Script de mise en miroir de base de données SQL Server

par SQLPro

Date de publication : 12/03/2009

Dernière mise à jour : 24/05/2009

Contrairement à une idée hélas de plus en plus répandue, il est inutile et même fortement nuisible d'arrêter MS SQL Server pour effectuer des sauvegardes. De même, la copie des fichiers d'une base de données, sans quelques précautions préalable, peut entraîner la perte irrémédiable de la base... Voici donc quelques explications sur ce qu'est une sauvegarde de bases de données et quels en sont les différents modes.

I - Principes

Dans un SGBDR de type Client/Serveur, toutes les opérations doivent être faite à chaud : création de bases, création de nouveaux espaces de stockage, déplacement des espaces de stockage, sauvegardes des données ou des transactions, reconstruction des index ou défragmentation logique des structures de données... toutes ces opérations de maintenance courante sont entreprise alors que les bases de données continuent d'être utilisées par de nombreuses applications. Le principe étant simple : un SGBDR doit pouvoir fonctionner 24 heures sur 24 et 365 jours par an. Aujourd'hui la plupart des constructeurs de serveurs (PC) et les grands éditeurs de SGBDR s'accordent pour offrir une continuité de services de cinq 9, c'est à dire 99,999 % du temps. Calculez par vous-même, cela ne vous laisse que 8 minutes et 46 secondes pendant lequel le SGBDR est à l'arrêt dans une année. Bref, à peine de quoi passer un patch critique nécessitant un arrêt machine !

Comment se fait-il alors que certains internautes demandent encore si on peut faire de la sauvegarde à chaud ???

Il y a plusieurs explications :

- de nombreux informaticiens n'ont qu'une vague idée de ce qu'est un SGBDR. Ils ont tout juste aperçut du Access ou du MySQL... Ils croient encore qu'une base de données ce sont des "fichiers client", "fichier facture", etc alimentés à coup "d'enregistrement"... Or le concept de "fichier" n'existe pas dans l'univers de SGBDR, pas plus que celui d'enregistrement, un SGBDR C/S n'ayant pas pour vocation de reproduire les mêmes mécanismes que ceux de Cobol des années 50 !
- beaucoup parmi ces informaticiens viennent de MySQL qui ne sait toujours pas réaliser des sauvegardes à chaud garantissant la consistance de la base de données. Tous les SGBDR moderne, à l'exception hélas de MySQL, savent depuis longtemps faire des sauvegardes à chaud sans altérer la continuité du service, mais la mode est au pseudo libre et MySQL AB (l'entreprise commerciale qui a créé MySQL, racheté par SUN puis pas Oracle) a réussit un excellent coup mercatique en noyant les hébergeurs de licences pseudo gratuites appauvrissant ainsi la culture base de données, déjà très faible au niveau scolaire !!!

II - Une commande de sauvegarde

Dans MS SQL Server, c'est la commande BACKUP qui permet d'effectuer les sauvegardes. Il existe différent types de sauvegarde pour différents usages, mais un point de passage obligé est de commencer par une sauvegarde complète de la base. Cette sauvegarde complète de la base consiste à capturer binairement toutes les pages de données et y ajouter toutes les transactions (donc toutes les modifications) entreprises depuis le démarrage de la sauvegarde jusqu'à ce que la sauvegarde prenne fin. Ainsi, la sauvegarde contient une image des données de la base, non pas à l'heure du début de celle-ci mais bien à l'heure de fin. Par exemple si vous avez lancé une sauvegarde à 22 heures et qu'elle se termine à 23h30, l'état de la base sera celui des données à 23h30 et non à 22 heures !

