DBToFile : L'Archiveur de base de données

A Microsoft Access Database Backup Engine

Présentation

Description

DBToFile est un système d'archivage d'un enchaînement d'enregistrements liés dans un fichier séparé de la base de données (sous forme texte et binaire pour les champs binaires de la base tels que les images, documents OLE …) en vue de le recharger ultérieurement dans une base de structure identique.

Versions disponibles de ce document

Ce document est disponible à l'adresse :

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

en français et en anglais dans les formats :

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

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

Avantages

- Les fichiers d'archives peuvent être transmis par disquette (ce sont de petits fichiers principalement de type texte qui peuvent en outre être compressé) ou bien par Internet sous forme de document attaché à un courrier électronique.

- Ce système peut être utilisé en guise de réplication hors ligne d'un jeu de plusieurs bases vers une seule base principale (puis réciproquement pour que la synchronisation des données s'effectue dans les deux sens) grâce à des fichiers de mise à jour, par exemple une base de clients pour une équipe de représentants de commerce.

- Il est possible de préciser ce qui doit figurer dans le fichier et ce qui doit être exclus, au niveau des tables et des champs des tables, voir Champs exclus.

- Les confirmations peuvent être désactivées afin d'automatiser le chargement et la sauvegarde ;

- DBToFile est un contrôle ActiveX, il peut donc être utilisé dans de nombreux environnements de développement, ou bien dans une page HTML, voir Démonstration.

- Si vous distribuez une application de base de données Access, vous pouvez proposer le système d'archivage simplement en insérant le contrôle ActiveX dans un formulaire Access, Visual Basic ou HTML, et en préparant les définitions des tables ainsi que les requêtes d'archivage des enregistrements de la base ; voir aussi Fichier de dépendance du contrôle ActiveX ;

- L'interface du contrôle ActiveX peut être invisible, car l'équivalent des boutons de commande est proposé sous forme de méthode du contrôle ;

- Des requêtes complexes sont possibles pour la sélection des enregistrements à archiver ;

- Exemples d'application liée à Internet : système de vote ou d'enquête par courrier électronique, système pour remplir en hors ligne un formulaire de base de données, …

Spécifications

- DBToFile peut archiver les bases Access 2, 95, 97 et 2000, voir aussi Développement futur avec ADO) ;

- La gestion de la sécurité est prise en compte pour accéder aux données à archiver en lecture / écriture (il est possible de préciser un compte utilisateur avec son mot de passe) ;

- DBToFile requiert les composants VBAJet32, le Runtime Visual Basic 6 et DAO 3.6, voir Installation

Limitations

- La définition des tables n'est pas automatique, il faut décrire l'ensemble des tables que l'on veut archiver

dans une table prévue à cet effet (DBToFile_Table de la base DBToFile.mdb). Cependant une documentation de la base de données peut être générée automatiquement pour faciliter cette étape, voir étapes 9 et 17 de la Démonstration. Une interface conviviale est prévue dans une version future ;

- Le fichier d'archives n'est pas destiné à être importé directement dans un logiciel de base de données ou tableur, car les données archivées dans ce fichier peuvent concerner la totalité des tables de la base, et elles peuvent intégralement être rechargées dans une base de structure identique sans perdre les liens entre les enregistrements. Cela est donc plus sophistiqué que la simple exportation d'une table (quoique cela aussi soit possible).

- Il n'est pas possible de restaurer l'état de la base après la validation du chargement (prévoir un backup si nécessaire), mais il y a une confirmation des remplacements.

- Champ index unique : il ne peut y avoir qu'un champ unique par table (ou une combinaison unique de champs correspondant à un index de la table) hormis le champ identifiant en numérotation automatique, car on ne peut éditer plusieurs enregistrements en même temps lors du chargement (si plusieurs champs uniques existent déjà dans la table pour plusieurs enregistrements distincts, il sera impossible de charger le fichier). Par exemple, dans la table Client, il n'est pas possible d'avoir simultanément [Code client] et Client comme champ unique.

Voir aussi les limitations techniques.

Fonctionnement

- Comparaison avec la réplication : le contrôle de la réplication est simplifié car il se réduit à confirmer le remplacement éventuel de certains enregistrements (il n'y a pas de règles de résolution de conflit lors du remplacement des enregistrements). Les champs qui vont être remplacés (ceux pour lesquels une notification à été demandée) sont affichés dans une boite de dialogue sous forme d'une liste déroulante, l'utilisateur est invité à confirmer ou bien à annuler l'ensemble de ces remplacements. Il n'est pas possible de voir la valeur des champs qui vont être remplacés (à moins de regarder le contenu du fichier).

