

# Optimisation des bases de données MS SQL Server

Seconde partie : Le serveur : ressources physiques, ressources logiques

par Frédéric Brouard

Date de publication : 30 mai 2008

Dernière mise à jour :

*Je pose souvent la question en ces termes lors des formations que je donne : parmi les différentes formes d'informatique, laquelle nécessite les machines ayant les plus grandes ressources ?*

*La plupart du temps, les étudiants et les stagiaires m'affirment que c'est l'informatique scientifique, gavée des exploits des antiques Cray et de Deep Blue et des monstres utilisés pour les calculs de la météo. D'autres pensent que ce sont les machines de la conquête spatiale... Peu savent que l'on trouve les systèmes les plus étoffés dans l'informatique de gestion. Un système comme SABRE de United Airlines fût longtemps l'un des systèmes de gestion de bases de données les plus énormes qui soit.*

*Ce besoin de ressources est lié à deux composantes : des calculs, certes souvent peu complexes, mais surtout la volumétrie des données à manipuler. Des quantités de données parfois si gigantesques qu'il convient de répartir la charge sur de multiples machines car pour certaines bases, aucun ordinateur au monde n'a encore la capacité de traiter seul et dans des temps de réponse acceptables les masses des données en jeu.*

Copyright et droits d'auteurs : la Loi du 11 mars 1957 n'autorisant aux termes des alinéas 2 et 3 de l'article 41, d'une part que des copies ou reproductions strictement réservées à l'usage privé et non [...] à une utilisation collective, et d'autre part que les analyses et courtes citations dans un but d'illustration, toute reproduction intégrale ou partielle faite sans le consentement de l'auteur [...] est illicite. Le présent article étant la propriété intellectuelle conjointe de Frédéric Brouard et de SQL Server magazine, prière de contacter l'auteur pour

toute demande d'utilisation, autre que prévu  
par la Loi à [SQLpro@SQLspot.com](mailto:SQLpro@SQLspot.com)

Finalement un serveur de bases de données relationnelle Client/Serveur n'est rien d'autre qu'un ordinateur dont on a sciemment atrophié certains éléments afin de les rendre plus performant qu'un PC de bureau.

Voyons ce qu'un serveur n'a peu ou prou besoin. Il n'a pas bien besoin d'un écran puisque qu'il se doit d'être scruté par l'intermédiaire d'autres machines. Il n'a pas non plus réellement besoin d'un clavier ni d'une souris pour les mêmes raisons.

En revanche, nous pouvons convenir qu'il a besoin de beaucoup de mémoire comme nous l'avons vu au chapitre précédent. Il a besoin de processeurs rapides (notez le pluriel), de disques de grande capacités dotés des temps de réponses les plus courts.

Ce sont donc ces trois axes que nous allons étudier dans cet article. Nous verrons cela du côté physique puis du côté logique.

Nous en tirerons quelques nouvelles règles propres à établir les préconisations que tout un chacun doit pouvoir spécifier afin de choisir une machine et la configurer au mieux en fonction de son budget.

## I - La mémoire

Plus il y en a et mieux c'est. À tous les niveaux. La mémoire vive de l'ordinateur, c'est à dire la RAM (Random Access Memory), mais aussi toutes les antémémoires possibles et imaginables, que l'on nomme " cache ". Cache au niveau processeur, cache au niveau disque.

Les caches servent la plupart du temps à stocker des données temporaires des fichiers du disque. Si une même demande intervient alors que le l'antémémoire la contient alors aucune lecture ne sera réalisée directement sur le disque et cela accélère le débit.

Mais il est bien évident qu'un tel cache est toujours limité en taille. Il possède aussi un inconvénient majeur : celui de bousculer le fonctionnement particulier de SQL Server si le cache est en écriture, à moins qu'il n'ait été spécialement conçu pour travailler avec un SGBDR C/S.

De la même façon, on trouve des caches côté processeurs.

Mais revenons à la mémoire vive. Sur un OS 32 bits, la mémoire directement adressable ne dépasse pas 4 Go. C'est à dire que seuls 4 milliards de blocs de 8 bits <sup>(1)</sup> peuvent être accédés de manière directe en spécifiant un numéro, un peu à la manière du téléphone qui, en France, permet de joindre 699 999 993 abonnés <sup>(2)</sup> et pas un de plus. En outre l'OS Windows serveur possède une petite particularité : si vous mettez 4 Go de données, il s'en réservera deux pour son usage exclusif et laissera les deux autres aux applications. Cela convient à la plupart des applications de serveurs qui utilisent en principe massivement l'OS. Or ce n'est pas le cas de SQL Server. Pourquoi ? *Parce que SQL Server possède son propre OS !* Un OS qui s'occupe de la gestion du stockage des données sur les disques, de la pagination des données et des procédures en mémoire, et enfin de la gestion concurrentielle des traitements. Bref, SQL Server fait plus de 50% du travail habituel de l'OS, ne laissant effectivement à Windows que les bribes de la gestion réseau. Dès lors, il serait intéressant de donner à SQL Server un peu plus de mémoire que le 50/50 de la répartition habituelle. Et bien cela est prévu. Un paramétrage spécial de l'OS au démarrage permet de lui indiquer qu'il ne devra prendre qu'un seul giga octet de RAM et réserver le reste aux applications, en l'occurrence à l'usage quasi exclusif de SQL Server (commutateur 3/GB).

Mais que se passe t-il si l'on a besoin de plus de mémoire vive que la fatidique barre des 4 Go ? Il existe deux techniques pour ce faire : la pagination de la mémoire virtuelle en RAM et le 64 bits...

### I-A - Pagination de la mémoire virtuelle en RAM

Peut être n'avez vous jamais remarqué ce fichier de nom pagefile.sys <sup>(3)</sup> situé en racine de votre disque C <sup>(4)</sup>. C'est lui qui assure la mémoire virtuelle de l'OS. En fait lorsqu'une application n'a pas assez de mémoire pour s'exécuter, ou encore pour charger toutes les données qu'elle doit utiliser en RAM, le fichier pagefile.sys sert de réservoir à ce trop plein. Devinez quel algorithme est utilisé pour décharger dans ce réservoir certaines parties de la RAM ? Notre fameux LRU <sup>(5)</sup> ! Autrement dit les pages de RAM les plus anciennement scrutées se déversent dans le fichier de mémoire virtuelle afin de faire profiter d'un espace RAM libre au nouvel arrivant.

