DBToFile : A Microsoft Access Database Backup Engine

Introduction

Description

DBToFile is a database backup system for saving linked records to a file (in text form or in binary form for the binary fields of the database such as pictures, OLE documents, etc.) so that it can be reloaded later into a database with the same structure.

Available versions of this document

This document is available at :

http://perso.worldonline.fr/ors/dbtofile/index.html

in French and English in the following formats :

French : HTML, Doc (Word 97/2000), Pdf, Postscript.

English : HTML, Doc (Word 97/2000), Pdf, Postscript.

Advantages

- The backup file can be sent by diskette or by Internet as a document attached to an e-mail. It is a small file, mainly of text type, and it can also be compressed ;

- This system can be used for the off-line replication of several databases towards a single main database using update files (and then reciprocally so that the data is synchronised in both directions). Example : a database of customer records shared by a team of sales agents ;

- It is possible to specify what must be included in the file and what must be excluded, at table level and at table field level See Excluded fields ;

- Automatic loading and saving is possible by disabling user confirmations ;

- DBToFile is an ActiveX control and can therefore be used in various development environments, for example in an HTML page, see Demonstration ;

- If you distribute software that uses data access, you can offer the backup system just by inserting the ActiveX control in an Access or Visual BASIC form or in an HTML page, and preparing the table definitions and the requests for backing up records in the database. ; see also Depending files of the ActiveX control ;

- The ActiveX control interface can be invisible, as command button functions are also available as a control method ;

- Complex queries are possible for selecting the records to be saved ;

- Examples of applications related to the Internet : enquiry or voting system by electronic mail, system for completing a database form off line, etc.

Specifications

- DBToFile can be used with Access 2, 95, 97 or 2000 databases (see also Future development using ADO) ;

- Security management is taken into account for read/write access to the data being backed up (it is possible to specify a user account with its password);

- DBToFile requires the following components : VBAJet32, Visual BASIC 6 runtime and DAO 3.6, see Installation

Limitations

- Table definition is not automatic; it is necessary to describe the tables to be backed up in a table designed for this purpose (DBToFile_Table of the backup control database DBToFile.mdb). However a database report can be generated automatically to make this work easier, see stages 9 and 17 of the Demonstration. A user-friendly interface is envisaged in a future version.

- The backup file is not intended to be imported directly into spreadsheet or database software, because the data stored in this file can relate to all the tables in the database, and it can be reloaded as a whole into a database with the same structure without losing the links between the records. Thus, the backup system is more sophisticated than just exporting a table (though that is also possible).

- It is not possible to restore the state of the database after validating the loading of the file (make a backup if necessary), but replacements are confirmed

- Unique index field : there can only be a single index field per table (or a single combination of fields corresponding to an index of the table), except the AutoNumber identifier field. This is because it is impossible to edit several records at the same time during loading (if several unique index fields already exist in the table for several distinct records, it will be impossible to load the file).

See also Technical limitations.

Operation

- Comparison with replication : replication control is simplified because it is reduced to confirming the possible replacement of some records (there are no rules for conflict resolution during record replacement). The fields to be replaced (those for which notification has been set) are displayed in a dialog box as a drop-down list; the user is asked to confirm or to cancel all the replacements. It is not possible to see the value of each of these fields that will be replaced (unless the content of the file is viewed).

Note : the examples in the original version (in English) Northwind.mdb are given in brackets after those of Comptoir.mdb in the following paragraphs)

- A checking system for identifiers is provided to avoid saving the same record several times. For example, if new daily orders from the Orders table are saved, and if several orders refer to the same product in the Products table via the [Order Details] table, the item will be only be saved once.

- Specific fields in the DBToFile_Table table of the backup control database DBToFile.mdb :

* UniqueFields (relates to file loading) : it is necessary to specify the unique field when there is a unique index to the table. At loading time, a search is made for the unique record in order to avoid the "duplicate key in index" error. If necessary, the record is then edited (modified) by the corresponding recording of the file. The index can be a combination of several fields; in that case it is the whole combination that must respect the uniqueness constraint, as is the case for the table index . On the other hand, only one unique index is authorized, see also the limitation about unique index fields and the technical

limitation on the Primary mnemonic field. It is possible to specify a unique field (or a combination) even if there is no corresponding index in the table. This prevents records being duplicated each time the file is loaded. For example, in the Products table, it is a good idea to specify that the ProductName field must be unique in the setting table, even if there is no uniqueness constraint on this field in the table. Tests using an empty database are necessary to detect this fault, see Demonstration ;

* Table identifier field (IDField, relates to file saving and file loading). This field is used to establish a simple link between two tables (a reference of numerical type). For example, EmployeeID is the identifier field of the table Employees. This identifier will be used for an employee's order in the Orders table.