Remarque : les exemples dans la version originale (en langue anglaise) Northwind.mdb sont présentés entre parenthèses après ceux de Comptoir.mdb dans les paragraphes suivants :

- Un système de vérification des identifiants est prévu pour éviter l'archivage à plusieurs reprises du même enregistrement. Par exemple, si on sauvegarde les nouvelles commandes du jour de la table Commandes (Orders), et si plusieurs commandes se réfèrent à un même article de la table Produits (Products) via la table [Détails commandes] ([Order Details]), l'article ne sera sauvé qu'une seule fois.

- Champs particuliers de la table DBToFile_Table de la base de contrôle de l'archivage DBToFile.mdb :

* Champs uniques (UniqueFields, concerne le chargement des fichiers) : il est nécessaire de préciser le champ unique lorsqu'il y a un index unique sur la table. Au moment de recharger un fichier, une recherche de l'enregistrement unique est effectuée afin d'éviter l'erreur de doublon dans l'index. Le cas échéant, l'enregistrement est alors remplacé (fusionné) par l'enregistrement correspondant du fichier. L'index peut comporter une combinaison de plusieurs champs, c'est alors la combinaison entière qui doit respecter la contrainte d'unicité, comme c'est le cas pour l'index de la table. En revanche, un seul index unique est autorisé, voir la limitation concernant les champs index uniques ainsi que la limitation technique sur les champs mnémoniques primaires. Il est possible de préciser un champ (ou bien une

combinaison) unique même s'il n'y a pas d'index correspondant dans la table. Cela évite que des enregistrements soient dupliqués à chaque chargement de fichier. Par exemple, dans la table des Produits (Products), il vaut mieux préciser que [Nom du produit] (ProductName) doit être unique dans la définition de la table, même s'il n'y a pas de contrainte d'unicité sur le champ de la table. Des tests sur une base vide sont nécessaires pour détecter ce défaut, voir la Démonstration ;

* Champ identifiant de la table (IDField, concerne la sauvegarde et le chargement des fichiers). Ce champ permet d'établir un lien simple entre deux tables (une référence de type numérique). Par exemple, la table Employés (Employees) aura pour identifiant [N° employé] (EmployeeID). Cet identifiant sera utilisé pour la commande d'un employé dans la table des Commandes (Orders).

* Champ mnémonique de la table (MnemonicField, concerne la sauvegarde et le chargement des fichiers). Le principe est le même que pour le champ identifiant, mais cette fois la référence n'est plus un entier long généré automatiquement par le moteur de base de données, mais un champ texte court défini par l'utilisateur (un texte mnémotechnique). Cela permet des références claires entre deux tables, ce qui est indispensable dans le cas de référence entre des tables de base différentes, ou bien dans le cas d'archivage facultatif de certaines tables : il est plus facile de retrouver un champ mnémonique de type texte qu'un champ numérique abstrait (surtout un numéro généré automatiquement). Voir aussi la limitation technique sur le Champ mnémonique primaire. Exemple : la table des clients (Customers) à pour champ mnémonique [Code Client] (CustomerID), qui est un champ texte et non un identifiant numérique.

- Tables de la base de contrôle de l'archivage DBToFile.mdb concernant la sauvegarde des fichiers :

* Table des identifiants équivalents (DBToFile_EqID) : liste de champs identifiants correspondant à un même champ dans une autre table. Par exemple, si la liste des sociétés clientes et la liste des sociétés fournisseurs sont regroupées dans une même table Société et qu'une table fait référence simultanément à un client et à un fournisseur pour un même enregistrement Commande, alors les champs identifiants seront IdSociete_Client et IdSociete_Fournisseur et la liste des Identifiants équivalents sera : Societe=IdSociete_Fournisseur, IdSociete_Client.

* Tables dépendantes (ou enfants, DBToFile_ChildTable) : cette liste permet l'archivage automatique de tables dépendantes d'une table référencée dans un enregistrement archivé. Par défaut, seule la table référencée par son identifiant (ou un identifiant équivalent) est sauvée ; pour que les tables dépendantes soient également sauvées, il faut alors le préciser à l'avance dans cette table. Exemple : lors de l'archivage de la table Commandes (Orders), il faut également sauver le détail des commandes sélectionnées. Pour cela, on indique dans la table Commandes (Orders) que la table dépendante est [Détails commandes] ([Order Details]).

