Labo.net SUPINFO

http://www.labo-dotnet.com

SQL Server 2000

SUPINFO DOT NET TRAINING COURSE

Auteur : Steve Beaugé

Version 1.2 – 2 novembre 2004
Nombre de pages : 61

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

Table des matières

1. QU'EST CE QUE SQL SERVER ? 4

2. INSTALLATION ET CONFIGURATION DE SQL SERVER 2000 5

2.1. INSTALLATION DU SERVEUR 5
2.1.1. Configuration requise 5
2.1.2. Etape n°1 : Menu CDRom SQL Server 5
2.1.3. Etape n°2 : Choix du programme à installer 6
2.1.4. Etape n°3 : Options d'installations 7
2.1.5. Etape n°4 : Choix des composants 8
2.1.6. Etape n°5 : Choix du type d'installation 9
2.1.7. Etape n° 6 : Choix du compte de service 10
2.1.8. Etape n°7 : Mode d'authentification 11
2.1.9. Etape n°8 : Fin d'installation 12

3. PRESENTATION DES OUTILS D'ENTREPRISE 14

3.1. SERVICE MANAGER 14
3.2. ENTERPRISE MANAGER 15
3.2.1. Ajouter un serveur à Enterprise Manager 15
3.2.2. Configurer le serveur 17
3.2.3. Créer une base de donnée 18
3.2.4. Créer ou modifier une table 19
3.2.5. Modéliser la base de données. 19
3.2.6. Construction visuelle de vues 20
3.2.7. Editeur de procédures stockées 21
3.2.8. Conclusion 22
3.3. ANALYSEUR DE REQUETES 22

4. TACHES ADMINISTRATIVES COURANTES 25

4.1. TACHES DE POST-INSTALLATIONS 25
4.1.1. Lancement du serveur 25
4.1.2. Configurations diverses 25
4.1.3. Configurer la prise en charge de SQL XML dans IIS. 27
4.1.4. Accéder aux données 29
4.1.5. Utilisation d'un modèle (template) 29
4.1.6. Encore plus fort, l'utilisation de feuilles de style 31

5. INTRODUCTION AU TRANSACT SQL 33

5.1. QU'EST CE QUE LE T-SQL (TRANSACT SQL) 33

5.2. « NORTHWIND » 33
5.3. COMMENTAIRES 35
5.4. VARIABLES 35
5.4.1. Déclaration 35
5.4.2. Types de données 36
5.4.3. Utilisation des variables : 37
5.4.4. Types de données utilisateur 38
5.5. IF / ELSE 38
5.6. WHILE 40
5.7. GOTO / LABEL 42
5.8. CASE (SIMPLE) 42
5.9. CASE (RECHERCHE) 44
5.10. INTEGRATION DE SQL 44
5.11. PROCEDURES STOCKEES 46
5.11.1. Déclarer une procédure stockée 46
5.11.2. Modification d'une procédure stockée existante 47
5.11.3. Appel d'une procédure stockée 47
5.11.4. Paramètres 47
5.11.5. Paramètres optionnels 48
5.11.6. Direction des paramètres 49
5.11.7. Procédures stockées systèmes et prédéfinies 50
5.12. TRANSACTIONS 51
5.12.1. Utilité 51
5.12.2. Exemple : 52
5.13. CURSEURS 58
6. CONCLUSION 61

1. Qu'est ce que SQL Server ?

SQL Server 2000 est la dernière version du SGBDR de Microsoft (Système de Gestion de Base de Données Relationnelles). Il est particulièrement adapté aux systèmes d'E-Business et de DataWare Housing (on parle aussi de Workflow). Cette dernière version inclut un support XML et HTTP, permettant d'accéder aux données depuis un navigateur, ou d'une application pouvant créer des requêtes HTTP.

Ses avantages sont multiples :

- Performant : SQL Server se classe parmi les SGBDR les plus rapides (www.microsoft.com/sql/worldrecord).
- Evolutif et fiable : vous pouvez répartir la charge sur plusieurs serveurs, bénéficier des avantages des systèmes multi-processeurs (SMP – Sysmetric Multi Processing) et profiter des performances de Windows 2000 DataCenter Server qui supporte 32 processeurs et 64 GO de ram).
- Rapidité de mise en œuvre : avec SQL Server, le développement, le déploiement et l'administration d'applications destinées au Web sont accélérés grâce aux nombreuses fonctionnalités dédiées, ainsi qu'au support du Web.

Pour découvrir les fonctionnalités de SQL Server, rendez vous à cette adresse : http://www.microsoft.com/france/sql/decouvrez/fonction.asp qui vous présentera chacune des fonctionnalités.

2. Installation et configuration de SQL Server 2000

2.1. Installation du serveur

Je vais présenter ici comment installer la version « développeur » de SQL Server car la principale cible de ce document sont les développeurs. L'installation d'une autre version ne change que peu, à l'exception de la version Desktop (MSDE) qui est prévu pour être automatisée et redistribuée. Les versions PocketPC sortent également du cadre de ce chapitre du fait de la spécificité de la plateforme. Dans ces cas, reportez vous aux documentations spécifiques de chaque version.

2.1.1. Configuration requise

Voici la configuration requise pour l'installation de SQL Server selon Microsoft :

- Processeur Intel Pentium 166Mhz ou supérieur.
- Mémoire vive de 64Mo si l'environnement est Windows 2000. Notez également que pour la version entreprise, 64Mo suffisent mais il est vivement recommandé d'avoir 128 ou plus pour pouvoir supporter une charge importante dans le cadre d'un serveur de production.
- Espace disque entre 95 et 270 Mo selon les options installées (250Mo pour une installation standard)
- Affichage de 800*600 pixels et une souris pour l'utilisation des outils graphiques
- Un lecteur CDRom pour l'installation depuis un CDRom
- Système d'exploitation : Windows NT (avec service pack 5 ou plus) ou 2000 (XP inclus). Notez que la version Entreprise ne peut s'installer que sur les versions Server de ces systèmes d'exploitation. La version Desktop et la version personnelle peuvent s'installer également sous Windows Me ou 98. Les outils clients et les options de connectivité peuvent quant à eux être installés sur tous les Windows depuis Windows 95.

2.1.2. Etape n°1 : Menu CDRom SQL Server

Voici le menu apparaissant après inséré le CDRom SQL Server Developer Edition :

Figure 1 - Autorun de SQL Server 2000

Si ce menu n'apparaît pas lors de l'insertion de votre CDRom ou si vous lancez l'installation depuis un lecteur réseau ou une autre source, vous pouvez lancez manuellement l'installation en exécutant le fichier « autorun.exe » ou « setup.bat » pour sauter l'étape 1.

Observons maintenant ce menu. Voici maintenant les 5 options proposées :

- Composants de SQL Server 2000
  - C'est l'option pour installer le serveur SQL en lui-même et/ou les composants et outils additionnels.
- Composants requis pour SQL Server 2000
  - Si vous êtes sous Windows 95, vous pouvez mettre à niveau votre système d'exploitation pour supporter les outils clients de SQL Server.
- Aide sur l'installation et la mise à niveau
  - Ouvre le fichier d'aide de SQL Server dans laquelle vous trouverez entre autre les détails de l'installation du serveur.
- Consulter les notes de mise à jour
  - Corrections apportées au manuel qui n'ont pas pu être incluses dans le fichier d'aide original
- Visitez notre site Web
  - Renvoie l'utilisateur sur le site web de SQL Server : http://www.microsoft.com/france/sql/default.asp si vous installez la version française de SQL Server.

L'option qui nous intéresse est la première : Composants de SQL Server 2000.

2.1.3. Etape n°2 : Choix du programme à installer

Vous devriez arriver dans ce menu :

Figure 2 - Choix du programme à installer

Vous devez ici choisir quel composant installer.

Vous pouvez installer :

- Le serveur : C'est le serveur SQL en lui-même et/ou ses outils clients et ses fichiers de connectivité.
- Analysis Services : C'est un outil permettant d'analyser les données du serveur et de modéliser des dataware house

- English Query : Cet outil vous permet de formuler vos requêtes en anglais.

Installons le serveur. Vous pourrez relancer le menu du CDRom plus tard pour installer les deux outils supplémentaires.

2.1.4. Etape n°3 : Options d'installations

Cliquez sur « suivant » sur la première page. Vous arrivez alors à cet écran :

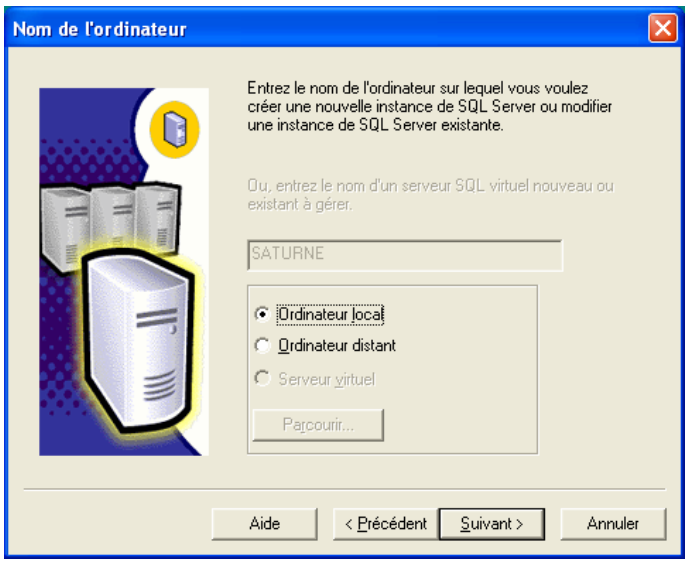


Figure 3 - Option d'installation

Cet écran vous permet de choisir entre :

- Installer le serveur SQL sur la machine locale, c'est cette option que nous choisirons
- Installer le serveur SQL sur une machine distante, ce qui permet d'installer le serveur SQL sur une autre machine. La procédure est un peu trop complexe à expliquer et sort du cadre de cet article. Reportez vous à l'aide fournie pour plus de détails.
- Installer un serveur SQL virtuel, permet d'installer, si l'ordinateur fait partie d'un cluster (et donc que MSCS - Microsoft Cluster Service est installé)

Gardons la première option. Vous devez maintenant choisir une installation :

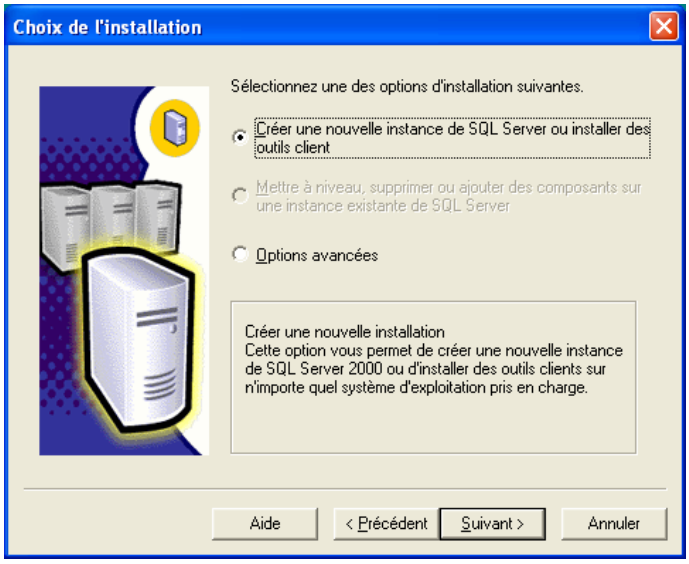


Figure 4 - Choix du type d'installation

Ce menu vous propose trois choix :

- Créer une nouvelle instance de SQL Serveur ou installer des outils clients
  - C'est le choix que nous allons faire. Cette option va nous permettre d'installer le serveur SQL, ou le outils clients.
- Mettre à niveau supprimer ou ajouter des composants sur une instance existante de SQL Server
  - Si une instance de SQL Server est déjà installée sur la machine, vous pourrez modifier son installation. Nous verrons plus loin ce qu'est une instance SQL
- Options avancées
  - Permet d'accéder aux options avancées. Ces options sortent du cadre de ce cours, mais voici brièvement leurs rôles :
    - « Enregistrer un fichier .ISS sans surveillance » : permet de créer un fichier de réponse pour automatiser une installation. Cette option est utile pour par exemple créer des installations du MSDE (SQL Server Desktop Engine) et ainsi de redistribuer ce dernier sans avoir besoin de demander à l'utilisateur de savoir configurer un serveur SQL.
    - « Reconstruire le registre » : permet de reproduire l'installation juste en recréant une ruche d'instance. Une ruche d'instance est une clef « racine » dans la base de registre dans laquelle les options de SQL Server sont sauvegardées. Chaque instance a sa propre ruche.
    - « Gérer un serveur virtuel pour la mise en clusters avec basculement » : Permet de procéder à des modifications sur des clusters existants, comme modifier le nom ou bien ajouter ou supprimer des nœuds de clusters.

Comme toujours, choisissons l'option par défaut. Après avoir validé, entrez votre nom et le nom de la société pour laquelle vous travaillez puis acceptez le CLUF (Contrat de Licence de l'Utilisateur Final) après l'avoir lu.

2.1.5. Etape n°4 : Choix des composants

Vous devriez arriver sur cet écran :

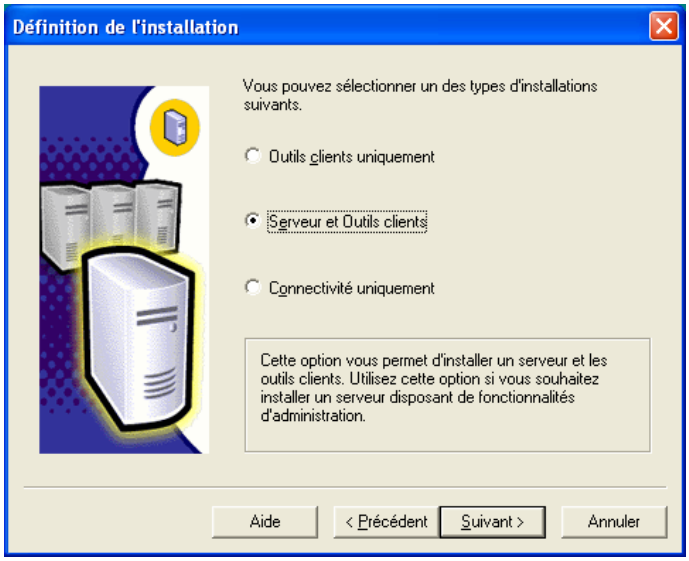


Figure 5 – Choix des composants

Trois options s'offrent alors à vous :

- Outils clients uniquement : Choisissez cette option si vous ne voulez que les outils clients et pas le serveur SQL. Utilisez par exemple cette option si vous disposez d'un serveur SQL centralisé pour n'installer que les outils permettant d'y accéder.

- Server et Outils clients : En plus d'installer les outils clients, on installe par le biais de cette option le serveur SQL.
- Connectivité uniquement : N'installe que des composants et des bibliothèques d'accès.

Choisissons d'installer le serveur SQL et les outils clients en cochant la deuxième option.

Il faut maintenant choisir une instance à installer :

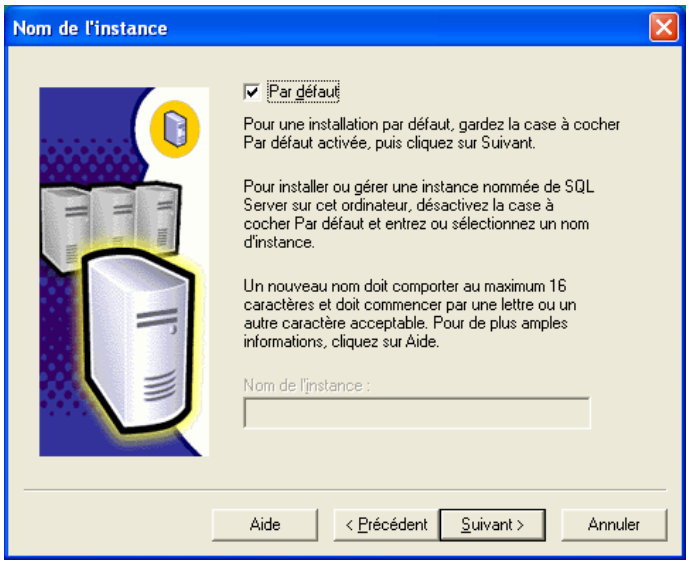


Figure 6 - Choix du nom de l'instance

SQL Server permet d'avoir sur la même machine plusieurs « instances ». Cela signifie que sur une machine quelconque vous pouvez avoir deux ou plus serveurs SQL en simultané. Chacun d'eux ayant alors ses propres bases de données, et sa propre ruche (une ruche est une clef « racine » de la base de registre dans laquelle sont stockées toutes les options du serveur SQL). Les instances sont différenciées par un nom attribué à chacune d'elle. Il est toutefois rare d'avoir besoin d'installer plusieurs instances sur une même machine.

L'instance par défaut est l'instance recherchée sur la machine lorsque aucune instance n'est spécifiée. Si vous installez SQL Server sur une machine sur laquelle il n'est pas installé et si vous ne prévoyez pas d'installer d'autres instances sur la machine, gardez l'option « Par défaut » cochée. Il sera alors plus simple d'accéder à cette base depuis les autres programmes.

Si vous installez une nouvelle instance sur la machine, spécifiez alors un autre nom pour l'instance.

Supposons que c'est l'instance par défaut que nous installons et cliquons sur « suivant ».

