**SQL Server Snapshot Isolation Test**

**Conclusion:**

Snapshot isolation will greatly increase concurrency read performance while transaction going on.

Meanwhile without snapshot isolation, Concurrent read commit will be blocked.

It works on SQL Server Standard Edition

**Concern:**

Based on table "B\_MASTER\_REPOSITORY\_ITEM", Snapshot isolation will add 2.2 GB to tempdb while updating 2.1 GB data.

Meanwhile without snapshot isolation, tempdb size will have no change.

Testing target table: B\_MASTER\_REPOSITORY\_ITEM

|  |  |  |  |
| --- | --- | --- | --- |
| Update Data Size (GB) | TempDB total file size (GB) | Row Number | TempDB/Table% |
| 2.2 | 2.06 | 2,090,913 | 94 |
| 6.0 | 4.56 | 3,471,346 | 76 |
| 11.5 | 9.06 | 6,942,692 | 79 |
| 22.5 | 17.56 | 13,885,384 | 78 |
| 44.4 | 34.6 | 27,770,768 | 78 |
| 88.9 | 69.56 | 55,941,550 | 78 |

We can only set snapshot isolation on database level. We can’t set snapshot isolation level on just one table. Also we can’t set it on only transaction neither (although ChatGPT indicates that we can set it on transaction level).

The transaction will be slower than non-snapshot transaction.

It will requires more compute resource (CPU/Memory) since both transaction and retrieving happens at the same time.

**Reference:**

<https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/sql/snapshot-isolation-in-sql-server>

**First test, executing queries with snapshot isolation enabled**

Testing table: **B\_MASTER\_REPOSITORY\_ITEM**

Table size: **6.8 GB**

Row count: 3,471,346

Updating data: REPOSITORY\_ID=10340 row count: 2,090,913 (60% of the table that’s about **4.1 GB**)

Steps:

1. **Get the tempdb info before change to ALLOW\_SNAPSHOT\_ISOLATION**

SELECT DB\_NAME(vsu.database\_id) AS DatabaseName,

vsu.reserved\_page\_count,

vsu.reserved\_space\_kb,

tu.total\_page\_count as tempdb\_pages,

vsu.reserved\_page\_count \* 100. / tu.total\_page\_count AS [Snapshot %],

tu.allocated\_extent\_page\_count \* 100. / tu.total\_page\_count AS [tempdb % used]

FROM sys.dm\_tran\_version\_store\_space\_usage vsu

CROSS JOIN tempdb.sys.dm\_db\_file\_space\_usage tu

WHERE vsu.database\_id = DB\_ID(DB\_NAME());

Result:

Each of tempdbfile size(total 8 files): reserved\_page\_count=0; reserved\_space\_kb=0; **tempdb\_pages=1024** each file; **Snapshot %=0**;

1. **Set the database to ALLOW\_SNAPSHOT\_ISOLATION. I'm using database EPIM**

*It might require restart SQL Server service to make the following 2 statements work.*

ALTER DATABASE [EPIM]

SET ALLOW\_SNAPSHOT\_ISOLATION ON

*Reboot SQL Server*

ALTER DATABASE [EPIM]

SET READ\_COMMITTED\_SNAPSHOT ON

*Reboot SQL Server*

Double Check READ\_COMMITED\_SNAPSHOT

SELECT CASE

WHEN transaction\_isolation\_level = 0 THEN 'Unspecified'

WHEN transaction\_isolation\_level = 1 THEN 'Read Uncommitted'

WHEN transaction\_isolation\_level = 2 AND d.snapshot\_isolation\_state\_desc = 'OFF' THEN 'Read Committed'

WHEN transaction\_isolation\_level = 2 AND d.snapshot\_isolation\_state\_desc = 'ON' AND d.is\_read\_committed\_snapshot\_on = 1 THEN 'Snapshot Read Committed'

WHEN transaction\_isolation\_level = 2 AND d.snapshot\_isolation\_state\_desc = 'ON' AND d.is\_read\_committed\_snapshot\_on = 0 THEN 'Snapshot'

WHEN transaction\_isolation\_level = 3 THEN 'Repeatable Read'

WHEN transaction\_isolation\_level = 4 THEN 'Serializable' END AS TRANSACTION\_ISOLATION\_LEVEL,

d.is\_read\_committed\_snapshot\_on,

d.snapshot\_isolation\_state\_desc

FROM sys.dm\_exec\_sessions

CROSS JOIN sys.databases AS d

where session\_id = @@SPID

AND d.database\_id = DB\_ID();

1. **Preparing queries, DO NOT execute any query at this step yet**

Find a Repository\_ID with large amount of rows.

select TOP 10 Repository\_ID, count(\*) as cnt from B\_MASTER\_REPOSITORY\_ITEM

group by Repository\_ID

order by 2 desc

I'm using Repository\_ID = 10340 that has 2,090,913 (out of 3,471,346) rows for testing 60% of data in 6.3GB table. Estimated modified data size will be **4.1 GB**

