



<http://www.laboratoire-microsoft.org>

Essentiel SQL Server 2000

INSTALLATION, CONFIGURATION ET
ADMINISTRATION DE MICROSOFT SQL
SERVER 2000 EDITION ENTERPRISE

Auteurs : ALET Yann, LHOMEL Guillaume, RIVALLAN
Jean-Yves

Versi on 0.99 – 5 Févri er 2004



Ecole Supérieure d'Informatique de Paris
23. rue Château Landon 75010 – PARIS
www.supinfo.com

Table des matières

1. VUE D'ENSEMBLE DE MICROSOFT SQL SERVER.....	7
1.1. PRESENTATION DE SQL SERVER	7
1.1.1. Composants client-serveur	7
1.1.2. Processus de communication client-serveur.....	7
1.1.3. Services SQL Server	8
1.2. INTEGRATION DE SQL SERVEUR 2000 AUX SYSTEMES D'EXPLOITATION	8
1.2.1. Active Directory.....	8
1.2.2. Sécurité.....	9
1.2.3. Prise en charge de plusieurs processeurs.....	9
1.2.4. Observateur d'événements Microsoft	9
1.2.5. Windows 2000 Component Services	9
1.2.6. Moniteur Système Windows 2000.....	9
1.2.7. Services Internet Microsoft.....	9
1.2.8. Windows Clustering.....	9
1.3. BASES DE DONNEES SQL SERVER	10
1.3.1. Types de bases de données	10
1.3.2. Objets de base de données	10
1.3.3. Référence aux objets SQL Server.....	11
1.3.4. Tables système	11
1.3.5. Extraction de métadonnées.....	12
1.4. SECURITE DE SQL SERVER	12
1.4.1. Authentification des comptes de connexion	12
1.4.2. Rôles et comptes d'utilisateur de base de données.....	12
1.5. UTILISATION DE SQL SERVER.....	13
1.5.1. Outils d'administration d'une base de données SQL Server	13
1.5.2. Implémentation d'une base de données SQL Server.....	13
2. PLANIFICATION DE L'INSTALLATION DE SQL SERVER.....	14
2.1. REMARQUES SUR L'INSTALLATION DU MATERIEL	14
2.1.1. Configuration matérielle minimale requise	14
2.1.2. Optimisation du matériel pour SQL Server	14
2.2. ÉDITIONS DE SQL SERVER 2000	15
2.3. REMARQUES SUR L'INSTALLATION DU LOGICIEL	15
2.3.1. Modes de licence	15
2.3.2. Choix du contexte de sécurité des comptes de service.....	16
2.3.3. Utilisation d'instances multiples et d'instances nommées de SQL Server.....	16
2.3.4. Choix d'un mécanisme de sécurité	16
2.3.5. Choix des classements et des règles de tri SQL Server.....	16
2.3.6. Choix des bibliothèques réseau	17
2.4. METHODES D'INSTALLATION DE SQL SERVER.....	17
2.4.1. Installation standard.....	17
2.4.2. Installation automatisée.....	18
2.4.3. Installation à distance.....	18
2.5. MISE A NIVEAU DE SQL SERVER.....	18

2.6.	VERIFICATION DE L'INSTALLATION	19
2.6.1.	Démarrage, arrêt et suspension des services SQL Server	19
2.6.2.	Connexion à SQL Server	19
2.7.	CONFIGURATION DE SQL SERVER ENTERPRISE MANAGER	19
2.8.	RESOLUTION DES PROBLEMES	20
3.	GESTION DES FICHIERS DE BASE DE DONNEES	21
3.1.	PRESENTATION DES STRUCTURES DE DONNEES	21
3.1.1.	Mode de stockage des données	21
3.1.2.	Fonctionnement du journal des transactions	22
3.2.	CREATION DE BASES DE DONNEES	22
3.2.1.	Processus de création d'une base de données	22
3.2.2.	Définition des options lors de la création d'une base de données	22
3.2.3.	Modification des options de base de données après la création de la base de données	23
3.2.4.	Affichage des propriétés de base de données	23
3.3.	GESTION DES BASES DE DONNEES	23
3.3.1.	Gestion de la croissance des fichiers journaux et de données	23
3.3.2.	Compaction automatique d'une base de données ou d'un fichier de base de données	23
3.3.3.	Compaction manuel d'une base de données ou d'un fichier de base de données	23
3.3.4.	Suppression d'une base de données	24
3.4.	EMPLACEMENT DES FICHIERS ET JOURNAUX DE BASE DE DONNEES	24
3.5.	OPTIMISATION D'UNE BASE DE DONNEES A L'AIDE D'UNE SOLUTION RAID MATERIELLE	24
3.5.1.	Création de groupes de fichiers définis par l'utilisateur	24
3.5.2.	Utilisation de groupes de fichiers pour améliorer les performances	24
3.6.	OPTIMISATION D'UNE BASE DE DONNEES EN UTILISANT DES GROUPES DE FICHIERS AVEC UNE SOLUTION RAID MATERIELLE	24
3.7.	PLANIFICATION DE LA CAPACITE	25
3.7.1.	Évaluation de la taille d'une base de données	25
3.7.2.	Évaluation de la quantité de données contenues dans les tables	25
4.	GESTION DE LA SECURITE	26
4.1.	IMPLEMENTATION D'UN MODE D'AUTHENTIFICATION	26
4.1.1.	Traitement de l'authentification	26
4.1.2.	Choix d'un mode d'authentification	26
4.1.3.	Emprunt d'identité et délégation	26
4.1.4.	Cryptage	26
4.1.5.	Étapes de l'implémentation du mode d'authentification	27
4.1.6.	Création de comptes de connexion	27
4.2.	ATTRIBUTION DE COMPTES DE CONNEXION A DES UTILISATEURS ET DES ROLES	27
4.2.1.	Rôles fixes de serveur	27
4.2.2.	Rôles fixes de base de données	28
4.2.3.	Rôles de base de données définis par l'utilisateur	28
4.3.	ATTRIBUTION D'AUTORISATIONS A DES UTILISATEURS ET DES ROLES	29
4.3.1.	Types d'autorisations	29
4.3.2.	Octroi, refus et révocation d'autorisations	29
4.4.	GESTION DE LA SECURITE DES APPLICATIONS	29
4.4.1.	Gestion de la sécurité à l'aide de vues et de procédures stockées	29

4.4.2.	<i>Gestion de la sécurité des applications clientes à l'aide de rôles d'application.....</i>	29
4.5.	GESTION DE LA SECURITE DE SQL SERVER DANS L'ENTREPRISE.....	30
4.5.1.	<i>Utilisation d'une stratégie de groupe pour sécuriser SQL Server.....</i>	30
4.5.2.	<i>Utilisation du cryptage pour la transmission des données en vue de leur sécurisation</i>	30
5.	EXECUTION DE TACHES ADMINISTRATIVES.....	31
5.1.	TACHES LIEES A LA CONFIGURATION.....	31
5.1.1.	<i>Configuration de l'Agent SQL Server.....</i>	31
5.1.2.	<i>Configuration de SQLAgentMail et de SQL Mail.....</i>	31
5.1.3.	<i>Configuration de serveurs liés.....</i>	31
5.1.4.	<i>Configuration des noms de source de données.....</i>	32
5.1.5.	<i>Configuration de la prise en charge du langage XML de SQL Server dans les services Internet</i>	32
5.1.6.	<i>Configuration de SQL Server pour partager les ressources de mémoire avec les autres applications serveur</i>	32
5.2.	TACHES DE ROUTINE LIEES A L'ADMINISTRATION DE SQL SERVER	33
5.3.	AUTOMATISATION DES TACHES DE MAINTENANCE DE ROUTINE.....	33
5.3.1.	<i>Automatisation de l'administration de SQL Server</i>	33
5.3.2.	<i>Création de travaux</i>	33
5.3.3.	<i>Définition des étapes d'un travail.....</i>	33
5.3.4.	<i>Création d'un organigramme des actions par étape de travail.....</i>	34
5.3.5.	<i>Planification de travaux</i>	34
5.3.6.	<i>Création d'opérateurs à notifier.....</i>	34
5.3.7.	<i>Analyse et configuration de l'historique des travaux</i>	35
5.4.	CREATION D'ALERTES	35
5.4.1.	<i>Utilisation d'alertes en réponse à des problèmes potentiels</i>	35
5.4.2.	<i>Consignation d'événements dans le journal applications</i>	35
5.4.3.	<i>Création d'alertes en réponse à des erreurs SQL Server</i>	35
5.4.4.	<i>Réponse à des alertes de conditions de performances.....</i>	36
5.4.5.	<i>Attribution d'un opérateur de prévention de défaillance.....</i>	36
5.5.	RESOLUTION DES PROBLEMES LIES A L'AUTOMATISATION DE SQL SERVER	36
5.6.	AUTOMATISATION DE TRAVAUX SUR PLUSIEURS SERVEURS.....	37
6.	SAUVEGARDE DE BASES DE DONNEES	39
6.1.	PROTECTION CONTRE LES PERTES DE DONNEES	39
6.2.	DEFINITION ET CHANGEMENT DE MODE DE RECUPERATION DE BASE DE DONNEES	39
6.3.	SAUVEGARDE DE SQL SERVER	40
6.4.	MOMENT APPROPRIE POUR SAUVEGARDER DES BASES DE DONNEES	40
6.4.1.	<i>Sauvegarde de bases de données système.....</i>	40
6.4.2.	<i>Sauvegarde de bases de données utilisateur.....</i>	40
6.4.3.	<i>Activités à éviter pendant les sauvegardes</i>	41
6.5.	EXECUTION DE SAUVEGARDES	41
6.5.1.	<i>Création d'unités de sauvegarde</i>	41
6.5.2.	<i>Utilisation de plusieurs fichiers de sauvegarde pour stocker les sauvegardes.....</i>	42
6.5.3.	<i>Spécification des options de bande.....</i>	42
6.6.	TYPES DE METHODES DE SAUVEGARDE	43
6.6.1.	<i>Sauvegarde de base de données complète</i>	43
6.6.2.	<i>Sauvegarde différentielle</i>	43
6.6.3.	<i>Sauvegarde du journal des transactions.....</i>	43
6.6.4.	<i>Sauvegarde d'un fichier ou d'un groupe de fichiers de base de données.....</i>	44

7.	RESTAURATION DE BASES DE DONNEES.....	45
7.1.	PROCESSUS DE RECUPERATION DE SQL SERVER.....	45
7.2.	RESTAURATION DE SAUVEGARDES	45
7.3.	RESTAURATION DE BASES DE DONNEES A PARTIR DE DIFFERENTS TYPES DE SAUVEGARDES	45
7.3.1.	<i>Restauration à partir d'une sauvegarde complète et différentielle de base de données.....</i>	45
7.3.2.	<i>Restauration à partir d'une sauvegarde du journal des transactions</i>	46
7.4.	RESTAURATION DE BASES DE DONNEES SYSTEME ENDOMMAGEES	46
8.	SURVEILLANCE DES PERFORMANCES DE SQL SERVER	47
8.1.	RAISONS JUSTIFIANT LA SURVEILLANCE DE SQL SERVER.....	47
8.2.	SURVEILLANCE ET AJUSTEMENT DES PERFORMANCES.....	47
8.2.1.	<i>Stratégies d'ajustement des performances.....</i>	47
8.2.2.	<i>Définition d'une référence en matière de performances et méthodologie d'ajustement.....</i>	47
8.3.	OUTILS DE SURVEILLANCE DE SQL SERVER	48
8.3.1.	<i>Observateur d'événements Windows 2000.....</i>	48
8.3.2.	<i>Moniteur système Windows avec SQL Server.....</i>	48
8.3.3.	<i>Fenêtre Activité en cours de SQL Server Enterprise Manager.....</i>	48
8.3.4.	<i>Outils Transact-SQL.....</i>	48
8.3.5.	<i>Générateur de profils SQL Server</i>	49
8.3.6.	<i>Analyseur de requêtes SQL Server</i>	49
9.	TRANSFERT DE DONNEES.....	50
9.1.	PRESENTATION DU TRANSFERT DE DONNEES	50
9.1.1.	<i>Raisons justifiant l'importation et l'exportation de données.....</i>	50
9.1.2.	<i>Raisons justifiant la transformation de données.....</i>	50
9.2.	OUTILS D'IMPORTATION ET D'EXPORTATION DE DONNEES DISPONIBLES DANS SQL SERVER....	50
9.2.1.	<i>Assistant Importation/exportation DTS</i>	50
9.2.2.	<i>Concepteur DTS</i>	50
9.2.3.	<i>Transfert d'objets DTS</i>	50
9.2.4.	<i>Insertion en bloc DTS</i>	50
9.2.5.	<i>Programme de copie en bloc (utilitaire bcp).....</i>	51
9.2.6.	<i>Réplication.....</i>	51
9.3.	PRESENTATION DES SERVICES DE TRANSFORMATION DE DONNEES	51
9.3.1.	<i>Vue d'ensemble des services DTS.....</i>	51
9.3.2.	<i>Outils des services DTS</i>	51
9.4.	TRANSFORMATION DE DONNEES A L'AIDE DES SERVICES DTS	52
9.4.1.	<i>Transformation et mappage des données</i>	52
9.4.2.	<i>Définition des tâches de transformation</i>	52
9.4.3.	<i>Définition des flux de travail</i>	52
9.4.4.	<i>Création d'un lot DTS</i>	53
9.4.5.	<i>Exécution et planification d'un lot DTS.....</i>	53
10.	MAINTIEN D'UNE HAUTE DISPONIBILITE	54
10.1.	PRESENTATION DE LA DISPONIBILITE	54
10.1.1.	<i>Définition de la disponibilité</i>	54
10.1.2.	<i>Détermination des besoins en matière de disponibilité</i>	54
10.1.3.	<i>Evolutivité.....</i>	54
10.1.4.	<i>Optimisation de la disponibilité avec les serveurs d'entreprise .NET de Microsoft.....</i>	55
10.2.	AMELIORATION DE LA DISPONIBILITE EN UTILISANT UN CLUSTER DE BASCULEMENT.....	55

10.2.1.	Utilisation d'un cluster de basculement SQL Server	55
10.2.2.	Clustering actif/passif.....	55
10.2.3.	Clustering actif/actif.....	56
10.3.	SERVEURS DE SECOURS ET TRANSMISSION DES JOURNAUX	56
10.3.1.	Maintien d'une haute disponibilité en utilisant un serveur de secours et la transmission des journaux	56
10.3.2.	Configuration de la transmission des journaux en utilisant l'Assistant Plan de maintenance de base de données.....	56
10.3.3.	Modification des rôles de transmission des journaux.....	56
11.	PRESENTATION DE LA REPLICATION	58
11.1.	PRESENTATION DES DONNEES DISTRIBUEES	58
11.1.1.	Besoin en données distribuées.....	58
11.1.2.	Facteurs à prendre en compte lors de la distribution des données	58
11.1.3.	Méthodes de distribution des données.....	58
11.2.	PRESENTATION DE LA REPLICATION SQL SERVER	59
11.2.1.	Métaphore éditeur-abonné	59
11.2.2.	Publications et articles	60
11.2.3.	Filtrage des données.....	60
11.2.4.	Abonnements.....	60
11.3.	AGENTS DE REPLICATION SQL SERVER	60
11.4.	TYPES DE REPLICATIONS SQL SERVER.....	61
11.4.1.	Vue d'ensemble des types de répliquions.....	61
11.4.2.	Remarques sur l'utilisation de la répliquion de fusion.....	62
11.5.	MODELES PHYSIQUES DE REPLICATION	62
11.5.1.	Le modèle Éditeur central/Distributeur distant.....	62
11.5.2.	Le modèle Abonné central/Éditeurs multiples	63
11.5.3.	Le modèle Éditeurs multiples/Abonnés multiples	64

1. Vue d'ensemble de Microsoft SQL Server

1.1. Présentation de SQL Server

SQL Server est un système de gestion de bases de données relationnelles (SGBDR) répondant aux exigences professionnelles du stockage de données. SQL Server prend en charge nativement pour la communication de requêtes entre client et serveur :

- Transact-SQL (variante du SQL pour l'écriture des procédures stockées, entre autres)
- XML (eXtensible Markup Language, un langage à balises permettant la représentation de données arbitraires)
- MDX (Multidimensional eXpressions)
- SQL-DMO (SQL-Distributed Management Objects)

Pour cela, deux modèles de stockage de données sont mis à disposition par SQL Server:

- OLTP (optimise la vitesse de mise à jour et de traitement)
- OLAP (permet la création de rapports et une analyse en entreprise)

1.1.1. Composants client-serveur

L'architecture de SQL Server est divisée en :

- une partie *serveur* : gestion des bases de données, répartition des ressources entre les différentes demandes des clients (effectue le traitement lourd)
- une partie *client* : affichage des résultantes de requêtes (effectue le traitement léger (renvoi du traitement lourd))

Composants Serveur :

- Bibliothèques réseau serveur (bibliothèque d'abstraction d'une pile réseau donnée ex : TCP/IP, IPX/SPX)
- Services Open Data (composant gérant les bibliothèques réseau serveur activées)
- Moteur Relationnel
- Moteur de stockage

Composants Clients :

- Bibliothèque réseau cliente (permet une abstraction de la pile réseau utilisée)
- API de base de données
- Application cliente (Analyseur de requêtes)

1.1.2. Processus de communication client-serveur

Lors de la soumission d'une requête client, les opérations suivantes sont réalisées :


- Le client fait appel à l'API de base de données et lui soumet la requête qui est transformée au format TDS (Tabular Data Stream)
- L'API soumet la requête à la bibliothèque réseau client qui encapsule les paquets TDS dans des paquets de couche 3 et les transmet sur le réseau
- La bibliothèque réseau serveur reçoit et réassemble les paquets sous forme exploitable par l'API (TDS) et les passe au service Open Data
- Open Data fait exécuter la requête par le moteur relationnel

- Le moteur relationnel analyse et interprète la requête et fait appel au moteur de stockage pour les actions à réaliser en lecture ou écriture sur les fichiers
- Le moteur relationnel définit la réponse et l'envoie au client suivant le chemin inverse de celui de la requête

1.1.3. Services SQL Server

Les fonctionnalités de SQL Server reposent sur l'exécution de 4 services :

- **MSSQLServer** gère les requêtes, les transactions et assure l'intégrité des données
- **MSSQLServerAgent** permet la remontée d'informations sur l'état de SQL Server
- **Microsoft Distributed Transaction Coordinator** prend en charge la gestion des transactions distribuées
- **Microsoft Search** gère l'indexation en texte intégral

 Il est possible d'héberger plusieurs instances de SQL Server sur une machine chacune étant indépendante des autres. Pour se connecter à une instance en particulier, le couple ordinateur/instance doit être précisé. En cas d'omission de l'instance, la requête est transférée à l'instance par défaut.

