# Volume

HELA - CATEGORIE ECONOMIQUE - ISAT

Section Informatique de Gestion - 2eme Annee

SQL 2008 Serveur

Volume1 : Creation et Mise en service d'une basedeDonneeSQL 2008 Serveur

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

## Introduction

**Rappels** **sur** **le** **stockage** **des** **donnees**

# Partie

Le stockage des donnees represente un probleme aussi vieux que l'informatique. Au fur et a mesure de l'evolution des capacites techniques du materiel et du volume des donnees manipulees, la fa�on de stocker et d'organiser les donnees a lui aussi evolue.

Dans le cadre d'une application de gestion, toutes les categories de donnees ne sont pas concernees de la meme fa�on par ces problemes d'organisation.

#### Les differentes categories de donnees

Dans un premier temps, il convient de definir la categorie des donnees. Cette categorisation est issue de quelques questions simples :

* + A quoi servent les donnees ?
  + Combien de temps est il necessaire de conserver ces donnees ?

#### Les donnees de base

Ce type de donnees est au coeur de tout systeme d'information. Il s'agit des donnees a partir desquelles il est possible de travailler. Ainsi, dans le cadre d'une gestion commerciale, les donnees de bases seront les informations sur les clients et sur les produits. Les donnees de ce type sont aussi volumineuses que possible et bien entendu elles ont une duree de vie tres longue. Comme ce sont des donnees de base, elles devront etre accessibles facilement et rapidement.

#### Les donnees de mouvement

Ces donnees sont generees a partir des donnees de base. Contrairement a ces dernieres, leur duree de vie sera limitee mais leur volume sera beaucoup plus important. Par exemple, toujours dans le cadre d'une gestion commerciale, les informations relatives a chaque commande sont considerees comme des donnees de mouvement. Le volume est important car l'entreprise compte bien que chaque client passe plusieurs commandes au cours d'une

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

meme annee comptable. Par contre, la duree de vie de ces informations est bien moindre. En effet, il n'est pas necessaire de conserver ce type d'informations plusieurs annees en ligne, mais plutot sur un support d'archivage autre et moins coOteux.

#### Les donnees de travail

Il s'agit de donnees generees dans un but precis, avec un volume parfois important mais une duree de vie tres courte. Des que le travail est realise il n'est pas necessaire de conserver ces donnees. Ainsi, par exemple, les donnees extraites de la base et qui vont servir a la realisation de graphiques sont a ranger dans cette categorie. Des que les graphiques sont realises, il n'est plus necessaire de conserver les donnees extraites de la base qui ont permis de les obtenir.

#### Les donnees d'archive

Il s'agit de donnees tres volumineuses et avec une duree de vie tres longue mais qui presentent la caracteristique de ne pas etre directement accessibles. Lorsqu'elles le sont, c'est uniquement en lecture. Par exemple, dans le cadre d'une application de gestion commerciale, il peut s'agir des donnees relatives aux annees comptables passees.

#### L'organisation des donnees

1. **Directe**

Cette organisation est sans doute la plus simple a utiliser. Les donnees sont enregistrees les unes a la suite des autres dans un fichier. Chaque ensemble de donnees possede une longueur fixe et les enregistrements sont stockes les uns derriere les autres. Ainsi, la connaissance de la longueur d'un enregistrement permet par simple calcul d'acceder directement au 10e m e enregistrement.

Ce type d'organisation est coOteux en espace disque et ne permet pas d'extraire facilement les informations sur des criteres autres que leur position dans l'ordre d'enregistrement.

#### Sequentielle

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Avec l'organisation sequentielle les donnees sont enregistrees les unes a la suite des autres. Un caractere special est utilise pour marquer la separation entre les differents champs tandis qu'un autre est utilise pour marquer la fin de chaque enregistrement. Les caracteres retenus sont couramment la virgule (,) et la fin de ligne (CR). Les fichiers qui retiennent ces separateurs sont alors decrits comme des fichiers CSV (*Comma* *Separated* *Values*).

Ce type d'organisation permet d'optimiser l'espace de stockage utilise et resoud ainsi l'un des problemes majeurs des fichiers avec un acces direct. Par contre, comme pour l'organisation directe, lorsque l'on recherche des donnees repondant a des criteres de selection bien precis il est necessaire de parcourir l'ensemble des donnees, ce qui s'avere d'autant plus long que le volume de donnees (nombre d'enregistrements) est important.

#### Sequentielle indexee

Les donnees sont toujours stockees au format sequentiel mais afin de permettre un acces plus rapide aux informations, des index peuvent etre definis pour chaque fichier. A l'interieur de ces index les donnees sont triees par ordre alphanumerique. Le parcours de l'index est realise de fa�on sequentielle et permet un acces direct aux informations stockees dans le fichier de donnees.

Le parcours de l'index, bien que sequentiel, est rapide car le volume de donnees manipule est faible. De plus, comme les donnees sont triees, il n'est pas necessaire de lire la totalite de l'index. Enfin, il est possible de definir plusieurs index sur un meme fichier de donnees. Par exemple, sur un fichier stockant des informations relatives aux clients, il est possible de definir un index sur les noms et un autre sur les villes.

Avec ce type d'organisation, la difficulte consiste a maintenir a jour les index lors des operations d'ajout, de suppression et de mise a jour. De plus, comme avec les organisations directe et sequentielle, les fichiers ne sont pas lies les uns aux autres et il n'existe pas de contexte de securite au niveau des donnees. Par exemple, rien ne s'oppose, au niveau des donnees, a la suppression d'un client meme s'il possede des commandes en cours.

De meme, toute personne en mesure de travailler avec les donnees, peut acceder a la totalite des donnees en lecture et en ecriture. Ces inconvenients posent plus de problemes avec l'organisation sequentielle indexee car des volumes de donnees important peuvent etre geres ainsi avec de nombreux utilisateurs connectes.

Cette solution sequentielle indexee a ete adoptee de fa�on massive pour des applications petites a moyennes car afin de faciliter les developpements, de nombreux langages de programmation proposaient un moteur de gestion de ce type d'organisation.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1. **Base** **de** **donnees** **hierarchique**

# Partie

Avec ces bases de donnees, les problemes de securite d'acces aux donnees ainsi que la liaison entre les donnees ont ete resolus. Par contre, chaque moteur a ete developpe de fa�on independante par les differents editeurs.

L'apprentissage du moteur est donc a recommencer a chaque fois que l'on developpe avec un nouveau moteur (le langage d'interrogation, l'API d'acces aux donnees). Ce a quoi il faut ajouter une organisation complexe des donnees.

Ces solutions hautement proprietaires sont souvent tres coOteuses pour l'entreprise qui les choisit.

#### Base de donnees relationnelle

Fondee sur une representation logique des donnees en respectant le modele relationnel, les bases de donnees relationnelles ont su s'imposer car elles s'appuient toutes sur le meme langage standardise et normalise qu'est le SQL.

#### La normalisation du schema relationnel

Lorsque le schema relationnel est defini afin de repondre a tous les besoins des utilisateurs, il est necessaire de le normaliser afin d'eviter toute redondance d'information ainsi que toute structure non conforme avec le modele relationnel.

Lorsque cette operation est realisee, le schema pourra alors etre denormalise bien que cette operation soit rarement la meilleure. Si le developpeur denormalise le schema, il doit egalement mettre en place l'ensemble du mecanisme qui permet de maintenir la coherence des donnees. En effet, le modele relationnel, et donc les SGBDR (Systeme de Gestion de Base de donnees Relationnelle), ne peuvent garantir la coherence des donnees que sur des modeles normalises.

Les formes normales permettent de s'assurer que le schema est bien conforme au modele relationnel. Il existe de fa�on theorique cinq formes normales, mais dans la pratique, seules les trois premieres sont appliquees. L'application des formes normales necessite de bien ma1triser le concept de dependance fonctionnelle. Une donnee depend fonctionnellement d'une autre lorsque la connaissance de la seconde permet de determiner la valeur de la

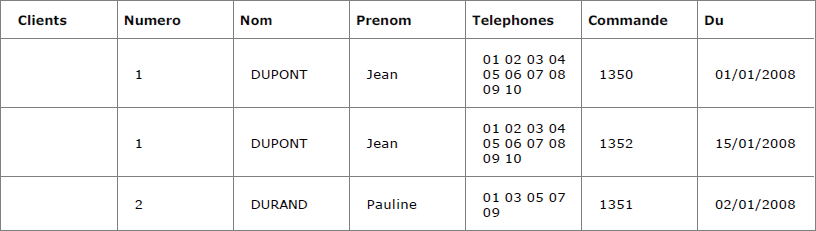
premiere. Par exemple, il est possible de dire que dans une application de gestion commerciale, il existe une dependance fonctionnelle entre un code TVA et le taux de TVA ou bien entre la reference d'un article et sa designation.

S Q L S E R V E U R 2 0 0 8

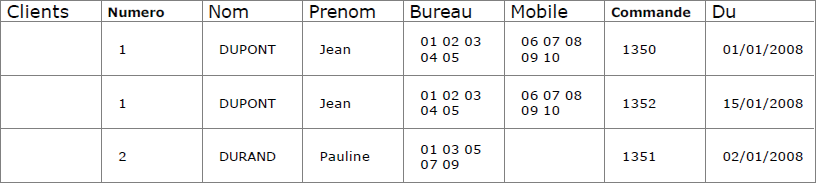
M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

**Premiere** **forme** **normale** **:** une table est dite en premiere forme normale lorsque toutes les colonnes contiennent des valeurs simples.

Par exemple, si une table des clients contient un champ Telephones dans lequel les differents numeros de telephone d'un client son stockes, alors cette table n'est pas en premiere forme normale. Il est alors necessaire de definir les colonnes Bureau et Mobile afin de mieux structurer les donnees.

La table presentee ci dessus ne respecte pas la premiere forme normale. Cette table respecte la premiere forme normale.



**Deuxieme** **forme** **normale** **:** une table est dite en deuxieme forme normale si elle est en premiere forme normale et si toutes les colonnes non cles dependent fonctionnellement de la cle primaire.

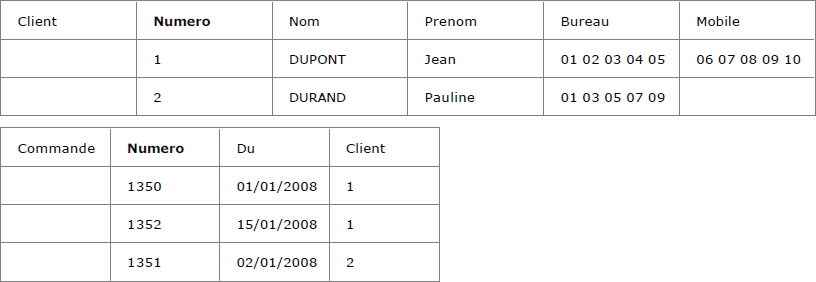
En reprenant l'exemple presente ci dessus, il est possible d'admettre dans un premier temps que la cle de la table des clients est composee des colonnes Numero et Commande.

Dans ce cas, les valeurs des colonnes Nom, Prenom, Bureau et Mobile dependent uniquement du numero tandis que la colonne Du est liee au numero de la commande. La table n'est donc pas en seconde forme normale. Il est donc necessaire de definir deux tables : clients et commandes.

S Q L S E R V E U R 2 0 0 8

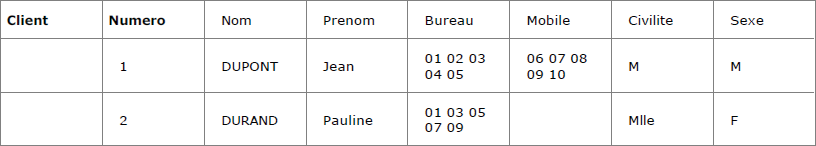
M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie



Les deux tables presentees ci dessus respectent la deuxieme forme normale.

**Troisieme** **forme** **normale** **:** une table est dite en troisieme forme normale si elle est en deuxieme forme normale et s'il n'existe pas de dependance fonctionnelle entre deux colonnes non cle.

Par exemple, si dans la table des clients les colonnes Civilite et Sexe sont ajoutees de la fa�on suivante :

Il est alors possible de dire qu'il existe une dependance fonctionnelle entre le sexe et la civilite. En effet, le fait de conna1tre la civilite (Mlle, Mme ou M) permet de deduire le sexe. La table des clients ne respecte donc pas la troisieme forme normale. La table des civilites est definie de fa�on a obtenir le schema suivant :



Les deux tables presentees ci dessus respectent la troisieme forme normale.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

#### Le modele relationnel

1. **Concepts** **et** **definitions**

Le modele relationnel repose sur des concepts de base simples (domaine, relation, attribut), auxquels s'appliquent des regles precises. La mise en oeuvre de la base est facilitee par un langage assertionnel (non procedural) simple, base sur une logique ensembliste.

C'est un ensemble de valeurs caracterise par un nom.

#### Cardinal

C'est le nombre d'elements d'un domaine.

*Exemple*

*Le* *dictionnaire* *des* *donnees* *de* *l'analyse* *d'une* *gestion* *commerciale* *peut* *comporter,* *entre* *autres,* *des* *specifications* *sur* *la* *gestion* *des* *etats* *de* *commande* *ou* *des* *numeros* *d'ordre* *a* *afficher.*

*Le* *modele* *relationnel* *les* *traduira* *de* *la* *maniere* *suivante* *:*

Etats des commandes = {"EC","LI","FA","SO"}Icardinal 4 Numeros d'ordre = {n I 1<=n<=9999}Icardinal 9999.

Le produit cartesien P entre plusieurs domaines D1, D2,..., Dn note P = D1 X D2 X ... X Dn est l'ensemble des n uplets(tuples) (d1, d2, ..., dn) ou chaque di est un element du domaine Di.

*Exemple*

*Si* *on* *veut* *gerer* *deux* *domaines* *(codes* *et* *taux},* *on* *pourra* *obtenir* *des* *2uplets* *composes* *d'un* *code* *et* *d'un* *taux.*

Codes = {1,2,3,4}

Taux de TVA = {0,5.5,19.6}

Codes X Taux de TVA ={(1,0),(1,5.5),(1,19.6), (2,0),(2,5.5),(2,19.6),(3,0),(3,5.5),(3,19.6),

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

(4,0),(4,5.5),(4,19.6)}

Une relation definie sur les domaines D1, D2,... , Dn est un sous ensemble du produit cartesien de ces domaines caracterise par un nom.

#### Attribut

C'est une colonne d'une relation caracterisee par un nom.

#### Degre

C'est le nombre d'attributs d'une relation.

*Exemple*

*Pour* *associer* *un* *seul* *taux* *par* *code,* *seuls* *trois* *2uplets* *doivent* *etre* *concernes.*

Relation TVA = {(1,0),(2,5.5),(3,19.6)}

Elle se fait sous forme de tableau (table), en extension : ou en comprehension :

TVA (CODE:codes, VALEUR:Taux de TVA) ou

TVA (CODE, VALEUR)

1. **Principales** **regles**

# Partie

Le modele relationnel gere donc un objet principal, la relation, associee aux concepts de domaine et d'attribut. Des regles s'appliquent a cette relation afin de respecter les contraintes liees a l'analyse.

