

Chapitre 3

Les ordres du SQL

1. Les bases du langage SQL

1.1 Les expressions

Dans la plupart des syntaxes Transact-SQL, il est possible d'utiliser des expressions ou des combinaisons d'expressions pour gérer des valeurs ou pour tirer parti de la programmabilité du langage. Les expressions peuvent prendre différentes formes :

Les valeurs constantes

Exemple

■ **Caractère** 'AZERTY', 'Ecole Nantaise d''Informatique'

Les chaînes de caractères sont entourées d'apostrophes. S'il doit y avoir une apostrophe dans la chaîne de caractères, il est nécessaire d'en mettre deux pour ne pas le confondre avec la fin de la chaîne de caractères.

■ **Numérique** 10, -15.26, 1.235e-5

Les valeurs numériques peuvent être écrites avec des chiffres en utilisant le point pour séparer la partie entière de la partie décimale. Il est possible d'utiliser la notation scientifique en utilisant le *e* avant la puissance de :

Date		
constante	date	heure
'231205'	5 Décembre 2023	00:00:00.000
'05/12/2023'	5 Décembre 2023	00:00:00.000
'05-12-23 8:30'	5 Décembre 2023	08:30:00.000
'8:30:2'	1 Janvier 1900	08:30:02.000
'5.12.2023 8:30pm'	5 Décembre 2023	20:30:00.000

Le format français jj/mm/aaaa est l'un des formats possibles pour les dates.

■ **Binaire** 0x05, 0xFF, 0x5aef1b
 ■ **Nulle** NULL

Les données binaires doivent être données en hexadécimal et sont préfixées par 0x.

Les noms de colonne

Un nom de colonne pourra être employé comme expression, la valeur de l'expression étant la valeur "stockée" de la colonne.

Les fonctions

On peut utiliser comme expression n'importe quelle fonction, la valeur de l'expression est le résultat retourné par la fonction.

Exemple

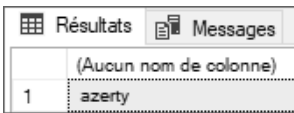
expression	valeur
SQRT(9)	3
substring('ABCDEF',2,3)	'BCD'

Les variables

Les variables peuvent être employées en tant qu'expression ou dans une expression, sous la forme @nom_de_variable. La valeur de l'expression est la valeur de la variable.

Exemple

```
DECLARE @x CHAR(10);  
SELECT @x='AZERTY';  
SELECT LOWER(@x);
```



Résultats		Messages	
(Aucun nom de colonne)			
1	azerty		

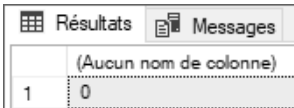
Les sous-requêtes

Une requête SELECT entre parenthèses peut être employée en tant qu'expression ayant pour valeur le résultat de la requête, soit une valeur unique, soit un ensemble de valeurs. Les requêtes SELECT sont présentées à la section L'extraction de lignes de ce chapitre.

Exemple

Stockage du nombre de clients dans une variable :

```
DECLARE @nombre INT;  
SELECT @nombre = (SELECT COUNT(*) FROM Clients);  
SELECT @nombre;
```



Résultats		Messages	
(Aucun nom de colonne)			
1	0		

Les expressions booléennes

Elles sont destinées à tester des conditions (IF, WHILE, WHERE, etc.). Ces expressions sont composées de la manière suivante :

```
expression1 opérateur expression2
```

1.2 Les opérateurs

Les opérateurs vont permettre de constituer des expressions calculées, des expressions booléennes ou des combinaisons d'expressions.

1.2.1 Les opérateurs arithmétiques

Ils permettent d'effectuer des calculs élémentaires et de renvoyer un résultat.

+ Addition

- Soustraction

* Multiplication

/ Division : division entière si les deux opérandes sont des entiers, division réelle sinon.

