



# **Manuel d'exploitation SQLserver 2012**

**Belkacem ABDELLAZIZ**

**Centre de formation spécialisé dans les systèmes d'informations**

**2023**

# **Manuel d'exploitation SQL server 2012**

**BELKACEM ABDELLAZIZ**

**CENTRE DE FORMATION SPECIALISE DANS LES SYSTEMES  
D'INFORMATIONS**

**UNE DISSERTATION PRÉSENTÉ EN VUE DE REMPLIR  
PARTIELLEMENT LES CONDITIONS D'ACCÈS  
AU MARCHÉ DE L'EMPLOI**

**2023**

## **Statement of Originality**

I hereby certify that the work embodied in this report is the result of original research and has not been submitted for any other institution.

**Operating manual of SQL server 2012 © 2023 by Belkacem ABDELLAZIZ is licensed under Attribution-NonCommercial 4.0 International. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/4.0/>**

.....27/10/2023  
Date

Belkacem ABDELLAZIZ  
Your Name

## **Student Declaration Statement**

This technical manual provide some tips and solutions to guide and help other students to optimaly manage, administrate SQL server database and more importantly resolutions for incidental events that may be occured.

I have prepared this report with love and put all my efforts on it, in order to make a better and comfortable place for all candidates who really wanted to succeed in their lives without paying attentions to money benefits.The part of humanity that we've lost for decates, need to be awakened, especially in professional environment to get at the same time height productivity and good relationship with a higher supervisor.

I really hope that this technical manual is going to help other youthful people in needs.

.....27/10/2023  
Date

Belkacem ABDELLAZIZ  
Student's Name

# Table de matière

<b>Abstract</b>	vii
<b>Acronyms</b>	viii
<b>Lists of Figures</b>	xiv
<b>1 Présentation de SQL server 2012</b>	1
1.1 Comprendre le fonctionnement de SQL server . . . . .	1
1.1.1 L'architecture OLTP (Client/Serveur) . . . . .	2
1.1.2 L'architecture OLTP 3 tiers . . . . .	2
1.1.3 l'architecture OLAP . . . . .	3
1.2 Présentation des différents composants de SQL server . . . . .	3
1.3 l'architecture de SQL server 2012 . . . . .	6
1.3.1 Qu'est ce qu'une instance . . . . .	6
1.3.2 Architecture d'une instance SQL server . . . . .	7
1.3.3 Les outils de SQL server 2012 . . . . .	7
1.4 Comprendre l'architecture d'une base de données . . . . .	8
1.4.1 Le rôle d'une base de données . . . . .	8
1.4.2 Architecture d'une base de données . . . . .	9
1.4.3 Les bases de données systèmes . . . . .	9
<b>2 Installation manuelle de SQL server 2012</b>	10
2.1 Les prérequis du processus d'installation de SQL server 2012 . . . . .	10
2.2 Les étapes du processus d'installation manuelle de SQL server 2012 . . . . .	13
<b>3 Installation silencieuse de SQL server 2012</b>	23
3.1 Création du fichier de configuration .ini . . . . .	23
3.2 Installation silencieuse . . . . .	24
3.2.1 Les étapes d'installation : . . . . .	24
<b>4 Les outils d'administration SSMS/SQLcmd</b>	27
4.1 Présentation de SQL server management studio (SSMS) . . . . .	27
4.1.1 Qu'est ce que c'est SQL server management studio . . . . .	27
4.1.2 Se familiariser avec l'interface graphique SSMS . . . . .	29
4.1.3 Présentation de l'arborescence au niveau répertoire de l'instance . . . . .	31

4.2	Presentation de SQLcmd . . . . .	33
4.2.1	Qu'est ce que SQLcmd . . . . .	33
4.2.2	SQLcmd ligne de commande . . . . .	33
<b>5</b>	<b>Configuration de SQL server 2012</b>	<b>35</b>
5.1	Introduction . . . . .	35
5.2	Le gestionnaire de configuration SQL server . . . . .	36
5.2.1	La gestion des différents Services . . . . .	36
5.2.2	La gestion des protocoles réseau du Serveur et du Client	37
5.3	Le gestionnaire de service Windows . . . . .	39
5.4	Inscription des serveurs . . . . .	40
5.5	Configuration et administration des serveurs . . . . .	41
5.5.1	Changement du mot de passe de l'administrateur (sa) . .	41
5.5.2	Dimensionnement du cache SQL server . . . . .	41
<b>6</b>	<b>Les bases de données sous SQL server 2012</b>	<b>42</b>
6.1	Architecture global d'une base de données . . . . .	42
6.1.1	Les fichiers de données . . . . .	44
6.2	Principe de fonctionnement des fichiers journaux . . . . .	46
6.3	Création d'une base de données . . . . .	47
6.3.1	En mode T-SQL . . . . .	47
6.3.2	En mode interface graphique . . . . .	49
6.4	La gestion d'une base de données . . . . .	50
6.4.1	Modifier un fichier data . . . . .	50
6.4.2	Ajouter un fichier data . . . . .	51
6.4.3	Suppression d'un fichier data . . . . .	51
6.4.4	Libération de l'espace de stockage . . . . .	51
6.4.5	Rapport d'utilisation du disque . . . . .	54
6.5	Les groupes de fichiers . . . . .	54
6.5.1	Structure logique . . . . .	54
6.5.2	Les types de groupe de fichiers . . . . .	55
6.5.3	Principe d'utilisation des groupes de fichiers . . . . .	56
6.6	Le partitionnement . . . . .	56
6.6.1	Objectif . . . . .	56
6.6.2	Le mécanisme de partitionnement . . . . .	57
6.7	Configuration d'une base de données SQL server . . . . .	62
6.7.1	Objectif . . . . .	62
6.7.2	Les options de configuration de la base de données . .	63
6.8	Les transactions . . . . .	66
6.8.1	C'est quoi une transaction ? . . . . .	66
6.8.2	Description d'une transaction explicite . . . . .	66
6.8.3	Les verrous sous SQL server . . . . .	67
6.8.4	Cas pratique d'utilisation d'une transaction explicite .	69
6.8.5	Les niveaux d'isolement sous SQL server . . . . .	72
6.9	Les modes de recuperation sous SQL server . . . . .	73
6.10	Fonction des fichiers journaux . . . . .	73
6.10.1	Le mode simple( <b>SIMPLE</b> ) . . . . .	73

6.10.2 Le mode complet ( <b>FULL</b> ) . . . . .	73
6.10.3 Le mode journalisé en bloc ( <b>BULK LOGGED</b> ) . . . . .	74
6.11 La compression des données . . . . .	74
6.11.1 Gérer la compression de données . . . . .	75
<b>7 Sauvegarde sous SQL server 2012</b>	<b>77</b>
7.1 Introduction à la sauvegarde . . . . .	77
7.1.1 Pourquoi sauvegarder ? . . . . .	77
7.1.2 Les caractéristiques de la sauvegarde . . . . .	78
7.2 La sauvegarde complète . . . . .	78
7.3 La sauvegarde différentielle . . . . .	79
7.4 La sauvegarde du journal des transactions . . . . .	80
7.5 Mise en oeuvre de la sauvegarde complète . . . . .	81
7.5.1 Destination des sauvegardes . . . . .	81
7.5.2 Les privilèges de sauvegarde . . . . .	81
7.5.3 L'instruction BACKUP . . . . .	83
7.5.4 Sauvegarde avec mise en miroir . . . . .	85
7.6 Mise en oeuvre de la sauvegarde différentielle . . . . .	86
7.6.1 la commande BACKUP . . . . .	87
7.7 Mise en oeuvre de la sauvegarde du journal des transactions . . . . .	89
7.7.1 Les caractéristiques . . . . .	89
7.7.2 La commande BACKUP . . . . .	90
7.7.3 Stratégie de sauvegarde sur un serveur de PROD . . . . .	90
7.8 Mise en oeuvre de la sauvegarde de groupes . . . . .	92
7.8.1 Caractéristiques . . . . .	92
7.8.2 La commande BACKUP . . . . .	92
7.9 La sauvegarde partielle . . . . .	95
7.9.1 La commande BACKUP . . . . .	95
7.10 Les snapshots . . . . .	96
7.10.1 C'est quoi un Snapshots ? . . . . .	96
7.10.2 Principe de fonctionnement d'un Snapshot . . . . .	96
7.10.3 Création d'une capture instantanée . . . . .	97
7.10.4 Comportement d'un SELECT sur un Snapshot . . . . .	98
<b>8 La Restauration sous SQL server 2012</b>	<b>100</b>
8.1 Les prérequis à la restauration . . . . .	100
8.1.1 ETAPE 1 : La vérification de la sauvegarde . . . . .	100
8.1.2 ETAPE 2 : Fermeture des sessions en cours . . . . .	102
8.1.3 ETAPE 3 : Restreindre toute nouvelles connexions au Serveur	103
8.1.4 ETAPE 4 : Exemple de restauration d'une base de données	105
8.2 La restauration d'une base de données . . . . .	105
8.2.1 Les différents type de restauration utilisés en PROD . . . . .	105
8.2.2 Les options de la commande RESTORE . . . . .	106
8.3 Exemple de stratégie de sauvegarde et restauration . . . . .	107
8.3.1 Processus de Sauvegarde . . . . .	107
8.3.2 Processus de Restauration . . . . .	107

<b>9 Importation et exportation</b>	<b>109</b>
9.1 Architecture de transfert de données . . . . .	109
9.2 Les outils de transfert de données . . . . .	111
9.2.1 L'outil SQL server Intégration Services . . . . .	111
9.2.2 L'outil BCP(Bulk Copy Program) . . . . .	116
9.2.3 L'outil BULK INSERT . . . . .	119
<b>10 Gestion de la sécurité d'accès</b>	<b>120</b>
10.1 Architecture de sécurité d'accès . . . . .	120
10.1.1 Terminologie . . . . .	120
10.2 L'authentification SQL server . . . . .	122
10.2.1 Création d'une entité via le mode Windows . . . . .	123
10.2.2 Création d'une entité via le mode SQL server . . . . .	125
10.3 Les crédenciales . . . . .	126
10.3.1 Définition : . . . . .	126
10.3.2 Crédentiel et mappage d'un créduential : . . . . .	127
10.4 Les utilisateurs de base de données . . . . .	128
10.4.1 Les caractéristiques : . . . . .	128
10.4.2 L'utilisateur "dbo" . . . . .	129
10.4.3 L'utilisateur "guest" . . . . .	130
10.4.4 Activation du compte "guest" . . . . .	131
10.4.5 Désactivation du compte "guest" . . . . .	131
10.4.6 Création d'un utilisateur de base de données . . . . .	132
10.4.7 Modification d'un utilisateur de base de données . . . . .	133
10.5 Les schémas . . . . .	134
10.5.1 Caractéristiques : . . . . .	134
10.5.2 Accéder aux objets d'un schéma . . . . .	134
10.5.3 Crédentiel et modification et suppression d'un schéma de base de données . . . . .	135
10.6 La gestion des droits sous SQL server . . . . .	136
10.6.1 Les niveaux d'attribution des priviléges . . . . .	136
10.6.2 Les priviléges au niveau Serveur . . . . .	137
10.6.3 Les priviléges d'utilisation d'instruction (liste non exhaustive) . . . . .	138
10.6.4 Les priviléges sur les objets . . . . .	138
10.6.5 Etude de cas . . . . .	138
10.7 La gestion des roles . . . . .	139
10.7.1 Caractéristiques : . . . . .	139
10.7.2 Les roles au niveau serveur . . . . .	140
10.7.3 Les roles au niveau base de données . . . . .	141
10.8 Contexte d'exécution des procédures stockées . . . . .	147
<b>11 Automatisation des tâches récurrentes</b>	<b>149</b>
11.1 L'outil SQL Agent . . . . .	149
11.1.1 Les caractéristiques : . . . . .	149
11.1.2 Les opérateurs . . . . .	151
11.1.3 Crédentiel et création d'un opérateur . . . . .	151

11.1.4 Les Alertes . . . . .	153
11.1.5 Caractéristiques : . . . . .	153
11.1.6 Création d'une Alerté . . . . .	154
11.1.7 Les travaux (les jobs) . . . . .	157
11.1.8 Création d'un travail . . . . .	158
11.2 Configuration du serveur de messagerie . . . . .	161
11.2.1 Caractéristiques : . . . . .	161
11.2.2 L'assistant de configuration de la messagerie de base de données . . . . .	162
11.3 Plan de maintenance . . . . .	166
11.3.1 Les Caratéristiques . . . . .	166
11.3.2 Crédation d'un plan de maintenance . . . . .	167
<b>12 Audit des environnements SQL server</b>	<b>172</b>
12.1 Audit au niveau Serveur . . . . .	172
12.1.1 Crédation de l'Audit . . . . .	174
12.1.2 Spécification de l'audit de serveur . . . . .	174
12.2 Audit au niveau base . . . . .	175
12.2.1 Crédation d'audit au niveau de la base de données . . . . .	176
12.3 L'outil SQL profiler . . . . .	177
12.3.1 Présentation de l'outil SQL server Profiler . . . . .	177
12.3.2 Relecture d'un fichier de trace . . . . .	183
12.4 Les déclencheurs DDL . . . . .	183
12.4.1 Caratéristiques . . . . .	183
12.4.2 Crédation d'un trigger DDL au niveau base . . . . .	184
<b>13 La surveillance sous SQL server 2012</b>	<b>186</b>
13.1 Travailler avec le moniteur de performance . . . . .	186
13.1.1 Caractéristiques . . . . .	186
13.1.2 Les compteurs de performances standard . . . . .	186
13.1.3 Lancement d'une charge de travail au niveau du serveur .	189
13.1.4 Les compteurs de performances personnalisés . . . . .	191
13.1.5 Caractéristiques : . . . . .	191
13.1.6 Crédier un nouveau compteur personnalisé . . . . .	191
<b>14 Dépannage des problèmes d'administration</b>	<b>194</b>
14.1 Corrélation des fichiers de trace . . . . .	194
14.1.1 Objectifs . . . . .	194
14.1.2 Principe de la corrélation . . . . .	195
14.1.3 La corrélation . . . . .	195
14.2 L'outil SQL tunning Advisor DTA . . . . .	199
14.2.1 Définition . . . . .	199
14.2.2 Le principe d'utilisation du DTA . . . . .	199
14.3 Les plans d'exécution des requetes . . . . .	201
14.3.1 C'est quoi un Plan d'exécution ? . . . . .	201
14.3.2 Le cout d'exécution d'une requete . . . . .	204
14.3.3 La mise en cache du plan . . . . .	205

14.3.4 Cas pratique de la mise en cache . . . . .	205
<b>15 La mise en miroir sous SQL server</b>	<b>207</b>
15.1 Configuration de la mise en miroir . . . . .	207
15.1.1 Les caractéristiques : . . . . .	207
15.1.2 Les différents états de la base de données miroir . . . . .	208
15.2 Architecture de la mise en miroir . . . . .	209
15.3 Les prérequis de la mise en place du processus de mise en miroir	210
15.4 Mise en place de la mise en miroir . . . . .	214
15.5 Le test de basculement (FAILOVER) . . . . .	218
<b>Appendix A Les objets de base de données système</b>	<b>221</b>
A.1 Les fonctions . . . . .	221
A.2 Les procédures stockés . . . . .	221
A.3 Les vues . . . . .	222
<b>Appendix B Les commandes SQL usuelles</b>	<b>225</b>
B.0.1 Les commandes de création de base de données . . . . .	225
B.0.2 Les commandes de configuration du Réseau SQL server .	225
B.0.3 Les commandes de sauvegarde et restauration . . . . .	226
B.0.4 Les commandes de transfert de données . . . . .	226
B.0.5 Les commandes de création d'un login de connexion .	226
B.0.6 La commande de création d'un Crédential . . . . .	227
B.0.7 Les commande de création d'un utilisateur . . . . .	227
B.0.8 Les commandes de création d'un schéma . . . . .	227
B.0.9 Les commandes de création d'un role de base de données	228
B.0.10 Les commandes de gestion de privilèges . . . . .	228
B.0.11 La commande de création d'un role d'application . . . . .	228
B.0.12 Les commandes de mise en cache . . . . .	229

# **Abstract**

I've used many methods an approch to resolve an Incidental events among these method.

I've uses the **SQL server logs files** to retreive error code, description.

Using **stack overflow web site** where all developper and administrators from all over the words are meet up including beginners et experienced ones to seek a same problem that have faced

To understand a batch or PowerShell script I've started by **diving the problem into peaces** (divide and concure rule), test each unit undividually and analyse by the end the standard output, if the first and second unit works correctly, combine them to see what will happened and so on..

My significant acheivement during my training was to organise my upcoming week and match my calendar with my to do list, execute each one, by the end of the week I wrote down 3 points : **what you I have acheived successfully, unfinished or bloqued task** that need to be reviewed, **overview of the upcoming tasks**

**VERY IMPORTANT** : Execute each planned task what ever happened

**Keywords:** Stack Overflow, logs file, Technical support, test, organise.

# Acronyms

<b>SGBDR</b>	Système de gestion de base de données
<b>DML</b>	Data Modification Langage
<b>DDL</b>	Data Definition Langage
<b>SQL</b>	Structured Query Langage
<b>ACID</b>	Atomicity Consistency Isolation Durability
<b>ETL</b>	Extraction Transformation Load
<b>OLTP</b>	Online transactional Processing
<b>OLAP</b>	Online Analytical Processing
<b>T-SQL</b>	Transact Structured Query Langage
<b>BCP</b>	Bulk Copy Programme
<b>DBCC</b>	Database Consistency Checker

# List of Figures

1.1	l'architecture OLAP Client/Serveur . . . . .	2
1.2	L'architecture trois tiers . . . . .	2
1.3	L'architecture OLAP . . . . .	3
1.4	Les principales composants de SQL server . . . . .	5
1.5	L'instance par défaut . . . . .	6
1.6	Architecture d'une instance SQL server . . . . .	7
1.7	Architecture d'une base de données . . . . .	9
2.1	Validation de la présence du composant .NET Framework . . . . .	11
2.2	Outil d'analyse de configuration système . . . . .	12
2.3	Résultat de l'analyse SCC . . . . .	12
2.4	Etape 1 : Lancement de l'installation d'une nouvelle instance SQL server . . . . .	13
2.5	Etape 2 : Démarrage de l'analyseur SCC . . . . .	13
2.6	Etape 3 : suspendre la recherche de mise à jour . . . . .	14
2.7	Etape 4 : Vérification des règles d'installation . . . . .	14
2.8	Etape 5 : spécifier la clé d'activation du produit . . . . .	15
2.9	Etape 6 : Cocher l'option d'installation de fonctionnalités SQL server . . . . .	15
2.10	Selection des fonctionnalités . . . . .	16
2.11	Etape 8 : Vérification des règles par SCC . . . . .	17
2.12	Etape 9 : Configuration de l'instance . . . . .	17
2.13	Etape 9 : Erreur d'installation de plusieurs instance par défaut . . . . .	18
2.14	Etape 10 : Espace disque nécessaire . . . . .	18
2.15	Etape 11 : Configuration du compte service . . . . .	19
2.16	Etape 11 : Configuration du compte service . . . . .	19
2.17	Etape 12 : Configuration du moteur de base de données . . . . .	20
2.18	Etape 12 : Configuration du moteur de base de données . . . . .	21
2.19	Etape 13 : Récapitulative de l'installation . . . . .	21
4.1	Etape 1 : Référencier les instances . . . . .	28
4.2	Etape 2 : Référencier les instances . . . . .	28
4.3	Etape 3 : Référencier les instances . . . . .	29
4.4	La boite de dialogue de connexion . . . . .	29
4.5	Présentation de l'interface utilisateur SSMS . . . . .	30
4.6	Le répertoire sécurité . . . . .	31
4.7	Le répertoire Objets serveur . . . . .	32

4.8 Le répertoire réPLICATION . . . . .	32
4.9 Le répertoire gestion . . . . .	33
4.10 afficher le manuelle d'utilisation . . . . .	34
5.1 Le gestionnaire de configuration SQL server . . . . .	36
5.2 Afficher les propriétés du compte service . . . . .	37
5.3 reconfigurer le port d'écoute de l'instance . . . . .	38
5.4 Gestionnaire de Service Windows . . . . .	39
5.5 Inscription des serveurs de base de données . . . . .	40
5.6 Changement du mot de passe du compte (sa) . . . . .	41
6.1 les fichiers de données et fichiers journaux . . . . .	43
6.2 Présentation de la propriété de la base de données . . . . .	44
6.3 La structure du fichier data . . . . .	45
6.4 Le principe de journalisation sous SQL server 2012 . . . . .	47
6.5 Création d'une base de donnée en mode T-SQL . . . . .	48
6.6 Création d'une base de données avec SSMS . . . . .	49
6.7 Modification d'un fichier de données . . . . .	51
6.8 Ajouter un fichier data . . . . .	52
6.9 supprimer un fichier data . . . . .	52
6.10 Cas d'utilisation de la commande SHRINKDATABASE . . . . .	53
6.11 affichage du rapport d'utilisation du disque . . . . .	54
6.12 Le groupe de fichier PRIMARY . . . . .	55
6.13 Créer une fonction de partition en mode T-SQL . . . . .	58
6.14 Créer un schéma de partitionnement en mode T-SQL . . . . .	59
6.15 Utilisation du schéma de partitionnement sur table . . . . .	61
6.16 l'assistant de gestion de partitionnement . . . . .	61
6.17 Création d'un Index Partitionné . . . . .	62
6.18 Utilisation de la fonction DATABASEPROPERTYEX . . . . .	65
6.19 Utilisation de la vue sys.database . . . . .	65
6.20 Transaction explicite . . . . .	66
6.21 Cas pratique d'utilisation de la procédure stockée sp_locks . . . . .	69
6.22 Transaction explicite d'insertion de données . . . . .	69
6.23 Transaction explicite avec insertion validé . . . . .	70
6.24 Transaction explicite avec des points d'arrêt . . . . .	70
6.25 Transaction avec des points d'arrêt . . . . .	71
6.26 Etape 1 : Lancer l'assistant de gestion de compression . . . . .	75
6.27 Etape 2 : Choisir le type de compression . . . . .	76
7.1 Exemple de combinaison d'une stratégie de sauvegarde FULL/DIFF	79
7.2 Combinaison des trois stratégies de sauvegarde sur SQL server .	81
7.3 Connaitre les roles pour les opérations BACKUP . . . . .	82
7.4 La syntaxe complète de sauvegarde sur SQL server . . . . .	83
7.5 Etape 1 : Création d'une unité de sauvegarde logique . . . . .	84
7.6 Etape 2 : Exécution de la sauvegarde complète en mode T-SQL	84
7.7 Etape 3 : vérification du contenu du support de sauvegarde . . .	85
7.8 La sauvegarde avec une mise en miroir . . . . .	86

7.9 La sauvegarde avec l'option de compression . . . . .	86
7.10 La syntaxe de la sauvegarde différentielle . . . . .	87
7.11 Sauvegarde différentielle sur un support de sauvegarde logique . . . . .	88
7.12 La syntaxe de la sauvegarde des fichiers journaux . . . . .	90
7.13 Exemple d'une stratégie de sauvegarde sous SQL server . . . . .	91
7.14 La syntaxe de sauvegarde de groupe de fichiers . . . . .	92
7.15 Etape 1 : stratégie de sauvegarde de groupe de fichiers . . . . .	93
7.16 Etape 2 : stratégie de sauvegarde de groupe de fichiers . . . . .	94
7.17 Etape 3 : stratégie de sauvegarde de groupe de fichier . . . . .	94
7.18 La syntaxe de la sauvegarde partielle . . . . .	95
7.19 Principe de fonctionnement d'un snapshot sous SQL server . . . . .	96
7.20 Création d'un snapshot sous SQL server . . . . .	97
7.21 Comportement d'un SELECT sur un Snapshot . . . . .	98
 8.1 La commande RESTORE HEADERONLY . . . . .	101
8.2 La commande RESTORE FILELISTONLY . . . . .	102
8.3 La commande RESTORE VERIFYONLY . . . . .	102
8.4 Fermeture des sessions en cours . . . . .	103
8.5 suspendre l'accès à l'instance . . . . .	103
8.6 Restreindre à l'accès à la base . . . . .	104
8.7 Processus de restauration . . . . .	105
8.8 Simulation d'un processus de sauvegarde . . . . .	107
8.9 Simulation d'un processus de sauvegarde (suite) . . . . .	107
8.10 Simulation d'un processus de restauration . . . . .	108
 9.1 architecture de transfert de données . . . . .	110
9.2 Etape 1 : Lancement de l'assistant SSIS . . . . .	112
9.3 Etape 2 : Définir la Source de données . . . . .	113
9.4 Etape 3 : Définir la Destination (The target) . . . . .	113
9.5 Etape 4 : Définir la copie des données . . . . .	114
9.6 Etape 5 : Configuration du fichier de destination . . . . .	114
9.7 Etape 6 : Création du package . . . . .	115
9.8 Utilisation de l'outil BCP . . . . .	117
9.9 Vérification des données exportées . . . . .	118
9.10 Exportation des données d'une requête . . . . .	118
9.11 Utilisation de la commande BULK INSERT . . . . .	119
 10.1 Architecture de sécurité d'accès sous SQL server . . . . .	121
10.2 Création d'une nouvelle connexion . . . . .	123
10.3 Etape 1 : Ajouter un compte d'administration Windows . . . . .	124
10.4 Etape 2 : Création d'un Login de connexion . . . . .	124
10.5 Etape 3 : Création d'un login de connexion . . . . .	125
10.6 Etape 1 : Création d'un compte SQL server . . . . .	125
10.7 Etape 2 : Création d'un compte SQL server . . . . .	126
10.8 Création d'un créduel . . . . .	127
10.9 Le mappage d'un Créduel . . . . .	127
10.10 Afficher la liste des utilisateurs de base de données . . . . .	129

10.11Mappage du compte "sa" à l'utilisateur"dbo" . . . . .	129
10.12Activer le compte "guest" . . . . .	131
10.13Désactiver le compte "guest" . . . . .	132
10.14Création d'un utilisateur de base de données . . . . .	132
10.15modifier un utilisateur de base de données . . . . .	133
10.16Accéder aux objets d'un schéma . . . . .	134
10.17Création, Modification et suppression d'un schéma . . . . .	135
10.18Les privilèges attribués au niveau serveur . . . . .	137
10.19Attribution des droits à l'utilisateur "belkacem_02" . . . . .	139
10.20Consulter les membres d'un role de base de données . . . . .	142
10.21La liste des roles fixes de base de données . . . . .	143
10.22Création d'un role utilisateur de base de données . . . . .	144
10.23Etape 1 : Création d'un role d'application . . . . .	146
10.24Etape 2 : Attribution des privilèges au role d'application . . . . .	146
10.25Créer une procédure en spécifiant le contexte d'exécution . . . . .	147
 11.1 Présentation de SQL Agent . . . . .	150
11.2 Le fichier journal des erreurs de SQL Agent . . . . .	151
11.3 Crédit d'un opérateur DBA . . . . .	152
11.4 choisir le type de notification envoyée à l'utilisateur . . . . .	152
11.5 Afficher la liste des messages d'erreurs . . . . .	153
11.6 Etape 1 : Création d'une Alerte SQL . . . . .	155
11.7 Etape 2 : Création d'une Alerte SQL . . . . .	156
11.8 Structure globale d'une alerte de type évènement SQL . . . . .	156
11.9 Structure globale d'une alerte sur une condition de performance . . . . .	157
11.10Etape 1 : création d'un travail . . . . .	158
11.11Etape 2 : création d'un travail . . . . .	159
11.12Etape 3 : création d'un travail . . . . .	159
11.13Etape 4 : création d'un travail . . . . .	160
11.14Etape 5 : création d'un travail . . . . .	160
11.15Etape 1 : Lancer l'assistant de configuration de la messagerie . . . . .	162
11.16Etape 2 : Créer un nouveau profil de messagerie . . . . .	162
11.17Etape 3 : Créer un nouveau profil de messagerie . . . . .	163
11.18Etape 4 : spécifier les attributs du compte SMTP . . . . .	163
11.19Etape 5 : Gérer la sécurité du profile . . . . .	164
11.20Etape 6 : Configuration des paramètres systèmes . . . . .	164
11.21Etape 7 : Tester le bon fonctionnement du serveur SMTP . . . . .	165
11.22Etape 1 : Définir un plan de maintenance . . . . .	167
11.23Etape 2 : Sélectionner les propriétés du plan . . . . .	168
11.24Etape 3 : Sélectionner les propriétés du plan . . . . .	168
11.25Etape 4 : Configuration de la tâche de maintenance . . . . .	169
11.26Etape 5 : Planification d'une tâche d'un plan de maintenance . . . . .	169
11.27Etape 6 : Enregistrement du fichier journal du plan de maintenance . . . . .	170
11.28Etape 7 : Affichage de l'historique du plan de maintenance . . . . .	170
11.29Etape 7 : Le fichier journal du plan de maintenance . . . . .	171

12.1 Emplacement de l'audit dans l'explorateur d'objet . . . . .	173
12.2 Création de l'audit au niveau serveur . . . . .	174
12.3 Créer une spécification de l'audit . . . . .	175
12.4 consulter le journal de l'audit . . . . .	175
12.5 Ajouter une spécification de l'audit au niveau base de données .	176
12.6 Ouverture d'une session de traçage . . . . .	178
12.7 Introduire les caractéristiques du fichier de trace . . . . .	179
12.8 Sélectionner la classe et les attributs d'évènements . . . . .	180
12.9 Personnaliser l'affichage des colonnes d'évènements . . . . .	181
12.10Le résultat de traçage des évènements sélectionnés . . . . .	181
12.11Création d'un trigger DDL_EVENT . . . . .	184
 13.1 Ajouter les compteurs standards de performances . . . . .	187
13.2 Les paramètres des compteurs de performances . . . . .	188
13.3 Le compteur Data File Size . . . . .	189
13.4 Exécution d'une charge de travail intense . . . . .	190
13.5 Etape 1 : choisir un compteur personnalisé dans l'objet UserSettable . . . . .	192
13.6 Etape 2 : Corréler le compteur avec l'évènement . . . . .	192
 14.1 Le principe de la corrélation . . . . .	195
14.2 Etape 1 : Ajouter les compteurs de performances . . . . .	196
14.3 Etape 2 : Enregistrer le fichier de trace du moniteur . . . . .	196
14.4 Etape 3 : Personnaliser les classes d'évènements sur SQL profiler	197
14.5 Etape 4 : Stopper la collecte de données . . . . .	197
14.6 Etape 5 : Importer les données de performances . . . . .	198
14.7 Etape 6 : Analyser le fichier de trace . . . . .	198
14.8 Etape 1 : Construire un fichier de trace . . . . .	200
14.9 Etape 2 : Configurer le DTA . . . . .	200
14.10Etape 3 : Le résultat de l'analyse DTA . . . . .	201
14.11Afficher le Plan d'exécution estimé en format XML . . . . .	203
14.12afficher le Plan d'exécution réel . . . . .	204
14.13Le plan d'exécution estimé . . . . .	205
14.14Exemple de mise en cache des requêtes couteuses . . . . .	206
 15.1 Architecture de la mise en miroir . . . . .	209
15.2 Etape 1 : Incrire vos différents serveurs . . . . .	210
15.3 Etape 2 : Sauvegarder la base de données principale . . . . .	211
15.4 Etape 3 : Restauration du jeu de sauvegarde sur le serveur miroir	211
15.5 Etape 4 : spécifier l'option NORECOVERY . . . . .	212
15.6 Etape 5 : sauvegarde du fichier journal de transaction . . . . .	212
15.7 Etape 6 : Restaurer le fichier journal des transactions . . . . .	213
15.8 Etape 7 : Activer l'option NORECOVERY . . . . .	213
15.9 Etape 1 : Démarrer la configuration de la mise en miroir . . . .	214
15.10Etape 2 : Démarrer la configuration de la mise en miroir . . . .	214
15.11Etape 3 : Inclure le serveur témoin . . . . .	215
15.12Etape 4 : configurer le serveur maître . . . . .	215

15.13Etape 5 : Configurer le serveur secondaire . . . . .	216
15.14Etape 6 : Configurer les comptes services des différents instances	216
15.15Etape 7 : Configuration terminée de la mise en miroir . . . . .	217
15.16Etape 8 : Afficher le moniteur de mise en miroir . . . . .	218
15.17Insérer un enregistrement dans une table . . . . .	219
15.18Arreter l'instance du serveur principal . . . . .	219
15.19Arreter l'instance du serveur principal . . . . .	220

# Chapter 1

## Présentation de SQL server 2012

### 1.1 Comprendre le fonctionnement de SQL server

Microsoft SQL sever ou couramment appelé "MSSQL" est **un système de gestion de base de données relationnelle** développé par Microsoft pour stocker et gérer les données volumétrique d'une entreprise de manière **cohérente et structuré**.

On peut le déployer sur **deux environnements distincts OLTP ou OLAP** :

1. SQL serveur peut fonctionner dans **un environnement OLTP (Online transaction processing)** dans lequel un certain nombre de transaction de type LMD (INSERT, UPDATE, DELETE) peuvent être exécuté, sa particularité est qu'il **consomme peu de ressource** compte tenu de la taille des données à traitées.
2. Il est également possible de le faire fonctionner en tant que serveur décisionnel ou datawarehouse (**OLAP : Online Analyses Processing**) pour une **analyse multidimensionnelle de grands volume de données provenant de plusieurs sources** à grande vitesse. **La charge est très importante, de grandes requêtes vont solliciter le serveur**

### 1.1.1 L'architecture OLTP (Client/Serveur)

On peut avoir **une architecture Client/serveur** dans laquelle un client lourd va attaquer un serveur de base de données, comme le montre la figure ci-dessous :

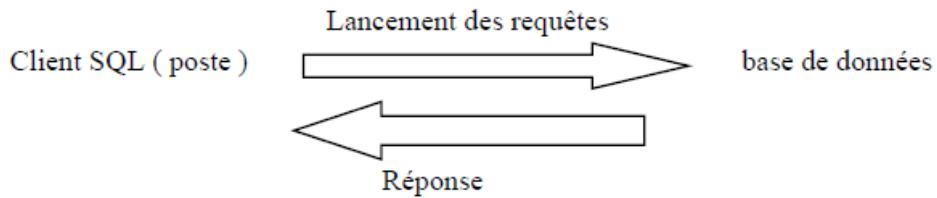


Figure 1.1: l'architecture OLAP Client/Serveur

### 1.1.2 L'architecture OLTP 3 tiers

C'est une architecture dans laquelle **le client va interroger avec des requêtes légères en premier lieu le serveur d'application**, puis ce dernier va interroger la base de données pour extraire les informations.

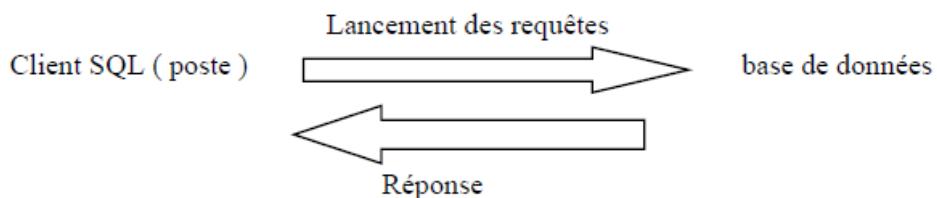
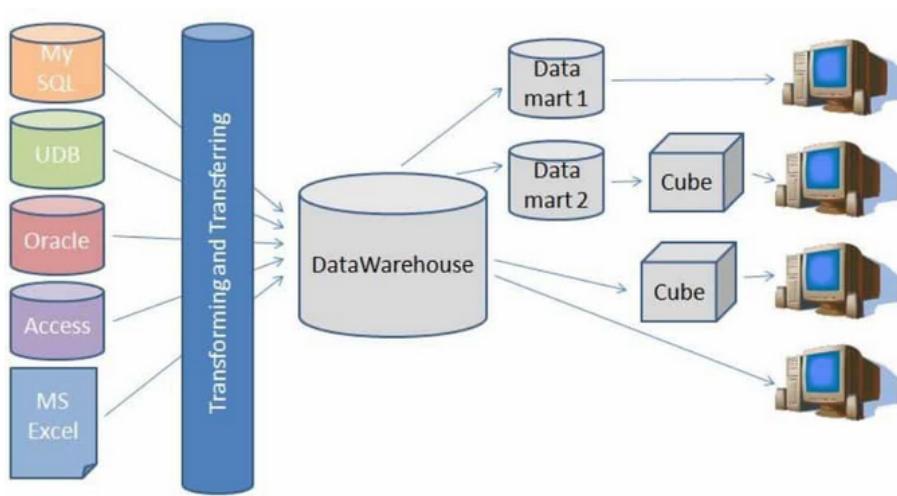


Figure 1.2: L'architecture trois tiers

---

<sup>1</sup>Le Serveur d'application : héberge et exécute des applications métier de l'entreprise (EXP : CRM, HRIS)

### 1.1.3 l'architecture OLAP



**Figure 1.3: L'architecture OLAP**

**Le principe de l'analyse multidimensionnelle** est illustré dans la **Figure 1.3**, vous avez d'une part les différents **sources de données** à partir desquelles **des outils ETL** (SSIS, DATASTAGE) vont extraire les données leur faire subir une transformation avant de les charger dans un entrepot de données (**Datawarehouse**) sous forme de cube OLAP.

Il peut etre avantageux de transférer ces données vers une base de données cible appelé **"Datamart"** exclusivement **conçue pour l'analyse, la prise de décision et génération de rapport personnalisé**.

## 1.2 Présentation des différents composants de SQL server

Avant d'installer SQL server, il est plus qu'important de **connaitre ces différents composants** :

1. **Le moteur de base de données** :

- (a) Il s'agit du composant responsable de la **gestion des données**, de leur **stockage**, de leur **manipulation** et de leur **récupération**.
- (b) Il s'exécute en tant Service Windows
- (c) Référencé en tant que MSSQLSERVER pour l'instance par défaut

**2. SQL Agent :**

- (a) C'est le composant qui **gère l'exécution de tâches planifiées**, la surveillance de SQL server et le suivi des alertes (un composant très important dans le cadre automatisation des tâches récurrentes)
- (b) Directement lié à une instance SQL server
- (c) Référencé dans le gestionnaire de service sous le nom **Agent SQL server (MSSQLSERVER)**

**3. Serveur Integration Services (SSIS) :**

- (a) Outil **d'importation et d'exportation** de données
- (b) **Transfert et transformation** des données
- (c) Intègre des assistants pour créer un **ETL**

**4. SQL Serveur Analyses Services (SSAS)**

- (a) Outil **d'analyse OLAP et Data mining** de Microsoft
- (b) Permet de **construire des cubes OLAP**
- (c) Idéal pour des **projets décisionnels**

**5. SQL serveur Reporting Services (SSRS)**

- (a) C'est un outil qui permet de **faire des extractions et d'afficher des rapports** sous format de tableau, de graphisme, afficher sous format lisible aux utilisateurs finaux.

**6. La réPLICATION SQL server**

- (a) C'est une fonctionnalité puissante qui permet de **copier et distibuer les données** et les objets de base de données **entre plusieurs serveurs**

## 7. Service Broker

- (a) Permet un **travail en mode asynchrone**, faciliter la gestion des pics de forte activité **en stockant les demandes de travail avant de les traiter**, il est très pratique dans le cas d'un server sous-dimensionner (crée une file d'attente et les traiter au fur et à mesure)

## 8. CLR :

- (a) L'intégration de cette fonctionnalité permet de développer des procédures et fonction en utilisant langages de développement d'application.

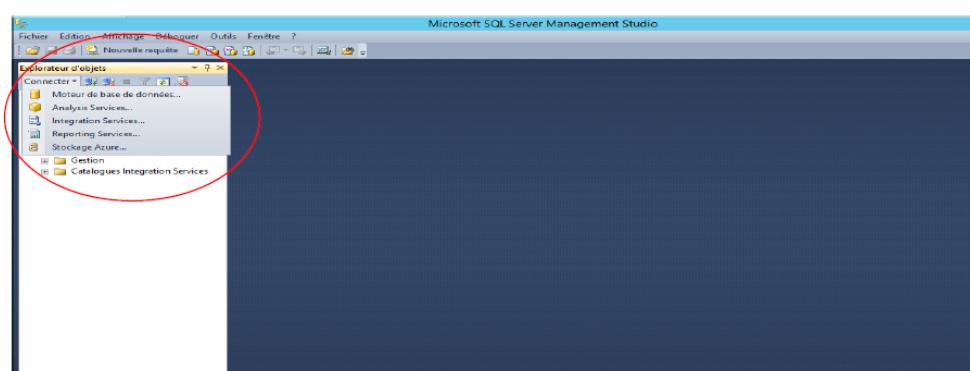


Figure 1.4: Les principales composants de SQL server

Pour visualiser ou se connecter à l'un des composants principaux de SQL server, ouvrez une nouvelle session SQL server management studio, une fenêtre de dialogue de connexion s'affichera pour vous demander se connecter à l'un des composants qui s'exécutent en tant que service (moteur, SSIS, SSAS, SSRS).

## 1.3 l'architecture de SQL server 2012

### 1.3.1 Qu'est ce qu'une instance

C'est une **zone mémoire**, c'est **des processus**, une **arborescence physique** au niveau des répertoires installé sur un server physique ou un server virtuel.

Lors du processus d'installation de SQL server, vous aurez au choix **deux type d'instances** à installer :

1. **Une Instance par défaut** : Il est **identifié** par **le nom réseau de l'ordinateur** sur lequel elle s'exécute.

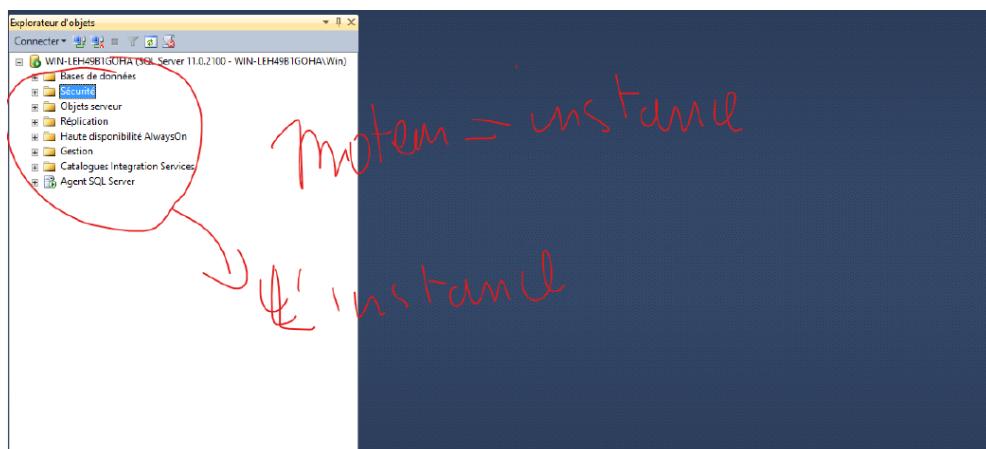


Figure 1.5: L'instance par défaut

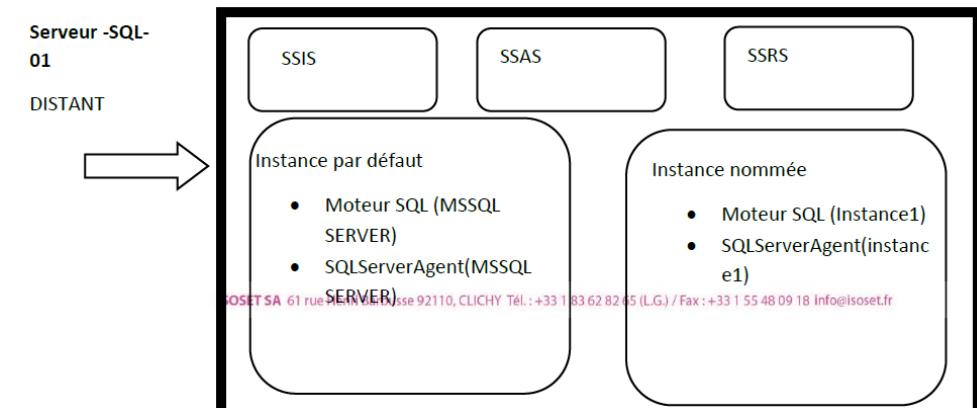
2. **Une instance nommée** : elle est **identifiée** par **le nom du réseau de l'ordinateur** sur lequel elle s'exécute, **suivie d'un nom attribué à l'instance**.



#### NOTE IMPORTANTE

Le choix du nom de l'instance est important lors de son installation

### 1.3.2 Architecture d'une instance SQL server



**Figure 1.6: Architecture d'une instance SQL server**

Voilà une représentation schématique de ce qui est une instance, pour pouvoir parler d'une instance **il faut obligatoirement un moteur de base de données et un SQL Agent**, elle peut comporter plusieurs base de données et partager divers fonctionnalités avec d'autre instance pour c'est le cas de SSIS

### 1.3.3 Les outils de SQL server 2012

Pour pouvoir travailler sur une instance, il faut avoir des outils, les outils principaux sur SQL SERVER sont :

1. **SSMS** : permet de réaliser toutes les opérations sur un server de base de données
2. **Le gestionnaire de configuration SQL server** : pour **gérer les services liés à SQL Server** (on l'utilise à la fin du processus d'installation pour vérifier que tous les services installée sont opérationnel)
3. **SQL server Profiler** : Pour suivre et analyser la charge de travail sur une

instance (celui-là on l'utilise pour identifier les opérations effectuer sur la base qui provoque des tremblements)

4. **Assistant de paramétrage du moteur de base de données** : qui permet l'optimisation globale du fonctionnement du serveur de base de données
5. : **SQLcmd** : c'est un outil de ligne de commande, permet d'exécuter des requêtes, exécuter des scripts de commande, **établir une connexion d'administration dédié (DAC)** (en cas de problème sur la base master, il faut se connecter en mode DAC pour pouvoir restaurer cette base de données)



#### NOTE IMPORTANTE

Il existe deux manières de se connecter au moteur de la base de données SQL server, **privilégier l'utilisation de l'outil SSMS pour la partie administration et sqlcmd pour toute l'opération délicate de maintenance**

## 1.4 Comprendre l'architecture d'une base de données

### 1.4.1 Le role d'une base de données

- Stocke des objets logiques de manière cohérente et structuré tel que **les données** : tables, indexe, contraintes d'intégrités, type de données, valeurs par défaut, règles..
- Stocke ce qui permet **d'accéder aux données** : vues et procédures stockés..
- Stocke **La gestion de l'intégrité complexe** : déclencheur

### 1.4.2 Architecture d'une base de données

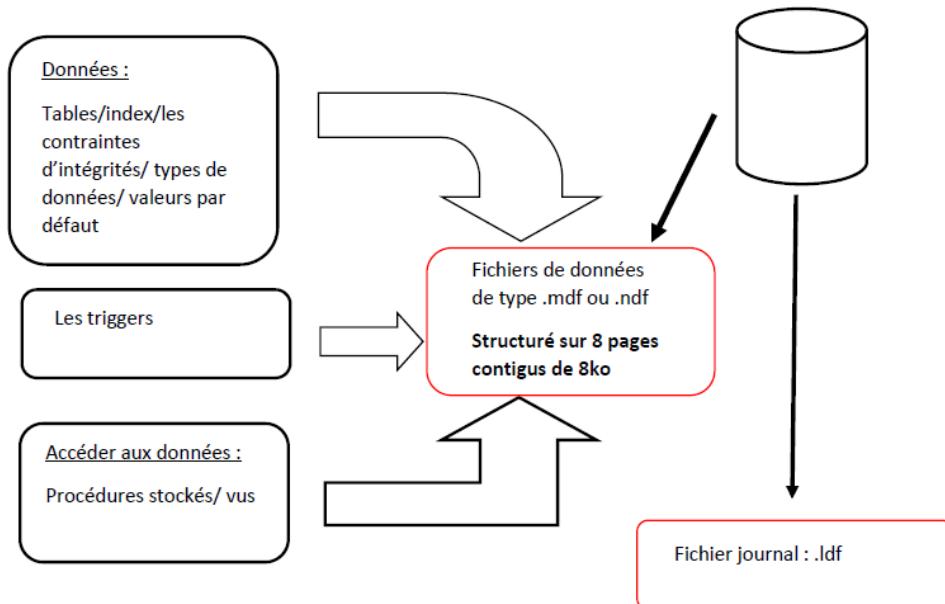


Figure 1.7: Architecture d'une base de données

Les points ci-dessous décrivent l'architecture des bases de données système et applicatives de SQL server

- Une base de données appartient obligatoirement à une instance SQL server
- Une base de données est composée physiquement d'au

### 1.4.3 Les bases de données systèmes

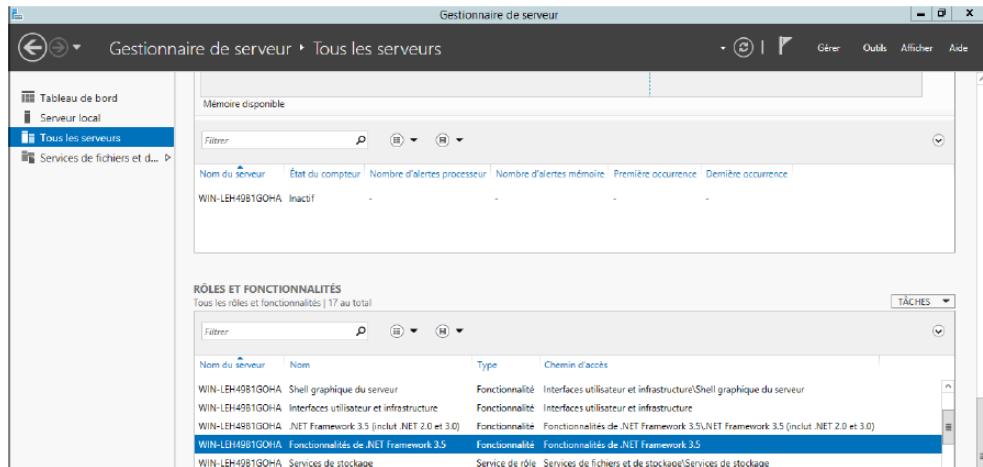
# Chapter 2

## Installation manuelle de SQL server 2012

### 2.1 Les prérequis du processus d'installation de SQL server 2012

Avant d'entamer le processus d'installation d'une instance SQL server, vous etre obliger de vérifier un certain nombre de prérequis :

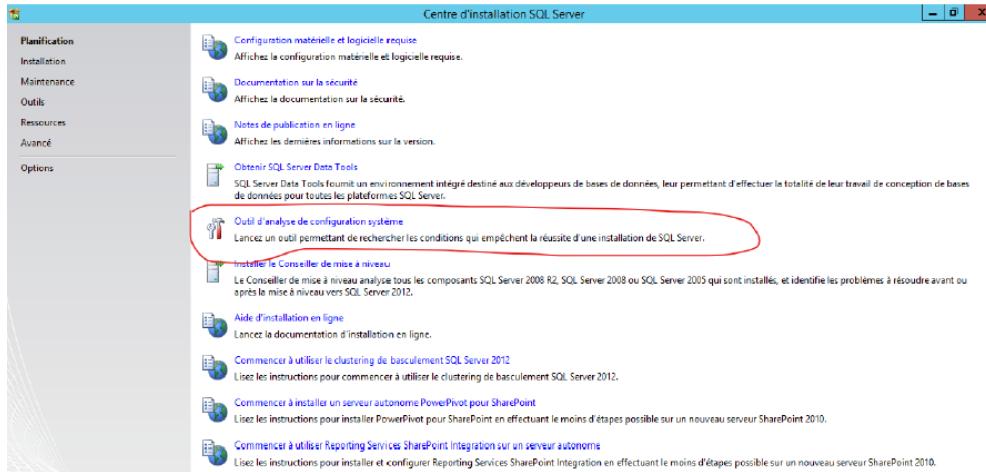
1. Assurez vous que **le composant .NET Framework 3.5.1 + SP** est **installé et mis à jour** en accédant au **gestionnaire de serveur** par le chemin suivant : **Gestionnaire de serveur \tous les servers \role et fonctionnalités**



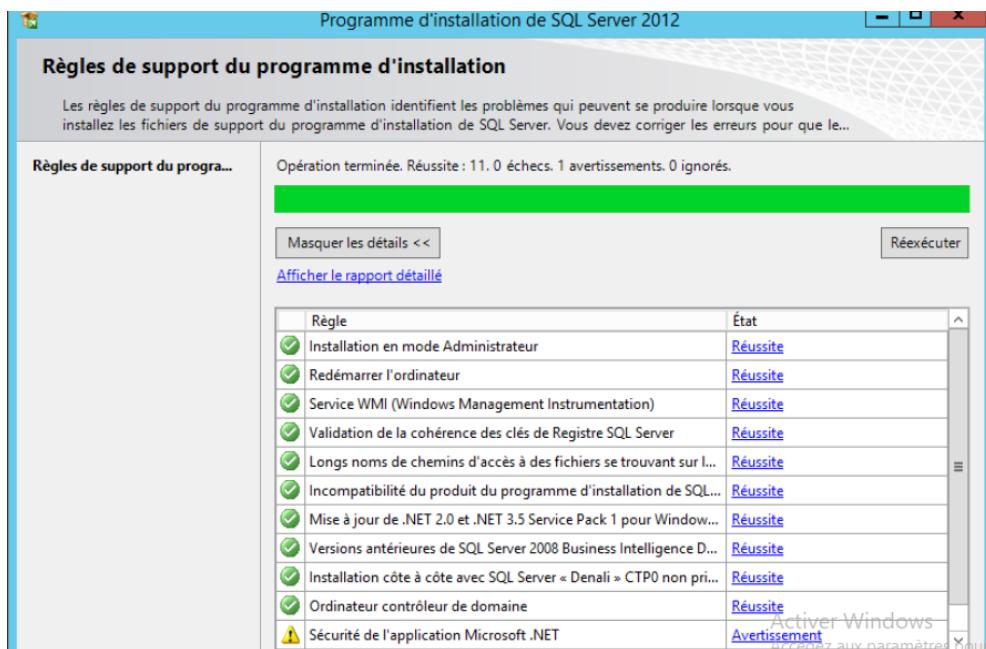
**Figure 2.1: Validation de la présence du composant .NET Framework**

Si par malheur le composant n'est pas installé sur votre système, je vous invite à l'installer en ajoutant la fonctionnalité .NET dans la liste des taches de Windows Server.

2. **Lancer l'outil d'analyse de configuration système (SCC)** : cette outil va vous permettre de checker le système avant de lancer l'installation. Le SCC va contrôler ou checker :
  - (a) **La configuration logicielle requise** : il vérifie par exemple si le système d'exploitation est compatible avec l'édition de SQL server 2012.
  - (b) **La configuration matérielle requise** : il vérifie si le serveur est conforme à la configuration minimale requise en matière de processeur, espace disque et de mémoire
  - (c) **Vérification des conditions de sécurité** : il vérifie si l'utilisateur qui tente d'installer SQL server 2012 possède les droits requis, il vérifie également si l'utilisateur possède les droit de Read/Write sur les répertoires de SQL server 2012
  - (d) **Vérification de l'état du système** : vérifie qu'aucun fichier n'est bloqué ou en attente.



**Figure 2.2: Outil d'analyse de configuration système**

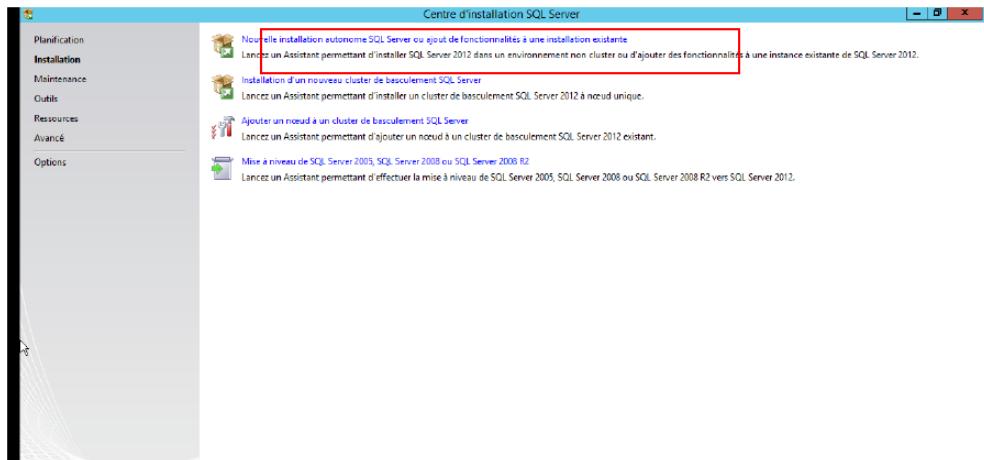


**Figure 2.3: Résultat de l'analyse SCC**

Le SCC a procédé à la vérification d'un ensemble de règle définis dans la **Figure 2.3**. Si les états de ces règles ne sont pas bloquant, l'installation de SQL server peut se faire sans aucun problème.

## 2.2 Les étapes du processus d'installation manuelle de SQL server 2012

Les étapes à suivre pour réaliser une installation manuelle d'une instance SQL server sont résumé dans la suite de figures ci-dessous :



**Figure 2.4:** Etape 1 : Lancement de l'installation d'une nouvelle instance SQL server



**Figure 2.5:** Etape 2 : Démarrage de l'analyseur SCC

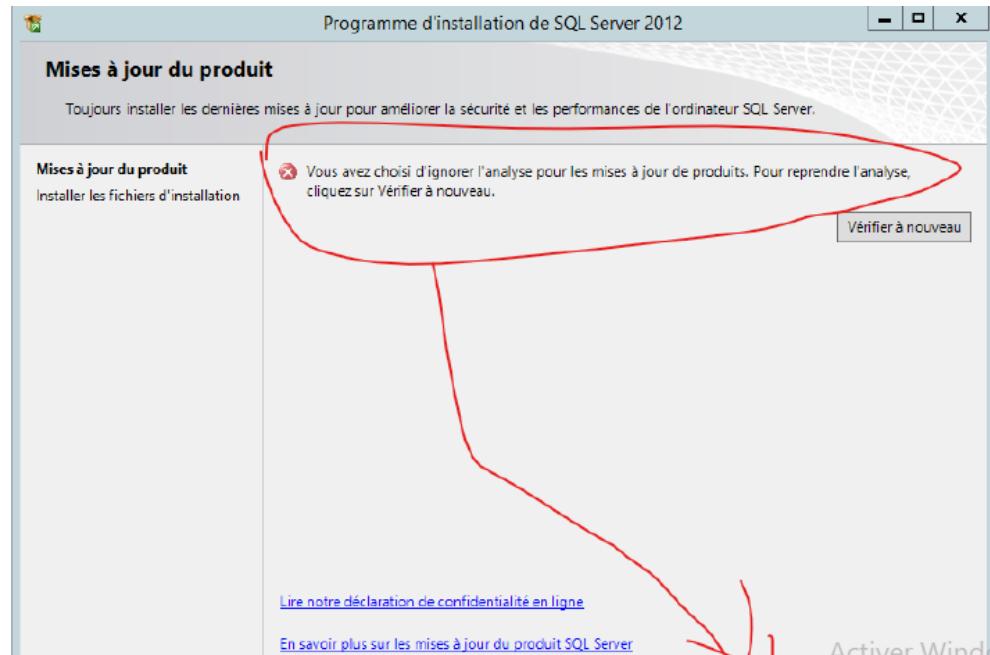


Figure 2.6: Etape 3 : suspendre la recherche de mise à jour

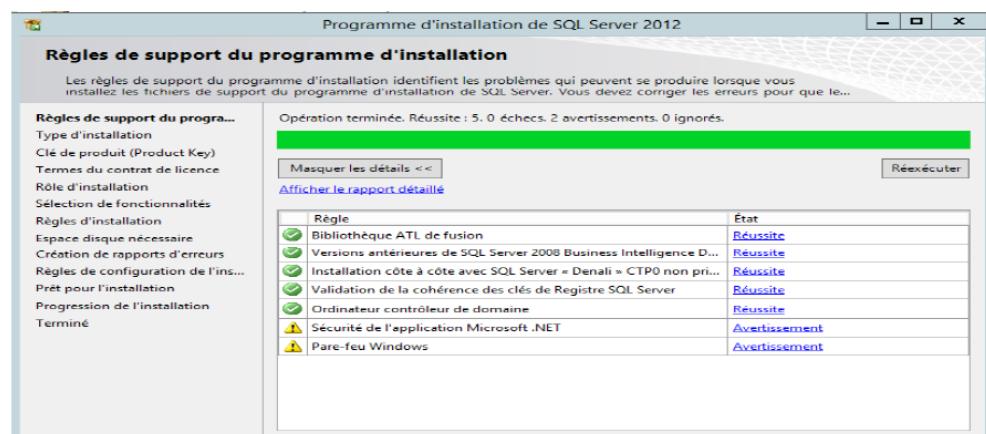


Figure 2.7: Etape 4 : Vérification des règles d'installation



**Figure 2.8: Etape 5 : spécifier la clé d'activation du produit**

Si vous êtes en entreprise, pour bénéficier de la version complète de SQL server, vous devez **introduire la clé d'activation** du produit fournie par Microsoft



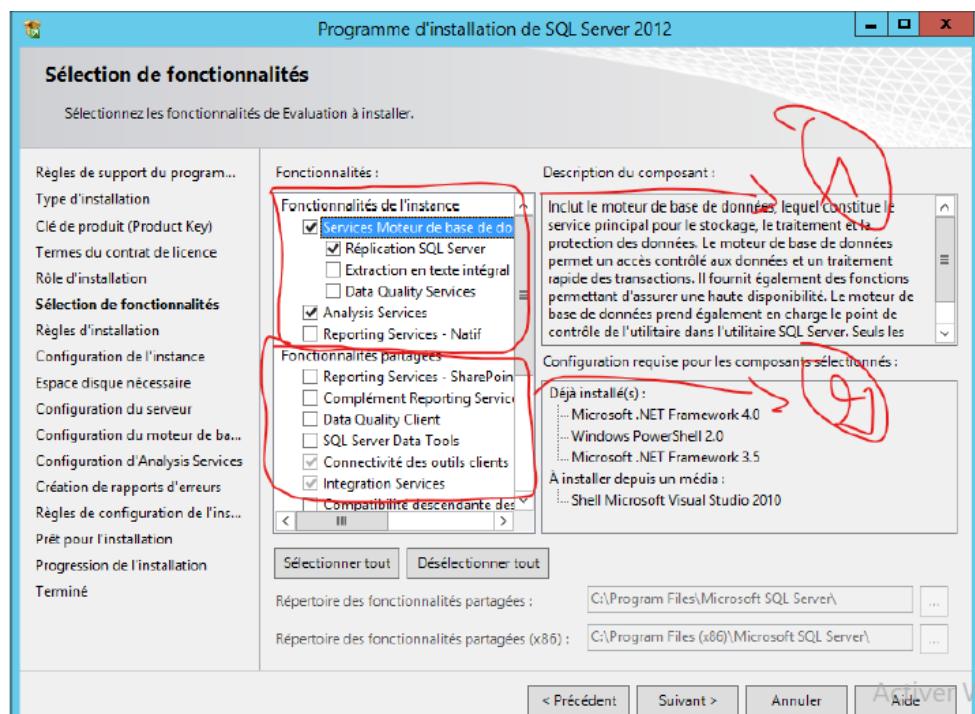
**Figure 2.9: Etape 6 : Cocher l'option d'installation de fonctionnalités SQL server**

Dans Cette étape importante du processus d'installation de sql server 2012, le programme d'installation nous propose trois options :

1. **Installer les fonctionnalités de SQL server 2012 individuellement** : nous avons la possibilité d'installer uniquement, le moteur de la base, le SSIS,

le SSAS, le SSRS, **cette option est nécessaire pour contrôler les différents composants à installer**

2. **Installer toutes les fonctionnalités par défaut** : elle est **déconseillée**, elle **consomme pas mal d'espace disque**, exécuter un certain nombre de services qui ne vont pas être utile comparant à nous besoin future. (**Faire un choix stratégique en termes de configuration**)



**Figure 2.10: Selection des fonctionnalités**

La selection des fonctionnalités sont divisés en deux catégories :

1. **Les fonctionnalités de l'instance** : sont des fonctionnalités qui sont lier directement à une seule et unique instance. Exemple : moteur de base de données, réPLICATION SQL server.
2. **Les fonctionnalités partagées** : ce sont des fonctionnalités qui peuvent être partager entre différentes instances de sql server comme : intégration

services, SSMS...

