

Main concept of SQL Server locking

📅 June 26, 2018 (<https://codingsight.com/main-concept-of-sql-server-locking/>) 👤 Esat Erkeç
(<https://codingsight.com/author/esaterkec/>) 📁 SQL Server (<https://codingsight.com/category/sql-server/>)

Total: 38 Average: 4

In this post, we will discuss the SQL Server lock mechanism and how to monitor SQL Server locking with SQL Server standard dynamic management views. Before we start to explain SQL Server lock architecture, let's take a moment to describe what the ACID (Atomicity, Consistency (<https://www.lifewire.com/database-consistency-definition-1019249>), Isolation (<https://www.lifewire.com/isolation-definition-1019173>), and Durability) database is. The ACID database can be explained as database theory. If a database is called relational database, it has to meet Atomicity, Consistency (<https://www.lifewire.com/database-consistency-definition-1019249>), Isolation (<https://www.lifewire.com/isolation-definition-1019173>), and Durability requirements. Now, we will explain these requirements briefly.

Atomicity: It reflects the principle of indivisibility that we describe as the main feature of the transaction process. A transaction block cannot be left unattended. Half of the remaining transaction block causes data inconsistency. Either the entire transaction is performed or the transaction returns to the beginning. That is, all changes made by the transaction are undone and returned to their previous state.

Consistency: There is a rule that sets the substructure of the non-divisibility rule. Transaction data must provide consistency. That is, if the update operation is performed in a transaction, either all remaining transactions must be performed or the update operation must be canceled. This data is very important in terms of consistency.

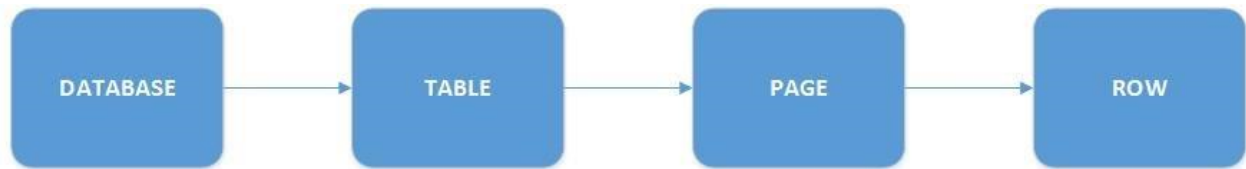
Isolation: This is a request packet for each transaction database. Changes made by a request packet must be visible to another transaction before it is complete. Each transaction must be processed separately. All transactions must be visible to another transaction after they occur.

Durability: Transactions can perform complex operations with data. In order to secure all these transactions, they must be resistant to a transaction error. System problems that may occur in SQL Server should be prepared and resilient against power failure, operating system or other software-induced errors.

Transaction: Transaction is the smallest stack of the process which cannot be divided into smaller pieces. Also, some group of transaction process can be performed sequentially but as we explained in the Atomicity principle if even one of the transactions fails, all transaction blocks will fail.

Lock: Lock is a mechanism to ensure data consistency. SQL Server locks objects when the transaction starts. When the transaction is completed, SQL Server releases the locked object. This lock mode can be changed according to the SQL Server process type and isolation level. These lock modes are:

Lock hierarchy: SQL Server has a lock hierarchy which acquires lock objects in this hierarchy. A database is located at the top of the hierarchy and row is located at the bottom. The below image illustrates the lock hierarchy of SQL Server.



Shared (S) Locks: This lock type occurs when the object needs to be read. This lock type does not cause much problem.

Exclusive (X) Locks: When this lock type occurs, it occurs to prevent other transactions to modify or access a locked object.

Update (U) Locks: This lock type is similar to the exclusive lock but it has some differences. We can divide the update operation into different phases: read phase and write phase. During the read phase, SQL Server does not want other transactions to have access to this object to be changed. For this reason, SQL Server uses the update lock.

