Les Procédures Stockées

Table des Matières

1. Définition et avantages des procédures stockées

Une procédure stockée est une collection précompilée d'instructions Transact-SQL stockée sous un nom et traitée comme une unité. Les procédures stockées de SQL Server permettent de gérer celui-ci et d'afficher les informations sur les bases de données et les utilisateurs. Les procédures stockées fournies avec SQL Server sont appelées procédures stockées du système.

Elles renvoient les données de quatre façons différentes :

- des paramètres de sortie, qui renvoient soit des données (entiers ou caractères) ou une variable de curseur, les curseurs étant des ensembles de résultats pouvant être extraits ligne par ligne ;
- des codes de retour, qui sont toujours un entier ;
- un ensemble de résultats pour chaque instruction SELECT contenue dans la procédure stockée ou toute autre procédure stockée appelée par cette dernière ;
- un curseur global qui peut être référencé en dehors de la procédure stockée.

Les curseurs :

Les opérations réalisées dans une base de données relationnelle s'exécutent sur un ensemble complet de lignes. L'ensemble de lignes renvoyé par une instruction SELECT contient toutes les lignes satisfaisant aux conditions de la clause WHERE de l'instruction. Cet ensemble complet de lignes renvoyées par l'instruction est appelé jeu de résultats. Les applications, en particulier les applications interactives en ligne, peuvent ne pas toujours fonctionner efficacement si le jeu de résultats est traité comme une unité. Ces applications ont besoin d'un mécanisme leur permettant de travailler avec une seule ligne ou un petit bloc de lignes à la fois. Les curseurs sont une extension des jeux de résultats et contiennent ce mécanisme.

```
DECLARE cursor_name [ INSENSITIVE ] [ SCROLL ] CURSOR
FOR select_statement
[ FOR { READ ONLY | UPDATE [ OF column_name [ ,...n ] ] } ]
```

Avantages des procédures stockées:

Les procédures stockées contribuent à mettre en œuvre une logique cohérente dans les applications. Les instructions SQL et la logique nécessaires à l'exécution d'une tâche fréquente peuvent être créées, codées et testées une seule fois dans une procédure stockée. Il suffit ensuite à chaque application devant effectuer la tâche d'exécuter la procédure stockée. Le codage de la logique de gestion en une seule procédure offre aussi un point de contrôle unique permettant de vérifier que les règles d'entreprise sont bien respectées.

Les procédures stockées peuvent également améliorer les performances. De nombreuses tâches sont mises en œuvre sous forme de séquences d'instructions SQL. La logique conditionnelle appliquée aux résultats des premières instructions SQL détermine les instructions suivantes à exécuter. Si ces instructions SQL et la logique conditionnelle sont

écrites dans une procédure stockée, elles deviennent partie intégrante d'un plan d'exécution unique sur le serveur. Les résultats n'ont pas besoin d'être renvoyés au client pour que la logique conditionnelle soit appliquée, car tout le travail est réalisé sur le serveur.

Les procédures stockées évitent aussi aux utilisateurs d'avoir à connaître les détails des tables de la base de données. Si un ensemble de procédures stockées prend en charge toutes les fonctions de gestion nécessaires aux utilisateurs, ceux-ci n'ont pas besoin d'accéder directement aux tables ; il leur suffit d'exécuter les procédures stockées qui modélisent les processus avec lesquels ils ont l'habitude de travailler.

Les procédures stockées du système SQL Server évitant aux utilisateurs d'accéder aux tables système en sont un exemple. SQL Server comprend un ensemble de procédures stockées système dont les noms commencent en général par **sp_**. Ces procédures prennent en charge toutes les tâches administratives nécessaires à l'exécution d'un système SQL Server. Vous pouvez administrer un système SQL Server à l'aide des instructions Transact-SQL associées à l'administration (telles que CREATE TABLE) ou des procédures stockées du système, sans jamais avoir à mettre à jour directement les tables système.

2. Les différents types de procédures stockées

2.1. Procédure stockée système

Ensemble de procédures stockées fournies par SQL Server pour la réalisation d'opérations telles que l'extraction d'informations du catalogue système ou l'exécution de tâches d'administration.

Nombre d'activités administratives dans Microsoft® SQL Server™ 2000 s'exécutent à l'aide d'un type spécial de procédure connu sous le nom de procédure stockée système. Les procédures stockées système sont créées et enregistrées dans la base de données **master** et ont le préfixe **sp_**. Les procédures stockées du système peuvent s'exécuter depuis n'importe quelle base de données, sans avoir à qualifier complètement le nom de la procédure stockée, en utilisant le nom de base de données **master**.

