Correction des bases SQL Server corrompues.

Introduction

En pratique il est extrêmement rare de voir une base de données SQL server se corrompre. Aucun bug concernant le moteur de stockage de Microsoft SQL Server n'a été détecté depuis près de 20 ans. Cela signifie en pratique que de telles erreurs ne peuvent intervenir qu'à l'extérieur du service SQL Server.
Ce peut être :

- Un disque défectueux (probabilité 99,5%) ;
- Un contrôleur disque défaillant (probabilité 0,1%) ;
- Tout dispositif entre les deux (bus, lien vers SAN... probabilité 0,4%).

Néanmoins, il est important de réagir au plus vite après détection pour corriger le problème.

Dans tous les cas, il est urgent de changer le dispositif en cause après l'avoir identifié.
Si c'est le stockage qui est en cause, changez immédiatement les disques après avoir déplacé les fichiers par le biais des commandes sp_detach_db et CREATE DATABASE ... FOR ATTACH... Voire annexe 1. En effet, tenter d'effectuer des opérations de correction sur un disque abîmé, ne peut qu'entrainer de plus en plus de corruptions.

Par **Frédéric BROUARD**
alias SQLpro
SQLpro@SQLspot.com
entreprise SQL SPOT (Paris / PACA)

Architecte de données :
expertise, audit,
optmisation, tuning,
conseil, formation,
modélisation, architecture

spécialiste SGBD relationnels, langage SQL,
Microsoft SQL Server et PostGreSQL.

le blog : http://blog.developpez.com/sqlpro

le site : http://sqlpro.developpez.com

---

1 – Détection des corruptions

La détection d'une corruption peut se faire de deux manières :

- Lors de la lecture ***physique*** d'une page (tables et index étant organisés en pages) ;
- Lors de l'exécution d'une commande DBCC CHECK... destinée à vérifier l'intégrité physique des pages de la base.

Du fait de la mise en cache des données, la lecture physique d'une page est rare en pratique si la base est bien organisée et si le cache est bien dimensionné. Or la probabilité de récupérer intégralement une page abîmée décroit en fonction du temps qui passe. En effet, dans le pire des cas, et pour récupérer intégralement la page, il faut rechercher la page avant sa corruption dans la dernière sauvegarde complète et réappliquer toutes les transactions concernant cette page en utilisant les sauvegardes transactionnelles. Si la corruption est plus ancienne que la dernière sauvegarde complète, alors l'opération s'avère impossible.

Il est alors fortement conseillé d'effectuer une vérification par DBCC CHECK... à la même fréquence que la sauvegarde complète ou différentielle.

Après avoir été informé d'une corruption, investiguez dans les journaux d'événement de SQL Server et Windows pour tenter d'avoir le maximum d'information sur le sujet. Vous pouvez être redirigé vers un « dump » qui donne des détails sur le problème.

Erreur de donnée ou de transaction ?

Si l'erreur concerne le journal de transaction, alors détachez la base et rattachez-là en demandant une reconstruction du journal de transaction.

Dans le cas contraire (erreur de données) il va falloir tenter de réparer...

1.1 – Détection par lancement d'une requête

Une correction détectée lors du lancement d'une requête renvoi généralement une erreur n°824. Exemple :

```
Msg 824, Niveau 24, État 2, Ligne 88
SQL Server a détecté une erreur d'E/S logique et relative à la cohérence. L'erreur somme de
contrôle incorrecte (somme de contrôle attendue : 0xf782e801 ; somme de contrôle réelle :
0xf7802511) s'est produite pendant une opération de lire de la page (1:80) dans la base de
données avec l'ID 12 au niveau du décalage 0x000000000a0000 dans le fichier 'C:\SQL
Server\2014F\MSSQL12.SQL2014FBIN2\MSSQL\DATA\DB_CORRUPTION.mdf'. Vous trouverez peut-être
plus de détails dans les messages supplémentaires qui figurent dans le journal des erreurs
et le journal des évènements système de SQL Server. Il s'agit d'une condition d'erreur
sévère qui met en péril l'intégrité de la base de données et qui doit être corrigée
immédiatement. Effectuez une vérification complète de la cohérence de la base de données
(DBCC CHECKDB). Cette erreur peut être due à de nombreux facteurs ; pour plus
d'informations, reportez-vous à la documentation en ligne de SQL Server.
```

Le message indique : « *une opération de lire de la page (1:80) dans la base de données avec l'ID 12* », nous en déduisons :

- l'identifiant de page 1:80 (fichier 1 page 80) ;

- l'identifiant de base d'ID 12.

1.2 – Détection lors d'une vérification DBCC CHECK...

Lors d'une vérification effectuée par une commande comme DBCC CHECKDB, les messages sont plus étoffés :

```
Résultats DBCC pour 'DB_CORRUPTION'.

...

Msg 8939, Niveau 16, État 98, Ligne 124
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data), page (1:80). Échec du test
(IS_OFF (BUF_IOERR, pBUF->bstat)). Valeurs 133129 et -4.
Msg 8928, Niveau 16, État 1, Ligne 124
ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID d'unité
d'allocation 72057594045792256 (type In-row data) : impossible de traiter la page (1:80).
Pour plus d'informations, consultez les autres erreurs.
Msg 8976, Niveau 16, État 1, Ligne 124
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data). La page (1:80) n'a pas été
détectée lors de l'analyse alors que sa page parent (1:78) et les (1:79) pages précédentes
la référencent. Vérifiez les erreurs précédentes.
Msg 8978, Niveau 16, État 1, Ligne 124
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data). Une référence de la page
précédente (1:80) est manquante à la page (1:89). Un problème de liaison de chaîne s'est
peut-être produit.
Résultats DBCC pour 'T'.
Il y a 85 lignes dans 6 pages pour l'objet "T".
CHECKDB a trouvé 0 erreurs d'allocation et 4 erreurs de cohérence dans la table 'T' (ID
d'objet 245575913).

...

CHECKDB a trouvé 0 erreurs d'allocation et 4 erreurs de cohérence dans la base de données
'DB_CORRUPTION'.
repair_allow_data_loss est le niveau minimum de réparation pour les erreurs trouvées par
DBCC CHECKDB (DB_CORRUPTION).
Exécution de DBCC terminée. Si DBCC vous a adressé des messages d'erreur, contactez
l'administrateur système.
```

Avec ces messages, le nom de la base apparait en clair (DB_CORRUPTION) ainsi que le nom de la table (T). Y figurent aussi :

- l'identifiant de l'objet : 245575913
- l'identifiant de l'index : 1

1.3 – Confirmation à l'aide de suspect_pages

Finalement, vous pouvez confirmer les informations de pages corrompues à l'aide de la requête :

```
SELECT * FROM msdb.dbo.suspect_pages
```

Qui donne par exemple :

```
database_id file_id   page_id            event_type  error_count last_update_date
----------- --------- ------------------ ----------- ----------- ----------------------
12          1         80                 4           2           2015-05-02 17:08:33.987
```

2 - Identification des objets en cause

À partir des identifiants de base, d'objet et d'index (et accessoirement de partition et d'allocation) vous pouvez retrouver les objets physiques en cause (nom des tables et index).