Intent Locks: The intent lock occurs when SQL Server wants to acquire the shared (S) lock or exclusive (X) lock on some of the resources lower in the lock hierarchy. In practice, when SQL Server acquires a lock on a page or row, the intent lock is required in the table.

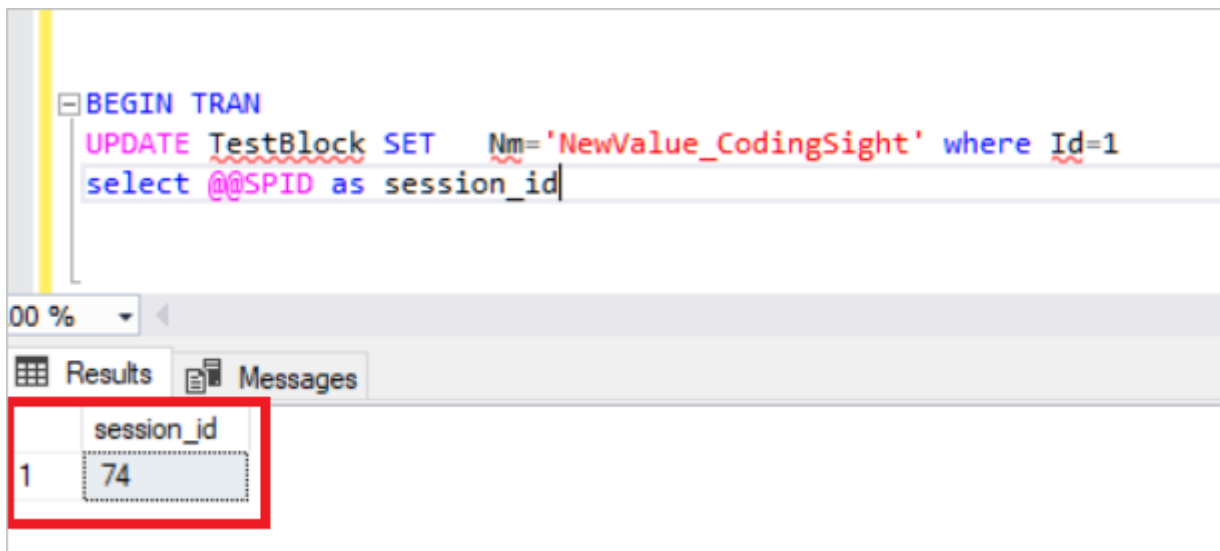
After all these brief explanations, we will try to find an answer to how to identify locks. SQL Server offers a lot of dynamic management views to access metrics. To identify SQL Server locks, we can use the **sys.dm_tran_locks** view. In this view, we can find a lot of information about currently active lock manager resources.

In the first example, we will create a demo table which does not include any indexes and try to update this demo table.

```
CREATE TABLE TestBlock
(Id INT ,
Nm VARCHAR(100))

INSERT INTO TestBlock
values(1,'CodingSight')
In this step, we will create an open transaction and analyze the locked resources.
BEGIN TRAN
UPDATE TestBlock SET Nm='NewValue_CodingSight' where Id=1
select @@SPID
```





The screenshot shows a SQL Server query window with the following code:

```
BEGIN TRAN
UPDATE TestBlock SET Nm='NewValue_CodingSight' where Id=1
select @@SPID as session_id
```

Below the query window, the 'Results' tab is active, displaying a single row of data:

	session_id
1	74

The first row of the results table is highlighted with a red border.