* Mnemonic field of the table (MnemonicField, relates to file saving and file loading). The principle is the same as for the identifier field, but this time the reference is not an AutoNumber one, but a short text field defined by the user (a mnemonic text). This allows comprehensible references between two tables. It is essential in the case of a reference between tables from different databases, or in the case of the optional saving of some tables, because it is easier to find a mnemonic field of text type than an abstract numerical field (especially an AutoNumber one). See also the technical limitation about the Primary mnemonic field. Example : the Customers' table has the CustomerID mnemonic field, which is a text field and not an AutoNumber identifier.

- Tables of the backup control database DBToFile.mdb concerning file saving :

* Equivalent identifiers (DBToFile_EqID) : list of identifier fields corresponding to a single field in another table. For example, if the list of customer companies and the list of supplier companies are collected in the same Companies table and a table refers simultaneously to a customer and to a supplier, then the identifier fields will be respectively CustomerCompanyID and SupplierCompanyID and the list of the equivalent identifiers would be : Company=CustomerCompanyID, SupplierCompanyID.

* Child tables (DBToFile_ChildTable) : this list allows automatic saving of tables that depend on a table referred in a saved record. By default , only tables referred by their identifier (or equivalent identifier) are saved ; so that dependent tables are also saved, you must specify them in this table. Example : when the Orders table is saved, the details of the selected orders must also be saved. For this, show in the Orders table that the child table is [Order Details].

See also Additional documentation ;

Installation

Download

- If Visual BASIC 6.0 (SP3) is already installed on your computer (in particular VBAJet32 and DAO 3.6), try the ocx dbtofile.zip (250 KB) directly, without the packaged installation. You just have to register the ocx using the command C:\Windows\RegSvr32.exe DBToFile.ocx (if the VB6Fr.Dll file does not exist in the directory C:\Windows\System\, copy it into it); if however you encounter errors (in particular 3446 or 3447), you may have to download the complete package (see also Depending files of the ActiveX control) :

- If not, download the complete package dbtofileinstall.zip (4 MB). Visual BASIC 6.0 runtime, VBAJet32

and DAO 3.6 for Windows 95 or 98 will be installed automatically.

Unzip

Unzip the dbtofileinstall.zip or dbtofile.zip file into a directory of your choice, using for example PkZip for Windows shareware.

Setup

- Launch setup.exe, or launch DBToFile.html directly if the OCX is already registered (see DBToFile registration).

Before starting

You can test the sample using Comptoir.mdb (Access 97 version), the French version of Northwind.mdb, or using the original database. If MS-Access is installed on your computer with the sample databases, you can copy Northwind.mdb from the directory :

C:\Program Files\MSOffice\Office\Sample\Northwind.mdb

or you can download the database at the following URL :

http://perso.worldonline.fr/ors/dbtofile/northwind.zip

Then copy Northwind.mdb into the install directory of the DBToFile software, which by default is : C:\Program Files\DBToFile\.

The original Access 97 French sample database Comptoir.mdb has a relation respecting referential integrity without the "Cascade Update Related Fields" option between [Clients] and [Commandes] tables. This is probably a small error by Microsoft France, because this relation respects cascade updating on the original version Northwind.mdb (furthermore, the Access 2000 version of Comptoir.mdb has been corrected). However, it is not possible to edit the records in the Client table when a primary mnemonic field has a relation respecting referential integrity without cascade updating (see the technical limitation about the Primary mnemonic field). The Comptoir.mdb database provided with DBToFile has therefore been corrected so that the sample works (the photographs have been deleted to make installation smaller).

Demonstration

After reading the Installation demonstration procedure, look at the sample using the ActiveX control DBToFile in HTML with VBScript : Demo.htm.

Here is the descriptive text of the demonstration stages :

Northwind.mdb backup

Northwind.mdb meta-backup
DBToFile.mdb meta-backup

Introduction

You do not have to use the DBToFile control buttons in any of the demonstrations; you can just use the 'Start !' button. You can then go to the next stage. You can change the language and start the stages again (from 1 to 13) with the other sample database Comptoir.mdb, which is the French version of Northwind.mdb. In this case, the name of the file will be Commande.cmd instead of Order.ord. Remember that Northwind.mdb is not included in the package.

Northwind.mdb backup

1-Saving an order

1-Saving an order in Order.ord file : the order with the number 10248 is selected by a query ; all the upstream records (e.g.: CustomerID) and the downstream records (e.g.: [Order Details]) related to this order will also be automatically saved in the file, respecting the following logical process : when the file is to be reloaded into a database with the same structure, referential integrity rules must be respected. The aim of the next stages from 2 to 7 is to test this backup file.