% Modulo (reste de la division entière)

(...) Parenthèses

Exemple

```
DECLARE @prixArticle NUMERIC(6,2) = 145.2;
DECLARE @TVA NUMERIC(2,2) = 20.0/100;
SELECT @prixArticle*(1+@TVA);
```

Résultats	
(Aucun nom de colonne)	
1	174.2400

```
DECLARE @nb INT = (SELECT COUNT(*) FROM Clients);
IF @nb%2=0 PRINT 'nombre de clients pair';
ELSE PRINT 'nombre de clients impair';
```

nombre de clients pair

■ Remarque

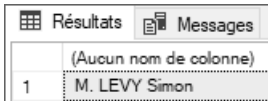
Les opérateurs + et - fonctionnent également sur les dates. L'unité est alors le jour. Exemple : `GETDATE() + 1` permet d'obtenir la date du lendemain.

1.2.2 La concaténation de chaînes de caractères

La concaténation permet de constituer une seule chaîne de caractères à partir de plusieurs expressions de type caractère. L'opérateur de concaténation est le signe plus (+).

Exemple

```
DECLARE @nom CHAR(4) = 'LEVY';  
DECLARE @prenom CHAR(5) = 'Simon';  
SELECT 'M. ' + @nom + ' ' + @prenom;
```



Résultats	
(Aucun nom de colonne)	
1	M. LEVY Simon

Il existe de plus la fonction `CONCAT()` qui permet de réaliser des concaténations. Contrairement à l'opérateur `+`, cette fonction appartient à la norme SQL. Elle a également l'avantage de pouvoir être utilisée avec des nombres sans que la somme soit effectuée.

1.2.3 Les opérateurs binaires

Ils permettent le calcul entre entiers, traduits implicitement en valeurs binaires.

& ET

| OU

^ OU Exclusif

~ NON

1.2.4 Les opérateurs de comparaison

Ils permettent la constitution d'expressions booléennes en comparant des expressions. Ces expressions peuvent être placées entre parenthèses.

`exp1 = exp2`

Égal.

`exp1 > exp2`

Supérieur.

`exp1 >= exp2`

Supérieur ou égal.

■ Remarque

`exp1 !> exp2` est également possible mais n'est pas dans la norme.

`exp1 < exp2`

Inférieur.

`exp1 <= exp2`

Inférieur ou égal.

■ Remarque

`exp1 !< exp2` est également possible mais n'est pas dans la norme.

`exp1 <> exp2`

Différent.

■ Remarque

`exp1 != exp2` est également possible mais n'est pas dans la norme.

`exp IN (exp1, exp2, ...)`

Indique si la valeur est parmi celles de la liste de valeurs.

`exp IS NULL` ou `exp IS NOT NULL`

Teste l'absence de valeur (NULL). Pour tester si une variable n'a pas de valeur affectée, il est indispensable d'utiliser l'opérateur IS NULL.

`exp LIKE 'masque'`

Filtre la chaîne de caractères ou la date suivant le masque spécifié.

Le masque peut être composé de :

—

Un caractère quelconque.

%

n caractères quelconques, avec n compris entre zéro et l'infini.

[ab...]

Un caractère dans la liste ab... .

[a-z]

Un caractère dans l'intervalle a-z.

[^ab...]

Un caractère en dehors de la liste ou de l'intervalle spécifié.

ab...

Le caractère lui-même.

Pour utiliser `_`, `%`, `[` et `^` en tant que caractères de recherche, il faut les encadrer de `[]`.

Exemple

masque	Chaînes correspondantes
'G%'	commençant par "G"
'_X%1'	deuxième caractère "X", dernier "1"
'%[1-9]'	se terminant par un chiffre compris entre "1" et "9"
'[^XW]%'	ne commençant ni par X ni par W
'LE[_]%'	commençant par "LE_"

exp LIKE 'masque' ESCAPE '\'

Lorsque le masque contient un caractère spécial comme `_` ou `&`, il est nécessaire de les faire précéder d'un caractère spécifique appelé caractère d'échappement. Afin de s'adapter à toutes les situations possibles, le caractère d'échappement est libre et il est simplement nécessaire de le spécifier après le mot-clé ESCAPE.

