Secteur Tertiaire Informatique
Filière étude - développement

Activité « Développer la persistance des données »

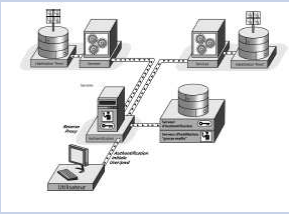
Sécurité dans SQL Server

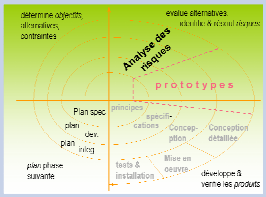
Accueil

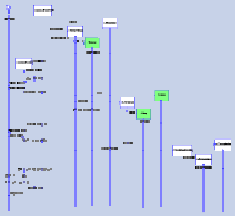
Apprentissage

Période en entreprise

Evaluation







Code barre

SOMMAIRE

I APERCU DE LA SECURITE SQL SERVER 2005

Dans SQL Server 2005, la sécurité par défaut a été implémentée : à l'installation, SQL Server 2005 ne permet pas à un outil client, localisé sur une autre machine de se connecter, un grand nombre de fonctionnalités sont désactivées (utilisation de la CLR) et il faudra une opération explicite de l'administrateur pour les mettre en place.

La sécurité au niveau du déploiement a été mise en place par le biais de l'outil **Configuration de la surface d'exposition de SQL Server** qui permet à l'administrateur de voir quels sont les services ouverts et de paramétrer ou non les accès.

Lorsqu'un utilisateur souhaite accéder à des données dans une base de données, trois niveaux d'accès sont vérifiés :

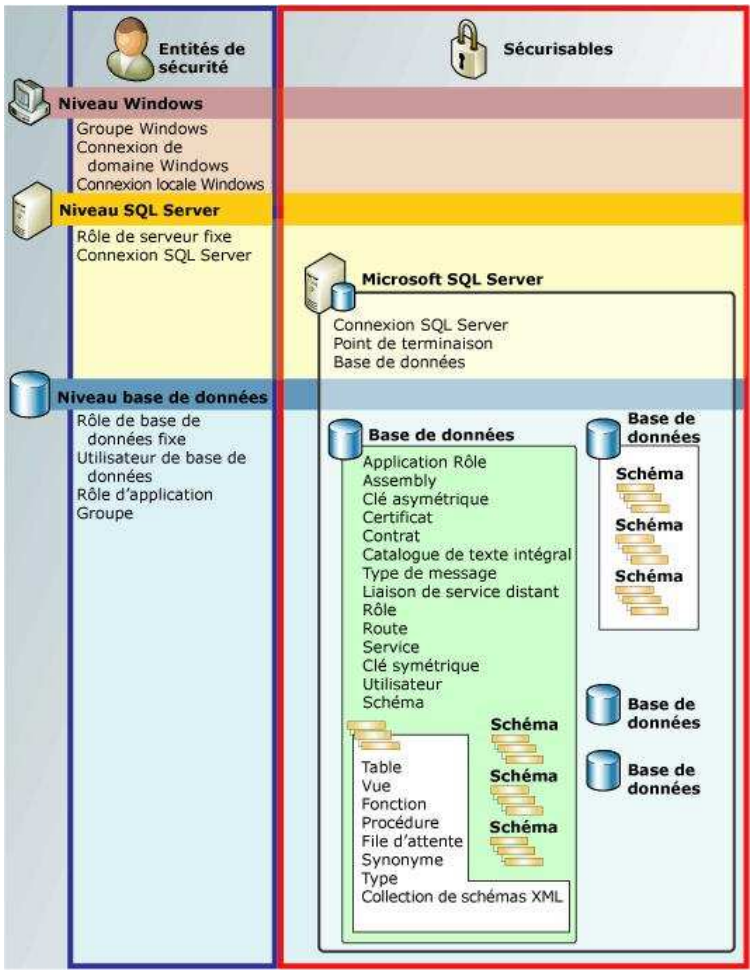
- L'accès de l'utilisateur à SQL Server est authentifié par rapport à un compte de connexion (par Windows ou SQL Server)
- L'accès de l'utilisateur à une base de données particulière est conditionné par son appartenance à un compte d'utilisateur ou à un rôle de base de données défini avec le compte de connexion
- L'accès de l'utilisateur à des objets ou travaux au sein d'une base est conditionné par les permissions octroyées au compte ou au rôle.

SQL Server 2005 accorde une grande attention, en matière de sécurité, aux éléments principaux et sécurisables.

Les **entités de sécurité** sont des comptes de connexion qui disposent d'un accès au serveur SQL.

Les trois éléments sécurisables de niveau supérieur sont **serveur**, **base de données** et **schéma**. Chacun d'eux en contient d'autres, qui à leur tour peuvent contenir d'autres éléments sécurisables. Ces hiérarchies imbriquées sont désignées par le terme de *portées*.

Des autorisations sont accordées aux entités de sécurité pour qu'elles puissent travailler avec les éléments sécurisables.


