

Guide d'optimisation de vos applications Microsoft ACCESS

par Fabrice CONSTANS ([autres articles](#))

Date de publication : 06/10/2007

Dernière mise à jour : 10/11/2007

Que vous utilisiez ACCESS pour créer des applications mono-utilisateur et locales ou multiutilisateur en réseau cette suite de recommandations vous est destinée.


Il s'agit de la version 2 contenant 1 ajout supplémentaire.


- I - Introduction
 - I-A - Avertissement
- II - Règles générales
 - II-A - Maintenance des postes
 - II-B - Compactage
 - II-B-1 - Compacter avec le Plannificateur de Tâches
 - II-C - Analyseur de performance
 - II-D - Compression et cryptage
- III - Liste des conseils
 - III-A - Architecture
 - III-B - Pendant le développement
 - III-C - Tables
 - III-D - Index
 - III-E - Requêtes
 - III-D-1 - L'optimisation Rushmore en détail
 - III-D-1-a - Observer l'optimisation Rushmore avec JetShowplan
 - III-D-1-b - Les inconvénients de JetShowplan
 - III-F - I.H.M.
 - III-G - Nommage
 - III-H - VBA
 - III-H-1 - ISAMstats une autre option non documentée
 - III-H-1-a - ISAMStats et SetOption
 - III-I - ODBC
 - III-J - Divers
- IV - Postes clients, serveurs et réseau
- V - Quelques solutions techniques
 - V-A - Convention Noms Longs / Noms Courts
 - V-A-1 - Transformer les noms longs en noms courts des tables liées
 - V-A-2 - Mettre en cache les noms longs (Postes clients uniquement)
 - V-A-3 - Désactiver la génération automatique de noms courts (Postes Serveur)
 - V-B - Délai de notification de partage (serveur uniquement)
 - V-C - Problème d'accès au fichier ldb (Moteur JET uniquement).
- VI - Conclusion
- VI - Liens importants
- VIII - Remerciements

I - Introduction

Vaste sujet que l'optimisation des applications ! Suivant l'architecture, la charge, l'infrastructure réseau, la version d'ACCESS ou encore la manière de programmer, une application ACCESS peut se comporter correctement ou faire preuve d'une inertie rédhibitoire. Par expérience, une approche multi-utilisateur nécessite TOUJOURS une optimisation à tous les niveaux. Cette suite de conseils non exhaustifs est une compilation de l'expérience de spécialistes Microsoft ACCESS, de la documentation Microsoft (disponible sur le MSDN), des conseils délivrés sur le forum DEVELOPPEZ.COM ainsi que de ma propre expérience.

I-A - Avertissement

 *L'utilisation de la touche F1 est vivement conseillée à tous les stades de l'utilisation d'ACCESS. L'amélioration constante de l'aide en fait un partenaire de choix dans l'apprentissage permanent d'ACCESS. Personnellement, je ne peux m'en passer, ne serait-ce que pour mémoire.*

 *Bien qu'ACCESS soit un produit puissant, les performances de son moteur SGBDR ne sont pas comparables avec SQL server, Oracle et autres. Si vous voulez gérer des giga-octets de données avec une ou plusieurs centaines d'utilisateurs simultanés ne choisissez pas le moteur natif d'ACCESS : JET.*

Rien ne vous empêche d'utiliser la partie conception d'IHM avec l'un des moteurs de bases de données citées précédemment.


II - Règles générales

II-A - Maintenance des postes

Vos postes clients et serveurs doivent toujours être à jour. Les Service packs et Service Releases corrigent des problèmes et leur déploiement n'est pas facultatif.

II-B - Compactage

N'hésitez pas à compacter souvent l'application pendant le développement ainsi lorsqu'elle est en production.

 **97 et antérieure** Menu Outils/Utilitaires de bases de données/Compacter une base de données.

Avec ces versions, la réparation est dissociée du compactage.

2000 à 2003 Menu Outils/Utilitaires de base de données/Compacter la base de données

2007 Menu principal / Gérer / Compacter une base de données

Lorsque vous créez ou modifiez une application, la taille de l'application a tendance à augmenter de 2 voir 3 fois la taille normale. Avec les tables, la taille augmente avec l'ajout de données, ce qui est logique, mais également lors de modifications ou de suppressions.

Pour récupérer la place, il est obligatoire de compacter régulièrement vos fichiers. En principe, un fichier applicatif n'a pas besoin d'un compactage aussi fréquent sauf dans le cas où il y a des modifications importantes du volume de données.

Ormis le gain de place, il y a celui de la performance. En effet, lors du compactage les requêtes sont recompilées et les statistiques **Rushmore** sont recalculées. Ce qui impacte directement la rapidité des requêtes.

Les données sont regroupées en fonction des indexes et classées dans les pages de données contiguës. Ceci dans le but d'avoir toujours le plus petit nombre de pages à consulter.

Le compactage peut être fait à la fermeture de l'application ou à l'aide d'un raccourci lancé dans le **Planificateur de tâches**.

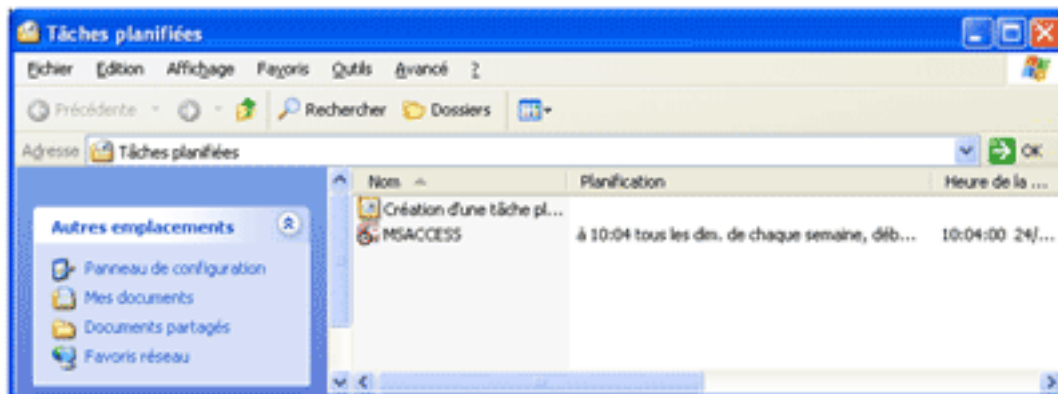
II-B-1 - Compactage avec le Planificateur de Tâches

Le compactage sur de gros fichiers peut durer plusieurs minutes ; c'est pour cela qu'il vaut mieux lancer les compactages lorsque le poste n'est pas sollicité.

Le **Planificateur de tâches** présent sur tous les systèmes Windows est idéal pour lancer cette commande.

Commencez par créer un raccourci de compactage. Voir le tutoriel  **Raccourci de compactage**

Le plus dur est fait ! Ouvrez le **Plannificateur de tâches** accessible par le menu **Démarrer/Programmes/Accessoires/Outils systèmes Tâches planifiées**




Fenêtre du plannificateur sous Windows XP


Créez une nouvelle tâche en cliquant sur la ligne **Création d'une tâche planifiée**. Un assistant vous guide tout au long de la création.


Celle-ci se fait en plusieurs phases simples :


- Choix du programme, c'est là que vous choisissez le raccourci.
- Choix de la fréquence, évitez **Une seule fois, Au démarrage...** et **A l'ouverture...**
- Suivant la fréquence choisie (mensuel, hebdomadaire ou quotidien) vous devrez sélectionner une heure, un jour, un mois.

 *Si la période est à votre discrétion, l'heure doit être méticuleusement choisie lorsque vous compactez une base située sur un serveur. En effet, les serveurs sont sollicités par les utilisateurs, par la maintenance et surtout par les sauvegardes. Choisissez une heure qui ne gênera pas les tâches existantes.*


- Entrez le nom du compte Windows et le mot de passe.

 *Ce compte devra avoir les autorisations nécessaires pour exécuter cette commande. En principe, si vous arrivez à lancer manuellement le compactage avec le raccourci, le **Plannificateur de tâches** n'en rencontrera pas de problème.*

 *Vous pouvez planifier une tâche à partir d'un poste client. Cependant, lancer la commande sur le poste contenant le fichier donne toujours de meilleurs résultats.*

 *Vous avez accès à d'autres paramètres importants en cliquant sur la case à cocher **Afficher les paramètres avancés**.*

Pour choisir la bonne fréquence de compactage, observez l'évolution de la taille de votre base de données sur plusieurs jours.