1. Open query session 1 for update statement

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

BEGIN TRAN;

UPDATE B\_MASTER\_REPOSITORY\_ITEM SET MERGED\_INTO\_ITEM\_ID = ITEM\_ID

WHERE Repository\_ID = 10340 AND MERGED\_INTO\_ITEM\_ID IS NULL;

/\* COMMIT; \*/

1. Open query session 2 for getting tempdb data file size

SELECT DB\_NAME(vsu.database\_id) AS DatabaseName,

vsu.reserved\_page\_count,

vsu.reserved\_space\_kb,

tu.total\_page\_count as tempdb\_pages,

vsu.reserved\_page\_count \* 100. / tu.total\_page\_count AS [Snapshot %],

tu.allocated\_extent\_page\_count \* 100. / tu.total\_page\_count AS [tempdb % used]

FROM sys.dm\_tran\_version\_store\_space\_usage vsu

CROSS JOIN tempdb.sys.dm\_db\_file\_space\_usage tu

WHERE vsu.database\_id = DB\_ID(DB\_NAME());

1. Open query session 3 try to retrieve data from un-committed snapshot

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SELECT COUNT(\*) as cnt FROM B\_MASTER\_REPOSITORY\_ITEM

WHERE Repository\_ID = 10340 AND MERGED\_INTO\_ITEM\_ID IS NULL;

1. Open query session 4 try to retrieve data from committed snapshot

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SELECT COUNT(\*) as cnt FROM B\_MASTER\_REPOSITORY\_ITEM

WHERE Repository\_ID = 10340 AND MERGED\_INTO\_ITEM\_ID IS NULL;

1. **Execute 4 query sessions**

Execute query session 1 (**without "COMMIT;"**)

Execute query session 2 to see the tempdb usage:

Each of tempdbfile size(total 8 files); reserved\_page\_count=211576; reserved\_space\_kb=1692608; **tempdb\_pages=33792; Snapshot %=626**;

Execute query session 3 to see the result: Read Committed returns count 2,090,913. **It will still return data while updating the same records.**

Execute query session 4 to see the result: Read Uncommited returns count 0, this is dirty read.

1. **Check result after commit**

Goes to query session 1, execute commit command:

COMMIT;

Execute query session 2 to see the tempdb usage:

Each of tempdbfile size(total 8 files); reserved\_page\_count=0; reserved\_space\_kb=0; **tempdb\_pages=1024; Snapshot %=0**;

The tempdb back to the original size after updating statement committed.

Execute query session 3 to see the result: Read Committed returns count 0, as expected

Execute query session 4 to see the result: Read Uncommited returns count 0, as expected

1. **Rollback testing setting and data that updated**

*It might require restart SQL Server service to make the following 2 statements work.*

ALTER DATABASE [EPIM] SET READ\_COMMITTED\_SNAPSHOT OFF

*Reboot SQL Server*

ALTER DATABASE [EPIM] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

*Reboot SQL Server*

UPDATE B\_MASTER\_REPOSITORY\_ITEM SET MERGED\_INTO\_ITEM\_ID = NULL

WHERE Repository\_ID = 10340 AND MERGED\_INTO\_ITEM\_ID = ITEM\_ID;

**Second test, executing querieswithout snapshot isolation enabled**

**Query session 1**, table will be locked, select statement will be blocked

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

BEGIN TRAN;

UPDATE B\_MASTER\_REPOSITORY\_ITEM SET MERGED\_INTO\_ITEM\_ID = ITEM\_ID

WHERE Repository\_ID = 10340 AND MERGED\_INTO\_ITEM\_ID IS NULL;

/\* COMMIT; \*/

**Query session 2** check tempdb size, it will have no change

SELECT DB\_NAME(vsu.database\_id) AS DatabaseName,

vsu.reserved\_page\_count,

vsu.reserved\_space\_kb,

tu.total\_page\_count as tempdb\_pages,

vsu.reserved\_page\_count \* 100. / tu.total\_page\_count AS [Snapshot %],

tu.allocated\_extent\_page\_count \* 100. / tu.total\_page\_count AS [tempdb % used]

FROM sys.dm\_tran\_version\_store\_space\_usage vsu

CROSS JOIN tempdb.sys.dm\_db\_file\_space\_usage tu

WHERE vsu.database\_id = DB\_ID(DB\_NAME());

**Query session 3** will not return result until the session 1 commited ( Can not do concurrent read commit)

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

SELECT COUNT(\*) as cnt FROM B\_MASTER\_REPOSITORY\_ITEM

WHERE Repository\_ID = 10340 AND MERGED\_INTO\_ITEM\_ID IS NULL;

**Query session 4** return 0 .it is dirty read any way

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

SELECT COUNT(\*) as cnt FROM B\_MASTER\_REPOSITORY\_ITEM

WHERE Repository\_ID = 10340 AND MERGED\_INTO\_ITEM\_ID IS NULL;

**Go to query session 1**, execute

COMMIT;

Both query session 3 and 4 will return count: 0 as expected.