Voir aussi Documentation complémentaire ;

Installation

Téléchargement

- Si Visual Basic 6.0 (SP3) est déjà installé sur votre machine (en particulier VBAJet32 et DAO 3.6) essayez directement l'ocx dbtofile.zip (250 Ko) sans le package d'installation. Il suffit d'enregistrer l'ocx grâce à la commande C:\Windows\RegSvr32.exe DBToFile.ocx (si le fichier VB6Fr.Dll n'existe pas dans le répertoire C:\Windows\System\, copiez le dedans) ; Si toutefois vous rencontrez des erreurs (notamment 3446 ou 3447), vous devrez télécharger le package complet (voir aussi Fichier de

dépendance du contrôle ActiveX) :

- Sinon, téléchargez le package complet dbtofileinstall.zip (4 Mo), qui se chargera d'installer le Runtime Visual Basic 6.0, VBAJet32 et DAO 3.6 pour Windows 95 ou 98.

Décompression

Décompressez le fichier dbtofileinstall.zip ou dbtofile.zip dans un répertoire de votre choix, grâce par exemple à l'utilitaire PkZip Pour Windows.

Installation

- Lancez setup.exe, ou sinon lancer directement DBToFile.htm si l'ocx est déjà enregistré (voir Registration du contrôle).

Préparation

Pour tester la démonstration avec la version anglaise Northwind.mdb de Comptoir.mdb (fournie avec Access 97), vous pouvez télécharger la base à l'adresse suivante :

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

Puis copiez Northwind.mdb dans le répertoire d'installation du logiciel DBToFile, par défaut il s'agit de C:\Program Files\DBToFile.

La base exemple Comptoir.mdb d'origine (de Access 97) contient une relation d'intégrité référentielle sans mise à jour en cascade entre les tables Clients et Commandes, ce qui est sans doute une petite erreur de la part de Microsoft France, car cette relation est avec mise à jour en cascade sur la version anglaise Northwind.mdb (d'ailleurs la version Comptoir.mdb de Access 2000 est corrigée). Or, il n'est pas possible d'éditer les enregistrements de la table Client lorsqu'un champ mnémonique primaire a une relation d'intégrité référentielle sans mise à jour en cascade (voir la limitation technique sur le Champ mnémonique primaire). La base Comptoir.mdb fournie avec DBToFile est donc corrigée pour que la démonstration fonctionne (les photos ont été supprimées pour alléger l'installation).

Démonstration

Après avoir lu la rubrique Installation, regardez la démonstration du contrôle ActiveX DBToFile en HTML avec VBScript : Exemples.htm

Voici le texte descriptif des étapes de la démonstration :

Archivage de Comptoir.mdb
Méta-archivage de Comptoir.mdb
Méta-archivage de DBToFile.mdb

Introduction

Pour toutes les démonstrations, vous n'avez pas besoin d'utiliser les boutons du contrôle DBToFile, vous pouvez utiliser le bouton Démarrer ! Vous pourrez ensuite passer à l'étape suivante. Vous pouvez changer la langue et recommencer les étapes (de 1 à 13) avec l'autre base exemple Northwind.mdb qui est la version anglaise de Comptoir.mdb. Dans ce cas, le nom du fichier d'archives sera Order.ord au lieu de Commande.cmd (rappel : Northwind.mdb n'est pas inclus dans le package).

Archivage de Comptoir.mdb

1-Archivage d'une commande

1-Archivage d'une commande dans le fichier Commande.cmd : la commande dont le n° est 10248 est sélectionnée par une requête d'archivage ; automatiquement, tous les enregistrements relatifs en amont (ex.: Code client) ou en aval (ex.: Détail commandes) à cette commande vont également être archivés dans le fichier, en respectant l'ordre logique suivant : lorsque le fichier doit être rechargé dans une base de structure identique, les règles d'intégrité référentielle doivent être respectées. Les étapes suivantes de 2 à 7 visent à tester ce fichier d'archives.

2-Vérification de Commande.cmd

2-Vérification du fichier créé Commande.cmd : l'en-tête du fichier est analysé pour déterminer si le fichier peut être importé dans la base de donnée Comptoir.mdb. A ce stade, on vérifie qu'une référence de fichier d'archives existe bien dans la base de contrôle de l'archivage DBToFile.mdb, et que son numéro de version est compatible avec la version du fichier.

