JetComp.exe Readme File

Table of Contents

Introduction

This Readme file includes sample code that allows you to easily view rows of data in which the Jet database engine found problems during the Compact process. The Jet database engine generates an error if it encounters a problem during the Compact process and then creates the MSysCompactError table in the destination database. You can use Jetcomp.exe on secured databases and databases that have a database password.

Jetcomp.exe is a standalone utility that allows you to compact databases created with Microsoft Jet database engine 3.x and 4.x. This utility may be used in conjunction with Microsoft Jet database engine 3.x and 4.x for recovering corrupted databases. Although it is possible to use the Microsoft Access Compact utility or the CompactDatabase method with Microsoft Jet database engine 3.x and 4.x, Jetcomp.exe may be able to recover some databases that these utilities could not. This is because the Microsoft Access Compact utility and the CompactDatabase method attempt to open and close a database before attempting to compact it. In certain cases where these utilities may not be able to reopen the database, Compact will be unable to proceed, preventing recovery of the database. Jetcomp.exe does not attempt to open and close the database before compacting, and may be able to recover some databases that the Microsoft Access compact utility and the CompactDatabase method could not. Jetcomp.exe is a freely distributable utility, but requires that you have one of the following products installed:

Microsoft Office 97
Microsoft Office 2000
Microsoft Access 97
Microsoft Access 2000
Microsoft Visual Basic® 5.0
Microsoft Visual Basic 6.0
A Microsoft Office Developer Edition 97 run-time application that includes the run-time version of Microsoft Access 97
A Microsoft Office Developer 2000 run-time application that includes the run-time version of Microsoft Access 2000
A Microsoft Visual Basic 5.0 run-time application that includes Microsoft Jet database engine 3.5
A Microsoft Visual Basic 6.0 run-time application that includes Microsoft Jet database engine 3.5 or 4.0

Localization and Support

The User Interface of JetComp.exe is in English only and is not supported by Microsoft Product Support Services. JetComp.exe can be used to compact databases in any language supported by the Microsoft Jet database engine.

Instructions

1. Double-click JetComp.exe.

2. In the **Database to Compact From (Source)** box, type the path and name of the database that you want to compact.
3. In the **Database to Compact Into (Destination)** box, type the path and name of the new compacted database that you want to create.
4. Under **Additional Compact Options**, set the appropriate options.
5. Click **Compact**.

Errors Encountered in Earlier Versions of the Microsoft Jet Database Engine

In the course of using Microsoft Jet database engine 3.x, you may have encountered the following errors.

Error #	Error Description
3197	The Microsoft Jet database engine stopped the process because you and another user are attempting to change the same data at the same time.
3343	Unrecognized database format ‘databasename.mdb'.
3015	‘databasename.mdb' isn't an index in this table. Look in the Indexes collection of the TableDef object to determine the valid index names.

These errors typically relate to some form of corruption in the database. Error 3015 resulted in the Jet database engine not being able to gain access to the database. Error 3343 required you to run the Jet database engine repair utility.

The following is a description of what the errors really signified.

Error #	Error Description	Cause/Problem
3197	The Microsoft Jet database engine stopped the process because you and another user are attempting to change the same data at the same time.	This is typically occurs when a long value column (an OLE or MEMO data-type) has bad data stored in it. Long value columns are typically stored in a separate page from the page that the row is stored in. If a long value column is present in the table schema, the Jet database engine will attempt to read the long value page when reading the row of data. In order to read the long value page, there is a pointer in the row of data. This error is generated when the Jet database engine cannot properly read the long value page from the pointer present in the data row. When viewing a row that exhibits this behavior in Microsoft Access data-windows, the user will typically see the number sign (#) for the entire row.
3343	Unrecognized database format ‘databasename.mdb'.	This is typically occurs when the Jet database engine was improperly shut down during the process of writing to disk.
3015	‘databasename.mdb' isn't an index in this table. Look in the Indexes collection of the TableDef object to determine the valid index names.	This is typically occurs when there is an index missing on the MSysObjects table. This is typically caused if the repair process is aborted.

In order to remedy these problems, Microsoft Jet database engine 3.51 and 4.x have enhanced the compact process. With Microsoft Jet database engine 3.51 and 4.x, all of the errors described above will be eliminated and repaired if possible. Compact now assumes all the functionality that the Jet database engine repair process included and it is no longer recommended that you use the Jet database engine repair process.

Compact should always be run on a regular basis because it creates a new database with the data and tables in contiguous and sorted order. It also refreshes the statistics in the database and causes all stored queries to recompile. Ultimately, this reduces the size of the database and can significantly increase performance.

MSysCompactError Table

If any of the errors described above are encountered, compact should be run as soon as possible. If an error is discovered, the Jet database engine will create a table called MSysCompactError. This table will have the following schema.