2-Checking Order.ord

2-Checking the created file Order.ord : the heading of the file is analyzed to check if the file can be imported into the database Northwind.mdb. At this stage, DBToFile checks if a reference to a backup file exists in the control database DBToFile.mdb and if its version number is compatible with the file version.

3-Loading test

3-Testing the reloading of the created file Order.ord in the same Northwind.mdb database : the goal is only to check that there is no duplicate key in index error using unique index fields ; as the data is identical, there should not be any modifications of the database contents. The order number that will be displayed after loading will remain unchanged. If you make modifications in the Northwind.mdb database concerning the records saved into the file, these modifications will of course be lost if you reload the file. However, you can repeat stage one of saving to update the backup file with those modifications.

4-Copying Northwind.mdb

4-Copying the database to create an empty Northwind0.mdb for testing; in order to check if all necessary records are actually stored in the backup file, Northwind.mdb will be copied to Northwind0.mdb, and its contents will be erased in the next stage.

5-Making Northwind0.mdb

5-Deleting all records of the test database Northwind0.mdb ; this new database will be empty and will have the same structure as Northwind.mdb, so the backup files will also be compatible with this database.

6-Loading Order.ord

6-Loading the created file Order.ord into the empty database Northwind0.mdb : records will be imported into the empty database and will thus be notified between brackets. The order number which will be displayed after loading will be different from the original one in the file that came from the first database, because it is automatically generated to a new record in an empty database (AutoNumber datatype field).

7-Reloading Order.ord

7-Reloading the created file Order.ord into the test database Northwind0.mdb that is no longer completely empty ; note that records are replaced (as in stage 3). The order number that will be displayed after loading will be unchanged since the previous stage. Important remark about the test methodology : it is necessary at this stage to check that records are not duplicated after each loading of the file into the database, in which case it would be necessary to specify the uniqueness constraints that records must respect in the DBToFile_Table table using the UniqueFields field of the DBToFile.mdb database. For this, check the contents of the Northwind0.mdb database. You can set a uniqueness constraint even if there is no corresponding index in a table.

8-Viewing Order.ord

8-Viewing the created file Order.ord (for information only). You cannot modify this file when it contains binary data (e.g.: employees' pictures), otherwise it could become unreadable.

9-Northwind.mdb database reporting

9-Reporting the database structure of Northwind.mdb : this functionality of DBToFile control is useful for defining tables to be saved from a database to which you want to add the backup system.

Northwind.mdb meta-backup

10-Northwind.mdb meta-backup

10-Backup of the backup system of Northwind.mdb into the Northwind.dbe file : the backup system of a given database can also be saved in a file, since it is completely described using Access tables from DBToFile.mdb. To do this, use the saving reference FileTypeID=5. The advantage is that for a database which is distributed among many customers, the backup system can be updated without modifying the source code. However, this advantage can only relate to a new selection of records (or a backup correction), because if the database structure changes, there will be in any case be a significant update to make. The goal of the

next stages 11 and 12 is to test this backup file.

11-DBToFile0.mdb creation

11-Creating an empty database DBToFile0.mdb for testing.

12-Loading Northwind.dbe

12-Loading Northwind.dbe file into an empty copy DBToFile0.mdb.

13-Viewing Northwind.dbe

13-Viewing the created file Northwind.dbe (for information only).

DBToFile.mdb meta-backup

14-DBToFile.mdb meta-backup

14-Backup of the backup system of DBToFile.mdb into the DBToFile.dbe file : the point is to illustrate the example of self-reference and to give an additional backup sample to better understand the possibilities. This system could make it easier to update the DBToFile utility while avoiding losing the backup settings of those that have already used a previous version of the software utility. However, this file can not be reloaded without first specifying the heading in the DBToFile_Main table and the list of tables to be saved in the DBToFile_Table table of the DBToFile.mdb database.

15-Loading DBToFile.dbe

15-Loading the created file DBToFile.dbe into an empty copy DBToFile0.mdb.

16-Viewing DBToFile.dbe

16-Viewing the created file DBToFile.dbe (for information only).

17-DBToFile.mdb database reporting

17-Reporting the database structure of DBToFile.mdb (for information only).

End of the demonstration

End of the demonstration. Thank you for your attention. For more information, consult the documentation joined to the software.