Quelques unes de ces regles sont :

* + Toute valeur prise par un attribut doit appartenir au domaine sur lequel il est defini.
  + Tous les elements d'une relation doivent etre distincts.
  + Attribut ou ensemble d'attributs permettant de caracteriser de maniere unique chaque element de la relation.
  + Identifiant minimum d'une relation.
  + Autres identifiants de la relation.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

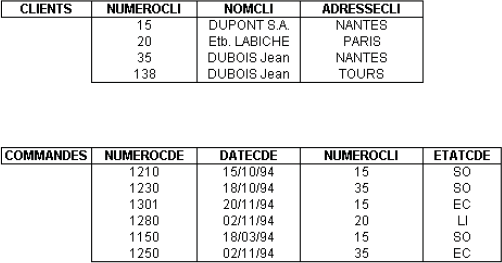
Cette regle impose qu'un attribut ou ensemble d'attributs d'une relation apparaisse comme cle primaire dans une autre relation.

Attribut ou ensemble d'attributs verifiant la regle d'integrite referentielle.

*Exemple*

*L'analyse* *d'une* *gestion* *commerciale* *nous* *impose* *de* *gerer* *des* *clients* *ayant* *des* *caracteristiques* *(Nom,* *adresse}* *et* *des* *commandes* *que* *passent* *ces* *clients.*

*On* *pourra* *proposer* *le* *modele* *suivant* *:*



CLIENTS (NUMEROCLI,NOMCLI,ADRESSECLI)

NUMEROCLI identifiant cle primaire de CLIENTS NOMCLI,ADRESSECLI identifiant cle secondaire de CLIENTS COMMANDES (NUMEROCDE,DATECDE,NUMEROCLI,ETATCDE)

NUMEROCDE identifiant cle primaire de COMMANDES NUMEROCLI cle etrangere de COMMANDES, referen�ant NUMEROCLI de CLIENTS

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Dans le modele relationnel, la notion de nullite est admise. C'est une valeur representant une information inconnue ou inapplicable dans une colonne. Elle est notee NULL.

Toute valeur participant a une cle primaire doit etre non NULL.

*Exemple*

*Dans* *la* *relation* *article,* *on* *admet* *que* *le* *prix* *ou* *le* *code* *TVA* *peuvent* *etre* *inconnus,* *mais* *la* *reference* *de* *l'article* *(cle* *primaire}* *doit* *etre* *renseignee.*



## Gerer une base de donnees

La creation et la maintenance d'une base de donnees SQL Server vont toucher des domaines d'activite varies, qui sont :

* + la gestion de l'espace de stockage,
  + la configuration de la base de donnees,
  + la gestion des objets de la base,
  + la traduction des contraintes de l'analyse,
  + la gestion de la securite d'acces,
  + les sauvegardes.

Certains de ces domaines touchent egalement l'administrateur et seront etudies ulterieurement. La gestion et la configuration de SQL Server peuvent se faire de deux manieres ; soit en Transact SQL, en interactif ou par script, soit par Microsoft SQL Server Management Studio, avec l'interface graphique.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Avec SQL Server 2008, il existe trois types de base de donnees :

* + Les bases OLTP (*OnLine* *Transaction* *Processing*), c'est a dire des bases qui vont supporter les transactions des utilisateurs. C'est ce type de base qui se trouve en production. Les principales caracteristiques de ce type de base sont que, malgre un volume de donnees consequent et de nombreux utilisateurs connectes, les temps de reponse doivent etre optimum. Heureusement, les utilisateurs travaillent sous forme de transaction courte et chaque transaction manipule un faible volume de donnees.
  + Les bases OLAP (*OnLine* *Analytical* *Processing*), c'est a dire une base qui va permettre de stocker un maximum d'informations afin de faire des requetes d'aide a la prise de decision. C'est le monde du decisionnel qui n'est pas aborde dans ce syllabus.
  + Les bases de type snapshot, qui sont des replications plus ou moins completes de la base d'origine afin d'acceder de fa�on rapide a des donnees eloignees par exemple.

#### Gerer l'espace de stockage

SQL Server utilise un ensemble de fichiers pour stocker l'ensemble des informations relatives a une base.

Il en existe un seul par base de donnees, c'est le point d'entree. Ce fichier porte l'extension

\*.mdf.

Il peut en exister plusieurs par base de donnees. Ils portent l'extension \*.ndf.

Ces fichiers (il peut y en avoir plusieurs) contiennent le journal des transactions et portent l'extension \*.ldf.

Il est possible de preciser un groupe de fichiers lors de la mise en place des fichiers. Ces groupes presentent l'avantage d'equilibrer les charges de travail sur les differents disques du systeme. Les donnees sont ecrites de fa�on equitable sur les differents fichiers du groupe.

#### Structure des fichiers de donnees

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

taille maximale d'une ligne est de 8060 octets hors types texte et image. Cette taille de 8 Ko autorise :

* + de meilleurs temps de reponse lors des operations de lecture/ecriture.
  + de supporter des lignes de donnees plus longues et donc de moins faire appel aux types texte et image.
  + une meilleure gestion des bases de tres grande taille.

Ces pages sont regroupees dans des extensions. Les extensions sont constituees de huit pages contigues (64 Ko).

Elles representent l'unite d'allocation d'espace aux tables et aux index. Pour eviter de perdre de la place disque, il existe deux types d'extensions :

* + Uniforme Reservee a un seul objet.
  + Mixte

Partagee par plusieurs objets, 8 au maximum.

Lorsqu'une table est creee, les pages sont allouees dans une extension mixte. Quand les donnees representent huit pages, alors une extension uniforme est allouee a la table.

#### Les fichiers de donnees

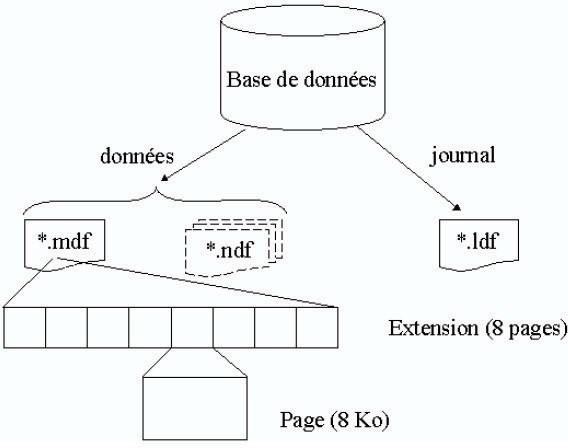
Les fichiers de donnees peuvent etre redimensionnes de fa�on dynamique ou manuelle. Lors de la creation du fichier, il faut preciser :

* + Le nom logique du fichier pour le manipuler avec le langage Transact SQL.
  + Le nom physique pour preciser l'emplacement du fichier.
  + Une taille initiale.
  + Une taille maximale.
  + Un pas d'incrementation.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie



#### Gerer l'objet DATABASE

Une DATABASE contient tous les autres objets :

* + Le catalogue de base de donnees.
  + Les objets utilisateurs (tables, defaults, views, rules, triggers, procedures).
  + Les index, les types de donnees, les contraintes.
  + Le journal de transactions.

La personne qui cree la base doit disposer des droits suffisants et devient le proprietaire de la base. SQL Server est capable de gerer 32767 bases de donnees.

Lors de leur creation, les fichiers sont initialises avec des zeros de fa�on a ecraser toutes les donnees deja presentes.

Cette operation entra1ne une surcharge de travail lors de la creation des fichiers mais elle permet d'optimiser les temps de reponse lorsque la base est en production.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Il est possible d'allouer de l'espace disque a la base sans qu'il soit initialise par des 0. Cette operation est identifiee

sous le terme initialisation instantanee. Les donnees anciennement presentes sur le disque sont ecrasees au fur et a mesure des besoins de la base.

SQL Server est capable d'utiliser des partitions brutes pour la creation des fichiers de bases de donnees. Cependant, dans la tres grande majorite des cas, la methode a privilegier est de creer les fichiers sur une partition NTFS.

En effet, l'utilisation d'une partition brute ne permet pas de signaler au systeme d'exploitation, et donc a l'administrateur, que cette partition est actuellement utilisee par SQL Server. L'espace, non utilise du point de vue systeme, peut ainsi etre facilement utilise pour etendre une partition ou bien creer une nouvelle partition. En utilisant des partitions brutes le risque de mauvaises manipulations augmente donc considerablement pour un gain qui n'est que peu significatif.

Les fichiers crees sur des partitions NTFS supportent sans soucis la compression NTFS et eventuellement certains groupes de fichiers peuvent etre positionnees en lecture seule. Ces considerations ne s'appliquent qu'aux bases de donnees utilisateur et ne peuvent pas etre appliquees sur les fichiers relatifs aux bases de donnees systeme.

#### Creer la base

Pour creer une base de donnees, il faut etre connecte en tant qu'administrateur systeme, ou avoir la permission d'utiliser CREATE DATABASE, et etre dans la base de donnees systeme master.

L'objet DATABASE doit etre cree en premier lieu. Une base de donnees contient tous les autres objets :

* + Le catalogue de base de donnees.
  + Les objets utilisateurs (tables, valeurs par defaut, vues, regles, declencheurs, procedures).
  + Les index, les types de donnees, les contraintes d'integrite.
  + Le journal des transactions.

Le nom de la base de donnees doit etre unique dans une instance SQL Server. Ce nom est limite a 128 caracteres en respectant les regles de construction des identificateurs. Cette longueur maximale est reduite a 123 caracteres si le nom du journal n'est pas precise lors de la creation de la base.

#### Syntaxe

CREATE DATABASE nom\_base [ ON [PRIMARY]

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

[( [ NAME = nom Logique, ] FILENAME = 'nom Physique' [, SIZE = taille]

[, MAXSIZE = { taille Maxi I UNLIMITED } ]

[, FILEGROWTH = valeur Increment] ) [,...]] [ LOG ON { fichier } ]

[COLLATE nom\_classement]

[ FOR ATTACH I FOR ATTACH\_REBUILD\_LOG ]

# Partie

### NAME

Nom logique du fichier.

### FILENAME

Emplacement et nom physique du fichier.

### SIZE

Taille initiale du fichier en megaoctets (MB) ou kilooctets (KB). La taille par defaut est de 1 megaoctet.

### MAXSIZE

Taille maximum du fichier indiquee en kilo ou megaoctets (par defaut megaoctets). Si aucune valeur n'est precisee, alors la taille du fichier sera limitee par la place libre sur le disque.

### UNLIMITED

Pas de taille maximum, la limite est la place libre sur le disque.

### FILEGROWTH

Precise le pas d'increment pour la taille du fichier, qui ne pourra jamais depasser la valeur maximale. Ce pas peut etre precise en pourcentage ou de fa�on statique en kilo ou megaoctets. Les extensions possedent une taille de 64 Ko.

C'est donc la valeur minimale du pas d'increment qu'il faut fixer.

### LOG ON

Emplacement du journal de transactions. Le journal de transactions stocke les modifications apportees aux donnees.

A chaque INSERT, UPDATE ou DELETE, une ecriture est faite dans le journal avant l'ecriture dans la base. La validation des transactions est egalement consignee dans le journal. Ce journal sert a la recuperation des donnees en cas de panne.

### COLLATE

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Indique le classement par defaut de la base de donnees. Le nom de classement peut etre un classement SQL ou Windows. S'il n'est pas precise, c'est le classement par defaut de l'instance SQL Server qui est utilise.

### FOR ATTACH

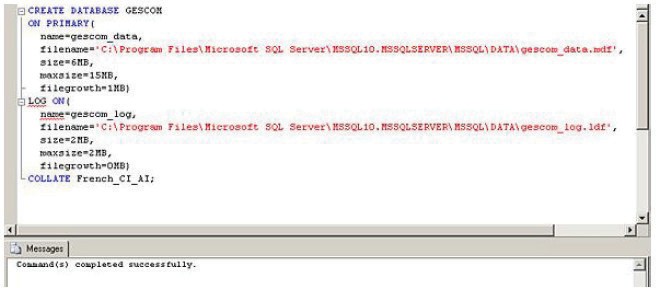
Pour creer une base en utilisant des fichiers deja crees. Cette commande est utile lorsque la base est creee avec plus de 16 fichiers.

### FOR ATTACH\_REBUILD\_LOG

Avec cette option, il est possible de creer la base en lui attachant les fichiers de donnees (mdf et ndf) mais pas necessairement les fichiers journaux. Les fichiers journaux sont alors reconstruits a vide. Si une base est attachee de cette fa�on, il est important d'effectuer rapidement une sauvegarde complete de la base et de planifier tous les processus de sauvegarde. En effet, il n'est pas possible de s'appuyer sur les sauvegardes faites avant l'attachement car les sequences des journaux ne correspondent plus.

*Exemple*

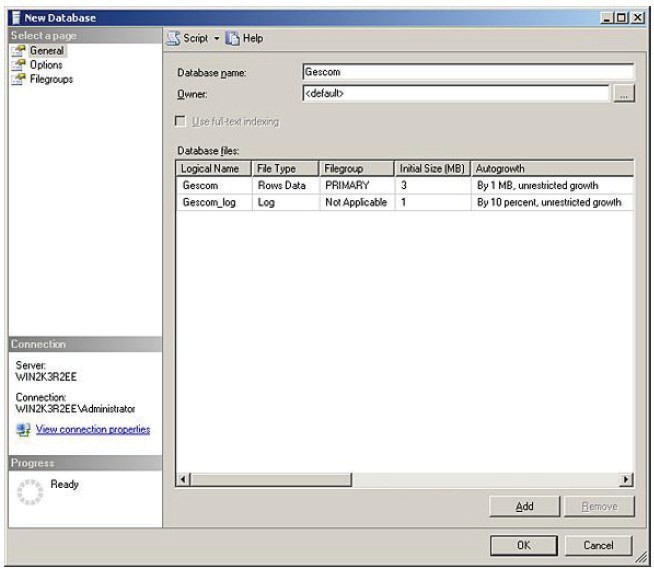
*Creation* *de* *la* *base* *de* *donnees* *Gescom* *(6* *Mo}* *avec* *le* *journal* *de* *transaction* *(2* *Mo}.*



Il est bien sOr possible de realiser cette operation depuis la console graphique SQL Server Management Studio. Pour cela, apres avoir selectionne le noeud Bases de donnees depuis l'explorateur, il faut faire le choix **New** **Database**

(Nouvelle Base de donnees) depuis le menu contextuel pour voir l'ecran suivant s'afficher.

S M I



Q L S E R V E U R 2 0 0 8

S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1Partie

Depuis cette bo1te de dialogue, il est possible de definir les differentes options de creation de la base de donnees.

#### Modifier la taille

Il est possible d'augmenter ou de diminuer la taille des fichiers de fa�on automatique ou manuelle. Si un pas d'increment (FILEGROWTH) et une taille maximum sont precises lors de la creation du fichier, le fichier changera de taille en fonction des besoins.