2.1.6. Etape n°5 : Choix du type d'installation

Vous arrivez en principe sur ce menu :

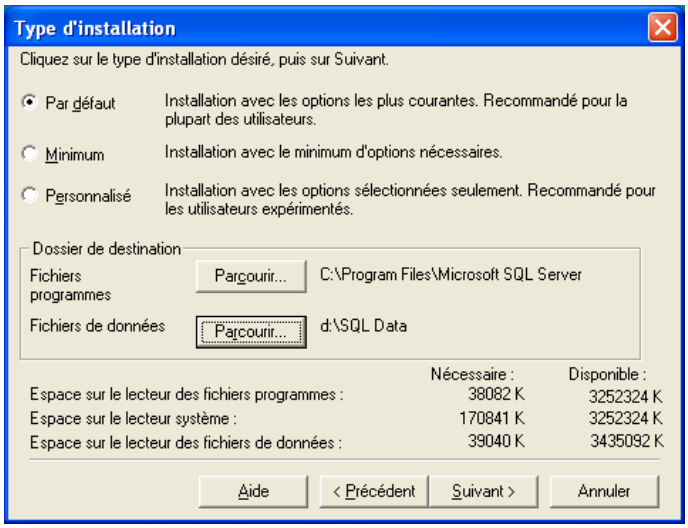


Figure 7 - Choix du type d'installation

Vous devez alors choisir le type d'installation

- Type : « Par défaut », « Minimum » ou « Personnalisé ».
  - L'option « Par défaut » convient pour la plupart des utilisateurs et comprend les outils clients et le serveur SQL. Utilise environ 250 Mo d'espace disque (hors données de vos bases de données)
  - L'option « Minimum » n'est à sélectionner que si vous manquez vraiment d'espace disque. Utilise environ 120 Mo d'espace disque.
  - L'option « Personnalisé » est utile si vous souhaitez installer des composants spécifiques qui ne figurent pas dans l'option par défaut ou si vous ne souhaitez installer uniquement ce qui vous semble nécessaire.
- Dossiers de destination
  - Le programme d'installation vous demande de choisir un dossier contenant les fichiers programmes et un autre contenant les fichiers de données. Ce dernier répertoire va contenir vos futures bases de données (vous pourrez toutefois choisir spécifiquement pour chaque base de donnée un emplacement particulier) et il est recommandé de choisir un autre lecteur pour ce dossier. Il sera alors plus facile d'un point de vue administratif de gérer le serveur. De plus, si le dossier est situé physiquement sur un autre disque les performances peuvent être accrues en distinguant le disque de données du disque d'application.
- Espace disque utilisé
  - Le programme récapitule l'utilisation disque que le serveur va prendre (hors fichiers de bases de données)

L'installation par défaut convient pour la plupart d'entre nous. Vous pouvez sélectionner « Personnalisé » et voir les différents composants pouvant être installés. Je ne les décrirai pas ici, reportez vous à l'aide pour plus de détails.
Cliquez sur suivant.

2.1.7. Etape n° 6 : Choix du compte de service

Si vous avez choisi d'installer le serveur avec les options par défaut vous devriez arriver à l'écran suivant (vous aurez d'abord choisi les options d'installation si vous avez cliqué sur « Personnalisé ») :

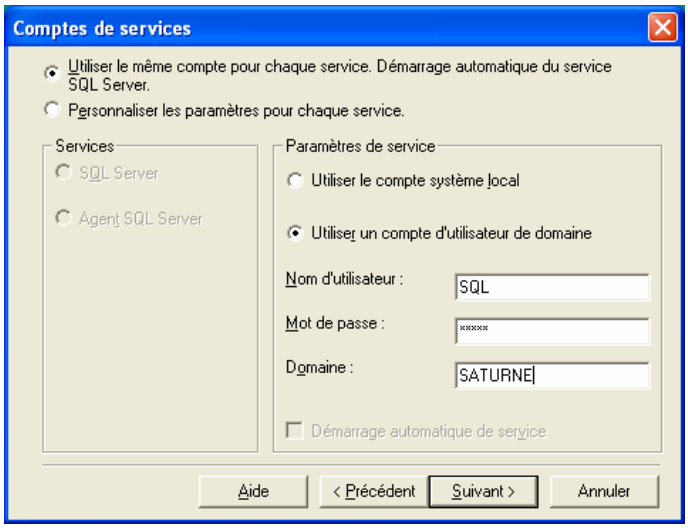


Figure 8 - Définition du compte sous lequel tourne SQL Server

Vous devez spécifiez le compte utilisé pour faire fonctionner le serveur SQL.

- Choisissez tout d'abord si vous utiliserez les mêmes informations pour chaque service ou si vous spécifierez séparément ces informations. A de rare exceptions près, vous pouvez utilisez le même compte pour tous les services.
- Choisissez alors un compte d'exécution de service. Le serveur SQL utilisera alors le compte spécifié pour s'exécuter. Il requit donc des droits administrateurs sur la machine, et même les droits d'administrateur du domaine si vous utilisez un cluster de serveurs.
    - « Utiliser le compte système local », cette option est intéressante si vous n'utilisez qu'un seul serveur SQL, ou si vous l'utilisez à des fins de tests. En effet, le compte système local n'a pas accès aux fonctionnalités réseaux de Windows 2000 et ne pourra donc pas communiquer avec d'éventuels autres serveurs SQL.
    - « Utiliser un compte d'utilisateur de domaine », spécifiez ici un utilisateur ayant les droits d'administrateur (ou administrateur du domaine si vous utilisez un cluster). Il peut être intéressant de créer un utilisateur spécifique ayant des droits limités (comme refuser l'ouverture de session par exemple) afin de pouvoir isoler le compte SQL et détecter plus facilement la source d'un éventuel piratage. Vous pouvez spécifier soit un utilisateur local, soit un utilisateur du domaine si vous souhaitez utiliser le même compte pour tous vos serveurs. Dans tous les cas, il faut que vous spécifiiez le mot de passe du compte.
- L'option « démarrage automatique du service » n'est disponible que si vous avez choisi de configurer séparément chaque service. Vous pourrez toujours modifier les options du serveur après l'installation pour changer ce paramètre.
    - Cochez la case pour que le service ce lance automatiquement lors du démarrage de la machine.

Validez en cliquant sur suivant.

2.1.8. Etape n°7 : Mode d'authentification

Il faut maintenant configurer l'authentification du serveur via cet écran :

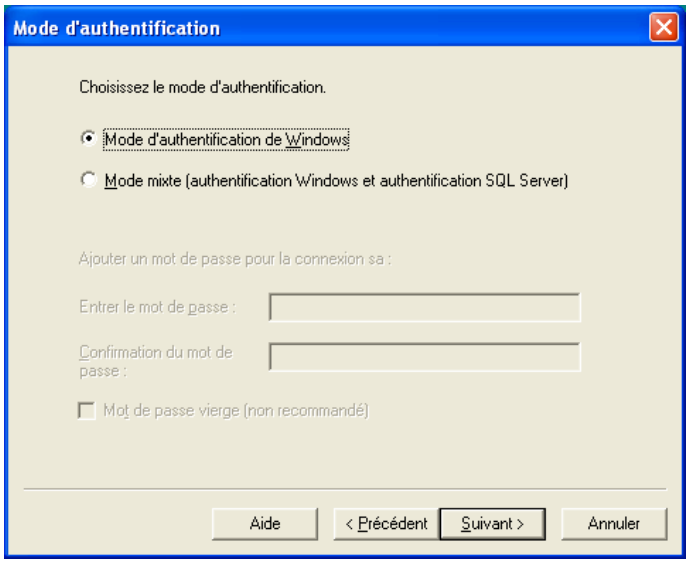


Figure 9 - Choix du type d'authentification utilisé

SQL Serveur propose deux modes d'authentification :

- Mode d'authentification de Windows
  - Ce mode d'authentification est très utile, surtout en entreprise. En effet, SQL Serveur utilise Windows pour authentifier les utilisateurs. Il fait ainsi appel à un contrôleur de domaine ou à la machine locale, et donc gère la sécurité de façon transparente et sûre. Par exemple, un utilisateur quelconque faisant parti du domaine se connectant à la base de données via un outil de gestion n'aura pas à rentrer le mot de passe. En effet le système aura identifié et authentifié l'utilisateur. L'authentification est sûre dans la mesure où tout se fait par Windows et est donc cryptée par le système d'exploitation. Le revers de la médaille est qu'il faut obligatoirement avoir ouvert une session sur le domaine pour avoir le droit d'accéder au serveur. Il y'a également un avantage administratif, puisqu'il suffit de choisir quels comptes du domaine ont accès au serveur.
  - Tous les administrateurs locaux sont donc administrateurs du serveur SQL (les administrateurs du domaine étant souvent inclus dans ce groupe local, ils le sont aussi en conséquence).
- Mode mixte (authentification Windows et authentification SQL Server)
  - Ce mode permet également de s'authentifier par le système d'exploitation, mais aussi en spécifiant un nom d'utilisateur et un mot de passe spécifique à SQL Server. Ces comptes ne peuvent être utilisés que depuis SQL Server et sont moins sûrs, à moins de crypter le canal de transmission. L'avantage de ce mode est de pouvoir accéder depuis n'importe quel ordinateur au serveur, en ayant un nom d'utilisateur et un mot de passe.
  - Le compte « sa » et un compte spécial. Il est un compte non Windows ayant totalement accès au serveur SQL (System Administrator). Il convient donc de lui donner un mot de passe long et difficile à trouver car c'est une faille potentielle de sécurité. Evitez absolument les mots de passe vide, sauf si vous utilisez le serveur à des fins de tests.

Nous choisissons de préférence le « mode d'authentification de Windows » qui est beaucoup plus simple à administrer et plus sécurisé, à moins d'avoir effectivement besoin du mode mixte.

Choisissez le mode qui vous semble le plus judicieux puis cliquez sur suivant.

2.1.9. Etape n°8 : Fin d'installation

Le programme d'installation copie alors les fichiers.

Le serveur est alors installé. Il est possible que l'on vous demande de relancer l'ordinateur pour finir l'installation si des fichiers du système d'exploitation ont été mis à jour.

Pensez également à mettre à jour le serveur via de services packs disponibles sur le site de Microsoft : http://www.microsoft.com/france/sql. A ce jour, deux services pack sont disponibles. L'installation de ces services pack est indispensable pour un serveur de production. Pour un serveur de développement (donc de tests), c'est un plus, mais ce n'est pas obligatoire.

Consulter le chapitre 4.1 pour des détails de fin d'installation.

3. Présentation des outils d'entreprise

3.1. Service Manager

Si vous avez installé Service Manager (inclus dans les options d'installation par défaut), une nouvelle icône sera apparue dans la barre des taches :

Figure 10 - Icône de Service Manager

C'est l'icône de Service Manager.

Vous pouvez lancer Service Manager depuis le menu démarrer (chemin par défaut) :

« Menu démarrer -> Programmes -> Microsoft SQL Server -> Service Manager »

En lançant Service Manager depuis le menu démarrer ou en double cliquant sur l'icône de notification, vous lancerez l'outil de gestion des services :

Figure 11 - Gestionaire de service

Le gestionnaire des services permet de configurer le démarrage des services.
Vous pouvez spécifier :

- Le serveur : entrez le nom de la machine à gérer (vous devez bien sûr être administrateur du serveur en question)
- Le service à gérer : généralement trois services sont installés avec SQL Serveur :
  - SQL Serveur : c'est le moteur SQL, le service principal. Les autres services sont dépendants de ce dernier donc si vous désactivez ce service, vous désactiverez les autres.
  - SQL Server Agent : c'est un service permettant l'automatisation des tâches courantes tels que la sauvegarde régulière, la publication d'une base de donnée dans le cadre de réplication, etc. Ce service n'est pas nécessaire pour l'exécution du serveur SQL mais offre une panoplie d'outils administratifs appréciables.
  - Distributed Transaction Coordinator : c'est un service permettant de gérer les transactions distribuées.

3.2. Enterprise Manager

Enterprise Manager est le couteau suisse de l'administrateur et du développeur SQL Server. En effet, depuis cet outil vous pouvez accéder à toutes les options et fonctionnalité du serveur et de toutes les bases (à condition d'en avoir le droit bien sûr !). Il se présente sous la forme d'une console enfichable MMC dont voici un aperçu :

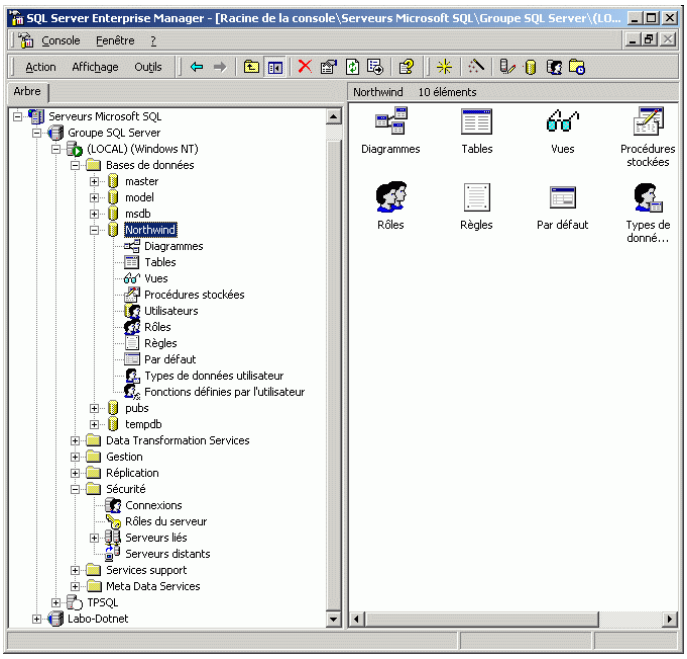


Figure 12 - Entreprise Manager

Comme vous pouvez le constater sur cette image, Enterprise Manager vous permet de manipuler depuis la même fenêtre plusieurs serveurs en même temps. Ces serveurs peuvent être regroupés en groupes ou sous groupes pour faciliter la gestion d'importants parcs de machines. Les fonctionnalités sont regroupées par nœud dans un sous-arbre unique à chaque instance de serveur :

- Bases de données : permet d'accéder aux bases de données, et à leur contenu
- Data Transformation Services : service de transfert et de transformation des données
- Gestion : utilitaire pour la gestion courante et la surveillance
- Réplication : permet de lier les serveurs entre eux pour répliquer les données, en tant que distributeur ou en tant qu'abonné
- Sécurité : gestion des accès au serveur
- Services Support : gestion des transactions distribuées et configuration de l'envoi de mails depuis SQL Server
- Meta Data Services : Permet d'utiliser un serveur de méta-données.

3.2.1. Ajouter un serveur à Enterprise Manager

Pour ajouter un serveur à administrer, rien de plus simple. Cliquez droit sur le groupe de serveur dans lequel vous voulez rajouter le serveur et faites « Enregister un nouveau serveur ». Vous pouvez également créer un nouveau groupe de serveurs en cliquant droit sur « Serveurs Microsoft SQL » ou sur un groupe déjà existant si vous souhaitez mettre le nouveau groupe dans un groupe existant.

Enregistrement d'un nouveau serveur :

La première page résume les différentes actions que vous devrez effectuer. On vous propose ensuite la liste des serveurs trouvés sur le réseau :

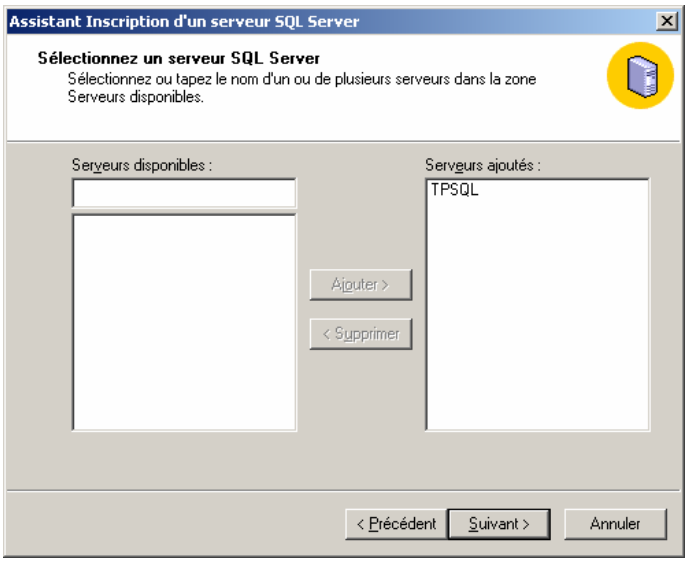


Figure 13 - Ajout de nouveau serveur dans l'Entreprise Manager

Choisissez ici le(s) les serveur(s) à rajouter, puis cliquez sur « Ajouter » pour les mettre dans la liste des serveurs à ajouter. Cliquez sur suivant.

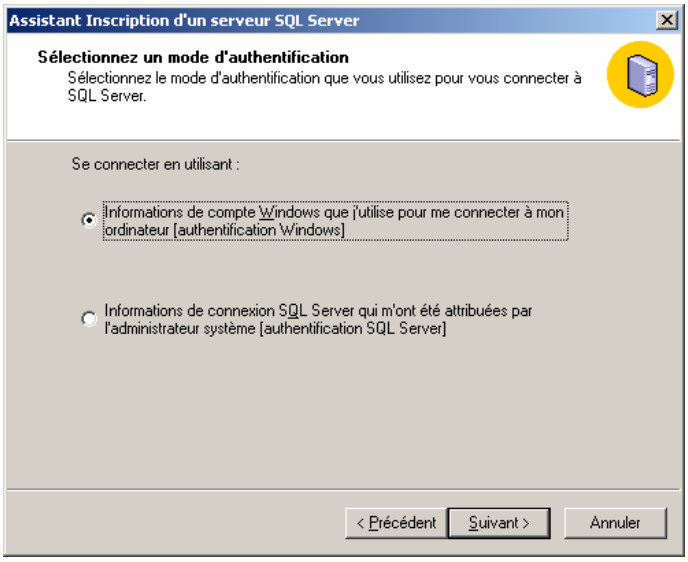


Figure 14 - Choix de l'authentification

Spécifiez ici si vous utiliserez votre compte Windows ou un compte SQL. Utilisez la première option de préférence, car vous n'aurez aucune information de login à fournir. Vous devez toutefois être sur le même domaine que le serveur (ou en local) pour bénéficier de cette option.

Si vous choisissez la deuxième option, il faudra spécifier un compte utilisateur SQL ayant le droit de se connecter, ou de demander à chaque connexion un login et un mot de passe :

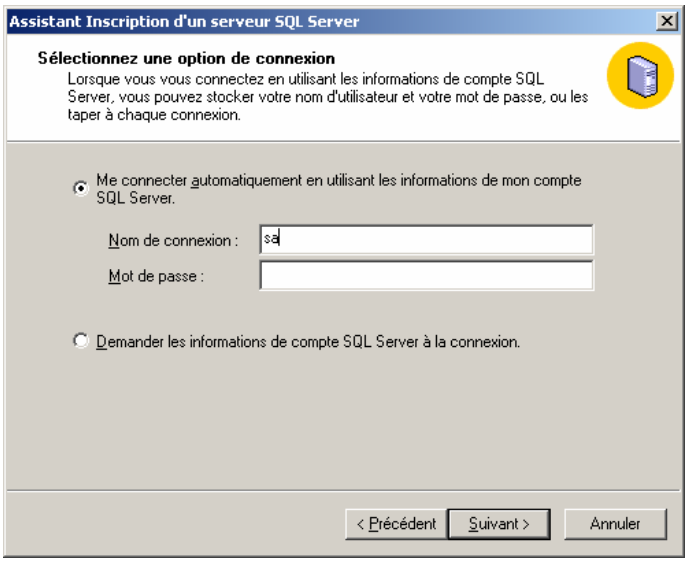


Figure 15 - Utilisation de l'authentification SQL Server

Dans tous les cas de figure, on vous demande dans quel groupe mettre ce serveur, soit un existant, soit on vous propose d'en créer un nouveau.

