Secteur Tertiaire Informatique
Filière étude - développement

Activité « Développer la persistance des données »

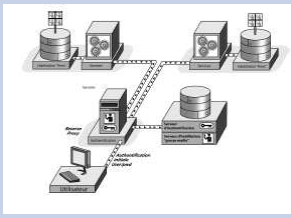
Mise en œuvre des fonctions

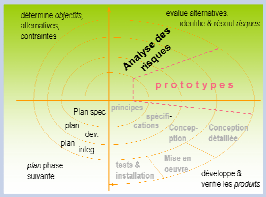
Accueil

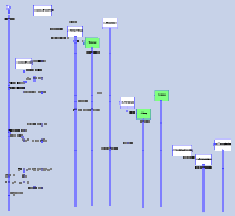
Apprentissage

Période en entreprise

Evaluation







Code barre

SOMMAIRE

Il existe deux types principaux de fonctions :

- Les fonctions scalaires renvoyant une seule valeur
- Les fonctions table, renvoyant un ensemble de lignes et de colonnes que l'on manipule comme une table

Par le biais d'une fonction,aucune modification de table de la base de données n'est possible . Seules les objets locaux à la fonction peuvent être modifiés.

Les fonctions sont créées de la même manière qu'une requête SQL, et se retrouvent sans l'arborescence de **Studio Management** dans le dossier de la base de données, sous-dossier **Programmabilité / Fonctions**°.

I CREATION DE FONCTIONS SCALAIRES

Une fonction scalaire est identique à une fonction mathématique : elle peut avoir plusieurs paramètres d'entrée (jusqu'à 1024), et renvoie une seule valeur.

Exemple: la fonction **fn_DateFormat** formate à l'aide d'un séparateur entré en paramètre, une date et la retourne sous forme de chaîne de caractères.

```
CREATE FUNCTION fn_DateFormat
(@pdate datetime, @psep char (1))
RETURNS char (10)
AS
    BEGIN
        RETURN
            CONVERT (varchar(10), datepart (dd,@pdate))
            + @psep  + CONVERT (varchar (10), datepart (mm, @pdate))
            + @psep  + CONVERT (varchar (10), datepart (yy, @pdate))
    END
```

- Chaque paramètre est défini par son nom obligatoirement préfixé du caractère @ et son type, et séparé du suivant par une virgule
- La clause **RETURNS** spécifie le type de données à retourner : les données de type timestamp, table, text, ntext et image, et définis par l'utilisateur ne peuvent être renvoyés.
- La valeur renvoyée par la fonction est toujours définie par le mot-clé **RETURN**
- La fonction est définie dans un bloc **BEGIN** … **END**

La fonction sera forcément appelée par son nom, préfixé du nom de son propriétaire:

```
SELECT [BaseTest.]dbo.fn_DateFormat(GETDATE(), '/')
```

Une fonction peut contenir un appel à une autre fonction, y compris elle-même, à une procédure stockée, et peut contenir toutes les instructions TRANSACT SQL.

II CREATION DE FONCTIONS TABLE

Les fonctions table sont de deux types :

- les fonctions table **en ligne,** qui ne contiennent qu'une seule instruction SELECT déterminant le format de la table renvoyée.

- Les fonctions table **multi instructions**, qui déclarent le format d'une table virtuelle, avant de la remplir par des instructions SELECT

Dans les deux cas, la fonction accepte des paramètres en entrée, et s'utilise comme une table dans une clause FROM : elles remplacent totalement les vues en y ajoutant la possibilité de passer des paramètres.

Exemple 1 : la fonction table en ligne, **fn_NomArticles** renvoie tous les articles d'une certaine couleur à partir d'une table ARTICLES de structure

ARTICLES (art_num, art_nom, art_coul, art_pa, art_pv, art_qte, art_frs)

```
CREATE FUNCTION fn_NomArticles
  (@pcoul char(10))
RETURNS table
AS
RETURN (
        SELECT art_nom,art_frs
        FROM dbo.articles
        WHERE art_coul =@pcoul)
```

- La clause **RETURNS** spécifie table comme type de données à retourner

- La valeur renvoyée par la fonction toujours définie par le mot-clé **RETURN** est une instruction **SELECT**

- On n'utilise pas de bloc **BEGIN** … **END**

La fonction sera appelée par son nom :

```
SELECT * from fn_NomArticles('ROUGE')
```

Exemple 2 : la fonction table multi instructions, **fn_PrixArticles** restitue selon la valeur de son paramètre , soit le code article, son nom et son prix de vente, soit le code article, son nom et son prix d'achat.

```
CREATE FUNCTION fn_PrixArticles
 ( @ptypeprix char(2) )
RETURNS  @fn_PrixArticles table
        ( art_num smallint PRIMARY KEY NOT NULL,
        art_nom char(20) NOT NULL,
        art_prix money)
AS
BEGIN
IF @ptypeprix ='PV'
        INSERT @fn_PrixArticles Select art_num, art_nom, art_pv from dbo.articles
else if @ptypeprix ='PA'
        INSERT @fn_PrixArticles Select art_num, art_nom, art_pa from dbo.articles
RETURN
END
```

- La clause **RETURNS** spécifie table comme type de données à retourner, la structure étant définie en suivant
- La valeur renvoyée par la fonction toujours définie par le mot-clé **RETURN** est une instruction **SELECT**
- **Le bloc BEGIN** … **END** délimitent les instructions multiples

La fonction sera forcément appelée par son nom préfixé du nom de son propriétaire:

SELECT * from dbo.**fn_PrixArticles**('PV')

III CREATION DE FONCTIONS CLR

L'intégration du CLR (Common Language RunTime) permet d'écrire des fonctions à l'aide de n'importe quel langage .Net framework et implémenté dans SQL Server au sein d'une assembly.