Additional documentation

Various remarks

- Example of an update of the database structure : the backup system makes it easy to update the database structure by for example adding a field (file version management is provided). Old files are loaded even if a field is missing or has been renamed(if the version is included in the interval >= MinFileVersion and < MaxFileVersion of the DBToFile_Main table). An error message will then indicate that such fields are missing from the database and that their value will be ignored. New fields will be supported in new backup files, generally without any modification of the backup table settings (except in the not very frequent case of a selection request in which there is not SELECT * FROM... but SELECT table1.field1, table1.field2 FROM...). These select queries are only used to start a sequence of cascade backups, the cascades of records being automatic in accordance with the settings of the backup tables : DBToFile_ChildTable (see Child Tables) and DBToFile_EqID (see Table of equivalent identifiers).

- Specific field in the DBToFile_Table table of the backup control database DBToFile.mdb :

* Excluded field (ExcludedField) : by default, all the fields of a table are saved. To avoid saving a given field, it is necessary to specify this field in the list of the excluded fields. This is useful in the case of a binary field that is considered to be too large or a temporary field that does not need to be saved. Example : picture field in the Employees table .

Speed optimization

Methodology : speed optimization should be considered only after the backup system works correctly.

- Specific fields of the DBToFile_Table table of the backup control database DBToFile.mdb :

* Presentation in table mode (bArrayLayout, relates to loading and saving speed optimization) : presentation of a record using only one line in order to optimize backup and loading. This technique is less effective than the Structure array one but it does not have its disadvantages.

* Index field (IndexField, relates only to loading optimization) : it has the same principle as Unique fields, but this time the index of the table is used in order to make searches much faster than when searching using RecordSet (the Seek method is applied to the table instead of the FindFirst method of RecordSet based on the table). By default, the index is automatically defined to PrimaryKey. This method is not allowed in the case of attached tables, for queries or for tables with no index or with several indexes (except the AutoNumber identifier field). The index may be a unique combination of several fields, in the same way as for Unique fields. In this case, the order of the fields in the index must be respected. See also Enabling index Enabling indexes and Disabling all index. Important remark about the backup test methodology : enabling the index should be considered only at the optimization stage, and only once the backup system works . It can also happen that a duplicate key in index error occurs during loading although nothing has changed in table settings ! it is possible that some locks remain after a crash (only during backup debugging) and that this causes some inexplicable errors ; in this case, exit the software containing ActiveX control (you may also have to reboot the computer if the problem persists) ;

* Enabling index (bUseIndex, relates only to loading speed optimization) : Boolean to enable the use of the index at specific table . When the uniqueness of the Unique fields does not correspond to an existing index in the table, it is necessary to disable record searching using the Indexes method. Example : for the table Employees, it's a good idea to set uniqueness constraint to LastName+FirstName, so these records are not duplicated each time the file is loaded. However, there is no corresponding index to this combination in the table, thus bUseIndex boolean must be left at false in this case.

* Structure array (special version of the ActiveX control, VB6StructName, relates to loading and backup speed optimization) : to optimize the backup of a great number of records, it is possible to write source code of a structure defining a particular data type and to build an array of this data type. Saving and loading this array will be achieved using only one BASIC instruction, respectively Put# and Get#, which is much faster than the field per field and record per record method ; on the other hand, this method requires specific source code which must be compiled in a DLL, thus losing the implicit upward compatibility (upgradeability) of the backup files if the database structure is upgraded. To enable backup using a structure array, get the special version of DBToFile.ocx control and specify the name of the structure in the VB6StructName field of the table.

- Properties of DBToFile.ocx control :

* bNewRecords (relates only to loading speed optimization) : when a backup file is loaded into an empty database, all records are necessarily new (this also applies in the file, which contains no redundant records), and it is then not necessary to check record uniqueness. This option thus gains time only in this example; in the other examples, this option will probably produce a duplicate key in index error.

* bIDsFieldNameEndsByID (relates to loading and backup speed optimization) : names of identifier fields all ends in ID; this applies to DBToFile.mdb tables (it would be also the case for Northwind.mdb tables if there were not a ShipVia field in Orders table ! It would have been better to use rather ShipperID, or not to save Shippers table by removing the NWSHIP option at the time of saving, see HTML source code of Demonstration) ;

* bIDsFieldNameStartsByID (relates to loading and backup speed optimization) : names of identifier fields all begins with ID (or Id), for example IdCustomer, as well as mnemonic fields, for example MnCustomer (French writing style) ;

* Index disabling (bNeverUseIndex, relates only to loading speed optimization) : to disable fast search of record all tables, for example in the case of attached tables, or for debugging the backup system (see also Speed optimization).