 *Veillez à ce que toutes les applications soient bien fermées sinon le compactage échouera. Il existe une méthode à mettre en place sur votre application pour qu'elle se ferme en cas d'inactivité.*

FAQ Méthode de déconnexion temporisée

II-C - Analyseur de performance

Avec les nouvelles versions d'ACCESS, il existe un **analyseur de performance** et un **analyseur de tables**. N'hésitez pas à les utiliser. Leurs recommandations sont souvent pertinentes.

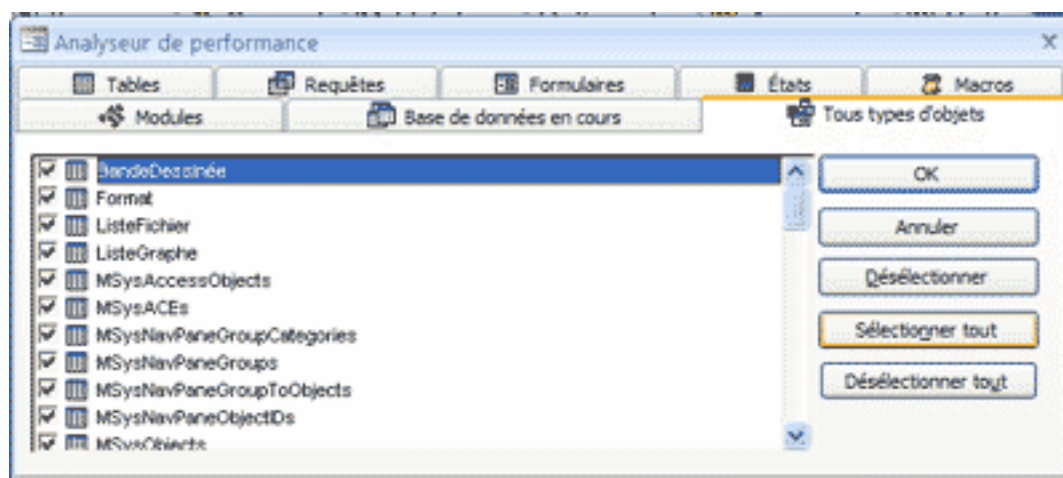
Suivez toujours les recommandations sur les index, celles-ci sont données en fonction des besoins de **Rushmore**.

 **2000 à 2003** Menu Outils/Analyse/Performances

2007 Ruban Outils de base de données / Analyser / Analyse des performances et Analyse table.

Ce tutoriel n'a pas pour objet d'apprendre la conception des modèles de données. Nous ferons donc l'impasse sur l'analyse des tables.

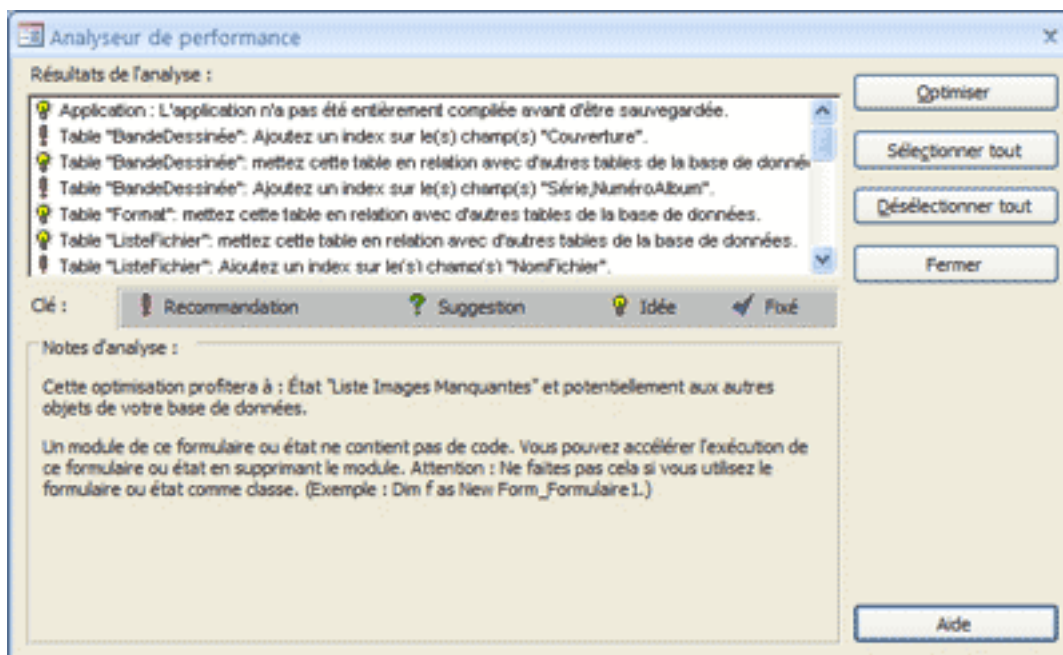
Dans l'**analyseur de performances**, cliquez sur l'onglet **Tous types d'objets**, sur le bouton **Sélectionner tout** puis **OK**.



L'analyseur de performance.

Vous obtenez un rapport clair sur les propositions d'optimisation déclinées en 3 groupes. **Recommandation, Suggestion, Idée**.

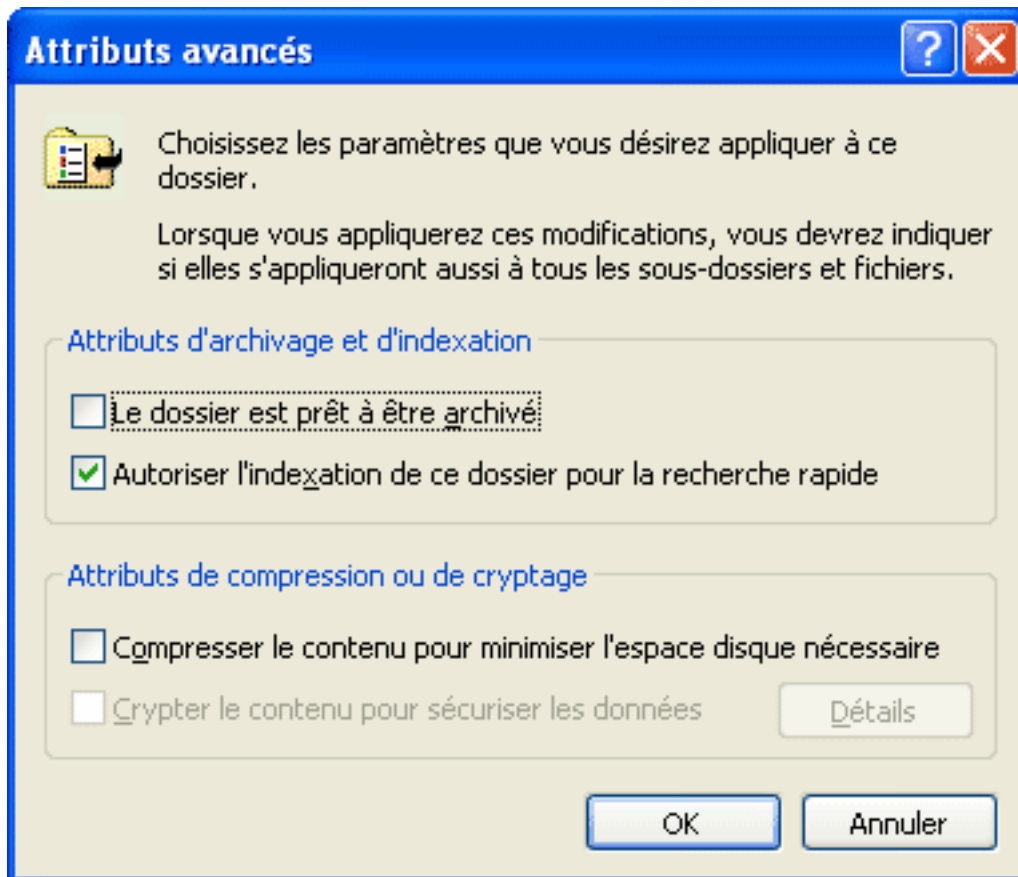
Lorsque vous cliquez sur **Optimiser**, les propositions sont effectuées et mise en statut **Fixé**.



L'analyse est faite.

II-D - Compression et cryptage

Windows permet la compression et le cryptage de vos dossiers. Chacune de ces opérations demande du temps à la machine, aussi bien dans un sens que dans l'autre. Il vaut mieux s'abstenir d'une telle pratique.