2.1 – Obtenir les identifiants

Pour retrouver la base de données (l'identifiant qui a été révélé est 12) [#01] :

```
SELECT name FROM sys.databases WHERE database_id = 12;
```

Pour retrouver l'objet contenu dans une page (pour nous la page 80 du fichier 1), lancez le lot de commande [#02] :

```
DBCC TRACEON (3604)
--> pour rediriger la sortie de la commande DBCC PAGE vers SSMS
GO
DBCC PAGE(DB_CORRUPTION, 1, 80, 1)
--> syntaxe : nom_base, n° fichier, n° page, mode de visu
GO
```

Dont la sortie sera par exemple la suivante :

```
Exécution de DBCC terminée. Si DBCC vous a adressé des messages d'erreur, contactez
l'administrateur système.

PAGE: (1:80)


BUFFER:


BUF @0x00000002FBF33FC0

bpage = 0x00000002EFFDA000          bhash = 0x0000000000000000          bpageno = (1:80)
bdbid = 12                          breferences = 1                     bcputicks = 0
bsampleCount = 0                    bUse1 = 11008                       bstat = 0x809
blog = 0x215a215a                   bnext = 0x0000000000000000

PAGE HEADER:


Page @0x00000002EFFDA000

m_pageId = (1:80)                   m_headerVersion = 1                 m_type = 1
m_typeFlagBits = 0x0                m_level = 0                         m_flagBits = 0x200
m_objId (AllocUnitId.idObj) = 120   m_indexId (AllocUnitId.idInd) = 256
Metadata: AllocUnitId = 72057594045792256
Metadata: PartitionId = 72057594040549376
Metadata: IndexId = 1
Metadata: ObjectId = 245575913      m_prevPage = (1:79)                 m_nextPage = (1:89)
pminlen = 508                       m_slotCnt = 15                      m_freeCnt = 401
m_freeData = 7761                   m_reservedCnt = 0                   m_lsn = (34:201:15)
```

```
m_xactReserved = 0                  m_xdesId = (0:0)                    m_ghostRecCnt = 0
m_tornBits = -142415871             DB Frag ID = 1

Allocation Status

GAM (1:2) = ALLOCATED               SGAM (1:3) = NOT ALLOCATED
PFS (1:1) = 0x60 MIXED_EXT ALLOCATED   0_PCT_FULL                          DIFF (1:6) = NOT
CHANGED
ML (1:7) = NOT MIN_LOGGED
Msg 0, Niveau 11, État 0, Ligne 110
Une erreur grave s'est produite sur la commande actuelle. Les résultats éventuels doivent
être ignorés.
```

Ou encore en mode « table » [#04] :

```
DBCC PAGE(DB_CORRUPTION, 1, 80, 0) WITH TABLERESULTS
GO
```

	ParentObject	Object	Field	VALUE
16	PAGE HEADER:	Page @0x00000002EFFDA000	m_level	0
17	PAGE HEADER:	Page @0x00000002EFFDA000	m_flagBits	0x200
18	PAGE HEADER:	Page @0x00000002EFFDA000	m_objId (AllocUnitId.idObj)	120
19	PAGE HEADER:	Page @0x00000002EFFDA000	m_indexId (AllocUnitId.idInd)	256
20	PAGE HEADER:	Page @0x00000002EFFDA000	Metadata: AllocUnitId	72057594045792256
21	PAGE HEADER:	Page @0x00000002EFFDA000	Metadata: PartitionId	72057594040549376
22	PAGE HEADER:	Page @0x00000002EFFDA000	Metadata: IndexId	1
23	PAGE HEADER:	Page @0x00000002EFFDA000	Metadata: ObjectId	245575913
24	PAGE HEADER:	Page @0x00000002EFFDA000	m_prevPage	(1:79)
25	PAGE HEADER:	Page @0x00000002EFFDA000	m_nextPage	(1:89)
26	PAGE HEADER:	Page @0x00000002EFFDA000	pminlen	508
27	PAGE HEADER:	Page @0x00000002EFFDA000	m_slotCnt	15
28	PAGE HEADER:	Page @0x00000002EFFDA000	m_freeCnt	401

Nous avons fait figurer en bleu et gras, les informations qui nous intéressent pour la commande en mode de sortie « texte » et en surlignage bleu celles de la commande en mode de sortie « grille ».

Ici c'est l'index d'identifiant 1 de l'objet d'identifiant 245575913 qui est en cause.

NOTA : un index_id 0 ou 1 est la table (0 sous forme "HEAP", 1 sous forme "CLUSTERED").

2.2 - Quel est la table et l'index concerné ?

Si l'identifiant d'index est 0 ou 1 c'est la table qui est concernée. 0 signifie que la table est sous forme de tas (« HEAP »), 1 sous forme « CLUSTERED ».

Pour trouver l'index et la table concernée, faite la requête suivante :

```
SELECT s.name AS TABLE_SCHEMA, o.name AS TABLE_NAME,
       o.type_desc AS TABLE_TYPE, i.name AS INDEX_NAME,
          CASE WHEN i.index_id <= 1 THEN 'TABLE'
               WHEN i.type_desc = 'NONCLUSTERED' THEN 'INDEX'
                     ELSE i.type_desc END as INDEX_TYPE
FROM   sys.objects AS o
```

```
       INNER JOIN sys.schemas AS s
               ON o.schema_id = s.schema_id
          INNER JOIN sys.indexes AS i
               ON o.object_id = i.object_id
WHERE  o.object_id = 245575913
  AND  i.index_id = 1;
```

À l'aide des informations récupérées ci-avant.

Notez les données de colonnes TABLE_TYPE et INDEX_TYPE qui vous seront nécessaire pour la méthode de correction.

3 - Correction

Les méthodes diffèrent en fonction de la nature de l'objet dont la page est corrompue.

