

Import de données : désactivation temporaire des index

Accélérer les insertions en désactivant les index



MVP SQL Server

Expert langage SQL, SGBDR, modélisation de données Enseigne à l'ISEN Toulon et aux Arts & Métiers

par **Frédéric Brouard** ([SQLpro](#))

Date de publication : Avril 2006

Dernière mise à jour :

Des performances exceptionnelles peuvent être obtenues lors des insertions massives de données telles que lors des imports à condition de minimiser les index et en particulier de supprimer tous les index qui ne sont pas fondamentalement nécessaires à assurer la cohérence des données.

Cet article présente une technique qui automatise la désactivation temporaire des index et sa réactivation afin de satisfaire un tel besoin.

Préambule

1 - Le Test

2 - La Solution

2.1 - méta données des index

2.2 - stockage des informations sur les index

3 - Utilisation

De plus amples informations vous sont nécessaires ?

Préambule

Disons le tout de suite, cet article ne concerne que les applications développées en MS SQL Server version 7 ou 2000. En effet depuis la version 2005, MS SQL Server possède un ordre qui rend caduque cet article et qui permet de désactiver un index quel qu'il soit :

```
ALTER INDEX { <nom_index> | ALL }
  ON <nom_objet>
  DISABLE

-- suivi au choix d'un

ALTER INDEX ... REBUILD
CREATE INDEX ... WITH DROP_EXISTING
```

1 - Le Test

Pour bien se rendre compte de l'intérêt d'une telle manœuvre, je vous propose de tester le temps d'exécution d'une insertion de données dans une table test comportant différents index. Puis de renouveler cette opération en supprimant préalablement les index puis en les remettant à la fin de l'opération. Bien entendu, dans ce dernier cas, notre mesure de temps comportera les actions de suppression et de remise en place des index...

Voici la table avec laquelle nous allons faire nos tests : la table test

```
CREATE TABLE T_TEST_INSERT_TIS
(TIS_ID          INTEGER NOT NULL
-- son index primaire (clef)
  CONSTRAINT PK_TIS PRIMARY KEY IDENTITY,
TIS_GUID         UNIQUEIDENTIFIER DEFAULT NEWID(),
TIS_DATA1        VARCHAR(32),
TIS_DATEHEURE    DATETIME,
TIS_DATA2        VARCHAR(64),
TIS_REEL         FLOAT,
TIS_DATA3        VARCHAR(128))
GO

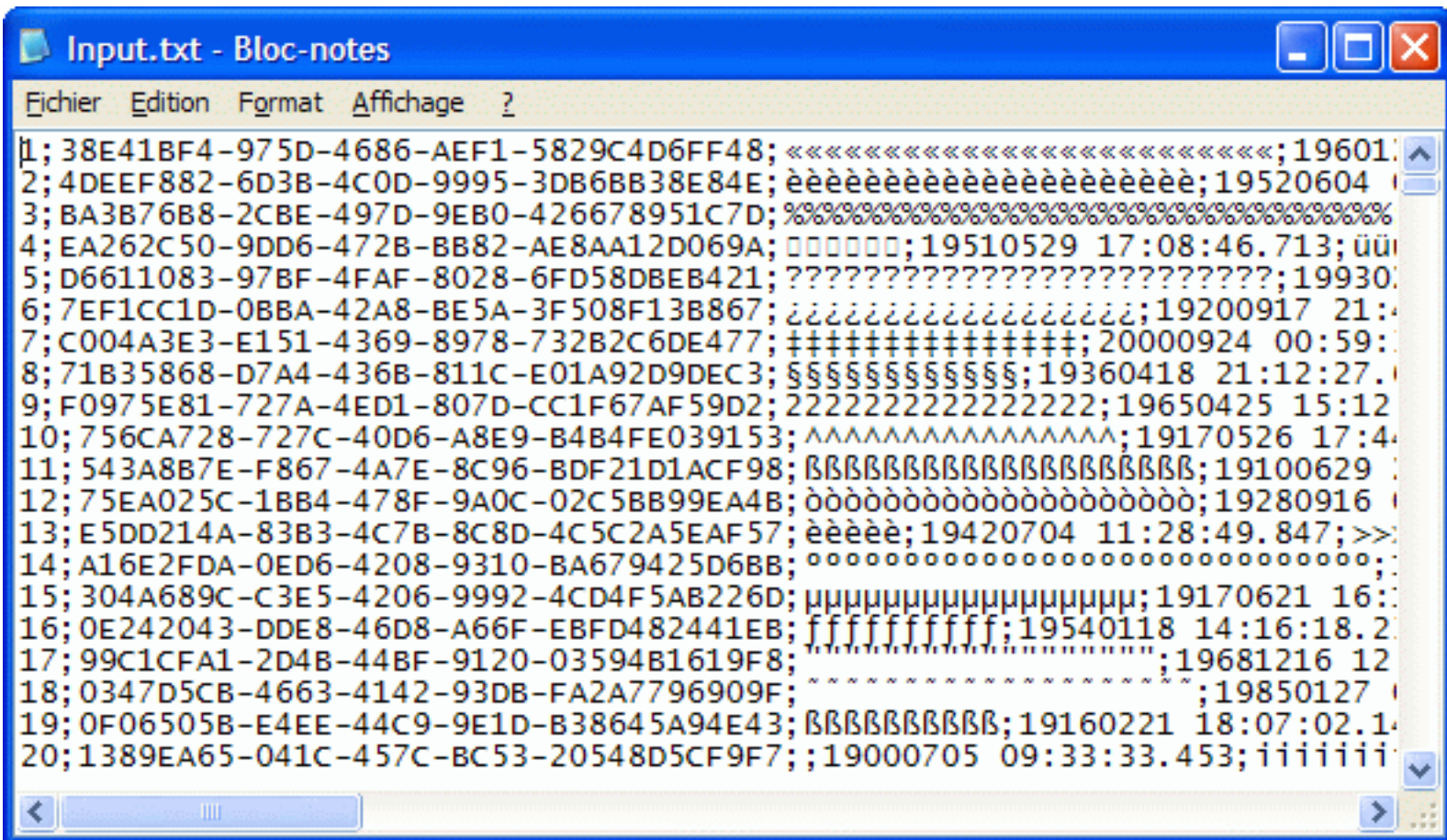
-- les index secondaires:

-- sur un DATETIME
CREATE INDEX X_TIS1 ON T_TEST_INSERT_TIS (TIS_DATEHEURE)
GO

-- sur deux colonnes VARCHAR
CREATE INDEX X_TIS2 ON T_TEST_INSERT_TIS (TIS_DATA1, TIS_DATA2)
GO

-- sur un réel
CREATE INDEX X_TIS3 ON T_TEST_INSERT_TIS (TIS_REEL)
GO
```