Il est possible de modifier manuellement la taille, la taille maximale et le taux d'augmentation d'un fichier de donnees avec la commande ALTER DATABASE.

#### Syntaxe

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

ALTER DATABASE nom MODIFY FILE

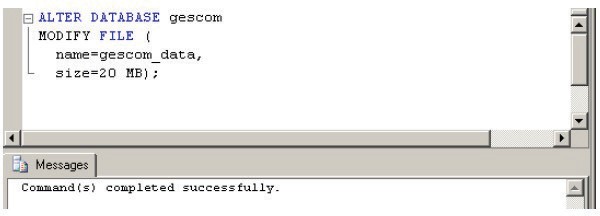
(NAME=nom Logique [,SIZE=taille] [,MAXSIZE=taille Maxi]

[FILEGROWTH=valeur Increment])

*Exemple*

*Augmenter* *la* *taille* *d'un* *fichier* *existant* *:*

# Partie



Il est egalement possible d'ajouter des fichiers.

ALTER DATABASE nom ADD FILE (

NAME = nom Logique, FILENAME = 'nom Physique' [, SIZE = taille]

[, MAXSIZE = { taille Maxi I UNLIMITED } ]

[, FILEGROWTH = valeu rIncrement] )

*Exemple*

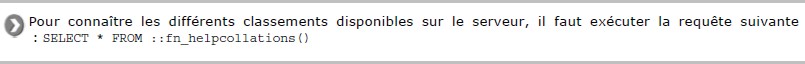
*Ajout* *d'un* *deuxieme* *fichier* *a* *la* *base* *GESCOM* *:*

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie



La commande ALTER DATABASE permet une action beaucoup plus importante sur la base que la simple modification de taille des fichiers. Il est ainsi possible d'ajouter et de supprimer des fichiers et des groupes de fichiers, de modifier le nom de la base, de preciser le mode de fin par defaut des transactions en cours et de changer le classement de la base.

#### Diminuer la taille

La taille des fichiers peut diminuer de fa�on automatique si l'option **autoshrink** a ete positionnee sur la base.

Les commandes DBCC SHRINKFILE et DBCC SHRINKDATABASE permettent d'executer manuellement la diminution de taille.

DBCC SHRINKFILE ne va porter que sur un fichier en particulier tandis que DBCC SHRINKDATABSE va scruter tous les fichiers de la base.

L'operation de reduction des fichiers commence toujours par la fin du fichier. Par exemple, si la base dispose d'un fichier de 500 Mo que l'on souhaite ramener a une taille de 400 Mo ce sont les 100 derniers megaoctets du fichier qui vont etre reorganises afin de ne plus detenir aucune donnee avant leur liberation.

La quantite d'espace reellement liberee est fonction de la taille ideale fixee en parametre a DBCC SHRINKFILE et de la realite des donnees. Si dans l'exemple precedent, le fichier contient 450 Mo d'extensions utilises, alors seulement 50 Mo d'espace seront liberes.

#### Syntaxe

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

DBCC SHRINKFILE **(**nom\_fichier {[ ,*taille\_cible* ]

I [ , { EMPTYFILE I NOTRUNCATE I TRUNCATEONLY } ] }**)**

DBCC SHRINKDATABASE **(***nom\_base* [ , *pourcentage\_cible* ] [ , { NOTRUNCATE I TRUNCATEONLY } ] **)**

# Partie

DBCC SHINKFILE

Reduit la taille du fichier de donnees ou du fichier journal pour la base de donnees specifiee.

DBCC SHRINKDATABASE

Reduit la taille des fichiers de donnees dans la base de donnees specifiee.

#### Taille\_cible

Indique la taille souhaitee du fichier apres reduction.

#### Pourcentage\_cible

Indique le pourcentage d'espace libre que l'on souhaite obtenir dans le fichier de donnees apres reduction de la base.

### EMPTYFILE

Permet de demander a la commande DBCC\_SHRINKFILE de transferer toutes les donnees presentes dans ce fichier de donnees vers un autre fichier du meme groupe. Une fois vide, le fichier pourra etre supprime a l'aide d'unecommande ALTER TABLE.

### NOTRUNCATE

Reorganise le fichier en positionnant les pages occupees en haut de fichier, mais il n'est pas diminue.

### TRUNCATEONLY

Coupe le fichier sans faire aucune reorganisation du fichier.

#### Supprimer la base

La commande DROP DATABASE permet de supprimer la base. Les fichiers physiques sont egalement supprimes.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Si certains fichiers sont detaches de la base avant sa suppression alors ils ne seront pas supprimes et il sera necessaire d'effectuer cette operation de fa�on manuelle depuis l'explorateur de fichiers. Enfin, si des utilisateurs sont connectes sur la base il n'est pas possible de la supprimer. Pour forcer la deconnexion de ces derniers et permettre la suppression de la base il est necessaire de basculer vers le mode SINGLE\_USER a l'aide de l'instruction ALTER DATABASE.

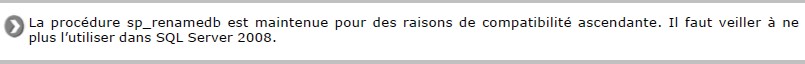
L'instruction DROP DATABASE ne peut etre executee que si le mode autocommit est active (ce qui est le cas par defaut). Il n'est pas possible de supprimer les bases systeme.

#### Renommer une base

Il est possible de renommer une base de donnees par l'intermediaire de l'instruction ALTER DATABASE.

#### Syntaxe

ALTER DATABASE *nom* *Base* MODIFY NAME= *nouveau* *NomBase*

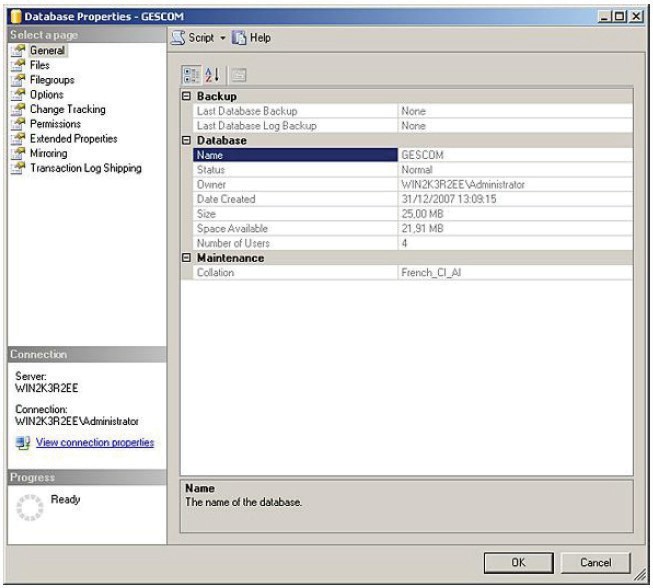


#### Configurer une base

Il est possible de configurer une base de donnees pour fixer un certain nombre d'options afin d'obtenir le comportement souhaite de la base en fonction des besoins des utilisateurs. On accede a ces differentes options, soit de fa�on graphique par SQL Server Management Studio en se positionnant sur la base puis en appelant la fenetre des proprietes par la touche [F4], soit par le menu contextuel associe a la base, ou bien encore par le menu **View** **Properties**

**Window** (Affichage Fenetre Proprietes) dans le menu general de SQL Server Management Studio.

S M I



Q L S E R V E U R 2 0 0 8

S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1Partie

#### Syntaxe

ALTER DATABASE nomBase SET option ;

*Etat* *de* *la* *base*

### ONLINE

Permet de rendre la base de donnees de nouveau visible.

### OFFLINE

Permet de rendre inaccessible la base de donnees. La base de donnees est fermee et arretee proprement. Il n'est pas possible de faire des operations de maintenance sur une base de donnees offline.

### EMERGENCY

La base de donnees est mise en lecture seule, la journalisation est desactivee et son acces est limite aux seuls administrateurs du serveur.

*Acces*

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

### SINGLE\_USER

Acces limite a un seul utilisateur.

### RESTRICTED\_USER

Seuls les membres des roles db\_owner, dbcreator, ou sysadmin peuvent se connecter a la base.

### MULTI\_USER

C'est le mode par defaut qui permet a tous les utilisateurs disposant de privileges suffisants d'acceder a l'information.

*Operations* *possibles*

### READ\_ONLY

La base est accessible uniquement pour les operations de lecture.

### READ\_WRITE

La base est accessible pour les operations de lecture/ecriture.

*Acces*

#### DBO use only

La base n'est accessible que par son proprietaire.

*Parametrage*

### ANSI\_NULL\_DEFAULT

Definit la valeur par defaut de la contrainte de nullite de colonne. Selon la norme ANSI, une colonne peut etre NULL par defaut.

### RECURSIVE\_TRIGGERS

Autorise la recursivite des triggers.

### TORN\_PAGE\_DETECTION

Permet de detecter les pages incompletes.

### AUTO\_CLOSE

La base est arretee et les ressources sont liberees apres la deconnexion du dernier utilisateur.

### AUTO\_SHRINK

Les fichiers de la base pourront etre automatiquement reduits.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

### AUTO\_CREATE\_STATISTICS

Toutes les statistiques manquantes lors de l'optimisation d'une requete sont creees. Cette option est active (positionnee a ON) par defaut.

### AUTO\_UPDATE\_STATISTICS

Toutes les statistiques obsoletes pour la bonne optimisation d'une requete sont recalculees.

### AUTO\_UPDATE\_STATISTICS\_ASYNC

Les statistiques qui permettent de representer la pertinence des index sont mises a jour de fa�on asynchrone. La requete qui provoque la mise a jour des statistiques n'attend pas que les statistiques soient a jour pour s'executer.

Ce sont les futures requetes qui profiteront de cette mise a jour. Cette option est active (positionnee a ON) par defaut.

### QUOTED\_IDENTIFIERS

Les identificateurs delimites peuvent etre encadres par des guillemets doubles.

### ANSI\_NULLS

Si le parametre est vrai (true), alors toutes les comparaisons avec une valeur NULL sont evaluees a inconnues. Si le parametre est faux (false) alors les comparaisons avec les valeurs NULL et les valeurs non unicode sont evaluees a VRAI si les deux valeurs sont NULL.

### ANSI\_WARNINGS

Permet de faire remonter des messages d'erreur ou des avertissements lorsque certaines conditions sont remplies.

### ARITHABORT

Permet d'arreter le traitement du lot d'instructions lors d'un depassement de capacite ou d'une division par zero.

### CONCAT\_NULL\_YIELDS\_NULL

Le resultat est NULL si l'un des deux operandes d'une operation de concatenation est NULL.

### CURSOR\_CLOSE\_ON\_COMMIT

Permet de fermer tous les curseurs lors de la definition d'une transaction ou lors de la fin d'une transaction.

### CURSOR\_DEFAULT

Les declarations de curseur ont pour valeur par defaut LOCAL.

### NUMERIC ROUNDABORT

Une erreur est levee si une perte de precision intervient au cours d'un calcul.

### RECOVERY

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Permet de preciser la strategie de sauvegarde planifiee au niveau de la base. Ce parametre a une incidence directe sur les informations conservees dans les journaux de transactions.

### PAGE\_VERIFY

Cette option permet de valider la qualite des informations stockees au niveau de chaque page. L'option par defaut CHECKSUM est celle recommandee par SQL Server.

### SUPPLEMENTAL\_LOGGING

En positionnant cette option a ON (OFF par defaut), des informations complementaires vont etre ajoutees au journal. Il est possible de conna1tre l'etat de cette option en examinant la valeur contenue dans la colonne is\_supplemental\_logging\_enabled de la vue sys.databases.

### PARAMETERIZATION

En mode SIMPLE, par defaut les requetes sont parametrees en fonction des regles en vigueur sur le serveur. En utilisant le mode FORCED, SQL Server parametre toutes les requetes avant de dresser le plan d'execution.

*Gestion* *des* *transactions*

**ROLLBACK** **AFTER** nombre

L'annulation des transactions est effective apres nombre secondes d'attente.

### ROLLBACK IMMEDIATE

L'annulation de la transaction est immediate.

### NO\_WAIT

Si la transaction n'accede pas immediatement aux ressources qui lui sont necessaires, elle est annulee.

### ANSI\_PADDING

Permet de specifier si les espaces a droite sur les donnees de type caractere doivent etre supprimes ou pas.

### COMPATIBILITY

Permet de fixer le niveau de compatibilite de la base de donnees : 80 pour SQL Server 2000, 90 pour SQL Server 2005 et 100 pour SQL Server 2008.

### DATE\_CORRELATION\_OPTIMISATION

Avec cette option SQL Server se charge de maintenir la correlation des statistiques entre deux tables liees par une contrainte de cle etrangere et qui possedent toutes les deux une colonne de type datetime.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

*Acces* *externe*

# Partie

### DB\_CHAINING

Permet de gerer les contextes de securite lors de l'acces a la base a partir d'une autre base.

### TRUSTWORTHY

Des modules internes (procedures stockees, fonctions) peuvent acceder a des ressources externes au serveur en utilisant un contexte impersonnate.

*Service* *Broker*

### ENABLE\_BROKER

Active le service Broker.

### DISABLE\_BROKER

Desactive le service Broker.

### NEW\_BROKER

Permet de preciser que la base doit recevoir un nouvel identifiant Service Broker.

### ERROR\_BROKER\_CONVERSATIONS

Les conversations en cours vont recevoir un message d'erreur et vont ainsi etre cloturees.

*Snapshot* *(Capture* *instantanee}*

### ALLOW\_SNAPSHOT\_ISOLATION

Lorsque ce mode est active, toutes les transactions sont capables de travailler avec une capture instantanee (snapshot) de la base telle qu'elle etait au debut de la transaction.

### READ\_COMMITTED\_SNAPSHOT

Lorsque ce mode est activite, toutes les instructions per�oivent les donnees telles qu'elles etaient au debut de l'instruction.

## Gerer les tables et les index

#### Identifiant

Tous les elements crees dans SQL Server sont parfaitement identifies par leur nom qui est utilise en tant qu'identifiant.

En effet, deux objets de meme type ne peuvent pas avoir le meme nom s'ils sont definis au meme niveau. Par exemple, sur une instance de SQL Server, il n'est pas possible d'avoir deux

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

bases de donnees avec le meme nom, par contre, c'est totalement possible si les bases sont definies sur deux instances distinctes de SQL Server.

De meme, au sein d'une base de donnees, il n'est pas possible d'avoir deux tables avec le meme nom. C'est a partir de l'identifiant qu'il est possible de manipuler via le SQL les objets. Il est donc important de definir correctement ces identifiants.

Les identifiants sont composes de 1 a 128 caracteres. Ils commencent toujours par une lettre ou l'un des caracteres suivants : \_, @, #.

Les caracteres suivants sont des caracteres alphanumeriques.

Bien entendu aucun identifiant ne peut correspondre a un mot cle du Transact SQL. Il existe deux categories d'identifiants : les reguliers et les delimites.

#### Les identifiants reguliers

Cette categorie d'identifiant est la plus communement utilisee et c'est celle a privilegier. En effet, ce type d'identifiant est present dans toutes les bases et presente beaucoup de souplesse au niveau de l'ecriture des requetes car la casse n'est pas conservee.