Now, we will check the sys.dm_tran_lock view.

```
select * from sys.dm_tran_locks WHERE request_session_id=74
```

This view returns a lot of information about active lock resources. But it is not possible to understand some of the data in this view. For this reason, we have to join the **sys.dm_tran_locks** view to other views.



```

SELECT dm_tran_locks.request_session_id,
       dm_tran_locks.resource_database_id,
       DB_NAME(dm_tran_locks.resource_database_id) AS dbname,
       CASE
         WHEN resource_type = 'OBJECT'
           THEN OBJECT_NAME(dm_tran_locks.resource_associated_entity_id)
         ELSE OBJECT_NAME(partitions.OBJECT_ID)
       END AS ObjectName,
       partitions.index_id,
       indexes.name AS index_name,
       dm_tran_locks.resource_type,
       dm_tran_locks.resource_description,
       dm_tran_locks.resource_associated_entity_id,
       dm_tran_locks.request_mode,
       dm_tran_locks.request_status
FROM sys.dm_tran_locks
LEFT JOIN sys.partitions ON partitions.hobt_id = dm_tran_locks.resource_associated_entity_id
LEFT JOIN sys.indexes ON indexes.OBJECT_ID = partitions.OBJECT_ID AND indexes.index_id = partitions.index_id
WHERE resource_associated_entity_id > 0
      AND resource_database_id = DB_ID()
      and request_session_id=74
ORDER BY request_session_id, resource_associated_entity_id

```

In the above image, you can see the locked resources. SQL Server acquires the exclusive lock in that row. **(RID: A row identifier used to lock a single row within a heap)** At the same time, SQL Server acquires the intent exclusive lock in the page and the **TestBlock** table. It means that any other process cannot read this resource until the SQL Server releases the locks. This is the basic lock mechanism in the SQL Server. ^

Now, we will populate some synthetic data on our test table.

```
TRUNCATE TABLE    TestBlock
DECLARE @K AS INT=0
WHILE @K <8000
BEGIN
INSERT TestBlock VALUES(@K, CAST(@K AS varchar(10)) + ' Value' )
SET @K=@K+1
END
```

After completing this step, we will run two queries and check the sys.dm_tran_locks view.

```
BEGIN TRAN
UPDATE TestBlock  set Nm ='New_Value' where Id<5000
```

In the above query, SQL Server acquires the exclusive lock on every single row. Now, we will run another query.

```
BEGIN TRAN
UPDATE TestBlock  set Nm ='New_Value' where Id<7000
```



In the above query, SQL Server creates the exclusive lock on the table, Because SQL Server tries to acquire a lot of RID locks for these rows which will be updated. This case causes a lot of resource consumption in the database engine. Therefore, SQL Server automatically moves this exclusive lock to an up-level object which is in the lock hierarchy. We define this mechanism as Lock Escalation ([https://technet.microsoft.com/en-us/library/ms172010\(v=sql.110\).aspx](https://technet.microsoft.com/en-us/library/ms172010(v=sql.110).aspx)). Lock Escalation can be changed at the table level.

```
ALTER TABLE XX_TableName
SET
(
    LOCK_ESCALATION = AUTO -- or TABLE or DISABLE
)
GO
```

I would like to add some notes about lock escalation. If you have a partitioned table, then we can set the escalation to the partition level.

In this step, we will execute a query which creates a lock in the AdventureWorks HumanResources table. This table has clustered and non-clustered indexes.

```
BEGIN TRAN
UPDATE [HumanResources].[Department] SET Name='NewName' where DepartmentID=1
```



As you can see in the below result pane, our transaction acquires exclusive locks in the PK_Department_DepartmentID cluster index key and also acquires exclusive locks in the AK_Department_Name non-clustered index key. Now, we can ask this question “Why SQL Server locks a non-clustered index?”

The **Name** column is indexed in the AK_Department_Name non-clustered index and we try to change the **Name** column. In this case, SQL Server needs to change any non-clustered indexes on that column. The non-clustered index leaf level includes every KEY value sorted out.

Conclusions

In this article, we mentioned the main lines of SQL Server lock mechanism and considered the use of sys.dm_tran_locks. The sys.dm_tran_locks view returns a lot of information about currently active lock resources. If you google, you can find a lot of sample queries about this view.