Le fichier que nous allons intégrer dans cette table comporte 100 000 lignes et "pèse" 19 218 432 octets.



Voici les conditions du test et les résultats sur un PC avec XPpro doté de 2 Go de RAM :

```

SET STATISTICS TIME ON

BULK INSERT T_TEST_INSERT_TIS
FROM 'C:\input.txt'
WITH (FIELDTERMINATOR = ';', ROWTERMINATOR = '\n')

SET STATISTICS TIME OFF
    
```

```

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

(100000 ligne(s) affectée(s))

Temps d'analyse et de compilation de SQL Server :
    temps UC = 0 ms, temps écoulé = 21 ms.

Temps d'exécution de SQL Server :
    Temps UC = 8063 ms, temps coulé = 32918 ms.

Temps d'exécution de SQL Server :
    Temps UC = 8063 ms, temps coulé = 32919 ms.
    
```

Soit 8 secondes d'utilisation de l'UC et 32 secondes en tout.

Réalisons maintenant le même test (la table a été reconstruite à l'identique) avec suppression préalable des index et remise en place desdits index :

```
SET STATISTICS TIME ON

DROP INDEX T_TEST_INSERT_TIS.X_TIS1
PRINT 'DROP INDEX 1 EXECUTÉ'
DROP INDEX T_TEST_INSERT_TIS.X_TIS2
PRINT 'DROP INDEX 2 EXECUTÉ'
DROP INDEX T_TEST_INSERT_TIS.X_TIS3
PRINT 'DROP INDEX 3 EXECUTÉ'

BULK INSERT T_TEST_INSERT_TIS
FROM 'C:\input.txt'
WITH (FIELDTERMINATOR = ';')

PRINT 'BULK INSERT EXECUTÉ'

CREATE INDEX X_TIS1 ON T_TEST_INSERT_TIS (TIS_DATEHEURE)
PRINT 'CREATE INDEX 1 EXECUTÉ'
CREATE INDEX X_TIS2 ON T_TEST_INSERT_TIS (TIS_DATA1, TIS_DATA2)
PRINT 'CREATE INDEX 2 EXECUTÉ'
CREATE INDEX X_TIS3 ON T_TEST_INSERT_TIS (TIS_REEL)
PRINT 'CREATE INDEX 3 EXECUTÉ'

SET STATISTICS TIME OFF
```

```
Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.
DROP INDEX 1 EXECUTÉ

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.
DROP INDEX 2 EXECUTÉ

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.
DROP INDEX 3 EXECUTÉ
```

```
Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

(100000 ligne(s) affectée(s))

Temps d'analyse et de compilation de SQL Server :
    temps UC = 0 ms, temps écoulé = 32 ms.

Temps d'exécution de SQL Server :
    Temps UC = 2109 ms, temps coulé = 10151 ms.

Temps d'exécution de SQL Server :
    Temps UC = 2109 ms, temps coulé = 10155 ms.
BULK INSERT EXECUTÉ
```

```

Temps d'exécution de SQL Server :
  Temps UC = 0 ms, temps coulé = 0 ms.
Temps d'analyse et de compilation de SQL Server :
  temps UC = 31 ms, temps écoulé = 37 ms.

Temps d'exécution de SQL Server :
  Temps UC = 188 ms, temps coulé = 189 ms.

Temps d'exécution de SQL Server :
  Temps UC = 188 ms, temps coulé = 196 ms.
CREATE INDEX 1 EXECUTÉ

Temps d'exécution de SQL Server :
  Temps UC = 0 ms, temps coulé = 0 ms.
Temps d'analyse et de compilation de SQL Server :
  temps UC = 0 ms, temps écoulé = 1 ms.

Temps d'exécution de SQL Server :
  Temps UC = 1531 ms, temps coulé = 1619 ms.

Temps d'exécution de SQL Server :
  Temps UC = 1531 ms, temps coulé = 1634 ms.
CREATE INDEX 2 EXECUTÉ

Temps d'exécution de SQL Server :
  Temps UC = 0 ms, temps coulé = 0 ms.
Temps d'analyse et de compilation de SQL Server :
  temps UC = 0 ms, temps écoulé = 2 ms.

Temps d'exécution de SQL Server :
  Temps UC = 250 ms, temps coulé = 254 ms.

Temps d'exécution de SQL Server :
  Temps UC = 250 ms, temps coulé = 265 ms.
CREATE INDEX 3 EXECUTÉ

