Secteur Tertiaire Informatique
Filière étude - développement

Activité « Développer la persistance des données »

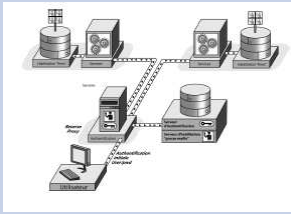
Mise en oeuvre des procédures stockées

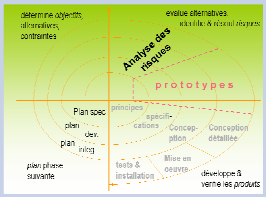
Accueil

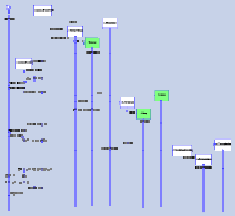
Apprentissage

Période en entreprise

Evaluation







Code barre

SOMMAIRE

Une procédure stockée est un ensemble nommé d'instructions SQL stocké sur le serveur. Elles offrent une méthode efficace pour encapsuler les instructions en vue d'une exécution répétitive. Elles prennent en charge des fonctionnalités de programmation puissante : variables définies par l'utilisateur, exécution conditionnelle...

Les **procédures stockées système** sont stockées dans la base de données master et identifiées par le préfixe sp_

Les **procédures stockées utilisateur** sont créées dans les bases de données concernées.

Les **procédures stockées temporaires** sont créées dans la base de données tempdb et sont supprimées automatiquement.

Les **procédures stockées distantes** sont apelées à partir d'un serveur distant.

A la création, les instructions qui la composent sont vérifiées syntaxiquement. SQL Server stocke ensuite le nom de la procédure stockée dans la vue catalogue **sys.objects** et son texte dans la vue catalogue **sys.sql_module** de la base de données courante : de ce fait, elles sont **accessibles à tous** les clients autorisés.

A la première exécution (ou si elle doit être recompilée), le processeur de requête la lit à partir de la la vue catalogue **sys.sql_module**. Elle est alors compilée.

L'optimiseur de requêtes l'optimise, et stocke alors un plan de requête compilé dans le cache de procédure(zone mémoire utilisée par SQL Server); la procédure stockée s'exécute.

L'exécution suivante des procédures stockées est **plus rapide**, car elles sont exécutées directement à partir du plan de requête optimisé du cache de procédure.

Le **trafic réseau est réduit** : Le client n'a à fournir que le nom et les paramètres de la procédure au lieu des instructions SQL.

I CREATION DE PROCEDURES STOCKEES

Les procédures stockées sont créées dans la base de données courante.
Elles peuvent référencer des tables, des vues, des tables temporaires et des procédures stockées.

Sous **SQL Server Management Studio**, on écrira sa requête et après un test positif, on incluera la création de la procédure : après avoir développé le dossier Bases de données du serveur, et sélectionné le dossier **Procédures stockées** dans le nœud **Programmabilité** de la base de donnée sélectionnée, vous choisirez **Nouvelle procédure** dans le menu contextuel.
Vous pourrez également sélectionner l'assistant Création de procédures stockées pour des procédure stockées de mise à jour de données.

Les procédures stockées sont créées avec l'instruction **Create Procedure** dont la syntaxe partielle est la suivante :

CREATE PROC[EDURE] [nom_schema.] nom_procédure[;number]
[(parametre1 [, parametre2] ... [parametre255])]
[WITH {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION
| EXECUTE AS}]
AS
instructions_SQL
EXTERNAL NAME nom_assembly.nom_classe.nom_méthode

Le nom de la procédure doit être unique dans la base de données, et ne pas excéder 128 caractères de long ; on peut adjoindre au nom un entier, facultatif pour indiquer qu'il s'agit d'une procédure de groupe, par exemple pour des procédures appartenant à la même application .(à éviter).

