Home **Articles** Questions & Answers Learning Zones Features Help! The Lounge Search

» Database » Database » SQL Server

VBScript to back up SQL Server data to pure SQL

By **alex turner** | 7 May 2008

Licence	BSD
First Posted	**7 May 2008**
Views	**23,467**
Bookmarked	**21 times**
Tags	VBScript, Windows, DBA, Dev, ADO, +

A VBScript which creates Delete and Insert statements to backup data from SQL Server.

Article **Browse Code** **Stats** **Revisions (2)** ☆☆☆☆☆ 4.85 (8 votes) 18

Download source code - 5.63 KB

Introduction

Repeatedly, over the last 8 years, I have needed to do the same thing - back up SQL Server (actually - it started with Sybase 11) to pure SQL. I started out using Perl - but here is a pure VBScript evolution of the technique I devised.

Let us get one thing straight here - this is not an official technique. If you are administering a SQL Server database and you use this technique - you are totally on your own. Yes - I know there are all sorts of official tools like bulk copy and data translation services. However, this technique is useful sometimes, and is flexible.

The idea is simple. Create a single SQL script that removes all the data from a database and then restores it all to a pre-recorded state. This boils down to a whole load of `DELETE` and `INSERT` statements. It is something that is done quite a bit in the MySQL world, but I have not seen it done in SQL Server land apart from by a loony like me!

So - what sort of thing do I use this for?

- Backing all the data up out of a SQL 2005 database and inserting it into a SQL Server 2000 database of the same structure.
- Backing up all the data from a test database, updating the database image from CVS (to get the latest sproc changes), and then resetting the data to that from the old image. This is kind of the whole point - this technique backs up the data, *not* the structure.
- Storing database data in CVS - the output of the script is text SQL.
- Backup and restore with extra logic (you can fiddle with the script to put COVERTS etc. to mess with data types and stuff).
- Overnight backup over a slow network. This one is risky - but as the SQL file compresses really small, it is sometimes an alternative to trying to back up to binary. I would not recommend it for a production system.
- Sending people the data from a database via email or FTP without them having to restore a binary image. You just say - unzip and run this script.
- I am sure there are others - but I have long since forgotten.

The Code

So here it is! I am 100% certain that there are many situations in which it will break. If you find one, let me know. If you find solutions, please let me know! *I have done a walk through below of how it works.*

⊟ Collapse

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Copyright (c) 2008, Dr Alexander J. Turner
' All rights reserved.
'
' Redistribution and use in source and binary forms, with or without
' modification, are permitted provided that the following conditions are met:
'     ' Redistributions of source code must retain the above copyright
'       notice, this list of conditions and the following disclaimer.
'     ' Redistributions in binary form must reproduce the above copyright
'       notice, this list of conditions and the following disclaimer in the
'       documentation and/or other materials provided with the distribution.
'     ' Neither the name of the  nor the
'       names of its contributors may be used to endorse or promote products
'       derived from this software without specific prior written permission.
'
' THIS SOFTWARE IS PROVIDED BY  ``AS IS'' AND ANY
```

See Also

- Articles like this
- Articles by this author

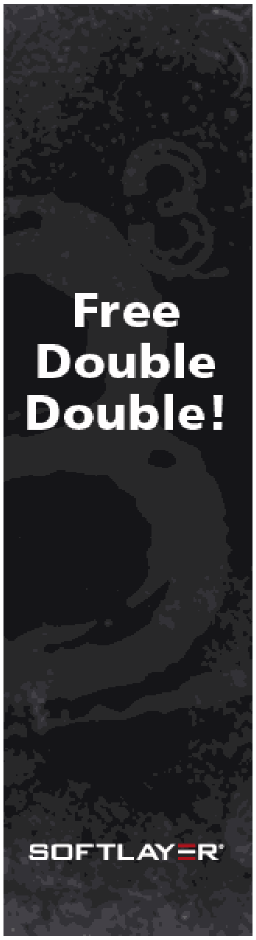
Sponsored Links

See Also...

Announcements

The Daily Insider

```
' EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
' WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
' DISCLAIMED. IN NO EVENT SHALL  BE LIABLE FOR ANY
' DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
' (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
' LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
' ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
' (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
' SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Option Explicit

