Dotnet France
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

Gestion des utilisateurs, des groupes et des rôles dans SQL Server 2008

Version 1.0

Microsoft CERTIFIED Technology Specialist
Grégory CASANOVA

Sommaire

1 Introduction

La sécurité est un domaine important dans les bases de données. Sans sécurité, nos données sont exposées à des risques d'altération de tous types, le plus souvent à cause des utilisateurs finaux, ou d'une mauvaise politique de gestion des bases. Il existe dans SQL Server 2008, trois types d'objets qui vont nous permettre de sécuriser nos bases et donc garantir une plus grande chance d'intégrités pour nos données. Ces trois types sont :

- **Les entités de sécurité** : compte de sécurité qui dispose d'un accès au serveur de données SQL.
- **Les sécurisables** : objets gérés par le serveur. Nous avons vu précédemment ce qu'était un objet dans une base de données ou dans un serveur.
- **Les autorisations** : celles-ci sont accordées aux entités de sécurité afin de pouvoir travailler avec les sécurisables.

Ces éléments de sécurité ont une organisation de type hiérarchique. Cette hiérarchie va nous permettre de mettre en place une politique sure, afin de maintenir l'intégrité de nos données, en faisant en sorte que cette gestion des droits d'accès soit simple, mais efficace. Par exemple, de façon hiérarchique, on pourra accorder l'autorisation « SELECT », à un sécurisable, par exemple un schéma, pour permettre à l'entité de sécurité qui reçoit cette autorisation d'effectuer l'opération SELECT sur tous les objets contenus dans le schéma définit. Parlons rapidement des schémas. Il en existe trois types définis. Les schémas de niveau serveur, qui vont s'appliquer à toutes les bases de données implantés sur le serveur. Les schémas de niveau base de données, actifs seulement sur la base de données ou il est définit. Et enfin les rôles d'application, définis sur la base de données utilisateur et qui vont permettre le bon fonctionnement d'une application cliente. Nous parlerons plus amplement des schémas dans la suite de ce chapitre. D'autres sécurisables existent, par exemple les utilisateurs simples, les groupes… Leur organisation vis-à-vis des autres éléments de sécurité ce fait aussi de manière hiérarchique, comme pour les schémas.

Il est possible d'obtenir un catalogue complet des utilisateurs et de leurs privilèges, grâce aux vues système. Voici quelques une de ces vues :

- **Sys.server_permissions** : Liste des permissions au niveau serveur et de leurs bénéficiaires.
- **Sys.server_principals :** Entités de sécurité définis au niveau serveur.
- **Sys.sql_logins :** Liste des connexions au niveau serveur.
- **Sys.Server_role_members :** Liste des bénéficiaires d'un rôle au niveau serveur.
- **Sys.database_permissions :** Liste des permissions et de leurs bénéficiaires au niveau base de données.
- **Sys.database_principals :** Entités de sécurité au niveau base de données.
- **Sys.database_role_members :** Liste des bénéficiaires d'un rôle de base de données.

Apprenons maintenant toutes les manières de préserver l'intégrité des données de nos bases, en mettant en place une politique de sécurité efficace.

2 Gestion des accès serveur

Avant de pouvoir commencer à travailler avec les données de nos bases, il est impératif de se logger sur le serveur SQL. Cette étape permet de s'identifier au niveau du serveur SQL, afin de pouvoir exploiter les droits qui ont étés attribués à notre connexion. Dans SQL Server, il existe plusieurs modes d'authentification que nous allons définir dans cette partie.

2.1 Mode de sécurité Windows

2.1.1 En général

Dans ce mode de sécurité, SQL Server s'appui sur les fonctionnalités de Windows. En effet, il se sert de la sécurité de Windows (login et mot de passe) pour créer des connexions aux instances et donc aux bases de données. Puisque le mode de sécurité Windows se sert de la sécurité du système sur lequel SQL Server est installé, la gestion des groupes est aussi possible. Il est donc plus facile d'administrer des connexions aux bases de données, de manière plus souple qu'utilisateur par utilisateur.

2.1.2 En détail

En réalité, seuls les login sont sauvegardés dans l'instance, de façon à permettre l'accès aux bases par la suite. Le mot de passe de chaque utilisateur est détenu par Windows. Vous comprendrez aisément qu'il est bien plus sécurisé de ne pas stocker le mot de passe d'un utilisateur directement dans une base système. De plus, la politique d'administration et de sécurité qui énonce la règle suivante : un mot de passe par utilisateur, est bien plus simple à mettre en place.

2.2 Mode de sécurité Mixte

2.2.1 En général

Le mode de sécurité mixte repose sur une double authentification : l'authentification Windows puis l'authentification SQL Server. C'est le mode le plus connu pour la sécurité des bases de données. Dans le principe, c'est SQL Server qui ce charge de vérifier que l'utilisateur existe et qu'il possède le bon mot de passe. Cela signifie que les utilisateurs sont entièrement gérés par SQL Server, autant les login que les mots de passe. Ce type d'identification est bien adapté pour une gestion des utilisateurs qui ne passent pas par une authentification Windows à la base.

2.2.2 En détail

Il est faux de penser que toute la sécurité du serveur est gérée par le serveur de base de données. Le fonctionnement mixte est basé sur un principe de vérification simple. Le système Windows vérifie si l'utilisateur existe et possède les droits de se connecter au serveur. Si c'est le cas, l'utilisateur sera connecté. Si ce n'est pas le cas, une erreur est levée, énonçant que l'utilisateur n'existe pas sur le système. Le serveur de base de données est alors chargé de demander le login et le mot de passe de la connexion.

2.3 Choisir son mode de sécurité

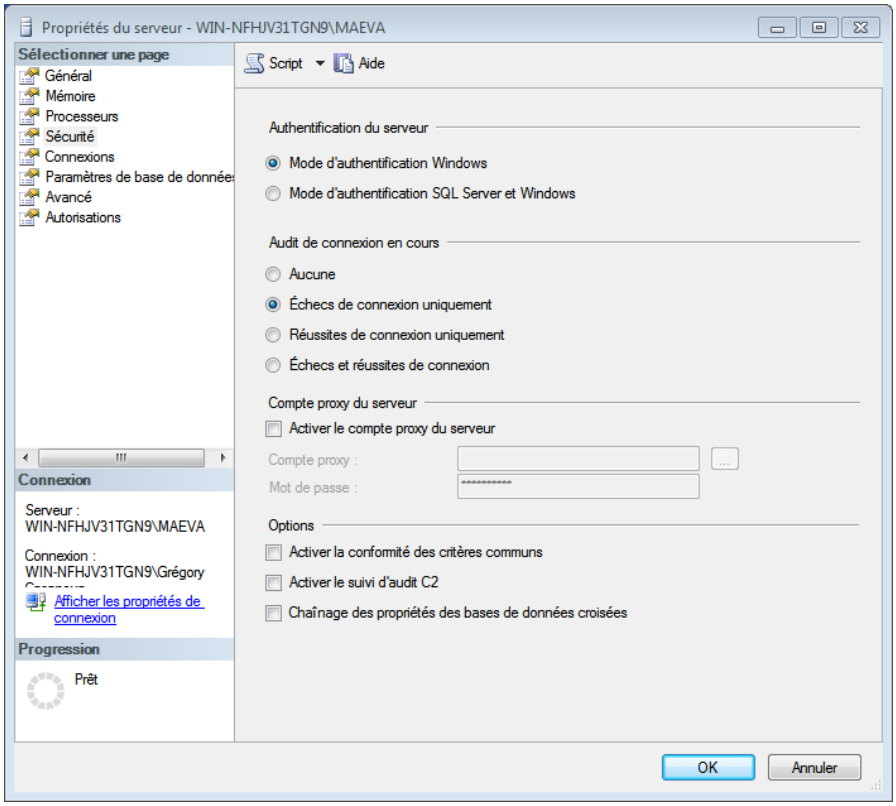
On peut directement choisir le mode de sécurité de SQL Server de deux manières différentes : lors de l'installation de l'instance SQL Server, ou bien directement dans SSMS, sans passer par l'assistant d'installation de SQL Server. Découvrons les différentes manières de changer de mode de sécurité.