Exemple : gescli ;1 et gescli ;2 représenteront 2 procédures stockées de l'application gescli

Les paramètres sont optionnels et leur nombre est de 1024 au maximum.
Les paramètres sont définis en tant que paramètres d'entrée, mais peuvent également retourner de l'information au programme appelant.
Un paramètre se déclare selon la syntaxe suivante :

@nom_parametre [nom_schema.] type_de_données [= valeur par défaut] [Output]

nom_parametre : Le nom du paramètre débute obligatoirement par @
[nom_schema.] type_de_données : Type du paramètre (tous les types SQL SERVER sont acceptés, sauf les images)
Si nom_schéma n'est pas précisé, le moteur de base de données de SQL Server 2005 pointe sur le type de données dans l'ordre suivant :

- types de données SQL Server fournis par le système ;
- schéma par défaut de l'utilisateur actuel dans la base de données actuelle ;
- schéma **dbo** dans la base de données actuelle.

valeur par défaut : Spécifie une valeur par défaut du paramètre (Constante ou la valeur NULL).
Output indique que le paramètre est un paramètre de sortie.

WITH RECOMPILE indique que le plan de requête de la procédure ne sera pas stocké en mémoire cache lors de son exécution ; chaque exécution de la procédure stockée donnera lieu à une recompilation.

Il peut être nécessaire de recompiler explicitement une procédure stockée lorsque :

- La procédure stockée est exécutée très rarement
- Elle sera exécutée avec des valeurs de paramètres très différentes (pas optimal d'utiliser le plan d'exécution en mémoire cache)

L'option **WITH ENCRYPTION** empêchera les utilisateurs d'afficher le texte des procédures stockées.
Créer une procédure stockée nécessite l'autorisation CREATE PROCEDURE dans la base de données, et l'autorisation ALTER sur le schéma dans lequel la procédure est créée.

La clause **EXECUTE AS** spécifie le contexte de sécurité dans lequel la fonction définie par l'utilisateur est exécutée.
(Référence Sécurité dans SQL Server)

II EXECUTION D'UNE PROCEDURE STOCKEE

Une procédure stockée peut être exécutée en spécifiant l'instruction EXECUTE (ou EXEC) avec le nom de la procédure et ses éventuels paramètres.

La valeur d'un paramètre d'entrée peut être définie en la passant à la procédure stockée par **référence** ou **position**.

La spécification d'un paramètre dans une instruction EXECUTE en utilisant le format

EXEC[UTE] nom_procédure [@parametre = valeur] [,…n]

s 'appelle un passage par référence : les noms des paramètres sont spécifiés, les valeurs des paramètres peuvent être spécifiées dans n'importe que ordre, les paramètres ayant une valeur par défaut peuvent être omis.

Le passage de valeurs sans référence aux noms des paramètres s'appelle le passage par position :

EXEC[UTE] nom_procédure [valeur] [,…n]

Les valeurs doivent alors être spécifiées dans l'ordre dans lequel les paramètres ont été définis dans le CREATE PROC.

Pour utiliser un paramètre de sortie, le mot clé OUTPUT doit être également spécifié dans l'instruction EXECUTE.

Aucune autorisation n'est requise pour exécuter l'instruction EXECUTE. Cependant, des autorisations sont requises sur les objets référencés dans la chaîne EXECUTE. Par exemple, si la chaîne contient une instruction INSERT, l'appelant de l'instruction EXECUTE doit posséder l'autorisation INSERT sur la table cible.

III EXEMPLES

Exemple 1: Création d'une procédure stockée sans paramètre : liste des employés dont le salaire n'est pas précisé

```
CREATE PROCEDURE Lst_SalNp
AS
      SELECT prénom, nom
      FROM Employés
      WHERE salaire is NULL
```

Cette procédure stockée pourra être exécutée, une fois créée avec l'instruction :

EXEC Lst_SalNp ou

EXECUTE Lst_SalNp

Exemple 2: Création d'une procédure stockée avec un paramètre d'entrée : liste des employés ayant un salaire supérieur à un salaire donné

```
CREATE PROCEDURE Lst_Sal
     @vsal int
AS
      SELECT prénom, nom, salaire
      FROM Employés
      WHERE salaire > @vsal
```