`exp BETWEEN min AND max`

Teste si la valeur de `exp` est incluse dans l'intervalle compris entre les valeurs `min` et `max` (bornes incluses).

`EXISTS` (sous-requête)

Renvoie VRAI si la sous-requête a renvoyé au moins une ligne.

Les opérateurs booléens

Ils permettent de combiner des expressions booléennes (`expb`) en renvoyant une valeur booléenne.

`expb1 OR expb2`

Vrai si une des deux expressions est vraie.

`expb1 AND expb2`

Vrai si les deux expressions sont vraies.

`NOT expb`

Vrai si `expb` est fausse.

1.3 Les fonctions

De nombreuses fonctions sont disponibles pour valoriser des colonnes ou pour effectuer des tests. Des exemples sont donnés avec l'instruction `SELECT`.

Parmi toutes les fonctions proposées en standard par SQL Server, il est possible de les regrouper par type : rowset, agrégation, ranking, scalaire. Pour ce dernier type de fonctions, il faut en plus les classer par catégories : mathématique, chaîne de caractères, date... car elles sont très nombreuses.

Certaines fonctions, plus particulièrement les fonctions permettant de manipuler les données de type caractère, prennent en compte le classement défini au niveau du serveur.

Chapitre 4

Gestion de la sécurité des accès

1. Introduction

Le contrôle d'accès représente une opération importante au niveau de la gestion de la sécurité sur un serveur de bases de données. La sécurisation des données nécessite une organisation des objets de façon indépendante des utilisateurs, ce qui est possible par les schémas. La sécurité passe également par un meilleur contrôle des autorisations et la possibilité d'accorder juste les privilèges nécessaires à chaque utilisateur pour qu'il puisse travailler de façon autonome.

Pour l'organisation de cette politique de sécurité, il faut prendre en compte l'organisation hiérarchique des éléments de sécurité, de façon à rendre la gestion des droits d'accès simple et efficace.

SQL Server s'appuie sur trois éléments clés qui sont :

- les entités de sécurité,
- les sécurisables,
- les autorisations.

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

Les sécurisables représentent les objets gérés par le serveur. Ici, un objet peut être une table, un schéma ou une base de données par exemple.

Les autorisations sont accordées aux entités de sécurité afin qu'elles puissent travailler avec les sécurisables.

L'organisation hiérarchique permet d'accorder une autorisation (par exemple SELECT) sur un sécurisable de niveau élevé (par exemple le schéma) pour permettre à l'entité de sécurité qui reçoit l'autorisation d'exécuter l'instruction SELECT sur toutes les tables contenues dans le schéma.

Les vues du catalogue système permettent d'obtenir un rapport complet et détaillé sur les connexions existantes, les utilisateurs de base de données définis et les privilèges accordés. Quelques-unes de ces vues sont présentées ci-dessous :

- sys.server_permissions : liste des permissions de niveau serveur et de leurs bénéficiaires.
- sys.sql_logins : liste des connexions.
- sys.server_principals : entité de sécurité définie au niveau serveur.
- sys.server_role_members : liste des bénéficiaires d'un rôle de serveur.
- sys.database_permissions : liste des permissions et de leur bénéficiaire au niveau base de données.
- sys.database_principals : entité de sécurité de niveau base de données.
- sys.database_role_members : liste des bénéficiaires d'un rôle de base de données.

Pour simplifier la gestion des droits d'accès, il est possible d'utiliser trois types de rôles. Les **rôles de serveur** qui regroupent des autorisations au niveau du serveur, ces autorisations sont valables pour toutes les bases installées. Les **rôles de base de données**, regroupent quant à eux des droits au niveau de la base de données sur laquelle ils sont définis. Et enfin les **rôles d'applications**, définis sur les bases de données utilisateur, permettent de regrouper les droits nécessaires à la bonne exécution d'une application cliente.