3-Test de chargement

3-Test de chargement du fichier créé Commande.cmd dans la même base Comptoir.mdb : on vérifie seulement qu'il n'y a pas d'erreur de doublons dans les index uniques ; les données étant identiques, il ne doit pas y avoir de modification du contenu de la base de données. Le n° de commande qui sera affiché après le chargement sera inchangé. Si vous effectuez des modifications dans la base Comptoir.mdb sur les enregistrements sauvés dans le fichier, ces modifications seront bien sûr perdues si vous recharger le fichier. Cependant vous pouvez répéter l'étape n°1 d'archivage pour remettre à jour le fichier d'archives avec ces modifications.

4-Copie de Comptoir.mdb

4-Copie de la base pour créer une base vide de test Comptoir0.mdb ; afin de vérifier que tous les enregistrements nécessaires figurent bien dans le fichier d'archives, nous allons effectuer une copie de Comptoir.mdb en Comptoir0.mdb, puis effacer son contenu à l'étape suivante.

5-Création de Comptoir0.mdb

5-Suppression des enregistrements de la base test Comptoir0.mdb ; cette nouvelle base sera vide et de structure identique à Comptoir.mdb. Ainsi les fichiers d'archives seront aussi compatibles avec cette base.

6-Chargement de Commande.cmd

6-Chargement du fichier créé Commande.cmd dans la base vide de test Comptoir0.mdb : les enregistrements vont être importés dans la base vide et seront donc notifiés entre parenthèses. Le n° de commande qui sera affiché après le chargement sera différent de celui du fichier de la base d'origine, car il est automatiquement attribué à un nouvel enregistrement dans une base vide.

7-Rechargement de Commande.cmd

7-Chargement à nouveau du fichier créé Commande.cmd dans la base test Comptoir0.mdb qui n'est plus tout à fait vide ; notez que les enregistrements sont remplacés (comme à l'étape n°3) et non plus importés. Le n° de commande qui sera affiché après le chargement sera identique à celui de l'étape précédante. Remarque importante concernant la méthodologie de test : il faut vérifier à cette étape que les enregistrements ne sont pas dupliqués à chaque rechargement du fichier dans la base, auquel cas il faudrait préciser les contraintes d'unicité que doivent respecter les enregistrements dans la table DBToFile_Table dans le champ UniqueFields de la base DBToFile.mdb. Pour cela, vérifiez le contenu de la base Comptoir0.mdb. Vous pouvez fixer une contrainte d'unicité même s'il n'y a pas d'index correspondant dans la table.

8-Visualisation de Commande.cmd

8-Visualisation du fichier créé Commande.cmd (pour information seulement). Vous ne pouvez pas modifier ce fichier lorsqu'il contient des données binaires (ex.: photos des employés), car sinon celles-ci ne pourraient plus être lues correctement.

9-Documentation de Comptoir.mdb

9-Documentation de la structure de la base de données Comptoir.mdb : Cette fonctionnalité du contrôle DBToFile est utile pour définir les tables d'archivage d'une base à laquelle vous souhaitez adjoindre le système d'archivage.

Méta-archivage de Comptoir.mdb

10-Méta-archivage de Comptoir.mdb

10-Archivage de l'archivage de Comptoir.mdb dans le fichier Comptoir.dbe : le système d'archivage d'une base particulière peut également être sauvé dans un fichier, puisqu'il est entièrement décrit à l'aide des tables Access de DBToFile.mdb. Il suffit d'indiquer la référence de l'archivage FileTypeID=5. L'intérêt est que dans le cas d'une base distribuée chez de nombreux clients, le système d'archivage peut être mis à jour sans modifier le code. Cependant, cet avantage ne peut concerner qu'une nouvelle sélection des enregistrements (ou bien une correction de l'archivage), car si la structure de la base change, il y aura de toute façon une mise à jour importante à faire. Les étapes suivantes 11 et 12 visent à tester ce fichier d'archives.

11-Création de DBToFile0.mdb

11-Création d'une base vide de test DBToFile0.mdb.

12-Chargement de Comptoir.dbe

12-Chargement du fichier Comptoir.dbe dans la copie vide DBToFile0.mdb.

13-Visualisation de Comptoir.dbe

13-Visualisation du fichier créé Comptoir.dbe (pour information seulement).