Cette procédure stockée pourra être exécutée une fois créée avec l'instruction :

EXEC Lst_Sal @vsal=16000

ou

EXEC Lst_Sal 16000

(si je veux connaître les salariés gagnant plus de 16000)

Exemple 3: Création d'une procédure stockée avec deux paramètres d'entrée et un paramètre de sortie : Addition de 2 entiers

```
CREATE PROCEDURE Addition
     @fact1 smallint,
     @fact2 smallint,
     @result smallint OUTPUT
AS
     SET @result = @fact1 + @fact2
```

Exécutée avec les instructions suivantes :

```
DECLARE @res smallint
EXECUTE Addition 5,2,@Res OUTPUT
SELECT 'L''addition est égale à :', @Res
```

Le résultat sera : « L'addition est égale à : 7 »

On peut également prévoir une valeur par défaut pour le paramètre : la procédure peut être exécutée sans spécifier de valeur pour ce paramètre.

La valeur par défaut doit être une constante ou elle peut avoir la valeur NULL. Elle peut contenir des caractères génériques si la procédure utilise le paramètre associé au mot clé LIKE : % _ [] et [^].

Exemple 4: L'addition de 2 entiers avec des valeurs par défaut :

```
CREATE PROCEDURE Addition
      @fact1 smallint=8,
      @fact2 smallint=6,
      @result smallint OUTPUT
AS
      SET @result = @fact1 + @fact2
```

Exécutée avec les instructions suivantes :

```
DECLARE @res smallint
EXECUTE Addition @result =@Res OUTPUT
SELECT 'L''addition est égale à :', @Res
```

Le résultat sera : « L'addition est égale à : 13 »

Exemple 5: Création d'une procédure stockée listant tous les employés dont le nom commence par une chaîne de caractères donnés.

```
CREATE PROCEDURE Empl_Commençantpar
      @vrech varchar(23)
AS
      BEGIN
        DECLARE  @vrechc varchar(24)
        SET @vrechc = @vrech + '%'
        SELECT noemp, prénom, nom,
              FROM Employés
              WHERE Nom like @vRechc
      END
```

Exemple 6: Utilisation de la clause EXECUTE AS

La clause EXECUTE AS Indique le contexte de sécurité dans lequel la procédure stockée doit être exécutée.

```
CREATE PROCEDURE Empl_Commençantpar
WITH EXECUTE AS 'KARINE'
      @vrech varchar(23)
AS
        ….
```

Karine ayant l'autorisation Select sur la table Employés, lorsque n'importe quel utilisateur appelle l'exécution de cette procédure, il a accès à la table Employés comme si c'était Karine.

Exemple 6 : Mise à jour d'un employé : on passe en paramètre le code et le nom de l'employé à mettre à jour ; dans la procédure stockée, on contrôlera que la mise à jour a bien été effectuée grâce à la fonction @@ROWCOUNT

```
CREATE PROCEDURE Employé_Update
    @vnum varchar(4)
    @vnom varchar(25)
AS
    BEGIN
        UPDATE Employés SET Nom = @vnom
            WHERE Noemp = @vnum
        IF (@@ROWCOUNT =0)
            BEGIN
                PRINT 'Avertissement : Aucune ligne modifiée'
                RETURN
            END
    END
```

IV UTILISATION DE CODES RETOUR

L'instruction RETURN provoque une sortie. Elle peut renvoyer une valeur d'état de type entier (code de retour).

Un code de retour égal à 0 indique un succès.

Les valeurs comprises entre –1 et –14 sont utilisées par SQL SERVER pour indiquer différentes causes d'échec.

Les valeurs comprises entre –15 et –99 sont réservées à une utilisation future.

Si aucune valeur de retour utilisateur n'est spécifiée, la valeur renvoyée est 0.