2.3.1 Avec l'assistant d'installation

Lors de l'installation d'une instance de SQL Server, il vous sera demandé de préciser quel mode de sécurité vous désirez mettre en place. Choisissez bien entendue le mode qui vous convient.

2.3.2 Avec SSMS

Avec SSMS, il va falloir vous rendre directement dans les propriétés de votre instance afin de modifier le mode de sécurité. Voici la fenêtre sur laquelle vous devez vous rendre afin d'opérer aux modifications :



Dans le menu Sécurité, dans la partie Authentification du Server, choisissez simplement le mode de sécurité à mettre en place. Cliquez sur OK pour valider votre choix.

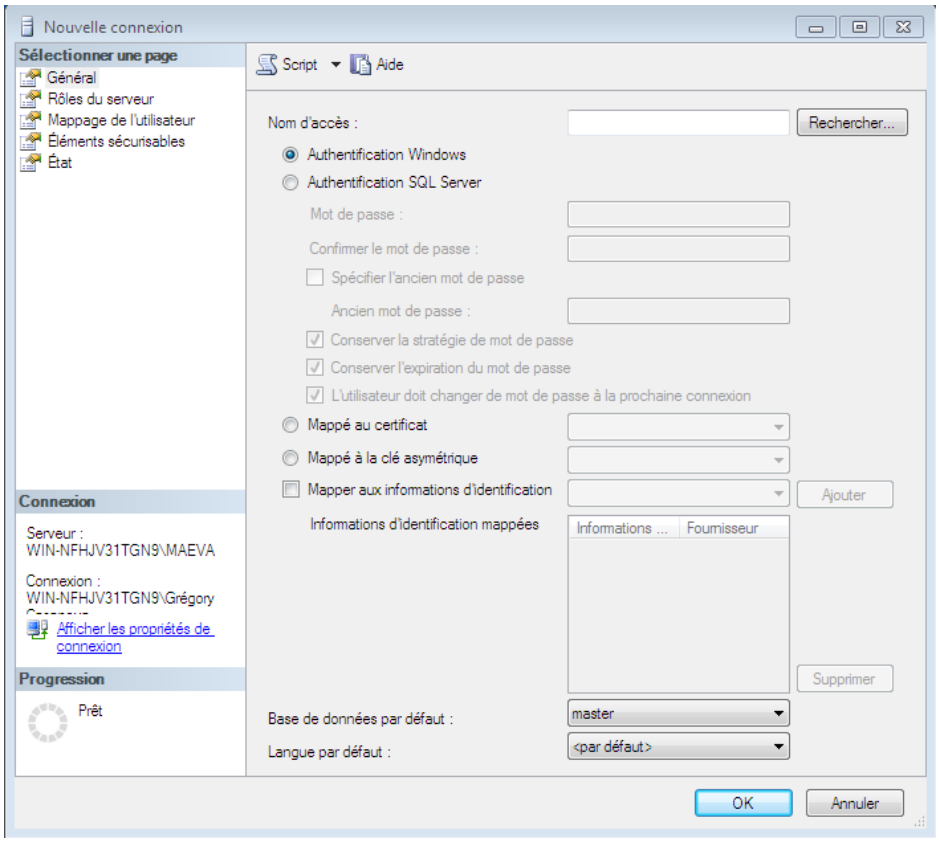
2.4 Gestion des connections à SQL Server

2.4.1 En mode sécurité Windows

Les noms de groupes ou les utilisateurs doivent être les mêmes que sous Windows. Il existe plusieurs façons de créer des connexions au serveur de base de données. Nous allons les détailler.

2.4.1.1 Avec SQL Server Management Studio

Pour créer une connexion en mode graphique, par l'intermédiaire de SSMS, il vous suffit de vous rendre dans **l'explorateur d'objet**, de déployer les nœuds jusqu'à **Sécurité-Connexion**. Dans le menu contextuel de ce nœud, choisir **nouvelle connexion**. La fenêtre suivante apparait :



L'interface graphique SSMS permet tout aussi bien de modifier les propriétés d'une connexion ou bien de la supprimer simplement.

2.4.1.2 *Avec du code T-SQL*

Avec du code T-SQL, la syntaxe de création d'une connexion SQL Server est la suivante :

```
CREATE LOGIN nom_connexion
FROM WINDOWS
WITH DEFAULT_DATABASE=Master
DEFAULT_LANGUAGE=langue
```

L'instruction CREATE LOGIN permet d'annoncer à SQL Server que nous allons créer une connexion. Il est nécessaire de donner le nom de la connexion Windows associée. La clause FROM WINDOWS, permet de dire que le login existe dans le système d'exploitation Windows. Les deux options suivantes permettent quand à elles de choisir, et la base par défaut de la connexion, et la langue par défaut de cette même connexion.

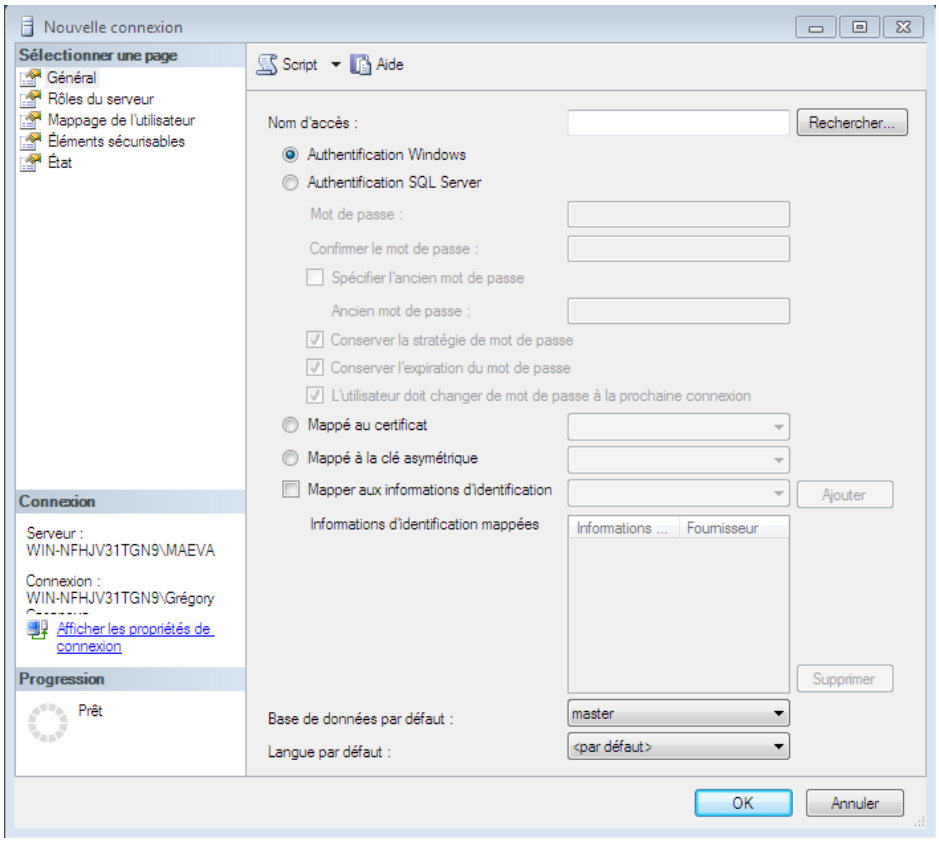
Note : Il est important de noter que les instructions UPDATE et DELETE peuvent être appliquées dans le cas d'un changement de propriété de connexion ou d'une suppression de connexion.