Mais cette gestion à travers le disque a un coût assez élevé : il faut écrire et lire ce fichier sans arrêt et l'on sait dorénavant que le temps de réponse d'un disque est incommensurablement plus long qu'une lecture en mémoire vive ! L'idée est alors de remplacer ce fichier par de la RAM... Puisque le processeur n'est pas capable d'adresser plus de 4 Go de RAM, faisons " glisser " la RAM supplémentaire vers le processeur pour que ces nouvelles parties

(1) 32 bits, signifie que l'adresse mémoire la plus " haute " est :  $2^{32} - 1 = 4\ 294\ 967\ 295$

(2) Il y a 7 préfixes : 01, 02, 03, 04, 05, 06 (portables) et 08 (numéros exotiques...) et pour chacun de ces préfixe 99999999 combinaisons possibles.

(3) S'il est caché, démasquez-le en paramétrantg votre explorateur windows proprement : menu Outils / Options des dossiers, onglet affichage, item fichiers et dossiers cachés, case à cocher " Afficher les fichiers et dossiers cachés " .

(4) Plus précisément de l'unité logique qui contient le boot système.

(5) Last RecentUse, voir chapitre précédent.

de mémoire lui soient accessibles. Imaginez un jardinier, un tuyau et un quelques bacs à fleurs. Le bac à fleur le plus éloigné est situé à 30 mètres du point d'eau. Mais son tuyau ne fait que 20 mètres. Comment lui sera t-il possible d'arroser ce maudit bac à fleurs ? Tout simplement en amenant ledit bac à portée du tuyau puis en le replaçant à son origine.

En informatique cette technique a pour nom la *translation d'adresse*.

En bref, Windows Server (2000 comme 2003) permet de faire ce que Microsoft appelle AWE pour *Address Windowing Extension*, c'est à dire de traduire des pans entiers de mémoire un peu à la manière des fenêtres coulissantes d'une baie vitrée !

Certes Windows 32 bits n'est toujours pas capable de dépasser la limite de 4 Go, et il faudra toujours consacrer du temps à la gestion de cette pagination, mais c'est sans commune mesure en matière de temps de réponse avec les exécrationnels délais que l'on obtient pour accéder à un disque. Et certaines éditions de Windows permettent de paginer en RAM jusqu'à 64 Go de mémoire vive.

Attention cependant : dépasser la limite de 4 Go n'est possible que pour certaines versions de l'OS Windows Server en combinaison avec certaines versions de SQL Server. Sachant qu'une réinstallation de l'OS et du SGBDR est la seule solution si l'on s'est trompé, il est donc nécessaire de bien anticiper son coup.

## I-B - 64 bits

Le 64 bits permet d'aller encore beaucoup plus loin dans l'adressage RAM. En fait, en théorie, il permet d'adresser 18 446 744 073 709 551 999 octets <sup>(6)</sup> ... En l'occurrence Windows édition 64 bits est encore limité à 512 Go de RAM, mais nous savons qu'en informatique, la réalité rattrape parfois rapidement la fiction !.

**Règle n°1 : adaptez la RAM à la fenêtre de données. Prévoyez la RAM en quantité suffisante pour que les données les plus utilisées soient toujours en mémoire.**

---

(6) Ou encore : 17 179 869 184 Go, 16 777 216 Po (Péta octets), 16 384 Ho (Hexa octets)

## II - Les disques

S'il est un point critique souvent négligé, tant au niveau hard qu'au niveau soft, c'est ce bon vieux disque...

Peu d'informaticien ont conscience qu'entre un disque ordinaire du commerce et un disque optimisé pour un SGBDR l'écart de prix peut aller du simple au quadruple.

Quels sont les facteurs du choix d'un disque ?

Parmi les facteurs à prendre en considération, il y a le temps d'accès moyen, l'algorithme d'accès aux fichiers, la vitesse de rotation, tout cela étant relatif à la technologie des disques (IDE, ATA, SCSI, SATA...).

Souvent le choix s'établit sur le temps d'accès du disque. En fait dans un SGBDR comme MS SQL Server, ce critère à peu d'importance. Le critère le plus important est en fait la vitesse de rotation du disque. Actuellement on trouve - uniquement chez les grands constructeurs : IBM, HP, Dell... - des disques dont la vitesse de rotation est de 18 000 tours par minute (rpm <sup>(7)</sup>).

C'est beaucoup comparé à la vitesse de rotation d'un disque du commerce qui est la plupart du temps de 7 200.

Pourquoi ce critère est-il si important ?

### II-A - La vitesse de rotation

En fait des SGBDR comme SQL Server sont capables de rechercher et réserver sur le disque les meilleurs emplacements de stockage d'un fichier afin de minimiser les temps d'accès, pour peu qu'on leur demande.

Par défaut, lorsque vous créez une base de données, SQL Server crée deux petits fichiers : l'un contient les données, l'autre le journal. Ces deux fichiers ont une taille minimale et chaque fois qu'il sera nécessaire de l'augmenter pour y stocker des informations, alors une opération de croissance de fichier sera entreprise.

Dans ce cas, si l'on observe ce qui se passe durant la vie de la base, on se trouve confronté à un phénomène qui va diminuer sensiblement les performances d'accès. Ce phénomène, c'est la fragmentation physique. Et il n'est pas possible d'y remédier après coup si la base de données doit être exploitée en continue !

Observons concrètement ce qui se passe dans le cas d'un serveur mono disque pour une base de données que l'on va créer avec les options par défaut.

Voici comment petit à petit, le disque se structure :



- En rouge figurent les fichiers de l'OS. Ils se trouvent en tête du disque parce qu'ils ont été installés en premier. En bleu figure « l'exécutable » SQL Server : en fait une collection de DLL et de fichier contenant divers programmes s'interfaçant avec les services SQL Server.
- En jaune figure le début de notre fichier journal de la base, et en vert, le fichier des données.
- Enfin, en gris figure un fichier quelconque, par exemple une sauvegarde.

On constate qu'au fur et à mesure du remplissage de la base de données, le fichier du journal, comme celui de la base de données se fragmente. Plus la base va vivre et plus la fragmentation sera prononcée.

(7) rpm = rotations per minute

Cette fragmentation impose des lectures du fichier des données qui vont, en terme de temps de réponse, du médiocre à l'exécration du fait des nombreux sauts que la tête de lecture doit subir pour lire une continuité de données.

Or MS SQL Server est doté d'un algorithme, qui, lorsque vous demandez, à la création de la base, des fichiers de taille fixe, recherche le meilleur emplacement sur le disque afin de minimiser le trajet des têtes de lecture...

Peu de gens le savent, et c'est particulièrement dommage.

Autrement dit, lors de la création de la base, vous pouvez déjà optimiser la lecture et l'écriture physique des disques. Il suffit pour cela de créer les fichiers de la base, sur un disque vierge ou préalablement formaté, avec une taille suffisante pour y prévoir 3 à 5 années d'exploitation, c'est à dire la durée de vie moyenne d'un serveur...