L'algorithme utilisé est en fait simple dans son principe, mais assez subtil : sachant qu'une base de données est découpées en page de 8 Ko, le mécanisme d'une sauvegarde complète consiste à copier en premier les pages les moins les moins souvent modifiées pour terminer par les pages les plus souvent modifiées. À la première page déjà sauvegardée, modifiée par un utilisateur, SQL Server place une marque dans le journal de transaction pour savoir à partir de quel endroit il va devoir copier les transactions depuis le journal. Pour savoir quelles sont les pages les plus ou les moins modifiées, SQL Server implémente un algorithme en logique floue (vous savez cette logique bretonne du "ni oui, ni non") c'est à dire une valeur décimale comprise entre 0 (jamais modifiée) et 1 (toujours modifiée). Bien entendu les pages vides ne sont pas sauvegardées. En revanche, les pages sauvegardées sont stockées telles quelles c'est à dire qu'elles contiennent exactement les mêmes données structurées de la même façon, fragmentation comprise).

Notons aussi que cette façon de procéder possède un avantage : si la base de données a été créée avec une taille prévisionnelle importante (par exemple 600 Go) alors qu'elle ne contient effectivement que peu de données (par exemple 3 Go), alors la sauvegarde fera au mieux 3 Go et au pire quelques gigaoctets de plus si une forte activité transactionnelle a été entreprise pendant la sauvegarde...

C'est pourquoi nous vous recommandons d'éviter les opérations lourdes pendant la sauvegarde et de la planifier aux heures creuses.

Enfin, disons que c'est le seule moyen d'avoir à la fois une sauvegarde consistante et cohérente de la base de de données et effectuée à chaud. C'est aussi comme cela que fonctionne IBM DB2 ou Oracle...

III - Un petit exemple

Voici comment lancer une sauvegarde d'une base appelée DB_mabase en produisant un fichier nommé SAUVE_mabase.bak dans le chemin "C:\MesSauevegardes\" :

```
BACKUP DB_mabase TO DISK = 'C:\MesSauevegardes\SAUVE_mabase.bak'
```

Notez qu'il n'y a aucune extension réservé pour les fichiers de SQL Server à quelque niveau que ce soit et que vous auriez donc pu nommer ce fichier SAUVE_mabase.sav ou encore tartemuche.oligore.

IV - FAQ

IV-A - Le mythe de l'arrêt du serveur Windows...

Une légende urbaine veut que les OS Windows doivent être arrêter régulièrement sinon on risque les pires ennuis... S'il est vrai que certains OS Windows ont connus quelques problèmes par le passé (NT 4 par exemple), cela était rarement dû à Windows lui même, mais à des logiciels mal écrits qui entrainaient notamment des fuites de mémoire que l'OS n'avait pas prévu de contrôler. Tous ces phénomènes pouvant conduire à des pannes ont été parfaitement annihilés à partir du service pack 4 de Windows NT 4. Or certains administrateurs systèmes avaient trouvés une parade peu subtile consistant à arrêter le serveur plutôt que de chercher quel service ou application ils devaient relancer ! Dès lors cette légende était née, savamment entretenu par quelques ayatollah Linuxiens (**il est a remarquer qu'une recrudescence importante des intégristes du monde Linux est en cours - un spécialiste reconnu de ce monde-là m'a fait remarquer récemment ce phénomène qui a pris de l'ampleur, sans doute à cause du fait que Linux ou Windows sont aujourd'hui aussi avantageux ou inconvenant l'un que l'autre !**).

Aujourd'hui il y a encore des informaticiens (?) qui croient encore que cet arrêt est nécessaire.

Visiblement, ils ne maîtrisent pas l'informatique et devraient plutôt s'orienter vers la mécanique, qui constitue une voie de garage intéressante pour ceux qui ne voient pas plus loin que leur bout de nez !

IV-B - Pourquoi ne pas arrêter le serveur ?

