

SQL Server - Introduction à la Gestion des Droits

par Pierre Caboche ([autres articles](#))

Date de publication : 19 Novembre 2007

La gestion des privilèges (= droits) en base de données est très importante. Malheureusement, cette phase est souvent reléguée au second plan, bien après la modélisation de données, c'est-à-dire quand il est déjà trop tard. Cet article a pour but de présenter une méthode simple pour gérer les privilèges sous SQL Server.

Introduction

1 - Généralités

- 1.1 - Pourquoi gérer les droits ?
- 1.2 - Les erreurs les plus fréquentes
- 1.3 - Gestion des droits dans le processus de développement

2 - Notions

- 2.1 - Notions de base
- 2.2 - Gestion des droits : principes
 - 2.2.1 - Implémentation naïve
 - 2.2.2 - Implémentation plus efficace
 - 2.2.3 - Qu'est-ce qu'un rôle exactement ?
- 2.3 - Gestion des droits : règles
- 2.4 - Notions supplémentaires
 - 2.4.1 - Login / User
 - 2.4.2 - Les Schémas
 - 2.4.3 - Chaînage de propriétaire (ownership chaining)
 - 2.4.4 - Vues et procédures stockées

3 - Management Studio

- 3.1 - Gestion des droits sous Management Studio
- 3.2 - Créer un login + user
- 3.3 - Créer et affecter des rôles
- 3.4 - Affecter les droits sur un objet

4 - En pratique

- 4.1 - Répertorier les droits pour une application .Net existante
- 4.2 - Affecter les droits suffisants
- 4.3 - Connaître les droits au niveau applicatif

Conclusion

Remerciements

Introduction

Après un bref résumé expliquant **pourquoi** il est nécessaire de définir une politique de sécurité, nous nous attarderons plus longuement à expliquer **comment** implémenter la gestion des droits de manière simple et efficace, afin que le développeur puisse se concentrer sur les nombreuses autres tâches qui lui incombent. Tout au long de l'article, nous expliquerons pourquoi la gestion des droits devrait se faire **le plus tôt possible** lors du développement.

Bien que cet article concerne principalement SQL Server (cela permet de donner des exemples précis et de présenter certaines spécificités de SQL Server), la plupart des notions abordées dans cet article (concernant la sécurité et la gestion des droits) peuvent s'appliquer à quasiment tous les SGBD.

1 - Généralités

1.1 - Pourquoi gérer les droits ?

La gestion des droits d'une base de données est un domaine assez vaste et il existe de nombreux articles sur le sujet. Voici les aspects qui me semblent les plus importants :

- **sécurité** : se protéger contre les attaques

Les réseaux informatiques sont de plus en plus souvent la cible d'attaques. Si, malgré les protections mises en place, un individu arrive à s'introduire dans votre base de données (en volant un mot de passe ou en se faisant passer pour une autre personne), il faut que son rayon d'action soit le plus limité possible

- **protection** : empêcher les utilisateurs d'effectuer certaines actions

L'erreur est humaine, c'est un fait, et il peut arriver que des utilisateurs parfaitement habilités à utiliser la base de données modifient par erreur certaines données qui devraient normalement être protégées. Une gestion correcte des droits permet de se prémunir contre ce genre de désagrément en empêchant l'exécution de certaines tâches

- **confidentialité** : restreindre l'accès à certaines données

Il est inconcevable que l'ensemble des salariés d'une entreprise aient accès aux données concernant, par exemple, les salaires. La gestion des droits permet de restreindre la visibilité de certaines données (seules les données strictement nécessaires doivent être accessibles)

Si vous recherchez des informations plus générales sur la gestion des droits, je vous recommande l'excellent article d'Hugo Etiévant : **la gestion des droits en base de données**.

1.2 - Les erreurs les plus fréquentes

Voici les erreurs les plus fréquentes en matière de gestion des droits en base de données :

- laisser les applications se connecter avec le login "sa" (administrateur)
- pire: laisser les applications se connecter avec le login "sa" et sans mot de passe !
- gérer les permissions entièrement au niveau applicatif quand il est possible de définir certains droits sur les objets de la base de données (tables, vues, procédures)
- accéder directement au contenu des tables (en lecture et écriture) plutôt que par le biais de vues ou de procédures stockées
- ne pas définir de rôles
- ne pas gérer les droits suffisamment tôt

Certaines de ces erreurs sont parfois très dures à récupérer par la suite (ex: remplacer l'accès à des tables par des vues, restreindre les permissions...), d'où l'intérêt de se soucier très tôt de la gestion des droits.