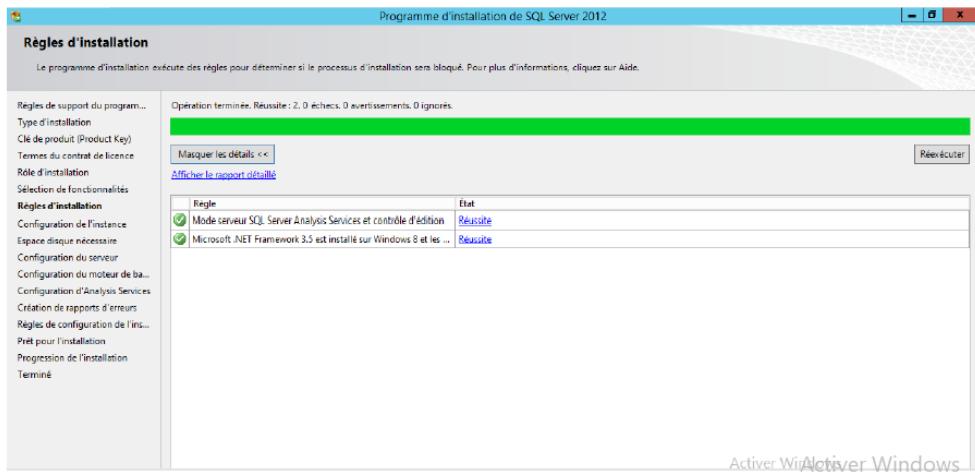


Figure 2.11: Etape 8 : Vérification des règles par SCC

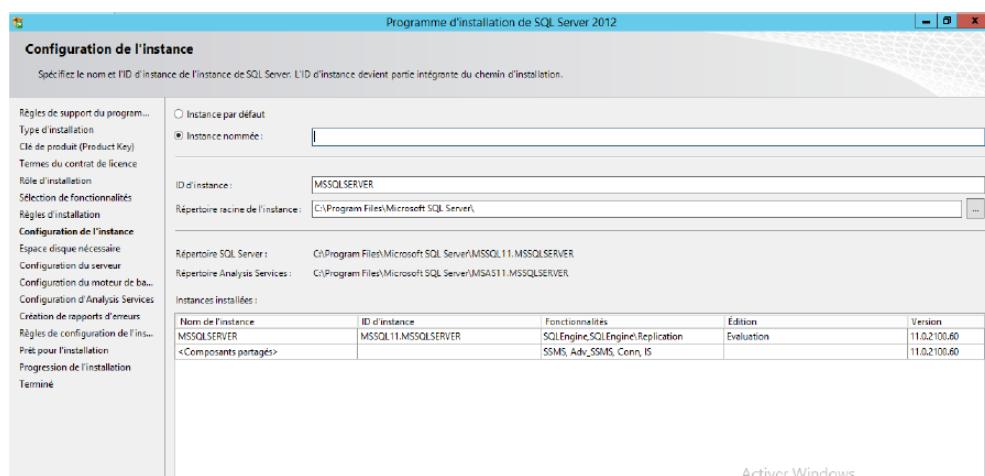
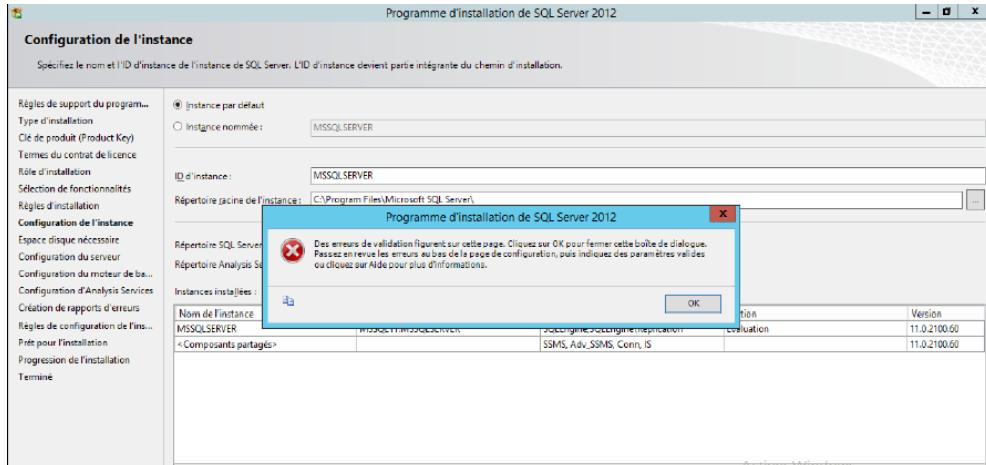


Figure 2.12: Etape 9 : Configuration de l'instance

Une étape TRES IMPORTANTE, c'est là que **le mode d'installation de l'instance sera choisi**, soit **une instance par défaut**, soit **une instance nommée**. En bas de la page vous pouvez identifier les différentes instances qui sont déjà installé sur la machine locale, dans notre cas une instance par défaut est déjà existante.

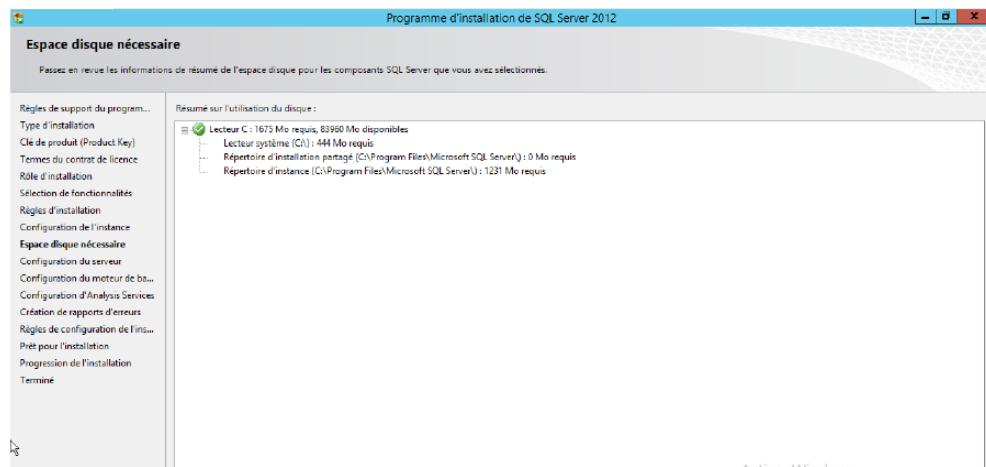


**Figure 2.13: Etape 9 : Erreur d'installation de plusieurs instance par défaut**



### NOTE IMPORTANTE

A la fin du processus d'installation d'une instance SQL server, il va falloir **confirmer et valider** votre installation soit en accédant à **l'outil Service** mis à disposition par Windows soit **d'ouvrir le gestionnaire de configuration SQL server** et voir si le **moteur de base de données et son SQL agent** sont présent dans la liste des services.



**Figure 2.14: Etape 10 : Espace disque nécessaire**

Dans cette étape nous obtenons un petit **récapitulative sur l'espace disque** nécessaire pour l'installation de l'instance nommée. **Chaque instance occupera son propre**

**volume disque** Pour pouvoir réaliser l'installation de l'instance nommé, il faut pouvoir disposer de :

1. D'un lecteur système ( 444Mo /83960 Mo)
2. D'un répertoire d'installation partagé de 0Mo
3. D'un répertoire d'instance 1231 Mo

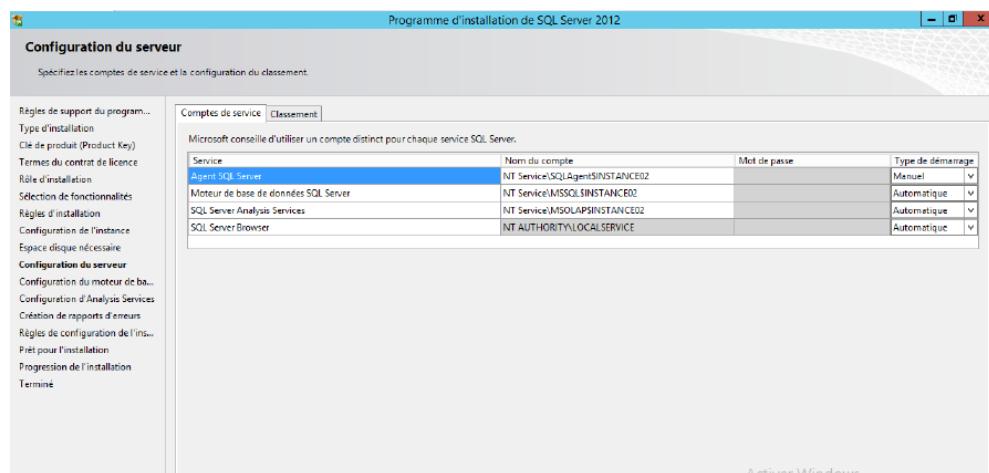


Figure 2.15: Etape 11 : Configuration du compte service

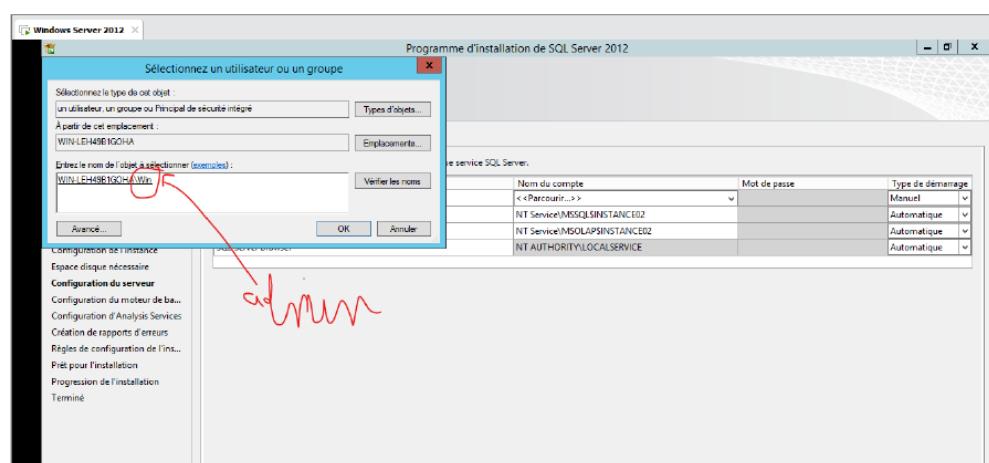
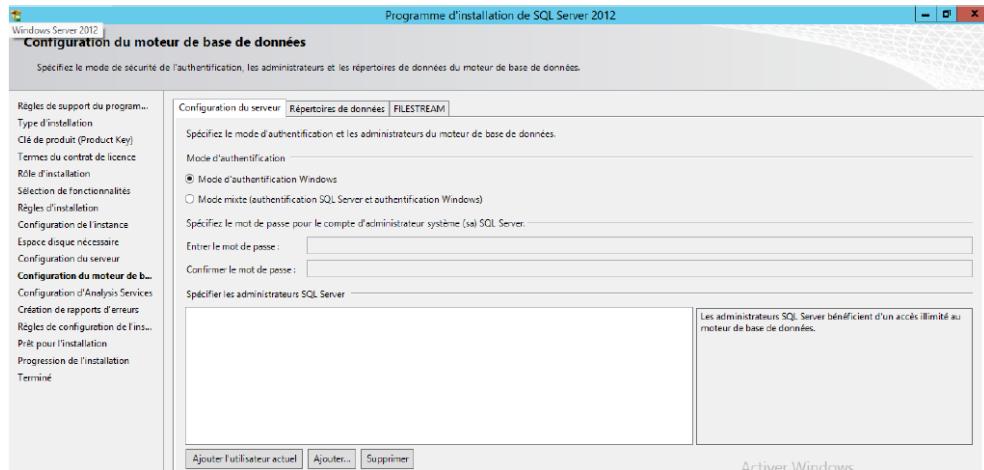


Figure 2.16: Etape 11 : Configuration du compte service

Nous avons précisé le compte d'utilisateur du domaine ( server ) qui sera utilisé pour l'ensembles des services proposé par SQL server, à l'occurrence

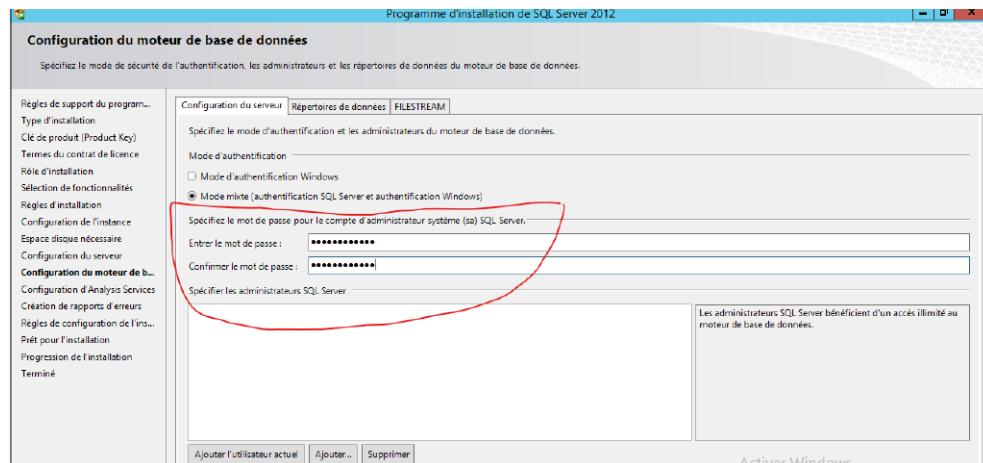
le compte local va faire l'affaire il possède tous les priviléges d'un administrateur. **Il est fortement recommander de spécifier un seul compte local qui a les priviléges d'administrateur** pour accéder aux différents services de sql server ( SQL agent, moteur de base de données, SQL SSIS...)



**Figure 2.17: Etape 12 : Configuration du moteur de base de données**

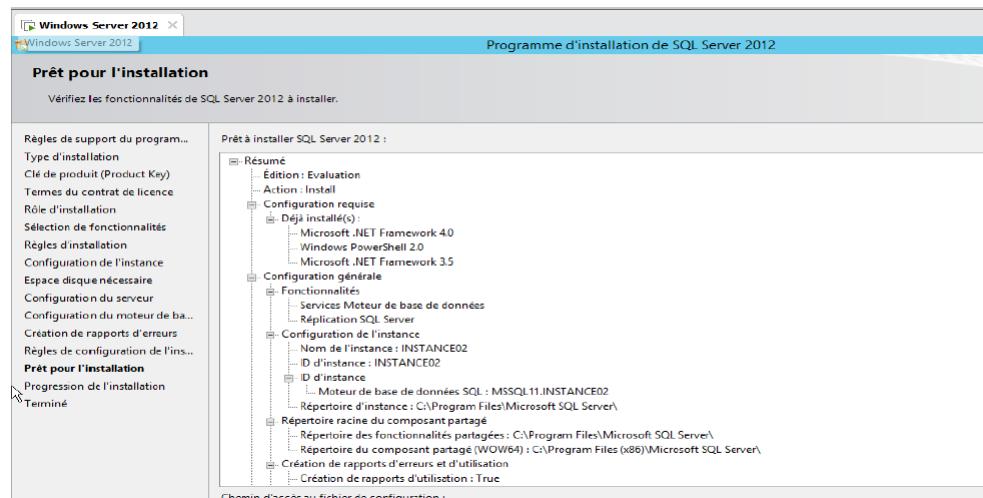
C'est dans cette partie qu'on va choisir le mode d'authentification de sql server, comment les accès vont être authentifier sur notre serveur, il y'a deux possibilité :

1. **Mode d'authentification Windows:** c'est le mode d'authentification du compte système à l'occurrence le compte de sql server 2012 ce qui est utiliser on production
2. **Le mode Mixte :** l'accès se fait par le système ou le server SQL server



**Figure 2.18: Etape 12 : Configuration du moteur de base de données**

Lorsque **le mode d'authentification mixte** est choisi, le programme d'installation **crée automatiquement un compte** qui possède les priviléges d'un administrateur nommé **(sa)**



**Figure 2.19: Etape 13 : Récapitulative de l'installation**



#### **NOTE IMPORTANTE**

A la fin du processus d'installation d'une instance SQL server, il va falloir **confirmer et valider** votre installation soit en accédant à **l'outil Service** mis à disposition par Windows soit **d'ouvrir le gestionnaire de configuration SQL server** et voir si le **moteur de base de données et son SQL agent** sont présent dans la liste des services.

# Chapter 3

## Installation silencieuse de SQL server 2012

### 3.1 Cration du fichier de configuration .ini



#### NOTE IMPORTANTE

Vous pouvez recuperer le fichier de ConFiguration.ini relative a une installation existante d'une instance SQL server **a partir du chemin physique suivant** : C:\Program Files \Microsoft SQL Server \110 \Setup Bootstrap \Log puis la deployer sur plusieurs environnements de travail creant ainsi plusieurs instances SQL server **avec la meme configuration**.

Afin de realiser ce type d'installation, **il va falloir mettre en place deux choses** :

1. Cre un fichier texte de configuration **avec l'extension .ini** qui va contenir

toutes les informations requises de l'installation.

2. Démarrage de l'installation automatique : exécuter **la ligne de commande ci-dessous** en faisant passer en paramètre le fichier de configuration créé au fichier exécutable **SETUPE.EXE**

**SETUPE.EXE /CONFIGURATIONFILE="CheminVersLeFichierDeConfiguration.ini"**

## 3.2 Installation silencieuse

### 3.2.1 Les étapes d'installation :

1. Depuis **les fichiers sources d'installation** identifier l'emplacement physique du ficheir exécutable **SETUPE.EXE**.
2. Créer le fichier de configuration s'il n'existe pas en ajoutant **les paramètres non exhaustifs** suivants :
  - (a) **IACCEPTSQLSERVERLICENSETERMS =”True”** : ce paramètre est généralement utilisé pour indiquer que l'utilisateur accepte les termes de la licence SQL server, **ce paramètre est obligatoire pour la poursuite de l'installation**
  - (b) **ACTION=”INSTALL”** : il s'agit également **d'un paramètre obligatoire**, il spécifie un flux de travail de programme d'installation, il accépte les valeurs suivantes **INSTALL,UNINSTALL ou UPGRADE**
  - (c) **QUIET=”True”** : En activant ce paramètre à ”True” le processus d'installation n'affichera aucune interface utilisateur.



**NOTE IMPORTANTE**

Lorsque le paramètre **QUIET** est désactivé, le programme d'installation d'une instance SQL server va lancer l'assistant en affichant l'interface graphique.

- (d) **FEATURES=»SQLENGINE»** : ce paramètre permet de spécifier les fonctionnalités à installer, désinstaller ou mettre à niveau, la liste des fonctionnalités pris en charge par ce paramètre comprend SQL, AS, RS, IS, MDS et Tools.
- (e) **INSTALLSHAREDDIR=»CheminPhysique»** : Permet de spécifier le répertoire racine d'installation racine pour les composants partagés. Ce répertoire reste inchangé après l'installation des composants partagés.
- (f) **INSTANCENAME=»NomInstance»** : ce paramètre permet de spécifier une instance par défaut ou instance nommée. MSSQLSERVER est l'instance par défaut.
- (g) **INSTANCEID=»NomInstance»** : permet de spécifier l'ID de l'instance des fonctionnalités SQL server
- (h) **ERRORREPORTING=»False»** : spécifier les erreurs peuvent être signalées à Microsoft afin d'améliorer les versions à venir de SQL server.
- (i) **INSTANCEDIR=»CheminPhysique»** : spécifier le répertoire d'installation.
- (j) **AGTSVCPASSWORD=»MotDePasse»** : cela permet de spécifier le mot de passe du compte service de l'Agent.
- (k) **AGTSVCACCOUNT=»CheminPhysique»** : cela permet de spécifier Le nom du compte service de l'Agent, dans la majorité des cas, le compte local ou du domaine d'administration Windows est choisi
- (l) **AGTSVCSTARTUPTYPE=»Automatic»** : il permet de démarrer le service automatiquement après l'installation

- (m) **SQLCOLLATION=»French\_CI\_AS»** : Spécifie un classement Windowsou SQL à utiliser pour le moteur de base de données.
- (n) **SQLSVCACCOUNT=»Domaine \Utilisateur»** : spécifier l'identifiant du compte service du moteur de base de données
- (o) **SQLSVCPASSWORD=»MotDePasse»** : Spécifier le mot de passe du compte service du moteur de base de données.
- (p) **SQLSYSADMINACCOUNTS=»Domaine \Utilisateur»** : Spécifier le compte Windows à configurer avec les privilèges d'accès d'administrateur système SQL server
- (q) **SECURITYMODE=»SQL»** : La valeur par défaut est l'authentification Windows, spécifier la valeur SQL” pour l'authentification Mixte.
- (r) **TCPENABLED=»1»** : Spécifier 0 pour désactiver le protocole TCP/IP ou 1 pour l'activer.

# Chapter 4

## Les outils d'administration SSMS/SQLcmd

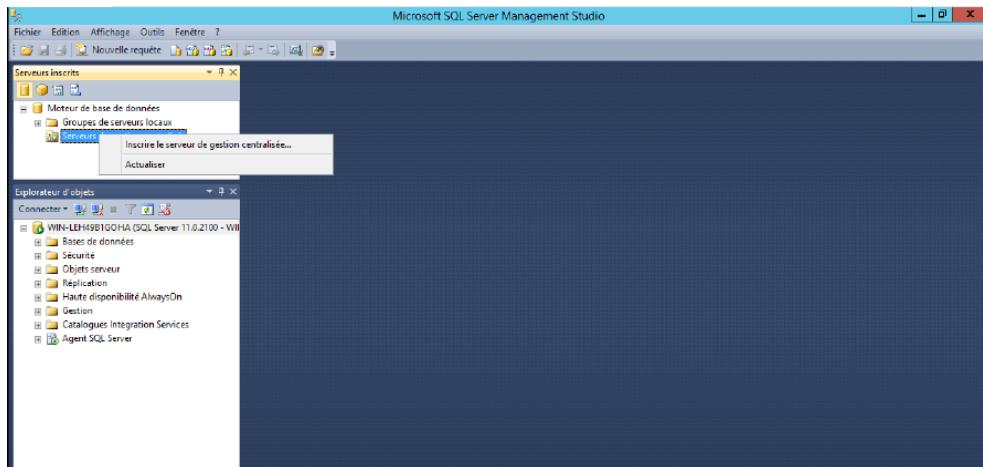
### 4.1 Presentation de SQL server management studio (SSMS)

#### 4.1.1 Qu'est ce que c'est SQL server management studio

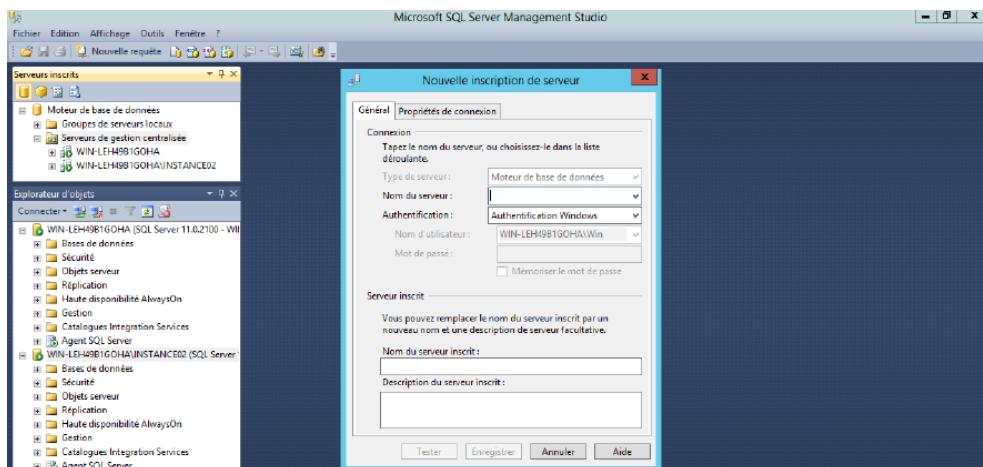
1. **Outil principal de gestion et d'administration** destiné à l'administrateur et aux développeurs ( qui permet de réaliser la programmation procédurale à travers le T-SQL )
2. Outil graphique des instances et des objets
3. **Gestion centralisée des instances distantes** ( dans le cas où vous avez plusieurs instances localisées sur plusieurs sites géographiques)

Pour vous permettre de réaliser la gestion centralisé des instances, il faut tous d'abord les référencier dans l'explorateur d'objet en suivant le

chemin suivant : **Affichage \ Serveurs Inscrits \Inscrire le serveur de gestion centralisée**



**Figure 4.1: Etape 1 : Référencier les instances**



**Figure 4.2: Etape 2 : Référencier les instances**

Dans cette nouvelle fenêtre de server, nous pouvons sélectionner toutes les instances dans le champ "nom du server" pour les rassembler dans un seul server de gestions centralisé, **on peut ajouter autant de server inscrit que d'instance à gérer**

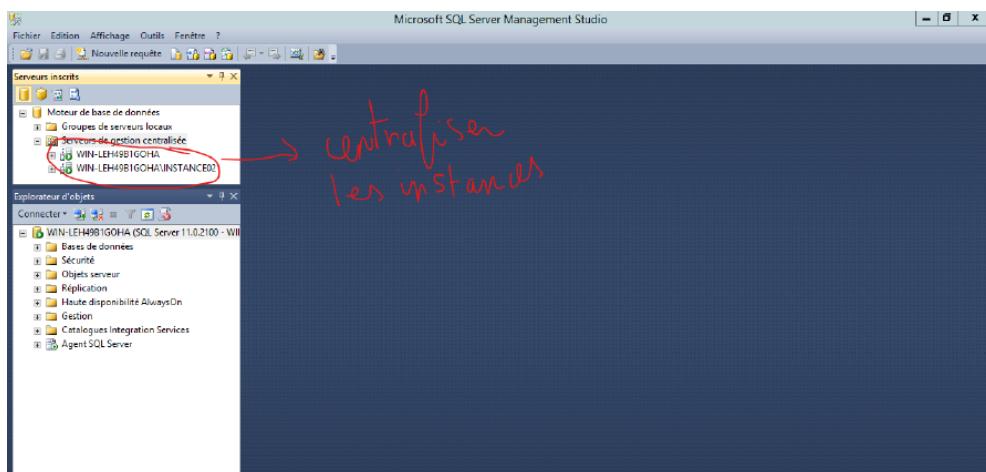


Figure 4.3: Etape 3 : Référencier les instances

Comme le montre la Figure 4.3, **deux instances** appartenant à deux serveurs distinct **ont été rassembler dans un seul groupe de gestion centralisé**

#### 4.1.2 Se familiariser avec l'interface graphique SSMS

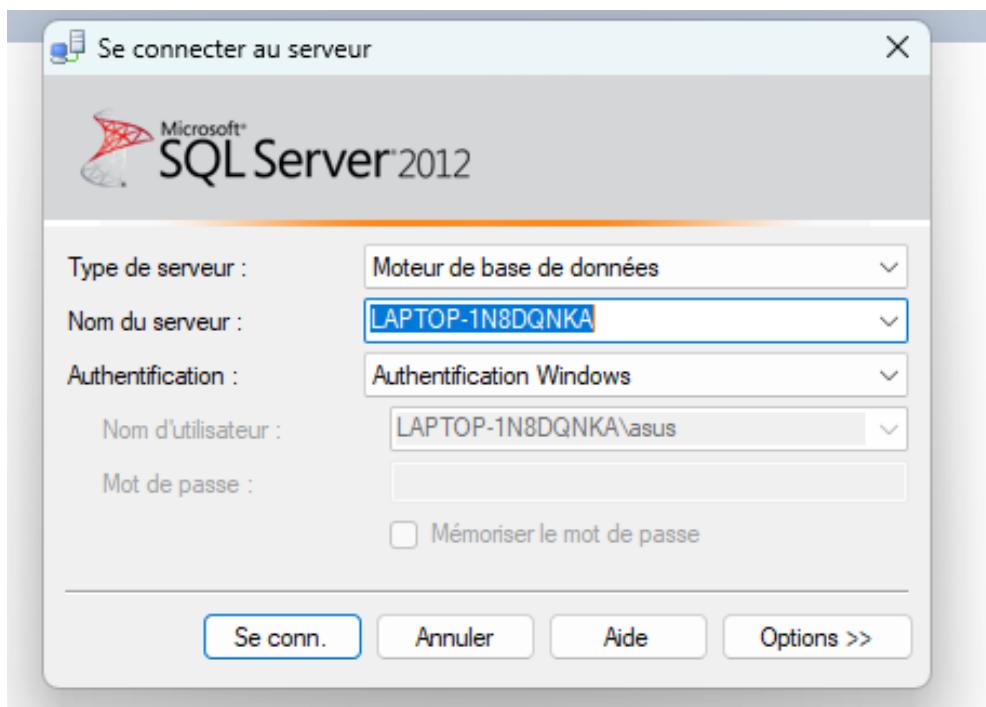


Figure 4.4: La boîte de dialogue de connexion

Quand vous allez exécuter une session SSMS, une boite de dialogue s'affichera pour introduire **les informations de connexion** à l'instance de base de données à savoir le type de serveur, Le nom du serveur (Instance par défaut ou Instance nommé), le mode d'authentification ( Mixe ou Windows). **Pour plus de détails, veuillez consulter la section 10.2 du chapitre 10**

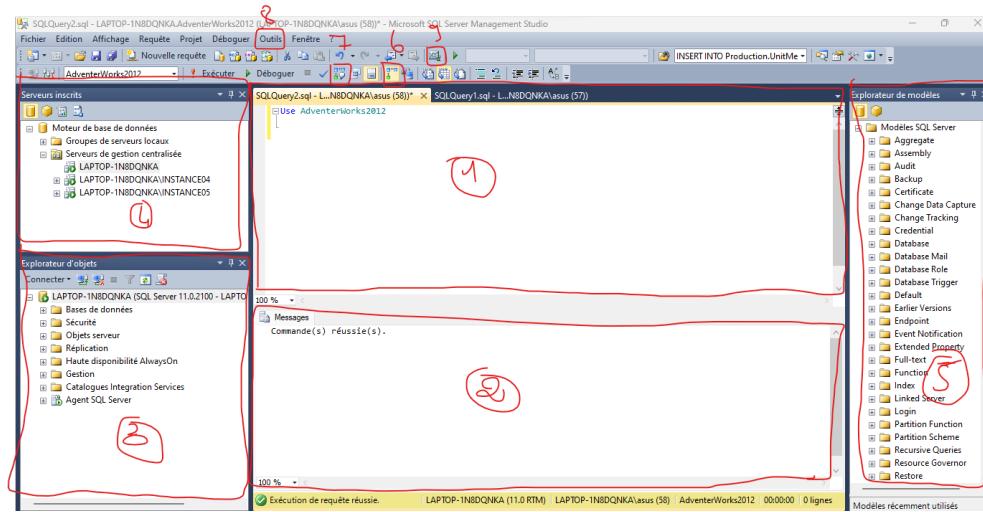


Figure 4.5: Présentation de l'interface utilisateur SSMS

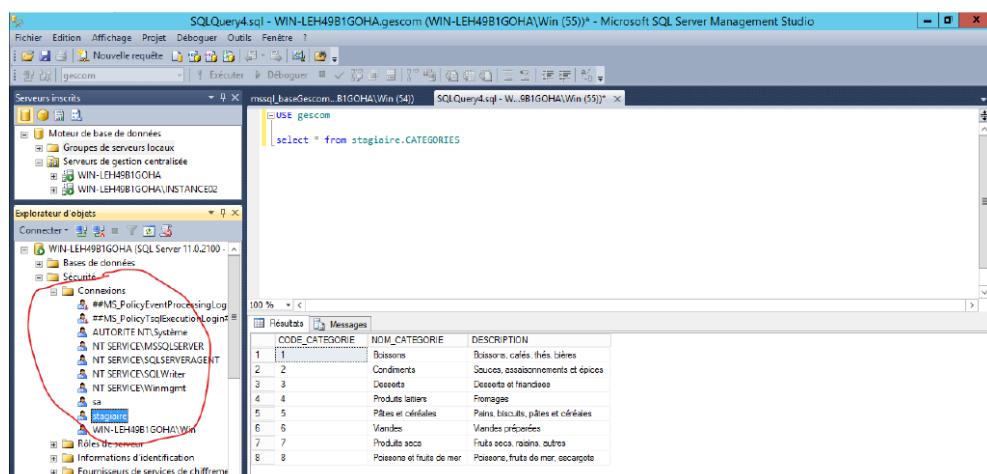
Une fois que les informations de connexion ont été introduites correctement, vous pouvez dès à présent se connecter et l'interface graphique SSMS s'affichera en vous offrant 4 principales sections. **La première section** représente la fenêtre de requetage dans laquelle vous allez exécuter vos requêtes SQL pour interroger les bases de données, développer vos scripts T-SQL pour les opérations d'administration classique, **la seconde section** ni plus ni moins que la sortie standard dans laquelle le serveur de base de données va répondre à l'utilisateur suite à sa demande formulée sous forme d'une requête SQL. **La troisième** traduit la racine de l'instance à laquelle l'utilisateur est connecté, elle est constituée de plusieurs nœuds dont on détaillera la signification dans la section suivante de ce chapitre. Enfin **la quatrième et la cinquième sections** ne sont pas affichées par défaut lorsque vous vous connectez pour la première fois à une instance SQL server. Je vous recommande vivement de les faire apparaître (voir comment centraliser les instances Section 4.1.1) et (comment afficher l'explorateur

de modèles). L'explorateur de modèle va vous offrir des scripts prédéfinis pour l'administartion que vous pouvez personnaliser selon vos préférences et vos besoins.

#### 4.1.3 Présentation de l’arborescence au niveau répertoire de l’instance

Dans les figures ci-dessous, nous allons présenter quelques répertoires de la racine de l'instance qui se trouvent dans l'explorateur d'objets :

A partir de l'onglet "sécurité", vous pouvez visualiser **la liste des comptes utilisateurs** qui peuvent se connecter à une instance SQL server.



**Figure 4.6:** Le répertoire sécurité

Dans la partie "**”objet server”**", créer les unités de sauvegardes logique, créer les déclencheurs au niveau serveur...

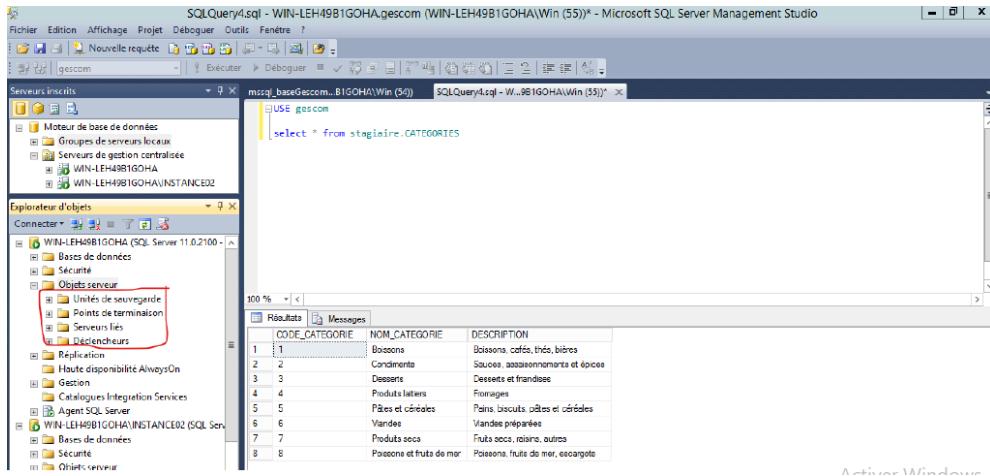


Figure 4.7: Le répertoire Objets serveur

Dans la partie "réPLICATION", on retrouve l'ensembles des informations qui peuvent gérer la copie et la distribution des données au niveau server

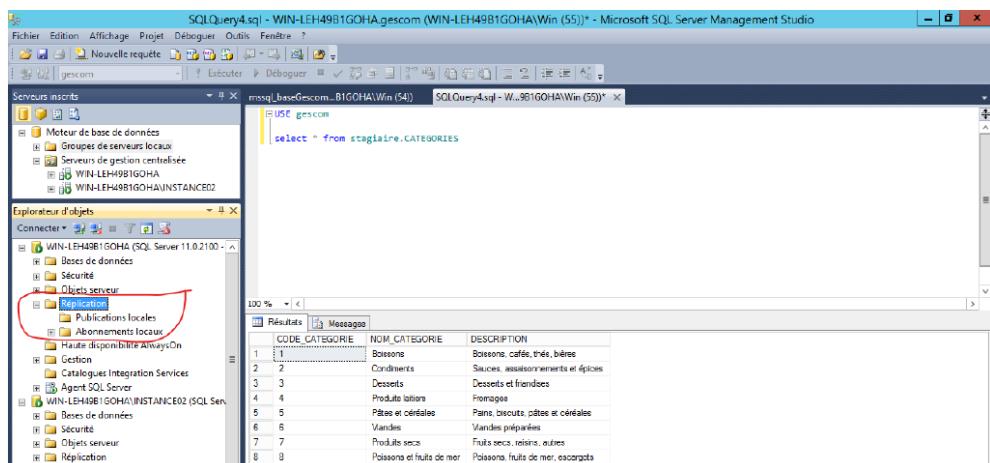


Figure 4.8: Le répertoire réPLICATION

Dans la partie "Gestion", on retrouve l'ensembles des informations qui peuvent réaliser des plans de maintenance, gérer la collecte de données, le catalogue Intégration services, la configuration de la messagerie de base de données "DataBaseMail"

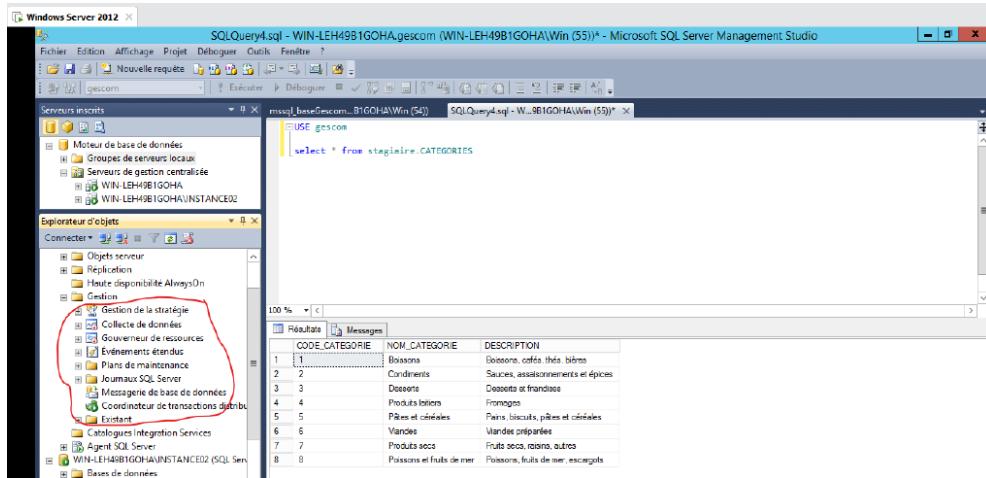


Figure 4.9: Le répertoire gestion

## 4.2 Présentation de SQLcmd

### 4.2.1 Qu'est ce que SQLcmd

1. Représente **un outil de gestion et d'administration en mode ligne de commande**
2. Exécuter des requêtes et de scripts
3. **Etablir une connexion d'administration dédié (DAC)** : une fonctionnalité de sql server 2008 qui permettra l'accès au server lorsque ce dernier sera bloqué ou non disponible. ( ex : restaurer la base de données master )

### 4.2.2 SQLcmd ligne de commande

1. La commande **sqlcmd - ?** : affiche le manuel d'utilisation qui contiendra l'ensemble des options et arguments associés à la commande sqlcmd.

```
C:\Users\Win>sqlcmd -?
Outil en ligne de commande Microsoft (R) SQL Server
Version 11.0.2100.60 NT x64
Copyright (c) 2012 Microsoft. Tous droits réservés.

utilisation : Sqlcmd      [-U ID de connexion]      [-P mot de passe]
[-S serveur]      [-H nom d'hôte]      [-E connexion approuvée]
[-N Chiffrer la connexion][ -C Faire confiance au certificat du serveur]
[-d utiliser le nom de base de données] [-l délai de connexion]      [-t délai
de requête]
[-h[en-têtes]]      [-s séparateur de colonnes]      [-w largeur d'écran]
[-a taille du paquet]      [-e entrée d'écho]      [-I Activer les identifi
icateurs marqués]
[-c fin de commande]      [-L[c] répertorier les serveurs[nettoyer la so
rtie]]
[-q « requête ligne de commande »]      [-Q « requête ligne de commande » et quit
ter]
[-m niveau d'erreur]      [-U niveau de gravité]      [-W supprimer les espac
es de fin]
[-u sortie Unicode]      [-r[0:1] messages vers stderr]
[-i fichier d'entrée]      [-o fichier de sortie]      [-z nouveau mot de
passe]
[-f <page de codes> : i:<page de codes>[,o:<page de codes>]] [-Z nouveau mot d
e passe et quitter]
[-k[1:2] supprimer[replacer] les caractères de contrôle]
[-y largeur d'écran de longueur variable]
[-Y largeur d'écran de longueur fixe]
[-p[1] imprimer les statistiques[colon format]]
[-R utiliser les paramètres régionaux du client]
[-K intention de l'application]
[-M basculement de plusieurs sous-réseaux]
[-b Abandonner le lot d'instructions après erreur]
[-v var = « valeur »...] [-A Connexion admin dédiée]
[-X[1] désactiver les commandes, le script de démarrage, les variables d'environnement]
```

Figure 4.10: afficher le manuelle d'utilisation

2. **L'option -E** : permet d'établir une connexion approuvée Windows
3. **L'option -H** : donne le nom ou l'adresse IP de la machine locale
4. **L'option -S** : permet de spécifier le nom du serveur sur lequel se trouve l'instance
5. **L'option -U** : spécifier l'ID de l'utilisateur
6. **L'option -P** : spécifier le mot de passe



#### NOTE IMPORTANTE

N'oubliez bien de taper l'instruction "GO" après l'exécution de la requête pour indiquer au moteur de base de données d'envoyer la demande au serveur à des fins de traitement, dans le cas contraire, le standard output vous affichera une ligne vide.

# Chapter 5

## Configuration de SQL server 2012

### 5.1 Introduction

La configuration des différents services SQL Server tel que le moteur de base de données et SQL agent vont se faire de deux moyens possibles :

1. **Le gestionnaire de configuration SQL server**
2. **Le gestionnaire de Services Windows**

A travers ces deux outils de configurations, vous êtes en mesure de connaître l'état des différents services de SQL Server 2012 (Démarrer, arrêt, suspendu)

1. **Démarrer** : prêt à accépter les connexions locale ou distante, traiter les requêtes et les différents Clients
2. **Suspendu** : Les nouvelles sessions ne sont plus accéptées, les sessions précédentes ne sont plus impactées par cet état.

3. **Arreté** : Le service est indisponible

## 5.2 Le gestionnaire de configuration SQL server

### 5.2.1 La gestion des différents Services

Utilisez le gestionnaire de configuration SQL server pour **démarrer, suspendre, reprendre ou arreter un Service, afficher les propriétés du compte service ou modifier les propriétés des services.**

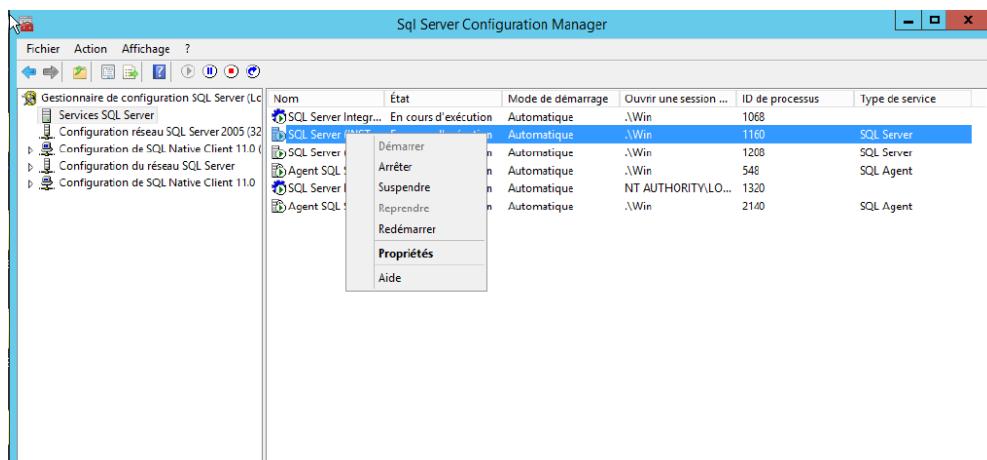


Figure 5.1: Le gestionnaire de configuration SQL server

**Gestionnaire de configuration SQL Server** est un outil permettant de gérer les services associés à SQL Server, de configurer les protocoles réseau utilisés par SQL Server et de gérer la configuration de la connectivité réseau à partir d'ordinateurs clients SQL Server.

## Modification des comptes utilisés par les services

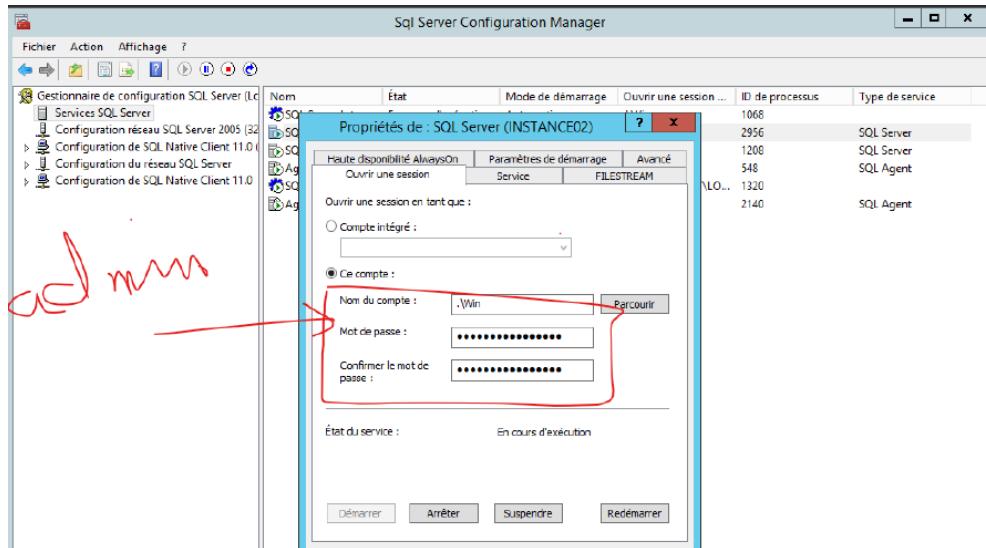


Figure 5.2: Afficher les propriétés du compte service

En cliquant sur propriétés du service SQLserver, une fenêtre s'ouvre pour afficher l'ensemble d'informations sur le compte possédant le droit d'administration système. **Ce compte a été définis lors du processus d'installation SQL server 2012.**

### 5.2.2 La gestion des protocoles réseau du Serveur et du Client

Le Gestionnaire de configuration SQL Server vous permet également de **configurer des protocoles réseau serveur et client** et des options de connectivité.

A partir de chaque instance , vous pouvez à l'aide de gestionnaire de configuration SQL server de **reconfigurer les connexions de serveurs** de façon à ce que SQL server écoute sur **un port** ou **un canal particulier** pour le faire suivie la figure suivante :

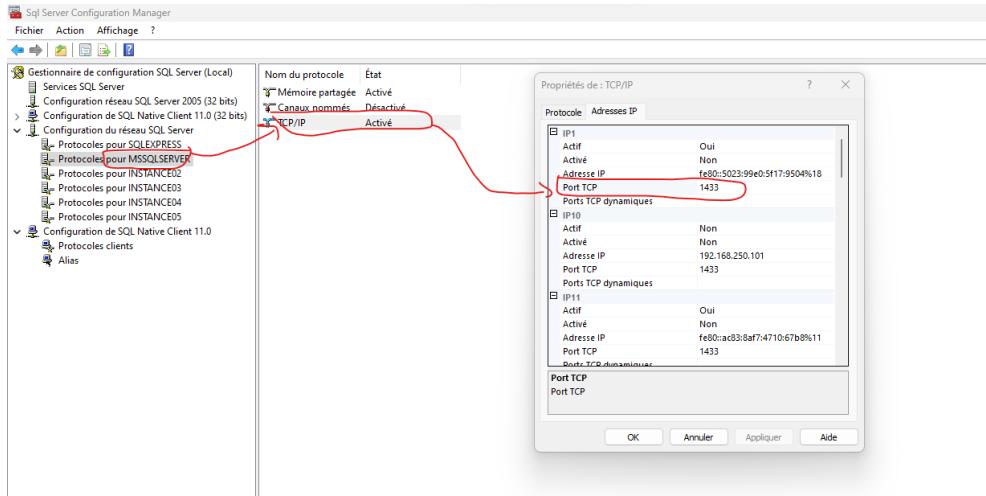


Figure 5.3: reconfigurer le port d'écoute de l'instance

Par défaut, les ports standard utilisés par SQL Server et les services de moteur de base de données associés sont : **TCP 1433, 4022, 135, 1434**

SQL server prends en charge plusieurs protocoles de communication qui permettent aux clients de se connecter à une instance SQL server, pour chaque instance, **il existe 3 protocole de communication :**

1. **TCP/IP** : C'est un protocole standard pour la communication réseau, adaptés aux connexions distantes via internet
2. **Named Pipes** : Est un protocole de communication qui utilise les canaux nommés du système d'exploitation.
3. **Shared Memory** : C'est un protocole de communication qui utilise la mémoire partagé du système d'exploitation



### NOTE IMPORTANTE

La principale distinction entre le protocole Shared Memory et Named pipe réside dans leur principe de fonctionnement. **Le Name Pipes** peut être utilisé à la fois pour **les communications locales et distantes**. tandis que **Shared memory** est adapté uniquement pour **les connexions locales**.

## 5.3 Le gestionnaire de service Windows

Cette fonctionnalité mis à disposition par Windows représente une autre alternative à l'utilisateur pour **la gestion et la configuration des différents services**.

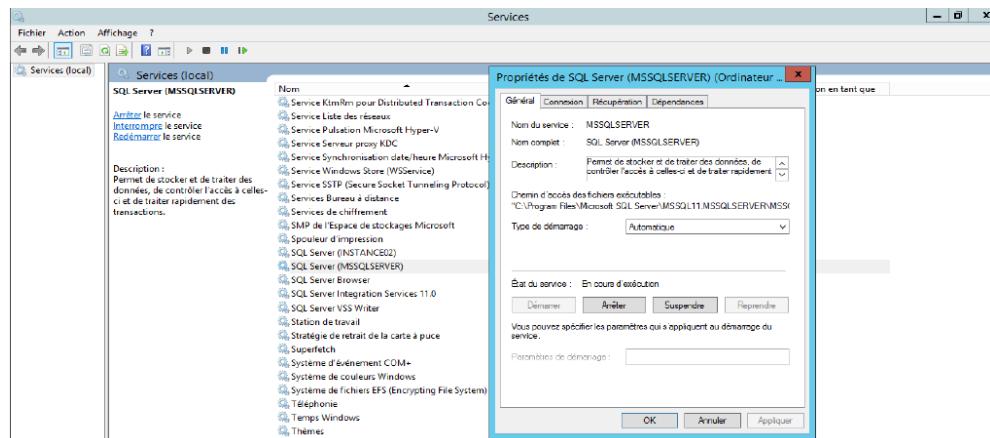


Figure 5.4: Gestionnaire de Service Windows

A partir du services Windows nous pouvons afficher les différents états dans lesquels les instances se trouvent, nous avons en plus de ces informations, le chemin d'accès vers le répertoire de la racine de l'instance.

## 5.4 Inscription des serveurs

Pour pouvoir gérer l'**administration et la configuration des différents servers distants**, il faut tout d'abord les inscrire en créant des **groupes de serveurs locaux**, cela facilitera à l'administrateur la **gestion centralisé** de plusieurs instances éparpiller sur plusieurs sites géographiques ( FRANCE, INDE, CHINE, ESPAGNE..). le raccourcis pour inscrire vos serveurs est **CTRL+ALT+G**

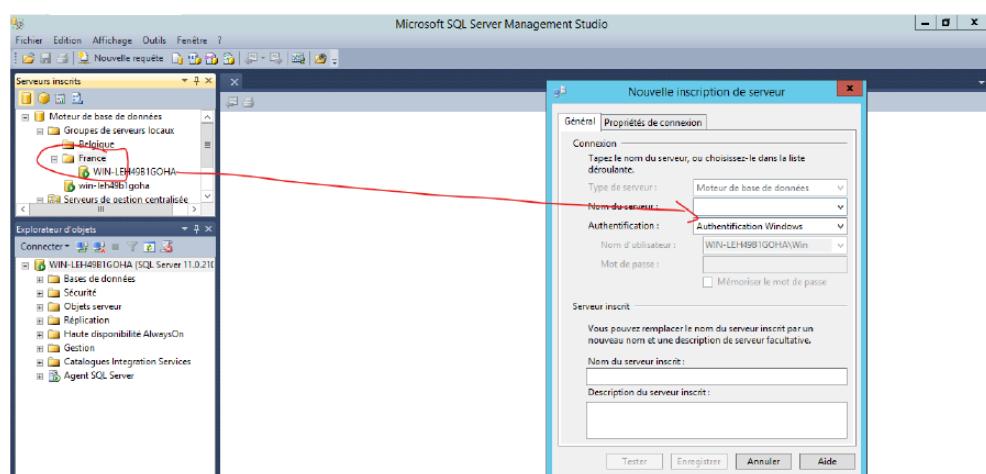
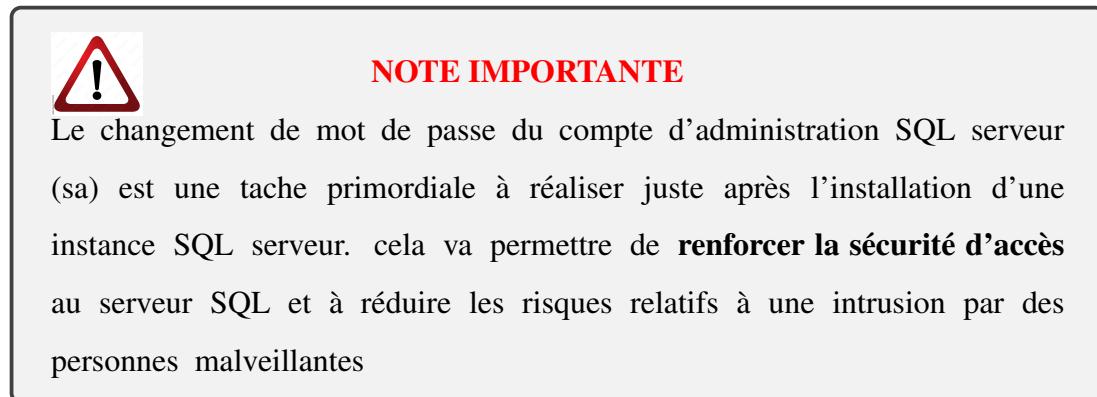


Figure 5.5: Inscription des serveurs de base de données

## 5.5 Configuration et administration des serveurs

### 5.5.1 Changement du mot de passe de l'administrateur (sa)



à l'aide de l'instruction T-SQL illustré dans la figure ci-dessous vous allez pouvoir changer le mot de passe du compte (sa). **Cette action est requise** après l'installation de SQL server.

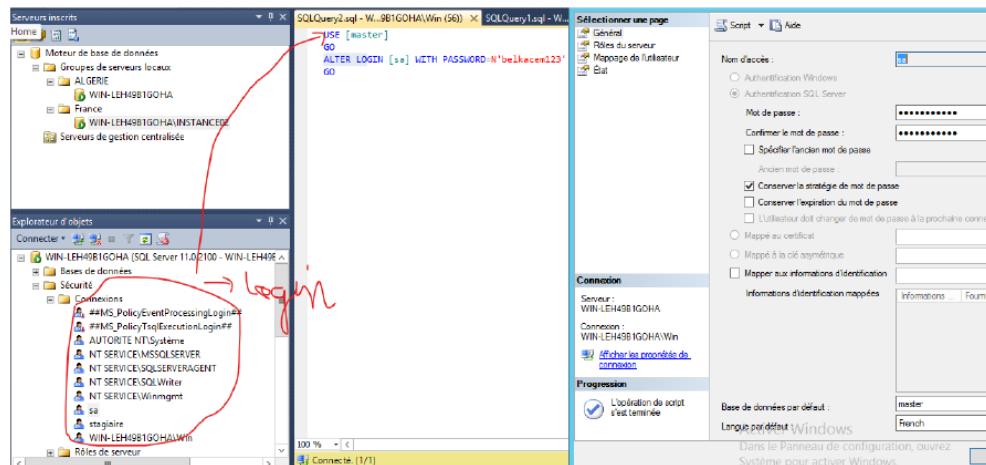


Figure 5.6: Changement du mot de passe du compte (sa)

### 5.5.2 Dimensionnement du cache SQL server

# Chapter 6

# Les bases de données sous SQL server 2012

## 6.1 Architecture global d'une base de données

Elle contient deux types de fichiers : ***les fichiers de données*** et ***les fichiers journaux***. ***Les fichiers de données*** sont réparties en fichiers de données **primaires obligatoires (.mdf)** qui stockent une collection de **métadonnées** sur la structure de la base de données en incluant des informations sur les tables, les index, les colonnes et les contraintes d'intégrités...etc et **Les fichiers secondaires (.ndf)** qui stockent à leurs tour les objets et les données utilisateurs au niveau applicatif.

***Les fichier journaux*** (.ldf) quant à eux, ils enregistrent **les opérations de modification de données** sur la base de données (**INSERT, UPDATE, DELETE**). L'objectif est de pouvoir restaurer les données de la base en cas de crash imprévisible.

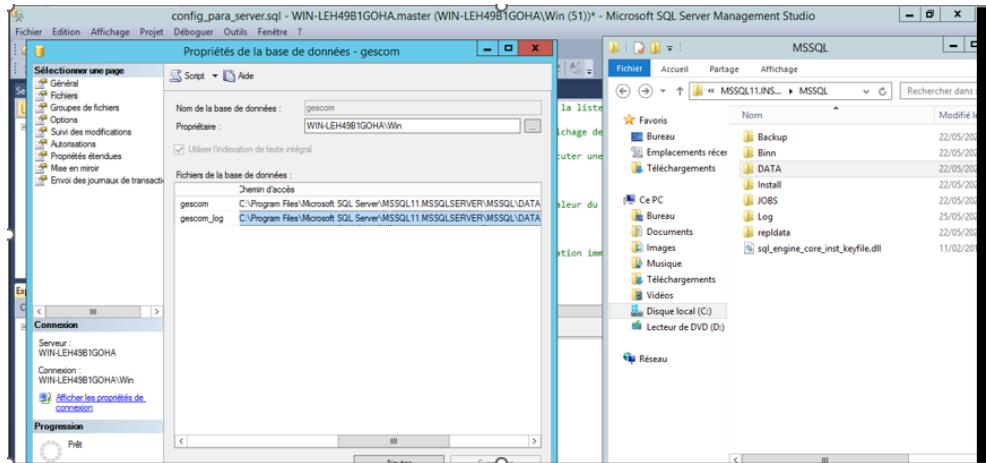


Figure 6.1: les fichiers de données et fichiers journaux

La figure 6.1 ci-dessous montre comment pouvoir connaître la composition et le chemin d'accès aux différents fichiers d'une base de données SQL server.

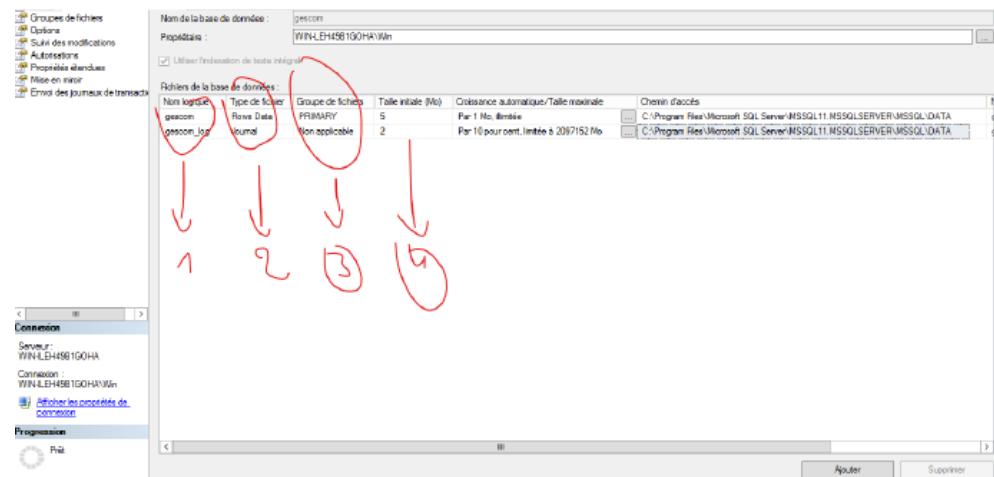
En cliquant sur propriété de la base de données, nous sommes en mesure d'identifier les noms logiques des différents fichiers qui composent la base de données ainsi que leur **rédertoires de stockage physique** au niveau du système de fichiers.



#### NOTE IMPORTANTE

Pour de meilleure performance système, il va falloir **répartir le stockage** des fichiers data sur un point de montage différent de celui des fichiers journaux. **Par défaut** l'emplacement physique des deux fichiers sont identiques

Une base de données par défaut contient au **minimum un fichier data et un fichier journal**. Le fichier data est rattaché obligatoirement au groupe de fichier **PRIMARY** qui représente l'unité de stockage logique choisie par SQL server. La figure 6.1 ci dessous montre un exemple d'une base de données "gescom" constitué d'un fichier de données avec **le nom logique "gescom"** et un fichier journal de transactions avec **le nom logique "gescom.log"**, le fichier data appartient **au groupe de fichier PRIMARY** dont la **taille initiale est de 5MO**, la



**Figure 6.2: Présentation de la propriété de la base de données**

**croissance des segments de données** est gérée **automatiquement** par le moteur de base de données en rajoutant un espace de stockage de 1MO à chaque fois que le fichier data à besoin de s'étendre.

En bas à droite de cette figure, vous trouverez le bouton "**ajouter**" si vous souhaitez ajouter d'autres fichiers data secondaires à la base de données qui porteront l'extension ".ndf", ce fichier aura pour rôle de **stocker les objets et les données utilisateurs au niveau applicatif**, pour vérifier son emplacement physique, il suffit d'y accéder via le chemin réel.

### 6.1.1 Les fichiers de données

les fichiers de données d'une base de données SQL server sont caractérisés par les points suivants :

- Le fichier Data est **rattaché obligatoirement à une seule base de données**
- Ils sont **structurés en pages** de 8ko, les pages contiennent les enregistrements
- La page 8ko représente **l'unité d'échange** entre deux éléments physiques du serveur, **le disque et la mémoire cache**
- **La structure de chaque ligne** d'une table est organisée de manière à ce qu'elle puisse tenir **sur seule page**



**Figure 6.3: La structure du fichier data**

**Description :** La figure montre **a structure physique d'un fichier de données**. ce dernier est structuré en **bloc de 8ko** qui forment à leur tour des **extensions de 64ko**. Le moteur SQL server va les formater de cette manière à chaque fois que vous créer une nouvelle base de données.



### NOTE IMPORTANTE

- **Les extension** : représente **un regroupement contigu** de blocs de 8k
- La taille d'une extension est de 64k 8\*8
- Les extensions sont créé automatiquement par le moteur de base de données lorsqu'il y'a besoin de plus de place
- La première extension créée automatiquement par le moteur sera en principe **mutalisé** mais dès qu'il y aura besoin de plus d'espace, le système va créer **des extensions spécialisées** afin d'éviter de gaspiller de l'espace de stockage dans le disque
- Il existe deux types d'extensions :
  1. Mixte : contient les données de plusieurs objets (**mutualisé**)
  2. Uniforme : Contient les données d'un seul objet (**spécialisé**)

## 6.2 Principe de fonctionnement des fichiers journaux

Pour bien pouvoir administrer un serveur, il faut savoir l'architecture du fonctionnement du moteur au niveau des fichiers journaux. La figure 6.4 décrit toutes les étapes de journalisation des données modifiées au niveau de la base de données.

1. **Etape 1** : Lorsqu'un client procède aux modifications de données à travers les instructions INSERT, UPDATE, DELETE ces modifications vont être faite au niveau du cache de données de SQL server.
2. **Etape 2** : le processus SQL server va ensuite charger les enregistrements à modifier dans la plus petite unité de stockage au niveau du disque pour les transporter vers la mémoire tampon.