2.4.2 En mode de sécurité mixte

Comme pour toute création d'objet dans SQL Server, et donc comme pour le mode de sécurité Windows, il est possible de créer une connexion par deux moyens. La mise en place de ce mode de sécurité est en tous points semblables à la mise en place du mode de sécurité Windows, à une chose près : il faut mettre en place les règles de gestion des mots de passe.

2.4.2.1 *Avec SQL Server Management Studio*

La procédure est la même que pour le mode de sécurité Windows. Nous ne la détaillerons pas une deuxième fois.



Dans le menu Général, nous choisirons Authentification SQL Server. Il ne vous restera donc plus qu'à donner un nom à la connexion et un mot de passe dans le champs prévu à cet effet. Comme précédemment, l'interface graphique permet de modifier ou bien supprimer une connexion.

2.4.2.2 Avec du code T-SQL

```
CREATE LOGIN nom_connexion
FROM PASSWORD='monmotdepasse'
Motdepasse HASHED MUST_CHANGED,
DEFAULT_DATABASE=Master,
DEFAULT_LANGUAGE=langue,
CHECK_EXPIRATION=ON,
CHECK_POLICY=ON,
CREDENTIAL=Nom
```

L'instruction CREATE LOGIN annonce à SQL Server qu'une nouvelle connexion va être créée. On donne impérativement un nom à cette connexion. La clause FROM permet de donner plusieurs propriétés à cette connexion, qui sont les suivantes :

- **PASSWORD** : Permet de préciser un mot de passe. L'option HASHED, permet de hacher le mot de passe lors de son stockage en fonction de la chaine de caractères précisée avant de mot clé.
- **DEFAULT_DATABASE** : Permet de préciser le nom de la base de données par default.
- **DEFAULT_LANGUAGE** : Permet de préciser un langage par défaut.
- **CHECK_EXPIRATION** : A OFF par défaut. Il n'est possible d'activer cette option que si **CHECK_POLICY** est aussi activé. Elle permet d'appliquer la politique de changement des mots de passe défini sur le serveur.
- **CHECK_POLICY** : A ON par défaut, cette option permet de récupérer au niveau serveur, les options définies pour la politique de sécurité.

- **CREDENTIAL** : Permet de relier la connexion à un credential créé auparavant. Nous verrons par la suite ce qu'est un credential.

2.4.3 Les certificats

SQL Server donne la possibilité de mettre en place des connexions au travers de certificats, qui sont un moyen sur de connexion. Il est possible d'utiliser des certificats déjà contenus dans un fichier, ou bien demander à SQL Server de générer ce dit certificat.

2.4.4 Les credentials

Ces objets permettent à des connexions en mode de sécurité SQL Server d'accéder à des ressources externes au serveur de base de données. Les credentials vont donc contenir un login et un mot de passe Windows qui sera éventuellement rattaché à une connexion de type mixte. Voyons maintenant comment créer un credential :

```
CREATE CREDENTIAL nomducredential
WITH IDENTITY = 'nom', SECRET = 'secret'
```

L'instruction **CREATE CREDENTIAL** permet d'annoncer à SQL Server que nous allons créer un credential. Par la suite, il est impératif de préciser un nom unique qui va nous permettre de reconnaître ce credential parmi les autres credentials présents sur le serveur de base de données. L'option **WITH IDENTITY** permet de préciser le compte Windows à utiliser dans le credential et **SECRET** permet de préciser le mot de passe. Les credentials peuvent être créés avec l'interface graphique dans l'explorateur d'objets en sélectionnant **Nouvelle information d'identification…** dans le menu contextuel du nœud **Information d'identification**.

3 Gestion des utilisateurs de base de données

Maintenant que nous avons définis les connexions (objets de niveau serveur), nous allons montrer comment définir les objets de niveau base de données que sont les utilisateurs. En effet, les droits d'accès, d'écriture, de lecture sur les bases de données ne sont pas attribués aux connexions, mais aux utilisateurs de bases de données. Il faut alors savoir que chaque utilisateur de base de données est affilié à une connexion de niveau serveur. Cependant, il existe des utilisateurs, qui eux ne sont pas associés à une connexion particulière (guest, sys…). Il est donc bon de savoir que si un utilisateur dispose d'une connexion de base de données et que celle-ci n'appartient à aucun utilisateur de base de données capable d'intervenir sur la base, les actions que celui-ci peut effectuer sont très limitées. Il existe alors deux types de droits :

- Ceux qui permettent ou non d'utiliser les objets de telle ou telle base.
- Ceux qui permettent d'utiliser telle ou telle instruction.

Comme nous l'avons dit plus tôt, ces droits sont accordés ou refusés aux utilisateurs de bases de données et non aux connexions serveur.

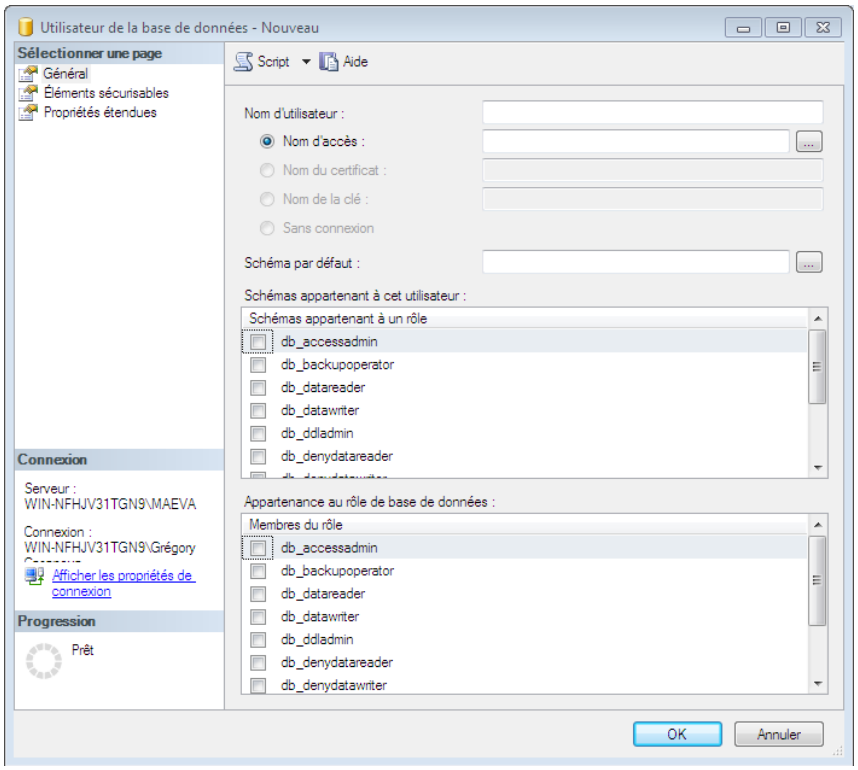
Note : Les utilisateurs de bases de données sont stockés dans la table système sysuser. De plus, à sa création, un utilisateur ne dispose d'aucun droit particulier si ce n'est de celui de guest.

