DataFlex Index Checking Utility

Co-developed By
Bob Worsley
David Martinko

November 11, 2002

Table of Contents

Index Checking

What you can and cannot do…

This utility will only work with a DataFlex database, it will not check SQL indexes.

Users may be on the system while the index checker is operating. Be aware that it is somewhat resource intensive so a slow-down may be encountered if run on a workstation being used for data entry or on a WTS server.

Why do it?

DataFlex databases are normally very stable, so why do we need such a program? If the users didn’t do such silly things as shut off their PCs without first exiting the programs, or have the occasional defective nic card or ran every workstation on a UPS, we wouldn’t have data problems. File servers once in a while have the audacity to crash, which can also make life interesting.

Going to a client server database would solve a number of these problems, but if that isn’t possible, you’re going to run into them every once in a while.

If it’s suspected that there are one or more bad indexes in your data, there are two actions that may be taken. You can either just reindex your data, in which case your users will all need to be off the system for the time it takes, or you can run the index checker which may tell you whether you need to or not. The gain you will get by doing this is a time savings if a reindex isn’t necessary, and possibly an indication of bad data if such exists. If your customers are experiencing problems, this utility may save them some production down time if your guess that a reindex is necessary is not indeed correct.

Included files

CkIndex.pkg	Class and object that actually performs the checking
FileUtil.prj	Project file for the IDE
FileUtil.src	Source module
FileUtil.mn	Menu file, will need to be registered in the IDE as an external component
CheckIndex.vw	View that calls the checker

The reason for externalizing the menu is to remove all of the normal VDF menu and toolbar options, they aren’t necessary for this program.

Additional tools may be included in this program, merely add them to the menu, .prj and the .src modules. Or, add this project to your IDE and do it there.

Program Structure

CkIndex.pkg is a class and object that may be included in either a front-end user view or in a back-end program that could be called from an NT service.

External calls

Procedure File_Loop [file #] [Message Object #]

If file # is 0 all files in filelist will be checked, if a valid number, only that file will be checked.

If the Message Object # is populated, the appropriate user screen message is returned to that object. This is useful for displaying busybox messages in the calling view rather than with Sentinel.

Example: Send File_Loop to oIndex_Check iFile iMsgObj

Function Show_Errors

Dumps the error array to an output box

Example: Send Show_Errors to oIndex_Check to iRC

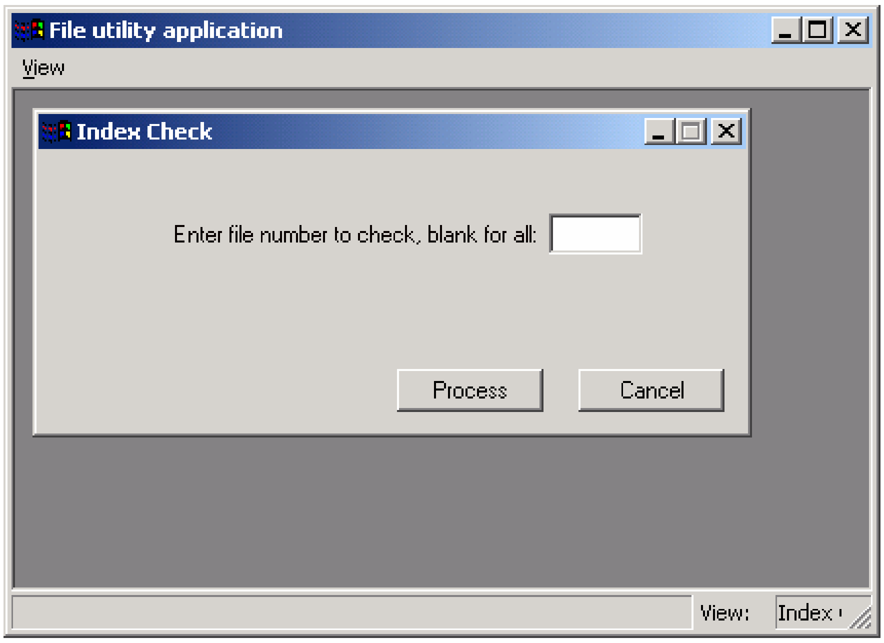
The code is fairly well documented, and isn't all that complicated, so extensive detail has not been provided here.

If run as a service…

The class contains function show_errors, which could be easily modified with an argument to redirect the array output from the output box to using Direct_Output to a file. Since this need – continuous data trouble -- would indicate that something is wrong with the system or application; this functionality is left to the developer.

Operation

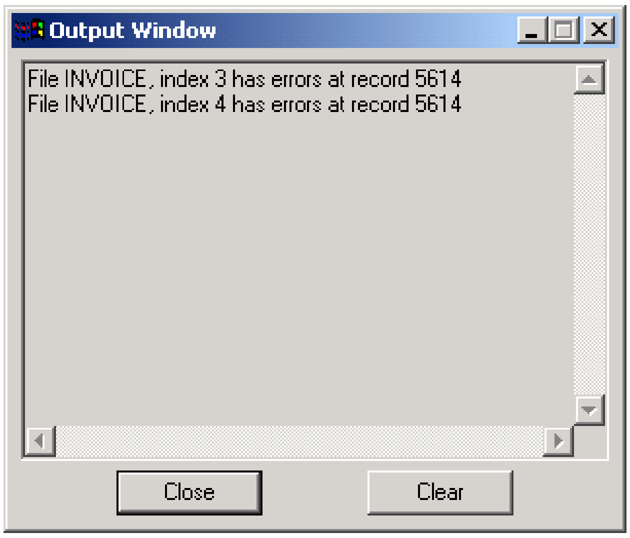
Double click on FleUtil.vd7 to start it, the following screen will appear.



To check indexes, either enter the file number as shown in dbExplor to check a single file, or click “Process” with the file number blank to check all. Depending on the number of records in your files, this could take a while, as each record in each file will be checked.

Problems you may encounter

If there’s a problem with a file, you will see an output screen as follows:



```
File INVOICE, index 3 has errors at record 5614
File INVOICE, index 4 has errors at record 5614
```

Bad Data

The checker will discontinue checking a file once an error has been detected which is why you may only see one error per index and file.

First, look at the record number of the indicated file using dbExplor. If bad data can be recognized, fix the record or delete it and rerun the utility. Be sure and use the index shown in the output window.

If the data for the record number shown appears correctly, look at other records above or below.

“Bad data” would be something in one of the fields that shouldn’t be there. For example, strange dates:

09/30/0100
01/14/0904

Both of these dates have really strange appearing years, but for some reason DataFlex handles them correctly one time and incorrectly the next. They don’t cause an error in a data field, but they do not “find” correctly in an index.

Other bad data would be something line “5:60” in a numeric field. The colon is incorrect.

In both cases, either fix the bad data or delete the record. Rerun the utility and if you get a clean run, you’re done. If not and a different record number shows, fix it.

Reindex

If no “bad data” can be found, reindex the file with dbBldr.

Rerun the utility afterwards and if a clean run, you are done. If still an indication of a problem, the next step might be to replace the file structure as it may be damaged internally – a dump and reload.

Test

How does one test such a utility? After all, bad indexes aren’t the “normal” and not everyone has them hanging around. Actually it’s easy; there are at least a couple of ways.

1. Edit a file with dbBldr, set <u>all</u> indexes to batch. Create a new record in the file with a duplicate key field. Reindex and check it. Reindexing will inform you that there's a problem.

2. Save a copy of the .dat file you want to test. Delete some records in the file, then copy the original .dat file back in and test.