Entités de sécurité
Sécurisables
Niveau Windows
Groupe Windows
Connexion de domaine Windows
Connexion locale Windows
Niveau SQL Server
Rôle de serveur fixe
Connexion SQL Server
Microsoft SQL Server
Connexion SQL Server
Point de terminaison
Base de données
Niveau base de données
Rôle de base de données fixe
Utilisateur de base de données
Rôle d'application
Groupe
Base de données
Application Rôle
Assembly
Clé asymétrique
Certificat
Contrat
Catalogue de texte intégral
Type de message
Liaison de service distant
Rôle
Route
Service
Clé symétrique
Utilisateur
Schéma
Table
Vue
Fonction
Procédure
File d'attente
Synonyme
Type
Collection de schémas XML
Schéma
Schéma
Schéma
Base de données
Schéma
Schéma
Schéma
Base de données
Base de données

II ACCES AU SERVEUR SQL

II.1 CONFIGURATION

Pour pouvoir utiliser une base de données dans SQL Server, un utilisateur doit pouvoir se connecter au serveur à l'aide d'un compte de connexion.

Seuls les administrateurs système ou chargés de la sécurité sont habilités à ajouter des comptes de connexion.

Le modèle de sécurité de SQL Server comporte deux modes d'authentification :

- **L'authentification Windows**
  Les comptes de connexion sont définis à partir des utilisateurs et groupes Windows : SQL Server autorise les utilisateurs Windows à se connecter au serveur SQL, et ne gère pas les mots de passe
- **L'authentification mixte**
  Dans ce mode, SQL Server vérifie l'existence de l'utilisateur qui demande la connexion, et vérifie son mot de passe.

Utilisez l'authentification Windows lorsque les accès à la base de données ne sont qu'internes à l'entreprise ; la .sécurité mixte est recommandée lorsque des utilisateurs externes sont autorisés à se connecter à la base.

Avec la sécurité mixte, SQL Server vérifie d'abord si l'utilisateur dispose d'une connexion SQL Server valide. Dans ce cas, la connexion est acceptée. Si le mot de passe est erroné, la connexion est refusée. SQL Server ne vérifie les connexions Windows que si la connexion est invalide.

Depuis la version 2005, l'authentification se fait de manière cryptée : les informations de login et password sont transmises encryptées.

L'authentification Windows offre un certain nombre d'avantages par rapport à l'authentification SQL Server. Elle offre plus particulièrement :

- Plus de fonctionnalités (cryptage des mots de passe, expiration, longueur minimale)
- l'ajout des groupes d'utilisateur dans SQL Server en ajoutant un seul compte de connexion
- la possibilité de se connecter rapidement sans avoir besoin d'entrer d'autres paramètres.

SQL Server 2005 peut appliquer les mêmes stratégies de complexité et de mot de passe que celles utilisées dans Windows, pour les mots de passe utilisés dans SQL Server. Cette fonctionnalité, qui dépend d'une API particulière, n'est disponible que dans Windows Server 2003.

La mise en œuvre du mode d'authentification peut être paramétrée dans SQL Server Management Studio (**Propriétés du serveur** dans l'explorateur d'objets/ page **Sécurité**).

Pour créer un compte de connexion, on utilisera une instruction Transact-SQL **CREATE LOGIN (**qui remplace la procédure système **sp_addlogin** de SQL Server 2000) ou SQL Server Management Studio.

Dans SQL Server Management Studio, après avoir développé le dossier Sécurité de votre serveur, vous créerez une **Nouvelle connexion** dans le menu contextuel **Connexions**.

Syntaxe générale sous Transact-SQL : Authentification Windows

CREATE LOGIN *nom_login*

FROM WINDOWS,
| CERTIFICATE *certname,*
| ASYMMETRIC KEY *asym_key_name,*
[WITH DEFAULT_DATABASE **=** *nom_database* | DEFAULT_LANGUAGE **=** *nom_langage]*

Les connexions peuvent être basées sur des certificats, qui permettent d'identifier de façon sure l'utilisateur en se basant sur des informations telles une clé publique, une identification et une période de validité, ou sur des clés asymétriques (voir les instructions CREATE CERTIFICATE et CREATE ASYMETRIC KEY dans l'aide en ligne de SQL Server).

Exemple 1:

Création d'un compte intégré SQL Server pour l'utilisateur Windows **camille** de la machine locale ou du domaine **ECRPSI** (les options de stratégie ne peuvent être définies car elles sont définies au niveau Windows)

CREATE LOGIN [**ECRPSI\camille**]
FROM WINDOWS,
WITH DEFAULT_DATABASE = **AdventureWorks**

Le nom de connexion est donné sous la forme domaine\utilisateur ; C'est sur la base de données par défaut (master par défaut) que sera positionné l'utilisateur utilisant cette connexion ; des droits d'utilisation resteront à définir ; la langue n'est à préciser que si elle est distincte de celle définie sur le serveur SQL.