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Configure this script then run from CMD using cscript
'
' Use the constants below to configure the script
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Database server name
Const server     = "localhost"
' Use trusted (windows authenitcation) or standard (SQL Server authentication)
Const trusted    = FALSE
' Database user name - not needed for trusted connection
Const userId     = "deployview"
' Database password  - not needed for trusted connection
Const password   = "deployview"
' Database
Const dataBase   = "dv"
' Set to true to create a unicode SQL File (safest)
' and false for an asci one (asci will loose data if you have
' unicode fields in the db)
Const useUnicode = TRUE
' Set the name of the created file
Const fileName   = "Data.sql"

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' DO NOT EDIT BELOW THIS LINE UNLESS YOU WANT TO ENCHANCE/CHANGE
' THE FUNCTIONALLITY OF THE SCRIPT
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Variables used in the script
Dim db,i,connectString,fields,rs

' Userful ADODB constants
Const adOpenStatic          = 3
Const adLockReadOnly        = 1
Const adCmdText             = 1
Const adUseClient           = 3
Const adLockBatchOptimistic = 4

' SQL that is used to get important info
Dim GetTriggers,GetUserTables_SQLServer,GetKeyOrder
GetTriggers = "SELECT spus.name + '.' + sp.name, s.name FROM " & _
" sysobjects s inner join sysobjects" & _
" sp on s.parent_obj = sp.id inner join sysusers spus on sp.uid = spus.uid  " & _
" WHERE s.xtype='TR' AND OBJECTPROPERTY(s.[id], 'ExecIsTriggerDisabled')=0"

GetUserTables_SQLServer = "SELECT usrs.name + '.' + obs.name 'Full Name' FROM " & _
" sysobjects obs, sysusers usrs WHERE obs.xtype = 'U' AND obs.uid = usrs.uid "

GetKeyOrder = "SELECT usrs1.name + '.' + o.name , usrs2.name + '.' + oo.name FROM " & _
" sysobjects o, sysforeignkeys f ,sysobjects oo,sysusers usrs1,sysusers usrs2 " & _
" WHERE o.id = f.rkeyid AND oo.id = f.fkeyid AND usrs1.uid=o.uid AND usrs2.uid=oo.uid"

' Connect to the db
If trusted Then
   connectString="Provider=SQLNCLI;Server=" & server & _
";Database=" & dataBase & ";Trusted_Connection=yes;"
Else
   connectString="Provider=SQLNCLI;Server=" & server & _
";Database=" & dataBase & ";Uid=" & _
    userId & ";Pwd=" & password & ";"
End If

Set db = CreateObject("ADODB.Connection")
db.Open connectString
db.Execute "USE " + dataBase

DumpDBDataToFile db,fileName,GetUserTables(db),dataBase,useUnicode
WScript.Echo "All done"
WScript.Quit
```



```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Pass in a connection and an array of table names
' and it will sort the tables names into dependency order.
' IE if table B depends on table A then A will be earlier in
' the list than B. Again, if B m->1 A, then A comes first
' in the list.
'
' Author: Alexander J Turner - 1 May 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub SortDepOrder(ado,tables)
   Dim recset
   Set recset = GetDisconRS(ado,GetKeyOrder)
   Dim inpa
   Dim rc
   Dim i
   i = 0
   rc = recset.RecordCount
   Dim pc()
   ReDim pc(rc, 2)
   recset.MoveFirst
   While Not recset.EOF
       pc(i, 0) = recset.fields(0)
       pc(i, 1) = recset.fields(1)
       recset.MoveNext
       i = i + 1
   Wend
   recset.Close
   Dim cnt
   cnt = True
   ' Keep sorting until no changes are made
   While cnt
       cnt = False
       Dim cfind
       ' scan over all elements
       For cfind = 0 To ubound(tables)
           Dim child
           child = tables(cfind)
           ' see if the current element is a reference child
           For i = 0 To rc
               ' if we find a child find the parent
               If pc(i, 1) = child Then
                   ' found child
                   ' so get parent
                   Dim prnt
                   prnt = pc(i, 0)
                   Dim pfind
                   ' loop over the whole input looking for the parent
                   For pfind = 0 To ubound(tables)
                       ' if we find it
                       If tables(pfind) = prnt Then
                           ' and it is after the child, swap
                           If pfind > cfind Then
                               ' parent lower than child swap
                               Dim tmp
                               tmp = tables(pfind)
                               tables(pfind) = tables(cfind)
                               tables(cfind) = tmp
                               WScript.Echo tables(pfind) & " X " & tables(cfind)
                               cnt = True
                           End If
                       End If
                   Next
               End If
           Next
       Next
   Wend
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Pass an database connection and get an array of all the user
' tables
'
' Author: Alexander J Turner - 1 May 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function GetUserTables(ado)
   Dim tabs(),ntab