Si vous avez déjà commis une ou plusieurs de ces erreurs, nous vous recommandons vivement la lecture de cet article.

Si vous n'avez commis aucune de ces erreurs (félicitations !), cet article pourra quand même vous intéresser et vous pourrez en faire profiter certains de vos collègues.

1.3 - Gestion des droits dans le processus de développement

En fait, gérer les droits est finalement très simple si on s'y prend suffisamment tôt.

A chaque création d'un nouvel objet dans la base de données (ex: une procédure stockée), il suffit de suivre les étapes suivantes :

- 1 on crée le nouvel objet
- 2 on fait la liste des rôles qui ont besoin d'accéder à cet objet et on affecte les droits en conséquence
- 3 on teste si les personnes concernées ont accès à l'objet (ex: peuvent exécuter la procédure stockée).

Si une exception est levée à ce moment là, il manque des droits

Au cours de ce processus, de nouveaux profils d'utilisateurs peuvent apparaître (des utilisateurs doivent avoir des droits particuliers, que les autres n'ont pas), d'où la nécessité de définir de nouveaux **rôles** (et d'attribuer ces rôles à différents utilisateurs).

Dans la section suivante, nous expliquerons plus en détails ces différentes notions (rôles, utilisateurs...)

2 - Notions

2.1 - Notions de base

Pour accéder à une base de données, un **utilisateur** utilise une **connexion**. Un utilisateur peut être soit une personne physique, soit une application (script, batch).

Une base contient de nombreux **objets** (tables, vues, procédures stockées, fonctions#). Pour entreprendre certaines **actions** sur ces objets (consulter, exécuter, modifier#) l'utilisateur doit avoir les **privileges** (aussi appelés **droits**) nécessaires. L'utilisateur peut obtenir ces droits de manière directe ou indirecte.

Le principe de base est donc finalement très simple. La mise en place peut être un peu plus complexe, comme nous allons le voir.

2.2 - Gestion des droits : principes

Dans cette section, nous allons comparer 2 implémentations pour la gestion des droits.

Pour que les exemples soient plus parlant, nous allons considérer une base de données avec :

- 10 utilisateurs aux droits différents suivant leur rôle
- 60 tables, vues, procédures stockées#

2.2.1 - Implémentation naïve

Pour prendre conscience des problèmes que peut représenter une mauvaise gestion des droits, nous allons imaginer l'implémentation suivante, très simple (mais très naïve) :

- pour chaque objet (table, vue, procédure stockée) on affecte les droits de manière individuelle à chaque utilisateur

Cette implémentation posera les problèmes suivants :

- à chaque fois que l'on ajoute un objet dans la base, il faut affecter les droits pour chacun des 10 utilisateurs
- à chaque ajout d'un utilisateur, il faudra affecter les droits sur chacun des 60 objets de la base de données

En résumé, avec une implémentation aussi naïve , la gestion des droits est loin d'être aisée.

2.2.2 - Implémentation plus efficace

Maintenant, considérons l'implémentation suivante :

- on définit la notion de rôle (= groupe d'utilisateurs partageant les mêmes droits)
- un utilisateur appartient à un ou plusieurs rôle(s)
- pour chaque objet de la base, on affecte les droits aux différents rôles. Tous les membres d'un rôle donné hériteront des droits du rôle

Avec cette implémentation :

- à chaque fois que l'on ajoute un objet dans la base, il suffit d'affecter les droits à 1 rôle pour que tous les utilisateurs du rôle bénéficient des droits d'accès
- lorsqu'on crée un nouvel utilisateur, il suffit de l'ajouter à 1 groupe d'utilisateurs (rôle) pour qu'il bénéficie de tous les droits du rôle
- pour changer les droits d'un utilisateur, il suffit de changer son appartenance aux différents rôles

La gestion des droits devient alors nettement plus facile !