Exemple : Contrôle de l'existence d'un employé dans la table Employé : s'il est inexistant, la procédure stockée renvoie le code de retour -100, sinon 0.

```
CREATE PROCEDURE Exist_Emp
    @codemp smallint
AS
    IF NOT EXISTS (SELECT noemp From Employés Where noemp = @codemp)
    RETURN –100
    ELSE
    RETURN 0
```

Exécutée avec les instructions suivantes :

```
DECLARE @retour int
EXECUTE @retour = Exist_Emp 00121
SELECT 'Le code retour est :', @Retour
```

Le résultat sera :

Le code retour est : -100

V LA GESTION DES ERREURS

La fonction @@**ERROR** contient le numéro de l'erreur de la dernière instruction Transact-SQL exécutée. Elle est effacée et ré-initialisée chaque fois qu'une instruction est exécutée. Si l'instruction s'est exécutée avec succès, @@ERROR renvoie la valeur 0.

Exemple : Division par un nombre pouvant être égal à 0

```
CREATE PROCEDURE Divis
    @fact1 smallint
    @fact2 smallint
AS
    SELECT @fact1 / @fact2
    PRINT CASE @@ERROR
        WHEN 8134 THEN 'Division par 0 impossible !'
        WHEN 0 THEN 'Aucune erreur détectée'
        ELSE 'Erreur inconnue'
        END
```

L'instruction RAISERROR renvoie un message d'erreur défini par l'utilisateur et place un indicateur système pour enregistrer le fait qu'une erreur s'est produite.

Elle permet à l'application de lire une entrée dans la table système **sysmessages** ou de construire un message dynamiquement en spécifiant sa gravité et son état. Cette instruction peut écrire des messages d'erreur dans le journal des erreurs de SQL Server et dans le journal des applications de Windows.

Exemple:

```
CREATE PROCEDURE Test
    @codemp smallint
AS
IF NOT EXISTS (SELECT noemp From Employés Where noemp = @codemp)
        BEGIN  /* Employé inexistant */
        RAISERROR ('N°employé incorrect, 16, 1) WITH LOG
        RETURN –100
        END
```

Après l'exécution de ce code, la valeur de retour contiendra la valeur –100 si l 'employé existe. Un message d'erreur sera écrit dans le journal Applications de Windows et dans le journal des erreurs de SQL SERVER.

Pour utiliser un message d'erreur préenregistré, on codera :

```
RAISERROR (n°message, 16, 1)
```

Pour inclure le code de l'employé dans le texte du message, on codera :

```
RAISERROR (n°message, 16, 1, @codemp)
```

Lorsqu'une chaîne est spécifiée dans l'instruction RAISERROR, RAISERROR génère un message d'erreur portant le numéro 50000.

Depuis la version 2005, Les erreurs dans le code Transact-SQL peuvent être traitées à l'aide d'un bloc TRY…CATCH similaire aux fonctionnalités de gestion des exceptions des langages.

Lorsqu'une condition d'erreur est détectée dans une instruction contenue dans un bloc **TRY**, le contrôle est transmis au bloc **CATCH** correspondant où elle peut être traitée.

Une fois que le bloc **CATCH** a traité l'exception, le contrôle est transféré à la première instruction qui suit l'instruction **END CATCH**. Si l'instruction **END CATCH** est la dernière instruction d'une procédure stockée ou d'un déclencheur, le contrôle est renvoyé au code qui a appelé la procédure stockée ou le déclencheur. Les instructions du bloc **TRY** situées après l'instruction qui génère une erreur ne sont pas exécutées.

En l'absence d'erreurs dans le bloc **TRY,** le contrôle est transmis à l'instruction située immédiatement après l'instruction **END CATCH** associée. Si l'instruction **END CATCH** est la dernière instruction d'une procédure stockée ou d'un déclencheur, le contrôle est transmis à l'instruction qui a appelé la procédure stockée ou le déclencheur.

Un bloc **TRY** commence par l'instruction **BEGIN TRY** et finit par l'instruction **END TRY**, et peut inclure une ou plusieurs instructions entre les instructions **BEGIN TRY** et **END TRY**.