Compactage et cryptage système

Notez qu'ACCESS permet également le cryptage des données, ce qui ralentit également l'application. Suivant les versions d'ACCESS cela s'appelle le cryptage ou le codage. Il s'agit de la même chose.

i **ACCESS 97 et 2003** Menu Outils/Sécurité/Coder-Décoder une base de données

ACCESS 2002 Menu Outils/Sécurité/Crypter-Décrypter une base de données

ACCESS 2007 Il faut ajouter la commande dans le ruban.

III - Liste des conseils

Après avoir vu quelques conseils simples, nous allons lister les conseils plus techniques touchant ACCESS, l'application ou l'environnement.

III-A - Architecture

- Mettez les données sur un serveur et l'interface sur chaque poste client lorsque vous êtes dans un environnement multiutilisateur.
- Compilez MDE si vous le pouvez. La compilation MDE accélère l'exécution du code VBA, tout en protégeant les sources de l'application. Elle n'a aucun effet sur les données.
- Evitez de mettre l'application et les données dans des sous-sous-sous répertoires et évitez l'utilisation des noms longs pour les répertoires et les fichiers.
- Si vous ne partagez pas vos applications, laissez-les en local et ouvrez-les en **Mode Exclusif**.
- Gardez à l'esprit que lorsque vous utilisez le moteur JET ; celui-ci est de type fichier/serveur, c'est donc le poste client qui réalise tous les travaux. Le serveur ne fait que mettre un fichier à disposition. Ces transferts de données peuvent surcharger la bande passante du réseau.

 *Le fichier contenant l'application est appelé **Frontal**.*

*Celui contenant les données qui est placé sur le serveur est nommé **Dorsal**.*

III-B - Pendant le développement

- Développez toujours vos applications en local.
- Faites souvent des sauvegardes.

Heureusement très rare avec les versions actuelles, des problèmes peuvent survenir pendant le développement d'une application rendant le fichier applicatif instable ou carrément illisible, seule la sauvegarde vous protégera efficacement.

- N'hésitez pas à compacter pendant le développement, de préférence avant la sauvegarde.
- Préférez l'utilisation de VBA aux macros, le gain de rapidité n'est plus à prouver.
- N'utilisez pas la réplication pour mettre à jour une IHM en exploitation ou comme méthode de sauvegarde.

III-C - Tables

- Evitez les champs OLE si vous pouvez vous en passer.
- Dimensionnez et typez correctement vos champs.
- Pour vos données statiques ou à très faible mise à jour, préférez des tables locales.
- Evitez de stocker des images ou fichiers dans les tables (champs OLE).
- Evitez d'activer les sous-feuilles de données dans les tables.
- Organisez correctement vos tables. Il vaut mieux une multitude de tables de faibles volumes qu'une seule de gros volume.
- Evitez la redondance inutile d'information. De rares cas rendent obligatoires cette pratique.

En général, toutes informations susceptibles de changer dans le temps mais dont la redondance permet de figer sa valeur à l'instant de son utilisation.

Les cas classiques sont :

Les adresses clients, les taux de TVA et les prix d'articles utilisés dans les factures et les commandes.

III-D - Index

- Pensez à indexer, mais uniquement les champs nécessaires. Trop d'index tuent les performances et nuisent aux accès concurrents.
- Dans le cas de tables en relation, pensez à indexer les 2 cotés de la relation.
- Si les données d'un champ doivent être fréquemment affichées triées, pensez à l'indexer.
- Dans le cas d'une base de données ayant peu ou pas de mise à jour, indexez tous les champs servant dans les relations et les restrictions (utiliser dans des conditions Where des requêtes).

Ceci améliorera notablement les statistiques **Rushmore**.

- La permission des valeurs Null dans un champ peut empêcher dans certain cas l'optimisation **Rushmore**.


III-E - Requêtes

- Le générateur de requête (QBE) optimise et corrige les requêtes pour les bases ACCESS, n'hésitez pas à recourir à ses services même si vous vous sentez un pro de SQL. Chaque langage SQL a ses particularités, celui de Microsoft ACCESS n'échappe pas à la règle.
- Pensez à l'imbrication des SELECT. Une solution tout SQL est souvent préférable à une solution VBA.
- Evitez de sélectionner des champs inutiles, ne remontez que les colonnes et les enregistrements dont vous avez besoin.
- Lors de création de requêtes paramétrées avec jointure, testez si l'exécution est plus rapide lorsque vous placez les critères du coté un ou du coté plusieurs de la relation.
- Evitez les formules dans les sous-requêtes.
- Evitez de faire des recherches sur des champs Mémo.
- Evitez de mettre des liens vers des objets de l'IHM dans vos requêtes paramétrées. Préférez une solution VBA pour intégrer le paramètre.
- Evitez l'imbrication de requête avec NOT IN() ; préférez une requête du type non correspondance.
- Evitez les critères sur des colonnes calculées.
- Pour compter le nombre d'enregistrements, utilisez COUNT(*) au lieu de COUNT([champ]).
- Evitez les fonctions de regroupement de domaine (Dlookup, Dcount, Dsum...) ; utilisez plutôt une sous-requête.
- La fonction Vraifaux() -Iif()- n'est pas optimisable, utilisez si possible cette fonction dans un formulaire ou un état.
- Evitez le Group By sur trop grand nombre de colonnes.
- Evitez le Group By après une jointure.
- Evitez d'insérer plusieurs fois la même table dans le QBE.

III-D-1 - L'optimisation Rushmore en détail

Rushmore fait appel aux index et à leur fusion pour optimiser les objets requêtes d'ACCESS. Cette technologie existe depuis la version 2.0 du moteur JET. Elle est issue de Foxpro.

L'optimisation **Rushmore** est employée sur des requêtes impliquant des restrictions (AND, OR, BETWEEN, =, LIKE, IN) multi-colonnes indexées.

 *LIKE est un opérateur un peu à part puisque qu'il bénéficie de l'optimisation uniquement dans les cas suivants :*


Optimisé

```
LIKE "mavaleur"  
LIKE "mavaleur*"
```

 *Si **IN** est optimisé, **Not IN** ne l'est absolument pas !*

Rushmore utilise un système de statistique pour déterminer la méthode la plus rapide pour retrouver un volume de données.


Les requêtes utilisant la fonction Count(*) et les index multiples sont les exemples les plus courants des capacités d'optimisation de **Rushmore**, mais pas seulement.

 ***Rushmore** ayant été porté de Microsoft FOXPRO, il fonctionne également avec les tables Dbase et Foxpro indexées liées mais pas au travers d'ODBC.*

Les listes suivantes indiquent les différents paramètres qui sont évalués par **Rushmore** pour choisir la stratégie la plus performante.

Première analyse

- Le nombre d'enregistrements dans la table,
- Le nombre de pages de données dans la table (plus il y a de pages plus le cout est élevé),
- L'endroit de stockage de la table (locale, liée, Jet, Foxpro, ODBC, ISAM...),
- Le jeu d'enregistrement est modifiable ou non.
- Les index de la table (tableau suivant).

 *Dans le cas d'un jeu d'enregistrements modifiables (Dynaset), **Rushmore** choisit la stratégie qui retourne le plus rapidement la 1ère page de données. Cela peut se faire au détriment du temps total pour remonter la totalité des enregistrements.*

Analyse des index des tables

- Index unique ou non.
- Nombre de pages d'index : Plus il y a de pages, plus le coût en temps est élevé.
- La valeur Null est permise : Cela peut annuler la fusion d'index.
- Analyse de jointure des tables

Une fois les paramètres évalués, **Rushmore** compare plusieurs techniques d'optimisations de recherches. La plus performante en temps est choisie.

Type de stratégie Rushmore

- **Itération imbriquée :**

Cette technique est utilisée lorsqu'il y a de gros volumes de données dans les tables.

- **Indexation** : Parcours les lignes de la 1ère table et recherche les enregistrements connexes dans la 2ème table.

Utilisée avec peu de données dans la 2ème table ou qu'aucune données de la 2ème table n'est à afficher par la requête.

- **Recherche** : Idem à la précédente sauf que le tri et la restriction se font avant la jointure.

Utilisée avec un faible volume de données dans la seconde table et jointure non indexée.

- **Fusion de jointure** : Tri les 2 tables par la jointure et combine les 2 tables.

Utilisée avec un gros volume de données triées sur le champ servant de jointure.


- **Fusion d'index** : Idem à la précédente sauf que les index servent au classement des 2 tables.

Utilisée lorsque les colonnes de la jointure sont indexées, sans valeur Null autorisée dans l'une des colonnes, ou si il y a plus d'une colonne servant de jointure.