*Exemple* *:*

RessourcesInformatiques

#### Les identifiants delimites

Cette categorie d'identifiant permet de conserver des caracteres speciaux dans les identifiants comme les caracteres accentues, les espaces, . mais egalement de conserver la casse. Ces identifiants sont utilises entre des crochets [] ou bien des guillemets "". L'utilisation de ce type d'identifiant ne permet que rarement de gagner en clarte car l'ecriture des requetes est plus lourde. Il est donc preferable d'utiliser des identifiants reguliers.

*Exemple* *:*

[editions numero 2], "Ressources Informatiques", ...

#### Les types de donnees

Lors de la definition d'une colonne, on precisera le format d'utilisation de la donnee ainsi que le mode de stockage par le type de la colonne.

#### Types de donnees systeme

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Ces types sont disponibles pour toutes les bases de donnees en standard.

#### Caracteres char[(n)]

Cha1ne de caracteres de longueur fixe, de n caracteres maximum. Par defaut 1, maximum 8000. varchar(nImax)

Cha1ne de caracteres a longueur variable, de n caracteres maximum. Par defaut 1, maximum 8000 caracteres. En precisant max, la variable peut contenir des donnees de type texte allant jusqu'a **231** caracteres.

#### nchar[(n)]

Cha1ne de caracteres unicode, maximum 4000 caracteres.

#### nvarchar (nImax)

Cha1ne de caracteres unicode, maximum 4000. En precisant max, la variable peut contenir des donnees de type texte allant jusqu'a **231** octets.

Numeriques

#### decimal [(p[,d])]

Numerique exact de precision p (nombre de chiffres total), avec d chiffres a droite de la virgule. p est compris entre 1 et 38, 18 par defaut.

d est compris entre 1 et p, 0 par defaut.

Exemple : pour decimal (8,3) l'intervalle admis sera de 99999,999 a +99999,999.

Les valeurs sont gerees de 1038 a 1038 1.

#### numeric [(p[,d])]

Identique a decimal. Pour le type decimal, la precision pourra etre parfois plus grande que celle requise.

#### bigint

Type de donnees entier code sur 8 octets. Les valeurs stockees avec ce type de donnees sont comprises entre (-9223 372 036 854 775 808) et (9 223 372 036 854 775 807).

#### int

Nombre entier entre ( -2147783648)et (+2147483647). Le type de donnees **int** est specifique SQL Serveret son synonyme **integer** est quant a lui compatible ISO.

#### smallint

Nombre entier entre (32768)et (+32767).

#### tinyint

Nombre entier positif entre 0 et 255.

#### float[(n)]

Numerique approche de n chiffres, n allant de 1 a 53.

#### real

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

Identique a float(24).

#### money

Numerique au format monetaire compris entre

337 203 685 477, 5808 et + 337 203 685 477, 5807 (8 octets).

#### smallmoney

Numerique au format monetaire compris entre 748,3648et + 748,3647 (4 octets).

*Binaires*

# Partie

#### binary[(n)]

Donnee binaire sur n octets (1 a 255), la longueur est fixe.

#### varbinary (nImax)

Donnee binaire de longueur variable de n octets (1 a 8000). L'option max permet de reserver un espace de 231 octets au maximum.

##### *Date*

Pour la gestion des donnees de types date et heure SQL Server 2008 propose de nouveaux types de donnees afin d'optimiser le stockage des donnees. Ces nouveaux types de donnees sont introduits pour permettre une gestionplus fine des donnees de types date et heure.

Tout d'abord, il existe des types particuliers pour stocker les donnees de type heure et d'autres types pour stocker les donnees de type date. Cette separation est benefique car elle permet de donner plus de precision tout en limitant l'espace utilise par les donnees de tel ou tel type.

Par exemple, Est il necessaire de conserver des donnees de type heures, minutes et secondes lorsque seule la date de naissance d'un client doit etre conservee ? Sans aucun doute, non.

Le simple fait de conserver ces informations peut induire des erreurs lorsque par la suite des calculs vont etre faits. Il est donc plus raisonnable et plus performant d'adopter pour cette donnee, un type qui ne conserve que la date.

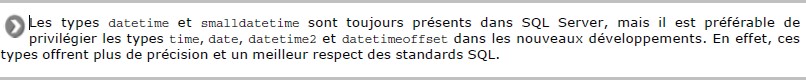
Ces nouveaux types de donnees, pour les donnees de type date et heure, vont egalement permettre une meilleure compatibilite avec les autres systemes de gestion de donnees et faciliter les operations de reprise de donnees.

Bien entendu SQL Server offre la possibilite avec les types datetime2 et datetimeoffset de conserver des informations de type date et heure de fa�on simultanee.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Le type datetime offset permet non seulement de stocker des informations de type date et heure avec une precision pouvant aller jusqu'a 100 nanosecondes, mais en plus c'est l'heure au format UTC qui est conservee ainsi que le decalage (en nombre d'heures) entre cette heure UTC et la zone horaire depuis laquelle travaille l'utilisateur qui introduit les informations dans la base.

#### datetime

Permet de stocker une date et une heure sur 8 octets. 4 pour un nombre de jours par rapport au 1er janvier 1900, 4 pour un nombre de millisecondes apres minuit. Les dates sont gerees du 1er janvier 1753 au 31 decembre 9999. Les heures sont gerees avec une precision de 3,33 millisecondes.

#### smalldatetime

Permet de stocker une date et une heure sur 4 octets. Les dates sont gerees du 1er janvier 1900 au 6 juin 2079, a laminute pres.

#### datetime2

Plus precis que le type datetime, il permet de stocker une donnee de type date et heure comprise entre le 01/01/0001 et le 31/12/9999 avec une precision de 100 nanosecondes. **datetimeoffset**

Permet de stocker une donnee de type date et heure comprise entre le 01/01/0001 et le 31/12/9999 avec une precision de 100 nanosecondes. Les informations horaires sont stockees au format UTC et le decalage horaire estconserve afin de retrouver l'heure locale renseignee initialement.

#### date

Permet de stocker une date comprise entre le 01/01/0001 et le 31/12/9999 avec une precision d'une journee.

#### time

Permet de stocker une donnee positive de type heure inferieure a 24h00 avec une precision de 100 nanosecondes.

*Speciaux*

#### bit

Valeur entiere pouvant prendre les valeurs 0, 1 ou null.

#### timestamp

Donnee dont la valeur est mise a jour automatiquement lorsque la ligne est modifiee ou inseree.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

#### uniqueidentifier

Permet de creer un identificateur unique en s'appuyant sur la fonction NEWID().

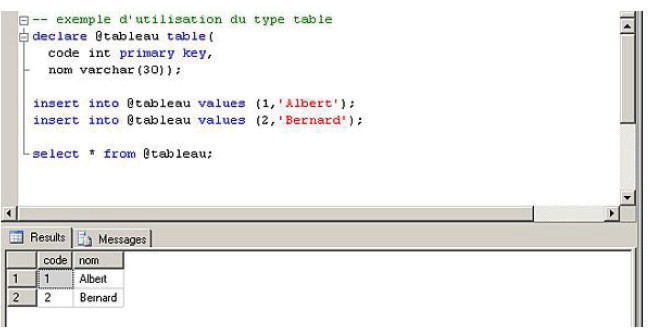
#### sql\_variant

Le type de donnees **sql\_variant** permet de stocker n'importe quel type de donnees a l'exception des donnees de type **text,** **ntext,** **timestamp** et **sql\_variant**. Si une colonne utilise ce type de donnees, les differentes lignes de la table peuvent stocker dans cette colonne des donnees de type different. Une colonne de type **sql\_variant** peut posseder une longueur maximale de 8016 octets. Avant d'utiliser une valeur stockee au format **sql\_variant** dans une operation,

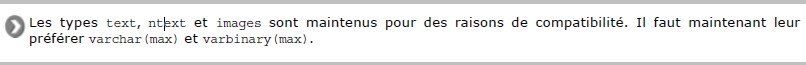
il est necessaire de convertir les donnees dans leur format d'origine. Les colonnes utilisant le type **sql\_variant** peuvent participer a des contraintes de cles primaires, de cles etrangeres ou d'unicite, mais les donnees contenues dans la cle d'une ligne ne peuvent exceder les 900 octets (limite imposee par les index). **sql\_variant** ne peut pas etre utilise dans les fonctions CONTAINSTABLE et FREETEXTTABLE.

#### table

C'est un type de donnees particulier qui permet de stocker et de renvoyer un ensemble de valeurs en vue d'une utilisation future. Le mode principal d'utilisation de ce type de donnees est la creation d'une table temporaire.



#### xml

Ce type permet de stocker un document xml dans une colonne au sein d'une table relationnelle.

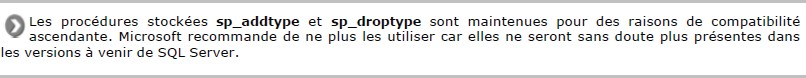
S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

* 1. **Types** **de** **donnees** **definis** **par** **l'utilisateur**

# Partie

Il est possible de definir ses propres types de donnees, soit par l'intermediaire de Management Studio, soit par la commande CREATE TYPE.



#### Syntaxe (creation)

CREATE TYPE nomType

{FROM typeDeBase [ ( longueur [ , precision ] ) ] [ NULL I NOT NULL ]

I EXTERNAL NAME nomAssembly [ .nomClasse]

} [ I ]

Il est possible de definir un type de donnees en s'appuyant sur la definition d'une classe. Cette option est liee a l'integration du CLR dans SQL Server. Cette integration est detaillee plus tard dans ce syllabus.

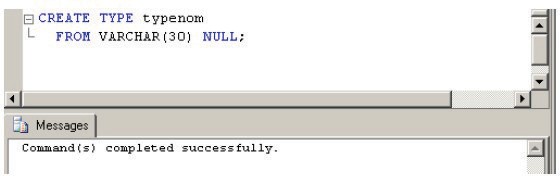
#### Syntaxe (suppression)

DROP TYPE [ schema\_name. ] type\_name [ I ]

Un type ne pourra pas etre supprime s'il est utilise dans une table de la base ou il a ete cree.

*Exemples*

*Creation* *d'un* *type* *pour* *les* *colonnes* *telles* *que* *nom* *client,* *nom* *fournisseur* *etc.* *:*

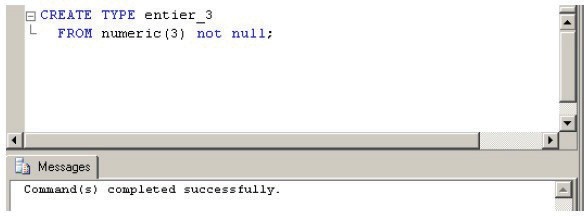


S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

*Creation* *d'un* *type* *pour* *des* *valeurs* *entre* *999* *et* *+999* *:*

# Partie



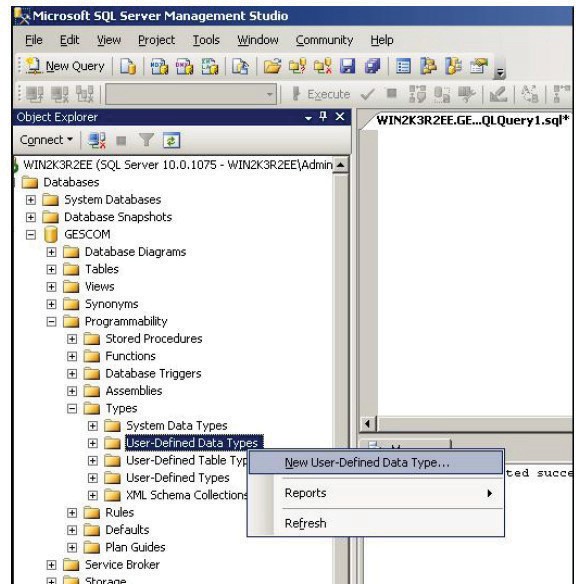
*Demander* *la* *creation* *d'un* *nouveau* *type* *de* *donnees* *depuis* *SQL* *Server* *Management* *Studio* *:*

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

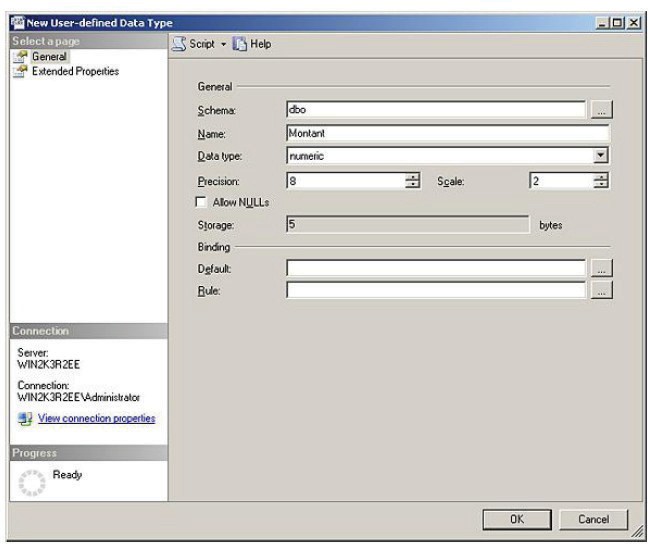
# Partie

1



*Definition* *du* *nouveau* *type* *de* *donnees* *"Montant"* *depuis* *SQL* *Server* *Management* *Studio* *:*

S M I



Q L S E R V E U R 2 0 0 8

S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1Partie

En plus de definir des alias (comme illustre cidessus), l'instruction CREATE TYPE permet de creer des types UDT (*User* *Defined* *Type*) c'est a dire definis a l'aide du CLR (ce point sera aborde dans le chapitre CLR).

Au niveau Transact SQL, l'instruction CREATE TYPE permet egalement de creer des types composes de plusieurs champs. Ces types sont frequemment nommes types structures dans les langages de programmation.

En ce qui concerne SQL Server, l'instruction CREATE TYPE permet de creer un type TABLE ou tableau. Chaque colonne qui participe a la definition de ce nouveau type est definie sur le meme principe qu'une colonne dans une table. Ainsi il est possible de definir les contraintes d'integrite de cle primaire (PRIMARY KEY), d'unicite (UNIQUE), de validation (CHECK) et de non nullite. Ces contraintes peuvent etre definies au niveau de la colonne ou bien de la table. Il est

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

egalement possible de definir une colonne de type identite.

# Partie

L'instruction CREATE TYPE permet ainsi de creer des types dits fortement types car les contraintes d'integrite permettent une definition plus precise du format possible des donnees. L'introduction de ce nouveau type va permettre de definir des parametres de fonctions ou procedures de type tableau. On parlera alors d'un table value parameter.

1

#### Syntaxe

CREATE TYPE nom Type AS TABLE (

colonne type Colonne[contrainte Colonne], ...)

#### nomType

Nom du type ainsi cree.

#### colonne

Nom de la colonne qui participe a la definition de ce nouveau type. Il est bien sOr possible de definir plusieurs colonnes.