Technical limitations

- Primary mnemonic field : it can be edited only if the relations respecting referential integrity based on this field authorize the "Cascade Update Related Fields" option. For example [Code client] (CustomerID) in the original Access 97 French version Comptoir.mdb can't be edited (but Northwind.mdb is Ok, see Installation). Generally, it is necessary to avoid primary fields being text fields, it is better to create an AutoNumber identifier field for the primary key; related error number 3200 : impossible to modify a record, because table X (child table) includes related records;

- Cyclic relation between tables : when there is a cyclic (or circular) relation with referential integrity between two tables, those tables cannot be saved. Indeed, it is impossible to determine which of the two tables must be saved first, because each one requires that the other one must be saved first in order to first attach the identifier field , which is then impossible. In the case of a cycle with more than two tables, it is possible to save the tables of the cycle in some cases that are rather complicated to describe : for example, in DBToFile database, there is a link between DBToFile_SaveFileType and DBToFile_SQL, between DBToFile_SQL and DBToFile_SelectedTable and also between DBToFile_SelectedTable and DBToFile_SaveFileType. Which table is it necessary to save first ? Answer : DBToFile_Main table is saved first. DBToFile_SaveFileType table is then saved as a child table of DBToFile_Main as it is specified in DBToFile_ChildTable table. DBToFile_SelectedTable and DBToFile_SQL tables are also child tables of DBToFile_SaveFileType. DBToFile_SelectedTable will be saved first if a record is referring to it is in DBToFile_SQL.

Dependent files of the ActiveX control

```
DBToFile files :
--------------
Exemples.htm
DBToFile.htm
DBToFile.doc
DBToFile.ocx
DBToFile.mdb
Comptoir.mdb (see original Comptoir.mdb database sample)
DBToFileEng.htm     (English presentation)
DBToFileEng.doc     (English documentation)
Demo.htm            (English demonstration)
```

The following list gives the components on which the DBToFile utility software depends (this can be useful if DBToFile ActiveX control is distributed in another package) :

```
Runtime VB6, just to copy into Windows\System\ :
```

```
-----------
MSVBVM60.DLL
VB6FR.DLL

DAO and system files, setup procedure is required :
---------
[Bootstrap Files]
VB6STKIT.DLL,$(WinSysPathSysFile),,,3/26/99 12:00:00 AM,101888,6.0.84.50
Vb6fr.dll,$(WinSysPath),,$(Shared),7/13/98 12:00:00 AM,119568,5.0.81.69
COMCAT.DLL,$(WinSysPathSysFile),$(DLLSelfRegister),,6/1/98 12:00:00 AM,22288,4.71.1460.1
stdole2.tlb,$(WinSysPathSysFile),$(TLBRegister),,5/25/99 12:00:00 AM,16896,2.40.4501.1
asycfilt.dll,$(WinSysPathSysFile),,,5/25/99 12:00:00 AM,147728,2.40.4501.1
olepro32.dll,$(WinSysPathSysFile),$(DLLSelfRegister),,5/25/99 12:00:00
      AM,164112,5.0.4501.1
oleaut32.dll,$(WinSysPathSysFile),$(DLLSelfRegister),,5/25/99 12:00:00
      AM,606480,2.40.4501.1
MSVBVM60.DLL,$(WinSysPathSysFile),$(DLLSelfRegister),,5/10/99 12:00:00
      AM,1384448,6.0.84.95

[Setup Files]
MSPRPFR.DLL,$(WinSysPath),,$(Shared),7/13/98 12:00:00 AM,7680,6.0.81.63
MSSTKPRP.DLL,$(WinSysPath),$(DLLSelfRegister),$(Shared),6/18/98 12:00:00
      AM,94208,6.0.81.69
VB5DB.DLL,$(WinSysPath),,$(Shared),6/18/98 12:00:00 AM,89360,6.0.81.69
msjtes40.dll,$(WinSysPathSysFile),$(DLLSelfRegister),,5/25/99 12:00:00
      AM,237840,4.0.2521.8
msrepl40.dll,$(WinSysPathSysFile),,,5/25/99 12:00:00 AM,553232,4.0.2521.9
msrd3x40.dll,$(WinSysPathSysFile),$(DLLSelfRegister),,5/25/99 12:00:00
      AM,315664,4.0.2521.8
msrd2x40.dll,$(WinSysPathSysFile),$(DLLSelfRegister),,5/25/99 12:00:00
      AM,422160,4.0.2521.9
mswdat10.dll,$(WinSysPathSysFile),,,5/25/99 12:00:00 AM,831760,4.0.2521.8
mswstr10.dll,$(WinSysPathSysFile),,,5/25/99 12:00:00 AM,614672,4.0.2521.8
expsrv.dll,$(WinSysPathSysFile),,,4/14/99 12:00:00 AM,379152,6.0.0.8269
vbajet32.dll,$(WinSysPathSysFile),,,5/25/99 12:00:00 AM,30992,6.0.1.8268
msjint40.dll,$(WinSysPathSysFile),,,5/25/99 12:00:00 AM,184592,4.0.2521.8
msjter40.dll,$(WinSysPathSysFile),,,5/25/99 12:00:00 AM,53520,4.0.2521.8
msjet40.dll,$(WinSysPathSysFile),$(DLLSelfRegister),,5/25/99 12:00:00
      AM,1499408,4.0.2521.8
dao360.dll,$(MSDAOPath),$(DLLSelfRegister),$(Shared),5/25/99 12:00:00
      AM,557328,3.60.2521.8
```