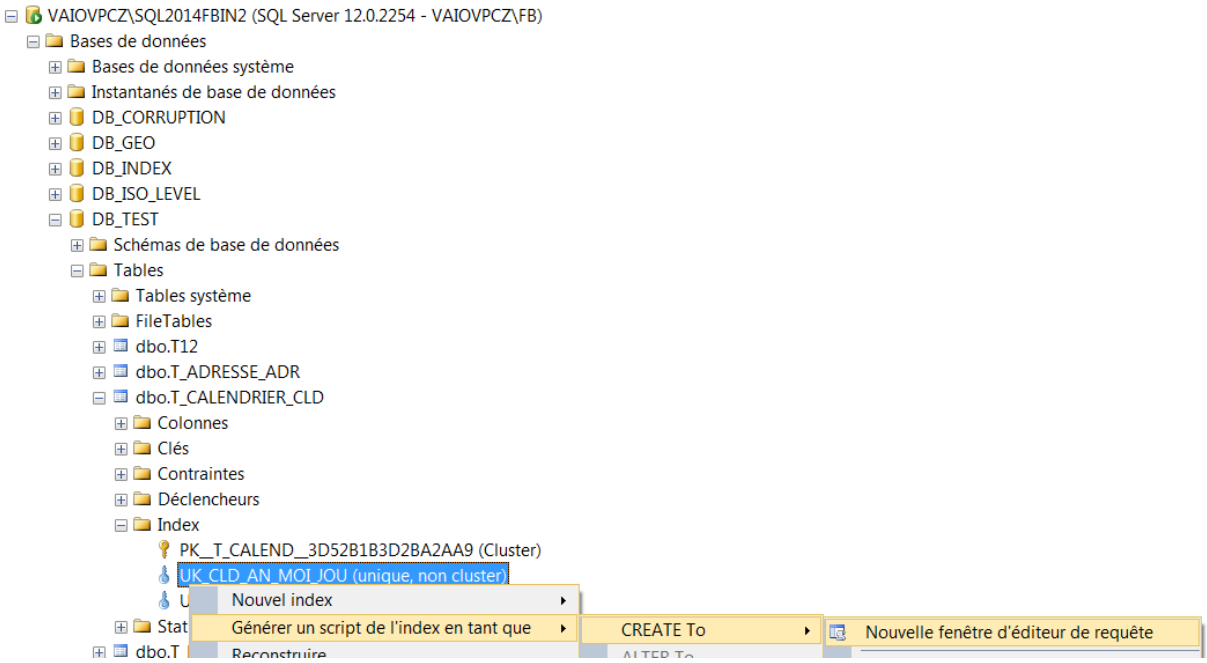
3.1 – La page concerne un index

Si la page concerne un index non clustered ou un index non table (INDEX_TYPE <> 'TABLE') alors la méthode la plus simple est de reconstruire l'index à l'identique, quelle que soit le type de table (système ou utilisateur).
Pour ce faire, utilisez la commande CREATE INDEX :

- soit après avoir supprimé l'index ;
- soit avec l'option DROP_EXISTING = ON.

NOTA : l'index peut résulter de la création d'une contrainte de type PRIMARY KEY ou UNIQUE.

ATTENTION : comme vous ne connaissez pas forcément la définition de l'index concerné, vous devez la retrouver avant suppression par exemple en utilisant SSMS.
Pour ce faire allez dans l'arborescence à la base concernée, trouvez la table dans la liste des tables, puis l'index dans la liste des index et obtenez le rétro script SQL de création de l'index par un clic droit sur « Générer un script de l'index en tant que » / « CREATE To » / « Nouvelle fenêtre d'éditeur de requête »



Dans le cas où l'index est une contrainte PRIMARY KEY ou UNIQUE, le script commence par la création d'une contrainte, suivie par des indications pour la création de l'index.

3.1.1 – Reconstruction d'un index hors contrainte

Voici une première manière pour reconstruire un index :

```
USE [DB_CORRUPTION]
GO

CREATE NONCLUSTERED INDEX [X_CLD_AN_SEMAINE]
    ON [dbo].[T_CALENDRIER_CLD]
               (
                      [CLD_AN] ASC,
                      [CLD_SEMAINE] ASC
               )
   WITH (PAD_INDEX = OFF,
         STATISTICS_NORECOMPUTE = OFF,
               SORT_IN_TEMPDB = OFF,
               DROP_EXISTING = OFF,
               ONLINE = OFF,
               ALLOW_ROW_LOCKS = ON,
               ALLOW_PAGE_LOCKS = ON,
               DROP_EXISTING = ON);
GO
```

Notez l'option DROP_EXISTING = ON, qui recréé l'index à l'identique.

Si vous n'avez pas déplacé les fichiers de la base, vous pouvez le faire à cette occasion en créant un nouveau groupe de fichiers, un ou plusieurs fichiers dans le groupe (voir annexe 2) et en précisant le nouvel emplacement pour cet index dans le groupe de fichiers nouvellement créé.
La commande est par exemple la suivante :

```
USE [DB_CORRUPTION]
GO

CREATE NONCLUSTERED INDEX [X_CLD_AN_SEMAINE]
    ON [dbo].[T_CALENDRIER_CLD]
               (
                      [CLD_AN] ASC,
                      [CLD_SEMAINE] ASC
               )
   WITH (PAD_INDEX = OFF,
         STATISTICS_NORECOMPUTE = OFF,
               SORT_IN_TEMPDB = OFF,
               DROP_EXISTING = OFF,
               ONLINE = OFF,
               ALLOW_ROW_LOCKS = ON,
               ALLOW_PAGE_LOCKS = ON,
               DROP_EXISTING = ON)
   ON FG_NEW;
GO
```

Notez l'option ON FG_NEW qui indique le nouvel emplacement ou sera recréé l'index.

3.1.2 - Reconstruction d'un index non clustered sous contrainte

Dans le cas ou votre index est le résultat d'une contrainte PRIMARY KEY (non clustered) ou UNIQUE, vous pouvez au choix, reconstruire l'index (CREATE INDEX... DROP_EXISTING) ou supprimer la contrainte pour la recréer.