3.1 Création d'utilisateurs de base de données

Il existe deux manières de créer des utilisateurs dans SQL Serveur, comme pour créer la plupart des objets présents dans une base de données. La création d'un utilisateur de base de données va permettre de lier l'utilisateur à la connexion, et donc autoriser l'utilisation de cette base.

3.1.1 Avec SSMS

Pour créer un utilisateur dans SSMS, ouvrez l'explorateur d'objets, et placez-vous sur la base de données concernée. Développez alors le nœud **Sécurité** et placez vous sur le nœud **Utilisateurs**. Dans le menu contextuel du nœud **Utilisateurs**, choisissez **Nouvel utilisateur**. Une nouvelle fenêtre s'ouvre.



Précisez dans un premier temps le nom de l'utilisateur de base de données et dans un second temps, la connexion serveur à lui associer. A partir de cette fenêtre, il est, entre autres, possible d'attacher des schémas à cet utilisateur ainsi que des rôles de base de données. Cliquez alors sur **OK**. Votre utilisateur est créé et mappé sur une connexion existante.

3.1.2 Avec du code T-SQL

En Transact SQL, l'instruction permettant de créer des utilisateurs et de les mapper aux différentes connexions est **CREATE USER**. Voici la syntaxe détaillée de cette instruction :

```
CREATE USER nom
FOR LOGIN connexion
CERTIFICATE nomcertificat
ASYMETRIC KEY nomcle
WITH DEFAULT_SCHEMA = nomschema
```

- **FOR LOGIN** : Définit la connexion où sera mappé l'utilisateur.
- **CERTIFICATE** : Définit le certificat à utiliser. Ne peut pas s'utiliser si une connexion est utilisée dans la clause **FOR LOGIN**.
- **ASYMETRIC KEY** : Définit la clé asymétrique à utiliser à utiliser.
- **WITH DEFAULT SCHEMA** : Nom du schéma de base de données à donner à l'utilisateur.

Note : Il existe des procédures stockées pour créer des utilisateurs et leurs donner des droits sur les bases de données qui sont sp_grantdbaccess et sp_adduser. Celles-ci sont maintenues pour des raisons de compatibilité ascendante, cependant, il est conseillé de ne plus les utiliser car elles sont vouées à disparaitre dans les prochaines versions de SQL Server.

3.2 Modification d'utilisateur de base de données

3.2.1 Avec SQL Server Management Studio

Pour modifier un utilisateur de base de données via l'interface graphique, il vous suffit de déployer l'ensemble des nœuds qui mène à cet utilisateur dans l'explorateur d'objets, afficher le menu contextuel et sélectionner **propriétés**. Vous arriverez alors directement sur la fenêtre de création d'un utilisateur, ou certain champs seront remplis. Il vous suffira de changer les informations que vous désirer modifier.

3.2.2 Avec du code T-SQL

Pour modifier un utilisateur avec du code, nous allons utiliser l'instruction **ALTER USER**. Voici la syntaxe de modification des propriétés d'un utilisateur.

```
ALTER USER nomutilisateur
WITH NAME=nouveaunom,
DEFAULT SCHEMA=nouveauschema
```

3.3 Suppression d'utilisateur de base de données

3.3.1 Avec SSMS

La procédure est la même que pour la modification. Simplement, au lieu d'afficher le menu contextuel et de sélectionner **propriété**, nous allons sélectionner **supprimer**.

3.3.2 Avec du code T-SQL

Pour supprimer un utilisateur, nous allons utiliser l'instruction **DROP USER**. Voici la syntaxe de cette instruction :

```
DROP USER nomutilisateur
```

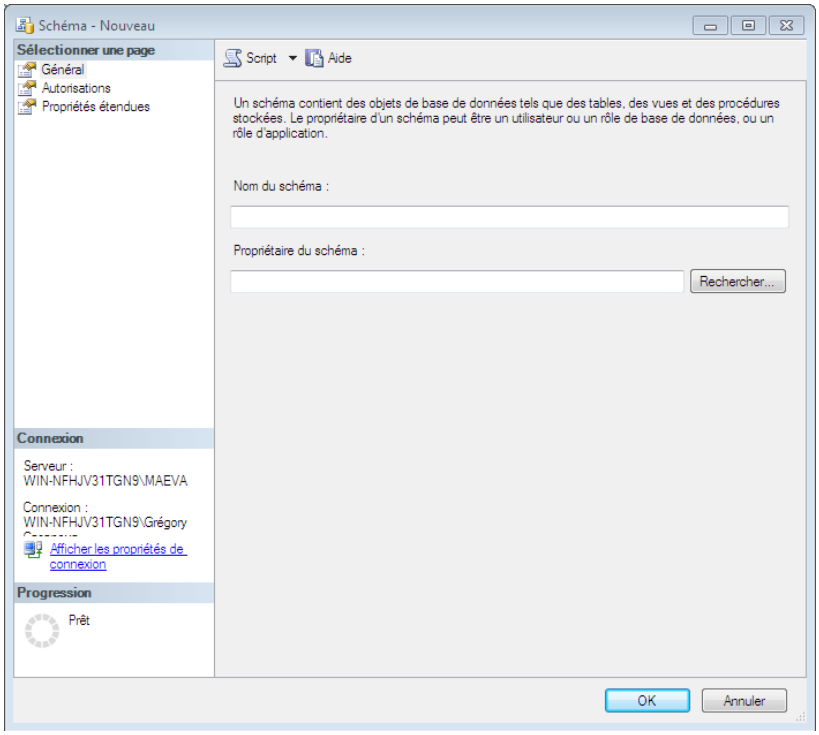
4 Gestion des schémas

Un schéma est un ensemble logique d'objets à l'intérieur des bases de données sur le serveur. Leur but est de faciliter, entre autre, l'échange de données entre les utilisateurs, sans pour autant affecter la sécurité. Concrètement, les schémas permettent une gestion plus aisée des privilèges d'utilisation des objets. Comme nous l'avons vu précédemment, un utilisateur est mappé sur un schéma dès sa création, obligatoirement. Si toute fois, aucun nom de schéma n'est précisé, alors l'utilisateur sera mappé sur **dbo** par défaut. Pour expliquer simplement le fonctionnement des schémas prenons un exemple. Un utilisateur est mappé sur un schéma nommé RU. Pour requêter sur les objets de la base, il pourra écrire directement le nom seul de cet objet si celui-ci est compris dans le schéma sur lequel il est mappé. Dans le cas contraire, l'utilisateur devra préciser le schéma de l'objet, le nom de l'objet et à ce moment là, SQL Server cherchera si le dit utilisateur possède les droit d'utiliser l'objet auquel il tente d'accéder. On peut retenir que les schémas servent essentiellement à faciliter le partage d'information entre les utilisateurs mappés sur ce schéma.