DBToFile.mdb Tables

For information, here is a sample file of the DBToFile.mdb database backup (see Demonstration) :

```
Table=DBToFile_Main
FileTypeID=7
IDField=FileTypeID
UniqueFields=FileType
bNotify=-1
NotifyField=FileType
bUseIndex=-1
IndexField=FileType

Table=DBToFile_Table
FileTypeID=7
IDField=TableID
UniqueFields=Table+FileTypeID
bNotify=-1
NotifyField=Table
IndexField=IdxTable

Table=DBToFile_ChildTable
FileTypeID=7
UniqueFields=TableID+ChildTableID
bUseIndex=-1
IndexField=IdxChildTable

Table=DBToFile_EqID
FileTypeID=7
UniqueFields=TableID+EqIDTableID
bUseIndex=-1
IndexField=IdxEq

Table=DBToFile_SaveFileType
FileTypeID=7
IDField=SaveFileTypeID
MnemonicField=SaveFileTypeMn
UniqueFields=SaveFileTypeMn
bNotify=-1
NotifyField=SaveFileTypeMn=SaveFileType
IndexField=SaveFileTypeMn

Table=DBToFile_SelectedTable
FileTypeID=7
```

```
IDField=SelectedTableID
UniqueFields=SaveFileTypeID+TableID
bUseIndex=-1
IndexField=IdxSelectedTable

Table=DBToFile_SQL
FileTypeID=7
UniqueFields=SaveFileTypeID+Number
bUseIndex=-1
IndexField=IdxSQL
```

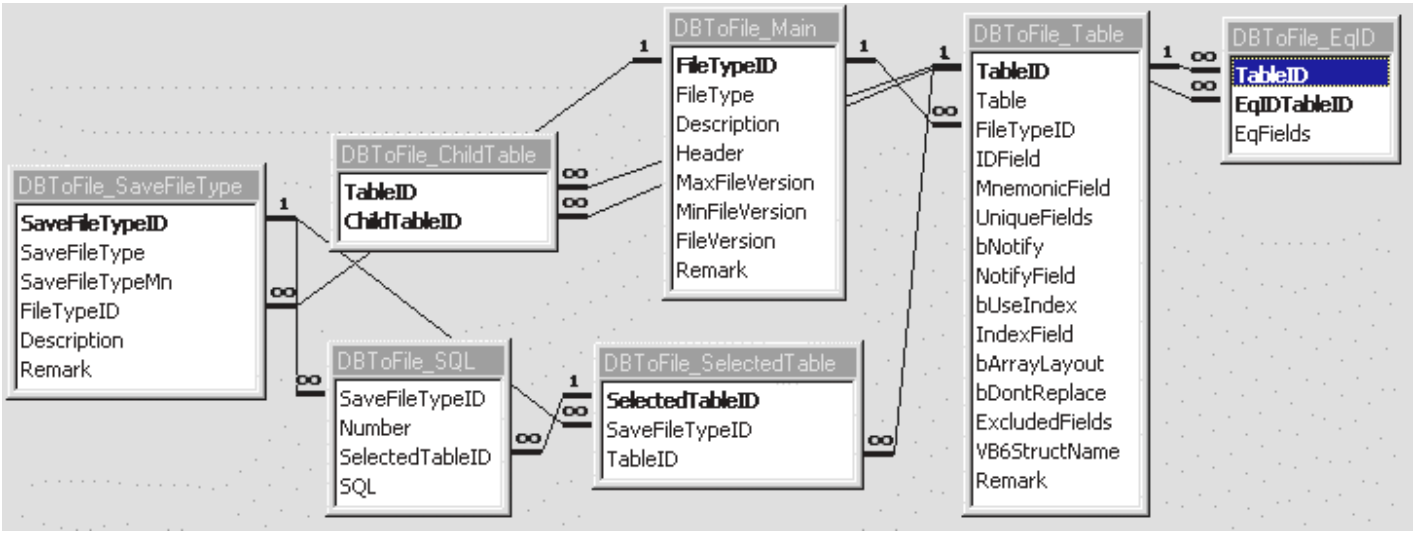
Saving query of DBToFile.mdb :

```
SQL=SELECT * FROM DBToFile_Main WHERE FileTypeID=[FileTypeID];
```

[FileTypeID] will be replaced by the value found in the property strSaveFileArg of ActiveX control before executing the request.