Méta-archivage de DBToFile.mdb

14-Méta-archivage de DBToFile.mdb

14-Archivage de l'archivage de DBToFile.mdb dans le fichier DBToFile.dbe : l'intérêt est d'illustrer le cas de l'autoréférence et de donner un exemple supplémentaire d'archivage à des fins didactiques. Ce système pourrait faciliter une mise à jour de l'utilitaire DBToFile en évitant de perdre les définitions d'archivage de ceux ayant déjà utilisé une version précédante de l'utilitaire. Cependant, ce fichier ne pourrait pas être rechargé sans préciser au préalable l'en-tête dans la table DBToFile_Main ainsi que la liste des tables d'archivage dans la table DBToFile_Table de la base DBToFile.mdb.

15-Chargement de DBToFile.dbe

15-Chargement du fichier créé DBToFile.dbe dans une copie vide DBToFile0.mdb.

16-Visualisation de DBToFile.dbe

16-Visualisation du fichier créé DBToFile.dbe (pour information seulement).

17-Documentation de DBToFile

17-Documentation de la structure de la base de données DBToFile (pour information seulement).

Fin de la démonstration

Fin de la démonstration. Merci de votre attention. Pour plus d'information, consultez la documentation jointe au logiciel.

Documentation complémentaire

Remarques diverses

- Cas d'une mise à jour de la structure de la base de données : l'archivage tolère facilement des mises à jour de la structure de la base (un contrôle de version des fichiers est prévu) telle que l'ajout d'un champ. En cas de suppression d'un champ ou bien dans le cas où un champ à été renommé, les anciens fichiers seront quand même chargés (si la version est comprise dans l'intervalle >=MinFileVersion et <MaxFileVersion de la table DBToFile_Main), et un message d'erreur indiquera que ce champ à disparu de la base et que sa valeur sera ignorée. Les nouveaux champs seront traités dans les nouveaux fichiers d'archives, le plus souvent sans aucune modification des tables d'archivage (sauf dans le cas peu fréquent d'une requête de sélection dans laquelle il n'y aurait pas SELECT * FROM … mais SELECT table1.champ1, table1.champ2 FROM …). Ces requêtes de sélection sont utilisées seulement pour initier un enchaînement de sauvegardes en cascade, ces cascades d'enregistrements étant automatiques si les tables d'archivage : DBToFile_ChildTable (voir Tables dépendantes) et DBToFile_EqID (voir Table des identifiants équivalents) sont correctement renseignées.

- Champ particulier de la table DBToFile_Table de la base de contrôle de l'archivage DBToFile.mdb :

* Champs exclus (ExcludedField) : par défaut, tous les champs d'une table sont sauvés. Pour éviter l'archivage d'un champ donné d'une table, il faut préciser ce champ dans la liste des champs exclus. Cela est utile dans le cas d'un champ binaire estimé trop gros ou bien d'un champ temporaire qui n'a pas de raison d'être conservé. Exemple : le champ Photo de la table Employées (Employees).

Optimisation en vitesse

Méthodologie : l'optimisation en vitesse ne doit être considérée qu'une fois que le système d'archivage est correctement configuré.

- Champs particuliers de la table DBToFile_Table de la base de contrôle de l'archivage DBToFile.mdb :

* Présentation en mode tableau (bArrayLayout, concerne l'optimisation du chargement et de la sauvegarde) : présentation d'un enregistrement sur une seule ligne afin d'optimiser l'archivage et le chargement. Cette technique est moins efficace que celle des Tableaux prédéfinis mais elle ne présente pas ses inconvénients.

* Champ index (IndexField, concerne l'optimisation du chargement) : il s'agit du même principe

que les Champs uniques, mais cette fois on utilise l'index au niveau de la table elle-même afin de faire des recherches beaucoup plus rapide que des recherches sur des RecordSet (la méthode Seek est appliquée sur la table au lieu de la méthode FindFirst sur le RecordSet fondé sur la table). Par défaut, l'index est automatiquement définit en PrimaryKey (clé primaire). Cette méthode n'est pas permise dans le cas des tables attachées, des requêtes et des tables sans index ou avec plusieurs indexes (hormis le champ identifiant en numérotation automatique). Une combinaison de plusieurs champs uniques peut être définie pour un index, de la même façon que pour les Champs uniques. Dans ce cas, l'ordre des champs de l'index doit être respecté. Voir Activation de l'index et Désactivation de tous les index. Remarque importante sur la méthodologie de test de l'archivage : n'activer les index qu'au moment de l'optimisation, une fois que l'archivage marche dans tous les cas de figure. Il peut arriver aussi qu'une erreur de doublons dans les index survienne au moment du chargement alors que rien n'a changé dans la définition des tables ! il est possible que des verrous subsistent après un plantage (seulement lors de la mise au point de l'archivage) et que cela entraîne des erreurs inexplicables ; dans ce cas, quittez l'application contenant le contrôle ActiveX (éventuellement, rebooter la machine si le problème persiste) ;