```

```

Temps d'exécution de SQL Server :
  Temps UC = 0 ms, temps coulé = 0 ms.

```

Traitement	UC (ms)	Coulé (ms)
Suppression des index	0	0
Bulk Insert	2 109	10 155
Reconstruction des index	1 969	2 095
Total	4 078	12 250

Nous avons donc économisé 50 % de l'UC et divisé presque par trois le temps global du traitement.

Poussons le test encore plus loin en désactivant tous les index, c'est à dire en retirant la clef primaire...

```

SET STATISTICS TIME ON

DROP INDEX T_TEST_INSERT_TIS.X_TIS1
PRINT 'DROP INDEX 1 EXECUTÉ'
DROP INDEX T_TEST_INSERT_TIS.X_TIS2
PRINT 'DROP INDEX 2 EXECUTÉ'
DROP INDEX T_TEST_INSERT_TIS.X_TIS3
PRINT 'DROP INDEX 3 EXECUTÉ'
ALTER TABLE T_TEST_INSERT_TIS
DROP CONSTRAINT PK_TIS
PRINT 'DROP PK'

```

```
BULK INSERT T_TEST_INSERT_TIS
FROM 'C:\input.txt'
WITH (FIELDTERMINATOR = ';')

PRINT 'BULK INSERT EXECUTÉ'

ALTER TABLE T_TEST_INSERT_TIS
ADD CONSTRAINT PK_TIS PRIMARY KEY (TIS_ID)
PRINT 'ADD PK'
CREATE INDEX X_TIS1 ON T_TEST_INSERT_TIS (TIS_DATEHEURE)
PRINT 'CREATE INDEX 1 EXECUTÉ'
CREATE INDEX X_TIS2 ON T_TEST_INSERT_TIS (TIS_DATA1, TIS_DATA2)
PRINT 'CREATE INDEX 2 EXECUTÉ'
CREATE INDEX X_TIS3 ON T_TEST_INSERT_TIS (TIS_REEL)
PRINT 'CREATE INDEX 3 EXECUTÉ'

SET STATISTICS TIME OFF
```

```
Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.
DROP INDEX 1 EXECUTÉ

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.
DROP INDEX 2 EXECUTÉ

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.
DROP INDEX 3 EXECUTÉ

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 1 ms.
DROP PK

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.

(100000 ligne(s) affectée(s))

Temps d'analyse et de compilation de SQL Server :
    temps UC = 15 ms, temps écoulé = 83 ms.

Temps d'exécution de SQL Server :
    Temps UC = 1938 ms, temps coulé = 2602 ms.

Temps d'exécution de SQL Server :
    Temps UC = 1953 ms, temps coulé = 2636 ms.
BULK INSERT EXECUTÉ

Temps d'exécution de SQL Server :
    Temps UC = 0 ms, temps coulé = 0 ms.
Temps d'analyse et de compilation de SQL Server :
    temps UC = 0 ms, temps écoulé = 1 ms.
```

Temps d'exécution de SQL Server :
Temps UC = 219 ms, temps coulé = 801 ms.

Temps d'exécution de SQL Server :
Temps UC = 219 ms, temps coulé = 855 ms.

ADD PK

Temps d'exécution de SQL Server :
Temps UC = 0 ms, temps coulé = 4 ms.

Temps d'analyse et de compilation de SQL Server :
temps UC = 0 ms, temps écoulé = 1 ms.

Temps d'exécution de SQL Server :
Temps UC = 175 ms, temps coulé = 175 ms.

Temps d'exécution de SQL Server :
Temps UC = 188 ms, temps coulé = 183 ms.

CREATE INDEX 1 EXECUTÉ

Temps d'exécution de SQL Server :
Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'analyse et de compilation de SQL Server :
temps UC = 0 ms, temps écoulé = 1 ms.