Syntaxe générale sous Transact-SQL : Authentification mixte

CREATE LOGIN *nom_login*
WITH PASSWORD **=** m*ot_passe* [HASHED] [MUST_CHANGE],
SID = sid,
DEFAULT_DATABASE **=** *nom_database* ,
DEFAULT_LANGUAGE **=** *nom_langage,*
| CHECK_EXPIRATION = { ON | OFF},
| CHECK_POLICY **=** { ON | OFF},
[CREDENTIAL **=** *nom_credential*]

SID = sid,
Spécifie le GUID de la nouvelle connexion SQL Server. Si cette option n'est pas sélectionnée, SQL Server attribue automatiquement un GUID.

CHECK_EXPIRATION = { ON | OFF}
Spécifie que les règles d'expiration des mots de passe Windows doivent être imposées sur cette connexion.

CHECK_POLICY **=** { ON | OFF}
Spécifie que les règles des mots de passe Windows doivent être imposées sur cette connexion.

CREDENTIAL **=** *nom_credential*
Des informations d'identification (compte Windows, mot de passe) peuvent être créées pour permettre à un utilisateur connecté en mode mixte d'accéder à des ressources extérieures au serveur SQL (voir l'instruction CREATE CREDENTIAL dans l'aide en ligne de SQL Server).

Exemple 2: Création du compte camille avec impératif que le mot de passe cRoc1822 respecte les règles d'expiration et de stratégie Windows, soit modifié à la première connexion, la base de données par défaut étant AdventureWorks.

```
CREATE LOGIN camille
    WITH PASSWORD='cRoc1822' MUST_CHANGE,
    DEFAULT_DATABASE = AdventureWorks,
    CHECK_POLICY =ON,
    CHECK_EXPIRATION = ON
```

L'instruction Transact-SQL **ALTER LOGIN** permettra de modifier des options de la connexion (nom de connexion, mot de passe, activation / désactivation, ….), l'instruction **DROP LOGIN** permettra de supprimer une connexion.

II.2 COMPTES DE CONNEXION PAR DEFAUT

Plusieurs connexions sont configurées par défaut à savoir Administrateurs locaux, Administrateur local, sa, NETWORK SERVICE et SYSTEM.

Ces comptes disposent de tous les droits sur toutes les bases de données.

III ACCES AUX BASES DE DONNEES

III.1 CONFIGURATION

Après avoir créé un compte (identité de serveur) pour la connexion au serveur SQL, il convient de lui accorder des droits d'accès à une ou plusieurs bases de données.

Une connexion ne pourra exécuter des opérations sur une base que s'il existe un compte utilisateur défini sur cette base, associé à la connexion.

Pour attribuer un compte d'utilisateur à une connexion, on pourra utiliser une instruction Transact-SQL CREATE USER (qui remplace la procédure système **sp_grantdbaccess** de SQL Server 2000) ou SQL Server Management Studio.

Une entrée est ajoutée à la table **sys.server_principals** de la base de données à laquelle l'accès a été ajouté.

Dans SQL Server Management Studio,

Après avoir développé le dossier **Sécurité** de votre serveur, dans le menu contextuel de la **connexion considérée/ Propriétés /** page **Mappage de l'utilisateur**, vous cocherez les noms des bases auxquelles cette connexion peut accéder.

Ou

Après avoir développé le dossier de la base de données considérée, dans le menu contextuel du dossier Utilisateurs vous cliquerez sur **nouvel Utilisateur ; dans le nom d'accès,** vous cliquerez sur le nom de votre choix après avoir ouvert la boite de dialogue **Sélectionner la connexion**.

Syntaxe sous Transact-SQL :

CREATE USER *nom_user*
[
{
FOR LOGIN *nom_login*
| CERTIFICATE *nom_cert*
| ASYMMETRIC KEY *nom_clé_asym*
}
| WITHOUT LOGIN
]
[WITH DEFAULT_SCHEMA **=** *nom_schéma*]

USER nom_user,
Spécifie le nom de l'utilisateur.

LOGIN *nom_login*
Spécifie le nom de la connexionà laquelle l'utilisateur est associé.

WITH DEFAULT_SCHEMA **=** *nom_schéma*
Indique le nom du premier schéma à explorer pour rechercher un objet de nom incomplet.

WITHOUT LOGIN
L'utilisateur ne sera pas associé à une connexion existante ; il ne pourra agir qu'au sein de sa propre base.

Exemple : Création de l'utilisateur camille dans la base de données AdventureWorks.

```
USE AdventureWorks
CREATE USER camille FOR LOGIN camille WITH DEFAULT_SCHEMA = Sales
```

Si les noms de l'utilisateur et du compte sont les mêmes, la clause FOR LOGIN n'est pas nécessaire. Si la clause optionnelle DEFAULT_SCHEMA n'est pas spécifiée, le schéma par défaut de l'utilisateur est dbo.

L'instruction Transact-SQL **ALTER USER** permettra de renommer le compte utilisateur ou de modifier son schéma par défaut, l'instruction **DROP USER** permettra de supprimer un compte.

La procédure système **sp_who** permet de connaître pour chaque connexion un certain nombre d'informations (nom de connexion, nom de l'ordinateur client, nom de la base de données courante).