Si vous faites cela, et en particulier sur un disque neuf ou dédié, vous constaterez deux phénomènes :

- le premier est que ce sont les pistes les plus proches du bord externe du disque qui seront utilisées;
- le second est que si le disque est multi plateaux (et ils le sont presque tous) alors SQL Server choisira ses granules <sup>(8)</sup> sur tous les plateaux.

Un jour, créant ainsi une base de données pour un des plus gros site web d'e-commerce avec cette technique, le DBA, peu au fait de la chose, me montra dans l'outil de d'analyse de fragmentation de Windows combien j'avais tort ! Il voyait que le fichier créé était coupé en quatre parties égales et me lança " vous voyez bien que ça marche pas vot'truc ! Le fichier est fragmenté ! " Alors je lui demandais de me fournir les caractéristiques techniques des disques du serveur et faisant la pari que les disques contenait quatre plateaux. A la lecture de la documentation, je criais " Bingo ! ".

En fait il faut comprendre que les têtes de lecture des différents plateaux sont solidaires. Si l'une lit au bord, l'autre ne peut lire l'extérieure. En groupant les granules de stockage sur les mêmes cylindres virtuels, SQL Server créé des fichiers qui minimisent le déplacement des têtes. Dès lors le temps d'accès qui est un temps moyen devient beaucoup moins proche de la mesure de ce que l'on obtiendra au final avec une telle stratégie. Ainsi, le critère le plus sensible sera donc la vitesse de rotation, car en admettant que la tête ne bouge plus, alors toute la différence se fera entre un disque avec un rpm de 7 200 et un autre de 18 000 (une amélioration de 250% !).

Un internaute a écrit un très bon article sur ce phénomène. Il s'intitule " le disque le plus rapide du monde " <sup>(9)</sup>.

## II-B - Vitesse de rotation et stratégie de fichiers

Nous savons que le fichier contenant les données ira toujours en croissance. Rares sont les bases de données dont le volume des données stockée diminue au fil du temps. De plus en organisant ses tables de la manière la plus intelligente qui soit (clefs cluster sur des colonnes auto incrément ou horodatage), alors les lignes les plus fréquemment accédées (généralement les plus récentes) seront regroupées plutôt vers la fin de la partie du fichier qui contient les données, et cela quelque soit les tables. Dans ce cas, nous bénéficions d'un double avantage : pas de fragmentation physique du fichier, faible fragmentation logique des lignes les plus lues des différentes tables.

**Règle n°2 : choisissez des disques dont la vitesse de rotation est la plus élevée possible.**

## II-C - La fragmentation physique

Car l'autre phénomène désagréable et auquel il est difficile de remédier, c'est la fragmentation physique du fichier. Lorsque l'on créé un fichier, quel qu'il soit, il est ordinairement créé avec une taille minimale, en fait la plus petite taille possible en regard des données à y stocker. Si ce fichier doit ensuite contenir une plus grande quantité d'informations, alors il faudra qu'il demande à l'OS de nouvelles granules à lui adjoindre afin de gérer cette croissance. Dès lors l'ensemble du fichier est constitué de granules éparses acquises au fur et à mesure des opérations de croissance. En effet, il y a fort peu de chance, surtout si l'activité de votre PC est intense, que ces granules soient contiguës. Or

<sup>(9)</sup> [http://www.onversity.net/cgi-bin/progarti/art\\_aff.cgi?Eudo=bgteob&P=a0400](http://www.onversity.net/cgi-bin/progarti/art_aff.cgi?Eudo=bgteob&P=a0400)

cette non contiguïté conduit à lire l'intégralité d'un fichier en sautant d'emplacement en emplacement ce qui dégrade singulièrement les performances de la lecture. Bien évidemment ce phénomène n'est pas grave s'il s'agit d'un fichier Word ! En revanche pour un fichier de base de données il en va tout autrement.

Cette fragmentation est bien connue puisqu'un outil de défragmentation est présent dans la palette des outils systèmes de Windows.

La particularité d'un fichier de données d'une base, est qu'il est en permanence ouvert à l'usage exclusif du serveur SQL. Pourquoi ? Tout simplement parce que le serveur doit pouvoir à tout moment et sans préavis écrire des données. D'où un verrou d'écriture permanent dès que la base est active.

Peut-on défragmenter un fichier de base de données fragmenté ? Oui, assurément. Mais pour cela il faut arrêter l'exploitation de la base, détacher le fichier de données de la base, procéder à la défragmentation au niveau de l'OS. De plus le chaînage interne effectué par SQL Server sur les pages du fichier de la base sera bouleversé. Il y a fort à parier qu'au redémarrage SQL Server mettra un certain temps avant de retrouver ses petits... Bref un travail incompatible avec un service efficace et continue des données !

Le plus simple est donc de créer des fichiers de données d'une taille suffisante à absorber le volume de plusieurs années d'exploitation de la base de données lors de la création de celle-ci.

**Règle n°3 : créez des fichiers de taille fixe.**

## II-D - Répartition des données

Quelques astuces supplémentaires nous permettent de gagner encore plus de temps :

- En prenant un disque très largement surdimensionné on bénéficie encore plus de cet effet vitesse de rotation.
- En dédiant un disque uniquement aux données et un autre au journal vous paralléliserez les lectures et écritures, ce qui fera que l'écriture du journal n'aura pas à attendre la fin d'une lecture des données et vice versa !
- En multipliant les fichiers de données et du journal sur différents disques physiques, vous accélérerez encore les accès, notamment des insertions car SQL Server les réparties de manière équitables dans tous les fichiers.
- En plaçant sur des fichiers distincts les données d'un côté et les index de l'autre, vous pourrez paralléliser mieux encore certaines requêtes.
- En partitionnant les données des plus grosses tables sur différents disques physiques, vous bénéficierez aussi d'un parallélisme de traitement pour chacune des partitions.

Mais veuillez noter que tout cela n'est intéressant que si l'on considère différents disques physiques. En aucun cas, des disques virtuels, partitions d'un même disque physique n'apporteront un tel gain. Ce serait même le contraire : le partitionnement d'un disque physique en multiples disques virtuels, diminue de manière très sensible les performances. Et si l'on opte pour un seul disque sur son serveur, mieux vaut ne pas le partitionner : l'équilibrage du taux d'occupation n'en sera que plus facile.

**Règle n°4 : répartissez les fichiers sur différents disques physiques et les données sur différents fichiers.**

## II-E - Baies de stockage : SAN et NAS

Dans les grosses bases de données, celles dont les volumes dépassent la capacité des disques actuels, on a de plus en plus souvent recours à un dispositif externe de stockage. En cette matière, deux technologies se distinguent : le NAS et le SAN.

