**3 ways of doing Optimistic Locking in .NET**

<http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx>

Posted By:**[Shivprasad koirala](http://www.dotnetspark.com/Member/questpond.aspx)**       Posted Date: September 18, 2010    Points: 25    Category: [**.Net Framework**](http://www.dotnetspark.com/kb/index.aspx?cid=2)   URL: [http://www.questpond.com](http://www.questpond.com/)

In this article we will touch base 3 ways of doing optimistic locking i.e. Using ADO.NET dataset, SQL Server Timestamp check and old / new value check. So we will start this article with a small introduction to concurrency, discuss the 5 concurrency problems and then get down to the actual meat to understand how to implement optimistic locking using the 3 methodologies.

|  |  |
| --- | --- |
| [Developer Express Ad](http://engine.adzerk.net/r?e=eyJhdiI6NTU0MywiYXQiOjUsImNtIjoxODcwNCwiY2giOjQ4MTEsImNyIjo0ODI1NywiZGkiOiI2ZTVjODU3MDRjNDM0OWIxYWJlNTkwMjVjMTAwZGJlNiIsImRtIjoxLCJmYyI6MTExMTE5LCJmbCI6NTQ1MDQsImt3IjoidW5kZWZpbmVkIiwibnciOjIwNzIsInJmIjoiaHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS5iZC8iLCJydiI6MCwicHIiOjc3NjAsInN0IjoxNzM1NCwidXIiOiJodHRwczovL3d3dy5kZXZleHByZXNzLmNvbS9nby9EWC1ORC1EWFAtMTItMi5hc3B4In0&s=g3WNo4KsQbTFwPT944TDopaUTpA) | [Pluralsight ad](http://engine.adzerk.net/r?e=eyJhdiI6MjM3MDksImF0Ijo1LCJjbSI6MzI5MTgsImNoIjo0ODExLCJjciI6ODAyNTgsImRpIjoiODdmYzgzNmJjYzVlNGY3NTkyMDYwOWI4ZGJkMDE0ZDgiLCJkbSI6MSwiZmMiOjExMTY1MiwiZmwiOjU0Njk3LCJrdyI6InVuZGVmaW5lZCIsIm53IjoyMDcyLCJyZiI6Imh0dHBzOi8vd3d3Lmdvb2dsZS5jb20uYmQvIiwicnYiOjAsInByIjo3NzYwLCJzdCI6MTczNTQsInVyIjoiaHR0cHM6Ly9wbHVyYWxzaWdodC5jb20vdHJhaW5pbmcvU3Vic2NyaWJlL1N0ZXAxP2lzVHJpYWw9VHJ1ZSZ1dG1fc291cmNlPW5ldWRlc2ljJnV0bV9tZWRpdW09MzAweDI1MCZ1dG1fY29udGVudD1mcmVlLXRyaWFsJnV0bV9jYW1wYWlnbj1zdW1tZXIyMDEzIn0&s=-3_Vhetrb97rSICfHxY2jQ6-JaM) |

**3 ways of doing Optimistic Locking in .NET**

[Introduction](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#Introduction)

[Why do we need locking?](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#Why%20do%20we%20need%20locking)

[How can we solve concurrency problems?](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#How%20can%20we%20solve%20concurrency%20problems)

[What kind of confusion is caused because of concurrency?](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#What%20kind%20of%20confusion%20is%20caused%20because%20of%20concurrency)

[So how can we solve the above problems?](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#So%20how%20can%20we%20solve%20the%20above%20problems)

[What is Optimistic locking?](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#What%20is%20Optimistic%20locking)

[How does optimistic lock work?](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#How%20does%20optimistic%20lock%20work)

[What are the different solutions by which we can implement optimistic locking?](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#What%20are%20the%20different%20solutions%20by%20which%20we%20can%20implement%20optimistic%20locking)

[Solution number 1:- Datasets](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#Solution%20number%201:-%20Datasets)

[Solution number 2:- Use timestamp data type](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#Solution%20number%202:-%20Use%20timestamp%20data%20type)

[Solution number 3:- Check old values and new values](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#Solution%20number%203:-%20Check%20old%20values%20and%20new%20values)

[Source code](http://www.dotnetspark.com/kb/2651-3-ways-doing-optimistic-locking-net.aspx#Source%20code)

**Introduction**

In this article we will touch base 3 ways of doing optimistic locking i.e. Using ADO.NET dataset, SQL Server Timestamp check and old / new value check. So we will start this article with a small introduction to concurrency, discuss the 5 concurrency problems and then get down to the actual meat to understand how to implement optimistic locking using the 3 methodologies.

This is a small Ebook for all my .NET friends which covers topics like WCF,WPF,WWF,Ajax,Core .NET,SQL etc you can download the same from [here](http://tinyurl.com/4nvp9t)  or else you can catch me on my daily free training @ from [here](http://tinyurl.com/257szt5)

**Why do we need locking?**

In multi-user environment it's possible that multiple users can update the same record at the same time causing confusion between users. This issue is termed as concurrency.

**How can we solve concurrency problems?**

Concurrency problems can be solved by implementing proper "Locking strategy". Locks prevent action on a resource to be performed when some other resource is already performing some action on it.