4.1 Création d'un schéma

4.1.1 Avec SSMS

Pour créer un schéma grâce à SSMS, il vous suffit de déployer tous les nœuds jusqu'à **sécurité**, d'afficher le menu contextuel (clic droit) du nœud **schéma** et de sélectionner **Nouveau schéma**. Une nouvelle fenêtre apparait.



4.1.2 Avec du code T-SQL

Voici la syntaxe de création d'un schéma de base de données :

```
CREATE SCHEMA nomschema
AUTHORIZATION nomproprietaire
options
```

Pour créer un schéma de base de données, nous nous servons de l'instruction **CREATE SCHEMA.** Comme tout objet dans une base de données, un schéma doit avoir un nom unique dans la

famille des schémas. Il est donc nécessaire de préciser un nom derrière cette instruction. La clause instruction va nous permettre en revanche de préciser l'utilisateur de base de données qui sera propriétaire du schéma. Pour finir, ce que nous avons représenté par option représente l'espace dans lequel nous pouvons directement faire la définition de nos tables, vues et privilèges rattachés à nos tables.

4.2 Modification d'un schéma

4.2.1 Avec SSMS

Avec l'interface graphique, la manipulation est la même que pour la création d'un schéma. Cependant, il faudra veiller à ne pas créer un nouveau schéma, mais à se rendre dans les **propriétés** du schéma que nous voulons modifier. Il faut en revanche prendre en compte que l'on ne peut pas changer le nom du schéma, seulement les tables, vue qui y sont contenues, les autorisations et le propriétaire du schéma.

4.2.2 Avec du code T-SQL

Pour modifier un schéma avec du code, nous allons utiliser cette syntaxe :

```
ALTER SCHEMA nomschema
TRANSFERT nomobjet
```

Pour modifier un schéma, nous allons utiliser l'instruction **ALTER SCHEMA** suivie du nom du schéma à modifier. Après la clause **TRANSFERT**, on peut spécifier les divers objets à déplacer dans le schéma. Il faut savoir que le nom de ces objets doit être au format suivant : **AncienSchema.NomObjet**.

4.3 Suppression d'un schéma

4.3.1 Avec SSMS

Pour supprimer un schéma de base de données avec SSMS, rendez vous sur le nœud **schéma** de la base de données ou est mappé le schéma. Déployez ce nœud pour laisser apparaitre les schémas existants. Faites un clic droit sur le schéma à supprimer et sélectionnez **Supprimer**.

4.3.2 Avec du code T-SQL

Pour supprimer un schéma, nous allons utiliser l'instruction **DROP SCHEMA,** avec la syntaxe suivante :

```
DROP SCHEMA nomschema
```


5 Gestion des droits

Les droits sont les autorisations qui vont nous permettre de travailler avec notre base de données. Ils sont organisés de façon hiérarchique par rapport aux éléments sécurisables du serveur.

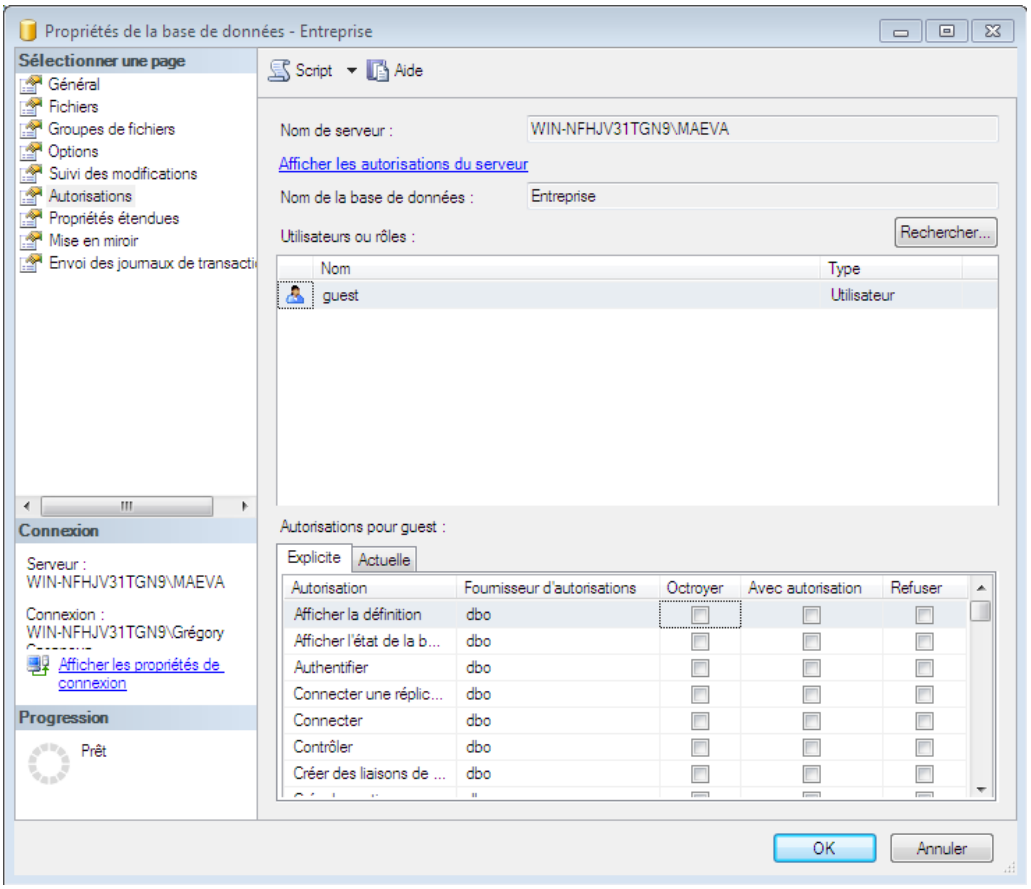
L'attribution des droits peut être faite à tous les niveaux, que se soit au niveau serveur, au niveau base de données ou encore directement sur les schémas ou les objets. Par conséquent, ils peuvent être accordés soit à un utilisateur, soit à une connexion. Il est possible de gérer ces permissions grâce à 3 instructions simples dans SQL Server : **GRANT**, **DENY**, **REVOKE**. **GRANT** permet de donner un privilège, **REVOKE** permet de le retirer si celui-ci a été donné auparavant et **DENY** permet de l'interdire même si il a été donné au travers d'un rôle.

5.1 Droits d'instruction

Ces droits correspondent aux droits qui permettent de créer (mettre à jour, supprimer) de nouveau objets dans la base. Les utilisateurs qui possèdent de tels droits sont donc capable de créer leurs propres tables… Voici les principaux droits disponibles : **CREATE DATABASE**, **CREATE TABLE**, **CREATE FUNCTION**, **CREATE PROCEDURE**, **CREATE VIEW**, **BACKUP DATABASE**, **BACKUP LOG**… Nous allons maintenant apprendre à donner, retirer ou interdire des droits.