2.2.3 - Qu'est-ce qu'un rôle exactement ?

Un **rôle**, c'est un ensemble de responsabilités.

Dans une entreprise, les employés ont diverses responsabilités. Chacune de ces responsabilités s'accompagne d'un certain nombre de tâches et l'entreprise doit fournir à ses employés les moyens nécessaires pour accomplir leur mission. Par ailleurs les employés peuvent avoir plusieurs responsabilités et donc cumuler les tâches.

En base de données, le principe est similaire : un rôle représente un ensemble de responsabilités au sein de l'application. Chacune de ces responsabilités s'accompagne d'un certain nombre de tâches (sous la forme de procédures stockées, par exemple). Pour accomplir ces différentes tâches, les membres d'un rôle doivent pouvoir accéder à différentes données (au travers de vues, procédures stockées, etc.), ce qui implique d'avoir les droits nécessaires pour accéder à ces données.

Lors de l'analyse, on doit donc identifier:

- les utilisateurs (users)
- les responsabilités (rôles)
- les tâches à effectuer et les moyens d'accès aux données (procédures stockées, vue, tables...)

2.3 - Gestion des droits : règles

Concernant la gestion des droits, quelques règles sont à respecter :

- **Identifier les rôles**
ainsi que les différents acteurs au sein de l'application
- **Donner le minimum de droits**

Pour chaque rôle, on ne doit fournir que les droits nécessaires et suffisants à l'exécution des différentes tâches.

- **Interdire l'accès direct aux tables**

Les tables sont le support de données et leur contenu ne devrait pas être accessible directement. Par exemple, une table "salarié" peut contenir des informations personnelles qui ne doivent être accessibles qu'à un petit groupe d'individus. C'est pourquoi on accède au contenu d'une table au moyen de vues, procédures stockées, fonctions, etc. Ceci permet également de spécifier si l'accès aux données se fait en lecture seule ou si elle autorise les modifications.

En résumé, pour accéder au contenu d'une table on crée une vue (ou une procédure stockée, ou une fonction) pour laquelle on affecte les droits aux différents rôles contenant plusieurs utilisateurs. Ceci permet un meilleur contrôle des accès.

- **Gérer les droits le plus tôt possible**

Plus on gère les droits de manière précoce, plus cette gestion est aisée car l'ajout de droits se fait au fur et à mesure du développement, et non de manière hâtive à la fin.

Au début du développement, quelques rôles sont clairement identifiés et d'autres seront ajoutés par la suite. Idem pour les utilisateurs. A chaque ajout de fonctionnalité dans le programme, on assigne les droits nécessaires pour son exécution (si les droits ne sont pas suffisants, on s'en rend très vite compte : une exception est levée). De cette façon, on gère les droits très facilement et avec un effort réduit.

Dans la suite de cet article, nous allons découvrir comment, dans le cas d'une application .Net au développement bien avancé, identifier les appels aux objets de la base de données (vues, procédures stockées#) en vue d'assigner les droits.

2.4 - Notions supplémentaires

2.4.1 - Login / User

En SQL Server, on distingue d'un part la notion de **login** et d'autre part la notion de **user**.

Le **login**, c'est ce qui permet de se connecter à un serveur SQL Server.

Cependant un même serveur peut accueillir plusieurs bases de données et dans chacune de ces bases on définit différents **users** pour la gestion des droits. Il est ensuite nécessaire de faire le lien entre les logins et les users (ce que nous verrons par la suite)

2.4.2 - Les Schémas

Auteur: **rudib**

Le schéma est un objet de base, au même titre que les tables. Il s'agit d'un objet intermédiaire entre le serveur et la table, qui est défini par la norme SQL. SQL Server implémentait très imparfaitement les schémas en 2000, et cette lacune a été corrigée en 2005.

Vous pouvez simplement considérer le schéma comme un espace de nom. C'est une sorte de coquille vide qui accueille des objets de base de données, tels que tables, vues, procédures, fonctions. A l'intérieur d'un schéma, un nom d'objet doit être unique. Vous pouvez par contr utiliser le même nom d'objet dans une base, s'il est dans un schéma différent.

