Introduction

I want to show the use of With (NoLock) hit with a simple example but before that let's brush up following term.

**Lock**it allows different type of resource to be lock by the transaction.

When user accessing data, data get locks so that other user cannot modify or delete data that someone is reading.

A **transaction**is unit of work submitted as hole to database for processing.

**Dirty Read** is a process of reading database record without locking the record being read.

(Reading uncommitted DATA)

Stand the risk of reading an uncommitted row that is subsequently rolled back. Resulting in reading of local copy of record that is not consistent with what is stored in database.

**Locks Types:**

**Shared Lock(s)** is read lock it is occur when SQL server perform select statement or any other statement to read the data tell to read but do not modify the data.

When one user is reading the data he/she issue read lock so that other user do not modify data.

**Exclusive Lock (x)** are generally use during modification activity use to lock data being modified by one transaction.it prevent modification by another concurrent transaction.

**Update Lock (u)** update lock a mix of shared and exclusive lock.

Update Lock is kind of Exclusive Lock except it can be placed on the row which already has Shared Lock on it. Update Lock reads the data of row which has Shared Lock, as soon as Update Lock is ready to change the data it converts itself to Exclusive Lock.

**WITH (NOLOCK)** hit

Do not issue shared locks and do not honor exclusive locks. When this option is in effect, it is possible to read an uncommitted transaction or a set of pages that are rolled back in the middle of a read. Dirty reads are possible. Only applies to the SELECT statement.

Background

In the following example we are denoting two seperate user by opening two sperate query window in SQL Server. I am using Northwind database on both the windows. So sotry is like this

User one update table saying that Dairy Products are Chees and Paneer but he did not commit data.

User one weight for an approval.

User two fire Select qury same time but qury run infinite loop as select qury issue share lock that is not granted as User one is updating data.

Another user fire Select qury same time WITH (NOLOCK) hit and qury run to show uncommitted data as Diry product "Chees and Paneer".

As our user one approval is rejected and he roll back his update to Dairy product to Chees only.

Oh the user with nolock hit is still considering that Diry product contin the Paneer.

But he still avoid the situation of being dead lock like user one.

Using the code

Let's open query window one and run the update query as mention bellow.

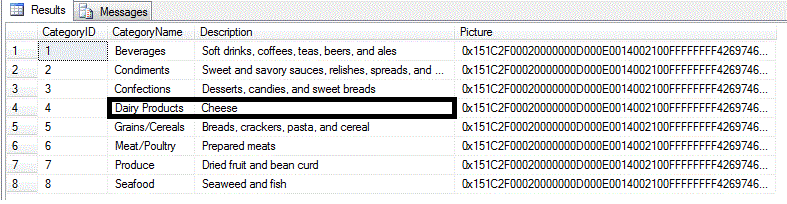
Hide   Copy Code

--Query window 1

USE Northwind

Hide   Copy Code

SELECT \* FROM Categories



Hide   Copy Code

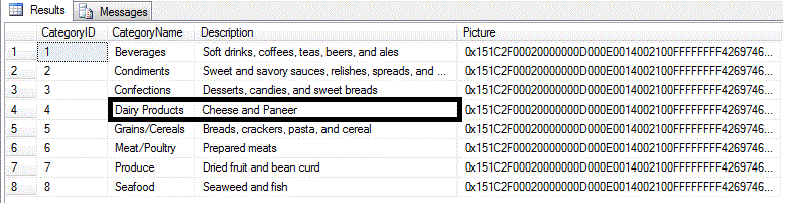
BEGIN TRANSACTION

UPDATE Categories SET Description='Cheese and Paneer'

WHERE CategoryName='Dairy Products'

Hide   Copy Code

SELECT \* FROM Categories



Open query window two and fire the same SELECT query.

Hide   Copy Code

*--Qury window 2*

USE Northwind

SELECT \* FROM Categories

I run the same query from above again and i have notice that it never completes, because the UPDATE has not yet been committed.