* Activation de l'index (bUseIndex, concerne l'optimisation du chargement) : Booléen pour activer l'utilisation de l'index au niveau d'une table particulière. Lorsque l'unicité choisit avec la rubrique Champs uniques ne correspond pas à un index existant dans la table, il est nécessaire de ne pas activer la recherche des enregistrements par la méthode des Indexes. Exemple : pour la table Employés (Employees), on veut que l'unicité soit déterminée par Nom+Prénom (LastName+FirstName), afin que ces enregistrements ne soient pas dupliqués lors du chargement d'un fichier. Or il n'y a pas d'index correspondant à cette combinaison dans la table, le booléen doit donc est désactivé dans ce cas.

* Tableau prédéfini (version spéciale du contrôle, VB6StructName, concerne l'optimisation du chargement et de la sauvegarde) : Pour optimiser l'archivage d'un grand nombre d'enregistrements, on peut écrire le code source pour définir une structure définissant un type de donnée particulier et construire un tableau de ce type. L'avantage est que l'archivage et le chargement de ce tableau s'effectueront à l'aide d'une seule instruction basic Put# et Get#, ce qui est beaucoup plus rapide que la méthode champ par champ et enregistrement par enregistrement ; mais l'inconvénient est que cette méthode requiert du code spécifique qui doit être compilée dans une Dll, ce qui a pour conséquence de perdre la compatibilité ascendante implicite (évolutibilité) des fichiers d'archives en cas de mise à jour de la structure de la base de données. Pour activer l'archivage avec un tableau prédéfini, procurez-vous la version spéciale du contrôle DBToFile.ocx et préciser le nom de la structure dans la définition de la table.

- Propriétés du contrôle DBToFile.ocx :

* bNewRecords (concerne l'optimisation du chargement) : lorsque l'on charge un fichier d'archives dans une base vide, tous les enregistrements sont forcement nouveaux (c'est le cas aussi dans le fichier, qui ne contient pas d'enregistrements redondants), et il n'est pas nécessaire alors de rechercher l'unicité des enregistrements. Cette option fait donc gagner du temps dans ce cas de figure seulement, dans les autres cas, cette option entraînera une erreur de doublon dans les index.

* bIDsFieldNameEndsByID (concerne l'optimisation du chargement et de la sauvegarde) : Les noms des champs identifiants se terminent tous par ID, c'est le cas de la table DBToFile.mdb (cela serait le cas dans Northwind.mdb s'il n'y avait pas ShipVia dans la table Orders ! il aurait fallu utiliser ShipperID, ou bien ne pas sauver la table Shippers en enlevant l'option NWSHIP lors de l'archivage : voir le code source de l'Exemple HTML) ;

* bIDsFieldNameStartsByID (concerne l'optimisation du chargement et de la sauvegarde) : Les noms des champs identifiants commencent tous par ID (ou Id), par exemple IdClient, ainsi que les champs mnémoniques, par exemple MnClient ;

* Désactivation des index (bNeverUseIndex, concerne l'optimisation du chargement) : pour désactiver la recherche rapide des enregistrements pour l'ensemble des tables, par exemple dans le cas de tables attachées, ou bien pour la mise au point du système d'archivage (voir aussi Optimisation en vitesse).