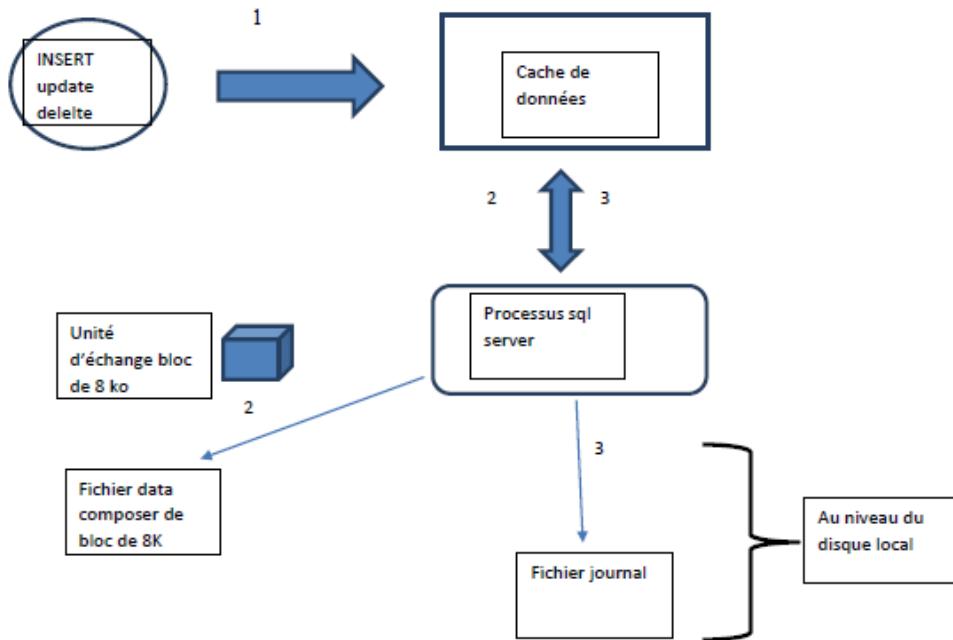


Figure 6.4: Le principe de journalisation sous SQL server 2012

3. **Etape 3 :** une fois les modifications sont faites, SQL server va écrire cette modification dans le fichier journal. **aucune données ne sera écrite dans le fichier data tant que la transaction n'a pas été écrite dans le fichier journal**
4. **Etape 4 :** En suite via **le processus Checkpoint** la modification va être faite via le cache de données dans les fichiers data sous forme de bloc 8ko qui représente l'unité d'échange entre le cache et le disque

## 6.3 Crédation d'une base de données

### 6.3.1 En mode T-SQL

Comme le montre la figure 6.5, vous avez le choix entre l'interface graphique SSMS ou en mode T-SQL pour créer une base de données en indiquant les informations spécifiques relatives aux fichiers data et aux fichiers journaux.

Les spécifications sont :

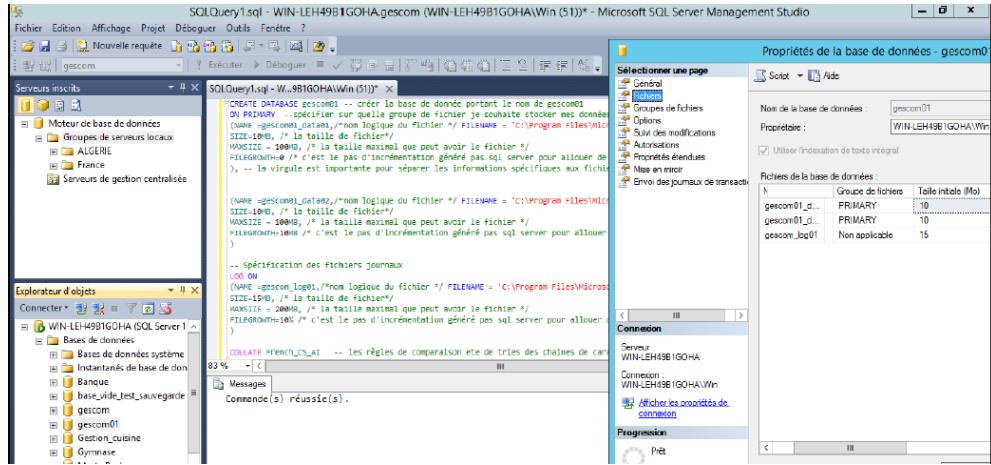
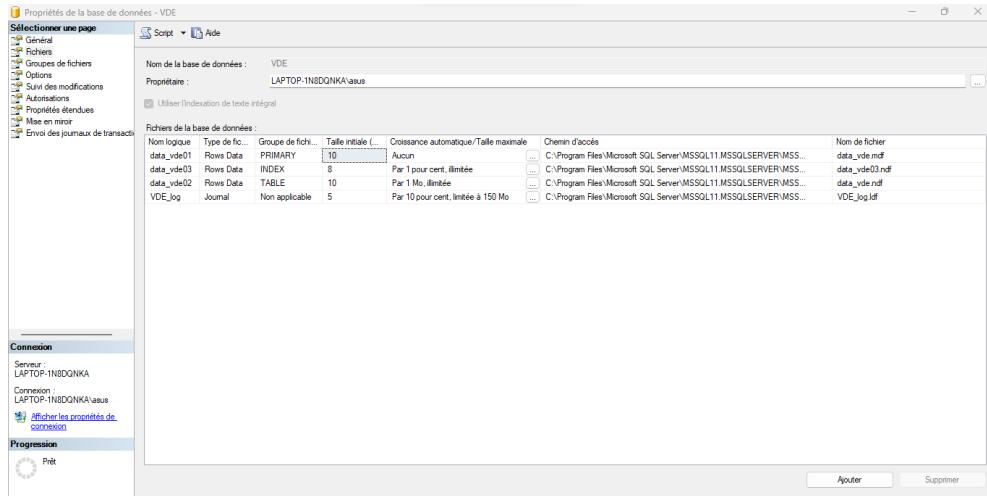


Figure 6.5: Création d'une base de données en mode T-SQL

1. **Name** = nomLogique
2. **FILENAME**= chemin physique d'accès dans le disque
3. **SIZE** = la taille en KB MB GB TB
4. **MAXSIZE**= la taille maximal du ficheir de données ou d'un fichier journal
5. **FILEGROWTH** : représente le pas d'incrémentation généré par SQL server dans un ficheir data en cas de besoin d'un espace supplémentaire. En d'autre terme c'est la atille à formaté par SQL server pour pouvoir stocker de nouvelles données dans le fichier data

Ce script T-SQL sert principalement à créer une nouvelle base de données en spécifiant les caractéristiques des ficheirs de données et le fichier journal de transaction. Dan cette étude de cas nous abons spécifier un fichier .mdf obligatoire pour qu'on puisse parler de base de données et fichier secondaire .ndf qui sont organisés logiquement dans un groupe de fichier PRIMARY.

Ensuite nous avons commencé par spécifié les caractéristiques de chaque fichier data en précisant : **le nom logique** du fichier, **l'endroit** où le fichier va etre stocker physiquement, **la taille initiale**, **la taille maximale** qui peuvent pccuper



**Figure 6.6: Création d'une base de données avec SSMS**

dans le disque et **le pas d'incrémentation** dont le rôle est d'ajouter des extants au fichier data en cas de besoin.

En ce qui concerne le fichier journal, la même démarche de spécification de caractéristique est effectuée, **la seule différence est dans l'endroit où on a choisi pour stocker physiquement notre fichier journal (LOG)**. Cette méthode est fortement recommandée à faire dès la rentrée de jeux pour optimiser le fonctionnement du système ( séparer l'emplacement du fichier data du fichier journal)

### 6.3.2 En mode interface graphique

La création de la base de données en mode graphique est assez simple à mettre en place, faites un clic droit sur **le noeud "Base de données"** au niveau de l'explorateur d'objet, puis il suffit de suivre le chemin d'accès suivant: **nouvelle base de données / générale**, dans la fenêtre qui s'affiche **"fichiers de la base de données"**, ajouter les spécifications des fichiers data et fichiers journaux que vous souhaitez.

## 6.4 La gestion d'une base de données

Gérer une base de données consiste à faire la gestion de l'espace de stockage qui peut se faire 4 modes possibles :

1. **AUTOEXTEND ON** (accroissement dynamique) : lorsque cette option de base de données est activé, les fichiers data vont croître automatiquement lorsque par exemple vous voulez ajouter des enregistrements dans une table



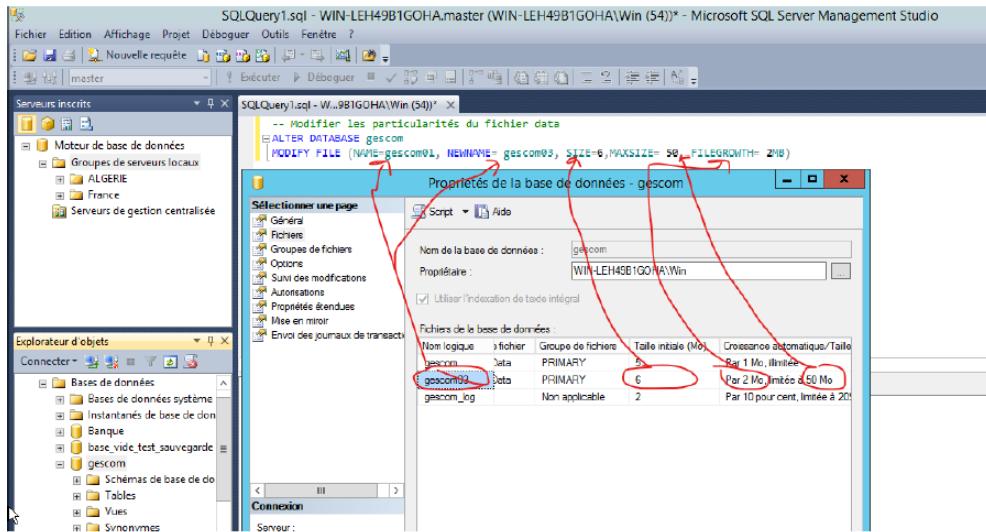
### NOTE IMPORTANTE

Cette manière de faire est déconseillé, tout simplement parce que quand le fichier d'autoextand ça induit une consommation des ressources qui peuvent pénaliser les performances du système.

2. **AUTOEXTEND OFF** : l'accroissement des fichiers est désactivé, il faut le faire manuellement si vous voulez ajouter des enregistrements supplémentaires ( si le fichier atteint sa taille maximal, c'est à vous d'ajouter de l'espace manuellement au fur et à mesure d'un besoin applicatif)
3. Ajout de fichier data ou fichier journal
4. **Libération de l'espace inutilisé** : quand vous faites un TRUNCATE OU UN DELETE sur une table, l'espace qui a été alloué pour ces données ne sera pas libéré automatiquement, vous devriez impérativement planifier manuellement la libération de l'espace inutilisé

### 6.4.1 Modifier un fichier data

Ce cas de figure montre un exemple de modification d'un fichier de données. Nous avons procéder à la modification des particularités du fichier data secondaire à l'aide de la commande **ALTER DATABASE** avec laquelle on a spécifié **un nouveau nom logique, un nouveau espace de stockage alloué à ce fichier, une**



**Figure 6.7: Modification d'un fichier de données**

**taille maximal** de 50Mo et enfin un **pas d'incrémentation des extants** de 2Mo.  
les flèches en rouge indiquent l'équivalence de ce changement fait en mode T-SQL vers le mode graphique.

Veuillez prendre note que cette démarche de modification est valable aussi pour les fichiers journaux de la base de données

#### 6.4.2 Ajouter un fichier data

Pour ajouter un fichier data supplémentaire à une base de données existante, il faut utiliser la commande décrite dans la figure ci-dessous

#### 6.4.3 Suppression d'un fichier data

Pour supprimer un fichier data d'une base de données existante, il faut utiliser la commande décrite ci-dessous

#### 6.4.4 Libération de l'espace de stockage

Quand vous faites un gros chargement de données ensuite ces données sont supprimées, est bien l'espace inutilisé n'est pas libéré ou rendu au système d'exploitation,

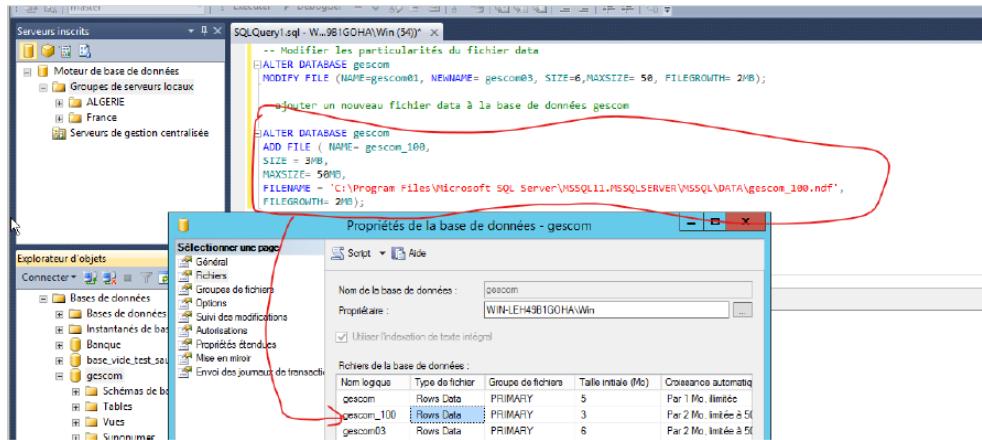


Figure 6.8: Ajouter un fichier data

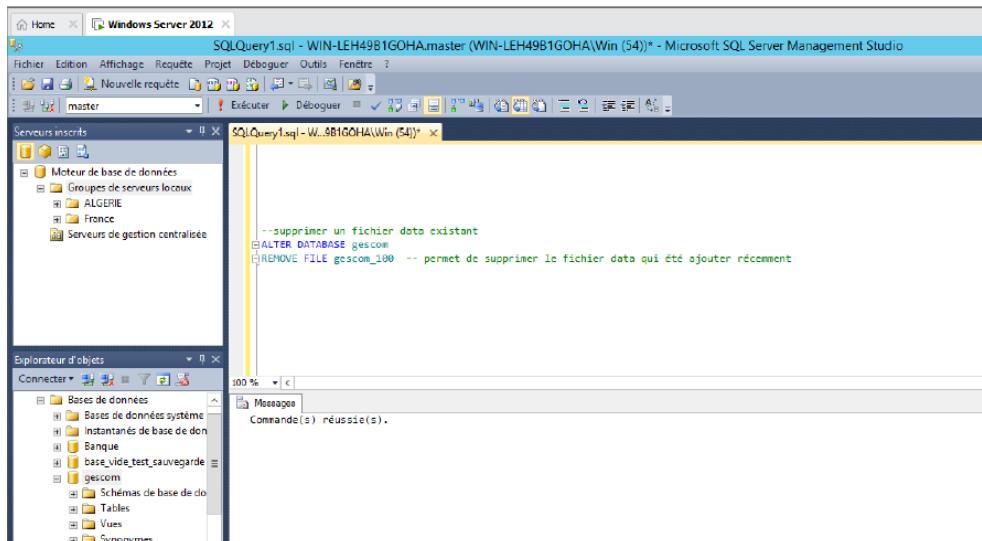


Figure 6.9: supprimer un fichier data

c'est à l'administrateur de mettre en place **un plan de maintenance** planifié avec **une tache de compactage**.

La commande de compactage qui va pouvoir réaliser cette tache est le **DBCC SHRINKDATABASE**, cette commande dispose de deux options à bien distinguer leur roles :

- 1. NOTRUNCATE** : Cette action compacte les données dans le fichier en déplaçant les pages affectés de la fin du fichier vers les pages non affectés du début du fichier. l'espace libre à la fin du fichier ne sera pas restitué au système d'exploitation et **la taille du fichier ne changera pas**

2. **TRUNCATEONLY** : cette action Libère tout l'espace libre à la fin du fichier pour le système d'exploitation. Le fichier de données est réduit seulement jusqu'à la dernière extension affectée. **Ignore target\_percent** s'il est spécifié avec TRUNCATEONLY

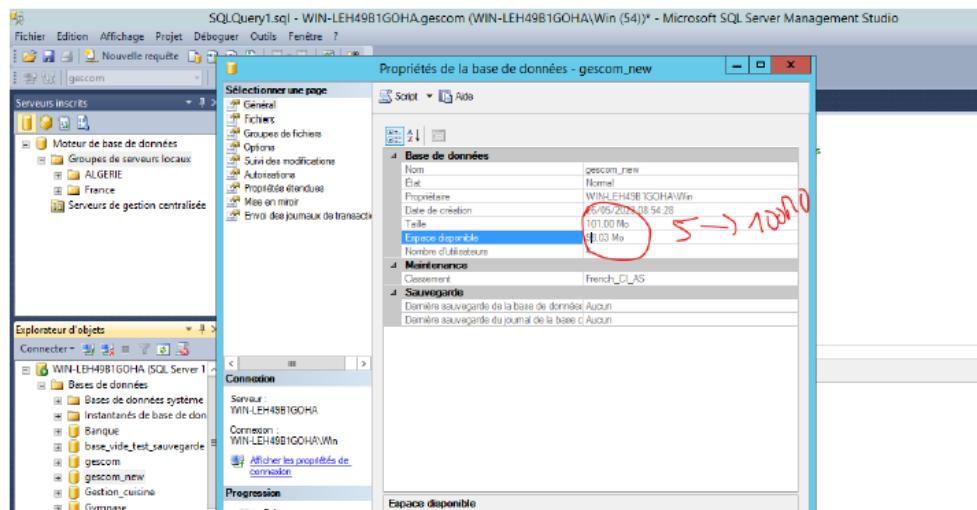
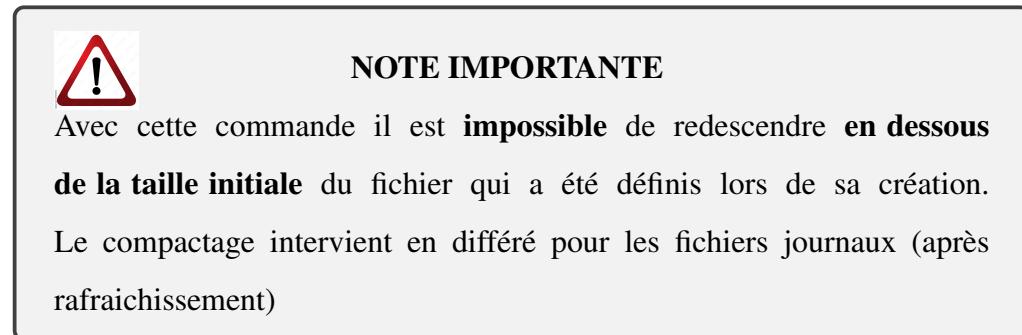


Figure 6.10: Cas d'utilisation de la commande SHRINKDATABASE

Afin de vous montrer le cas d'utilisation de la commande **SHRINKDATABASE**, nous avons **intentionnellement augmenter la taille du fichier data** de la base de données gescom\_new à **100Mo**, le but est d'utiliser cette commande pour **réduire la taille de la base de données** à sa taille initiale (la taille lors de la création du fichier), c'est-à-dire libérer de l'espace de stockage inutilisé.

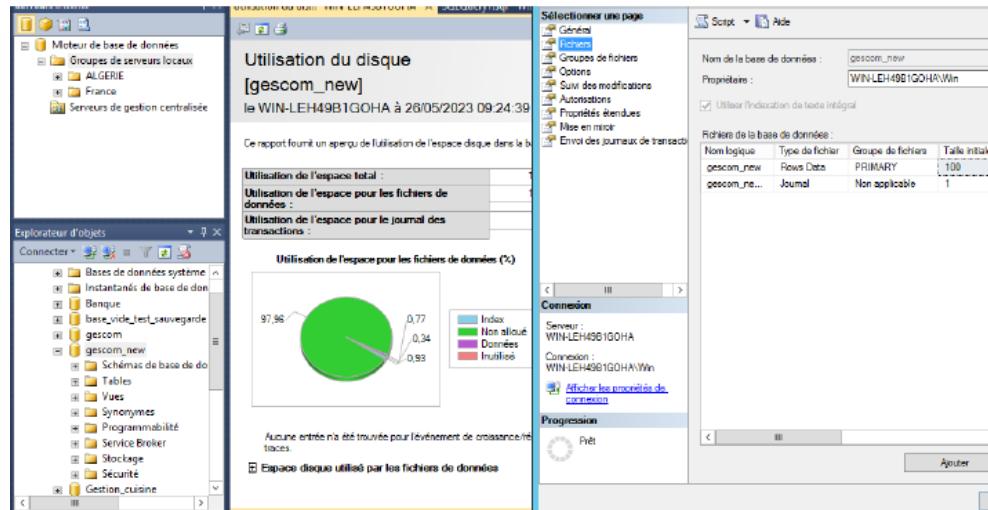


Figure 6.11: affichage du rapport d'utilisation du disque

#### 6.4.5 Rapport d'utilisation du disque

Comme vous pouvez le remarquer dans **le rapport de l'espace de stockage** du fichier data qu'on a augmenté, il existe maintenant **97.96%** d'espace occupé et inutilisé par la base de données, on souhaite avec le **SHRINKDATABASE** libérer cette espace pour le rendre au système d'exploitation avec **le paramètre TRUNCATEONLY**, ce dernier et j'insiste sur cela **libère tous les extensions allouées à la fin du fichier au système d'exploitation**



#### NOTE IMPORTANTE

La libération de l'espace inutilisé par les fichiers data rentre dans **le cadre d'un plan de maintenance**, qu'un DBA doit obligatoirement le mettre en place.

### 6.5 Les groupes de fichiers

#### 6.5.1 Structure logique

- Permet de **regrouper des fichiers de données** et de les gérer comme des **unités logiques** (elle est connue sous le nom de tablespace sous des SGBDR)

Oracle)

- Ils sont utilisés pour stocker les tables, les index...et indirectement ces groupes de fichiers vont être lier à des fichiers physiques.
- Un groupe de fichier ni plus ni moins qu'une **unité logique** auxquelles vont être rattachés les **fichiers physiques .mdf .ndf**

### 6.5.2 Les types de groupe de fichiers

1. **PRIMARY** : représente l'unité de stockage logique par défaut de SQL server à chaque fois que vous créer une nouvelle base de données
2. **Défini par l'utilisateur** : l'administrateur est en mesure de **créer un groupe de fichier de stockage logique personnalisé** puis l'associer à un ou plusieurs fichiers de données

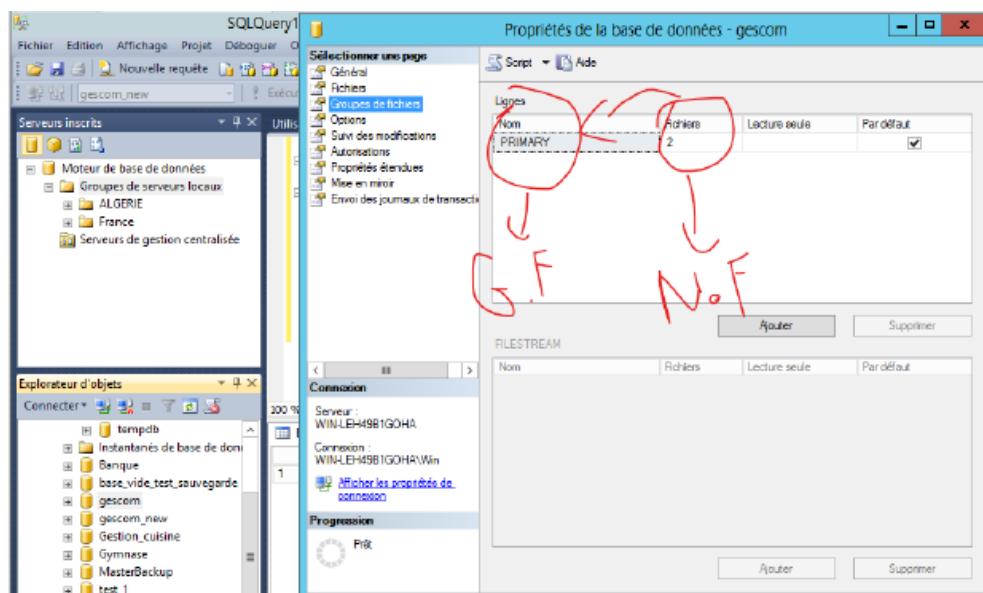


Figure 6.12: Le groupe de fichier PRIMARY

Dans la figure 6.12, on est en mesure d'identifier l'unité logique (groupe de fichier) auxquelles sont rattaché les fichiers de données. **L'unité logique PRIMARY** est créé par défaut par le SQL server à chaque fois que vous créez une base de données.

### 6.5.3 Principe d'utilisation des groupes de fichiers

Les groupes de fichiers sont principalement utilisés pour deux raisons à bien retenir :

1. **Répartir les données par types** (Tables, Index) pour une meilleur organisation logique.
2. **Répartir la charge des I/O** au niveau du disque. Dans le cas où on possède deux disques, on crée un groupe de fichier sur le premier disque et un autre groupe sur le second.



#### NOTE IMPORTANTE

La différence entre **le stockage physique et logique** réside dans **le niveau d'abstraction**. Le stockage logique se concentre sur la **structure conceptuelle** et la manière dont les données sont organisées, tandis que le stockage physique se concentre sur **l'implémentation réelle des données** dans le disque sous forme de fichiers.mdf .ndf

## 6.6 Le partitionnement

### 6.6.1 Objectif

- **Diviser** des tables volumineuses en plusieurs tables
- **Optimiser le stockage**
- **Montée en charge**



#### NOTE IMPORTANTE

Quand vous êtes amenée à travailler sur de grands entrepôts de données, le partitionnement va servir principalement à diviser les données d'une table volumineuse sur plusieurs tables physique pour pourvoir :

1. **Physiquement montée en charge**
2. **Logiquement** permettre l'accès à certaines tables **en lecture**, d'autre **en écriture** par exemple
3. **Organisation au niveau sécurité** adapté à un besoin spécifique

### 6.6.2 Le mécanisme de partitionnement

Pour pouvoir implémenter la notion de partitionnement sur une table volumineuse. Veuillez suivre chronologiquement les étapes suivantes :

- **Etape 1 : Créez un groupe de fichiers ou des groupes de fichiers** et des fichiers de données correspondants qui contiennent les partitions spécifiées par le schéma de partition
- **Etape 2 : Créez une fonction de partition** qui mappe les lignes d'une table ou d'un index dans des partitions en fonction des valeurs d'une colonne spécifiée (**clé de partitionnement**)
- **Etape 3 : Créez un schéma de partition** qui mappe les partitions d'une table ou d'un index partitionné à un groupe de fichiers ou à plusieurs groupes de fichiers
- **Etape 4 : Créez ou modifiez une table ou un index** et **spécifiez le schéma de partition** comme emplacement de stockage, ainsi que **la colonne qui servira de colonne de partitionnement**.



### NOTE IMPORTANTE

Il vaut mieux qu'un index ai **le même schéma de partitionnement** que la table sur laquelle est positionné

## La fonction de partitionnement

- Il faut tout d'abord avant de définir la fonction, définir la clé de partitionnement (le champ de répartition doit être pertinent : pertinent c'est à dire qu'il doit être capable de répartir les données de manière à équilibrer les nombres d'enregistrement par table)
- Permet de **répartir les données entre les différentes partitions**
- Utilise **des plages de valeurs bornées** : pour dire quelles valeurs vont être stockés dans une telle ou une telle partition
- Sert à définir les plages de valeur de chaque partition
- **Pour définir une fonction de partition en mode T-SQL**, exécuter les instructions suivantes :

```

USE gescom
GO

CREATE PARTITION FUNCTION partition_date (date) -- créer une fonction de partitionnement nommée "partition_date" sur un champ de type "date"
AS RANGE RIGHT FOR VALUES ('01/01/1996','01/01/1997','01/01/1998') -- on a défini trois borne de partition

/* AS RANGE sert à définir une partition en utilisant une plage de valeurs */

/* concrètement RANGE RIGHT signifie que les données inférieur à la première borne seront contenue dans la première partition,
les données supérieur à la première borne et inférieur à la deuxième borne seront stocké dans la deuxième partition et ainsi de suite */
    
```

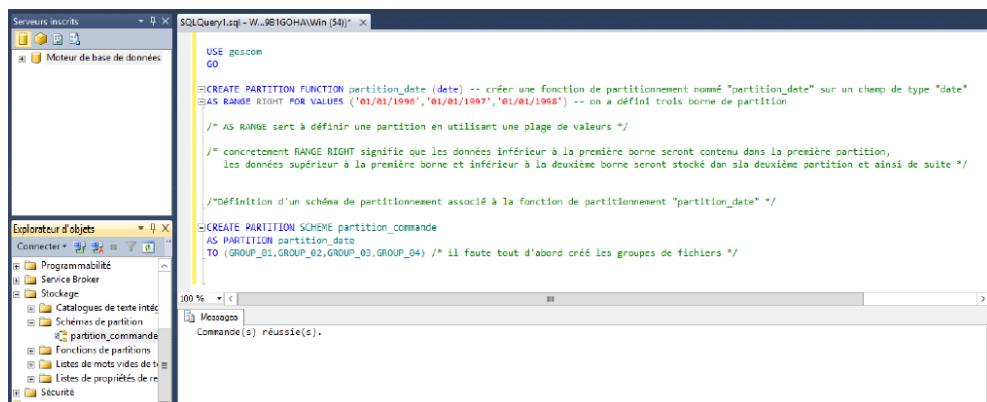
Figure 6.13: Crée une fonction de partition en mode T-SQL

L'argument de la fonction de partitionnement à bien retenir en tête est **RIGHT — LEFT** : Spécifie à quel côté de chaque intervalle de valeur limite, gauche ou droite, appartient les valeurs bornées.

- **RIGHT** : signifie que chaque valeur bornée appartienne à la partition suivante ou à la partition de droite
- **LEFT** : signifie que chaque valeur bornée appartienne à la partition de gauche ou à la partition courante

## Schéma de partitionnement

- Permet **d'affecter chaque partition à un groupe de fichiers**
- Possibilité de spécifier plus de groupe de fichiers que de partitions définis
- **Possibilité d'affecter toutes les partitions à un seul groupe de fichiers (non recommandé)** puisque l'objectif de pourvoir monter en charge
- Pour créer un schéma de partitionnement, exécuter les instructions suivantes :



The screenshot shows the SSMS interface with two windows open. The top window is titled 'SQLQuery1.sql - W...\961GOHA\Win (54)' and contains the following T-SQL code:

```

USE gescom
GO

CREATE PARTITION FUNCTION partition_date (date) -- créer une fonction de partitionnement nommé "partition_date" sur un champ de type "date"
AS RANGE RIGHT FOR VALUES ('01/01/1996','01/01/1997','01/01/1998') -- on a défini trois borne de partition

/* AS RANGE permet de définir une partition en utilisant une plage de valeurs */

/* concrètement RANGE RIGHT signifie que les données inférieur à la première borne seront contenu dans la première partition,
les données supérieur à la première borne et inférieur à la deuxième borne seront stocké dans la deuxième partition et ainsi de suite */

/*Définition d'un schéma de partitionnement associé à la fonction de partitionnement "partition_date" */

CREATE PARTITION SCHEME partition_commande
AS PARTITION partition_date
TO (GROUP_01,GROUP_02,GROUP_03,GROUP_04) /* il faut tout d'abord créer les groupes de fichiers */

```

The bottom window is titled 'Explorateur d'objets' and shows the database structure. The 'Schémas de partition' node under 'Stockage' has a child node 'partition\_commande'. The status bar at the bottom of the interface indicates 'Messages: Commande(s) réussie(s)'.

**Figure 6.14: Créer un schéma de partitionnement en mode T-SQL**

Dans cette figure, nous avons procédé à **la création d'un schéma de repartitionnement sur la fonction de répartition** qui a été créé antérieurement, ce schéma de repartitionnement va nous permettre de rattacher chaque partition au groupe de fichier correspondant, bien entendu cela implique à priori la création des groupes de fichiers définit dans le schéma de répartition ainsi que les fichiers data qui vont contenir les données de

chaque partition. **La clause AS PARTITION** quant à elle va définir la fonction de répartition que le schéma de repartitionnement va utiliser.



#### **NOTE IMPORTANTE**

**L'ordre de groupe de fichier est important** car l'ordre de valeurs de la fonction de repartitionnement va être associer à l'ordre des groupes de fichier qui vont être définis dans le schéma de partitionnement.

## Créer une table partitionnée

```

CREATE PARTITION SCHEME partition_commande
AS PARTITION partition_date
TO (GROUP_01, GROUP_02, GROUP_03, GROUP_04) /* il faut tout d'abord créer les groupes de fichiers */

/* Utilisation du schéma de partitionnement sur la table commande */

CREATE TABLE COMMANDES (
    [NO_COMMUNE] [numeric](6, 0) NOT NULL,
    [CODE_CLIENT] [char](5) NOT NULL,
    [NO_EMPLOYE] [numeric](6, 0) NOT NULL,
    [DATE_COMMANDE] [datetime] NOT NULL,
    [DATE_ENVOI] [datetime] NULL,
    [PONTE] [numeric](8, 2) NULL,
)
    ) ON partition_commande (DATE_COMMANDE)

Msg 102, Level 15, State 1, Line 1
    Comme(s) réussi(s).

```

Figure 6.15: Utilisation du schéma de partitionnement sur table

Cette figure nous montre **la dernière étape du mécanisme de partitionnement** qui consiste à **appliquer le schéma de partitionnement sur l'objet** qui possède la colonne qui va servir **de clé de partitionnement**.

Il est totalement possible **de gérer le partitionnement à travers l'interface SSMS** en cliquant sur **Propriété de la table > stockage > gérer le partitionnement**

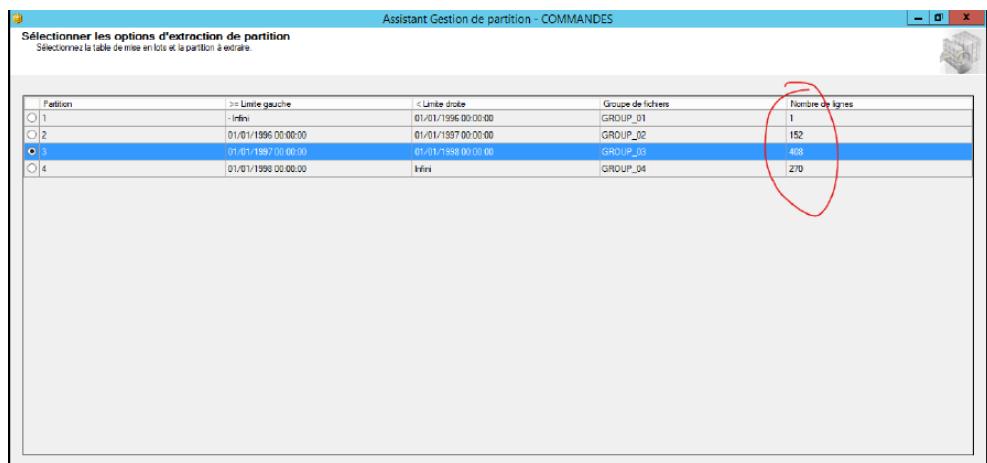


Figure 6.16: l'assistant de gestion de partitionnement

à l'aide de l'**assistant de gestion de partition** intégré à SQL server, nous avons pu diviser notre table "commande" avec les critères définis dans la fonction de partitionnement. On peut distinguer facilement la plage de données comprise dans chaque intervalle de partition avec le nombre d'enregistrements correspondant.

### Création d'un Index partitionné

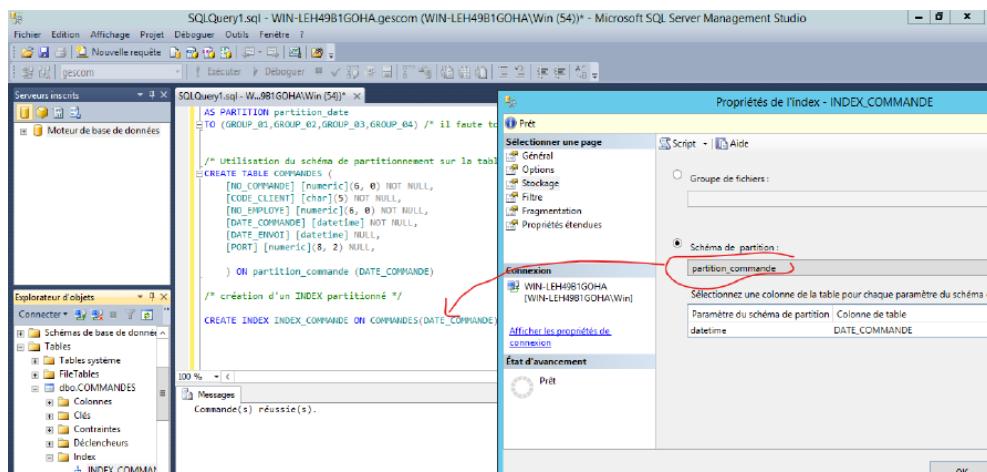


Figure 6.17: Création d'un Index Partitionné

Nous constatons qu'après avoir **créé un index sur la table partitionnée, le SQL server par défaut et automatiquement crée un index partitionné avec le même schéma de partitionnement** qui a servie à diviser notre table (commande) mais rien n'empêche de créer un schéma de partitionnement propre à l'index

## 6.7 Configuration d'une base de données SQL server

### 6.7.1 Objectif

- Apprendre à configurer les options d'une base de données en mode T-SQL et en mode Interface graphique

### 6.7.2 Les options de configuration de la base de données

Les options de configurations les plus couramment utilisés dans le cadre de l'administration d'une instance sont :

- **AUTO\_SHRINK(ON—OFF)** : si cette option est activée, les fichiers qui vont atteindre un seuil de 25% d'espace libre va réduire la taille de fichier données, on peut modifier cette option à travers  
**le paramètre (isAutoShrink)**
- **READ\_ONLY, READ\_WRITE** : la première option permet de rendre la base de données en lecture seul, cela veut dire que seul les requêtes SELECT sont possibles sur l'ensembles des objets de la base de données qui a été configuré en lecture, l'option READ\_WRITE est l'option par défaut de sql server qui signifie que la base de données est en mode lecture et écriture, on peut modifier ces options à travers  
**le paramètre (Updateability)**
- **SINGLE\_USER, RESTRICTED\_USER, MULTI\_USER** : la première option signifie qu'un seul utilisateur peut se connecter à l'instance, **RESTRICTED\_USER** signifie que seul les DB owner ou les propriétaires de la base et ces admins ou les utilisateurs qui ont les privilèges de DB owner et DB créateur peuvent se connecter au système. La dernière options **MUTLI\_USER** qui est la valeur par défaut, elle permet à tous les utilisateurs de se connecter à une base de données de sql server. Ces options peuvent être modifiées à travers le paramètre (**UserAccess**)
- **AUTO\_CREATE\_STATISTICS** : c'est une option qui permet d'activer l'**AUTO\_CALCULE** des statistiques, si cette option est positionnée à ON donc les statistiques lors de l'optimisation de la requête seront calculés de façon automatique. Le paramètre qui permet de modifier cette option est **IsAutoCreateStatestic**

- **AUTO\_UPDATE\_STATISTICS** : c'est le paramètre (**IsAutoUpdateStatistics**) qui va modifier cette option. Si elle est positionnée à ON les statistiques obsolètes (existante) vont être recalculer automatiquement.
- **AUTO\_CLOSE** : cette option si elle est positionnée à ON permet d'arrêter un server lorsque la dernière session est fermée, le paramètre qui contrôle la modification de cette option est (**IsAutoClose**). son activation est très gourmande en terme de ressource provoquant une dégradation des performances.
- **RECOVERY** : le mode précise la manière de **logger les transactions dans les fichiers journaux**
- **STATUS** : est permet de connaitre l'état du server et de la base
- **Collation** : permet de connaitre les règles de comparaison et de trie des chaines de caractère dans une base de données



#### NOTE IMPORTANTE

Les statistiques aident **l'optimiseur de requête à générer des plans d'exécution efficace** en fournissant la distribution des données dans une table ou une colonne spécifique.

Pour afficher l'état actuel des options de configuration, il existe divers manière:

- Utilisation de la fonction **DATABASEPROPERTYEX()**
- Utilisation de la procédure stocké **sp\_configure**
- Utilisation de la vue système **sys.database**

Cette fonction système décrite dans **la Figure 6.18** va nous permettre de vérifier avec **le paramètre IsAutoShrink** si dans la base de données qui a été introduit comme premier paramètre, la réduction automatique de la taille des fichiers

The screenshot shows the SQL Server Management Studio interface. In the center pane, a query window displays the following T-SQL code:

```

-- première méthode pour afficher les options de configuration de la base de données--
-- Utilisation de la fonction DATABASEPROPERTYEX

SELECT DATABASEPROPERTYEX('gescom','isautoshrink'); -- on va vérifier si la réduction de la taille des fichiers est automatique ou pas

--2-- Utilisation des tables systèmes sys.databases

```

A red arrow points from the text "on va vérifier si la réduction de la taille des fichiers est automatique ou pas" to the result set in the bottom pane. The result set shows a single row with the value 0 under the column "isautoshrink". A red circle highlights the value 0.

Figure 6.18: Utilisation de la fonction DATABASEPROPERTYEX

data est activé ou non, nous constatons que la requête nous a renvoyé une valeur nulle indiquant qu'AUTO\_SHRINK est désactivé.

- Utilisation de la vue système sys.database

The screenshot shows the SQL Server Management Studio interface. In the center pane, a query window displays the following T-SQL code:

```

-- les méthodes pour afficher les options de configuration de la base de données--
--1-- Utilisation de la fonction DATABASEPROPERTYEX

--SELECT DATABASEPROPERTYEX('gescom','updateability'); -- on va vérifier si la réduction de la taille des fichiers data est automatique ou pas
--SELECT DATABASEPROPERTYEX('gescom','useraccess') -- afficher les utilisateurs qui ont droit d'accéder à la base
--SELECT DATABASEPROPERTYEX ('gescom','isautoclose') -- afficher si le serveur s'arrête automatiquement lorsque une session est fermée

--2-- Utilisation des tables systèmes sys.databases
SELECT collation_name,user_access_desc,is_auto_close_on,is_auto_shrink_on,is_auto_update_stats_on
FROM sys.databases
WHERE name = 'gescom'

--3-- Utilisation des procédures stockées

```

A red box highlights the section starting with "--2-- Utilisation des tables systèmes sys.databases". Below it, a red arrow points from the text "afficher les utilisateurs qui ont droit d'accéder à la base" to the result set in the bottom pane. The result set shows a single row with the following data:

collation_name	user_access_desc	is_auto_close_on	is_auto_shrink_on	is_auto_update_stats_on
French_CI_AS	MULTI_USER	0	0	1

Figure 6.19: Utilisation de la vue sys.database

Comme vous pouvez le constater dans le jeu de résultat, il y'a un certain de nombre d'information sur les paramètres de configuration de la base de données (gescom) comme par exemple sa collation qui est French\_CI\_AS, elle est ouverte en mode MULTI\_USER, le serveur sur lequel elle est hébergé ne va pas se déconnecté automatiquement lorsque une session est fermée, sa taille ne va pas être réduite automatiquement, les statistiques utilisés par l'optimiseur des

requêtes vont être générées automatiquement.

## 6.8 Les transactions

### 6.8.1 C'est quoi une transaction ?

Une transaction est un **ensemble atomique**, soit la totalité des instructions est validées, soit la totalité est annulées



#### NOTE IMPORTANTE

Les instructions **INSERT, UPDATE et DELETE** sont des instructions transactionnelles de modification de données avec un **COMMIT implicite**.

### 6.8.2 Description d'une transaction explicite

```
BEGIN TRANSACTION <nom_transaction>
Instruction 1
SAVE TRAN <nomPointArret>
Instruction 2
SAVE TRAN < nomPointArret >
Instruction n
ROLLBACK TRAN <nomTransaction |nomPointArret>
COMMIT TRAN <nomTransaction>
```

Trans  
atom

Figure 6.20: Transaction explicite

- Une transaction explicite commence toujours par l'instruction **BEGIN TRANSACTION** suivie du nom de la transaction
- Les instructions de 1 à n peuvent être dans la majorité des cas un **UPDATE, DELETE, INSERT ou un SELECT**

- Le **SAVE TRAN** représente un point d'annulation de la transaction, quand cela est définis, le moteur donne la possibilité à l'utilisateur **d'annuler toutes les instructions** qui arrivent **après le point d'arrêt**
- Une transaction explicite **se termine obligatoirement** par **un COMMIT ou un ROLLBACK**
- **Le ROLLBACK** peut etre définis de deux manières possibles, soit on **définit le nom de la transaction** ce qui provoquera **l'annulation totale de la transaction**, soit on définit **le nom du point d'arrêt** ce qui annulera la transaction après le point d'arrêt



#### NOTE IMPORTANTE

L'utilisation des flags le long d'une transaction évitera à l'utilisateur de recommencer la transaction depuis le début



#### NOTE IMPORTANTE

Sous SQL server on ne voit que les transactions qui ont été validé par les autres utilisateurs

### 6.8.3 Les verrous sous SQL server

En SQL, **un verrou** est **un mécanisme de sécurité**. il permet de **limité ou d'interdire l'accès** à un ou plusieurs éléments de base de données. ça peut etre une ligne d'enregistrement ou une table entière. l'utilité des verrous est simple **garantir l'ACIDité** de la base.

Il existe différent **type de verrous** pris en charge par SQL server :

- **RID** = verrou sur une seule ligne d'une table, identifiée par un identificateur de ligne RID.

- **KEY** = verrou dans un index qui protège une plage de clés dans les transactions sérialisables
- **PAG** = verrou sur une page de données ou un index
- **EXT** = verrou sur une extension
- **TAB** = verrou sur une table complète comprenant l'ensemble des index et des données.
- **DB** = verrou sur une base de données
- **FIL** = verrou sur un fichier de base de données
- **APP** = verrou sur une ressource spécifiée par l'application
- **MD** = un verrou sur des métadonnées ou information de catalogue
- **HBT** = Verrou sur un tas ou B-Tree (HoBT). Ces informations sont incomplètes dans SQL server
- **AU** verrou sur une unité d'allocation. Ces informations sont incomplètes dans SQL server



#### NOTE IMPORTANTE

Pour obtenir **des informations sur les verrous** utilisés par le moteur de base de données SQL server, consulter la vue système **sys.dm\_tran\_locks** ou la procédure stockée **sp\_locks**

La figure 6.21 nous montre l'intérêt d'utiliser la procédure stockée **sp\_lock** pour obtenir des informations sur les verrous positionnés sur les objets détenu par la session en cours d'exécution. Dans un premier temps, il va falloir identifier l'ID de la session en cours en interrogeant la vue système **sys.dm\_exec\_sessions** puis vous l'introduisez en paramètre d'entrée de la procédure stockée **sp\_lock**.

Parmis les données renvoyés par la requête exécutée, on trouve **l'identifiant de l'objet** et **le type de verrou** qui a été employé pour verrouiller l'objet en question. **Le numéro d'identification de l'objet (ObjId)** sur lequel le verrou est maintenu va servir à identifier l'objet( Table, index, champs, un tuple ...) dans la base de données.

```

exec sp_lock @spid=55 -- identification des verrous positionnée sur les objets des base de données ouverte par la session 55
SELECT DB_ID() AS id_database
SELECT DB_NAME() AS name_database
SELECT OBJECT_NAME(1467152272,1) AS name_object_verrouillé

SELECT *
FROM master.dbo.spt_values

/*********je souhaite récupérer les informations sur les verrous positionné sur les objets des base de données ouvertes par la session 55
exec sp_lock @spid=55

*****poser un verrouille sur une ligne de la table catégorie appartenant que schéma stagiaire*****/
USE gescom
    
```

spid	dbid	ObjId	IndId	Type	Resource	Mode	Status
1	55	0	0	DB		S	GRANT
2	55	1	1467152272	TAB		IS	GRANT
3	55	32767	-571204656	TAB		Sch-S	GRANT

Figure 6.21: Cas pratique d'utilisation de la procédure stockée sp\_locks

#### 6.8.4 Cas pratique d'utilisation d'une transaction explicite

```

SELECT * FROM voiture
/* sous sql server 2012 on ne voit que les transactions qui ont été validé par les autres utilisateurs */

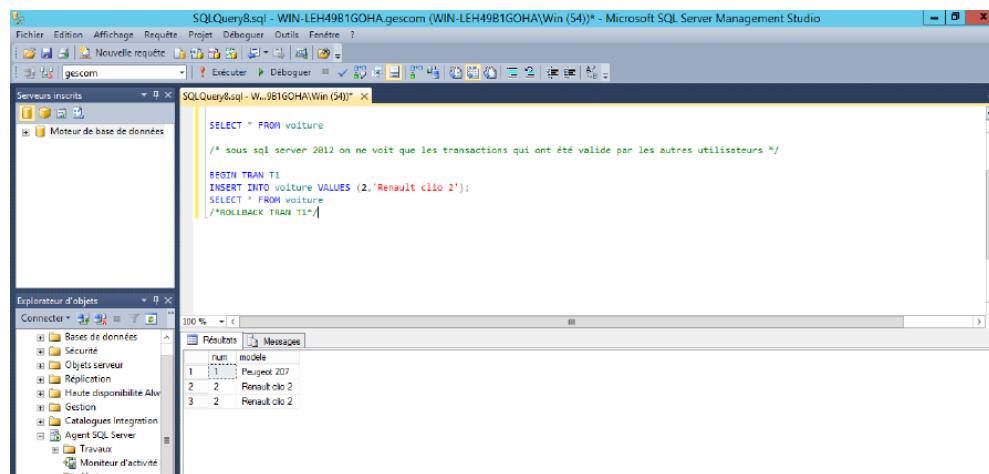
BEGIN TRAN T1
INSERT INTO voiture VALUES (2,'Renault clio 2');
SELECT * FROM voiture
/*ROLLBACK TRAN T2*/
    
```

num	marque
1	Peugeot 207
2	Renault clio 2
3	Renault clio 2

Figure 6.22: Transaction explicite d'insertion de données

Nous avons défini dans ce code une transaction explicite simple qui va servir à insérer des données dans la table voiture , un point important à bien retenir en tête que **tant que vous avez pas défini un COMMIT ou un ROOLBACK à la**

fin de la transaction, les données qui sont affiché dans le jeu de résultat sont en cours d'exécution autrement dit, les données n'ont pas encore été définitivement validées dans la base de données(**les données ne sont pas encore persistant**), cela vous permettra de revenir en arrière en cas de mauvaise manipulation comme dans notre cas on a inséré des doublons dans notre table, il suffit de faire un ROLLBACK pour pouvoir annuler les modifications.



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, there is a query window titled "SQLQuery8.sql - WIN-LEH49B1GOHA.gescom (WIN-LEH49B1GOHA\Win (54))". The query code is:

```

SELECT * FROM voiture
/* sous sql server 2012 on ne voit que les transactions qui ont été valide par les autres utilisateurs */

BEGIN TRAN T1
INSERT INTO voiture VALUES (2,'Renault clio 2');
SELECT * FROM voiture
/*ROLLBACK TRAN T1*/

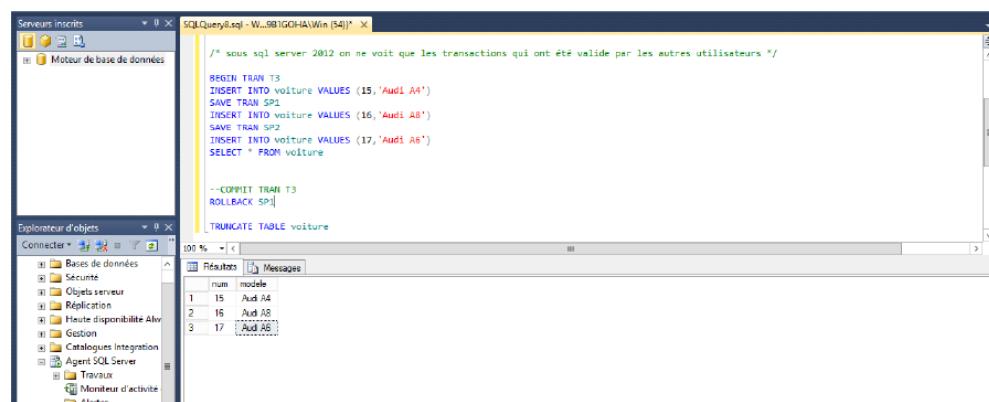
```

In the bottom right pane, the "Résultats" tab displays the following data:

	num	modele
1	1	Peugeot 207
2	2	Renault clio 2
3	2	Renault clio 2

Figure 6.23: Transaction explicite avec insertion validé

Dans cette transaction, nous avons bien précisé à la fin l'instruction **COMMIT** pour rendre **les données persistantes et visible par les utilisateurs**



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center pane, there is a query window titled "SQLQuery8.sql - WIN-LEH49B1GOHA\Win (54)". The query code is:

```

/* sous sql server 2012 on ne voit que les transactions qui ont été valide par les autres utilisateurs */

BEGIN TRAN T3
INSERT INTO voiture VALUES (15,'Audi A4')
SAVE TRAN SP1
INSERT INTO voiture VALUES (16,'Audi A8')
SAVE TRAN SP2
INSERT INTO voiture VALUES (17,'Audi A6')
SELECT * FROM voiture

--COMMIT TRAN T3
--ROLLBACK SP2
TRUNCATE TABLE voiture

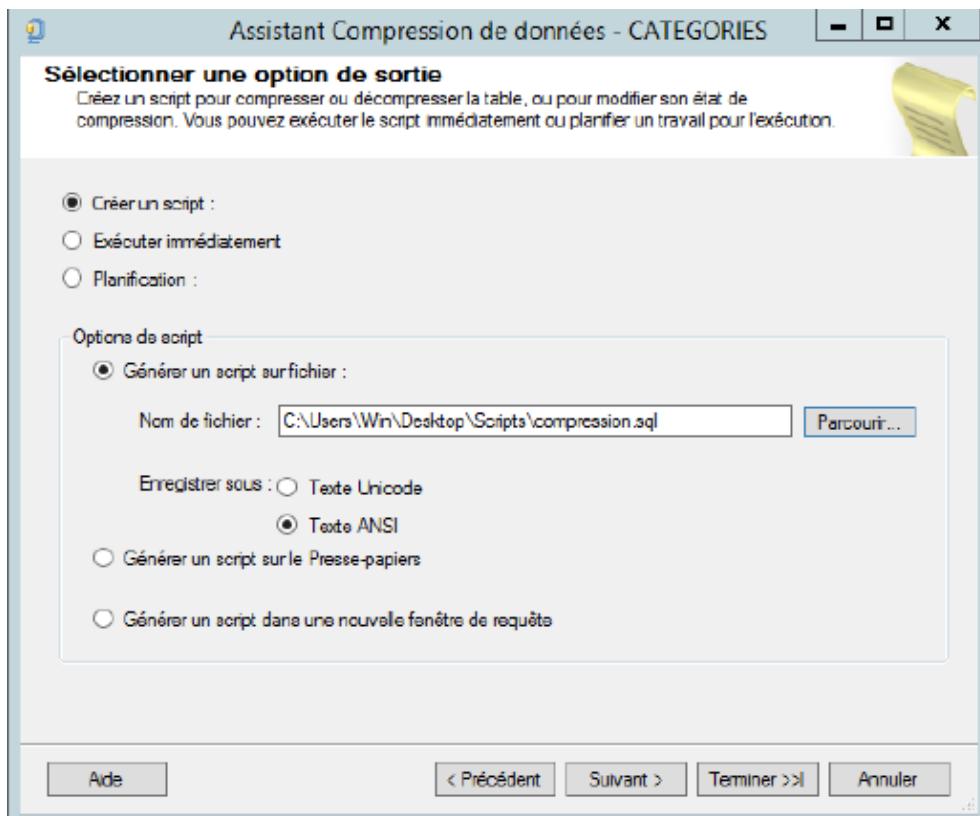
```

In the bottom right pane, the "Résultats" tab displays the following data:

	num	modele
1	15	Aud A4
2	16	Aud A8
3	17	Aud A6

Figure 6.24: Transaction explicite avec des points d'arrêt

Dans ce cas particulier de transaction où l'utilisateur va positionner des points d'arrêts le long d'une transaction, les instructions peuvent être rollbacker à par-



**Figure 6.25:** Transaction avec des points d’arrêt

tir de ce flag. Dans notre exemple typique, nous avons voulu annuler le dernier enregistrement qui apparait, cela peut être réaliser facilement en précisant le nom du flag qui précède les ou l'instruction qu'on souhaite annulée.

On constate bien que **l’ensemble des transactions qui viennent après le point d’arrêts SP2** ont bien été annulé, les autres instructions sont prêtes pour être Committer de manière persistante



#### NOTE IMPORTANTE

Pour pouvoir travailler sur un serveur en production, il est recommandé de commencer toujours par un **BEGIN TRAN puis ROLLBACKER ou COMMITTER** en fonction des vérifications que vous allez réaliser. Ceci est très pratique dans le cas où par exemple **vous avez procédez à une mise à jour sans spécifier de clause WHERE ça vous évitera d'impacter l'ensemble de vos enregistrement**

### 6.8.5 Les niveaux d'isolement sous SQL server

- **READ COMMITED** : Spécifie que les instructions ne peuvent pas lire les données modifiées et non validées par d'autres transactions.

Le comportement de READ COMMITED dépend de la valeur affectée à l'option de base de données **READ\_COMMITED\_SNAPSHOT** si :

1. **READ\_COMMITED\_SNAPSHOT a la valeur OFF** , le moteur SQL server utilise des verrous partagés pour empêcher d'autres transactions de modifier des lignes pendant que la transaction active exécute une opération de lecture.

- **REPEATABLE READ** : Les transactions en mode **REPEATABLE READ** utilisent des verrous de lecture sur les données qu'elles consultent. Ces verrous de lecture sont maintenus pendant toute la durée de la transaction, tant que la transaction en cours n'est pas validée, les autres transactions ne peuvent que lire de manière répétée l'ancienne valeur.

- **SERIALIZABLE** : représente le niveau d'isolation le plus élevé qui offre une sécurité maximale **au détriment des performances** en raison du nombre de verrous excessifs qui sont utilisés

## 6.9 Les modes de recuperation sous SQL server

### 6.10 Fonction des fichiers journaux

- Permettent de journaliser les transactions de type LMD
- Enregistre les valeurs avant et après modification
- Garantir la cohérence et la durabilité des données (COMMIT) : Lorsque un utilisateur COMMIT une transaction, elle est directement loggée dans le fichier journal de transaction (**Voir Mécanisme de journalisation**)

#### 6.10.1 Le mode simple(SIMPLE)

- Limite la journalisation de la majorité des transactions ( **Ecrit le minimum d'information**)
- **Le journal est tronqué après chaque CHECKPOINT**, cela signifie qu'après chaque point de contrôle qui se produit dans le système de gestion de base de données, les informations enregistrées sur le journal sont effacées.
- Ne permet pas **la restauration en mode PITR**
- Impossible de restaurer des pages de données de 8ko individuellement
- Rapide et **limite la taille des fichiers journaux** ( Le fichier est tronqué après chaque CHECKPOINT

#### 6.10.2 Le mode complet (FULL)

- C'est le mode qui **offre une sécurité maximale**, écrit le maximum d'information dans le fichier journal.
- Doit être utilisé sur **un serveur de production** afin de **récupérer l'ensemble des transactions** qui ont été validé sur le système **en cas de crash ou de défaillance**.

- Le fichier journal **n'est pas tronqué** après chaque CHECKPOINT et peut grossir rapidement voir **saturer votre espace de stockage**
- Permet une restauration PITR jusqu'à un point de défaillance
- Ce mode de journalisation **impacte les performances** en générant beaucoup d'IO sur le disque

### 6.10.3 Le mode journalisé en bloc (**BULK LOGGED**)

- Minimise la journalisation lors de l'exécution des opérations de masse tel que SELECT INTO et BULK INSERT.
- Ne permet pas une restauration PITR ou un point de défaillance.



#### NOTE IMPORTANTE

Le **RECOVERY** est une option de configuration de la base de données que vous allez paramétrier pour sélectionner le mode de récupération souhaité (**FULL, SIMPLE, BULK LOGGED** )

## 6.11 La compression des données

Les opérations de compression de données se font généralement sur les objets applicatifs tel que :

1. Les tables volumineuses
2. Les index
3. Les partitions



### NOTE IMPORTANTE

La compression sert principalement à **gagner de l'espace disque**, en revanche elle **dégrade les performances de lecture /écriture** due à l'exécution de deux algorithmes, l'un qui va compresser les données et l'autre lors de l'extraction pour décrypter les données.

#### 6.11.1 Gérer la compression de données



### NOTE IMPORTANTE

Un DBA PROD **combine généralement la compression avec le partitionnement**. Chaque partition peut être compressé indépendamment des autres, ce qui particulièrement utile lorsque plusieurs partitions ont des caractéristiques de données distinctes

Pour gérer la compression des données via l'assistant de compression des données, il va falloir **suivre les étapes décrites dans les figures suivantes** :

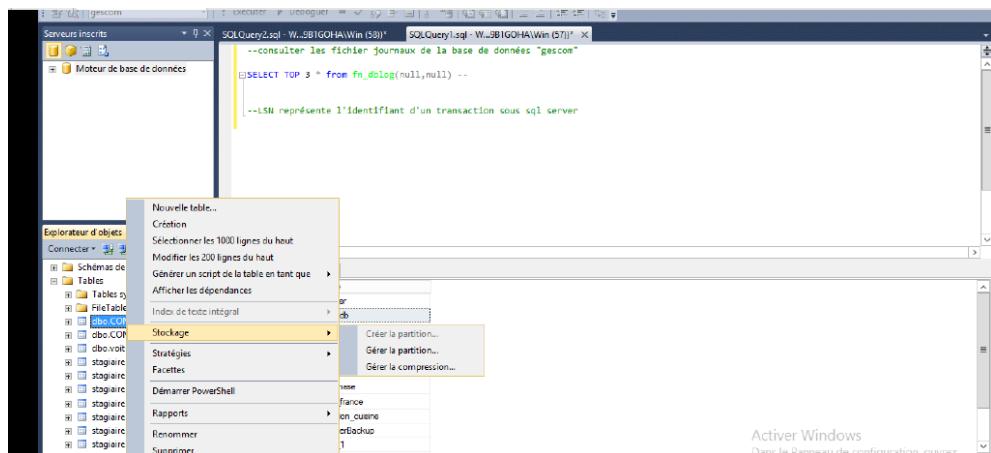


Figure 6.26: Etape 1 : Lancer l'assistant de gestion de compression

Comme **la figure 6.26** le montre, faites un clic droit sur l'objet dont vous souhaitez compresser, dans notre cas d'étude, nous avons sélectionné un objet de type table appartenant à la base de données "gescom". une nouvelle

fenêtre va s'ouvrir pour **choisir le type de compression** illustrée dans la figure ci-dessous

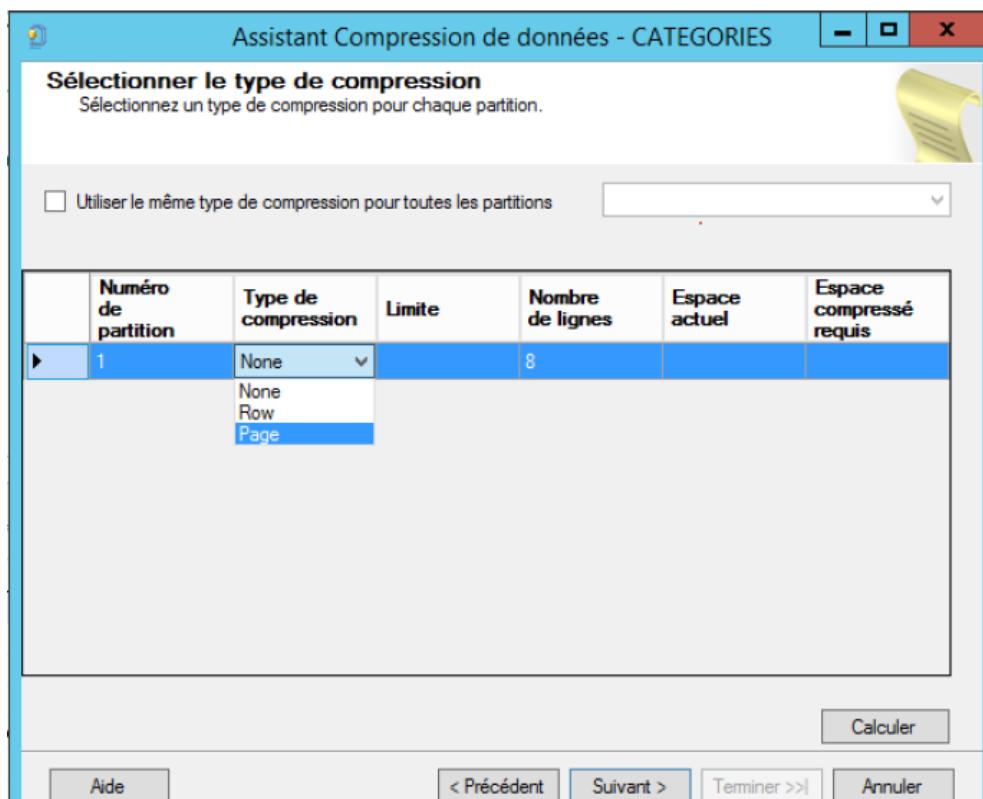


Figure 6.27: Etape 2 : Choisir le type de compression

Dans cette étape, il faut **choisir le type de compression** de la table, soit par page soit par enregistrement. Vous pouvez éventuellement si votre table est partitionnée de réaliser une compression par partition. Le lancement du calcul va nous permettre d'estimer l'espace gagné sur le disque local en faisant la différence entre l'espace actuelle et l'espace compressé requi.