**What kind of confusion is caused because of concurrency?**

There are 4 kinds of major problems caused because of concurrency, below table shows the details of the same.

|  |  |  |
| --- | --- | --- |
| **Problems** | **Short description** | **Explanation** |
| Dirty reads | "Dirty Read" occurs when one transaction is reading a record, which is part of a half, finished work of other transaction. | . User A and user B are seeing value as "5". . User B changes the value "5" to "2". . User A is still seeing the value as "5"..Dirty read has happened. |
| Unrepeatable read | In every data read if you get different values then it's an "Unrepeatable Read" problem. | . User A is seeing value as "5". . User B changes the value"5" to "2". . User A refreshes see values "5", he is surprised..unrepeatable read has happened. |
| Phantom rows | If "UPDATE" and "DELETE" SQL statements does not affect the data then it can be "Phantom Rows" problem. | . User A updates all value "5' to "2". . User B inserts a new record with value "2". . User A selects all record with value "2' if all the values have changed, he is surprised to still find value "2" records...Phantom rows have been inserted. |
| Lost updates | "Lost Updates" are scenario where one updates which is successfully written to database is overwritten with other updates of other transaction. | . User A updates all value form "5" to "2". . User B comes and updates all "2" values to "5". . User A has lost all his updates. |

**So how can we solve the above problems?**

By using optimistic or pessimistic locking, the further coming article discusses the same.

**What is Optimistic locking?**

http://www.dotnetspark.com/kbPicture/questpond_634203811505540000_lock.JPG

As the name suggests "optimistic" it assumes that multiple transaction work without affecting each other. In other words no locks are enforced while doing optimistic locking. The transaction just verifies that no other transaction has modified the data. In case of modification the transaction is rolled back.

**How does optimistic lock work?**

You can implement optimistic locking by numerous ways but the fundamental to implement optimistic locking remains same. It's a 5 step process as shown below:-

. Record the current timestamp.  
. Start changing the values.  
. Before updating check whether anyone else has changed the values by checking the old time stamp and new time stamp.  
. If it's not equal rollback or else commit.

http://www.dotnetspark.com/kbPicture/questpond_634203811586910000_1.JPG

**What are the different solutions by which we can implement optimistic locking?**

There are 3 primary ways by which we can implement optimistic locking in .NET:-

. **Datasets**: - Dataset by default implement optimistic locking. They do a check of old values and new values before updating

. **Timestamp Data type**: - Create a timestamp data type in your table and while updating check if old timestamp is equal to new timestamp.

. **Check old and new value**: - Fetch the values, do the changes and while doing the final updates check if the old value and current values in database are equal. If they are not equal then rollback or else commit the values.

**Solution number 1:- Datasets**

As said in the previous section dataset handles optimistic concurrency by itself. Below is a simple snapshot where we held the debug point on Adapter's update function and then changed the value from the SQL Server. When we ran the "update" function by removing the break point it threw "Concurrency" exception error as shown below. 

http://www.dotnetspark.com/kbPicture/questpond_634203811930020000_2.JPG

If you run the profiler at the back end you can see it fires the update statement checking of the current values and the old values are same.

exec sp\_executesql N'UPDATE [tbl\_items] SET [AuthorName] = @p1 WHERE (([Id] = @p2) AND ((@p3 = 1 AND [ItemName] IS NULL) OR ([ItemName] = @p4)) AND ((@p5 = 1 AND [Type] IS NULL)   
OR ([Type] = @p6)) AND ((@p7 = 1 AND [AuthorName] IS NULL) OR ([AuthorName] = @p8)) AND ((@p9 = 1 AND [Vendor] IS NULL) OR ([Vendor] = @p10)))',N'@p1 nvarchar(11),@p2 int,@p3   
int,@p4 nvarchar(4),@p5 int,@p6 int,@p7 int,@p8 nvarchar(18),@p9 int,@p10 nvarchar(2)',@p1=N'this is new',@p2=2,@p3=0,@p4=N'1001',@p5=0,@p6=3,@p7=0,@p8=N'This is Old   
Author',@p9=0,@p10=N'kk'

In this scenario we were trying to change the field value "AuthorName" to "This is new" but while updating it makes a check with the old value "This is old author". Below is the downsized code snippet of the above SQL which shows the comparison with old value.

,@p8=N'This is Old Author'

**Solution number 2:- Use timestamp data type**

The other way of doing optimistic locking is by using 'TimeStamp' data type of SQL Server. Time stamp automatically generates a unique binary number every time you update the SQL Server data. Time stamp data types are for versioning your record updates. 

http://www.dotnetspark.com/kbPicture/questpond_634203812112640000_3.JPG

To implement optimistic locking we first fetch the old 'TimeStamp' value and when we are trying to update we check if the old time stamp is equal to the current time stamp as shown in the below code snippet.

update tbl\_items set itemname=@itemname where CurrentTimestamp=@OldTimeStamp

We then check if any updates has happened, in case updates has not happened then we raise a serious error '16' using SQL Server 'raiserror' as shown in the below code snippet.

if(@@rowcount=0)

begin

raiserror('Hello some else changed the value',16,10)

end

If any concurrency violation takes place you should see the error propagated when you call 'ExecuteNonQuery' to the client side as shown in the below figure. 

http://www.dotnetspark.com/kbPicture/questpond_634203812322640000_4.JPG

**Solution number 3:- Check old values and new values**

Many times we would like to check concurrency on only certain fields and omit fields like identity etc. For those kind of scenarios we can check the old value and the new value of the updated fields as shown in the below code snippet.