Après validation, une fenêtre vous indiquera le succès ou l'échec de l'enregistrement du serveur.

3.2.2. Configurer le serveur

Pour accéder aux options de configuration du serveur, faites un clic droit sur le nom du serveur puis choisissez options. Remarquez au passage le menu conceptuel du serveur dans lequel vous avez accès à certaines tâches courantes comme l'arrêt/démarrage du service, l'import/export de données, etc.
Les options se présentent sous la forme d'une fenêtre à plusieurs onglets :

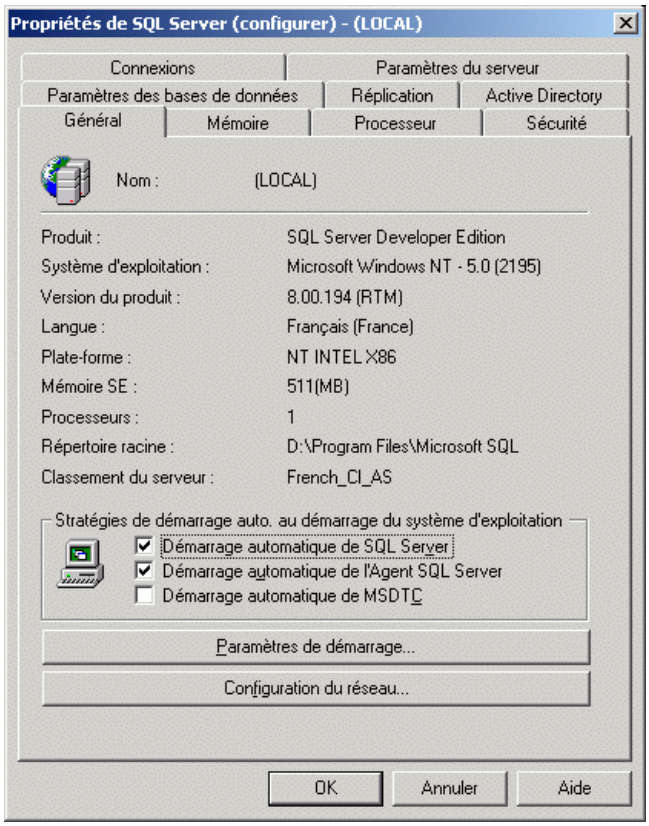


Figure 16 - Configuration de SQL Server

Reportez vous à la partie « Options de configuration du serveur » du chapitre « Tâches administratives courantes » pour plus de détails.

3.2.3. Créer une base de donnée

Pour créer un base de donnée, il suffit de cliquer droit sur le nœud « Bases de données » et de choisir « Nouvelle base de donnée ».

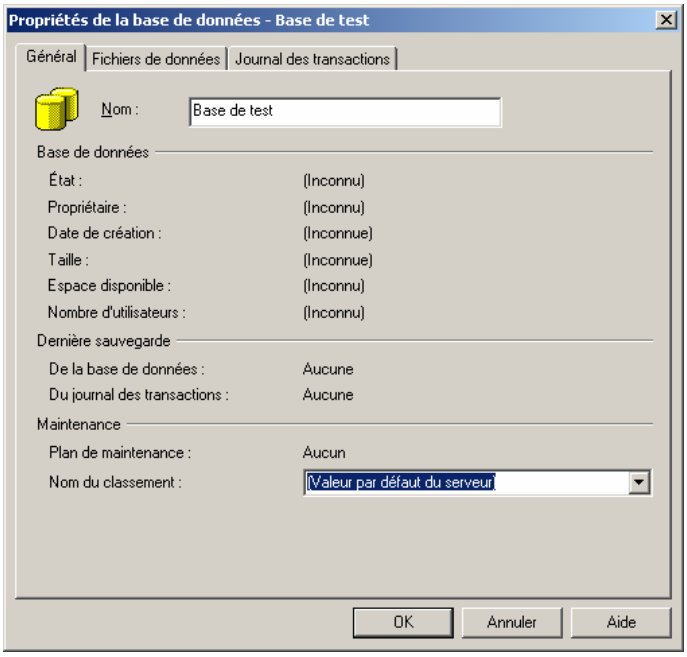


Figure 17 - Création d'une base de données

Vous devez alors spécifier un nom de base de donnée, ici « Base de test », un nom de classement (langue de la base de donnée, la valeur par défaut étant celle du serveur) et les fichiers de donnée et de transactions, ainsi que leurs propriétés :

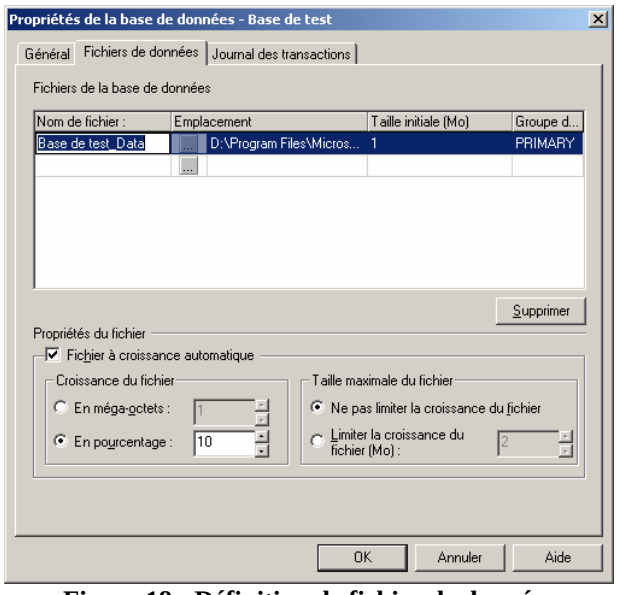


Figure 18 - Définition de fichier de données

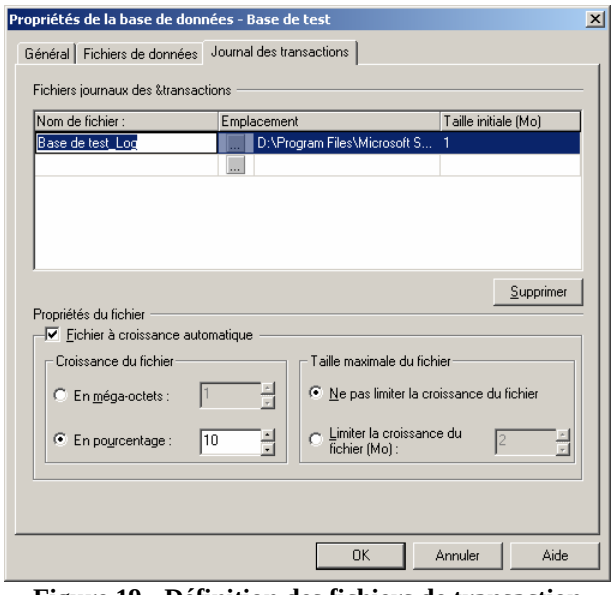


Figure 19 - Définition des fichiers de transaction

Pour les deux types de fichier, vous pouvez spécifier le comportement de l'expansion des fichiers. Vous pouvez en effet définir de quelle façon les fichiers de données vont augmenter de taille (pas à pas ou en pourcentage), une taille maximale et les fichiers dans lesquels les données vont être réparties. Plus de détails dans l'aide fournie avec SQL Server.

3.2.4. Créer ou modifier une table

Pour créer une table, cliquez droit sur le nœud « tables » de la base de données sur laquelle vous voulez créer la table.
Pour modifier une table existante, cliquez droit sur la table existante et faîtes « Modifier la table ».
Dans les deux cas, vous devez arriver à un écran comme celui-ci :

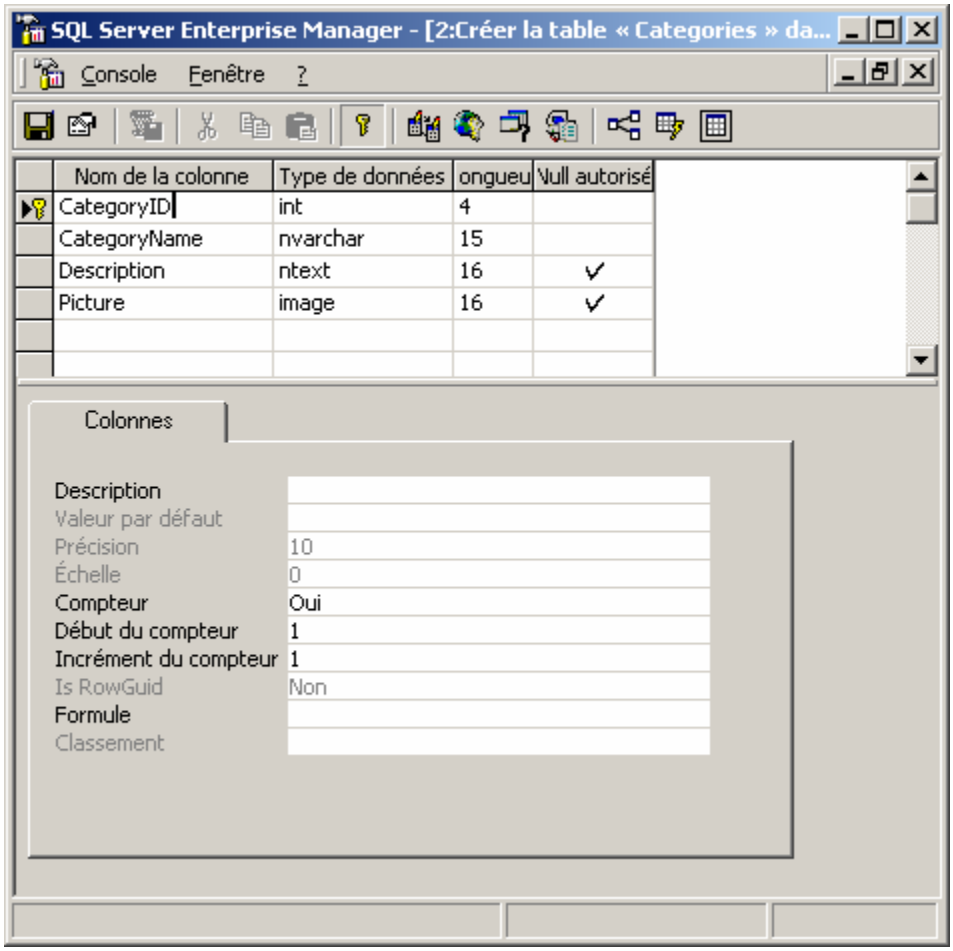


Figure 20 - Création d'une table

Depuis cette fenêtre, vous pouvez spécifier créer/modifier/supprimer une colonne à la table, en spécifiant le type de donnée, la longueur, le droit d'y mettre NULL, etc.
Vous avez également un certain nombre d'icônes pour gérer les clefs, les index, les triggers, etc.
Pour appliquer les modifications ou enregistrer la nouvelle table, cliquez sur la disquette et la table sera enregistrée. Si vous modifiez une table existante, il vaut veiller à l'intégrité des données et il se peut que vous ne soyez pas autorisé à modifier la table si des données existent déjà.

3.2.5. Modéliser la base de données.

Enterprise Manager propose un éditeur de diagramme dans lequel vous pouvez afficher les tables et leurs relations. Depuis ce diagramme, vous pouvez modifier le modèle de donnée directement :

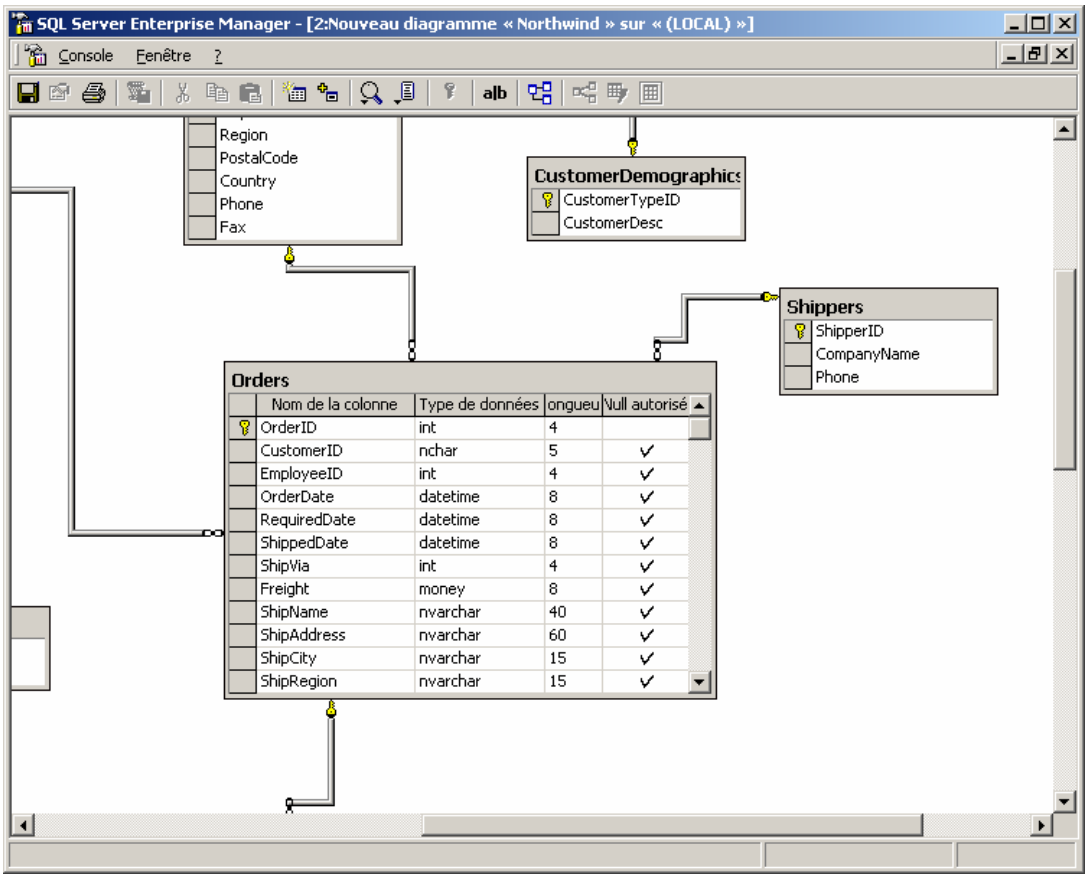


Figure 21 - Diagramme de la base de données

Toutefois, je recommande plutôt l'utilisation de Visio pour les personnes ayant ce logiciel. En effet Visio permet de modéliser les bases de données d'un point de vue détaché du serveur, et possède de nombreux outils tels que la génération de rapports que SQL Server ne propose pas.

3.2.6. Construction visuelle de vues

Enterprise Manager permet également de construire visuellement des vues.
Pour créer une nouvelle vue, cliquez sur le nœud « Vues » et choisissez « Nouvelle vue ». Pour modifier une vue existante, cliquez droit sur la vue en question et faites « Modifiez la vue ».

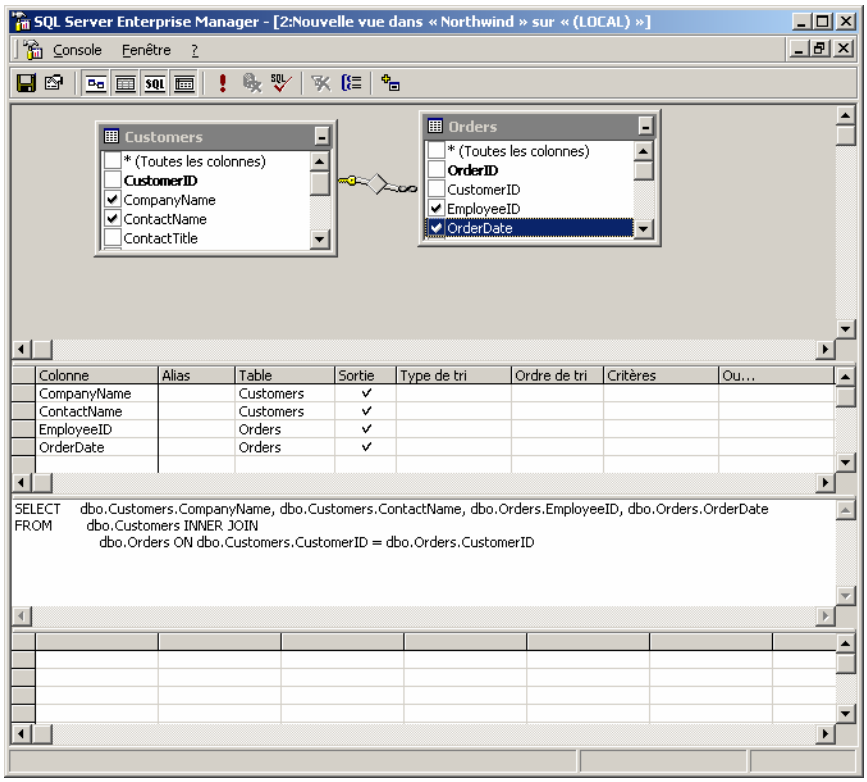


Figure 22 - Création d'une vue

Vous pouvez très facilement créer des vues depuis cet écran en spécifiant les tables à utiliser, les champs à afficher et les relations entre les tables.

3.2.7. Editeur de procédures stockées

Le développement de procédure stockée est facilité par Enterprise Manager. Vous pouvez soit créer une procédure stockée, soit en modifier une existante depuis le nœud « Procédures stockées » :

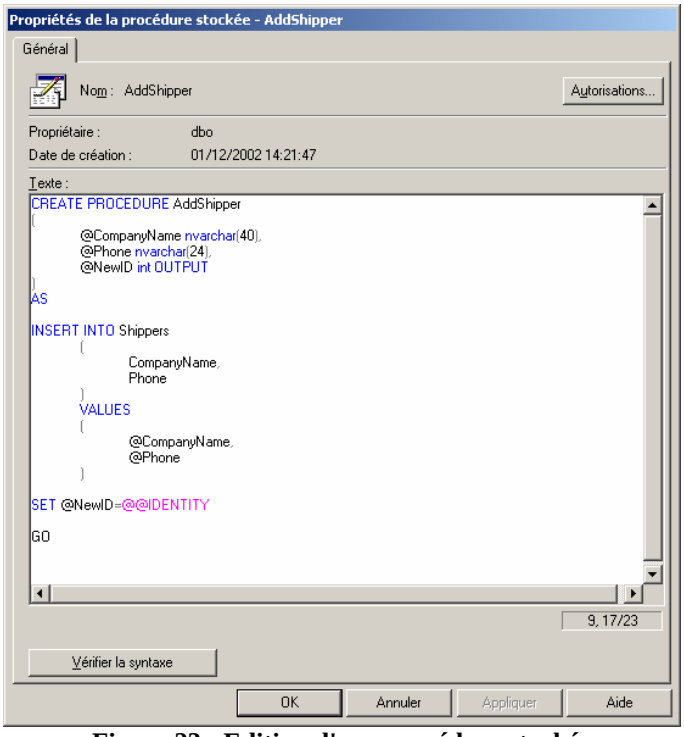


Figure 23 - Edition d'une procédure stockée

Nous parlerons des procédures stockées dans la présentation du Transact SQL…

3.2.8. Conclusion

Enterprise Manager est le plus complet des outils permettant d'administrer SQL Serveur. Vous pouvez également visualiser/modifier/supprimer des données, mais il est souvent plus intéressant de passer par une application pour cela. Pour les environnements où vous n'avez pas Enterprise Manager, sachez que toutes les commandes sont disponibles en T-SQL pour configurer et administrer le serveur.