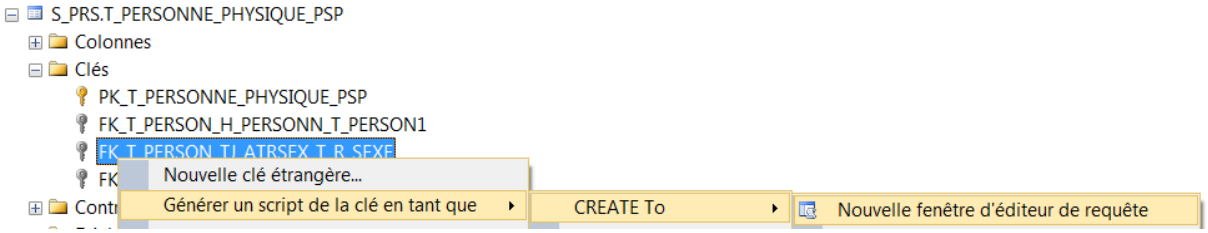
Le problème est que votre contrainte PRIMARY KEY ou UNIQUE peut être utilisée par une clef étrangère. Il faut donc aussi les supprimer préalablement puis les recréer après.

Pour identifier les contraintes FOREIGN KEY à « débrancher », en connaissant le nom de la table dont l'index de contrainte PRIMARY KEY ou UNIQUE est à réparer, utilisez la requête suivante [#05] :

```
SELECT RC.CONSTRAINT_NAME, FK.TABLE_SCHEMA, FK.TABLE_NAME,
       'ALTER TABLE [' + FK.TABLE_SCHEMA + '].[' + FK.TABLE_NAME
          + '] DROP CONSTRAINT [' + FK.CONSTRAINT_NAME + '];' AS DROP_COMMAND
FROM   INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS UK
       INNER JOIN INFORMATION_SCHEMA.REFERENTIAL_CONSTRAINTS AS RC
                ON UK.CONSTRAINT_NAME = RC.UNIQUE_CONSTRAINT_NAME
                      AND UK.CONSTRAINT_SCHEMA = RC.UNIQUE_CONSTRAINT_SCHEMA
          INNER JOIN INFORMATION_SCHEMA.TABLE_CONSTRAINTS AS FK
                ON RC.CONSTRAINT_NAME = FK.CONSTRAINT_NAME
                      AND RC.CONSTRAINT_SCHEMA = FK.CONSTRAINT_SCHEMA
       INNER JOIN sys.objects AS o
                ON o.object_id = OBJECT_ID(UK.TABLE_SCHEMA+'.'+UK.TABLE_NAME)
          INNER JOIN sys.indexes AS i
                ON o.object_id = i.object_id
                      AND i.name = UK.CONSTRAINT_NAME
WHERE  UK.TABLE_SCHEMA = 'S_PRS'           --> schéma SQL de la table concernée
AND    UK.TABLE_NAME   = 'T_PERSONNE_PRS'  --> nom de la table concernée
AND    index_id        = 1                 --> identifiant de l'index concerné
```

NOTA : la colonne DROP_COMMAND vous donne le script de suppression des contraintes FOREIGN KEY nécessaires. Vous pouvez aussi désactiver les contraintes FOREIGN KEYs puis les réactiver après correction.

Une fois ces contraintes identifiées, vous pouvez obtenir le script de création de ces contraintes, par l'interface graphique de SSMS :



Cliquez droit sur la contrainte FOREIGN KEY considérée et sélectionnez « Générer un script de la clé en tant que » / « CREATE To » / « Nouvelle fenêtre d'éditeur de requête ». Ceci conduit par exemple à un script similaire à l'exemple ci-dessous :

```
USE [DB_GRAND_HOTEL]
GO

ALTER TABLE [S_PRS].[T_PERSONNE_PHYSIQUE_PSP]
   WITH CHECK
   ADD  CONSTRAINT [FK_T_PERSON_H_PERSONN_T_PERSON1]
      FOREIGN KEY([PRS_ID])
      REFERENCES [S_PRS].[T_PERSONNE_PRS] ([PRS_ID])
GO

ALTER TABLE [S_PRS].[T_PERSONNE_PHYSIQUE_PSP]
   CHECK CONSTRAINT [FK_T_PERSON_H_PERSONN_T_PERSON1]
GO
```

Conservez l'ensemble de ces scripts pour reconstruire les clefs étrangères après reconstruction de la clef primaire ou étrangère.

Le scénario final est alors le suivant :

1) suppression des contraintes FOREIGN KEY
2) reconstruction de l'index corrompu
3) remise en place des contraintes FOREIGN KEY

3.2 – La page concerne une table utilisateur

S'il s'agit d'un index CLUSTERED (donc la table sous forme d'index, inex_id = 1) ou d'une table en HEAP (index_id = 0), il n'est pas possible d'utiliser la technique de recréation d'index. Le moyen est alors de tenter une réparation, et si elle échoue, une restauration, soit page par page, soit globale de la base.

3.2.1 – Réparation à l'aide de DBCC CHECK...

Tentez de réparer en mode REPAIR_REBUILD à l'aide de la commande DBCC CHECK... Par exemple pour une table T corrompue située dans une base de nom DB_CORRUPTION, la commande à passer est [#06] :

```
ALTER DATABASE [DB_CORRUPTION]
```

```
   SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO
DBCC CHECKTABLE ('T', REPAIR_REBUILD)
```

Le résultat peut être une réparation, mais aussi un échec :

```
Résultats DBCC pour 'T'.
Msg 8928, Niveau 16, État 1, Ligne 10
ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID d'unité
d'allocation 72057594045792256 (type In-row data) : impossible de traiter la page (1:80).
Pour plus d'informations, consultez les autres erreurs.
        Le niveau de réparation pour l'instruction DBCC est tel que la réparation est
ignorée.
Msg 8939, Niveau 16, État 98, Ligne 10
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data), page (1:80). Échec du test
(IS_OFF (BUF_IOERR, pBUF->bstat)). Valeurs 2057 et -4.
        Résoudre cette erreur impose de résoudre d'abord les autres erreurs.
Msg 8976, Niveau 16, État 1, Ligne 10
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data). La page (1:80) n'a pas été
détectée lors de l'analyse alors que sa page parent (1:78) et les (1:79) pages précédentes
la référencent. Vérifiez les erreurs précédentes.
        Résoudre cette erreur impose de résoudre d'abord les autres erreurs.
Msg 8978, Niveau 16, État 1, Ligne 10
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data). Une référence de la page
précédente (1:80) est manquante à la page (1:89). Un problème de liaison de chaîne s'est
peut-être produit.
        Résoudre cette erreur impose de résoudre d'abord les autres erreurs.
Il y a 385 lignes dans 26 pages pour l'objet "T".
CHECKTABLE a trouvé 0 erreurs d'allocation et 4 erreurs de cohérence dans la table 'T' (ID
d'objet 245575913).
repair_allow_data_loss est le niveau minimum de réparation pour les erreurs trouvées par
DBCC CHECKTABLE (DB_CORRUPTION.dbo.T, repair_rebuild).
Exécution de DBCC terminée. Si DBCC vous a adressé des messages d'erreur, contactez
l'administrateur système.
```