Un bloc **TRY** doit être immédiatement suivi d'un bloc **CATCH**. Un bloc **CATCH** commence par l'instruction **BEGIN CATCH** et finit par l'instruction **END CATCH**.

Chaque bloc **TRY** est associé à un seul bloc **CATCH.**

TRY…CATCH utilise des fonctions de gestion d'erreur pour capturer les informations d'erreur.

- **ERROR_NUMBER()** retourne le numéro de l'erreur.
- **ERROR_MESSAGE()** retourne le texte complet du message d'erreur. Le texte comprend les valeurs fournies pour tous les paramètres substituables, tels que les longueurs, les noms d'objets ou les heures.
- **ERROR_SEVERITY()** retourne le niveau de gravité.
- **ERROR_STATE()** retourne le numéro d'état de l'erreur.
- **ERROR_LINE()** retourne le numéro de ligne de la routine à l'origine de l'erreur.
- **ERROR_PROCEDURE()** retourne le nom de la procédure stockée ou du déclencheur dans lequel l'erreur s'est produite.

Exemple 3: Division par un nombre pouvant être égal à 0

```
CREATE PROCEDURE Divis
      @fact1 smallint,
      @fact2 smallint
AS
      BEGIN TRY
        SELECT @fact1 / @fact2
        PRINT ‘Aucune erreur détectée’
      END TRY
      BEGIN CATCH
        SELECT ERROR_NUMBER() as ‘N° erreur’
      END CATCH
```

VI VISUALISATION D'INFORMATIONS

Pour obtenir des informations supplémentaires sur tous les types de procédures stockées, vous pouvez utiliser les procédures stockées système suivantes ou exécuter SQL Server Management Studio.

Procédure stockée	Informations
Sp_help nom_proc	Affiche la liste de paramètres et leur type
Sp_helptext nom_proc	Affiche le texte de la procédure stockée si non cryptée
Sp_depends nom_proc	Enumère les objets qui dépendent de la procédure et les objets dont dépend cette procédure
Sp_stored_procedures	Renvoie la liste des procédures stockées de la base de données en cours

VII CREATION DE PROCEDURES STOCKEES CLR

La clause **EXTERNAL NAME** permet d'utiliser une méthode d'une classe assembly .NET Framework pour créer une référence à une procédure stockée CLR.
Si la classe possède un nom qualifié par un espace de noms utilisant un point (**.**) afin de séparer les parties constituant l'espace de noms, le nom de la classe doit alors être délimité par des crochets (**[ ]**) ou des guillemets droits (**" "**).
La méthode indiquée doit correspondre à une méthode statique de la classe.

Exemple : Création de la procédure stockée ProcX du schéma Gescli, qui fait référence à la méthode GetProcX de la classe LesProc 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 = SAFE

CREATE PROCEDURE Gescli.ProcX
AS EXTERNAL NAME  SQLAssembly.LesProc.GetProcX
```

La clause **EXTERNAL NAME** se code sous la forme

nom_assembly **.**nom_classe.nom_methode

Les paramètres d'une procédure stockée CLR peuvent être de n'importe quel type de données scalaire système SQL Server.

La méthode du CLR doit présenter les caractéristiques suivantes :

- elle doit être déclarée en tant que méthode statique
- elle doit compter le même nombre de paramètres que la procédure
- elle ne doit pas être un constructeur ou un destructeur de sa classe correspondante
- les types de paramètres utilisés doivent être compatibles avec ceux des paramètres correspondant de la procédure SQL Server.
- la méthode doit retourner une valeur « void » ou de type **SQLInt32**, **SQLInt16**, **System.Int32** ou **System.Int16** ;
- elle doit retourner ses paramètres par référence et non par valeur si OUTPUT est spécifié pour toute déclaration de paramètre donné.

VIII MODIFICATION / SUPPRESSION

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

La commande de suppression DROP PROCEDURE permet de supprimer une procédure stockée.

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