# Chapter 7

## Sauvegarde sous SQL server 2012

### 7.1 Introduction à la sauvegarde

#### 7.1.1 Pourquoi sauvegarder ?

- **Sinistre ou Panne matérielle** : si on a un problème au niveau du disque par exemple, nous pouvons restaurer nos données perdues.
- **Une mauvaise manipulation des utilisateurs** : si un utilisateur final effectue un DROP TABLE ou un DELETE d'un enregistrement, c'est à l'administrateur de la base de données de **restaurer les sauvegardes justes avant la mauvaise manipulation**
- **Une panne serveur** comme le déni de service, c'est à l'administrateur de remonter sur un autre serveur qui a été **configuré pour accepter la nouvelle instance**.
- **Déplacement de base de données** (clonnage de la base) : Le BACKUP/RESTORE pour pouvoir se déplacer d'un serveur à un autre est couramment utiliser

par les administrateurs de base de données, il existe bien entendu d'autre techniques de déplacement comme le **SELECT INTO, BCP, SSIS**

### 7.1.2 Les caractéristiques de la sauvegarde

- **Sauvegarde à chaud** : cela veut dire que l'utilisateur étant connecté, on ne perturbe pas le bon fonctionnement normal du serveur, en d'autre terme, l'utilisateur peut lancer la sauvegarde alors que l'instance est en plein fonctionnement.
- **Sauvegarde cohérente des données** : SQL server va réaliser un certain nombre d'opération qui vont garantir la cohérence des données, cela veut dire que les transactions qui ont été validés pendant le BACKUP vont etre restauré de manière cohérente.
- **Aucune opération de création ou de modification** de la base n'est possible pendant une opération de sauvegarde.
- **Impossible de créer des indexes** pendant la sauvegarde
- **Exécution d'opération non journalisée non autorisé** : il n'est pas possible de lancer des transactions qui ne vont pas etre écrites dans les fichiers journaux

## 7.2 La sauvegarde complète

**La sauvegarde complète** : la plus utilisé pour les bases de données **moyennes et petites volumétries**, elle est d'autant plus caractérisée par les éléments suivants :

1. C'est une sauvegarde où **l'ensemble des fichiers data et une partie des fichier journaux seront sauvegardés**, cela **permettra à l'utilisateur de restaurer la base sur le même ou sur un autre serveur en cas de crash**.

2. Sauvegarde les modifications **validées pendant la sauvegarde**.
3. Représente un **point de départ** pour **toute stratégie de sauvegarde**



#### NOTE IMPORTANTE

Qui dit stratégie de **sauvegarde complète** dit aussi **volumétrie d'espace disque occupé !!!!**



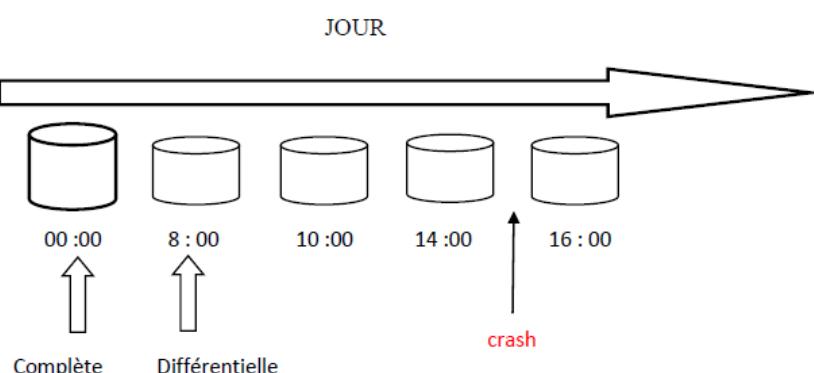
#### NOTE TRES IMPORTANTE

Il est recommandé de faire une **sauvegarde complète le Week-end** là où il n'a pas d'activité sur le serveur

### 7.3 La sauvegarde différentielle

**La sauvegarde différentielle** : est une méthode de sauvegarde qui va sauvegarder tous les extants de 64k modifiés depuis la dernière sauvegarde complète

1. Sauvegarde uniquement les parties de la base de données modifiés depuis la dernière sauvegarde complète.
2. Sauvegarde toutes les transactions intervenues pendant la sauvegarde différentielle



**Figure 7.1: Exemple de combinaison d'une stratégie de sauvegarde FULL/DIFF**



#### NOTE IMPORTANTE

Pour pouvoir mettre en place une stratégie de sauvegarde différentielle, le prérequis est qu'il faut qu'au préalable une sauvegarde complète a été réalisé.

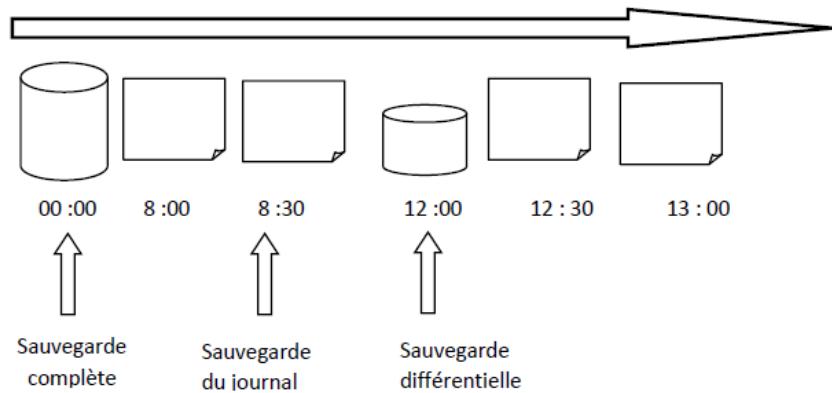
En cas de crash survenu entre 14h00 et 16h00, il suffit de restaurer la sauvegarde complète de minuit pour pouvoir ensuite appliquer la sauvegarde différentielle de 16h00 comme ça on recouvre l'ensemble des transactions qui ont été joués entre 00 :00 et 14h00. **La mise en place de la sauvegarde différentielle est essentiellement pour réduire le cout et la durée de la sauvegarde.**

**La combinaison entre la sauvegarde différentielle et la sauvegarde du journal des transaction est la plus optimale**

## 7.4 La sauvegarde du journal des transactions

**La Sauvegarde du journal des transactions** : consiste à sauvegarder uniquement la partie journal des transactions qui contient les transactions qui n'ont pas encore été sauvegardé. Pour pouvoir appliquer cette méthode de sauvegarde,**un certain nombre de prérequis** doivent être remplis :

1. **Sauvegarde complète INDISPENSABLE**
2. Nécessite le mode de récupération complet (**RECOVERY=FULL**)



**Figure 7.2: Combinaison des trois stratégies de sauvegarde sur SQL server**

## 7.5 Mise en oeuvre de la sauvegarde complète

### 7.5.1 Destination des sauvegardes

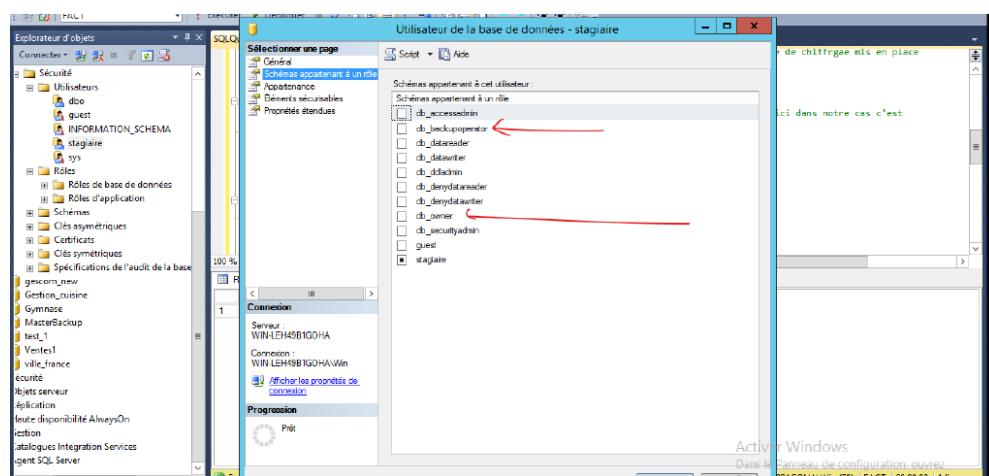
- **Disque** :
  1. **Les unités physiques** : les fichiers de sauvegarde vont etre rediriger vers le système de fichier du système de'exploitation.
  2. **Les unités logique** : s'appuient obligatoirement sur des fichiers physiques

- **Bande**

### 7.5.2 Les priviléges de sauvegarde

Pour que l'utilisateur de connexion puisse réaliser les opérations de sauvegarde sur in stancce de base de données, il devra obligatoirement etre au minimum **membre de l'un des roles** suivants :

1. **Sysadmin** ( rôle de serveur): Représente l'administrateur de l'instance, il possède tous les priviléges
2. **db\_owner** (rôle de base de données) : Représente le propriétaire de la base de données
3. **db\_backupoperator** (rôle de base de données) : les membres de ce rôle auront la possibilité de réaliser des sauvegardes



**Figure 7.3: Connaitre les rôles pour les opérations BACKUP**

Cette figure va nous servir à connaitre l'ensemble des rôles de la base de données gescom auquel le schéma stagiaire appartient, les flèches en rouges nous indique les rôles auxquelles il faut impérativement devenir membre pour avoir les priviléges nécessaires pour effectuer des BUCKUP. le chemin d'accès vers la liste des roles de base de données est le suivant :

**Base de données > sécurité > utilisateurs**

### 7.5.3 L'instruction BACKUP

- La syntaxe

```
|BACKUP DATABASE <nombase> TO <unite logique> | DISK =
<chemin_nom> WITH INIT|NOINIT|FORMAT|CHECKSUM|COMPRESSION
```

1. WITH INIT : pour pouvoir écraser une sauvegarde existante s'il y'en a une
2. NOINIT : nous permet de faire l'inverse de WITH INIT
3. FORMAT : permet de réinitialiser l'en-tête du volume de sauvegarde
4. CHECKSUM : permet de réaliser un contrôle sur les pages sauvegardés

**Figure 7.4: La syntaxe complète de sauvegarde sur SQL server**



#### NOTE IMPORTANTE

il est recommandé d'utiliser **l'option CHECKSUM** pour vérifier si votre sauvegarde est opérationnelle, autrement dit elle vérifie l'intégrité des données lors de la sauvegarde d'une base de données.

Le SQL server va calculer un checksum pour chaque page de données sauvegardés et stocke ce checksum dans le fichier de sauvegarde. Lors de la restauration, SQL server vérifie les checksums pour s'assurer que les données n'ont pas été corrompus

Pour mettre en place une sauvegarde complète d'une base de données applicative, veuillez bien suivre en respectant chronologiquement les étapes suivantes :

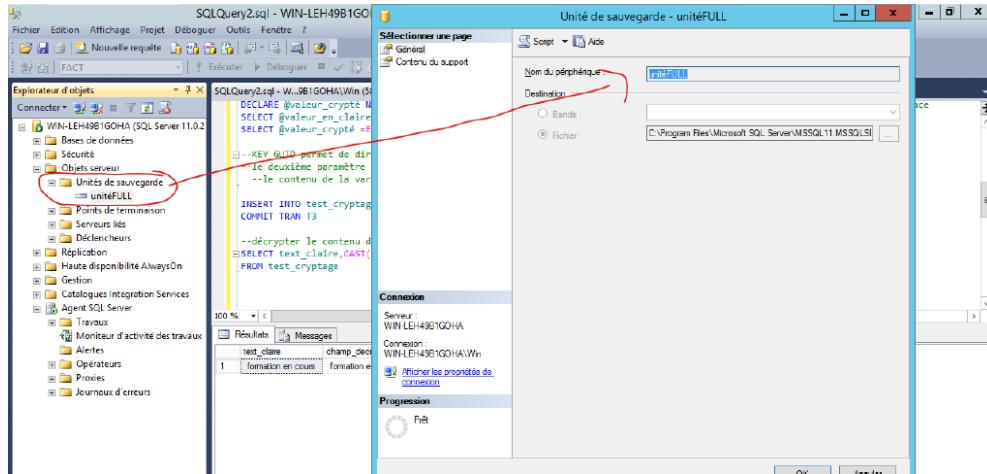


Figure 7.5: Etape 1 : Création d'une unité de sauvegarde logique

La création d'une unité de stockage logique va nous servir à rattacher physiquement le jeu de sauvegarde à un fichier au niveau du système d'exploitation. En d'autre terme cela va nous permettre de créer une arborescence physique vers le fichier .bak

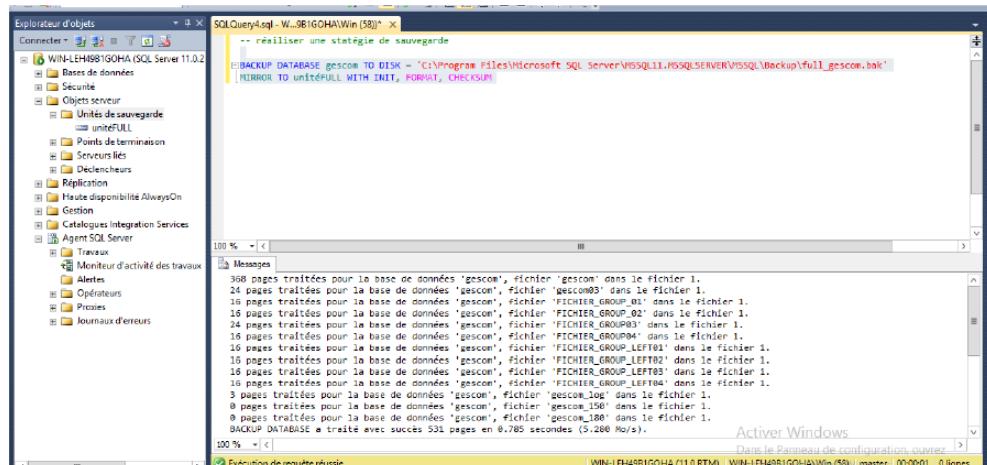
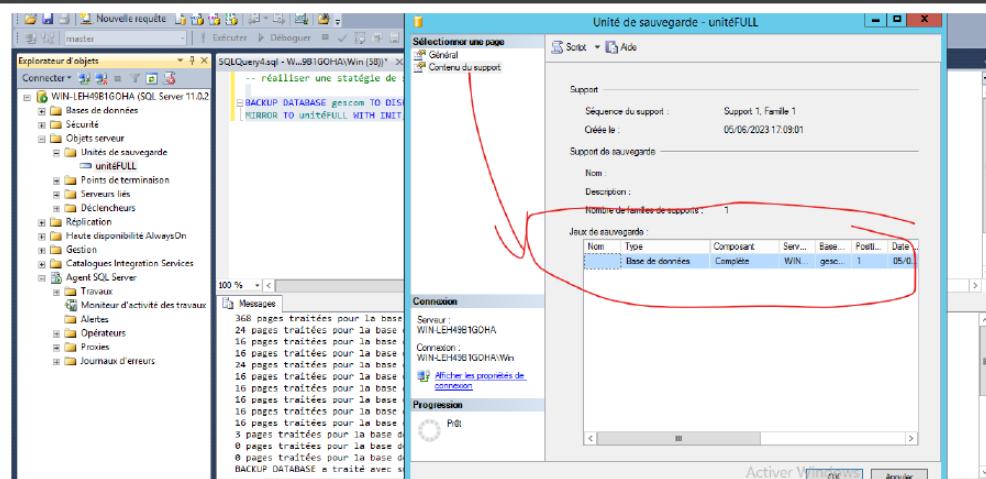


Figure 7.6: Etape 2 : Exécution de la sauvegarde complète en mode T-SQL

L'option **MIRROR** va servir à dupliquer les données d'une base de données sur l'unité logique qui vient d'être créé qui s'appelle unitéFULL

### NOTE IMPORTANTE

Il est plus que recommandé d'utiliser **le MIRROR TO** sur un serveur de production lors de la sauvegarde d'une base de données afin de pouvoir dupliquer les données sur plusieurs supports de stockage et assurer par la suite **l'intégrité et la disponibilité des données sauvegardés**



**Figure 7.7:** Etape 3 : vérification du contenu du support de sauvegarde

On remarque maintenant qu'il y'a un certain nombre d'informations qui sont disponibles dans le support après avoir réalisé le processus de sauvegarde. **La vérification du contenu du support vous confirmera du bon déroulement ou non de la sauvegarde.**

#### 7.5.4 Sauvegarde avec mise en miroir

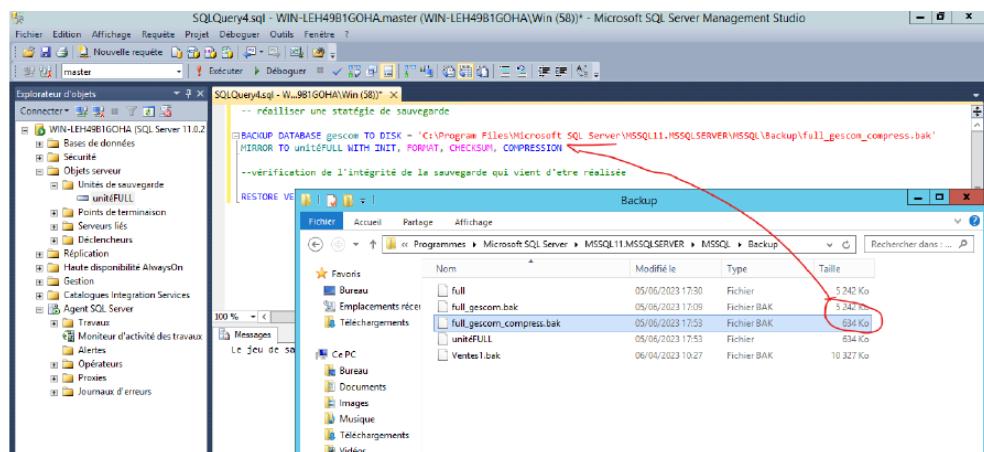
La mise en miroir consiste à diviser ou **duplicer les sauvegardes sur plusieurs supports, ça permettra d'être protégé d'une éventuelle panne au niveau support..**

La syntaxe de sauvegarde avec une mise en miroir est représenté ci-dessous :

BACKUP DATABASE <nombase> TO <unité logique> | DISK =  
 <chemin\_nom> MIRROR TO <unité logique> | DISK = <chemin\_nom>

**Figure 7.8: La sauvegarde avec une mise en miroir**

Autre option à ne pas négliger son utilisation est l’option de compression illustré dans la figure ci-dessous :



**Figure 7.9: La sauvegarde avec l’option de compression**

Cette figure nous montre l’importante d’adopter **l’option COMPRESSION** lors de la stratégie de sauvegarde afin de **gagner en volumétrie dans l’espace de stockage**

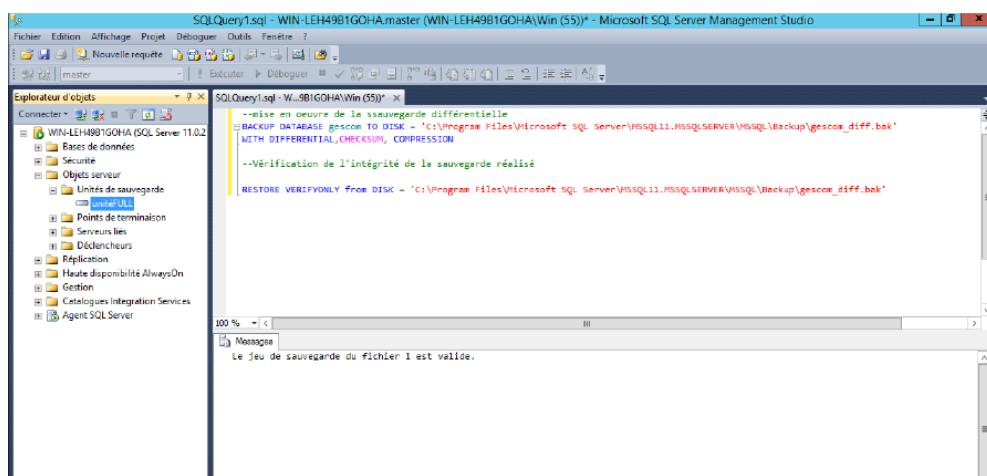
## 7.6 Mise en oeuvre de la sauvegarde différentielle

La sauvegarde différentielle est **un complément** de la sauvegarde complète, l’un ne va pas sans l’autre.

- Caractéristiques de la sauvegarde DIFF

1. **Sauvegarde toutes les extensions modifiées depuis la dernière sauvegarde complète** : le SQL server est capable d'identifier l'ensemble des fichiers qui ont été modifiés en consultant les extensions qui ont la byte =1
2. **Principe du fonctionnement de la sauvegarde différentielle** : pour chaque extension créée dans la base de données, **sql server va créer un bite qui est à la base égale à 0**, un fois que l'extant a été **modifié par un INSERT ou un UPDATE**, le sql server est capable de **pointer** sur **les extants** dont le bite à **basculer à 1** pour pouvoir les prendre en compte dans la sauvegarde.
3. **Permet de réduire le nombre de sauvegarde du journal des transactions** : tout les fichiers journaux qui ont été sauvegardés avant la sauvegarde différentielle ne seront plus utilisé pour pouvoir restaurer.
4. Utilisable **quelque soit le mode de récupération** choisie pour les transactions (**FULL, BULK LOGGED et SIMPLE**)

### 7.6.1 la commande BACKUP



The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery1.sql - WIN-LEH49B1GOHA.master (WIN-LEH49B1GOHA\Win (55)) - Microsoft SQL Server Management Studio". The main window displays the T-SQL command for a differential backup:

```

--alors en cours de la sauvegarde différentielle
BACKUP DATABASE gescom TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\Backup\gescom_diff.bak'
WITH DIFFERENTIAL, CHECKSUM, COMPRESSION
--Vérification de l'intégrité de la sauvegarde réalisé
RESTORE VERIFYONLY FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\Backup\gescom_diff.bak'

```

The status bar at the bottom of the interface shows the message: "Le jeu de sauvegarde du fichier 1 est valide."

Figure 7.10: La syntaxe de la sauvegarde différentielle

Cette figure nous montre l'**instruction BACKUP** qui permettra de réaliser une sauvegarde différentielle de la base de données gescom en lui associant les options de sauvegarde tel que **le CHECKSUM et COMPRESSION**, le fichier.back a été stocké dans un emplacement indiqué par le chemin physique **vers le disque**, il était possible également d'ajouter une unité logique dans le répertoire de l'instance **objet serveur** dans laquelle on stocke les données de sauvegarde différentielle.

Concrètement cette commande va réaliser une sauvegarde des extensions de 64k qui ont subis des modifications depuis la dernière sauvegarde complète.

#### NOTE IMPORTANTE

Il est fortement recommandé de privilégier l'utilisation d'un **support de sauvegarde logique** pour stocker tous les sauvegardes différentielles sous forme de liste classée par ordre croissant par date de début de sauvegarde

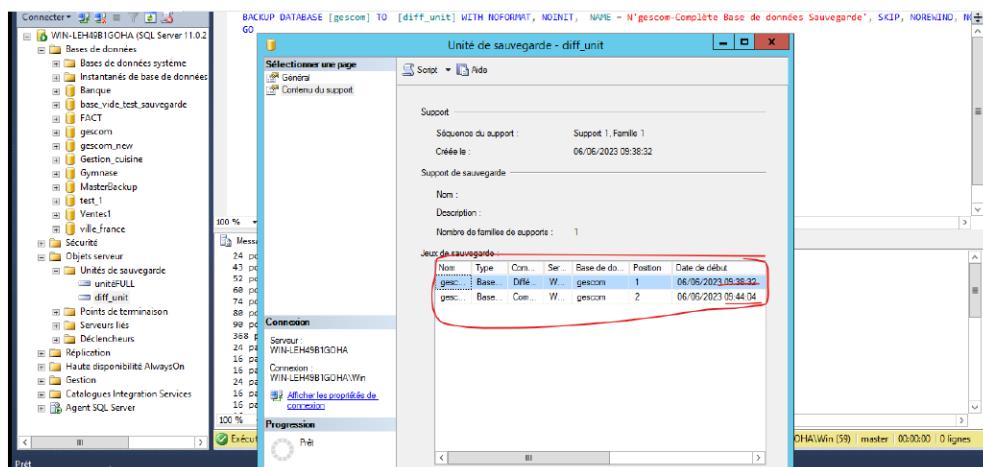


Figure 7.11: Sauvegarde différentielle sur un support de sauvegarde logique

### NOTE IMPORTANTE

**Une sauvegarde différentielle** nécessite au préalable une sauvegarde complète pour pouvoir restaurer les transactions faites depuis la dernière sauvegarde complète. La sauvegarde différentielle **sauvegarde uniquement les extensions de 64 ko qui ont subis de modifications (valeur bite =1)**. L'unité logique permette de centraliser tous vos sauvegardes dans un fichier physique sur lequel elle pointe.

## 7.7 Mise en oeuvre de la sauvegarde du journal des transactions

- Est complément de la sauvegarde complète et la sauvegarde différentielle, **la combinaison des trois permettra mettre en place une stratégie de sauvegarde optimale** adaptés aux **contrainte de production !!!**

### 7.7.1 Les caractéristiques

1. Utilisables uniquement si l'option de configuration de la base de données **RECOVERY** est configuré à **COMPLET ou JOURNALISE EN BLOC**.
2. Prérequis : autorisée uniquement après une sauvegarde complète
3. Principe de fonctionne de la sauvegarde du journal des transactions : **le sql server va remonter vers le dernier LSN** ( Log sequence Number ) qui représente l'identifiant des transactions aux niveau système, une fois qu'il est remonté, **il sauvegarde les transactions qui ont arrivé après le dernier LSN**
4. Sauvegarde toutes les transactions jusqu'à **la transaction actuelle ouverte**

5. Toutes les transactions sauvegardées peuvent alors être supprimés du journal des transactions

**NOTE IMPORTANTE**

Toutes les transactions qui ont étaient sauvegardés n'ont pas lieu d'être, c'est-à-dire que le fichier journal peut être tronqué pour libérer ainsi de l'espace de stockage dans le système d'exploitation

### 7.7.2 La commande BACKUP

`BACKUP LOG <nom_base> TO DISK = 'chemin' WITH INIT`

Figure 7.12: La syntaxe de la sauvegarde des fichiers journaux

### 7.7.3 Stratégie de sauvegarde sur un serveur de PROD

il est essentiel de savoir que la combinaison des trois méthodes de sauvegarde représente la stratégie BACKUP la plus optimal couramment utilisé dans les environnements RUN et BUILD

Voici un exemple basique d'une stratégie de sauvegarde en combinant les trois méthodes de sauvegardes FULL, DIFF et LOG :

**Figure 7.13:** Exemple d'une stratégie de sauvegarde sous SQL server

Dans cette figure nous avons réalisé **une stratégie de sauvegarde** qui consiste à **combiner les trois méthodes de sauvegarde** en une seule fois, rappelez-vous que pour pouvoir faire un BACKUP DIFF vous êtes dans l'obligation au préalable de faire un BACKUP FULL, dans ce cas toutes les transactions qui ont étaient jouées entre le dernier backupfull et le backup diff vont être sauvegardé dans le backup diff. La sauvegarde des journaux de transactions vont quant à eux servir à **augmenter et garantir la disponibilité des données** en cas de perte ou sinistre inattendu survenue entre le BACKUP FULL et le BACKUP DIFF. D'où **LA NECESSITE DE COMBINER LES TROIS METHODE POUR ASSURER LA DISPONIBILITE DE VOS DONNEES.**



## **NOTE IMPORTANTE**

Sur un serveur en production il est indispensable de combiner les trois méthodes BACKUP sur **les grosses bases de données**. La **DIFFERENTIAL** vous permet de **gagner du temps de restauration**, au lieu de jouer tous les sauvegardes des journaux des transactions.

## 7.8 Mise en oeuvre de la sauvegarde de groupes

### 7.8.1 Caractéristiques

- C'est une stratégie de sauvegarde **alternative** aux sauvegardes complètes : **très utile** dans le cas des bases de données de grosse volume **VLD (very large database)**.
- Cette méthode de sauvegarde est **nécessaire** lorsque la base de données **devient volumineuse** et que les **temps de sauvegarde** vont être **trop longues**.
- Les différentes méthodes de sauvegarde tel que : la **méthode de sauvegarde différentielle** et la **méthode de sauvegarde du journal des transactions** peuvent se faire **sur des groupes de fichiers**.
- **Point de départ** : réaliser une **sauvegarde FULL** de la base de données.
- Nécessite de paramétrier l'option **RECOVERY à FULL (Complet) ou BULK LOGGED (journalisé en bloc)**.

### 7.8.2 La commande BACKUP

```
BACKUP DATABASE <nom_base> FILEGROUPE =  
<nom_groupe_fichier> TO DISK =<nom_fichier>
```

```
BACKUP DATABASE <nom_base> FILEGROUPE =  
<nom_groupe_fichier> TO DISK =<nom_fichier> WITH  
DIFFERENTIAL
```

Figure 7.14: La syntaxe de sauvegarde de groupe de fichiers

La première syntaxe va effectuer **une sauvegarde complète** du groupe de fichier rattacher à une base de données, la seconde va effectuer **une sauvegarde différentielle** du groupe de fichier.

**Avant de débuter la méthode de sauvegarde de groupe de fichiers**, il faut au préalable **vérifier un certain nombre d'informations** qui nous autorisent à réaliser cette stratégie de sauvegarde à savoir :

1. **Le mode de récupération** des transactions est configuré à **FULL ou BULK FULL**
2. Vérifier **l'existence d'une unité de stockage logique** qui va contenir l'ensemble des sauvegardes de groupes de fichier, dans le cas contraire il faut la créer

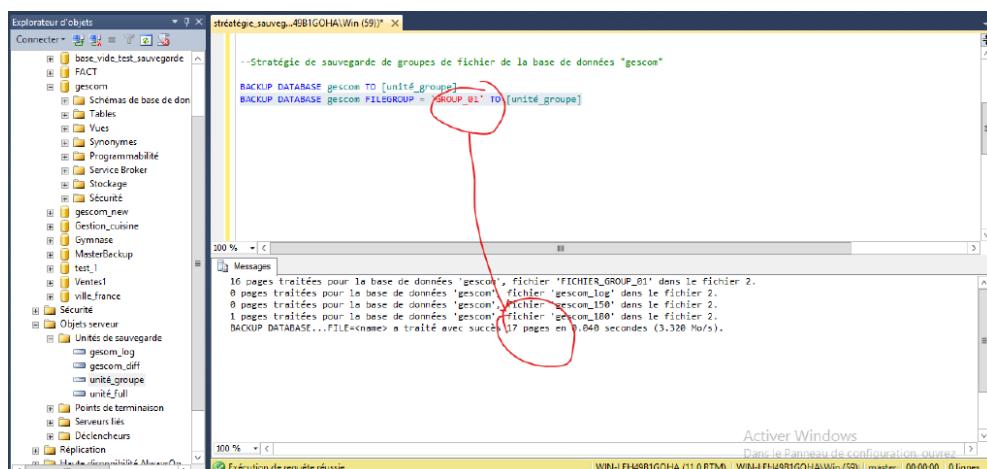


Figure 7.15: Etape 1 : stratégie de sauvegarde de groupe de fichiers

Dans cette partie, nous avons au préalable réalisé **une sauvegarde complète** qui est l'une des prérequis indispensables pour pouvoir effectuer la sauvegarde de groupe de fichier, comme vous pouvez le constater, le processus de sauvegarde a été réalisé avec succès, **le nombre de pages** qui ont étaient sauvegardées

dans l'unité de stockage logique s'élève à 17 ce qui signifie que le groupe de fichier pointe sur un fichier data **composé en moyenne de 2 extensions**.

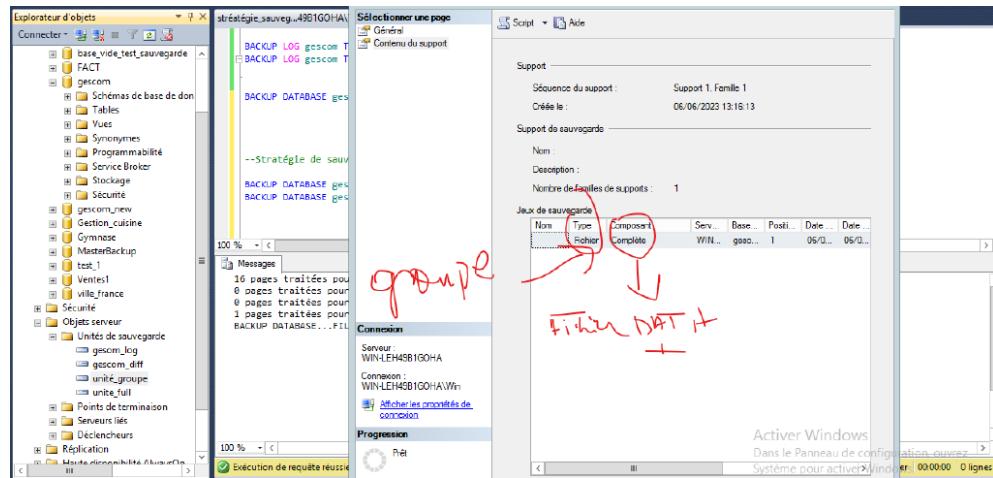


Figure 7.16: Etape 2 : stratégie de sauvegarde de groupe de fichiers

Dans cette figure nous avons procédé à la **vérification du contenu du support de sauvegarde** ou de l'unité logique contenant les fichiers data rattaché au **groupe de fichier GROUP\_01**, le jeu de sauvegarde nous informe sur le type de sauvegarde et également sur le composant ce dernier quand il est en **mode complet** signifie que l'ensemble des fichiers data appartenant au groupe de fichier ainsi qu'une partie du journal de transaction qui contient les informations sur le groupe de fichiers GROUP\_01 sont sauvegardés dans l'unité logique.

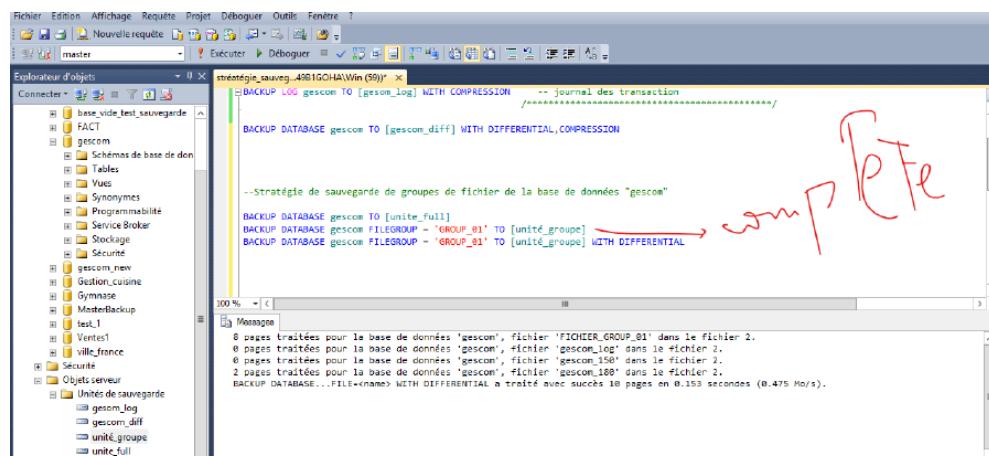


Figure 7.17: Etape 3 : stratégie de sauvegarde de groupe de fichier

nous avons réalisé **les deux méthodes de sauvegarde (FULL, DIFF)** sur **un groupe de fichier** de la base de données "gescom". En résumé la stratégie de sauvegarde qui combine le FULL et le DIFF peuvent s'appliquer soit sur la **base de données complète** soit sur l'un **des groupes de fichier** qui la compose.



#### NOTE IMPORTANTE

Lorsque vous procéder à une stratégie de sauvegarde sur SQL server 2012, il faut s'appuyer sur l'interface graphique à partir duquel vous **générer votre script en mode T-SQL** puis l'adapter selon vos préférences.  
**(Méthode d'un expert consultant dans les SGBDR)**

## 7.9 La sauvegarde partielle

### 7.9.1 La commande BACKUP

---

```
BACKUP DATABASE <nom_base> READ_WRITE_FILEGROUPS TO
DISK = <fichier_nom>
```

**Figure 7.18: La syntaxe de la sauvegarde partielle**



#### NOTE IMPORTANTE

Dans un groupe de fichier **en lecture seule**, il y'a très peu de chance qu'il y'a des extensions qui vont être modifier et par conséquence lors de l'application d'une sauvegarde, **le SQL server va les contourner ce qui réduira le temps de sauvegarde**

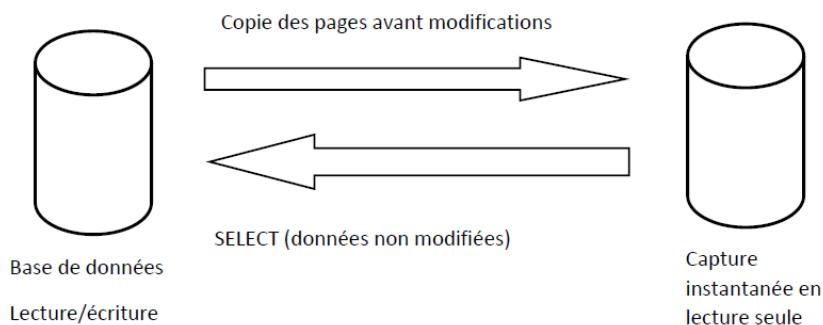
## 7.10 Les snapshots

Ce sont **des captures instantanées de la base de données** qui permet de créer **une image à un instant de T** et de conserver l'ensemble des modifications de données (figer l'état des données de la base de données)

### 7.10.1 C'est quoi un Snapshots ?

- C'est une **copie en lecture seule** des bases de données
- Elle est dépourvue de journal des transactions : ce qui est normal puisque **la copie de la base de données** créé est en **lecture seule**
- Impossible **de sauvegarder, restaurer ou détacher** une capture instantanée de base de données
- Le snapshot doit **se trouver dans la même instance** que la base de données source

### 7.10.2 Principe de fonctionnement d'un Snapshot

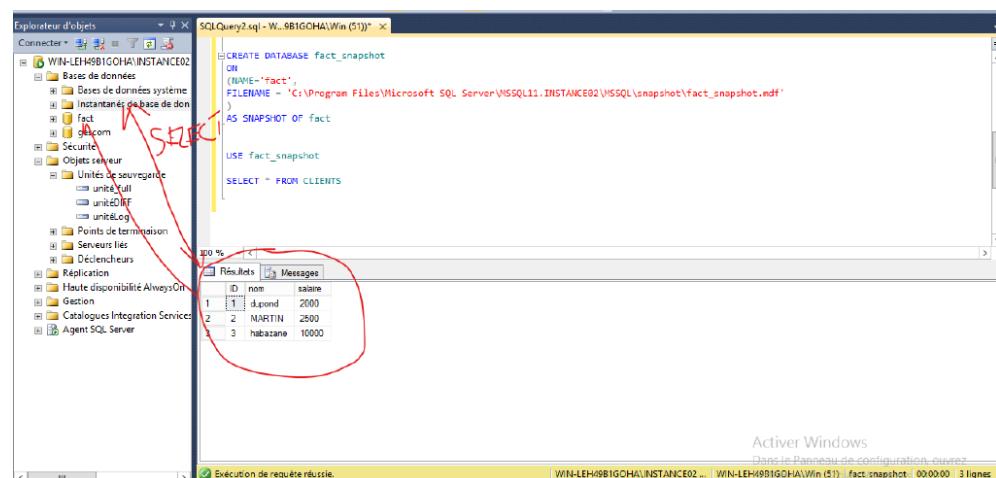


**Figure 7.19: Principe de fonctionnement d'un snapshot sous SQL server**

- **Explication :**

- Lorsque vous avez **une base de données source**, il existe un certain nombre de blocs, qui forme une suite contiguë de 64ko sur laquelle vous pouvez appliquer vos transactions de modification de données SELECT INSERT UPDATE.
- Une fois la base de données **en lecture seule** est créée qui est **initiallement vide**, le SQL serveur va faire une **copie des pages** de la base source **avant modification** puis les modifications peuvent être faites sur les pages 8ko.
- **Est-ce qui se passe si un SELECT est appliqué sur la base de données en lecture seule ?** : si la donnée a été **modifier depuis que le snapshot a été créé**, le sql server va nous **répondre à partir des pages qui se trouvent dans la capture instantanée** mais si un SELECT se fait sur une table qui **n'a pas subis de modification** sur la base de données source, **le SQL server va rediriger la requête vers la base de données source.**

### 7.10.3 Crédit d'une capture instantanée



The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure including 'fact' and 'fact\_snapshot'.
- SQL Query Editor:** Contains the T-SQL script for creating the snapshot and executing a SELECT statement.
 

```

CREATE DATABASE fact_snapshot
ON
NAME='fact',
FILENAME = 'C:\Program Files\Microsoft SQL Server\MSSQL11.INSTANCE02\MSSQL\snapshot\fact_snapshot.mdf'
AS SNAPSHOT OF fact

USE fact_snapshot
SELECT * FROM CLIENTS
      
```
- Results Grid:** Displays the query results showing three rows of data from the 'CLIENTS' table.
- Annotations:** Red circles highlight the 'fact\_snapshot' database name in the Object Explorer and the 'fact\_snapshot' database name in the T-SQL script.
- Status Bar:** Shows 'Exécution de requête réussie.' (Query execution successful).

Figure 7.20: Crédit d'un snapshot sous SQL server

Ce qui est intéressant à connaitre et bien retenir en tête de cette figure est que la requête qui a été envoyé au serveur va permettre au moteur de base de données d'aller **interroger directement la base de données source pour répondre au besoin de l'utilisateur** en affichant tous les enregistrements de la table CLIENT sans passer par la capture instantanée puisque aucun page de la base de données source n'a subi de modification (**Revoir le principe de fonctionnement des snapshots**).



#### NOTE IMPORTANTE

La capture instantanée de la base **ne contient pas de fichier journal de transaction** ce qui normal car la base créée est **en lecture seule**

#### 7.10.4 Comportement d'un SELECT sur un Snapshot

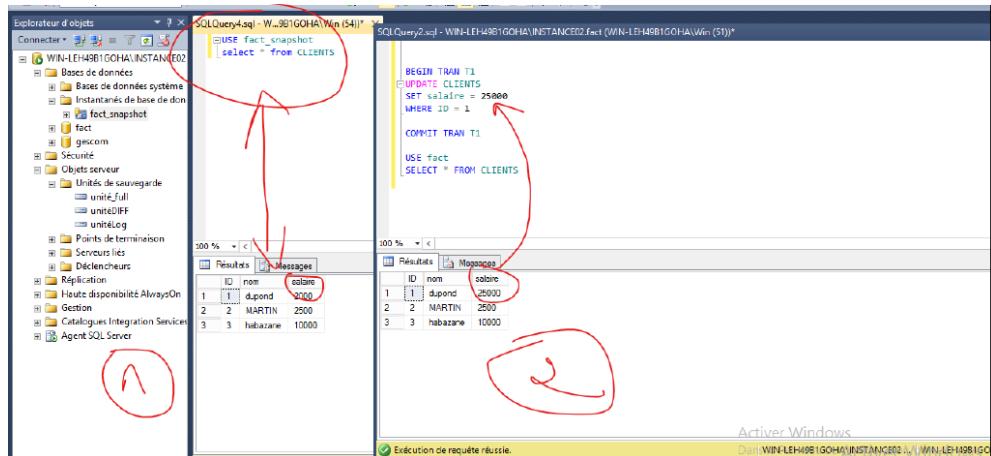


Figure 7.21: Comportement d'un SELECT sur un Snapshot

La partie 1 de la figure 7.21 montre bien le comportement d'un SELECT sur une capture instantané, après une petite mise à jour faite sur les données du client dupond qui se trouvent dans la base de données source, nous confirmons bien que le jeu de résultat 1 provient bien du snapshot qui a figé les données et qu'il n'y a pas eu lieu de redirection de requête vers la base de données.

source.

# Chapter 8

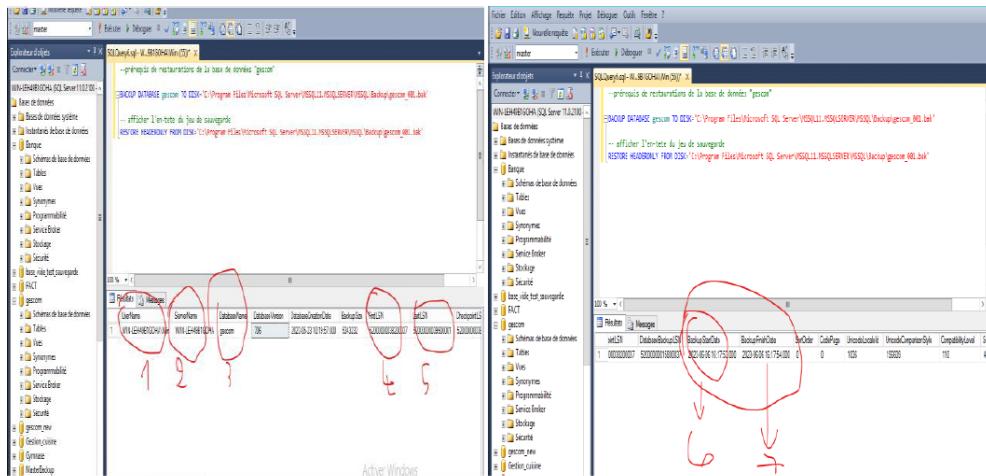
## La Restauration sous SQL server 2012

### 8.1 Les prérequis à la restauration

#### 8.1.1 ETAPE 1 : La vérification de la sauvegarde

La vérification des sauvegardes : SQL server dispose d'un certain nombre de commandes qui nous permettent de **lire ou de valider un jeu de sauvegarde** :

- **RESTORE HEADERONLY** : permet de lire le contenu l'en-tete d'un fichier ou d'un jeu de sauvegarde pour afficher les informations concerant les sauvegardes, cela est très utile pour s'assurer l'en-tete correspond bien aux informations de la base de données que vous voulez restaurer.



**Figure 8.1: La commande RESTORE HEADERONLY**

Cette commande va lire l'en-tête du jeu de sauvegarde dont l'emplacement est indiqué par le chemin vers le disk, un certain nombre d'informations très utile vont être affichés dans le jeu de résultat comme :

1. **le nom du serveur** sur lequel le jeu de sauvegarde a été réalisé.
  2. **le nom de l'instance** auquel l'utilisateur était connecté lors de la réalisation de la sauvegarde.
  3. **le nom de la base de données** contenu dans le jeu de sauvegarde.
  4. **l'identifiant de la première transaction** qui a été enregistré dans cette sauvegarde.
  5. **l'identifiant de la dernière transaction** qui a été enregistré dans cette sauvegarde.
  6. **date et heure de début** de la sauvegarde.
  7. **date et heure** de fin de la sauvegarde.
- **RESTORE FILELISTONLY** : permet d'afficher **des renseignements sur les fichiers de données et les fichiers journaux** contenu dans le jeu de sauvegarde.

```

SQLQuery6.sql - W...BTIGOHAVm (55) | x
-- prérequis de restaurations de la base de données "gescom"
BACKUP DATABASE gescom TO DISK='C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\Backup\gescom_001.bak'
-- afficher l'en-tête du jeu de sauvegarde
RESTORE HEADERONLY FROM DISK='C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\Backup\gescom_001.bak'

RESTORE FILELISTONLY FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\Backup\gescom_001.bak'
    
```

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure for 'gescom'. In the center, a query window titled 'SQLQuery6.sql' contains T-SQL code for performing a backup and then restoring it with the 'FILELISTONLY' option. The results pane at the bottom shows a table with file details, including file names, sizes, and file groups.

Figure 8.2: La commande RESTORE FILELISTONLY

- **RESTORE VERIFYONLY** : la plus intéressante et la plus pertinente pour valider que le jeu de sauvegarde est bien opérationnel.

```

RESTORE VERIFYONLY FROM DISK='C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\Backup\Adventureworks2012.bak'
    
```

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer displays the database structure for 'Adventureworks2012'. In the center, a query window contains the 'RESTORE VERIFYONLY' command. The results pane at the bottom shows a message indicating that the restore operation was successful.

Figure 8.3: La commande RESTORE VERIFYONLY

### 8.1.2 ETAPE 2 : Fermeture des sessions en cours

**Assurez-vous qu'aucune connexion utilisateur n'est en cours sur la base de données que vous voulez restaurer.** Le moyen le plus simple est d'exécuter la procédure stockée **sp\_who ou sp\_who2** pour identifier les sessions en cours et de faire un **KILL** pour les mettre en état d'arrêt et de ne laisser que la session principale. Un autre moyen s'offre à vous est d'intéroger la vue système **sys.dm\_exec\_sessions**



Figure 8.4: Fermeture des sessions en cours

### 8.1.3 ETAPE 3 : Restreindre toute nouvelles connexions au Serveur

Une fois que **les sessions en cours ont bien été annulées**, il va falloir bannir toute nouvelle connexion à l'instance de base de données à restaurer, de ce fait vous avez **le choix entre deux possibilités** :

1. Suspendre l'instance de base de données via **le gestionnaire de configuration de SQL server**

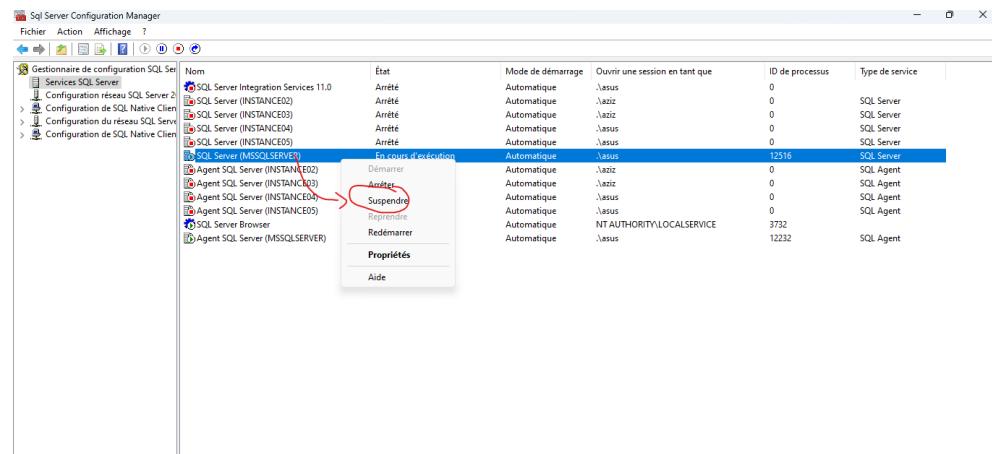


Figure 8.5: suspendre l'accès à l'instance

2. Configurer le paramètre **"restreindre l'accès"** à **"SINGLE\_USER"** dans propriétés de la base de données à restaurer.

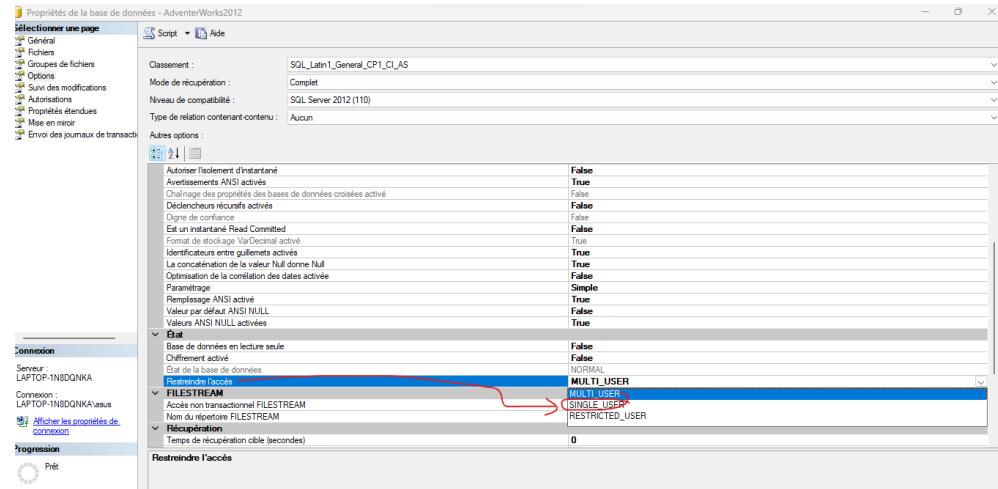


Figure 8.6: Restreindre à l'accès à la base



#### NOTE IMPORTANTE

L'option **MOVE** de l'instruction **RESTORE DATABASE** dans SQL Server est utilisée pour spécifier le nouvel emplacement physique des fichiers de données (fichiers .mdf et .ndf) lors de la restauration d'une base de données. **Cette option est particulièrement utile lorsque vous restaurez une base de données sur un serveur avec une configuration de stockage différente**

### 8.1.4 ETAPE 4 : Exemple de restauration d'une base de données

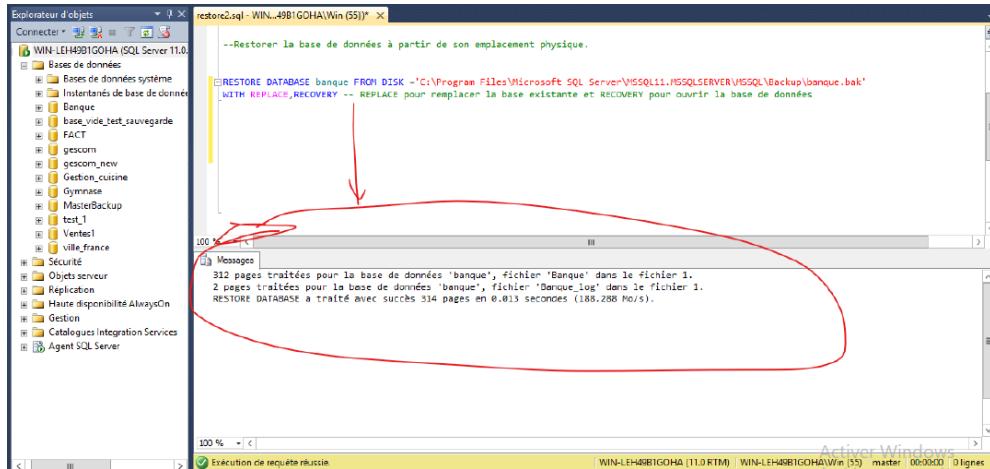


Figure 8.7: Processus de restauration

Cette instruction va nous servir à restaurer le jeu de sauvegarde qui se trouve dans l'emplacement physique indiqué en rouge, **WITH REPLACE** va remplacer la base existante par les données sauvegardés et **RECOVERY** va indiquer à SQL SERVER d'effectuer le mode de récupération complète du journal des transactions après la restauration et **d'ouvrir la base en mode READ/WRITE**.

## 8.2 La restauration d'une base de données

Dans cette partie nous allons voir **les différents type de restauration** en fonction de la méthode de sauvegarde choisie (FULL, DIFF, journal des transactions)

### 8.2.1 Les différents type de restauration utilisés en PROD

- Restauration **d'une sauvegarde complète** (FULL)

- Restauration **d'une sauvegarde différentielle** (DIFF)
- Restauration **d'une sauvegarde de journal de transactions**
- Restauration **partielle**
- Restauration **d'une page corrompue**

### 8.2.2 Les options de la commande RESTORE

- **RECOVERY** : lorsque cette option est ajoutée, elle va permettre de **mettre en fin la restauration** puis ouvrir la base aux utilisateurs.
- **NORECOVERY** : permet de faire l'inverse, le SQL server va laisser la base en mode RESTORE (il va rester en attente d'une nouvelle restauration). Par exemple si vous avez plusieurs restaurations à réaliser comme BACKUP FULL, BACKUP DIFF l'utilisation de cette option est nécessaire.
- **FILE** : qui permet de spécifier le fichier de sauvegarde que vous souhaitez restaurer si vous avez stocker vos sauvegardes dans une unité logique qui contient plusieurs fichiers physiques de sauvegarde.
- **MOVE ..TO** : permet de déplacer les fichiers de sauvegarde vers la cible, si par exemple les fichiers data et les fichiers journaux sont stocké dans un chemin différent.
- **REPLACE** : permet d'écraser la base de données existante.
- **STOPAT** : offre la possibilité de pouvoir remonter dans le temps.
- **STOPATMARK** : arreter à un identifiant de transaction bien spécifique (LSN)

## 8.3 Exemple de stratégie de sauvegarde et restauration

Les figures ci-dessous décrivent un exemple d'une stratégie de sauvegarde et restauration qui peut être appliquée sur un serveur PROD.

### 8.3.1 Processus de Sauvegarde

```

SQLQuery1.sql - WIN-LEH49B1GOHA\INSTANCE02.gescom (WIN-LEH49B1GOHA\Win (54))
=====
/******TP_complet de sauvegarde et restauration*****/
/******IMPORTANT IMPORTANT IMPORTAT*****/

--1-- commencer par créer de nouvelles unités de sauvegarde en mode SSMS
--2-- créer une sauvegarde sur une unité logique full

***** créer un jeu de sauvegarde complet le WEEK-END de la semaine 29/05/2023 *****/
BACKUP DATABASE gescom TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.INSTANCE02\MSSQL\Backup\full_gescom.bak'
MIRROR TO [unité_full] WITH INIT, CHECKSUM, FORMAT

*****vérifier si le jeu de sauvegarde n'est pas corrompu *****/
RESTORE VERIFYONLY FROM DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.INSTANCE02\MSSQL\Backup\full_gescom.bak'
RESTORE VERIFYONLY FROM [unité_full]

*****lancer des transactions de modification de données non incluse dans le jeu de sauvegarde complet*****
INSERT INTO [stagiaire].[CATEGORIES] VALUES (100,'backup diff','avant backup diff');
INSERT INTO [stagiaire].[CATEGORIES] VALUES (101,'backup diff','avant backup diff');

*****lancer une sauvegarde différentielle de la base de données gescom *****/
BACKUP DATABASE gescom TO DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL11.INSTANCE02\MSSQL\Backup\diff_gescom.bak'
MIRROR TO [unitéDiff] WITH DIFFERENTIAL, FORMAT
  
```

Figure 8.8: Simulation d'un processus de sauvegarde

```

***** lancer d'autres transactions supplémentaires *****/
INSERT INTO [stagiaire].[CATEGORIES] VALUES (102,'backup log1','avant backup log1');
INSERT INTO [stagiaire].[CATEGORIES] VALUES (103,'backup log1','avant backup log1');

***** Réaliser une jeu de sauvegarde du journal de transactions avec un intervalle de 1h *****/
BACKUP LOG gescom to [unitéLog] WITH INIT;

***** lancer d'autres transactions de modifications de données *****/
INSERT INTO [stagiaire].[CATEGORIES] VALUES (104,'backup log2','avant backup log2');
INSERT INTO [stagiaire].[CATEGORIES] VALUES (105,'backup log2','avant backup log2');

*****Réaliser un autre jeu de sauvegarde du journal de transaction après 1h *****/
BACKUP LOG gescom to [unitéLog] WITH INIT; -- WITH INIT écrase lancière sauvegarde

SELECT *
FROM [stagiaire].[CATEGORIES]
  
```

Figure 8.9: Simulation d'un processus de sauvegarde (suite)

### 8.3.2 Processus de Restauration

Avant de restaurer vos données perdues , veuillez vous assurer que **la restriction est levée** sur l'accès à la base de données. En d'autre terme le paramètre de

configuration MULTI\_USER à bien été définis.

```

TP.BACKUP_RESTORE\IMPORTANT.sql - WIN-LEH49B1GOHA\INSTANCE02.gescom (WIN-LEH49B1GOHA\Win (54))
-----+
SELECT s.login_name,s.host_name,s.program_name
FROM sys.dm_exec_sessions s
WHERE s.is_user_process = 1

ALTER DATABASE gescom
SET MULTI_USER

-----+
RESTORE DATABASE gescom FROM [unité_full] /*****          -- cette commande va permettre de mettre en standby le système sur le mode restauration
WITH REPLACE,NORECOVERY                                     -- restauration du backup full -- la base de données n'est pas ouverte aux utilisateurs
-----+
-----+
RESTORE DATABASE gescom FROM [unitéDIFF]                  -- restauration du backup DIFF en lui disant de sortir du mode de restauration
WITH REPLACE,NORECOVERY
-----+
-----+
RESTORE LOG gescom FROM [unitéLog]                         -- restauration du dernier backup du journal des transactions
WITH FILE=2,RECOVERY
-----+

```

Figure 8.10: Simulation d'un processus de restauration

# **Chapter 9**

## **Importation et exportation**

### **9.1 Architecture de transfert de données**

La figure ci-contre montre la composition d'une architecture de transfert de données quelque soit l'outil que vous allez employer pour réaliser la tache.

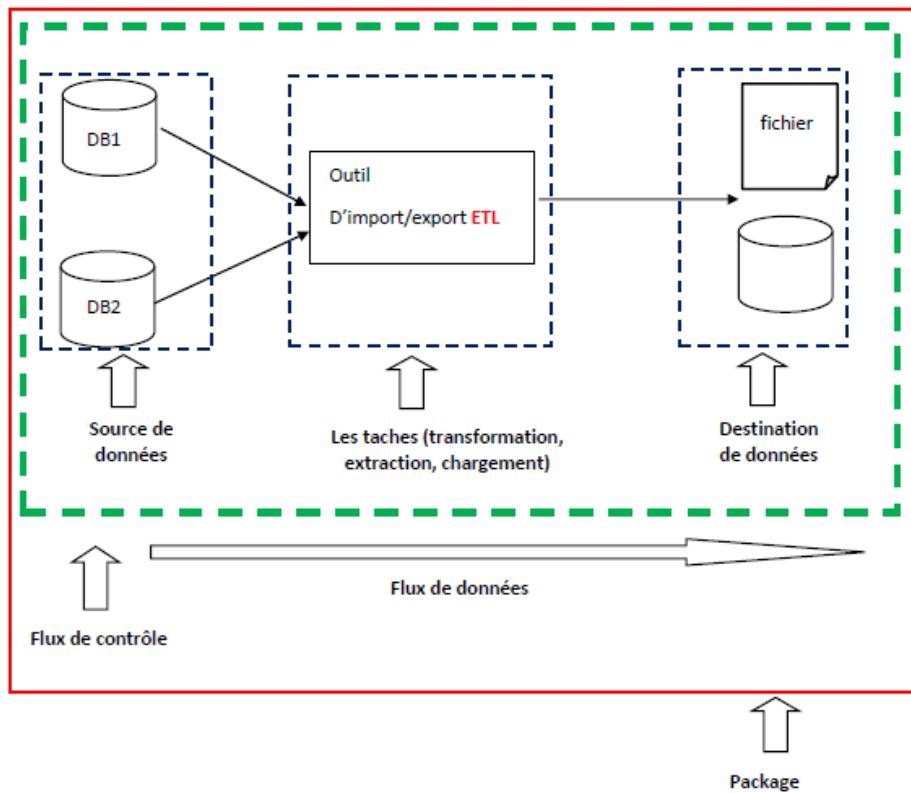


Figure 9.1: architecture de transfert de données

Le package contient trois parties à bien distinguer leur rôle, la première définit **la source de données**, c'est-à-dire à partir de quelle base ou type de fichier vous allez importer ou exporter vos données, la seconde est **un bloc d'extraction, de transformation et de chargement des données de la source** et la dernière partie représente **la destination de données** vers laquelle les données transformées vont être chargées.

Le Flux de données dans le cadre du processus d'import peut se faire entre deux bases de données ou d'une base de données vers un fichier plat, Comme le montre **la Figure 9.1**

## 9.2 Les outils de transfert de données

L'intérêts **des outils Microsoft d'import-export** est de pouvoir se connecter à d'autres source de données **autre que Microsoft**, il est totalement possible d'importer par exemple les données d'une base de données Oracle, d'une base de données MySQL... pour l'intégrer à SQL server 2012. Les outils permettant de réaliser ce genre d'opération sont :

1. **SSIS** SQL Server Intégration Services est un outil qui fonctionne en tant que Service dans SQL server, il permet de concevoir des ETL.
2. **BCP** : est un programme intégré à SQL server, il permet de réaliser les opérations de chargement de masse en mode ligne de commande entre un fichier plat et une base de données SQL server. Vous trouvez le programme dans le répertoire **tools** en suivant le chemin physique suivant :  
`C:\Program Files\Microsoft SQL Server\110\Tools\Binn`
3. **Le BULK INSERT** : est **une commande de chargement de masse** pour pouvoir importer les données à partir d'un fichier plat et les déployer sur une base de données SQL server.
4. **SELECT INTO** : Représente un méthode alternative au chargement de masse, elle permet de récupérer les données d'une table et de créer une nouvelle table à partir des données de la source.

### 9.2.1 L'outil SQL server Intégration Services

#### Les Terminologies SSIS

- **Les packages** : sont des unités de travail. Tous les source de données, les taches à réaliser et la destination de données vont etre contenu dans

des packages.

- **Les tâches** : sont contenu dans des packages, ils se charge **des opérations de transformation, de suppressions et de chargement**. Toute les opérations créées dans les packages seront gérée à travers les tâches.
- **Les conteneurs** : Permettent de fournir une structure pour des packages.
- **Le Flux de contrôle**: il permet de géré l'ensemble des étapes d'importation et d'exportation des données
- **Le Flux de données** : c'est le process qui débute à partir de l'extraction de données source pour les transformer à l'aide des ETL puis au finale les charger dans une base de données de destination.