   ado.Execute "BEGIN TRANSACTION"
   Dim recset
   Set recset = GetDisconRS(ado,GetUserTables_SQLServer)
   recset.MoveFirst
   ntab=0
```

```
   ntab=0
   While Not recset.EOF
       ntab=ntab+1
       recset.MoveNext
   Wend
   recset.MoveFirst
   redim tabs(ntab-1)
   ntab=0
   While Not recset.EOF
       tabs(ntab)= recset.fields(0).value
       recset.MoveNext
       ntab=ntab+1
   Wend
   recset.Close
   ado.Execute "COMMIT"
   GetUserTables = tabs
   Exit Function
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Pass an database connection and get an array of all the enabled user
' table triggers as TABLE,TRIGGER strings
'
' Author: Alexander J Turner - 1 May 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Function GetUserTriggers(ado)
   Dim trigs(),ntrig

   ado.Execute "BEGIN TRANSACTION"
   Dim recset
   Set recset = GetDisconRS(ado,GetTriggers)
   recset.MoveFirst
   ntrig=0
   While Not recset.EOF
       ntrig=ntrig+1
       recset.MoveNext
   Wend
   recset.MoveFirst
   redim trigs(ntrig-1)
   ntrig=0
   While Not recset.EOF
       trigs(ntrig)= recset.fields(0).value & "," & recset.fields(1)
       recset.MoveNext
       ntrig=ntrig+1
   Wend
   recset.Close
   ado.Execute "COMMIT"
   GetUserTriggers = trigs
   Exit Function
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This function writes SQL to restore all the data into a set of tables
' without changing the structure - IE a data only backup of the tables
' in pure SQL (ie loads of delete and insert statements).
'
' Parameters:
' ado         - a ADODB database connection objects
' fileName    - the file to which to write the SQL
' tabs        - a list of tables owner.name (like dbo.mytab)
' dataBase    - the name of the database the tables are in
' userUnicode - is the file to be unicode (recommended)
'
' Author: Alexander J Turner - 1 May 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Public Sub DumpDBDataToFile(ado, fileName, tabs,dataBase,useUnicode)
   Dim trc
   trc=0
   Dim fs
   ' Open the output file and select the chosen format
   Set fs = CreateObject("Scripting.FileSystemObject")
   Dim ts
   If useUnicode Then
       Set ts = fs.OpenTextFile(fileName, 2, True,-1)
   Else
       Set ts = fs.OpenTextFile(fileName, 2, True)
   End If

   Dim t,tt
   Dim rec
   Dim c
   Dim trigs
```

```
    ' Putting no count in the output script makes it run faster
    ts.WriteLine "SET NOCOUNT ON"
    ts.WriteLine "GO"
    ts.WriteLine "USE " & dataBase
    ts.WriteLine "GO"
    ' I had trouble with transactions, though under some conditions
    ' running with transactions was faster, often the transactions are
    ' so large that SQL Server 'jams up' and takes ages (even hours) to
    ' recover - this way is safer!
    ts.WriteLine "SET IMPLICIT_TRANSACTIONS OFF"
    ts.WriteLine "GO"

    ' It is important to turn off all enabled triggers else the db will
    ' be updating as it is loading and so all sorts of problems will ensue
    trigs=GetUserTriggers(ado)
    For Each t In trigs
        t=Split(t,",")
        For Each tt In tabs
            If UCase(Trim(tt))= UCase(Trim(t(0))) Then
                WScript.Echo "Disabling trigger: " & t(1) & " on " & t(0)
                ts.WriteLine "ALTER TABLE " & t(0) & " DISABLE TRIGGER " & t(1)
                ts.WriteLine "GO"
                Exit For
            End If
        Next
    Next