Il n'y a en aucune raison d'arrêter un serveur Windows régulièrement en pensant lui faire du bien. C'est plutôt lui faire du mal, tant au niveau logique que physique ! D'ailleurs notez que pour tenter d'enrayer ce phénomène, Windows vous oblige à spécifier quelle est la raison qui à motivé l'arrêt du serveur... Au niveau physique, de nombreux éléments militent en faveur de la continuité de service. Prenons un seul exemple : les disques durs ont statistiquement beaucoup plus de "chance" (**nous devrions dire risque, mais les statisticiens raisonnent en chances !**) de lâcher au démarrage que pendant leur activité ordinaire de lecture/ecriture. Au niveau logique, de nombreuses informations sont collectées afin d'auditer le fonctionnement du serveur (compteurs de performances par exemple), mais ces données sont volatiles, car uniquement en mémoire. L'arrêt du serveur réinitialise la plupart de ces compteurs et les mesures en sont perdues à tout jamais !

IV-C - Pourquoi ne pas copier les fichiers de la base ?

La copie pure et simple des fichiers de la base semble être une idée séduisante en matière de sauvegarde. C'est d'ailleurs le seul moyen qu'il y a de faire une sauvegarde consistante et cohérente d'une base de données MySQL du fait de sa structure particulière (n tables = n fichiers). Pour comprendre pourquoi cette manière de faire peut s'avérer dangereuse, il faut comprendre quel est la relation entre la notion de fichier (aspect physique) et une base de données (aspect logique).

Dans SQL Server, les bases de données sont constituées d'au moins deux fichiers (mais souvent plus) : l'un contient des données, l'autre est le journal. Peu importe les extensions (mdf, ndf, ldf... df signifiant data file) ce n'est qu'au niveau logique que SQL Server les reconnaîtra. Mais vous pouvez avoir autant de fichiers de données que nécessaire. Par exemple pour des bases de données de grande dimension, on placera la base sur plusieurs disques physiques, et c'est pourquoi on créera autant de fichiers physiques, chacun sur un disque différent. De la même manière on peut créer un journal de transaction sur plusieurs fichiers. Il n'y a pas de lien logique direct entre un fichier de données et les objets logique ce qui signifie que sans spécification particulière, pour les bases ayant plusieurs fichiers de données, les tables seront équi-réparties dans tous les fichiers et cela afin de paralléliser les lectures et les écritures...

Il est à noter que cette manière de faire est identique pour Oracle, Sybase ou IBM DB2...

Lorsque le serveur est en activité, donc que les bases sont en ligne, SQL Server ouvre tous les fichiers de toutes les bases en accès exclusif pour le moteur de stockage. Aucun autre service ou application ne peut donc y accéder. On ne peut donc copier les fichiers tant que SQL Server tourne ou bien que la base est en ligne.

Il est bien entendu possible de **mettre la base de données hors ligne** par une commande Transact SQL de manière à ce que les fichiers soient libérés de leur accès exclusif.

Exemple :
```
ALTER DATABASE <nomBase> SET OFFLINE
```

La base de données est toujours reconnue, mais tout accès y est impossible. Mais cette commande provoque le vidage du cache, et notamment des tables système, et provoque la remise à zéro des compteurs (DMV, compteur des performances...) pour cette base... Ce qui signifie que les requêtes qui devront être faites sur cette base vont devenir épouvantablement lente (recalcul de plan de requête systématique et lecture disque impérative...) donc des performances particulièrement mauvaises pour toutes les requêtes qui vont être entreprises à la reprise du service des données, car, il ne faut pas l'oublier, un SGBDR travaille exclusivement en cache, c'est à dire qu'il extrait de la RAM les données et les plans de requête précompilés qui stagnent en mémoire.

L'avantage de cette commande est qu'elle ferme proprement la base de données sans rompre le lien logique entre les fichiers et le nom de la base. On peut alors copier les fichiers de cette base.

Bien entendu, pour remettre la base en ligne il suffit de faire :

```
ALTER DATABASE <nomBase> SET ONLINE
```

Autre possibilité, **détacher la base de données**. Cela s'effectue à l'aide de la procédure stockée de détachement.