5.1.1 Avec SSMS

Avec SSMS, on peut administrer ces droits via la fenêtre propriété de la base de données concernée. Pour cela, rendez-vous sur le menu Autorisations de cette même page.



Pour ajouter des droits à un utilisateur, sélectionnez l'utilisateur dans le premier champ, où nous avons notre utilisateur **Guest**, et octroyez-lui, enlevez-lui ou refusez-lui des permissions en cochant ou décochant les checkbox en conséquence.

5.1.2 Avec du code T-SQL

Pour ajouter, supprimer ou interdire un droit à un utilisateur de base de données, nous allons utiliser les instructions **GRANT**, **DENY** et **REVOKE** comme énoncé précédemment. Voici la syntaxe de don, suppression ou interdiction de droits :

```
--Ajout :
GRANT CREATE TABLE TO Guest

--Suppression :
REVOKE CREATE TABLE FROM Guest

--Interdiction :
DENY CREATE TABLE  TO Guest
```

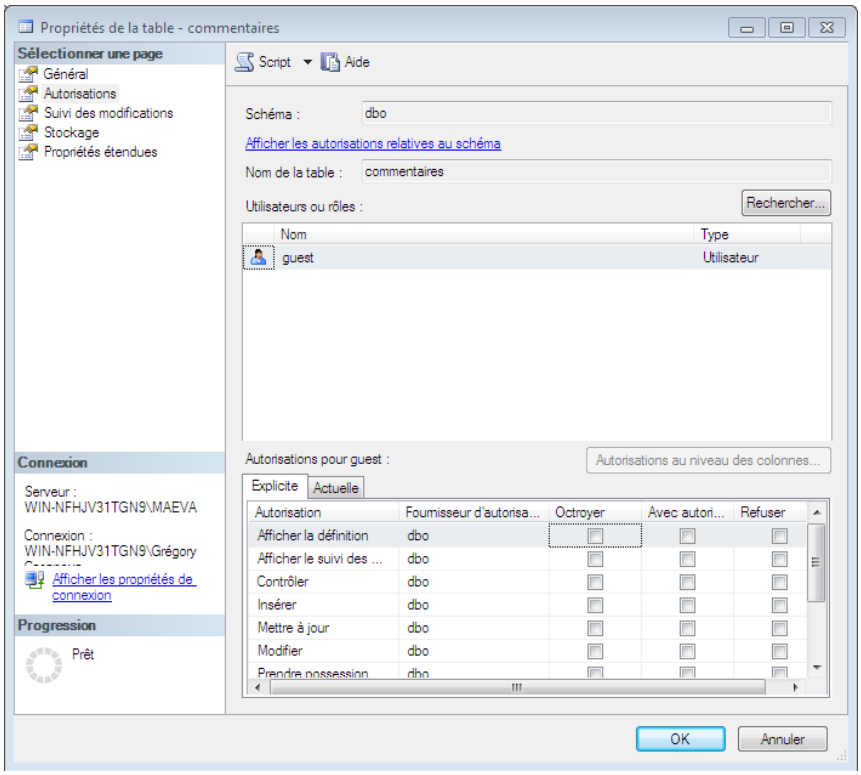
5.2 Droits d'utilisation

Les droits d'utilisation permettent de déterminer si un utilisateur possède les droits pour travailler sur les objets, par exemple lire les informations, insérer des données… Dans le cas général, c'est le propriétaire de l'objet qui définit les droits d'utilisation. En ce qui concerne les droits de lecture et de mise à jour des données, le propriétaire possède la faculté de choisir quelles colonnes l'utilisateur peut lire ou mettre à jour. Voici la liste des principaux droits d'utilisation : **INSERT**, **UPDATE**, **SELECT**, **DELETE**, **EXECUTE**.

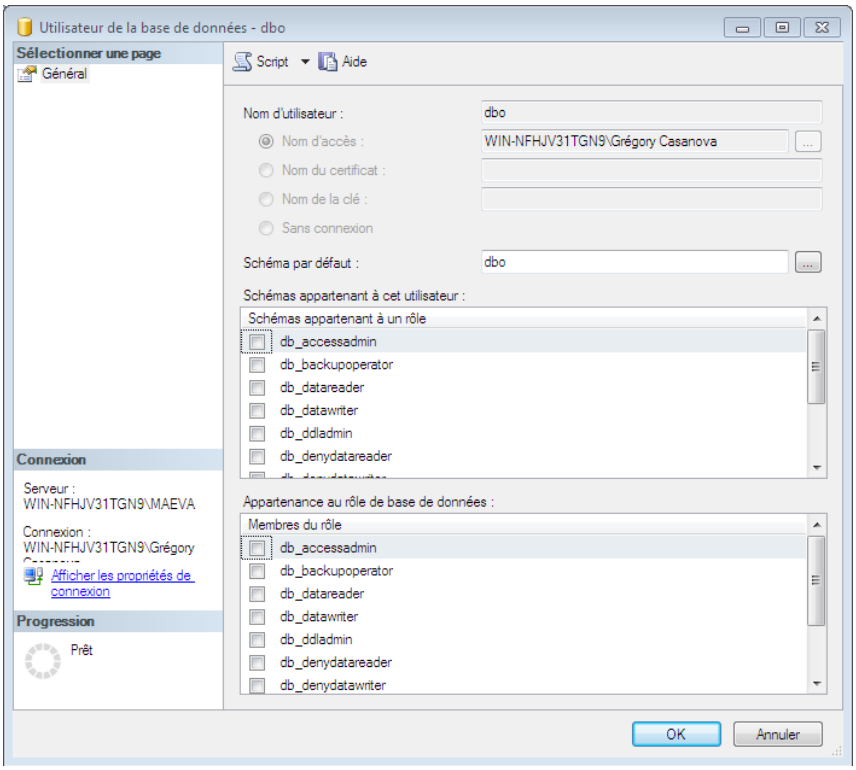
5.2.1 Avec SSMS

Les droits d'utilisation peuvent être gérés à deux niveaux dans SQL Server, au niveau utilisateur et au niveau objet. Dans SSMS, pour les deux niveaux cités, la gestion de ces droits ce fait grâce à la fenêtre de propriété. Voici les deux cas de fenêtre dans lesquelles vous pouvez changer les droits d'utilisation.