References

SQL Server Transaction Locking and Row Versioning Guide (<https://technet.microsoft.com/en-us/library/jj856598%28v=sql.110%29.aspx?f=255&MSPPErrors=-2147217396>)

SQL Server, Locks Object (<https://docs.microsoft.com/en-us/sql/relational-databases/performance-monitor/sql-server-locks-object?view=sql-server-2017>)

Esat Erkeç (<https://www.linkedin.com/in/Esat-Erkeç-22433882>**)**

Esat Erkeç is an SQL Server professional that began his career as a Software Developer over 8 years ago. He is an SQL Server Microsoft Certified Solutions





Expert. Most of his career has focused on SQL Server Database Administration and Development. His current interests are in database administration and Business Intelligence.



(<https://www.linkedin.com/in/esat-erkeç-22433882>)

[performance](https://codingsight.com/tag/performance/) (<https://codingsight.com/tag/performance/>)

[sql server](https://codingsight.com/tag/sql-server/) (<https://codingsight.com/tag/sql-server/>)

ALSO ON CODINGSIGHT.COM

Learn to Store and Analyze Documents ...

8 months ago • 1 comment

The final part goes into every detail and exemplifies storing and analyzing ...

Restoring the SQL Server Master ...

7 months ago • 1 comment

It is crucial to back up the SQL Server master database daily. Read to ...

Implementing a TeamCity Plugin – ...

a year ago • 1 comment

The article details the development of a TeamCity plugin. The main steps ...

Seven for Dev

5 months

This article covers seven in SQL Se



0 Comments

codingsight.com

 Disqus' Privacy Policy Login ▾ Recommend Tweet Share

Sort by Best ▾



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Be the first to comment.

 Subscribe  Add Disqus to your site  Add  Do Not Sell My Data[◀ Migrating SQL Server Database to Azure SQL \(PaaS\) \(https://codingsight.com/migrating-sql-server-database-to-azure-sql-paas/\)](https://codingsight.com/migrating-sql-server-database-to-azure-sql-paas/)[Introduction to Docker and Docker Toolbox ▶ \(https://codingsight.com/introduction-to-docker-and-docker-toolbox/\)](https://codingsight.com/introduction-to-docker-and-docker-toolbox/)

SIGN UP FOR CODINGSIGHT DIGEST

MOST POPULAR POSTS

Introduction to Temporary Tables in SQL Server (<https://codingsight.com/introduction-to-temporary-tables-in-sql-server/>)

Calculating Running Total with OVER Clause and PARTITION BY Clause in SQL Server (<https://codingsight.com/calculating-running-total-with-over-clause-and-partition-by-clause-in-sql-server/>)

Grouping Data using the OVER and PARTITION BY Functions (<https://codingsight.com/grouping-data-using-the-over-and-partition-by-functions/>)

10 Best MySQL GUI Tools (<https://codingsight.com/10-best-mysql-gui-tools/>)

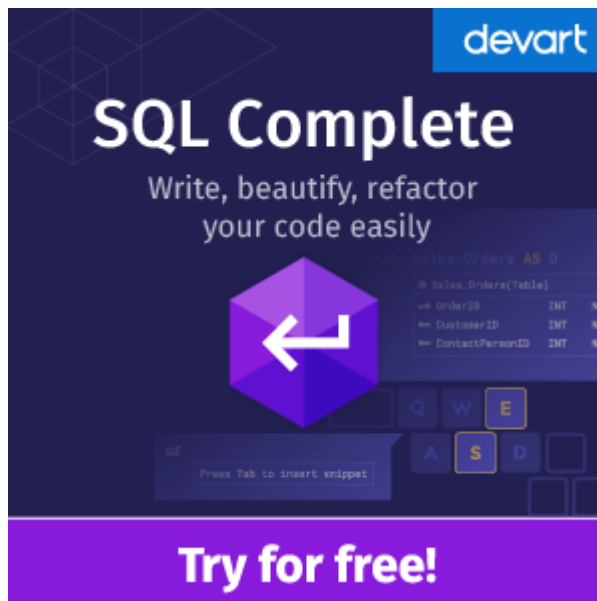
Similarities and Differences among RANK, DENSE_RANK and ROW_NUMBER Functions (https://codingsight.com/similarities-and-differences-among-rank-dense_rank-and-row_number-functions/)



