Chapitre 6
Les types évolués

1. Introduction

Les données utilisées dans le contexte des applications ne cessent d'évoluer. Il est donc normal que les serveurs de bases de données évoluent également en proposant des types adaptés à ces nouveaux formats. C'est ce que fait SQL Server en offrant la possibilité de stocker des données au format XML, des données géographiques ainsi qu'une meilleure gestion des documents annexes (image, vidéo, son, document numérisé...) afin de ne pas alourdir le processus de gestion de la base tout en liant les données relationnelles à ces informations stockées directement sur le système de fichiers.

2. Travailler avec le format XML

Les données au format XML sont de plus en plus présentes dans l'environnement de travail. Il est donc normal qu'une base de données s'adapte afin d'être en mesure de stocker et de gérer de façon optimale les données définies dans ce format. C'est ce que fait SQL Server en offrant la possibilité de travailler directement avec des données au format XML et de les stocker dans la structure relationnelle d'une table. Étant donné que XML représente avant tout un format d'échange de données, SQL Server propose également les outils nécessaires pour produire un document XML à partir de données relationnelles ou bien au contraire d'intégrer dans les tables relationnelles des données issues d'un document XML.

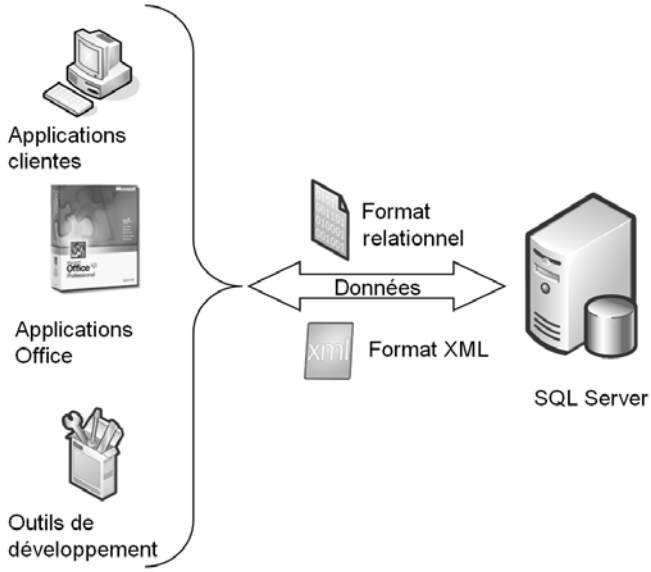
Il est possible de stocker les informations, soit au format relationnel, soit au format XML. Chaque format possède ses avantages et ses inconvénients.

SQL Server héberge un moteur relationnel pour le stockage et le travail avec les données conservées à ce format. Mais SQL Server propose également de gérer des données au format XML. Ainsi, quel que soit le mode de stockage retenu, SQL Server peut héberger ces données dans leur format natif.

L'objectif de SQL Server est d'être capable de s'adapter au mode de stockage des données en fonction du format avec lequel travaille l'application cliente.

Microsoft Office permet aux utilisateurs de Word, Excel Visio et Infopath de générer leurs documents au format XML, par l'intermédiaire du format OpenXML.

Le schéma ci-dessous illustre le fait que les applications travaillent aussi bien avec des données au format relationnel qu'au format XML.



Pour répondre correctement aux différentes demandes, SQL Server a considérablement amélioré sa gestion du format XML.

Choisir un format

Les deux formats ne sont pas en concurrence mais sont complémentaires. Le moteur de base de données doit donc être capable de gérer de façon optimum les données, quel que soit leur format de stockage.

Le XML est particulièrement bien adapté pour l'échange d'informations entre des applications, pour la gestion documentaire, pour l'envoi de message (SOAP)...

Le XML présente l'avantage d'être autodescriptif. Il offre ainsi la possibilité de transférer des structures de données complexes. Cette représentation des données est faite sous forme d'arborescence.

Cependant, le format relationnel permet de garantir une meilleure homogénéité des données car les tables sont fortement structurées. La structuration tabulaire des données donne la possibilité de stocker un grand volume d'informations de façon fiable et les requêtes pour extraire ces données sont rapides et performantes. C'est sans aucun doute le meilleur format pour stocker et travailler avec de gros volumes d'informations.

Le tableau suivant permet de privilégier un format ou un autre en fonction de la structure initiale des données :

Formatage des données	XML	Relationnel
Fichier plat	Bien adapté	Bien adapté
Structure hiérarchique	Bien adapté	Possible
Données semi structurées	Bien adapté	Possible
Langage de description	Bien adapté	Possible
Conserver l'ordre	Bien adapté	Possible
Récursivité	Bien adapté	Possible

2.1 Le type XML

SQL Server propose un type de données XML pour stocker les données au format natif XML. Ce type n'est pas un champ texte de grande dimension, nvarchar(max) par exemple. En effet, si pour des raisons de stockage pur cela ne change pas grand-chose, le type XML permet de faire des recherches précises d'informations. Il est également possible de définir des index sur des colonnes de type XML afin d'accélérer le traitement des requêtes.

Il est possible de créer des tables relationnelles qui, en plus des colonnes de types habituels, peuvent compter une ou plusieurs colonnes de type XML.