III.2 COMPTES UTILISATEUR PAR DEFAUT

Chaque base de données de SQL Server dispose de 2 comptes utilisateurs par défaut : **dbo** et **guest.**

Dans toutes les bases de données, le compte de connexion **sa** et les membres du rôle **administrateurs,** qui disposent par défaut du rôle **sysadmin,** sont mappés à un compte d'utilisateur spécial appelé **dbo** : tous les objets crées par ces comptes appartiendront à l'utilisateur **dbo,** tous les objets crées par des utilisateurs autres appartiennent à l'utilisateur qui les créent.

Le compte d'utilisateur **guest** (invité) permet aux connexions dépourvues de compte d'utilisateur d'accéder à une base de données. Par défaut, **guest** existe dans la base de données *model*, donc existera dans toutes les nouvelles bases de données (création à partir de *model*).Ce compte peut être ajouté ou supprimé dans toutes les bases de données, sauf master et tempdb, car la plupart des utilisateurs accèdent à ces bases en tant qu'invité.

A noter :

- L'utilisateur guest est membre du rôle de serveur public et hérite des autorisations de ce rôle
- Pour qu'une base soit accessible à des invités, l'utilisateur guest doit lui être affecté et activé
- L'utilisateur guest n'est utilisé que lorsqu'un compte d'utilisateur a accès à SQL Server mais pas à la base de données

Lorsque SQL Server est installé, le programme d'installation n'attribue pas de mot de passe au profil **sa** : Vous devez attribuer un mot de passe à ce profil pour empêcher les utilisateurs non autorisés de se connecter à SQL Server avec des privilèges d'administrateur (à effectuer avec SQL Server Management Studio). Il est parfois conseillé de supprimer ou désactiver ce compte.

NETWORK SERVICE et SYSTEM sont des comptes locaux intégrés du serveur. Que des connexions soient créées ou non pour ces comptes dépend de la configuration du serveur par exemple, si le serveur est configuré en tant que générateur de rapports, on dispose d'un compte Network Service et la connexion, membre d'un rôle RSExecRole est utilisé pour gérer le schéma Report Server.

Tous les objets system sont contenus dans des schémas nommés **sys** ou **INFORMATION_SCHEMA**. Ce sont 2 schémas particuliers crées dans chaque base. Ces vues des schémas fournissent une vue du système interne des métadonnées de tous les objets de cette base.

Chaque base de données inclut deux entités qui apparaissent comme des utilisateurs dans les affichages catalogue : Les utilisateurs sys et INFORMATION_SCHEMA, requis par SQL Server fournissent des renseignements sur ces vues et ne peuvent être ni modifiés, ni supprimés.

IV ACCEDER AUX OBJETS

Un schéma, on l'a vu lors de la création de la base de données, est un ensemble logique d'objets à l'intérieur de la base.

Exemple : Création du schéma Gescli dans la base Ventes, de propriétaire Gestion.

```
USE Ventes;
CREATE SCHEMA Gescli AUTHORIZATION Gestion
```

Un même utilisateur de base de données peut-être le propriétaire de plusieurs schémas.

Lorsqu'on attribue un schéma par défaut à un utilisateur, il n'est pas nécessaire que cet utilisateur fasse référence au nom du schéma lorsqu'il accède à un objet de ce schéma.

En cas de conflit d'accès entre plusieurs objets de différents schémas, SQL Server utilisera en priorité l'objet qui existe dans le schéma **sys**, puis l'objet qui existe dans le **schéma par défaut** de l'utilisateur, puis l'objet qui existe dans le schéma **dbo**.

Exemple 1: Si le schéma par défaut de Camille est Sales, et si la table Contact existe dans Sales et dbo, c'est la table de Sales qui sera utilisée.

Exemple 2: Si le schéma par défaut de Camille est Sales, et si la table Contact existe dans HumanResources et dbo, c'est la table de dbo qui sera utilisée.

Exemple 3: Si le schéma par défaut de Camille est Sales, et si la table Contact n'existe que dans HumanResources, une erreur se produira.

V AUTORISATIONS

Les autorisations déterminent les actions permises aux utilisateurs dans SQL Server ou dans une base de données.

Elles sont gérées par SQL Server avec 3 instructions :

- **GRANT** permet d'accorder des autorisations
- **REVOKE** permet de retirer des autorisations précédemment accordées
- **DENY** interdit une action particulière

Elles sont accordées en fonction de l'Id de connexion, de l'appartenance à des groupes, et de l'appartenance à des rôles. La modification des définitions de la base ou l'accès à des données suppose l'établissement préalable des autorisations nécessaires qui peuvent être de trois types :

V.1 AUTORISATIONS D'OBJET

Les autorisations peuvent être gérées pour des objets spécifiques, d'un type particulier ou appartenant à un schéma spécifique, mais dépendant de la portée (voir schéma).

Au niveau serveur, les autorisations portent sur les points de terminaison, les connexions et les rôles.

Au niveau base de données, les autorisations portent sur des rôles d'application, des assemblies, des clés et certificats, des schémas, des tables …

L'accès à tous ces objets est contrôlé en octroyant, refusant ou annulant la possibilité d'utiliser certaines instructions ou procédures stockées ; par exemple, on peut accorder à un utilisateur le droit de lire (SELECT) mais lui refuser le droit d'ajout (INSERT), de mise à jour (UPDATE) et de suppression (DELET). Il est également possible de préciser certaines colonnes de la table dans le cas de lecture ou mise à jour : cette possibilité n'est pas conseillée : il est préférable de passer par une vue ou une procédure stockée pour limiter la gestion des droits.