Limitations techniques

- Champ mnémonique primaire : il ne peut être édité que si les relations basées sur ce champ autorise la mise à jour en cascade, ce qui n'est pas le cas par exemple de [Code client] dans la version d'origine de Comptoir.mdb (la version anglaise Northwind.mdb est correcte, voir Installation). D'une manière générale, il faut éviter de mettre les champs primaires sur des champs textes, il vaut mieux créer un champ NuméroAuto numérique pour la clé primaire ; Erreur relative n°3200 : impossible de modifier un enregistrement, car la table X (table liée) comprend des enregistrements connexes ;

- Relation cyclique entre table : lorsqu'il existe une relation cyclique (ou circulaire) avec intégrité référentielle entre deux tables, celles-ci ne peuvent être archivées. En effet, on ne peut pas déterminer laquelle des deux tables doit être sauvée en premier, car chacune d'elle nécessite que l'autre soit archivée en premier afin de fixer le champ identifiant au préalable, ce qui est alors impossible. Dans le cas d'un cycle avec plus de deux tables, il est possible d'archiver les tables du cycle dans certains cas assez compliqués à décrire : par exemple, dans la base DBToFile, il y a un lien entre DBToFile_SaveFileType et DBToFile_SQL, entre DBToFile_SQL et DBToFile_SelectedTable et aussi entre DBToFile_SelectedTable et DBToFile_SaveFileType. Quelle table faut-il sauver en premier ? Réponse : La table DBToFile_Main est sauvée en premier. Puis la table DBToFile_SaveFileType est sauvée ensuite en tant que table fille de DBToFile_Main comme il est précisé dans la table DBToFile_ChildTable. Ensuite les tables DBToFile_SelectedTable et DBToFile_SQL sont aussi des tables filles de DBToFile_SaveFileType. DBToFile_SelectedTable sera sauvée en premier si un enregistrement est référencé dans DBToFile_SQL.

Fichier de dépendance du contrôle ActiveX

```
Fichiers DBToFile :
-----------------
Exemples.htm
DBToFile.htm
DBToFile.doc
DBToFile.ocx
DBToFile.mdb
Comptoir.mdb (voir base exemple Comptoir.mdb d'origine)
DBToFileEng.htm     (présentation en anglais)
DBToFileEng.doc     (documentation en anglais)
Samples.htm         (exemples en anglais)
```

La liste suivante présente les composants dont dépend l'utilitaire DBToFile (elle peut servir dans le cas de la redistribution du contrôle dans un autre package) :

```
Runtime VB6, une simple copie dans Windows\System suffit :
-----------
MSVBVM60.DLL
VB6FR.DLL

Fichiers DAO et système, une installation est requise :
------------
[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
```

Tables de DBToFile.mdb

Pour information, voici un exemple d'archivage de la base DBToFile (voir la Démonstration) :

```
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
```

Requête d'archivage de DBToFile.mdb :

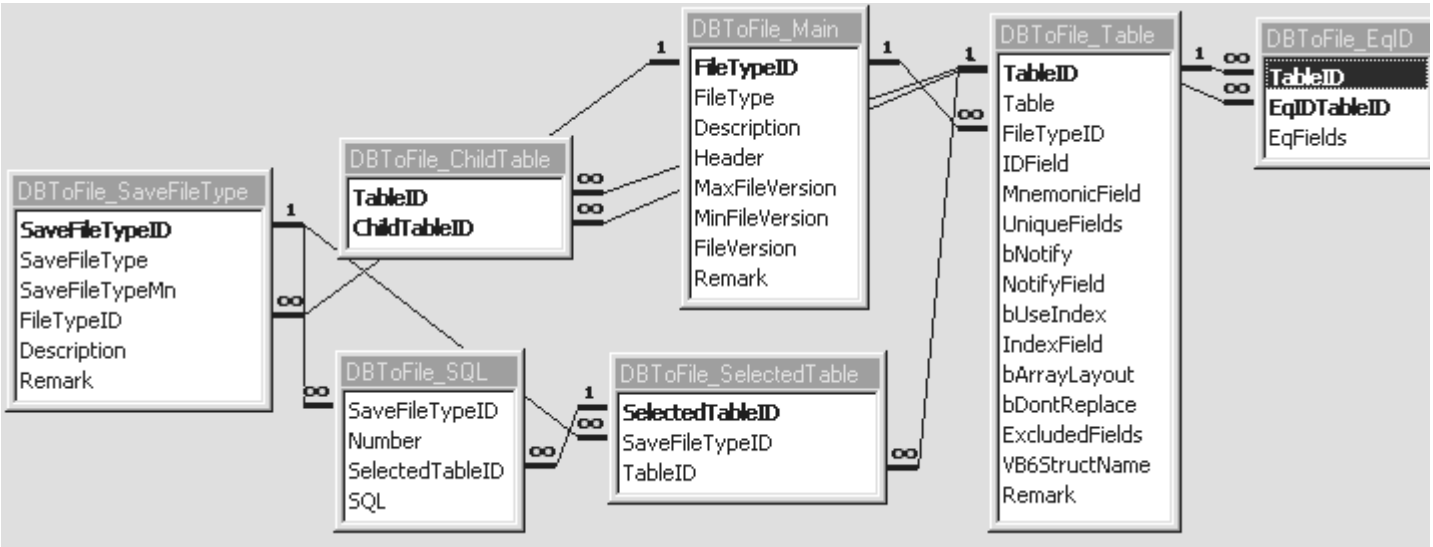
```
SQL=SELECT * FROM DBToFile_Main WHERE FileTypeID=[FileTypeID];
```

