

Installation de dbcc checkstorage



par Fabien Celaia ([Fadace](#))

Date de publication : 2.3.2006

Dernière mise à jour : 8.11.2006

Configuration du DBCC CHECKSTORAGE utilisatble sur Sybase ASE.

1 - Préambule.....	3
2 - Installation de dbccdb.....	4
3 - Configuration de checkstorage.....	10
4 - Utilisation de checkstorage.....	11

1 - Préambule

Sybase Adaptive Server offre un éventail d'utilitaires permettant de tester l'intégrité physique d'une base de données: les fameux dbcc. Il existe une kyrielle de dbcc:

- checkalloc
- checkdb
- checktable
- checkverify
- indexalloc
- rebuild_text
- reindex
- tablealloc

... sans compter ceux, nombreux et non documentés. Tous souffrent de la même carence:

- ils sont en mesure d'analyser une base en ligne
- en ligne, ils ne peuvent corriger les erreurs
- en multi-utilisateurs, ils relèvent parfois des erreurs qui n'en sont pas (spurious error) puisque l'état d'un bloc de donnée peut être modifié pendant l'analyse
- l'analyse des rapports générés est fastidieuse et compliquée

Depuis sa version 11.5, Sybase Adaptive Server fournit un puissant outil permettant de contrôler rapidement et en ligne l'intégrité des bases de données. Il répond au doux nom de dbcc checkstorage.

Il s'oppose aux autres DBCC car il se repose sur un référentiel qui lui permet

- de gérer le cas des erreurs inconsistantes
- d'offrir des rapports de meilleure facture, mis en forme par des procédures stockées idoines
- de ne pas parasiter les performances de la base de production analysée.

Ces avantages ont malheureusement un prix : le processus d'installation et de configuration de la base référentielle dbccdb est lourds et fastidieux. Nombre de DBA Sybase n'utilisent donc pas cet outil puissant par méconnaissance ou en étant rebuté par la phase d'installation et de configuration.

La procédure qui suit a comme but de vous guider dans l'installation de la dbccdb et de vous offrir des outils qui vous permettront de la gérer aisément.

2 - Installation de dbccdb

Première chose à faire, créer une base dbccdb. Afin que celle-ci impacte le moins possible sur les autres devices, je vous conseille de la créer sur 2 devices bien spécifiques: La commande suivante vous permet de déterminer quels sont les volumétries attendues pour dbccdb.

```
use master
exec sp_plan_dbccdb
```

Riche de ces valeurs créez ensuite votre base:

```
create database dbccdb on dbccdb_data01=100M log on dbccdb_log01=10M
```

Exécutez ensuite le script suivant

- Sous Windows: %SYBASE%\%SYBASE_ASE%\scripts\installdbccdb
- Sous Unix: \$SYBASE/\$SYBASE_ASE/scripts/installdbccdb