Syntaxe sous Transact-SQL :

GRANT {ALL | *permission* [(colonne, [**,**...n])]}
ON [OBJECT ::] *objet* [**,**...n]
TO *nom_user* [**,**...n]
[WITH GRANT OPTION]

ALL,
Permet d'accorder tous les privilèges d'utilisation de l'objet.

objet,
Correspond au nom complet de l'objet (nom_schéma.nom_objet).

permission,
Spécifie l'autorisation pouvant être accordée.

Nom_user
Nom de l'utilisateur à qui l'autorisation sera accordée.

WITH GRANT OPTION
Cet utilisateur pourra accorder cette autorisation à d'autres utilisateurs.

Exemple : Accorder à l'utilisateur camille tous les privilèges d'utilisation de la table sales.Store, et l'autorisation Select sur la table sales.SalesPerson

```
USE AdventureWorks
GRANT ALL ON sales.Store TO camille
GRANT SELECT ON sales.SalesPerson TO camille
```

Cette autorisation pourra par la suite être retirée par l'instruction REVOKE.

Syntaxe sous Transact-SQL :

REVOKE [GRANT OPTION FOR]
{ALL | *permission* [(colonne, [**,**...n])]}
ON *objet* [**,**...n]
{FROM | TO } *nom_user* [**,**...n]
[CASCADE]

GRANT OPTION FOR
Indique que l'utilisateur ne pourra plus accorder cette autorisation spécifique
Peut servir à n'interdire que cette possibilité.

permission,
Spécifie l'autorisation à lever.

objet,
Correspond au nom complet de l'objet (nom_schéma.nom_objet).

FROM |*TO nom_user*
Nom de l'utilisateur à qui l'autorisation sera retirée.

CASCADE
Tous les utilisateurs ayant reçu cette autorisation par l'intermédiaire de cet utilisateur se verront leur autorisation retirée.

Exemple : Retirer à l'utilisateur camille l'autorisation Select sur la table sales.SalesPerson de la base de données AdventureWorks

```
USE AdventureWorks
REVOKE SELECT ON sales.SalesPerson FROM camille
```

L'instruction DENY permet d'interdire à un utilisateur l'utilisation d'un privilège, même s'il en reçoit l'autorisation par son appartenance à un groupe..

DENY {ALL | *permission* [(colonne, [**,**...n])]}
ON *objet* [**,**...n]
{ TO } *nom_user* [**,**...n]
[CASCADE]

Exemple : Interdire à l'utilisateur camille l'autorisation Select sur la table sales.SalesPerson de la base de données AdventureWorks

```
USE AdventureWorks
DENY SELECT ON sales.SalesPerson TO camille
```

Dans SQL Server Management Studio, les privilèges d'utilisation des objets peuvent être gérés à deux niveaux :

- En développant les propriétés de l'utilisateur de la base de données, onglet éléments sécurisables, ce qui permet de connaître toutes les autorisations pour un utilisateur
- En développant les propriétés de l'objet , onglet autorisations, ce qui permet d connaître toutes les utilisations de l'objet.

Les autorisations sur les objets au niveau serveur seront accessibles en développant les **propriétés** du serveur, grâce à l'onglet **Autorisations**

V.2 AUTORISATIONS D'INSTRUCTIONS

Les autorisations d'instruction contrôlent les actions d'administration comme la création d'une base ou l'ajout d'objets. Par défaut les connexions normales ne disposent pas de tels droits qui doivent être ajoutés spécifiquement en cas de besoin.

Au sein d'une base, un utilisateur pourrait être capable de créer ses propres tables, ses procédures, de sauvegarder …

Syntaxe sous Transact-SQL :

GRANT *permission* [**,**...n]
TO *nom_user* [**,**...n] [WITH GRANT OPTION]

permission,
Spécifie l'autorisation pouvant être accordée.

Nom_user
Nom de l'utilisateur à qui l'autorisation sera accordée.

WITH GRANT OPTION
Cet utilisateur pourra accorder cette autorisation à d'autres utilisateurs.

Exemple : Accorder à l'utilisateur camille l'autorisation de créer des tables sur la base de données AdventureWorks, avec le droit d'accorder CREATE TABLE à d'autres utilisateurs

```
USE AdventureWorks
GRANT CREATE TABLE TO camille WITH GRANT OPTION
```

Cette autorisation pourra par la suite être retirée par l'instruction REVOKE.

Syntaxe sous Transact-SQL :

REVOKE [GRANT OPTION FOR]
permission [**(** *column* [**,**...n] **)**] [**,**...n]
FROM *nom_user* [**,**...n]
[CASCADE]

GRANT OPTION FOR
Indique que l'utilisateur ne pourra plus accorder cette autorisation spécifique
Peut servir à n'interdire que cette possibilité.

permission,
Spécifie l'autorisation à lever.

nom_user
Nom de l'utilisateur à qui l'autorisation sera accordée.