Git Branching Naming Convention: Best Practices (<https://codingsight.com/git-branching-naming-convention-best-practices/>)

Methods to Rank Rows in SQL Server: ROW_NUMBER(), RANK(), DENSE_RANK() and NTILE() (<https://codingsight.com/methods-to-rank-rows-in-sql-server-rownumber-rank-denserank-and-ntile/>)

Passing Data table as Parameter to Stored Procedures (<https://codingsight.com/passing-data-table-as-parameter-to-stored-procedures/>)



(<https://bit.ly/353NqFM>)

Exclusively for the Codingsight readers, **Devart** (<https://bit.ly/2JB7zvm>) offers a 10% discount on all product license purchases!

Simply use a **CODINGSIGHT** coupon code.

BECOME AN AUTHOR

What does it take to start writing for us? Learn more (<https://codingsight.com/write-for-us/>)

Read what our contributors say about us (<https://codingsight.com/testimonials/>)

TAGS



[.net \(https://codingsight.com/tag/net/\)](https://codingsight.com/tag/net/) [.net framework \(https://codingsight.com/tag/net-framework/\)](https://codingsight.com/tag/net-framework/)
[asp.net core \(https://codingsight.com/tag/asp-net-core/\)](https://codingsight.com/tag/asp-net-core/) [azure \(https://codingsight.com/tag/azure/\)](https://codingsight.com/tag/azure/)
[azure sql \(https://codingsight.com/tag/azure-sql/\)](https://codingsight.com/tag/azure-sql/) [c# \(https://codingsight.com/tag/c/\)](https://codingsight.com/tag/c/)
[database administration \(https://codingsight.com/tag/database-administration/\)](https://codingsight.com/tag/database-administration/)
[database backup \(https://codingsight.com/tag/database-backup/\)](https://codingsight.com/tag/database-backup/)
[database security \(https://codingsight.com/tag/database-security/\)](https://codingsight.com/tag/database-security/)
[design patterns \(https://codingsight.com/tag/design-patterns/\)](https://codingsight.com/tag/design-patterns/)
[docker \(https://codingsight.com/tag/docker/\)](https://codingsight.com/tag/docker/)
[entity framework \(https://codingsight.com/tag/entity-framework/\)](https://codingsight.com/tag/entity-framework/)
[execution plan \(https://codingsight.com/tag/execution-plan/\)](https://codingsight.com/tag/execution-plan/)
[failover cluster instance \(https://codingsight.com/tag/failover-cluster-instance/\)](https://codingsight.com/tag/failover-cluster-instance/)
[full-text search \(https://codingsight.com/tag/full-text-search/\)](https://codingsight.com/tag/full-text-search/)
[indexes \(https://codingsight.com/tag/indexes/\)](https://codingsight.com/tag/indexes/) [json \(https://codingsight.com/tag/json/\)](https://codingsight.com/tag/json/)
[linked server \(https://codingsight.com/tag/linked-server/\)](https://codingsight.com/tag/linked-server/) [linq \(https://codingsight.com/tag/linq/\)](https://codingsight.com/tag/linq/)
[linux \(https://codingsight.com/tag/linux/\)](https://codingsight.com/tag/linux/) [Microsoft Azure \(https://codingsight.com/tag/microsoft-azure/\)](https://codingsight.com/tag/microsoft-azure/)
[mysql \(https://codingsight.com/tag/mysql/\)](https://codingsight.com/tag/mysql/) [oracle \(https://codingsight.com/tag/oracle/\)](https://codingsight.com/tag/oracle/)
[performance \(https://codingsight.com/tag/performance/\)](https://codingsight.com/tag/performance/) [power bi \(https://codingsight.com/tag/power-bi/\)](https://codingsight.com/tag/power-bi/)
[query performance \(https://codingsight.com/tag/query-performance/\)](https://codingsight.com/tag/query-performance/)
[sql \(https://codingsight.com/tag/sql/\)](https://codingsight.com/tag/sql/) [sql constraints \(https://codingsight.com/tag/sql-constraints/\)](https://codingsight.com/tag/sql-constraints/)
[sql database \(https://codingsight.com/tag/sql-database/\)](https://codingsight.com/tag/sql-database/)
[sql functions \(https://codingsight.com/tag/sql-functions/\)](https://codingsight.com/tag/sql-functions/)
[sql operator \(https://codingsight.com/tag/sql-operator/\)](https://codingsight.com/tag/sql-operator/)
[sql server \(https://codingsight.com/tag/sql-server/\)](https://codingsight.com/tag/sql-server/)
[sql server 2016 \(https://codingsight.com/tag/sql-server-2016/\)](https://codingsight.com/tag/sql-server-2016/)
[sql server 2017 \(https://codingsight.com/tag/sql-server-2017/\)](https://codingsight.com/tag/sql-server-2017/) [ssis \(https://codingsight.com/tag/ssis/\)](https://codingsight.com/tag/ssis/)
[ssms \(https://codingsight.com/tag/ssms/\)](https://codingsight.com/tag/ssms/) [t-sql \(https://codingsight.com/tag/t-sql/\)](https://codingsight.com/tag/t-sql/)
[t-sql queries \(https://codingsight.com/tag/t-sql-queries/\)](https://codingsight.com/tag/t-sql-queries/)
[t-sql statements \(https://codingsight.com/tag/t-sql-statements/\)](https://codingsight.com/tag/t-sql-statements/) [tdd \(https://codingsight.com/tag/tdd/\)](https://codingsight.com/tag/tdd/)
[tips and tricks \(https://codingsight.com/tag/tips-and-tricks/\)](https://codingsight.com/tag/tips-and-tricks/) [tools \(https://codingsight.com/tag/tools/\)](https://codingsight.com/tag/tools/)
[transaction log \(https://codingsight.com/tag/transaction-log/\)](https://codingsight.com/tag/transaction-log/)
[unit test \(https://codingsight.com/tag/unit-test/\)](https://codingsight.com/tag/unit-test/) [visual studio \(https://codingsight.com/tag/visual-studio/\)](https://codingsight.com/tag/visual-studio/)