Temps d'exécution de SQL Server :
Temps UC = 1546 ms, temps coulé = 1572 ms.

Temps d'exécution de SQL Server :
Temps UC = 1546 ms, temps coulé = 1582 ms.

CREATE INDEX 2 EXECUTÉ

Temps d'exécution de SQL Server :
Temps UC = 0 ms, temps coulé = 0 ms.

Temps d'analyse et de compilation de SQL Server :
temps UC = 0 ms, temps écoulé = 2 ms.

Temps d'exécution de SQL Server :
Temps UC = 235 ms, temps coulé = 253 ms.

Temps d'exécution de SQL Server :
Temps UC = 235 ms, temps coulé = 265 ms.

CREATE INDEX 3 EXECUTÉ

Temps d'exécution de SQL Server :
Temps UC = 0 ms, temps coulé = 0 ms.

Traitement	UC (ms)	Coulé (ms)
Suppression des index	0	10
Bulk Insert	1953	2636
Reconstruction des index	2188	2885
Total	4 141	5 522

Si le gain est nul en terme d'UC par rapport à la solution précédente (en fait notre clef primaire de type cluster à la chance de recevoir des données pré triées...), en temps coulé l'affaire se présente mieux. Mais cette solution de suppression des contraintes n'est pas toujours réaliste en solution d'exploitation. En effet si d'aventure la table cible est utilisée concurremment par des insertions ou des modifications, la suppression de la clef primaire comme de tout index unique risque d'entraîner des incohérences de données du fait que le garde fou constitué par l'impossibilité de violer les contraintes PRIMARY KEY et UNIQUE n'est plus actif.

La conclusion de l'ensemble de ces tests est simple : toute insertion massive de données est fortement pénalisée par le recalcul des index qui s'opère pour chaque ligne. Mon expérience m'a montré que le gain en exploitation sur de grandes tables avec de multiples index pouvait aller de 2 à ... plus de 10 ! D'où l'intérêt de mettre en #uvre une solution souple permettant de retirer et replacer les index sans même en connaître préalablement la composition...


```

        ON o.id = k.id
        and i.indid = k.indid
INNER JOIN dbo.syscolumns c
        ON k.colid = c.colid
        and o.id = c.id
INNER JOIN INFORMATION_SCHEMA.COLUMNS ISC
        ON u.name = ISC.TABLE_SCHEMA
        AND o.name = ISC.TABLE_NAME
        AND c.name = ISC.COLUMN_NAME
LEFT OUTER JOIN INFORMATION_SCHEMA.TABLE_CONSTRAINTS TCT
        ON u.name = TCT.CONSTRAINT_SCHEMA
        AND i.name = TCT.CONSTRAINT_NAME

WHERE i.status & 64 <> 64

ORDER BY IXD_SCHEMA_NAME, IXD_TABLE_NAME, IXD_INDEX_NAME, IXD_COL_IDX_ORDER

GO

```

2.2 - stockage des informations sur les index

Deux tables vont permettre de stocker, l'une, les informations sur les index, l'autre, quel a été le demandeur de la man#uvre :

```

-- table mère du processus
CREATE TABLE dbo.T_A_MAINTENANCE_PROCESS_MPC
(MPC_ID          INT NOT NULL IDENTITY CONSTRAINT PK_MPC PRIMARY KEY,
 MPC_ID_PRECEDENT INT NULL
  FOREIGN KEY REFERENCES T_A_MAINTENANCE_PROCESS_MPC (MPC_ID),
 MPC_SQL_USER    sysname DEFAULT CURRENT_USER,
 MPC_SYS_USER    sysname DEFAULT SYSTEM_USER,
 MPC_SPID        INT,
 MPC_DATE_TIME   DATETIME DEFAULT CURRENT_TIMESTAMP,
 MPC_HOST_NAME   NCHAR(128),
 MPC_APPLI       NCHAR(128),
 MPC_NETNAME     NCHAR(128),
 MPC_NETADR      NCHAR(12),
 MPC_PROCESS     VARCHAR(512),
 MPC_SUCCESS     BIT DEFAULT 0)
GO

-- table fille de détail du processus lié à la désindexation/réindexation
CREATE TABLE dbo.T_A_MAINTENANCE_INDEX_MIX
(MIX_ID          INT NOT NULL IDENTITY CONSTRAINT PK_MIX PRIMARY KEY,
 MPC_ID          INT NOT NULL
  FOREIGN KEY REFERENCES T_A_MAINTENANCE_PROCESS_MPC (MPC_ID),
 MIX_DATE_TIME   DATETIME DEFAULT CURRENT_TIMESTAMP,
 MIX_OPERATION    CHAR(5) CONSTRAINT CK_MIX_OPR
  CHECK (MIX_OPERATION IN ('DROP', 'RESET')),
 MIX_SCHEMA_NAME sysname,
 MIX_TABLE_NAME  sysname,
 MIX_INDEX_NAME  sysname,
 MIX_UNIQUE      BIT DEFAULT 0,
 MIX_CLUSTER     BIT DEFAULT 0,
 MIX_COL_ORDER_LIST VARCHAR(2160),
 MIX_INDEX_WITH  VARCHAR(200),
 MIX_NO_ROW_LOCK BIT DEFAULT 0,
 MIX_NO_PAGE_LOCK BIT DEFAULT 0,
 MIX_ON_FILE     sysname NULL,
 MIX_COMPLETE    BIT DEFAULT 0)
GO

```

La première table enregistre la demande de suppression des index. La seconde sert à décrire les index supprimés, donc ceux qui vont devoir être remis en place.