Les schémas sont utiles pour la gestion des privilèges, car ils permettent de les attribuer sur l'étendue du schéma. Par exemple, permettre en une instruction l e privilège d'exécution sur toutes les procédures appartenant à un schéma.

2.4.3 - Chaînage de propriétaire (ownership chaining)

Auteur: *rudib*

Un schéma appartient à un utilisateur de la base.

Dès lors qu'un objet de code, comme une fonction, un déclencheur, une procédure ou une vue, fait appel à des objets qui appartiennent au même utilisateur (donc, soit au même schéma, soit à des schémas qui ont le même propriétaire), les privilèges spécifiques de ces objets sous-jacents ne sont pas vérifiés par SQL Server. Il estime simplement que si un utilisateur a créé tous les objets, il sait ce qu'il fait dans son code, et donc, il ne va pas vérifier toute la chaîne des privilèges sur les objets.

Il s'agit d'une optimisation de performance, mais aussi d'une facilité pour gérer l'accès aux objets à travers procédures et vues : on donne des privilèges à ces objets seulement, et pas aux tables sous-jacentes, et donc on contrôle plus finement les accès (voir paragraphe suivant).

2.4.4 - Vues et procédures stockées

Les vues et procédures stockées présentent plusieurs avantages pour le développeur.

Par exemple, les procédures stockées permettent de définir une suite d'instructions qui seront exécutées côté serveur (meilleure performance, limite la bande passante), en une seule transaction, et de manière paramétrable.

Du point de vue de la gestion des droits, l'utilisation de vues et de procédures stockées permettent un meilleur contrôle des droits. Par exemple, on autorise des utilisateurs à exécuter un procédure stockée mais on lui interdit l'accès direct aux données de la table.

De cette façon, le contrôle des droits d'accès s'effectue au niveau des vues et procédures stockées (qui représentent les "tâches" à effectuer au sein de la base).

Bien sûr, ceci n'est qu'une façon parmi d'autres de contrôler les droits en base de données. Par ailleurs, dans certains cas, il peut être tout à fait justifié d'autoriser les utilisateurs à accéder directement au contenu des tables. C'est le cas par exemple si, dans une application cliente .Net, on utilise des Dataset pour mettre à jour les tables.

Dans tous les cas, il est important d'essayer de n'affecter aux utilisateurs que les droits strictement nécessaires à l'exécution leurs différentes tâches. Toute digression de cette règle doit être justifiée (ex: contraintes techniques).

3 - Management Studio

Cette section porte sur les principales opérations liées à la gestion des droits sous Management Studio.

3.1 - Gestion des droits sous Management Studio

Voici un résumé de l'interface permettant la gestion des droits sous SQL Server Management Studio. Tout se passe dans l'Explorateur d'Objets (*Object Explorer*) :

- dans l'arborescence, pour un serveur donné, on trouve le répertoire "Security". C'est là que se gère la sécurité au niveau **du serveur**, notamment la liste des logins
- pour un serveur donné, on a plusieurs bases de données (répertoire "Databases") et pour chaque base de données on trouve également un répertoire "Security". C'est là que se gère la sécurité au niveau **de la base**, notamment les users, rôles, etc

Le fait que le mot "Security" apparaisse à la fois au niveau de la base et du serveur peut prêter à confusion au début.

Au niveau du serveur, on gère :

- les logins
- les rôles serveur

Au niveau de la base, on gère :

- les users
- les rôles
- les schémas

C'est dans les logins que l'on définit les liens entre un login et un user (voir paragraphe suivant).

3.2 - Créer un login + user

A présent nous allons créer à la fois un nouveau login (pour que l'utilisateur puisse se connecter au serveur), un user dans la base de données (pour l'affectation des droits) et le lien entre le login et le user.

Sous SQL Server Management Studio :

- après vous être connecté au serveur désiré, celui-ci apparaît dans l'Object Explorer (Explorateur d'Objets).
- dans l'explorateur d'objets, pour le serveur désiré, allez dans Security/Logins (Sécurité/Connexions)
- effectuez un click droit et sélectionner "New login" (Nouvelle connexion)
- une fenêtre apparaît. Saisissez alors les infos sur le login (nom, mot de passe, etc)
- dans le menu de gauche de la fenêtre, cliquez sur "User mapping" (Mappage utilisateur)
- sélectionnez les bases pour lesquelles vous souhaitez créer un utilisateur lié à la connexion
- cliquez sur "Ok"

A présent les login et user sont créés, le lien (mapping) entre les deux est correctement effectué, reste à affecter les droits.