3.3. Analyseur de requêtes

L'analyseur de requêtes est un client SQL Server destiné à l'exécution de requêtes. De plus, il vous permet d'afficher le plan d'exécution, d'afficher des informations de traçage et d'afficher des statistiques. Vous pouvez ainsi optimiser les requêtes en identifiant les goulots d'étranglement.
Tout comme Enterprise Manager, vous pouvez soit spécifier un compte SQL, soit utiliser le compte Windows.

Ouverture de l'analyseur de requête :

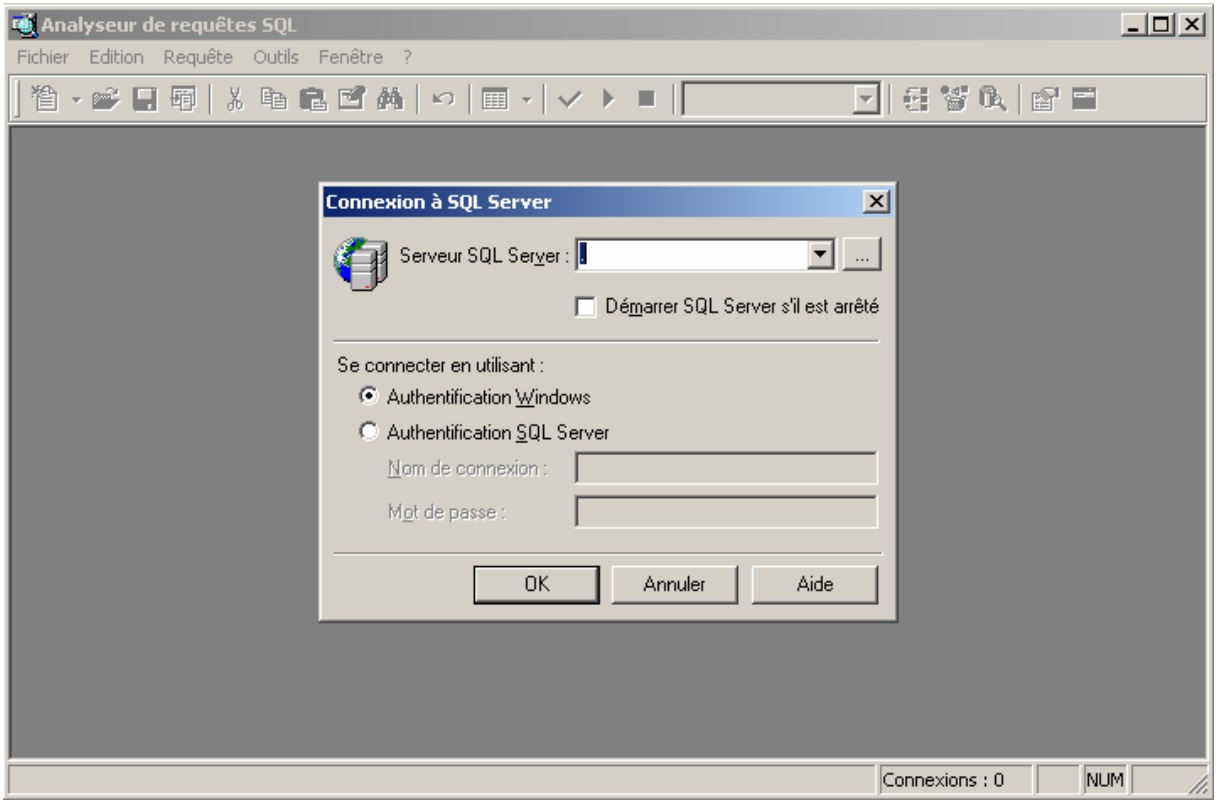


Figure 24 - Ouverture de l'analyseur de requête

Depuis cette fenêtre vous devez spécifier des informations de connexion valides, qu'elles soient du type authentification Windows (c'est donc l'utilisateur en cours sur la machine qui va transmettre sont authentification) ou du type SQL Server (des comptes spécifiques à SQL). Un bouton parcourir vous montrera la liste des serveurs disponibles sur le réseau, le « . » étant le serveur local. Attention toutefois, un administrateur peut cacher pour des raisons de sécurité la découverte de son serveur par ce biais.

Une fois connecté, vous avez à votre disposition une fenêtre dans laquelle vous pourrez entrez vos requêtes :

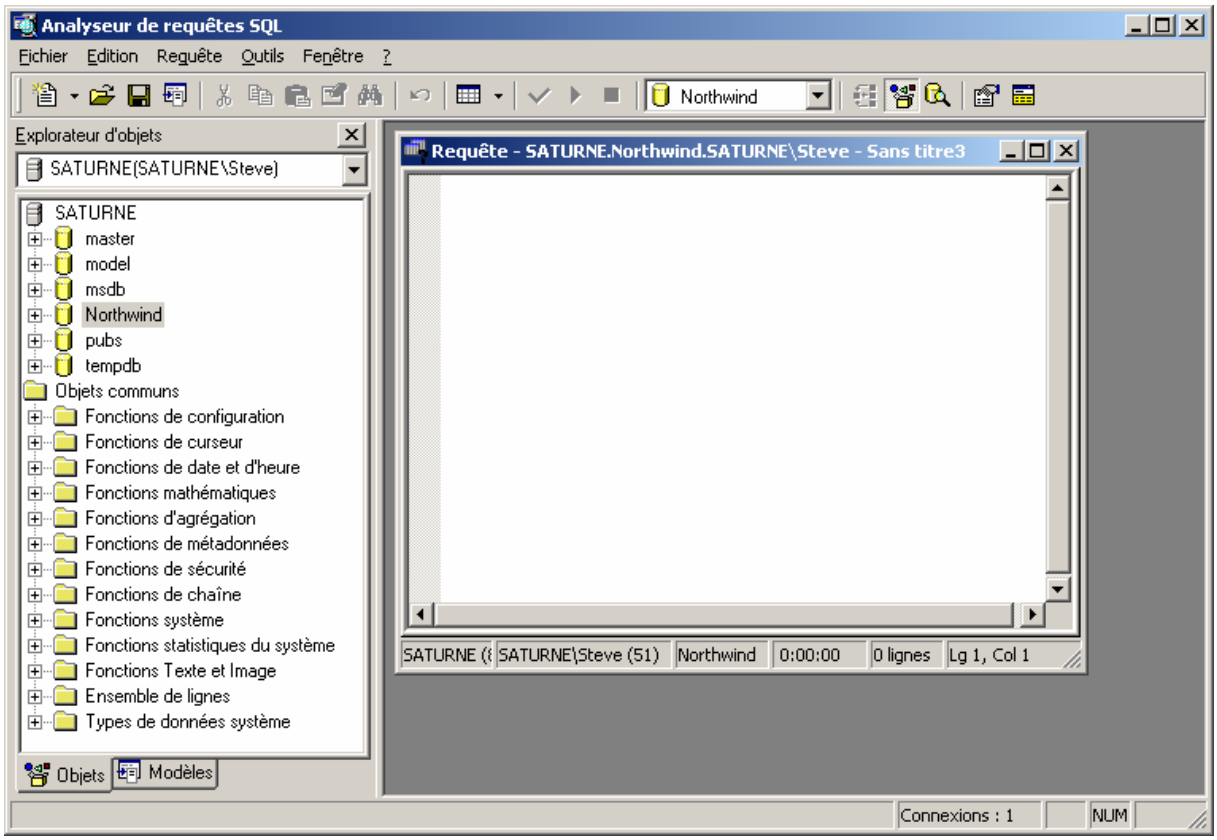


Figure 25 - SQL Query Analyser

Cet outil étant MDI (Multiple Documents Interface – Interface à documents multiples), vous pouvez ouvrir plusieurs de ces fenêtres depuis le menu fichier. « Nouveau » pour ouvrir une nouvelle fenêtre utilisant la même connexion, « Connecter » pour ouvrir une nouvelle connexion.

Remarquez le volet gauche qui vous permet d'explorer le serveur, avec, en premier les bases de données, puis les objets communs (fonctions, types de données, etc.) et des modèles dans un second onglet du volet qui vous aidera à la rédaction de requêtes courantes.
Notez également le menu déroulant de la barre d'outil dans lequel il est affiché la base de donnée en cours pour la fenêtre ayant le focus.

Entrez donc une requête quelconque dans la fenêtre et cliquez soit sur la flèche bleu de la barre d'outils, soit appuyez sur F5 pour exécuter la requête.
La requête que j'emploie ici est :

```
SELECT * FROM Products
```

Vous pouvez afficher les résultats de deux manières. Cliquez sur le bouton déroulant pour choisir entre le mode texte ou le mode grille, ou encore dans un fichier :

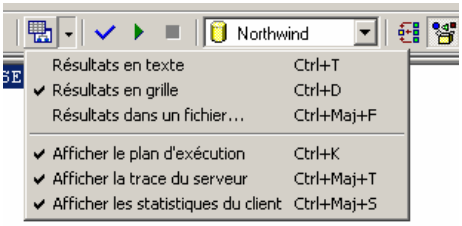


Figure 26 - Définition du type d'affichage

Le plan d'exécution vous permet d'afficher quelles parties des requêtes sont les plus lourdes. Sélectionnez de l'afficher depuis ce même menu et exécutez la requête.
Un nouvel onglet sera apparu :

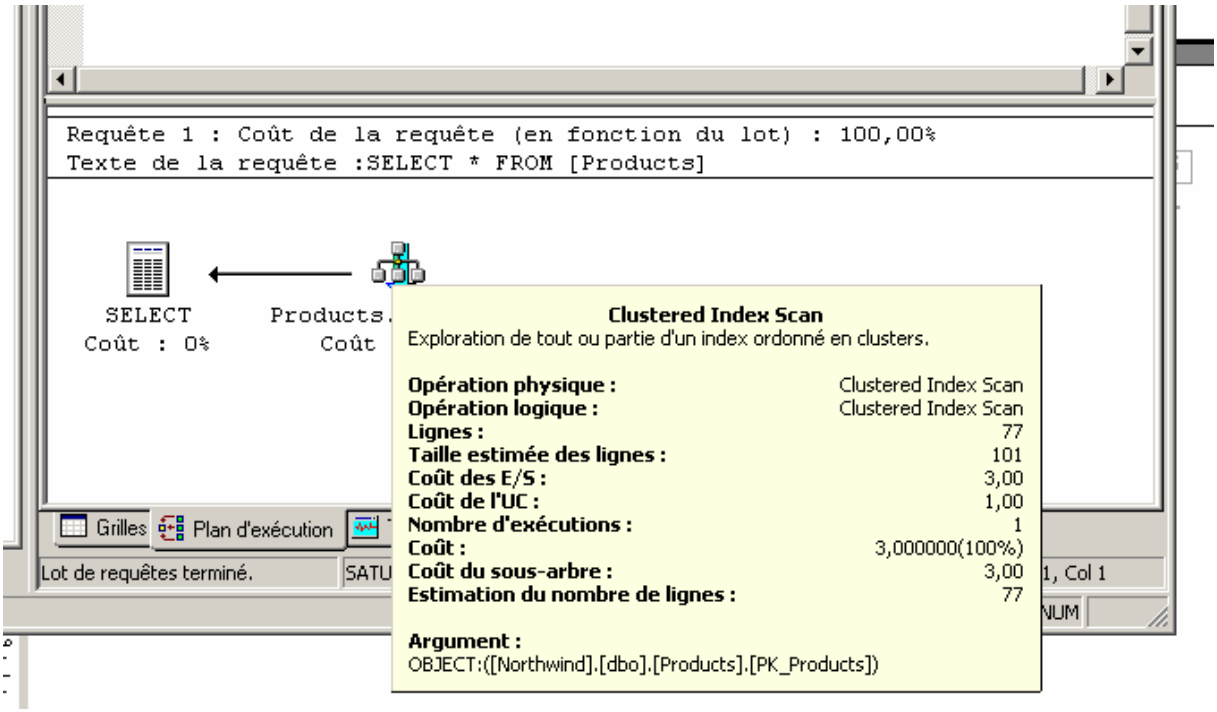


Figure 27 - Affichage du plan d'exécution

Vous pouvez depuis cet onglet afficher le coût de chaque étape de la requête. Ici, il n'y a q'une étape donc l'intérêt est limité, mais pour les requêtes importantes, cela permet d'isoler les charges lourdes.

De plus, depuis l'analyseur de requêtes, vous pouvez gérer les index de la table en cours depuis le menu « Fichier -> Gérer les index ».

4. Tâches administratives courantes

4.1. Tâches de post-installations

4.1.1. Lancement du serveur

Si vous venez d'installer le serveur SQL et que vous n'avez pas redémarré, il faudra lancer manuellement le serveur. Si vous avez redémarré et à moins que vous ayez spécifié le contraire lors de l'installation, le serveur sera lancé.

Si ce n'est pas le cas, vous avez plusieurs possibilités pour lancer le serveur :

- Utilisez le Gestionnaire de service (Service Manager). Pour plus de détails, référez vous au chapitre 3.1 concernant Service Manager.
- Utilisez le composant enfichable « Services » depuis une console MMC et démarrez le service MSSQLSERVER

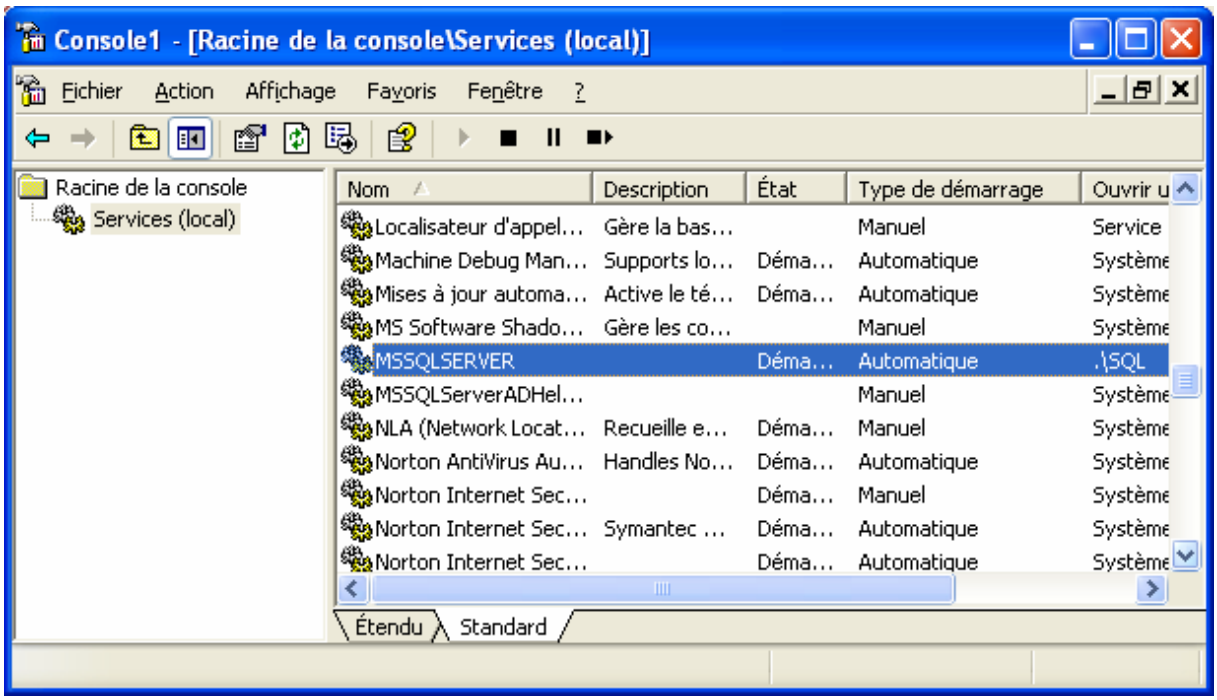


Figure 28 - Console MMC des services Windows

- Utilisez Enterprise Manager (qui est en fait également un composant enfichable dans une console MMC). Enterprise Manager est l'outil idéal pour l'administrateur du serveur SQL. Je présente plus en détail l'outil au chapitre 3.2.

4.1.2. Configurations diverses

Utilisez Enterprise Manger pour configurer les options de votre serveur. Vous pouvez également utiliser des commandes Transact SQL pour modifier la plupart des options. Il est toutefois plus aisé d'utiliser Enterprise Manager. Reportez vous à la documentation si vous souhaitez tout de même utiliser le Transact SQL.

Lancez Enterprise Manager. Selon que vous souhaitiez administrer un serveur installé localement ou un serveur installé sur une autre machine, vous devrez ou non enregistrer un serveur. Reportez vous la présentation de Enterprise manager pour plus de détails.

SQL Server installé localement

Si vous avez installé localement SQL Server, vous aurez votre instance du serveur déjà enregistrée. Vous pouvez donc cliquer avec le bouton droit sur l'instance et sélectionner « propriétés » pour accéder aux options du serveur.

SQL Server installé sur une autre machine

Si vous avez installé SQL Serveur sur une autre machine, vous devrez renseigner le serveur. Pour se faire cliquez droit sur « Groupe SQL Server » et faite « Enregistrez un nouveau serveur SQL ». Suivez alors les indications de l'assistant. Pour plus d'information, reportez vous à la présentation de Enterprise Manager.

Une fois le serveur enregistré, vous pouvez cliquer avec le bouton droit sur l'instance et sélectionner « propriétés » pour accéder aux options du serveur.

Options de configuration du serveur

Les différentes options de configuration du serveur apparaissent sous forme d'onglets :

Figure 29 - Configuration du Serveur SQL

Les différents onglets regroupent les options par catégories. Pour des détails spécifiques à chaque options, reportez vous à la documentation, mais voici un bref descriptif de chaque onglets :

- Général :
  - Cet onglet vous donne des informations systèmes à propos du serveur SQL.