## Assistant d'exportation des données

Partant du principe qu'on souhaite effectuer **une tache d'exportation des données** de la table CLIENT appartenant à la base de données "gescom" vers un fichier Plat, pour le faire veuillez bien vouloir suivre les étapes ci-dessous :

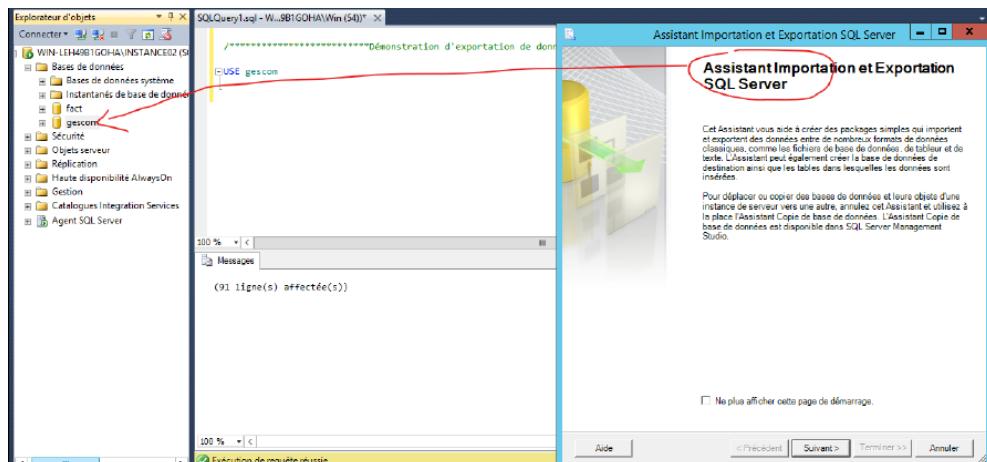


Figure 9.2: Etape 1 : Lancement de l'assistant SSIS

Cette étape nous montre l'utilisation de l'assistant de l'outil d'import-export

SSIS, pour pouvoir l'afficher il faut suivre l'arborescence suivante :

Clique droit sur la base \tache \export

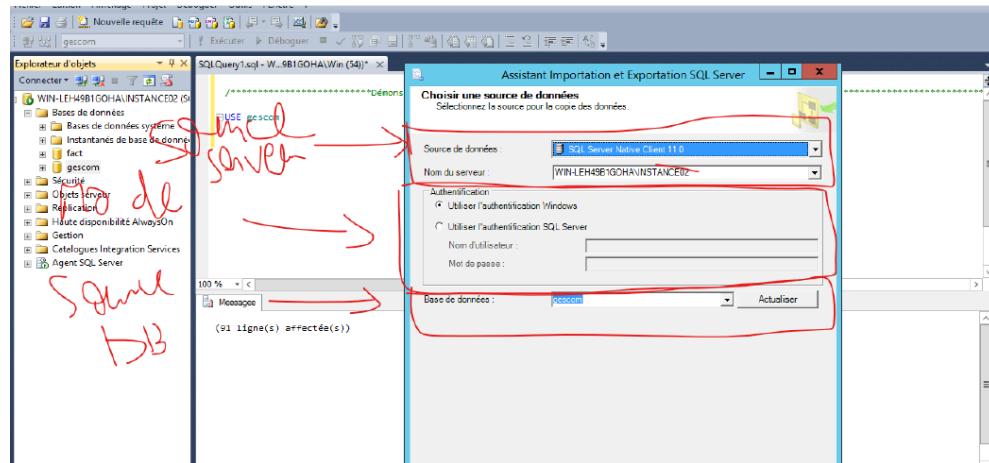


Figure 9.3: Etape 2 : Définir la Source de données

Dans cette étape l'utilisateur doit paramétriser ou **définir la source de données** pour pourvoir les extraire, il va en première partie sélectionner **le nom du serveur source** sur lequel vous voulais vous connecter, la deuxième partie consiste à **définir comment** le compte avec lequel vous accéder à **la base de données source est authentifié sur le serveur** puis à la fin vous précisez votre **base de données source**. En résumé cette interface vous permet de définir la source de données

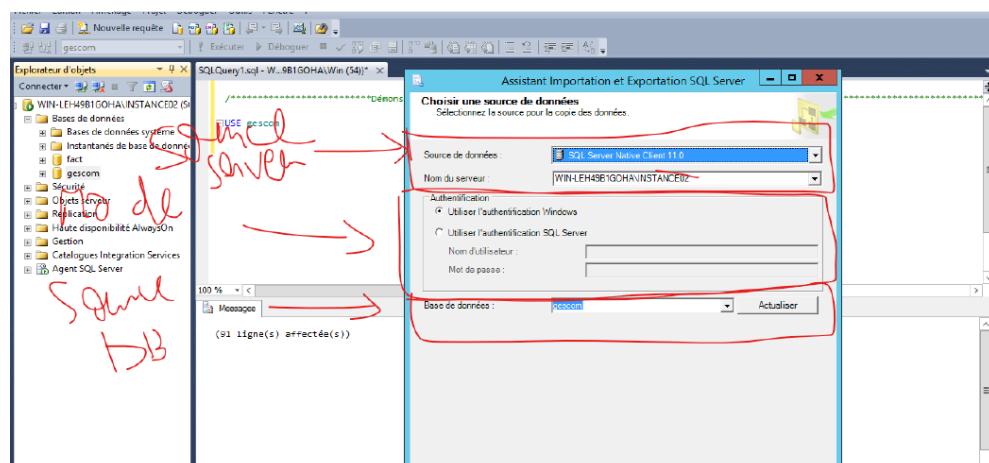


Figure 9.4: Etape 3 : Définir la Destination (The target)

Vous allez pouvoir **définir dans cette interface la Destination de vos données**, il est totalement possible de les charger soit dans **un fichier Excel**, ou vers d'autre serveur tel **qu'Oracle ou Analyses services** ou vers d'autres source de destinations qui supportent **les connexions OLE DB**.

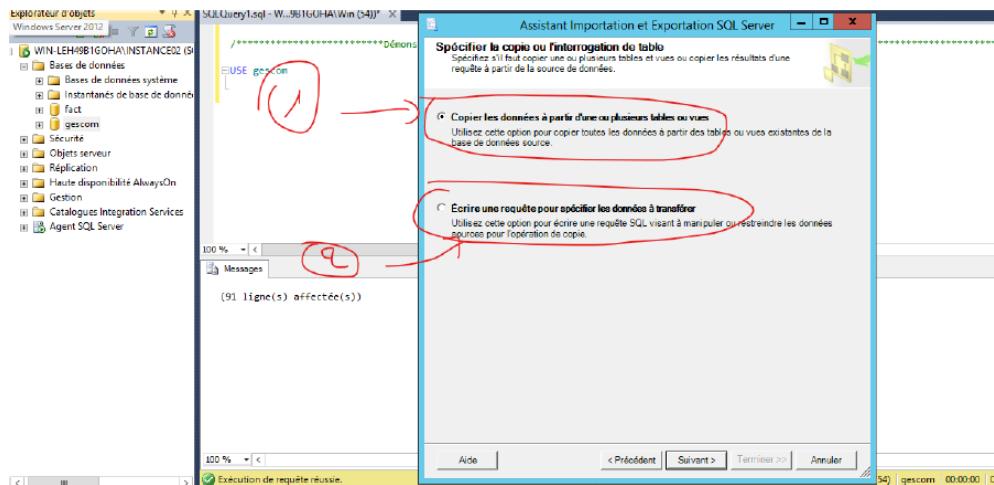


Figure 9.5: Etape 4 : Définir la copie des données

Dans cette partie, l'**objectif est de spécifier la copie des données** à partir de la source, en d'autre terme comment les données vont pouvoir être extraite de la source. Deux possibilités s'offrent à vous, soit de **copier la totalité d'une ou plusieurs tables ou vues**, soit **d'exporter à partir d'une requête SQL** si vous souhaitez restreindre les données sources.

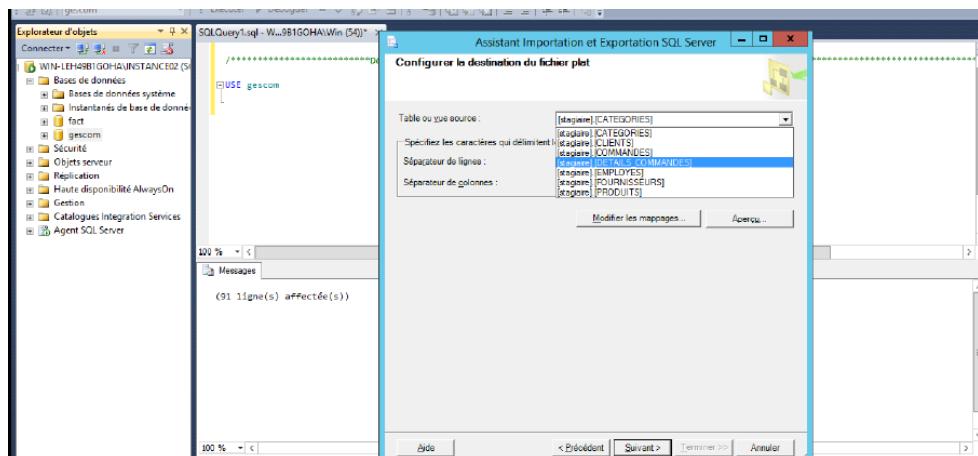
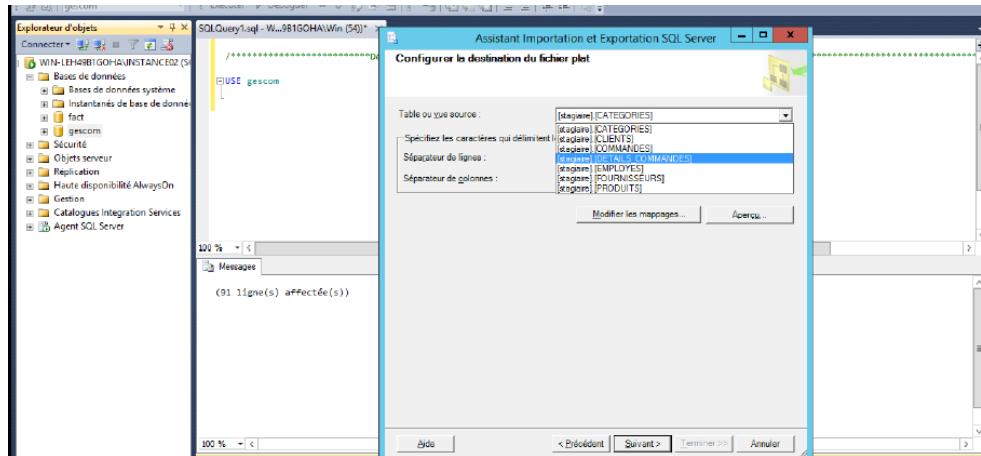


Figure 9.6: Etape 5 : Configuration du fichier de destination

Cette étape vous permet de **sélectionner la table ou la source de données**, vous êtes en mesure également de **modifier les mappages comme modifier le nom du champ, transformer le type pour le chargement**.



**Figure 9.7: Etape 6 : Création du package**

SSIS ici va nous permettre avec l'assistan de créer un package, qu'on **peut soit l'exécuter immédiatement** ou choisir de **le stocker directement sur SQL server** dans **la base MSDB**, ou choisir un emplacement physique **dans le système de fichiers**.



#### NOTE IMPORTANTE

1. Il fortement recommandé **d'enregistrer le package dans le système de fichier** sous le format fichier.dtsx pour pouvoir le déployer dans d'autres serveur en cas de besoin
2. Il totalement possible aussi de **chiffrer les données du package** avec une clé utilisateur
3. Tous **les packages** qui vont **réaliser l'opération d'import-export** des données portent **l'extension .dtsx** dans le système de fichier.

**NOTE IMPORTANTE**

Il faut toujours penser à **plannifier vos travaux d'exportation** après chaque création d'un package sur des **périodes de faible activités** du serveur pour éviter de **dégrader les performances**.

### Assistant d'Importation des données

**La démarche d'importation** sur SQL server 2012 est **similaire à la démarche d'exportation**, c'est-à-dire qu'il faut déployer l'assistant Intégration services pour définir la source de données puis la destination des données **à la seule différence que le flux de données va se faire dans le sens inverse** du flux de données lors du processus d'exportation.

### 9.2.2 L'outil BCP(Bulk Copy Program)

#### Caractéristiques

1. C'est un outil ligne de commande
2. Permet d'exporter les données d'une table ou d'une requête SQL
3. Permet d'importer un fichier texte dans une table SQL server

#### Exportation des données d'une table

La démarche à adopter pour **exporter** les données **en mode ligne de commande** est décrite dans la figure ci-dessous :

```

Microsoft Windows [version 6.3.9600]
(c) 2013 Microsoft Corporation. Tous droits réservés.

C:\Users\Win> cd\

C:\>bcp gescon.stagiaire.employees_Biz out C:\export_BCP\employees.txt -c -t ";" -r \t -T -S WIN-LEH49BIG0HA\INSTANCE02
Démarrage de la copie...
SQL Server Native Client : 0
Error = [Microsoft][SQL Server Native Client 11.0]Avertissement : l'importation de fichier BCP avec un fichier de format convertira les cha
limentées en NULL.

9 lignes copiées.
Taille du paquet réseau (octets) : 4096
Heure <n> Total : 18? Moyenne : <48.13 lignes par seconde>

C:\>-

```

**Figure 9.8: Utilisation de l'outil BCP**

Nous allons expliquer ici chaque partie numéroté de 1 à 8 qui compose la ligne de commande représenté dans la figure 9.8 :

- **1** : Spécifie l'objet de type Table qu'on souhaite importer ou exporter en définissant la base, et son schéma logique.
- **2** : "out" indique qu'on souhaite exporter les données
- **3** : spécifie le répertoire racine dans lequel on va stocker notre fichier.txt
- **L'option -C** : qui permet de spécifier que j'importe en mode caractère.
- **L'option -t** : qui spécifie la caractére de fin de champs
- **L'option -T** : qui spécifie le mode d'authentification Windows
- **L'option -S** : indique sur quelle instance je souhaite me connecter.
- **L'option -r** : spécifie l'indicateur de fin de ligne

Le standard output nous confirme que **le processus d'exportation a été réalisé avec succès**, nous pouvons ensuite le vérifier en accédant au répertoire de stockage du fichier qui contient les données exportées sur le système de fichiers. (**Voir la Figure 9.9**)

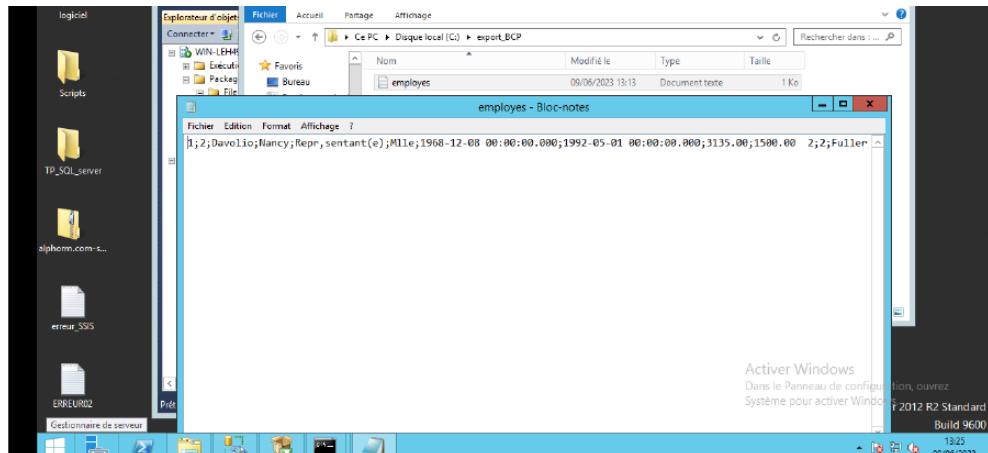


Figure 9.9: Vérification des données exportées

### Exporter les données d'une requête

Il est également possible au lieu d'exporter l'intégralité de la table, d'effectuer une restriction sur le processus d'exportation des données à travers une requête dotée d'une clause WHERE comme le montre la figure ci-dessous.



#### NOTE IMPORTANTE

Faites **ATTENTION** à bien préciser dans la ligne de commande **l'option "queryout"** pour faire savoir au programme bcp d'exporter seulement le résultat d'une requête.

```
C:\>bcp "SELECT * FROM gescom.stagiaire.employees_Bis WHERE salaire > 3000" queryout C:\export_BCP\emp_vip.txt -c -t ";" -r "\n" -S VIN-LEH49BIGORH\INSTANCE6
Démarrage de la copie...
SQLState = 01000; NativeError = 0
T-SQL State = 01000; SQL Server Native Client 11.0Révertissement : L'importation de fichier BCP avec un fichier de format convertira les chaînes vides dans le
limitées en NULL.
4 lignes copiées.
Taille du paquet réservé (octets) : 4096
Heure (ms) Total : 344 Moyenne : 811.63 lignes par secondes
C:\>
bcp
```

Figure 9.10: Exportation des données d'une requête

### 9.2.3 L'outil BULK INSERT

Représente une commande au niveau T-SQL elle permet d'effectuer une opération de chargement de masse à partir d'un fichier txt dans une table ou une vue de base de données dans un format spécifier par l'utilisateur.

```

USE gescom
SELECT * INTO stagiaire.employes_Bis FROM stagiaire.EMPLOYES
TRUNCATE TABLE [stagiaire].[employes_Bis]
/*
***** Import des données avec la commande T-SQL BULK/INSERT *****/
BULK INSERT [stagiaire].[employes_Bis] FROM 'C:\export_BCP\emp_vip.txt'
WITH (FIELDTERMINATOR='|', FIRSTROW=1 /* permet de sauter la première ligne si cette dernière contient le nom des champs */)
SELECT * FROM [stagiaire].[employes_Bis]

```

NO_EMPLOYEE	REND_COMPTE	NOM	PRENOM	FONCTION	TITRE	DATE_NAISSANCE	DATE_EMBAUCHE	SALAIRE	COMMISSION
1	2	Davolio	Nancy	Réprésentant(e)	Mme	1989-12-09 00:00:00.000	1992-05-01 00:00:00.000	3135.00	1500.00
2	2	Fürst	Andrew	Vice-Président	Dr.	1952-02-19 00:00:00.000	1992-09-14 00:00:00.000	10000.00	NULL
3	2	Levene	Jani	Réprésentant(e)	Mme	1962-08-30 00:00:00.000	1992-04-14 00:00:00.000	3500.00	1000.00
4	2	Buchenro	Steven	Chef des ventes	M.	1955-03-04 00:00:00.000	1993-10-17 00:00:00.000	8000.00	NULL

Figure 9.11: Utilisation de la commande BULK INSERT

La commande T-SQL BULK INSERT va permettre d'importer des données stockées sur le disque, il faut commencer par spécifier le nom du schéma dans lequel se trouve la table qui va contenir les données qu'on souhaite importer, le répertoire racine du fichier externe puis les différentes options comme FIELDTERMINATOR qui précise le séparateur de fin de champs utilisé dans le fichier.txt et FIRSTROW qui permet de sauter la première ligne si cette dernière contient les noms des champs.

# Chapter 10

## Gestion de la sécurité d'accès

### 10.1 Architecture de sécurité d'accès

#### 10.1.1 Terminologie

- **Les entités de sécurité** : des comptes de sécurités qui dispose d'un accès au serveur de données SQL, toute objet qui permet de se connecter au serveur est qualifier d'entité de sécurité
  - 1. **Windows** : Groupe Windows, Compte d'utilisateur de domaine, Compte d'utilisateur local, tous ces comptes peuvent être mapper, s'authentifier, déclaré à un compte de SQL server
  - 2. **SQL server**: Role server, les connexions déclarés au niveau SQL server.
  - 3. **Base de données** : Utilisateur de base de données, Role de base de données, Role d'application
- **Les sécurisables** : ce sont les objets gérés par le serveur (serveur, base,

schéma)

- **Les autorisations** : sont accordés aux entités de sécurités afin de pouvoir travailler avec les sécurisables



### NOTE IMPORTANTE

**Le compte** d'administration **Windows** est qualifié **d'entité de sécurité**, n'importe quels utilisateurs peuvent se connecter au serveur avec ce compte

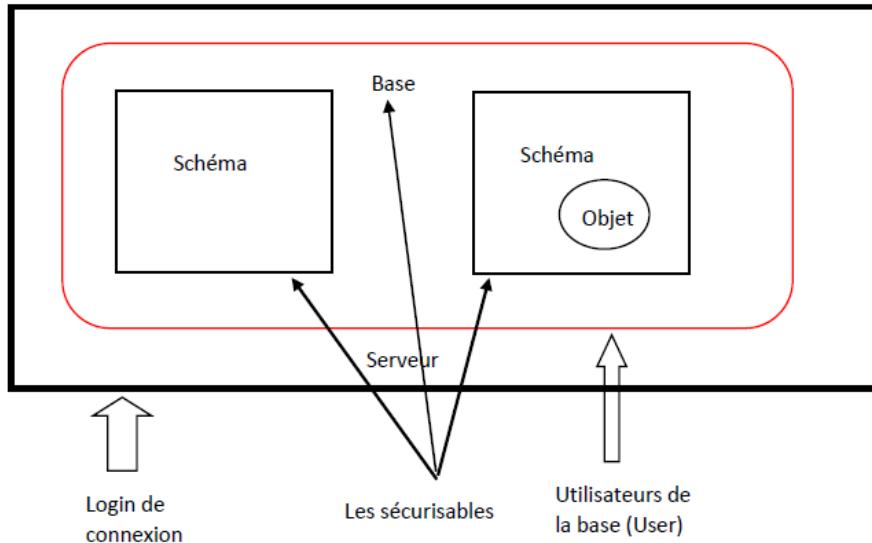


Figure 10.1: Architecture de sécurité d'accès sous SQL server

**NIVEAU 1** : La première barrière de sécurité est **le niveau serveur** accessible uniquement par **un compte qualifié d'entité de sécurité**. La liste des comptes disposant des droits d'accès à une instance SQL server sont stockés dans le répertoire de l'instance **"Sécurité"** facilement identifiables dans l'explorateur d'objet SQL server.

**NIVEAU 2** : la deuxième barrière n'est d'autre que la base de données elle-même. Afin que l'utilisateur puisse manipuler les objets au niveau base de

données, il doit en premier lieu disposer **d'un compte utilisateur** de base de données **mappé à son compte de connexion** à l'instance de base de données, ensuite lui gratifier les privilèges de manipulation d'objet en fonction des besoins applicatifs.

**NIVEAU 3 : le niveau de granularité minimal de sécurité d'accès** n'est d'autre que **le schéma**, son rôle est de pouvoir regrouper de manière logique les objets de types tables pour pouvoir par la suite attribuer des privilèges particulier à l'utilisateur de base de données. cela va permettre entre autre une **gestion optimal de la sécurité**

## **10.2 L'authentification SQL server**

Il existe sur SQL server management studio (SSMS) deux manières avec laquelle SQL server va gérer les sécurités d'accès à une instance de base de données :

### **1. Le Mode d'authentification Windows**

- (a) Lorsque ce mode est choisi **tous les logins de connexions sont authentifié par le système d'exploitation** avant d'accorder ou non le droit d'accès.
- (b) L'accès des utilisateurs se fait via un compte mappé à leur compte Windows ( déclarer le compte d'administration Windows à SQL server).

### **2. Le Mode d'authentification Mixte**

- (a) Tous les utilisateurs connectés via une connexion Windows déclarés sur SQL server sont validés.

- (b) Les utilisateurs connectés via un compte non Windows (un client Linux par exemple ) déclarés sous SQL server sont également validés.



#### NOTE IMPORTANTE

Quand le mode d'authentification mixte est choisi, est bien SQL server va en premier faire un test en utilisant la sécurité Windows si l'utilisateur du réseau dans SQL server appartient à un groupe approuvé par SQL server alors l'utilisateur sera connecter au serveur, si l'utilisateur n'est pas approuvé via Windows, SQL server celui qui va s'en charger pour authentifier l'utilisateur en vérifiant le compte et le mot de passe qui seront stocké au niveau de l'instance.

### 10.2.1 Crédation d'une entité via le mode Windows

Pour déclarer une entité de sécurité via le mode Windows en T-SQL, veuillez taper le code T-SQL suivant :

```
CREATE LOGIN [ <domaine>\<nom_connexion> ] FROM WINDOWS |  
WITH DEFAULT_DATABASE=<base_de_données> |  
DEFAULT_LANGUAGE = <langue>
```

Figure 10.2: Crédation d'une nouvelle connexion

Si vous souhaitez le créer via l'interface graphique SSMS, veuillez suivre ces étapes simples à réaliser :

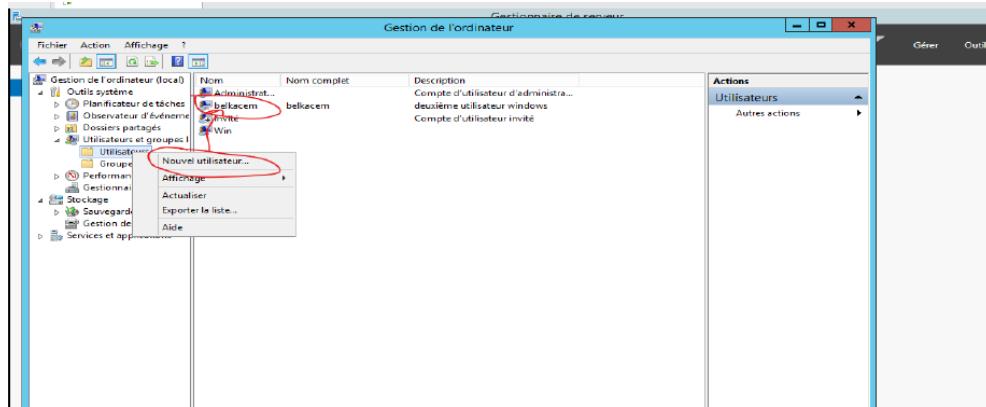


Figure 10.3: Etape 1 : Ajouter un compte d'administration Windows

La première étape à faire est **d'ajouter un compte utilisateur Windows** puis le mapper, le déclarer à SQL server. Afin d'ajouter un compte utilisateur windows, accéder à **gestion de l'ordinateur** ensuite faites un **clique droit sur l'utilisateur** que vous souhaitez **définir en tant que compte possédant les privilèges d'administration**.

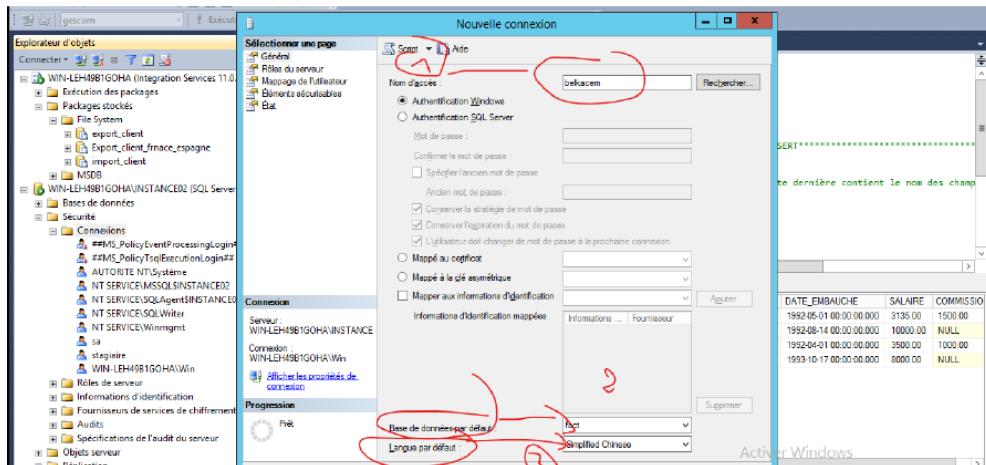
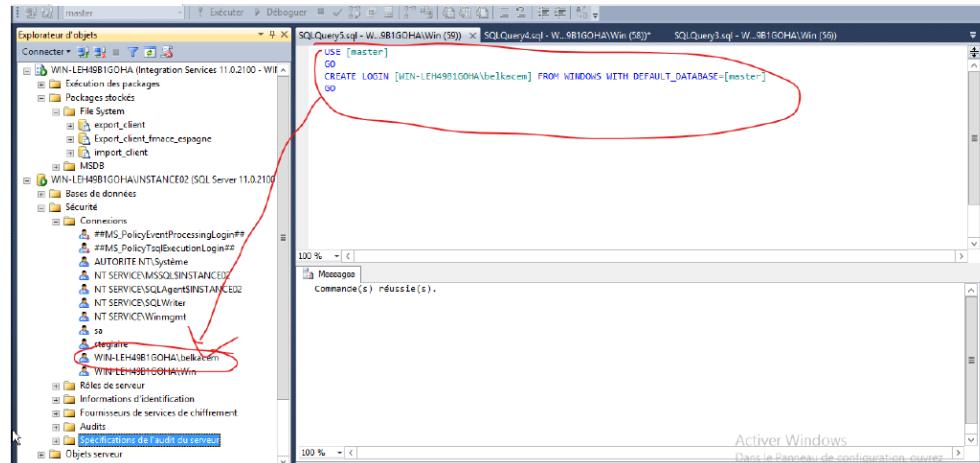


Figure 10.4: Etape 2 : Création d'un Login de connexion

Au niveau de **cette étape**, nous avons spécifié sur cette fenêtre, **l'identifiant de l'entité de sécurité** avec lequel l'utilisateur accède à SQL server, nous l'avons **rattacher** à la base de données **"fact"**, avant de spécifier à la fin **la langue par défaut**.



**Figure 10.5: Etape 3 : Création d'un login de connexion**

Le mappage du compte de connexion à SQL server se fait à travers l'instruction T-SQL décrite dans **la Figure 10.5**

### 10.2.2 Crédation d'une entité via le mode SQL server

La Figure ci-dessous représente l'instruction T-SQL il faut exécuter pour créer un compte avec l'authentification SQL server

```

CREATE LOGIN <nom_connexion> WITH PASSWORD = <mot_de_passe>
[MUST_CHANGE] | DEFAULT_DATABASE =<base_de_données>
DEFAULT_LANGUAGE =<langue>
    
```

**Figure 10.6: Etape 1 : Crédation d'un compte SQL server**

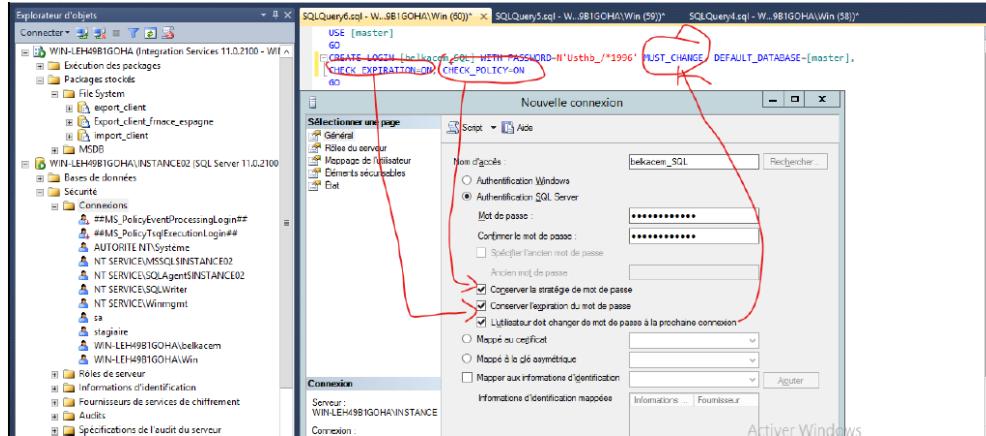


Figure 10.7: Etape 2 : Crédentiel SQL Server

Vous êtes en mesure de **générer la correspondance en mode T-SQL** depuis l'interface graphique SSMS en suivant les indications montrées sur la figure ci-dessus.



#### NOTE IMPORTANTE

Il existe une **ASTUCE PRATIQUE** pour identifier la provenance du compte de connexion à l'instance. **Si l'identifiant du compte est précédé par le nom du domaine** cela signifie que **le compte appartient au système d'exploitation**, en revanche **si l'identifiant n'est pas précédé par un domaine**, la provenance est automatiquement **SQL server**

## 10.3 Les crédenciales

### 10.3.1 Définition :

- Des entités de sécurité permettent à **des connexions en mode sécurité SQL serveur** d'accéder à une **ressource externe** au serveur de base de données il peut s'agir d'un nom utilisateur et un mot de passe qui permettent à SQL serveur d'authentifier et de se connecter à d'autres système ou ser-

vices

- Un credential = un compte windows
- Un compte SQL server est rattaché à un credential

### 10.3.2 Crédit et mappage d'un credential :

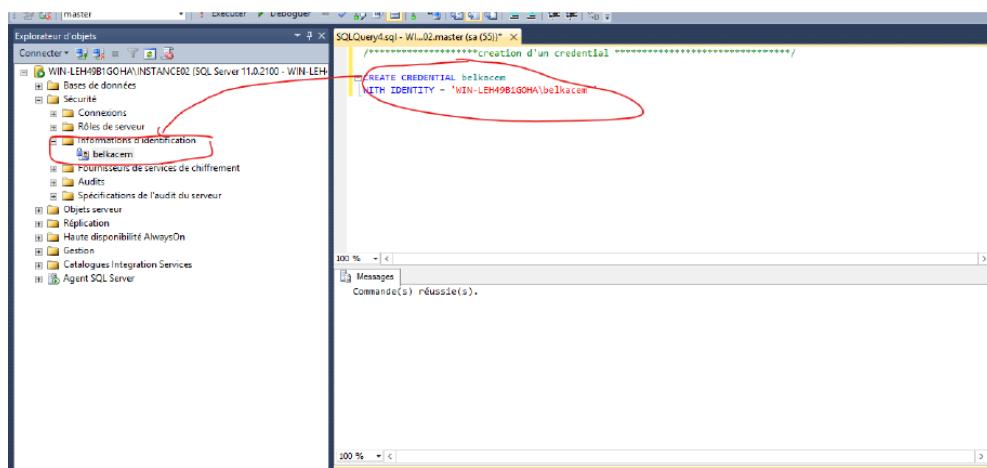


Figure 10.8: Crédit et mappage d'un credential

La figure ci-contre nous montre l'instruction T-SQL à utiliser pour créer un credential. Vous devez au préalable choisir un compte parmis la liste proposé par l'onglet "sécurité au niveau serveur" dont les informations d'identification vont être utilisées pour créer un credential.

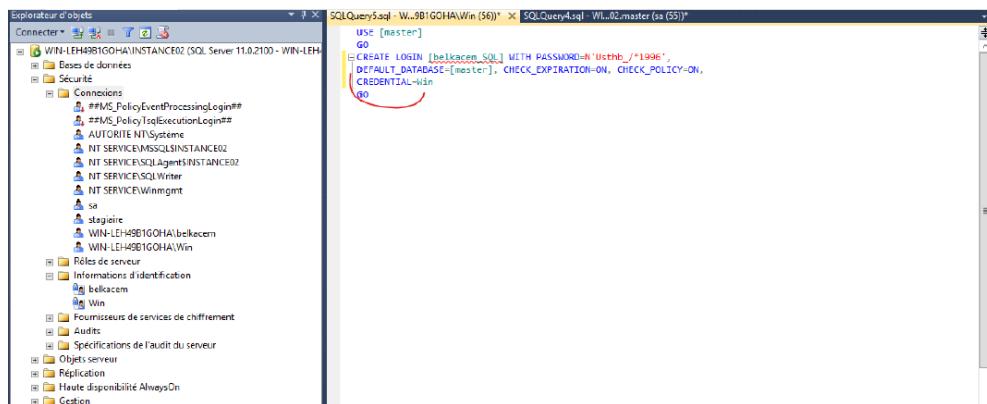


Figure 10.9: Le mappage d'un Crédential

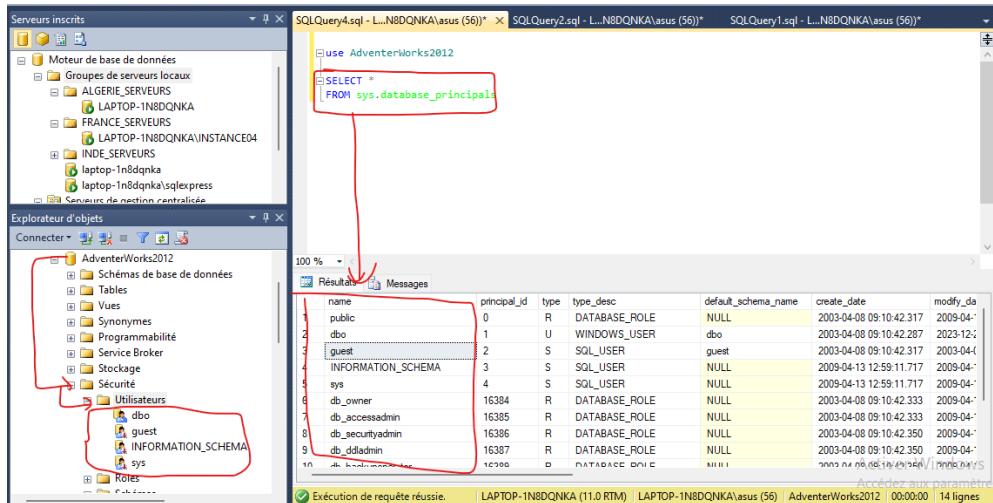
**Le compte SQL server "belkacem\_SQL"** va pouvoir accéder à des ressources externes à SQL serveur auxquelles Win a déjà le droit au niveau du système d'exploitation **en créant un mappage entre un LOGIN SQL serveur et un LOGIN windows**

## **10.4 Les utilisateurs de base de données**

### **10.4.1 Les caractéristiques :**

1. Crée au niveau de la base de données
2. Rattaché à un Login de connexion au niveau serveur
3. Les droit de manipulation d'objet (Schéma, Table, Index..) sont attribués aux utilisateurs de basxe de données.

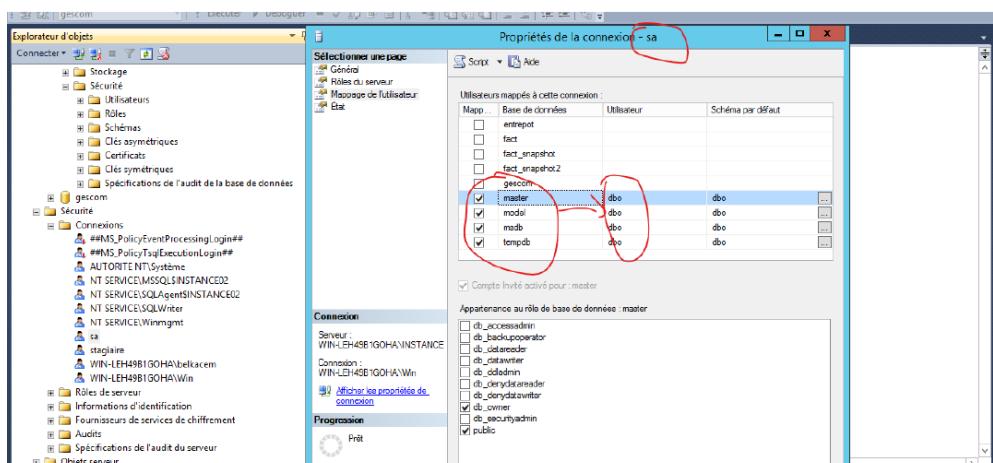
Pour afficher **la liste des utilisateurs d'une base de données applicative**, vous pouvez soit utiliser l'interface graphique SSMS pour accéder à l'aborescence suivante au niveau répertoire de l'instance "**DatabaseName \ Sécurité \ Utilisateurs**" soit d'intéroger **le vue dynamique "sys.database\_principals"**



**Figure 10.10:** Afficher la liste des utilisateurs de base de données

#### 10.4.2 L'utilisateur "dbo"

1. **Présent systématiquement** sur toutes les bases de données par défaut
  2. Les membres du rôle **SYSADMIN** sont mappés automatiquement à "**dbo**"
  3. **Impossible de le supprimer**
  4. **Tous les objets applicatifs** créés avec un membre du rôle **SYSADMIN** appartiennent automatiquement à "**dbo**"



**Figure 10.11: Mappage du compte "sa" à l'utilisateur "dbo"**

On remarque que pour chaque base de données système, le login de connexion "sa" est mapper à l'utilisateur "dbo" de la base de données. Par exemple pour la base de données "master" **si je me connecte avec le LOGIN de connexion "sa", il est directement mappé à l'utilisateur de base de données "dbo"** idem pour les autres.

#### **10.4.3 L'utilisateur "guest"**

1. **Présent dans tous les bases de données par défaut**, mais faites attention il n'est pas toujours activé (voir **la flèche rouge**)
2. **Autorise les connexions sans compte utilisateur** à accéder à la base de données.
3. **Autorise également l'accès des utilisateurs non authentifiés** (les utilisateurs orphelin) à certaine ressource à la base de données avec des permissions limitées.



#### **NOTE IMPORTANTE**

Les utilisateurs non authentifiés sont utiles si vous avez besoin **d'autoriser des connexions spécifiques** ou des **applications** à accéder à certains objets sans avoir à créer un compte de connexion distinct. d'où le terme GUEST=INVITE

#### 10.4.4 Activation du compte "guest"

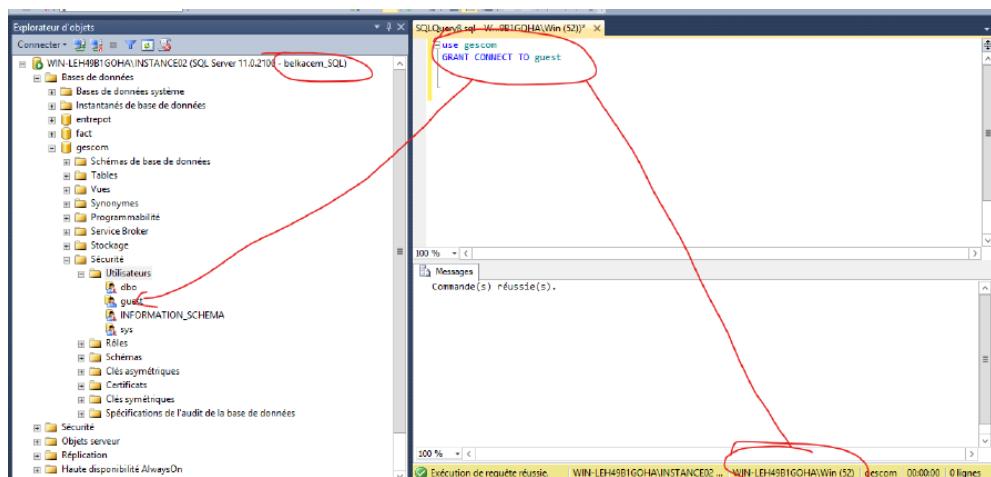


Figure 10.12: Activer le compte "guest"

En se connectant avec le compte disposant des priviléges d'administration, activer le compte "guest" pour permettre à d'autres utilisateurs créés avec le mode de sécurité SQL d'accéder à une base de données applicative avec des autorisations restreints

#### 10.4.5 Désactivation du compte "guest"

En utilisant l'instruction REVOKE CONNECT TO "guest" vous allez révoquer les droits d'accès au base de données à l'utilisateur qui n'est rattaché à aucun compte USER

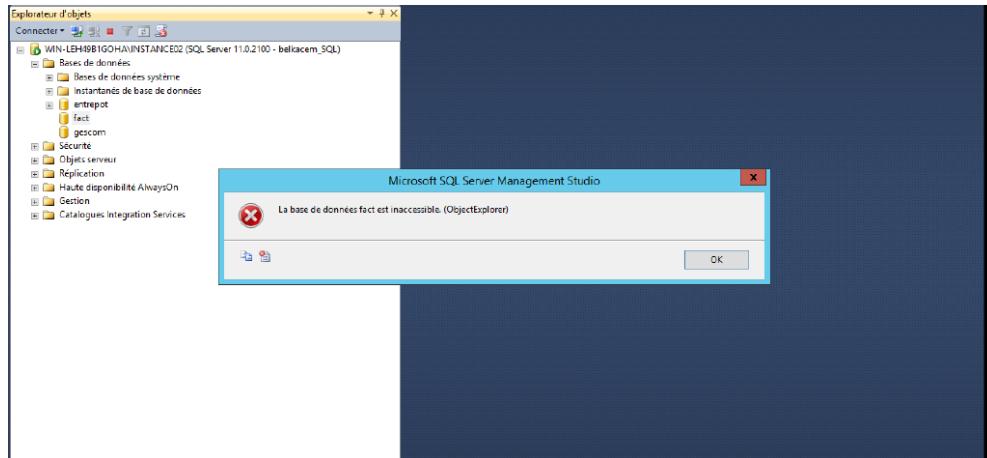


Figure 10.13: Désactiver le compte "guest"

#### 10.4.6 Création d'un utilisateur de base de données

```
CREATE USER <utilisateur> FOR LOGIN <login>
WITH DEFAULT_SCHEMA =<schema>
```

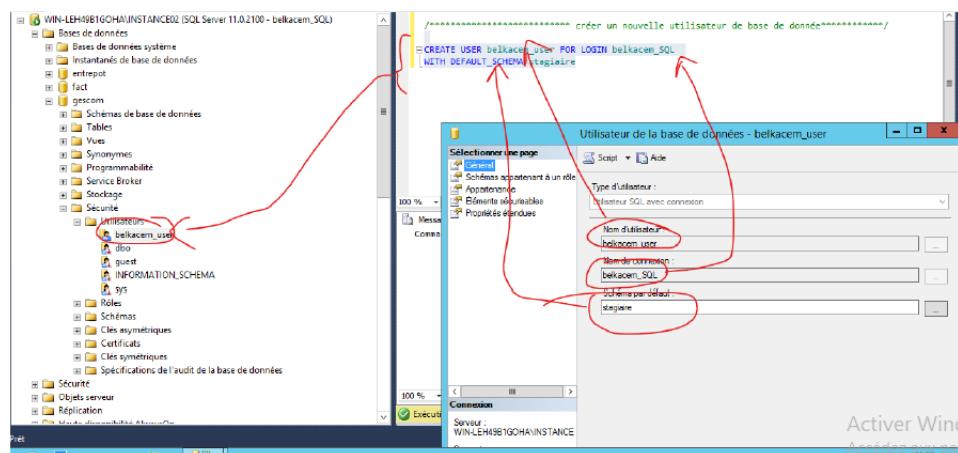


Figure 10.14: Création d'un utilisateur de base de données

Cette commande va servir à créer un nouveau utilisateur de base de données "belkacem\_user" qui va être **mapper, rattacher au LOGIN de connexion créé avec le mode d'authentification mixte et en précisant l'option de schéma par défaut**

### 10.4.7 Modification d'un utilisateur de base de données

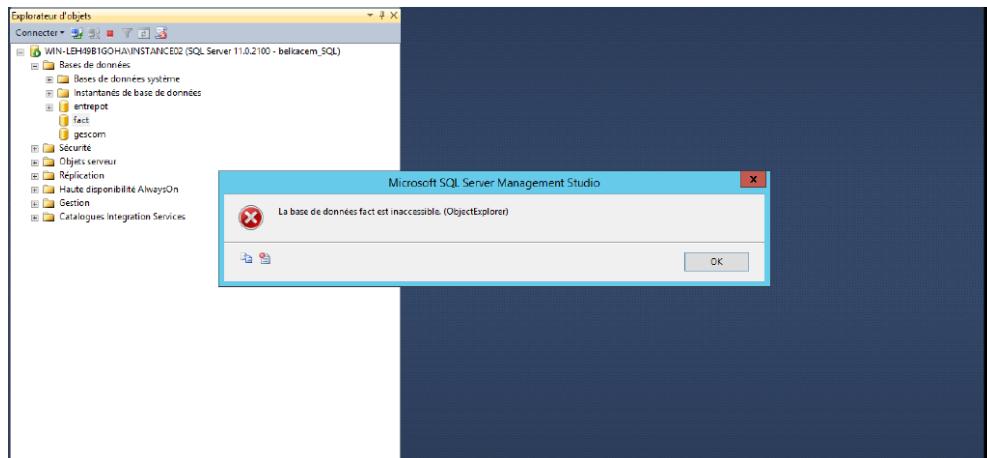


Figure 10.15: modifier un utilisateur de base de données



#### NOTE IMPORTANTE

1. Quand je souhaite créer un utilisateur qui a besoin de se connecter à la base de données, **il faut au préalable créer LOGIN de connexion** au niveau serveur.
2. Par défaut **si aucun mappage n'a été réalisé** à un compte utilisateur, **le login de connexion va utiliser le compte utilisateur "guest"** qui doit être **activé** pour pouvoir accéder à la base de données par défaut.
3. **Attribuer les privilèges** d'insertion, modification, création des objets au niveau de la base **au USER DE CONNEXION**

## 10.5 Les schémas

### 10.5.1 Caractéristiques :

1. Objet logique (enveloppe)
2. Permet un **regroupement logique** des objets
3. **Permet de gérer d'une manière plus optimale les droits** sur les objets
4. Associé à un utilisateur
5. **Le schéma par défaut** associé à l'utilisateur de base de données est **le schéma "dbo"**
6. Pour **accéder à des objets en dehors de son schéma**, il faut faire précéder **le nom de l'objet par le nom du schéma.**

### 10.5.2 Accéder aux objets d'un schéma

**Figure 10.16: Accéder aux objets d'un schéma**

Cette manipulation, nous a permis de conclure un point très important sur la manière d'accéder aux objets d'une base de données, là comme vous pouvez le constater, **le nom de l'objet de type table n'a pas été précédé par le nom du schéma**, cela s'explique par le fait que l'utilisateur s'est connecté au serveur avec le LOGIN de connexion "stagiaire" qui a été **mappé à l'utilisateur de base de données "stagiaire" dont le schéma par défaut est stagiaire**

### 10.5.3 Crédation et modification et suppression d'un schéma de base de données

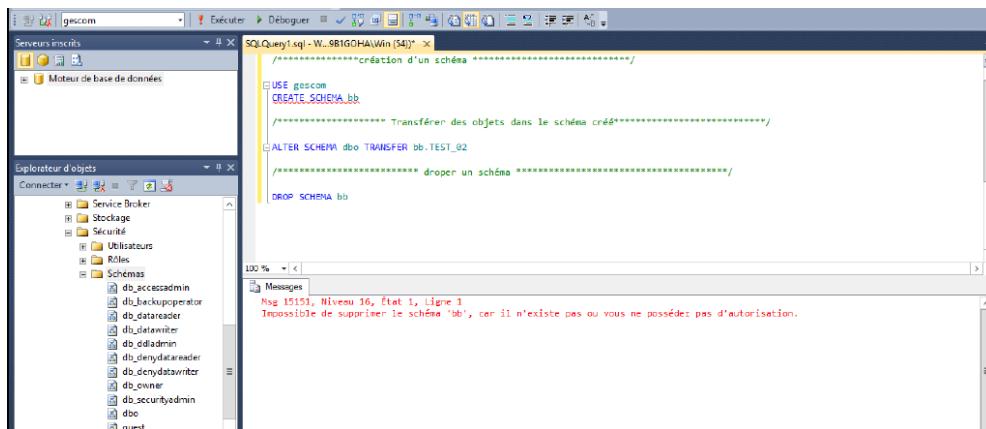


Figure 10.17: Crédation, Modification et suppression d'un schéma

Cette figure nous montre comment créer un schéma de sécurité sur SQL server avec la commande T-SQL CREATE SCHEMA, ce schéma est facilement repérable au niveau du sous répertoire "schéma" du répertoire parent "sécurité". Une fois le schéma est créé, il est maintenant possible de transférer des objets pour les rattacher au schéma de la base de données souhaitée avec la commande ALTER SCHEMA .... TRANSFER.

Avant de procéder à la suppression, il faut s'assurer que le schéma n'est pas référencier par aucun objet de base de données, dans le cas contraire, le SGBDR vous empêchera de supprimer l'objet pour préserver la cohérence et l'intégrité de la base.



#### NOTE IMPORTANTE

De point vue organisation, il est recommandé de créer vos propres schéma au lieu de tous mettre dans le schéma "dbo", cela simplifiera l'organisation et la gestion de droits

## 10.6 La gestion des droits sous SQL server



### NOTE IMPORTANTE

Les droits sont bien entendu les autorisations qui vont nous permettre de travailler avec les bases de données qui vont être créer sur SQL serveur. Nous allons voir que la gestion des droits fonctionne de manière hiérarchique

### 10.6.1 Les niveaux d'attribution des privilèges

1. **Le niveau Serveur** : tous les privilèges qui vont être attribuer au niveau de l'instance vont impacter l'ensemble des bases de données de votre instance
2. **Le niveau base de données** : pour pouvoir attribuer les privilèges au niveau base de données, il faut accéder aux propriétés de la base en suivant le chemin suivant : **propriété \autorisation**.  
Il existe deux type de droit au niveau de la base :
  - (a) **Les droits d'utilisation d'instruction** : comme CREATE TABLE, CREATE USER, CREATE LOGIN..
  - (b) **Les droits sur les objets** : SELECT, INSERT, UPDATE..
3. **Le niveau schéma** : pareille nous pouvons afficher la liste des autorisations dans propriétés du schéma **propriété \autorisation**
4. **Le niveau objet** : idem nous pouvons afficher la liste des autorisations dans propriété de la table **propriété \autorisation**

## 10.6.2 Les priviléges au niveau Serveur

1. Accordés au **Login de connexion**
2. Pour accorder **les droits au niveau serveur**, il faut être positionné sur la **base Master**

Pour accorder **des droits au niveau de l'instance**, il faut accéder au propriété du serveur dans l'onglet "**Autorisation**"

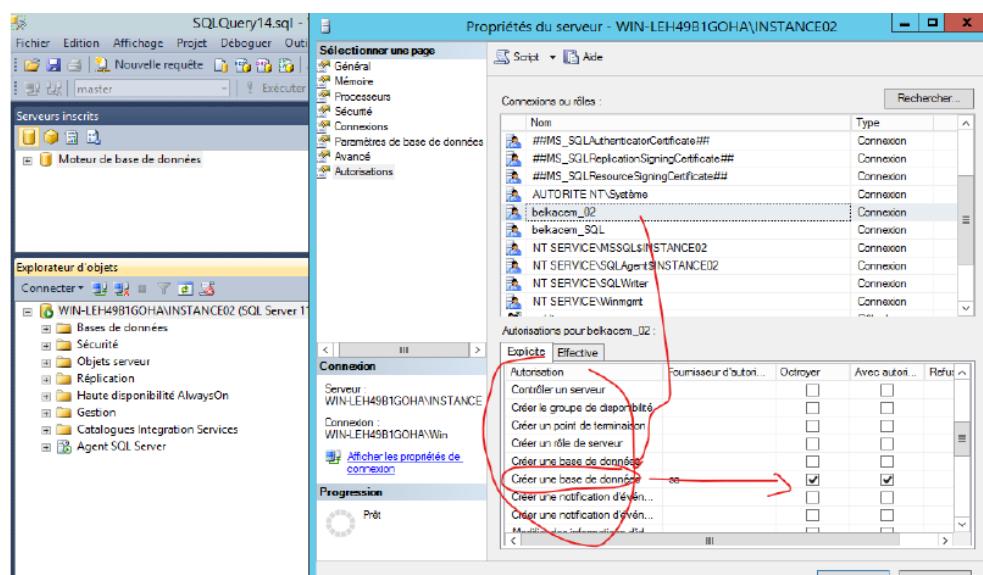


Figure 10.18: Les priviléges attribués au niveau serveur



### NOTE IMPORTANTE

Quand on attribue **des priviléges à un niveau supérieur**, il impacte tous les **niveaux inférieurs** par exemple si j'attribue des priviléges au niveau de la base, cela va impacter le niveau schéma et le niveau des objets

### 10.6.3 Les privilèges d'utilisation d'instruction (liste non exhaustive)

- **CREATE DATABASE** : pour créer une base de données
- **CREATE PROCEDURE** : pour créer une procédure stockée
- **CREATE TABLE** : pour créer une table
- **BACKUP DATABASE** : pour une réaliser une sauvegarde
- **CREATE RULE** : pour créer un rôle
- **CREATE VIEW** : pour créer une vue
- **BACKUP LOG** : pour réaliser une sauvegarde du journal des transactions

### 10.6.4 Les privilèges sur les objets

- **TABLE:** SELECT, INSERT, UPDATE, DELETE
- **PROCEDURE** : EXECUTE pour pouvoir exécuter une procédure ou pas.

### 10.6.5 Etude de cas

On souhaite à travers cet exemple **accorder les privilèges sur les sécurisables** de la base de données "**gescom**" à l'utilisateur SQL server "**belkacem\_02**".

Dans **propriétés de la base de données** vous avez la possibilité de **lister l'ensemble des privilèges** qui ont été attribués à un utilisateur de base de données.

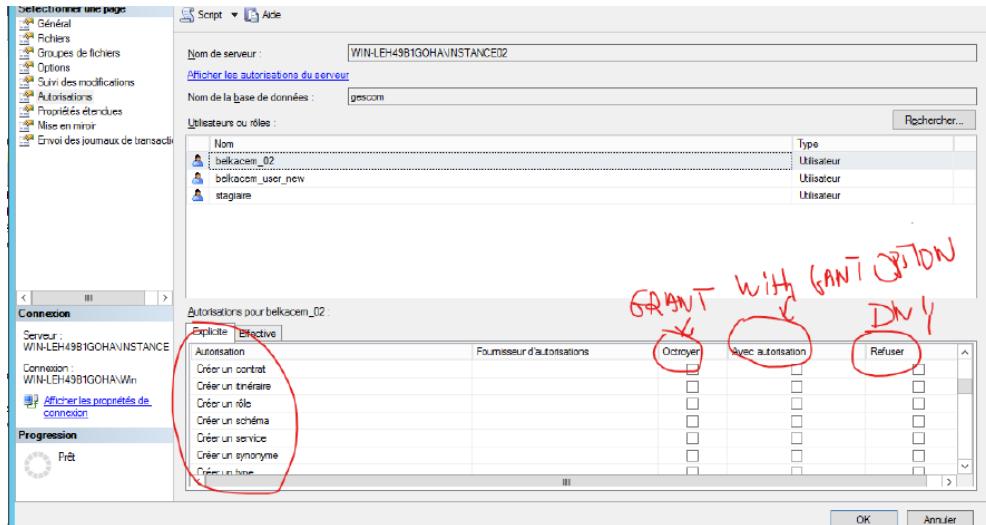


Figure 10.19: Attribution des droits à l'utilisateur "belkacem\_02"

À droite de la liste des privilèges, on retrouve les différentes **commandes de gestion de droit** comme **Octroyer** qui est équivalent au **GRANT**, "avec autorisation" c'est l'équivalent à **l'option WITH GRANT OPTION**, "refuser" est l'équivalent de **la commande DENY** qui va interdire un ou plusieurs privilèges.

Par défaut le fournisseur d'autorisation est mappé au schéma "dbo"

## 10.7 La gestion des rôles

### 10.7.1 Caractéristiques :

Un rôle n'est pas moins qu'un **ensemble de privilèges**, c'est un regroupement de privilèges (INSERT, UPDATE, DELETE)



#### NOTE IMPORTANTE

Quand vous allez avoir **plusieurs utilisateurs** qui vont avoir **besoin des mêmes privilèges** sur une base ou sur des objets, il préférable **au lieu d'attribuer des privilèges de manière unitaire à chaque utilisateur**, il vaut mieux **créer des rôles**, rajouter des privilèges à ces rôles puis **rajouter les utilisateurs comme étant membre de ces rôles**.

1. Existe dans **les trois niveaux** dans l'architecture SQL server (**Au niveau serveur, au niveau base de données, au niveau application**)
2. il existe deux types de rôle :
  - **Role Utilisateurs** : ces des roles créés et personnalisés par l'utilisateur lui-même.
  - **Role fixes** (au niveau base et serveur) : ces role sont prédéfinis lorsque vous réaliser une installation d'une instance SQL server

### 10.7.2 Les roles au niveau serveur

La liste des roles fixes au niveau serveur sont :

1. **SYSADMIN** : **(super administrateur de l'instance)** : permet à un utilisateur qui en est membre d'être un super utilisateur de l'instance, exécute n'importe quelle opération sur le serveur.
2. **SERVERADMIN** : permet la configuration des paramètres au niveau serveur, tout login de ce rôle va pouvoir paramétrier les options de configuration du serveur comme par exemple : paramétrier la mémoire cash du serveur, restreindre le nombre d'utilisateur qui peuvent se connecter simultanément au serveur

3. **SECURITYADMIN** : gestion de connexion d'accès au serveur, un utilisateur membre de ce rôle pourra créer, modifier, supprimer un LOGIN de connexion et le mapper à un compte utilisateur etc....
4. **PROCESSADMIN** : gestion des traitements sous SQL server, un utilisateur membre de ce rôle pourra gérer, surveiller l'ensemble des processus exécutés sur l'instance.
5. **DBCREATOR** : Un utilisateur membre de ce rôle va pouvoir créer et modifier des bases de données.
6. **DISKADMIN** : un utilisateur membre de ce rôle pourra faire la gestion des fichiers sur le disque, grossièrement créer et gérer des fichiers au niveau du système d'exploitation.
7. **PUBLIC** : est un moyen de définir des autorisations de base pour tous les utilisateurs d'une base de données.



#### NOTE IMPORTANTE

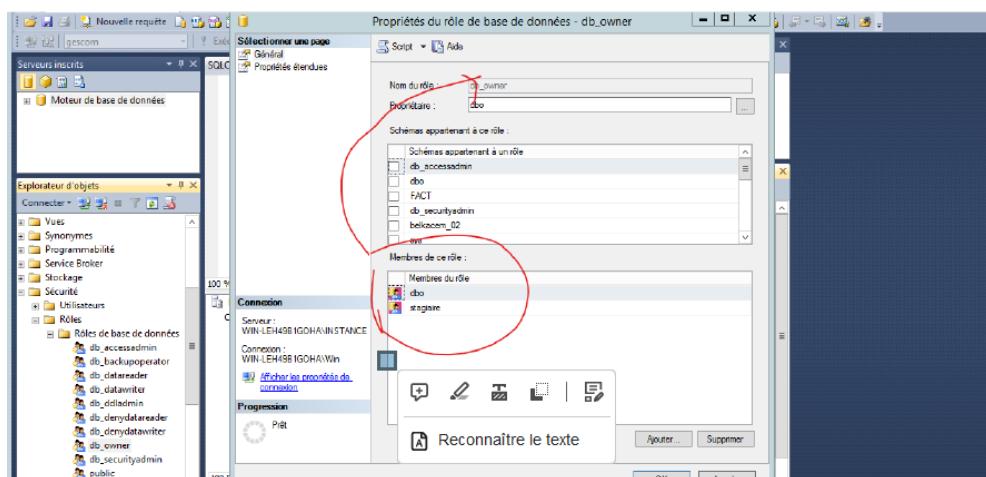
Tous les Login de connexion créés sur l'instance vont être par défaut membre du rôle "public".

### 10.7.3 Les roles au niveau base de données

#### Les roles fixes

1. **Db\_owner** : propriétaire de la base
2. **Db\_accessadmin** : ajouter ou supprimer des utilisateurs de base de données
3. **Db\_datareader** : un utilisateur membre de ce rôle va pouvoir faire un SELECT sur toutes les tables de la base de données

4. **Db\_ddladmin** : un membre de ce rôle va pouvoir exécuter les instructions DDL tel que CREATE, ALTER
5. **Db\_securityadmin** : faire la gestion des rôles , des autorisations sur les instructions et les objets
6. **Db\_backupoperator** : réalise les sauvegardes de la base de données
7. **Db\_denydatareader** : Pour interdir le SELECT/INSERT sur toute la base de données
8. **Db\_denydatawriter** : pour interdire n'importe quelle opération de modification de données INSERT UPDATE DELETE sur la base



**Figure 10.20: Consulter les membres d'un rôle de base de données**

En faisant clique droit sur le rôle de base de données "**db\_owner**", nous pouvons ainsi visualiser la liste des utilisateurs membre de ce rôle qui leur permettront d'obtenir le **statut de propriétaire** de la base de données.

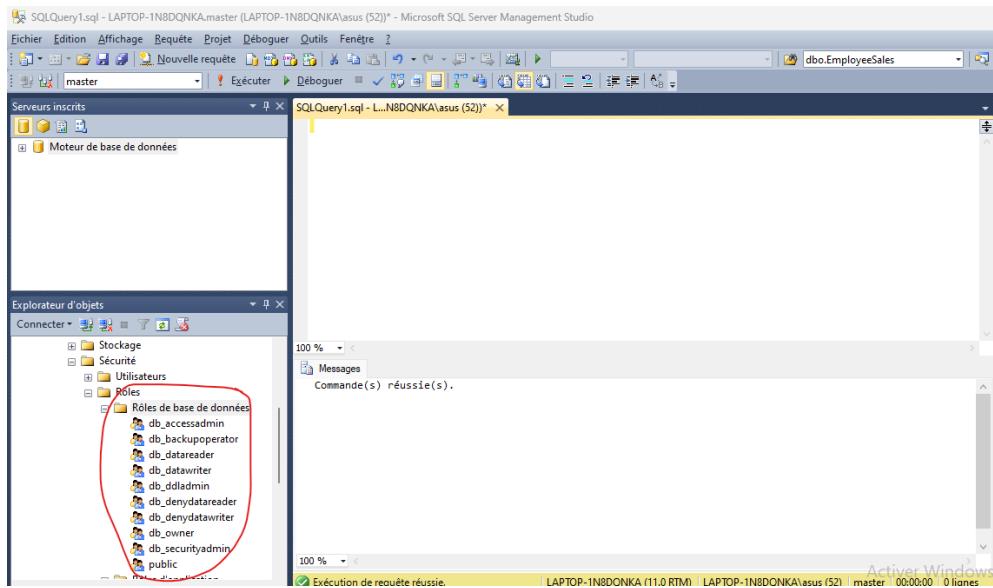


Figure 10.21: La liste des rôles fixes de base de données

### Les roles utilisateurs

C'est les plus courants, ce sont les rôles que vous allez créer vous même pour avoir un certain nombre de priviléges sur les objets.