ATTENTION : avant de lancer la commande DBCC en mode REPAIR_ALLOW_DATA_LOSS qui supprime les données fautives, tentez une réparation par restauration de page.

```
ALTER DATABASE [DB_CORRUPTION]
   SET SINGLE_USER WITH ROLLBACK IMMEDIATE;
GO
DBCC CHECKTABLE ('T', REPAIR_ALLOW_DATA_LOSS)
```

Cette fois-ci l'erreur semble résolue :

```
Résultats DBCC pour 'T'.
Réparation : l'index Clustered a été reconstruit pour l'objet "dbo.T" dans la base de
données "DB_CORRUPTION".
Réparation : la page (1:80) a été libérée de l'objet ID 245575913, index ID 1, partition ID
72057594040549376, unité d'allocation ID 72057594045792256 (type In-row data).
Réparation : l'index Nonclustered a été reconstruit pour l'objet "dbo.T, X" dans la base de
données "DB_CORRUPTION".
Msg 8945, Niveau 16, État 1, Ligne 10
Erreur de table : ID d'objet 245575913, l'ID d'index 1 sera reconstruit.
        L'erreur a été résolue.
Msg 8928, Niveau 16, État 1, Ligne 10
```

```
ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID d'unité
d'allocation 72057594045792256 (type In-row data) : impossible de traiter la page (1:80).
Pour plus d'informations, consultez les autres erreurs.
        L'erreur a été résolue.
Msg 8939, Niveau 16, État 98, Ligne 10
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data), page (1:80). Échec du test
(IS_OFF (BUF_IOERR, pBUF->bstat)). Valeurs 2057 et -4.
        L'erreur a été résolue.
Msg 8976, Niveau 16, État 1, Ligne 10
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data). La page (1:80) n'a pas été
détectée lors de l'analyse alors que sa page parent (1:78) et les (1:79) pages précédentes
la référencent. Vérifiez les erreurs précédentes.
        L'erreur a été résolue.
Msg 8978, Niveau 16, État 1, Ligne 10
Erreur de table : ID d'objet 245575913, ID d'index 1, ID de partition 72057594040549376, ID
d'unité d'allocation 72057594045792256 (type In-row data). Une référence de la page
précédente (1:80) est manquante à la page (1:89). Un problème de liaison de chaîne s'est
peut-être produit.
        L'erreur a été résolue.
Msg 8945, Niveau 16, État 1, Ligne 10
Erreur de table : ID d'objet 245575913, l'ID d'index 2 sera reconstruit.
        L'erreur a été résolue.
Il y a 385 lignes dans 26 pages pour l'objet "T".
CHECKTABLE a trouvé 0 erreurs d'allocation et 4 erreurs de cohérence dans la table 'T' (ID
d'objet 245575913).
CHECKTABLE a corrigé 0 erreurs d'allocation et 4 erreurs de cohérence dans la table 'T' (ID
d'objet 245575913).
Exécution de DBCC terminée. Si DBCC vous a adressé des messages d'erreur, contactez
l'administrateur système.
```

En vérifiant le contenu de la table :

```
SELECT * FROM T
```

Vous constaterez que certaines lignes ont été perdues.

ATTENTION : si la table est mère d'une intégrité référentielle, alors certaines références contenues dans les tables filles sont devenues obsolètes. Dans ce cas vous devez identifier les tables filles ayant des erreurs d'intégrité référentielle à l'aide de DBCC CHECKCONSTRAINTS et supprimer aussi les lignes filles devenues orphelines à l'aide de la commande DELETE.

NOTA : n'oubliez pas de replacer votre base en mode multi utilisateur :

```
ALTER DATABASE [DB_CORRUPTION]
   SET MULTI_USER;
```

3.2.2 – Réparation par restauration de page

La technique consiste à récupérer la page dans la dernière sauvegarde complète puis de procéder à une ré-application de tous les journaux de transaction.
Pour ne perdre aucune donnée il convient de commencer par une sauvegarde du journal de transaction en faisant un « tail log backup » après avoir isolé la base pour que de nouvelles mises à jour n'affectent pas le processus.

Si vous êtes en version Enterprise, ceci peut être fait « on line ». Sinon il vous faudra passer la base en mode « off line » à l'aide de la commande ALTER DATABASE ... SET OFFLINE.

L'enchainement est donc le suivant :

1) RESTORE DATABASE ... PAGE ... WITH NORECOVERY
2) RESTORE LOG ... WITH NORECOVERY : autant de fois qu'il y a de journaux sauvegardé normalement
3) RESTORE LOG ... WITH RECOVERY : du journal sauvegardé en mode « tail log backup »

NOTA : une sauvegarde du journal de transaction en mode « tail log backup » consiste à rajouter l'option NO_TRUNCATE, qui n'affecte aucun des fichiers de la base et par conséquent ne purge pas le journal des transactions.

Exemple :

```
USE master;
GO

RESTORE DATABASE DB_CORRUPTION
   PAGE = '1:80' --> pages concernées
   FROM  DISK = 'C:\SAVE\DB_CORRUPTION_20150501.BAK'
   WITH NORECOVERY;

RESTORE LOG DB_CORRUPTION
   FROM  DISK = 'C:\SAVE\DB_CORRUPTION_20150501_08H00.TRN'
   WITH NORECOVERY;

RESTORE LOG DB_CORRUPTION
   FROM  DISK = 'C:\SAVE\DB_CORRUPTION_20150501_09H00.TRN'
   WITH NORECOVERY;

RESTORE LOG DB_CORRUPTION
   FROM  DISK = 'C:\SAVE\DB_CORRUPTION_TAIL.TRN'
   WITH RECOVERY;
```

Il faut restaurer tous les journaux de transaction entre la sauvegarde complète et la sauvegarde du JT en mode « tail log backup » et pour ce dernier le passer en mode RECOVERY.