    - Vous pouvez également configurer les paramètres de démarrage des services ainsi que la ligne de commande de démarrage, ainsi que les différents moyens de connexion au serveur.
- Mémoire :
    - Vous pouvez configurer ici l'occupation mémoire du serveur.
    - Par exemple, vous pouvez allouer une certaine quantité de mémoire vive réservée au serveur, et indiquer des paramètres spécifiques de gestion mémoire pour optimiser les performances du serveur (au détriment des autres applications souvent)
- Processeur :
    - Pour les ordinateurs ayant plusieurs processeurs, vous pouvez dédier au serveur SQL un ou plusieurs processeurs, et configurer diverses options de priorité du serveur SQL.
- Sécurité :
    - Précisez dans cet onglet le mode d'authentification et le compte de service de SQL Server.
- Connexions :
    - Cet onglet permet de configurer les connections au serveur.
- Paramètres du serveur :
    - Précisez depuis cet onglet la langue par défaut du serveur, certains paramètres de sécurité ainsi que l'utilisation de mails depuis le serveur.
- Paramètre de base de données :
    - C'est ici que vous spécifiez des options de maintenance et d'administration, telles que les répertoires de création de base de donnée, des options de sauvegarde, etc.
- Réplication :
    - Configurer ici si le serveur est capable de répliquer une base de données.
- Active Directory :
    - Vous pouvez enregistrer votre serveur dans une base Active Directory.

4.1.3. Configurer la prise en charge de SQL XML dans IIS.

Comme nous l'avons vu dans la présentation de SQL Server, il est possible de récupérer des données directement au format XML depuis SQL Serveur. Pour cela, il faut que IIS soit installé.

Lancement de l'assistant

Lancer l'assistant depuis le menu démarrer en ouvrant :

« Menu démarrer -> Microsoft SQL Server -> Configurer la prise en charge de SQL XML dans IIS »

Une console MMC va alors s'ouvrir avec le composant enfichable « IIS Virtual Directory Management for SQL Server »

Créer un répertoire virtuel

Explorer l'arborescence jusqu'au site Web accueillant la génération de flux XML depuis SQL Server. Cliquez droit sur le site Web et faites « Nouveau -> Répertoire virtuel ».

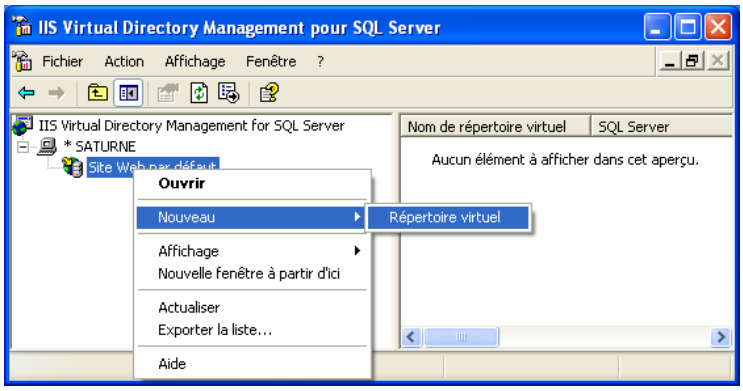


Figure 30 - Création d'un répertoire virtuel IIS

Vous devriez alors arriver à l'écran de configuration de ce répertoire virtuel :

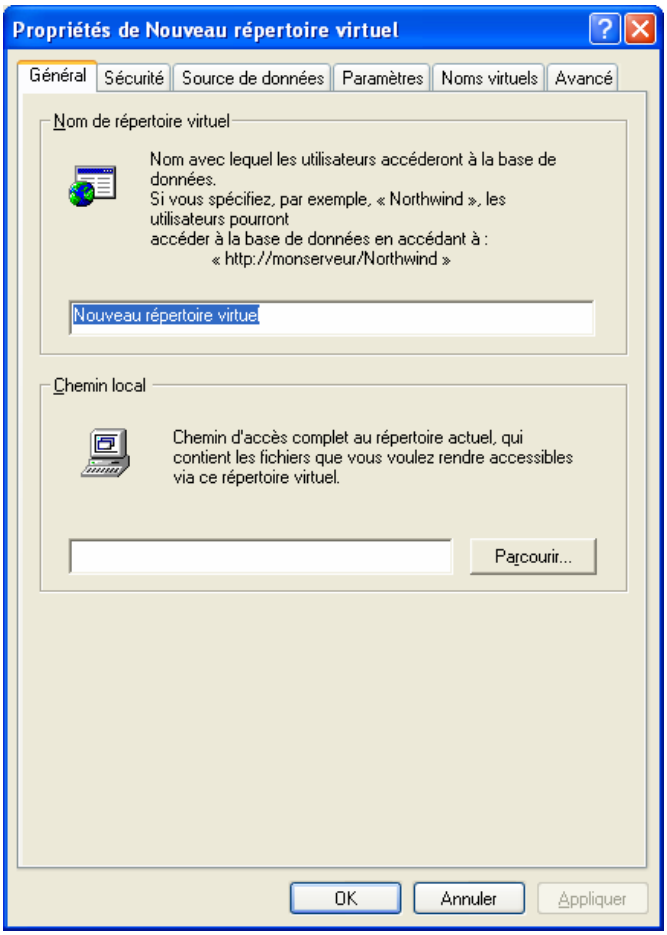


Figure 31 - Propirétés du repertoire virtuel

Onglet « Général »

Spécifiez le nom du répertoire virtuel. Pour cet exemple, nous utiliserons la base de donnée « Northwind » qui est une base de donnée exemple installée avec SQL Server.

Mettons alors, comme le suggère l'assistant :

http://monserveur/Northwind où « nomserveur » est le nom de la machine.

Spécifier également un chemin ou stocker ces fichiers. Faites attention aux droits des répertoires.

Pour notre exemple, mettons « D:\SQLXML\Northwind » comme chemin des fichiers.

Onglet « Sécurité »

Spécifiez dans cet onglet des informations de connexion valides. Pour notre exemple, choisissons la deuxième option « utiliser l'authentification intégrée de Windows ». Vous aurez tout loisir d'affiner ses options lorsque vous utiliserez vos propres bases de données.

Onglet « Source de données »

Cet onglet permet de choisir le serveur SQL. Si le serveur Web est situé sur une autre machine vous pourrez alors choisir le bon serveur SQL.

Choisissez également la bonne base de données. Pour notre exemple, choisissons Northwind.

Onglet « Paramètres »

Spécifiez depuis cet onglet comment les utilisateurs peuvent accéder au contenu. Pour l'exemple, cochez tout. Vous devrez ensuite spécifier uniquement les options dont vous aurez besoin pour éviter des commandes 'dangereuses'.

Onglet « Noms virtuels »

Configurez depuis cet onglet des schémas de requêtes. Cette notion étant assez complexe, je vous conseille d'étudier la documentation de SQL Server avant de configurer cet onglet.

Pour l'exemple, cliquez sur « Nouveau ». Dans la boite dialogue alors ouverte, choisissez un nom virtuel. Prenons « templates » (ou ce que vous voulez) pour l'exemple. Sélectionnez « template » comme type de nom virtuel et choisissez le chemin d'accès : « D:\SQLXML\Northwind\templates » (qui doit exister !!!).

Procédez de même pour créer un nom virtuel pour les schémas : cliquez sur « Nouveau », puis mettez « schemas » (ou ce que vous voulez) dans le nom virtuel, « schema » pour le type et « D:\SQLXML\Northwind\schemas » pour le chemin d'accès, qui, bien sûr, doit exister.

Enfin, cliquez à nouveau sur « Nouveau » puis mettez « dbobjects » dans le nom virtuel (ou ce que vous voulez) et en type choisissez « dbobject ». Vous n'avez pas besoin ici de spécifier un chemin.

Onglet « Avancé »

Cet onglet contient d'autres paramètres spécifiques que nous laissons tels quels.

4.1.4. Accéder aux données

Pour tester le répertoire virtuel crée, tapez la chaîne suivante dans le navigateur : http://monserveur/northwind?sql=SELECT * FROM Employees FOR XML AUTO&root=root et appuyez sur ENTRÉE.

Si vous ne voyez rien, affichez la source de la page. Vous verrez alors le résultat des données au format XML.

Reportez vous à la documentation pour personnaliser le flux XML de sortie. Il suffit alors près depuis n'importe quelle application capable de récupérer des données au format XML pour après les traiter.

4.1.5. Utilisation d'un modèle (template)

Il est beaucoup plus intéressant d'utiliser un template que de spécifier la requête directement dans l'url.

Créez ce fichier XML dans votre répertoire templates (d:\SQLXML\Northwind\templates) :

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql" >
<sql:query>
SELECT * from [Products by Category] FOR XML AUTO
</sql:query>
</ROOT>
```

Enregistrez le sous le nom : « products.xml ».
On déclare ici un modèle (ou template) SQL. Ce premier exemple consiste à exécuter une requête simple pour afficher tous les éléments d'une vue appelée « Product by catégory ». Notez bien le FOR XML AUTO qui permet de générer le flux au format XML.
Notez également que tout le modèle est inclut dans un élément ROOT, mais vous pouvez l'appeler comme vous le souhaitez, ROOT étant juste plus explicite.

Ouvrez ensuite l'url : http://nomserveur/northwind/templates/products.xml pour voir le résultat :

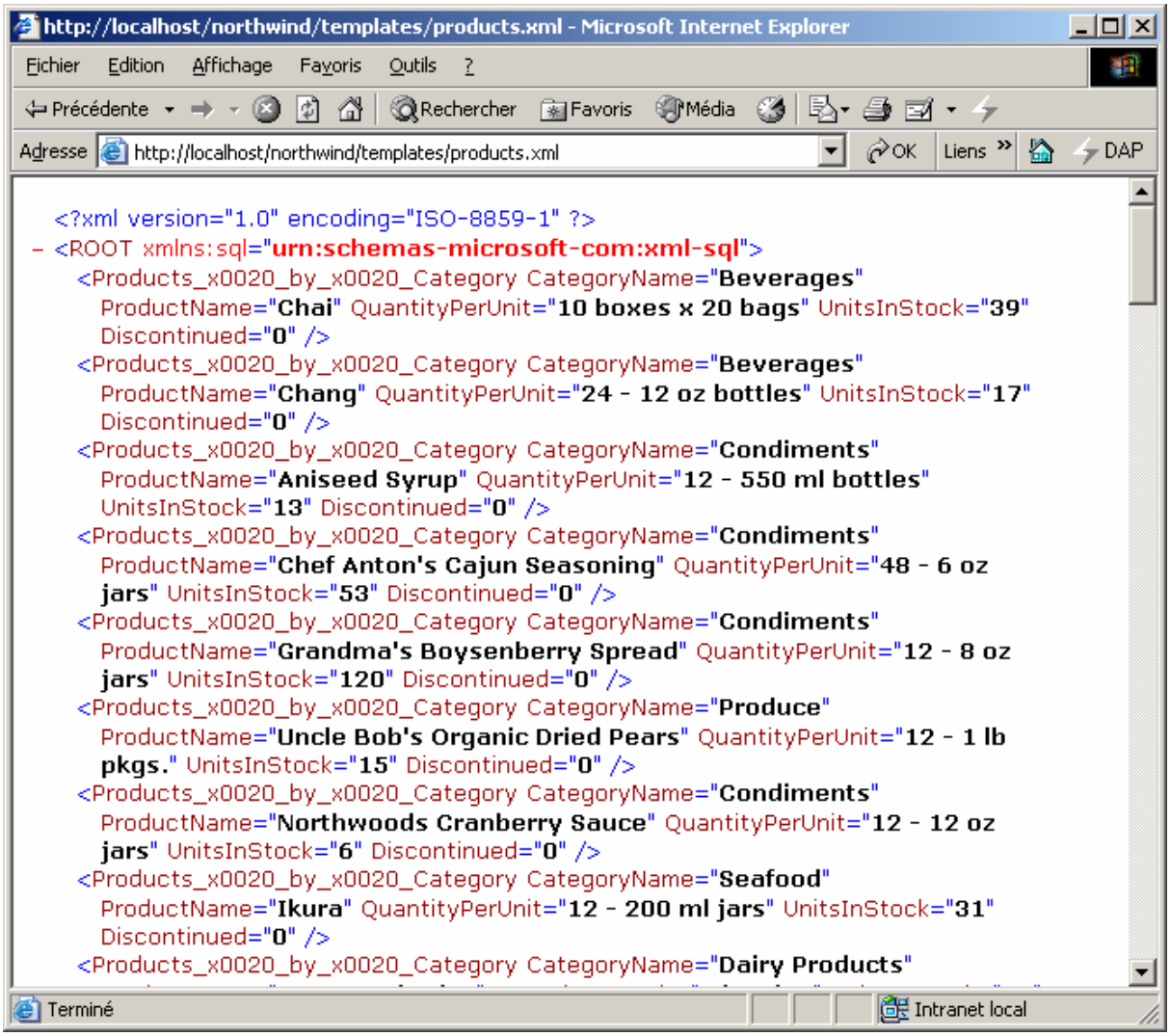


Figure 32 - Résultat d'affichage d'un requête SQLXML

Comme vous pouvez le voir, vous avez récupéré le résultat sous forme XML. Vous pouvez déjà imaginer l'avantage d'une telle technique. Par exemple, on utilisant le support de XML, n'importe quelle application capable d'effectuer une requête http de récupérer des données. Pas besoin de créer une connexion avec le serveur SQL ! De plus, XML étant standard, on pourra accéder à ces données depuis n'importe quelle plateforme possédant un parseur XML.

4.1.6. Encore plus fort, l'utilisation de feuilles de style

Le flux étant au format XML, rien ne nous empêche d'y associer une feuille de style.
La preuve par l'exemple :

Utilisons cette fois ci ce template :

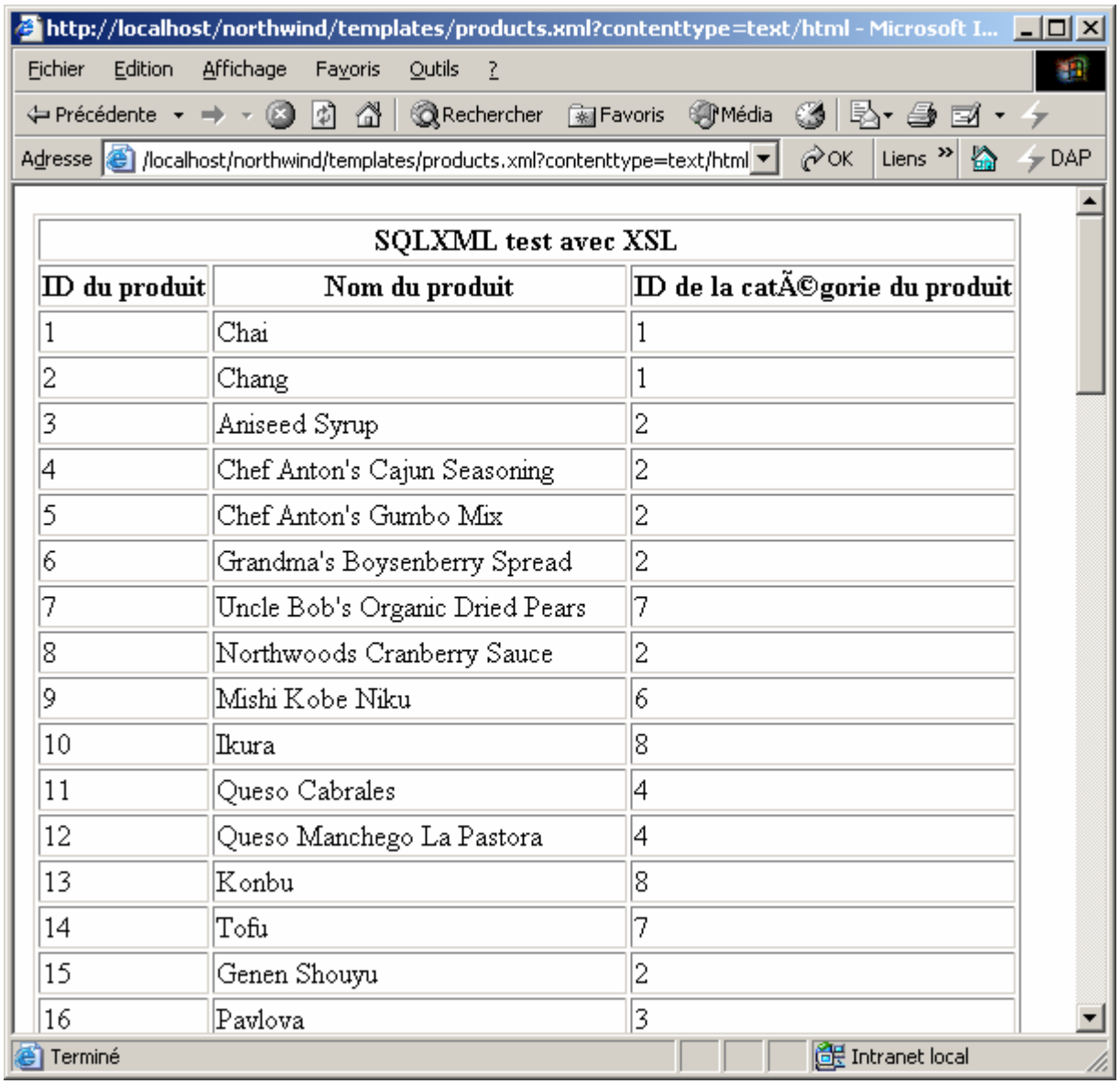
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<ROOT xmlns:sql="urn:schemas-microsoft-com:xml-sql" sql:xsl='products.xsl'>
      <sql:query>
SELECT ProductID, ProductName, CategoryID FROM Products FOR XML AUTO
      </sql:query>
</ROOT>
```

Puis copiez cette feuille de style dans le même répertoire en l'appelant « products.xsl » :

```
<?xml version='1.0' encoding='UTF-8' ?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      version="1.0">
      <xsl:template match='Products'>
            <tr>
                  <td>
                        <xsl:value-of select='@ProductID' />
                  </td>
                  <td>
                        <xsl:value-of select='@ProductName' />
                  </td>
                  <td>
                        <xsl:value-of select='@CategoryID' />
                  </td>
            </tr>
      </xsl:template>
      <xsl:template match='/'>
            <html>
                  <body>
                        <table border='1'>
                              <tr>
                                    <th colspan='3'>SQLXML test avec
XSL</th>
                              </tr>
                              <tr>
                                    <th>ID du produit</th>
                                    <th>Nom du produit</th>
                                    <th>ID de la catégorie du produit</th>
                              </tr>
                              <xsl:apply-templates select='ROOT' />
                        </table>
                  </body>
            </html>
      </xsl:template>
</xsl:stylesheet>
```

Et observez le résultat en ouvrant l'url suivante :
http://monserveur/northwind/templates/products.xml?contenttype=text/xml.
N'oubliez pas de préfixer les colonnes par « @ » car le flux généré les transforme en attributs.