1.2.Intégration de SQL Serveur 2000 aux systèmes d'exploitation

Les composants clients de SQL serveur 2000 sont compatibles avec :

- Windows NT4 et supérieur
- Windows 95 et supérieur

Les différentes éditions de SQL Server permettent aux composants serveurs de s'exécuter sur toutes les éditions de Windows 2000, les versions de Windows NT, Windows Me, Windows 98 et Windows CE. Certaines versions spécifiques des systèmes d'exploitation et les éditions de SQL Server limitent les composants serveur.

Microsoft Windows NT Server 4.0, Service Pack 5 (SP5) ou ultérieur doit au minimum être installé pour toutes les éditions de SQL Server 2000. Seuls les composants serveur, tels que le moteur de base de données et Analysis server, sont limités à des versions précises de systèmes d'exploitation. Ainsi, bien que le moteur de base de données pour l'Édition Entreprise de Microsoft SQL Server 2000 ne s'exécute pas sur Microsoft Windows 2000 Professionnel, Microsoft Windows NT Workstation, Windows Me ou Windows 98, vous pouvez utiliser le CD-ROM SQL Server 2000 Édition Entreprise pour installer les composants clients sur l'un de ces systèmes d'exploitation.

Windows NT 4.0 Terminal Server ne prend pas en charge SQL Server 2000.

SQL Server est entièrement intégré à Windows 2000 et bénéficie de plusieurs de ses fonctionnalités.

1.2.1. Active Directory

Les serveurs et leurs attributs sont inscrits automatiquement dans le service d'annuaire Active Directory™ au démarrage du serveur. Les utilisateurs peuvent rechercher un serveur particulier et le localiser à l'aide de la fonctionnalité de recherche d'Active Directory. Par exemple, un utilisateur peut utiliser l'annuaire pour localiser tous les serveurs exécutant une ou plusieurs instances de SQL Server avec un nom de base de données particulier qui leur est associé.

1.2.2. Sécurité

SQL Server est intégré au système de sécurité de Windows 2000. Ainsi, avec un nom d'utilisateur et un mot de passe uniques, il est possible d'accéder à la fois à SQL Server et à Windows 2000. SQL Server utilise également les fonctionnalités de cryptage dans Windows 2000 pour assurer la sécurité du réseau, et prend notamment en charge Kerberos. SQL Server fournit sa propre sécurité aux clients qui doivent accéder à SQL Server sans être authentifiés par Windows 2000.

1.2.3. Prise en charge de plusieurs processeurs

SQL Server prend en charge les fonctions de multitraitement symétrique (SMP, *Symmetric Multi-Processing*) de Windows 2000 et tire automatiquement parti de tous les processeurs supplémentaires ajoutés au serveur.

1.2.4. Observateur d'événements Microsoft

SQL Server écrit des messages dans les journaux Système, Sécurité et Application de Windows 2000, offrant ainsi un mécanisme cohérent pour la détection et l'affichage des problèmes.

1.2.5. Windows 2000 Component Services

Component Services repose sur des extensions du modèle COM (*Component Object Model*) et de Microsoft Transaction Server. Il offre de meilleures performances dans les domaines suivants : gestion des threads, sécurité, gestion des transactions, réutilisation des objets, composants mis en attente, administration des applications et conditionnement d'applications. Par exemple, les développeurs de logiciels peuvent utiliser Component Services pour configurer de manière visuelle le comportement habituel des composants et des applications, comme la sécurité et la participation aux transactions, ainsi que pour intégrer des composants à des applications COM+.

1.2.6. Moniteur Système Windows 2000

SQL Server envoie au Moniteur système Windows 2000 des informations sur les performances, ce qui vous permet de surveiller les performances système de SQL Server.

1.2.7. Services Internet Microsoft

SQL Server utilise les services Internet (IIS, *Internet Information Services*) Microsoft pour que les navigateurs Internet puissent accéder à une base de données SQL Server par l'intermédiaire du protocole HTTP.

1.2.8. Windows Clustering

Windows Clustering, un composant de Windows 2000 Advanced Server, prend en charge la connexion de deux serveurs, ou nœuds, au sein d'un cluster pour une disponibilité plus élevée et une plus grande facilité de gestion des données et des applications. SQL Server fonctionne conjointement avec Windows Clustering pour basculer automatiquement vers le nœud secondaire en cas de défaillance du nœud principal.

1.3. Bases de données SQL Server

1.3.1. Types de bases de données

SQL Server héberge deux types de bases de données, les bases de données système et les bases de données utilisateur. Les bases de données système contiennent des informations nécessaires au bon fonctionnement du moteur SQL Server et des services associés. Les bases de données utilisateur sont les bases de données créées et utilisées par les utilisateurs.

Lors de l'installation de SQL Server, le programme d'installation crée des bases de données système et des exemples de bases de données utilisateur. La base de données distribution est installée lorsque vous configurez SQL Server pour des activités de réplication.

Description des bases de données installées par défaut :

- **master** : Contrôle les bases de données utilisateur et le fonctionnement global de SQL Server en effectuant le suivi d'informations telles que les comptes d'utilisateur, les variables d'environnement configurables et les messages d'erreur du système.
- **model** : Offre un modèle ou prototype pour les nouvelles bases de données utilisateur.
- **tempdb** : Offre une zone de stockage pour les tables temporaires et les autres besoins de stockage temporaire.
- **msdb** : Offre une zone de stockage pour les informations de programmation et l'historique des travaux.
- **distribution** : Stocke les données relatives à l'historique et aux transactions utilisées dans la réplication.
- **pubs** : Propose un exemple de base de données comme outil d'apprentissage.
- **Northwind** : Propose un exemple de base de données comme outil d'apprentissage.

1.3.2. Objets de base de données

Une base de données est un ensemble de données, de tables et autres objets. Les objets de base de données vous aident à structurer vos données et à définir les mécanismes d'intégrité des données.

Description des objets de base de données SQL Server :

- **Table** : Définit un ensemble de lignes ayant des colonnes associées.
- **Type de données** : Définit les valeurs de données autorisées pour une colonne ou une variable. SQL Server offre les types de données fournis par le système. Les utilisateurs créent des types de données définis par l'utilisateur.
- **Contrainte** : Définit les règles relatives aux valeurs autorisées dans les colonnes et constitue le mécanisme standard pour garantir l'intégrité des données.
- **Valeurs par défaut** : Définit une valeur qui est stockée dans une colonne si aucune autre valeur n'est fournie.
- **Règle** : Contient des informations qui définissent les valeurs valides qui sont stockées dans une colonne ou un type de données.
- **Index** : Constitue une structure de stockage offrant un accès rapide pour l'extraction de données, et pouvant garantir l'intégrité des données. Dans un index ordonné, les enregistrements de la table sont physiquement stockés suivant l'ordre des valeurs de la ou des clés de l'index. Dans un index non ordonné, l'ordre logique de l'index ne correspond pas à l'ordre stocké physique des lignes de la table.
- **Vue** : Permet de visualiser des données provenant d'une ou de plusieurs tables ou vues d'une même base de données. Fonction définie par l'utilisateur Renvoie soit une valeur scalaire, soit une table. Les fonctions permettent d'encapsuler une logique exécutée fréquemment. Tout code qui doit exécuter la

logique incorporée dans une fonction peut appeler cette fonction plutôt que répéter toute la logique de la fonction.

- **Procédure stockée** : Constitue un ensemble nommé d'instructions Transact-SQL pré compilées devant être exécutées ensemble.

- **Déclencheur** : Constitue une forme spéciale de procédure stockée exécutée automatiquement lorsqu'un utilisateur modifie des données d'une table ou d'une vue.

1.3.3. Référence aux objets SQL Server

Vous pouvez faire référence aux objets SQL Server de plusieurs manières. Vous pouvez spécifier le nom entier de l'objet (nom complet) ou ne spécifier qu'une partie du nom de l'objet et laisser SQL Server déterminer le reste du nom en fonction de votre contexte de travail.

Noms complets

Le nom complet d'un objet de SQL Server comprend quatre identificateurs :

- le nom du serveur
- le nom de la base de données
- le nom du propriétaire
- le nom de l'objet, au format suivant : *serveur.base_données.propriétaire.objet*

Un nom d'objet qui spécifie les quatre composants est un nom complet. Chaque objet créé dans SQL Server doit posséder un nom complet unique. Par exemple, la même base de données peut comporter deux tables appelées **Orders** (commandes) si elles ont des propriétaires différents. De plus, les noms de colonne doivent être uniques au sein d'une table ou d'une vue.

Noms incomplets

Lorsque vous faites référence à un objet, vous n'êtes pas toujours tenu de spécifier le serveur, la base de données et le propriétaire. Des identificateurs intermédiaires peuvent être omis si leur position est indiquée par des points.

La liste suivante présente les formats valides pour les noms d'objet :

serveur.base_données.propriétaire.objet
base_données.propriétaire.objet
base_données..objet
propriétaire.objet
objet

1.3.4. Tables système

SQL Server stocke des informations (métadonnées) relatives au système et aux objets dans les bases de données pour une instance de SQL Server. Les *métadonnées* sont des informations relatives aux données.

Les métadonnées comprennent des informations sur les propriétés des données, telles que le type de données d'une colonne (numérique, texte, etc.) ou la longueur d'une colonne. Il peut également s'agir d'informations sur la structure des données, ou d'informations qui spécifient la conception des objets.

Tables systèmes : Les informations sur les données dans les tables système incluent des informations sur la configuration ainsi que des définitions de toutes les bases de données et de tous les objets de base de données dans l'instance de SQL Server. Les utilisateurs ne doivent pas directement modifier les tables système.

Catalogue de base de données : Chaque base de données (y compris **master**) contient un ensemble de tables système qui stockent les métadonnées relatives à cette base de données. Cet ensemble de tables système constitue le catalogue de la base de données. Il contient la définition de tous les objets de la base de données, ainsi que des autorisations.

Catalogue système : Le catalogue système, situé uniquement dans la base de données **master**, est un ensemble de tables système qui stockent les métadonnées relatives au système entier et à toutes les autres bases de données.

1.3.5. Extraction de métadonnées

Lorsque vous développez des applications conçues pour extraire des informations à partir des tables système, vous devez utiliser :

- les procédures stockées système
- les fonctions système
- les vues de schémas d'informations fournies par le système.

Vous pouvez exécuter une requête sur une table système de la même manière que sur une table d'une base de données pour extraire des informations concernant le système. Toutefois, il est déconseillé de rédiger des scripts qui interrogent directement les tables système car en cas de modification de ces dernières dans des versions ultérieures du produit, vos scripts peuvent échouer ou ne pas fournir les informations souhaitées.

1.4.Sécurité de SQL Server

1.4.1. Authentification des comptes de connexion

Deux types d'authentification sont disponibles dans SQL Server.

- L'**authentification Windows** qui utilise le compte Active Directory ou SAM de l'utilisateur pour vérifier son identité.
- L'**authentification SQL Serveur** qui utilise des comptes disjoints des comptes d'ouverture de session. Dans ce mode, un login et un mot de passe sont requis.

SQL Server peut être configuré pour n'accepter que l'authentification Windows ou les deux types d'authentification (mode mixte).

Avant de traiter la requête d'un utilisateur, SQL serveur effectue une vérification des autorisations de l'utilisateur en question. Ceci permet de vérifier si ce dernier a oui ou non le droit d'accéder aux informations ciblées par la requête. Si oui, le serveur SQL traite la requête, si non, le serveur SQL renvoie une erreur à l'utilisateur.

1.4.2. Rôles et comptes d'utilisateur de base de données

Un rôle est à SQL serveur ce qu'est un groupe à Windows. Il s'agit d'une entité qui regroupera un ensemble d'utilisateurs devant avoir les mêmes autorisations d'accès à une ressource.

Nous pouvons distinguer 3 types de rôles :

- **Rôles fixes de serveur :** permettent de gérer le serveur (Gestion de la base de données, gestions des fichiers physiques, gestions des processus, Audit, etc...)
- **Rôles fixe de base de données :** permettent de gérer les bases de données (Gestion des autorisations, gestions d'utilisateurs, lecture des données, etc...)

- **Rôles de base de données définis par l'utilisateur** : permettent aux utilisateurs de créer de nouveaux rôles en fonctions des travaux devant être effectués par un ensemble d'utilisateurs.

1.5.Utilisation de SQL Server

1.5.1. Outils d'administration d'une base de données SQL Server

SQL Server est fourni avec une suite d'outils dédiés aux taches d'administration :

Outil graphique	Objet
Utilitaire Réseau Client	Utilitaire de gestion de la configuration des bibliothèques réseau des clients
Utilitaire Réseau Serveur	Utilitaire de gestion de la configuration des bibliothèques réseau des serveurs
Générateur de profils	Utilitaire permettant de capturer un enregistrement continu de l'activité du serveur et offrant des fonctions d'audit
Analyseur de requêtes	Outils d'interrogation graphique utilisés pour analyser le plan d'une requête, analyser les informations statistiques et gérer simultanément plusieurs requêtes dans différentes fenêtres
Service Manager	Utilitaire graphique permettant de démarrer, arrêter et suspendre des services SQL Server
Programme d'installation de SQL Server	Application qui permet d'installer et de configurer SQL Server
Assistants de SQL Server	Ensemble d'outils qui guident les utilisateurs au cours de tâches complexes
Outil en ligne de commande	Objet
osql	Utilitaire utilisant la connectivité ODBC pour communiquer avec SQL Server (utilisé principalement pour exécuter des fichiers de traitement par lots contenant une ou plusieurs instructions SQL)
bcp	Utilitaire de traitement par lots permettant d'importer et d'exporter des données depuis/vers SQL Server (c'est-à-dire copier les données depuis/vers un fichier de données dans un format spécifié par l'utilisateur)

1.5.2. Implémentation d'une base de données SQL Server

L'implémentation d'une base de données SQL Server passe par plusieurs étapes :




- **Conception de la base de données** à l'aide d'une méthode de modélisation
- **Création de la base de données** avec les outils de gestion de SQL Server
- **Test et ajustement de la base de données**, vérification de la juste utilisation des index
- **Planification du déploiement de la base de données**

2. Planification de l'installation de SQL Server

2.1. Remarques sur l'installation du matériel

2.1.1. Configuration matérielle minimale requise

La fréquence du processeur et la quantité de mémoire nécessaires varient selon l'édition de SQL Server désirée. Cependant, voici les configurations minimales :

	Windows 2000 Professionnel
 Processeurs	Pentium 166 ou plus
 Mémoire vive	256 Mo au minimum (toutes les versions serveur de Windows 2000 et 2003) 64 Mo au minimum (toutes les versions serveur de Windows NT 4.0)
 Disque dur	250 Mo (installation minimum), 100 Mo (outils clients), 270 Mo (installation complète), 50 à 130 Mo (Analysis Services), 80 Mo (English Query)

2.1.2. Optimisation du matériel pour SQL Server

Pour des gains en performance, il peut être intéressant d'optimiser la machine serveur, spécialement les points suivants :

- **Processeur** : SQL Server supporte jusqu'à 32 processeurs, sur des ordinateurs SMP (Symmetrical MultiProcessing). Seuls Windows 2000 et 2003 Datacenter peuvent supporter ce type de configuration
- **Sous-système disque** : l'implémentation du RAID (matériel ou logiciel) augmentera performance, fiabilité, stockage et capacité.

Type de Raid	Avantages	Inconvénients
Raid 1	Gains de performances en lecture / écriture Très bonne redondance des données	
Raid 5	Gains de performance en lecture Très bonne redondance des données	Performances en écriture moyennes
Raid 10	Performances en lecture / écriture et redondance des données maximales	Beaucoup de disques

- **Mémoire** : plus le système disposera de mémoire physique, moins l'ordinateur aura besoin de faire des lectures sur disques pour répondre aux requêtes. En ajoutant de la mémoire physique, la mise en cache accélèrera les traitements.

- **Réseau** : dans une architecture client-serveur, la qualité du réseau est essentielle, tant au niveau de la bande passante disponible qu'au niveau de la qualité et vitesse des cartes réseaux.

2.2.Éditions de SQL Server 2000

Il existe 6 éditions de SQL Server 2000 :

- **Entreprise** : En environnement de production Client/Serveur, cette version regroupe toutes les fonctionnalités avec la prise en charge de sites Web, OLTP, le Data Warehousing. Voici dans le tableau suivant toutes les fonctionnalités :

Moteur de base de données relationnelle	Services d'analyse
Cluster avec basculement	Partitions OLAP
Transmission des journaux	Assistant partitions
Instruction CREATE INDEX parallèle	Cubes OLAP
Vues indexées	Prise en charge ROLAP
Prise en charge SAN	Prise en charge http
Prise en charge d'utilitaires graphiques	Dimensions activées en écriture

- **Entreprise d'évaluation** : Cette version est identique à la précédente mais n'est utilisable que 120 jours. Elle est téléchargeable gratuitement.
- **Standard** : En environnement de production Client/Serveur, cette version regroupe toutes les fonctionnalités de l'édition Entreprise, sauf pour ce qui est des sites Web, OLTP et le Data Warehousing. Cette version convient à de petites exploitations, avec des bases de données plus légères.
- **Développeur** : Cette version, identique à l'édition entreprise, est réservée au développement d'applications basées sur SQL Server, aux tests et non à l'exploitation. Cette version peut être installée sur Windows XP, 2000 Professionnel et NT Workstation.
- **Personnelle** : Cette version est destinée aux utilisateurs ayant besoin d'exploiter et stocker localement leurs données. Cette version peut être installée sur Windows XP, 2000 Professionnel, NT Workstation, 98 et ME.
- **Windows CE** : Cette version allégée permet le stockage de données sur des périphériques exploitant Windows CE. Il est alors possible pour l'utilisateur de synchroniser ses données avec des éditions Entreprise et Standard d'SQL Server.

2.3.Remarques sur l'installation du logiciel

2.3.1. Modes de licence

Il existe 2 types de licences pour SQL Server 2000 :

- **par processeur** : une licence est nécessaire pour chaque processeur exécutant SQL Server 2000. Le nombre de connexions réseaux vers cette licence est illimité.
- **par siège** : chaque serveur SQL dispose de sa licence. Chaque client se connectant à une instance SQL Server doit également disposer de sa licence (CAL, *Client Access License*). Pour une entreprise comptant un grand nombre d'utilisateurs, la licence par processeur est souvent plus rentable que le système de licence par sièges.