- Au niveau objet :



- Au niveau utilisateur :



5.2.2 Avec du code T-SQL

Pour donner des droits d'utilisation ou les retirer avec du code T-SQL, nous allons retrouver les trois mots clés **GRANT**, **DENY** et **REVOKE**. La syntaxe en revanche sera changeante. La voici :

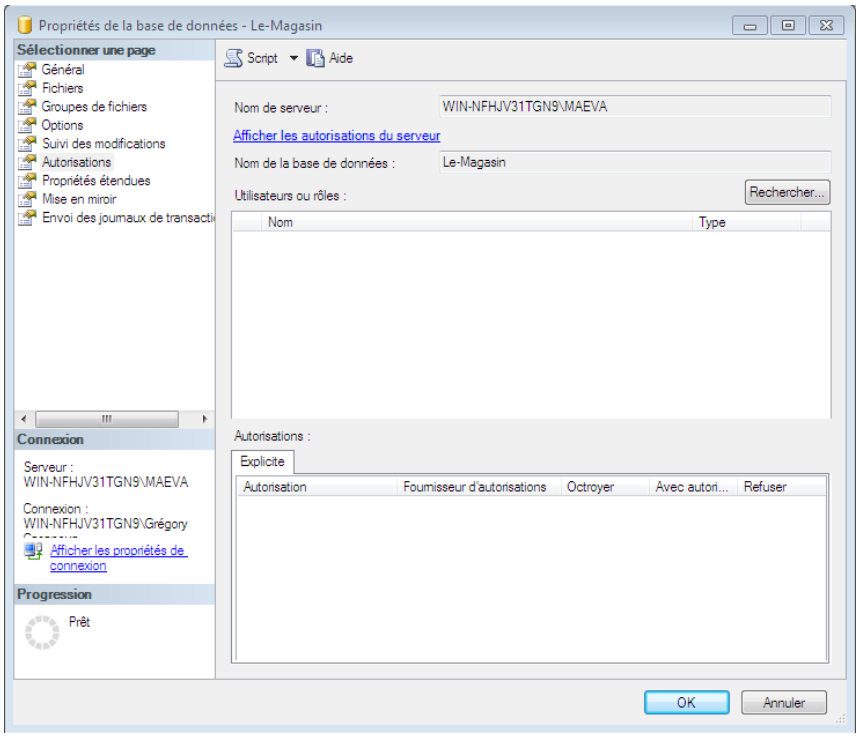
```
--Ajout :
GRANT CREATE TABLE TO Guest

--Suppression :
REVOKE CREATE TABLE FROM Guest

--Interdiction :
DENY CREATE TABLE  TO Guest
```

5.3 Droits au niveau base de données

Les droits au niveau des bases de données vont donner des droits aux utilisateurs qui ne seront valables que sur une base de données précise. Au niveau base de données, il est possible de donner des droits à un utilisateur, à un schéma, à une assembly ou encore à un objet service broker. Ces droits peuvent être attribués de plusieurs manières, soit par du code Transact SQL, soit par les propriétés de la base de données.

5.3.1 Avec SSMS

Pour accorder des droits, vous procèderez de la façon suivante. Déroulez la totalité des nœuds qui mènent à votre base de données, affichez le menu contextuel de cette base de données en effectuant un clic droit et sélectionnez **propriété**. Vous aboutissez sur la fenêtre suivante (Pensez à vous rendre dans la partie **Autorisations**) :



Il vous suffit d'ajouter des utilisateurs dans la première partie nommée **Utilisateurs ou rôles** et de leur donner des droits dans la seconde partie nommée **Autorisations**. Cliquez enfin sur OK pour valider vos choix.

5.3.2 Avec du code T-SQL

Avec du code T-SQL, nous utiliserons la syntaxe suivante :

```
GRANT permissiondebasededonnées
TO utilisateur
AS groupeourole
```

- **Permissiondebasededonnées** : La ou les permissions de base de données à accorder à l'utilisateur de base de données.
- **Utilisateur** : L'utilisateur qui va recevoir les permissions de base de données précisées après l'instruction **GRANT**.
- **Groupeourole :** Constitue le contexte de sécurité qui va nous permettre d'accorder les privilèges.

Note : Il est évident que nous pouvons utiliser l'instruction GRANT, aussi bien que les instructions DENY et REVOKE, pour interdire et supprimer des droits aux utilisateurs.

Nous avons noté plus haut que nous précisons le nom des permissions de bases de données après l'instruction principale **GRANT**. Nous allons lister la quasi-totalité de ces permissions ci-après :

- **ALTER**
- **ALTER ANY APPLICATION ROLE**
- **ALTER ANY ASSEMBLY**
- **ALTER ANY ASYMMETRIC KEY**
- **ALTER ANY CERTIFICATE**
- **ALTER ANY CONTRACT**
- **ALTER ANY DATABASE DDL TRIGGER**

- **ALTER ANY DATABASE EVENT NOTIFICATION**
- **ALTER ANY DATASPACE**
- **ALTER ANY FULLTEXT CATALOG**
- **ALTER ANY MESSAGE TYPE**
- **ALTER ANY REMOTE SERVICE BINDING**
- **ALTER ANY ROLE**
- **ALTER ANY ROUTE**
- **ALTER ANY SCHEMA**
- **ALTER ANY SERVICE**
- **ALTER ANY SYMMETRIC KEY**
- **ALTER ANY USER**
- **AUTHENTICATE**
- **BACKUP DATABASE**
- **BACKUP LOG**
- **CHECKPOINT**
- **CONNECT**
- **CONNECT REPLICATION**
- **CONTROL**
- **CREATE AGGREGATE**
- **CREATE ASSEMBLY**
- **CREATE ASYMMETRIC KEY**
- **CREATE CERTIFICATE**
- **CREATE CONTRACT**
- **CREATE DATABASE**
- **CREATE DATABASE DDL EVENT NOTIFICATION**
- **CREATE DEFAULT**
- **CREATE FULLTEXT CATALOG**
- **CREATE FUNCTION**
- **CREATE MESSAGE TYPE**
- **CREATE PROCEDURE**
- **CREATE QUEUE**
- **CREATE REMOTE SERVICE BINDING**
- **CREATE ROLE**
- **CREATE ROUTE**
- **CREATE RULE**
- **CREATE SCHEMA**
- **CREATE SERVICE**
- **CREATE SYMMETRIC KEY**
- **CREATE SYNONYM**
- **CREATE TABLE**
- **CREATE TYPE**
- **CREATE VIEW**
- **CREATE XML SCHEMA COLLECTION**
- **DELETE**
- **EXECUTE**
- **INSERT**
- **REFERENCES**

- **SELECT**
- **SHOWPLAN**
- **SUBSCRIBE QUERY NOTIFICATIONS**
- **TAKE OWNERSHIP**
- **UPDATE**
- **VIEW DATABASE STATE**
- **VIEW DEFINITION**

5.4 Droit au niveau serveur

Les privilèges au niveau serveur s'attribuent de la même manière que ceux de niveau base de données, simplement, ils ne sont pas les mêmes. En revanche, ces privilèges ne sont pas attribués à un utilisateur, mais à une connexion. Nous ne présenterons donc pas les façons d'attribuer ces droits puisque elles sont les mêmes que pour le niveau base de données, mais nous allons les citer :

- **ADMINISTER BULK OPERATIONS**
- **ALTER ANY CONNECTION**
- **ALTER ANY CREDENTIAL**
- **ALTER ANY DATABASE**
- **ALTER ANY ENDPOINT**
- **ALTER ANY EVENT NOTIFICATION**
- **ALTER ANY LINKED SERVER**
- **ALTER ANY LOGIN**
- **ALTER RESOURCES**
- **ALTER SERVER STATE**
- **ALTER SETTINGS**
- **ALTER TRACE**
- **AUTHENTICATE SERVER**
- **CONNECT SQL**
- **CONTROL SERVER**
- **CREATE ANY DATABASE**
- **CREATE DDL EVENT NOTIFICATION**
- **CREATE ENDPOINT**
- **CREATE TRACE EVENT NOTIFICATION**
- **EXTERNAL ACCESS ASSEMBLY**
- **SHUTDOWN**
- **UNSAFE ASSEMBLY**
- **VIEW ANY DATABASE**
- **VIEW ANY DEFINITION**
- **VIEW SERVER STATE**

6 Les rôles

On peut dire que les rôles sont des sortes de groupements de droits. Il existe trois niveaux d'actions pour les rôles : Server, Base de données et Application. L'utilité des rôles réside dans le fait

qu'il est plus simple d'attribuer des droits à des rôles puis d'attribuer des rôles à des utilisateurs ou des connexions plutôt que d'attribuer directement des droits à ces derniers. Il faut donc retenir que les rôles sont des **ensembles de droits** qui portent un nom et qu'on peut les attribuer aux utilisateurs.