#### NOTE IMPORTANTE

Les rôles facilitent l'audit des activités SQL, en associant pour chaque rôle des actions. Elle offre une **approche structurée et flexible** pour gérer l'accès aux données dans les environnements métier.

- ✓ **CREATE ROLE <nom\_role> AUTHORIZATION <nom\_proprietaire>**
- ✓ **Sp\_addrolemember ajoute un membre au rôle**
- ✓ **Sp\_droprolemember : retirer un membre au rôle**
- ✓ **DROP ROLE <nom\_role>**

**Figure 10.22: Crédit d'un rôle utilisateur de base de données**

### Les rôles d'application

- Définis au niveau base
- Ne possède aucun utilisateur
- **Protégé par un mot de passe**
- Permet de **donner les droits nécessaires** pour l'exécution d'une application
- **Nécessite d'être activé** par un utilisateur
- **Prend le dessus sur les priviléges de l'utilisateur** qui a activé le rôle d'application



#### NOTE IMPORTANTE

Utiliser les rôles d'application pour permettre l'accès à des données spécifiques aux utilisateurs **qui se connectent via une application spécifique**



#### NOTE IMPORTANTE

**Les rôles d'application** vont permettre à des applications clientes de gérer leur propre sécurité indépendamment du contexte de sécurité lié à SQL server, en accédant directement à certaines tables de la base de données

### Ce Connecter avec le role d'application

Les étapes suivantes composent le processus par lequel un rôle d'application fait basculer les contextes de sécurité :

1. **Un utilisateur exécute une application cliente** : supposons qu'un utilisateur souhaite à travers une application métier de gestion de ressource humaine d'accéder à certaines ressources de la base de données d'une entreprise comme par exemple le nombre de salarié, les antécédents professionnels. Un rôle d'application doit être défini au niveau de la base
2. L'application cliente se connecte à une instance de SQL Server en tant qu'utilisateur
3. L'application exécute ensuite **la sp\_setapprole** procédure stockée **avec un mot de passe** connu uniquement pour l'application.
4. Si le nom et le mot de passe du rôle d'application sont valides, **le rôle d'application est activé**.
5. à ce stade la connexion perd les autorisations de l'utilisateur et **assume les autorisations du rôle d'application**

### Création d'un rôle d'application

Comme n'importe quelle tâche d'administration, **la création d'un rôle d'application** peut se faire soit à travers l'interface graphique SSMS soit à travers l'instruction T-SQL décrite dans la figure ci-dessous, à noter qu'il faut **impérativement l'activer en exécutant la procédure sp\_setapprole pour permettre à l'application d'acquérir les priviléges du rôle d'application..**

La démarche de création d'un rôle d'application via le mode interface graphique ou le mode T-SQL est décrit ci-dessous :

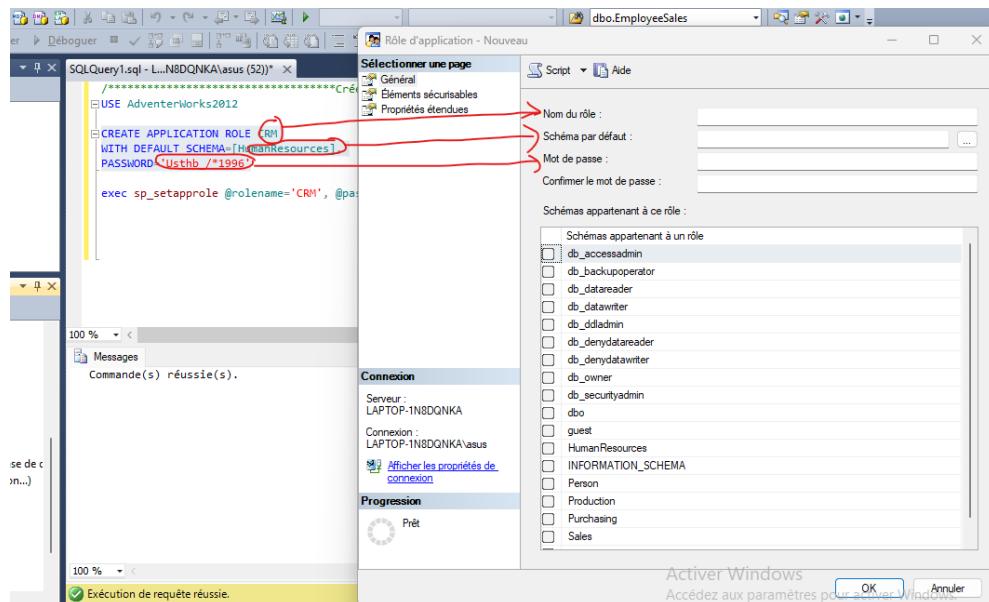


Figure 10.23: Etape 1 : Création d'un rôle d'application

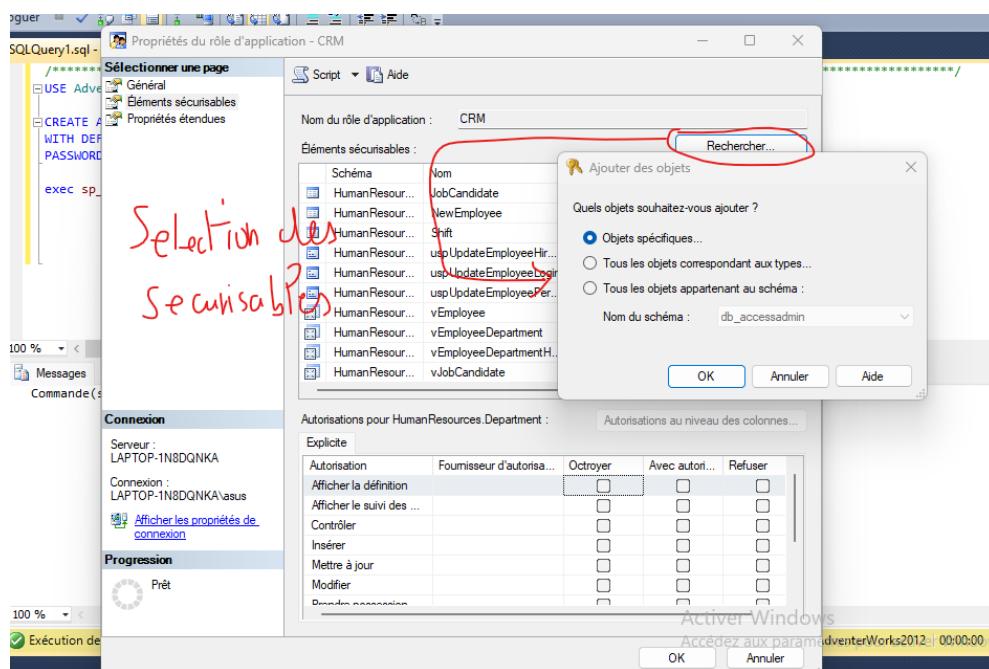


Figure 10.24: Etape 2 : Attribution des privilèges au rôle d'application

## 10.8 Contexte d'exécution des procédures stockées

### Définition

**Le contexte d'exécution** des procédures stockées fait référence à l'environnement dans lequel la procédure est exécutée, cela inclus la sécurités, les autorisations, les paramètres, les transactions, la gestion des erreurs et d'autres conditions qui influent sur le déroulement de l'exécution de la procédure stockée

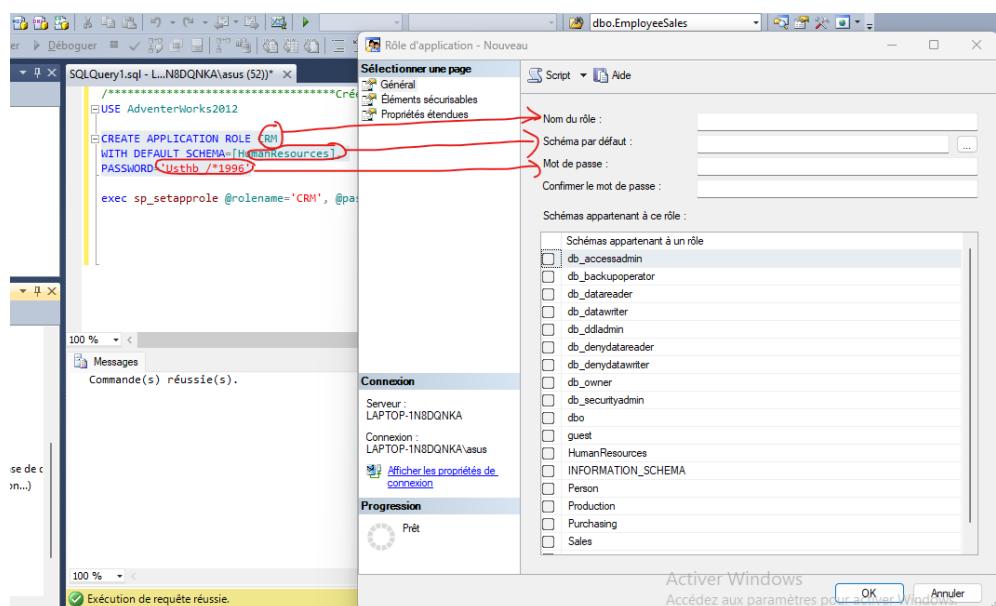


Figure 10.25: Créer une procédure en spécifiant le contexte d'exécution

Les arguments de l'option **WITH EXECUTE** déterminera le contexte dans lequel le module s'exécutera, les valeurs possibles que peut prendre l'option sont :

1. **SELF** : cela signifie que la procédure stockée s'exécutera dans le contexte de l'utilisateur qui a créé ou modifié la procédure stockée.
2. **OWNER** : Spécifie que les instructions à l'intérieur du module s'exécutent

dans le contexte du propriétaire actuel du module. Si le module n'a pas de propriétaire spécifié, le propriétaire du schéma du module est utilisé

3. **CALLER** : Spécifie que les instructions à l'intérieur du module sont exécutées dans le contexte de l'appelant du module. **L'utilisateur** qui exécute le module **doit disposer des autorisations nécessaires non seulement sur le module lui-même, mais également sur tout objet de base de données référencé** par le module.

# Chapter 11

## Automatisation des taches récurrente

### 11.1 L'outil SQL Agent

#### 11.1.1 Les caractéristiques :

1. Permet **d'automatiser les taches récurrentes** sur SQL server
2. Chaque **Agent** est rattaché à **une seule instance**
3. Il enregistre **les erreurs SQL server** dans **l'observateur des évènements Windows**
4. Il stocke **les informations sur les jobs** dans **la base de données MSDB**
5. Il est responsable de la gestion :
  - (a) **des Alertes** : c'est la remonter d'information au opérateurs.
  - (b) **des Jobs** : c'est une suite d'étapes qui s'exécute en fonction d'une planification.

(c) **des opérateurs** : représente dans la majorité des cas un groupe DBA à qui les notifications seront envoyées pour prendre les mesures proactives.

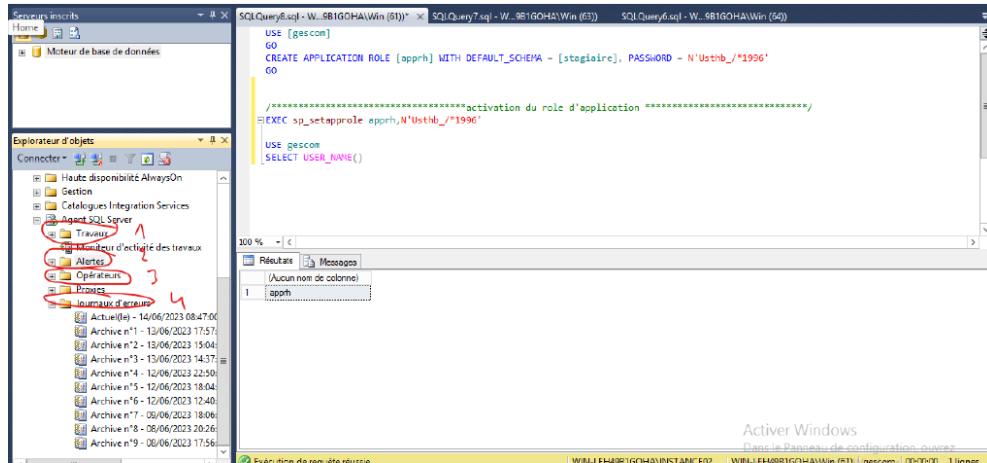


Figure 11.1: Présentation de SQL Agent

**La fonctionnalité SQL agent** se présente sous SQL server **sous forme d'arborescence de dossier**, vous allez trouver en premier **le répertoire "Travaux"** dans lequel nous allons écrire, planifier nos tâches pour les faire exécuter par le SQL Agent lui-même, en deuxième **le répertoire "Alerte"** fait référence à une fonctionnalité qui permet de définir des conditions et des seuils spécifiques, et d'être notifié lorsque ces conditions sont remplies.

Il est possible de **consulter le journal des erreurs relatif à SQL server** en cliquant sur **propriété de l'agent** comme le montre la figure ci-dessous, ou tout simplement **accéder au fichier log stocké dans le système de fichier**. (Voir la figure ci-dessous)

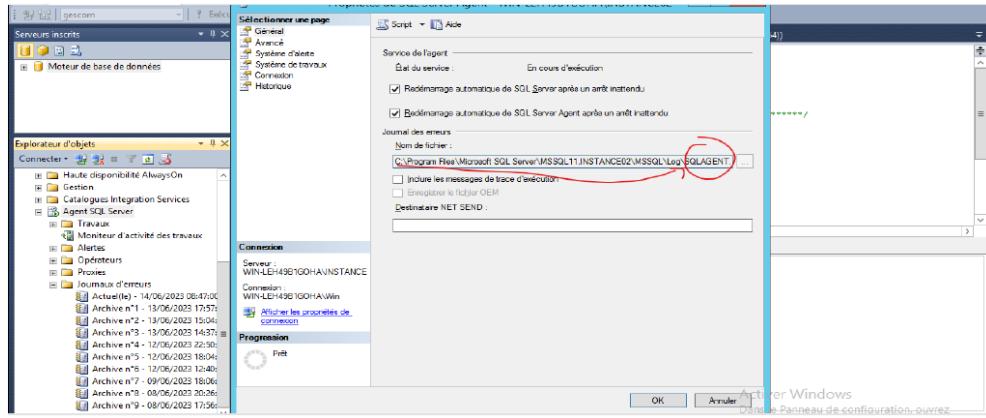


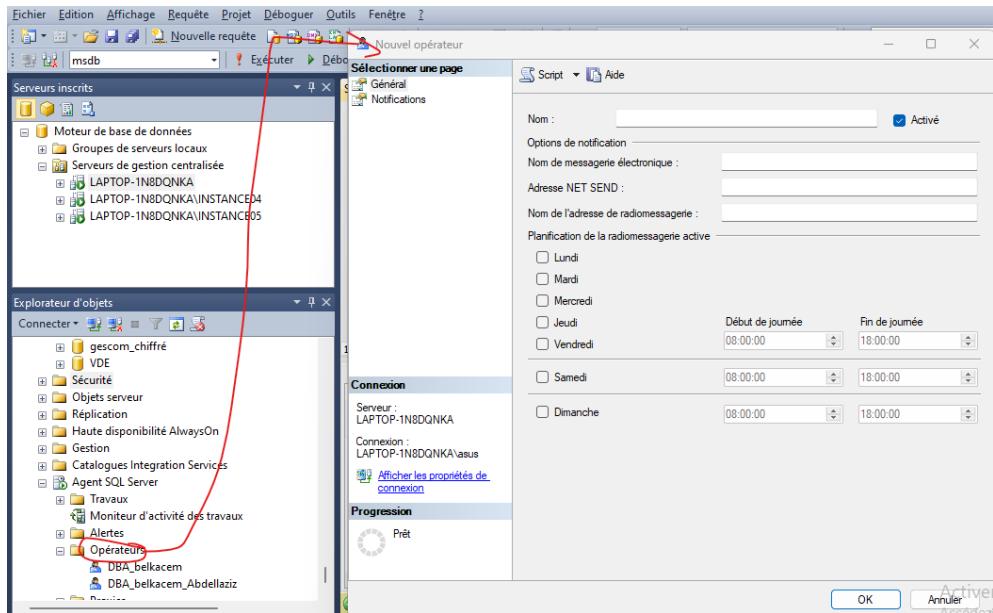
Figure 11.2: Le fichier journal des erreurs de SQL Agent

### 11.1.2 Les opérateurs

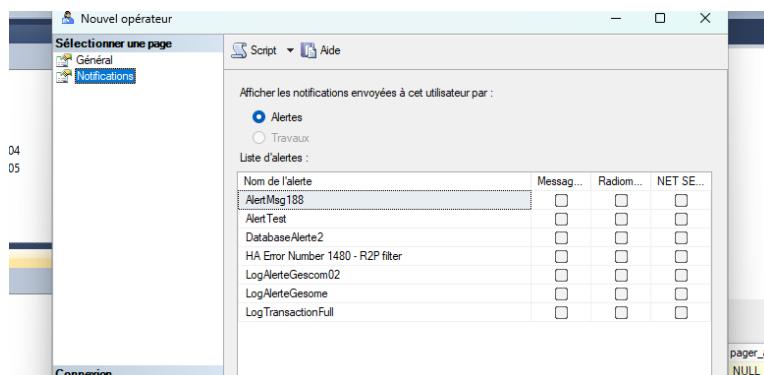
1. **Les opérateurs sont des alias** pour les personnes ou un groupe de personne qui peuvent recevoir une notification de fin de travaux ou en cas d'alerte.
2. l'outil SQL agent prends en charge **la notification des administrateurs** par le biais des opérateurs
3. Les informations sur les opérateurs sont stockés dans la base de données **”msdb”**

### 11.1.3 Création d'un opérateur

Pour spécifier à qui les alertes et les notifications de fin de travaux devraient être envoyées pour permettre à l'ensemble des administrateurs d'être en permanence au courant de ce qui se passe au niveau serveur, **il va falloir créer un opérateur.**



**Figure 11.3: Crédit d'un opérateur DBA**



**Figure 11.4: Choisir le type de notification envoyée à l'utilisateur**



### NOTE IMPORTANTE

Afin que l'opérateur reçoit bien les notifications de travaux et alertes SQL, **il doit impérativement être activé**, une fois que l'opérateur créé et activé, il est possible maintenant de mettre en place des travaux et dire que c'est cet opérateur-là qui doit recevoir les notifications.

### 11.1.4 Les Alertes

#### 11.1.5 Caractéristiques :

1. Permettent d'avertir un opérateur lorsqu'un évènement survient : **en cas d'erreur, en cas d'une panne serveur, en cas de l'atteinte d'un seuil définis, une violation de contrainte d'intégrité, une erreur d'exécution...**
2. Permettent de réaliser un traitement pour **résoudre un problème de manière proactive.** (Exp : déclenchement d'un travail suite à une alerte de gravité élevé )

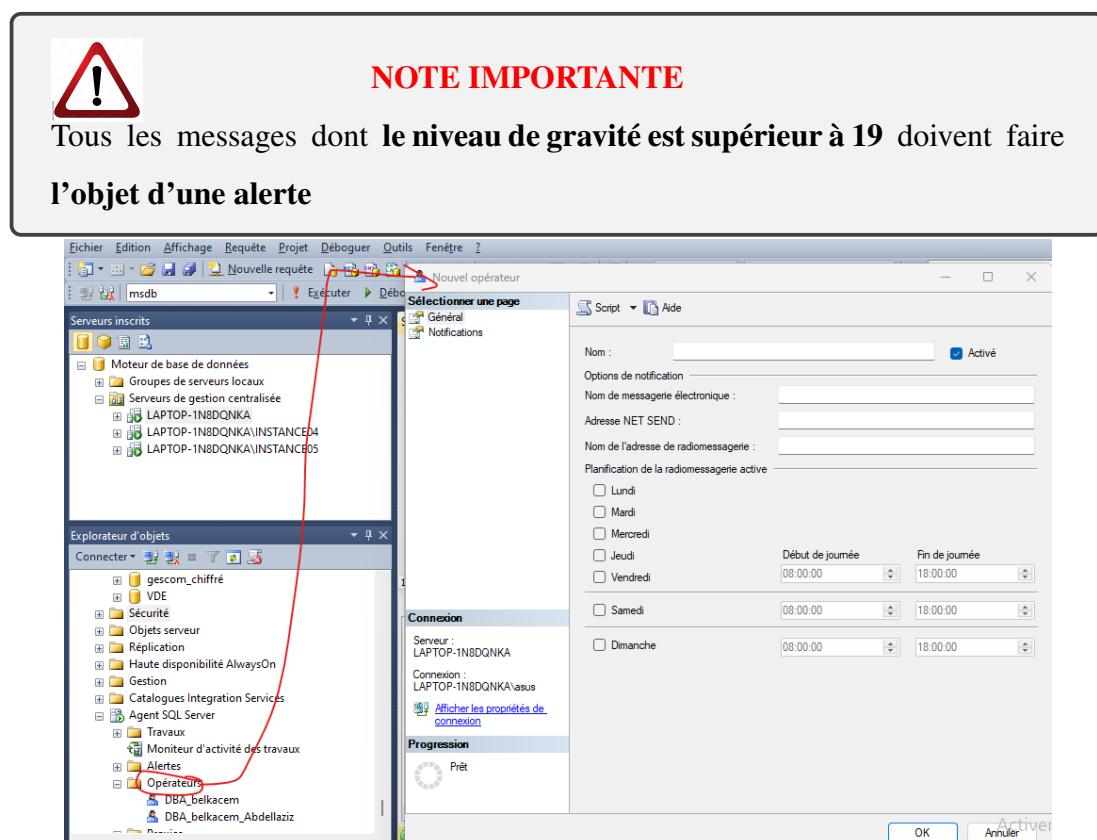


Figure 11.5: Afficher la liste des messages d'erreurs

En interrogant la table système "sysmessages", nous avons récupérer dans le jeu de résultat, **l'ensemble des erreurs prédéfinis au niveau de SQL serveur,**

la première colonne représente l'identifiant de l'erreur (clé primaire), la seconde colonne décrit le niveau de gravité, **sous Windows tous les messages dont le niveau de gravité est au dessus de 19 sont enregistrés dans l'observateur d'évènements** et enfin l'avant dernière colonne stocke la description des messages d'erreur.

On peut positionner une alerte sur chacune des messages d'erreurs stocké dans la table système. Par exemple on peut dire que si un message d'erreur dont le niveau de sévérité est supérieur à 24 se déclenche, on envoie une alerte à un opérateur pour le prévenir, autre exemple très intéressant est de récupérer le code d'erreur du message pour lequel vous souhaitez être alerté puis vous positionnez votre alerte.



#### NOTE IMPORTANTE

Pour détecter les erreurs au niveau de SQL serveur, **l'agent SQL va scanner le journal des évènements de Windows** et quand il détecte une erreur qui correspond à ce que vous avez spécifier dans votre alerte alors il va réaliser la tâche programmée

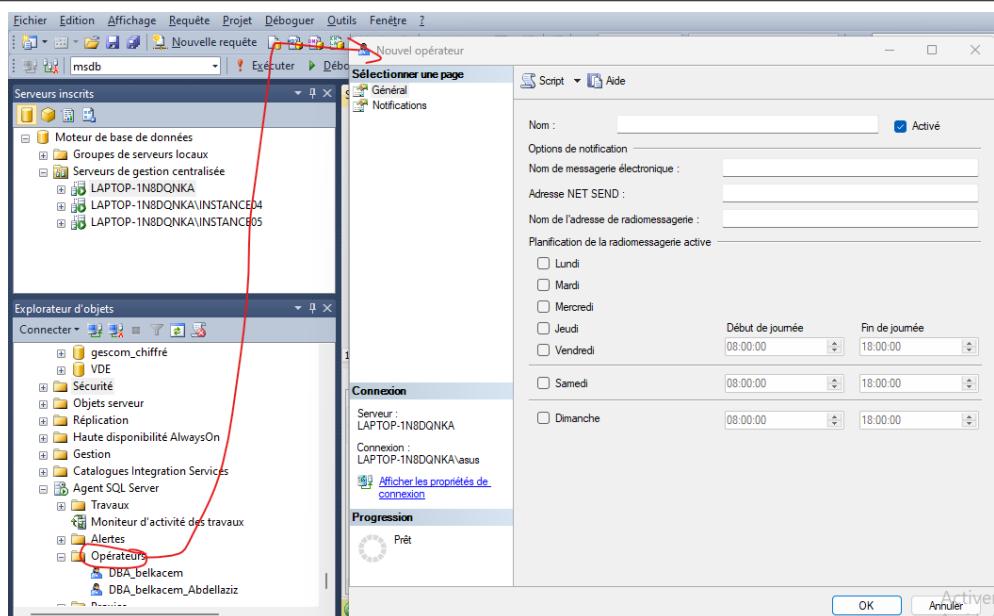
#### 11.1.6 Crédit d'une Alert

La création d'alerte peut se faire de deux façon possibles :

1. via l'interface graphique de SQL server management studio
2. Via un script T-SQL en utilisant les procédures stockées suivantes :
  - (a) **sp.add\_alert** : pour créer une nouvelle alerte
  - (b) **sp.update\_alert** : pour mettre à jour une alerte
  - (c) **sp.delete\_alert** : pour supprimer une alerte

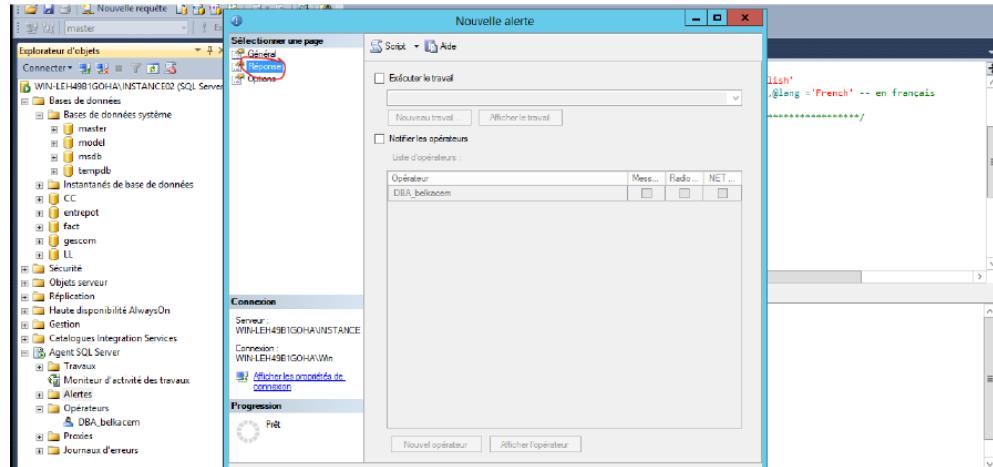

**NOTE TRES IMPORTANTE**

Il est possible de **centraliser l'ensemble des erreurs sur un serveur déporté** (serveur distant dont l'accès se fait uniquement via le réseau) en suivant le chemin suivant : **propriété agent SQL \Avancé \cocher la case transférer les évènements sur un autre serveur.** Ce qui permettra de pointer toutes les erreurs survenues dans plusieurs serveurs sur un seul serveur de gestionnaire d'erreur. (Voir la figure ci-dessous)



**Figure 11.6: Etape 1 : Création d'une Alerté SQL**

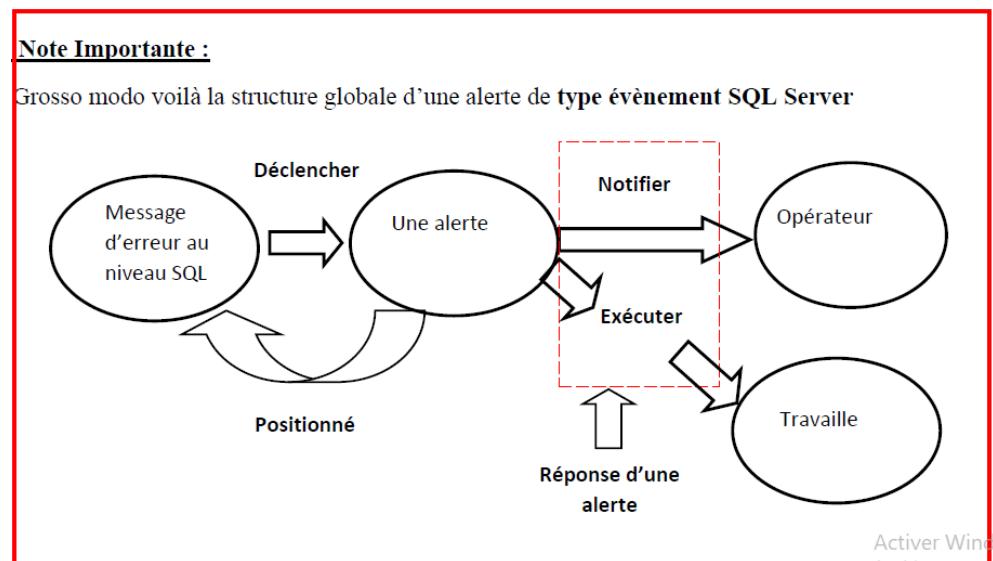
Lorsque vous allez mettre en place une alerte SQL, une fenêtre de ce type s'affiche sur laquelle **un certain nombre d'informations doivent être renseignés** à savoir : **le nom de l'alerte, le types d'alerte**, quand ce que l'alerte sera déclenché c'est-à-dire **sur quel message d'erreur je souhaite positionner mon alerte**, vous avez le choix entre le numéro d'erreur ou le niveau de gravité ou en fonction de la description du message d'erreur pour identifier l'erreur, **définir nom de la base de données** dans laquelle les événements se produisent.



**Figure 11.7: Etape 2 : Création d'une Alerte SQL**

Une fois que **l'alerte à bien été définis**, il faut **définir une réponse** pour remédier au problème, c'est-à-dire **quelle est la tâche à exécuter pour résoudre le problème** ou tout simplement notifier les opérateurs ( dba) pour dire **ATTENTION il y'a une défaillance matérielle !!!**

Pour **finaliser la création de l'alerte**, vous devez spécifier dans **"options"** **la réponse de l'alerte à la suite de son déclenchement**, dans la majorité des cas on opte pour la notification des opérateurs.



**Figure 11.8: Struture globale d'une alerte de type évènement SQL**

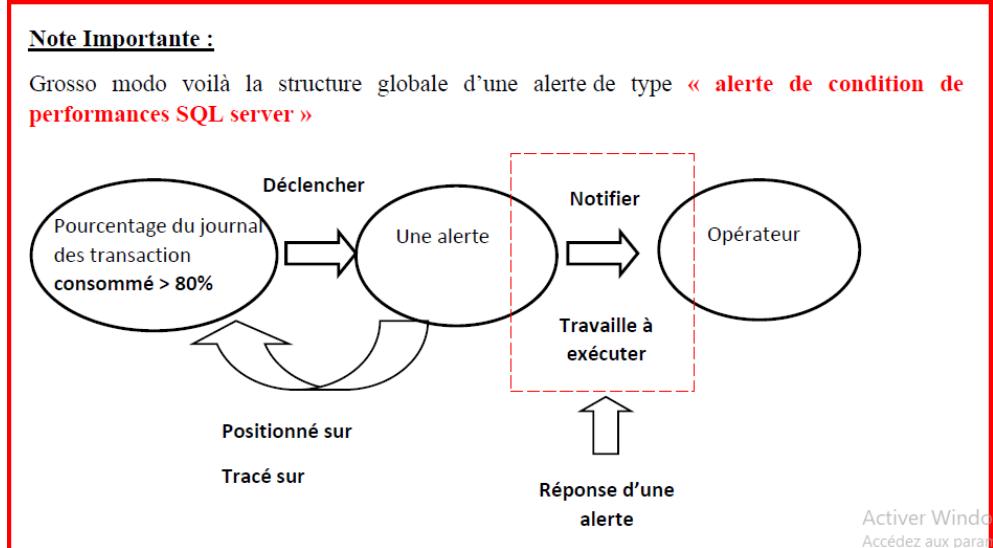


Figure 11.9: Structure globale d'une alerte sur une condition de performance



#### NOTE TRES IMPORTANTE

Il faut apprendre à utiliser les alertes sans modération, ça vous permettre de corriger tous les cas de figure possible.

### 11.1.7 Les travaux (les jobs)

1. Constitué d'une ou plusieurs étapes (tâches)
2. Deux états possibles pour une tâche : Echec ou Succès
3. Enchainement possible entre les étapes
4. Plusieurs type d'étape :
  - (a) **Transac-SQL** : utilisé par défaut, le plus courant
  - (b) **Commande système** : créer des étapes qui vont faire appelle à des commandes du système d'exploitation
  - (c) **Package** : il est possible également d'exécuter des packages SSIS
  - (d) **RéPLICATION**

(e) Langage de script

(f) Les étapes planifiées sont stockées dans la table "sysjobs" de la base de données MSDB

### 11.1.8 Création d'un travail

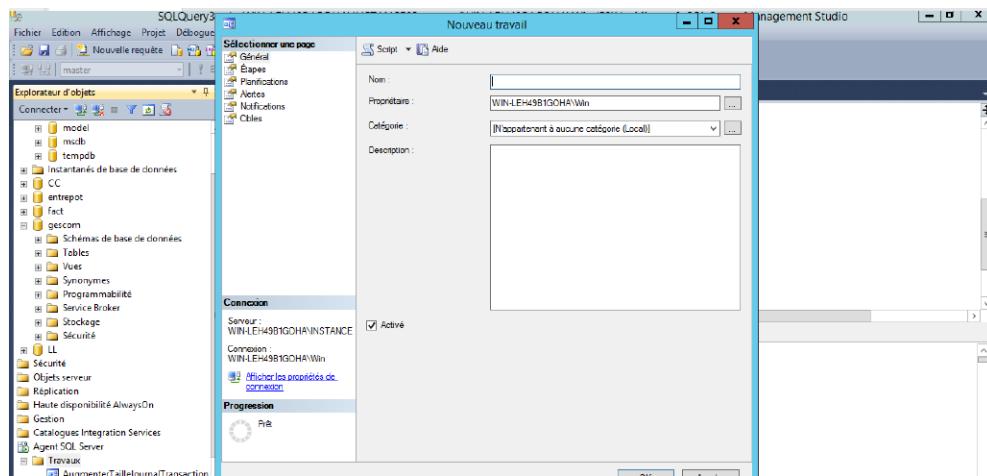


Figure 11.10: Etape 1 : création d'un travail

Dans cette figure, il faut renseigner un certain nombre d'information comme **le nom qui doit être unique** sur serveur, **spécifier une catégorie** qui permet d'organiser les travaux en fonction des opérations qu'ils exécutent, **le propriétaire de la base** qui est par défaut l'administrateur de l'instance, **on peut spécifier également si un travail est activé ou pas.**

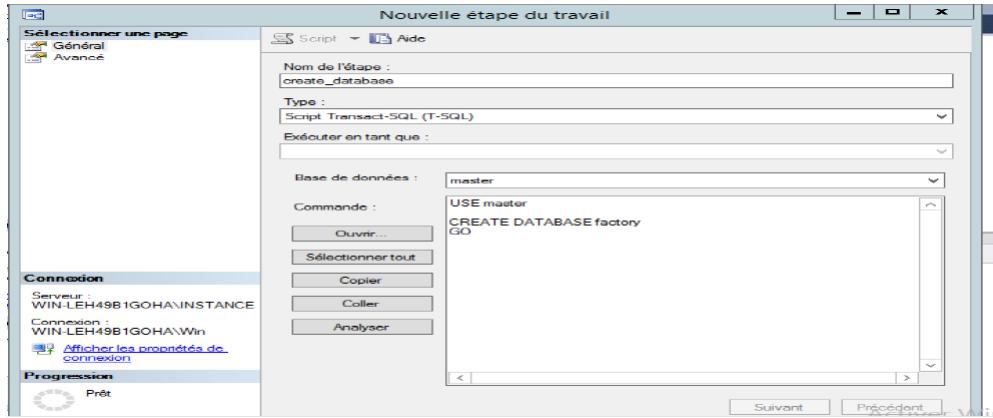


Figure 11.11: Etape 2 : création d'un travail

Dans cette figure, nous choisissons le nom de l'étape qui constitue **le travail** ainsi que son type, ici à l'occurrence c'est **un script Transact-SQL**, puis en bas vous écrivez votre script T-SQL associé à votre étape.

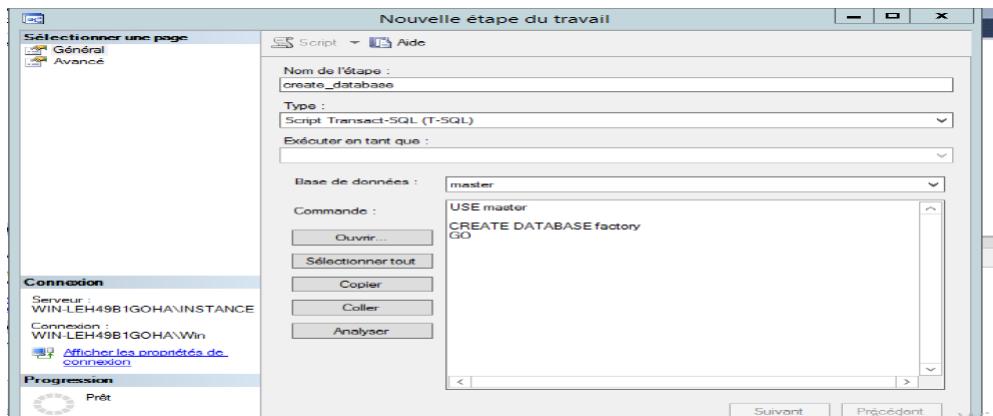


Figure 11.12: Etape 3 : création d'un travail

Dans l'onglet "avancé", vous allez spécifier **quelle est l'action à réaliser en cas de succès de l'étape**. Ici à l'occurrence nous avons dit qu'il passe à l'étape suivante, on aurait pu dire aussi de **quitter le travail en signalant l'opérateur par mail** ou de **quitter le travail en signalant l'échec**.

Vous avez également en cas où l'étape échoue, **spécifier le nombre de tentative à faire en cas d'échec** avant de quitter l'étape, **spécifier l'intervalle de reprise**

entre deux tentatives successives.

Il est possible pour chaque étape d'enregistrer ou **rediriger la sortie sur un fichier.txt** vers le système de fichier.

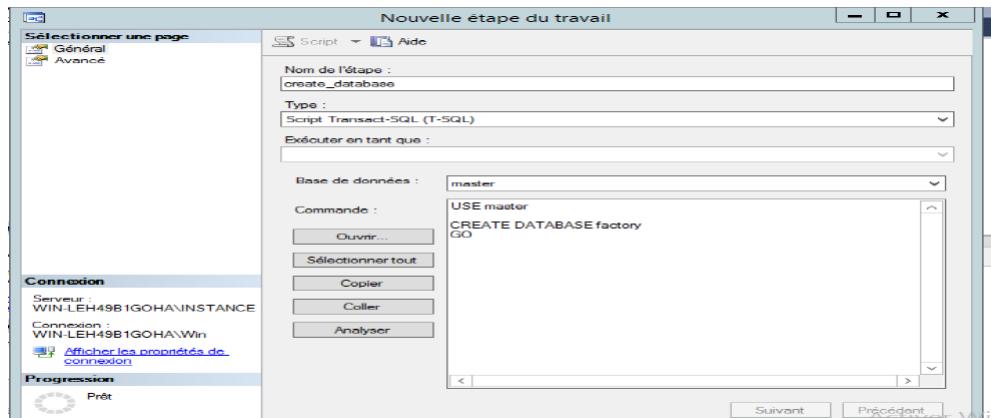


Figure 11.13: Etape 4 : création d'un travail

Dans cette figure, nous avons **définis 3 étapes d'un travail**, la première étape va servir à créer une nouvelle base de données dès que cette étape se réalise avec succès, on passe à l'étape suivante qui à son tour va servir à créer une table dans la base de données définis précédemment et une fois que cette étape se termine avec succès, le travail passe à la dernière étape dans laquelle un tuple va être insérer dans la table.

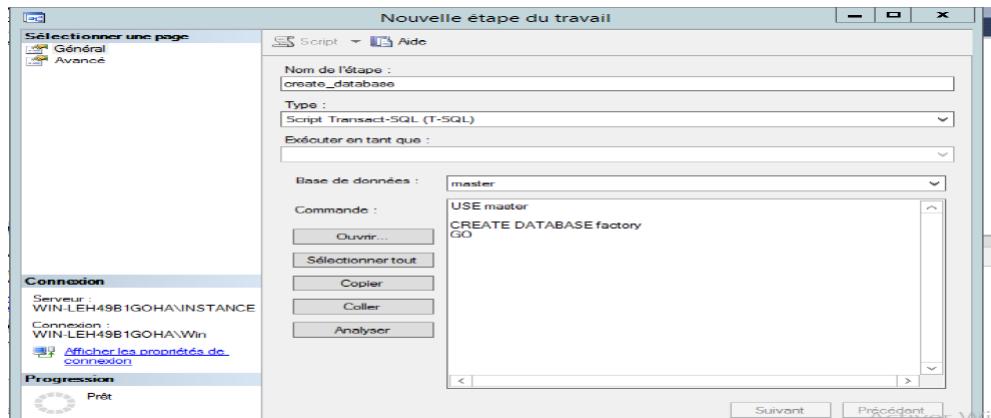


Figure 11.14: Etape 5 : création d'un travail

Cette dernière étape consiste à **planifier l'exécution du job** afin qu'il se déclenche selon d'une fréquence bien déterminée.

## 11.2 Configuration du serveur de messagerie

### 11.2.1 Caractéristiques :

1. Utilise le **protocole SMTP** pour transférer des courriers électroniques à travers un réseau informatique
2. Possède son propre processus de fonctionnement : cela évitera d'impacter le bon fonctionnement de SQL server en cas de crash survenu au niveau du serveur de messagerie.
3. La fonctionnalité interne **DataBaseMail n'est pas active** par défaut.
4. Il va falloir disposer des privilèges du rôle **serveur fixe SYSADMIN** pour configurer le serveur de messagerie.
5. Pour envoyer un mail avec le serveur de messagerie de base de données, Vous devez être **membre du rôle DatabaseMailUserRole** dans la base de données "msdb"



#### NOTE IMPORTANTE

Utiliser la procédure **sp\_configure** pour activer la fonctionnalité DataBase-Mail et pouvoir ainsi **configurer le profile et le compte SMTP**

## 11.2.2 L'assistant de configuration de la messagerie de base de données

Pour pouvoir configurer la messagerie de base de données, utiliser l'**assistant dédié à cet effet** ou tout simplement exploiter le **script T-SQL** présent dans la **liste de modèle** prédéfinis sur SQL server.

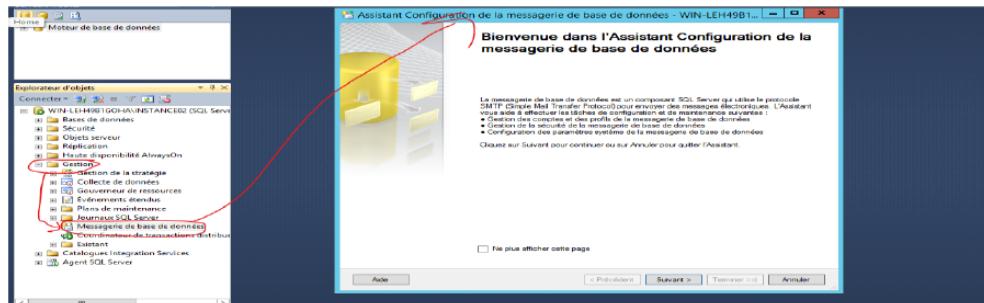


Figure 11.15: Etape 1 : Lancer l'assistant de configuration de la messagerie

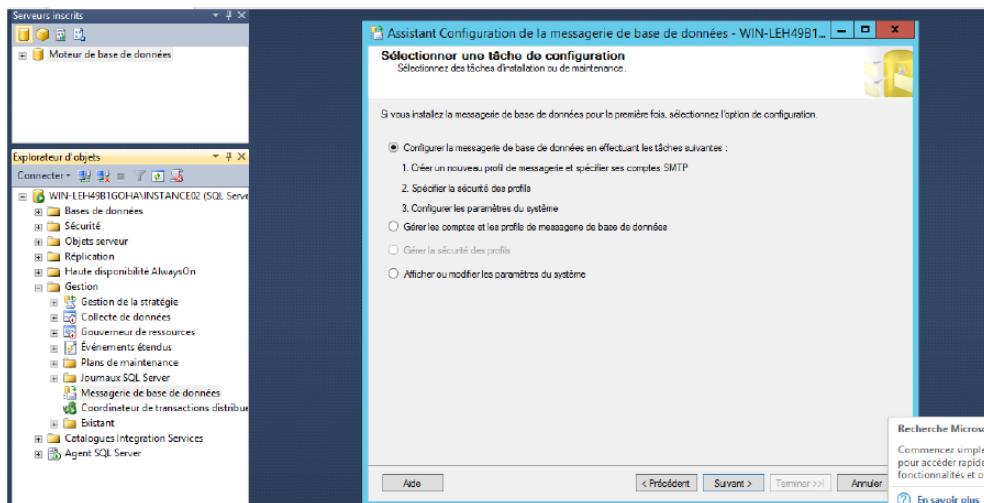
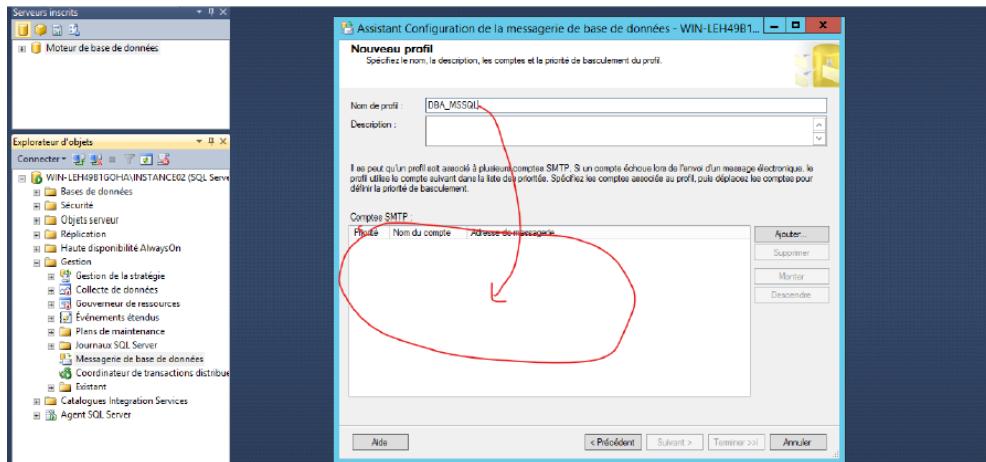
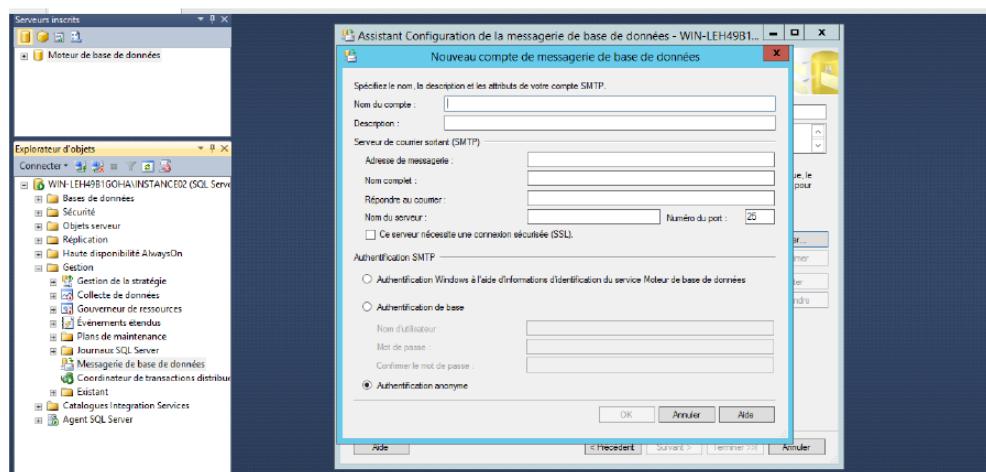


Figure 11.16: Etape 2 : Créer un nouveau profil de messagerie



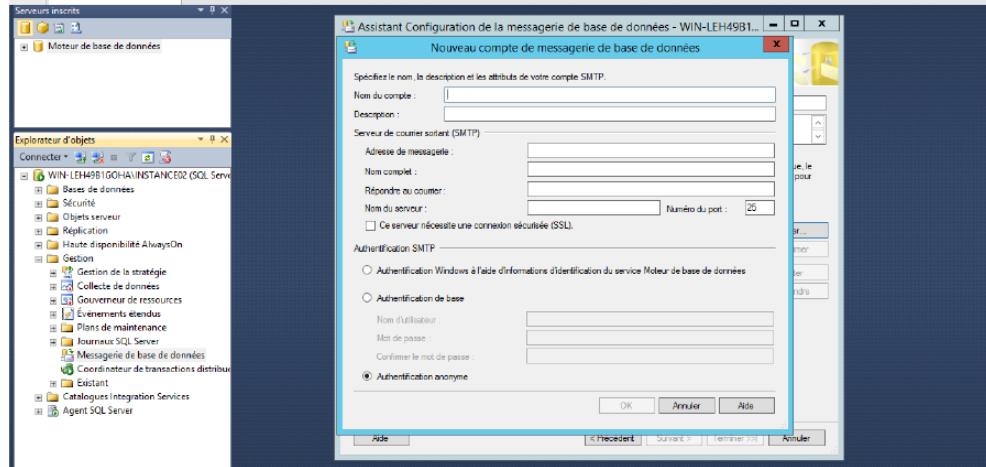
**Figure 11.17: Etape 3 : Créer un nouveau profil de messagerie**

C'est au niveau de cette étape qu'on va **spécifier le profil** ou les informations sur l'identité de l'utilisateur de la messagerie, il doit être **rattacher** obligatoirement à **un compte SMTP**.



**Figure 11.18: Etape 4 : spécifier les attributs du compte SMTP**

Ici vous allez définir un certain nombre d'informations descriptives sur le compte SMTP à savoir **le nom du serveur, l'adresse électronique, le numéro de port de communication et le mode d'authentification** de la base au serveur SMTP....

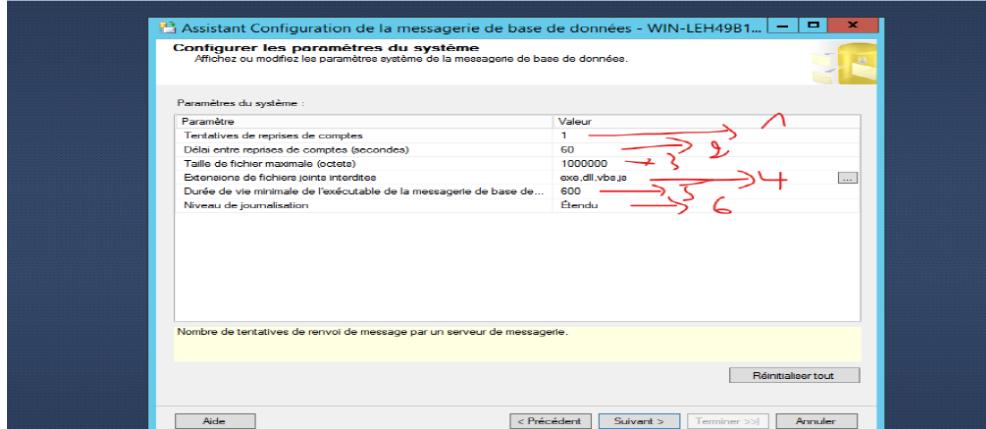


**Figure 11.19: Etape 5 : Gérer la sécurité du profile**



### NOTE IMPORTANTE

Quand la case "public" est cochée cela veut dire que **l'ensemble des utilisateurs de base de données pourront utiliser ce profile** pour envoyer les notifications et les alertes

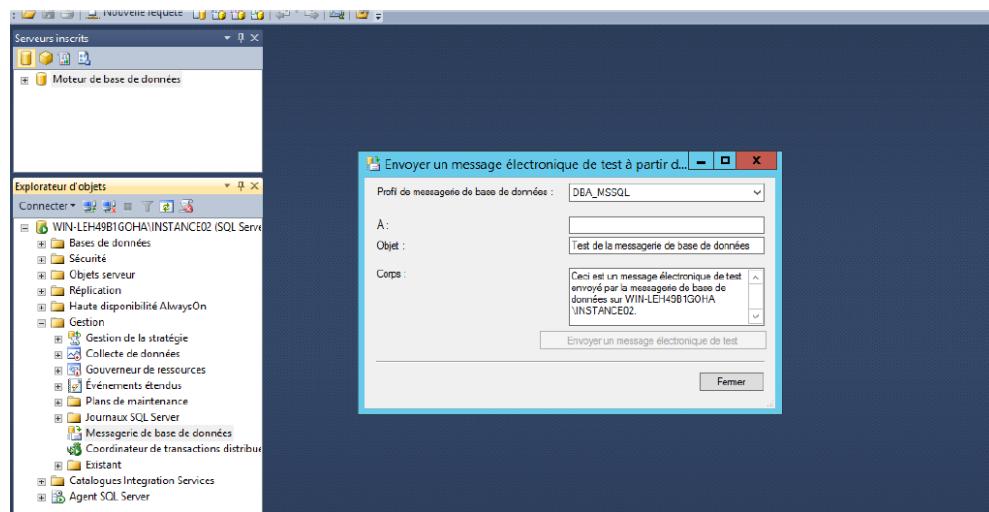


**Figure 11.20: Etape 6 : Configuration des paramètres systèmes**

Les paramètres systèmes à configurer à la fin du processus de configuration de la messagerie de base de données sont les suivants :

1. **Tentatives de reprise de comptes** : correspond au nombre d'essais que va effectuer le serveur en cas de difficultés à transmettre l'alerte.

2. **Délai entre reprise de comptes** : en indiquant 60s cela veut dire qu'il essayera de relancer tous les 60s.
3. **Taille de fichier maximale** : c'est la taille maximale qui peut être envoyée via le compte SMTP.
4. **Extension de fichiers joint interdite** : spécifier les extensions des fichiers à exclure par le serveur SMTP.



**Figure 11.21:** Etape 7 : Tester le bon fonctionnement du serveur SMTP



#### NOTE IMPORTANTE

Cette démarche de configuration de messagerie est un prérequis souhaitable pour pouvoir ensuite travailler avec SQL agent qui va gérer vos travaux planifiés.

## 11.3 Plan de maintenance

### 11.3.1 Les Caractéristiques

1. Permettent **d'automatiser la maintenance** des bases de données
2. **La liste des taches** prédéfinis dans SQL server et qu'on peut automatiser à travers les **plans de maintenances** :
  - (a) **Vérification de l'intégrité** d'une base de données
  - (b) **Réaliser le compactage** de base de données
  - (c) réaliser **la réorganisation des indexes**
  - (d) Reconstruire des indexes
  - (e) **Mise à jours des statistiques** : utilisé par l'optimiseur de requetes pour optimiser le plan d'accès aux données
  - (f) Réaliser **un nettoyage de l'historique** de l'Agent SQL server
  - (g) **Exécution d'un travail** de l'Agent SQL server
  - (h) **Sauvegarder les bases de données** (FULL, DIFF, journal des transactions ) : il est possible d'automatiser les méthodes de sauvegardes de base de données en utilisant un plan de maintenance
  - (i) Nettoyer des sauvegardes et des états
  - (j) Notifier les opérateurs
  - (k) Exécution d'instruction T-SQL
3. Stocké sous **forme de paquetage SSIS** : c'est un genre de groupe qui contient un certain nombre de taches ou d'actions, en exécutant le package cela va lancer les programmes qui ont été stockés à l'intérieur.
4. Chaque paquetage possède sa propre planification : lorsque on **crée un plan de maintenance** qui contient **plusieurs taches** et bien le moteur va

créer plusieurs paquetage SSIS et chaque paquetage SSIS va avoir sa propre planification.

## 5. l'automatisation d'un plan de maintenance passe par l'Agent SQL

### 11.3.2 Crédit d'un plan de maintenance

Pour mettre en place un plan de maintenance d'une tache quelconque, Veuillez suivre les étapes décrites dans les figures ci-dessous :

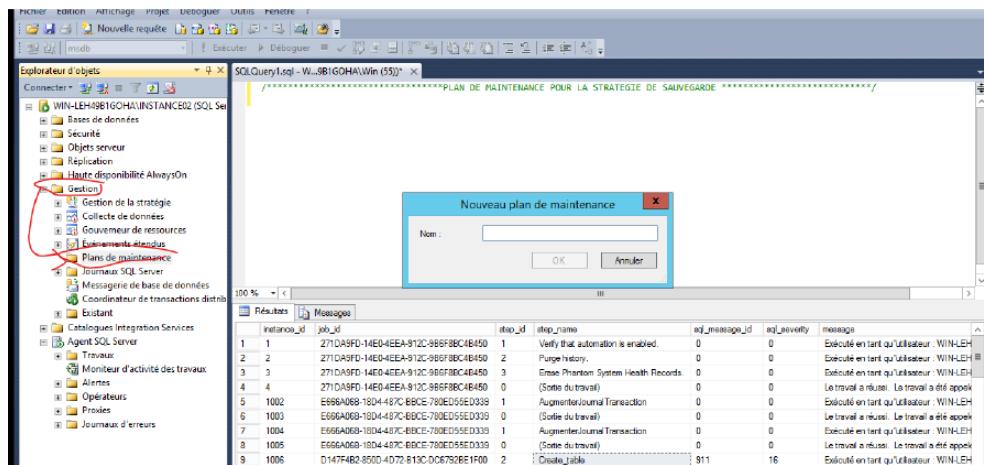


Figure 11.22: Etape 1 : Définir un plan de maintenance

Pour pouvoir définir un plan de maintenance quelconque vous devriez accéder au répertoire "plan de maintenance" en suivant l'arborescence de l'instance suivant : gestion \Plan de maintenance, faites un clic droit pour afficher l'assistant du plan de maintenance avec lequel vous allez parcourir l'ensemble des étapes essentiel pour créer un plan de maintenance.

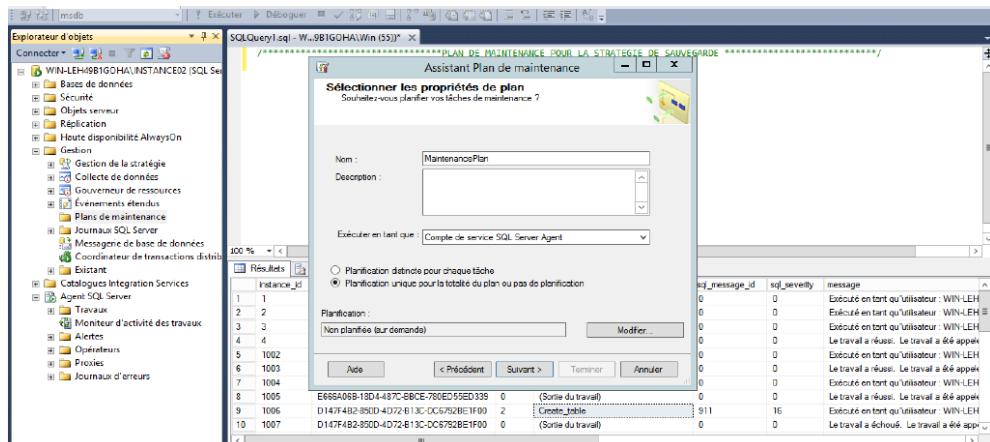


Figure 11.23: Etape 2 : Sélectionner les propriétés du plan

Dans cette étape vous allez définir, **le nom du plan de maintenance**, **la description du plan de maintenance** et aussi si on souhaite **planifier distinctement les tâches ou créer une seule planification** pour la totalité du plan de maintenance, pour notre cas nous allons choisir "planification distincte pour chaque tache" tous simplement parce que notre plan de maintenance va contenir 3 jeu de sauvegardes qui vont être lancer à des instants différents.

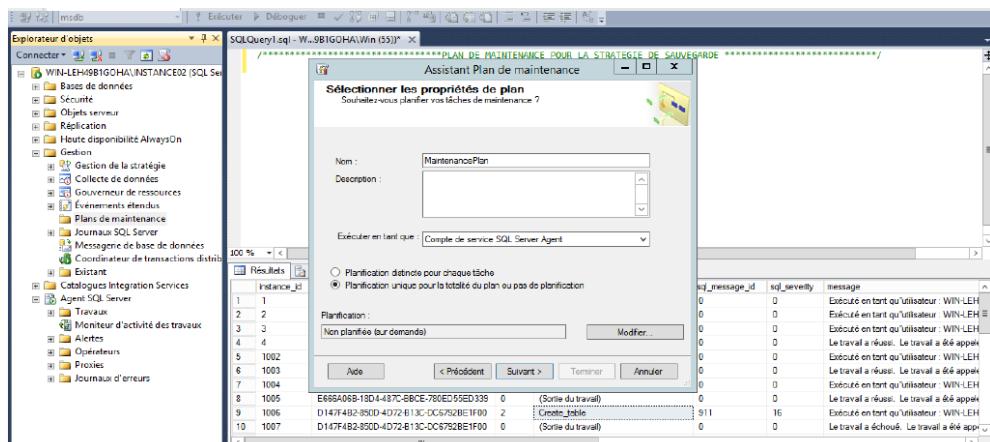
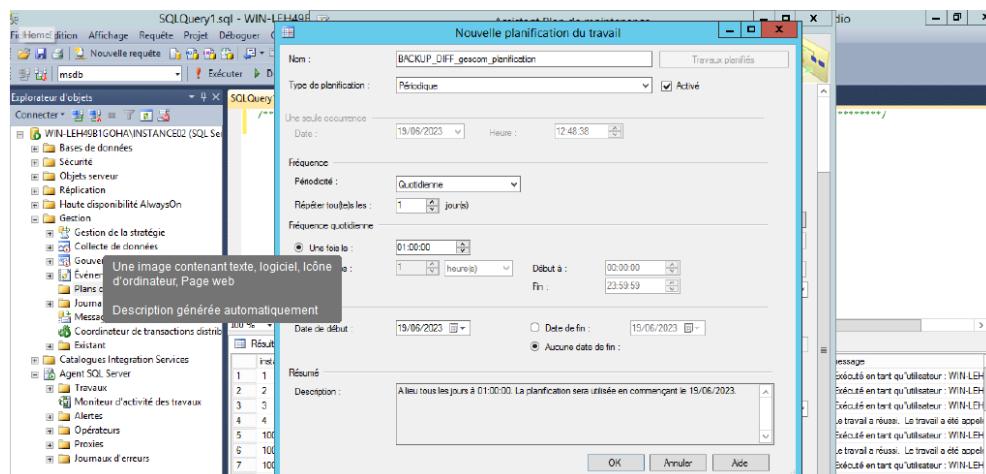
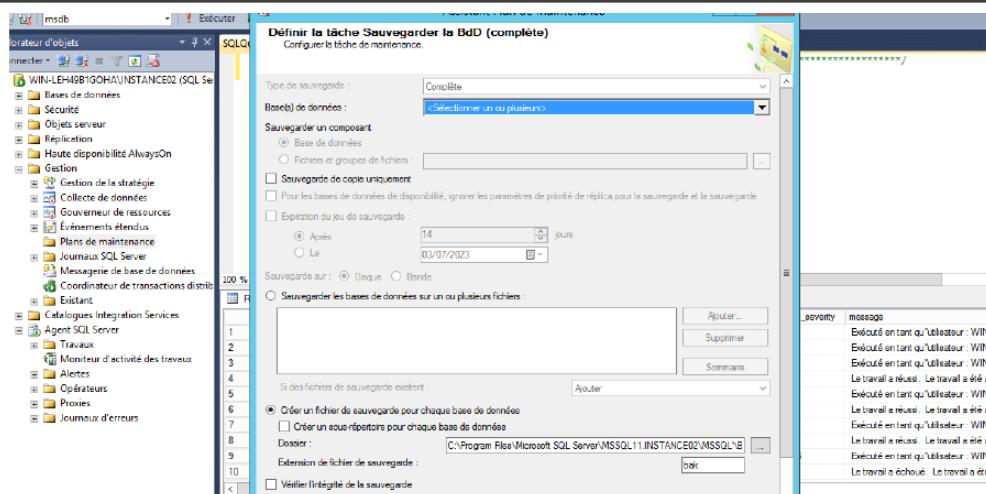
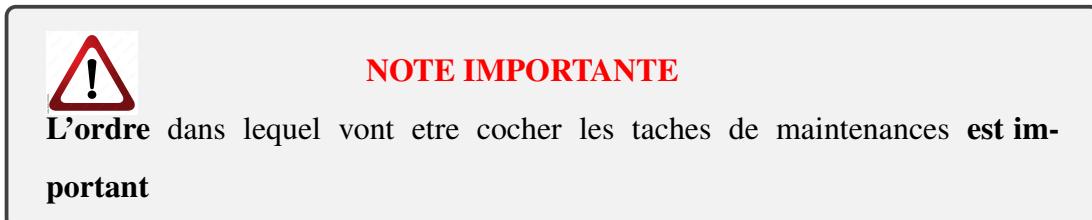
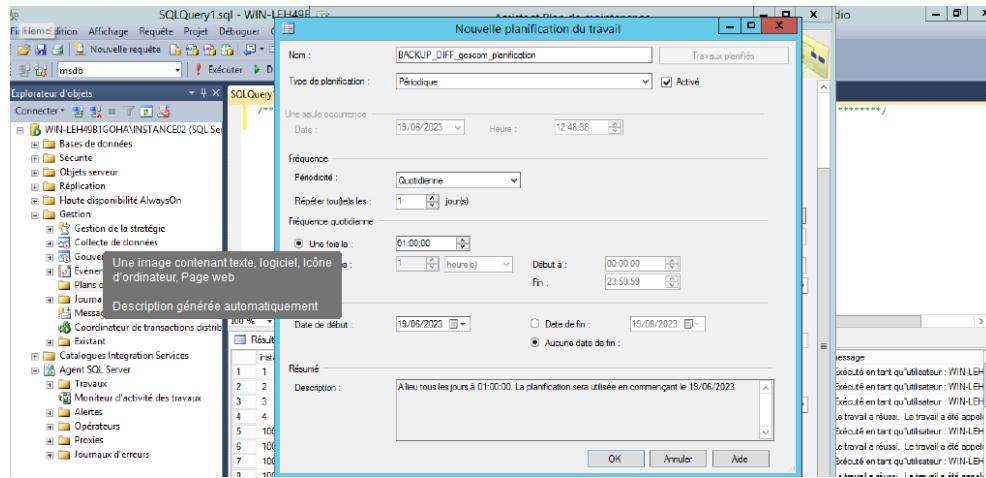


Figure 11.24: Etape 3 : Sélectionner les propriétés du plan

Cette étape nous visualise **la liste des tâches de maintenance** qui peuvent être automatisé. Vous êtes en mesure de cocher un ou plusieurs tâches de maintenance.