Exemple 1: : Création de la procédure stockée Fn01 du schéma Gescli, qui fait référence à la méthode GetFn01 de la classe LesFonctions se trouvant dans l'assembly SQLAssembly. Avant la création, l'assembly est chargée dans le moteur de base de données.

```
CREATE ASSEMBLY SQLAssembly
FROM 'C:\Documents and Settings\Util\Mes documents\Projets Visual Studio\TestAssembly\DLLTest.dll'
WITH PERMISSION_SET = EXTERNAL_ACCESS

CREATE FUNCTION Gescli.Fn01
AS EXTERNAL NAME SQLAssembly.LesFonctions.GetFn01
```

Exemple 2: Création de la procédure stockée Fn02 du schéma Gescli, qui fait référence à la méthode GetFn02 de la classe LesFonctions se trouvant dans l'assembly SQLAssembly.

```
CREATE FUNCTION Gescli.Fn02 (@str nvarchar(4000))
RETURNS bigint
AS EXTERNAL NAME SQLAssembly.lesFonctions.GetFn02
```

La méthode GetFn02 prend comme paramètre d'entrée une chaîne et retourne en sortie un entier long.

Pour que SQL Server référence la méthode adéquate dans une classe, elle doit présenter les caractéristiques suivantes :

- Recevoir le même nombre de paramètres que ceux spécifiés dans la définition de la fonction
- Recevoir tous les paramètres par valeur, non par référence.
- Utiliser des types de paramètre compatibles avec ceux spécifiés dans la fonction SQL Server.

IV LES OPTIONS DE FONCTION

De manière générale, chaque fonction peut être définie avec des options :

CREATE FUNCTION [nom_schema.] nom_fonction
([{ @nom_parametre [AS][nom_schéma.] type_données }]
RETURNS {type_données | TABLE | @return_variable TABLE }
[WITH { ENCRYPTION | SCHEMABINDING
| RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT]
| EXECUTE AS}]
AS
Corps de la fonction tel que défini §1 et §2

ENCRYPTION
Indique que le Moteur de base de données chiffre les colonnes d'affichage catalogue qui contiennent le texte de l'instruction CREATE FUNCTION (ne peut pas être spécifié pour les fonctions CLR).

SCHEMABINDING
Indique que la fonction est liée aux objets de base de données auxquels elle fait référence. Toutes modification (ALTER) ou suppression (DROP) de ces objets est vouée à l'échec. (ne peut pas être spécifié pour les fonctions CLR).

La liaison de la fonction aux objets auxquels elle fait référence est supprimée uniquement lorsqu'une des actions suivantes se produit :

- La fonction est supprimée.
- La fonction est modifiée, avec l'instruction ALTER, sans spécification de l'option SCHEMABINDING.

Une fonction peut être liée au schéma uniquement si les conditions suivantes sont vérifiées :

- La fonction est une fonction Transact-SQL.
- Les fonctions utilisateur et vues référencées par la fonction sont également liées au schéma.
- La fonction fait référence aux objets à partir d'un nom en deux parties.
- La fonction et les objets auxquels elle fait référence appartiennent à la même base de données.
- L'utilisateur qui exécute l'instruction CREATE FUNCTION dispose de l'autorisation REFERENCES pour les objets de base de données auxquels la fonction fait référence.

RETURNS NULL ON NULL INPUT | CALLED ON NULL INPUT

CALLED ON NULL INPUT est implicite par défaut. le corps de la fonction est exécuté même si NULL est transmis comme argument.

RETURNS NULL ON NULL INPUT indique que SQL Server peut retourner NULL lorsque n'importe lequel des arguments qu'il reçoit a pour valeur NULL, sans réellement appeler le corps de la fonction

La clause **EXECUTE AS** spécifie le contexte de sécurité dans lequel la fonction définie par l'utilisateur est exécutée.

(Référence Sécurité dans SQL Server)

V MODIFICATION / SUPPRESSION DE FONCTION

La commande de modification ALTER FUNCTION accepte les mêmes paramètres que CREATE FUNCTION.

```
ALTER FUNCTION fn_DateFormat
(@pdate datetime, @psep char(1))
RETURNS char(10)
WITH ENCRYPTION
AS
    BEGIN
        RETURN
            CONVERT (varchar(10), datepart(dd,@pdate))
            + @psep  + CONVERT (varchar(10), datepart(mm, @pdate))
            + @psep  + CONVERT (varchar(10), datepart(yy, @pdate))
    END
```

La commande de suppression DROP FUNCTION permet de supprimer une fonction.

```
DROP FUNCTION dbo.fn_DateFormat
```

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