3.3 – La page concerne une table système

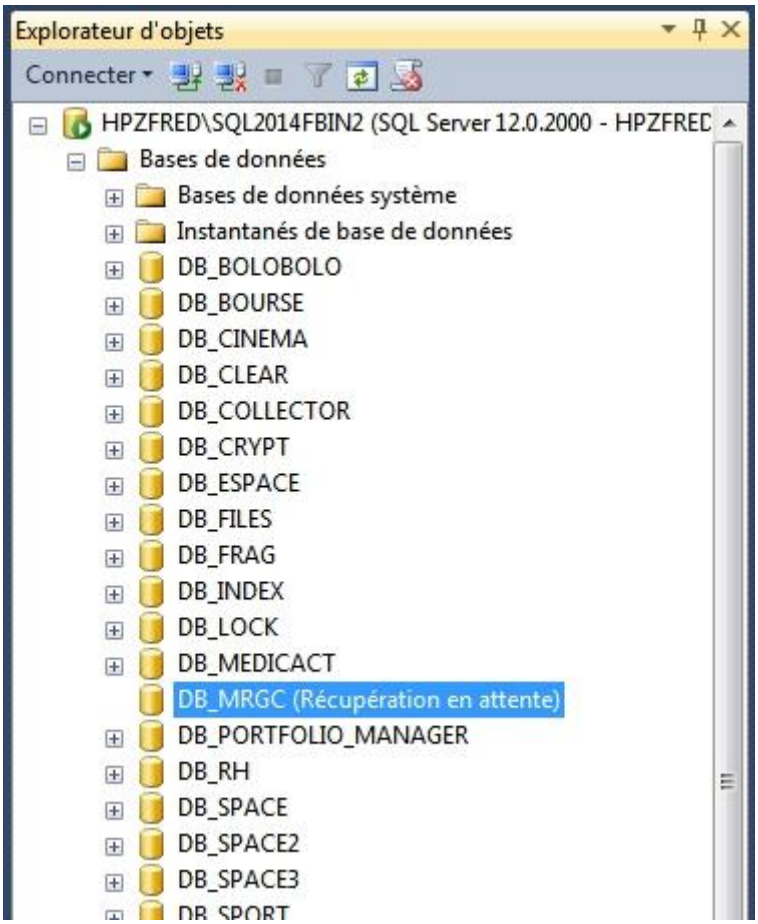
Pour un TABLE_TYPE comme SYSTEM_TABLE ou INTERNAL_TABLE, voire SERVICE_QUEUE[1], la solution est identique à celle précédemment indiquée (table utilisateur).

3.4 – La base est en mode « récupération en attente »

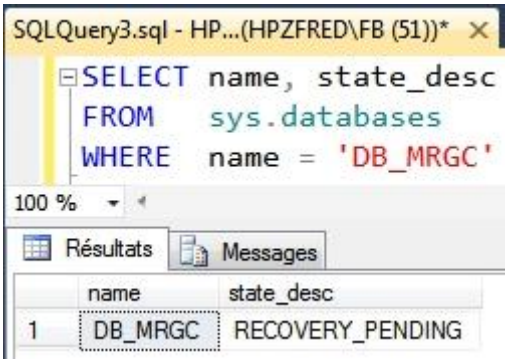
***NOTA** : dans les exemples et images qui suivent notre base s'appelle DB_MRGC.*

[1] Pour ce dernier type « SERVICE QUEUE », il est possible de supprimer et remettre en place les différents éléments de Service Broker que vous avez mis en place, sauf si cette table est une file d'attente interne (par exemple pour database mail...).

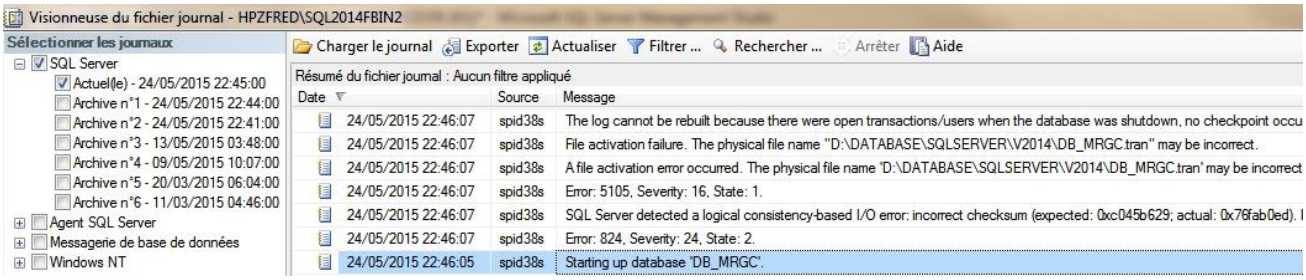
Dans le cas où la base serait dans un mode de récupération en attente…



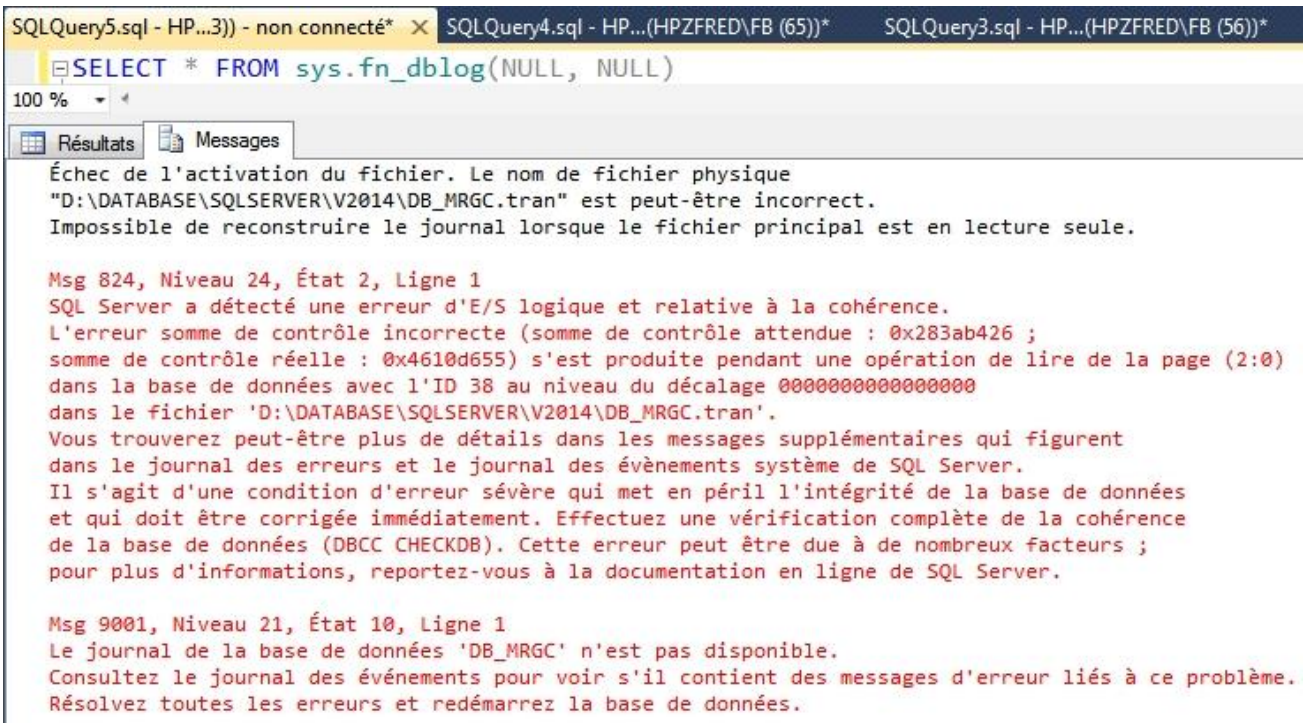
Confirmé par une requête telle que celle-ci :