Disons le tout de suite, le NAS est l'épouvante de SQL Server. En effet, un NAS ou Network Area Storage suppose un découplage total entre le PC et le stockage. Des données sont envoyées sur le réseau pour que le NAS écrive

ou lise le disque. SQL Server ne s'adresse donc plus à un disque, mais à une boîte noire pour laquelle il ne dispose d'aucune information et qui lui répond quand elle veut, car n'oublions pas qu'un réseau de type Ethernet comme TCP/IP est par nature indéterministe et ne garantit en aucune manière que ce qui est envoyé sur le réseau parviendra un jour à son destinataire... Situation idéale, on en conviendra, pour obtenir les performances les plus lamentables, d'autant que plusieurs serveurs et même certains postes clients peuvent s'adresser au NAS pour de la lecture comme pour de l'écriture ! Heureusement SQL Server ne permet pas un tel fonctionnement de manière native. Mais si vous voulez quand même pouvoir disposer d'une telle fonctionnalité, alors sachez qu'il y a moyen de le faire, à vos risques et périls, en positionnant des flags de trace particuliers <sup>(10)</sup>.

En revanche la technologie SAN est bien plus intéressante : le "Storage Area Network" est en fait un espace de stockage multi disque dédié à un serveur. Dès lors la connexion entre le serveur et le SAN est direct (la plupart du temps sous la forme d'une fibre optique) et exclusif.

Le SAN apporte donc un gros intérêt pour les, VLDB <sup>(11)</sup>. Mais pour être efficace avec SQL Server, un SAN se doit d'être hautement administrable. S'il se comporte comme une boîte noire (partitionnement, agrégats incontrôlés...), on y retrouve les défauts inhérents à la fragmentation. A l'inverse, si ce SAN permet de définir soit même des zones de stockage sur des LUN (Logical Unit Number) et en les créant de manière intelligente, on se trouve dans le meilleur des deux mondes. Pensez donc à éviter les SAN qui ne permettent que le "Target-Level Zoning" et masquent ainsi la structure physique des agrégats de disques.

**Règle n°5 : bannissez les NAS. Préférez les SAN hautement administrables.**

## II-F - Estimer la taille des fichiers

Que vous ayez ou non des difficultés à estimer la taille nécessaire de vos fichiers, le mieux est de toujours leur donner une grande taille, mais surtout de les prévoir en croissance automatique (avec un pas fixe assez large, que je fixe pour ma part à 250 Mo généralement).

Le calcul du volume des données, n'est pas toujours chose aisée. Mais il existe différentes techniques...

- Calcul manuel : calculer la longueur de chaque ligne de chaque table et multiplier par le nombre de ligne probable. Multiplier par 1,25 à 1,60 pour l'indexation et prévoyez une marge (pour des petites bases, je multiplie par deux).
- Calcul automatique : certains outils de modélisation de données comme Power Designer (ex AMC Designor) de Power Soft / Sybase permettent de calculer automatiquement le volume de la base et de chaque table, pourvu qu'on leur fournisse le nombre de lignes de chaque table.
- Calcul estimatif : il suffit de remplir la base de données avec un jeu d'essais assez consistant. En doublant par exemple ce jeu d'essais, on peut estimer le volume à terme de la base à l'aide d'une simple règle de trois.

Ceci pour les fichiers de données. Pour le journal, ce calcul est plus difficile qu'il n'y paraît. Les paramètres qui influent le plus sont : le nombre de transaction par minute, le volume des transactions, la concurrence, la fréquence et le mode des sauvegardes et le mode de récupération de la base. Pour ma part je considère qu'en mode full <sup>(12)</sup>, une base OLTP de taille et d'accès moyen (une centaine d'utilisateurs) un journal de transaction de 30 à 50% de la taille de la base est généralement très suffisant.

**Règle n°6 : estimez la taille de votre base au terme de 3 à 5 années d'exploitation.**

---

<sup>(10)</sup>DBCC TRACEON(1807)

<sup>(11)</sup>Very Large DataBases (très grandes bases de données, c'est à dire celles dépassant le téra octet).

<sup>(12)</sup>SQL Server permet de journaliser avec différents niveaux de précision. Dans le mode "full" le maximum d'information est journalisé afin qu'après une éventuelle reprise après panne, le délai pour que le serveur recouvre un point de cohérence soit le plus court possible. Dans les autres modes, cette journalisation est minimisée, mais le délai de recouvrement d'allonge.

## II-G - Gérez la croissance...

De toutes les manières, il conviendra d'auditer régulièrement la base de données et ses fichiers afin de savoir si vous êtes dans le vrai. Ainsi en laissant des fichiers de grandes tailles avec une croissance automatique par grand blocs fixes (j'utilise 250 Mo comme pas d'incrément pour la croissance de mes fichiers), vous n'aurez pas de surprise. D'autant que les opérations de croissance de fichier, telle la Loi de Murphy, ont le désagrément de se produire généralement aux plus mauvais moments. Croyez vous statistiquement qu'une telle opération sera entreprise au moment ou le serveur est au repos ? Que nenni ! Elle a toutes les chances de se produire dans la transaction la plus longue, celle justement qui insère 32 847 nouvelles lignes dans la plus grosse et la plus scrutée de vos tables, augmentant considérablement le temps global de la transaction, et par là même la durée des verrous, favorisant ainsi la contention !

Dernier avantage de prévoir un fichier de données (et accessoirement un fichier de journal) suffisamment dimensionné : vous éviterez le coup du disque plein, surtout si vous êtes un éditeur de logiciel et que vous ne maîtrisez pas le contenu des disques du serveur de votre client. Ce coup du disque plein se caractérise généralement par le fait qu'au tout début de l'exploitation, les disques du serveur étant quasi vide, la tentation est grande d'y stocker n'importe quoi : les photos de la belle mère lors de la dernière réunion de famille, comme les rapports du service comptabilité que personne ne lit et qui font tous 4 Mo du fait des nombreux graphiques statistiques. Dans le cas ou le disque commencerait à être saturé, la base de données ainsi proprement dimensionnée pourra toujours continuer à vivre.

**Règle n°7 : justifiez votre travail de DBA par une technique de prévention : auditez régulièrement la croissance et gérez là par anticipation.**

## II-H - Répartition des fichiers

Nous avons déjà commencé à parler de la répartition des fichiers et des données sur différents disques. Voyons cela plus en détail et notamment sous l'angle des mécanismes de redondance de type RAID...

Vous le savez sans doute un élément essentiel de la sécurité des données est la redondance. Celle introduite par un système RAID (Originellement *Redundant Arrays of Inexpensive Disks*, puis modifié en *Redundant Arrays of Independant Disks*) est indispensable à une bonne tolérance de panne afin que la base de données puisse continuer son exploitation en cas de perte d'un disque (2eme cause de panne après celle de l'alimentation des serveurs).