DBToFile.mdb Relations

(for information)



Report of the DBToFile database structure

For information, here is the database report created at stage 17 of the demonstration :

```
DBToFile Database Reporting : http://perso.worldonline.fr/ors/dbtofile/index.html
Database : C:\Program Files\DBToFile\Samples\DBToFile.mdb
Warning : some index may be relation's index !

DBToFile_ChildTable : Tables to be save if a record from a parent table is saved / Tables devant
être sauvées lorsqu'un enregistrement d'une table parent est sauvé
  TableID : Identifier of the parent table, Ex.: Orders
  ChildTableID : Identifier of the child table, Ex.: [Order Details]
    Index   : {94177CEA-A8C4-11D3-8214-BA8E2F36C603}
      field : TableID
    Index   : {94177CEB-A8C4-11D3-8214-BA8E2F36C603}
      field : ChildTableID
    Index   : ChildTableID
      field : ChildTableID
    Index   : IdxChildTable, 2 fields, Unique, Primary
      field : TableID
      field : ChildTableID
    Index   : TableID
      field : TableID
```

```
DBToFile_EqID : Corresponding identifier field names between tables / Table de correspondance des
noms de champ identifiant entre tables
  TableID : Identifier of the table, Ex.: table Orders
  EqIDTableID : Identifier of the corresponding table, Ex.: table Shippers
  EqFields : List of fields corresponding to a identifier field name of a table, Ex.: ShipVia in
table Orders
    Index   : {94177CEC-A8C4-11D3-8214-BA8E2F36C603}
      field : TableID
    Index   : {94177CED-A8C4-11D3-8214-BA8E2F36C603}
      field : EqIDTableID
    Index   : EqIDTableID
      field : EqIDTableID
    Index   : IdxEq, 2 fields, Unique, Primary
      field : TableID
      field : EqIDTableID
    Index   : TableID
      field : TableID

DBToFile_Main : Table of backup file types / Table des types de fichiers d'archives
  FileTypeID : Identifier of backup file type
  FileType : Name of backup file type
  Description : Description of backup file type
  Header : File header of backup file type
  MaxFileVersion : Max file version that can be load (max compatible version)
  MinFileVersion : Min file version that can be load (min compatible version)
  FileVersion : Actual file version
  Remark : Remark
    Index   : FileType, Unique
      field : FileType
    Index   : PrimaryKey, Unique, Primary
      field : FileTypeID

DBToFile_SaveFileType : Description of save file types / Tables des références de sauvegarde
  SaveFileTypeID : Identifier of the save file type
  SaveFileType : Save file type
  SaveFileTypeMn : Mnemonic code of the save file type used in save arguments
  FileTypeID : Identifier of backup file type
  Description : Description of the save file type included in the backup file header
  Remark : Remark
    Index   : {94177CE5-A8C4-11D3-8214-BA8E2F36C603}
      field : FileTypeID
    Index   : PrimaryKey, Unique, Primary
      field : SaveFileTypeID
    Index   : SaveFileTypeMn, Unique
      field : SaveFileTypeMn

DBToFile_SelectedTable : Selected tables for each save file type / Tables sélectionnées dans
chaque référence de sauvegarde
  SelectedTableID : Identifier of the selected table
  SaveFileTypeID : Identifier of the save file type
  TableID : Identifier of the table
    Index   : {94177CE7-A8C4-11D3-8214-BA8E2F36C603}
      field : SaveFileTypeID
    Index   : {94177CEE-A8C4-11D3-8214-BA8E2F36C603}
      field : TableID
    Index   : IdxSelectedTable, 2 fields, Unique
      field : SaveFileTypeID
      field : TableID

    Index   : PrimaryKey, Unique, Primary
      field : SelectedTableID
    Index   : SaveFileTypeID
      field : SaveFileTypeID
    Index   : TableID
      field : TableID
```

```
DBToFile_SQL : List of SQL queries to select records to be save in backup file / Liste des
requêtes SQL de sélection des enregistrements à archiver
  SaveFileTypeID : Identifier of the save file type
  Number : Order number of SQL query to respect if necessary referential integrity rules during
loading
  SelectedTableID : Identifier of the selected table
  SQL : SQL query to select records to be save in backup file
  QueryName : Name of an Access query to be used instead of SQL string -Future OCX version-
    Index   : {94177CE8-A8C4-11D3-8214-BA8E2F36C603}
      field : SaveFileTypeID
    Index   : {94177CE9-A8C4-11D3-8214-BA8E2F36C603}
      field : SelectedTableID
    Index   : IdxSQL, 2 fields, Unique
      field : SaveFileTypeID
      field : Number
    Index   : Number
      field : Number
    Index   : SaveFileTypeID
      field : SaveFileTypeID
    Index   : SelectedTableID
      field : SelectedTableID

DBToFile_Table : Description of the tables of the backup / Description des tables de l'archivage
  TableID : Identifier of the table
  Table : Name of the table
  FileTypeID : Identifier of backup file type
  IDField : Name of the identifier field of the table, Ex.: "OrderID" in table Orders
  MnemonicField : Name of the mnemonic field of the table, Ex.: "CustomerID" in table Customers
  UniqueFields : List of unique fields, Ex.: "OrderID+ProductID" in table [Order Details]
  bNotify : Boolean to notify loading (saving) a record into (from) a table
  NotifyField : Name of the field to notify, Ex.: "CompanyName=Shipper" in table Shippers
  bUseIndex : Boolean to use or not index of the table for seeking existing records while loading
to speed up processing
  IndexField : Name of the index field of the table which is unique
  bArrayLayout : Boolean to write and read array of records in a single line to speed up
processing
  bDontReplace : Boolean to avoid replacement of records into this table -Futur OCX version-
  ExcludedFields : List of fields to be excluded into to file, Ex.: Photo in table Employees
  VB6StructName : Name of a Visual Basic 6 Type to speed up processing -Special OCX version-
  Remark : Remark
    Index   : {94177CE6-A8C4-11D3-8214-BA8E2F36C603}
      field : FileTypeID
    Index   : IdxTable, 2 fields, Unique
      field : Table
      field : FileTypeID
    Index   : PrimaryKey, Unique, Primary
      field : TableID
```