    ' sort the dependency order so that deletes and inserts will fit
    ' with FK restraints. There might be a way of turning off the restraints
    ' but this works as well.
    WScript.Echo "Sorting table order"
    SortDepOrder ado, tabs
    For c = ubound(tabs) To 0 Step -1
        ts.WriteLine "DELETE FROM " & tabs(c) & " WITH (TABLOCKX) "
        ts.WriteLine "GO"
    Next

    ' Now we write out the inserts to restore the data. The tables are
    ' loaded in the opposite order to that in which they are deleted from
    For Each t In tabs
        ado.Execute "BEGIN TRANSACTION"
        ' This allows insertion into identity columns
        ts.WriteLine _
            "IF OBJECTPROPERTY ( object_id('" & t & "'),'TableHasIdentity') = 1 " + _
                "SET IDENTITY_INSERT " & t & " ON "
        ts.WriteLine "GO"
        Set rec = GetDisconRS(ado,"SELECT * FROM " & t)
        Dim sql
        Dim sql1
        Dim first
        first = True
        If Not rec.EOF Then
        rec.MoveFirst
        While Not rec.EOF
            Dim i
            If first Then
                sql1 = "INSERT INTO " & t & " ("
                For i = 0 To rec.fields.count - 1
                    If i > 0 Then sql1 = sql1 + ","
                    sql1 = sql1 + rec.fields(i).name
                Next
                sql1 = sql1 + ") VALUES ("
                first = False
                WScript.Echo "Dumping " & t
            End If
            sql = sql1
            Dim vt
            Dim f
            ' Use the returning data type to work out how to escape the SQL
            ' this is far from perfect, I am sure that some translations
            ' will not work properly, but for now it seems to work on the DBs
            ' I am working with
            For i = 0 To rec.fields.count - 1
                f = rec.fields(i).value
                vt = varType(f)
                If vt = 1 Then
                    f = "Null"
                ElseIf vt = 2 Or vt = 3 Or vt = 4 Or vt = 5 Or vt = 6 Or vt = 14 Then
                    f = DBEscapeNumber(CStr(f))



                ElseIf vt = 11 Then
```

```
                   If vt Then
                       f = "1"
                   Else
                       f = "0"
                   End If
               ElseIf vt = 8 Then
                   f = DBEscapeString(CStr(f))
               ElseIf vt = 7 Then
                   f = DBEscapeDate(CStr(f))
               ElseIf vt = 17 Then
                   f = "0x" + Right( "0" & Hex(f),2)
               ElseIf vt = 8209 Then
                   f = "0x" + BinToHex(f)
               Else
                   WScript.Echo "Could not reformat", "Table=" & _
                     t & " Col=" & rec.fields(i).name & " vt=" & vt
                   WScript.Quit
               End If
               If i > 0 Then sql = sql + ","
               sql = sql + f
           Next
           sql = sql + ")"
           ts.WriteLine sql
           ts.WriteLine "GO"
           trc=trc+1
           ' I like to see some record of what is going on
           if trc mod 1000 = 0 Then
               WScript.Echo "Total row count=" & trc
           End If
           rec.MoveNext
       Wend

       End If
       rec.Close
       ' Turn back on normal identity rules
       ' It would be better to check if identity insert was on before we
       ' turned it off - this way we might turn it off when it is supposed to
       ' on for the DBs normal function. I should fix this some time soon
       ts.WriteLine _
           "IF OBJECTPROPERTY ( object_id('" & t & "'),'TableHasIdentity') = 1 " + _
               "SET IDENTITY_INSERT " & t & " OFF "
       ts.WriteLine "GO"
   Next

   ' Turn back on triggers
   For Each t In trigs
       t=Split(t,",")
       For Each tt In tabs
           If UCase(Trim(tt))= UCase(Trim(t(0))) Then
               WScript.Echo "Enabling trigger: " & t(1) & " on " & t(0)
               ts.WriteLine "ALTER TABLE " & t(0) & " ENABLE TRIGGER " & t(1)
               ts.WriteLine "GO"
               Exit For
           End If
       Next
   Next
   ts.Close
End Sub