Pour autant, les systèmes RAID ne permettent pas toujours les mêmes performances qu'un simple disque <sup>(13)</sup>. Voici un tableau comparant les différents niveaux de RAID à un disque unique :

Niveau de RAID	Lecture	Écriture
0	plus rapide	plus rapide
1	plus rapide	identique
5	un peu moins rapide	moins rapide
10	plus rapide	plus rapide

Le RAID 5 est un compromis entre redondance et coût. Il constitue un bon point d'entrée pour SQL Server. Néanmoins on peut améliorer la chose de la manière suivante :

- placer les fichiers du journal (plus fréquemment écrit) sur un RAID 1
- placer tous les fichiers (données et journal) sur un RAID 1
- placer les fichiers du journal sur un RAID 10 (ou 0+1) et les fichiers de données sur du RAID 5 ou 1.
- placer tous les fichiers (données et journal) sur un RAID 10

Tout ceci doit se faire avec des considérations d'estimation du volume à stocker et du coût du sous système disque.

(13) Par opposition au RAID, un disque unique est appelé SLED pour "Single Large Expensive Disk" !

Si l'on conçoit en outre qu'il est nécessaire pour des raisons de récupération en cas de défaillance grave, de séparer sur des agrégats RAID différents, les fichiers de données des fichiers du journal, alors le sous système disque doit comprendre au moins deux agrégats <sup>(14)</sup> soit au minimum 4 disques.

Bien entendu ce qui est applicable aux fichiers de la base de données est aussi intéressant à appliquer sur les différents fichiers fortement sollicités par le serveur SQL et l'OS.

On pourra par exemple placer sur un agrégat RAID spécifique les éléments suivants :

- fichiers de l'OS et exécutables (SQL Server et autres services associés), hormis fichier de pagination pagefile.sys
- fichier de pagination pagefile.sys
- fichiers de la base de données tempdb

On gagnera encore plus pour les très grandes bases à utiliser plusieurs agrégats pour les fichiers de données voire aussi pour le journal. Dans ce cas il faudra choisir entre performances à la mise à jour (écriture) ou à la lecture.

Si l'on veut de la performance en écriture il suffit de disposer de plusieurs fichiers sur plusieurs agrégats pour les données comme pour le journal sans rien spécifier de particulier au niveau de la base.

En revanche si l'on veut des performances en lecture, il est intéressant de séparer les index des données brutes, voire même d'isoler certaines tables ou de les partitionner <sup>(15)</sup> sur des agrégats spécifiques.

Notez que les modes d'écriture des fichiers de données et des fichiers du journal sont fondamentalement différents. Autant l'écriture dans un fichier de données se faire à un emplacement fréquemment aléatoire et finalement peu fréquent, autant le journal est écrit en fin de fichier de manière séquentielle (write ahead), pour un volume assez fort et assez fréquent. C'est pourquoi il est fréquent de placer les données et le journal sur des modes RAID différents.

Dans tous les cas bannissez l'utilisation de systèmes RAID logiciels. Ces derniers diminuent singulièrement les performances et reportent la criticité globale du système sur le soft. N'hésitez pas à utiliser différents contrôleurs RAID si votre base est grande. Chacun des contrôleurs indépendants apportera une solution de sécurité supplémentaire par sa redondance.

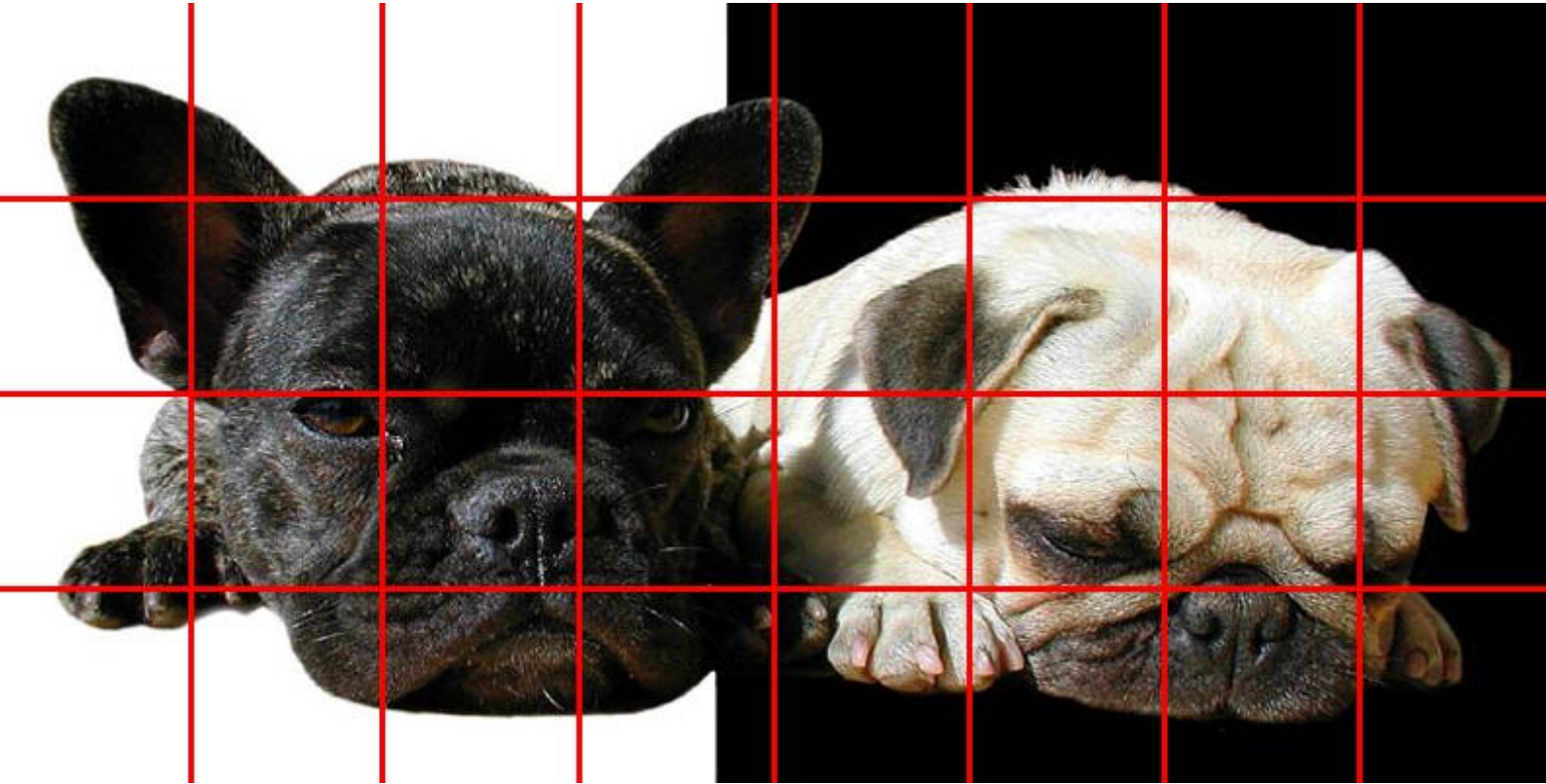
**Règle n°8 : choisissez le bon niveau de RAID et répartissez les fichiers de données, comme ceux du journal sur différents agrégats. Faites de même avec les exécutables, le fichier de pagination de la mémoire virtuelle (pagefile.sys) et les fichiers de la base tempdb.**