3.3 - Créer et affecter des rôles

Pour créer un rôle :

- sélectionnez la base de données concernée
- allez dans Security/Roles
- on distingue les rôles utilisateurs (User Roles) des rôles applicatifs (Application Roles).
- sur "User Roles", effectuez un click droit. Cliquez sur "New Role"
- une fenêtre apparaît. Saisissez les informations concernant le rôle (nom du rôle...)
- cliquez sur "Ok"

Pour ajouter des utilisateurs à un rôle :

- dans Security/Roles/User Roles, sélectionnez rôle
- effectuez un click droit. Cliquez sur "Properties"
- cliquez sur "Add"
- cliquez sur "Browse"
- sélectionnez les utilisateurs à ajouter au rôle et cliquez sur "Ok"
- cliquez sur "Ok" pour enregistrer les changements

3.4 - Affecter les droits sur un objet

Pour affecter les droits :

- sélectionnez un objet (table, vue, procédure...)
- effectuez un click droit. Cliquez sur "Properties"
- sélectionnez "Permissions"
- cliquez sur "Add"
- cliquez sur "Browse"
- sélectionnez les utilisateurs/rôles à ajouter au rôle et cliquez sur "Ok"
- **ne pas oublier** : pour chaque utilisateur/rôle sélectionnez les droits (SELECT, EXECUTE...). Si vous ne le faites pas, l'utilisateur (ou le rôle) n'aura aucun droit et sera donc supprimé de la liste
- cliquez sur "Ok" pour enregistrer les droits

Tout au long de cet article (et surtout dans la section 2), nous avons parlé de l'importance des rôles. C'est dans cette phase que l'on affecte des droits à un rôle plutôt qu'à des utilisateurs isolés.

4 - En pratique

4.1 - Répertorier les droits pour une application .Net existante

Voici un scénario courant lors de la mise en place d'un Système d'Information :

- au départ, on analyse les besoins utilisateur
- ensuite on identifie les différentes fonctionnalités (les tâches que l'application doit effectuer) et éventuellement quelques rôles (qui fait quoi dans l'application)
- lors du développement, on concentre ses efforts sur la mise en place des différentes fonctionnalités plutôt que sur la gestion des droits (les utilisateurs ayant une notion plutôt vague de ce qu'est la "sécurité", les chefs de projet préfèrent généralement les se concentrer sur le nombre de fonctionnalités implémentées)
- une fois le développement bien avancé, le chef de projet fait de la gestion des droits une priorité (parce que le mot "sécurité" revient à la mode au sein du service)

A partir de ce moment là, le développeur est confronté aux difficultés suivantes :

- le nombre d'objets en base et dont il faut sécuriser l'accès est généralement important
- l'analyse des besoins en termes de sécurité est souvent superficielle
- le développeur n'a que peu de temps pour implémenter les droits et a souvent d'autres choses à faire (tester l'application et corriger les bugs, entre autres#)

Toute ressemblance avec des situations existantes ou ayant existé serait loin d'être fortuite# C'est pourquoi il est nécessaire de disposer d'une méthode pour affecter les droits nécessaires et suffisant à une application existante. C'est justement le but de ce paragraphe, dans le cadre d'une application .Net.

La première étape consiste à dresser la liste des appels aux différents objets de la base de données à partir de l'application.

Sous Visual Studio 2005, on procède ainsi :

- 1 repérez dans le code un appel au constructeur de *SqlCommand*
- 2 effectuez un click droit au dessus du constructeur de *SqlCommand*. Un menu contextuel apparaît. Sélectionnez "Find All References" (1) dans ce menu
- 3 la liste de tous les appels au constructeur de *SqlCommand* apparaît alors.

Ceci permet de dresser la liste des requêtes SQL et des appels à des procédures stockées effectués par l'application, ainsi que le contexte dans lequel sont ces appels apparaissent. Connaissant cela, il est possible d'affecter les droits de manière précise.