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' This function returns a disconnected RS
' given a connection to the db and some SQL
'
' Author: Alexander J Turner - 1 May 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function GetDisconRS(ado,sql)
   Dim recset
   Set recset = CreateObject("ADODB.Recordset")
   recset.CursorLocation = adUseClient
   recset.CursorType = adOpenStatic
   recset.LockType = adLockBatchOptimistic
   recset.Open sql, ado, , , adCmdText
   Set recset.ActiveConnection = Nothing
   Set GetDisconRS = recset
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Given a variable of type Date returns  a variable of type String in
' long date format in English. For example a Date 01/01/2008 will
' become "1 January 2008". If that passed variable is a String, not
' a Date, then the results will still be a long date if VBScript can
' parse the passed String as a Date. However, the Date created  will
' be dependent upon the local in which VBScript is running.
```

```
'
' Author: Alexander J Turner - 12 Feb 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function DateLong(myDate)
  Dim months
  months=Split("january,february,march,april,may,june,july,august," & _
               "september,october,november,december",",")
  DateLong= _
      DatePart("D",mydate)       & " " & _
      months(    DatePart("M",myDate)-1) & " " & _
      DatePart("YYYY",mydate)
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Given any variable, will return a String which is safe for direct
' inclusion in an SQL Server SQL Statement. E.g. 01/01/2008 will
' result in '1 January 2008'. Note that the ' marks are included imn
' the returned String.
'
' Author: Alexander J Turner - 12 Feb 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function DBEscapeDate(myDate)
  ' The full String escape should never be required but it is here
  ' to ensure that a malevalent injection cannot cause
  ' commands to be passed via a Date field
  DBEscapeDate=DBEscapeString(DateLong(myDate))
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Given any variable, will return a String which is safe for direct
' inclusion in an SQL Server SQL Statement.
' Note that the ' marks are included in the returned String.
'
' Author: Alexander J Turner - 12 Feb 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function DBEscapeString(myString)
  DBEscapeString="'" & Replace(myString,"'","''") & "'"
End Function

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Given any variable, will return a Number which is safe for direct
' inclusion in an SQL Server SQL Statement. Note than non numeric
' values will be converted to 0.
'
' Author: Alexander J Turner - 12 Feb 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function DBEscapeNumber(myNumber)
  If NOT IsNumeric(myNumber) Then myNumber=0
  myNumber=myNumber*1.0
  DBEscapeNumber=Replace(myNumber & "","'","''")
End Function


''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Pass in an array of numbers (byte or between 0 and 255)
' and get out a string of hex representing the same numbers
'
' Author: Alexander J Turner - 1 May 2008
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
Function BinToHex(data)
   Dim ret
   Dim l
   Dim i
   Dim lb
   Dim h
   Dim d
   Dim o
   lb = LBound(data) - 1
   l = UBound(data) - LBound(data) + 1
   ret = String(l * 2, "0")
   Redim o(l-1)

   ' Use arrays and join as just adding to the end of a
   ' string scales badly as the length of the string increases
   For i = 1 To l
       d = 255 and ascb(midb(data,i,1))
       If d > 15 Then
           o(i-1) = Hex(d)
       Else
           o(i-1) = "0" + Hex(d)
       End If
   Next
```

```
    BinToHex = Join(o,"")
End Function
```

How it Works

Step one is just to log onto the database using ADODB. This connects to the server using the OLEDB database provider. I have set it so it can use a trusted (Windows authentication) or SQL Server authentication connection.

Next, all the activity goes to the function `DumpDBDataToFile`. This opens the file to which the SQL is to be written. A constant at the top of the script sets if this is to be written in Unicode or not. I would recommend Unicode as SQL Server supports Unicode data fields. After it has the output file sorted, it needs to start creating the SQL.

A lot of the development work I did on this surrounded speed. Not the speed of writing the SQL backup, but the speed of the resulting SQL reloading the database. I found that, counter to expectation, it worked faster if implicit transactions are turned off.

Before any `Delete` or `Insert` statements can be used reliably in the database restore, all triggers have to be turned off. By joining the sysobjects table to itself, it is possible to get all the active triggers and the tables on which they depend. The SQL to do this is in the script, but here it is again:

Collapse