III-D-1-a - Observer l'optimisation Rushmore avec JetShowplan

Une option non-documentée permet d'observer le travail de l'optimisateur **Rushmore**. Cette option permet de générer un fichier nommé *showplan.out* contenant l'ensemble des analyses **Rushmore** de chaque requête du fichier.

Pour qu'ACCESS génère ce fichier, il faut procéder à une modification de la base de registre.

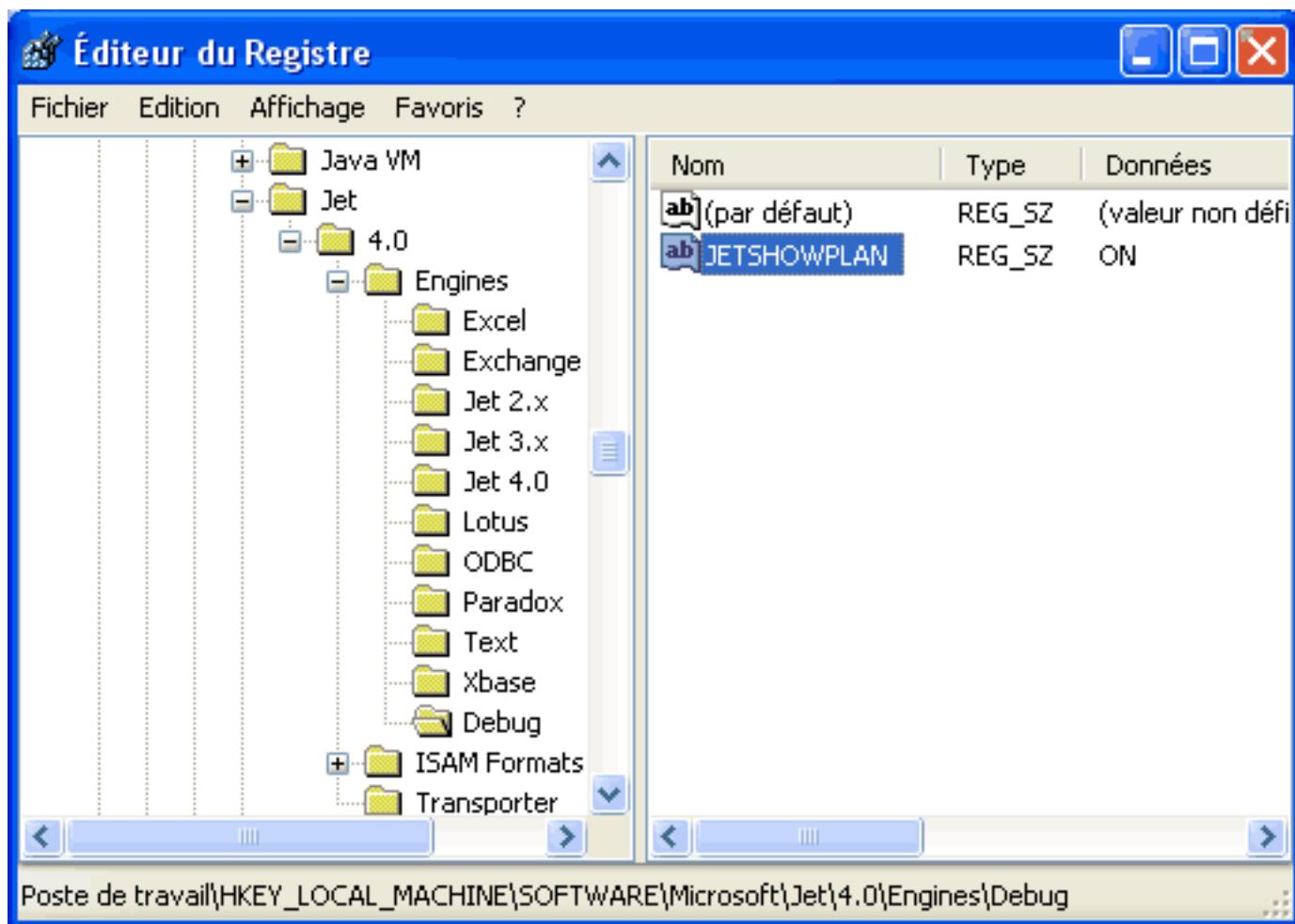
 *La modification de la base de registre comporte toujours des risques. Procédez avec précaution !*

Pour ACCESS 95 à 2003

- 1 Ouvrez la base de registre avec Regedit.
- 2 Allez dans le registre `\\HKEY_LOCAL_MACHINE\\SOFTWARE\\MICROSOFT\\Jet\\x.x\\Engines\\Debug` ou `x.x` représente la version du moteur Jet que vous utilisez.

Actuellement la dernière version de JET est la 4.0

- 3 En principe **Debug** n'est pas créé. Créez cette clef.
- 4 A l'intérieur créez la valeur **JETSHOWPLAN** de type **Chaîne**
- 5 Paramétrez la valeur **ON**




Exemple pour une version 2002-XP

Pour ACCESS 2007

- 1 Ouvrez la base de registre avec Regedit.
- 2 Allez dans le registre *HKLM\SOFTWARE\Microsoft\Office\12.0\Access Connectivity Engine\Engines\Debug*
- 3 En principe, **Debug** n'est pas créé. Créez cette clef.
- 4 A l'intérieur créez la clef **JETSHOWPLAN** de type **Chaîne / REG_SZ**
- 5 Paramétrez la valeur **ON**

 **Pour annuler la génération, vous devez mettre la valeur JETSHOWPLAN à OFF.**

 **Debug** doit obligatoirement respecter la casse. La **première lettre est en majuscule** ; les suivantes sont en minuscules.

JETSHOWPLAN, ON et OFF doivent obligatoirement être en **majuscules**.


 **Notez qu'une fois la clef JETSHOWPLAN est en place, la position ON/OFF est prise en compte sans redémarrage du poste.**

Une fois cette modification faite, redémarrez le poste de travail pour prendre en compte la modification de la Base de registre.

Le fichier *showplan.out* se génère sous deux conditions :


L'ouverture, la modification ou la création d'un objet mettant en oeuvre un accès à une table ; que ce soit par un accès direct, par une requête ou encore par les fonctions de domaines. Le fichier se génère également lors du compactage de la base.


 *Pour les autres cas de génération d'un fichier *showplan.out*, voir le chapitre suivant.*


 *Pour forcer l'analyse d'une requête, il faut la modifier puis la sauvegarder. L'analyse apparaîtra dans le fichier *showplan.out*.*

Lors de la génération du fichier, l'analyse des tables systèmes, locales et attachées est systématique. On y trouve donc des données intéressantes comme le chemin d'attache, le nombre d'enregistrements, les nombres de pages de données et tous les renseignements concernant les index.

Les erreurs éventuelle sont inscrites en entête du fichier. Chaque nouvelle action du moteur Jet est écrite à la suite du fichier

 Avec **Windows XP**, le fichier *showplan.out* est écrit dans le répertoire **MES DOCUMENTS**.

 Avec les versions antérieures du système, le fichier *showplan.out* est écrit dans le répertoire courant.

 *Pour connaître le répertoire courant :*

- Démarrez l'invite de commande, le prompt (c:\xxx\>) indique le répertoire courant.

- Sous ACCESS, lisez le résultat de la fonction VBA Curdir().

Ouvrez le fichier *showplan.out*.