Vous devriez arriver à un résultat comme celui-ci :



SQLXML test avec XSL		
ID du produit	**Nom du produit**	**ID de la catÃ©gorie du produit**
1	Chai	1
2	Chang	1
3	Aniseed Syrup	2
4	Chef Anton's Cajun Seasoning	2
5	Chef Anton's Gumbo Mix	2
6	Grandma's Boysenberry Spread	2
7	Uncle Bob's Organic Dried Pears	7
8	Northwoods Cranberry Sauce	2
9	Mishi Kobe Niku	6
10	Ikura	8
11	Queso Cabrales	4
12	Queso Manchego La Pastora	4
13	Konbu	8
14	Tofu	7
15	Genen Shouyu	2
16	Pavlova	3

Figure 33 - Résultat d'affichage d'un requête SQLXML associée à une feuille de style

Vous pouvez voir alors que la feuille de style a transformé le flux XML en HTML. Le « ?contenttype=text/html » indique au navigateur que le résultat est au format HTML, et non XML comme l'extension le laisse supposer.
Les possibilités de SQL XML sont immenses, on peut passer des paramètres par l'url, utiliser XPath pour parcourir un document et même y mettre des scripts clients. Il est donc possible de réaliser des applications Web complètes sans avoir une seule ligne de code « standard » (C#, VB, PHP ou autre) à taper.

Pour aller plus loin, je vous invite à consulter la documentation fournie avec SQL Server.

5. Introduction au Transact SQL

5.1. Qu'est ce que le T-SQL (Transact SQL)

Le Transact SQL, ou T-SQL est un langage dédié à l'accès aux données. Il est une extension de SQL et propose de nombreuses améliorations, comparé à ce dernier :

- Structures de contrôle (if/then/switch/…)
- Déclaration de variables
- Curseurs (pointeurs de données)
- Et bien d'autres encore

Ce langage est une extension de la norme SQL-92.

5.2. « Northwind »

Pour les exemples qui vont suivre, j'utiliserai l'analyseur de requêtes, mais vous pouvez très bien utiliser n'importe quel logiciel permettant l'exécution de requêtes. Nous utiliserons la base de donnée « Northwind » qui est livré en exemple avec SQL Server. Voici les tables de cette base :

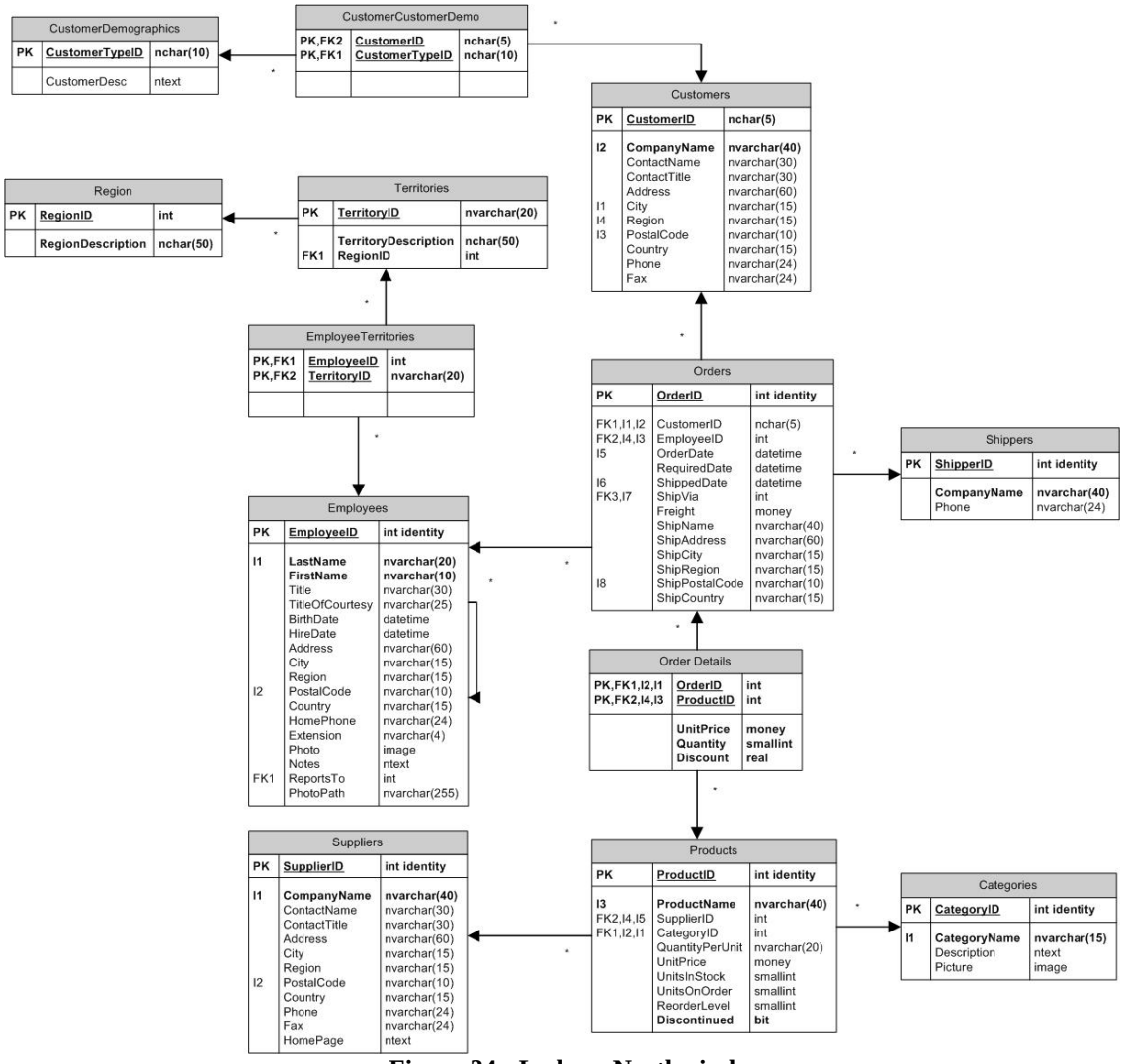


Figure 34 - La base Northwind

Ainsi que quelques vues qui ont été définies :

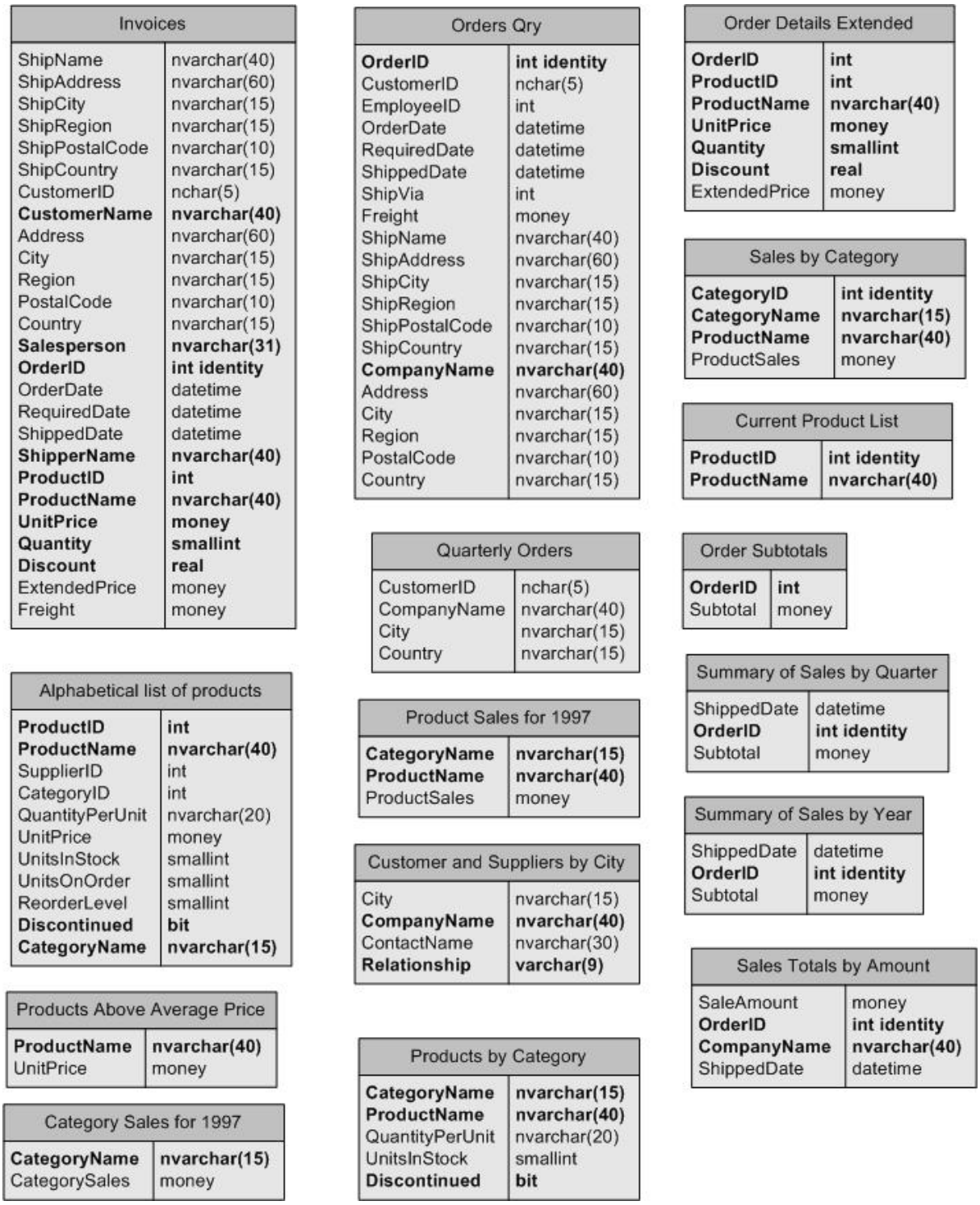


Figure 35 - Vues de la base Northwind

5.3. Commentaires

Il existe deux façons de mettre des commentaires dans une requête T - SQL :

```
-- Commentaire
```

Permet de mettre en commentaire tout ce qui suit sur la même ligne.

```
/* Commentaire */
```

Permet de mettre en commentaire tout ce qui est inclut entre les /* */

Exemple :

```
/*
      Exemple de commentaire
*/

-- Ceci est un commentaire d'une ligne
DECLARE @ProductCount int

/* On met un commentaire
sur plusieurs
lignes */

SET @ProductCount = (SELECT COUNT(ProductID)
FROM Products)

SELECT @ProductCount -- On met un commentaire à la fin de la ligne
```

Nous verrons plus loin à quoi correspondent les différentes instructions, cet exemple permettant juste de montrer la syntaxe des commentaires.

5.4. Variables

5.4.1. Déclaration

Sans les variables, le T-SQL ne serait qu'une implémentation du SQL standard. Vous pouvez déclarer des variables de n'importe quel type SQL Server.

Exemple :

```
DECLARE @MyInteger int
```

Que signifie cette ligne ?

Simplement, nous avons déclaré une variable locale du nom de @MyInteger et de type int. On dit qu'une variable est locale (marquée par le « @ ») car seule cette requête pourra accéder à cette dernière.

5.4.2. Types de données

Il existe un certain nombre de types de données pour le T-SQL :

- Types numériques exacts (c'est-à-dire sans perte d'information) :
  - entiers :
    - bigint : entier 64 bits signé
    - int : entier 32 bits signé
    - smallint : entier 16 bits signé
    - tinyint : entier 8 bits non signé (équivalent à un octet)
  - binaire :
    - bit : 0 ou 1 (attention, ce n'est pas true/false, mais bien les entiers 0 et 1)
  - décimaux et numériques :
    - decimal : nombre à précision fixe
    - numeric : équivalent au type decimal
  - monétaires :
    - money : représente une somme (4 chiffres après la virgule)
    - smallmoney : équivalent de money avec une précision moindre
- Types approchés (il peut résulter une perte de données du à l'imprécision de ces types)
  - numériques approchés:
    - float : nombre à virgule flottante
    - real : nombre à virgule flottante de grande précision
  - date :
    - datetime : date et heure allant du 1er janvier 1753 au 31 décembre 9999 avec une précision de 3.33 millisecondes.
    - smalldatetime : date et heure allant du 1er janvier 1900 au 6 juin 2079 avec une précision d'une minute.
  - chaînes de caractères :
    - char : chaîne de caractère fixe (8000 caractères maximum)
    - varchar : chaîne de caractère à longueur variable (8000 caractères maximum)
    - text : chaîne de caractère de taille pouvant aller jusqu'à 2^31 – 1 caractères.
  - chaînes de caractères incluant les caractères Unicode.:
    - nchar : chaîne de caractères fixe (8000 caractères maximum) incluant les caractères Unicode.
    - nvarchar : chaîne de caractères à longueur variable (8000 caractères maximum) incluant les caractères Unicode.
    - ntext : chaîne de caractères de taille pouvant aller jusqu'à 2^31 – 1 caractères incluant les caractères Unicode.
  - chaînes binaires :
    - binary : chaîne binaire de taille fixe (8000 octets maximum)
    - varbinary : chaîne binaire de taille variable (8000 octets maximum)
    - image : chaîne binaire d'une taille pouvant aller jusqu'à 2^31 – 1 octets
  - autres :
    - cursor : référence vers un curseur. Plus de détails au chapitre sur les curseurs.
    - sql_variant : pouvant contenir tous les types, à l'exception de text, ntext, image et sql_variant.
    - table : type utilisé pour stocker des résultats pour une utilisation ultérieure.
    - timestamp : nombre unique mis à jour à chaque mise à jour d'un enregistrement.
    - uniqueidentifier : identifiant global unique de type GUID

Nous verrons plus tard que nous pouvons créer nos propres types de donnée.

A quoi sert le « n » devant les types de chaînes de caractères ? Il correspond en fait aux types utilisant les caractères Unicode. Ces types permettent d'utiliser les caractères spéciaux liés aux langues (les accents, etc.). On peut toutefois utiliser des caractères spéciaux, si on est sur le même jeu de caractère que le serveur, mais il est peu recommandé de partir du principe que le serveur le restera (mise à jour, changement de configuration, de logiciel, etc.). Utilisez donc les « n-types » lorsque vous risquez d'avoir des caractères spéciaux. Toutefois, les « n-types » utilisent deux fois plus de place (les caractères étant codés sur 16 bits au lieu de 8 comme avec les caractères « normaux »), il faut donc bien étudier le modèle de données avant de faire le choix.

Les types « variables » (varchar, nvarchar, varbinary) permettent de limiter le gaspillage de place en ne stockant que les données effectives. Par exemple, déclarer un char(20) prendra systématiquement 20 octets en mémoire, alors que déclarer un varchar(20) ne prendra que la taille réelle de la valeur spécifiée.

5.4.3. Utilisation des variables :

Pour pouvoir utiliser une variable, il faut d'abords qu'elle ait été déclarée.
Vous pourrez ensuite l'affecter de deux manières, et récupérer sa valeur très simplement.
Exécutez la requête ci-dessus, en vérifiant que vous êtes bien sur la base de données « Northwind ».

```
DECLARE @AvgUnitPrice money

SELECT @AvgUnitPrice = AVG(UnitPrice)
FROM Products

SELECT @AvgUnitPrice
```

Sortie :

```
28.8663
```

Que font ces quelques lignes de code ?
Tout d'abord, on déclare un variable @AvgUnitPrice de type money. On l'affecte ensuite via la clause SELECT, et on lui donne la valeur moyenne de la colonne UnitPrice.
Enfin, on affiche la variable grâce au SELECT

Autre exemple :

```
DECLARE @ProductCount int

SET @ProductCount = (SELECT COUNT(ProductID)
FROM Products)

SELECT @ProductCount
```

Sortie :

```
77
```

La différence avec l'exemple précédent est que cette fois ci nous utilisons le mot SET au lieu de SELECT pour affecter la variable. Il faut bien faire attention avec SET au type de donnée et au nombre de champs retournés qui doit être un. A l'opposé, avec SELECT, il faut faire attention car si on renvoie plusieurs lignes, la valeur affectée sera celle de la dernière ligne affectée.

On peut également affecter plusieurs variables en une fois avec SELECT, tandis qu'avec SET on ne peut pas :

```
DECLARE @AvgUnitPrice money
DECLARE @ProductCount int

SELECT      @AvgUnitPrice = AVG(UnitPrice),
            @ProductCount = COUNT(ProductID)
FROM Products

SELECT @AvgUnitPrice, @ProductCount
```

Sortie :

```
28.8663                 77
```

Vous voyez ici comment affecter deux variables avec un seul SELECT, ainsi que comment en afficher plusieurs, toujours avec SELECT.

5.4.4. Types de données utilisateur

Vous pouvez créer vos propres types de données. Par exemple, pour représenter un code postal, on utilise couramment varchar(5) comme type de donnée.
Nous allons donc créer un type de donnée ZipCode pour mapper ce type.

```
EXEC sp_addtype @typename='ZipCode', @phystype='varchar(5)'
```

Il faut indiquer le nom du nouveau type et le type physique des données.
Il est alors très facile d'utiliser le nouveau type :

```
DECLARE @cp ZipCode

SET @cp ='75010'

SELECT @cp
```

Il faut utiliser les types de données utilisateur avec précaution, car si l'on modifie le type, toutes les données de toutes les tables de ce type risquent de ne pas pouvoir être converties.

5.5. IF / ELSE

Avec T-SQL vous pouvez écrire des routines de test avec IF et ELSE

Exemple :

```
DECLARE @ProductCount int

SET @ProductCount = (SELECT COUNT(ProductID) FROM Products)

IF (@ProductCount > 100)
      PRINT 'Plus de 100 références'
ELSE
      PRINT 'Moins de 100 références'
```

Sortie, mais attention, la commande PRINT transmet des messages. Si vous utilisez un logiciel qui va afficher sous forme de grille les résultats, vous ne verrez rien. Utilisez par exemple l'analyseur de requête et affichez les résultats sous forme de texte.

```
Moins de 100 références
```

Le mot clef IF attend une expression booléenne. Vous pouvez utiliser les opérateurs de comparaison courants (=, >, <, >=, <=, <>, != , !<, !>) et utiliser les combinaisons logiques (AND, ALL, ANY, BETWEEN, EXISTS, IN, LIKE, NOT, OR, SOME). Voyez la MSDN pour plus de détails concernant les opérateurs.