```
SELECT spus.name + '.' + sp.name, s.name
FROM sysobjects s inner join sysobjects sp on
     s.parent_obj = sp.id inner join sysusers spus on sp.uid = spus.uid
WHERE s.xtype='TR' AND OBJECTPROPERTY(s.[id], 'ExecIsTriggerDisabled')=0
```

The script stores the trigger information so that it can create SQL to turn off all active triggers at the head of the output file and then turn them back on at the tail.

Once the triggers are all off, the script needs to write out the `Delete` statements to clear any data from the tables before new data is inserted. To do this, the dependency order of the tables needs to be worked out. This is done via sysobjects once again, in the function `SortDepOrder`. Once this is found, the `Delete` statements work in the opposite order to the `Insert`s.

Just to explain, the `TABLOCKX` on the `DELETE` statements just mean that the server will use a table lock straight away, and so saves a lot of lock escalation and performance issues. Chances are it may not have much effect most of the time; however, it appears to help sometimes. Any and all comments welcome on the subject.

Once all the `Delete`s are out the way, the `Insert`s which reload the data are created. The trick here is to know the format of each `Insert` statement. The field names can be acquired from the result set. Then, the way the data is added into the `Insert` statement is worked out from the data type of the elements in the fields of the recordset. The code below does that bit:

Collapse

```
For i = 0 To rec.fields.count - 1
    f = rec.fields(i).value
    vt = varType(f)
    If vt = 1 Then
        f = "Null"
    ElseIf vt = 2 Or vt = 3 Or vt = 4 Or vt = 5 Or vt = 6 Or vt = 14 Then
        f = DBEscapeNumber(CStr(f))
    ElseIf vt = 11 Then
        If vt Then
            f = "1"
        Else
            f = "0"
        End If
    ElseIf vt = 8 Then
        f = DBEscapeString(CStr(f))
    ElseIf vt = 7 Then
        f = DBEscapeDate(CStr(f))
    ElseIf vt = 17 Then
        f = "0x" + Right( "0" & Hex(f),2)
    ElseIf vt = 8209 Then
        f = "0x" + BinToHex(f)
    Else
        WScript.Echo "Could not reformat", "Table=" & _
          t & " Col=" & rec.fields(i).name & " vt=" & vt
        WScript.Quit
    End If
    If i > 0 Then sql = sql + ","
    sql = sql + f
Next
```

To be honest, this is probably the most important piece of the whole script. It is also where the script is going to fail for data types for which I have not yet accounted. `DBEscapeNumber`, `DBEscapeString`, `BinToHex` are all functions in the script which I have written to make string representations which can

be put straight into SQL. They take care of things like date format and escaping of ' in CHAR/VARCHAR.

Running the Script

To run, set the constants at the top of the script and then run it from *cscript*, like this:

Collapse

```
C:\Documents and Settings\user\Desktop>cscript DBBackup.vbs
```

The script will report its progress like this (from a real example):

Collapse