Column Name	DataType	Description
ErrorCode	Integer	Error number generated from the Jet database engine. Note that this error number is an internal Jet database engine error number and does not correlate to error numbers that are returned from Visual Basic for Applications.
ErrorDescription	Memo	Error message from the Jet database engine error file. Microsoft Jet database engine 3.51 currently uses a non-descriptive error message because the Jet database engine could not modify the error file for this release. The Jet database engine 4.x provides a more descriptive error message.
ErrorRecId	Binary	The bookmark of the row as it relates to the destination database for a row that had long value column values changed to ################. Note that this bookmark is only guaranteed to be good when using the OpenRecordSet method's dbOpenTable type. The bookmark may also become invalid if rows are modified in the table after compact.
ErrorTable	TEXT (64)	The name of the table that contained the partial bad row, if the partial bad row was retrievable.

It is important to note that this table will only persist in the database until the following compact. Compact will not copy any tables where the table name starts with MSysCompactError to the destination database. This is done for maintenance, and the existence of tables where the name starts with MSysCompactError indicates that the previous compact encountered problems**. It is important that you do not have any tables that start with MSysCompactError or they will not be present in the destination database after running compact.**

In addition to the Jet database engine creating the MSysCompactError table, the Jet database engine will return an error message at the completion of compact to signify that a problem was encountered.

Unfortunately, modifying the error DLL (msjter35.dll for 3.51) was not an option due to localization issues and Microsoft was forced to use a non-descriptive error message. The error message for 4.x has been update to be more meaningful, as stated below.

Jet Version	Error #	The Jet database engine error message
3.51x	8119	Record(s) can't be read; no read permission on 'databasename.mdb'.
4.x	8119	Errors encountered in compact process

Issues Fixed by Updated Compact Utility

Compact will currently fix the following issues with a database.

Problem	Result	Resolution
Missing index(s) on MSysObjects.	Database cannot be opened and is unusable.	Compact will ignore the existence of indexes on the MSysObjects table and will scan the table instead. The destination database will contain the necessary indexes on the MSysObjects table to allow the database to be opened.
Commit bytes are left in a suspect state.	Database cannot be opened and typically required the repair process to be run.	Compact will ignore the status of the commit bytes and continue to process while correcting for any errors.
Properties are missing for a table.	The table cannot be open from Microsoft Access, but may be able to be open from DAO's OpenRecordSet method. Properties for a table are stored in MSysObjects in the LVProp column. This column is a long value column falling under the issues discussed above.	The Jet database engine will reset the LVProp column to NULL, effectively removing all properties for the table. It will also log a row in MSysCompactError, but there will not be a bookmark to identify the row in the destination database.
Pointers in non-system table long value columns point to invalid locations.	The Jet database engine will typically return error # 3197, which appears to be a multi-user error.	The Jet database engine will reset the LV column to sixteen number signs (#). This is done to allow the user to see what LV column could not be read. The Jet database engine will also log a row in MSysCompactError with a bookmark to identify the row in the destination database.
Size of LV column in data row is different than size reported in LV page.	The Jet database engine will typically return error # 3197, which appears to be a multi-user error.	The Jet database engine will reset the LV column to sixteen number signs (#). This is done to allow the user to see what LV column could not be read. The Jet database engine will also log a row in MSysCompactError with a bookmark to identify the row in the destination database.

When the Jet database engine compact encountered errors, it would delete the destination database upon completion of compact. This has now been modified so that the destination database will not be deleted if

the compact process encounters errors described above. However, if some other error is encountered that will prevent the destination database from being usable, compact will delete the destination database. If compact is run from the Microsoft Access toolbar and you choose the destination database name to be the same as the source database name and an error described above is encountered, a database called DB?.MDB will be left on disk. The source database will remain unaltered.

Sample Code to View Rows Containing Modified Column Data

While the MSysCompactError table will give a bookmark value to the row that had corruption on it, Microsoft is providing code to make it easier for you to view what rows had column data that the Jet database engine could not read. The code below will create additional tables from the rows in the MSysCompactError table that compact created. The code will create tables that start with MSysCompactError and will have the original table name appended to MSysCompactError. The created table will have the exact schema as the original table and will only contain the rows that had column data value modified from the compact process. This will allow you to easily view rows that were affected by the compact process. **It is important to remember that any subsequent compact will remove these tables**.