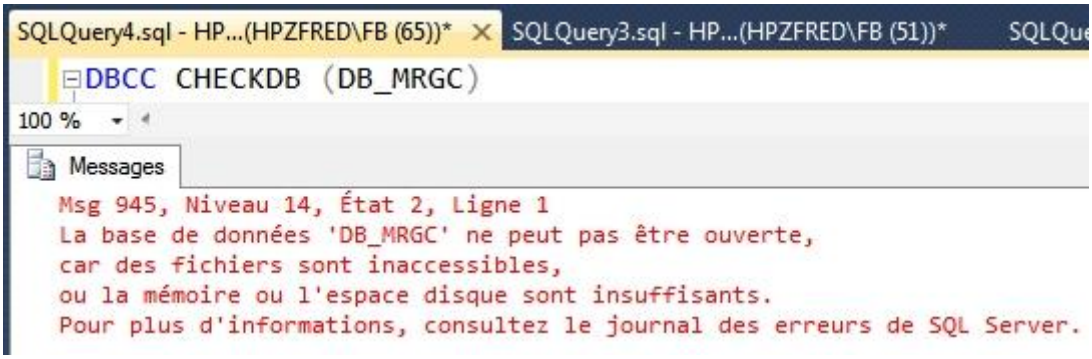
Par exemple suite à un dommage survenu au journal de transaction (vérifiez le contenu du journal d'événement de SQL Server) :



Et que certaines requêtes se soldent par une erreur :



Et qu'une vérification de la base part en échec :



Et finalement qu'une tentative de réparation n'aboutit pas non plus :

```
DBCC CHECKDB (DB_MRGC, REPAIR_ALLOW_DATA_LOSS)
```

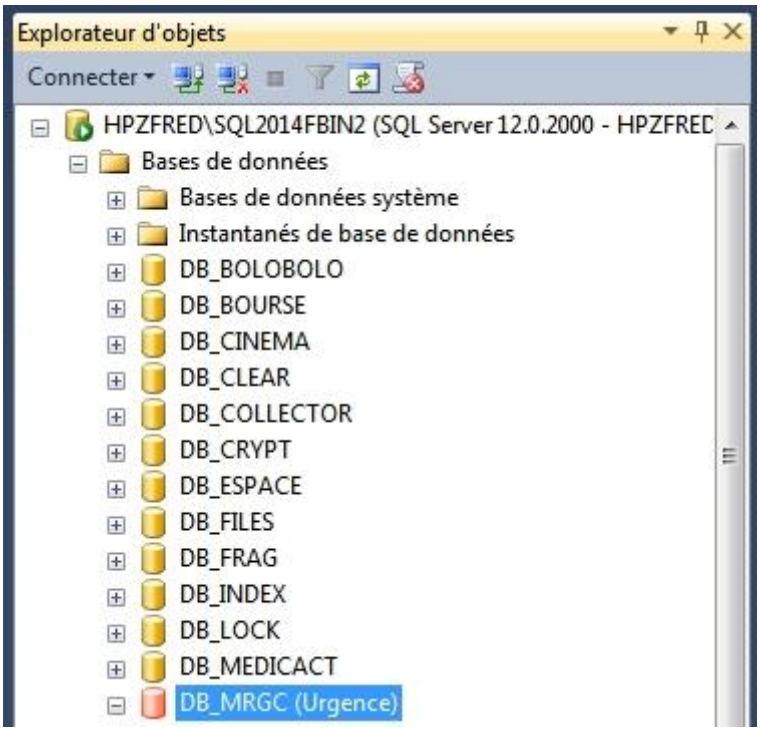
Alors il reste à tenter de mettre la base en mode d'urgence et réparer :

3.4.1 - Placement de la base en mode EMERGENCEY (URGENCE)

Lancez la commande suivante [#08] :

```
ALTER DATABASE DB_MRGC SET EMERGENCY
GO
```

Ce qui a pour effet de la placer en mode EMERGENCY :



Dès lors, vous pouvez tenter de réparer par le script de commandes suivant [#09] :

```
ALTER DATABASE DB_MRGC SET SINGLE_USER;
GO
DBCC CHECKDB (DB_MRGC, REPAIR_ALLOW_DATA_LOSS) WITH NO_INFOMSGS;
GO
ALTER DATABASE DB_MRGC SET MULTI_USER;
GO
```

Potentiellement ce script fait perdre des données. Vous devez revérifier la base après coup en lançant les commandes suivantes [#10] :

```
DBCC CHECKDB (DB_MRGC);
GO
USE DB_MRGC
GO
DBCC CHECKCONSTRAINTS ();
```

4 – Vérification après réparation

Une fois toutes les erreurs réparées, il est conseillé de vérifier ses réparations en effectuant un DBCC CHECKTABLE sur les tables concernées et de purger la table indiquant les pages corrompues à l'aide de la commande :

```
DELETE FROM msdb.dbo.suspect_pages
```

5 – Particularité du « mirroring » ou de « AlwaysOn »

La haute disponibilité mise en place à l'aide du mirroring (depuis la version 2005) où de AlwaysOn (depuis la version 2012) procède à la réparation automatique des pages. Ce qui ne veut pas dire qu'il ne faut pas procéder à une vérification d'intégrité de la base à l'aide de DBCC CHECK... et de temps à autre purger votre table msdb.dbo.suspect_pages.

ATTENTION : dans tous les cas cela indique une très forte suspicion de problème physique sur un disque et il faut toujours commencer par remplacer les supports physiques.

ANNEXE 1 – déplacement des fichiers d'une base

Pour déplacer les fichiers d'une base, il suffit de 4 étapes :

- Identification des fichiers composant la base
- Détachement de la base
- Déplacement des fichiers
- Recréation de la base à l'aide des fichiers une fois déplacés

ATTENTION : lisez attentivement une première fois cet article avant de procéder.