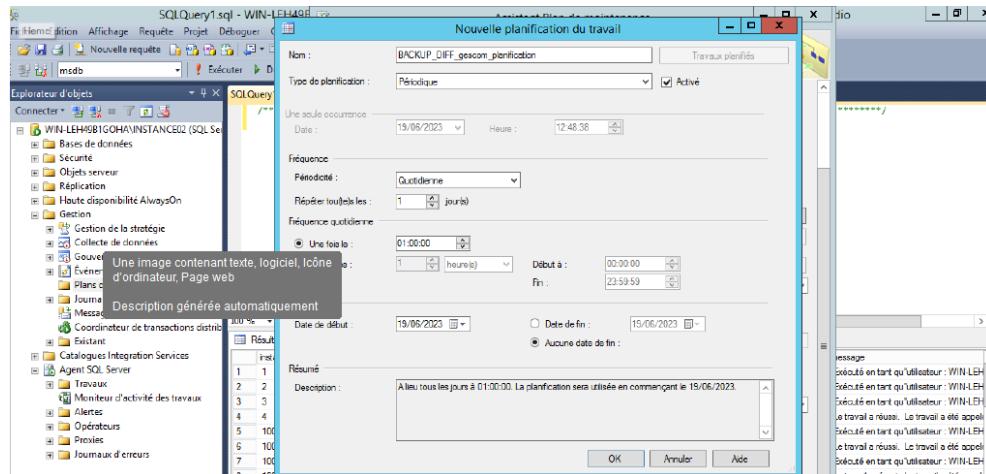


Une **planification distincte** vous donnera la possibilité de planifier chaque tâche d'un plan de maintenance individuellement, cela est très utile par exemple dans **le cadre de la mise en place d'une stratégie de sauvegarde** composé des trois méthodes exécutées à des instants différents.



**Figure 11.27: Etape 6 : Enregistrement du fichier journal du plan de maintenance**

Dans cette étape vous avez **la possibilité d'enregistrer le fichier Log** de l'ensemble des transactions relatives au plan de maintenance **dans le système de fichier**, soit de **l'envoyer via le serveur de messagerie de base de données** qui doit être au préalable configuré !!!!, aux différents opérateurs.



**Figure 11.28: Etape 7 : Affichage de l'historique du plan de maintenance**

Dans cette figure, nous avons visualisé **l'historique du plan de maintenance** après avoir finaliser la planification des différentes tâches d'entretiens, vous avez la possibilité de **modifier les tâches**, par exemple définir une nouvelle fréquence d'exécution d'un BACKUP, **renommer, supprimer et exécuter le plan** etc....

Autre information très importante est que la liste des taches vont être enregistré également en tant que travail de type package integration services SSIS afin qu'ils puissent être gérées par l'agent SQL.

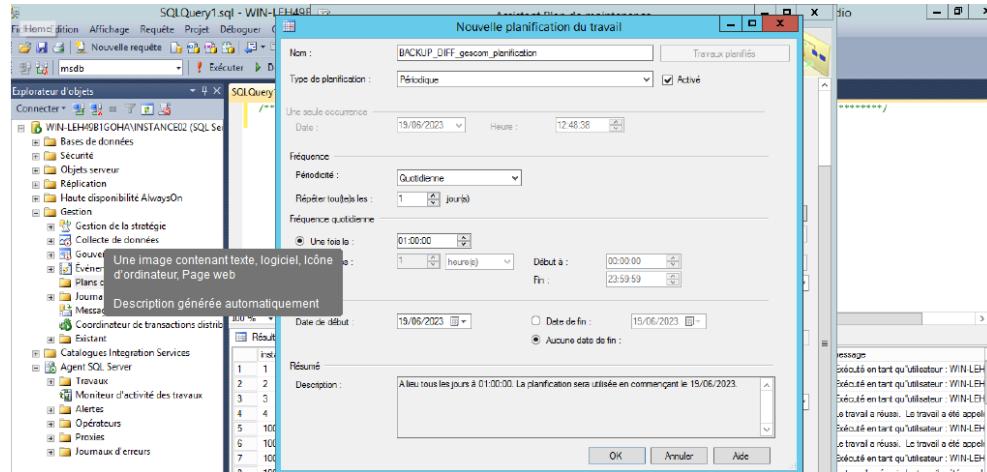


Figure 11.29: Etape 7 : Le fichier journal du plan de maintenance

Il est opportun d'avoir ce réflexe d'aller consulter le **fichier journal du plan de maintenance** pour confirmer que les taches du plan se sont déroulé correctement.



#### NOTE IMPORTANTE

Une fois que le **plan de maintenance soit mis en place**, n'attendez pas que la planification se lance pour confirmer le bon déroulement du plan, **il faut faire le test de manière unitaire**



#### NOTE IMPORTANTE

le **plan de maintenance** SQL server contient un certain nombre de taches stockés sous forme de paquetage SSIS. Dans la majorité des cas pratique, on mis en place un plan de maintenance qui stocke les taches suivantes : **Vérification de l'intégrité, la reconstruction des indexe, la mise à jours des statistiques et la stratégie de sauvegarde à chaud.**

# Chapter 12

## Audit des environnements SQL server

Les audites peuvent inclure **les catégories suivantes d'actions** :

1. **Niveau serveur** : Ces actions incluent des opérations de serveur, telles que des modifications de gestion et des opérations d'ouverture de session et de fermeture de session.
2. **Niveau Base de données** : ces actions incluent les opérations DML et DDL
3. **Niveau Audit** : Ces actions incluent des actions appartenant au processus d'audit

### 12.1 Audit au niveau Serveur

1. L'objet SQL server audit va permettre de **tracer des actions et des groupes d'actions au niveau serveur** comme par exemple : **CREATE DATABASE**, **DROP DATABASE...** etc, il est possible de tracer un certain nombre d'actions dans le fichier de trace de l'audit

2. L'audit s'effectue au **niveau de l'instance SQL server**.
3. Possibilité d'avoir plusieurs audits par instance SQL server : on est en mesure de **tracer une action de type BACKUP RESTORE, un audit qui va tracer tous les logins d'authentification en échec ou un audit qui trace le changement de mot de passe** des logins au niveau serveur.
4. **Par défaut** l'audit est dans **un état désactivé**

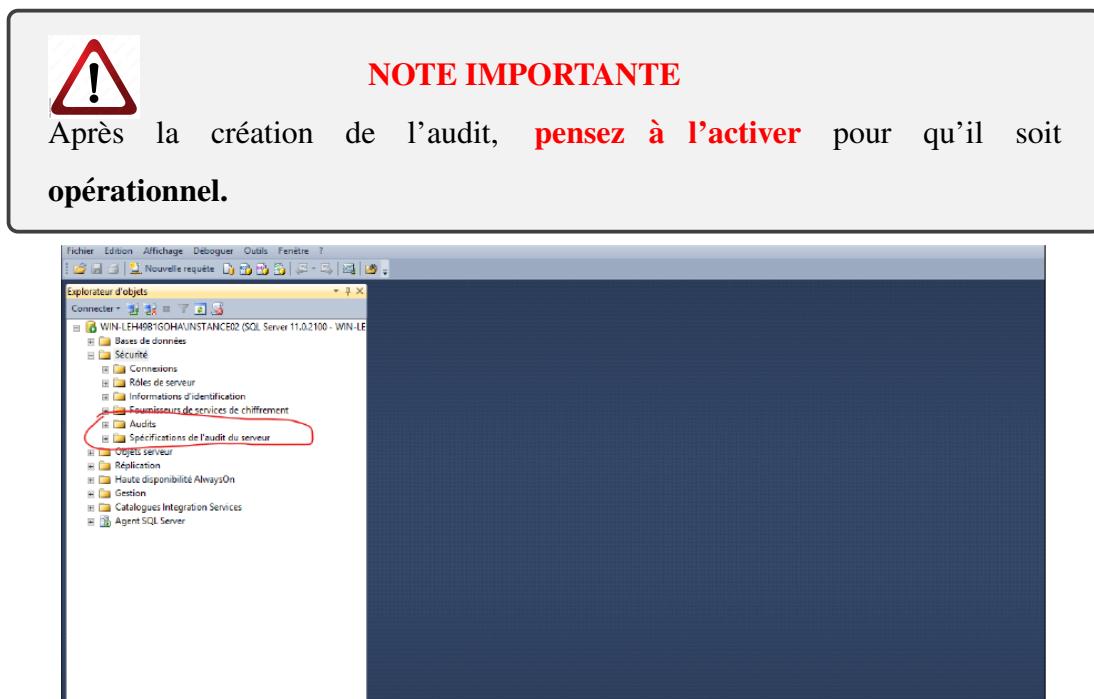


Figure 12.1: Emplacement de l'audit dans l'explorateur d'objet

Dans cette figure nous pouvons distinguer **deux répertoires** utiliser dans le **cadre de l'audit**, le premier **"audits"** va permettre de **définir la cible** c'est-à-dire où va être stocker **les caractéristiques des traces**. Le deuxième **"spécifications de l'audit du serveur"** va permettre de contenir **les évènements, les actions et groupe d'action que vous souhaitez tracer**. Voila grossso modo comment s'organise la mise en place d'un audit.

### 12.1.1 Cration de l’Audit

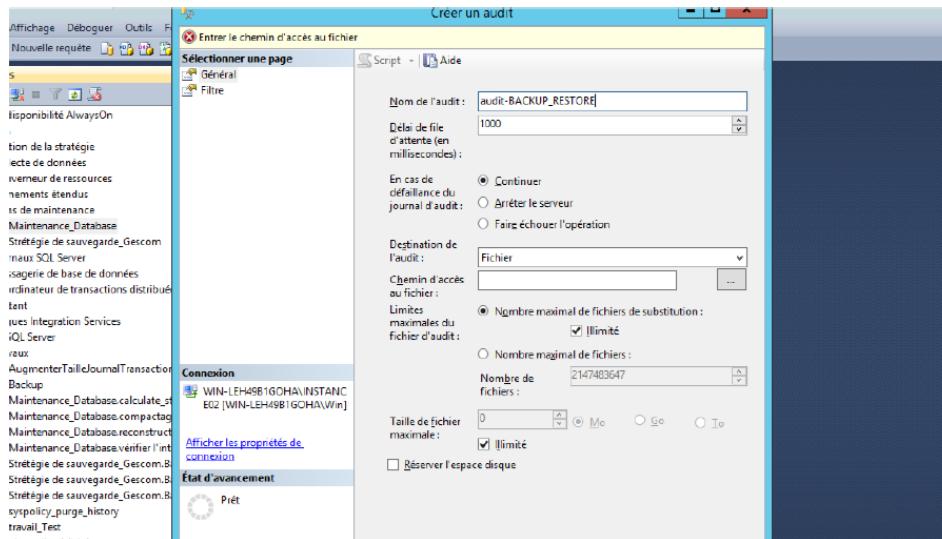


Figure 12.2: Cration de l’audit au niveau serveur

Cette interface nous offre **plusieurs champs** dont on a besoin de renseigner, vous avez **le nom de l’audit** à spécifier, **le délai de file d’attente** qui va indiquer la durée qui va s’écouler avant que le traitement des audits sera forcer, une valeur 0 indique une remise synchrone, **en cas de défaillance du journal d’audit** que dois-je faire, **la destination de l’audit** généralement choisie comme fichier de sortie stocker dans le système de fichier.

### 12.1.2 Specification de l’audit de serveur

Dans le deuxième répertoire, nous avons créé **une nouvelle spécification de l’audit du serveur** en indiquant dans le premier champ **le nom de la spécification**, **le fichier cible** de l’audite dans le second champs et enfin **sélectionner les évènements ou groupe d’action** à tracer puis stockés dans le fichier de destination.

On peut également **visionner le journal de l’audit** comme le montre la figure

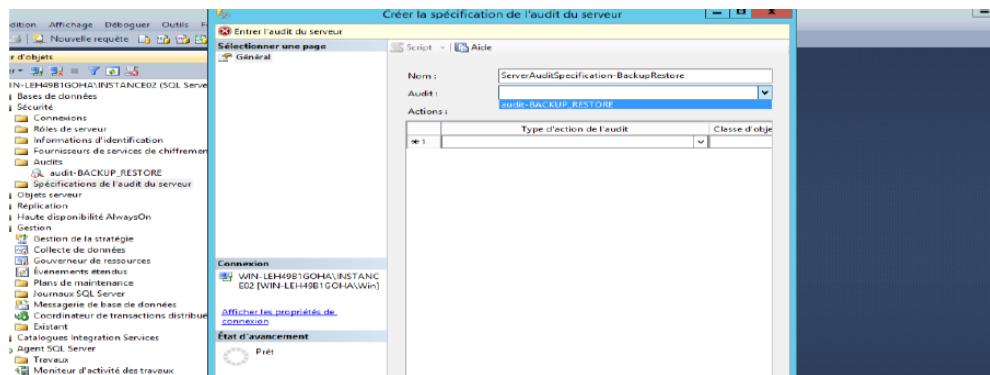


Figure 12.3: Créer une spécification de l'audit

ci-dessous :

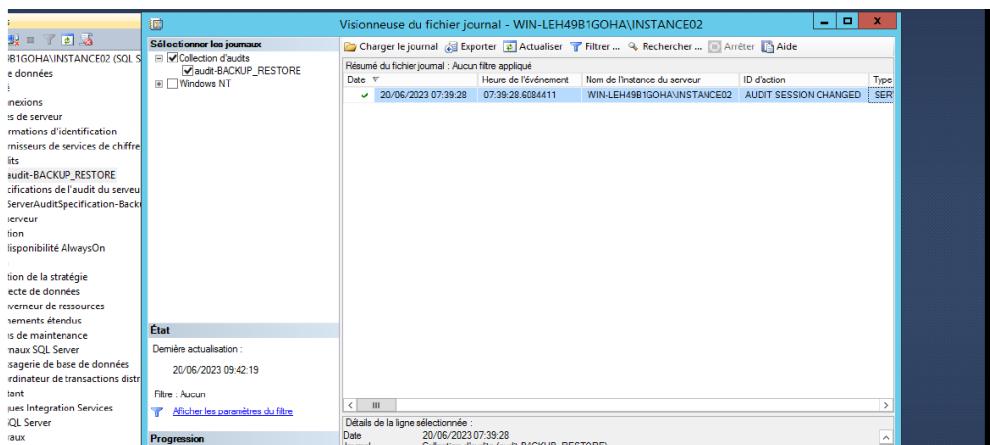


Figure 12.4: consulter le journal de l'audit

Dans **la liste des évènements tracés par l'audit**, il se peut que vous allez rencontré des évènements **tracés implicitement** par le système.

## 12.2 Audit au niveau base

1. L'audit s'effectue au niveau de la base de données
2. Possibilités d'avoir plusieurs audits par base de données
3. **Par défaut** l'audit créé est **désactivé**



### NOTE IMPORTANTE

1. Toujours une spécification d'audit est liée à **un fichier d'audit**
2. **L'audit générale** fait référence au **procédure de contrôle de la gestion d'une entreprise ou d'une organisation** par contre **l'audit de sécurité** se concentre sur **l'évaluation des contrôles de sécurités**

#### 12.2.1 Création d'audit au niveau de la base de données

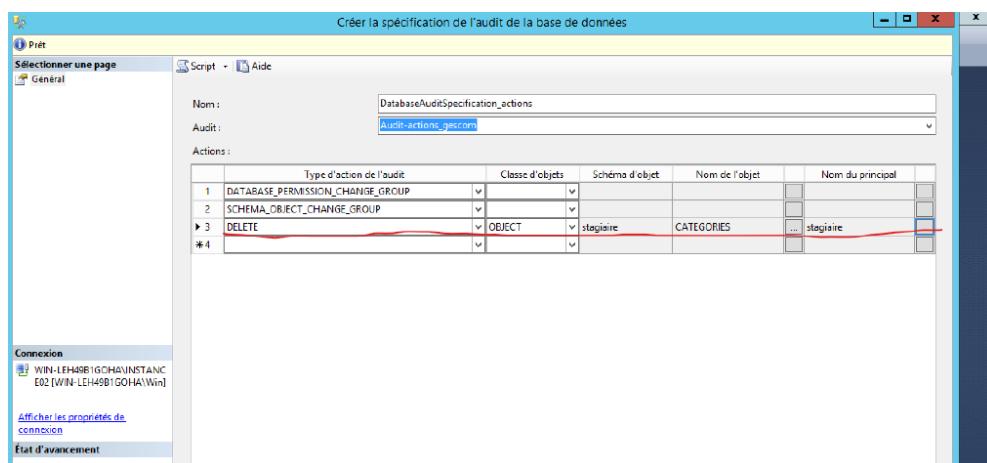


Figure 12.5: Ajouter une spécification de l'audit au niveau base de données

**La première action** va être tracé au niveau de la base de données "gescom" pour **toute instruction de modification de droit de type GRANT, REVOKE, DENY** attribués un à utilisateur.

**La deuxième action** va être tracé au niveau de la base de données "gescom" lorsqu'un **événement de type CREATE, ALTER, DROP** est réalisé au niveau des objets du schéma "stagiaire".

la dernière action consiste à **tracer le lancement de l'instruction DELETE** sur la base de données "gescom", sur la table catégorie qui se trouve dans le schéma

”stagiaire” à partir de l’utilisateur connecté ”stagiaire”



#### NOTE IMPORTANTE

Pour éviter que SQL server soit une boite noire pour l’administrateur, il est préférable **régulièrement d’activer ces traces** pour voir ce qui se passe sur le système, mais faite **ATTENTION** de spécifier un nombre important de trace pouvant **rapidement saturer votre FS**. Il est **IMPORTANT** de cibler les actions sensibles de votre organisation

### 12.3 L’outil SQL profiler

#### 12.3.1 Présentation de l’outil SQL server Profiler

- Permet de capturer des traces de l’activités de la base de données, détecte les problèmes au niveau système.
- Utilisé en corrélation avec l’analyseur de performance Windows pour l’analyse de problèmes de performance.
- Plusieurs **modèles** de traces **sont prédefinis** : il est recommandé de **repartir avec un modèle de trace prédefinis** et de le personnalisé !! et de l’adapter selon le besoin.
- Permet de **capturer une charge de travail** pour **la rejouer ailleurs** : Dans le cadre du Benchmaking, il peut etre utile de capturer la charge de travail d’un serveur source et de la rejouer ailleurs dans un serveur cible (**Analyse comparative des performances**)
- Permet de faire de **la corrélation avec d’autres trace** au niveau système



### NOTE IMPORTANTE

L'utilisation de SQL profiler est précieux dans de nombreux scénarios notamment : **l'optimisation des performances, le débogage des requêtes lentes, l'identification des goulets d'étranglement** du système et la compréhension du comportement des applications

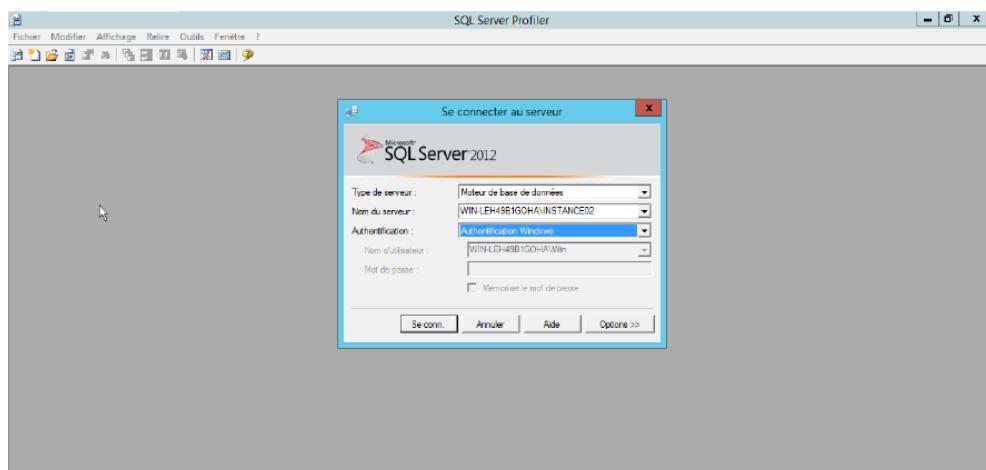


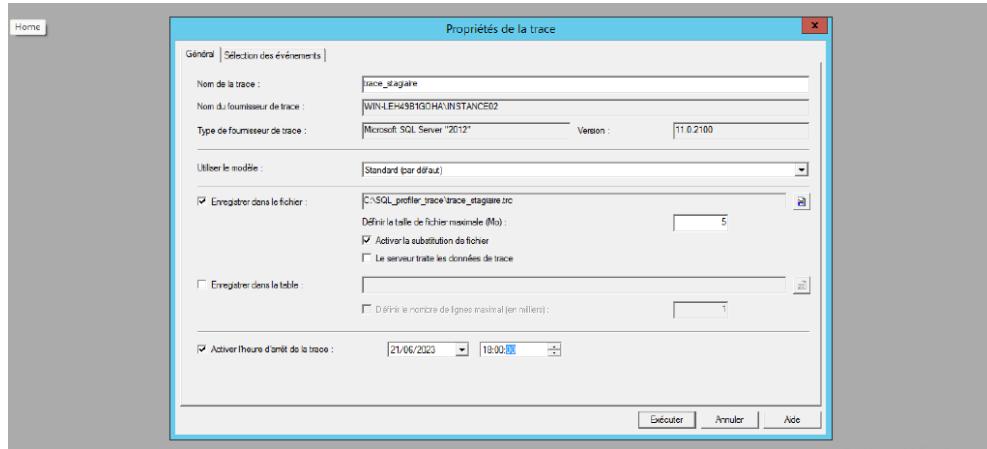
Figure 12.6: Ouverture d'une session de traçage

Pour **démarrer une nouvelle session de trace SQL profiler**, vous disposer des solutions suivantes :

1. Utiliser **le raccourci CTL+ALT+P** pour démarrer une session SQL profiler avec le contexte d'exécution actuelle
2. A partir du **menu démarrer** de votre système d'exploitation
3. A partir de **management studio** au niveau de la **barre d'outil** de la fenêtre active, sélectionner "Outils"

Avant de manipuler les fichiers de traces, il faut au préalable **se connecter à l'instance** sur laquelle vous souhaitez créer vos fichiers de traces.

Une fenêtre de dialogue de connexion s'affiche pour **introduire les informations de connexion** en utilisant le mode d'authentification Windows ou SQL server



**Figure 12.7: Introduire les caractéristiques du fichier de trace**

Une trace sous SQL server profiler est caractérisé par son nom, le nom du serveur sur lequel sera définis le fichier de trace, vous avez également plusieurs modèles de trace, les plus utilisé sont "**Tuning**" et "**TSQL Replay**" ce dernier est principalement utiliser pour capturer la charge de travaille ensuite la rejouer ailleurs dans un autre serveur afin de tester les impacts, vous pouvez enregistrer la trace dans un fichier ou dans une table, vous allez définir aussi la taille du fichier de trace qui doit être choisie soigneusement pour enregistrer tous les traces à des fins d'analyse (500 Mo par exemple).



#### NOTE IMPORTANTE

Il n'est pas recommandé d'enregistrer la trace dans une table mais plutot dans un fichier de trace ".trc" comme ça vous ne dépendais pas d'une base, vous aurez une maitrise totale de l'espace de stockage

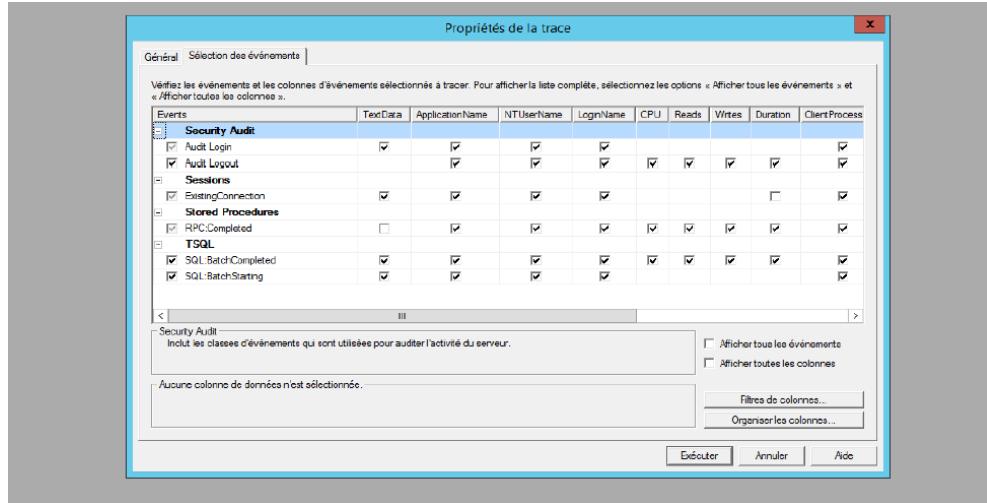
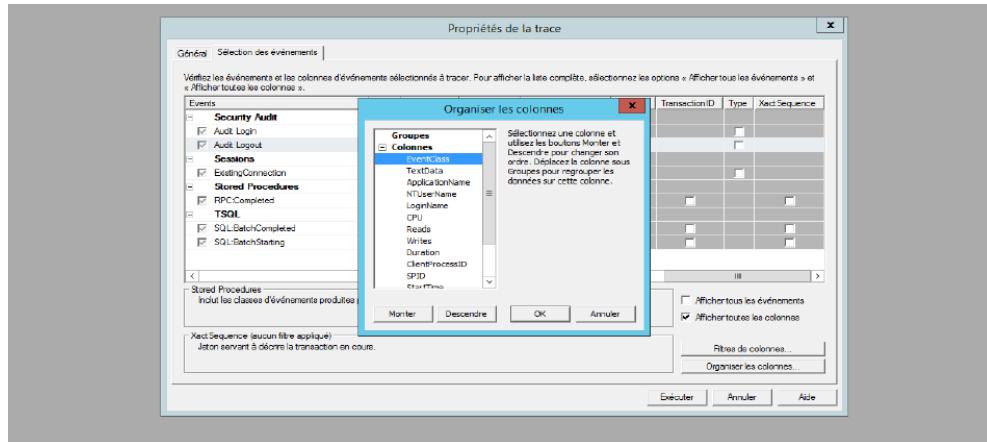


Figure 12.8: Sélectionner la classe et les attributs d'évènements

**TRES IMPORTANT**, c'est ici qu'on va avoir **l'ensemble des événements** qu'on souhaite **capturer sur l'instance** en fonction du modèle de trace , vous avez **les évènements de type T-SQL** c'est à dire l'ensembles des instructions qui s'exécute dans une fenêtre de requêtage ainsi nous pouvons **tracer les attributs d'événements** associés à savoir **le login de connexion qui a lancé cette action**, **la consommation des ressources CPU** causé par l'exécution de cette instruction, vous avez également **la durée d'exécution**, quand est ce que la commande s'est lancé et quand est ce qu'elle s'est terminée.

Il est possible de **rajouter dans le modèle** spécifié et pour **chaque type d'événements**, **les colonnes d'évènements** que vous souhaitez tracer.

Par ailleurs, **organiser et personnaliser l'affichage** est également possible de l'ensemble des colonnes d'évènements que vous souhaitez auditer au niveau serveur.



**Figure 12.9: Personnaliser l'affichage des colonnes d'évènements**

Il est possible de **filtrer sur des colonnes d'événements**, cela est équivalent à **l'instruction LIKE** qui permet de dire seul les requêtes provenant d'un l'utilisateur données pourront être tracés.

**Figure 12.10: Le résultat de traçage des évènements sélectionnés**

Ce fichier de trace nous **affiche un certain nombre d'évènements** qui se sont produits au niveau de **l'instance INSTANCE02**, on retrouve tous les **colonnes d'événements audités**. Au niveau de la première colonne, on remarque la présence de l'instruction T-SQL qui a été exécuté par login de connexion "hamid", nous pouvons voir également **à partir de quel environnement cette instruction a été lancée, la durée d'exécution, le nombre de bloc 8ko** qui ont été lus, vous avez

le moment à partir duquel l'instruction a été lancé, vous avez le "HostName" dans laquelle l'évènement s'est produit.

En d'autre terme ce fichier de trace a **loggé tous les évènements qui se sont produit sur l'instance** notamment les évènements de type T-SQL qui ont été lancés à partir du login de connexion "hamid"



#### NOTE IMPORTANTE

L'administrateur de l'instance est en mesure **d'arrêter la session de traçage en cours** d'exécution, **récupérer le fichier de trace** à partir du système de fichier pour **le lire de nouveau à l'aide de l'outil SQL server profiler** pour une **analyse ultérieur**.



#### NOTE IMPORTANTE

Le SQL server profiler est un outil très utilisé par les DBA pour pouvoir **rejouer la charge** de travail sur un autre serveur, **capturer des requêtes et des transactions qui posent des problèmes** en termes de performance, ce qui permettra d'analyser de manière très fine ce qui se passe sur notre système.

La connaissance et la maîtrise de l'outil SQL server profiler est **indispensable pour pouvoir diagnostiquer les problèmes** que vous pouvez rencontrée au niveau d'un environnement SQL server.

Dans le cas où vous allez **rencontrer de grosses requêtes** qui posent des problèmes, sans l'utilisation de l'outil SQL server profiler, **la détection de l'anomalie peut prendre 2 à 3 jours** au revanche avec l'utilisation de cette outil **le problème sera détecter en espace de quelques minutes** en faisant une corrélation avec d'autres compteur de performances

### 12.3.2 Relecture d'un fichier de trace

La relecture est **la possibilité d'ouvrir un fichier de trace enregistré et de le relire**. La relecture est très pratique pour **résoudre les problèmes d'application ou de processus**. Une fois la correction apporté, faite relire le fichier de trace sur l'application ou le processus qui causé problème. **Relisez ensuite la trace d'origine et comparer les résultats**



#### NOTE IMPORTANTE

Des **classes d'évènements doivent être capturés pour permettre la relecture**. Ces classes d'évènements sont capturés par défaut si vous utilisez **le modèle de trace TSQL\_Replay**

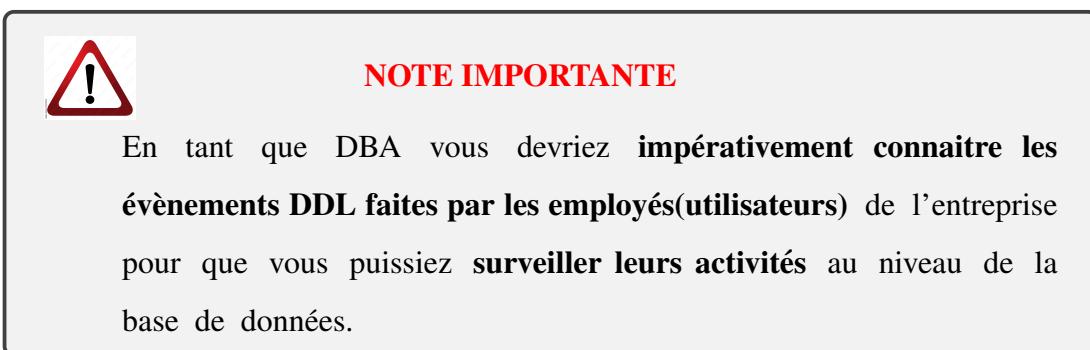
Pour plus de détails sur la relecture. **Veuillez consulter le Support Technique.**

## 12.4 Les déclencheurs DDL

### 12.4.1 Caractéristiques

1. La notion de déclencheur est **un outil puissant** qui va nous permettre de tracer les instructions qui **agissent sur la structure** de la table comme **CREATE, ALTER, DROP**
2. Très **intéressant** dans le cas où vous travailler dans une base de données en production pour **sécuriser votre environnement**.
3. Permet de **créer un déclencheur sur les opérations CREATE, ALTER, DROP** table—base— login

4. Permet de **limiter les opérations DDL même si un utilisateur dispose des droits** nécessaires : par exemple vous voulez **sécuriser votre environnement de production** et vous voulez même si l'utilisateur fait partie du rôle serveur "sysadmin" ne puisse pas supprimer ces objets, **vous pouvez créer des triggers qui vont annuler l'ordre de suppression** tel que le DROP et afficher un message d'erreur à l'utilisateur.



#### 12.4.2 Crédit d'un trigger DDL au niveau base

```

CREATE TRIGGER log_DDL_user
ON DATABASE
FOR DDL_TABLE_EVENTS -- sur les événements de type DDL effectués sur la table ( vous êtes mesure de le faire sur une base, LOGIN --USER)
AS
BEGIN
    INSERT INTO journal (date_heure,utilisateur,événement,instruction,context)
    VALUES (GETDATE(),
            CURRENT_USER,
            EVENTDATA().value('/EVENT_INSTANCE/EventType')[1] , 'varchar(2000)' ,
            EVENTDATA().value('/EVENT_INSTANCE/TSQLCommand')[1] , 'varchar(2000)' ,
            EVENTDATA())
    print(' récupération des informations relatives à l''utilisateur a été faite avec succès')
END
-- GETDATE() permet de récupérer la date actuelle du système
-- CURRENT_USER() permet de récupérer le nom de l'utilisateur qui a lancé l'instruction DDL
-- EVENTDATA () est une fonction qui permet de récupérer les informations relatives à l'événement qui se produit sur l'instance
-- ainsi que le type d'événement CREATE ALTER DROP
CREATE TABLE aa ( num int )
DROP TABLE aa
SELECT *
FROM journal
    
```

Identifiant	Date_heure	Utilisateur	Événement	Instruction	Contexte
1	2023-06-21 14:25:23.580	dbo	CREATE_TABLE	CREATE TABLE aa (num int)	<EVENT_INSTANCE>:PrintType:CREATE_TABLE</EventTyp...<EVENT_INSTANCE>:PrintType:DROP_TABLE</EventTyp...
2	2023-06-21 14:25:42.387	dbo	DROP_TABLE	DROP TABLE aa	

Figure 12.11: Crédit d'un trigger DDL EVENT

Ce trigger illustré dans la Figure 12.11 est très utile pour extraire les informations lier à un utilisateur qui a déclenché un évènement DDL qu'on appellera les **EVENTDATA**, ce qui permettra à l'administrateur d'être notifié de cette modification de structure de la base.



#### **NOTE IMPORTANTE**

Les triggers sont très intéressants à utiliser dans un environnement en production surtout pour logger **les EVENTDATA liées aux évènements de type DDL CREATE, DROP, ALTER et pour interdire à des utilisateurs** bien spécifiques de **COMMITER les événements DDL**

# Chapter 13

## La surveillance sous SQL server 2012

### 13.1 Travailler avec le moniteur de performance

#### 13.1.1 Caractéristiques

- Permet de rassembler ou de **collecter les données de surveillance**.
- Permet de déceler **les conditions qui peuvent affecter les performances** tel que les IO disque, La consommation mémoire, le temps CPU.
- Fonctionnalité mis à disposition par Windows.
- Il est possible de **rajouter ses propres compteurs de performances**.

#### 13.1.2 Les compteurs de performances standard

Il y'a un certain nombre de composants **à surveiller systématiquement** sur vos serveurs de base de données pour **identifier les goulets d'étranglements** :

1. **lecture et d'écriture physique** (IO Disque) : Surveiller les opérations d'entrée/sortie du disque pour éviter les goulots d'étranglement due à des temps d'accès disque élevés
2. **L'utilisation du processus CPU** : La surveillance de la charge CPU peut détecter les pic de consommation qui pourrait indiquer **le besoin d'optimisation des requêtes ou d'augmentation de la capacité matérielle**
3. **La consommation de la mémoire tampon** : Assurez vous que **la mémoire physique est suffisamment disponible** pour éviter des ralentissement due à une utilisation excessive de la mémoire virtuelle

**Les compteurs standards** peuvent être ajoutés facilement à travers **l'analyseur de performance** mis à disposition par Windows. La figure ci-dessous vous montre comment ajouter un compteur **en cliquant sur le signe +** après avoir choisi le **nom du serveur** que vous souhaitez monitorer, collecter les données de performances

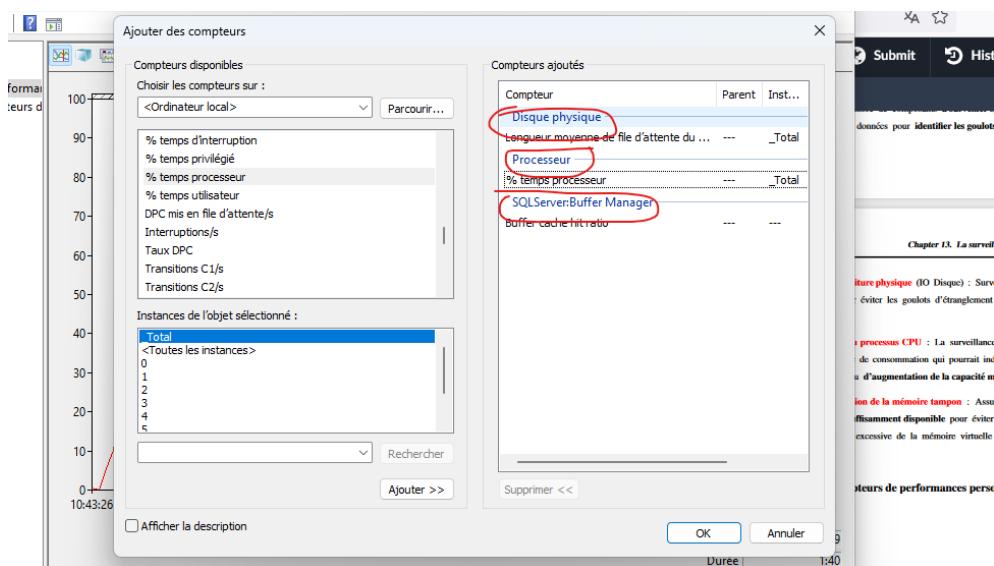


Figure 13.1: Ajouter les compteurs standards de performances

Pour **chacun des compteurs sélectionnés**, il y a un paramètre prédéfini qui nous permis de collecter les données de performance à partir du serveur de base de données.

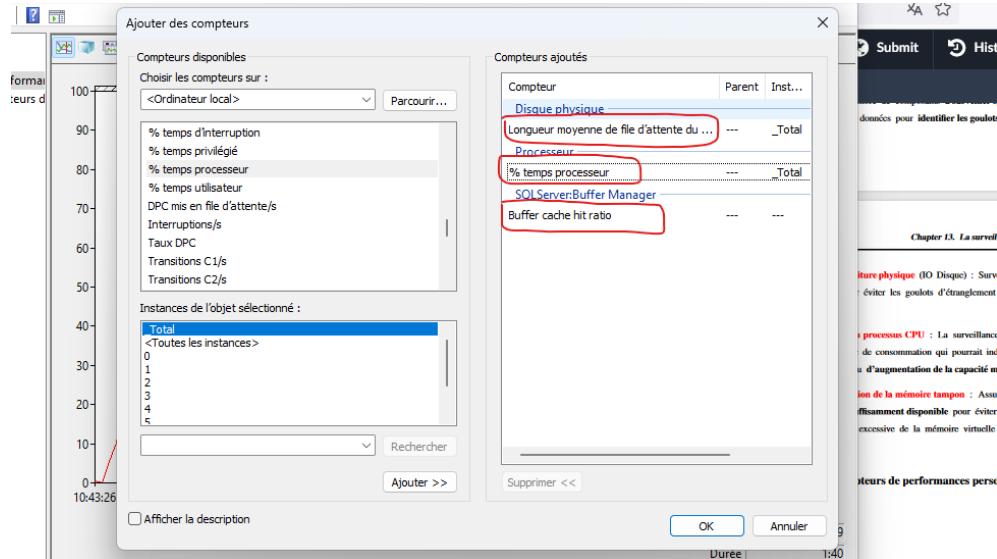


Figure 13.2: Les paramètres des compteurs de performances

1. **La longueur moyenne de file d'attente** : Représente **le nombre moyen de demande** de lecture et écriture **placé en file d'attente** pour le disque sélectionnée, **un nombre important indique la présence d'un goulot d'étranglement** sur une entrée/sortie disque.
2. **Le temps processeur** : le pourcentage du temps processeur ne doit iméprativement pas dépasser **les 58%** sinon un problème CPU est enregistré.
3. **Buffer cache hit ratio** : représente le pourcentage trouvée dans les pages de pool de mémoire tampons sans avoir à effectuer de lecture à partir du disque. **Un taux d'accès au cache tampon inférieur à 90% indique une baisse de performance**



### NOTE IMPORTANTE

**Le Buffer hit ratio** est très important car il nous **indique si notre cash SQL server est bien dimensionner** pour pouvoir gérer l'ensemble de vos données

**Un autre compteur** peut être ajouté à la liste des compteurs à surveiller régulièrement pour **garder un oeil** sur l'état de santé de la base de données comme le **Data File Size** qui concerne directement la database.

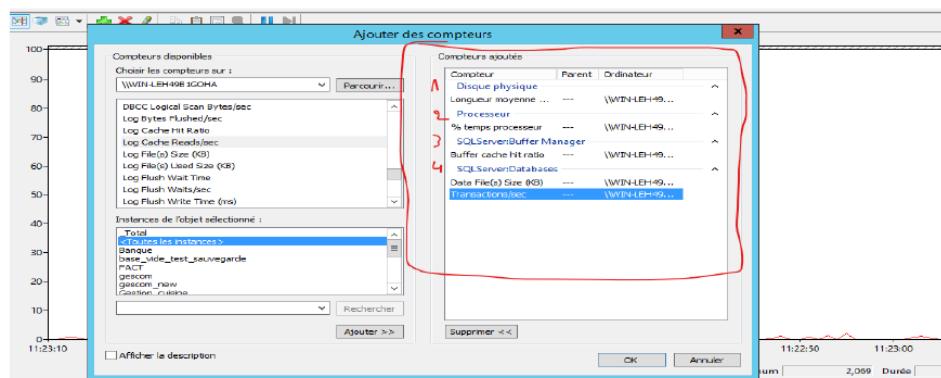


Figure 13.3: Le compteur Data File Size

Cette figure nous montre **l'ensemble des compteurs** qu'on a récupérés pour **surveiller les performances collectées** à partir du serveur WIN-LEH49B1GOHA, comme vous pouvez le voir nous avons sélectionné **un compteur pour le disque physique, un compteur pour le processeur, un compteur pour la mémoire et deux compteurs pour la base de données**.

### 13.1.3 Lancement d'une charge de travail au niveau du serveur

L'exemple qu'on va illustrer dans la figure ci-dessous consiste à **exécuter une charge de travail** assez lourde qui va insérer des enregistrements au niveau d'une table tant que condition n'est pas vérifiée. **L'objectif est de voir l'impact**

**des performances sur le système** et comment les compteurs vont enregistrer les tremblements au niveau des différents compteurs qu'on a ajouté via le moniteur de performance.

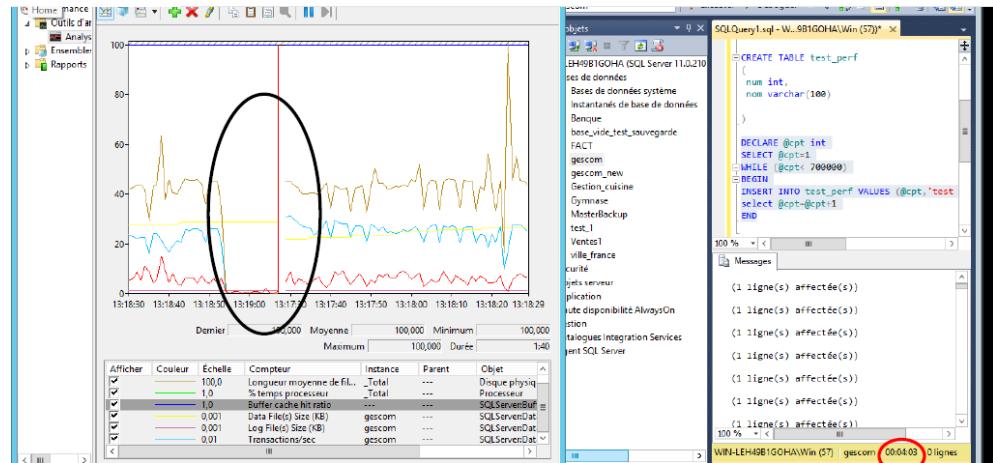


Figure 13.4: Exécution d'une charge de travail intense

On constate que le moniteur de performance **a enregistré un goulot d'étranglement** au niveau de la longueur moyenne de la file d'attente, au niveau du nombre de transactions lancés par seconde, en ce qui concerne la consommation des ressources CPU on peut la considérer acceptable tant qu'elle n'a pas atteint **un pourcentage de 65 %**.

On remarque également que la sollicitation **des entrées/sortie** au niveau du disque ainsi que **le nombre de transaction/s** est plus au moins **constant dans le temps** ce qui est tout à fait normal **compte tenu du temps écoulé** pour finaliser l'exécution de la charge de travail.



#### NOTE IMPORTANTE

Il est possible **d'enregistrer le fichier log du moniteur de performance** (enregistrement de tous les compteur défini) dans le système de fichier pour pouvoir ensuite les corréler avec d'autre fichiers de trace de SQL server profiler et connaître ainsi **quelle est la requête qui consomme le plus de ressource CPU**, celle qui consomme le plus d'entrée disque où celle qui consomme le plus de mémoire.

### 13.1.4 Les compteurs de performances personnalisés

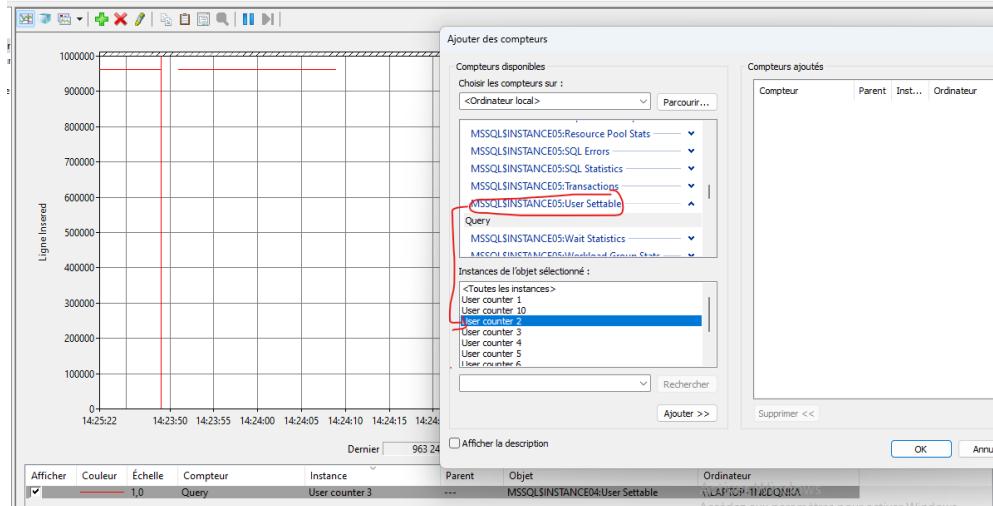
Les compteurs de performances personnalisés vont nous servir principalement à **affiner l'analyse** après la collecte de données de performances de SQL Server.

### 13.1.5 Caractéristiques :

- Les compteurs personnalisés sont disponibles pour chaque instance dans **l'objet UserSettable**
- Le moniteur de performances vous mis à disposition **10 procédures stockés réservés** pour vos compteurs de **sp\_user\_counter1 à sp\_user\_counter10**

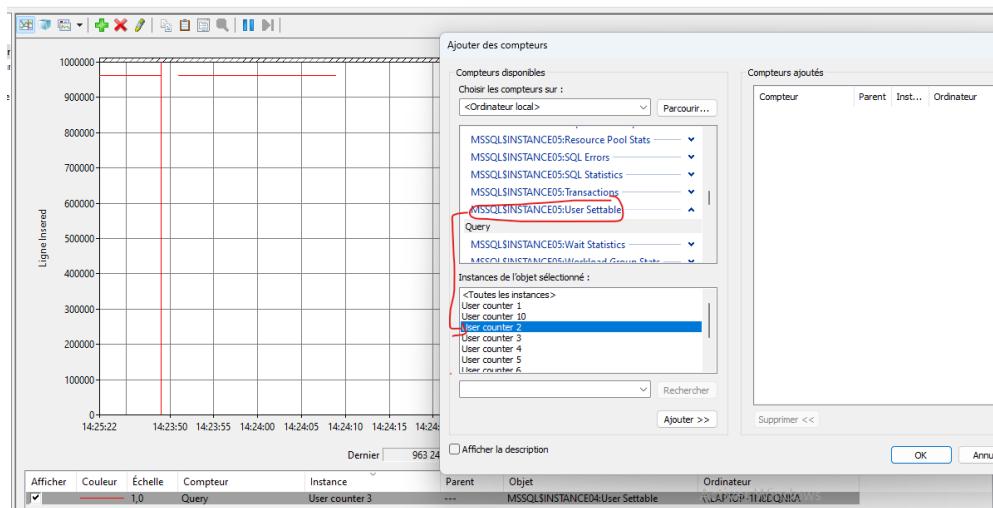
### 13.1.6 Créer un nouveau compteur personnalisé

Pour pouvoir **créer un compteur de performance personnalisé** en fonction de vos besoins, veuillez vous appuyer sur l'exemple suivant qui consiste à **surveiller le nombre d'enregistrement inséré dans une table** :



**Figure 13.5: Etape 1 : choisir un compteur personnalisé dans l'objet UserSettable**

En développant le **noeud UserSettable** de l'instance sur laquelle vous souhaitez **surveiller un évènement** quelconque, vous avez le choix de **sélectionner jusqu'à 10 procédures stockés** destinés pour créer des compteurs personnalisés.



**Figure 13.6: Etape 2 : Corréler le compteur avec l'évènement**

L'évènement SQL de la figure consiste à **calculer** à l'aide d'une fonction scalaire **le nombre d'enregistrements insérés**, **la valeur de sortie** de la fonction est directement **stockée dans l'une des procédures stockés réservées pour les compteurs de performances**, en suivant cette démarche, vous allez pouvoir corrélérer

l'évènement à votre compteur à des fins de surveillance et/ou d'analyse ultérieur.

# Chapter 14

## Dépannage des problèmes d'administration

### 14.1 Corrélation des fichiers de trace

#### 14.1.1 Objectifs

1. Apprendre à **dépanner les problèmes** rencontrées sur un environnement SQL server
2. Effectuer une **corrélation entre les compteurs de performances Windows avec les traces de SQL profiler** pour identifier les goulots d'étranglements
3. Identifier rapidement **un point de contention ou un point de conflit** sur l'un des paramètres de performance (Exp : la longueur de file d'attente, consommation des IO ou de ressource CPU.. )
4. Identification des **requêtes à charge élevé** ou les requêtes couteuses

### 14.1.2 Principe de la corrélation

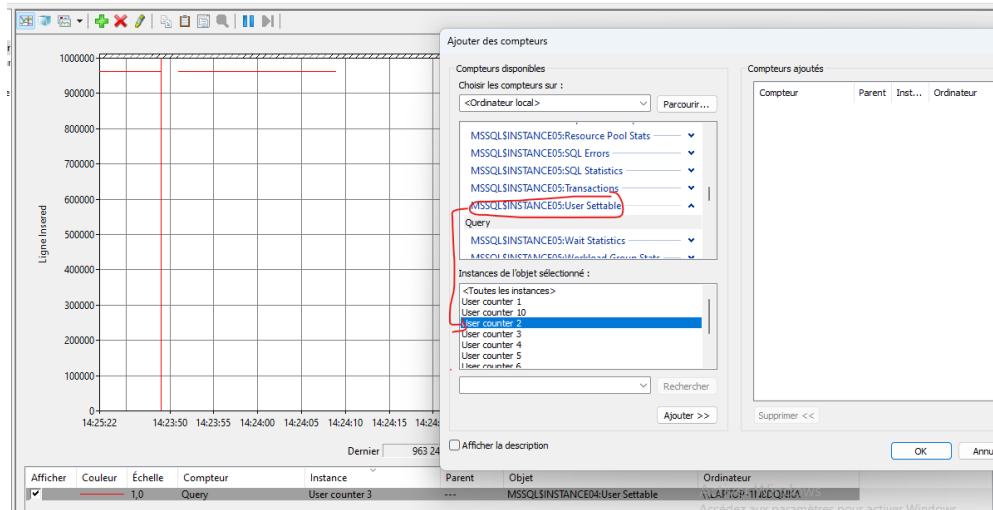
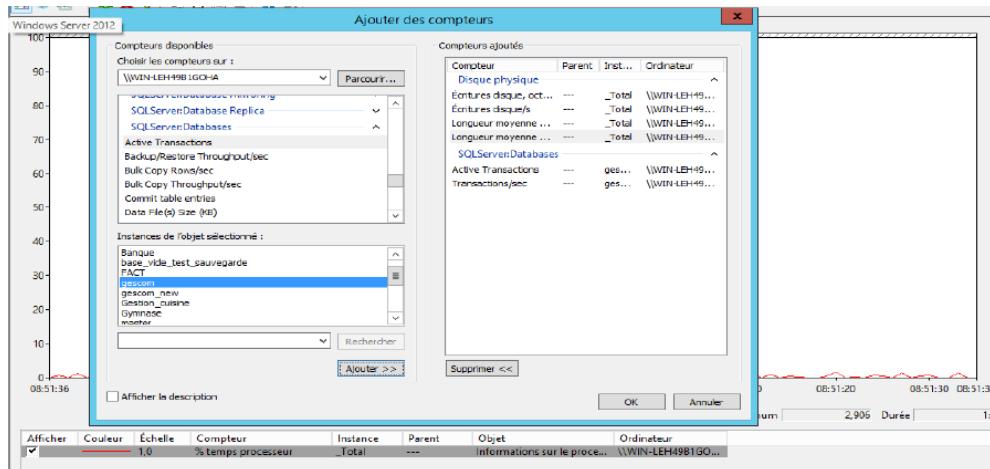


Figure 14.1: Le principe de la corrélation

Le schéma de la Figure 14.1 montre le principe de la corrélation entre le compteur et le fichier de trace de SQL profiler, la corrélation elle-même va permettre de combiner le rôle des deux outils et d'en tirer l'avantage **d'identifier rapidement la clause, l'instruction ou le lot de code SQL qui est à l'origine du goulot d'étranglement.**

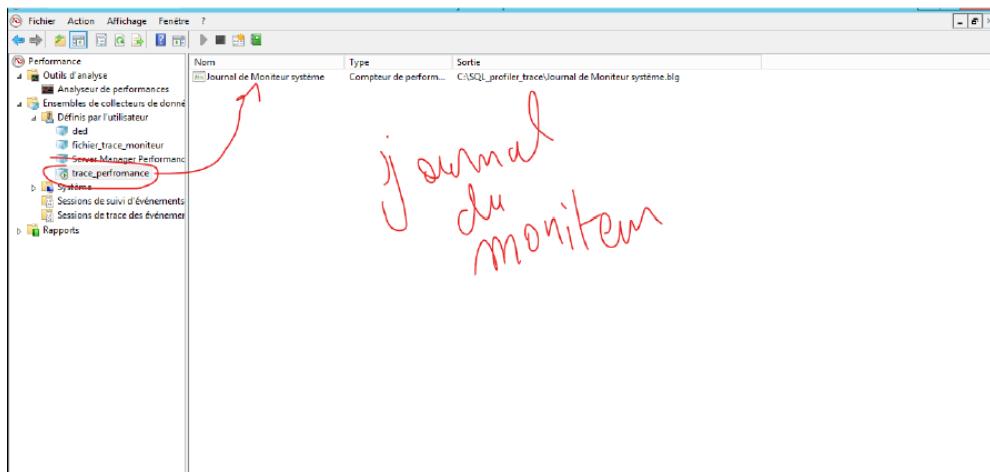
### 14.1.3 La corrélation

La première étape pour réaliser la corrélation est la **configuration de l'analyseur de performance** avec les compteurs que vous voulez surveiller.

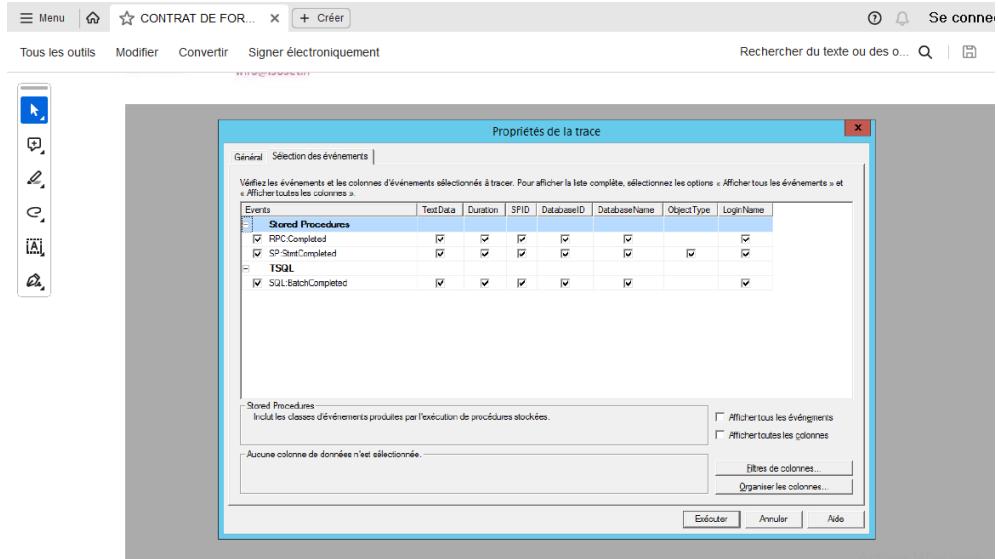


**Figure 14.2:** Etape 1 : Ajouter les compteurs de performances

**La seconde étape** est de **démarrer le processus de collecte de données de performances** puis les enregistrées dans un fichier de trace au niveau du système de fichiers.



**Figure 14.3:** Etape 2 : Enregistrer le fichier de trace du moniteur

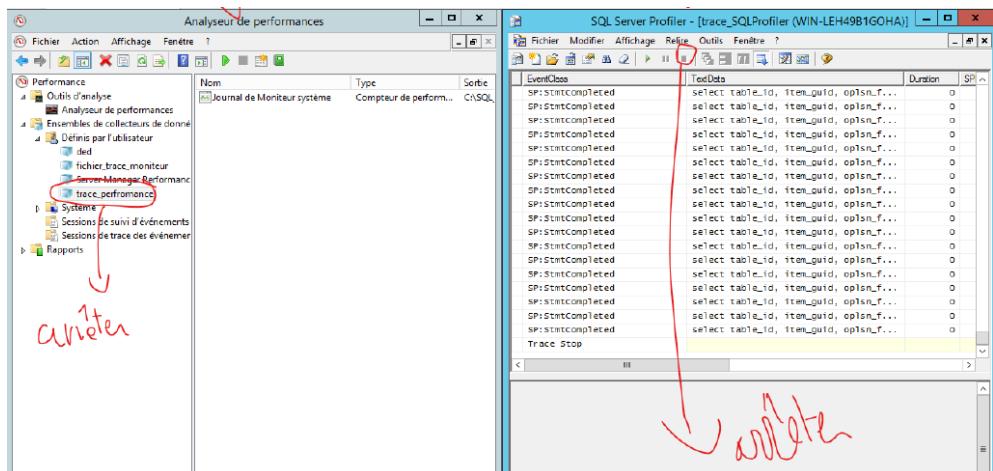


**Figure 14.4: Etape 3 : Personnaliser les classes d'évènements sur SQL profiler**



### NOTE IMPORTANTE

Dans le cadre de la corrélation, il faut impérativement cocher l'attribut **StartTime**. Choisir le **Modèle Tuning** pour **optimiser les requêtes**



**Figure 14.5: Etape 4 : Stopper la collecte de données**

Afin de pouvoir **corréler les fichiers de trace**, il faut **stopper les sessions de trace en cours** à partir du moniteur de performance et à partir de SQL server profiler.

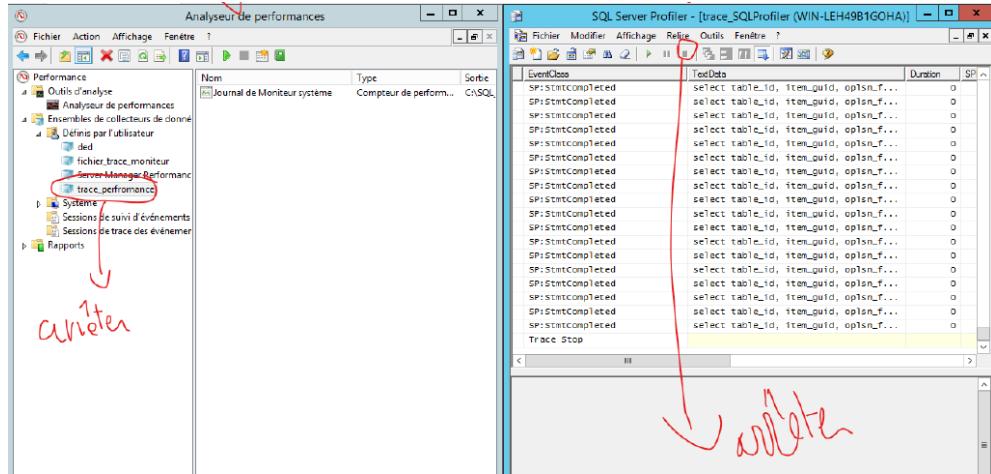


Figure 14.6: Etape 5 : Importer les données de performances

Cette étape du processus de corrélation consiste à **ouvrir le fichier de trace de SQLprofiler** qui a été au préalable enregistré dans le système de fichier puis **importer les données de performance**

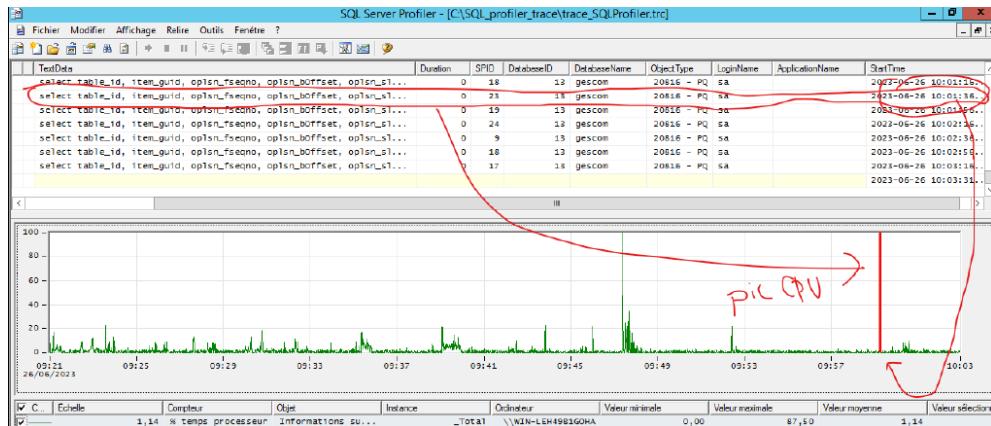


Figure 14.7: Etape 6 : Analyser le fichier de trace

Cette figure montre **le principe de la corrélation**, à chaque fois que vous **cliquer sur un évènement** qui s'est produit au niveau de la base de données "gescom", une barre rouge apparaît pour se positionner à l'instant exacte du lancement de l'instruction (d'où l'importante de la colonne d'événement "StartTime" )



#### NOTE IMPORTANTE

La corrélation des données du moniteur de performance avec le fichier de trace du générateur de profiles est une utilisation très pertinente, sont utilisation doit se faire systématiquement par le DBA pour identifier de manière précise quelle est la requête ou la transaction qui provoque les goulots d'étranglement

## 14.2 L'outil SQL tuning Advisor DTA

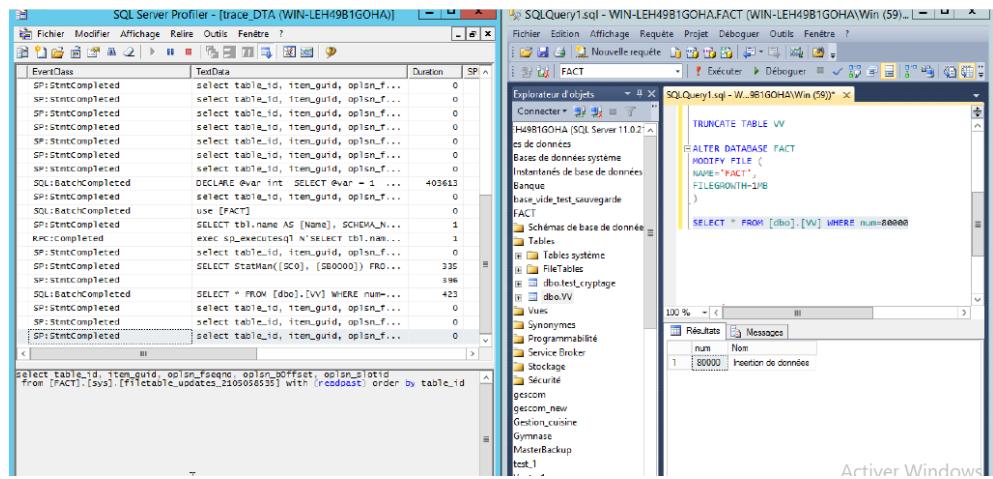
### 14.2.1 Définition

1. C'est un outil très puissant qu'il faut impérativement utiliser en production en cas de soucis de performance sur les requêtes, il va énormément aider le DBA dans les phases d'analyses des problèmes de performances rencontrés.
2. Il apporte une solution globale d'amélioration des performances
3. Le DTA va ajouter une touche importante de préconisation d'amélioration en terme d'indexation ou de partitionnement.