Par contre, les licences par sièges conviennent aux petites entreprises ayant peu d'utilisateurs mais une utilisation intensive.

2.3.2. Choix du contexte de sécurité des comptes de service

Un service SQL Server nécessite un compte d'ouverture de session affecté. Ce compte peut être un compte d'utilisateur local ou de domaine. Exemples de services : SQL Server et SQL Server Agent.

Lors de l'installation de SQL Server, à partir d'un compte utilisateur du domaine, SQL Server autorise ce compte à ouvrir une session en tant que service sur l'ordinateur SQL Server.

Cette fonctionnalité est intéressante dans les cas suivants :

- SQL Server doit accéder à des fichiers sur d'autres ordinateurs du domaine
- SQL Server devra notifier l'avancement de travaux par envois de courriers électroniques
- Cohabitation avec d'autres produits serveurs sur la même machine (Exchange par exemple)

Il ne faut pas oublier aussi de paramétrer le compte utilisateur : le mot de passe ne doit pas expirer, ...

Dans le cas d'un compte d'utilisateur local, SQL Server ne pourra pas communiquer avec d'autres serveurs requerrant des connexions approuvées Windows.

Lors de l'installation, vous pouvez choisir de démarrer manuellement ou automatiquement le service SQL Server. Cette fonctionnalité permet à l'administrateur de ne pas se connecter pour démarrer le service, lors d'un redémarrage de la machine par exemple.

2.3.3. Utilisation d'instances multiples et d'instances nommées de SQL Server

Plusieurs types d'instances existent :

- **instance par défaut** : une seule instance par défaut est autorisée. Elle reprend le nom réseau de la machine exécutant SQL Server.
- **Instance nommée** : Elle reprend le nom réseau de la machine exécutant SQL Server suivi d'un nom d'instance. Pour ce connecter à ce type d'instance, il est indispensable d'utiliser des composants client de SQL Server 2000.

Il est possible d'exécuter plusieurs instances sur un même serveur, qui fonctionneront toutes indépendamment les unes des autres. Il pourra alors y avoir une instance par défaut et plusieurs instances nommées configurées avec leurs paramètres d'annuaire personnels.

2.3.4. Choix d'un mécanisme de sécurité

De même, il existe plusieurs possibilités concernant l'authentification :

- **Mode d'authentification Windows** : Ce mode reprend l'authentification du compte d'utilisateur Windows 2003, 2000 ou NT valide. Le système d'exploitation devra alors valider l'utilisateur.
- **Mode mixte** : l'utilisateur a le choix d'utiliser soit l'authentification Windows, soit l'authentification SQL Server.

2.3.5. Choix des classements et des règles de tri SQL Server

Lors de l'installation de SQL Server, il vous est demandé de sélectionner un type de classement. Ceci correspond à un ensemble de règles qui régissent la manière dont les données vont être triées et comparées. Ces règles vont définir entre autres la séquence de caractères corrects, la casse ...

Vous avez alors le choix entre 2 types de classements :

- **classement Windows** : le classement est basé sur les paramètres régionaux de Windows

- **classement SQL** : dans le cas où une réplication sera nécessaire avec des versions antérieures de SQL Server, ou lorsque le code application dépend de classements SQL précédents. SQL Server mappe les attributs de combinaisons courantes de numéro de page de code et d'ordre de tri qui ont été spécifiés lors d'installation de versions antérieures de SQL Server.

Si jamais vous ne spécifiez pas de classement particulier, le classement Windows et le classement SQL sont sélectionnés par défaut.

2.3.6. Choix des bibliothèques réseau

Par défaut, à l'installation de SQL Server, voici les bibliothèques réseau sélectionnées :

- canaux nommés, bibliothèque réseau serveur sockets TCP/IP pour SQL Server
- bibliothèque réseau cliente sockets TCP/IP pour les outils de gestion
- canaux nommés comme deuxième bibliothèque réseau cliente sur toutes les versions de Windows 2003, 2000 et NT4.

Voici maintenant la liste des bibliothèques réseaux serveur prises en charge par SQL Server :

Sockets TCP/IP
Canaux nommés
NWLink IPX/SPX
VIA ServerNet II SAN
VIA GigaNet SAN
Multiprotocole
AppleTalk ADSP (<i>AppleTalk Data Stream Protocol</i>)
Banyan Vines

2.4. Méthodes d'installation de SQL Server

Le programme d'installation propose 3 types d'installations : minimum, par défaut et personnalisée.

L'installation se fait soit depuis un CD-ROM, soit depuis le réseau (répertoire partagé).

Pour installer SQL Server, il est nécessaire de se connecter sur la station d'accueil avec un compte membre du groupe local Administrateurs.

2.4.1. Installation standard

Lors de l'installation standard, si vous ne touchez à rien en particulier, une installation par défaut sera sélectionnée. Par contre, en cliquant sur le bouton *Personnalisée*, il est possible de choisir manuellement les composants à installer ou non.

Voici un tableau récapitulatif des composants compris dans les différents types d'installation :

Composant	Installation Minimum	Installation par défaut	Installation personnalisée
Serveur de base de données	Oui	Oui	Facultatif
Outils de mise à niveau	Non	Oui	Facultatif

Support de réplication	Oui	Oui	Facultatif
Texte Intégral	Non	Oui	Facultatif
Outils de gestion de client	Non	Oui	Facultatif
Connectivité du client	Oui	Oui	Non
Documentation en ligne	Non	Oui	Facultatif
Outils de développement	Non	Débogueur seul	Choix
Exemples de code	Non	Non	Choix
Paramètres de classement	Oui	Oui	Choix

2.4.2. Installation automatisée

L'installation automatisée va permettre d'installer plusieurs instances ayant la même configuration sans passer par l'installation interactive.

Cette installation se fait à l'aide de deux fichiers :

- **un fichier de commandes** : ce fichier détecte la plate-forme d'installation, spécifie les arguments d'installation et fait appel au fichier d'initialisation d'installation
- **un fichier d'initialisation d'installation** spécifiant toutes les options choisies par l'utilisateur avant l'installation. Ce fichier peut être créé soit en utilisant un éditeur de texte, soit en utilisant le programme d'installation de manière interactive.

Des fichiers exemples et modèles sont fournis avec SQL Server 2000.

Par exemple, pour supprimer SQL Server :

- prendre **Sqlrem.bat** en tant que fichier de commandes
- prendre **Sqlrem.iss** en tant que fichier d'initialisation d'installation.

2.4.3. Installation à distance

Pour l'installation à distance, il va vous être demandé de spécifier un compte d'utilisateur autorisé à effectuer l'installation sur l'ordinateur distant. Ce compte doit être administrateur sur l'ordinateur distant et disposer des droits de lecture sur le dossier source des fichiers d'installation.

L'installation se déroule alors en deux étapes :

- le programme d'installation enregistre les paramètres spécifiés par l'utilisateur dans un fichier setup.iss
- démarrage de l'installation avec démarrage d'un service distant, puis copie des fichiers dans le dossier partagé Admin\$ et enfin exécution de l'installation avec les options spécifiées dans le fichier setup.iss

2.5. Mise à niveau de SQL Server

Il est possible de mettre à niveau à partir de 3 versions différentes de SQL Server :

- **SQL Server 7.0** : soit on installe une instance par défaut qui fera une mise à jour de l'instance existante, soit on installe une nouvelle instance nommée et l'installation existante sera alors conservée.
- **SQL Server 6.5** : il est possible d'installer une instance par défaut ou une instance nommée, ce qui reviendra à la création de nouvelles instances.
- **SQL Server 6.0** : une mise à jour vers SQL Server 7.0 ou 6.5 doit d'abord être effectuée.

2.6. Vérification de l'installation


2.6.1. Démarrage, arrêt et suspension des services SQL Server

Comme nous l'avons vu précédemment, pour utiliser SQL Server, le service SQL Server doit être démarré soit automatiquement, soit manuellement.

Pour configurer le type de démarrage, il vous est possible d'utiliser les outils suivants : SQL Server Enterprise Manager, le Gestionnaire de Services SQL Server, Services (Gestion de l'ordinateur).

Vous pouvez utiliser les mêmes outils pour tout ce qui est manuel et utiliser également **net** à l'invite de commande DOS. Par exemple, **net stop sqlserveragent**, est la commande permettant d'arrêter une instance par défaut.

Service	Démarrage	Suspension	Arrêt
SQL Server	Les utilisateurs peuvent établir de nouvelles connexions	Empêche les nouvelles connexions	Désactive les ouvertures de sessions Réalise un point de contrôle pour chaque base de données Ferme SQL Server une fois que les instructions T-SQL et les procédures stockées sont traitées
SQL Server Agent	Automatisation d'activités Activation des alertes	Empêche les activités automatiques et les alertes	Empêche les activités automatiques et les alertes

 Pour arrêter une instance de SQL Server sans attendre la fin des transactions, l'instruction Transact-SQL « SHUTDOWN WITH NOWAIT » doit être utilisée.

2.6.2. Connexion à SQL Server

2 outils permettent de vérifier la connexion à SQL Server :

- un outil graphique tel que l'**analyseur de requêtes SQL** ou **SQL Server Enterprise Manager**
- un outil en ligne de commande tel que **osql** qui utilise la connectivité ODBC pour la communication avec SQL Server.

2.7. Configuration de SQL Server Enterprise Manager

Il est nécessaire d'inscrire un serveur local ou distant SQL Server afin de pouvoir l'administrer grâce à SQL Server Enterprise Manager.

Pour ce faire, il suffit de spécifier le nom du serveur en question, l'instance nommée (si il y a lieu), l'authentification et un groupe de serveurs.

Un groupe de serveurs est utilisé pour des tâches de gestion et d'administration de plusieurs serveurs ou plusieurs instances.

Afin d'administrer SQL Server, vous devez être membre du rôle sysadmin sur l'instance en question.

2.8. Résolution des problèmes

Pour la résolution des problèmes, une démarche type est recommandée :

- **Lecture des messages d'erreur** : ce sont les messages apparaissant pendant l'installation
- **Lecture du fichier Sqlstp.log** : fichier stocké dans c:\Winnt\ détenant toute la sortie des scripts d'installation générant des erreurs de contrôle de cohérence de bases de données
- **Lecture des fichiers journaux** : journal Sqlstp, journal Applications de Windows 2000, Journal des erreurs de SQL Server
- **Tests des connexions réseaux**
- **Recherche d'une solution connue** parmi les solutions existantes en ligne.

3. Gestion des fichiers de base de données

3.1. Présentation des structures de données

3.1.1. Mode de stockage des données

La création d'une base de données entraîne la création de 3 types de fichiers :

- **Un fichier de données .mdf** : fichier de données principal. 1^{er} fichier créé par défaut.
- **Un fichier de données .ndf** : fichier de données secondaire. Fichiers créés par l'utilisateur.
- **Un ou plusieurs fichiers journaux (.ldf)** : Stockage de toutes les informations requises pour la restauration de la base de données en cas de panne système (on parle aussi de fichier journal de transaction). Fichiers dans lesquels sont enregistrés toutes les opérations effectuées au sein de la base de données.

Dans le processus de création d'une base de données, il faut également considérer les points suivants :

- Une base de donnée peut stocker jusqu'à 128 pages (blocs de données de 8Ko) par Mo.
- Les lignes (ou enregistrements) ne peuvent pas s'étendre sur plusieurs pages.
- Vous pouvez stocker jusqu'à 8060 octets de données sur une ligne (en déduisant l'espace requis pour l'en-tête).

Les modifications issues d'une transaction ne sont sauvegardées au niveau d'une base de données que si toutes les instructions de cette transaction ont pu être exécutées correctement.

Nous pouvons distinguer 2 types de transactions :

Transactions Implicites	Transactions explicites
Elles sont effectuées si les instructions suivantes sont utilisées : <ul style="list-style-type: none">• ALTER TABLE• CREATE• DELETE• DROP• FETCH• GRANT• INSERT• OPEN• REVOKE• SELECT• TRUNCATE TABLE	Elles sont précisées par l'utilisateur. Toutes les instructions devant faire partie d'une transaction devront être précédées de l'instruction : BEGIN TRANSACTION et se terminer par la commande : COMMIT TRANSACTION .

- UPDATE

3.1.2. Fonctionnement du journal des transactions

Toutes les transactions sont enregistrées dans le journal des transactions, ceci permettant une éventuelle restauration.

Les étapes d'une transaction s'enchaînent dans l'ordre ci-dessous :

- Une modification des données est envoyée par l'application
- Les pages de données concernées sont chargées en mémoire (cache) et sont modifiées
- La modification est enregistrée dans le journal des transactions sur le disque
- Le point de contrôle écrit les transactions effectuées dans la base de données

3.2. Création de bases de données

3.2.1. Processus de création d'une base de données

Tout utilisateur voulant créer une base de donnée doit nécessairement avoir les autorisations d'accès à la base **master**. En effet toutes les bases de données créées sous SQL Server sont répertoriées dans 2 tables de la base de données **master** : **sysdatabases** et **sysaltfiles**.

3.2.2. Définition des options lors de la création d'une base de données

Voici la liste des options paramétrables sous SQL Server :

Options	Descriptions
Fichier principal	Le fichier principal contient le groupe de fichiers de la base de données. Ce groupe de fichiers stocke toutes les tables systèmes de la base de données ainsi que les objets et les données non attribuées aux groupes de fichiers définis par l'utilisateur.
Fichiers secondaires	Utilisés pour les bases de données très volumineuses qui requièrent plus de fichiers de données
Journal de transactions	Occupent 10 à 15% de la base de données. Voir ci-dessus pour plus de détails.
Nom de fichier et emplacement	Tous les fichiers de base de données possèdent un nom logique et un emplacement physique. Les fichiers doivent généralement être répartis sur plusieurs disques durs pour optimiser les performances et assurer la redondance.
Taille	512 K par défaut.
Croissance des fichiers	Donner la possibilité à un fichier d'occuper plus de place si besoin. (par défaut : oui)
Taille Maximale	Taille maximale que peut atteindre un fichier
Classement	Ce paramètre spécifie le classement par défaut de la base de données.

3.2.3. Modification des options de base de données après la création de la base de données

Pour modifier les options d'une base de données, nous pouvons passer par

- SQL Server Enterprise Manager
- l'instruction ALTER DATABASE
- la procédure stockée sp_dboption.

3.2.4. Affichage des propriétés de base de données

Pour afficher les propriétés d'une base de données, nous pouvons passer par

- SQL Server Enterprise Manager
- les procédures stockées système (sp_helpdb, sp_spaceused)
- les fonctions systèmes
- les instructions DBCC dans l'analyseur de requête SQL Server (DBCC SQLPERF (LOGSPACE)).

3.3. Gestion des bases de données

3.3.1. Gestion de la croissance des fichiers journaux et de données

La gestion de l'espace utilisé par les fichiers de la base de données est l'une des tâches les plus importantes. L'administrateur du serveur SQL dispose de deux solutions pour la croissance des fichiers :

Croissance Automatique	Croissance Manuelle
Les fichiers peuvent dynamiquement occuper plus ou moins de place sur le disque en se basant sur une marge définie par l'administrateur. Il est cependant conseillé de prévoir une taille maximale pour ces fichiers.	L'administrateur doit manuellement intervenir pour modifier l'espace disque pouvant être utilisé par un fichier. Ceci entraîne donc nécessairement plus de tâches administratives.

3.3.2. Compactage automatique d'une base de données ou d'un fichier de base de données

SQL Server permet de compacter les fichiers de base de données afin d'optimiser les opérations d'accès aux informations.

Cette option, désactivée par défaut sur toutes les éditions de SQL Server sauf SQL Server Desktop Edition, ne peut s'exécuter que si l'espace libre est supérieur à 25 %.

Pour activer le compactage automatique, l'administrateur a le choix entre :

- SQL Server Enterprise Manager
- L'instruction ALTER DATABASE AUTO_SHRINK
- La procédure stockée système sp_dboption

3.3.3. Compactage manuel d'une base de données ou d'un fichier de base de données

Un compactage manuel permet de réduire les fichiers de données et de journal des transactions en groupe ou individuellement. Pour ce faire, on peut utiliser :

- SQL Server Enterprise Manager
- DBCC SHRINKDATABASE
- DBCC SHRINKFILE

3.3.4. Suppression d'une base de données

L'instruction DROP DATABASE permet de supprimer une base de données.

Cependant cette suppression ne peut pas être effectuée dans les cas suivants :

- Quand la base de données est en cours de restauration
- Quand la base de données est ouverte par un autre utilisateur en lecture / écriture
- En cas de publication de l'une des tables de la base dans le cadre d'une réplication SQL Server
- Vous ne pouvez pas supprimer une base de données système

3.4.Emplacement des fichiers et journaux de base de données

La solution la plus optimale consiste à utiliser un seul fichier de données par disque physique, ce qui allègera les accès Entrées/Sorties pour chaque disque dur.

Mieux encore, utiliser le RAID donnera les mêmes avantages, avec en plus une tolérance de panne (sauf pour le RAID 0).

3.5.Optimisation d'une base de données à l'aide d'une solution RAID matérielle

3.5.1. Création de groupes de fichiers définis par l'utilisateur

Pour créer un groupe de fichier personnalisé, l'utilisateur doit passer par :

- SQL Server Enterprise Manager
- CREATE DATABASE
- ALTER DATABASE

3.5.2. Utilisation de groupes de fichiers pour améliorer les performances

Lorsque vous créez une table, vous pouvez l'attribuer à un groupe de fichiers défini par l'utilisateur. Les groupes de fichiers obéissent à une stratégie de remplissage réparti sur tous les fichiers du groupe de fichiers. À mesure que des données sont écrites dans le groupe de fichiers, chaque fichier est rempli en parallèle.

Chaque fichier est placé physiquement sur un disque ou un ensemble de disques. SQL Server conserve un plan de fichiers qui associe chaque objet de la base de données avec son emplacement sur le disque.

3.6.Optimisation d'une base de données en utilisant des groupes de fichiers avec une solution RAID matérielle

La solution RAID permet de bénéficier des mêmes avantages que la répartition des fichiers de données sur plusieurs disques, tout en étant plus simple à administrer.