CASCADE
Tous les utilisateurs ayant reçu cette autorisation par l'intermédiaire de cet utilisateur se verront leur autorisation retirée.

Exemple : Retirer à l'utilisateur camille l'autorisation de créer des tables sur la base de données AdventureWorks

```
USE AdventureWorks
REVOKE CREATE TABLE FROM camille CASCADE
```

L'instruction DENY permet d'interdire à un utilisateur l'utilisation d'un privilège, même s'il en reçoit l'autorisation par son appartenance à un groupe..

DENY *permission* [**,**...n]
TO *nom_user* [**,**...n] [CASCADE]

Exemple : Interdire à l'utilisateur camille l'autorisation de créer des tables sur la base de données AdventureWorks

```
USE AdventureWorks
DENY CREATE TABLE TO camille
```

SQL serveur 2005 dispose d'un ensemble d'instructions conséquent pour la gestion des autorisations: chaque élément sécurisable de SQL Server 2005 comporte des autorisations qui peuvent être accordées à un élément principal de sécurité.

Ces autorisations débutent par un ou des mots-clés qui identifient l'autorisation accordée.

CONTROL
Confère des fonctionnalités de type propriété au bénéficiaire de l'autorisation. L'élément principal dispose de toutes les autorisations définies sur l'objet sécurisable, et peut aussi en accorder des autorisations sur cet élément. Le modèle de sécurité de SQL Server étant hiérarchique, accorder CONTROL sur une portée particulière inclut implicitement CONTROL à tous les objets sécurisables inclus dans cette portée.
Par exemple, CONTROL sur une base de données implique toutes les autorisations sur la base de données, toutes les autorisations sur tous les assemblys de la base de données, toutes les autorisations sur tous les schémas de la base de données et toutes les autorisations sur les objets de tous les schémas de la base de données.

ALTER ANY <base de données> | <serveur>
Accorde la possibilité de CREATE, ALTER, DROP sur des éléments individuels sécurisables de la base de données / serveur
Exemple 1 : accorder ALTER ANY SCHEMA sur une base de données à un élément principal, donne la permission à cet élément de CREATE, ALTER, DROP sur tout schéma de la base.
Exemple 2: accorder ALTER ANY LOGIN sur le serveur à un élément principal, donne la permission à cet élément de CREATE, ALTER, DROP sur toute connexion du serveur.

ALTER
Accorde la possibilité de modifier les propriétés, excepté l'appartenance, d'un élément sécurisable particulier ; Lorsque ALTER est accordé sur une portée, ALTER octroie également la capacité de modifier, de créer ou de supprimer tous les objets sécurisables contenus dans cette portée
Exemple : accorder ALTER sur un schéma à un élément principal, donne la permission à cet élément de CREATE, ALTER, DROP sur tous les objets du schéma.

TAKE OWNERSHIP
Accorde la possibilité d'obtenir la propriété de l'élément sécurisable.

CREATE <élément sécurisable du serveur | base de données | schéma >
Accorde la possibilité de créer un objet sécurisable du serveur, de la base de données, du schéma
Toutefois, l'autorisation ALTER sur le schéma est requise pour créer l'objet sécurisable dans un schéma particulier.

VIEW DEFINITION
Permet au bénéficiaire d'accéder aux métadonnées.

BACKUP et **DUMP** sont synonymes.
RESTORE et **LOAD** sont synonymes.

Dans SQL Server Management Studio, après avoir développé les **propriétés** de la base de données, vous accéderez à l'onglet **Autorisations**, dans lequel pour chaque utilisateur de la base, vous pourrez octroyer (GRANT) ou révoquer (REVOKE), octroyer avec autorisation (GRANT WITH GRANT OPTION), refuser (DENY) les autorisations sur les objets de base de sonnées.

Les autorisations sur les objets au niveau serveur seront accessibles en développant les **propriétés** du serveur, grâce à l'onglet **Autorisations**

V.3 AUTORISATIONS IMPLICITES

Seuls les membres de rôles système prédéfinis ou de propriétaires de base de données ou d'objets de la base disposent d'autorisation implicites.

Pour accorder des droits implicites à des utilisateurs, il suffit de les faire membre du rôle correspondant.

VI ATTRIBUTION DE CONNEXIONS A DES ROLES

Les rôles constituent un moyen de regrouper des utilisateurs dans une unité unique à laquelle il est possible d'octroyer des permissions.

SQL Server offre des rôles des bases de données et de serveur fixes prédéfinis qui facilitent les opérations administratives courantes et permettent d'octroyer facilement des permissions courantes à des utilisateurs. Il est également possible de créer ses propres rôles de base de données.