Allez ensuite dans la base dbccdb, créez la procédures stockée sp_dbcc_init. Lors de son premier appel, cette procédure configure toutes les DBs afin qu'elles puissent utiliser dbcc checkstorage Lors des appels suivants, elle reconfigure les paramètres pour qu'ils répondent aux critères de sp_dbcc_evaluatedb

```
create procedure sp_dbcc_init (@cache varchar(30) = NULL)
/*
 * Auteur   : Fabien Celaia
 * Date     : 23.1.2002
 * Certif.  : ASE 12.5 (sinon quelques changements de paramètres nécessitent un reboot)
 * Préreq.  : 1) dbccdb est créée
 *          : 2) Script installdbccdb déjà exécuté
 * Desc.    : 1) Premier appel : configure toutes les DBs afin qu'elles puissent utiliser dbcc checkstorage
 *          : 2) Ensuite : reconfigure les paramètres pour qu'ils répondent aux critères de sp_dbcc_evaluatedb
 * Syntaxe  : exec dbccdb..sp_dbcc_init
 * Param    : Entrée : -
 *          : Sortie : -
 */
as
begin
declare @max_text int
declare @max_wt int
declare @max_cache int
declare @dev_name varchar(30)
declare @dbid smallint, @dbid1 smallint, @dbid2 smallint
declare @scanws_size int
declare @textws_size int
declare @cache_size int
declare @wt_count smallint
declare @dbname varchar(30)
declare @dbcc_dbname varchar(30)
declare @size float
declare @msg varchar(1024)
declare @min_wssize int /* taille maximale du workspace */
declare @new_min_wssize int /* contrôle la taille minimum du workspace */
declare @min_cachesize int /* Taille minimale du cache */
declare @err int
/*
** ces variables sont utiles au formatage de l'output
*/
declare @str1 varchar(30)
declare @twc varchar(30)
declare @sws varchar(30)
/*
** Empêche l'utilisation de cette proc dans une transaction puisque des tables temporaires sont créées
*/
if @@trancount > 0
begin
```

```

        /*
        ** 17260, "Ne pas utiliser %! dans une transaction ouverte."
        */
        raiserror 17260, "sp_plan_dbccdb"
        return (1)
end
else
begin
    set chained off
end

set transaction isolation level 1
set nocount on

/*
** Création de la table temporaire stockant les résultats et déclaration d'un curseur
*/
create table #dbcc_dblist(dbid int, ldev_name varchar(30))

declare cursor_dbcc_dblist cursor
for select dbid from #dbcc_dblist

/*
** ajout de dbname et dbid pour toutes les bases valides de #dbcc_dblist, à l'exception de model et tempdb
*/
insert #dbcc_dblist(dbid, ldev_name)
select distinct su.dbid, dev.name
from master..sysdevices dev, master..sysusages su
where dev.cntrltype = 0 and
su.vstart between dev.low and dev.high
and dbid between 3 and 31000

/*
** Taille minimum de workspace à 24 pages
*/
select @min_wssize = 24

/*
** Création de la table temporaire permettant de stocker les informations de la configuration
*/
create table #dbcc_config(dbid int, dbname varchar(30) null,
scanws_size int null, textws_size int null,
wt_count smallint null, cache_size int null)

declare cursor_dbcc_config cursor for
select dbname, scanws_size, textws_size, wt_count, cache_size
from #dbcc_config

/*
** Pour chaque dbid de la table #dbcc_dblist, calcul de ses paramètres de configuration
*/
open cursor_dbcc_dblist
fetch cursor_dbcc_dblist into @dbid
while (@@sqlstatus = 0)
begin

    /*
    ** Récupère le nom de la base
    */
    select @dbname = db_name(@dbid)

    /*
    ** Contrôle si le nom de la base est déjà présent.
    ** Si tel est le cas, continue avec la prochaine entrée. La table #dbcc_dblist
    ** peut avoir plusieurs lignes pour une base donnée si cette base réside sur
    ** plusieurs devices. Le calcul n'est utile qu'une seule fois cependant.
    */
    if exists (select * from #dbcc_config where dbid = @dbid)
    begin
        fetch cursor_dbcc_dblist into @dbid
        continue
    end
end
    
```

```

/*
** Calcul le nombre de devices impacté par cette base
** Détermine le nombre de worker processes basés sur ce nombre
*/
select @wt_count = count(*) from #dbcc_dblist
where #dbcc_dblist.dbid = @dbid

if (@wt_count < 1 )
    select @wt_count = 1
else
    if (@wt_count > 128)
        select @wt_count = 128

/*
** Taille minimale du cache de 640K * @wt_count
*/
select @min_cachesize = 640 * @wt_count

/*
** La taille minimum du workspace dépend aussi du nombre de worker
** processes. Ajustement de la valeur.
*/
select @new_min_wssize = ((@wt_count + 1) * 8)
if (@new_min_wssize < @min_wssize)
    select @new_min_wssize = @min_wssize
/*
** Calcul de la taille du workspace scanws pour cette base, qui équivaut au 1.2%
** de la taille de la base. Unité : la page
*/
select @scanws_size = ceiling(sum(size) * 0.012)
from master..sysusages
where dbid = @dbid

/*
** Arrondi au multiple de 8 supérieur (unité d'allocation = 1 extent = 8 pages)
*/
select @scanws_size = ((@scanws_size / 8) + 1) * 8
/*
** Taille le workspace texte au 25% du workspace scan et l'arrondit au multiple de 8
*/
select @textws_size = ceiling(@scanws_size * 0.25)
select @textws_size = ((@textws_size / 8) + 1) * 8

/*
** La taille minimum requise pour les workspaces et de @new_min_wssize pages.
*/
if (@scanws_size < @new_min_wssize)
begin
    select @scanws_size = @new_min_wssize
end
if (@textws_size < @new_min_wssize)
begin
    select @textws_size = @new_min_wssize
end

/*
** Pour les petites bases (< 20MB), augmentation de la taille du
** scanws de 8 pages, checkstorage utilisant un extent plein pour
** le mapping des pages. 1.2% de la taille d'une petite base ne
** pourrait suffir.
*/
if (@scanws_size <= 128)
    select @scanws_size = @scanws_size + 8
/*
** conversio de la taille des workspaces en Ko
*/
select @scanws_size = @scanws_size * @@maxpagesize / 1024
select @textws_size = @textws_size * @@maxpagesize / 1024

/*
** Dimensionne la taille du cache à 20% de la taille des workspaces.