Exemple de l'analyse d'une requête

```
--- Requête1 ---

- Inputs to Query -
Table 'Payeur'
  Using index 'PrimaryKey'
  Having index:
  PrimaryKey 122 entries, 1 page, 122 values
    which has 1 column, fixed, unique, primary-key, no-nulls
  NUMERO 122 entries, 1 page, 122 values
    which has 1 column, fixed, unique
  NOM 122 entries, 1 page, 118 values
    which has 1 column, fixed
Table 'CLIENT'
Table 'CONTRAT_APPAREIL'
Table 'CONTRAT'
  Using index 'CLIENTCONTRAT'
  Having index:
  CLIENTCONTRAT 336 entries, 1 page, 281 values
    which has 1 column, fixed
Table 'Appareil'
  Using index 'PrimaryKey'
```

Exemple de l'analyse d'une requête

```

Having index:
PrimaryKey 412 entries, 3 pages, 412 values
  which has 1 column, fixed, unique, primary-key, no-nulls
N_APPAREIL 412 entries, 3 pages, 412 values
  which has 1 column, fixed
Table 'CONTRAT_APPAREIL'
Using index 'N_CONTRAT'
Having index:
N_CONTRAT 478 entries, 3 pages, 301 values
  which has 1 column, fixed
N_APPAREIL 478 entries, 3 pages, 404 values
  which has 1 column, fixed
CONTRATCONTRAT_APPAREIL 478 entries, 3 pages, 301 values
  which has 1 column, fixed
APPAREILCONTRAT_APPAREIL 478 entries, 3 pages, 404 values
  which has 1 column, fixed
Table 'CONTRAT'
- End inputs to Query -

01) Restrict rows of table CONTRAT
    using rushmore
    for expression "CONTRAT.N_CLIENT Like "411*"
02) Outer Join table 'CLIENT' to table 'CONTRAT'
    using index 'CONTRAT!CLIENTCONTRAT'
    join expression "CLIENT.N_CLIENT=CONTRAT.N_CLIENT"
03) Sort table 'CONTRAT_APPAREIL'
04) Outer Join result of '02)' to result of '03)'
    using temporary index
    join expression "CONTRAT.N_CONTRAT=CONTRAT_APPAREIL.N_CONTRAT"
05) Outer Join result of '04)' to table 'Payeur'
    using index 'Payeur!PrimaryKey'
    join expression "CLIENT.PAYEUR=PAYEUR.N_PAYEUR"
06) Outer Join result of '05)' to table 'Appareil'
    using index 'Appareil!PrimaryKey'
    join expression "CONTRAT_APPAREIL.N_APPAREIL=APPAREIL.N_APPAREIL"
07) Sort result of '06)'
08) Outer Join result of '01)' to result of '07)'
    using temporary index
    join expression "CONTRAT.N_CLIENT=[Vérification Payeur-Client-Contrat-Appareil].Cleint"
09) Inner Join result of '08)' to table 'CONTRAT_APPAREIL'
    using index 'CONTRAT_APPAREIL!N_CONTRAT'
    join expression "CONTRAT.N_CONTRAT=CONTRAT_APPAREIL.N_CONTRAT"

```

La requête correspondante : Requête1


```

SELECT CONTRAT_APPAREIL.N_CONTRAT, CONTRAT.N_CLIENT,
[Vérification Payeur-Client-Contrat-Appareil].T_SERVICE,
CONTRAT_APPAREIL.MODEENTR, CONTRAT.D_ECHEANCE
FROM [Vérification Payeur-Client-Contrat-Appareil] RIGHT JOIN (CONTRAT INNER JOIN
CONTRAT_APPAREIL ON CONTRAT.N_CONTRAT = CONTRAT_APPAREIL.N_CONTRAT) ON
[Vérification Payeur-Client-Contrat-Appareil].Cleint = CONTRAT.N_CLIENT
WHERE (((CONTRAT.N_CLIENT) Like "411*"));

```

Il s'agit d'une partie du fichier *showplan.out* comprenant l'analyse de la requête nommée **Requête1**. Ce résultat d'analyse se découpe en 2 segments :

- l'analyse des tables utilisées dans la requête.
- le détail du processus **Rushmore**.

 On peut remarquer qu'en 01 la restriction est prise en charge par **Rushmore**. En 04 et 08, **Rushmore** crée un index temporaire pour les jointures.

L'analyse et la compréhension de ce fichier peut apporter beaucoup dans l'optimisation des requêtes. Notez qu'il ne s'agit que d'un résultat et non du processus complet d'analyse.

III-D-1-b - Les inconvénients de JetShowplan

Jetshowplan a un gros inconvénient : il ne se déclenche pas uniquement lors de la modification des requêtes, mais lors de toutes manipulation d'objets. De plus, il reçoit des informations en permanence. La solution est de le renommer entre 2 écritures ou encore de le détruire sinon il prend de l'embonpoint. Il peut passer de quelques Kilo-octets à plusieurs Méga-octets, ralentissant d'autant l'exécution de l'application.

Opérations du fichier showplan

Opérations du fichier showplan	Commentaires
Création d'un Ko formulaire	60 Analyse de la requête source à plusieurs reprises.
Ouverture d'un Mo état en mode consultation	4027 Analyse de la requête source et chaque fonction de domaine autant de fois qu'elles sont sollicitées. Dans le cas de cet état élaboré, 10 appels de fonctions de domaines par page, 4040 enregistrements.
Ouverture d'un Ko état en mode création	4 Analyse de la requête source
Création d'un Ko table	2 3 champs 2 index, le nom de la nouvelle table apparait avec l'analyse de chaque index.
3 Ko	3 Analyse de la table,

Ouverture d'une table	nombre d'enregistrements, de pages de données, d'index et d'entrées dans les index.
Exécution d'une requête Select sous VBA	Un mythe tombe ! La requête est bien prise en charge par Rushmore .

Code VBA / Requête

```

Function testjet()
Dim rst As DAO.Recordset
Dim sql As String

sql = "SELECT FACTURE.N_FACTURE, FACTURE.MONTANTTTC, FACTURE.MTVA55, FACTURE.MTVA196,
sql = sql + " FACT_DETAILS.MONTANT, FACT_DETAILS.MONTANTTTC, FACT_DETAILS.TVA"
sql = sql + " FROM FACTURE LEFT JOIN FACT_DETAILS ON (FACTURE.N_FACTURE = FACT_DETAILS.N_FACTURE)"
sql = sql + " AND (FACTURE.N_FACTURE = FACT_DETAILS.N_FACTURE)"
sql = sql + " WHERE (((FACTURE.N_FACTURE) > 204050279))"
sql = sql + " ORDER BY FACTURE.N_FACTURE DESC;"

Set rst = CurrentDb.OpenRecordset(sql)

Debug.Print rst.Fields(0).Value

rst.Close

End Function
    
```

Le fichier résultat ci-dessous et plus particulièrement le processus 01 indique que **Rushmore** est actif.

```

- Inputs to Query -
Table 'FACTURE'
  Database 'D:\ACCESS\wga\WGA_fact.mdb'
Table 'FACT_DETAILS'
  Database 'D:\ACCESS\wga\WGA_fact.mdb'
- End inputs to Query -

01) Restrict rows of table FACTURE
   using rushmore
   for expression "FACTURE.N_FACTURE>204050279"
02) Sort result of '01)'
03) Sort table 'FACT_DETAILS'
04) Outer Join result of '02)' to result of '03)'
   using Merge Join
   join expression "FACTURE.N_FACTURE=FACT_DETAILS.N_FACTURE And
FACTURE.N_FACTURE=FACT_DETAILS.N_FACTURE"
05) Sort result of '04)'
    
```

Notez que tous les traitements du moteur Jet sont inscrits comme le démontre la présence des requêtes temporaires (~tmp..., ~sql...)

La conclusion sur Jetshowplan est que nous sommes en lien direct avec le moteur Jet. Cette fonction ne doit jamais être mise en place sur une application en exploitation et encore moins en cours de développement, mais uniquement dans le cas d'un besoin ponctuel d'analyse des requêtes.

III-F - I.H.M.

- Evitez un trop grand nombre d'objets, préférez des solutions dynamiques.
 - ⚠ *Les solutions dynamiques basées sur l'ouverture des objets en mode création sont incompatibles avec la compilation MDE.*
- Restreignez l'utilisation des zones de listes et modifiables dans un environnement réseau. Celles-ci demandent beaucoup de transfert de données.
- Evitez les colonnes inutiles dans les zones listes.
- Lors de l'utilisation d'onglets contenant beaucoup de données (sous-formulaire, zone de liste...) chargez les données au moment de la consultation.
- Evitez les accès directs aux tables dans la propriété Source ; utilisez plutôt des requêtes pour réduire au maximum le volume de données affichées.
- Prévoyez un bouton de saisie pour l'ouverture de chacun de vos formulaires. Cela évitera de récupérer des enregistrements inutiles à cette action.
- Dans les formulaires, utilisez la lecture seule si vous ne faites que de la consultation.
- Choisissez les méthodes de verrouillage adaptées aussi bien générale qu'au niveau des formulaires.
- Evitez les conflits de verrouillage en réglant les paramètres de temporisation.
- Convertissez vos applications et fichiers de groupe de travail dans la version utilisée.
- Même si la cohabitation de diverses versions d'ACCESS sur une même application est possible, mettez vos versions d'ACCESS à niveau.
- Dans le cas de formules ou de calculs, placez-les dans les formulaires ou états plutôt que dans la requête source.
- Préférez le nom de la requête dans la source des formulaires/états plutôt que la chaîne SQL.
- Evitez l'affichage systématique d'un champ Mémo, préférez un bouton pour le charger.
- Limitez le nombre de polices de caractères dans vos applications, préférez toujours les polices standard.
- N'ouvrez jamais directement une table (liée ou non) avec une application en exploitation.
- Pour les liens formulaires/sous-formulaires composés de plusieurs champs, créez l'expression de liaison dans les requêtes sources.
- Si vous utilisez toujours les mêmes images dans un état ou un formulaire, utilisez plutôt un sous-état (ou sous-formulaire) vous n'aurez ainsi qu'un objet image au lieu d'une multitude stockée dans votre application.

