

Performance Problem Using ADO and Stored Procs

Source: <http://www.tech-archive.net/Archive/Data/microsoft.public.data.ado/2005-01/0360.html>

From: MichaelPyles (*MichaelPyles_at_discussions.microsoft.com*)

Date: 01/27/05

Date: Thu, 27 Jan 2005 07:41:03 -0800

I have a stored proc that executes in < 15 seconds through Query Analyzer. This proc updates rows in a table, inserts rows in another table, then returns the new rows via a select statement. If I execute this proc programmatically via ADO, it works fine when the number of rows updated and inserted are around 25000 or less. However, with 32000 rows, SQL server seems to become extremely slow, taking more than one hour to execute the proc (I didn't wait for it to complete). While the proc was executing, there was no CPU activity on the server hosting my VB app, but there was a HUGE amount of CPU activity on the server hosting SQL server for the SQL server process. Also, there were no page faults on the SQL server box and consistently about 300 I/O operations per second.

In addition to returning rows, the proc has two output values and a return value. Contrary to popular belief, the return value and output values CAN be accessed before iterating through all the rows if you use a client-side cursor because all the rows are returned to the client (See "HitchHiker's Guide To Visual Basic & SQL Server, 6th ed., page 765). Is there any workaround without a massive rewrite? The proc is OK. It's VB and ADO that's the problem. I'm suspicious it has something to do with the client-side cursor, but I need this to get my return and output values BEFORE iterating through the rows.

Here's my VB code. The ADO connection has already been established.

```
' Exec stored proc that returns output value
' In: vntDB - ADO connection object
' strProcName - name of calling procedure
' objParameters - Scripting.Dictionary of parameters (name is key)
' objReturnValues - Scripting.Dictionary of return value (key is
RETURN_VALUE)
' rsResultSet - ADODB.RecordSet of results returned from stored proc
(optional)
Private Function RunProc(vntDB As Variant, strProcName As String,
objParameters As Variant, objReturnValues As Variant, Optional rsResultSet As
Variant) As Boolean
    On Error GoTo err_handler

    ' allocate dictionary to contain return value and/or error info
```

microsoft.public.data.ado: Performance Problem Using ADO and Stored Procs

```
Set objReturnValues = CreateObject("Scripting.Dictionary")
objReturnValues.Add "ErrNo", ""
objReturnValues.Add "ErrDescription", ""

' set up the command object that we will build from each parameter
Dim adoCmd As ADODB.Command
Set adoCmd = CreateObject("ADODB.Command")
adoCmd.ActiveConnection = vntDB
adoCmd.CommandText = strProcName
adoCmd.CommandType = adCmdStoredProc
adoCmd.CommandTimeout = 30 * 60 ' 30 minutes

' get the parameter info
adoCmd.Parameters.Refresh

If IsObject(objParameters) Then
    Dim objParameter As ADODB.Parameter
    For Each objParameter In adoCmd.Parameters
        ' each name is preceded by the @ sign – ignore it for comparisons
        Dim strName As String
        strName = Mid(objParameter.Name, 2)
        If objParameters.Exists(strName) Then
            On Error Resume Next
            objParameter.Value = objParameters(strName)
        End If
    Next
End If

Dim rs As Recordset
Set rs = New Recordset
rs.CursorLocation = adUseClient
Set rs.Source = adoCmd
rs.Open

If Not IsMissing(rsResultSet) Then
    If rs.State = adStateClosed Then
        rsResultSet = Empty
    Else
        Set rsResultSet = rs
    End If
End If

' put the return value into the output dictionary
Dim objParm As ADODB.Parameter
For Each objParm In rs.ActiveCommand.Parameters
    Select Case objParm.Direction
        Case ADODB.adParamOutput, ADODB.adParamReturnValue,
ADODB.adParamInputOutput
            Select Case objParm.Name
                Case "@ErrNo"
                    objReturnValues("ErrNo") = objParm.Value
```

microsoft.public.data.ado: Performance Problem Using ADO and Stored Procs

```
Case "@ErrDescription"  
    objReturnValues("ErrDescription") = objParm.Value  
Case Else  
    objReturnValues.Add Mid(objParm.Name, 2),  
objParm.Value  
    End Select  
End Select  
Next  
  
RunProc = True  
  
err_handler:
```