Enabling SQL Server Snapshot Isolation

Clients who have very high transaction volume might encounter some SQL deadlock issues. In these cases, Aptify recommends using snapshot isolation to significantly reduce or eliminate database blocking issues in high-volume situations. The purpose of this topic is to provide information about how to configure the SQL Server transaction isolation level.

Please read thoroughly

It's important to understand the implications of using this technique, which utilizes row-level isolation of data being updated in a transaction. One specific situation is inventory, where potentially stale inventory counts could be read if another transaction is in the process of updating that count. In the case of two transactions updating the same row at the same time, the second should fail and rollback, which would cause the transaction to have to be resubmitted. This technique should be tested thoroughly before being applied.

Understanding Snapshot Isolation Levels

READ_COMMITTED_SNAPSHOT vs ALLOW_SNAPSHOT_ISOLATION

We recommend setting the READ_COMMITTED_SNAPSHOT database option ON. If you set the READ_COMMITTED_SNAPSHOT database option to ON, the database engine uses row versioning and snapshot isolation as the default, instead of using locks to protect the data. This topic also discusses setting ALLOW_SNAPSHOT_ISOLATION database option ON. This isolation state can be used but is not required as per Aptify's analysis and testing.

SQL Server 2005 added two new isolation levels, Read Committed Snapshot and Snapshot, to enhance concurrency for OLTP applications. These isolation levels determine what locks SQL Server takes when accessing data and, therefore, by extension they determine the level of concurrency and consistency that statements and transactions experience. In earlier versions of SQL Server, concurrency was based solely on locking, which caused blocking and deadlocking problems for some applications. Snapshot isolation depends on enhancements to row versioning and is intended to improve performance by avoiding reader-writer blocking scenarios. All of these isolation levels are described in the following Microsoft Development Network article:

http://msdn2.microsoft.com/en-us/library/ms173763.aspx

The Snapshot Isolation Level and Read Committed Snapshot features improved SQL Server, providing optimistic locking as well as pessimistic locking. This is a feature that allows writers not to block readers, and readers will not block writers. Instead, the readers will look at the most recent row, and ignore the fact that it's currently being written to. This will help, in that long running transactions won't block other transactions.

Read Committed Snapshot is a modification to the Read Committed Isolation level that uses row versioning to read the previous value. Turn on Read Committed Snapshot, and a lot of your blocking will be a thing of the past. Once snapshot isolation is enabled, updated row versions for each transaction are maintained in **tempdb**. A unique transaction sequence number identifies each transaction, and these unique numbers are recorded for each row version. The transaction works with the most recent row versions having a sequence number before the sequence number of the transaction. Newer row versions created after the transaction has begun are ignored by the transaction.

You may want to consider moving tempdb to an isolated disk for performance purposes. You may also want to consider setting the initial size of tempdb to 2 GB.

The term "snapshot" reflects the fact that all queries in the transaction see the same version, or snapshot, of the database, based on the state of the database at the moment in time when the transaction begins. No locks are acquired on the underlying data rows or data pages in a snapshot transaction, which permits other transactions to execute without being blocked by a prior uncompleted transaction. Transactions that modify data do not block transactions that read data, and transactions that read data do not block transactions that write data, as they normally would under the default READ COMMITTED isolation level in SQL Server. This non-blocking behavior also significantly reduces the likelihood of deadlocks for complex transactions.

Snapshot isolation uses an optimistic concurrency model. If a snapshot transaction attempts to commit modifications to data that has changed since the transaction began, the transaction will roll back and an error will be raised. You can avoid this by using UPDLOCK hints for SELECT statements that access data to be modified. Search for "Locking Hints" for your version of SQL Server for more information.

Snapshot isolation must be enabled by setting the ALLOW_SNAPSHOT_ISOLATION ON database option before it is used in transactions. This activates the mechanism for storing row versions in the temporary database (**tempdb**). You must enable snapshot isolation in each database that uses it with the Transact-SQL ALTER DATABASE statement. In this respect, snapshot isolation differs from the traditional isolation levels of READ COMMITTED, REPEATABLE READ, SERIALIZABLE, and READ UNCOMMITTED, which require no configuration.

> Refer to the following Microsoft Development Network article for more information on understanding isolation:
>
> https://msdn.microsoft.com/en-us/library/ms345124.aspx#sql2k5snapshotisol_topic05
>
> In particular, review the section on Administrative Best Practices.

Configuring SQL Server Snapshot Isolation

Perform this procedure to enable SQL Server snapshot isolation. Ensure that you understand the information in the Understanding SQL Server Snapshot Isolation Levels before going ahead and enabling snapshot isolation.

1. Log into SQL server management studio with a sa-level user account.
2. Open a new query.
3. Run the following query to enable the READ_COMMITTED_SNAPSHOT option:

```
/** Run this statement to change the isolation level of the database.
When the READ_COMMITTED_SNAPSHOT database option is set ON, the mechanisms used to support the option are
activated immediately. When setting the READ_COMMITTED_SNAPSHOT option, only the connection executing the A
LTER DATABASE command is allowed in the database. There must be no other open connection in the database until
ALTER DATABASE is complete. The database does not have to be in single-user mode.
**/

ALTER DATABASE APTIFY SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
ALTER DATABASE Aptify SET READ_COMMITTED_SNAPSHOT ON;
ALTER DATABASE APTIFY SET MULTI_USER;
```

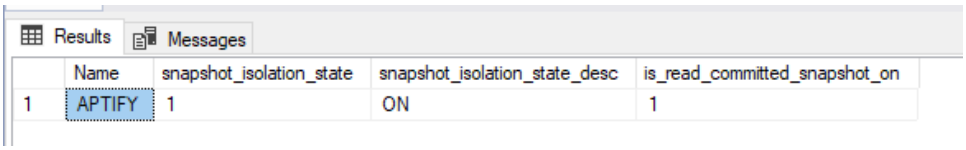
4. Run the following query to enable the ALLOW_SNAPSHOT_ISOLATION option:

```
/** Statement below can be used to turn on the snapshot isolation, so that transactions can set the
snapshot isolation level. Per our analysis, there is no Aptify code that currently does this, therefore
this is not required.**/

ALTER DATABASE Aptify SET ALLOW_SNAPSHOT_ISOLATION ON;
```

5. Run the following query to verify the values of these options:

```
/** Query to determine snapshot states **/
SELECT
Name
, snapshot_isolation_state
, snapshot_isolation_state_desc
, is_read_committed_snapshot_on
FROM sys.databases
WHERE name = 'APTIFY';
```



Results | Messages

	Name	snapshot_isolation_state	snapshot_isolation_state_desc	is_read_committed_snapshot_on
1	APTIFY	1	ON	1