FOLLOW US ON SOCIAL NETWORKS



(<https://twitter.com/codingsight>)

(<https://www.facebook.com/codingsight/>)

(<https://bit.ly/2KmQsgB>)

(<https://jooble.org/jobs-information-technology->

specialist)

SQL SERVER ([HTTPS://CODINGSIGHT.COM/CATEGORY/SQL-SERVER/](https://codingsight.com/category/sql-server/))

.NET ([HTTPS://CODINGSIGHT.COM/CATEGORY/NET/](https://codingsight.com/category/net/))

C# ([HTTPS://CODINGSIGHT.COM/CATEGORY/C/](https://codingsight.com/category/c/))

AZURE ([HTTPS://CODINGSIGHT.COM/CATEGORY/MICROSOFT-AZURE/](https://codingsight.com/category/microsoft-azure/))

ORACLE ([HTTPS://CODINGSIGHT.COM/CATEGORY/ORACLE/](https://codingsight.com/category/oracle/))

POSTGRESQL ([HTTPS://CODINGSIGHT.COM/CATEGORY/POSTGRESQL/](https://codingsight.com/category/postgresql/))

ABOUT ([HTTPS://CODINGSIGHT.COM/ABOUT/](https://codingsight.com/about/))

sparkling Theme by Colorlib (<http://colorlib.com/>) Powered by WordPress (<http://wordpress.org/>)