Pour alimenter ces tables, et opérer la man#uvre, deux procédures stockées sont créées. L'une P_A_MAINTENANCE_DROP_INDEXES retire les index d'une table appartenant à un schéma et au passage, notons que l'on à prévu de pouvoir recréer les index sur un nouveau groupe de fichiers si le besoin s'en fait sentir. L'autre P_MAINTENANCE_RESET_INDEXES les remets en place simplement en lui donnant en paramètre l'identifiant généré par la précédente, identifiant qui est en fait l'entrée (clef) de la table T_A_MAINTENANCE_PROCESS_MPC.

La procédure P_A_MAINTENANCE_DROP_INDEXES, utilise en outre une fonction SQL de nom F_A_INDEX_LIST_COL_ORDER qui permet d'obtenir la liste ordinale des colonnes composant l'index et le sens d'indexation (ASC ou DESC).

Voici le code Transact SQL de ces trois routines :

```
CREATE FUNCTION F_A_INDEX_LIST_COL_ORDER
(@SCHEMA_NAME sysname,
 @TABLE_NAME sysname,
 @INDEX_NAME sysname)
RETURNS VARCHAR(2160)
AS
BEGIN
DECLARE @RETVAL VARCHAR(2160)
SET @RETVAL = ''

SELECT @RETVAL = @RETVAL +
    c.name + ' ' +
    CASE
        WHEN INDEXKEY_PROPERTY
(o.id , i.indid , k.keyno , 'IsDescending' ) = 0 THEN 'ASC'
        WHEN INDEXKEY_PROPERTY
(o.id , i.indid , k.keyno , 'IsDescending' ) = 1 THEN 'DESC'
        WHEN INDEXKEY_PROPERTY
(o.id , i.indid , k.keyno , 'IsDescending' ) IS NULL THEN ''
    END + ', '

FROM sysindexes i
INNER JOIN sysobjects o
    ON i.id = o.id
INNER JOIN sysusers u
    ON o.uid = u.uid
INNER JOIN sysindexkeys k
    ON o.id = k.id
    and i.indid = k.indid
INNER JOIN syscolumns c
    ON k.colid = c.colid
    and o.id = c.id
INNER JOIN INFORMATION_SCHEMA.COLUMNS ISC
    ON u.name = ISC.TABLE_SCHEMA
    AND o.name = ISC.TABLE_NAME
    AND c.name = ISC.COLUMN_NAME

WHERE u.name = @SCHEMA_NAME
    AND o.name = @TABLE_NAME
    AND i.name = @INDEX_NAME

ORDER BY k.keyno

IF LEN(@RETVAL) >= 2
SET @RETVAL = SUBSTRING(@RETVAL, 1, LEN(@RETVAL) - 1)
```

```
RETURN @RETVAL

END

GO
```

```
CREATE PROCEDURE dbo.P_A_MAINTENANCE_DROP_INDEXES
    @SCHEMA_NAME sysname = 'dbo',
    @TABLE_NAME sysname,
    @CREATE_ON sysname = 'PRIMARY',
    @MPC_ID INT = NULL OUTPUT
AS
DECLARE @ROWS INT, @ERROR INT
SET @MPC_ID = NULL
-- RETURN NULL ON NULL INPUT (norme SQL)
IF @TABLE_NAME IS NULL OR @CREATE_ON IS NULL OR @SCHEMA_NAME IS NULL
BEGIN
    RAISERROR ('Entrée NULL détectée dans au moins un des paramètres de la procédure
        P_A_MAINTENANCE_DROP_INDEXES. Abandon immédiat', 16, 1)
    RETURN
END
-- schéma.table n'existe pas dans le catalogue présent
IF NOT EXISTS (SELECT *
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = @SCHEMA_NAME
        AND TABLE_NAME = @TABLE_NAME)
BEGIN
    RAISERROR ('Le couple schéma.table : %s.%s,
        n''existe pas dans le catalogue courant', 16, 1, @SCHEMA_NAME, @TABLE_NAME)
    RETURN
END
-- vérification file group
IF @CREATE_ON <> 'PRIMARY'
    IF NOT EXISTS (SELECT *
        FROM sysfilegroups
        WHERE groupname = @CREATE_ON)
BEGIN