2. Gestion des accès serveur

Avant de pouvoir travailler avec les données gérées par les bases, il faut dans un premier temps se connecter au serveur SQL. Cette étape permet de se faire identifier par le serveur SQL et d'utiliser par la suite tous les droits qui sont accordés à notre connexion. Il existe dans SQL Server deux modes de gestion des accès au serveur de base de données.

Attention : dans cette section, seule la partie connexion au serveur est abordée. Il est important de bien distinguer la connexion au serveur et l'utilisation de bases de données. La connexion au serveur permet de se faire identifier par le serveur SQL comme un utilisateur valide, pour utiliser, par la suite, une base de données : les données et les objets. L'ensemble de ces droits seront définis ultérieurement. Ces droits sont associés à un utilisateur de base de données, pour lequel correspond une connexion.

Remarque

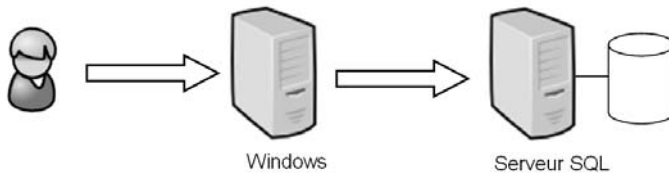
On parlera de connexion au serveur ou de logins.

2.1 Mode de sécurité Windows

Ce type de gestion de la sécurité permet de s'appuyer sur les utilisateurs et les groupes Windows pour le domaine et le poste local. SQL Server utilise la gestion des utilisateurs de Windows (gestion des mots de passe...) et récupère uniquement les noms pour créer des connexions au serveur.

Une fonctionnalité très importante de SQL Server est de pouvoir autoriser des groupes Windows à venir se connecter. La gestion des groupes permet de simplifier grandement la gestion de l'accès aux ressources. Les mêmes groupes Windows peuvent donc être utilisés pour donner accès à des fichiers ou à des objets de bases de données SQL Server.

Avec une telle méthode de fonctionnement, comme un utilisateur Windows peut appartenir à plusieurs groupes, il peut posséder plusieurs droits de connexion à SQL Server.



Authentication Windows

En mode sécurité Windows, seuls les noms d'utilisateurs sont stockés. La gestion des mots de passe et de l'appartenance aux différents groupes est laissée à Windows. Ce mode de fonctionnement permet de bien discerner les tâches de chacun et de spécialiser SQL Server sur la gestion des données en laissant Windows gérer l'authentification des utilisateurs, ce qu'il sait bien faire. De plus, avec un tel schéma de fonctionnement, il est possible d'appliquer la politique suivante : un utilisateur = un mot de passe. L'accès au serveur SQL est transparent pour les utilisateurs approuvés par Windows.

Remarque

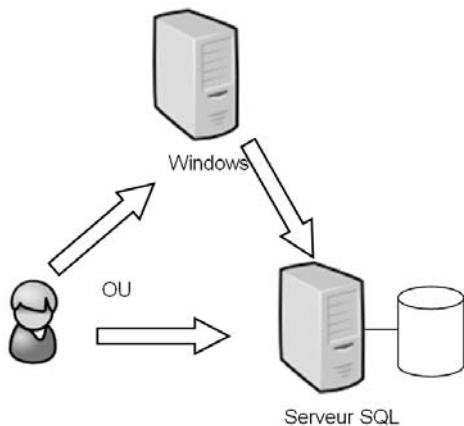
SQL Server s'appuie sur les groupes auxquels appartient l'utilisateur lors de la connexion au serveur. Si des modifications d'appartenance aux groupes sont effectuées depuis Windows, ces modifications ne seront prises en compte que lors de la prochaine connexion de l'utilisateur au serveur SQL.

Remarque

SQL Server s'appuie sur le SID de Windows pour identifier le groupe. Si un groupe est supprimé puis recréé dans Windows, il est important de réaliser la même opération en ce qui concerne la connexion du groupe dans SQL Server.