Additional samples

Backup samples containing many complex queries for selecting the records to be saved to the file :

http://perso.worldonline.fr/ors/dbtofile/samples.zip

History of the DBToFile software utility

Origin

The idea of saving a sequence of linked records from a database to a file so that it can subsequently be

reloaded later into a database with the same structure goes back to Herve Gaucher, software manager and developper at LIM Geotechnologies (www.limgeo.com). As a software developer for several years in this company (which manufactures drilling parameter recorders) I had to extend this principle to new data types added to the structure of a geotechnical database; this gave the idea of extending the backup system to an unspecified database structure. To isolate the source code from the database structure, an elementary syntax described in character strings is used to isolate the backup mechanism from the description of the table list, in particular for unique fields and for fields that trigger cascade saving.

Development

This made it possible to build the backup system for two distinct versions of the geotechnical database with the same backup engine. In addition, analysing the backup files made it possible to convert one version to the other using an algorithm usually known by the name of of "moulinette", meaning that a great deal of source code is necessary for this operation which is of similar complexity to a true translation from one language into another.

Generalization

Thereafter, with the aim of generalizing the backup system to any database, I personally created the following improvement to the system. Settings of tables to be saved and queries for selecting the data to be saved are stored in Access tables in order to make it possible to update the backup system without modifying the source code. In addition, this made it possible for the backup system itself to be saved, because it is also MS-Access data (see the stages from 14 to 17 of Demonstration : DBToFile meta-backup).

All searches on character strings were replaced using indexed Visual Basic collections in order to optimize the backup speed.

The backup system source code has been compiled as an ActiveX control, which can be used with or without a visible interface, thanks to its methods and properties, in a large number of development environments supporting ActiveX controls.

Future developments

- Wizard that analyzes the structure of the database to be saved and creates settings of the tables to be saved ; interface for checking and editing DBToFile.mdb tables ;

- Multilingual version using a DLL with the source code to add a language ;

- Optimization with structure array in Visual BASIC 6, see Structure array ;

- Multiple indexes support for a table (see also Index field) ;

- Option of disabling record replacement of a table (field bDontReplace of DBToFile_Table) ;

- Generalization to other databases using ADO (Active Data Object) ;

- Backup queries stored (and so compiled) in the database and referenced in the QueryName field in the

DBToFile_SQL table ;

Email & Web

patrice.dargenton@worldonline.fr

http://perso.worldonline.fr/ors/dbtofile/index.html