

Re: Multiuser Problem

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.vb/2004-04/2670.html>

From: Gary Hunt (*betaplace_at_codequest.co.uk*)

Date: 04/13/04

Date: Tue, 13 Apr 2004 17:32:52 +0100

Prabhat,

Database locking:

1. Pessimistic locking – in today's modern languages this is not really the way to go and implementing it is getting harder. Pessimistic locking requires that you have a database transaction locking the row of the record that the user is editing preventing any other user from editing that row.
2. Optimistic locking is the way to go but requires that you check to see if the record has changed since you allowed the user to edit the row.
3. No locking is common amongst developers and leads to business rule chaos. You must check that the rows you are about to update are the same ones that you or your user thought they were changing.

Some examples.

To perform pessimistic locking with ADO.Net is gonna be really hard as you don't know if the user will ever return back to the code that caused the lock to be put on in the first place. I'll do a quick pseudo code Forms app to show you how you'd do it – but like I said above – don't do this it's not good practice!

Protected _cn as SqlConnection

Protected _tr as SQLTransaction

Sub cmdEditRow()

 cn = New SqlConnection(.... fill in the details here.....)

 tr = cn.BeginTrans

 Dim cm As New SqlCommand("SELECT * FROM row WHERE id=10",cn, tr)

 cm.ExecuteNonQuery

 cm.Dispose

End Sub

```
Sub cmdUpdateRow()  
    Dim cm as New SqlCommand("UPDATE row SET field = new_value WHERE id=10",  
cn, tr)  
    cm.ExecuteNonQuery  
    tr.Commit  
    cm.Dispose()  
    _tr.Dispose()  
    _cn.Dispose()  
    _tr = Nothing  
    _cn = Nothing  
End sub
```

So what you'd have to do is call cmdEditRow before presenting the data to the user and because the database transaction is left un-committed the row will be left locked. Any attempt to play with that row in the database will timeout as there is a database lock on that row – e.g. a bad situation to be in.

The once the user has changed what he/she wants and clicked Update or whatever you the call cmdUpdateRow which, updates the rows with the new values. Because this is within the same transaction this will be allowed and after updating the row the transaction is committed and the member variables cleared as there is no longer a transaction in place.

As you can see this is not a good way to write software and you should consider the next option.

ADO.Net helps out a lot with the chore of writing opportunistic check code.

The theory is that you get the values from the row and present them to the user. Allow the user to edit the row but before you write the data back to the row you ensure that nothing has changed in the row. The CommandBuilder classes create update statements that check the before values from within the where clause and generate a DataConcurrencyException if the old values are not the same as when the row was first read. You'll need to ensure that if you are using ASP.Net that you persist the DataSet / DataTable. A common mistake is to present the user with the values on one HTML page. Allow the user to edit and then post back the results to the page. This page goes and gets the values to compare with – but they might not be the same as when the user first saw the page – you've no longer got a locking strategy.

If you want to get cute and program like a pro then you end up doing the following. Rather than compare the fields of the row you can use a "timestamp" column. This column is guaranteed to change whenever any of the other fields change in the row – so if the timestamp column has changed then you know that the row has changed since you last looked.

I'll show you an example that uses timestamps and datatables. Because this is only doing a single read there is no need for a database transaction as such but it would be good practice to use them throughout.

So some code:

```
Sub Page_Load()
    ' first call into the page so lets display the data
    If Not IsPostBack Then
        Dim cn As new SqlConnection(.....)
        Dim da as new SqlDataAdapter("SELECT * FROM row WHERE id = 10")
        Dim dt as new DataTable
        da.Fill(dt)
        Dim dr as DataRow = dt.Rows(0)
        ViewState.Add("ts", dr.Item("ts").ToString)
        txtCustomerName.Text = dr.Item("name").ToString
        dt.Dispose()
        da.Dispose()
        cn.Dispose()
    End If
End Sub
```

All of the above should be inside a try..catch..finally to ensure that the Connection object is closed/disposed if things fail.

You'll note that I've added the original timestamp column "ts" into the viewstate so that I can retrieve it at a later date – such as in the routine below which involves the save.

```
Sub cmdSave_Click( ByVal sender as Object, ByVal args As System.EventArgs )
    ' this would be called after a click on the page indicating that the
    user wants to save
    Dim cn as New SqlConnection(.....)
    Dim tr as SQLTransaction = cn.BeginTrans
    Dim cmTS as New SqlCommand("SELECT ts FROM row WHERE id = 10", cn, tr)
    Dim ts As Object = cmTS.ExecuteScalar
    If (ts.ToString <> DirectCast(ViewState.Item("ts"), String) Then
        ' someone else has edited this row so show an error
        tr.Rollback
    Else
        Dim cm as New SqlCommand("UPDATE row SET name = @name WHERE id =
10", cn, tr)
        cm.Parameters.Add("@name", SqlDbType.String).Value =
txtCustomerName.Text
        cm.ExecuteNonQuery
        tr.Commit
        ' update the row
    End If
```

The key to the above is the check to ensure that the "ts" field (which is a timestamp column) hasn't changed. If it has then somebody else has edited the row. You'll also note that the update is within a transaction as there is more than one call being made.

I hope that helps – the code is straight from my head so give it a go and expect to make a few changes to make it work.

cheers,

g

"Prabhat" <not_a_mail@hotmail.com> wrote in message
news:uAjWwAXIEHA.2524@TK2MSFTNGP11.phx.gbl...

> *How do I lock a particular record that one user has opened for editing?*

>

> *If I use the pessimistic type, can other users view the record (but not
> edit it) and return a message telling that another person is editing the
> record, or does this type lock the record such that it is unavailable
> until the editor releases it?*

>

> *An explanation of pessimistic and optimistic lock types would be really
> useful, as would some example code.*

>

> *My project is to maintenance system with multiple users who can
> all edit every record. However, I need to control the edit function so
> that nobody edits a record that is in the process of being edited by
> someone else. Also I Don't want to to use a Database Column for FLAG to
> Maintain the Edit Mode.*

>

> *My Database is SQL Server with VB.NET with ADO.NET (or Lagecy ADO) is the
> Programming TOOL.*

>

> *Any ideas?*

> *Thanks*

> *Prabhat*

>

>