[FileTypeId] sera remplacé par la valeur trouvée dans la propriété strSaveFileArg du contrôle ActiveX

avant d'exécuter la requête.

Relations de DBToFile.mdb

(pour information)



Rapport de la structure de DBToFile.mdb

Pour information, voici le rapport généré lors de l'étape 17 de la démonstration :

```
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 -Futur 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
```

Exemples complémentaires

Exemples d'archivage contenant de nombreuses requêtes de sélection des enregistrements à archiver : http://perso.worldonline.fr/ors/dbtofile/samples.zip

Historique de l'utilitaire DBToFile

Origine

L'idée d'archiver un enchaînement de données liées d'une base de données vers un fichier en vue de les recharger ultérieurement dans une base de structure identique revient à Hervé Gaucher, responsable informatique et développeur cher LIM Géotechnologies (www.limgeo.com). Egalement développeur pendant plusieurs années dans cette entreprise fabriquant des appareils enregistreurs de paramètres de forage, j'ai eu à étendre ce principe à de nouveaux types de données ajoutés à la structure d'une base de données géotechniques, d'où l'idée de généraliser le principe de l'archivage pour une structure quelconque de base de données. Pour s'affranchir de la structure de la base, une syntaxe élémentaire décrite dans des chaînes de caractères a permis d'isoler le mécanisme d'archivage de la description de la liste des tables, en particulier des champs uniques ainsi que des champs déclenchants des archivages en cascades.

Evolution

Cela a permis de réaliser le système d'archivage de deux versions distinctes de la base géotechniques avec le même moteur d'archivage. Par ailleurs, l'analyse des fichiers d'archives a permis leur conversion d'une

version à l'autre à l'aide d'un algorithme usuellement qualifié de "moulinette", ce qui signifie que beaucoup de code est nécessaire pour cette opération dont la complexité est analogue à une véritable traduction d'une langue en autre une.

Généralisation

Par la suite, dans l'objectif de généraliser le système d'archivage d'une base quelconque, j'ai développé à titre personnel un système dans lequel la description des tables à archiver et des requêtes de sélection des données à archiver sont placées dans des tables Access afin de pouvoir être complétées sans modification du code source. Par ailleurs, cela a permis que les différentes possibilités d'archivage puissent elles même être archivées parce que ce sont également des données d'une base MS Access (voir les étapes de 14 à 17 de la démonstration sur le méta-archivage de DBToFile).

Toutes les recherches sur des chaînes de caractères ont été remplacées par la création de collections VB indexées afin d'optimiser la vitesse de la sauvegarde et du chargement des fichiers.

L'ensemble du système d'archivage à été intégré dans un contrôle ActiveX, qui peut être utilisé avec ou sans interface visible, grâce à ces méthodes et propriétés, et cela dans un grand nombre d'environnement de développement supportant les contrôles ActiveX. Voir aussi les Développements futurs.

Développements futurs

- Assistant installateur qui analyse la structure de la base de données à archiver et crée les définitions des tables à archiver ; interface pour la vérification et la modification des tables de DBToFile.mdb ;

- Version multilingue grâce à une Dll avec le code source pour ajouter une langue ;

- Optimisation avec les types prédéfinis en Visual Basic 6, voir Tableau prédéfini ;

- Gestion des index multiples dans une table (voir Champ index) ;

- Possibilité de ne pas remplacer les enregistrements d'une table (champ bDontReplace de DBToFile_Table) ;

- Généralisation à d'autre base de données qu'Access grâce à ADO (Active Data Object) ;

- Requêtes d'archivage stockées (donc compilées) dans la base de données et référencées dans le champ QueryName de la table DBToFile_SQL ;

Contact

patrice.dargenton@worldonline.fr

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