    RAISERROR ('Le groupe de fichier %s, n''existe pas dans la liste des groupe
        de fichiers pour cette base de données', 16, 1, @CREATE_ON)
    RETURN
END
-- les tables de maintenance existent-elles dans la base ?
DECLARE @T1 sysname, @T2 sysname, @SC sysname
SET @T1 = 'T_A_MAINTENANCE_INDEX_MIX'
SET @T2 = 'T_A_MAINTENANCE_PROCESS_MPC'
SET @SC = 'dbo'
IF NOT EXISTS (SELECT *
    FROM INFORMATION_SCHEMA.TABLES
    WHERE TABLE_SCHEMA = @SC
        AND TABLE_NAME = @T1
        AND EXISTS (SELECT *
            FROM INFORMATION_SCHEMA.TABLES
            WHERE TABLE_SCHEMA = @SC
                AND TABLE_NAME = @T2))
BEGIN
    RAISERROR ('Au moins l''une des tables de travail (%s.%s, %s.%s)_
        n''existe pas dans la liste objets du
        catalogue', 16, 1, @SC, @T1, @SC, @T2)
    RETURN
END
```

```

END

BEGIN TRANSACTION

-- insertion des informations de maintenance
INSERT INTO dbo.T_A_MAINTENANCE_PROCESS_MPC (MPC_SPID, MPC_HOST_NAME, MPC_APPLI,
MPC_NETNAME, MPC_NETADR, MPC_PROCESS )
SELECT spid,      hostname,      program_name, nt_username, net_address,
'Suppression des index non cluster et non unique
(dont clef) table '+' @SCHEMA_NAME+'.'+'@TABLE_NAME
FROM      master.dbo.sysprocesses p
WHERE     p.spid = @@SPID
IF @@ERROR <> 0
    GOTO LBL_ERROR

-- reprise de l'indentifiant auto généré pour report dans table fille
SET @MPC_ID = @@IDENTITY

-- insertion des informations sur les index
INSERT INTO dbo.T_A_MAINTENANCE_INDEX_MIX (MPC_ID, MIX_OPERATION,
MIX_SCHEMA_NAME, MIX_TABLE_NAME, MIX_INDEX_NAME, _
MIX_UNIQUE, MIX_CLUSTER, MIX_INDEX_WITH, MIX_COL_ORDER_LIST,
MIX_NO_ROW_LOCK, MIX_NO_PAGE_LOCK, MIX_ON_FILE)
SELECT @MPC_ID, 'DROP', @SCHEMA_NAME, @TABLE_NAME, i.name,
INDEXPROPERTY(o.id, i.name, 'IsUnique'),
INDEXPROPERTY(o.id, i.name, 'IsClustered'),
-- propriété PAD
CASE
    WHEN INDEXPROPERTY(o.id, i.name, 'IsPadIndex') = 1 THEN ' PAD_INDEX '
    ELSE ''
END +
-- propriété FILL FACTOR
CASE
    WHEN INDEXPROPERTY(o.id, i.name, 'IndexFillFactor') > 0
        THEN 'FILLFACTOR = ' + CAST(INDEXPROPERTY
(o.id, i.name, 'IndexFillFactor') AS VARCHAR(3))
    ELSE NULL
END,
dbo.F_A_INDEX_LIST_COL_ORDER (@SCHEMA_NAME, @TABLE_NAME, i.name),
INDEXPROPERTY(o.id, i.name, 'IsRowLockDisallowed'),
INDEXPROPERTY(o.id, i.name, 'IsPageLockDisallowed'),
NULLIF(@CREATE_ON, 'PRIMARY')
FROM      dbo.sysindexes i
INNER JOIN dbo.sysobjects o
    ON i.id = o.id
INNER JOIN dbo.sysusers u
    ON o.uid = u.uid
WHERE     i.status & 64 <> 64 -- sauf les index "stat"
AND     u.name = @SCHEMA_NAME
AND     o.name = @TABLE_NAME
AND     INDEXPROPERTY(o.id, i.name, 'IsUnique') <> 1
AND     INDEXPROPERTY(o.id, i.name, 'IsClustered') <> 1
SELECT @ERROR = @@ERROR, @ROWS = @@ROWCOUNT
IF @ERROR <> 0
    GOTO LBL_ERROR

-- aucun index secondaire n'est présent dans la table
IF @ROWS = 0
BEGIN
    SET @MPC_ID = NULL
    GOTO LBL_ERROR
RETURN
END

-- suppression des index
DECLARE @SQL VARCHAR(8000)

SET @SQL = 'DROP INDEX'