3.7. Planification de la capacité

3.7.1. Évaluation de la taille d'une base de données

Les paramètres suivants sont à prendre en compte pour évaluer la taille qu'occupera une base de données sur le disque:

- La taille de la base de données **model** et de ses tables système, en considérant les prévisions de croissance de la base;
- La nature et la quantité de données à insérer dans les tables, en considérant les prévisions de croissance de la base;
- La taille et le nombre des index, en particulier le nombre de lignes, la taille de la valeur de clé et le taux de remplissage ;
- La taille du journal des transactions, qui dépend de la fréquence et de la quantité des modifications, de la taille de chaque transaction et de la fréquence de sauvegarde et de vidage du journal ;
- La taille des tables système, avec le nombre d'utilisateurs, d'objets, etc... Elle constitue une part souvent très minoritaire de la taille d'une base de données.

3.7.2. Évaluation de la quantité de données contenues dans les tables

Après avoir estimé l'espace accordé à la base de données « model », vous devez examiner la quantité de données qu'auront à contenir vos tables, en prenant en considération d'éventuelles prévisions de croissances. Cette estimation est effectuée en déterminant pour chaque table le nombre de lignes, leur taille, le nombre de lignes contenues dans une page.

Pour évaluer le nombre de pages requises pour une table, utilisez la méthode suivante :

- Additionnez le nombre d'octets requis par chaque colonne. Pour les colonnes à taille variable (chaînes de caractères, par exemple), estimez la moyenne que vos données consommeront.
- Déterminez le nombre de lignes que contiendra une page en divisant 8060 par le nombre d'octets par ligne et en arrondissant le résultat.
- Divisez le nombre de lignes prévues pour la table par le nombre de lignes que peut contenir une page de données.

4. Gestion de la sécurité

4.1. Implémentation d'un mode d'authentification

4.1.1. Traitement de l'authentification

Dans le cas d'une authentification Windows, le client transmet les références de sécurité Windows 2000 de l'utilisateur. Comme il s'agit d'une authentification Windows, SQL Server sait que le mot de passe est correct car il a déjà été validé par un contrôleur de domaine. Si le compte utilisateur ou de groupe est présent dans la table système sysxlogins, la connexion est acceptée par SQL Server.

Dans le cas d'une authentification SQL Server, le login est recherché dans la table sysxlogins. Si ce dernier est présent, le mot de passe fourni est comparé au mot de passe stocké correspondant à l'utilisateur. S'ils sont identiques, la connexion est acceptée.

4.1.2. Choix d'un mode d'authentification

Les modes d'authentification mixte et Windows présentent tous deux des avantages. L'authentification Windows permet de profiter d'une sécurité avancée avec, entre autres, l'expiration de mots de passe, la longueur minimale des mots de passe, l'audit, et le verrouillage de comptes après la saisie d'un mot de passe invalide.

L'authentification mixte offre l'avantage de permettre aux clients non-windows 2000 de se connecter à SQL Server.

4.1.3. Emprunt d'identité et délégation

L'emprunt d'identité est la faculté de SQL Server à accéder à des ressources locales (typiquement le système de fichiers) en utilisant l'identité de l'utilisateur pour lequel il effectue ces tâches.

La délégation consiste, lorsque qu'un serveur se connecte à un autre serveur pour le compte d'un utilisateur, à utiliser les informations d'authentification de l'utilisateur. Si un utilisateur ESI\rivall_j se connecte à un premier serveur SERVERA, et que ce serveur doit se connecter à SERVERB, SERVERB reconnaîtra la connexion comme venant de ESI\rivall_j. La délégation n'est disponible qu'en environnement Windows 2000 Active Directory.

La configuration de la délégation nécessite certaines actions au niveau d'Active Directory et de SQL Server. L'option « Le compte est approuvé pour la délégation » doit être cochée pour le compte utilisateur devant utiliser les fonctionnalités de délégation et l'option « L'ordinateur est approuvé pour la délégation » doit être cochée sur le compte d'ordinateur du serveur SQL.

4.1.4. Cryptage

SQL Server crypte automatiquement les mots de passe des tables système. Aucun utilisateur, y compris un utilisateur administrateur, ne peut visualiser un mot de passe de connexion dans la base de données système. Il est également possible de crypter les définitions des vues, triggers et procédures stockées dans la table syscomments.

Il est également possible de crypter la connexion réseau entre le client et le serveur. Cette démarche est fortement conseillée lorsque des données confidentielles sont susceptibles d'être échangées entre client et serveur (le mot de passe circule en clair lors de l'authentification SQL Server).

4.1.5. Étapes de l'implémentation du mode d'authentification

La définition d'un mode d'authentification sur SQL Server s'effectue dans **SQL Server Enterprise Manager** et est effective après redémarrage du service MSSQLServer. Une fois le mode d'authentification défini, il est possible de définir les droits de connexion à des utilisateurs ou groupes Windows 2000 et de créer des comptes de connexion SQL Server (dans le cas d'une authentification en mode mixte).

4.1.6. Création de comptes de connexion

La création d'un compte de connexion Windows 2000 se réalise grâce à **SQL Server Enterprise Manager** ou à la procédure stockée **sp_grantlogin**. Le nom de compte doit être un nom de compte utilisateur ou de groupe Windows 2000 pleinement qualifié (de la forme DOMAINE\utilisateur). Ce nom est limité à 128 caractères unicode.

✎ La suppression d'un utilisateur ou d'un groupe Windows 2000 ne supprime pas l'entrée correspondante dans SQL Server (pour supprimer un compte de connexion, utilisez la procédure stockée **sp_revokelogin**).

Lorsqu'une permission de connexion est accordée à un groupe, tous ses membres reçoivent cette permission. Un utilisateur peut à tout moment changer le mot de passe de son compte de connexion avec la procédure stockée **sp_password**.

La création d'un compte de connexion SQL Server se réalise grâce à **SQL Server Enterprise Manager** ou à la procédure stockée **sp_addlogin**. La longueur d'un nom d'un compte de connexion SQL Server ne peut excéder 128 caractères, il ne doit pas contenir de caractère '\' (backslash) ou '\0' (NULL) ni être une chaîne vide ou égale à un compte réservé (ex : « sa »).

SQL Server possède deux comptes de connexion par défaut, BUILTIN\Administrateurs et sa. Le premier permet la connexion des administrateurs Windows 2000, il possède tous les droits sur SQL Server et sur toutes les bases de données. Le second est un compte par défaut en mode mixte et possède tous les droits sur SQL Server et toutes les bases de données. Il est présent à des fins de compatibilité et ne doit pas être utilisé systématiquement.

4.2. Attribution de comptes de connexion à des utilisateurs et des rôles

Pour ajouter un utilisateur à une base de données, utilisez **SQL Server Enterprise Manager** ou la procédure stockée **sp_grantdbaccess**.

Deux utilisateurs ont des statuts spéciaux dans SQL Server : **dbo** et **guest**.

- **dbo** est l'utilisateur propriétaire de tous les objets créés par un administrateur et ne peut être supprimé.
- **guest** est un compte utilisateur ne requérant pas de mot de passe. Cet utilisateur doit être ajouté manuellement à une base de données pour permettre son utilisation.

4.2.1. Rôles fixes de serveur

Un compte de connexion membre d'un rôle fixe de serveur a des droits particuliers sur l'environnement SQL Server.

✎ Tous les membres d'un rôle fixe de serveur peuvent ajouter d'autres comptes de connexion à ce rôle.


Pour ajouter ou supprimer un compte de connexion dans un rôle de serveur, utilisez la boîte de dialogue Propriétés de connexion SQL Server dans Enterprise Manager ou les procédures stockées `sp_addsrvrolemember` et `sp_dropsrvrolemember`.

Les rôles fixes de serveur sont les suivants :

Rôle	Autorisation
sysadmin	Toute activité autorisée
dbcreator	Créer et modifier une base de données
diskadmin	Gérer les fichiers de base de données
processadmin	Gérer les processus SQL Server
serveradmin	Changer la configuration de SQL Server
setupadmin	Installer la réplication
securityadmin	Gérer les connexions au serveur
bulkadmin	Exécution de commandes BULK INSERT

4.2.2. Rôles fixes de base de données

Les comptes de connexion membres d'un rôle fixe de base de données ont des droits particuliers sur la base.

 Seuls les membres du rôle `db_owner` peuvent ajouter ou supprimer un compte de connexion d'un rôle fixe de base de données.

Pour ajouter ou supprimer un compte de connexion dans un rôle de base de données, utilisez SQL Server Enterprise Manager ou les procédures stockées `sp_addrolemember` et `sp_droprolemember`.

Les rôles fixes de base de données sont :

Rôle	Autorisations
public	Autorisation par défaut
db_owner	Droits illimités sur la base de données
db_accessadmin	Ajout, suppression d'utilisateur, de groupe et de rôle de base de données
db_ddladmin	Ajout, suppression et modification des objets de base de données
db_securityadmin	Donner des autorisations à des objets
db_backupoperator	Sauvegarde de la base de données
db_datareader	Lire les données de toutes les tables
db_datawriter	Modifier les données de toutes les tables
db_denydatareader	Refuser la lecture de toutes les tables
db_denydatawriter	Refuser la modification des données de toutes les tables

4.2.3. Rôles de base de données définis par l'utilisateur

Les rôles de base de données définis par l'utilisateur permettent de regrouper les utilisateurs de manière logique lorsque aucun groupe Windows 2000 n'existe à cet usage. Un rôle de base de données défini par l'utilisateur peut être ajouté en tant que membre de rôles fixes de serveur ou de base de données. Les autorisations correspondantes s'appliquent donc à tous les membres du groupe comme si elles leur étaient attribuées directement.

4.3. Attribution d'autorisations à des utilisateurs et des rôles

4.3.1. Types d'autorisations

Les instructions Transact-SQL impliquant la création ou la suppression d'objets ont chacune une autorisation correspondante nécessaire à leur utilisation (ex : CREATE DATABASE).

☞ Seuls les membres de sysadmin, db_owner ou db_securityadmin peuvent accorder des autorisations sur instruction.

Certains types d'objet peuvent avoir des autorisations propres, en particulier les tables (SELECT, INSERT...), colonnes (SELECT, UPDATE, REFERENCES...) et procédures stockées (EXECUTE).

☞ Un propriétaire d'objet a implicitement tous les droits permettant toute activité sur un objet lui appartenant.

4.3.2. Octroi, refus et révocation d'autorisations

Une autorisation peut être accordée ou révoquée à plusieurs niveaux hiérarchiques (directement sur un utilisateur, au niveau d'un rôle dont l'utilisateur est membre, un utilisateur pouvant être membre de plusieurs rôles). Pour ôter toute ambiguïté quant aux autorisations d'un utilisateur, trois instructions concernent l'octroi ou le refus d'autorisation à un utilisateur :

Instruction	Action de l'instruction	Description
GRANT	Positif	L'utilisateur se voit accorder l'autorisation
REVOKE	Neutre	L'utilisateur ne peut réaliser l'action. Cette interdiction peut être ôtée par l'appartenance à un rôle.
DENY	Négatif	L'utilisateur ne peut réaliser l'action. Cette interdiction ne peut pas être ôtée par l'appartenance à un rôle.

4.4. Gestion de la sécurité des applications

4.4.1. Gestion de la sécurité à l'aide de vues et de procédures stockées

Pour une gestion souple de la sécurité des données, l'utilisation des vues et procédures stockées peut être fort utile car elle permet de gérer la sécurité à leur niveau, plutôt qu'au niveau des données auxquelles elles font références.

☞ SQL Server inclut des outils assistant la création de vues et procédures stockées de manière graphique. Dans certains cas, néanmoins, l'écriture directe en Transact-SQL peut être nécessaire.

4.4.2. Gestion de la sécurité des applications clientes à l'aide de rôles d'application

Les rôles d'applications permettent de gérer l'environnement de sécurité accordé à une application particulière. Ils peuvent permettre de restreindre indirectement les autorisations d'un utilisateur. Leur utilisation est similaire aux rôles définis par l'utilisateur vis-à-vis de l'octroi ou de la révocation d'autorisation, mais certaines de leurs propriétés sont différentes : un rôle d'application ne contient pas de membres, il est affecté dynamiquement à n'importe quel utilisateur utilisant l'application en question, ce dernier perdant automatiquement ses autorisations propres.

Utilisez SQL Server Enterprise Manager ou la procédure stockée système `sp_addapprole` pour créer un rôle d'application. Seuls les membres des rôles `db_owner`, `db_securityadmin` et `sysadmin` peuvent exécuter la procédure `sp_addapprole`.

Une fois qu'un client s'est connecté à SQL Server au moyen d'un compte de connexion, il doit exécuter la procédure stockée système `sp_setapprole` pour activer les autorisations associées à un rôle d'application. La procédure stockée système `sp_setapprole` ne peut être exécutée qu'à l'aide d'instructions Transact-SQL directes. Elle ne peut pas l'être au sein d'une autre procédure stockée ou d'une transaction définie par l'utilisateur.

4.5. Gestion de la sécurité de SQL Server dans l'entreprise

4.5.1. Utilisation d'une stratégie de groupe pour sécuriser SQL Server

L'utilisation de l'authentification Windows 2000 par SQL Server permet l'utilisation de toutes les fonctionnalités de la gestion des utilisateurs et des ordinateurs présentes dans Windows 2000, en particulier les Stratégies de groupe (GPO) qui permettent de configurer de manière centralisée nombre de paramètres (stratégies de mots de passe, longueur, expiration, verrouillage de comptes).


4.5.2. Utilisation du cryptage pour la transmission des données en vue de leur sécurisation

SQL Server peut tirer parti du support IPSec inclus dans Windows 2000. IPSec est un ensemble de standards Internet qui ajoutent des fonctionnalités de sécurité cryptographiques au protocole IP :

- Confidentialité des données. Le trafic IPSec est chiffré, donc illisible à tout personne ignorant la clé de chiffrement.
- Authentification de l'expéditeur. IPSec permet de s'assurer de l'identité de l'expéditeur des paquets grâce à l'authentification Kerberos, à un certificat numérique, ou à un secret partagé.
- Intégrité des données. IPSec permet de s'assurer que les données n'ont pas été modifiées depuis leur envoi par l'expéditeur.

La stratégie de sécurité IPSec peut être configurée au niveau d'un domaine ou d'un ordinateur Windows 2000.

SQL Server peut également utiliser un chiffrement de type SSL (Secure Sockets Layer) pour chiffrer ses communications avec les clients. SSL nécessite un certificat émanant d'une autorité de certification publique. Ce certificat permet au client de s'assurer de l'identité du serveur avec lequel il communique, grâce à un certificat émis par l'autorité de certification ayant fourni le certificat du serveur. Il est aussi possible pour le client de disposer de son propre certificat pour s'authentifier auprès du serveur, via SSL.

 L'activation de SSL se fait grâce à l'utilitaire « SQL Server Network Utility ». Si SSL est activé, il l'est sur tous les protocoles serveur présents sur SQL Server.

5. Exécution de tâches administratives

5.1. Tâches liées à la configuration

5.1.1. Configuration de l'Agent SQL Server

Comme nous l'avons vu dans le chapitre concernant l'installation de SQL Server, il est possible de configurer le service de sorte à ce qu'il démarre automatiquement.

Il est également possible de configurer ce service pour qu'il redémarre en cas d'arrêt inattendu. Dans ce cas, le compte doit être membre du groupe Administrateurs local.

Il faut également attribuer le rôle sysadmin au compte d'ouverture de session de l'agent SQL Server. Dans le cas d'un compte système local, on ne pourra pas accéder aux ressources réseau le plus souvent.

Pour un compte d'utilisateur de domaine, vous aurez la possibilité d'accéder aux ressources réseau et de communiquer avec des systèmes de messagerie par exemple. Soit vous attribuez le rôle sysadmin de SQL Server au compte d'ouverture de session, soit vous ajoutez le compte d'ouverture de session au groupe local Windows auquel est attribué le rôle sysadmin.

5.1.2. Configuration de SQLAgentMail et de SQL Mail

SQL Server a la possibilité de communiquer avec des serveurs de messagerie pour envoyer ou recevoir du courrier. Il utilise alors la session SQLAgentMail ou SQL Mail.

Les messages envoyés concernent les alertes et les résultats de travaux planifiés.

Un compte d'utilisateur de domaine avec un profil de messagerie doit être paramétré. Il suffit ensuite de spécifier le profil à utiliser dans SQL Server Enterprise Manager.

SQL Mail est un composant constitué de bon nombre de procédures stockées étendues utilisées par le service SQL Server.

SQL Mail va alors répondre aux messages électroniques en renvoyant des résultats.

Les conditions de fonctionnement de SQL Mail et SQLAgentMail sont :

- un serveur de messagerie compatible MAPI
- un client de messagerie configuré sur la machine exécutant SQL Server
- un compte de connexion pour les services SQL Server et SQLServerAgent
- un profil de messagerie
- sélection du profil de messagerie dans SQL Server Enterprise Manager.

5.1.3. Configuration de serveurs liés

Un serveur lié est défini par un **fournisseur OLE DB** et une **source de données**. Le fournisseur OLE DB qui est rattaché à une source de données spécifique est en fait une bibliothèque de liaisons dynamiques (DLL).

La source de données peut également être constituée de fichiers textes ou de résultats de recherches de contenu de texte intégral.

La définition d'un serveur lié passe par son inscription auprès de SQL Server avec un nom logique. Pour son inscription, on utilise SQL Server Enterprise Manager ou la procédure stockée **sp_addlinkedserver**.

Lors de l'inscription du serveur lié, un automappage a lieu, c'est-à-dire que les informations d'identification de sécurité en cours du compte d'ouverture de session sont émuloées sur le serveur lié.

Si le serveur lié ne reconnaît pas le mode d'authentification Windows ou si la délégation de compte de sécurité n'est pas disponible sur l'ordinateur client ou serveur, alors l'automappage ne fonctionnera pas. Il faudra paramétrer un mappage de connexion locale d'un compte d'ouverture de session authentifié Windows vers un compte d'ouverture de session spécifique sur le serveur lié. Le compte d'ouverture de session distant prendra donc un compte authentifié SQL Server si le serveur lié est une instance de SQL Server.

5.1.4. Configuration des noms de source de données

Un nom de source de données doit être défini par 3 éléments :

- le pilote ODBC permettant la connexion à la source de données
- les informations du pilote ODBC pour la connexion à la source de données. Ces informations - regroupent entre autres : nom et chemin de la source de données, compte de connexion ...
- les options propres au pilote avec des enregistrements divers, statistiques ...

