La vérification d'intégrité et les problématiques liés aux VLDB

par David BARBARIN (http://mikedavem.developpez.com/) (Blog)

Date de publication : 18 juillet 2009

Dernière mise à jour :

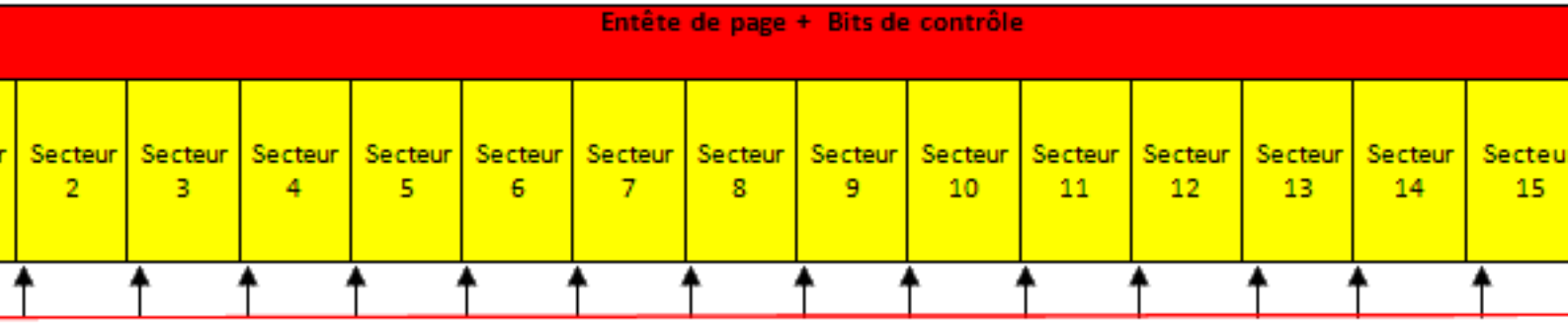
I - Introduction

Quel DBA ignore ce qu'est la vérification d'intégrité ? Ce processus est important car il permet de prévenir une anomalie d'une base de données. Cette opération, en apparence simple à intégrer dans un plan de maintenance, l'est beaucoup moins lorsqu'il s'agit de VLDB (Very Large DataBases).

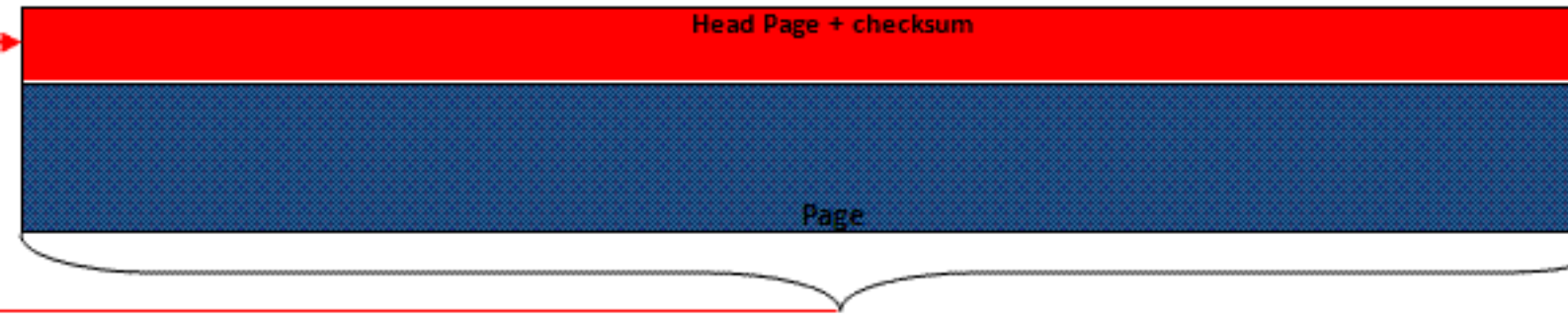
II - Détection des problèmes d'intégrité

SQL Server peut vérifier l'intégrité des données de 2 façons : La 1ère concerne l'option de bases de données PAGE_VERIFY qui indique à SQL Server le procédé qu'il doit utiliser pour détecter les pages endommagées. Depuis la version 2005 de SQL Server il est possible de paramétrer l'option PAGE_VERIFY avec la valeur ***TORN_PAGE_DETECTION*** ou ***CHECKSUM***.

Si l'option *TORN_PAGE_DETECTION* est activée, un bit spécifique à chaque secteur de 512 octets d'une page de 8Ko (soit 16 secteurs) est inscrit dans l'en-tête de la page. Lorsqu'une page est lue depuis le disque, ces bits sont comparés aux informations réelles de chaque secteur. Si les informations diffèrent, alors il y a corruption de la page.