```

```

SELECT @SQL = @SQL + ' ' + MIX_SCHEMA_NAME + '.' +
  MIX_TABLE_NAME + '.' + MIX_INDEX_NAME + ','
FROM   dbo.T_A_MAINTENANCE_INDEX_MIX
WHERE  MPC_ID = @MPC_ID
      AND MIX_UNIQUE = 0
      AND MIX_CLUSTER = 0
IF @@ERROR <> 0
  GOTO LBL_ERROR

SET @SQL = SUBSTRING(@SQL, 1, LEN(@SQL) -1)

EXEC (@SQL)
IF @@ERROR <> 0
  GOTO LBL_ERROR

UPDATE dbo.T_A_MAINTENANCE_INDEX_MIX
SET    MIX_DATE_TIME = CURRENT_TIMESTAMP,
      MIX_COMPLETE = 1
WHERE  MPC_ID = @MPC_ID

UPDATE dbo.T_A_MAINTENANCE_PROCESS_MPC
SET    MPC_SUCCESS = 1
WHERE  MPC_ID = @MPC_ID

COMMIT TRANSACTION

RETURN

LBL_ERROR:
ROLLBACK TRANSACTION
RAISERROR('Maintenance d''index : opération inachevée. Veuillez prendre
connaissance des informations des tables dbo.T_A_MAINTENANCE_PROCESS_MPC (et
dbo.T_A_MAINTENANCE_INDEX_MIX) pour les entrées
d''identifiant %i.', 18, 1, @MPC_ID) WITH LOG

GO

```

```

CREATE PROCEDURE P_MAINTENANCE_RESET_INDEXES
  @MPC_ID INTEGER
AS
-- RETURN NULL ON NULL INPUT (norme SQL)
IF @MPC_ID IS NULL
BEGIN
  RAISERROR ('Entrée NULL détectée dans au moins un des paramètres de la
procédure P_MAINTENANCE_RESET_INDEXES. Abandon immédiat', 16, 1)
  RETURN
END

-- les tables de maintenance existent-elles dans la base ?
DECLARE @T1 sysname, @T2 sysname, @SC sysname
SET @T1 = 'T_A_MAINTENANCE_INDEX_MIX'
SET @T2 = 'T_A_MAINTENANCE_PROCESS_MPC'
SET @SC = 'dbo'
IF NOT EXISTS (SELECT *
  FROM   INFORMATION_SCHEMA.TABLES
  WHERE  TABLE_SCHEMA = @SC
        AND TABLE_NAME = @T1
        AND EXISTS (SELECT *
  FROM   INFORMATION_SCHEMA.TABLES
  WHERE  TABLE_SCHEMA = @SC
        AND TABLE_NAME = @T2))
BEGIN
  RAISERROR ('Au moins l''une des tables de travail (%s.%s, %s.%s) n''existe pas
dans la liste objets du catalogue', 16, 1, @SC, @T1, @SC, @T2)

```

```

RETURN
END

-- pas d'entrée dans la table T_A_MAINTENANCE_PROCESS_MPC
IF NOT EXISTS(SELECT *
              FROM   dbo.T_A_MAINTENANCE_PROCESS_MPC
              WHERE  MPC_ID = @MPC_ID)
BEGIN
    RAISERROR ('pas d'entrée pour la clef de valeur %i dans la
              table T_A_MAINTENANCE_PROCESS_MPC. Abandon immédiat', 16, 1, @MPC_ID)
    RETURN
END

-- pas d'entrée dans la table T_A_MAINTENANCE_INDEX_MIX
IF NOT EXISTS(SELECT *
              FROM   dbo.T_A_MAINTENANCE_INDEX_MIX
              WHERE  MPC_ID = @MPC_ID)
BEGIN
    RAISERROR ('pas d'entrée pour la clef étrangère de valeur %i
              dans la table T_A_MAINTENANCE_INDEX_MIX. Abandon immédiat', 16, 1, @MPC_ID)
    RETURN
END

-- informer la table dbo.T_A_MAINTENANCE_PROCESS_MPC avec chainage
DECLARE @MPC_ID_NEW INT

INSERT INTO dbo.T_A_MAINTENANCE_PROCESS_MPC
           (MPC_ID_PRECEDENT, MPC.MPC_SPID, MPC_HOST_NAME, MPC_APPLI,
            MPC_NETNAME, MPC_NETADR, MPC_PROCESS
           )
SELECT DISTINCT MPC.MPC_ID,          spid,          hostname,          program_name,
nt_username, net_address, 'Réindexation de la table ' + MIX_SCHEMA_NAME + '.' + MIX_TABLE_NAME
FROM   dbo.T_A_MAINTENANCE_PROCESS_MPC MPC
       INNER JOIN dbo.T_A_MAINTENANCE_INDEX_MIX MIX
                ON MPC.MPC_ID = MIX.MPC_ID
       CROSS JOIN master.dbo.sysprocesses s
WHERE  s.spid = @@SPID
       AND MPC.MPC_ID = @MPC_ID
IF @@ERROR <> 0
    GOTO LBL_ERROR

SET @MPC_ID_NEW = @@IDENTITY

DECLARE @SQL NVARCHAR(4000)
DECLARE @MIX_ID INT
DECLARE @NO_PG_LCK BIT, @NO_RW_LCK BIT
DECLARE @IDX_COMPOSITE_NAME NVARCHAR(257)
DECLARE @I INT
SET @I = 256

-- au travail, sans curseur
WHILE EXISTS(SELECT *
            FROM   dbo.T_A_MAINTENANCE_INDEX_MIX
            WHERE  MPC_ID = @MPC_ID
            AND    MIX_OPERATION = 'DROP')
BEGIN
    -- sélection du premier index à remettre
    SELECT TOP 1 @MIX_ID = MIX_ID, @NO_PG_LCK = MIX_NO_PAGE_LOCK, @NO_RW_LCK = MIX_NO_ROW_LOCK,
               @IDX_COMPOSITE_NAME = '[' + MIX_TABLE_NAME + '].[' + MIX_INDEX_NAME + ']',
               @SQL = 'CREATE NONCLUSTERED INDEX ' + MIX_INDEX_NAME +
                    ' ON [' + MIX_SCHEMA_NAME + '].[' + MIX_TABLE_NAME + ']' +
                    ' (' + MIX_COL_ORDER_LIST + ') ' + COALESCE(' WITH ' + MIX_INDEX_WITH + ' ', '') +
                    ' ON ' + COALESCE(MIX_ON_FILE, 'PRIMARY')
    FROM   dbo.T_A_MAINTENANCE_INDEX_MIX
    WHERE  MPC_ID = @MPC_ID
           AND MIX_OPERATION = 'DROP'

    -- exécution de la réindexation

