Chapitre 4
Transact-SQL : le langage procédural

1. Le SQL procédural

SQL Server est un serveur de base de données relationnelle et à ce titre, il fournit tous les éléments pour stocker de façon structurée les données mais aussi les outils nécessaires pour travailler avec les données au travers de SQL. Avec le Transact-SQL, il est également possible de définir des traitements procéduraux directement dans la base de données. Ces traitements vont pouvoir être utilisables par tous les utilisateurs de la base sous réserve qu'ils possèdent les privilèges nécessaires. Il est possible de conserver la définition de ces traitements et de les rendre paramétrables par l'intermédiaire de la création de procédures et de fonctions.

Des traitements procéduraux pourront également être mis en place pour définir des contraintes d'intégrité complexes, il s'agira alors de déclencheurs de base de données (TRIGGER).

Le Transact-SQL est un langage procédural qui intègre complètement et nativement le langage SQL. Il est ainsi possible de tirer parti des deux langages. Par exemple, il va être très facile de définir une variable en Transact-SQL puis d'inclure cette variable dans une requête SQL. Lors de l'exécution de la requête, c'est la valeur contenue par la variable qui est prise en compte.

Le Transact-SQL n'a pas pour objectif de se substituer aux requêtes SQL mais plutôt de permettre un complément par rapport aux tâches réalisables en SQL.

À l'aide du Transact-SQL, il va être possible de définir des procédures et des fonctions dans la base de données. Les déclencheurs de base de données vont permettre la mise en place au niveau de la base de règles métiers complexes.

1.1 Les variables

1.1.1 Les variables utilisateur

Une variable est une zone mémoire, caractérisée par un nom et un type, permettant de stocker une valeur. Les variables Transact-SQL doivent obligatoirement être déclarées avant leur utilisation. Elles peuvent ensuite remplacer n'importe quelle expression dans les instructions SQL.

Déclaration de variables

```
DECLARE @nom_variable type [= expr][,...][;]
```

`nom_variable`

Nom précédé du caractère @.

`type`

Type système ou défini par l'utilisateur.

Valorisation des variables

```
SELECT @nom_variable = expr [,...][FROM...]
```

Exemple

Dans les scripts suivants, les variables sont tout d'abord définies. La variable @choix est ensuite valorisée avec la valeur 20. Puis une requête de type SELECT permet de renseigner le contenu des variables @nom et @prenom. Cette requête fonctionne correctement car la restriction sur une valeur unique de la clé primaire permet de garantir qu'une seule ligne de données sera remontée depuis la base. Enfin, la dernière instruction SELECT permet de visualiser le contenu des variables @nom et @prenom.

```
DECLARE @choix int;
DECLARE @nom CHAR(30), @prenom CHAR(30);
SELECT @choix=20;
SELECT @nom=nom, @prenom=prenom
FROM clients
WHERE idClient = @choix;
SELECT @nom, @prenom;
```


Résultats Messages

	(Aucun nom de colonne)	(Aucun nom de colonne)
1	LABICHE	Adeline

1.1.2 Les variables système

Ces variables sont définies par le système, et peuvent être utilisées seulement en lecture. Elles se distinguent des variables utilisateur par le double @.

`@@CONNECTIONS`

Nombre de connexions ou de tentatives de connexion depuis le dernier démarrage de SQL Server.

`@@CPU_BUSY`

Temps consacré par l'unité centrale à SQL Server depuis le dernier démarrage de celui-ci. Le résultat est exprimé en unité CPU. Il faut multiplier par @@TIMETICKS pour obtenir un résultat en microsecondes.

`@@CURSOR_ROWS`

Nombre de lignes affectées dans le dernier curseur ouvert. Les curseurs sont présentés dans la section Les curseurs de ce chapitre.

`@@DATEFIRST`

Renvoie la valeur courante du paramètre SET DATEFIRST.

`@@DBTS`

Valeur du type de données timestamp courant pour la base de données.

`@@ERROR`

Dernier numéro d'erreur généré par le système.

`@@FETCH_STATUS`

Contient le statut d'une commande de curseur FETCH.

`@@IDENTITY`

Enregistre la dernière valeur IDENTITY insérée.

`@@IDLE`

Temps, en millisecondes, pendant lequel SQL Server est resté inactif depuis son dernier démarrage.

`@@IO_BUSY`

Temps, en millisecondes, consacré par SQL Server à effectuer des opérations d'entrée/sortie depuis son dernier démarrage.

`@@LANGID`

Identificateur de la langue actuellement utilisée.

`@@LANGUAGE`

Langue actuellement utilisée.

`@@LOCK_TIMEOUT`

Timeout, en millisecondes, de la session en cours.

`@@MAX_CONNECTIONS`

Nombre maximal de connexions simultanées qu'il est possible d'établir avec SQL Server.

`@@MAX_PRECISION`

Renvoie le niveau de précision utilisé par les types de données **decimal** et **numeric**.

`@@NESTLEVEL`

Niveau d'imbrication de l'instruction en cours d'exécution.