Exemple :
```
EXEC sp_detach_db '<nomBase>'
```

Cette commande permet de détacher la base de données et ses fichiers, mais réalise préalablement le transfert des données des transactions validées dans les fichiers de données correspondant. En revanche, cette base de données n'est plus connue du serveur.
De la même manière que la précédente, cache et compteurs sont définitivement perdus.

Avant de pouvoir exécuter l'une ou l'autre de ces commandes, il faut s'assurer que plus aucune connexion ne s'effectue à cette base. Pour cela vous pouvez voir les connexions sur la base de votre choix en exécutant la requête :

```
SELECT *
FROM sys.dm_exec_connections AS ec
  INNER JOIN sys.dm_exec_sessions AS es
  ON ec.session_id = es.session_id
  INNER JOIN sys.dm_exec_requests AS er
  ON ec.session_id = er.session_id
WHERE database_id = DB_ID('<nomBase>')
```

Bien entendu vous pouvez **forcer une déconnexion impérative et immédiate de tous les utilisateurs** à l'aide de la commande :

```
ALTER DATABASE <nomBase> SET SINGLE_USER WITH ROLLBACK IMMEDIATE
```

Vous devenez alors l'unique utilisateur de cette base de données (notez qu'il vous faut les privilèges du plus haut niveau relatif à cette base pour ce faire).

Vous pouvez être moins violent dans cette commande en utilisant les options :

```
ROLLBACK AFTER <em>n</em> SECONDS
```

qui permet d'attendre in nombre déterminé de seconde avant de commencer l'annulation des transaction en cours, avec l'assurance qu'aucune nouvelle transaction ne va être entreprise)

```
NO WAIT
```

qui permet de passer en utilisateur unique immédiatement si aucune transaction n'est en cours.

Revenir en arrière consiste à **rattacher la base** et cela s'effectue par la commande :

```
CREATE DATABASE <nomBase>
ON ( <???> )
FOR ATTACH
```

Dans laquelle il faut remplacer le marqueur <???> par les spécifications de tous les fichiers composant la base.

Exemple :
```
CREATE DATABASE DB_AIRBUS_A380
ON ( FILENAME = 'E:\DATASQL\db_A380_tab1.dt',
     FILENAME = 'F:\DATASQL\db_A380_tab2.dt',
     FILENAME = 'G:\DATASQL\db_A380_idx1.dt',
     FILENAME = 'H:\DATASQL\db_A380_jt1.dt'    )
FOR ATTACH
```

On peut même en profiter pour oublier les fichiers du journal de transactions fin que cette commande les recrée vierge à l'aide de l'option :

```
FOR ATTACH_REBUILD_LOG
```

La dernière solution, la pire bien entendu, consiste donc à arrêter le serveur et copier tous les fichiers.... Là la perte de mise en cache et des compteurs est totales : toutes les bases de production ainsi que les bases systèmes voient leurs caches vidés et tous les compteurs sont remis à zéro... au redémarrage, le serveur repart comme un nouveau né auquel il faut tout réapprendre !

IV-D - Perte du cache, perte des compteurs... Si important que cela ?

Comme indiqué dans la phase précédente, pendant que votre serveur de bases de données travaille, il accumule une quantité d'information destiné à le faire travailler de plus en plus vite. En cas d'arrêt il perd tout et doit réapprendre ce qu'il a accumulé pendant de nombreux jours. Ainsi les performances vont chuter au plus bas chaque fois que vous redémarrez votre serveur...

Ainsi, pour aller le plus vite possible, toutes les requêtes sont effectuées sur des données mise en cache. Mais pour cela il faut savoir lesquelles mettre en cache. C'est le rôle du moteur de stockage que de collecter suffisamment de données pour faire en sorte que les données les plus utilisées soient plus souvent en mémoire que celles qui y sont peu. Mais pour faire cela il faut que votre serveur ait vécu de manière continuelle longtemps, afin d'accumuler

suffisamment d'information pour le faire à l'optimum. Chaque fois que vous arrêterez une base, ou pire votre serveur, toutes les données du cache pour la base ou l'ensemble du serveur sont vidées, y compris toutes les statistiques accumulées et les tables systèmes (**celles qui permettent de savoir ce que contient la base de données, comment est formé telle ou telle table...**).