5.1.5. Configuration de la prise en charge du langage XML de SQL Server dans les services Internet

Il est possible d'accéder à SQL Server par le protocole http. Voici les possibilités :

- **accéder aux objets de la base de données** à partir d'une URL (tables par exemple)
- **exécuter des fichiers modèles**. Ces fichiers sont des fichiers XML contenant des instructions Transact-SQL. Il est possible de passer directement ces instructions dans l'URL mais ce n'est pas recommandé pour des raisons de sécurité
- **exécuter des requêtes XPath** (*XML Path Language*).

L'accès en XML à SQL Server se fait en plusieurs étapes :

- indication du nom du serveur Web dans l'URL
- le serveur Web vérifie dans la racine virtuelle indiquée dans l'URL que l'extension de nom de fichier DLL ISAPI a bien été inscrite
- le serveur Web transmet alors la demande à la bibliothèque DLL
- Sqlisapi.dll communique avec le fournisseur OLE DB et se connecte à l'instance de SQL Server identifiée.

Pour paramétrer cette fonction, il est nécessaire de créer **un répertoire virtuel** avec **IIS Virtual Directory Management pour SQL Server**. Sous cet utilitaire, il faut créer une nouvelle **racine virtuelle** et une association sera alors créée (grâce aux services Internet) entre le nouvel annuaire et une instance de SQL Server.

5.1.6. Configuration de SQL Server pour partager les ressources de mémoire avec les autres applications serveur

SQL Server gère dynamiquement sa charge de mémoire. Il prend et libère la mémoire selon ses besoins de mise en cache des données.

Cependant, même si ce n'est pas conseillé, vous avez la possibilité de paramétrer le tout manuellement. Vous pouvez alors définir un minimum et un maximum de mémoire à utiliser pour SQL Server.

Il est évident que plus la quantité de mémoire allouée à SQL Server est grande, moins le fichier d'échange sera sollicité. Ce qui est un gain de temps et de productivité pour SQL Server.

5.2. Tâches de routine liées à l'administration de SQL Server

Comme pour tout SGBD en production, il est nécessaire de planifier certaines tâches :

- sauvegarde de base de données
- importation et exportation de données

Mais il est également essentiel de mener une surveillance accrue du serveur. Pour cela, il est conseillé de surveiller les journaux de base de données et de transactions : espace libre, erreurs ...

De même, une surveillance des performances est recommandée. Nous citerons par exemple les verrous utilisateurs.

5.3. Automatisation des tâches de maintenance de routine

5.3.1. Automatisation de l'administration de SQL Server

Les services utilisés pour l'automatisation de SQL Server sont :

- **Agent SQL Server**
- **SQL Server**
- **Observateur d'événements Windows**

Les composants de l'Agent SQL Server permettant les automatisations sont :

- **les alertes**
- **les travaux**
- **les opérateurs**

On peut alors associer les alertes et les travaux pour par exemple l'envoi d'une alerte lorsque qu'un travail planifié échoue.

5.3.2. Création de travaux

Pour définir un travail, vous pouvez utiliser SQL Server Enterprise Manager ou exécuter la procédure stockée système **sp_add_job**. La définition du travail est stockée dans la table système **msdb..sysjobs**. Cette table est conservée en mémoire cache pour améliorer les performances.

Voici alors les étapes à ne pas manquer :

- vérification que le travail est activé
- vérification du propriétaire du travail. Il faut mentionner son compte d'ouverture de session utilisateur Windows ou SQL Server.
- vérification si le travail doit être effectué en local ou sur d'autres serveurs
- classification du travail dans une catégorie pour faciliter la gestion de l'ensemble des travaux.

5.3.3. Définition des étapes d'un travail

Vous pouvez utiliser SQL Server Enterprise Manager ou exécuter la procédure stockée système **sp_add_jobstep** pour définir chaque étape d'un travail. Les définitions des étapes d'un travail sont stockées dans la table système **msdb..sysjobsteps**.

Il est possible de définir des étapes de travail pour :

- **des instructions Transact-SQL** : il faut identifier la base de données, inclure variables et paramètres requis pour le travail, et éventuellement préciser un fichier de sortie
- **des commandes du système d'exploitation** : Il faut préciser un code de sortie du processus pour avertir de la réussite de la commande, et indiquer le chemin complet de la commande

- **des scripts ActiveX** : il faut préciser le langage de script utilisé et écrire ou ouvrir le script actif
- **des tâches de réplication SQL Server.**

5.3.4. Création d'un organigramme des actions par étape de travail

Dans la planification des travaux, il est possible de gérer les finalités des travaux. Nous pouvons donc préciser à SQL Server quelle action mener en cas d'échec ou de réussite d'un travail.

Par défaut, SQL Server exécute les étapes les unes après les autres et s'arrête en cas d'échec. Vous pouvez indiquer le nombre de fois que SQL Server doit tenter d'exécuter une étape de travail en cas d'échec de celle-ci. Vous pouvez également définir des intervalles de reprise (en minutes).

5.3.5. Planification de travaux

Vous pouvez utiliser SQL Server Enterprise Manager ou exécuter la procédure stockée système **sp_add_jobschedule** pour définir la planification de chaque travail. Les planifications des travaux sont stockées dans la table système **msdb..sysjobschedules**.

En environnement multi-serveur, il est possible de planifier les travaux sur plusieurs serveurs cibles.

Il est possible de planifier le démarrage automatique des travaux uniquement dans les conditions suivantes :

- l'Agent SQL Server est démarré
- une heure spécifique est réglée
- une régularité est réglée (hebdomadaire, mensuel ...)
- le processeur est inactif

Vous pouvez également paramétrer des planifications multiples. Plusieurs planifications peuvent être réglées pour un même travail avec par exemple une alternance jour-nuit.

5.3.6. Création d'opérateurs à notifier

Vous pouvez utiliser SQL Server Enterprise Manager ou exécuter la procédure stockée système **sp_add_operator** pour définir un nouvel opérateur. La définition de l'opérateur est stockée dans la table système **msdb..sysoperators**.

Une fois le travail exécuté ou en cas d'échec d'une des étapes du travail, vous pouvez écrire l'événement dans le journal applications de Windows, supprimer le travail ou notifier un opérateur par radiomessagerie, courrier électronique ou par une commande **net send**.

Pour éviter d'éventuels conflits, il est important de préciser les adresses de messagerie complètes pour les noms de messagerie.

Pour créer un opérateur :

- utiliser un alias de messagerie de groupe pour le contact de plusieurs personnes
- établir une planification de travail pour chaque opérateur
- utiliser une commande **net send** pour les envois de messages aux opérateurs et serveurs sous Windows 2003, 2000 et NT.

En cas de problème d'envoi de notification, il faut vérifier :

- que l'opérateur est disponible
- que le service Affichage des messages est bien activé sur l'ordinateur de l'opérateur

- l'heure de la dernière notification pour mieux trouver d'où provient l'erreur
- que vous pouvez bien notifier l'opérateur sans passer par SQL Server

5.3.7. Analyse et configuration de l'historique des travaux

L'agent SQL Server reporte l'état d'exécution des travaux dans la table système `msdb..sysjobhistory`. Vous pouvez afficher l'historique des travaux dans Enterprise Manager.

Afin de trouver les réponses à un éventuel échec d'un travail, l'historique enregistre les éléments suivants :

- date et heure du travail
- échec ou réussite du travail
- opérateur notifié
- méthode de notification (mail, net send ...)
- durée du travail
- erreurs et messages

5.4. Création d'alertes

5.4.1. Utilisation d'alertes en réponse à des problèmes potentiels

Les alertes répondent aux erreurs présentes dans le journal Applications de Windows. Le schéma des alertes est le suivant :

- l'utilisateur génère une erreur
- l'erreur est consignée dans le journal Applications de Windows
- l'Agent SQL Server est prévenu d'un événement par le journal Applications
- l'Agent SQL Server recherche l'alerte correspondant à l'erreur dans la table système `msdb..sysalerts` conservée en mémoire cache
- l'Agent SQL Server passe en revue la table système `msdb..sysnotifications` pour envoyer le message électronique et la table système `msdb..sysoperators` pour identifier l'opérateur à joindre.

5.4.2. Consignation d'événements dans le journal applications

Des événements sont consignés dans le journal Applications Windows dans les cas suivants :

- les erreurs SQL dont le niveau de gravité est compris entre 19 et 25
- l'instruction **RAISERROR WITH LOG** est exécutée
- la procédure stockée étendue `xp_logvent` est exécutée
- les procédures stockées système `sp_addmessage` et `sp_altermessage` sont exécutées.

5.4.3. Création d'alertes en réponse à des erreurs SQL Server

Lorsque vous associez une alerte à une erreur SQL Server, vous pouvez indiquer un seul numéro d'erreur, comme 9003, ou toutes les erreurs d'un niveau de gravité particulier, tel que 16.

Une nouvelle alerte peut être créée au moyen d'Enterprise Manager ou de la procédure stockée système `sp_add_alert`. Sa définition sera stockée dans la table système `msdb..sysalerts`.

Il est possible de personnaliser les messages d'erreurs, en plus du numéro.

Pour les alertes concernant les niveaux de gravité, les niveaux compris entre 20 et 25 concernent les erreurs fatales.

L'Agent SQL Server peut être configuré pour envoyer tous les messages d'événements (ou seulement ceux qui ne sont pas gérés) dont le niveau de gravité excède une valeur donnée à un même serveur. Il peut être judicieux de transférer ces événements vers le serveur dont la charge est la moins élevée de votre réseau.

5.4.4. Réponse à des alertes de conditions de performances

Vous pouvez également utiliser les alertes en réponse à des alertes de performance, comme par exemple la taille d'un journal qui dépasserait une valeur critique. Il est alors possible de paramétrer l'alerte de sorte à ce qu'elle effectue un travail.

Les alertes de performance peuvent concerner les sujets suivants :

- Méthodes d'accès
- Bases de données
- Gestionnaire de cache
- Gestionnaire de tampons
- Verrous
- Statistiques SQL

5.4.5. Attribution d'un opérateur de prévention de défaillance

Un opérateur de prévention de défaillance peut être alerté par l'échec d'une notification envoyée par radiomessagerie à des opérateurs définis.

Un opérateur de prévention de défaillance est notifié dans les cas suivants :

- Echec de notifications par messagerie électronique
- Si il est défini
- La session de messagerie de l'Agent SQL Server doit être en service

Cependant, il ne peut y avoir qu'un seul opérateur de messagerie.

Pour supprimer un opérateur de prévention de défaillance, il faut dans un premier temps supprimer l'attribution d'opérateur de prévention de défaillance, puis supprimer l'opérateur.

5.5. Résolution des problèmes liés à l'automatisation de SQL Server

Voici un tableau récapitulatif des automatisations et leurs problèmes potentiels :

Problèmes d'automatisation	Vérifications
Agent SQL Server	Est-ce que le rôle sysadmin de SQL Server est bien attribué au compte de connexion de l'agent SQL Server ? Le mot de passe du compte de connexion est-il correct ?
Planification, Alertes	Est-ce que le travail, la planification, l'alerte ou l'opérateur est activé ?
Le Compte Proxy	Est-ce que les utilisateurs ont les autorisations d'exécuter ces types de travaux ? Est-ce que le compte d'utilisateur du domaine utilisé en tant que compte proxy a les autorisations pour exécuter les travaux ?

Les journaux d'erreurs	Un journal des erreurs est créé à chaque arrêt et redémarrage de l'Agent SQL Server Ne pas hésiter à augmenter la taille du journal Applications de Windows pour les événements SQL Server Comparez date et heure entre journal des erreurs SQL Server, journal Applications Windows et journal Agent SQL Server
L'historique	L'historique des alertes et des opérateurs capture uniquement l'activité la plus récente Vous devez vérifier pour des raisons d'espace disque que la taille du fichier et la taille de croissance de la base de données sont cohérentes
Le client de messagerie	Se connecter au client de messagerie avec un compte d'utilisateur du domaine de l'Agent SQL Server et tenter d'envoyer une notification par courrier électronique à un opérateur.

Un problème plus important concerne parfois les retards d'alertes.

Les causes de ce problème peuvent être :

- trop de réponses aux alertes
- journal Applications Windows plein
- utilisation du processeur anormalement élevée

Les solutions proposées sont :

- désactivation temporaire de l'alerte
- plus de délai entre les réponses
- vidage du journal Application Windows
- régler le problème d'utilisation des ressources anormalement élevée

5.6. Automatisation de travaux sur plusieurs serveurs

Il est possible, pour des questions de gestion, de répartir des travaux sur plusieurs serveurs.

Vous pourrez alors gérer plusieurs serveurs à partir d'un seul, et répartir les types de travaux selon des emplacements de serveur.

Pour cela, il est nécessaire de définir un serveur principal qui va distribuer les différents travaux à des serveurs cibles.

Un serveur principal ne peut être défini que sur un Windows 2003, 2000 ou NT Server pour la simple raison qu'un grand nombre de connexions est nécessaire.

L'utilitaire SQL Server Enterprise Manager va permettre l'inscription de ce serveur en tant que serveur principal. Vous pouvez également utiliser la procédure stockée système suivante : **sp_msx_enlist**.

Lors de la définition d'un serveur principal, un serveur cible est inscrit. Une ligne est alors insérée dans la table système **systargetservers** du serveur principal. Un compte d'ouverture de session SQL Server est automatiquement créé pour chaque serveur cible avec le suffixe **_msx_probe**.

Un opérateur **MSXOperator** est créé sur le serveur principal et sur chaque serveur cible.

Enfin, le serveur principal peut être désigné en tant que serveur de transferts d'événements pour les serveurs cibles.

Les serveurs cibles ne peuvent s'inscrire qu'à un seul serveur principal. Ils doivent faire partie du même domaine Windows que le serveur principal.

De même que pour le serveur principal, il existe 2 outils pour inscrire un serveur cible à un serveur principal :


- **Assistant Création d'un serveur cible** dans SQL Server Enterprise Manager
- Exécution de la procédure stockée système **sp_msx_enlist**

La définition du serveur cible est stockée dans la table système **msdb..systargetservers**.

L'intérêt de cette démarche est de créer ultérieurement des travaux sur le serveur principal qui seront exécutés sur les serveurs cibles.

Voici la procédure :

- la liste des travaux définis sur le serveur principal se trouve dans une liste de téléchargement de la table système **msdb..sysdownloadlist**
- les serveurs cibles se connectent régulièrement au serveur principal pour voir si la liste a subi des mises à jour. Une fois le travail terminé, le serveur cible envoie un message d'état du travail au serveur principal pour qu'il l'enregistre dans la table système **msdb..sysjobservers**.

 La modification d'un travail doit être effectuée sur le serveur principal uniquement, et non pas sur les serveurs cibles.

6. Sauvegarde de bases de données

6.1. Protection contre les pertes de données

Un autre point important dans l'administration d'un Serveur SQL est l'implémentation d'une stratégie de sauvegarde. En effet ceci permettra en cas de perte majeure de données (incendie, vol, défaillance logicielle ou matérielle, etc...) de pouvoir restituer les données perdues.


6.2. Définition et changement de mode de récupération de base de données

SQL Server supporte trois modes de restauration de données :

Mode	Description	Avantage	Inconvénient
Mode de récupération complète	Il s'agit du mode par défaut. Il permet une récupération totale ou partielle de la base de données, ce qui comprend la base de données en elle-même, ainsi que les informations contenues dans les journaux.	Aucune perte de données. Possibilité de restaurer partiellement les données en se basant sur un point temporel spécifique.	Espace de stockage requis important.
Mode de récupération Bulk_Logged	Ce modèle diminue le nombre d'espaces de journalisation inactifs, et conserve les espaces actifs. Cela apporte un avantage lors d'une restauration complète, car celle-ci sera alors plus rapide. Ceci est dû au fait que les opérations de chargements en blocs sont enregistrées sans contrôle précis.	Presque tous les avantages d'une récupération complète, en occupant moins de place.	Les opérations de chargements en blocs sont enregistrées sans contrôle précis.
Mode de récupération simple	Dans ce mode, la base de données est restaurée intégralement telle qu'elle était à sa dernière sauvegarde uniquement.	Aucune perte d'informations. Moins d'espace disque nécessaire.	Impossibilité de restaurer en se basant sur un point temporel.

Voici la syntaxe qui permet de changer le mode de restauration d'une base de données :

```
ALTER DATABASE base_données  
SET RECOVERY {FULL | SIMPLE | BULK_LOGGED}
```

 Vous devez obligatoirement effectuer une sauvegarde de la base de données quand vous changez le mode de restauration

6.3. Sauvegarde de SQL Server

Avant de tenter toute restauration d'une base de données, il faut bien entendu la sauvegarder au préalable. Durant ce processus de sauvegarde, l'administrateur système doit prendre en considération les points suivants :

- Il est possible d'effectuer une sauvegarde quand SQL Server est en cours d'utilisation
- La sauvegarde contient : le schéma et la structure des fichiers, les données, et les modifications de la base de données survenues depuis le début du processus de sauvegarde.

Le processus de sauvegarde peut être lancé par une instruction Transact-SQL ou via SQL Server Enterprise Manager. Vous devez être membre d'un des rôles suivant pour lancer le processus de sauvegarde :


- rôle fixe de serveur **sysadmin**
- rôle fixe de base de données **db_owner** ;
- rôle fixe de base de données **db_backupoperator**.

6.4. Moment approprié pour sauvegarder des bases de données

6.4.1. Sauvegarde de bases de données système

Vous devez sauvegarder la base de données **master** à *chaque* création de base de données définie par l'utilisateur. En effet la base de données **master** contient des informations sur toutes les bases de données hébergées par un serveur SQL Server. Vous pourrez ainsi récupérer et restaurer plus facilement les bases de données utilisateur si la base de données **master** est endommagée.

Une fois la base de données **master** reconstruite et restaurée, vous pouvez restaurer d'autres sauvegardes de bases de données système et faire référence à des bases de données utilisateur existantes.

 Si vous ne possédez pas une sauvegarde de la base de données master contenant des références vers vos bases de données utilisateur, utilisez l'utilitaire 80\Tools\Binn\Rebuildm.exe. Cet utilitaire en ligne de commande recrée toutes les bases de données système d'un bloc.

Certaines procédures stockées ou instructions modifient la base de données **master**. Par conséquent, l'administrateur doit sauvegarder cette base de données à chaque fois que ces procédures ou instructions sont appelées. Voilà ci-dessous la liste de ces procédures et instructions concernées :

- Instructions CREATE DATABASE, ALTER DATABASE, DROP DATABASE
- Procédures sp_logdevice, sp_addserver, sp_dropserver, sp_addlinkedserver, sp_addmessage

Les bases de données **msdb** et **model** doivent également être sauvegardées si elle sont modifiées.