```

```

** La taille du cache est actuellement la taille du buffr pool de 16K.
** La taille doit être de @cache_size + 512K
*/
select @cache_size = ceiling((@scanws_size + @textws_size) * 0.2)
if (@cache_size < @min_cachesize)
begin
    select @cache_size = @min_cachesize
end

/*
** Insertion des valeurs calculées pour cette base dans la table
** #dbcc_config table. Cette table est utilisée à la génération d'un rapport
*/
insert into #dbcc_config(dbid, dbname, scanws_size, textws_size,
    wt_count, cache_size)
values(@dbid, @dbname, @scanws_size, @textws_size,
    @wt_count, @cache_size)

    fetch cursor_dbcc_dblist into @dbid
end

close cursor_dbcc_dblist

print ""

/*
** Calcule (en Mo) la taille de la base dbccdb. Même si checkstorage
** peut être exécuté en parallèle sur plusieurs bases,
** il est peu vraisemblable et pas recommandé de scanner plus de 2 bases.
** Déterminons donc la taille en nous basant sur les 2 plus grosses bases.
*/

select @dbid1 = dbid from #dbcc_config
    group by dbid having sum(scanws_size) = max(sum(scanws_size))

select @dbid2 = dbid from #dbcc_config where dbid != @dbid1
    group by dbid having sum(scanws_size) = max(sum(scanws_size))

select @size = (sum(scanws_size) + sum(textws_size))
    from #dbcc_config
    where dbid = @dbid1 or dbid = @dbid2

/* Retour des valeurs de configuration */
select @max_wt=max(wt_count),
    @max_cache=max(cache_size)
from #dbcc_config

/* Le premier paramètre est le nom du cache. S'il n'est pas renseigné, nous
* prendrons soit le premier cache contenant "%dbcc%" soit "default data cache"
*/
if @cache is null
select @cache = name
    from master..sysconfigures
    where name like "%dbcc%"
    and parent = 19

if @cache is null
select @cache="Default data cache"

/* Population du cache */

create table #cache
(caches varchar(30) not null,
POOL2K int null,
POOL16K int null)

insert into #cache
select convert(varchar(30), comment), null, null
from master..syscurconfigs
where config = 19
and comment != "0"
```

```

update #cache set POOL2K = s.value
from #cache c, master..syscurconfigs s
where s.config = 22
and c.caches = s.comment

update #cache set POOL16K = s.value
from #cache c, master..syscurconfigs s
where s.config = 24
and c.caches = s.comment

select @str1= convert(varchar(30),@max_cache)+"K"
exec sp_poolconfig @cache, @str1 , "16K"

/* Configuration des worker processes */
select @wt_count = value from master..sysconfigures where name ="number of worker processes"
if @wt_count < @max_wt
    exec sp_configure "number of worker processes", @max_wt

/* Adjonction des 2 segments nécessaires */
select @dev_name=name
from master..sysdevices,
     master..sysusages
where dbid=db_id("dbccdb")
and vstart between low and high
and segmap != 4

if not exists (select * from dbccdb..syssegments where name = "scan_seg")
    exec dbccdb..sp_addsegment scan_seg, dbccdb, @dev_name

if not exists (select * from dbccdb..syssegments where name = "text_seg")
    exec dbccdb..sp_addsegment text_seg, dbccdb, @dev_name

/
* Configuration pour chaque base utilisateur (il s'agit là aussi d'un choix: vous pouvez ajouter les bases système)
open cursor_dbcc_config
fetch cursor_dbcc_config into @dbname, @scanws_size, @textws_size, @wt_count, @cache_size

while (@@sqlstatus != 2)
    begin

        /* Création des Workspaces*/
        select @str1=convert(varchar(30), @scanws_size)+"K",
               @sws=@dbname+"_scan_ws"
        if not exists (select * from dbccdb..sysobjects where name=@sws and type="U")
            exec dbccdb..sp_dbcc_createws dbccdb, scan_seg, @sws,"scan", @str1

        select @str1=convert(varchar(30), @textws_size)+"K",
               @tws=@dbname+"_text_ws"
        if not exists (select * from dbccdb..sysobjects where name=@tws and type="U")
            exec dbccdb..sp_dbcc_createws dbccdb, text_seg, @tws,"text", @str1

        select @str1=convert(char(1),@wt_count)
        exec dbccdb..sp_dbcc_updateconfig dbccdb, "max worker processes", "5"
        select @str1=convert(varchar(6),@cache_size)+"K"
        exec dbccdb..sp_dbcc_updateconfig @dbname, "dbcc named cache", @cache, @str1
        exec dbccdb..sp_dbcc_updateconfig @dbname, "scan workspace", @sws
        exec dbccdb..sp_dbcc_updateconfig @dbname, "text workspace", @tws
        fetch cursor_dbcc_config into @dbname, @scanws_size, @textws_size, @wt_count, @cache_size
        print ""
        end
    close cursor_dbcc_config

/*
** Nettoyage des tabels temporaires
*/
deallocate cursor cursor_dbcc_dblast
deallocate cursor cursor_dbcc_config

drop table #dbcc_config
drop table #dbcc_dblast
    
```

```
drop table #caches
return(0)
end
go
grant execute on sp_dbcc_init to sa_role
go
```

3 - Configuration de checkstorage

Reste à récolter ce que nous venons de semer. Configurons donc notre outil en lançant simplement

```
execute dbccdb..sp_dbcc_init
```

Puis validons nos paramètres en les contrôlant via la SP officielle de Sybase

```
execute dbccdb..sp_dbcc_evaluatedb
```

4 - Utilisation de checkstorage

Il ne nous reste plus qu'à exécuter le dbcc checkstorage. Couplé au dbcc checkcatalog, il couvre le spectre de la totalité des outils de contrôle.

```
dbcc checkcatalog (MaBase)
if @@error=0
  print "Catalogue en ordre"
else
  print "Erreur lors du checkcatalog"

dbcc checkstorage (MaBase)
if @@error=0
  print "Checkstorage en ordre"
else
  print "Erreur lors du checkstorage"
```