`@@OPTIONS`

Informations sur les valeurs courantes des options SET.

`@@PACK_RECEIVED`

Nombre de paquets entrants lus par SQL Server depuis son dernier démarrage.

`@@PACK_SENT`

Nombre de paquets sortants écrits par SQL Server depuis son dernier démarrage.

`@@PACKET_ERRORS`

Nombre d'erreurs qui se sont produites alors que SQL Server envoyait ou recevait des paquets depuis son dernier démarrage.

`@@PROCID`

Identificateur de la procédure stockée Transact-SQL du traitement en cours d'exécution.

`@@REMSERVER`

Renvoie le nom du serveur contenu dans l'enregistrement des noms d'accès d'un serveur distant.

`@@ROWCOUNT`

Nombre de lignes affectées par la dernière instruction.

`@@SERVERNAME`

Nom du serveur SQL local.

`@@SERVICENAME`

Nom du service en cours d'exécution.

`@@SPID`

Numéro d'identification du processus courant sur le serveur.

`@@TEXTSIZE`

Longueur maximale, en octets, des données texte ou image renvoyées par une instruction SELECT.

`@@TIMETICKS`

Nombre de millisecondes par pulsation.

`@@TOTAL_ERRORS`

Nombre d'erreurs rencontrées par SQL Server en lisant ou en écrivant des données depuis son dernier démarrage.

`@@TOTAL_READ`

Nombre de lectures de données sur le disque effectuées par SQL Server depuis son dernier démarrage.

`@@TOTAL_WRITE`

Nombre d'écritures de données sur le disque effectuées par SQL Server depuis son dernier démarrage.

`@@TRANCOUNT`

Nombre de transactions actuellement actives pour l'utilisateur courant.

`@@VERSION`

Date, numéro de version et type de processeur de la version courante de SQL Server.

Ces variables système contiennent de nombreuses informations et en les interrogeant il est déjà possible d'obtenir beaucoup d'informations sur le serveur.

Exemple

Dans l'exemple ci-dessous, la variable @@VERSION est interrogée et la version exacte de SQL Server est retournée. Avec ce numéro de version, il est par exemple possible de savoir quels Service Packs sont appliqués sur le serveur en s'appuyant sur les tableaux de correspondance disponibles sur le site web Technet de Microsoft.

```
SELECT @@VERSION;
Microsoft SQL Server 2019 (RTM-GDR) (KB4517790) - 15.0.2070.41 (X64)
Oct 28 2019 19:56:59   Copyright (C) 2019 Microsoft Corporation
Developer Edition (64-bit) on Windows 10 Pro 10.0 <X64> (Build 18363: )
```

1.1.3 L'affichage

L'instruction PRINT affiche un message.

Syntaxe

```
PRINT {'texte'|@variable|@@variablesystème}
```

Exemple

```
DECLARE @nb INT;
SELECT @nb=COUNT(*) FROM Clients;
PRINT 'Nombre de clients :'
PRINT @nb;
Affichage
Nombre de clients :
5
```

1.2 Les transactions

1.2.1 Le principe

Il est possible de définir une transaction comme un ensemble indivisible d'instructions, c'est-à-dire dans lequel toutes les instructions réussissent ou bien aucune. Cette notion de transaction est bien présente dans le monde réel, par exemple lors d'un retrait d'argent à un distributeur automatique bancaire les deux éléments indivisibles sont le débit du compte et la distribution des billets. Si l'un de ces éléments ne peut être exécuté correctement (et quelle qu'en soit la raison) alors c'est l'ensemble qui est mis en échec.

En effet, il n'est pas envisageable de distribuer des billets sans que le compte ne soit débité du montant correspondant.

1.2.2 La gestion des transactions

Le premier point à prendre en considération est le verrouillage des informations. En effet, lorsque SQL Server lit des données ou les modifie, il verrouille les lignes d'informations manipulées. Ce verrouillage dure le temps de l'exécution de l'instruction ou le temps de la transaction.

Suivant le type de verrous posés, il est possible ou non à d'autres transactions d'accéder simultanément à la même information.

Une transaction est un ensemble d'instructions de manipulations de données s'exécutant dans une même unité de travail. La validation d'une transaction assure que toutes les instructions en faisant partie se sont correctement terminées, l'annulation de la transaction assurera l'annulation de l'ensemble des instructions.

Seules les instructions du DML (SELECT, INSERT, UPDATE, DELETE) sont prises en compte dans une transaction.

Les transactions sont utiles pour assurer l'intégrité et la cohérence des données lors de modifications multiples, pour améliorer les performances, pour tester les effets de modification, pour gérer les verrouillages.

Une transaction est caractérisée par le mot-clé ACID (*Atomicity Consistency Isolation Durability*) qui peut se traduire en français par Atomique Consistance Indépendance Durée.

- Atomique car une transaction représente une unité atomique (non divisible) de travail pour le serveur.
- Consistance car à la fin de la transaction les informations présentes dans la base doivent être consistantes, c'est-à-dire cohérentes par rapport aux règles de structuration des données mises en place.