6.4.2. Sauvegarde de bases de données utilisateur

Voilà ci-dessous un tableau montrant la liste des actions qui devraient être suivies d'une sauvegarde :

Action	Raison
Après création de bases de données	Car on ne peut pas restaurer un journal de transactions, celui-ci s'appliquant à une base de données.

Après création d'index	Ceci permet d'écourter le processus de restauration en cas de perte de la base de données.
Après vidage du journal des transactions (BACKUP LOG WITH TRUNCATE_ONLY ou BACKUP LOG WITH NO LOG)	Dans ce cas, le journal des transactions ne contient plus d'enregistrements de l'activité de la base de données et ne peut pas être utilisé pour récupérer des modifications apportées à la base de données.
Après exécution d'opérations non journalisées	Avec certains modes de récupération, vous ne pouvez pas récupérer les modifications effectuées par les opérations non journalisées suivantes : <ul style="list-style-type: none"> • Instruction BACKUP LOG WITH TRUNCATE_ONLY ou BACKUP LOG WITH NO_LOG. • Instruction WRITETEXT ou UPDATETEXT. • Instruction SELECT...INTO lors de la création d'une table permanente ou de l'utilisation du programme de copie en bloc

6.4.3. Activités à éviter pendant les sauvegardes

Bien que la sauvegarde de la base de données soit possible pendant que celle-ci subit d'éventuelles modifications, il faut tout de même éviter d'entamer ce processus dans les cas suivant :

- création ou modification de bases de données à l'aide de l'instruction **CREATE DATABASE** ou **ALTER DATABASE** ;
- exécution d'opérations de croissance automatique ;
- création d'index ;
- exécution d'opérations non journalisées, y compris le chargement en bloc de données et les instructions **SELECT...INTO**, **WRITETEXT** et **UPDATETEXT** ;
- compactage d'une base de données.

6.5.Exécution de sauvegardes

6.5.1. Création d'unités de sauvegarde

Une unité de sauvegarde permanente est un fichier qui hébergera les futures sauvegardes.

La création d'une unité de sauvegarde peut se faire en passant par SQL Server Enterprise Manager, ou en exécutant la procédure stockée système **sp_addumpdevice**.

Syntaxe

```
sp_addumpdevice [ @devtype = ] 'type_unité',
[ @logicalname = ] 'nom_logique',
[ @physicalname = ] 'nom_physique'
[, { [ @cntrltype = ] type_contrôleur | [ @devstatus = ] 'état_unité' } ]
```

Où *type_unité* représente {DISK | TAPE | PIPE}

Exemple 1

Dans cet exemple, un fichier de sauvegarde permanent est créé sur un disque dur.

```
USE master
```

```
EXEC sp_addumpdevice 'disk', 'mybackupfile',
```

```
'C:\Backup\MyBackupFile.bak'
```

Exemple 2

Dans cet exemple, une unité de sauvegarde est créée sur une bande avec le nom logique **Mytape1** et le nom physique **\\.\tape0**

```
USE master
```

```
EXEC sp_addumpdevice 'tape', 'mytape1', '\\.\tape0'
```

SQL Server ne peut utiliser un lecteur de bandes que si ce dernier est connecté localement au serveur.

Il est possible également de se passer d'une unité de sauvegarde permanente en utilisant l'instruction **BACKUP DATABASE**, dans ce cas les données sont stockées directement dans un fichier.

Exemple

```
USE master
```

```
BACKUP DATABASE Northwind
```

```
TO DISK = 'C:\Temp\MyCustomers.bak'
```

6.5.2. Utilisation de plusieurs fichiers de sauvegarde pour stocker les sauvegardes

SQL Server offre la possibilité de répartir les sauvegardes sur plusieurs fichiers de sauvegardes. Ceci permet d'optimiser les processus de sauvegardes et de restaurations (si ces fichiers sont sur des bandes ou des contrôleurs disques différents, un peu comme le RAID).

Syntaxe

```
BACKUP DATABASE {base_données | @variable_nom_base_données}
```

```
TO <unité_sauvegarde> [, ...n]
```

```
[WITH [ [MEDIANAME = {support | @variable_nom_support}]]
```

6.5.3. Spécification des options de bande

En cas de sauvegarde sur bande, plusieurs options de sauvegardes sont disponibles :

Option de bande	Description
UNLOAD (par défaut)	Une fois l'enregistrement terminé, SQL Server rembobine et extrait automatiquement la bande.
NOUNLOAD	SQL Server ne rembobine pas, et n'extrait pas automatiquement la bande après la sauvegarde.
BLOCKSIZE	Permet de définir en octets la taille des blocs physiques.
FORMAT	Utilisez cette option pour écrire un en-tête sur tous les volumes (fichiers) utilisés pour une sauvegarde. SQL Server remplace tous les en-têtes et toutes les sauvegardes existantes dans les

	fichiers.
SKIP	SQL Server ignore les étiquettes de bandes ANSI figurant sur le lecteur de bande. L'étiquette ANSI d'une bande peut signaler la date d'expiration de la bande et autoriser son accès en écriture.
NOSKIP (par défaut)	SQL Server vérifie l'étiquette de date d'expiration avant d'autoriser le remplacement des données.
RESTART	Utilisez cette option pour redémarrer l'opération de sauvegarde à partir du point d'interruption pour les sauvegardes sur bandes réparties sur plusieurs volumes. Vous devez relancer le processus de sauvegarde manuellement en exécutant l'instruction BACKUP d'origine avec l'option RESTART.

6.6.Types de méthodes de sauvegarde

6.6.1. Sauvegarde de base de données complète

Les sauvegardes de base de données complètes entraînent les actions suivantes :

- Création de points de références.
- Sauvegarde les fichiers, objets, et données d'origine.
- Sauvegarde des parties du journal des transactions.

6.6.2. Sauvegarde différentielle

Les sauvegardes différentielles sous SQL Server fonctionnent de la même manière que sous **ntbackup**. C'est-à-dire que tous les fichiers modifiés depuis la dernière sauvegarde complète sont à nouveau sauvegardés à chaque processus de sauvegarde différentielle.

Exemple :

```
BACKUP DATABASE Northwind TO  
DISK = 'D:\MyDiffBackup.bak'  
WITH DIFFERENTIAL
```

Ce type de sauvegarde permet de bénéficier d'une durée de sauvegarde réduite par rapport à une sauvegarde complète, ainsi qu'une durée de restauration également réduite car il n'est pas nécessaire d'appliquer une série de journaux de transactions.

6.6.3. Sauvegarde du journal des transactions

La sauvegarde d'un journal de transaction permet d'enregistrer toutes les modifications apportées à une base de données. Cette sauvegarde s'effectue via l'instruction **BACKUP LOG**.

Cet exemple crée une unité de sauvegarde pour le journal et sauvegarde le journal des transactions de la base de données **Northwind**.

```
USE master  
EXEC sp_addumpdevice 'disk', 'NwindBacLog',  
'D:\NwindBacLog.bak'  
BACKUP LOG Northwind TO NwindBacLog
```


6.6.4. Sauvegarde d'un fichier ou d'un groupe de fichiers de base de données

SQL Serve offre la possibilité de sauvegarder partiellement les fichiers de données d'une base de donnée. Ceci peut être pratique dans le cas de la gestion d'une VLDB (*Very Large DataBase*).

Syntaxe partielle

```
BACKUP DATABASE {base_données | @variable_nom_base_données}  
[<fichier_ou_groupe_fichiers> [, ...n]] TO <unité_sauvegarde> [, ...n]]
```


Où <fichier_ou_groupe_fichiers> représente :

```
{FILE = {nom_fichier_logique | @variable_nom_fichier_logique}  
|  
FILEGROUP = {nom_groupe_fichiers_logique |  
@variable_nom_groupe_fichiers_logique}  
}
```

Exemple

Dans cet exemple, le fichier Orders2 de la base de données PhoneOrders est sauvegardé dans le fichier OrderBackup2. Un rapport concernant cette sauvegarde sera généré dans le fichier OrderBackupLog.

```
BACKUP DATABASE PhoneOrders  
FILE = Orders2 TO OrderBackup2  
BACKUP LOG PhoneOrders to OrderBackupLog
```

 Pour assurer la cohérence des fichiers restaurés par rapport à la base de données, vous devez sauvegarder le journal des transactions.

7. Restauration de bases de données

7.1. Processus de récupération de SQL Server

Avant de lancer une mise à jour de la base de données en se basant sur une sauvegarde, SQL Server effectue un tri automatique des informations à sauvegarder.

Si le journal des transactions contient des **transactions validées** mais encore non inscrites dans la base de données, ces dernières le sont au moment de la restauration.

Alors que si le journal des transactions contient des **transactions non validées**, ces dernières ne sont pas traitées par SQL Server au moment de la restauration, et sont supprimées du journal de transactions.

☞ En cas de panne ou arrêt du Server SQL, le processus de restauration est automatiquement lancé au redémarrage du serveur.

7.2. Restauration de sauvegardes

Avant de lancer une procédure de restauration, il revient à l'administrateur de vérifier l'intégrité ainsi que le contenu des données à restaurer. Pour se faire, plusieurs instructions TRANSAC-SQL sont disponibles :

Instruction	Description
RESTORE HEADERONLY	Cette instruction renvoie les informations d'en-tête d'un fichier de sauvegarde ou d'un jeu de sauvegardes particulier (nom du fichier de sauvegarde, description du fichier de sauvegarde, date, heure, méthode de sauvegarde, etc...).
RESTORE FILELISTONLY	Cette instruction renvoie les informations concernant les fichiers de base de données ou de journal de transactions contenues dans un fichier de sauvegarde.
RESTORE LABELONLY	Cette instruction renvoie des informations sur le support de sauvegarde contenant le fichier de sauvegarde.
RESTORE VERIFYONLY	Vérifie l'intégrité des données sauvegardées.

7.3. Restauration de bases de données à partir de différents types de sauvegardes

7.3.1. Restauration à partir d'une sauvegarde complète et différentielle de base de données

En cas de restauration complète de la base de données, toutes les données sont restaurées. Il n'est pas nécessaire de reconstruire le schéma de la base de données.

En cas de restauration différentielle, il est possible de restituer uniquement les modifications effectuées sur la base de données depuis la dernière sauvegarde complète.

7.3.2. Restauration à partir d'une sauvegarde du journal des transactions

Lors d'une restauration d'une base de données depuis une sauvegarde du journal de transactions, SQL Server restaure les modifications apportées à la base de données.

Cette procédure permet d'appliquer à la base de données les modifications apportées depuis la dernière sauvegarde complète ou différentielle. Elle permet également de récupérer une base de données jusqu'à un certain point dans le temps.

SQL Server offre également la possibilité de restaurer la base de données telle qu'elle était à une heure précise via l'option STOPAT suivie de la date et l'heure. Par exemple :

```
USE master
RESTORE LOG MaBase
FROM NwindBacLog
WITH FILE = 2, RECOVERY,
STOPAT = '4 Mars 2002 06:00'
```

7.4. Restauration de bases de données système endommagées

Il peut arriver que les bases système soient endommagées. Dans ce cas, l'administrateur a la possibilité de restaurer une sauvegarde de cette base de données toujours avec l'instruction **RESTORE DATABASE**. Cependant cette méthode sous-entend que le service SQL Server est démarré. Si ce n'est pas le cas, l'administrateur peut également restaurer la base de données **master** en utilisant l'utilitaire en ligne **Rebuildm.exe** situé dans le dossier 80/Tools/Binn.

Si la base de données **master** a été restaurée et qu'aucune sauvegarde valide n'a été appliquée, il convient de rattacher toutes les bases de données utilisateur restaurées par la suite, en utilisant les procédures stockées systèmes **sp_attach_db** ou **sp_attach_single_file_db**.

8. Surveillance des performances de SQL Server

8.1. Raisons justifiant la surveillance de SQL Server

SQL Server offre un service de stockage de données aux applications. Le principal besoin justifiant la surveillance des performances de SQL Server est l'optimisation de ces applications. Le temps de réponse doit particulièrement être étudié car c'est ce qui pénalisera principalement les applications utilisant SQL Server.

Une analyse de la structure des données et du matériel hébergeant SQL Server est nécessaire pour en améliorer les performances. Les temps de réponses élevés sont souvent causés par un goulet d'étranglement au niveau matériel (CPU, RAM, disque dur...) ou par un système sous-jacent mal configuré (taille de pagefile insuffisante, répartition des fichiers sur le disque inadéquate...).

8.2. Surveillance et ajustement des performances

8.2.1. Stratégies d'ajustement des performances

L'ajustement des performances doit suivre deux axes principaux :

- la réduction du temps de réponse
- l'augmentation du débit

Le temps de réponse correspond au temps écoulé entre l'envoi d'une requête et la réception de la première ligne de la réponse par le client.

Le débit représente le nombre de requêtes qu'un serveur peut prendre en charge pendant un intervalle donné.

Ces deux paramètres se détériorent proportionnellement à mesure que le nombre d'utilisateurs d'un serveur augmente ou que les besoins se complexifient.

La surveillance du temps de réponse et du débit d'une instance SQL Server est essentielle pour fournir une expérience utilisateur optimale. Malgré tout, l'ajustement des performances peut aussi passer par l'analyse des applications utilisant SQL Server, car ils peuvent aussi, localement, dégrader le débit et le temps de réponse.

L'optimisation du débit passe principalement par l'architecture matérielle du serveur. Un goulet d'étranglement matériel (CPU, RAM...) ou logiciel (pagefile ou disque insuffisant, fragmentation...) est souvent la cause d'un débit faible. L'optimisation du temps de réponse, par contre, nécessite une connaissance détaillée de l'utilisation que les utilisateurs ont des données et des requêtes qu'ils effectuent. Elle passe par une optimisation des requêtes, des index, et des procédures stockées.

8.2.2. Définition d'une référence en matière de performances et méthodologie d'ajustement

L'ajustement des performances doit intervenir tout au long du cycle de développement d'une application. Prendre soin et documenter cette dernière peut permettre de gagner un temps précieux.

Cinq facteurs peuvent influencer sur les performances d'une instance SQL Server :

- La charge de travail. C'est le taux d'activité du serveur.
- Le débit. Le nombre de requêtes reçues par le serveur durant un intervalle donné.

- Les ressources système. Liées au matériel hébergeant le serveur.
- L'optimisation. C'est la conception de la base de données et des applications qui en tirent parti.
- Les conflits. Surviennent lors des accès concurrentiels à un même objet de base de données.

La référence des performances doit tenir compte de certains paramètres, comme la charge du serveur aux heures creuses et de pointes ou les temps de sauvegarde et de restauration. Une fois cette référence établie, vous pouvez vous en servir pour identifier les points à traiter plus particulièrement.

8.3. Outils de surveillance de SQL Server

8.3.1. Observateur d'événements Windows 2000

L'observateur d'événements Windows 2000 permet d'identifier les goulets d'étranglement de plusieurs natures. Il contient trois journaux différents : Applications, Système et Sécurité. Tout événement ayant une incidence quelconque sur le système y est consigné. Les événements spécifiques à SQL Server se trouvent dans le journal Applications.

8.3.2. Moniteur système Windows avec SQL Server

Le moniteur système permet de mesurer de nombreux paramètres de fonctionnement d'un système Windows 2000 à l'aide de compteurs. SQL Serveur ajoute ses propres compteurs dans le moniteur système. Parmi eux :

- E/S SQL Server
- utilisation de la mémoire de SQL Server
- connexions des utilisateurs à SQL Server
- verrouillage de SQL Server
- activité liée à la réplication

8.3.3. Fenêtre Activité en cours de SQL Server Enterprise Manager

Cette fenêtre permet la visualisation de plusieurs informations importantes quant aux performances de SQL Server : le nombre de connexions ouvertes, l'identité des connectés, les verrous et blocages.

8.3.4. Outils Transact-SQL

Transact-SQL fournit plusieurs moyens de surveiller SQL Server :

Procédures stockées système	Informations obtenues
sp_who	Processus et utilisateurs en cours de SQL Server.
sp_lock	Informations concernant les verrous (verrous actifs, blocages et interblocages).
sp_spaceused	Espace disque utilisé par une table ou une base de données.
sp_helpdb	Bases de données et leurs objets.
sp_monitor	Statistiques de SQL Server (temps total de traitement, nombre de lectures & écritures, connexions).
sp_helpindex	Index d'une table.
sp_statistics	Tous les index d'une table en particulier.
Variable globale	Objet
@@connections	Contient le nombre de connexions ou de tentatives de connexion effectuées depuis le dernier démarrage de SQL Server.
@@error	Contient le numéro d'erreur de la dernière instruction Transact-

	SQL exécutée.
@@spid	Contient l'identificateur de processus du serveur pour le processus utilisateur en cours. Permet d'identifier le processus utilisateur en cours dans le résultat de sp_who.
@@procid	Contient l'identificateur de la procédure stockée en cours.
Instruction	Description
set statistics IO	Affiche des informations concernant le volume d'activité disque généré par les instructions Transact-SQL.
set statistics time	Affiche le temps en millisecondes requis pour analyser, compiler et exécuter chaque instruction.
set statistics profile	Affiche le profil d'exécution de la requête.
set showplan_text on/off	Empêche SQL Server d'exécuter la requête et renvoie des informations détaillées sur l'exécution des instructions.

8.3.5. Générateur de profils SQL Server

Le générateur de profils SQL Server permet de tracer certains événements SQL et de stocker les résultats dans une table, un fichier ou un script Transact-SQL. Les événements les plus judicieux à tracer sont :

- Les requêtes qui ne s'exécutent pas convenablement
- Les requêtes qui causent des analyses de tables
- Les problèmes d'interblocage
- Les tentatives et les échecs de connexion, les connexions et les déconnexions
- Les opérations d'écriture et de lecture sur disque
- L'utilisation du processeur pour chaque instruction
- Le délai d'attente pour tous les événements suivant l'exécution

Deux procédures stockées peuvent aider à l'utilisation du générateur de profils :

- sp_trace_create trace des événements spécifiques sur le serveur
- trace_produce_blackbox crée un résumé des 5 derniers Mo d'événements.

