

Re: Reapplying successfull batch updates

Source: <http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.database.ado/2005-02/0082.html>

From: Dave Blake (barnswood_at_hotmail.com)

Date: 02/04/05

Date: Fri, 4 Feb 2005 19:30:39 -0000

Brian, thanks for the SaveToStream suggestion.

- > *The behavior you site above is not quite what I observer.*
- > *A cloned TadoDataset keeps its own set of updated*
- > *values after a batchUpdate and RecordStatus reflects that there are still*
- > *records to be update. However the oldValue of fields reverts to the*
- > *current value after a batchupdate to the source Dataset.*

Having you and Del insist that clones are more independant than my experiences has made me curious (and persistent). I have knocked up some simple code that shows the lack of independence (and that what you say doesn't happen). It uses ADO/ADOX straight (no 3rd-party /ADOExpress in case that contributes, nor data linked components) so is a bit ugly but stands alone. All you need is a form with two buttons and a TMemo, and the code below.

It shows that a) edits to the original dataset are visible from the clone, b) after UpdateBatch records in clone have same changed status as the original and c) neither have any updates pending.

Now maybe a TAdoDataset does something different, and if it does I would love to know why.

Meanwhile, it seems a bit perverse to have to save/load from file when after a rollback the local recordset has the correct values but just does not know that it has to send them to the server. It means that the user can see the values on screen, but if they don't edit them again they won't get saved. I'm thinking that since I have the values then all I have to do is loop through all the field values setting them to themselves, of course there is no way to differentiate between the ones that have changed (since the user started editing) and the ones that are the same, but then update the whole lot – that's what the save/load from stream is doing isn't it?

Many thanks for your attention

Dave

uses ADODB_TLB, ADOX_TLB, OleCtrls;

```
procedure CreateWidgetTable(strDBPath: String);
var
  catDB: Catalog;
  tblNew: Table;
  AdoRS: Recordset;
  conn: Connection;
  i: Integer;
begin
  try
    catDB := CoCatalog.Create;
    catDB.Create('Provider=Microsoft.Jet.OLEDB.4.0;Data Source=' +
strDBPath);
    tblNew := CoTable.Create;
    with tblNew do begin
      Name := 'Widget';
      with Columns do begin
        Append('ID', AdInteger, 0);
        Append('Desc1', adVarChar, 30);
        Append('Desc2', adVarChar, 30);
        Append('Desc3', adVarChar, 30);
      end;
    end;
    catDB.Tables.Append(tblNew);

    conn := CoConnection.Create;
    conn.ConnectionString := 'Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
+ strDBPath;
    conn.CursorLocation := adUseClient;
    conn.Open(conn.ConnectionString, 'Admin', '', 0);

    AdoRS := CoRecordset.Create;
    AdoRS.CursorLocation := adUseServer;
    AdoRS.Open('SELECT * FROM Widget', conn, adOpenKeyset, adLockOptimistic,
0);
    with AdoRS do begin
      for i := 1 to 5 do begin
        AddNew(Emptyparam, EmptyParam);
        Fields[0].Value := i;
        Fields[1].Value := 'Blah ' + inttostr(i);
        Fields[2].Value := 'Pop ' + inttostr(i);
        Fields[3].Value := inttostr(i*1000);
        Update(Emptyparam, EmptyParam);
      end;
    end;
  finally
    AdoRS.Close;
    AdoRS := nil;
    conn.Close;
    conn := nil;
    catDB := nil;
  end;
end;
```

end;

```
function GetUpdateCount(dt: Recordset): integer;
```

```
var
```

```
    CloneRS: Recordset;
```

```
begin
```

```
    CloneRS := dt.Clone(adLockUnspecified);
```

```
    try
```

```
        CloneRS.Filter := adFilterPendingRecords;
```

```
        Result := CloneRS.RecordCount;
```

```
    finally
```

```
        CloneRS.Close;
```

```
        CloneRS := nil;
```

```
    end;
```

```
end;
```

```
procedure TForm1.EditWidgetTable(strDBPath: String);
```

```
var
```

```
    RS, cloneRS: Recordset;
```

```
    conn: Connection;
```

```
    i: Integer;
```

```
procedure DisplayValues;
```

```
var
```

```
    i: Integer;
```

```
begin
```

```
    with memo1 do begin
```

```
        with RS do begin
```

```
            Lines.Add('Original');
```

```
            MoveFirst;
```

```
            while not EOF do begin
```

```
                for i := 0 to 3 do
```

```
                    Lines.Add(Format(' %s %s %s', [Fields[i].Value,
```

```
Fields[i].originalValue, Fields[i].UnderlyingValue]));
```

```
                    Lines.Add(Format('Record Status = %x ', [status]));
```

```
                    movenext;
```

```
                end;
```

```
            end;
```

```
            with CloneRS do begin
```

```
                Lines.Add('CLONE');
```

```
                MoveFirst;
```

```
                while not EOF do begin
```

```
                    for i := 0 to 3 do
```

```
                        Lines.Add(Format(' %s %s %s', [Fields[i].Value,
```

```
Fields[i].originalValue, Fields[i].UnderlyingValue]));
```

```
                        Lines.Add(Format('Record Status = %x ', [status]));
```

```
                        movenext;
```

```
                    end;
```

```
                end;
```

```
                Lines.Add(format('Pending org = %d clone = %d',[GetUpdateCount(RS),  
GetUpdateCount(CloneRS)]));
```

```
    Lines.Add("");
end;
end;

begin
try
    conn := CoConnection.Create;
    conn.ConnectionString := 'Provider=Microsoft.Jet.OLEDB.4.0;Data Source='
+ strDBPath;
    conn.CursorLocation := adUseClient;
    conn.Open(conn.ConnectionString, 'Admin', "", 0);

    RS := CoRecordset.Create;
    with RS do begin
        CursorLocation := adUseClient;
        Open('SELECT * FROM Widget', conn, adOpenKeyset,
adLockBatchOptimistic, 0);
        CloneRS := Clone(adLockUnspecified); //clone before edits
        MoveFirst;
        for i := 1 to 3 do begin //Edit 3 records
            Fields[1].Value := 'EDITED ' + Fields[1].Value;
            Fields[2].Value := 'Hip ' + inttostr(i);
            Fields[3].Value := inttostr(i*1000+1);
            movenext;
        end;
    end;
// CloneRS := rs.Clone(adLockUnspecified); // Or after

    DisplayValues;
    memo1.Lines.Add('***Update Batch');
    conn.begintrans;
    RS.UpdateBatch(adAffectAll);
    conn.RollbackTrans; //Data rolledback on server, still local
    DisplayValues;

finally
    if rs.state = adStateOpen then
        RS.Close;
    RS := nil;
    if Cloners.state = adStateOpen then
        CloneRS.Close;
    CloneRS := nil;
    conn.Close;
    conn := nil;
end;
end;

procedure TForm1.MakeDBBtnClick(Sender: TObject);
begin
    CreateWidgetTable(ExtractFilePath(ParamStr(0)) + 'test2.mdb');
end;
```

borland.public.delphi.database.ado: Re: Reapplying successfull batch updates

```
procedure TForm1.EditBtnClick(Sender: TObject);  
begin  
  EditWidgetTable(ExtractFilePath(ParamStr(0)) + 'test2.mdb');  
end;
```

Click on MakeDB button, click on Edit button, look through the output – sorry for length of this post!