Pour faciliter la gestion des droits, SQL Server propose des droits dits fixes. En effet, ils sont prédéfinis et donc non modifiables. Ils sont définis à deux niveaux : Server et Base de données. En revanche, les rôles définis par l'utilisateur de SQL Server atteignent deux niveaux différents : Base de données et Application. Au final, un utilisateur de base de données peut avoir accès à des droits de quatre manières différentes : grâce aux droits de la connexion qu'il utilise, grâce aux rôles fixes, grâce aux rôles créés par l'utilisateur et enfin grâce aux droits qui lui ont été directement affectés.

Note : Il existe un rôle qui est public et qui est attribué à tous les utilisateurs de base de données et à toutes les connexions. Ce rôle ne peut pas être enlevé à aucun des utilisateurs. En revanche, il peut être modifié. Il faut simplement prendre en compte que TOUS les utilisateurs auront les droits ajoutés à public.

6.1 Les rôles prédéfinis

Rôles serveur :

- **Sysadmin** : Administrateur du serveur.
- **Serveradmin** : Permet de configurer les paramètres niveau serveur.
- **Setupadmin** : Permet d'exécuter certaines procédures stockées et d'ajouter des serveurs liés.
- **Securityadmin** : Permet de gérer les connexions serveur.
- **Processadmin** : Permet de gérer les traitements au sein de SQL Server.
- **Dbcreator** : Permet de créer ou modifier des bases de données.
- **Diskadmin** : Permet de gérer les fichiers sur le disque.
- **Bulkadmin** : Permet d'exécuter l'instruction **BULK INSERT**.

Rôles base de données :

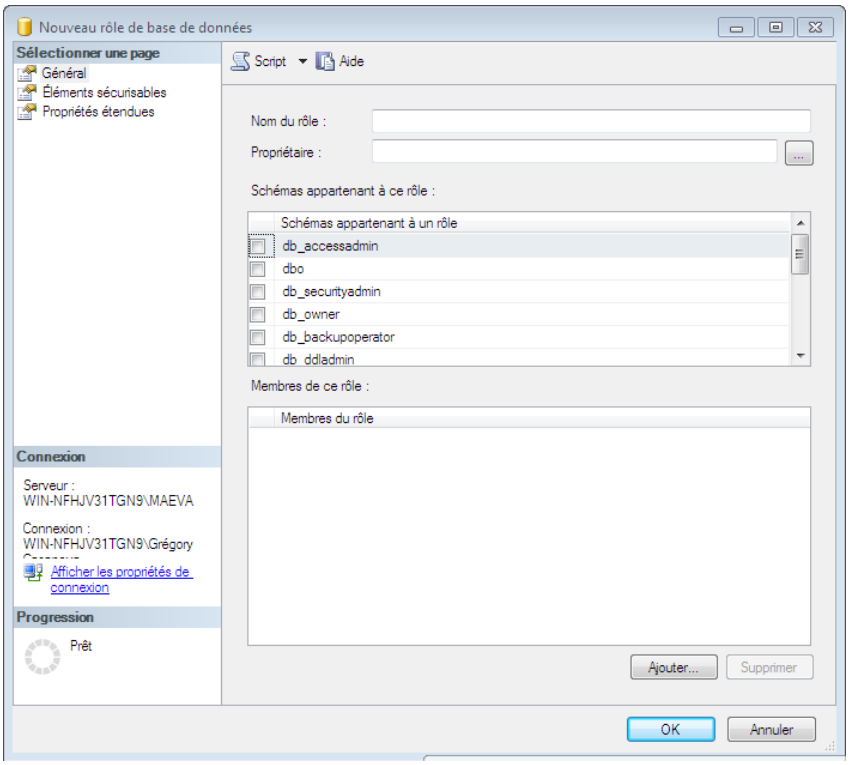
- **Db_owner** : Equivalent à propriétaire base de données.
- **Db_accessadmin** : Permet d'ajouter et supprimer des utilisateurs de base de données.
- **Db_datareader** : Permet d'utiliser l'instruction **SELECT**.
- **Db_datawriter** : Permet les instructions **INSERT**, **UPDATE** et **DELETE**.
- **Db_ddladmin** : Permet les opérations sur les objets de base de données.
- **Db_securityadmin** : Permet de gérer les éléments de sécurité sur la base de données.
- **Db_backupoperator** : Permet l'utilisation des backups.
- **Db_denydatareader** : Interdit l'instruction **SELECT**.
- **Db_denydatawriter** : Interdit l'écriture sur la base de données.

6.2 Les rôles définis par l'utilisateur

Il est possible de définir ses propres rôles afin de faciliter l'administration des droits dans SQL Server. Logiquement, on créera un rôle dit personnalisé lorsque plusieurs utilisateurs doivent avoir les mêmes droits et que ces droits n'existent pas dans les rôles prédéfinis. Les rôles peuvent être accordés soit directement à un utilisateur, soit à un autre rôle.

6.2.1 Avec SQL Server Management Studio

Pour créer un nouveau rôle, procédez de la manière suivante. Déployez successivement le nœud de votre base de données puis le nœud sécurité. Affichez alors le menu contextuel du nœud rôles et sélectionnez **Nouveau rôle**. Vous obtenez cette nouvelle fenêtre :



Il vous suffit alors de préciser le nom et le propriétaire du rôle. En revanche, vous n'êtes pas obligé de préciser les autres informations de suite. Vous pouvez les modifier ultérieurement en vous rendant dans les propriétés du rôle voulu.

6.2.2 Avec du code T-SQL

Voici la syntaxe de création d'un rôle personnalisé :

```
CREATE ROLE nomdurole
AUTHORIZATION propriétaire
```

Nous allons bien entendu créer un rôle grâce à l'instruction **CREATE ROLE**. Il est alors nécessaire de préciser le nom du schéma après cette instruction pour assurer son unicité au sein de la base de données sur le serveur. On peut enfin préciser un propriétaire grâce à la clause **AUTHORIZATION**.

7 Conclusion

Dans ce chapitre, vous avez appris à mettre en place la sécurité de vos bases de données au travers des droits. D'autres formes de sécurités sont possible, entre autres, l'encryptions transparente des données. Ce chapitre est désormais terminé. Si toutefois vous avez besoin de précisions sur le cours, ou bien sur les objets de bases… n'hésitez pas à consulter les cours de développement de base de données, ou bien à poster sur le forum de ce même site (http://forums.dotnet-france.com/forums/).