III-G - Nommage

- Evitez les noms longs pour vos objets, tables, champs...
- Evitez les espaces et caractères cabalistiques nécessitant l'ajout de crochet [] dans les noms de tables, de champs, de contrôles, de requêtes, d'états..., tous les objets que vous devez ou pouvez nommer. L'underscore ou trait de soulignement _ permet de remplacer avantageusement l'espace.

III-H - VBA

- Si vous utilisez uniquement des bases JET (fichier de données MDB) préférez l'utilisation de DAO nettement plus rapide. Pour d'autres bases de données utilisez ADO.
- Pour des traitements EXCEL / ACCESS utilisez le passage de recordset via DAO au lieu de la modification de cellules (Range, Cell#)
- Lors des accès en automation à une feuille Excel, désactivez le recalcul automatique et l'affichage de la feuille. Les applications ouvertes en automation peuvent être cachées, n'hésitez pas à utiliser cette option.
- Evitez les recordsets modifiables si vous n'en avez pas besoin.
- Préférez toujours du code VBA à des macros.
- Lorsque vous ouvrez des objets recordset, automation et autres, pensez à les vider et les refermer.
- Evitez d'utiliser les anciennes méthodes disponibles pour compatibilité. Convertissez votre code avec les nouveaux objets et propriétés.
- Si vous faites référence plusieurs fois à un même objet dans le code, stockez-le dans une variable objet.
- Les variables **Integer** sont systématiquement converties en **Long**. Economisez cette conversion en déclarant vos variables **Integer** directement en **Long**.
- Pour savoir si une chaîne est vide, utilisez la fonction **Len()** -Longueur de la chaîne-. VBA stocke en mémoire la longueur de chaque chaîne de caractère, **Len** ne fait donc que lire cette valeur. La valeur renvoyée par **Len** est toujours de type **Long**.
- Les fonctions renvoyant une chaîne de caractère (fonctions précédées du signe dollars \$) sont plus rapides que les mêmes sans \$ renvoyant une variable Variant.
- Lorsque vous utilisez les dates, vous pouvez utiliser 2 séparateurs différents, le # ou le ". Dans le second cas VBA est obligé de faire une conversion.
- Préférez le type **Currency** à **Single** qui sollicite le FPU et donc plus long à traiter.
- Il existe deux opérateurs pour la division. \ (**anti-slash**) pour les entiers et / (**slash**) qui renvoie un **Double** et donc utilise le FPU.
- Utilisez les constantes mis à votre disposition par VBA pour remplacer certaines fonctions. **vbCrLf** remplace **chr(13)+chr(10)**.
- Evitez d'utiliser les fonctions de domaines (**Dlookup, Dsum...**) sur des tables ayant de gros volumes.
- Evitez d'utiliser le Find (ADO) ou FindFirst (DAO) sur des gros volumes de données, créez une requête ou un filtre avant la recherche.
- Si vous faites une recherche sur un champ indexé, utilisez Seek (ADO, DAO) plutôt que Find / FindFirst.
- Utilisez les tableaux de variables au lieu d'une table temporaire.

III-H-1 - ISAMstats une autre option non documentée

ISAMstat est une fonction qui fournit des statistiques de lecture/écriture/locking et mise en cache du moteur JET. Elle est donc l'indicateur des résultats des réglages effectués sur les options suivantes :

- PageTimeout
- SharedAsyncDelay
- ExclusiveAsyncDelay
- LockRetry
- UserCommitSync
- ImplicitCommitSync
- MaxBufferSize
- MaxLocksPerFile
- LockDelay
- RecycleLVs

- FlushTransactionTimeout

Ces options sont en base de registre, mais peuvent être modifiées grâce à la méthode **SetOption** de **DBEngine** avec DAO ou les propriétés d'une connexion ADO. (voir le chapitre suivant pour l'utilisation et la signification de ces paramètres.

Pour utiliser cette fonction, il suffit simplement d'y faire appel dans le code VBA.

Exemple d'un code ISAMStats fourni par le MSDN

```
Sub Main()

Dim dbs As Database, ws As Workspace
Dim strSQL As String
Dim lngDiskRead As Long, lngDiskWrite As Long, lngCacheRead As Long, lngLocksReleased As Long
Dim lngCacheReadAheadCache As Long, lngLocksPlaced As Long

' Affecte explicitement 0 aux variables
lngDiskRead = DBEngine.ISAMStats(0, True)
lngDiskWrite = DBEngine.ISAMStats(1, True)
lngCacheRead = DBEngine.ISAMStats(2, True)
lngCacheReadAheadCache = DBEngine.ISAMStats(3, True)
lngLocksPlaced = DBEngine.ISAMStats(4, True)
lngLocksReleased = DBEngine.ISAMStats(5, True)

Set dbs = OpenDatabase("chemin et nom du fichier applicatif", False, False)
Set ws = Workspaces(0)
strSQL = "Indiquez ici votre chaîne SQL action"

dbs.Execute strSQL, dbFailOnError

' La transaction nulle assure qu'il n'y pas d'activité
' après l'exécution qui pourrait fausser les statistiques
ws.BeginTrans
ws.CommitTrans

' Les appels ISAMStats retourne les statistiques
' Les valeurs ne sont pas remise à zéro.

lngDiskRead = DBEngine.ISAMStats(0)
lngDiskWrite = DBEngine.ISAMStats(1)
lngCacheRead = DBEngine.ISAMStats(2)
lngCacheReadAheadCache = DBEngine.ISAMStats(3)
lngLocksPlaced = DBEngine.ISAMStats(4)
lngLocksReleased = DBEngine.ISAMStats(5)

Debug.Print "0 - Disk reads " & lngDiskRead
Debug.Print "1 - Disk writes " & lngDiskWrite
Debug.Print "2 - Cache reads " & lngCacheRead
Debug.Print "3 - Cache reads from RA cache " & lngCacheReadAheadCache
Debug.Print "4 - Locks placed " & lngLocksPlaced
Debug.Print "5 - Locks released " & lngLocksReleased

End Sub
```

Les résultats ISAMStats sont surtout significatifs pour des applications réseaux.

Pour en savoir plus sur ISAMStats, consultez le lien suivant :  [ISAMStats](#)

III-H-1-a - ISAMStats et SetOption

L'utilisation de SetOption n'est pas compliquée ; seul le paramétrage idéal peut être difficile. Il est réservé à des développeurs chevronés.

```
' pour DAO
DBEngine.SetOption DAO.paramètre, MaValeur

' pour ADO : ADOc est un objet Connection ADO
ADOC.Properties("ADO.paramètre") = MaValeur
```

paramètre DAO.SetOption paramètre ADO	Paramètre Base de Registre Valeur d'initial	Description
dbPageTimeout Jet OLEDB:Page Timeout	PageTimeout	5000 Temps (en ms) entre le moment où les données non verrouillées en lecture sont placées dans une mémoire cache interne et le moment où elles sont invalidées.
dbSharedAsyncDelay Jet OLEDB:Shared Async Delay	SharedAsyncDelay	0 Indique le délai (en ms) pour différer un vidage asynchrone d'une base de données partagée.
dbExclusiveAsyncDelay Jet OLEDB:Exclusive Async Delay	ExclusiveAsyncDelay	2000 Indique le délai (en millisecondes) pour différer un vidage asynchrone d'une base de données en accès exclusif.
dbLockRetry Jet OLEDB:Lock Retry	LockRetry	2 Nombre de tentatives d'accès à une page verrouillée avant l'affichage d'un message d'erreur
dbUserCommitSync Jet OLEDB>User Commit Sync	UserCommitSync	Yes Indique si le système doit attendre une validation de fin de transaction. La valeur Yes demande au système d'attendre, et la valeur No lui demande d'exécuter la validation en mode asynchrone.
dbImplicitCommitSync Jet OLEDB:Implicit Commit Sync	ImplicitCommitSync	No Indique si le système doit attendre une validation de fin de transaction. La valeur No demande au système de poursuivre sans attendre la validation de fin, et la valeur Yes lui demande d'attendre la validation de fin de transaction.
dbMaxBufferSize Jet OLEDB:Max Buffer Size	MaxBufferSize	0 Taille de la mémoire cache interne du moteur de base de données, mesurée en kilo-octets (Ko). La valeur

