://docs.microsoft.com/en-us/sql/2014-toc/media/lockconflicttable.gif?view=sql-server-2014) taken from Microsoft:



	NL	SCH-S	SCH-M	S	U	X	IS	IU	IX	SIU	SIX	UIX	BU	RS-S	RS-U	RI-N	RI-S	RI-U	RI-X	RX-S	RX-U	RX-X
NL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
SCH-S	N	N	C	N	N	N	N	N	N	N	N	N	N	I	I	I	I	I	I	I	I	I
SCH-M	N	C	C	C	C	C	C	C	C	C	C	C	C	I	I	I	I	I	I	I	I	I
S	N	N	C	N	N	C	N	N	C	N	C	C	C	N	N	N	N	N	C	N	N	C
U	N	N	C	N	C	C	N	C	C	C	C	C	C	N	C	N	N	C	C	N	C	C
X	N	N	C	C	C	C	C	C	C	C	C	C	C	C	C	N	C	C	C	C	C	C
IS	N	N	C	N	N	C	N	N	N	N	N	N	C	I	I	I	I	I	I	I	I	I
IU	N	N	C	N	C	C	N	N	N	N	N	C	C	I	I	I	I	I	I	I	I	I
IX	N	N	C	C	C	C	N	N	N	C	C	C	C	I	I	I	I	I	I	I	I	I
SIU	N	N	C	N	C	C	N	N	C	N	C	C	C	I	I	I	I	I	I	I	I	I
SIX	N	N	C	C	C	C	N	N	C	C	C	C	C	I	I	I	I	I	I	I	I	I
UIX	N	N	C	C	C	C	N	C	C	C	C	C	C	I	I	I	I	I	I	I	I	I
BU	N	N	C	C	C	C	C	C	C	C	C	C	N	I	I	I	I	I	I	I	I	I
RS-S	N	I	I	N	N	C	I	I	I	I	I	I	I	N	N	C	C	C	C	C	C	C
RS-U	N	I	I	N	C	C	I	I	I	I	I	I	I	N	C	C	C	C	C	C	C	C
RI-N	N	I	I	N	N	N	I	I	I	I	I	I	I	C	C	N	N	N	N	C	C	C
RI-S	N	I	I	N	N	C	I	I	I	I	I	I	I	C	C	N	N	N	C	C	C	C
RI-U	N	I	I	N	C	C	I	I	I	I	I	I	I	C	C	N	N	C	C	C	C	C
RI-X	N	I	I	C	C	C	I	I	I	I	I	I	I	C	C	N	C	C	C	C	C	C
RX-S	N	I	I	N	N	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C
RX-U	N	I	I	N	C	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C
RX-X	N	I	I	C	C	C	I	I	I	I	I	I	I	C	C	C	C	C	C	C	C	C

Key

N	No Conflict	SIU	Share with Intent Update
I	Illegal	SIX	Shared with Intent Exclusive
C	Conflict	UIX	Update with Intent Exclusive
		BU	Bulk Update
NL	No Lock	RS-S	Shared Range-Shared
SCH-S	Schema Stability Locks	RS-U	Shared Range-Update
SCH-M	Schema Modification Locks	RI-N	Insert Range-Null
S	Shared	RI-S	Insert Range-Shared
U	Update	RI-U	Insert Range-Update
X	Exclusive	RI-X	Insert Range-Exclusive
IS	Intent Shared	RX-S	Exclusive Range-Shared
IU	Intent Update	RX-U	Exclusive Range-Update
IX	Intent Exclusive	RX-X	Exclusive Range-Exclusive

Lock compatibility controls whether concurrent transactions can acquire locks on the same resource at the same time. If a transaction requests a lock on an already locked resource, it will only be granted if it is compatible. If it is not, the transaction will have to **wait** for the lock to be released. This is an important point, because if a transaction spends a long time waiting for a lock, we can start to see problems. We might assume the process is running slowly, but in reality it is blocked as it can't get a lock on the resource until the other transaction has released it.

Summary

This is only a fundamental overview, but how does it help you on a practical level? First, knowing the basics will help you have a better understanding of how different queries will interact and which operations will take locks that will block. This is a useful thing to have in mind when writing your code. Secondly, it is vital to know a bit about it, to make it easier to understand other concepts, particularly isolation levels, which I will come to discuss in a future blog.

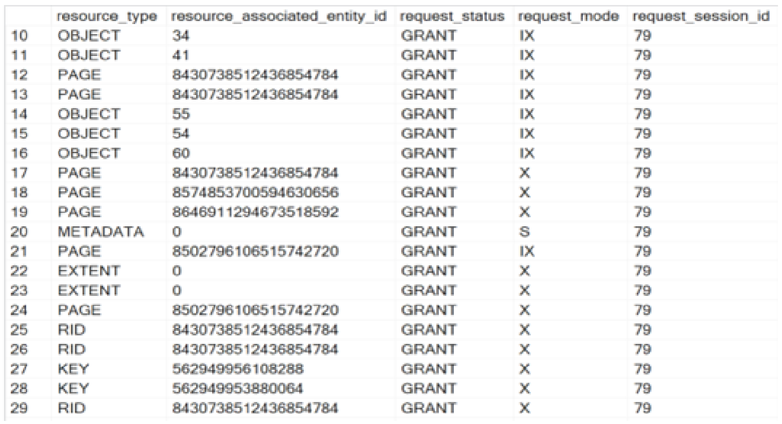
It also allows you to do some basic troubleshooting if you see blocking taking place on your server and gives you an insight into why it might be happening. Hopefully, this blog has given you the basics to let you delve deeper into the internals of SQL Server.