Le bloc ELSE est optionnel, et vous pouvez imbriquer autant de IF que vous le souhaitez. Si l'un des blocs faits plus d'une ligne, imbriquez les lignes dans un block BEGIN END. Exemple :

```
DECLARE @ProductCount int

SET @ProductCount = (SELECT COUNT(ProductID) FROM Products)

IF (@ProductCount > 100)
      BEGIN
            PRINT 'Plus de 100 références'
            PRINT @ProductCount
      END
ELSE
      BEGIN
            PRINT 'Moins de 100 références'
            PRINT @ProductCount
      END
```

Il aura fallu mettre ici entre un BEGIN END les lignes, sinon il y aurait eu une erreur. Notez au passage que l'on peut afficher directement une variable avec PRINT, si la conversion en nvarchar est possible.

Autre exemple :

```
DECLARE @ProductCount int

SET @ProductCount = (SELECT COUNT(ProductID) FROM Products)

IF (@ProductCount < 100)
      BEGIN
            PRINT 'Plus de 100 références'
            PRINT @ProductCount

            DECLARE @MaxPrice money

            SELECT @MaxPrice = MAX(UnitPrice)
            FROM Products

            PRINT 'Le prix maximum est de ' +
                  CONVERT(nvarchar(10),@MaxPrice) +
                  ' €'
      END
ELSE
      BEGIN
            PRINT 'Moins de 100 références'
            PRINT @ProductCount
      END
```

Sortie :

```
Plus de 100 références
77
```

```
Le prix maximum est de 263.50 €
```

Cet exemple est un peu plus complexe que l'exemple précédent. Observez tout d'abord que l'on peut déclarer une variable dans un bloc BEGIN END, et que cette variable ne sera disponible que dans le bloc où elle a été déclarée.
Observez également que pour pouvoir afficher une variable, il faut parfois la convertir, en fonction de sont type. Ici, le type money n'est pas affichable directement, c'est pourquoi il faut utiliser CONVERT en spécifiant le type de sortie et l'entrée (variable ou colonne). On concatène ensuite les chaînes avec les « + ».

5.6. WHILE

Vous pouvez utiliser la structure WHILE dans une requête T-SQL. Le principe est le même que le while du C, C++, C#, ou la plupart des autres langages, c'est-à-dire que tant que l'expression évaluée est vraie, on exécute la boucle, sinon, on quitte le WHILE.

Exemple :

```
DECLARE @index int

SET @index = 0

WHILE @index < 10
      BEGIN
            PRINT 'Index=' + CONVERT(varchar(2),@index)
            SET @index = @index + 1
      END
```

Sortie :

```
Index=0
Index=1
Index=2
Index=3
Index=4
Index=5
Index=6
Index=7
Index=8
Index=9
```

Vous voyez donc que la boucle affiche le contenu de la variable, puis l'incrémente. La boucle sera réitérée si l'expression évaluée est vraie, ici, l'index devant être inférieur à 10.

Vous pouvez également utiliser CONTINUE et BREAK pour influencer sur le comportement de la boucle.

Par exemple :

```
DECLARE @index int

SET @index = 0

WHILE @index < 10
      BEGIN
            PRINT ''
            PRINT 'Index=' + CONVERT(varchar(2),@index)
            SET @index = @index + 1

            IF @index = 7
                  BEGIN
                        PRINT 'Le nombre 7 fait quitter le while'
                        BREAK
                  END

            IF (@index -1) % 3 = 0
                  BEGIN
                        PRINT 'On passe à la prochaine itération'
                        CONTINUE
                  END

            PRINT 'Le nombre n''est pas un multiple de 3'

      END
```

Sortie :

```
Index=0
On passe à la prochaine itération

Index=1
Le nombre n'est pas un multiple de 3

Index=2
Le nombre n'est pas un multiple de 3

Index=3
On passe à la prochaine itération

Index=4
Le nombre n'est pas un multiple de 3

Index=5
Le nombre n'est pas un multiple de 3

Index=6
Le nombre 7 fait quitter le while
```

Explications :

A chaque passage dans la boucle WHILE, on va afficher la variable @index, puis l'incrémenter. La nouveauté de cet exemple est que dans chaque itération, on teste si @index est égal à 7. Dans ce cas, on quitte le WHILE avec BREAK. On teste également si le nombre est un multiple de 3. Si c'est le cas, on passe à la prochaine itération avec CONTINUE. Si aucun de ces tests n'est vérifié, on exécute

la fin du WHILE qui consiste simplement ici à prévenir l'utilisateur que l'index en cours n'est pas un multiple de 3.
Observez également que pour afficher une « ' », il faut la doubler, pour indiquer que ce n'est pas la fin de la chaîne de caractère, mais une apostrophe. C'est le même principe qu'en C et dérivés où il faut doubler le « \ » pour l'afficher.

5.7. GOTO / LABEL

Cette structure permet de déclarer des étiquettes et de sauter à ces dernières. Cette structure étant peu utilisée, je ne ferai que donner un exemple simple :

```
DECLARE @index int

SET @index = 0

debut:

PRINT 'Index=' + CONVERT(varchar(2),@index)
SET @index = @index + 1

IF @index != 10
      GOTO debut
```

Sortie :

```
Index=0
Index=1
Index=2
Index=3
Index=4
Index=5
Index=6
Index=7
Index=8
Index=9
```

Explications :

Le mot clef GOTO, change le flux d'exécution pour indiquer au moteur SQL de passer à ligne définie par une étiquette. On utilise le « : » pour marquer une étiquette.

5.8. CASE (simple)

Il existe deux utilisation de CASE en T-SQL : le CASE « simple » ou le CASE « de rechercher »
Le CASE « simple » du T-SQL est l'équivalent du Switch en C ou du Select Case en Visual Basic. Le principe est de proposer une expression, et plusieurs propositions possible. Le moteur SQL basculera alors sur la proposition égale à l'expression.

Exemple :

```
DECLARE @anyNumber int, @anyString nvarchar(20)

SET @anyNumber = 42

SELECT @anyString =
      CASE @anyNumber
```

```
            WHEN 10 THEN 'dix'
            WHEN 24 THEN 'vingt quatre'
            WHEN 42 THEN 'quarante deux'
            WHEN 68 THEN 'soixante huit'
            ELSE 'aucun'
      END

PRINT 'Le nombre est ' + @anyString
```

Sortie :

```
Le nombre est quarante deux
```

Explications :

La structure CASE retourne la valeur spécifiée après le THEN lorsque l'expression spécifiée est égale à la valeur donnée avec WHEN.
Si aucune valeur n'est trouvée, on utilise alors celle spécifié avec ELSE. La clause ELSE étant obligatoire.
Attention aux types, utilisez au besoin les fonctions de convertion.

5.9. CASE (recherche)

La deuxième variante du CASE consiste à ne pas fournir d'expression à comparer. Il faudra alors fournir à chaque WHEN une expression booléenne qui sera évaluée. Si cette dernière est vérifiée, alors l'expression retournée sera celle du THEN correspondant.

Modifions notre précédent exemple :

```
DECLARE @anyNumber int, @anyString nvarchar(20)

SET @anyNumber = 42

SELECT @anyString =
      CASE
            WHEN @anyNumber < 20 THEN 'inférieur à vingt'
            WHEN @anyNumber < 40 THEN 'inférieur à quarante'
            WHEN @anyNumber < 60 THEN 'inférieur à soixante'
            WHEN @anyNumber < 80 THEN 'inférieur à quatre-vingt'
            ELSE 'très grand !'
      END

PRINT 'Le nombre est ' + @anyString
```

Sortie :

```
Le nombre est inférieur à soixante
```

Dans cette variante du CASE, on évalue une à une chacune des expressions fournies aux WHEN, jusqu'à en trouver une qui soit vraie. Dans ce cas, on retourne l'expression fournie avec le THEN correspondant.

5.10. Intégration de SQL

Dans les exemples précédents, nous n'utilisions que des variables locales. Vous pouvez bien sûr utiliser le SQL standard pour traiter les données.

Exemple avec un IF :

```
SELECT
      UnitPrice,
      CASE  WHEN UnitPrice < 30 THEN 'pas cher'
            WHEN UnitPrice < 60 THEN 'moyen cher'
            ELSE 'très cher'
      END AS Comment
FROM Products
```

Sortie (tronquée pour la mise en page) :

```
UnitPrice             Comment
--------------------- ----------
18.0000               pas cher
19.0000               pas cher
10.0000               pas cher
...
30.0000               moyen cher
```

```
62.5000               très cher
13.0000               pas cher

(77 ligne(s) affectée(s))
```

De la même façon, on peut utiliser des variables dans des requêtes INSERT, UPDATE, etc. C'est très utile pour les procédures stockées car vous pouvez utiliser les arguments de la procédure stockée. Plus de détails lors du chapitre sur les procédures stockées.

Exemple :

```
DECLARE @NewPrice money
DECLARE @ProductID int

SET @NewPrice = 4.5
SET @ProductID = 5

-- Affichage des données actuelles
SELECT ProductID, ProductName, Unitprice
FROM Products

-- Mise à jour des données
UPDATE Products
SET UnitPrice = @NewPrice
WHERE ProductID = @ProductID

-- Affichage des données mises à jour
SELECT ProductID, ProductName, Unitprice
FROM Products
```

Sortie (tronquée):

```
ProductID   ProductName                              UnitPrice
----------- ---------------------------------------- ---------------------
1           Chai                                     18.0000
2           Chang                                    19.0000
3           Aniseed Syrup                            10.0000
4           Chef Anton's Cajun Seasoning             22.0000
5           Chef Anton's Gumbo Mix                   18.5000
6           Grandma's Boysenberry Spread             25.0000

.......................

76          Lakkalikööri                             18.0000
77          Original Frankfurter grüne Soße          13.0000

(77 ligne(s) affectée(s))


(77 ligne(s) affectée(s))

ProductID   ProductName                              UnitPrice
----------- ---------------------------------------- ---------------------
1           Chai                                     18.0000
2           Chang                                    19.0000
3           Aniseed Syrup                            10.0000
4           Chef Anton's Cajun Seasoning             22.0000
5           Chef Anton's Gumbo Mix                   4.5000
6           Grandma's Boysenberry Spread             25.0000
```

```
.....................

76          Lakkalikööri                               18.0000
77          Original Frankfurter grüne Soße            13.0000

(77 ligne(s) affectée(s))
```

Explications :

C'est un exemple simple dans lequel sont mixés l'utilisation de variables et l'utilisation de requêtes SQL.

5.11. Procédures stockées

5.11.1. Déclarer une procédure stockée

Les procédures stockées peuvent s'apparenter à des fonctions du coté serveur. En effet, on va « programmer » des requêtes avec d'éventuels arguments, retours, etc. que l'on appellera par la suite.
Voici un premier exemple de procédure stockée :

```
CREATE PROCEDURE GetProductID
AS
SELECT ProductID
FROM Products

GO
```

Cet exemple montre comment créer la procédure stockée. La première ligne, indique que l'on crée une procédure stockée (CREATE PROCEDURE) du nom de GetProductID. Le nom est de la forme suivante :

- [owner].name

Où « owner » est le propriétaire (optionnel, utilisateur de la session par défaut) et « name » le nom de la procédure stockée.
Vous auriez donc aussi pu écrire les lignes suivantes pour arriver au même résultat :

```
CREATE PROCEDURE dbo.GetProductID
AS
SELECT ProductID
FROM Products

GO
```

« dbo » étant le propriétaire de la base de donnée (**D**ata**B**ase **O**wner)

Ou :

```
CREATE PROCEDURE Steve.GetProductID
AS
SELECT ProductID
FROM Products

GO
```

Si l'utilisateur « Steve » a les droits nécessaires.

Le mot clef GO permet d'exécuter les opérations en cours. C'est utile si vous créez la procédure dans une requête qui va immédiatement l'utiliser.

5.11.2. Modification d'une procédure stockée existante

Pour modifier une procédure stockée existante, vous pouvez utiliser le mot clef ALTER et non plus CREATE pour changer le corps de la procédure. Changeons notre procédure stockée de tout à l'heure pour afficher également le nom des produits :

```
ALTER PROCEDURE GetProductID
AS
SELECT ProductID, ProductName
FROM Products

GO
```

La procédure est alors modifiée.

5.11.3. Appel d'une procédure stockée

Pour appeler une procédure stockée, qu'elle soit développée par un utilisateur ou qu'elle soit système (nous verrons plus loin ces différentes procédures stockées), il faut employer le mot clef EXEC ou EXECUTE.
Ainsi, pour appeler notre procédure stockée GetProductID citée plus haut, utilisez :

```
EXEC GetProductID
```

Sortie (tronquée) :

```
ProductID   ProductName
----------- ----------------------------------------
17          Alice Mutton
3           Aniseed Syrup
...
64          Wimmers gute Semmelknödel
47          Zaanse koeken

(77 ligne(s) affectée(s))
```

Vous pouvez donc constater que la sortie est celle programmée dans la procédure stockée. Vous pouvez ainsi exécuter tout type de requête dans une procédure stockée.

5.11.4. Paramètres

Les procédures ne seraient pas intéressantes si on ne pouvait pas spécifier de paramètres. Heureusement, il est très facile d'écrire un procédure stockée paramétrable.
Ainsi, si on souhaite afficher le détail d'un des produits, on filtrera les données par son ProductID, que l'on spécifiera en paramètre :

```
CREATE PROCEDURE GetSingleProduct
(
      @ProductID int
)
AS
SELECT ProductID, ProductName
FROM Products
WHERE ProductID=@ProductID
GO
```

Pour déclarer un paramètre, il suffit donc de le spécifier dans l'entête de la procédure en lui indiquant :

- Son nom : ici @ProductID (n'oubliez pas le @), qui sera utilisable comme une variable dans la procédure stockée.

- Un type de donnée, choisi parmi les types SQL ou les types utilisateurs.
- Une valeur par défaut optionnelle.
- Une direction, En mettant OUTPUT derrière le nom d'un paramètre, vous pouvez indiquer que le paramètre est un paramètre de sortie, mais nous y reviendrons.

Pour appeler alors une telle procédure, il faut toujours utiliser EXEC ou EXECUTE, en lui adjoignant la liste des paramètres, soit explicitement en mettant autant de couple : « @paramètre=*Valeur* » que nécessaire, séparés par une virgule :

```
EXEC GetSingleProduct @ProductID=20
```

Soit en mettant les paramètres les uns à la suite des autres dans l'ordre attendu par la procédure stockée, toujours séparés par une virgule :

```
EXEC GetSingleProduct 20
```

Les deux méthodes donneront ce même résultat :

```
ProductID   ProductName
----------- ----------------------------------------
20          Sir Rodney's Marmalade

(1 ligne(s) affectée(s))
```

Il est recommandé d'utiliser explicitement les noms des paramètres, car si la procédure venait à être changée, les paramètres risquent de ne plus être les mêmes ou d'avoir le même ordre.
De plus, il faut faire attention au type de données. En effet, le type du paramètre fourni, doit soit avoir le même type que celui du paramètre attendu, soit être convertible implicitement.

5.11.5. Paramètres optionnels

Vous pouvez spécifier pour chaque paramètre une valeur par défaut. Modifions notre précédente procédure comme ceci (n'oubliez pas d'utiliser ALTER et non plus CREATE puisque la procédure existe déjà) :

```
ALTER PROCEDURE GetSingleProduct
(
      @ProductID int = null
)
AS
IF @ProductID IS null
      BEGIN
            SELECT ProductID, ProductName
            FROM Products
      END
ELSE
      BEGIN
            SELECT ProductID, ProductName
            FROM Products
            WHERE ProductID=@ProductID
      END

GO
```

Cette procédure stockée a deux usages : soit on ne spécifie pas de paramètre (équivaut à lui donner la valeur null, puisque c'est la valeur par défaut) et il affiche tous les produits, soit on lui donne un entier en paramètre, et il affichera le produit correspondant.

- Premier cas de figure :

Appel avec :

```
EXEC GetSingleProduct @ProductID=20
```

Ou :

```
EXEC GetSingleProduct 20
```

Donnera :

```
ProductID   ProductName
----------- ----------------------------------------
20          Sir Rodney's Marmalade

(1 ligne(s) affectée(s))
```

- Deuxième cas de figure :

Appel avec :

```
EXEC GetSingleProduct @ProductID=default
```

Ou :

```
EXEC GetSingleProduct
```

Donnera :

```
ProductID   ProductName
----------- ----------------------------------------
17          Alice Mutton
3           Aniseed Syrup
...
64          Wimmers gute Semmelknödel
47          Zaanse koeken

(77 ligne(s) affectée(s))
```

En effet, ne pas spécifier un paramètre lui donne sa valeur par défaut. Bien sûr, si le paramètre ne possède pas de valeur par défaut, une erreur surviendrait.

5.11.6. Direction des paramètres

Vous pouvez également définir des paramètres de sortie. Ces paramètres sont modifiés dans la procédure stockée puis retournés à l'appelant.

Exemple : Insertion d'un nouveau fournisseur (Shippers)

```
CREATE PROCEDURE AddShipper
(
      @CompanyName nvarchar(40),
      @Phone nvarchar(24),
      @NewID int OUTPUT
)
AS
```

```
INSERT INTO Shippers
      (
            CompanyName,
            Phone
      )
      VALUES
      (
            @CompanyName,
            @Phone
      )

SET @NewID=@@IDENTITY

GO
```

Puis pour utiliser ce paramètre :

```
DECLARE @InsertedID int

EXEC AddShipper
            @CompanyName ='MS Press',
            @Phone='01 45 00 49 87',
            @NewID= @InsertedID OUTPUT

SELECT @InsertedID
```

On a donc dans un premier temps déclaré notre paramètre @NewID comme étant un paramètre de sortie. Dans la procédure stockée, on lui donne à la fin la valeur de @@IDENTITY qui est une fonction système retournant le dernier ID inséré dans la table, ce signifie que @NewID contiendra l'ID du fournisseur créé à l'instant.
Ensuite, lors de l'appel, nous avons tout d'abord déclaré une variable « réceptacle », puis nous l'avons lié au paramètre de sortie @NewID via : « @NewID = @Inserted OUTPUT » du EXEC. N'oubliez pas le mot clef OUTPUT, sinon, @Inserted ID gardera sa valeur d'avant l'appel.

5.11.7. Procédures stockées systèmes et prédéfinies

Chaque base de données contient un certain nombre de procédures stockées prédéfinies. Soit ce sont des procédures stockées système, soit incluses dans la base lors de sa création. En effet, la création de la base de donnée se fait en dupliquant la base de donnée « model ». Si vous avez écrit des procédures stockées dans cette dernière, elles seront automatiquement copiées dans la nouvelle base de données.
Pour ce qui est des procédures stockées « systèmes », ce sont des procédures stockées incluses dans « master » que tout le monde peut appeler (à condition d'y avoir le droit). Ces procédures servent principalement à l'administration du serveur, et nous ne pourrons pas les détailler toutes ici sans sortir du cadre de ce cours. Je vais simplement présenter un exemple concret d'utilisation : l'ajout d'un nouvel utilisateur à une base de donnée.