VI.1 ROLES DE SERVEUR

Les **rôles de serveur fixes** offrent des privilèges administratifs au niveau du serveur (administrateur système, créateur de base de données, responsable de la sécurité). Ils sont gérés indépendamment des bases de données et sont stockés dans la table système **sys.server_principals**. (syslogins pour SQL Server 2000)

Rôle de serveur fixe	Description
Sysadmin	Peut contrôler entièrement SQL Server et les bases de données installées
Serveradmin	Peut configurer les paramètres serveur
Setupadmin	Peut mettre en œuvre la réplication et contrôler les procédures de démarrage
Securityadmin	Peut gérer des connexions serveur, créer des autorisations de base de données.
Processadmin	Peut gérer des processus exécutés dans SQL Server
Dbcreator	Peut créer, modifier et restaurer des bases de données
Diskadmin	Peut gérer des fichiers disque
Bulkadmin	Peut effectuer des opérations d'insertion en bloc

Le rôle **sysadmin** est l'équivalent de la connexion **sa**.

Tout membre d'un des rôles peut ajouter des membres à ce rôle.

Pour ajouter un compte de connexion en tant que membre d'un rôle de serveur fixe, on pourra utiliser une procédure système **sp_addsrvrolemember**

Exemple : Accorder à la connexion **ECRPSI\camille** le rôle de serveur fixe **sysadmin**

```
EXEC sp_addsrvrolemember 'ECRPSI\camille', 'sysadmin';
```

On retirera un compte de connexion d'un rôle de serveur fixe en utilisant la procédure stockée système sp_dropsrvrolemember

Dans SQL Server Management Studio,

Après avoir développé le dossier **Sécurité** de votre serveur, vous ajouterez un rôle de serveur à un compte de connexion, en affichant ses propriétés, et dans la page **Rôles de serveur**, en cochant/décochant le rôle choisi.

VI.2 ROLES DE BASE DE DONNEES

Pour accorder des autorisations au niveau des bases de données, SQL Server 2005 reconnaît des types de rôles de bases de données, qui permettent de regrouper les utilisateurs et de gérer plus facilement les autorisations d'accès à la base de données.

- **le rôle public**

Le rôle **public** est un rôle auquel appartiennent tous les utilisateurs d'une base de données. Il contient toutes les autorisations par défaut dont bénéficient les utilisateurs de la base.

- **les rôles de bases de données fixes**

Ils offrent des privilèges administratifs prédéfinis, non modifiables au niveau des bases de données (sauvegarde et restauration, responsable de la sécurité, lecture et modification des données).

Ils sont stockés dans la table système sys.server_principals (sysusers dans SQL Server 2000) de chaque base de données.

Rôle de base de donnée fixe	**Description**
db_owner	Peut exécuter les activités de tous les rôles de base de données, ainsi que des taches de maintenance et configuration dans la base.
db_accessadmin	Peut ajouter ou supprimer des connexions à la base.
db_datareader	Peut afficher toutes les données de toutes les tables utilisateur dans la base.
db_datawriter	Peut ajouter, modifier ou supprimer des données de toutes les tables de la base.
db_ddladmin	Peut accomplir des tâches associées au langage DDL, sauf Grant, Revoke et Deny.
db_securityadmin	Peut gérer les autorisations, la propriété des objets et les rôles.
db_backupoperator	Peut sauvegarder la base de données
db_denydatareader	Ne peut pas lire les données des tables utilisateur.
db_denydatawriter	Ne peut pas ajouter,modifier ou supprimer des données dans les tables utilisateur.

Les permissions du rôle **db_owner** couvrent toutes celles des autres rôles.

Pour ajouter un compte de sécurité en tant que membre d'un rôle de base de données, on pourra utiliser une procédure système **sp_addrolemember** ou SQL Server Management Studio.
Seuls les membres du rôle db_owner sont habilités à exécuter cette procédure ;

Exemple : Ajouter l'utilisateur camille au rôle db_datareader de la base de données active

Exec sp_addrolemember **'db_datareader'**, **'camille**'

- **les rôles standard** définis par l'utilisateur

Ils permettent de regrouper les utilisateurs qui bénéficient des mêmes autorisations de manière logique :

Exemple : Créer un rôle nommé **Lecture** qui n'autorise que le Select sur certaines tables, et ajouter l'utilisateur **camille** à ce rôle

CREATE ROLE **Lecture**

GRANT **SELECT** ON **Table1**, **Table2** TO **Lecture**

EXEC sp_addrolemember **'Lecture', 'camille'**

La procédure système **sp_droprolemember** permet de supprimer un utilisateur d'un rôle de base de données fixe ou d'un rôle standard.

L'instruction **DROP ROLE** permettra de supprimer un rôle standard ; les rôles de base de données fixes ne peuvent pas être supprimés.

- **les rôles d'application** définis par l'utilisateur

Ils permettent d'attribuer des autorisations temporaires pour des applications spécifiques. L'application active alors le rôle par mot de passe, ce qui fait bénéficier à l'utilisateur de ces autorisations temporaires.

Exemple : l'application peut accorder l'autorisation de mettre à jour (Update) certaines tables aux utilisateurs qui n'ont que des droits d'affichage.

Le rôle sera créé de la même manière que précédemment, l'application l'activera par la procédure système

Exec sp_setapprole **'nom_du_role' 'mot_de_passe'**

Le mot de passe ayant été défini à la création du rôle.

L'instruction DROP APPLICATION ROLE permet de supprimer un rôle d'application.