2.2 Mode de sécurité mixte

Le mode de sécurité mixte repose sur une authentification Windows puis sur une authentification SQL Server. C'est ce mode d'authentification qui va être détaillé ici. Dans un tel mode de fonctionnement, c'est SQL Server qui se charge de vérifier que l'utilisateur qui demande à se connecter est bien défini puis il se charge également de vérifier le mot de passe.

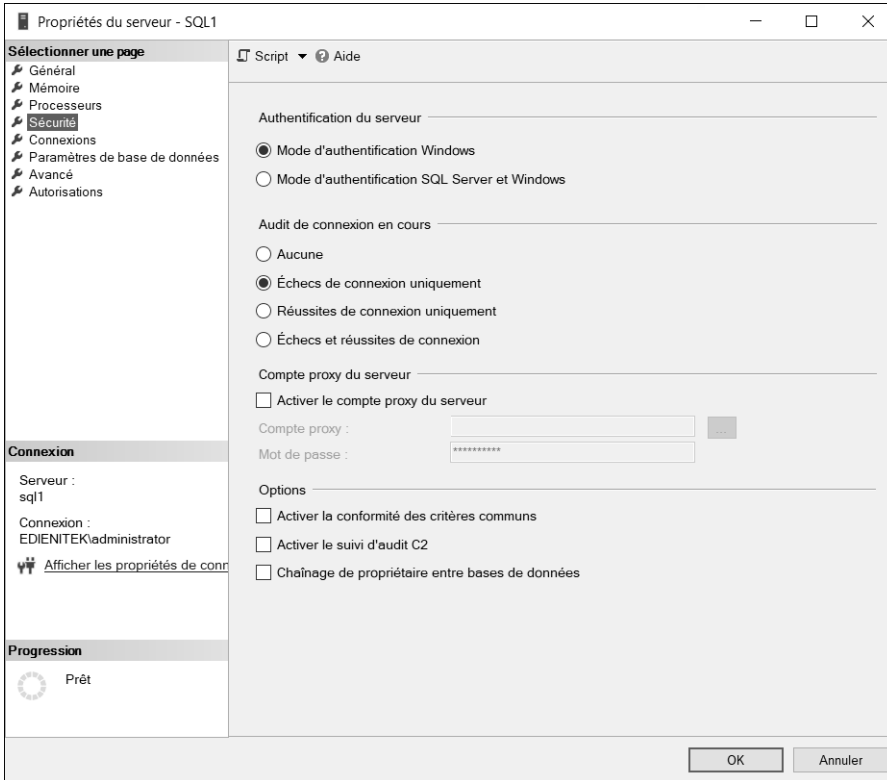


Mode de sécurité mixte

Tous les utilisateurs sont entièrement gérés par SQL Server (nom et mot de passe). Ce type de gestion des connexions est bien adapté pour des clients qui ne peuvent pas s'identifier auprès de Windows.

2.3 Comment choisir un mode de sécurité ?

Il est possible de modifier le mode de sécurité utilisé par le serveur SQL directement depuis SQL Server Management Studio en allant modifier les propriétés de l'instance comme ci-après.



Lors de l'installation du serveur SQL, des connexions administrateurs SQL (**sysadmins**) sont créées. En mode Windows, le groupe local des Administrateurs peut être autorisé à se connecter, et en mode de sécurité SQL Server, l'utilisateur **sa** est déjà créé, seul un mot de passe est à paramétrer.