Les colonnes de type XML utilisent un format binaire (blob) pour stocker l'information dans la base relationnelle, ainsi le document XML est conservé dans l'état. De fait, l'espace pour chaque donnée XML est limité à 2 Go. De plus, le document XML ne doit pas être structuré avec une hiérarchie contenant plus de 128 niveaux.

Remarque

Les données XML sont typées en UTF-16 par SQL Server.

Avec le type de données XML, pour stocker les données directement à ce format, SQL Server évite de concevoir un travail long et fastidieux de mappage entre le format XML et la structure relationnelle des données telles qu'elles sont organisées dans la base. Ce type dispose des méthodes **query()**, **exist()**, **value()**, **nodes()** et **modify()** afin de pouvoir travailler sur les données de façon simple. Ces méthodes s'appuient sur XQuery, sous ensemble de XML spécifique aux requêtes.

Afin de satisfaire aux exigences du W3C (organisme de validation des normes utilisées sur Internet), il est possible d'associer une collection de schémas à une colonne de type XML. Les données stockées dans la colonne devront alors respecter les contraintes du schéma. Cette colonne sera alors dite XML typé, sinon il s'agit d'une colonne XML non typé.

XML non typé

Le type XML, tel qu'il est défini dans SQL Server, respecte la définition donnée par le standard ISO SQL-2003. À ce titre, il se doit d'accueillir des documents XML 1.0 bien formés ainsi que des fragments XML.

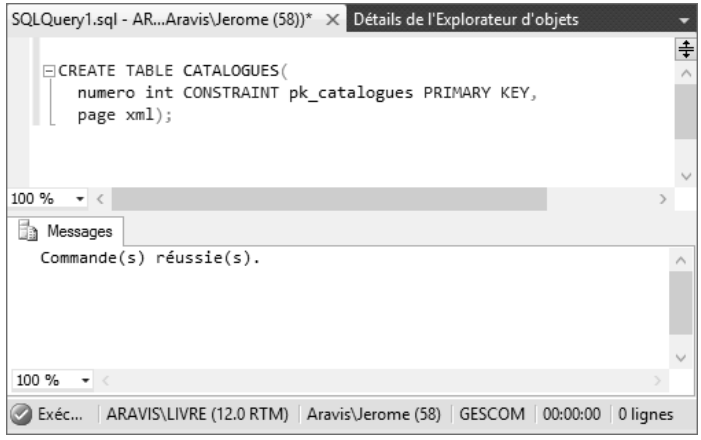
La colonne qui utilise un type XML non typé, c'est-à-dire non relié à un schéma XML, ne contiendra toutefois que des informations conformes à un document XML 1.0 bien formé, ou bien un fragment XML.

Cette méthode de fonctionnement est certainement la plus souple, cependant, lorsque l'on dispose d'un schéma XML, il est préférable de passer par un type XML typé.

Elle doit être utilisée lorsque l'on ne dispose pas de schéma XML, ou lorsqu'il existe plusieurs schémas XML avec des liaisons à des sources de données externes.

Exemple

La table Catalogues est créée avec une colonne de type XML non typé :



XML typé

Dans le cas où les informations qui vont être contenues dans une colonne de type XML, sont décrites dans une collection de schémas XML, il est possible d'associer cette collection de schémas XML à la colonne. En réalisant cette liaison, au niveau de la définition de la colonne, toutes les informations saisies dans la colonne doivent respecter un schéma XML associé. La colonne est alors dite définie sur un type XML typé.

Les schémas XML agissent comme une sorte de contrainte d'intégrité forte, en garantissant une structure clairement identifiée à toutes les informations présentes dans cette colonne. Les mises à jour des informations XML sont également mieux contrôlées et réalisées plus rapidement dans le processus d'exécutions des requêtes.

Lors de la définition de la colonne, il est possible, par l'intermédiaire des mots clés DOCUMENT et CONTENT, de spécifier que la colonne ne va contenir que des documents XML bien formés, c'est-à-dire avec un seul élément au niveau supérieur. Dans le deuxième cas, CONTENT, la colonne contient des données au format XML. Si rien n'est précisé lors de la définition de la colonne, c'est le choix CONTENT qui est appliqué par défaut.

La collection de schémas XML doit être créée avant de pouvoir être référencée par une colonne XML. La gestion des collections de schémas passe par les instructions CREATE XML SCHEMA COLLECTION, ALTER XML SCHEMA COLLECTION et DROP XML SCHEMA COLLECTION. Chaque collection va pouvoir contenir un ou plusieurs schémas XML. Cette utilisation des collections offre une gestion beaucoup plus souple des colonnes XML typées, car il est toujours possible d'ajouter un schéma XML à la collection pour pouvoir répondre à de nouvelles contraintes, fonctionnalités ou formats de gestion d'informations.

Cependant, lors de la définition du schéma, il n'est pas toujours possible de définir l'ensemble des possibilités. SQL Server supporte les déclarations **any**, **anyAttribute** et **anyType** dans la définition de schémas.

Syntaxe

```
CREATE XML SCHEMA COLLECTION nomSchéma
AS définitionDuSchéma;
```