## II-I - Taux d'occupation

Un disque, lorsqu'il commence à être rempli, ressemble à un jeu de taquin. Ce jeu mathématique inventé par Sam Loyd en 1873 consiste à découper une image en petits carrés, les mélanger et en retirer un. Pour gagner il faut recomposer l'image en faisant glisser les petits carrés verticalement ou horizontalement.

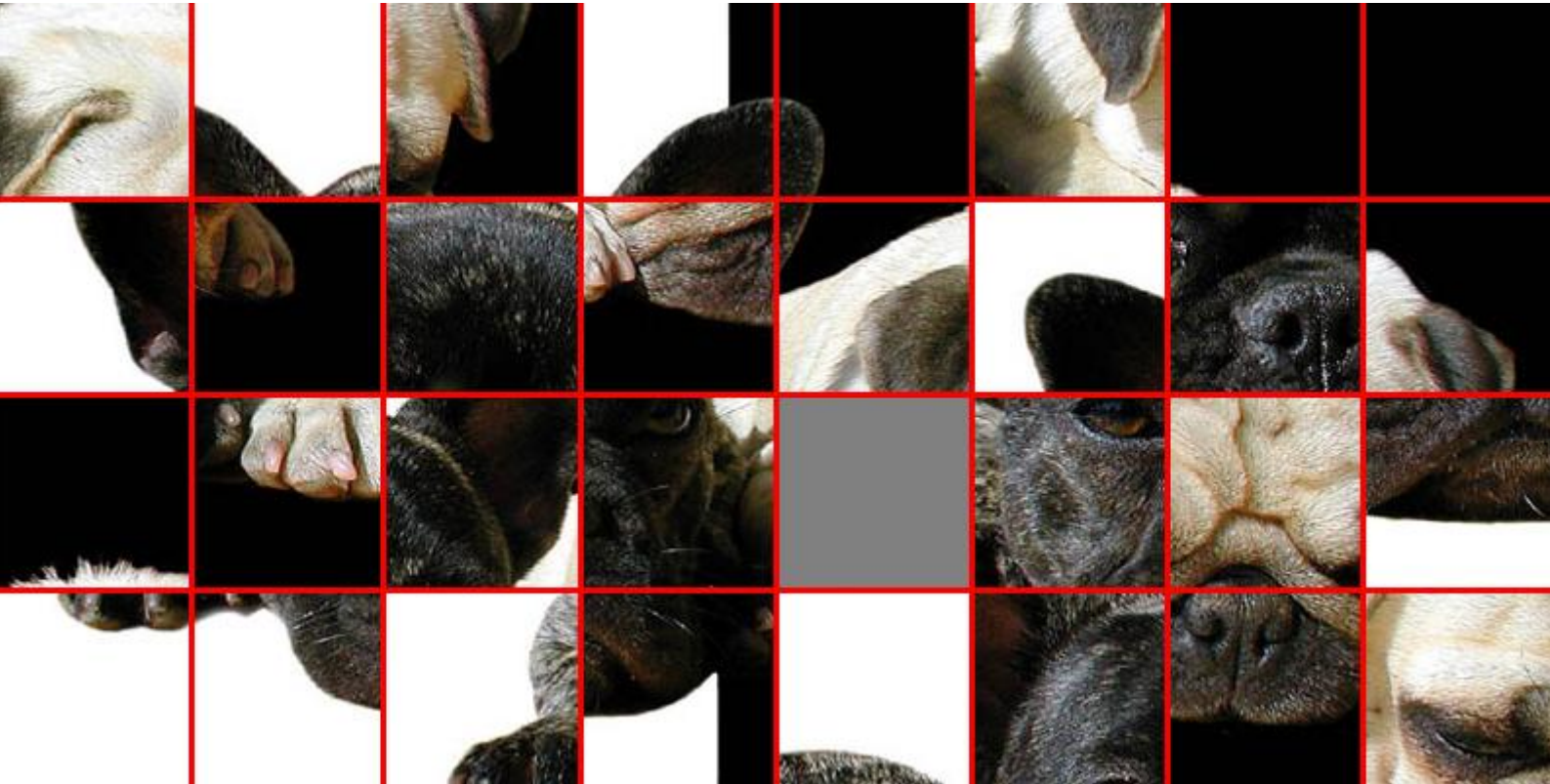
<sup>(14)</sup>Un agrégat RAID est un ensemble de disques mis en RAID afin de ne plus former qu'une seule unité logique correspondant à la capacité maximale de l'ensemble des disques mis en RAID, par rapport au niveau de RAID choisi. Par exemple pour un RAID 1 avec 2 disques de 300 GO, le résultat est une unité logique de stockage de 300 GO. pour un RAID 5 avec 3 disques de 300 GO, le résultat est une unité logique de stockage de 600 GO

<sup>(15)</sup>SQL Server 2005 dispose d'un nouveau système de partitionnement des données, facile à mettre en oeuvre et transparent pour le développeur. Nous en reparlerons.



*figure 1 - Le jeu du taquin constitue une bonne analogie d'un disque saturé : l'optimum est l'image proprement composé.*

En fait un disque dont le taux de remplissage voisine la saturation est un gigantesque taquin dans lequel une seule granule de stockage est libre. Au moment d'enregistrer un fichier que vous venez de modifier, il faudra au système opérer l'enregistrement du fichier granule par granule. L'opération va devenir extrêmement longue et mobiliser tous les accès disques.



*figure 2 - lorsque l'on mélange les différentes parties de l'image, sa recomposition est extrêmement longue à opérer si l'on a qu'un seul espace libre disponible. Bien évidemment, plus on a d'espaces libres, plus rapide sera la recomposition. De manière analogue, dans un disque dans lequel il reste peu d'espaces disponibles, toute manipulation d'écriture de fichier va prendre un temps important car il faut passer par des zones de stockage qui servent de tampons pour les données en cours de modification.*

C'est pourquoi on recommande de ne pas dépasser un certain taux d'occupation d'un disque. En général il faut prévenir à 70 % et guérir à 80. Au delà de 90% de taux d'occupation, l'allongement des temps de réponse commencent à se faire sentir !