Si l'option *CHECKSUM* est activée, une somme de contrôle pour la page entière est calculée puis stockée dans son entête. Lorsqu'une page est lue depuis le disque, une somme de contrôle est recalculée et comparée à celle écrite dans l'en-tête de la page. Si celles-ci ne correspondent pas, la page est corrompue.



La détection d'erreur par l'option *CHECKSUM* est beaucoup plus précise que celle utilisée par l'option *TORN_PAGE_DETECTION* mais elle monopolise quelques ressources CPU de plus.

A partir de la version 2005, lors SQL Server détecte une page corrompue, il génère une erreur. La commande qui voulait accéder à la page est annulée et une entrée est créée dans la table *suspect_pages* dans la base de données *msdb*.

Avec cette 1ère méthode il faut attendre que la page soit lue par le moteur de bases de données pour détecter une corruption. Une approche beaucoup plus proactive est de lancer la commande DBCC CHECKDB pour forcer la lecture de toutes les pages d'une base de données.

III - Fonctionnement de la commande DBCC

La commande DBCC CHECKDB vérifie l'intégrité physique et logique d'une base de données et s'effectue en 3 phases bien distinctes.

1ère phase : CHECKALLOC : Une vérification de la cohérence d'allocation d'espace disque est effectuée. Les informations contenues dans les pages IAM, GAM, SGAM sont vérifiées pour chaque Extent de la base de données.

2ème phase : CHECKTABLE : Une vérification de l'intégrité des pages et des structures pour chaque table, vue indexée et index d'un point vue logique et physique est effectuée. Dans le cas des vues indexées chaque ligne de données est régénérée en se basant sur sa requête de création puis comparée aux données réelles de cette même vue.

3ème phase : CHECKCATALOG : Un contrôle de la cohérence des informations entres les différentes tables de métadonnées système est effectuée.

IV - Facteurs affectant les performances de la commande DBCC CHECKDB

Afin d'optimiser la phase de vérification des bases de données, il est utile de connaître les facteurs affectant sa vitesse. Celle-ci varie en fonction de différents facteurs physiques et logiques, dont les principaux sont :

La taille de la base de données : Un facteur important. En effet plus la taille d'une base de données est importante, plus le processus de vérification sera long car la totalité des pages d'allocation de la base doit être lue.

La complexité du schéma de bases de données : Le schéma est également un facteur important. Plus le schéma de la base de données est complexe, plus la vérification d'intégrité prendra de temps.

Les ressources matérielles : Les lectures effectuées par le processus de vérification se font séquentiellement et dans l'ordre physique des pages de la base de données. Le processus de vérification, grâce à son algorithme de fonctionnement, essaie autant que possible d'optimiser et de réduire les éventuels déplacements aléatoires des têtes de lecture des disques de données. Si une concurrence d'accès disque apparaît, celle-ci peut interférer et réduire la bande passante utilisée par le processus de vérification et le temps d'exécution en est alors augmenté. C'est une des raisons pour lesquelles il est conseillé de lancer un tel processus dans les périodes creuses d'activité dans la mesure du possible. On peut soulever une deuxième explication : Lorsque le processus de vérification est lancé, il utilise une capture instantanée interne qui se situe sur le même volume disque que la base de données elle-même afin d'éviter une concurrence d'accès aux données de la base. Il faut savoir qu'une capture instantanée est vide à sa création mais se remplit à chaque mise à jour des données de la base. Cela signifie que pour chaque mise à jour il y aura un déplacement des têtes de lecture de disque qui perturbera inévitablement le processus de vérification. En d'autres termes, le processus risque d'être d'autant plus long que les mises à jour lancées au cours du processus de vérification d'intégrité des bases sont nombreuses.

Les disques et l'utilisation de la base TEMPDB : Le processus de vérification utilise principalement des entrées / sorties disques. Sa vitesse d'exécution est donc dépendante de la bande passante du sous système disque. La base

de données tempdb peut être également utilisée par le processus de vérification en particulier dans le cas des VLDB où la quantité de mémoire allouée est parfois insuffisante. En effet le processus de vérification nécessite le stockage de certaines informations internes au cours de son processus. Il utilise pour cela un espace mémoire de travail mais dans le cas des VLDB cet espace mémoire est très souvent insuffisant et le processus stockera ses informations dans la base Tempdb prévue à cet effet. Dans ce deuxième cas, la vitesse d'exécution du processus de vérification dépend aussi du sous système disque des fichiers de bases tempdb.

Les options d'exécution de la commande DBCC CHECKDB : En fonction des options d'exécution de la commande DBCC CHECKDB, le processus peut être plus ou moins long car la vérification s'effectuera partiellement ou complètement sur la base de données.