8.3.6. Analyseur de requêtes SQL Server

L'analyseur de requêtes SQL Server est un outil graphique de rédaction, d'exécution et d'optimisation de requêtes. Il permet de consulter les résultats d'une requête, d'en analyser le plan d'exécution et assiste à l'optimisation. Les différentes options d'optimisation disponibles dans l'analyseur de requêtes sont :

- Afficher le plan d'exécution de requête. Le plan d'exécution détaille les choix retenus par l'optimiseur de requêtes. Il montre comment les tables sources sont accédées, et les méthodes utilisées pour en extraire les données (utilisation d'index, par exemple).
- Afficher la trace du serveur. La trace du serveur permet de visualiser les données capturées en fonction des événements, des colonnes de données et des filtres que vous avez sélectionnés. Les données de trace sont enregistrables pour pouvoir être analysées ultérieurement.
- Afficher les statistiques du serveur. Ces statistiques permettent d'évaluer les performances d'une requête. Elles s'activent au moyen de l'instruction SET pour les statistiques SHOWPLAN, STATISTICS IO et STATISTICS TIME.
- Afficher les statistiques du client. Cette option permet d'afficher les statistiques liées au client. Elles peuvent être de trois types : application, réseau et horaires.
- Assistant Paramétrage d'index. Avec l'utilisation de cet assistant, l'Analyseur de requêtes donne des suggestions quant aux index qui peuvent permettre d'optimiser les requêtes.

9. Transfert de données

9.1. Présentation du transfert de données

9.1.1. Raisons justifiant l'importation et l'exportation de données

Il est souvent nécessaire d'importer ou d'exporter des données. Ceci peut donner des nouvelles versions de fichiers. Il peut être également intéressant d'enregistrer le fichier sous un autre format lisible par un autre produit.

Dans tous les cas, il est indispensable de connaître la source et la destination des données. Vous pouvez être amené à opérer des transformations de données pendant l'importation ou l'exportation.

9.1.2. Raisons justifiant la transformation de données

La transformation des données est nécessaire dans de nombreux cas :

- changement des formats de données (conversions de date, booléen ...)
- mappage des données (regroupement de données provenant de plusieurs sources de données)
- homogénéisation des données (conversion des données importées pour correspondre aux données du fichier destinataire)
- validation des données (contrôler que les données remplissent bien une condition pour être importées ou exportées)
- planification de l'importation ou l'exportation (gain de temps et simplification des opérations)
- importation ou exportation entre environnements hétérogènes (entre bases Access/SQL Server, Oracle/SQL Server ...)

9.2. Outils d'importation et d'exportation de données disponibles dans SQL Server

9.2.1. Assistant Importation/exportation DTS

Cet assistant est utilisé pour les transferts de données entre environnements hétérogènes. Des lots DTS peuvent être créés de manière interactive pour les données hétérogènes.

9.2.2. Concepteur DTS

Utilisé dans le cas de sources de données hétérogènes provenant de multiples sources. Les administrateurs peuvent alors spécifier des flux de travail complexes sur les données.

9.2.3. Transfert d'objets DTS

Le transfert concerne ici un transfert entre 2 bases de données SQL Server. Il est possible de transférer tous les objets d'une base ou seulement une partie. En effet, il est possible de transférer des tables, vues, procédures stockées, règles, valeurs par défaut, connexions, utilisateurs, rôles, index, contraintes ...

9.2.4. Insertion en bloc DTS

Aucune transformation n'est possible mais c'est le transfert de loin le plus rapide. Son utilisation est un gain très important de performances. Ce mode permet la copie de données à partir d'un fichier texte.

9.2.5. Programme de copie en bloc (utilitaire bcp)

Ce mode permet un transfert en mode natif, de données entre une table SQL Server et un fichier de données. bcp est un utilitaire en ligne de commande.

9.2.6. Réplication

Maintien de copies de données sur plusieurs bases de données. Ces copies peuvent régulièrement être mises à jour. La réplication a généralement lieu entre différents serveurs. Le problème réside dans le fait que la cohérence des données n'existe pas en permanence.

9.3. Présentation des services de transformation de données

9.3.1. Vue d'ensemble des services DTS

Les services DTS permettent de manipuler des sources de données au format texte, OLE DB, ou ODBC. Ces services permettent et sont utilisés pour :

- l'importation et exportation de données de diverses sources vers diverses destinations
- transformer des données
- transférer des objets entre différentes bases de données SQL Server
- créer des objets de transformation personnalisés compatibles avec d'autres produits
- utiliser les fournisseurs OLE DB d'autres éditeurs de produits pour accéder à certaines applications
- construire des Data Warehouses et data marts à partir de transferts de plusieurs sources hétérogènes ou de planifications régulières

9.3.2. Outils des services DTS

Afin de définir plusieurs tâches à effectuer dans un ordre précis pour l'importation, l'exportation et la transformation de données, 3 outils sont mis à disposition de l'administrateur par SQL Server.

Outil	Description
Assistant importation/exportation DTS	Il permet : <ul style="list-style-type: none">• la copie entre plusieurs sources de données hétérogènes• transférer un schéma de base de données• copier des tables et des requêtes (mêmes complexes avec jointures) avec leur résultat• planifier l'exécution de lots DTS
Concepteur DTS	Création et modification graphique de lots DTS. Il permet de réaliser les mêmes tâches que l'assistant vu précédemment, mais aussi de créer des transformations complexes et spécifier une séquence complexe d'opérations à réaliser pour un lot DTS.
Utilitaire dtsrun	Utilisé pour extraire, exécuter, supprimer, et remplacer un lot DTS

9.4. Transformation de données à l'aide des services DTS

9.4.1. Transformation et mappage des données

Le mappage de données va permettre d'adapter le mieux possible les données d'une source à la destination. Pour cela, on va pouvoir préciser des indicateurs de transformation définissant des méthodes de conversion (type de données, taille colonne, valeur NULL ...).

Chaque base de données a ses propres règles de fonctionnement (le nommage par exemple). Les services DTS vont se charger de trouver les meilleures correspondances entre 2 bases de données.

Il est possible aussi de changer manuellement les mappages DTS pour des valeurs personnalisées.

Plusieurs intégrations de données sont possibles :

- combinaison de plusieurs sources pour une destination unique
- à partir d'une source pour plusieurs destinations
- combinaison de colonnes pour une colonne unique
- à partir d'une colonne pour plusieurs colonnes

9.4.2. Définition des tâches de transformation

Un lot DTS est composé d'une ou plusieurs tâches qui sont définies par l'administrateur. Chaque tâche représente une unité de travail.

Voici les actions pouvant être définies pour une tâche :

- exécution d'une instruction Transact-SQL (requête SQL, insertion par bloc, requête contrôlée par les données)
- exécution d'un script (JScript, VBScript, PerlScript, ActiveX...)
- exécution d'une application externe (exécution de processus, envoi de courrier)
- transfert d'objets SQL Server
- exécution d'un autre lot DTS

9.4.3. Définition des flux de travail

Un flux de travail contrôle la séquence d'exécution de chaque étape d'un lot DTS et est défini par des contraintes. Il permet de contrôler la priorité des étapes.

Une contrainte de priorité organise le flux et les étapes du lot DTS. Elle permet le contrôle de chaque étape d'un flux.

Exemple de contrainte de priorité : **A l'achèvement.** Cette contrainte s'exécute automatique à l'issue d'une étape.

Il est intéressant d'organiser les étapes selon les besoins :

- Etapes organisées selon un certain ordre
- Etapes exécutées en parallèle pour des questions de performances
- Etapes organisées selon un certain ordre et exécutées en parallèle pour des questions de performances

Il est également possible de spécifier une priorité pour une étape. Il en existe 3 types : inactive, normale ou élevée.

9.4.4. Création d'un lot DTS

La création d'un lot DTS passe par 2 étapes essentielles :

- **son enregistrement**
- **sa sécurisation**

L'enregistrement du lot est essentiel si l'on désire pouvoir le modifier, le réutiliser et le planifier par la suite. Un lot DTS est exécuté immédiatement si il n'est pas enregistré. Le lot DTS peut être stocké à plusieurs endroits :

- **dans un fichier de stockage structuré de type COM** : un fichier d'extension .dts est créé et peut être accédé sans SQL Server. Il est possible d'enregistrer plusieurs versions d'un même lot et plusieurs lots dans un même fichier
- **dans les services de métadonnées SQL Server** : les services sont en fait une base de données recensant des informations sur les objets et leurs relations. Le lot DTS est enregistré en tant qu'entrée de cette base
- **dans la base de données msdb** : il n'est pas possible d'enregistrer plusieurs lots dans un même fichier
- **dans un fichier Visual Basic** : le lot pourra alors être intégré dans une application Visual Basic.

Sa sécurisation peut passer par son cryptage. En effet, il est possible d'enregistrer le lot en fichier de stockage structuré pour protéger certaines informations telles que les noms d'utilisateur et mots de passe.

Il existe 2 niveaux de sécurité pour les lots DTS :

- **Mot de passe utilisateur** : les lots sont utilisables et ne permettent pas l'accès aux noms d'utilisateur et mots de passe
- **Mot de passe propriétaire** : il est possible d'accéder aux collections et propriétés.

9.4.5. Exécution et planification d'un lot DTS

L'extraction ou l'exécution d'un lot DTS se fait à l'aide de **SQL Server Enterprise Manager** ou de l'utilitaire en ligne de commande **dtsrun**.

Il est possible de planifier l'exécution d'un lot DTS à un moment précis de la journée, ou de manière récurrente dans le temps (une fois par mois par exemple). 2 méthodes permettent de paramétrer cette planification :

- **l'Assistant Importation / exportation DTS** si l'enregistrement du lot DTS se fait dans la base de données **msdb**
- **SQL Server Enterprise Manager** si le lot DTS est exécuté avec l'utilitaire **dtsrun**.

10. Maintien d'une haute disponibilité

10.1. Présentation de la disponibilité

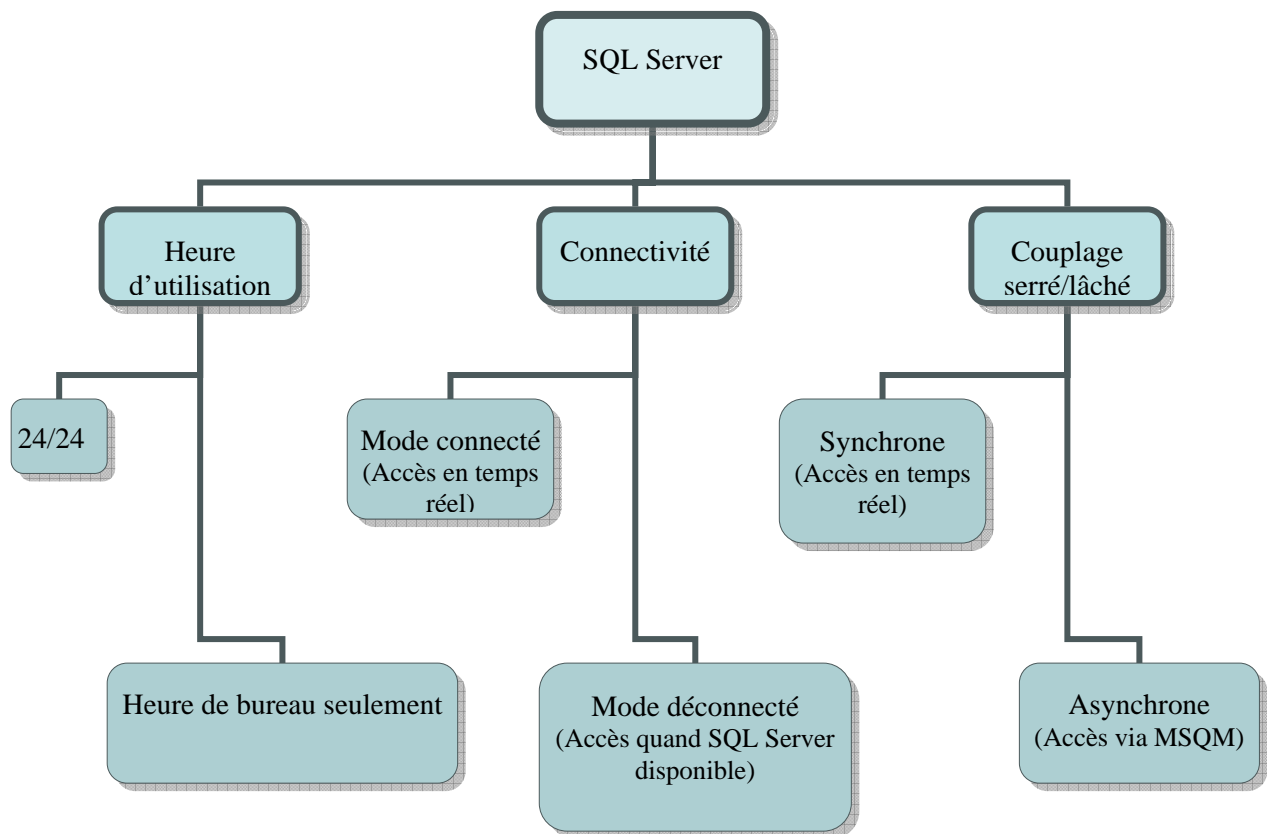
10.1.1. Définition de la disponibilité

Il peut arriver que les services s'appuyant sur SQL Server doivent être disponibles 24h/24h, 7 jours sur 7. Cela implique que le serveur SQL Server doit lui aussi être disponible sur ces mêmes périodes. Cependant, certains facteurs comme la fiabilité du réseau, les défaillances matérielles ou logicielles, et les catastrophes naturelles, peuvent provoquer un arrêt inopiné du serveur SQL Server.

L'administrateur doit donc implémenter une stratégie qui limitera les risques d'interruptions, et ainsi augmenter la disponibilité du système.

10.1.2. Détermination des besoins en matière de disponibilité

Nous pouvons distinguer trois types de besoin en matière de disponibilité :



10.1.3. Evolutivité

L'évolutivité permet de fournir un service permanent aux utilisateurs. Nous pouvons distinguer deux types d'évolutivités :

Nom	Description	Solution
Evolutivité verticale	Augmentation du nombre de requêtes pouvant être traitées simultanément.	Augmentation de la mémoire vive. Utilisation de plusieurs processeurs.
Evolutivité horizontale	Répartition de charge sur plusieurs serveurs SQL Server pour un traitement distribué.	Réplication. Serveur de secours en lecture seule. Fédération de serveur.

10.1.4. Optimisation de la disponibilité avec les serveurs d'entreprise .NET de Microsoft

Les serveurs d'entreprise .NET permettent de concevoir, déployer et gérer des services évolutifs basés sur le Web. Une application distribuée se divise en trois niveaux répartis sur plusieurs ordinateurs. Chaque niveau étant indépendant, la maintenance et la configuration de chaque niveau peuvent être indépendants, fournissant une grande souplesse pour l'administration de l'application.

Niveau présentation :

Il s'agit de la couche la plus haute. C'est à ce niveau qu'est fourni à l'utilisateur un service logiciel lui permettant d'interagir avec l'application. Il peut s'agir d'une application Windows locale, ou d'une application Web accessible via un navigateur Web.

Niveau Métier :

Ce niveau est généralement constitué de composants COM qui servent d'interface entre l'utilisateur et la source de données. C'est à ce niveau qu'interviennent les groupes de serveurs en cluster pour la répartition de charge, ou encore MSQM.

Niveau Données :

C'est à ce niveau que résident physiquement les données, c'est-à-dire le ou les serveurs SQL Server. On peut éventuellement mettre en place une solution RAID1 ou RAID5 pour garantir une tolérance de panne.

10.2. Amélioration de la disponibilité en utilisant un cluster de basculement

10.2.1. Utilisation d'un cluster de basculement SQL Server

Au cours de l'installation de SQL Server, il est possible de créer des serveurs virtuels. Un serveur virtuel est affecté à chaque groupe de ressources du cluster Windows sous-jacent. Le programme d'installation de SQL Server peut également servir à récupérer un cluster de basculement à la suite d'une défaillance. Dans ce cas, il sert à supprimer un nœud du cluster puis à le rajouter, une fois qu'il est sain.


10.2.2. Clustering actif/passif

Quand SQL Server fonctionne en mode Actif/Passif, l'instance n'est active que sur le premier nœud. Le deuxième nœud ne passe en mode actif que si le premier nœud subit une défaillance. Dans ce cas le deuxième nœud récupère les ressources et la charge de travail du nœud principal.

10.2.3. Clustering actif/actif

Dans une configuration Actif/Actif chaque nœud héberge une ou plusieurs copies de SQL Server. Il y a donc un serveur virtuel par nœud. Ainsi, si un des nœuds tombe, tout le traitement qu'aurait dû effectuer ce nœud défectueux sera transféré sur un autre nœud exécutant un SQL Server.

Par contre, dans ce cas, les performances de ce deuxième nœud se verront amoindries car il exécute les travaux de deux serveurs.

-  Pour configurer votre serveur SQL Server en clustering, vous devez vous assurer que les conditions suivantes sont validées :
- Chaque nœud doit exécuter Windows Advanced Server ou supérieur.
 - Une copie de SQL Server est installée sur le disque ou la batterie de disques SCSI partagé(e) par les deux nœuds.
 - Exécuter Comclust.exe sur chaque nœud du cluster pour que le service MSDTC s'exécute.

10.3. Serveurs de secours et transmission des journaux

10.3.1. Maintien d'une haute disponibilité en utilisant un serveur de secours et la transmission des journaux

Un serveur de secours fonctionne de la même manière qu'un nœud passif dans un environnement de cluster. A savoir qu'il peut remplacer un serveur SQL Server principal si ce dernier tombe en panne. La différence réside dans le fait qu'il n'est pas nécessaire d'installer le service cluster pour bénéficier de ce type de tolérance de panne.

Pour maintenir la synchronisation entre le serveur principal et le serveur de secours, vous pouvez copier et charger les journaux de transaction du serveur principal sur le serveur de secours. Vérifiez également que votre serveur SQL utilise le mode de restauration complète pour permettre la sauvegarde des journaux de transaction.

10.3.2. Configuration de la transmission des journaux en utilisant l'Assistant Plan de maintenance de base de données

Sous SQL Server 2000 Edition Entreprise, la transmission des journaux entre le serveur principal et un serveur de secours peut s'effectuer via l'**Assistant plan de maintenance de base de données**. Vous devez cependant considérer les 3 points suivants avant d'utiliser cet assistant :

- L'utilisateur exécutant cet assistant doit être membre du rôle de serveur **sysadmin**
- Il n'est pas possible de planifier une transmission des journaux sur plusieurs bases de données en même temps.
- Il n'est possible de sauvegarder les journaux que sur disque, pas sur des bandes.