Copy and paste the following code into a new Microsoft Access module and run the code. Note that this code will not work with Visual Basic because of the CurrentDB call, but it can be easily modified to DAO's OpenDatabase method.

```
Sub main()
    On Error GoTo ErrorHandler
    Dim db As DAO.Database, vBookMark As Variant, _
      rsMSysCompactError As DAO.Recordset, strErrorTable As String, _
      rsErrorTable As DAO.Recordset, fldErrorField As DAO.Field, _
      strSQLSEL As String, strColumnValue As Variant, _
      qdTemp As QueryDef, strSQLINS As String, intLoop As Integer, _
      lngTableNameLength As Long, _
      colErrorCollection As New Collection, intErrorCount As Integer

    Set db = CurrentDb()
    ' Walk through the MSysCompactError table to find rows that reflect
    ' lost data values.
    Set rsMSysCompactError = db.OpenRecordset("SELECT * FROM
MSysCompactError WHERE ErrorRecId IS NOT NULL", dbOpenDynaset)
    intErrorCount = 0
    While Not rsMSysCompactError.EOF
        ' Get the name of the table that had column data missing.
        strErrorTable = rsMSysCompactError!ErrorTable
        ' Check to see that tablename is not greater than 48 characters
        ' to stay under 64 character tablename limit.
        lngTableNameLength = Len(strErrorTable)
        If lngTableNameLength > 48 Then
            strErrorTable = Mid(strErrorTable, 1, 48)
            ' See if this truncated table name already exists.
            On Error Resume Next
            colErrorCollection.Add strErrorTable, strErrorTable
            ' If this already exists in the collection, then there is a
            ' duplicate table name.
            If Err = 457 Then
                ' Truncate one more digit to append on the intErrorCount
                ' number to eliminate the duplicate table name.
                strErrorTable = Mid(strErrorTable, 1, 47)
```

```
                strErrorTable = strErrorTable &
Mid((Str(intErrorCount)), 2, 1)
                intErrorCount = (intErrorCount + 1)
            End If
        End If

        ' Get the bookmark value of the row that had lost column data.
        vBookMark = rsMSysCompactError!ErrorRecId
        ' Open table that has lost column data.
        Set rsErrorTable = db.OpenRecordset(strErrorTable, dbOpenTable,
dbReadOnly)
        ' Move to row that has lost column data.
        rsErrorTable.Bookmark = vBookMark
        ' Start to build SQL string to call up in a table window.
        strSQLSEL = "SELECT * INTO MSysCompactError" & strErrorTable & "
FROM " & strErrorTable & " WHERE "
        strSQLINS = "INSERT INTO MSysCompactError" & strErrorTable & "
SELECT * FROM " & strErrorTable & " WHERE "
        intLoop = 0
        For Each fldErrorField In rsErrorTable.Fields
            strColumnValue = fldErrorField.Value
            ' Logic to build predicate based on various data types.
            If Not IsNull(strColumnValue) Then
                ' Can't use ordinal as no guarantee of first column
                ' being zero.
                ' Check to see if this is the first column or not to
                ' build SQL statement.
                If intLoop = 0 Then
                    If fldErrorField.Type = dbDate Then
                        strSQLSEL = strSQLSEL & "[" & fldErrorField.Name
& "] = " & "#" & strColumnValue & "#"
                        strSQLINS = strSQLINS & "[" & fldErrorField.Name
& "] = " & "#" & strColumnValue & "#"
                    Else
                        If fldErrorField.Type = dbText Or
fldErrorField.Type = dbChar Or fldErrorField.Type = dbMemo Then
                            strSQLSEL = strSQLSEL & "[" &
fldErrorField.Name & "] = " & "'" & strColumnValue & "'"
                            strSQLINS = strSQLINS & "[" &
fldErrorField.Name & "] = " & "'" & strColumnValue & "'"
                        Else
                            strSQLSEL = strSQLSEL & "[" &
fldErrorField.Name & "] = " & strColumnValue
                            strSQLINS = strSQLINS & "[" &
fldErrorField.Name & "] = " & strColumnValue
                        End If
                    End If
                Else
                    If fldErrorField.Type = dbDate Then
                        strSQLSEL = strSQLSEL & " AND " & "[" &
fldErrorField.Name & "] = " & "#" & strColumnValue & "#"
                        strSQLINS = strSQLINS & " AND " & "[" &
fldErrorField.Name & "] = " & "#" & strColumnValue & "#"
                    Else
                        If fldErrorField.Type = dbText Or
fldErrorField.Type = dbChar Or fldErrorField.Type = dbMemo Then
```

```
                            strSQLSEL = strSQLSEL & " AND " & "[" &
fldErrorField.Name & "] = " & "'" & strColumnValue & "'"
                            strSQLINS = strSQLINS & " AND " & "[" &
fldErrorField.Name & "] = " & "'" & strColumnValue & "'"
                        Else
                            strSQLSEL = strSQLSEL & " AND " & "[" &
fldErrorField.Name & "] = " & strColumnValue
                            strSQLINS = strSQLINS & " AND " & "[" &
fldErrorField.Name & "] = " & strColumnValue
                        End If
                    End If
                End If
            End If
            intLoop = (intLoop + 1)
            ' QJet limitation for maximum conditions is reached.
            If intLoop = 39 Then
                Exit For
            End If
        Next fldErrorField
        On Error Resume Next
        ' Create error table if it does not exist.
        db.Execute strSQLSEL, dbFailOnError
        If Err = 3010 Then
            On Error GoTo ErrorHandler
            ' Add rows to error table if it already exists.
            db.Execute strSQLINS, dbFailOnError
        End If
        rsErrorTable.Close
        rsMSysCompactError.MoveNext
    Wend
    rsMSysCompactError.Close
    MsgBox "Done!"
    Exit Sub
ErrorHandler:
    MsgBox "An error has occurred " & Err & " " & Error
    Resume Next
End Sub
```