V - Les problématiques liées aux VLDB

Dans la plupart des situations, les options de vérification d'intégrité des bases proposées dans les plans de maintenance suffisent et satisfont aux besoins des entreprises. Pourtant, il n'en va pas de même lorsqu'il s'agit des VLDB où le temps d'exécution du processus de vérification d'intégrité devient très important et peut rapidement atteindre les limites imposées par une fenêtre de maintenance. Il existe une multitude de solutions pour essayer de pallier ce problème :

V-A - Utiliser les options de la commande DBCC CHECKDB

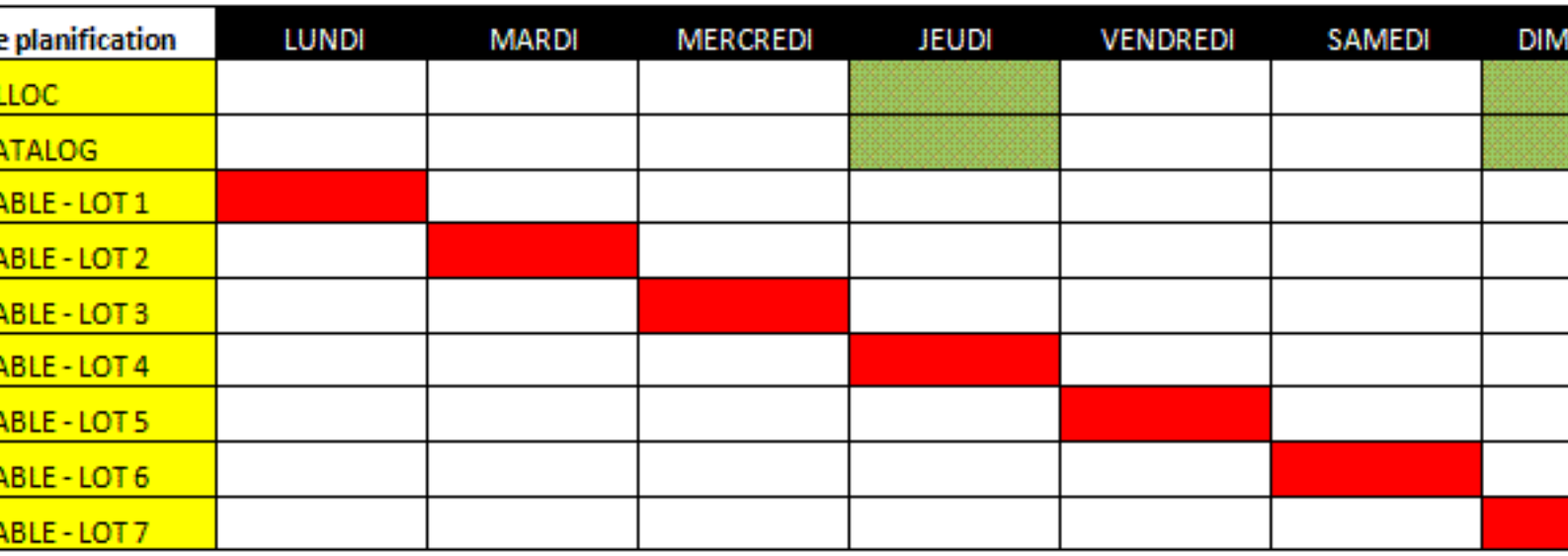
La commande DBCC CHECKDB comporte plusieurs options intéressantes qui peuvent considérablement réduire les temps d'exécution de bases de données.

Option **PHYSICAL_ONLY**: Cette option supprime l'étape de vérification logique de la base de données. Seules les structures physiques des pages et des en-têtes ainsi que celles des arbres B-Tree sont vérifiées afin de prévenir tout problème de corruption matérielle (TORN pages, CHECKSUM Failure ?). La suppression de l'étape vérification logique des bases est un bon moyen de diminuer considérablement le temps d'exécution et minimise les ressources monopolisées par le processus.

Option **CHECKFILEGROUP** : Cette option peut être utile si la structure physique de la base de données le permet. Celle-ci doit suivre un schéma de répartition des tables dans plusieurs FILEGROUPS. L'option CHECKFILEGROUP, dans ce cas, peut jouer pleinement son rôle et permet de vérifier l'intégrité de toutes les tables d'une base de données pour un FILEGROUP déterminé. On peut, grâce à cette option, répartir et planifier la vérification de chaque groupe de fichiers sur plusieurs jours de la semaine afin de réduire le temps et la charge d'exécution du processus de vérification de la base.

V-B - Personnaliser ses propres partitions de vérification

Une autre solution consiste à partitionner la base de données en lots et planifier l'exécution des 3 phases de la commande DBCC CHECKDB (DCBCC CHECKCATALOG, DBCC CHECKALLOC et DBCC CHECKTABLE) sur ces lots. Pour cela, il faut au préalable repérer les tables les plus importantes selon le nombre de pages, puis les grouper pour former des lots de taille équivalente et planifier pour chaque lot l'exécution les différentes phases de vérification d'intégrité. Le but est de minimiser la charge de travail du processus en répartissant les phases de vérification sur une semaine.