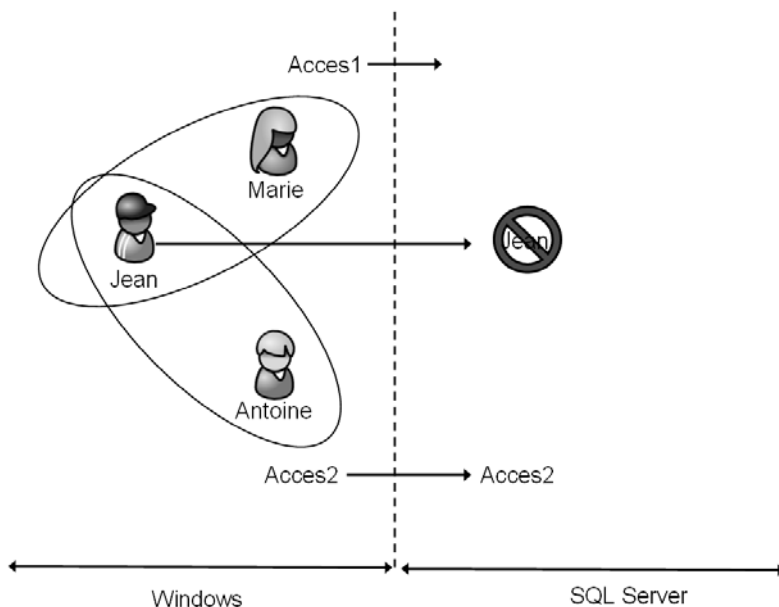
Remarque

Il est recommandé d'utiliser la sécurité Windows, qui offre une plus grande souplesse au niveau de la gestion des utilisateurs.

Les changements de mode entre mode mixte et mode Windows ne sont pris en compte qu'après redémarrage du service SQL Server.

2.4 Gérer une connexion à SQL Server

Étant donné qu'un utilisateur Windows peut appartenir à plusieurs groupes, il peut se voir accorder plusieurs fois le droit de connexion au serveur SQL. Ceci peut poser un problème lorsqu'un utilisateur ou un groupe d'utilisateurs ne doit jamais pouvoir se connecter au serveur SQL. Afin de remédier à ce souci, Microsoft fournit, parmi les ordres Transact SQL pour la gestion des connexions, l'ordre `DENY` qui permet de refuser explicitement un utilisateur ou un groupe Windows. Le `DENY` constitue un refus explicite et est prioritaire par rapport aux différentes autorisations de connexion.



Création d'une connexion

Dans le schéma ci-dessus, il y a trois utilisateurs Windows et deux groupes. Une connexion SQL a été mise en place pour le groupe `Acces2`, le groupe `Acces1` ne possède pas de connexion, mais pour que l'utilisateur Jean soit interdit d'accès à SQL Server, une connexion a été créée sur SQL Server avec la propriété `DENY`.

Remarque

*Il faut posséder une permission d'administrateur (**sysadmin**) ou une permission de gestionnaire de sécurité (**securityadmin**) pour pouvoir réaliser les différentes opérations relatives à la gestion des connexions.*

Les connexions, qu'elles soient de type SQL Server ou bien Windows, doivent être définies au niveau de l'instance SQL Server pour permettre aux utilisateurs de se connecter. Elles sont stockées dans la base de données système **Master**.

La gestion des connexions peut être réalisée de façon graphique par l'intermédiaire de l'explorateur d'objets depuis SQL Server Management Studio ou bien sous forme de script Transact SQL. Toutes les opérations de gestion des connexions sont réalisables en Transact SQL avec l'aide des instructions **CREATE LOGIN**, **ALTER LOGIN** et **DROP LOGIN**. Chaque solution possède ses avantages et ses inconvénients.

Remarque

Les procédures `sp_addlogin` et `sp_grantlogin` ne doivent plus être utilisées. Elles sont encore présentes dans SQL Server pour assurer la compatibilité des scripts.

2.4.1 En mode de sécurité Windows

Les noms des groupes ou des utilisateurs devront correspondre à ceux définis dans Windows.

SQL Server Management Studio

Il est possible de créer les connexions depuis l'interface graphique de SQL Server Management Studio, en procédant de la sorte :

- Depuis l'explorateur d'objets, se placer sur le nœud **Sécurité - Connexions**.
- Depuis le menu contextuel associé au nœud connexion, faire le choix **Nouvelle connexion**.

L'écran suivant apparaît alors. Il ne reste plus qu'à sélectionner le compte d'utilisateur Windows ou bien le groupe qui est autorisé à se connecter.