#### typeColonne

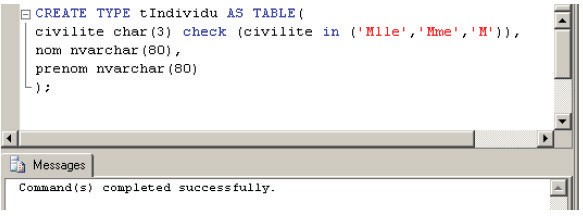
Type de donnees Transact SQL sur lequel est definie la colonne. Toutes les colonnes d'un meme type de table ne sont pas forcement definies sur le meme type ni avec la meme precision.

Definition de la contrainte d'integrite associee a la colonne.

*Exemple*

*Dans* *l'exemple* *suivant* *un* *type* *representant* *un* *individu* *est* *defini.* *Ce* *type* *est* *compose* *des* *champs* *civilite,* *nom* *et* *prenom.* *Pour* *le* *champ* *civilite* *seules* *certaines* *valeurs* *sont* *autorisees.*

S Q L M I S



S E R V E U R 2 0 0 8

E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1Partie

#### Gerer les tables

Une table represente une structure logique dans laquelle les donnees vont etre rangees. Pour permettre une bonne organisation des informations, chaque table est constituee de colonnes afin de structurer les donnees. Chaque colonne est parfaitement identifiee par son nom, qui est unique a l'interieur de la table, et par son type de donnees. Les donnees sont reparties entre plusieurs tables. Les contraintes d'integrite permettent de garantir la coherence des donnees. Les trois operations de gestion de table sont la creation (CREATE TABLE), la modification (ALTER TABLE) et la suppression (DROP TABLE). Ces operations peuvent etre realisees en Transact SQL ou par SQL Server Management Studio par un utilisateur "dbo" ou ayant re�u le droit CREATE TABLE.

#### Creer une table

L'etape de creation des tables est une etape importante de la conception de la base car les donnees sont organisees par rapport aux tables. Cette operation est ponctuelle et elle est en general realisee par l'administrateur (DBA : *DataBase* *Administrator*) ou tout au moins par la personne chargee de la gestion de la base.La creation d'une table permet de definir les colonnes (nom et type de donnees) qui la composent ainsi que les contraintes d'integrite. De plus, il est possible de definir des colonnes calculees, un ordre de tri specifique a la colonne ainsi que la destination des donnees de type texte ou image.

#### Syntaxe

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

CREATE TABLE [nom Schema.] *nom\_table*

( *nom\_colonne* {*type* *colonne*IAS *expression\_calculee*} [,*nom\_colonne* ... ][,*contraintes*...])

[ON *groupefichier*] [TEXTIMAGE\_ON *groupe\_fichier*]

# Partie

#### nomSchema

1

Nom du schema dans lequel la table va etre definie.

#### nom\_table

Peut etre sous la forme base.proprietaire.table.

#### nom\_colonne

Nom de la colonne qui doit etre unique dans la table.

#### Typecolonne

Type systeme ou type defini par l'utilisateur.

#### contraintes

Regles d'integrite (traitees ulterieurement dans ce syllabus).

#### groupefichier

Groupe de fichiers sur lequel va etre creee la table.

#### AS expression\_calculee

Il est possible de definir une regle de calcul pour les colonnes qui contiennent des donnees calculees. Bien entendu,ces colonnes ne sont accessibles qu'en lecture seule, et il n'est pas possible d'inserer des donnees ou de mettre a jour les donnees d'une telle colonne.

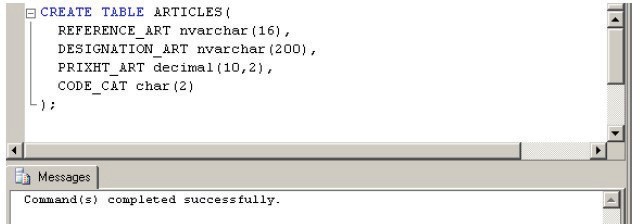
### TEXTIMAGE\_ON

Permet de preciser le groupe de fichiers destination pour les donnees de type texte et image. Il est possible de creer 2 milliards de tables par base de donnees.

*Exemples*

*Creation* *de* *la* *table* *ARTICLES* *:*

S M I

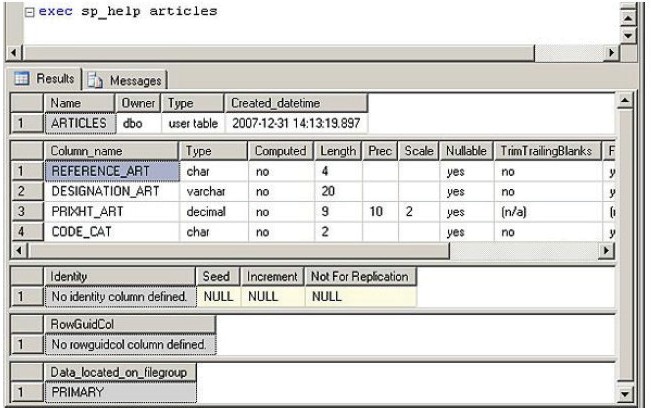


Q L S E R V E U R 2 0 0 8

S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1Partie

*Affichage* *des* *informations* *sur* *la* *table* *a* *l'aide* *de* *la* *procedure* *stockee* *sp\_help* *:*



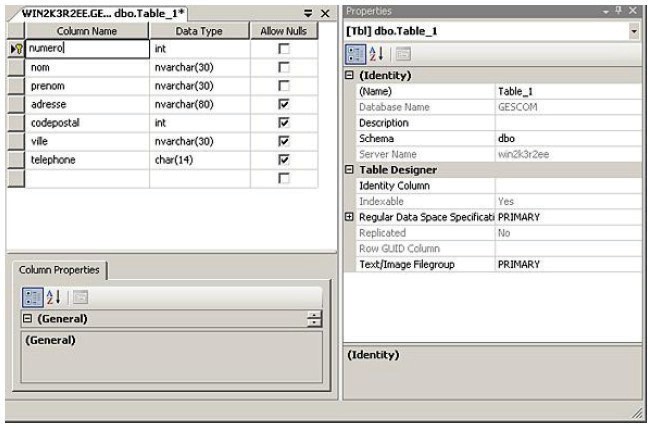
*Creation* *de* *la* *table* *CLIENTS* *(depuis* *l'interface* *graphique}* *:*

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

1



#### Modifier une table

La modification de table est effectuee par la commande ALTER TABLE ou par l'interface graphique de SQL Server Management Studio. Lors d'une modification de table, il est possible d'ajouter et de supprimer des colonnes et des contraintes, de modifier la definition d'une colonne (type de donnees, classement et comportement vis a vis de la valeur NULL), d'activer ou de desactiver les contraintes d'integrite et les declencheurs. Ce dernier point peut s'averer

utile lors d'import massif de donnees dans la base si l'on souhaite conserver des temps de traitements coherents.

*Syntaxe*

ALTER TABLE [nomSchema.] nomtable

{ [ ALTER COLUMN nom\_colonne

{ nouveau\_type\_donnees [ ( longueur [ , precision ] ) ] [ COLLATE classement ] [ NULL I NOT NULL ] } ]

I ADD nouvelle\_colonne

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

I [ WITH CHECK I WITH NOCHECK ] ADD contrainte\_table

I DROP { [ CONSTRAINT ] nom\_contrainte I COLUMN nom\_colonne } I { CHECK I NOCHECK } CONSTRAINT { ALL I nom\_contrainte }

I { ENABLE I DISABLE } TRIGGER { ALL I nom\_declencheur } }

# Partie

#### nomSchema

Nom du schema dans lequel la table va etre definie.

### WITH NOCHECK

Permet de poser une contrainte d'integrite sur la table sans que cette contrainte soit verifiee par les lignes deja presentes dans la table.

### COLLATE

Permet de definir un classement pour la colonne qui est different de celui de la base de donnees.

### NULL, NOT NULL

Permettent de definir une contrainte de nullite ou de non nullite sur une colonne existante de la table.

### CHECK, NOCHECK

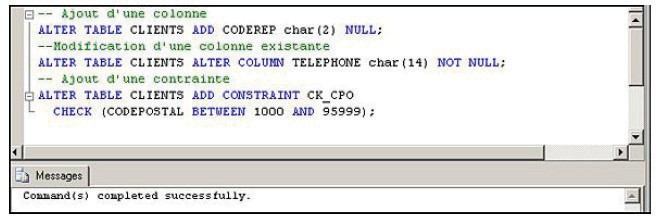
Permettent d'activer et de desactiver des contraintes d'integrite.

### ENABLE, DISABLE

Permettent d'activer et de desactiver l'execution des declencheurs associes a la table.

*Exemple*

*Ajout* *d'une* *colonne* *:*



#### Supprimer une table

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

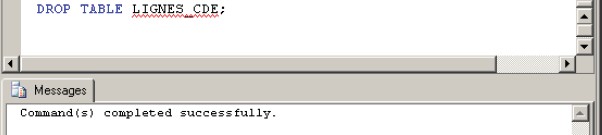
La suppression d'une table entra1ne la suppression de toutes les donnees presentes dans la table. Les declencheurs et les index associes a la table sont egalement supprimes. Il en est de meme pour les permissions d'utilisation de la table. Par contre, les vues, procedures et fonctions qui referencent la table ne sont pas affectees par la suppression de la table. Si elles referencent la table supprimee, alors une erreur sera levee lors de la prochaine execution.

Syntaxe

DROP TABLE [nomSchema.] nomtable [,nomtable...]

La suppression d'une table supprimera les donnees et les index associes. La suppression ne sera pas possible si la table est referencee par une cle etrangere.

*Suppression* *d'une* *table* *:*



#### Nom complet d'une table

En fonction de l'emplacement depuis lequel la table, et plus generalement l'objet, est reference, il est necessaire d'utiliser un nom plus ou moins precis. Le nom complet d'une table, et donc d'un objet, est de la forme suivante :

#### nomBase.nomSchema.nomObjet

Cependant, comme habituellement les objets references sont presents sur la base courante, il est possible d'omettre le nom de la base.

Le nom du schema peut egalement ne pas etre specifie. Dans un tel cas de figure, le moteur de base de donnees cherchera l'objet dans le schema associe a l'utilisateur, puis si cette recherche est infructueuse dans le schema dbo.

## Mise en oeuvre de l'integrite des donnees

S Q L S E R V E U R 2 0 0 8

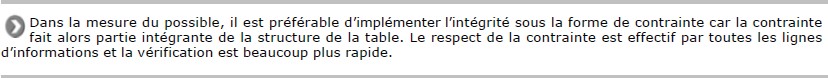
M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Pour assurer la coherence des donnees dans la base, il est possible de gerer au niveau du serveur un ensemble defonctionnalites qui permettent de centraliser les controles et les regles de fonctionnement dictes par l'analyse.

La mise en oeuvre de l'integrite des donnees peut se faire de maniere procedurale par les valeurs par defaut (DEFAULT)et les declencheurs (TRIGGER) ou de maniere declarative par les contraintes (CONSTRAINT) et la propriete IDENTITY.

L'integrite des donnees traduit les regles du modele relationnel, regle de coherence (integrite de domaine), existence de valeurs nulles, regle d'unicite (integrite d'entite) et cles etrangeres (integrite referentielle).



#### Les valeurs par defaut

Depuis SQL Server 2005, les objets DEFAULT n'ont plus cours et ne doivent pas etre mis en place dans le cadre de nouveaux developpements. En effet, ce type d'objet n'est pas conforme a la norme du SQL.

Meme si les instructions CREATE DEFAULT, DROP DEFAULT, sp\_bindefault et sp\_unbindefault sont toujours presentes, il est preferable de definir la valeur par defaut lors de la creation de la table (CREATE TABLE) ou de passer par une instruction de modification de table (ALTER TABLE). Comme toujours, ces operations peuvent etre executees sous forme de script ou par l'intermediaire de SQL Server Management Studio.

#### Les regles

Afin de proposer une gestion plus uniforme des differents elements de la base, en generalisant l'utilisation des instructions CREATE, ALTER et DROP, et pour etre plus proche de la norme, SQL Server 2008 ne propose plus la gestion des regles en tant qu'objet independant. Les contraintes d'integrite qui pouvaient etre exprimees sous forme de regle doivent etre definies lors de la creation de la table par l'instruction CREATE TABLE. Elles peuvent egalement etre ajoutees/supprimees sur une table existante par l'intermediaire de l'instruction ALTER TABLE. Pour assurer la continuite des scripts, SQL Server continue d'interpreter correctement les instructions CREATE RULE, DROP RULE, sp\_bindrule, sp\_unbindrule.

#### La propriete Identity

Cette propriete peut etre affectee a une colonne numerique entiere, a la creation ou a la modification de la table et permet de faire generer, par le systeme, des valeurs pour cette colonne. Les valeurs seront generees a la creation de la

ligne, successivement en partant de la valeur initiale specifiee (par defaut 1) et en augmentant ou diminuant ligne apres ligne d'un increment (par defaut 1).

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

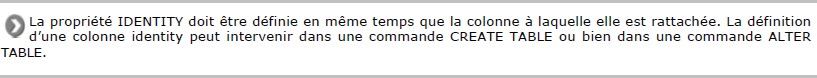
*Syntaxe*

# Partie

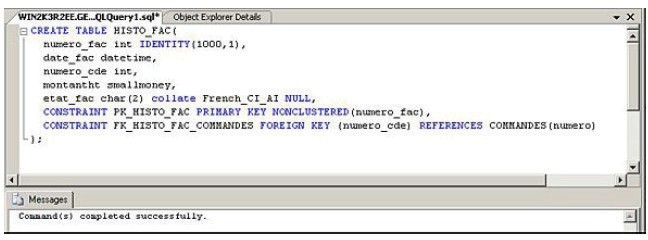
CREATE TABLE nom (colonne typeentier IDENTITY [(depart, increment)],

...)

Il ne peut y avoir qu'une colonne IDENTITY par table !



*Exemple*



Lors des creations de ligne (INSERT), on ne precisera pas de valeur pour NUMERO\_FAC. La premiere insertion affectera le NUMERO\_FAC 1000, la deuxieme le NUMERO\_FAC 1001, etc. Le mot cle IDENTITYCOL pourra etre utilise dans une clause WHERE a la place du nom de colonne.

La variable globale @@**IDENTITY** stocke la derniere valeur affectee par une identite au cours de la session courante. La fonction **SCOPE\_IDENTITY** permet d'effectuer le meme type de travail mais limite la portee de la visibilite au seul lot d'instructions courant. La fonction **IDENT\_CURRENT** permet quant a elle de conna1tre la derniere valeur identite generee pour la table specifiee en parametre, quelles que soient les sessions.

Pour pouvoir inserer des donnees sans utiliser la propriete IDENTITY et donc la numeration automatique, il faut faire appel a l'instruction IDENTITY\_ INSERT de la fa�on suivante :

SET IDENTITY\_INSERT nom\_de\_table ON

Le parametre ON permet de desactiver l'usage de la propriete IDENTITY tandis que la meme instruction avec le parametre OFF reactive la propriete.