La vérification du catalogue des métadonnées et de la cohérence d'allocation des pages s'effectuera deux fois par semaine. La vérification d'intégrité des pages et des structures physiques se fera tout au long de la semaine et sur chaque lot distinct. On minimise ainsi l'impact et la durée du processus de vérification. L'avantage de cette solution est de réduire considérablement le temps d'exécution de la vérification d'intégrité ainsi que les ressources monopolisées. Elle convient bien dans un scénario où la fenêtre de maintenance est extrêmement réduite où l'on doit effectuer d'autres opérations par la suite telle qu'une ré-indexation ou une sauvegarde de bases de données. Cette méthode présente néanmoins quelques inconvénients : une capture instantanée est créée à chaque fois qu'une opération DBCC est réalisée et, dans ce cas précis, pour l'opération CHECKTABLE. Si l'activité de mise à jour est importante à ce moment-là, comme nous l'avons vu plus haut, le processus de vérification risque d'être perturbé et le temps d'exécution allongé. Il faudra alors adapter le nombre de lots en fonction de la durée imposée dans la fenêtre de maintenance. Elle nécessite également plus de travail de préparation pour constituer les lots de table.

V-C - Utiliser un système dédié

Il s'agit dans ce cas de déporter l'opération de vérification des bases sur un système dédié. Il suffit de remonter une sauvegarde et de lancer une opération de vérification sur les bases de données restaurées. Comme on peut le deviner, vous l'aurez devinez, le principal avantage est que les ressources ne sont plus monopolisées sur l'environnement de production. Mais il y a tout de même quelques inconvénients majeurs. D'une part cela exige d'avoir un système dédié et d'autre part de posséder une place disque aussi importante qu'en environnement de production

V-D - Optimiser tempdb

Comme nous l'avons vu, la bande passante offerte par le sous système disque est en relation directe avec la vitesse d'exécution du processus de vérification des bases de données. Il faut choisir un sous système de disque rapide pour tempdb (RAID 1 + 0 par exemple) car le processus de vérification a de très grandes chances de l'utiliser si votre base est une VLDB. Il faut également créer autant de fichiers que l'on a de processeur pour profiter pleinement de l'algorithme de répartition des données utilisé par SQL Server.

VI - De la théorie à la pratique

VI-A - Le contexte

Un des problèmes que rencontrait le client était la durée d'exécution d'un de ses plans de maintenance et qui contenait une tâche de vérification d'intégrité sur les bases de données de son application principal. Le temps d'exécution global que prenait cette tâche était de 8H, ce qui correspondait à durée maximum permise par la fenêtre de maintenance. Il ne restait donc plus assez de temps pour les autres étapes du plan comme la sauvegarde et les différentes tâches d'optimisation. Pour l'exemple nous nous focaliserons sur une base de données en particulier ayant les caractéristiques suivantes : - Taille de la base de données : **210 Go** - Temps d'exécution de la vérification d'intégrité : **2H 15min**

VI-B - Résultats

Méthode appliquée	Temps d'exécution / jo
ar défaut du plan de maintenance (DBCC CHECKDB sans option)	2H30
entation de l'option PHYSICAL_ONLY (DBCC CHECKDB ... WITH PHYSICAL_ONYL)	1h28
entation partitionnement personnalisé avec lots (DBCC CHECKTABLE + CHECKALLOC + CATALOG) lus important : 100 Go	40 min pour le lot le plus imp

Comme on le constate dans le meilleur des cas, le gain de temps est de 73% pour cette seule base de données avec l'application du partitionnement personnalisé.

VII - Conclusion

La vérification d'intégrité des bases de données dans le cas des VLDB n'est pas simple à mettre en oeuvrre. Le choix d'une méthode doit s'effectuer en adéquation avec l'architecture, les contraintes techniques et fonctionnelles de l'environnement dans lequel l'on se trouve. Il faut garder également à l'esprit qu'une méthode mise en place est amenée à changer en fonction des changements que subissent les bases de données. Il est donc important d'implémenter en parallèle un système de surveillance permettant d'anticiper et d'indiquer vers quelles méthodes il faudra se tourner. Ceci fera l'objet d'un autre article...

VIII - Références

VIII-A - Webographie

- **Msdn - CHECKDB**
- **Msdn - Optimisation des performances CHECKDB**
- **SQL Server Storage Engine**
- **SQLServerPedia**
- **Rob s SQL Server Blog**

VIII-B - Bibliographie

Inside SQL Server 2005 - The Storage Engine

IX - Remerciements