- Création du login sur le serveur :

En effet, chaque utilisateur de base de données doit être logué sur le serveur avant de pouvoir utiliser une base de données. Il faut donc créer un login pour une authentification SQL Server, ou accorder à un utilisateur le droit de se loguer, pour une authentification Windows. Dans notre cas, nous allons ajouter un utilisateur Windows, donc accordons lui le droit de se loguer :

```
EXEC sp_grantlogin @Loginame='SATURNE\Matthieu'
```

Résultat :

```
Connexion d'accès accordée pour 'SATURNE\Matthieu'.
```

Nous avons ici utilisé la procédure stockée système « sp_grantlogin ». Pour plus de détails, consultez l'aide, mais sachez que cette procédure attend entre autre, un nom d'utilisateur Windows pour lui accorder le droit de se connecter.

- Ajout de cet utilisateur à la base de données NorthWind

```
USE NorthWind

EXEC sp_grantdbaccess 'SATURNE\Matthieu'
```

Résultat :

```
Accès accordé à la base de données 'SATURNE\Matthieu'.
```

Ici, c'est la procédure stockée système « sp_grantdbaccess » que nous avons utilisé. « USE Northwind » est nécessaire pour se placer dans la base de données NorthWind, pour donner l'accès à cette dernière.

- Attribution des droits :

Enfin, il faut indiquer quels sont les droits de cet utilisateur dans cette base de donnée. Pour cela, plutôt que de donner les droits manuellement, nous allons l'inclure dans un groupe (ou rôle) :

```
EXEC sp_addrolemember
      @rolename='db_datareader',
      @membername= 'SATURNE\Matthieu'
```

Enfin, nous avons inclut l'utilisateur ainsi crée dans le rôle « db_datareader ».

Pour conclure avec les procédures stockées, je vous invite à consulter l'aide pour connaître l'utilité de chacune, et leur syntaxe. La plupart de ses procédures étant destinées à l'administrateur, vous n'aurez que peu souvent besoin de vous en servir.

5.12. Transactions

5.12.1. Utilité

Les transactions permettent de définir des blocs de code dont le succès d'exécution doit être traité comme un seul bloc. Plus clairement, toutes les instructions comprises dans la transaction doivent être exécutées avec succès, ou tout le bloc sera annulé.
Quel est l'intérêt ? Imaginons l'insertion d'une facture et des lignes de facturation associées. La première chose à faire est d'insérer l'enregistrement d'une facture, puis, pour chaque ligne insérée l'enregistrement correspondant. Si on essaye d'insérer une ligne de facturation dont le produit n'existe pas (erreur de clef primaire/étrange), une erreur va être générée. Deux cas de figure :

- Sans transaction :
  - La facture sera insérée
  - la première ligne de facturation va être insérée
  - la seconde ligne
  - …
  - la nième ligne : provoque une erreur
  - la requête est interrompue
  - au final : une facture ajoutée, et n-1 lignes alors que la facture n'est pas valide.

- Avec transaction :
  - La facture sera insérée
  - la première ligne de facturation va être insérée
  - la seconde ligne
  - …
  - la nième ligne : provoque une erreur
  - la requête est interrompue ET ANNULEE.
  - au final : les modifications (insertion d'une facture et n-1 lignes) ne seront pas prises en compte car la transaction n'aura pas été exécutée avec succès.

L'intérêt des transactions est donc de permettre d'attendre qu'une requête soit complètement terminée avant de valider les modifications.

5.12.2. Exemple :

Les transactions sont faciles à mettre en place en T-SQL. Observez cet exemple qui exécute deux fois la même chose, une fois sans et une fois avec une transaction:

```
-- Etat initial
PRINT 'Lignes actuelles de la commande 10248 :'
PRINT ''
SELECT OrderID, ProductID, UnitPrice
FROM [Order Details]
WHERE OrderID=10248

/* Ajout sans transaction d'une ligne valide et
   d'une ligne non valide (product ID inexistant)*/

PRINT ''
PRINT 'Ajout du produit n°14'
PRINT ''
INSERT INTO [Order Details]
      ( OrderID, ProductID, UnitPrice, Quantity, Discount )
      VALUES
      ( 10248 , 14, 10, 1, 0)

PRINT ''
PRINT 'Ajout du produit n°14000'
PRINT ''
INSERT INTO [Order Details]
      ( OrderID, ProductID, UnitPrice, Quantity, Discount )
      VALUES
      ( 10248 , 14000, 10, 1, 0)

-- Etat actuel :
PRINT ''
PRINT 'Lignes actuelles de la commande 10248 :'
PRINT ''
SELECT OrderID, ProductID, UnitPrice
FROM [Order Details]
WHERE OrderID=10248

/* Ajout avec transaction d'une ligne valide et
   d'une ligne non valide (product ID inexistant)*/

PRINT ''
PRINT 'Début de la transaction'
PRINT ''
BEGIN TRANSACTION myTran
```

```
PRINT ''
PRINT 'Ajout du produit n°65'
PRINT ''
INSERT INTO [Order Details]
      ( OrderID, ProductID, UnitPrice, Quantity, Discount )
      VALUES
      ( 10248 , 65, 10, 1, 0)

PRINT ''
PRINT 'Ajout du produit n°1111'
PRINT ''
INSERT INTO [Order Details]
      ( OrderID, ProductID, UnitPrice, Quantity, Discount )
      VALUES
      ( 10248 , 1111, 10, 1, 0)

IF @@ERROR <>0
      ROLLBACK TRANSACTION myTran
ELSE
      COMMIT TRANSACTION myTran

PRINT ''
PRINT 'Fin de la transaction'
PRINT ''

-- Etat actuel
PRINT ''
PRINT 'Lignes actuelles de la commande 10248 :'
PRINT ''
SELECT OrderID, ProductID, UnitPrice
FROM [Order Details]
WHERE OrderID=10248
```

Sortie :

```
Lignes actuelles de la commande 10248 :

OrderID     ProductID   UnitPrice
----------- ----------- ---------------------
10248       11          14.0000
10248       42          9.8000
10248       72          34.8000

(3 ligne(s) affectée(s))


Ajout du produit n°14


(1 ligne(s) affectée(s))


Ajout du produit n°14000

Serveur : Msg 547, Niveau 16, État 1, Ligne 1
Conflit entre l'instruction INSERT et la contrainte COLUMN FOREIGN KEY
'FK_Order_Details_Products'. Le conflit est survenu dans la base de données
'Northwind', table 'Products', column 'ProductID'.
L'instruction a été arrêtée.

Lignes actuelles de la commande 10248 :
```

```
OrderID     ProductID   UnitPrice
----------- ----------- ---------------------
10248       11          14.0000
10248       14          10.0000
10248       42          9.8000
10248       72          34.8000

(4 ligne(s) affectée(s))


Début de la transaction


Ajout du produit n°65


(1 ligne(s) affectée(s))


Ajout du produit n°1111

Serveur : Msg 547, Niveau 16, État 1, Ligne 1
Conflit entre l'instruction INSERT et la contrainte COLUMN FOREIGN KEY
'FK_Order_Details_Products'. Le conflit est survenu dans la base de données
'Northwind', table 'Products', column 'ProductID'.
L'instruction a été arrêtée.

Fin de la transaction


Lignes actuelles de la commande 10248 :

OrderID     ProductID   UnitPrice
----------- ----------- ---------------------
10248       11          14.0000
10248       14          10.0000
10248       42          9.8000
10248       72          34.8000

(4 ligne(s) affectée(s))
```

Déroulement de la requête.

- Tout d'abord, on affiche l'état actuel de la commande n°10248 en affichant toutes les lignes de facturation associées. Pour le moment il y en a 3
- On tente ensuite d'insérer deux nouveaux enregistrements. Le deuxième est volontairement invalide (contraintes de clef étrangère) pour générer une erreur. Le déroulement passe alors sur le premier INSERT sans problème tandis que le second provoque l'erreur souhaitée. On affiche alors l'état actuel, et on se rend compte qu'il y'a 4 lignes, celle en plus provenant du premier INSERT, le deuxième INSERT étant annulé.
- On retente la même chose, cette fois ci en englobant le même genre d'instructions dans une transaction.
    - La première chose à faire est de déclarer la transaction avec « BEGIN TRANSACTION nom_de_la_transaction ». Le nom est facultatif mais permet d'identifier précisément la transaction.
    - Viennent ensuite les instructions potentiellement génératrices d'erreur.
    - Vous pouvez alors ordonner à la transaction soit de valider les modifications via la commande « COMMIT nom_de_la_transaction », soit d'annuler les modifications en utilisant « ROLLBACK nom_de_la_transaction ». Ainsi, dans notre exemple, on teste

la présence d'une erreur avec @@ERROR qui n'est à 0 que si le déroulement c'est bien passé. Si il y'a eu une erreur, on annule les modifications, sinon on les valide.

- Dans le corps de la transaction, le premier INSERT est valide, le second provoquera une erreur. Mais comme nous l'avons englobé dans une transaction, les deux seront ignorés. Il n'est donc pas surprenant de voir qu'il n'y a toujours que 4 lignes et non 5.

Les transactions sont donc utiles pour définir des points de validités au sein d'une requête. Il faut toutefois noter qu'une procédure stockée est une transaction « implicite ». Ainsi, si dans une procédure stockée une erreur devait survenir, toute la procédure est annulée.

Exemple :

```
-- Création de la procédure stockée
CREATE PROCEDURE InsertSomething
(
	@OrderID    int,
	@ProductID  int,
	@UnitPrice  money,
	@Quantity   smallint,
	@Discount   real
)
AS
INSERT INTO [Order Details]
	(
		OrderID,
		ProductID,
		UnitPrice,
		Quantity,
		Discount
	)
	VALUES
	(
		@OrderID ,
		@ProductID,
		@UnitPrice,
		@Quantity,
		@Discount
	)

GO


-- Etat initial
PRINT ''
PRINT 'Lignes actuelles de la commande 10248 :'
PRINT ''

SELECT OrderID, ProductID, UnitPrice
FROM [Order Details]
WHERE OrderID=10248

-- Exécution de la procédure stockée avec des valeurs "valables"
EXEC InsertSomething 10248, 65, 10, 1, 0

-- Etat actuel
PRINT ''
PRINT 'Lignes actuelles de la commande 10248 :'
PRINT ''

SELECT OrderID, ProductID, UnitPrice
```

```
FROM [Order Details]
WHERE OrderID=10248

-- Exécution de la procédure stockée avec des valeurs non "valables"
EXEC InsertSomething 10248, 11111, 10, 1, 0

-- Etat actuel
PRINT ''
PRINT 'Lignes actuelles de la commande 10248 :'
PRINT ''

SELECT OrderID, ProductID, UnitPrice
FROM [Order Details]
WHERE OrderID=10248
```

Sortie :

```
Lignes actuelles de la commande 10248 :

OrderID     ProductID   UnitPrice
----------- ----------- ---------------------
10248       11          14.0000
10248       42          9.8000
10248       72          34.8000

(3 ligne(s) affectée(s))


(1 ligne(s) affectée(s))


(7 ligne(s) affectée(s))


Lignes actuelles de la commande 10248 :

OrderID     ProductID   UnitPrice
----------- ----------- ---------------------
10248       11          14.0000
10248       42          9.8000
10248       65          10.0000
10248       72          34.8000

(4 ligne(s) affectée(s))

Serveur : Msg 547, Niveau  16, État 1, Procédure InsertSomething, Ligne 10
Conflit entre l'instruction INSERT et la contrainte COLUMN FOREIGN KEY
'FK_Order_Details_Products'. Le conflit est survenu dans la base de données
'Northwind', table 'Products', column 'ProductID'.
L'instruction a été arrêtée.

Lignes actuelles de la commande 10248 :

OrderID     ProductID   UnitPrice
----------- ----------- ---------------------
10248       11          14.0000
10248       42          9.8000
10248       65          10.0000
10248       72          34.8000
```

```
(4 ligne(s) affectée(s))
```

Vous pouvez constater que sans avoir eu à déclarer une transaction, les INSERT non valables sont ignorés. En effet, ces INSERT sont inclus dans une procédure stockée. Cette dernière contient une transaction implicite, et donc, s'il y a une erreur, toute la procédure sera annulée.

5.13. Curseurs

Les curseurs permettent de parcourir un jeu de données ligne par ligne.

Exemple :

```
-- Déclaration du curseur
DECLARE myCursor CURSOR LOCAL
FOR
SELECT OrderID, ProductID, UnitPrice, Quantity
FROM [Order Details]
WHERE OrderID = 10248

-- Ouverture du curseur
OPEN myCursor

-- Première récupération
FETCH NEXT FROM myCursor

-- Puis on boucle tant qu'il reste des lignes à récupérer
WHILE @@FETCH_STATUS = 0
      BEGIN
            -- Ceci est exécuté pour chaque ligne du curseur
            FETCH NEXT FROM myCursor
      END

-- Fermeture et déchargement du curseur
CLOSE myCursor
DEALLOCATE myCursor
```

Sortie :

```
OrderID     ProductID   UnitPrice             Quantity
----------- ----------- --------------------- --------
10248       11          14.0000               12

(0 ligne(s) affectée(s))

OrderID     ProductID   UnitPrice             Quantity
----------- ----------- --------------------- --------
10248       42          9.8000                10

(0 ligne(s) affectée(s))

OrderID     ProductID   UnitPrice             Quantity
----------- ----------- --------------------- --------
10248       72          34.8000               5

(0 ligne(s) affectée(s))

OrderID     ProductID   UnitPrice             Quantity
----------- ----------- --------------------- --------

(0 ligne(s) affectée(s))
```

L'utilisation d'un curseur se fait en plusieurs étapes :

- La déclaration : un curseur étant une variable (bien qu'un peu particulière), il faut la déclarer avec DECLARE, le type étant alors CURSOR. Il y a ensuite le mot clef LOCAL, qui indique que le curseur ne sera disponible que pour la session en cours. Vous pouvez également spécifier GLOBAL, la valeur par défaut étant définie par l'option du serveur « default to local cursor » qui est « false » par défaut.
- FOR *select statement* : On spécifie ici la requête voulue. Le curseur va ensuite balayer ligne par ligne les résultats de cette requête. Dans notre exemple, c'est un simple SELECT
- OPEN : permet d'ouvrir le curseur. C'est à ce moment que la requête spécifiée sera exécutée pour fournir un certain nombre d'enregistrements au curseur
- FETCH NEXT : permet de lire la ligne en cours. Le résultat est retourné dans la fonction système @@FETCH_STATUS qui contiendra 0 si le curseur n'a plus de ligne disponible. Il existe d'autres options que NEXT : PRIOR, FIRST, LAST, ABSOLUTE, RELATIVE mais il faut que vous ayez spécifié dans la déclaration du curseur le droit de l'utiliser. Consultez l'aide si vous avez besoin de telles fonctionnalités.
- CLOSE : permet de fermer le curseur
- DEALLOCATE : permet de libérer les ressources. Optionnel ici car le curseur est déclaré en LOCAL et est donc automatiquement libéré à la fin de la requête.

Cette requête récupère tout d'abord une première ligne. Si @@FETCH_STATUS vaut 0, la boucle WHILE sera de toute façon non exécutée, tandis que si c'est différent de 0, elle sera exécutée, car on sait qu'il y a une ligne disponible au curseur. Nous sommes obligés de passer par un double FETCH (un seul au début puis un dans la boucle WHILE) car l'équivalent du DO WHILE de certains langages n'existent pas.

Pour le moment cette requête ne fait que parcourir les lignes. Il serait bien plus intéressant de traiter les lignes une à une.

Modifions donc notre requête précédente :

```
-- Déclaration de variables tampon
DECLARE     @OrderID    int,
            @ProductID  int,
            @UnitPrice  money,
            @Quantity   smallint

-- Déclaration du curseur
DECLARE myCursor CURSOR LOCAL
FOR
SELECT OrderID, ProductID, UnitPrice, Quantity
FROM [Order Details]
WHERE OrderID = 10248

-- Ouverture du curseur
OPEN myCursor

-- Première récupération
FETCH NEXT FROM myCursor
INTO @OrderID, @ProductID, @UnitPrice, @Quantity

-- Puis on boucle tant qu'il reste des lignes à récupérer
WHILE @@FETCH_STATUS = 0
      BEGIN
            -- Ceci est exécuté pour chaque ligne du curseur
            PRINT 'OrderID: ' + CONVERT(nvarchar(50),@OrderID)
            PRINT CONVERT(nvarchar(50), @Quantity) +
                  ' exemplaires de produit ayant ' +
```

```
                  CONVERT(nvarchar(50), @ProductID) +
                  ' pour référence'
            PRINT 'Prix à l''unité : ' +
                  CONVERT(nvarchar(50), @UnitPrice)
            PRINT 'Soit un total de : '  +
                  CONVERT(nvarchar(50), @UnitPrice * @Quantity)
            PRINT ''

            -- on récupère les valeurs pour la prochaine ligne
            FETCH NEXT FROM myCursor
            INTO @OrderID, @ProductID, @UnitPrice, @Quantity
      END

-- Fermeture et déchargement du curseur
CLOSE myCursor
DEALLOCATE myCursor
```

Résultat :

```
OrderID: 10248
12 exemplaires de produit ayant 11 pour référence
Prix à l'unité : 14.00
Soit un total de : 168.00

OrderID: 10248
10 exemplaires de produit ayant 42 pour référence
Prix à l'unité : 9.80
Soit un total de : 98.00

OrderID: 10248
5 exemplaires de produit ayant 72 pour référence
Prix à l'unité : 34.80
Soit un total de : 174.00
```

Cette fois, nous avons spécifié à FETCH ce qu'il doit faire des lignes de résultat, c'est-à-dire les copier dans les variables @OrderID, @ProductID, @UnitPrice et @Quantity. Faites bien attention, c'est dans l'ordre des sorties de la requête déclarée dans le curseur. Ici, la colonne OrderID du SELECT du curseur sera copiée dans @OrderID et ainsi de suite.
Ces variables servent ensuite aux traitements.
Notez au passage l'utilisation de CONVERT qui permet de faire des conversions explicites. Ici, on convertit des entiers et un monétaire en chaîne nvarchar(50).

6. Conclusion

Ce document devrait vous avoir permis de découvrir SQL Server. C'est un SGBDR puissant et complet avec lequel vous pourrez développer des applications très performantes et très rapidement, notamment grâce aux supports de XML et de HTTP ainsi que grâce au Transact SQL.
Vous pourrez également voir dans le cours sur ADO.Net (http://www.labo-dotnet.com, rubrique Supinfo Dotnet Training Courses) que vous pourrez accéder aux données depuis le framework avec une simplicité déconcertante.