Overall, we can see that locks are very useful, they ensure that multiple users cannot modify the same data at the same time. As a result of this it can occasionally appear that processes are running slowly, however it won't be the process that is running slowly, it will be due to it spending time waiting on other processes to finish what they're doing and release the locks. If this behaviour did not exist, you could quickly end up with inconsistent data in your database, which nobody wants!

There are a few good DMVs to investigate if you want to watch locking in action on your servers:

sys.dm_tran_locks (link (https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-tran-locks-transact-sql?view=sql-server-ver15))

The below image shows some useful columns in it, such as the status and the lock mode, as well as the resource type that the lock is targeted at.



	resource_type	resource_associated_entity_id	request_status	request_mode	request_session_id
10	OBJECT	34	GRANT	IX	79
11	OBJECT	41	GRANT	IX	79
12	PAGE	8430738512436854784	GRANT	IX	79
13	PAGE	8430738512436854784	GRANT	IX	79
14	OBJECT	55	GRANT	IX	79
15	OBJECT	54	GRANT	IX	79
16	OBJECT	60	GRANT	IX	79
17	PAGE	8430738512436854784	GRANT	X	79
18	PAGE	8574853700594630656	GRANT	X	79
19	PAGE	8646911294673518592	GRANT	X	79
20	METADATA	0	GRANT	S	79
21	PAGE	8502796106515742720	GRANT	IX	79
22	EXTENT	0	GRANT	X	79
23	EXTENT	0	GRANT	X	79
24	PAGE	8502796106515742720	GRANT	X	79
25	RID	8430738512436854784	GRANT	X	79
26	RID	8430738512436854784	GRANT	X	79
27	KEY	562949956108288	GRANT	X	79
28	KEY	562949953880064	GRANT	X	79
29	RID	8430738512436854784	GRANT	X	79

and joining it with **sys.dm_os_wait_stats (link)** (https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-os-wait-stats-transact-sql?view=sql-server-ver15) can help you look into blocking on your server. More details and examples can be found in the links.

That is all for me in this blog, I hope you have found it useful and learned something, and I look forward to seeing you again on my next post!

Get the latest blogs sent to your inbox (https://blog.coeo.com/cs/c/?cta_g
5a6ab62c29dc&click=6c3d8d42-d029-4cc4-a8ba-4bb
server&utm_referrer=https%3A%2F%2Fwww.google.com%2F&portal_id=3356718
l3rlTYG4a_1sdiuBEbUJEoSWe7e6MKFS
g9JLyDHPM0weo&__hstc=145057003.cb0625040a865b430a0b3d240

Rawoof Shaik 14/10/2020, 06:28:20

Very nice explainations

Reply to *Rawoof Shaik*

Vasyl Kosovan 30/10/2020, 17:08:31

What are the differences between conflict and Illegal?

Reply to *Vasyl Kosovan*

Dan Jackson 12/11/2020, 18:00:13

Hi Vasyl, thank you for your question, unfortunately I'm not 100% sure what you're asking. Would it be possible for you to provide a little bit more detail about what you want to know and I'll endeavor to get back to you with a more helpful answer.
Cheers

Leave comment:

First Name*

Last Name

Email*

Website

Comment*

reCAPTCHA محمي بخدمة
الخصوصية - البنود

Submit Comment

Subscribe to Email Updates

Email*

Coeo needs the contact information you provide to us to contact you about our products and services. You may unsubscribe from these communications at any time. For more information on how to unsubscribe and our commitment to your privacy, please review our **Privacy Policy (https://www.coeo.com/privacy/)**.

Subscribe

Related posts

Concurrency with Azure Backup for SQL Server

(https://blog.coeo.com/concurrency-with-azure-backup-for-sql-server)

8 Reasons to Choose Dedicated Support

(https://blog.coeo.com/8-reasons-to-choose-dedicated-support)

Podcast: Data Platform Modernisation (with Simon Osborne)

(https://blog.coeo.com/podcast-data-platform-modernisation)

Cyber Security Monthly News - October 2020

(https://blog.coeo.com/cyber-security-monthly-news-october-2020)

+44 (0)20 3051 3595 | info@coeo.com

Contact Us

Name

Email *

Message

We would like to keep in touch with information about our products and services, as well as other content that may be of interest to you.

☐
I agree to receive other communications from Coeo.

reCAPTCHA محمي بخدمة
الخصوصية - البنود

SUBMIT

Upcoming Events

See all events (https://www.coeo.com/events/)

710 Wharfedale Road, Winnersh, Wokingham, Berkshire, RG41 5TP.

(https://www.glassdoor.co.uk/Overview/Working-at-Coeo-EI_IE959052.11,15.htm) (https://www.linkedin.com/company/coeo-ltd) (https://twitter.com/CoeoLtd) (https://www.facebook.com/coeoltd/)

BACK TO TOP