Bien entendu la survenance d'un tel seuil est facilitée si les disques ne sont pas partitionnés.

**Règle n°9 : surveillez les taux d'occupation des différents disques et tentez de ne jamais dépasser 70%.**

## II-J - Base tempdb en RAM

Une solution ultime pour obtenir de bonnes performances si votre base tempdb est fortement saturée, consiste à la placer en mémoire. Différents fabricants proposent ce que l'on appelle des "Ram disque". Autrement dit une mémoire vive qui se substitue à un disque. Pas de panique, les fabricants ont bien prévu les filets de sécurité nécessaires. Ces dispositifs sont généralement vendus avec une batterie et un disque de secours qui en cas de panne de courant s'assure que les données en mémoire sont flashées sur le disque. Avec un tel système combiné à une forte et judicieuse utilisation de la tempdb, les résultats peuvent être spectaculaires !

Jetez donc un coup d'oeil aux produits de Texas Memory Systems par exemple...

## II-K - Algorithme d'accès aux fichiers

Enfin, l'un des points maintenant le mieux maîtrisé dans la technologie des disques est l'algorithme d'accès au fichier. Dans un mauvais disque, les demandes d'accès (écriture ou lecture) sont effectuées dans l'ordre de la demande (FCFS : First Come First Serve). Dans un bon disque ils peuvent être effectués dans l'ordre du plus près servit

(Shortest Seek Time First) mais cet algorithme peut conduire à la famine du système. En général il est amendé afin de ne jamais conduire à une telle situation. Mieux le parcours en "scan" aussi appelé "lift", car c'est l'algorithme de parcours des ascenseurs intelligents, permet de parcourir le disque dans toute la plage des pistes et de servir au passage les demandes, sans tenir compte de l'ordre de la demande. Enfin des améliorations notables ont été apportées à cet algorithme tel que le C-Scan ou le "look".

### III - les processeurs

Pour une base de données chargée, il est nécessaire de disposer de plusieurs processeurs. Mais qu'est-ce qu'une base chargée ? C'est soit un grand nombre d'utilisateurs potentiellement simultanés (pôle de saisie, site web marchand...), soit des requêtes portant sur des masses importantes d'informations (calculs d'agrégats par exemple).

Dans ce cas le nombre de processeurs du serveur peut être important.

De manière générale je considère qu'il faut un processeur pour 250 à 300 process simultanés. Ou encore qu'il faut plusieurs processeur dès que la durée de certaines requêtes avoisine la minute.

Attention cependant à l'hyperthreading avec des processeurs Intel. En effet Slava Ocks, un des développeurs du team SQL de Microsoft a mis en lumière un phénomène dont certains clients de SQL Server et de Citrix ont été victimes... En effet dans certaines conditions : forte charge, multiples petit process... le cache du processeur est encombré par des processus en tâche de fond qui semblent ne plus vouloir partir. Et les performances sont en large baisse. Comme il est difficile de mettre en lumière ce phénomène pour savoir si vous en êtes victimes, les spécialistes considèrent qu'il est préférable de désactiver l'hyperthreading.

Autrement dit vous avez deux moyens : ne pas utiliser les processeurs Intel en mode hyperthreading (prendre par exemple des processeurs Opteron ou Athlon d'AMD) ou bien doubler les "vrais" processeurs Intel de votre machine.

Dans tous les cas le passage du 32 au 64 bits, à caractéristiques physique équivalentes pour les CPU, diminue d'environ 30 à 40% les temps de réponse des requêtes. En effet, 64 bits, c'est une chaîne de caractères traitée en deux fois moins de temps qu'en 32 bits, tout simplement !

***Règles n°10 : utilisez plusieurs processeurs physiques. Évitez le multithreading Intel. Préférez le 64 bits.***

## IV - les réglages au niveau de SQL Server

Mis à part la mémoire haute dans le cas d'un OS 32 bits <sup>(16)</sup>, l'ensemble des réglages pour la mémoire, les disques et les processeurs, sont à faire sur le serveur SQL.

Mais il y a deux catégories de réglages :

- les réglages stricts, préventif destinés à indiquer à SQL Server dans quelle mode il tourne,
- les réglages fins, curatifs, destinés à améliorer le fonctionnement du serveur en fonction de la charge.

### IV-A - réglages de la mémoire

La procédure sp\_configure permet de régler les principaux paramètres pour piloter la mémoire.

Réglages stricts :

AWE enabled	doit être activé si vous utilisez un système 32 bits avec plus de 4 Go de RAM
-------------	---

Réglages fins :

index create memory	contrôle la quantité de mémoire associée au tri des index.
max et min server memory	attribuez au minimum entre 4 et 16 Mo de mémoire à un SQL Server 32 bits. Pour le max, laissez une valeur ouverte.
min memory per query	diminuez à 512 Ko ou augmentez si besoin est.
set working set size	verrouille la mémoire utilisée par SQL Server. A n'utiliser que pour un serveur dédié. (plus nécessaire dans la version 2005)

Autres réglages entraînant de la consommation de mémoire :

max worker thread	nombre de thread dans le pool de travail
open objects	limite le nombre d'objets ouverts simultanément

### IV-B - réglages au niveau des disques

Il n'y a pas de réglage disponible pour la gestion des disques. Souvenez vous simplement qu'il vaut mieux créer une base avec des fichiers de grande taille, si possible fixe, répartir ses fichiers sur différents agrégats de disques et surveiller leur taux de remplissage. Vous éviterez ainsi la fragmentation physique des fichiers et ne serez pas pénalisés en cas de fort flux de mise à jour.

N'oubliez pas non plus d'auditer vos disques afin d'en limiter le taux d'occupation.

Bien que l'installation de SQL Server provoque l'installation de toutes les bases dans un même répertoire, vous pouvez modifier l'emplacement de la base tempdb à l'aide de la commande ALTER TABLE.

<sup>(16)</sup>Pour 4 Go de RAM basse, ajoutez le switch /3GB dans une ligne du fichier boot.ini. Pour plus de 4 Go vous devez ajouter en sus le switch /PAE. Au delà de 16 GO, ne pas activer le switch /3GB.

Sachez que, lorsque vous créez des index (CREATE INDEX), vous pouvez préciser à l'aide de l'option "*sort in tempdb*" que le tri pour la création dudit index se fera dans la base de données des objets temporaires.