S Q L S E R V E U R 2 0 0 8

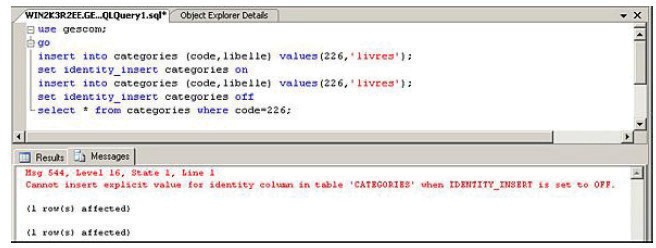
M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

*Exemple*

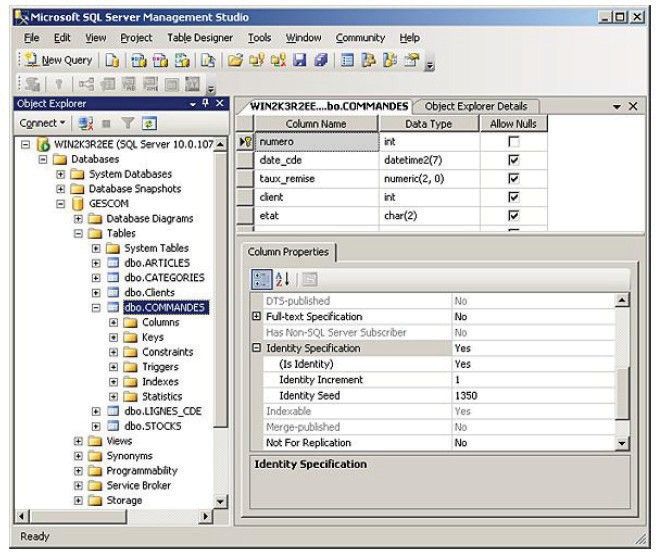
*Dans* *l'exemple* *suivant,* *une* *nouvelle* *categorie* *est* *ajoutee.* *La* *premiere* *insertion* *se* *solde* *par* *un* *echec* *car* *la* *propriete* *IDENTITY* *est* *active.*

*Apres* *sa* *desactivation* *par* *l'instruction* *SET* *IDENTITY\_INSERT* *categories* *ON,* *il* *est* *possible* *d'inserer* *la* *ligne* *de* *donnees.*



Il est possible de definir la propriete identity depuis SQL Server Management Studio en affichant l'ecran de modification ou de creation d'une table (**Design/Modifier** depuis le menu contextuel associe a la table).

S M



Q L S E R V E U R 2 0 0 8

I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1Partie

Il est possible d'utiliser les fonctions suivantes pour obtenir plus informations sur les types identites :

* IDENT\_INCR pour conna1tre le pas d'increment de la valeur identity.
* IDENT\_SEED pour conna1tre la valeur de depart fixee lors de la creation du type identity. Toutes ces fonctions ont pour objectif de permettre au programmeur de mieux controler la valeur generee afin de pouvoir la recuperer lorsqu'il s'agit de la cle primaire.

#### Les contraintes d'integrite

Les contraintes permettent de mettre en oeuvre l'integrite declarative en definissant des controles de valeur au niveau de la structure de la table elle meme.

La definition des contraintes se fait sous forme de script par l'intermediaire des instructions CREATE et ALTER TABLE. Il est egalement possible de les definir depuis SQL Server Management Studio.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Il est recommande de passer, lorsque cela est possible, par des contraintes d'integrite a la place de declencheurs de base de donnees car les contraintes d'integrites sont normalisees et elles limitent le codage donc le risque d'erreur.

Elles sont integrees dans la definition de la structure de la table et leur verification est plus rapide que l'execution d'un declencheur.

*Syntaxe*

ALTER TABLE nom Table

{ADDIDROP} CONSTRAINT nom Contrainte ...[I]

Il est possible d'obtenir des informations sur les contraintes d'integrites definies en interrogeant

**sys.key\_constraints** ou en utilisant les procedures stockees, **sp\_help** et **sp\_helpconstraint**. Les contraintes seront associees a la table ou a une colonne de la table, en fonction des cas.

Il est possible de verifier l'integrite des contraintes au travers de DBCC CHECKCONSTRAINT. Cet utilitaire prend tout son sens lorsque les contraintes d'une table ont ete desactivees puis reactivees sans verification des donnees dans la table.

1. **NOT** **NULL**

SQL Server considere la contrainte de nullite comme une propriete de colonne. La syntaxe est donc :

CREATE TABLE nom table (nom colonne type [{NULL I NOT NULL}] [? ...])

### NOT NULL

Specifie que la colonne doit etre valorisee en creation ou en modification.

Il est preferable de preciser systematiquement NULL ou NOT NULL, car les valeurs par defaut de cette propriete dependent de beaucoup de facteurs :

* + Pour un type de donnees defini par l'utilisateur, c'est la valeur specifiee a la creation du type.
  + Les types bit et timestamp n'acceptent que NOT NULL.
  + Les parametres de session ANSI\_NULL\_DFLT\_ON ou ANSI\_NULL\_DFLT\_OFF peuvent etre actives par la commande SET.
  + Le parametre de base de donnees 'ANSI NULL' peut etre positionne.

Depuis SQL Server 2005, il est possible de modifier la contrainte de NULL/NOT NULL avec une commande ALTER TABLE pour une colonne qui existe deja. Bien entendu, les donnees deja presentes dans la table doivent respecter ces contraintes.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1. **PRIMARY** **KEY**

# Partie

Cette contrainte permet de definir un identifiant cle primaire, c'est a dire une ou plusieurs colonnes n'acceptant que des valeurs uniques dans la table (regle d'unicite ou contrainte d'entite).

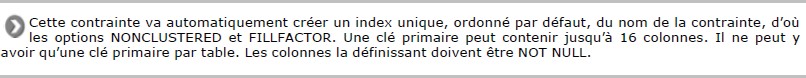
*Syntaxe*

[CONSTRAINT nom contrainte]PRIMARY KEY[CLUSTEREDINONCLUSTERED]

(nom colonne[,...])[WITH<$I[]FILLFACTOR> FILLFACTOR=x][ON groupe\_de\_fichiers]

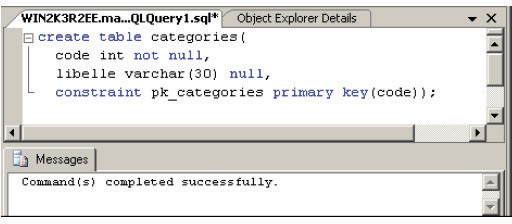
#### nomcontrainte

Nom permettant d'identifier la contrainte dans les tables systeme. Par defaut, SQL Server donnera un nom peu facile a manipuler.



*Exemples*

*Table* *categorie,* *identifiant* *CODE\_CAT* *:*

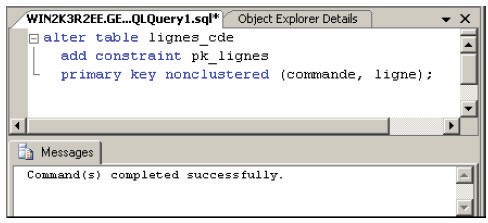


S Q L S E R V E U R 2 0 0 8

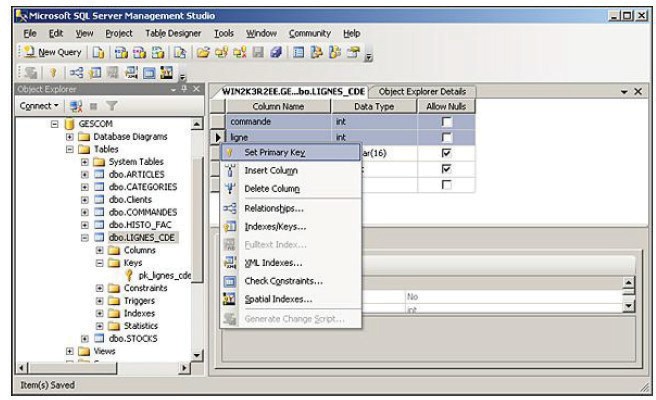
M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

*Ajout* *de* *la* *cle* *primaire* *a* *la* *table* *Lignes\_cdes* *(un* *index* *ordonne* *existe* *deja* *sur* *Numero\_cde}* *:*



*Gestion* *de* *la* *cle* *primaire* *par* *SQL* *Server* *Management* *Studio* *:*



Il n'est pas possible de supprimer la cle primaire si :

* elle est referencee par une contrainte de cle etrangere.
* un index xml primaire est defini sur la table.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

1. **UNIQUE**

# Partie

Cette contrainte permet de traduire la regle d'unicite pour les autres cles uniques de la table ou identifiants cles secondaires.

Cette contrainte possede les memes caracteristiques que PRIMARY KEY a deux exceptions pres :

* + il est possible d'avoir plusieurs contraintes UNIQUE par table ;
  + les colonnes utilisees peuvent etre NULL (non recommande !).

Lors de l'ajout d'une contrainte d'unicite sur une table existante, SQL Server s'assure du respect de cette contrainte par les lignes deja presentes avant de valider l'ajout de la contrainte.

La gestion de cette contrainte est assuree par un index de type UNIQUE. Il n'est pas possible de supprimer cet index par l'intermediaire de la commande DROP INDEX. Il faut supprimer la contrainte par l'intermediaire de ALTER TABLE.

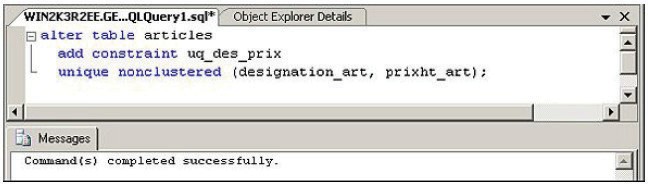
*Syntaxe*

[CONSTRAINT nom contrainte] UNIQUE [CLUSTERED I NONCLUSTERED ]

(nom colonne [,...]) [WITH FILLFACTOR=x] [ON groupe\_de\_fichiers]

*Exemple*

*L'association* *des* *colonnes* *Designation* *et* *Prixht* *doit* *etre* *unique* *dans* *la* *table* *ARTICLES* *:*

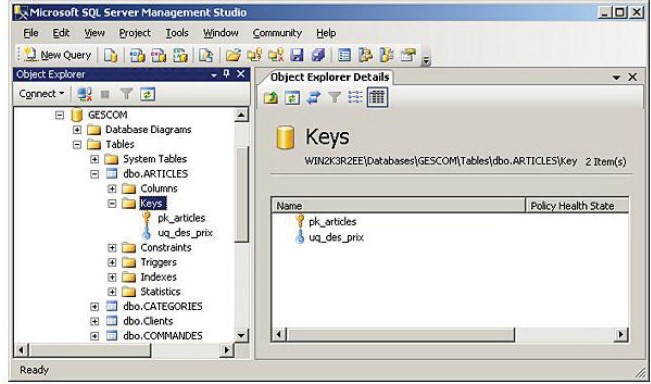


S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

*Gestion* *des* *cles* *secondaires* *par* *SQL* *Server* *Management* *Studio* *:*

# Partie



1. **REFERENCES**

Cette contrainte traduit l'integrite referentielle entre une cle etrangere d'une table et une cle primaire ou secondaire d'une autre table.

*Syntaxe*

CONSTRAINT nom\_contrainte [FOREIGN KEY (colonne[,\_])]

REFERENCES *table* [ ( *colonne* [<+>,... ] ) ]

[ ON DELETE { CASCADE I **NO** **ACTION** I SET NULL I SET DEFAULT } ] [ ON UPDATE { CASCADE I **NO** **ACTION** I SET NULL I SET DEFAULT } ]



L'option de cascade permet de preciser le comportement que doit adopter SQL Server lorsque l'utilisateur met a jour ou tente de supprimer une colonne referencee. Lors de la definition d'une contrainte de reference par les instructions CREATE TABLE ou ALTER TABLE, il est possible de preciser les clauses ON DELETE et ON UPDATE.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

### NO ACTION

Valeur par defaut de ces options. Permet d'obtenir un comportement identique a celui present dans les versions precedentes de SQL Server.

### ON DELETE CASCADE

Permet de preciser qu'en cas de suppression d'une ligne dont la cle primaire est referencee par une ou plusieurs lignes, toutes les lignes contenant la cle etrangere faisant reference a la cle primaire sont egalement supprimees. Par exemple, avec cette option, la suppression d'une ligne d'information dans la table des commandes provoque la suppression de toutes les lignes d'information de la table lignes\_commande.

### ON UPDATE CASCADE

Permet de demander a SQL Server de mettre a jour les valeurs contenues dans les colonnes de cles etrangeres lorsque la valeur de cle primaire referencee est mise a jour.

### SET NULL

Lorsque la ligne correspondant a la cle primaire dans la table referencee est supprimee, alors la cle etrangere prend la valeur null.

### SET DEFAULT

Lorsque la ligne correspondant a la cle primaire dans la table referencee est supprimee, alors la cle etrangere prend la valeur par defaut definie au niveau de la colonne.



Toute reference circulaire est interdite dans les declenchements des cascades.

La contrainte de reference ne cree pas d'index. Il est recommande de le creer par la suite.

Bien que SQL Server ne comporte pas de limite maximale en ce qui concerne le nombre maximal de contraintes de cle etrangere definies sur chaque table, il est recommande par Microsoft de ne pas depasser les 253. Cette limite est a respecter en ce qui concerne le nombre de cles etrangeres definies sur la table et le nombre de cles etrangeres qui font reference a la table. Au dela de cette limite, il peut etre interessant de se pencher a nouveau sur la conception de

la base pour obtenir un schema plus optimum.

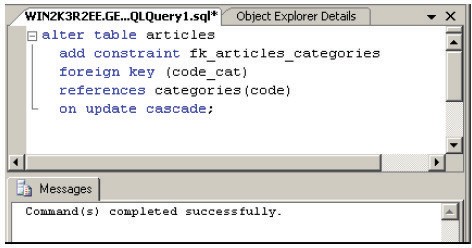
S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

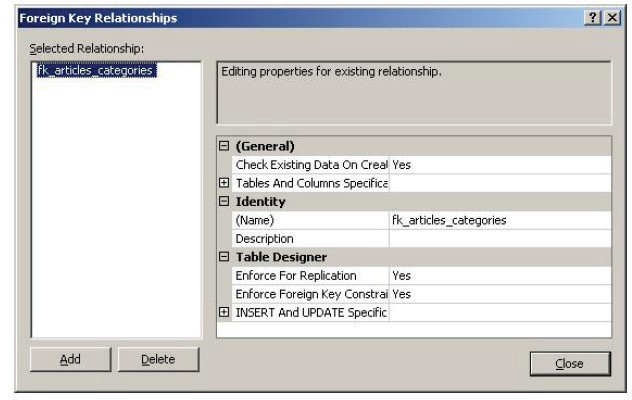
*Exemple*

*Creation* *de* *la* *cle* *etrangere* *code\_cat* *dans* *la* *table* *ARTICLES* *:*

# Partie



*Gestion* *des* *cles* *etrangeres* *par* *SQL* *Server* *Management* *Studio* *:*



S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

1. **DEFAULT**

La valeur par defaut permet de preciser la valeur qui va etre positionnee dans la colonne si aucune information n'est precisee lors de l'insertion de la ligne.