NOTA : des particularités s'appliquent si votre base contient du stockage de type FILESTREAM ou FILETABLE.

1 – identification des fichiers d'une base

Vous pouvez utiliser l'une quelconque de ces deux requêtes :

```
USE [MA_BASE] --> nom de la base à déplacer

SELECT name, type_desc, physical_name,
       size * 8 / 1024.0 AS SIZE_MB
FROM   sys.database_files;
```

```
SELECT name, type_desc, physical_name,
       size * 8 / 1024.0 AS SIZE_MB
FROM   sys.master_files
WHERE  database_id = DB_ID('MA_BASE'); --> nom de la base à déplacer
```

2 – Détachement de la base

Pour détacher une base, il faut utiliser la procédure stockée sp_detach_db.
Exemple :

```
EXEC sp_detach_db 'MA_BASE', 'true';
```

Dès lors votre base n'est plus visible dans l'IHM SSMS, mais les fichiers ont été conservés sur les disques.

NOTA : le premier paramètre est le nom de la base, le second à « true » indique d'ignorer le réajustement des statistiques.

3 – déplacement des fichiers

Une fois les fichiers identifiés dans le résultat de la requête en 1) la colonne *physical_name* vous montre les noms et chemins initiaux des fichiers à déplacer.

Vous pouvez alors utiliser la commande DOS MOVE ou un script PowerShell pour déplacer vos fichiers

Lors de l'étape 1) vous pouvez générer le script de déplacement des fichiers à l'aide d'une requête SQL.
Exemple - soit à déplacer tous les fichiers de la base vers le chemin « D:\DATABASES\ » :

```
DECLARE @NEWPATH NVARCHAR(256);
SET @NEWPATH = 'D:\DATABASES\'; --> nouveau chemin à prendre en compte

SELECT name, type_desc, physical_name,
       size * 8 / 1024.0 AS SIZE_MB,
          N'MOVE /Y "' + physical_name + N'" "'
          + @NEWPATH + N'"' AS OS_COMMAND
FROM   sys.master_files
WHERE  database_id = DB_ID('MA_BASE'); --> nom de la base à déplacer
```

La colonne OS_COMMAND contient les commandes DOS pour déplacer

Ou encore :

```
DECLARE @OSCMD VARCHAR(8000);
SET @OSCMD = 'C:\DATABASES\'; --> nouveau chemin à prendre en compte
SET @OSCMD = '"' + @OSCMD +'"';
SELECT @OSCMD = '"' + physical_name +'" ' + @OSCMD
FROM   sys.master_files
WHERE  database_id = DB_ID('MA_BASE'); --> nom de la base à déplacer
SELECT 'COPY /Y ' + @OSCMD;
```

Qui fournit une seule commande pour déplacer tous les fichiers.

4 – rattachement de la base

Le rattachement d'une base est en fait une création de base et par conséquent se fait à l'aide de la commande CREATE DATABASE avec l'option FOR ATTACH.
Exemple :

```
CREATE DATABASE [DB_MABASE]
   ON (FILENAME = 'D:\DATABASES\DB_MABASE.mdf') ,
      (FILENAME = 'D:\DATABASES\DB_MABASE_log.ldf')
   FOR ATTACH;
```

Nous vous déconseillons le mode ATTACH_REBUILD_LOG car cela reconstruit un nouveau journal et par conséquent certaines options de réparation ne seront pas possible.

Pour préparer une telle commande, vous pouvez utiliser la requête suivante dans l'étape 1 :

```
DECLARE @NEWPATH NVARCHAR(256);
SET @NEWPATH = N'D:\DATABASES\'; --> nouveau chemin à prendre en compte
DECLARE @SQLCMD NVARCHAR(max);

WITH
T0 AS (SELECT DB_NAME(database_id) AS DATABASE_NAME,
              REVERSE(LEFT(REVERSE(physical_name),
                      CHARINDEX('\', REVERSE(physical_name)) - 1))
```

```
                AS A_FILE,
              ROW_NUMBER() OVER(ORDER BY file_id) AS N,
              COUNT(*) OVER() AS P
       FROM   sys.master_files
       WHERE  database_id = DB_ID('MA_BASE')), --> nom de la base à déplacer
T1 AS (SELECT DISTINCT N'CREATE DATABASE [' + DATABASE_NAME + N'] ON ' AS SQLCMD_HEAD
       FROM   T0),
T2 AS (SELECT CAST(N' (FILENAME = ''' + @NEWPATH + A_FILE + N''') ' AS NVARCHAR(max))
                AS SQLCMD, N, P
       FROM   T0
       WHERE  N = 1
       UNION ALL
       SELECT SQLCMD + N', (FILENAME = ''' + @NEWPATH + A_FILE + N''') ', T0.N, T0.P
       FROM   T0
              INNER JOIN T2 ON T2.N + 1= T0.N)
SELECT SQLCMD_HEAD + N' ' + SQLCMD + N' FOR ATTACH;'
FROM   T1
       CROSS JOIN T2
WHERE  N = P;
```

ANNEXE 2 – Création d'un groupe de fichier et d'un fichier dans le nouveau groupe

Pour mettre en place un nouvel espace de stockage, vous devez créer un groupe de fichier et au moins un fichier dans ce nouveau groupe de fichier.
Les scripts SQL pour ce faire passent par toutes par la commande ALTER DATABASE.

Ajouter un groupe de fichier à une base.

Exemple : base de nom MA_BASE, groupe de fichier à créer de nom FG_NEW :

```
USE master;
GO
ALTER DATABASE MA_BASE
   ADD FILEGROUP FG_NEW;
GO
```

Ajouter un fichier à un groupe de fichier

Exemple : ajout d'un fichier de nom DATA.ndf, situé sur D:\DATABASE à ce nouveau groupe de fichiers en lui donnant le nom logique MA_BASE_DATA :

```
USE master;
GO
ALTER DATABASE MA_BASE
   ADD FILE
      (NAME = 'MA_BASE_DATA',
          FILENAME = 'D:\DATABASE\DATA.ndf'
      )
   TO FILEGROUP FG_NEW;
GO
```

Vous pouvez préciser la taille et le pas d'incrément du fichier à l'aide des options SIZE ert FILEGROWTH. Exemple :

```
USE master;
GO
ALTER DATABASE MA_BASE
   ADD FILE
      (NAME = 'MA_BASE_DATA',
          FILENAME = 'D:\DATABASE\DATA.ndf',
          SIZE = 10 GB,
          FILEGROWTH = 50 MB
      )
   TO FILEGROUP FG_NEW;
GO
```