```

```
EXEC (@SQL)
IF @@ERROR = 0
-- succès on informe avec complète = 1
UPDATE dbo.T_A_MAINTENANCE_INDEX_MIX
SET     MIX_OPERATION = 'RESET',
        MIX_COMPLETE  = 1,
        MIX_DATE_TIME = CURRENT_TIMESTAMP
WHERE  MIX_ID = @MIX_ID
ELSE
-- échec on informe avec complète = 0
UPDATE dbo.T_A_MAINTENANCE_INDEX_MIX
SET     MIX_OPERATION = 'RESET',
        MIX_COMPLETE  = 0,
        MIX_DATE_TIME = CURRENT_TIMESTAMP
WHERE  MIX_ID = @MIX_ID

-- si @NO_PG_LCK = 1 remplacer l'interdiction d'utiliser le verrous de page dans l'index
IF @NO_PG_LCK = 1
EXEC sp_indexoption @IDX_COMPOSITE_NAME, 'DisallowPageLocks' , '1'
-- si @NO_RW_LCK = 1 remplacer l'interdiction d'utiliser le verrous de ligne dans l'index
IF @NO_RW_LCK = 1
EXEC sp_indexoption @IDX_COMPOSITE_NAME, 'DisallowRowLocks' , '1'

-- a ton bouclé indéfiniment ? (max 256 index)
SET @I = @I - 1
IF @I = 0
BREAK

END

-- tous les index ont-ils été recréés ?
IF EXISTS (SELECT *
           FROM   dbo.T_A_MAINTENANCE_INDEX_MIX
           WHERE  MPC_ID = @MPC_ID
           AND    MIX_COMPLETE = 0)
-- non : il en existe au moins 1 qui n'a pas été remis
GOTO LBL_ERROR

-- oui : on met les pendules à l'heure dans la table dbo.T_A_MAINTENANCE_PROCESS_MPC
UPDATE dbo.T_A_MAINTENANCE_PROCESS_MPC
SET     MPC_SUCCESS = 1
WHERE  MPC_ID = @MPC_ID_NEW
RETURN

LBL_ERROR:
RAISERROR('Maintenance d''index : opération inachevée.
          Veuillez prendre connaissance des informations des tables
          dbo.T_A_MAINTENANCE_PROCESS_MPC (et dbo.T_A_MAINTENANCE_INDEX_MIX) pour
          les entrées d''identifiant %i et %i', 18, 1, @MPC_ID, @MPC_ID_NEW) WITH LOG

GO
```

3 - Utilisation

Dans notre exemple test, cet outil s'utilise comme suit :

```
DECLARE @MPCID INT

EXEC dbo.P_A_MAINTENANCE_DROP_INDEXES
        @SCHEMA_NAME = 'dbo',
        @TABLE_NAME = 'T_TEST_INSERT_TIS',
        @CREATE_ON = 'PRIMARY',
        @MPC_ID = @MPCID output

BULK INSERT T_TEST_INSERT_TIS
FROM 'C:\input.txt'
WITH (FIELDTERMINATOR = ';', ROWTERMINATOR = '\n')

EXEC dbo.P_MAINTENANCE_RESET_INDEXES @MPCID
```

Traitement	UC (ms)	Coulé (ms)
Suppression des index	0	7
Bulk Insert	2 125	8881
Reconstruction des index	2000	2378
Total	4 125	11 259

On constate que le temps rajouté par ce traitement générique de collecte des informations d'index, et traitement générique est parfaitement négligeable.

De plus amples informations vous sont nécessaires ?

Venez en discuter sur le forum public français de Microsoft SQL Server :

news.msnews.microsoft.com/microsoft.public.fr.sqlserver

Ou sur le forum SQL Server de developpez.com :

<http://www.developpez.net/forums/forumdisplay.php?f=49>