```
Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

Disabling trigger: Trig_ChckBusnUnitProjSlots on dbo.T_ProjBusnUnit
Disabling trigger: TRG_ChckSlotStat  on dbo.T_Slot
Disabling trigger: TRIG_BusnUnitUpdate on dbo.T_BusnUnit
Disabling trigger: Trig_ChckLocnSlots on dbo.T_Locn
Disabling trigger: TRIG_LocnUpdate on dbo.T_Locn
Disabling trigger: Trig_ChckLocnProjSlots on dbo.T_LocnProj
Disabling trigger: TRIG_MchnPropTypeUC on dbo.T_MchnPrpsType
Disabling trigger: TRIG_UpdtSlotSlotHist1 on dbo.T_Slot
Disabling trigger: TRIG_UpdtSlotSlotHist2 on dbo.T_SlotStatHist
Disabling trigger: TRIG_UserNameChck on dbo.T_User
Disabling trigger: Trig_ChckUserBusnUnitSlots on dbo.T_UserBusnUnit
Sorting table order
dbo.T_Bndl X dbo.T_Proj
dbo.T_BndlDepl X dbo.T_Depl
dbo.T_BndlPrps X dbo.T_Bndl
dbo.T_BndlSlot X dbo.T_Slot
dbo.T_BndlTypeBULocn X dbo.T_BusnUnit
dbo.T_BusnUnit X dbo.T_Locn
dbo.T_BndlTypeMchn X dbo.T_Mchn
dbo.T_BndlTypeSlotCata X dbo.T_SlotCata
dbo.T_BndlTypeBULocn X dbo.T_BusnUnit
dbo.T_DyncExpn X dbo.T_DyncExpnType
dbo.T_RedyTrak X dbo.T_TrakStat
dbo.T_TrakStat X dbo.T_TrakStep
dbo.T_TrakStep X dbo.T_User
dbo.T_SlotEvntTrig X dbo.T_SlotStatType
dbo.T_SlotHistData X dbo.T_SlotHistDataType
dbo.T_SlotHistDataType X dbo.T_SlotStatHist
dbo.T_SlotHistData X dbo.T_SlotHistDataType
dbo.T_RedyTrak X dbo.T_TrakStat
dbo.T_TrakStat X dbo.T_TrakStep
dbo.T_RedyTrak X dbo.T_TrakStat
dbo.T_UserEvnt X dbo.T_UserEvntStat
dbo.T_UserEvntStat X dbo.T_UserEvntType
dbo.T_UserEvnt X dbo.T_UserEvntStat
dbo.T_Slot X dbo.T_Locn
dbo.T_Locn X dbo.T_SlotCata
dbo.T_Slot X dbo.T_Locn
dbo.T_Mchn X dbo.T_User
dbo.T_BusnUnit X dbo.T_BusnUnitType
dbo.T_BndlTypeMchn X dbo.T_Mchn
dbo.T_SlotCata X dbo.T_Actn
dbo.T_Locn X dbo.T_Lang
dbo.T_Lang X dbo.T_LocnType
dbo.T_User X dbo.T_Locn
dbo.T_Slot X dbo.T_SlotCata
dbo.T_MchnSlot X dbo.T_Slot
dbo.T_Locn X dbo.T_Lang
dbo.T_User X dbo.T_Locn
Dumping dbo.T_Proj
Dumping dbo.T_Depl
Dumping dbo.T_Bndl
Dumping dbo.T_Actn
Dumping dbo.T_BndlType
Dumping dbo.T_LocnType
Dumping dbo.T_Lang
Dumping dbo.T_BndlTypePrps
Dumping dbo.T_SlotCata
Dumping dbo.T_BrknData
Dumping dbo.T_BusnUnitType
Dumping dbo.T_BusnUnitPrnt
Dumping dbo.T_BusnUnitTree
Dumping dbo.T_BusnUnit
Dumping dbo.T_CodeBook
Total row count=1000
Total row count=2000
...
Dumping dbo.T_ActnSubActn
Enabling trigger: Trig_ChckBusnUnitProjSlots on dbo.T_ProjBusnUnit
Enabling trigger: TRG_ChckSlotStat  on dbo.T_Slot
Enabling trigger: TRIG_BusnUnitUpdate on dbo.T_BusnUnit
Enabling trigger: Trig_ChckLocnSlots on dbo.T_Locn
Enabling trigger: TRIG_LocnUpdate on dbo.T_Locn
Enabling trigger: Trig_ChckLocnProjSlots on dbo.T_LocnProj
Enabling trigger: TRIG_MchnPropTypeUC on dbo.T_MchnPrpsType
Enabling trigger: TRIG_UpdtSlotSlotHist1 on dbo.T_Slot
Enabling trigger: TRIG_UpdtSlotSlotHist2 on dbo.T_SlotStatHist
Enabling trigger: TRIG_UserNameChck on dbo.T_User
Enabling trigger: Trig_ChckUserBusnUnitSlots on dbo.T_UserBusnUnit
All done
```

So here are some extracts from a produced SQL file:

Collapse