Enfin, notez qu'il est difficile et coûteux de mouvoir les fichiers d'une base en exploitation, comme de déplacer certaines structures de la base (tables, index...) <sup>(17)</sup>. Mieux vaut donc avoir prévu la répartition de ces éléments lors de la création de la base, par une étude sérieuse des volumétries en jeu.

## IV-C - réglages au niveau des processeurs

La procédure `sp_configure` permet de régler les principaux paramètres pour piloter la gestion des processeurs. De plus le tag `MAXDOP` permet de limiter l'usage de plusieurs processeurs.

Nous avons vu que SQL Server utilisait peu l'OS puisqu'il utilise son propre OS "SQL OS". Nous en avons déduit que contrairement à d'autres applications (IIS par exemple) il était parfaitement logique d'assigner 3/4 des ressources RAM à SQL Server et 1/4 à l'OS (swith /3GB dans le fichier `boot.ini`).

Il en est de même pour les processeurs. SQL Server installé sur une machine dédiée n'a besoin que de 75% des processeurs physiques.

Ainsi à partir du quadri processeur, il est intéressant de dédier à SQL Server certains processeurs pour laisser les autres à disposition de l'OS.

La répartition de fait à l'aide du paramètre "`affinity mask`" de la procédure stockée `sp_configure`.

Un masque d'affinité est une combinaison de bits qui indique par un 1 si le processeur de rang `n` est utilisé par SQL Server, sinon à 0 c'est l'OS qui l'utilisera. Ainsi pour indiquer à SQL Server que les processeurs 0, 1 et 2 lui sont réservés (et que par conséquent le processeur 3 sur un système quadri est utilisé par l'OS) alors il faut donner au paramètre "`affinity mask`" la valeur 7 obtenue par la somme de  $2^0 + 2^1 + 2^2$ .

Attention : dédiez toujours les processeurs de poids faible (0, 1, 2...) à SQL Server et ceux de poids forts (... `n-2`, `n-1`, `n`) à l'OS. En effet le processeur de poids le plus fort est la plupart du temps utilisé par les cartes réseaux. Or c'est la seule action que SQL dédie à l'OS !

Même si vous avez dédié à SQL Server 6 processeurs sur 8, vous pouvez exiger qu'aucune requête ne prenne plus de 4 processeurs par exemple. Cela fluidifiera les petites requêtes au détriment du temps de réponse des grosses. C'est possible à l'aide du paramètre "`max degree of parallelism`" modifiable par la procédure stockée `sp_configure`.

Vous pouvez aussi définir le coût <sup>(18)</sup> minimum à partir duquel le parallélisme va entrer en jeu. C'est le paramètre "`cost threshold for parallelism`" de la même procédure. N'oubliez pas qu'un traitement parallèle engendre un travail plus important qu'en série. Il y a donc gain sur la durée, mais perte sur la consommation des ressources.

Enfin, le tag `MAXDOP` peut être utilisé dans une requête pour lui indiquer de ne pas utiliser plus de `n` processeur pour être traitée. Cela se fait dans la clause `OPTION` de la requête, clause spécifique à SQL Server.

---

<sup>(17)</sup>C'est possible, mais pour cela il faut interrompre le service de la base de données en tout ou partie (détachement des fichiers, reconstruction des index...)

<sup>(18)</sup>Pour toutes les requêtes, les coûts sont estimés en seconde d'exécution.

## V - Conclusion

Avant de régler finement votre système; il faut l'auditer régulièrement, notamment à l'aide du moniteur de performances. De là, vous obtiendrez une masse d'informations que l'observation vous permettra d'analyser. De cette analyse découleront les paramètres à faire évoluer.

Mais, comme tout bon artisan, il convient de n'ajuster qu'un seul paramètre à la fois et constater si son effet est positif. C'est pourquoi l'optimisation n'est pas une science absolue. Si elle l'était, Microsoft aurait fourbit SQL de quelques dizaines d'assistants. Or ce n'est pas le cas. Même l'assistant de paramétrage d'index a été retiré dans la version 2005 devant sa médiocre qualité, au profit d'un outil plus modeste : "database tuning advisor" autrement dit "indicateur" de possibilité d'optimisation...

### Références webgraphiques :

#### Le disque dur le plus rapide du monde :

- [http://www.onversity.net/cgi-bin/progarti/art\\_aff.cgi?Eudo=bgteob&P=a0400](http://www.onversity.net/cgi-bin/progarti/art_aff.cgi?Eudo=bgteob&P=a0400)

#### Technologie des disques durs :

- <http://pagnotte.ftp-developpez.com/chapitre6.pdf>
- <http://www.enseirb.fr/~pelegrin/enseignement/enseirb/systeme/cours/c.ps>
- <http://www.dil.univ-mrs.fr/~massat/ens/systeme/transparentes-cours-sgf.pdf>
- [http://www-id.imag.fr/Laboratoire/Membres/Marangozova-Martin\\_Vania/teaching/cse\\_m1/etudiants/C9.pdf](http://www-id.imag.fr/Laboratoire/Membres/Marangozova-Martin_Vania/teaching/cse_m1/etudiants/C9.pdf)

#### Disques, fichiers de bases de données et RAM :

- [http://www.site.uottawa.ca/~kiringa/courses05/csi3717/ch9\\_disks\\_files\\_csi3717-05.ppt](http://www.site.uottawa.ca/~kiringa/courses05/csi3717/ch9_disks_files_csi3717-05.ppt)

#### Systèmes RAID :

- <http://www.cs.cmu.edu/~garth/RAIDpaper/Patterson88.pdf>
- [http://fr.wikipedia.org/wiki/RAID\\_\(informatique\)](http://fr.wikipedia.org/wiki/RAID_(informatique))
- <http://www.commentcamarche.net/protect/raid.php3>
- <http://www.ibeast.com/content/tools/RaidCalc/RaidCalc.asp>

#### Les SAN :

- <http://www.yafla.com/dennisforbes/SAN-NAS-and-iSCSI-SQL-Server-/SAN-NAS-and-iSCSI-SQL-Server-.html>
- <http://www.ossir.org/resist/supports/cr/20060320/SecuriteSAN.pdf>
- <http://www.comptechdoc.org/docs/craig/sanzoning/>

#### Ram Disk :

- [http://www.superssd.com/products\\_sub.htm](http://www.superssd.com/products_sub.htm)
- <http://www.amtsoftware.com/Ramdisk-Plus/docs/SQL-Performance>

#### Le bug de l'hyperthreading Intel :

- <http://www.zdnet.fr/actualites/informatique/0,39040745,39289582,00.htm?xtor=1>
- <http://blogs.msdn.com/slavao/archive/2005/11/12/492119.aspx>

*Photographie et dessin de l'auteur (Tjader et Ubik)*