Les valeurs par defaut sont particulierement utiles lorsque la colonne n'accepte pas les valeurs NULL car elles garantissent l'existence d'une valeur.

Il faut toutefois conserver a l'esprit que la valeur par defaut est utilisee uniquement lorsqu'aucune valeur n'est precisee pour la colonne dans l'instruction INSERT. Il n'est pas possible de completer ou d'ecraser une valeur saisie

par l'utilisateur. Pour realiser ce type d'operation, il est necessaire de developper un declencheur de base de donnees.

Une valeur par defaut peut etre definie pour toutes les colonnes a l'exception des colonnes de type timestamp ou bien celles qui possedent un type identity.

*Syntaxe*

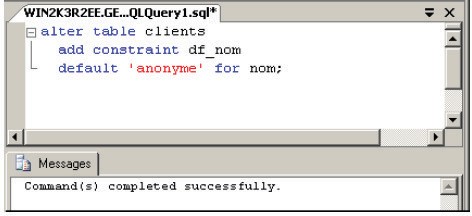
[CONSTRAINT Nom contrainte] DEFAULT valeur [FOR nom colonne]

#### valeur

La valeur doit etre exactement du meme type que celui sur lequel est definie la colonne. Cette valeur peut etre une constante, une fonction scalaire (comme par exemple : USER, CURRENT\_USER, SESSION\_USER, SYSTEM\_USER.) ou bien la valeur NULL.

*Exemple*

*Valeur* *par* *defaut* *pour* *le* *Nom* *du* *client* *:*

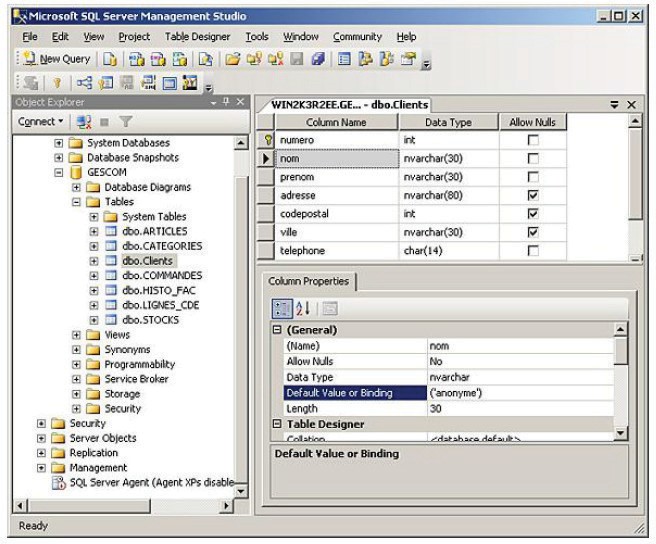


S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

*Definition* *d'une* *valeur* *par* *defaut* *depuis* *SQL* *Server* *Management* *Studio* *:*

# Partie



1. **CHECK**

La contrainte de validation ou contrainte CHECK, permet de definir des regles de validation mettant en rapport des valeurs issues de differentes colonnes de la meme ligne. Ce type de contrainte permet egalement de s'assurer que les donnees respectent un format precis lors de leur insertion et mise a jour dans la table. Enfin, par l'intermediaire d'une contrainte de validation, il est possible de garantir que la valeur presente dans la colonne appartient a un domaine precis de valeurs.

*Syntaxe*

[CONSTRAINT Nom contrainte] CHECK [NOT FOR REPLICATION]

(expression booleenne)

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

### NOT FOR REPLICATION

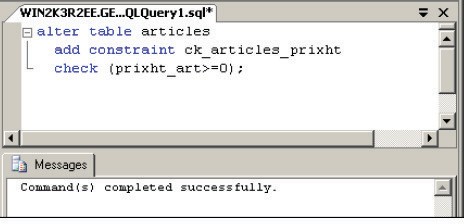
Permet d'empecher l'application de la contrainte lors de la replication.

# Partie

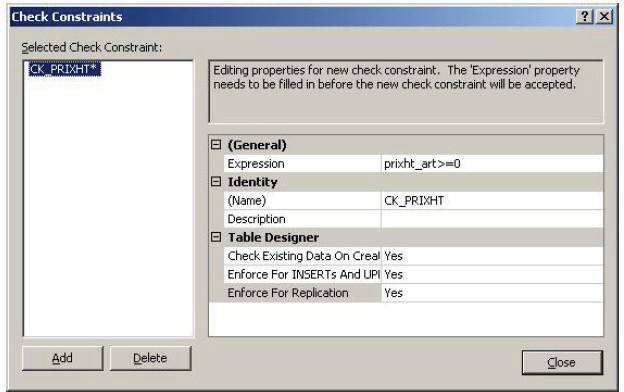


*Exemple*

*Mise* *en* *oeuvre* *du* *contr6le* *du* *prix* *positif* *de* *l'article* *:*



*Gestion* *des* *contraintes* *CHECK* *par* *SQL* *Server* *Management* *Studio* *:*



S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

#### Gerer les index

**Utilisation** **des** **index** **ou** **pourquoi** **indexer** **?**

L'objectif des index est de permettre un acces plus rapide a l'information dans le cadre des extractions (SELECT) mais egalement dans le cadre des mises a jour (INSERT, UPDATE, DELETE) en reduisant le temps necessaire a la localisation

de la ligne. Cependant, les index vont etre coOteux dans le cas de mise a jour de la valeur contenue dans la colonne indexee.

Une bonne strategie d'indexation doit prendre en consideration ces differents points et peut deja en deduire deux regles :

* + Il est preferable d'avoir trop peu d'index que trop d'index. Dans le cas de multiples index, les gains en acces a l'information sont annules par les temps necessaires pour mettre a niveau les index.
  + Les index doivent etre le plus "large" possible afin de pouvoir etre utilises par plusieurs requetes.

Enfin, il faut s'assurer que les requetes utilisent bien les index qui sont definis.

Les requetes doivent egalement etre ecrites pour manipuler le minimum d'informations de la fa�on la plus explicite possible.

Par exemple, dans le cas d'une projection, il est preferable de lister les colonnes pour lesquelles l'information doit etre affichee a la place du caractere generique \*.

Pour les restrictions, il est preferable de faire des comparaisons entre constantes et la valeur contenue dans une colonne.

Par exemple, si la table des Articles contient le prix HT de chaque article, pour extraire la liste des articles dont le prix TTC est inferieur ou egal a 100 €, il est preferable d'ecrire la condition suivante prixht<=100/1.196 a la place de prixht\*1.196<=100. En effet, dans le second cas, le calcul est effectue pour chaque article, tandis que dans le premier, le calcul est fait une fois pour toute. Il faut egalement prendre en compte le fait que les donnees sont stockees dans les index et donc qu'ils vont occuper un espace disque non negligeable. Le niveau feuille d'un index ordonne (clustered) contient l'ensemble des donnees. Pour un index non ordonne, le niveau feuille de l'index contient une reference directe vers la ligne d'information liee a la cle

de l'index.

Les autres niveaux de l'index sont utilises pour naviguer dans l'index et arriver ainsi tres rapidement a l'information. Il est possible d'ajouter de l'information au niveau des feuilles de l'index, sans que ces colonnes soient prises en compte par l'index. Cette technique est pratique dans le cas de la definition d'index qui couvre des requetes. Par exemple, il est

necessaire d'extraire la liste des codes postaux et des villes de la table des clients. Pour cela, un index non ordonne est defini par rapport a la colonne des codes postaux, et la colonne qui

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

represente le nom de la ville est ajoute au niveau feuille. Ainsi, l'index couvre la requete qui est capable de produire le resultat sans faire d'acces a la table.

#### Qu'est ce qu'un index ?

La notion d'index est deja connue de tous. En effet, chacun a deja utilise l'index d'un livre pour aboutir directement a lapage ou aux pages d'informations recherchees. Peut etre avez vous d'ailleurs utilise l'index de ce livre pour aboutir directement a cette explication en parcourant l'index a la recherche du mot cle "index".

Si pour l'index, derriere chaque mot cle n'appara1t qu'un seul numero de page, on parle alors d'index unique.

Les index que SQL Server propose de mettre en place sont tres proches des index que l'on peut trouver dans les livres.

En effet, il est possible de parcourir l'ensemble de l'index pour retrouver toutes les informations, comme il est possible de lire un livre a partir de l'index, au lieu de suivre l'ordre propose par la table des matieres.

Il est egalement possible d'utiliser l'index pour acceder directement a une information precise. Afin de garantir des temps d'acces a l'information homogenes, SQL Server structure l'information sous forme d'arborescence autour de la propriete indexee. C'est d'ailleurs la demarche que l'on adopte lorsque l'on parcourt un index en effectuant une premiere localisation de l'information par rapport au premier caractere, puis un parcours sequentiel afin de localiser le mot cle recherche.

Maintenant, imaginez un livre sur lequel il est possible de definir plusieurs index : par rapport aux mots cles, par rapport aux themes, par rapport au type de manipulations que l'on souhaite faire... Cette multitude d'index, c'est ce que propose SQL Server en donnant la possibilite de creer plusieurs index sur une table.

Parmi tous les index du livre, un en particulier structure le livre : c'est la table des matieres, qui peut etre per�ue comme un index sur les themes. De meme, SQL Server propose de structurer physiquement les donnees par rapport a un index. C'est l'index CLUSTERED.

#### Organiser ou non les donnees ?

SQL Server propose deux types d'index : les index organises, un au maximum par table, qui reorganisent physiquement la table et les index non organises.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

La definition ou bien la suppression d'un index non organise n'a aucune influence sur l'organisation des donnees dans la table. Par contre, la definition ou la suppression d'un index organise aura des consequences sur la structure des index non organises.

#### Table sans index organise

Si une table possede uniquement ce type d'index, alors les informations sont stockees au fil de l'eau sans suivre une organisation quelconque.

Ce choix est particulierement adapte, lorsque :

* + la table stocke de l'information en attendant un partitionnement,
  + les informations vont etre tronquees,
  + des index sont frequemment crees ou bien supprimes,
  + un chargement en bloc de donnees a lieu,
  + les index sont crees apres le chargement des donnees, et leur creation peut etre parallelisee,
  + les donnees sont rarement modifiees (UPDATE) afin de conserver une structure solide,
  + l'espace disque utilise par l'index est minimise, ce qui permet de definir des index a moindre coOt.

Cette solution est un choix performant pour l'indexation d'une table non presente sur un serveur OLTP, comme un serveur dedie a l'analyse des informations.

#### Les index organises

Sur chaque table, il est possible de definir un et un seul index organise. Ce type d'index permet d'organiser physiquement les donnees contenues dans la table selon un critere particulier. Par exemple, un index organise peut etre defini sur la cle primaire.

Ce type d'index est coOteux en temps et en espace disque pour le serveur lors de sa construction ou de sa reconstruction.

Si ce type d'index est defini sur une table deja valorisee, sa construction sera longue. Elle sera d'ailleurs d'autant plus longue si des index non ordonnes existent deja.

Idealement, et afin d'eviter une maintenance trop coOteuse de l'index ordonne, il est defini sur une colonne qui contient des donnees statiques et qui occupe un espace limite, comme la cle primaire par exemple.

La definition d'un index organise sur une colonne non stable, comme le nom d'une personne ou son adresse, conduit irremediablement a une degradation significative des performances.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

**Les** **index** **non** **organises**

# Partie

Suivant qu'elles soient definies sur une table munie ou non d'un index organise, les feuilles de l'index non organise ne font pas reference a la ou les lignes d'informations de la meme fa�on. Tout d'abord, dans le cas ou la table ne possede pas d'index organise, le RID (Row IDentifier) de la ligne d'information est stocke au niveau des feuilles de l'index. Ce RID correspond a l'adresse physique (au sens SQL Server) de la ligne. Si au contraire la table possede un index organise, alors, au niveau de la feuille de l'index non organise, est stockee la cle de la ligne d'information recherchee. Cette cle correspond au critere de recherche de l'index organise.

Les index non organises sont a privilegier pour definir un index qui couvre une ou plusieurs requetes. Avec ce type d'index, la requete trouve dans l'index l'ensemble des informations dont elle a besoin et evite ainsi un acces inutile a la table, puisque seul l'index est manipule. Les performances sont alors considerablement ameliorees car le volume de donnees manipule est beaucoup plus leger.

Bien entendu, chaque requete peut etre optimisee a l'aide de cette technique, mais ce n'est pas non plus l'objectif recherche car la multitude d'index serait coOteuse a maintenir.

#### Index et calcul d'agregat

Pour les tables volumineuses, des index doivent etre mis en place afin de s'assurer que les requetes courantes ne provoquent pas un balayage complet de table mais s'appuient simplement sur les index. Ce point est particulierement important pour les calculs d'agregat qui necessitent une operation de tri avant de pouvoir effectuer le calcul. Si un index permet de limiter le tri a effectuer, alors le gain de performance est tres net.

Toujours dans cette optique d'optimisation des performances lors de l'acces aux donnees, il est possible de definir des index sur des colonnes d'une vue, meme si le resultat present dans la colonne est le resultat d'un calcul d'agregat.

L'index defini sur une vue est limite a 16 colonnes et 900 octets de donnees pour chaque entree de l'index.

Contrairement a l'inclusion de colonnes non indexees dans les feuilles de l'index, les index definis sur une vue peuvent contenir le resultat d'un calcul d'agregat.

#### Creer un index

Un index peut etre cree n'importe quand, qu'il y ait ou non des donnees dans la table. Cependant, dans le cas ou des donnees doivent etre importees, il est preferable d'importer les donnees en premierpuis de definir les index. Dans le cas contraire (les index sont definis avant une importation majeure de donnees), il est necessaire de reconstruire les index afin de garantir une repartition equilibree des informations dans l'index.

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

#### Syntaxe

CREATE [ UNIQUE ] [ CLUSTERED I NONCLUSTERED ] INDEX nom\_*index*

ON { nom\_*table* I nom\_*vue* } ( colonne [ ASC I DESC ] [ ,...n ]) [INCLUDE (colonne[ ,...n ])]

[ WITH [ PAD\_INDEX = {ON I OFF} ] [,FILLFACTOR = x] [,IGNORE\_DUP\_KEY = {ON I OFF} ] [,DROP\_EXISTING = {ON I OFF} ] [,ONLINE = {ON I OFF} ]

[,STATISTICS\_NORECOMPUTE = {ON I OFF} ] [,SORT\_IN\_TEMPDB = {ON I OFF} ]

[ ON *groupe\_fichier* ]

# Partie

### UNIQUE

Precise que la ou les colonnes indexees ne pourront pas avoir la meme valeur pour plusieurs lignes de la table.