### 14.2.2 Le principe d'utilisation du DTA

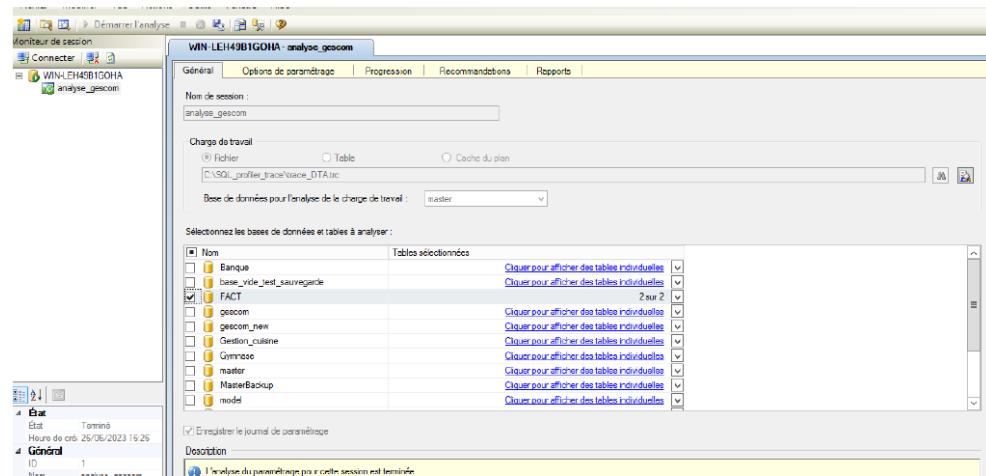
L'utilisation du DTA repose sur trois étapes à réaliser dans l'ordre :

1. Construire un fichier de charge de travail (audit d'environnement SQL ou lancer une session de trace SQL profiler)



**Figure 14.8: Etape 1 : Construire un fichier de trace**

### 2. Configurer le DTA : soumettre le fichier de trace pour analyse au DTA



**Figure 14.9: Etape 2 : Configurer le DTA**

### 3. enregistrer ou appliquer les recommandations du DTA

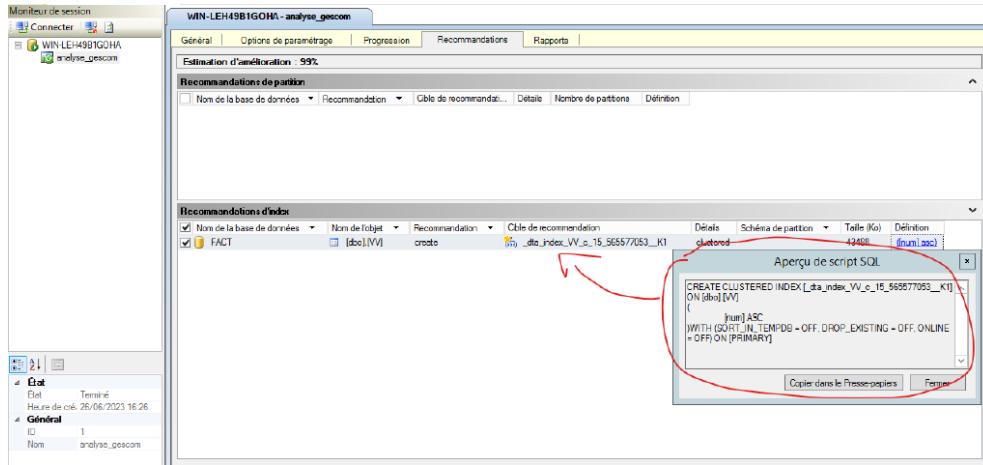


Figure 14.10: Etape 3 : Le résultat de l'analyse DTA

Le DTA dans cette figure, **préconise le positionnement des index** sur la table qui contient 1 million d'enregistrement pour permettre **un accès rapide aux données** et ainsi améliorer les performances du système.



#### NOTE IMPORTANTE

**ATTENTION**, il n'est pas recommander d'appliquer les recommandations sur un environnement de PROD, **enregistrer** plutot le fichier **puis le soumettre au DTA dans un environnement de test** pour voir l'impacte des recommandations.

## 14.3 Les plans d'exécution des requêtes

### 14.3.1 C'est quoi un Plan d'exécution ?

C'est la manière avec laquelle le moteur de base de données accède aux données (FULL SCAN ou INDEX).

Le Plan d'exécution **fournit des estimations sur le cout de chaque opération** en terme de ressource système (CPU, IO disque, mémoire ...)



#### NOTE IMPORTANTE

Pour utiliser cette fonctionnalité, **les utilisateurs doivent disposer des autorisations appropriées** pour exécuter les requêtes Transact-SQL pour lesquelles un plan d'exécution graphique est généré, et ils doivent **disposer de l'autorisation SHOWPLAN** pour toutes les bases de données référencées par la requête.

#### Le plan d'exécution estimé

Un plan d'exécution estimé **ne contient aucune information d'exécution**. Au lieu de cela, le plan d'exécution généré affiche le plan d'exécution de requête utilisé par le moteur de base de données SQL Server **si les requêtes ont été réellement exécutées** et affiche les lignes estimées qui transitent par plusieurs opérateurs du plan.

Pour afficher le plan d'exécution estimé d'une requête :

1. **Sélectionner la requête ou le lot T-SQL** dont vous voulez afficher le plan d'exécution estimé.
2. Dans la barre d'outils, sélectionner **"requête"** puis plan d'exécution estimé (**CTL+L**)
3. Le plan utilisé par **l'optimiseur de requête** est affiché sous l'onglet **Plan d'exécution** dans le volet de **résultats**.



### NOTE IMPORTANTE

Vous pouvez également utiliser **SET SHOWPLAN XML ON** pour **retourner les informations du plan d'exécution pour chaque instruction sans l'exécuter**. Si elle est utilisée dans SQL Server Management Studio, **l'onglet Résultats a un lien pour ouvrir le plan d'exécution au format graphique**.

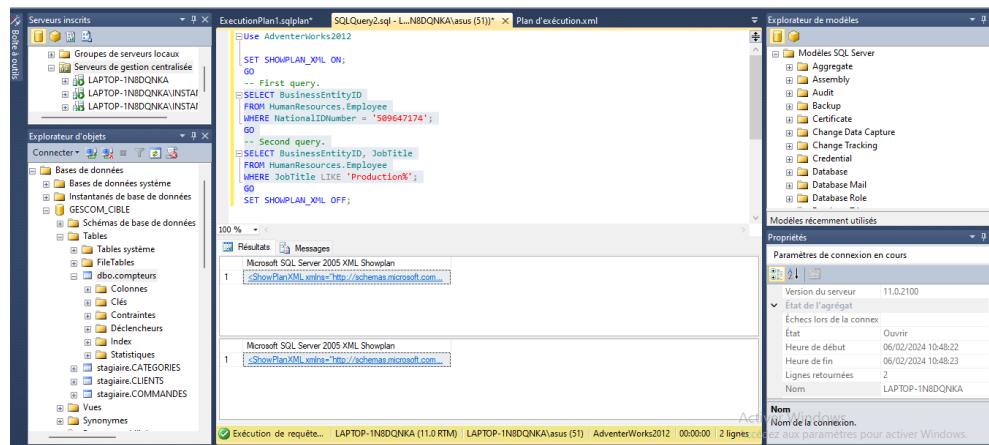


Figure 14.11: Afficher le Plan d'exécution estimé en format XML



### NOTE IMPORTANTE

**Les plans d'exécution estimé** offre à l'administrateur la possibilité d'analyser les performances ou **le cout potentielle d'une requete avant de la lancer dans un environnement de PROD**

Pour afficher le plan d'exécution réel d'une requête :

1. Utilisez le raccourci **CTL+M** pour inclure le plan d'exécution réel **avant d'exécuter la requête**
2. Le plan utilisé par l'optimiseur de requête est affiché sous **l'onglet Plan d'exécution** dans le volet de résultats.

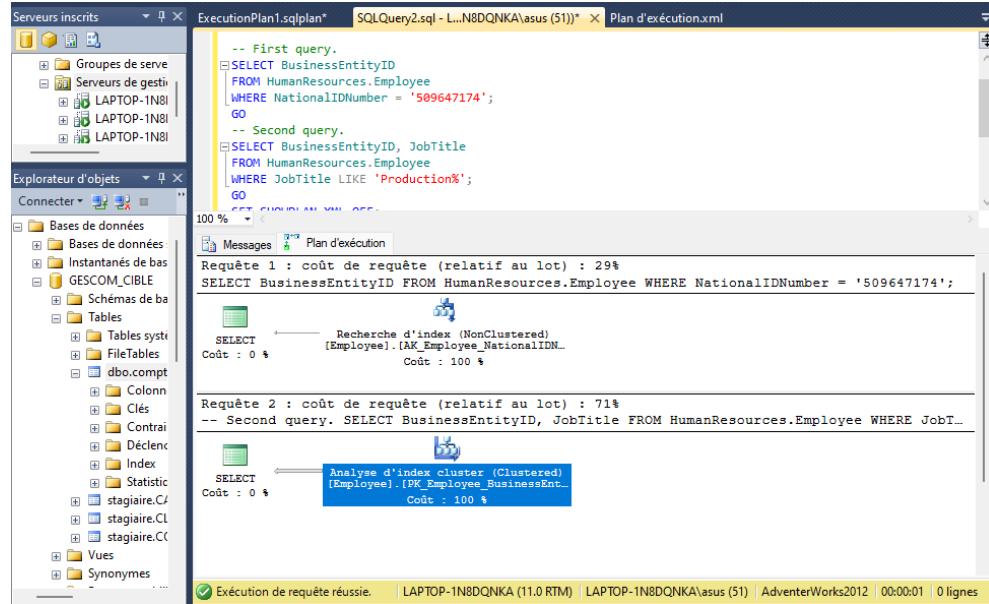


Figure 14.12: afficher le Plan d'exécution réel

### Le plan d'exécution réel

Les plans d'exécution réels sont générés **après l'exécution des requêtes ou lots T-SQL**. ils contiennent des **informations d'exécution** réelles et des métriques **d'utilisation de ressource** réelles.

#### 14.3.2 Le cout d'exécution d'une requête

Afin d'apprendre à **examiner et analyser le plan d'exécution d'une requête**, nous avons construit **une requête complexe** très couteuse avec un **CROSS JOIN**. Comme le montre la figure ci-dessous :

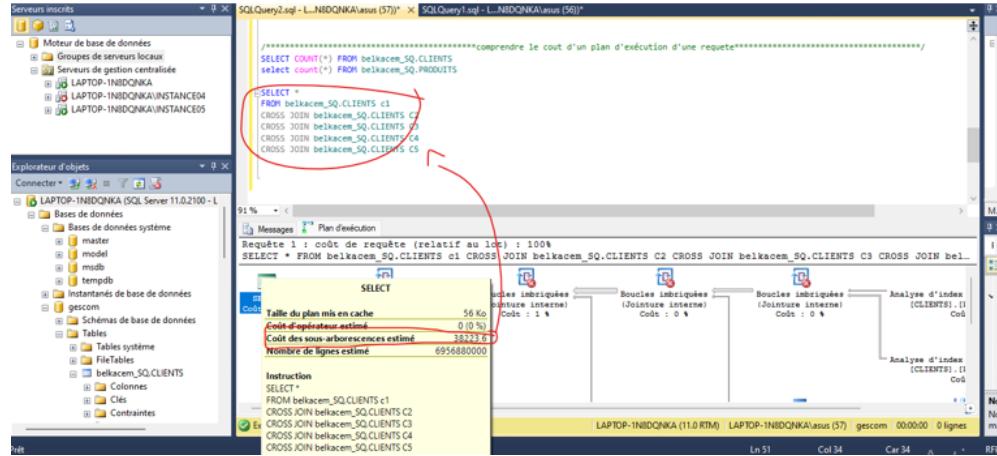


Figure 14.13: Le plan d'exécution estimé

Pour évaluer le cout d'exécution d'une requête, il faut analyser systématiquement l'opérateur "cout des sous-arborescence", pour obtenir un aperçus globale sur la performance de la requête. là à l'occurrence, la requête va couter environ 38223.6 s . Ce qui est énorme !!!

### 14.3.3 La mise en cache du plan

La recherche de la stratégie d'exécution par l'optimiseur (Plan d'exécution) peut prendre du temps, il est donc nécessaire de stocker le plan d'exécution dans le cache de plan (le cache du procédure) afin d'éviter au moteur de compiler plusieurs fois la même requête

### 14.3.4 Cas pratique de la mise en cache

Afin de mettre en pratique comment fonctionnent la mise en cache des plans d'exécution des requêtes, nous avons choisi d'exécuter les instructions illustrées dans la figure ci-dessous :

The screenshot shows the SQL Server Management Studio interface. In the center, there are three numbered queries:

- `SELECT DISTINCT t1.SOCIETE, t1.ADRESSE  
FROM stagiaire.CLIENTS t1  
JOIN stagiaire.COMMANDES t2 ON t1.CODE_CLIENT=t2.CODE_CLIENT  
WHERE PAYS='France'`
- `SELECT DISTINCT t1.SOCIETE, t1.ADRESSE  
From stagiaire.CLIENTS t1  
JOIN stagiaire.COMMANDES t2 ON t1.CODE_CLIENT=t2.CODE_CLIENT  
WHERE PAYS='Canada'`
- `SELECT DISTINCT t1.SOCIETE, t1.ADRESSE  
FROM stagiaire.CLIENTS t1  
JOIN stagiaire.COMMANDES t2 ON t1.CODE_CLIENT=t2.CODE_CLIENT  
WHERE t1.CODE_CLIENT='WELL'`

Below the queries, a results grid is shown with the following data:

text	usecounts	cacheobjtype	size_KB
1. SELECT DISTINCT t1.SOCIETE, t1.ADRESSE FROM stagiaire.CLIENTS t1 JOIN stagiaire.COMMANDES t2 ON t1.CODE_CLIENT=t2.CODE_CLIENT WHERE PAYS='France'	1	Compiled Plan	56
2. SELECT DISTINCT t1.SOCIETE, t1.ADRESSE From stagiaire.CLIENTS t1 JOIN stagiaire.COMMANDES t2 ON t1.CODE_CLIENT=t2.CODE_CLIENT WHERE PAYS='Canada'	1	Compiled Plan	56
3. SELECT DISTINCT t1.SOCIETE, t1.ADRESSE From stagiaire.CLIENTS t1 JOIN stagiaire.COMMANDES t2 ON t1.CODE_CLIENT=t2.CODE_CLIENT WHERE t1.CODE_CLIENT='WELL'	1	Compiled Plan	56

Red annotations include a bracket over the first two queries labeled "3 requêtes mises en cache" and a bracket over the results table labeled "F".

Figure 14.14: Exemple de mise en cache des requêtes couteuses

Cet exemple consiste tout simplement à **vider en premier lieu le cache de procédure** afin de forcer la mise en cache de chaque plan d'exécution associé à chaque requête par l'optimiseur de requête (compiler à nouveau la requête) **en exécutant l'instruction DBCC FREEPROCCACHE**. L'étape suivante consiste quant à elle à exécuter en même temps les 3 requêtes contenant chacune une jointure interne avant d'exécutant à la fin la dernière requête qui nous permet d'extraire les informations sur chaque plan stocké dans le cache de plan d'exécution.

Le résultat nous indique que chaque plan dont le text SQL est affiché sur la première colonne a occupé un espace au niveau du cache qui s'élève à 56Ko, de l'autre côté, le plan d'exécution estimé a indiqué des valeurs strictement inférieures soit 24Ko. La **première conclusion** qu'on peut faire est que **les données du plan estimé ne sont pas forcément les données réelles** utilisées par l'optimiseur de requête, **il donne uniquement un aperçu général sur le cout total de la requête**, une **seconde conclusion** est que la mise en cache des plans fait partie de l'une des techniques d'optimisation employée par l'optimiseur pour **éviter au moteur de base de données de recompiler la même requête plusieurs fois et donc les performances ne seront pas dégradées**.

# Chapter 15

## La mise en miroir sous SQL server

### 15.1 Configuration de la mise en miroir

#### 15.1.1 Les caractéristiques :

1. Une fonctionnalité interne qui permet de **créer une copie en temps réel d'une base de données** SQL d'un **serveur principale** sur un autre serveur appelé **serveur miroir (esclave)**
2. Garantis une **haute disponibilité** des données et une **protection contre les défaillances** du serveur principal
3. Permet de **maintenir à jours** une base de données cible à partir d'une base de données source **en réplicant en temps réel les fichiers journaux de transactions**
4. Ne peut être configurer que sur les bases de données applicatifs, **les bases de données système ne sont pas éligible à la mise en miroir.**
5. La base de données miroir est **en mode RESTORE**, elle rejoue systématiquement

l'ensemble des transactions jouées sur le serveur principale, **les utilisateurs n'y ont pas accès**



**NOTE IMPORTANTE**

**ATTENTION**, quand vous allez mettre en place **une base de données miroir** pour une base de données en production, **la réplica est en mode RESTORE** et non pas en mode **READ\_WRITE**



**NOTE IMPORTANTE** La configuration de **la mise en miroir** vous permettra en cas de crash du serveur principal, de **faire basculer les utilisateurs** vers le serveur **secondaire de secours**

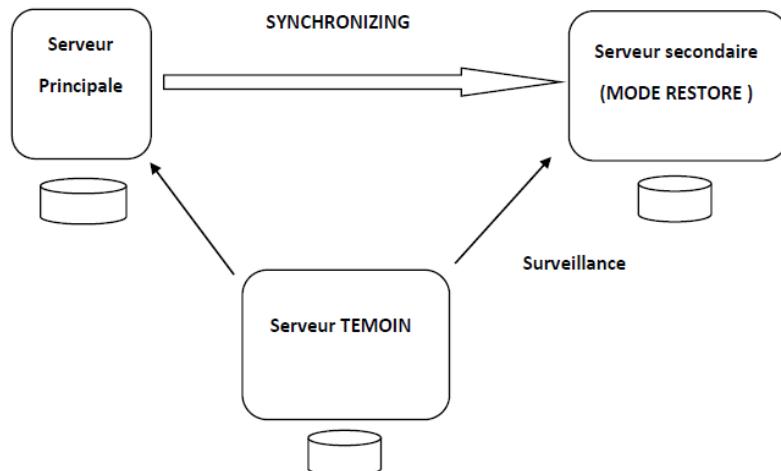
### 15.1.2 Les différents états de la base de données miroir

1. **SYNCHRONIZING** : cela veut dire que **les deux bases ne contiennent pas les mêmes informations**, la base de données principal **communique le journal des transactions** à la base de données miroir pour reproduire les modifications
2. **SYNCHRONIZED** : les deux bases **contiennent les mêmes informations**
3. **SUSPENDED** : cela veut dire que **la base principale travaille sans envoyer le journal de transactions à la base miroir**
4. **PENDING FAILOVER** : cela veut dire qu'une **demande de bascule manuelle a été faite mais pas encore accepter par les deux partenaires**, c'est l'état dans laquelle va se trouver vos bases de données lors d'une bascule.
5. **DISCONNECTED** : cela veut dire que le **contacte avec le partenaire est perdu** et que **le témoin ne communique plus avec le serveur principale et secondaire**.

**NOTE IMPORTANTE**

Faites **ATTENTION** quand vous avez des bases mirroré qui se trouvent dans un état **DISCONNECTED**, cela veut dire qu'il y'a soit **un problème réseaux** soit **un problème de disponibilité de l'instance principale et de l'instance de secours**

## 15.2 Architecture de la mise en miroir



**Figure 15.1: Architecture de la mise en miroir**

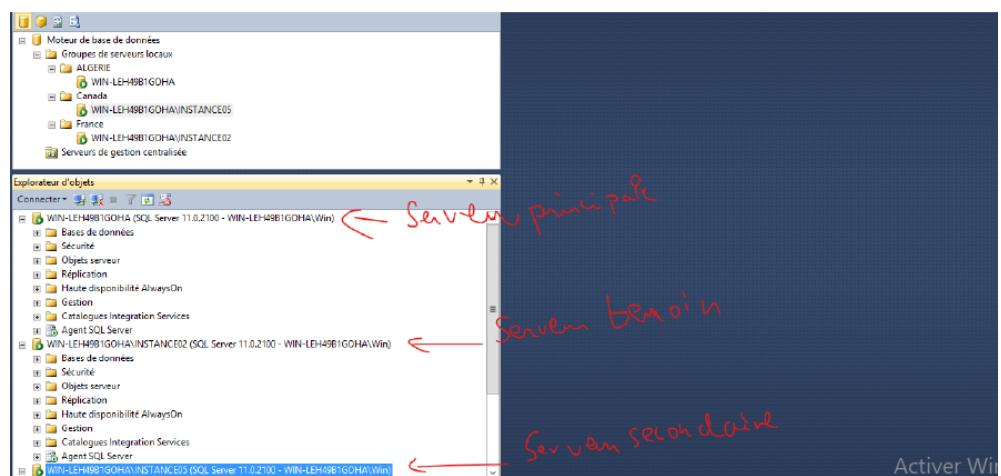
La figure Montre **le principe de database mirroring**, vous avez d'une part un serveur principal , un serveur de secours secondaire qui se trouve en **mode RESTORE** et d'autre part un serveur témoin, ce dernier va assurer en temps réel la surveillance des deux autres.

Dans **une architecture optimale** de la mise en miroir, ce **troisième serveur est obligatoire** pour déclencher le FAILOVER en cas de défaillance du serveur maître .

**NOTE IMPORTANTE**

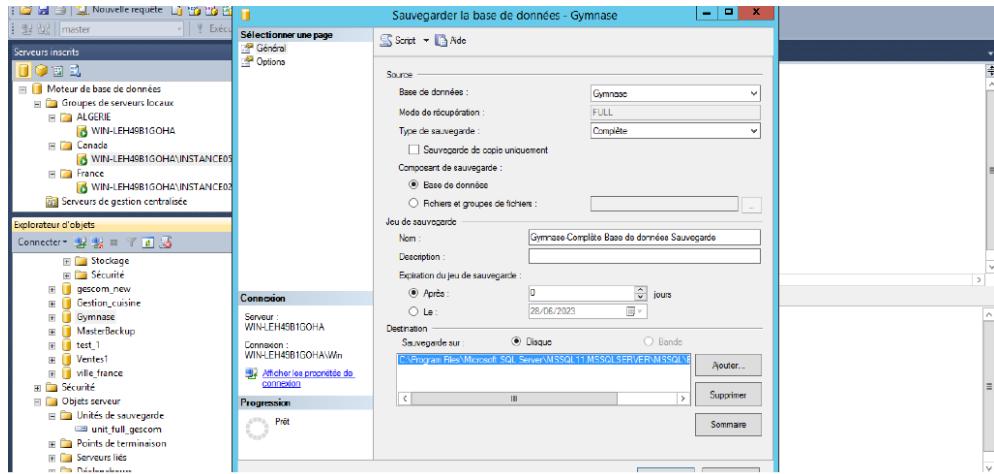
Le serveur secondaire peut jouer le rôle du serveur témoin mais **il est toutefois déconseillé**, si vous avez un problème matériel ou autre sur le serveur secondaire et bien même votre serveur témoin va se trouver impacté et par conséquence il ne sera pas en mesure de réaliser le basculement.

### 15.3 Les prérequis de la mise en place du processus de mise en miroir



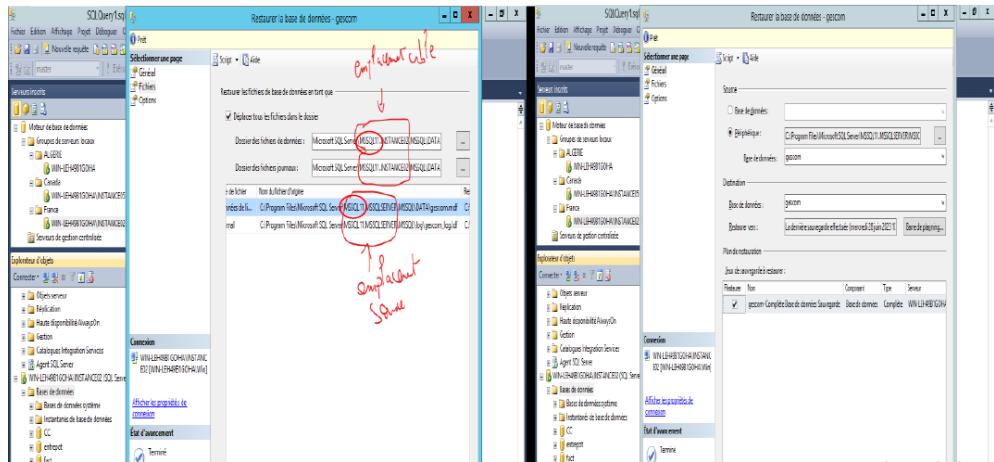
**Figure 15.2: Etape 1 : Incrire vos différents serveurs**

La première chose à faire est **d'inscrire vos différents serveurs**, identifier votre **serveur maître**, votre **serveur de secours** et **le serveur témoin** qui vont participer à la mise en miroir.



**Figure 15.3: Etape 2 : Sauvegarder la base de données principale**

Durant cette Etape, vous devriez lancer **une sauvegarde complète** de la base de données principale dont le jeu de sauvegarde sera déployer dans l'instance de la base de données de secours.



**Figure 15.4: Etape 3 : Restauration du jeu de sauvegarde sur le serveur miroir**

Avant de procéder à la restauration, **assurez vous que vous êtes bien connecter à l'instance sur laquelle vous souhaiter restaurer la base de données**, sélectionner ensuite **le support de sauvegarde** à partir du système de fichiers.

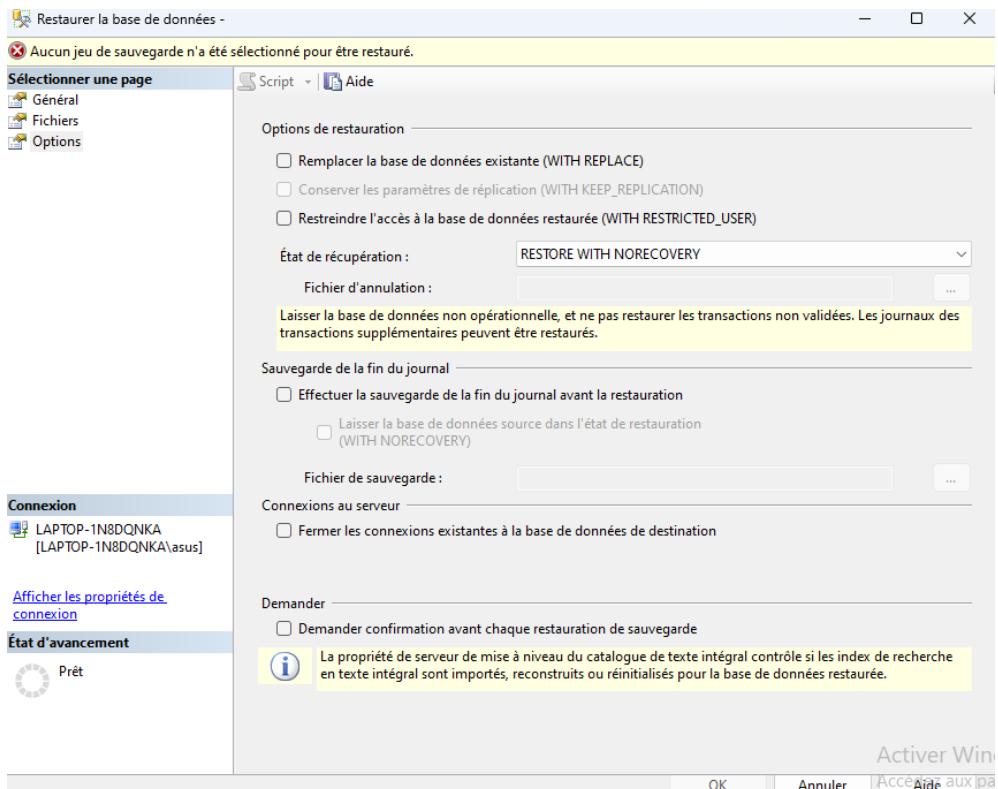


Figure 15.5: Etape 4 : spécifier l'option NORECOVERY

Faites ATTENTION à bien définir cette option lors de la mise en miroir, comme on l'a déjà expliqué en haut, la base de données cible **doit être en mode RESTORE** pour jouer l'ensemble des transactions qui proviennent du serveur principal.

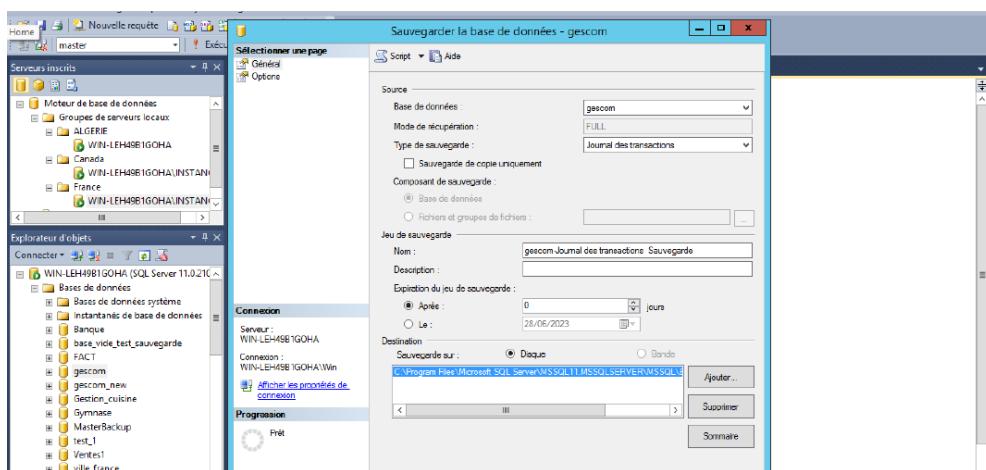


Figure 15.6: Etape 5 : sauvegarde du fichier journal de transaction

Cette étape consiste à **lancer la sauvegarde des fichiers journaux** pour pouvoir réappliquer ces fichiers journaux sur l'instance secondaire afin d'initialiser le processus des rejoues des transactions.

Une fois que le fichier journal des transactions de la base a bien été sauvegardé, il faut **rejouer le jeu de sauvegarde dans le serveur miroir** comme le montre la figure ci-dessous :

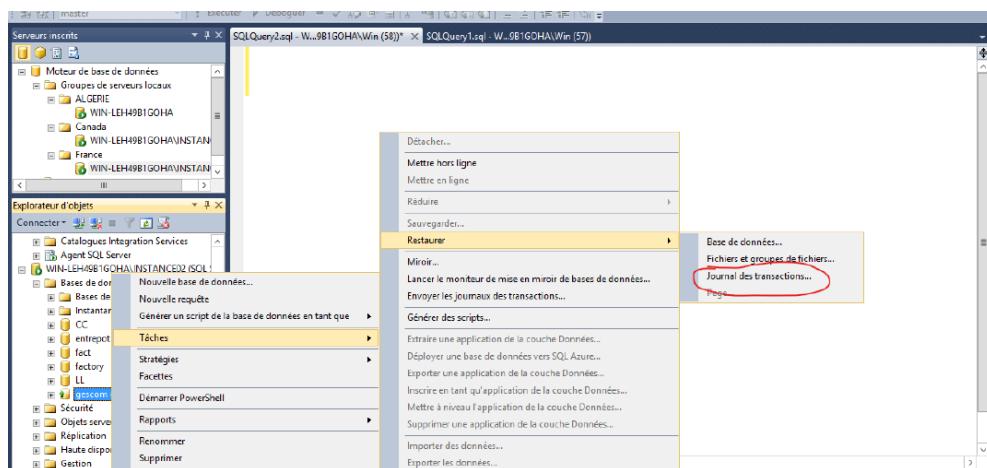


Figure 15.7: Etape 6 : Restaurer le fichier journal des transactions

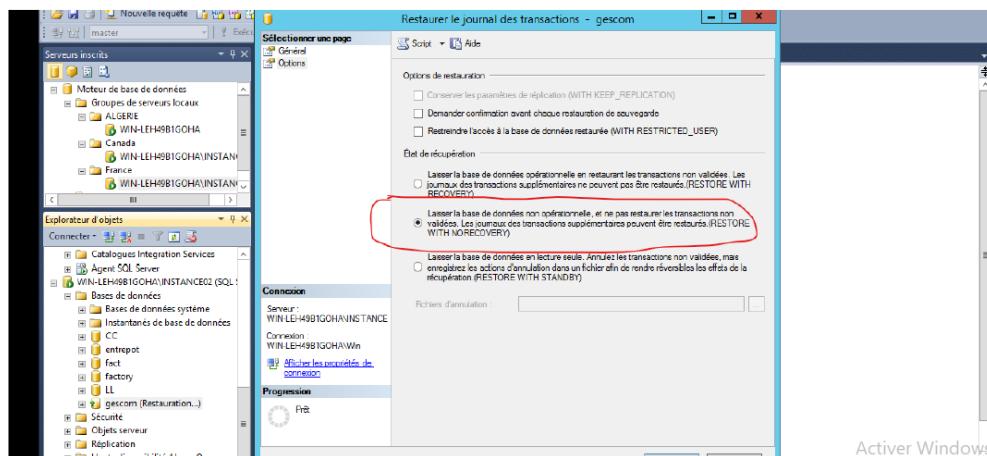


Figure 15.8: Etape 7 : Activer l'option NORECOVERY



### NOTE IMPORTANTE

faites **ATTENTION** à bien conserver **le mode RESTORE** lorsque vous allez restraurer votre fichier journal des transactions

## 15.4 Mise en place de la mise en miroir

Une fois que les étapes de prérequis ont bien été réalisés avec succès, la configuration de la mise en miroir peut être mis en place :

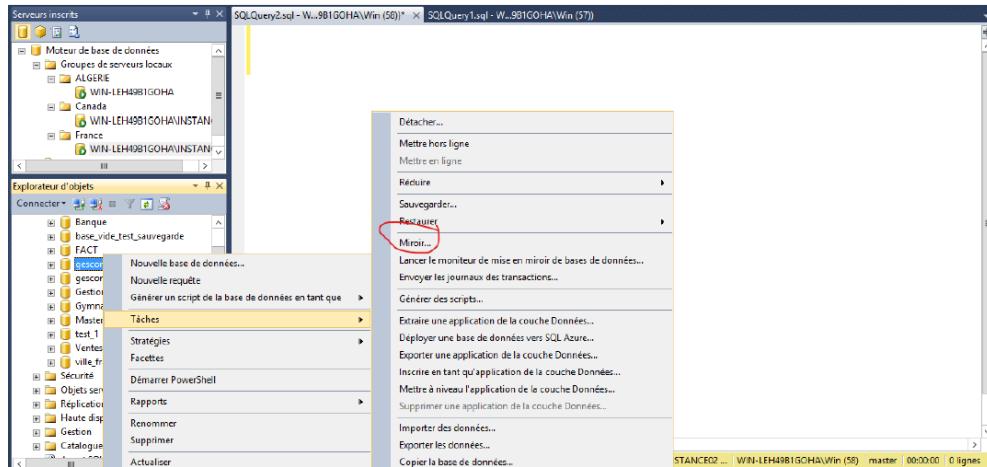


Figure 15.9: Etape 1 : Démarrer la configuration de la mise en miroir

A partir du serveur principal, vous faites **un clic droit sur la base de données** que vous voulez copier vers le serveur de secours puis vous **sélectionnez "Miroir"**

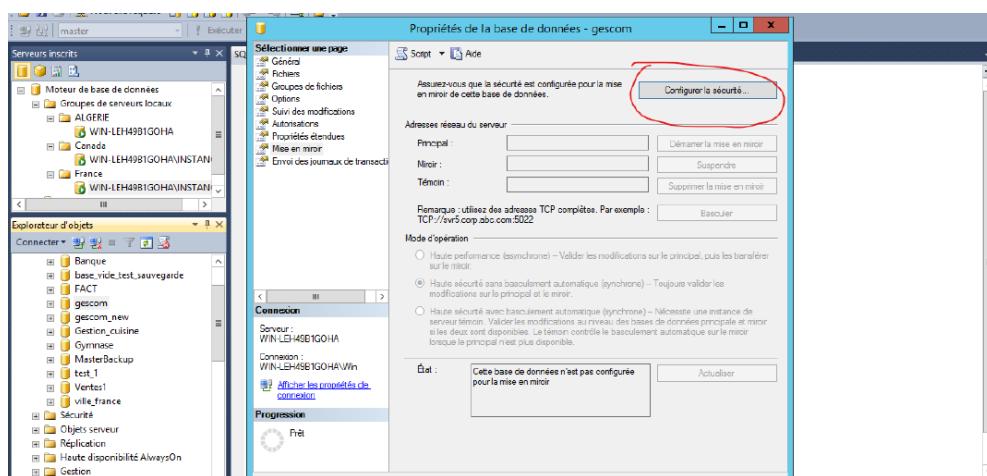
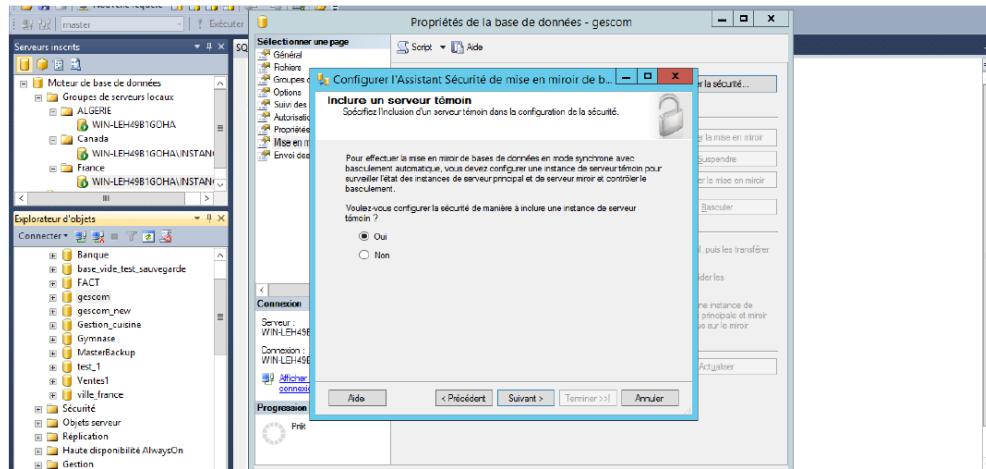


Figure 15.10: Etape 2 : Démarrer la configuration de la mise en miroir

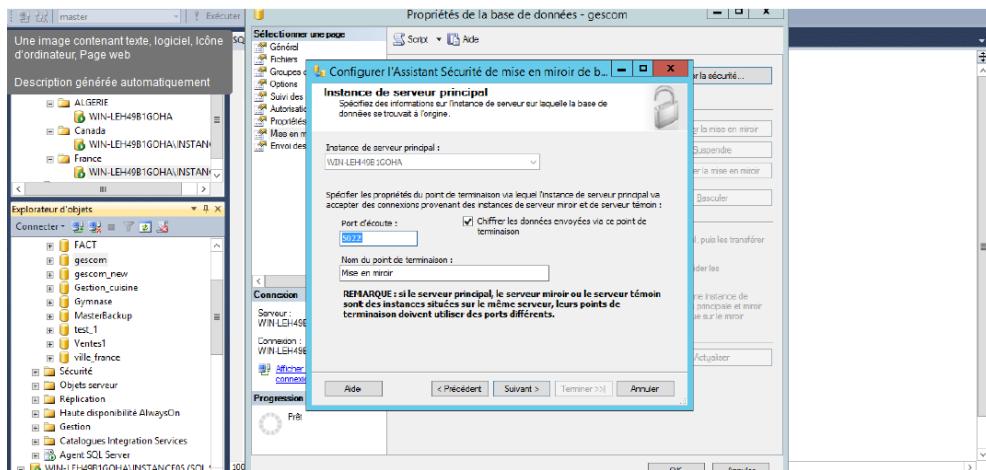
Il faut cliquer sur **"configurer la sécurité"** pour permettre aux 3 instances de

communiquer entre elles et surtout permettre au serveur témoin d'aller communiquer avec le serveur maître et le serveur secondaire.



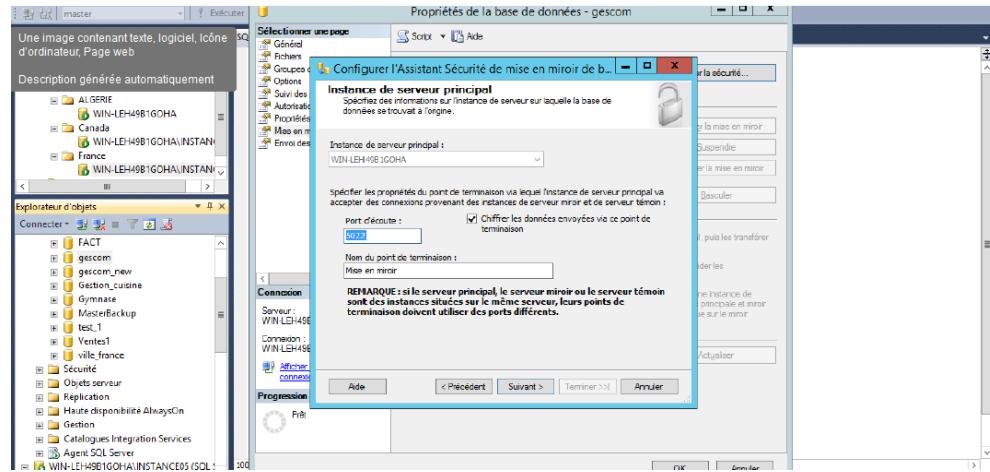
**Figure 15.11: Etape 3 : Inclure le serveur témoin**

A l'aide de **l'assistant de configuration**, nous allons **ajouter un serveur témoin** dans le rôle est d'assurer le basculement automatiquement en cas de sinistre qui survient sur le serveur principal.



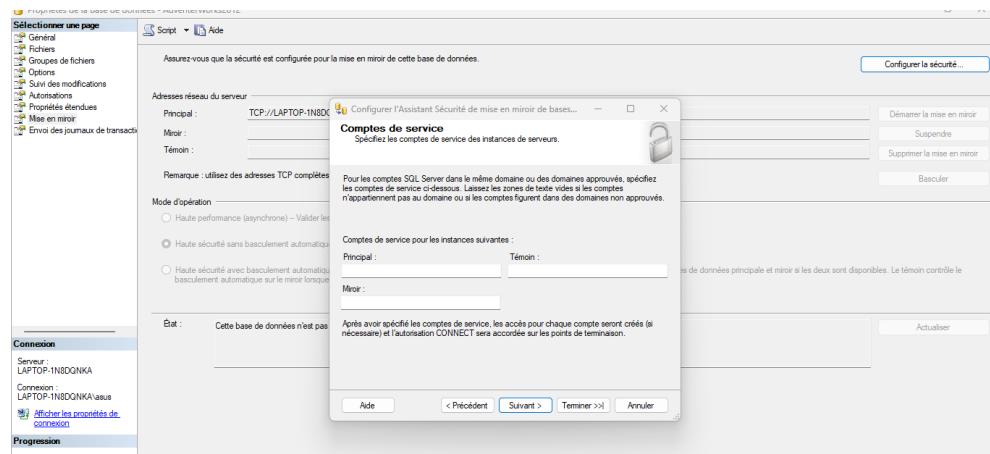
**Figure 15.12: Etape 4 : configurer le serveur maître**

Spécifier **les informations sur l'instance du serveur** sur laquelle la base de données se trouvait à l'origine en indiquant **le numéro de port d'écoute**, **le nom du serveur** ainsi que **le nom du point de terminaison**



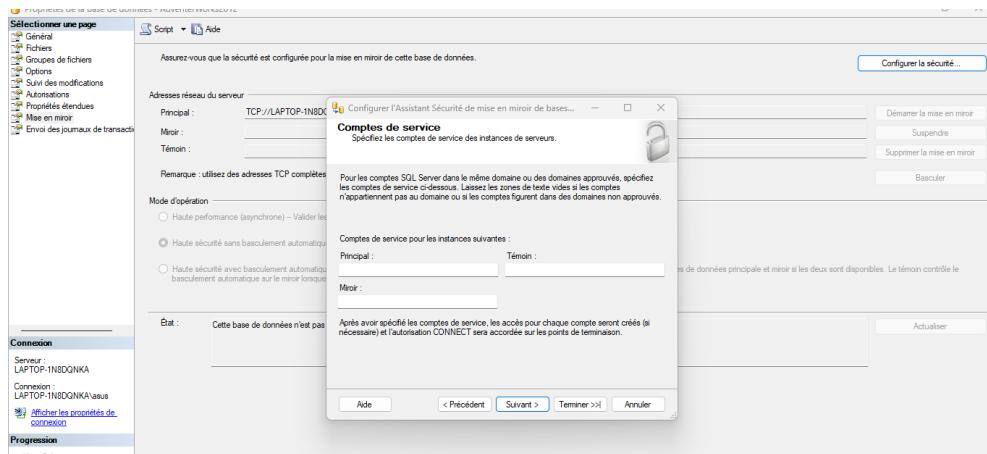
**Figure 15.13: Etape 5 : Configurer le serveur secondaire**

Dans cette interface nous allons **inclure les informations concernant l'instance de serveur sur laquelle se trouve la copie miroir** de la base de données.



**Figure 15.14: Etape 6 : Configurer les comptes services des différents instances**

Il faut spécifier ici **le compte administrateur de chaque instance**, si c'est un compte local (SQL server) vous devriez spécifier pour chaque instance le compte administrateur sinon vous pouvez spécifier directement le compte du domaine (compte Windows).



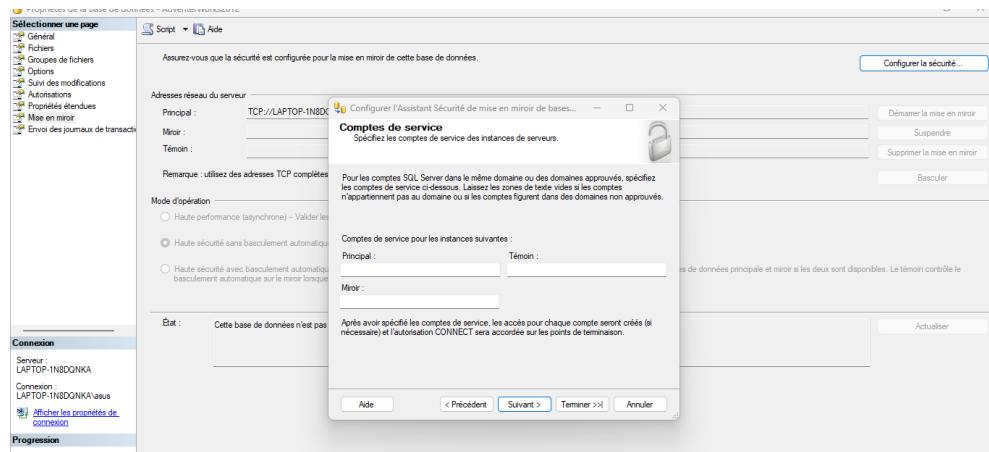
**Figure 15.15: Etape 7 : Configuration terminée de la mise en miroir**

Dans cette figure nous pouvons facilement identifier le mode de mise en mémoire là à l'occurrence c'est **le mode synchrone** qui permet le basculement automatique du serveur principal vers le serveur secondaire, ce mode comme on la déjà expliqué nécessite la configuration du serveur témoin.

Nous pouvons également voir **l'état de la mise en miroir** qui est en cours de synchronisation, toutes les données situées dans le serveur principal sont en cours de transfert vers le serveur secondaire. (Etat = SYNCHRONIZING).

Faites un clic sur **”actualiser”** pour synchroniser définitivement les données entre le serveur principale et le serveur miroir. (Les deux serveurs contiennent les mêmes informations).

Pour valider que les deux instances sont bien synchronisées entre eux après avoir finalisé la mise en miroir, il faut afficher **”le moniteur de mise en miroir de bases de données”** comme le montre la figure ci-dessous :



**Figure 15.16: Etape 8 : Afficher le moniteur de mise en miroir**

Nous pouvons définitivement confirmer que **la synchronisation des deux serveurs à bien été réalisé avec succès.**

## 15.5 Le test de basculement (FAILOVER)

Pour mettre en pratique le teste de basculement, **réaliser chronologiquement** les étapes suivantes :

1. **Lancer une opération transactionnelle d'insertion de données** dans une table appartenant à une base de données principale.

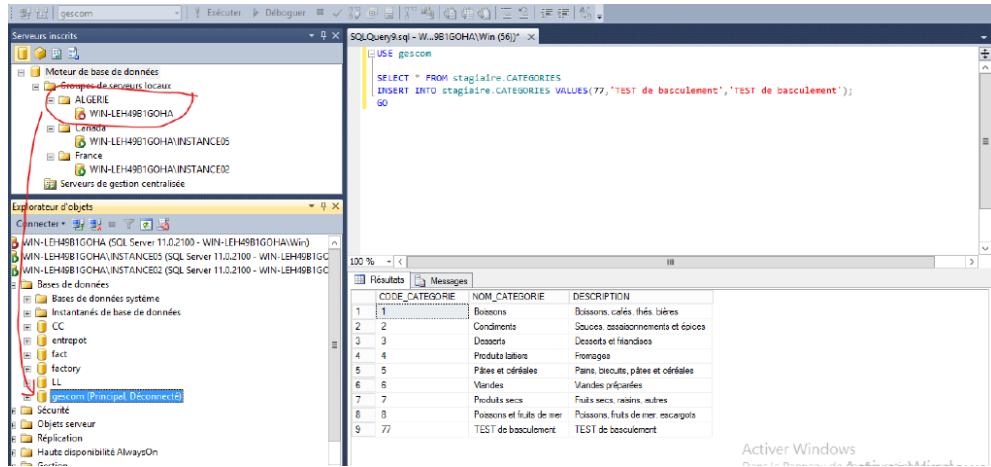


Figure 15.17: Insérer un enregistrement dans une table

2. Faite tomber volontairement en échec le serveur principal en mettant en arrêt le service moteur de base de données à partir du gestionnaire de configuration SQL server

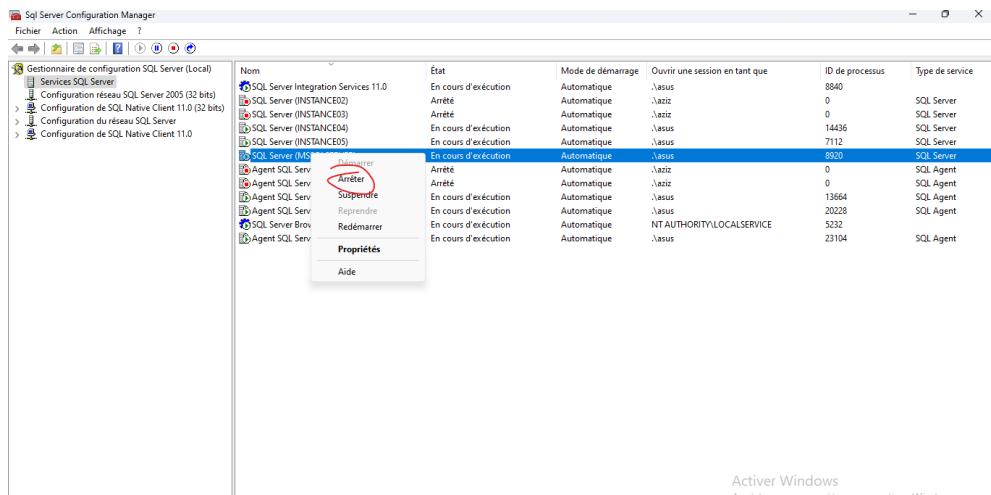
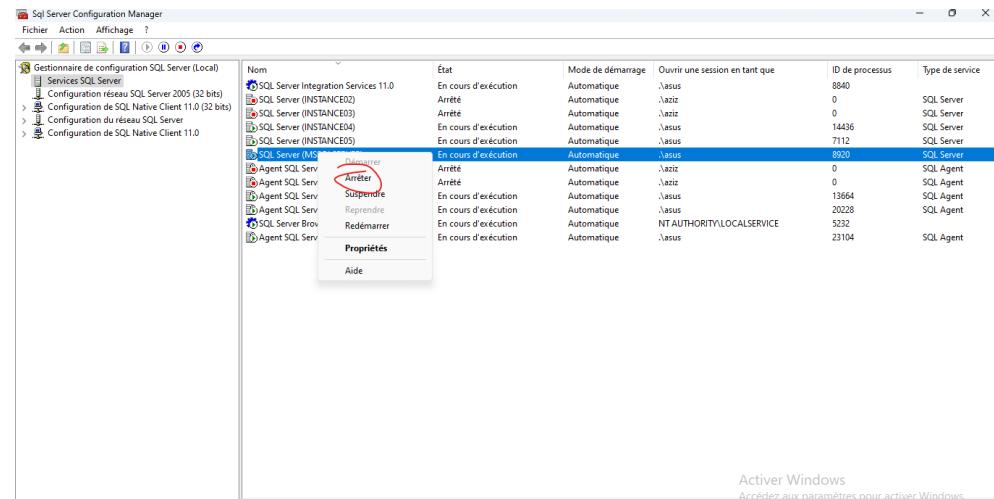


Figure 15.18: Arreter l'instance du serveur principal

3. Allez consulter à présent les données répliquées vers le serveur secondaire en faisant un SELECT sur la table dans laquelle vous avez ajouté l'enregistrement



**Figure 15.19: Arreter l'instance du serveur principal**

Si l'opération de SELECTION s'est déroulée correctement sur la copie de la base (La base de donnée sera ouverte en mode LECTURE/ECRITURE) et l'enregistrement insérée se trouve belle et bien dans la table, vous pouvez conclure que le serveur témoin à bien basculer les données utilisateurs vers le serveurs de secours qui en l'occurrence est devenu le serveur maître.



**NOTE IMPORTANTE** Eviter de forcer le basculement **manuellement** vers le serveur de secours à partir de "**propriétés de la base de données**", cela ne **nous vous permet pas de valider si votre serveur témoin joue bien son rôle de bascule.**

# Appendix A

## Les objets de base de données système

### A.1 Les fonctions

1. **DB\_ID()** : récupère l'identifiant de la base de données en cours d'exécution
2. **DB\_NAME()** : récupère le nom de la base de données en cours d'exécution
3. **SERVERPROPERTY(argument)** : récupère les propriétés principales du serveur de base de données
4. **DATABASEPROPERTYEX ( database , property )** : permet de retourner l'état de l'option de configuration courante de la base de données spécifiée

### A.2 Les procédures stockées

1. **sp\_helpdb** : affiche les informations sur les bases de données systèmes et applicatives
2. **sp\_configure** : permet d'afficher et de modifier les options de configura-

tion

3. **sp\_who** : affiche des informations sur les sessions actives sur le serveur
4. **sp\_spaceused** : Permet de connaitre l'espace utilisé, réservé et libre au niveau des objets.
5. **sp\_helpsrvrole** : affiche la liste des roles à l'échelle serveur
6. **sp\_addsrvrolemember** : permet d'ajouter un login en tant que membre d'un role serveur
7. **sp\_dropsrvrolemember** : cette procédure stockée va permettre de supprimer un membre d'un role serveur.
8. **sp\_locks** : affiche les informations sur les verrous
9. **sp\_setapprole** : Permet de créer un role d'application au niveau de la base de données, elle accepte deux paramètre d'entrée **@rolename** et **@password**
10. **sp\_addmessage** : procédure destiner à **créer un message** d'erreur
11. **sp\_altermessage** : procédure destiner à **modifier un message** d'erreur
12. **sp\_dropmessage** : procédure destiner à **supprimer un message** d'erreur

### A.3 Les vues

1. La vue **sys.index**
2. La vue **sys.filegroups**
3. La vue **sys.files**
4. La vue **sys.tables**
5. La vue **sys.dm\_tran\_active\_transactions**

6. La vue **sys.symmetric\_keys** : Nous permet de consulter les clé de chiffrement disponible au niveau de l'instance notamment la clé principale de service
7. La vue **sys.dm\_exec\_sessions** : la vue système qui affiche des informations sur les sessions actives sur le serveur
8. **sys.dm\_exec\_connections** : cette vue dynamique fournie des informations sur les connexions établie sur l'instance SQL server en cours d'exécution.
9. **La vue sys.server\_principal** : liste tous les utilisateurs qui peuvent se connecter au niveau de l'instance avec d'autre informations utiles comme les roles auxquels ils appartiennent.
10. **La vue sys.sql\_login** : liste les connexions au niveau serveur qui ont été créées avec **le mode d'authentification SQL server**
11. **La vue sys.database\_principal** : affiche la liste de l'ensemble des utilisateurs de base de données
12. **La vue sys.database\_permissions** : c'est une vue système qui stocke des informations sur les autorisations au niveau de la base de données
13. **La vue sys.dm\_tran\_locks** : Permet d'obtenir des informations sur les verrous dans le moteur de base de données SQL server
14. **La vue sys.tcp\_endpoints** : cette vue système nous permet d'obtenir des informations sur les modules d'écoutes
15. **sys.dm\_exec\_cached\_plans** : cette vue dynamique nous offre des informations associée au plan d'exécution mis en cache.
16. **sys.dm\_exec\_cached\_plans** : cette vue dynamique nous offre des informations associée au plan d'exécution mis en cache.
17. **msdb.dbo.sysoperators** : cette vue récupère une liste d'informations sur les opérateurs actuellement disponible et pris en charge par l'agent SQL

18. **msdb.dbo.sysmessages** : affiche la liste des messages d'erreur au niveau de la base de données SQL server
19. **sys.database\_mirroring\_endpoints** : affiche **l'état et les points de terminaison des serveurs** qui participent à la mise en miroir

## Appendix B

# Les commandes SQL usuelles

### B.0.1 Les commandes de création de base de données

1. **ALTER DATABASE "nom de la base" MODIFY FILE (spécificationFichier)**
2. **ALTER DATABASE "nom de la base" ADD FILE (spécificationFichier)**
3. **ALTER DATABASE "nomBase" REMOVE FILE NomLogique**
4. **ALTER DATABASE "nomBase" SET RECOVERY FULL—SIMPLE—BULK LOGGED**
5. **DBCC SHRINKDATABASE(id\_db ou db\_name,"target\_percent",NOTRUNCATE ou TRUNCATEONLY)**

### B.0.2 Les commandes de configuration du Réseau SQL server

1. **SELECT desc\_type, port FROM sys.tcp\_endpoints**

### B.0.3 Les commandes de sauvegarde et restauration

1. **BACKUP DATABASE** "databse\_name" **TO DISK** = "Chemin\_physique" **WITH NOFORMAT, NOINIT, CHECKSUM**
2. **BACKUP DATABASE** "databse\_name" **TO DISK** = "Chemin\_physique" **WITH NOFORMAT, NOINIT, CHECKSUM, DIFFERENTIAL**
3. **BACKUP LOG** "databse\_name" **TO DISK** = "Chemin\_physique" **COMPRESSION**
4. **RESTORE HEADERONLY FROM DISK** = "Chemin\_physique"
5. **RESTORE FILELISTONLY FROM DISK** = "Chemin\_physique"
6. **RESTORE VERIFYONLY FROM DISK** = "Chemin\_physique"

### B.0.4 Les commandes de transfert de données

1. **BULK INSERT** database.schema.table **FROM** = "Chemin\_physique" **WITH ( FIRSTROW= 2, FIELDTERMINATOR =',', ROWTERMINATOR='\'n')**

### B.0.5 Les commandes de création d'un login de connexion

1. **CREATE LOGIN** "name" **FROM WINDOWS**  
**WITH DEFAULT\_DATABASE="nom\_base"**  
**DEFAULT\_LANGUAGE="langue"**
2. **CREATE LOGIN** "name" **WITH PASSWORD = "motDePasse**  
**MUST\_CHANGED DEFAULT\_DATABASE="nom\_base"**  
**DEFAULT\_LANGUAGE="langue"**

3. **ALTER LOGIN "name" WITH DEFAULT\_DATABASE="nomBase", DEFAULT\_LANGUAGE="langue"**
4. **ALTER LOGIN "nom\_utilisateur" DISABLE**
5. **DENY CONNECT SQL TO "utilisateur\_name"**

### B.0.6 La commande de création d'un Crédential

1. **CREATE CREDENTIAL "NomCredential"**  
**WITH IDENTITY="Domaine \Utilisateur"**

### B.0.7 Les commandes de création d'un utilisateur

1. **CREATE USER "nameUser" FOR LOGIN "NameLogin"**  
**WITH DEFAULT\_SCHEMA = "schema"**
2. **CREATE USER "nameUser" WITHOUT LOGIN**
3. **GRANT CONNECT TO "guest"**
4. **ALTER USER "userName" WITH NAME="newName",**  
**DEFAULT\_SCHEMA="NomSchema"**

### B.0.8 Les commandes de création d'un schéma

1. **CREATE SCHEMA "nomSchema" AUTHORIZATION "UserName"**
2. **ALTER SCHEMA "nomSchema" TRANSFER "schema.objet"**
3. **DROP SCHEMA "NomSchéma"**

### B.0.9 Les commandes de création d'un rôle de base de données

1. **CREATE ROLE "nomRole" AUTHORIZATION "NameUSER"**
2. **DROP ROLE "nomRole"**

### B.0.10 Les commandes de gestion de privilèges

1. **GRANT "NomPrivilège" ON "objectName" TO "UserName"**  
**WITH GRANT OPTION**
2. **REVOKE "NomPrivilège" FROM "UserName"**  
**CASCADE**
3. **SELECT name, type\_desc, class\_desc, permission\_name**  
**FROM sys.database\_principals d1**  
**INNER JOIN sys.database\_permissions d2 ON**  
**d1.principal\_id=d2.grantee\_principal\_id**  
**WHERE name='guest'**
4. **SELECT name, type\_desc, class\_desc, permission\_name**  
**FROM sys.server\_principals d1**  
**INNER JOIN sys.server\_permissions d2 ON**  
**d1.principal\_id=d2.grantee\_principal\_id**  
**WHERE name='guest'**

### B.0.11 La commande de création d'un rôle d'application

1. **CREATE APPLICATION ROLE "NomApplication"**  
**WITH DEFAULT\_SCHEMA=NomSchema,**  
**PASSWORD='PWD'**

### B.0.12 Les commandes de mise en cache

1. **SELECT cp2.text, cp.cacheobjtype, cp.size\_in\_bytes/1024 AS size\_KB FROM sys.dm\_exec\_cached\_plans cp CROSS APPLY sys.dm\_exec\_sql\_text(cp.plan\_handle) cp2 CROSS APPLY sys.dm\_exec\_query\_plan(cp.plan\_handle) cp3 OPTION (RECOMPILE) GO**