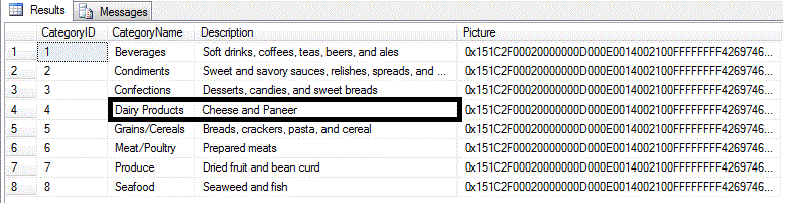
Open query window third and fire the same SELECT query WITH (NOLOCK) hit.

Hide   Copy Code

*--Qury window 3*

USE Northwind

SELECT \* FROM Categories WITH (NOLOCK)



Hide   Copy Code

*--Let me rollback the query in Window 1.*

ROLLBACK

SELECT \* FROM Categories

But our window  3 user is still considering that  Diary Products are Cheese and Paneer reading uncommitted data that is rolled back after time.

And our window 2 user is fall in to dead lock condition.

We can find the lock issue in each window using **sp\_lock**.

If we run our SELECT without NOLOCK we can see the locks that are taken if we use **sp\_lock**.

For window 2 while our query is in dead lock we can veryfy lock on the page by firing **sp\_lock**on separate window.

You can find that there is share lock issue when you run the select query with out (NOLOCK) and there is NO Share lock issue when you run the qury with (NOLOCK). Hence when you perform update in that is yet to commit exclusive lock is issue.

Where (NOLOCK) does not honor exclusive lock

Do not use WITH (NOLOCK) without fully understanding the ramifications of a dirty read

### Advantages:

* Reduces the locking which intends to less chance to block ,[Deadlocks](http://sqlserverplanet.com/optimization/understanding-sql-server-deadlocks) will not occur against other queries running against the same data
* Typically allows for much higher concurrency due to lower footprint
* Disadvantages:
* Uncommitted data can be read leading to dirty reads
* Explicit hints against a table are generally bad practice

More about Locks: http://www.mssqltips.com/sqlservertip/2470/understanding-the-sql-server-nolock-hint/

**Alternative for Nolock is Isolation Level**

# [SET TRANSACTION ISOLATION LEVEL](http://sqlserverplanet.com/optimization/set-transaction-isolation-level)

This statement is used to set the isolation level for either a connection or a stored procedure. The most typical use I’ve seen is at the top of a stored procedure in order to avoid locking and deadlocks. This is a cleaner alternative to using WITH (NOLOCK) hints on tables. If you set the isolation level within a procedure, the isolation level will revert to its previous level once the procedure finishes.

The syntax is:

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;

The available options are:

* READ UNCOMMITTED – Allows dirty reads, meaning data can be read from a transaction that is not yet complete.
* READ COMMITTED – Only reads data from transactions that have already been committed.  Current transactions must wait for transactions referencing common resources to finish.
* REPEATABLE READ – Data that is being read is exclusively locked until the transaction completes.
* SNAPSHOT – The default for Oracle.  This level allows all data to be read without using locks by maintaining a snapshot of all the data being modified in “versioning” tables.  This is the only isolation level not lock based.
* SERIALIZABLE – Data that is being read is exclusive locked and inserts are prevented within this range until the transaction completes.

|  |  |  |  |
| --- | --- | --- | --- |
| * Isolation level | * Dirty read | * Nonrepeatable read | * Phantom read |
| * Read uncommitted | * ✔ | * ✔ | * ✔ |
| * Read committed | * ✗ | * ✔ | * ✔ |
| * Repeatable read | * ✗ | * ✗ | * ✔ |
| * Serializable | * ✗ | * ✗ | * ✗ |
| * Snapshot | * ✗ | * ✗ | * ✗ |

<https://en.wikipedia.org/wiki/Isolation_(database_systems)>