Autrement dit, tant que votre serveur travaille, et le plus longtemps possible de manière continue, plus de 95% des requêtes s'effectuent sur des données déjà en RAM. Chaque fois que vous arrêtez le serveur, vous l'obliger à lire le disque pour toutes les nouvelles requêtes après le redémarrage. Bref, avant arrêt vous faites du client/server en RAM et après arrêt vous faites de l'accès aux fichiers style COBOL. Or la différence est de taille : 9 ns pour un accès RAM, 9 ms pour un accès disque soit 1000000 (un million...) de fois plus rapide intrinsèquement. En fait, à cause des bus et autres registres, on considère que l'écart de vitesse entre lectures RAM et lecture disque est d'au moins 1000... Si vous cela ne vous parle pas, vos utilisateurs sauront faire la différence !

IV-E - Perdre une base en faisant une sauvegarde par copie, c'est possible ?

Lorsque vous arrêtez le serveur (**soit logique, à savoir SQL Server, soit physique, à savoir la machine**), c'est en fait une opération d'urgence que vous effectuez. Le service SQL Server reçoit l'ordre de s'arrêter et doit obtempérer en un minimum de temps. Certaines opérations d'écriture qui sont en attente ne vont pas être effectuées, mais resterons en "stand by" dans le journal de transaction. Tant est si bien qu'en cas de corruption de l'un ou l'autre des fichiers (données ou journal), la base ne pourra pas être reconstruite. En revanche, les procédures Transact SQL destinées à détacher une base de données ou bien à la mettre hors ligne, finalise les écritures des transactions qui sont achevées mais non encore reportées dans les fichiers de données.

Vous comprenez maintenant pourquoi il est dangereux d'effectuer une sauvegarde avec des actions aussi brutales que l'arrêt du service SQL Server...

V - Les différents modes de sauvegardes

Nous avons dit que SQL Server permettait de faire différentes sauvegardes. En voici les détails :

V-A - Sauvegarde complète

Point de départ de toutes sauvegardes. Aucune autre espèce de sauvegarde ne peut être entreprise sans qu'une sauvegarde complète soit opérée. Conseil ; faites une sauvegardes complète le plus tôt possible, c'est à dire juste après l'avoir créée. Nous verrons pourquoi plus loin.

V-B - Sauvegarde différentielle

Consiste à ne sauvegarder que les pages de données (data, index, blobs) qui ont été modifiées depuis la dernière sauvegarde complète.

La syntaxe est :

```
BACKUP DATABASE <nomBase>
TO DISK = '<destination>'
WITH DIFFERENTIAL
```

V-C - Sauvegarde du journal de transaction

Consiste à capturer depuis le fichier actif du journal de transaction, toutes les transactions qui se sont déroulées depuis la dernière sauvegardes, quelle qu'elle soit (complète, différentielle ou journal de transaction).

La syntaxe est :

```
BACKUP LOG <nomBase>
```

La syntaxe est :
```
TO DISK = '<destination>'
```

V-D - Sauvegarde de groupe de fichiers

Consiste à sauvegarder un ou plusieurs groupe de fichiers spécifiques (espaces de stockage) mais indépendant logiquement du reste de la base de données (en particulier pas de lien d'intégrité référentielle). Pour en comprendre la logique et l'intérêt il vous faut savoir comment sont structurés les espaces de stockage d'une base MS SQL Server.