		<p>doit être un nombre entier supérieur ou égal à 512.</p> <p>La valeur par défaut est calculée sur la base de la formule suivante :</p> $((\text{TotalRAM en Mo} - 12 \text{ Mo}) / 4) + 512 \text{ Ko}$ <p>Exemple pour 512 Mo de RAM :</p> $(512 - 12) / 4 = 125 \text{ (Mo)} + 512 \text{ (Ko)} = 125.512 \text{ Ko}$ <p>0 permet au moteur d'utiliser la mémoire dont il a besoin.</p>
dbMaxLocksPerFile Jet OLEDB:Max Locks Per File	MaxLocksPerFile	<p>9600 paramètre empêche les transactions de Microsoft Jet de dépasser la valeur spécifiée. Si le verrou d'une transaction tente de dépasser cette valeur, la transaction est divisée en deux parties ou plus et entérinée partiellement. Ce paramètre a été ajouté pour éviter que le serveur Netware 3.1 se bloque si la limite du verrou Netware spécifié est dépassée, et pour améliorer les performances avec Netware et NT.</p>
dbLockDelay Jet OLEDB:Lock Delay	LockDelay	<p>100 paramètre est associé à LockRetry car un délai d'attente de 100 millisecondes est affecté à ce dernier pour qu'une autre demande de verrouillage puisse être émise. Le paramètre LockDelay a été ajouté pour éviter les « rafales » éventuelles avec certains systèmes de gestion de réseau.</p>
dbFlushTransactionTimeout Jet OLEDB:Flush Transaction Timeout	FlushTransactionTimeout	<p>5000ms (en ms) au delà duquel les écritures asynchrones sont faites si aucune page n'a été ajoutée à la mémoire cache. Si la mémoire cache dépasse</p>

		la valeur de MaxBufferSize l'écriture est déclenchée.
--	--	--

Informations issues du MSDN

III-I - ODBC

ODBC est une couche supplémentaire permettant de mettre en relation ACCESS et n'importe quel moteur de base de données. Son utilisation implique une bonne connaissance de la base à laquelle on accède.

- Favorisez toujours un pilote Microsoft (ODBC par exemple).
- Evitez les accès directs aux tables, préférez un attachement et des requêtes.
- Mettez la propriété **FailOnError** du serveur à **Oui**.
- Assurez-vous toujours que les requêtes SQL peuvent être exécutées par le serveur.

III-J - Divers

- Avec les versions ACCESS 2000 et antérieures, désactivez la correction automatique des noms.
- N'hésitez pas à faire des essais de timing et rafraîchissement pour optimiser vos temps de réponses. Ces options se paramètrent soit pour l'application, soit pour la globalité d'ACCESS.
- Si vous avez trop d'accès concurrents (utilisateurs au même instant sur la ou les mêmes tables), choisissez un autre moteur de base de données que JET. MSDE, SQL Server par exemple.
- Gardez toujours Windows et MS Office à jour et au même niveau, évitez les bêta-versions en production.

IV - Postes clients, serveurs et réseau

- N'oubliez pas que JET n'est pas un client/serveur, c'est le poste client qui effectue toutes les opérations donc les données doivent transiter sur le réseau. Vous ne devez pas hésiter à tout mettre en oeuvre pour limiter ce trafic.
- Ne faites pas cohabiter serveur de fichiers et serveur de domaine primaire ou secondaire.
- Sur le serveur, préférez l'économiseur "Ecran noir" simple plutôt que la tuyauterie bondissante et les autres animations gadgets.
- Evitez autant que possible les fonds d'écran volumineux.
- Evitez de mettre l'application comme le fichier des données dans des sous-sous-sous répertoires et évitez les noms longs pour les répertoires et le fichier de données. Plus vous êtes proche de la racine et moins vos noms de répertoire sont courts (convention 8.3), plus les performances seront accrues.
- Certains Anti-virus trop agressifs nuisent à la bonne marche de nos machines, les applications en font les frais. N'hésitez pas exclure de la vérification les applications ACCESS. Sachez que les virus ou macros-virus attaquant ACCESS sont très peu nombreux.
- Vérifiez que le serveur dispose d'un nombre d'accès suffisant pour accueillir l'ensemble des connexions.
- ACCESS a besoin de beaucoup de bande passante sur le réseau. A moins de 100 Mb/s, ne faites pas d'applications partagées.
- Vérifiez ou faites vérifier votre réseau pour éliminer tout goulet d'étranglement, les coupures qui peuvent survenir pourraient endommager les fichiers de données.
- L'utilisation des nouveaux disques autonomes Ethernet (NAS, NDAS) ne sont pas conseillés pour héberger les données d'une application. Le SAN n'est pas concerné par cette restriction.
- Placez vos bases partagées sur des volumes NTFS plutôt que FAT ou FAT32.
- Evitez l'utilisation du chemin UNC pour l'attachement des bases.

Si votre fichier de données est hébergé sur un serveur Windows : Sauf 2003 SERVER !

- Désactivez le délai de notification de violation de partage.
- Désactivez la génération automatique de nom de fichier court.
- Activez la mise en cache des noms longs (sauf si vous utilisez des noms courts)

V - Quelques solutions techniques

Dans ce chapitre, nous allons commenter quelques techniques énoncées précédemment.

V-A - Convention Noms Longs / Noms Courts

Comme nous l'avons vu dans le chapitre concernant les tables, vous devez proscrire l'utilisation des noms longs dans vos répertoires au profit de chemin de type 8.3 (huit caractères " . " trois caractères).

Rassurez-vous ! Vous n'avez pas besoin de tout changer sur votre serveur, vous pouvez vous même traduire un nom long en nom court à l'aide des tildes ~.

Exemple d'un chemin nom long et de sa traduction en 8.3.

```
C:\Program files\  
C:\progra~1\  

```

Le code VBA suivant permet de transformer automatiquement les noms longs en noms au format 8.3.

```
Option Compare Database  
Option Explicit  
  
Declare Function GetShortPathName Lib "kernel32" _  
    Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _  
    ByVal lpszShortPath As String, ByVal cchBuffer As Long) As Long  
  
Public Function GetShortName(ByVal sLongFileName As String) As String  
    Dim lRetVal As Long, sShortPathName As String, iLen As Integer  
    sShortPathName = Space(255)  
    iLen = Len(sShortPathName)  
    lRetVal = GetShortPathName(sLongFileName, sShortPathName, iLen)  
    GetShortName = left(sShortPathName, lRetVal)  
End Function  
  
Appel de la fonction :  
  
NomCourt = GetShortName("C:\Program Files\Microsoft Office\")
```

V-A-1 - Transformer les noms longs en noms courts des tables liées

A l'aide des API et de VBA, nous pouvons transformer automatiquement les liens longs en 8.3.

source MSDN traduction Fabrice Constans

```
Declare Function GetShortPathName Lib "kernel32" _  
    Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _  
    ByVal lpszShortPath As String, ByVal cchBuffer As Long) As Long  
  
Function RefreshLinks()  
    On Error GoTo ErrorHandler
```

source MSDN traduction Fabrice Constans

```

'Défini l'objet catalogue ADOX
Dim objCat As New ADOX.Catalog
'Défini l'objet table ADOX
Dim objTbl As ADOX.Table

'Nom de la base pour la table liée
Dim strFilename As String
'Chemin et nom de la base de la table liée
Dim strFullName As String

Dim blnIsMapi As Boolean
Dim blnIsImex As Boolean
Dim blnIsTemp As Boolean
Dim blnLongFileName As Boolean
Dim blnFailedLink As Boolean
Const srtImex = "IMEX"
Const strMapi = "MAPILEVEL="

'ouvre le catalogue
objCat.ActiveConnection = CurrentProject.Connection

'parcours la collection de tables et met à jour le lien
For Each objTbl In objCat.Tables
    'Vérifie que la table est une table liée
    If objTbl.Type = "LINK" = True Then
        blnIsTemp = objTbl.Properties("Temporary Table") Or Left(objTbl.Name, 1) = "~"
        blnIsImex = (InStr(1, objTbl.Properties("Jet OLEDB:Link Provider String"), srtImex,
vbTextCompare) > 0)
        blnIsMapi = (InStr(1, objTbl.Properties("Jet OLEDB:Link Provider String"), strMapi,
vbTextCompare) > 0)

        If Not blnIsTemp And Not blnIsImex And Not blnIsMapi Then
            'vérifie que c'est une table JET.
            strFullName = objTbl.Properties("Jet OLEDB:Link Datasource")
            strFilename = Mid(strFullName, InStrRev(strFullName, "\", Len(strFullName)) + 1,
Len(strFullName))
            'détermine si la base existe
            If DoesFileExist(strFullName) = True Then
                objTbl.Properties("Jet OLEDB:Link Datasource") = GetShortName(strFullName)
            'met à jour le lien de la table en nom court
            Else
                MsgBox "Mise à jour impossible : '" & objTbl.Name & "'" &
                _String(2, vbCrLf) & "Fichier non-trouvé: " & vbCrLf & strFullName
                blnFailedLink = True
            End If
            If InStr(strFilename, ".") > 9 Then blnLongFileName = True
        End If
    End If
Next

If blnFailedLink = False Then
    If blnLongFileName = True Then
        MsgBox "Le lien de la table a été mis à jour avec succès,
cependant le nom de la base serveur n'est pas au format 8.3" & _
vbCrLf & "Veuillez renommer le fichier, refaire le lien et relancer la procédure.",
vbExclamation
    Else
        MsgBox "Le lien a été correctement mis à jour. ", vbInformation
    End If
Else
    MsgBox "Les liens n'ont pu être mis à jour." & vbCrLf & "Vérifier la cohérence des liens.",
vbExclamation
End If

ExitHandler:
Exit Function
    
```