SQL Server propose des procédures stockées système concernant la sécurité :

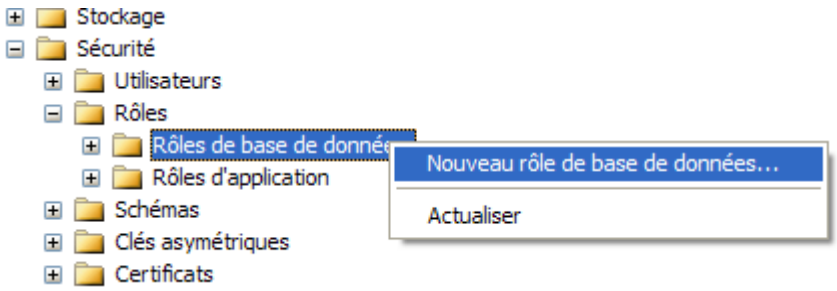
- **sp_helprole** renvoie les informations concernant les rôles de la base de données active
- **sp_helpsrvrole** renvoie les informations concernant les rôles de serveur fixe
- **sp_helprolemember** renvoie les membres d'un rôle
- **sp_helsrvprolemember** renvoie les membres d'un rôle de serveur fixe
- **sp_dbfixedrolepermission** affiche les autorisations d'un rôle de base de données fixe
- **sp_srvrolepermission** affiche les autorisations d'un rôle de serveur fixe

La version 2005 propose des vues de catalogue concernant la sécurité :

- **sys.database_permissions** / **sys.server_permissions** renvoie une ligne par autorisation de niveau base de données / serveur
- **sys.database_rolemember** / **sys.server_rolemember** renvoie une ligne par membre de rôle de base de données / serveur fixe
- **sys.database_principal** / **sys.server_principal** renvoie une ligne par entité de sécurité de la base de données / du serveur

Dans SQL Server Management Studio,

Vous créerez un rôle standard ou un rôle d'application dans le dossier sécurité de votre base de données.



Après avoir développé le dossier **Utilisateurs** de la base de donnée choisie dans le dossier **Bases de données** de votre serveur, vous affecterez un rôle de base de données dans la page **Général** de l'utilisateur concerné.

Vous accéderez à la liste des membres d'un rôle de base de données en développant le dossier **Rôles** et en cliquant dans le volet détail sur le rôle choisi.

VII LA CLAUSE EXECUTE AS

Sous SQL Server 2005, **EXECUTE AS** permet de modifier le contexte d'exécution d'un module (procédures stockées ou UDF). L'utilisateur concerné doit bien évidemment, disposer au préalable de l'autorisation EXECUTE sur le module en question.

Lorsque ce dernier est exécuté, le contexte de sécurité est celui défini par les paramètres de la clause EXECUTE AS.

Exemple :

Jean dispose d'une autorisation EXECUTE sur pour la procédure stockée Ventes.ProcX, le schéma Ventes étant la propriété de Karine. La procédure accède à plusieurs tables, dont certaines se trouvant dans différents schémas, tous propriété de Karine.

Bien que, compte tenu des règles d'appropriation, Jean n'ait pas l'autorisation Select sur ces différentes tables, la procédure s'exécute tant que le propriétaire des différents schémas reste le même.

Si l'une des tables appartient à un schéma dont le propriétaire est différent, les règles d'appropriation ne sont plus respectées, et Jean devra faire état d'une autorisation spécifique Select sur la table en question.

VII.1 EXECUTE AS CALLER

Cette option force le module à s'exécuter dans le contexte de l'appelant : c'est l'option de contexte par défaut, il n'est donc pas utile de le préciser.

VII.2 EXECUTE AS 'USER'

Cette option spécifie une exécution dans le contexte d'exécution de l'utilisateur user passé en paramètre. Le basculement de contexte se fait à l'exécution de la clause EXECUTE AS.

Pour reprendre notre exemple, imaginons que Jean exécute la procédure stockée Ventes.ProcX dans le contexte d'exécution de Karine.

Karine a l'autorisation Select sur toutes les tables requises par la procédure stockée. Lorsque Jean exécute la procédure stockée, c'est exactement comme s'il s'agissait de Karine. Ce sont donc ses droits d'accès qui s'appliquent.

C'est ce qu'on appelle le principe **d'impersonation.** Il est prudent de n'utiliser cette fonctionnalité que lorsqu'on souhaite passer outre les autorisations normales de la base de données.

L'option EXECUTE as SELF n'est qu'un raccourci de EXECUTE AS 'user', le paramètre user étant dans ce cas l'utilisateur qui exécute l'instruction.

Afin d'éviter d'éventuels conflits, il est préférable de mentionner explicitement les noms d'utilisateurs.

Etablissement référent
Marseille Saint Jérôme

Equipe de conception
Elisabeth Cattanéo

Remerciements :

Date de mise à jour 05/05/2008
afpa © Date de dépôt légal mai 08

afpa / Direction de l'Ingénierie13 place du Générale de Gaulle / 93108 Montreuil Cedex
association nationale pour la formation professionnelle des adultes
Ministère des Affaires sociales du Travail et de la Solidarité