La syntaxe est :
```
BACKUP DATABASE <nomBase>
{ FILE = <nomFichier> | FILEGROUP = <nomGroupeFichiers> , ... }
TO DISK = '<destination>'
```

V-E - Sauvegarde différentielle de groupe de fichiers

Consiste en une sauvegarde différentielle appliquée à un ou plusieurs groupes de fichiers logiquement indépendants du reste de la base.

```
BACKUP DATABASE <nomBase>
{ FILE = <nomFichier> | FILEGROUP = <nomGroupeFichiers> , ... }
TO DISK = '<destination>'
WITH DIFFERENTIAL
```

V-F - Autres options

La commande de sauvegarde possède plus d'un trentaine d'options. En voici les principales...

V-F-1 - Sauvegardes sur un device

Vous pouvez créer un "device", en fait un super fichier pour y mettre toutes vos sauvegardes. Cela est très pratique notamment pour éviter de chercher ou sont les différentes sauvegardes d'une même base. Voir l'article que j'ai écrit à ce sujet : **De l'intérêt des devices pour les sauvegardes**.

V-F-2 - Sauvegarde multiples

Il est possible de générer jusqu'à 3 sauvegardes simultanée. Par exemple pour générer une sauvegarde locale en même temps qu'une sauvegarde distante. C'est l'option MIRROR TO.

V-F-3 - Sauvegarde distante

Il est possible de sauvegarder sur une ressources distante, mais à condition d'utiliser la convention de nommage UNC (\\MonServeur\MonRépertoire ...). Il faut bien entendu s'assurer que la connexion qui lance cette opération possède les droits systèmes adéquats.

V-F-4 - Sauvegarde en parallèle

Permet de générer une sauvegarde éclatée sur différents dispositifs (fichiers ou bande).

V-F-5 - Sauvegarde sur bande

Permet de piloter directement un dispositif de sauvegarde à bande (à la place de DISK, mettez TAPE) : commandes REWIND, NOREWIND, UNLOAD, NOUNLOAD.

V-F-6 - Compression de la sauvegarde

Option COMPRESSION.

V-F-7 - Options de conservation (pour les devices)

NOINIT, INIT (ajoute ou remplace les nouvelles sauvegardes au jeu de sauvegardes déjà présents sur le support de sauvegarde), NOSKIP, SKIP (vérifie la date et l'heure d'expiration des jeux de sauvegardes figurant sur le support avant de les écraser), NOFORMAT, FORMAT (écrase ou non l'en-tête et les jeux de sauvegardes existants dans le device), EXPIREDATE = 'date', RETAINDAYS = days (spécifie la date ou le nombre de jour écoulé après laquelle le jeu de sauvegarde peut être écrasé).

V-F-8 - Vérification de la sauvegarde

MEDIAPASSWORD (crypte la sauvegarde), NO_CHECKSUM, CHECKSUM (effectue ou non une somme de contrôle), STOP_ON_ERROR, CONTINUE_AFTER_ERROR (s'arrête ou pas à la première erreur), RESTORE VERIFYONLY (vérifie la consistance logique du fichier de sauvegarde).

On comprend à la richesse de ces options qu'il faut être prudent dans le maniement de cette commande.

VI - Conclusion

La sauvegarde est une opération importante qui s'étudie de manière rigoureuse. On ne fait pas une sauvegarde pour sauvegarder des données. On fait une sauvegarde dans le but d'avoir un jour à restaurer... Ça n'est pas la même chose.

Cela veut simplement dire que l'élaboration d'une stratégie de sauvegarde commence par les deux questions fondamentales :

- combien de données puis-je me permettre de perdre ?
- combien de temps vais-je mettre pour restaurer mon système ?

Tout ceux qui n'ont pas étudié cette problématique et bien entendu ne se sont pas préparés en conséquence, notamment en simulant une restauration, n'ont généralement aucune chance de récupérer leur base de données.

Pour vous donner une idée de l'importance et de la complexité de la chose, les cours d'administration de SQL Server nécessitent de passer au minimum 3 heures (et bien souvent 5 à 6) entre présentation de la chose et TP...