La deuxième étape consiste à affecter les droits, c'est-à-dire que l'on définit quels rôles ont le droit d'accéder à quelles vues / tables (2) / procédure stockée.

Important :

Ceci tend à montrer qu'il est beaucoup plus facile de mettre en place une gestion des droits efficace si on la définit dès le départ et qu'on l'implémente au fur et à mesure du développement. Si on s'y prend trop tard, cela devient beaucoup plus difficile et laborieux (il faut relire tout le code à la recherche d'appels à des requêtes SQL à un moment critique où le temps vient à manquer).

4.2 - Affecter les droits suffisants

Le but ici est d'affecter les droits en respectant la règle : "Pour chaque rôle, on ne doit fournir que les droits nécessaires et suffisants à l'exécution des différentes tâches".

Pour cela, nous allons utiliser deux connexions différentes à la base de données : La première nous servira à affecter les droits La deuxième nous servira à vérifier les droits d'un utilisateur particulier

1. Ouvrez une première connexion (utilisateur dont on veut vérifier les droits) :

- cliquez sur "Connect"
- entrez les informations de connection (serveur, login, mot de passe...)

Sauf cas particuliers, les objets qui apparaissent sont ceux pour lesquels l'utilisateur connecté a les droits d'accès.

Si l'utilisateur a le droit VIEW DEFINITION (octroyé par l'instruction GRANT VIEW DEFINITION), l'utilisateur aura en plus la possibilité de voir des objets auxquels il n'a pas accès (c'est un des cas particuliers mentionné précédemment).

Si un objet n'apparaît pas, c'est que l'utilisateur n'a pas les droits suffisants. Nous allons donc ouvrir une autre connexion afin d'affecter les droits manquants.

2. Ouvrez une deuxième connexion (utilisateur avec pouvoirs) :

- cliquez sur "Connect"
- loguez-vous en tant que super utilisateur ("sa", ou autre utilisateur avec pouvoirs)

Grâce à cette connexion, il est possible de gérer les droits (voir section 3 sur la Gestion des droits)

Après avoir modifié les droits, vous pouvez vérifier que l'utilisateur en question a bien accès aux objets désirés. Pour cela, sélectionnez la première connexion (utilisateur dont on veut vérifier les droits), et rafraîchissez l'affichage (bouton droit > Propriétés > Refresh).

Grâce à cette technique (utilisation de deux connexions : l'une pour affecter les droits, l'autre pour vérifier que les droits affectés sont les bons), il est possible d'affecter les droits de manière très fine (et de ne donner que les droits nécessaires).

4.3 - Connaître les droits au niveau applicatif

Dans SQL Server, il existe un certain nombre de vues systèmes qui permettent de connaître les droits affectés. Par exemple grâce aux vues **sys.database_principals** et **sys.database_role_members**, il est possible de savoir à quels rôles appartiennent les différents utilisateurs.

Ceci peut se révéler utile au niveau applicatif : en fonction des différents rôles attribués, on peut activer ou désactiver certaines options dans l'interface utilisateur.

Conclusion

Avec cet article, j'espère vous avoir convaincu de l'importance de la gestion des droits en base de données, mais surtout que la mise en place d'**une bonne gestion des droits est grandement facilitée si celle-ci est mise en place dès le début du développement.**

Je vous ai également donné les moyens de faciliter la gestion des droits:

- définition de rôles
- ajout des droits, création des rôles au fur et à mesure du développement
- vérification que pour un rôle donné, on a bien les droits nécessaires et suffisants à l'exécution des différentes tâches

Maintenant, vous devriez avoir toutes les clefs en main pour mettre en place une gestion correcte des droits sans que cela nuise à la productivité.

Remerciements

Je tiens à remercier **rudib** et **fadace** pour leur contribution lors de la relecture de l'article.

1 : Dans les versions de Visual Studio antérieures à 2005, on ne dispose malheureusement pas de la fonctionnalité "Find All References", ce qui complique énormément la tâche.

2 : Comme expliqué en section 2.4.4, il est souvent préférable d'utiliser des vues et procédures stockées plutôt que d'accéder directement au contenu des tables (on peut ainsi définir plus finement les droits d'accès). Néanmoins, dans la pratique, ce cas est très répandu.