source MSDN traduction Fabrice Constans

```

ErrorHandler:
    MsgBox Err.Description & " " & Err.Number
    Resume ExitHandler

End Function

Function GetShortName(ByVal sLongFileName As String) As String
    Dim lRetVal As Long, sShortPathName As String, iLen As Integer
    'paramètre le tampon de retour de l'API.
    sShortPathName = Space(255)
    iLen = Len(sShortPathName)

    'appel de la fonction
    lRetVal = GetShortPathName(sLongFileName, sShortPathName, iLen)
    'supprime les caractères inutile
    GetShortName = Left(sShortPathName, lRetVal)
End Function


Function DoesFileExist(strFileSpec As String) As Boolean
    'True si le fichier spécifié dans strFilespec existe.
    'False si strFileSpec n'est pas valide
    'Nom du fichier si strFileSpec est un répertoire
    Const INVALID_ARGUMENT As Long = 53
    On Error GoTo DoesfileExist_Err
    If (GetAttr(strFileSpec) And vbDirectory) <> vbDirectory Then
        DoesFileExist = CBool(Len(Dir(strFileSpec)) > 0)
    Else
        DoesFileExist = False
    End If
DoesfileExist_End:
    Exit Function
DoesfileExist_Err:
    DoesFileExist = False
    Resume DoesfileExist_End
End Function
    
```

V-A-2 - Mettre en cache les noms longs (Postes clients uniquement)

Ce problème touche surtout Windows 2000. Le système ne met pas en cache les noms longs. En réseau les performances peuvent être catastrophique lors de l'ouverture d'une base de données ou le lancement de requête. On a pu observer des temps d'attentes de 35 secondes avant une réponse.

 Le Service Pack 4 de Windows 2000 corrige ce problème.

 Article correspondant sur MSDN

 Les modifications de la base de registres sont à faire en dernier recours. Elles peuvent endommager le système d'exploitation. Celles-ci sont effectuées à vos risques et périls. Je ne pourrais être tenu responsable d'une quelconque perte de données ou d'exploitation.

Recherchez la clef suivante :


```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\MRxSmb\Parameters
```

Créez la valeur suivante :

```
InfoCacheLevel  
Type = Hexadécimal  
Valeur = 0x10
```

- 1 0 : désactive la mise en cache pour tous les fichiers.
- 2 1 : active la mise en cache si fichiers short le nom de fichier (8.3) - si cela est la valeur par défaut.
- 3 10 : active la mise en cache pour tous les fichiers.

V-A-3 - Désactiver la génération automatique de noms courts (Postes Serveur)

 *Vous ne devez exécuter cette modification que si vous n'utilisez pas de programmes 16bits. Dans le cas contraire, les fichiers ne seront plus accessible par ces applications.*

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem  
NtfsDisable8dot3NameCreation  
Valeur = 1
```

V-B - Délai de notification de partage (serveur uniquement)

Cette opération modifie le délai de réaction du serveur lors des notifications d'échecs de partage. Plus le délai est court plus vite la demande est réitérée et donc susceptible d'aboutir.

Il faut agir directement sur la base de registre du serveur. Trouvez la clef suivante :

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters
```

Normalement vous devez voir 2 valeurs :

```
SharingViolationRetries  
SharingViolationDelay
```

Mettez 0 pour les 2. Si elles n'existent, pas créez-les ! Elles sont du type DWORD.

V-C - Problème d'accès au fichier ldb (Moteur JET uniquement).

Nous allons simplifier le mécanisme du fichier ldb pour une meilleure compréhension.

Lors de la connexion du premier utilisateur à un fichier mdb ou mde, un fichier .ldb est créé.

A la fermeture du fichier par le dernier utilisateur, ce fichier est supprimé.

Tous les utilisateurs entre le premier et le dernier se connectent à ce fichier unique.

Chaque fois qu'il accède à des enregistrements, une information est inscrite dans ce fichier indiquant le ou les enregistrements verrouillés. Ce fichier est donc fondamental pour le fonctionnement de JET.

Il arrive que cette connexion échoue ce qui retarde les interrogations aux fichiers de données.

Pour palier à ce problème, vous pouvez créer un accès permanent grâce à une technique simple.

- Dans le fichier dorsal (fichier sur le serveur contenant les données).
- Créez une table que vous nommez **tbl_ldb**.
- Créez un champ de type booléen.
- Attachez cette nouvelle table dans le fichier frontal (fichier du poste client contenant l'IHM).
- Ouvrez l'éditeur VB. Dans le Menu **Outils/Références**, vérifiez que vous avez bien activé le **Microsoft DAO x.x** correspondant à votre version (3.6 en général).
- Dans un module standard ajoutez le code suivant :

```
Public rstLienOuvert As DAO.Recordset
```

- Sauvez le module et fermez l'éditeur VB.
- Créez un formulaire que vous nommerez **Frm_ldb**
- Dans les propriétés du formulaire, repérez **Sur Ouverture** et créez un code VBA.
- Ajoutez ce code :

```
Set rstLienOuvert = CurrentDb.OpenRecordset("tbl_ldb")
```

- Toujours dans les propriétés, repérez l'événement **Sur Fermeture** et créez comme précédemment un code VBA.
- Insérez le code suivant.

```
rstLienOuvert.Close  
Set rstLienOuvert = Nothing
```

- Au lancement, ouvrez ce formulaire avec le statut caché.


```
DoCmd.OpenForm "Frm_ldb", , , , , acHidden
```

A la fermeture de l'application, le formulaire caché se fermera ; pendant l'intervalle, il n'y aura pas de déconnexions intempestives donc pas d'attente de reconnexion.

VI - Conclusion

La mise en réseau d'une application ACCESS n'est pas compliquée, le plus dur est de trouver un consensus entre rapidité acceptable et IHM fonctionnelle. Si malgré l'application de ces consignes vous n'arrivez pas à obtenir des performances raisonnables, passez sur MSDE, SQL Server ou autres base de données client/serveur. Attention cependant à bien adapter le code VBA et SQL en conséquence.

Rushmore, showplan, Setoption et réglage de la Base de registre auraient nécessité un tutoriel à eux seuls. Rassurez-vous, nous y reviendrons prochainement !

 *Notez que si votre réseau a des faiblesses, vous pourrez choisir le plus performant des moteurs de base de données vous n'en tirerez aucun bénéfice.*

VI - Liens importants

Pour en savoir plus :

 **ACCESS 2007 valables pour les versions antérieures**

 **En savoir plus (Système Windows 2000 et XP avec bases partagées)**

 **Spécial ODBC**

 **Requêtes paramétrées et ODBC**

 **Tableau croisé dynamique**

 **Moteur Jet et Expressions**

 **Optimisation client/serveur avec Access**

 **Perte de performance NT4, 2000 et XP**

VIII - Remerciements

Je tiens à remercier : **Starec** pour le temps passé en relecture et pour ses remarques judicieuses.

Maxence Hubiche pour ses bons liens et ses observations. **Tofalu** pour ses conseils. L'équipe de **Developpez.com** pour la qualité du site.

Nono40 pour son superbe éditeur XML et tout le temps qu'il passe pour nous donner toujours plus de fonctionnalités.

Je présente mes plus plates excuses à ceux que j'aurais omis de remercier.