#### CLUSTERED ou NONCLUSTERED

Avec un index CLUSTERED (ordonne), l'ordre physique des lignes dans les pages de donnees est identique a l'ordre de l'index. Il ne peut y en avoir qu'un par table et il doit etre cree en premier. La valeur par defaut est NONCLUSTERED.

### INCLUDE

Avec cette option, il est possible de dupliquer l'information pour inclure une copie des colonnes specifiee en parametre directement dans l'index. Cette possibilite est limitee aux index non organises et les informations sont stockees au niveau feuille. Il est possible d'inclure dans l'index de 1 a 1026 colonnes de n'importe quel type hormis varchar(max), varbinary(max) et nvarchar(max). Cette option INCLUDE permet de definir des index qui couvrent la requete, c'est a direque la requete va parcourir uniquement l'index pour trouver la totalite des besoins qui lui sont necessaires.

#### FILLFACTOR=x

Precise le pourcentage de remplissage des pages d'index au niveau feuille. Cela permet d'ameliorer les performances en evitant d'avoir des valeurs d'index consecutives qui ne seraient plus physiquement contigues. La valeur par defaut est 0 ou fixee par sp\_configure. Pour x = 0, le niveau feuille est rempli a 100%, de l'espace est reserve au niveau non feuille.

Pour x entre 1 et 99, le pourcentage de remplissage au niveau feuille est x%, de l'espace est reserve au niveau non feuille. Pour x = 100 les niveaux feuille et non feuille sont remplis.

PAD\_INDEX

S Q L S E R V E U R 2 0 0 8

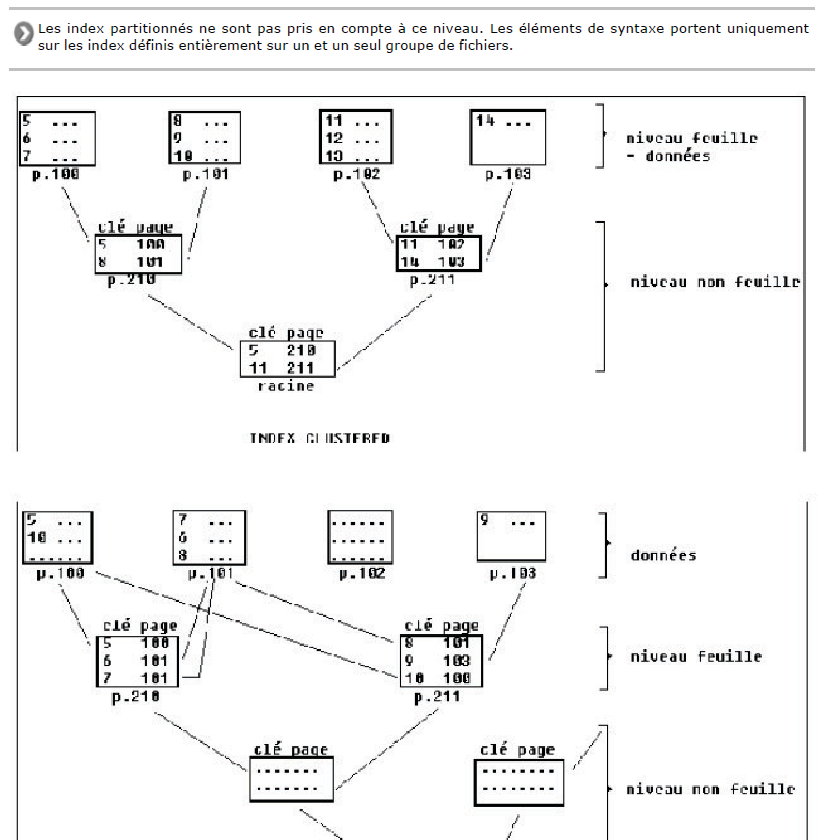
M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

Precise le taux de remplissage des pages non feuille de l'index. Cette option n'est utilisable qu'avec FILLFACTOR dontla valeur est reprise.

#### Groupefichier

Groupe de fichiers sur lequel est cree l'index.



S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

### IGNORE\_DUP\_KEY

Cette option autorise des entrees doubles dans les index uniques. Si cette option est levee, un avertissement est genere lors de l'insertion d'un doublon et SQL Server ignore l'insertion de ligne. Dans le cas contraire, une erreur est generee.

### DROP\_EXISTING

Precise que l'index deja existant doit etre supprime.

### ONLINE

Lorsque cette option est activee (ON) lors de la construction de l'index, les donnees de la table restent accessibles en lecture et en modification pendant la construction de l'index. Cette option est disponible a partir de SQL Server 2005 et elle est desactivee par defaut.

### STATISTICS\_NORECOMPUTE

Les statistiques d'index perimes ne sont pas recalculees, et il faudra passer par la commande

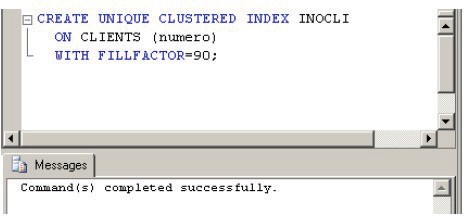
**UPDATE** **STATISTICS** pour remettre a jour ces statistiques.

#### Si le serveur sur lequel s'execute SQL Server possede plusieurs processeurs, alors la creation d'index peut etre parallelisee afin de gagner du temps pour la construction d'index sur des tables de grandes dimensions. La mise en place d'un plan d'execution parallele pour la construction d'un index prend en compte le nombre de processeurs du serveur fixe par l'option de configuration max degree of parallelism (sp\_configure) et le nombre de processeurs

**n'etant** **pas** **deja** **trop** **charges** **par** **des** **threads** **SQL** **Server.**

*Exemples*

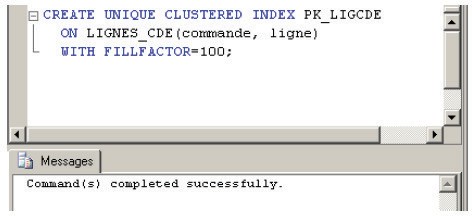
*Creation* *d'un* *index* *ordonne* *:*



S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

*Creation* *d'un* *index* *ordonne* *sur* *des* *donnees* *deja* *triees* *:*



1. **Supprimer** **un** **index**

# Partie

Seuls les index definis avec l'instruction CREATE INDEX peuvent etre supprimes avec DROP INDEX. Un index peut etresupprime lorsque sa presence ne permet pas un gain de performances significatif en comparaison du coOt de maintenance.

Si la suppression intervient dans le cadre d'une reconstruction de l'index, il est alors preferable d'activer l'option DROP\_EXISTING de l'instruction CREATE INDEX car elle offre de meilleures performances.

#### Syntaxe

DROP INDEX nomIndex ON nomTableI

#### Reconstruire un index

La commande DBCC DBREINDEX est encore disponible pour des raisons de compatibilite. Il est preferable d'utiliser la commande ALTER INDEX qui permet de maintenir les index.

La commande ALTER INDEX permet de reconstruire un index en particulier ou tous les index associes a une table. Lors de la reconstruction de l'index, il est possible de preciser le facteur de remplissage des pages feuilles.

#### Syntaxe

ALTER INDEX { *nomIndex* I ALL }

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

ON *nomTable* REBUILD [WITH](

[PAD\_INDEX = { ON I OFF },] [FILLFACTOR = *x* ,] [SORT\_IN\_TEMPDB = { ON I OFF },] [IGNORE\_DUP\_KEY = { ON I OFF },]

[STATISTICS\_NORECOMPUTE = { ON I OFF }] [ONLINE = { ON I OFF },]]

[I]

# Partie

### FILLFACTOR

Permet de specifier le pourcentage de remplissage des pages au niveau feuille de l'index.

### PAD\_INDEX

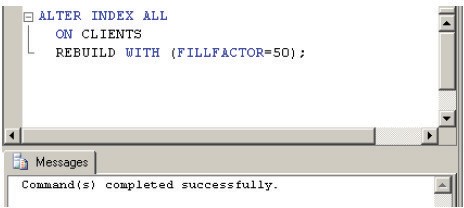
Permet de reporter dans les pages non feuilles de l'index, le meme niveau de remplissage que celui precise par l'intermediaire de FILLFACTOR.

Les autres options de l'instruction ALTER INDEX possedent la meme signification que celles utilisees avec l'instruction CREATE INDEX. Il est a remarquer que l'option ONLINE prend tout son sens dans ce cas precis car elle permet de reconstruire l'index en laissant les utilisateurs travailler avec les donnees de la table sous jacente.

Certes lesoperations seront plus lentes mais elles ne seront pas bloquees. Si la reconstruction d'index est planifiee sur un moment de faible utilisation du serveur elle peut meme etre transparente pour les quelques utilisateurs qui travaillent sur la base a ce moment.

*L'exemple* *ci* *dessous*

*illustre* *la* *reconstruction* *de* *tous* *les* *index* *d'une* *table* *:*

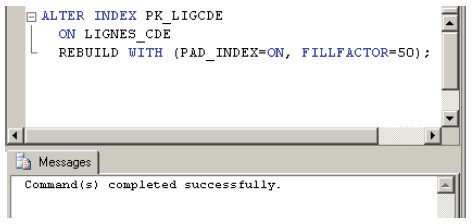


S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

*Ce* *second* *exemple* *illustre* *la* *reconstruction* *d'un* *index* *en* *specifiant* *une* *valeur* *pour* *l'option* *FILLFACTOR,* *ainsi* *que* *la* *prise* *en* *compte* *du* *facteur* *de* *remplissage* *dans* *les* *fonctions* *non* *feuilles.*



#### Mettre a jour les statistiques

SQL Server utilise des informations sur la distribution des valeurs de cles pour optimiser les requetes. Ces informations doivent etre mises a jour apres des modifications importantes de donnees.

Bien que la procedure manuelle de creation et de mise a jour des statistiques soit exposee ici, il est tres nettement preferable de configurer la base pour une creation et une mise a jour automatique des statistiques. En effet, trop souvent la degradation des performances d'un serveur est due en partie ou en totalite a des statistiques non mises a jour.

*Syntaxe*

UPDATE STATISTICS nomtable [,nomindex]

[WITH {FULLSCANISAMPLE n {PERCENTIROWS}IRESAMPLE}]

Si le nom d'index est omis, tous les index sont pris en compte.

### FULLSCAN

Les statistiques sont creees a partir d'un balayage complet de la table, soit 100% des lignes. SAMPLE n{PERCENTIROWS}

Les statistiques sont etablies a partir d'un echantillon representatif des informations contenues dans la table. Cet echantillon peut etre exprime en pourcentage ou bien en nombre de lignes. Si la taille de l'echantillon precise n'est pas suffisante, SQL Server corrige de lui meme

cette taille pour s'assurer d'avoir parcouru environ 1000 pages de

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

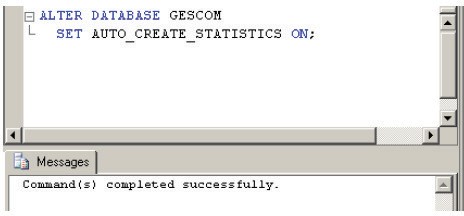
donnees. C'est le mode d'echantillonnage de statistiques par defaut.

### RESAMPLE

Permet de redefinir les statistiques a partir d'un nouvel echantillonnage.

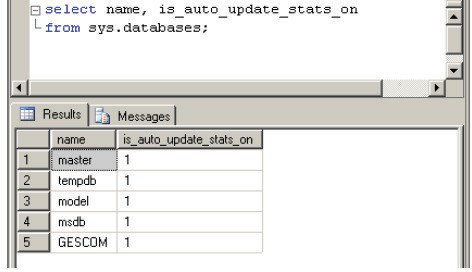
Les statistiques peuvent egalement etre mises a jour de fa�on automatique. Cette option doit etre definie, soit lors de la construction de la base a l'aide de la commande **ALTER** **DATABASE**, soit en utilisant la procedure stockee **sp\_autostats**.

*Configuration* *de* *la* *base* *pour* *une* *mise* *a* *jour* *automatique* *des* *statistiques* *d'index* *:*



En mode automatique, c'est le moteur qui se charge de calculer les statistiques manquantes, de maintenir a jour les statistiques en fonction des operations faites sur les donnees, mais aussi de supprimer les statistiques inutiles.

Il est possible de savoir si une base est configuree en mise automatique des statistiques en interrogeant la colonne **is\_auto\_update\_stats\_on** de la table **sys.databases** ou bien en visualisant la valeur de la propriete **IsAutoUpdateStatistics** de la fonction **databasepropertyex**.

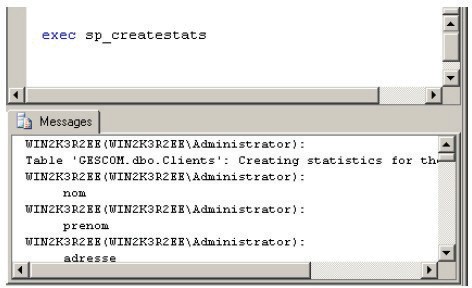


S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

*Creation* *des* *statistiques* *de* *tous* *les* *index* *sur* *les* *donnees* *non* *systeme* *de* *la* *base* *de* *donnees* *GESCOM* *:*



#### Informations sur les index

Des informations concernant la structure des index peuvent etre obtenues par les procedures stockees **sp\_help** ou **sp\_helpindex**.

Des informations concernant la taille et la fragmentation des tables et des index peuvent etre obtenues par l'intermediaire de la fonction **sys.dm\_db\_index\_physical\_stats**.

##### *Syntaxe*

dm\_db\_index\_physical\_stats(idBase I NULL, idObjet I NULL,

idIndex I NULL I 0, numeroPartition I NULL , mode I NULL I DEFAULT)

#### idBase

Identifiant de la base de donnees. Il est possible d'utiliser la fonction **db\_id()** pour conna1tre celui de la base

S Q L S E R V E U R 2 0 0 8

M I S E E N S E R V I C E D ' U N E B A S E D E D O N N E E

# Partie

courante. La valeur NULL prend en compte l'ensemble des bases definies sur le serveur et implique d'utiliser la valeur NULL pour l'identifiant de l'objet, de l'index et de la partition.

#### idObjet

Identifiant de l'objet (table ou vue) sur lequel on souhaite des informations. Cet identifiant peut etre obtenu en faisant appel a la fonction **object\_id()**. L'utilisation de la valeur NULL permet de signaler que l'on souhaite des informations sur toutes les tables et vues de la base courante. Cela implique egalement d'utiliser la valeur NULL pour l'option

#### idIndex et numeroPartition idIndex

Identifiant de l'index a analyser. Si la valeur NULL est specifiee alors l'analyse porte sur la totalite

des index de la table.

#### numeroPartition

Numero de la partition concernee. Si la valeur NULL est specifiee alors toutes les partitions sont prises en compte.

#### mode

Precise le mode d'obtention des informations : DEFAULT, NULL, LIMITED ou DETAILED. La valeur par defaut (NULL) correspond a LIMITED

La procedure stockee **sp\_spaceused** permet de conna1tre l'espace disque utilise par les index. La fonction **INDEXPROPERTY** permet d'obtenir les differentes informations relatives aux index.