```
SET NOCOUNT ON
GO
USE dv
GO
SET IMPLICIT_TRANSACTIONS OFF
GO
ALTER TABLE dbo.T_ProjBusnUnit DISABLE TRIGGER Trig_ChckBusnUnitProjSlots
GO
ALTER TABLE dbo.T_Slot DISABLE TRIGGER TRG_ChckSlotStat
GO
ALTER TABLE dbo.T_BusnUnit DISABLE TRIGGER TRIG_BusnUnitUpdate
GO
ALTER TABLE dbo.T_Locn DISABLE TRIGGER Trig_ChckLocnSlots
GO
...
DELETE FROM dbo.T_AuthRuleDlig WITH (TABLOCKX)
GO
DELETE FROM dbo.T_AuthRuleBU WITH (TABLOCKX)
GO
DELETE FROM dbo.T_AuthRule WITH (TABLOCKX)
GO
DELETE FROM dbo.T_ActnSubActn WITH (TABLOCKX)
GO
...
IF OBJECTPROPERTY ( object_id('dbo.T_CodeBook'),'TableHasIdentity') = 1
   SET IDENTITY_INSERT dbo.T_CodeBook ON
INSERT INTO dbo.T_CodeBook (Sequ,Valu) VALUES (559,0xF5B2AADA26EAA6CE)
GO
INSERT INTO dbo.T_CodeBook (Sequ,Valu) VALUES (560,0xEDAC5014F0F50B0D)
GO
...
IF OBJECTPROPERTY ( object_id('dbo.T_MchnPrps'),'TableHasIdentity') = 1
   SET IDENTITY_INSERT dbo.T_MchnPrps ON
GO
INSERT INTO dbo.T_MchnPrps (MchnId,Sequ,Type,Data,Prnt)
       VALUES (1,1,1,'RbsManuf1',1)
GO
INSERT INTO dbo.T_MchnPrps (MchnId,Sequ,Type,Data,Prnt)
       VALUES (1,2,2,'IBM T42',1)
GO
```

Note that the script turns `IDENTITY_INSERT` on and off to allow exact re-creation of the identity columns on the tables.

That's all Folks

This is a bit fast and brief. I hope to add more explanation in the future, and I will try to answer any questions asked.

For more stuff like this - see my blog index page: http://nerds-central.blogspot.com/2008/01/excel-vbscript-index-page.html.

License

About the Author

alex turner

Web Developer

United Kingdom

Member

I am now a Software Systems Developer - Senior Principal at Micro Focus Plc.

My past includes a Ph.D. in computational quantum mechanics, software consultancy and several/various software development and architecture positions.

For more - see

blog: http://nerds-central.blogspot.com

twitter: http://twitter.com/alexturner

Article
Top

Comments and Discussions

You must Sign In to use this message board.

FAQ

Search

Noise Tolerance Medium Layout Normal Per page 25 Update

Msgs 1 to 18 of 18 (Total in Forum: 18) (Refresh) First Prev Next

SQLBackupAndFTP	**Ruslan Sudentas**	**8:27 19 May '09**
Re: SQLBackupAndFTP	alex turner	10:49 19 May '09
Nice Straight Forward	**spoodygoon**	**4:01 8 Feb '09**
Schema Objects Bug...	**Member 437390**	**8:28 29 Sep '08**
Re: Schema Objects Bug...	Member 437390	8:53 29 Sep '08
Re: Schema Objects Bug...	Member 437390	9:42 29 Sep '08
BUG: SortDepOrder get into infinite loop	**harishbalakrishnan**	**20:37 28 May '08**
Re: BUG: SortDepOrder get into infinite loop	alex turner	21:25 28 May '08
Bug. VBScript Error with Read Only Database	**versacid**	**7:51 14 May '08**
Re: Bug. VBScript Error with Read Only Database	alex turner	8:49 14 May '08
How to run the resultant script	**alex turner**	**3:52 8 May '08**
Bug. Database name [modified]	**Danila Korablin**	**21:10 7 May '08**
Re: Bug. Database name	alex turner	22:19 7 May '08
Re: Bug. Database name	Danila Korablin	2:56 8 May '08
Re: Bug. Database name	alex turner	3:32 8 May '08
modify article please	**Alex Kucherenko**	**3:41 7 May '08**
Re: modify article please	alex turner	4:16 7 May '08
Re: modify article please [modified]	alex turner	5:18 7 May '08

Last Visit: 14:02 5 Jan '11 Last Update: 16:36 5 Jan '11 **1**

General News Question Answer Joke Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+PgUp/PgDown to switch pages.