10.3.3. Modification des rôles de transmission des journaux

Pour remplacer un serveur principal par un serveur de secours, exécutez la procédure présentée ci-dessous :

Procédure	Description
sp_change_primary_role	Exécutée sur le serveur SQL Server principal, cette procédure désactive la tâche sauvegarde du journal, puis sauvegarde le journal
sp_change_secondary_role	Transforme un serveur SQL Server de secours en serveur SQL Server principal.
sp_change_monitor_role	L'exécution de cette procédure sur le serveur de

	surveillance courant met à jour la surveillance pour refléter les nouvelles bases de données principale et secondaire, ainsi que le nouveau répertoire de sauvegarde des journaux de transactions.
sp_resolve_logins	Met à jour les connexions de la base de données sur le nouveau serveur principal en utilisant la sauvegarde provenant du serveur principal d'origine.

Le basculement d'un serveur de secours en serveur principal peut nécessiter la modification du nom du nouveau serveur principal pour que les clients puissent toujours communiquer avec ce serveur en toute transparence. Pour changer le nom d'un serveur SQL Server, il suffit de lancer à nouveau le programme d'installation de SQL Server.

11. Présentation de la réplication

11.1. Présentation des données distribuées

11.1.1. Besoin en données distribuées

L'avantage d'un environnement à données distribuées réside dans le fait que plusieurs copies des mêmes informations et données se trouvent sur plusieurs serveurs. Les objectifs d'un tel environnement sont :

- rapprocher l'utilisateur des données qu'il recherche
- diviser le réseau en sites qui pourront être indépendants les uns des autres (sites autonomes)
- diviser les besoins : un site pour la lecture des informations par exemple, et un autre pour les transactions de mise à jour
- réduire les conflits de toute nature (transactions, réseau ...)

11.1.2. Facteurs à prendre en compte lors de la distribution des données

Pour un environnement distribué, il existe deux stratégies : la réplication et les transactions distribuées. Si vous désirez concevoir un tel environnement, il vous est possible de mixer les 2 stratégies. Voici un tableau récapitulatif de ces 2 stratégies :

Stratégie	Description
Réplication	C'est une duplication récente des données d'une base de données source vers une base de données de destination. Cette stratégie peut être mise en place sur des sites autonomes et dans ce cas, il peut y avoir une intermittence de la mise en ligne de ces sites.
Transactions distribuées	Cette solution ne convient que si il y a un besoin absolu que les données soient synchronisées en permanence sur tous les serveurs. Toutes les copies de données doivent être identiques au même moment sur tous les serveurs. Lors d'une transaction distribuée, celle-ci doit pouvoir s'exécuter sur tous les serveurs inclus dans la transaction. Si un serveur fait défaut, la transaction ne peut avoir lieu sur aucun serveur.

Afin de choisir la stratégie correspondant le mieux à vos besoin, veuillez à bien prendre en compte vos besoins en termes de :

- **durée/latence** : À quelle fréquence doivent être synchronisées les données ?
- **cohérence** : De quel délai dispose-t-on pour atteindre la cohérence transactionnelle ?
- **sites autonomes** : Quelles sont les possibilités temporelles pour se connecter à un serveur central pour synchroniser les données ?

11.1.3. Méthodes de distribution des données

Il existe différentes méthodes de distribution de données offrant plus ou moins de latence ou d'autonomie pour les sites. Les voici :

Méthodes	Description
Transactions distribuées	Tous les sites ont les mêmes données au même moment. On utilise un protocole de validation à 2 phases qui garantit la réalisation simultanée d'une transaction sur tous les sites. Le moins d'autonomie et de latence
Réplication transactionnelle avec abonnements mis à jour immédiatement ou en attente	Lorsque des données locales sont modifiées, ces modifications s'appliquent à la fois aux données sources et aux données locales. MS DTC met automatiquement à jours ces données, évitant les conflits de mise à jour. Une réplication s'opère dès que les mises à jour ont lieu. Autonomie et latence très légères
Réplication transactionnelle	Dans ce type de réplication, seules les données qui ont subi des modifications sont distribuées. La séquence des transactions est conservée. Les conflits sont évités du fait que les données sont modifiées à un seul endroit. Autonomie et latence légères
Réplication de capture instantanée avec abonnements mis à jour immédiatement ou en attente	Cette méthode est similaire à la précédente, mis à part que la réplication n'est effectuée que de manière périodique, ce qui permet une plus grande autonomie. Autonomie et latence moyennes
Réplication de capture instantanée	Une photo des données est prise à n'importe quel moment (décidé par l'utilisateur) et remplace les données sur un serveur de destination. Cette action peut être automatisée ou avoir lieu sur simple demande. Autonomie et latence élevées
Réplication de fusion	Chaque site effectue les modifications souhaitées. Ils mettent à jour un serveur central périodiquement. Il y a un certain risque de conflit. Autonomie et latence très élevées

11.2. Présentation de la réplication SQL Server

11.2.1. Métaphore éditeur-abonné

Le mécanisme de réplication SQL Server est entre autre basé sur la métaphore éditeur-abonné. Cette métaphore se compose de trois entités : **l'éditeur**, le **distributeur**, et **l'abonné**.

L'éditeur :

- Il héberge les données et les met à disposition d'un distributeur en vue d'une réplication.

Le distributeur :

- Il stocke les métadonnées, les données d'historique, et les transactions. Il stocke les modifications afin de les transmettre aux abonnés.

L'abonné :

- L'abonné possède une copie de la base de données et applique les modifications transmises par le distributeur.

11.2.2. Publications et articles

Une publication peut être comparée à une revue ou un magazine. Ce dernier est composé d'articles, c'est-à-dire un ensemble d'objets de base de données, ou de parties de tables utilisateurs.

Le processus de réplication distribue ensuite des publications (magazines) aux abonnés.

11.2.3. Filtrage des données

SQL Serveur implémente également un système de filtre des données qui permet de publier uniquement certaines parties d'une ou plusieurs tables. Nous allons pouvoir filtrer verticalement ou horizontalement.

Un filtre horizontal permettra de sélectionner uniquement certaines lignes d'une table, et permettra donc d'accéder à tous les champs d'une entrée de cette table.

Un filtre vertical permettra de sélectionner certaines colonnes d'une table, et ainsi avoir uniquement accès aux informations stockées dans ces colonnes.

Exemple de filtre horizontal :

Nom	Prénom	Sexe	Tel	Adresse	Département	Spécialité
McLane	John	1	0115478963	11 rue des Champs	Sécurité	ISA
Fox	Mike	1	0536548532	15 Av Martin	Sécurité	ISA
Lopez	Julie	0	0412365898	32 rue des Pics	Développement	C++
Bauer	Yann	1	0236521478	23 rue Albin	Administration	Win2K

Exemple de filtre vertical :

Nom	Prénom	Sexe	Tel	Adresse	Département	Spécialité
McLane	John	1	0115478963	11 rue des Champs	Sécurité	ISA
Fox	Mike	1	0536548532	15 Av Martin	Sécurité	ISA
Lopez	Julie	0	0412365898	32 rue des Pics	Développement	C++
Bauer	Yann	1	0236521478	23 rue Albin	Administration	Win2K

11.2.4. Abonnements

Deux types d'abonnements sont disponibles sous SQL Server :

Abonnement envoyé :

- L'abonnement est défini au niveau du serveur de publication. Ceci permet une administration centralisée des abonnements. (sécurité élevée)

Abonnement extrait :

- L'abonnement est défini au niveau du serveur d'abonnement, l'abonné s'inscrit donc lui-même à une publication (sécurité faible).

11.3. Agents de réplication SQL Server

L'Agent SQL Server peut lancer des travaux d'agents de réplication SQL Server de manière périodique et automatique. Voici une description des différents agents de réplication existants :

Agents de réplication	Description
Agent de capture instantanée	<p>Cet agent s'exécute au niveau du distributeur. Il a pour rôle de :</p> <ul style="list-style-type: none"> • stocker les instantanés (photos de données) • préparer les schémas • préparer les fichiers de données initiales des tables publiées • préparer les procédures stockées <p>Toutes les informations relatives aux synchronisations sont stockées dans la base de données distribution.</p>
Agent de distribution	<p>Cet agent est utilisé pour la réplication de capture instantanée et la réplication transactionnelle. Il envoie aux abonnés les transactions et les travaux de capture instantanée présents dans la base de données « distribution ». Cet agent s'exécute généralement sur le distributeur pour les abonnements envoyés et au niveau de l'abonné pour les abonnements extraits.</p>
Agent de lecture du journal	<p>Cet agent est dédié à la réplication transactionnelle. Il veille sur les journaux de transactions de chaque base de données et recherche les transactions à répliquer.</p> <p>Il copie alors les transactions à répliquer vers la base de données distribution. La base de données les distribuera alors par la suite aux bases de données des abonnés.</p> <p>Chaque base de données dispose de son propre agent de lecture du journal</p>
Agent de fusion	<p>Cet agent est utilisé avec la réplication de fusion et applique la capture instantanée initiale à l'abonné, puis déplace et réconcilie les modifications incrémentielles apportées aux données.</p> <p>Chaque abonnement de fusion à un agent de fusion connecté à l'éditeur et à l'abonné. Il les met tous les deux à jour.</p>
Agent de lecture de file d'attente	<p>Cet agent applique les messages d'une file d'attente à la publication appropriée. Il est utilisé avec la réplication de capture instantanée ou transactionnelle avec l'option de mise à jour en attente, ou si mise à jour immédiate avec mise à jour en attente en tant qu'option de basculement est activée.</p> <p>Cet agent est un agent multi-thread qui s'exécute sur le distributeur.</p> <p>Un seul agent de lecture répond aux éditeurs et publications d'un distributeur donné.</p>

11.4. Types de réplications SQL Server

11.4.1. Vue d'ensemble des types de réplications

Nous pouvons distinguer trois types de réplications sous SQL Server :

Réplication	Description
Réplication de capture instantanée	Transfert en blocs périodique de nouvelles captures instantanées de données
Réplication transactionnelle	Les données sont répliquées en temps réel (temps de latence minimal)

Réplication de fusion	Les modifications autonomes apportées aux données répliquées sont fusionnées ultérieurement
-----------------------	---

11.4.2. Remarques sur l'utilisation de la réplication de fusion

La réplication de fusion modifie le schéma pour éviter ou résoudre les conflits.

Modifications du schéma

Pour que la réplication de fusion fonctionne correctement, SQL Server apporte trois modifications importantes au schéma de la base de données de publication.

SQL Server identifie une colonne unique pour chaque ligne de la table répliquée. Ainsi, la ligne peut être identifiée de manière unique parmi plusieurs copies de la table.

Il ajoute plusieurs tables système pour opérer le suivi des données, prendre en charge une synchronisation efficace et la détection, la résolution et le compte-rendu des conflits.

SQL Server crée des déclencheurs sur les tables au niveau de l'éditeur et de l'abonné pour assurer le suivi des modifications apportées aux données de chaque ligne ou éventuellement de chaque colonne. Ces déclencheurs capturent les modifications apportées à la table et les enregistrent dans des tables système de fusion.

Étant donné que SQL Server peut prendre en charge plusieurs déclencheurs du même type sur une table de base, les déclencheurs de la réplication de fusion n'interfèrent pas avec ceux définis par l'application ; en d'autres termes, les déclencheurs définis par l'application et ceux de la réplication de fusion peuvent coexister.

Résolution des conflits

Ce type de réplication permettant des mises à jour indépendantes, des conflits peuvent survenir. Pour résoudre ces conflits, la réplication de fusion se base sur des priorités.

- L'Agent de fusion analyse chaque ligne mise à jour.
L'historique des modifications apportées à une ligne s'appelle le lignage. Lorsque l'Agent de fusion fusionne les modifications et rencontre une ligne pouvant avoir subi plusieurs modifications, il examine le lignage pour déterminer l'existence éventuelle d'un conflit. La détection des conflits peut avoir lieu au niveau de la ligne ou de la colonne.
- L'Agent de fusion évalue les valeurs entrantes et en cours des données, et tout conflit entre les anciennes et les nouvelles valeurs est automatiquement résolu en fonction des priorités affectées.
- Les valeurs des données ne sont répliquées sur d'autres sites que pendant la phase de synchronisation, qui peut se produire à quelques minutes, jours ou même semaines d'intervalle.
- Vous pouvez également personnaliser les déclencheurs pour définir votre propre résolution de conflit.

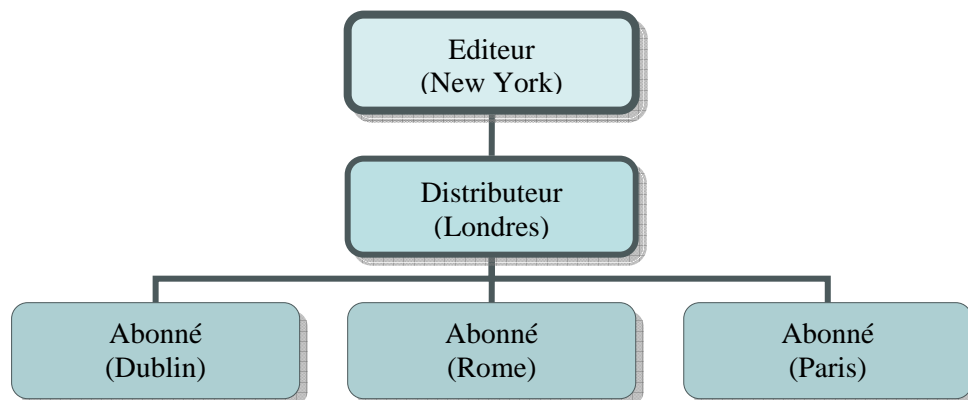
11.5. Modèles physiques de réplication

11.5.1. Le modèle Éditeur central/Distributeur distant

Ce modèle est très utile pour partitionner une base de données via l'utilisation du filtrage horizontal. L'intérêt de ce modèle se révèle par exemple pour une entreprise ayant plusieurs sites distants ayant chacun une liaison bas débit vers un site de centralisation des données. Dans ce cas, le site

central joue le rôle d'éditeur, un site équipé d'une liaison haut débit devient distributeur, et chaque site distant est abonné.

Exemple :

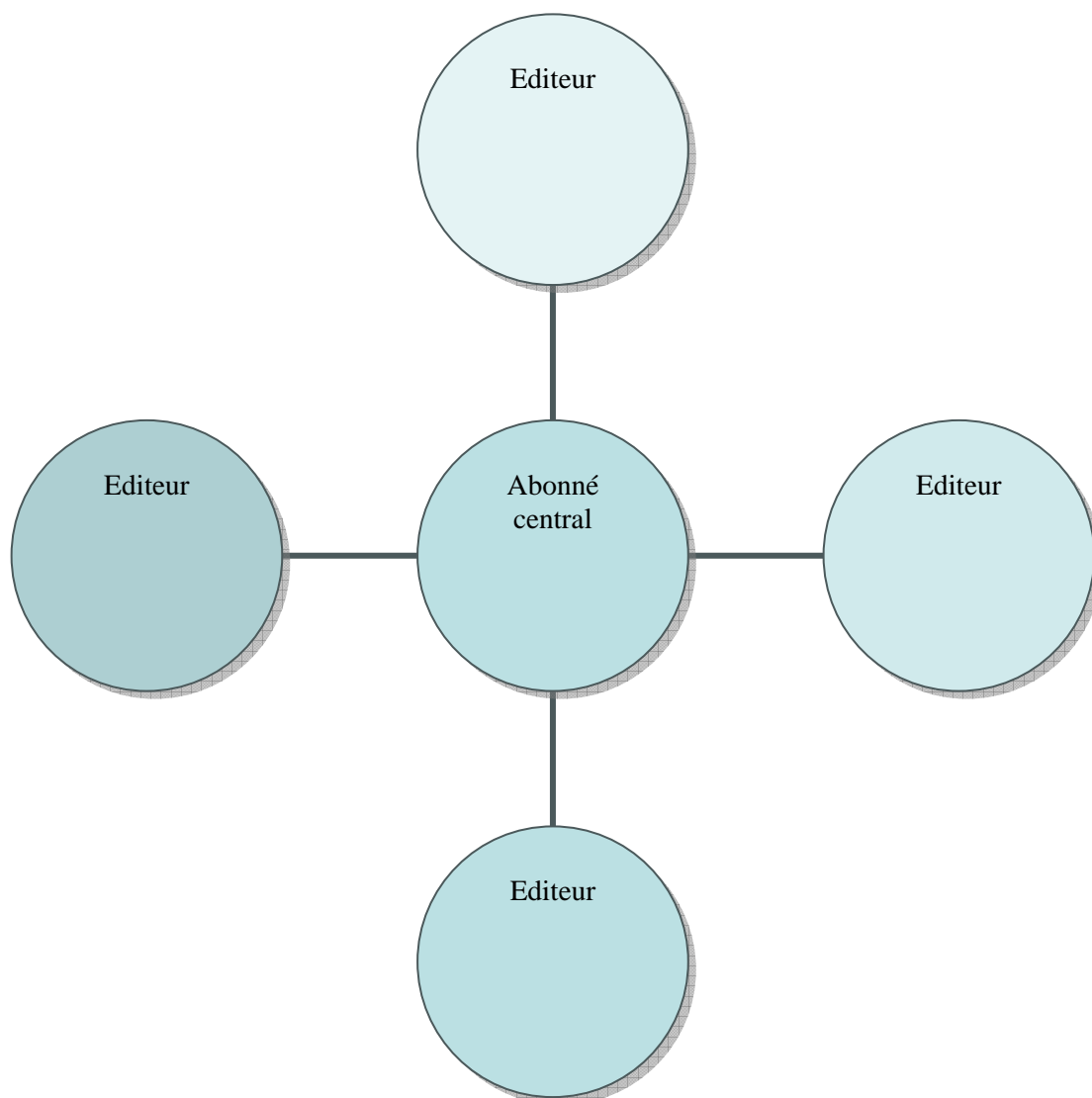


Ce modèle, par l'usage du filtrage horizontal, permet d'alléger la bande passante nécessaire à la synchronisation entre l'éditeur et les abonnés.

11.5.2. Le modèle Abonné central/Éditeurs multiples

Ce modèle, à l'inverse du précédent, permet la centralisation des données entre plusieurs sites. Dans ce cas, tous les sites distants sont éditeurs/distributeurs et un site central est abonné à tous ces sites. Ainsi, chaque site héberge sa propre base maîtresse (au détriment de la bande passante).

Exemple :



11.5.3. Le modèle Éditeurs multiples/Abonnés multiples

Ce modèle s'apparente au modèle réseau « Full mesh » : chaque site est à la fois éditeur et abonné aux autres sites.

Exemple :