Il est fortement recommandé de ne pas créer de procédures stockées avec le préfixe **sp_**. SQL Server recherche toujours une procédure stockée en commençant par **sp_** dans l'ordre suivant :

1. elle existe dans la base de données **master** ;
2. ensuite, en fonction des éventuels identificateurs fournis (nom de base de données ou propriétaire) ;
3. enfin, avec **dbo** comme propriétaire si aucun propriétaire n'est spécifié.

Par conséquent, bien qu'il puisse exister dans la base de données en cours une procédure stockée créée par l'utilisateur ayant le préfixe **sp_**, la base de données **master** est toujours analysée la première, même si la procédure stockée est qualifiée avec le nom de la base de données.

Informations sur les procédures stockées

Pour afficher le texte utilisé pour créer la procédure, exécutez **sp_helptext** dans la base de données dans laquelle la procédure se trouve en passant le nom de la procédure en paramètre.

Pour obtenir une liste des objets référencés par une procédure, utilisez **sp_depends**.

Pour renommer une procédure, utilisez **sp_rename**

> **Important** Si une procédure stockée créée par un utilisateur porte le même nom qu'une procédure stockée système, celle de l'utilisateur ne s'exécutera jamais.

2.2. Procédures stockées temporaires

Les procédures stockées temporaires privées et globales, comme les tables temporaires, peuvent être créées en ajoutant les préfixes # et # # à leur nom. # désigne une procédure stockée temporaire locale, et # #, une procédure stockée temporaire globale. Ces procédures n'existent plus après l'arrêt de SQL Server.

Les procédures stockées temporaires locales sont disponibles au sein d'une seule session d'utilisateur. Tandis que les procédures stockées temporaires globales sur l'ensemble des sessions d'utilisateur.

Les procédures stockées temporaires sont utiles lorsque vous vous connectez à des versions antérieures de SQL Server qui ne prennent pas en charge la réutilisation des plans d'exécution des instructions ou lots d'instructions Transact-SQL.

Pour créer et exécuter les procédures stockées temporaires :

```
Create procedure # #procedure_name

As

sql_statement

Exec sp_executesql # #procedure_name
```

2.3. Procédures stockées distantes

Les procédures stockées distantes sont une ancienne fonctionnalité de Microsoft® SQL Server™ 2000. Leur fonctionnalité dans Transact-SQL est limitée à l'exécution d'une procédure stockée sur une installation SQL Server distante. Les requêtes distribuées introduites dans la version 7.0 de SQL Server prennent en charge cette possibilité ainsi que l'accès à des tables dans des sources de données OLE DB hétérogènes directement à partir d'instructions Transact-SQL locales. Au lieu d'utiliser un appel de procédure stockée distante sur SQL Server 2000, utilisez des requêtes distribuées et une instruction EXECUTE pour exécuter une procédure stockée sur un serveur distant.

Une instance SQL Server 2000 peut envoyer et recevoir des appels de procédures stockées distantes à d'autres instances de SQL Server 2000 et SQL Server version 7.0. Une instance SQL Server 2000 peut également envoyer des appels de procédures stockées distantes vers des instances SQL Server 6.0 ou 6.5 et en recevoir. Un serveur exécutant SQL Server 2000 peut recevoir des appels de procédures stockées distantes d'une instance SQL Server 4.21a, mais l'instance SQL Server 2000 ne peut pas faire des appels de procédures stockées distantes à l'instance SQL Server 4.21a. L'instance SQL Server 4.21a ne peut pas reconnaître la version du flux de données tabulaires (TDS, *Tabular Data Stream*) utilisée par SQL Server 2000.

2.4. Procédures stockées étendues

Les procédures stockées étendues vous permettent de créer vos propres routines externes dans un langage de programmation comme le langage C. Les procédures stockées étendues apparaissent aux utilisateurs comme des procédures stockées normales et s'exécutent de la même façon. Des paramètres peuvent être passés à une procédure stockée étendue pour renvoyer des résultats et un état. Les procédures stockées étendues permettent d'étendre les fonctionnalités de Microsoft® SQL Server™ 2000.

Les procédures stockées étendues sont des bibliothèques de liaison dynamique (DLL, *dynamic-link library*) que SQL Server peut charger et exécuter dynamiquement. Elles s'exécutent directement dans l'espace d'adresse de SQL Server et sont programmées au moyen de l'API Open Data Services de SQL Server.

Une fois que la procédure stockée étendue est écrite, les membres du rôle de serveur fixe **sysadmin** peuvent l'inscrire dans SQL Server, puis donner l'autorisation de l'exécuter à d'autres utilisateurs. Les procédures stockées étendues ne peuvent être ajoutées qu'à la base de données **master**.

Les procédures stockées étendues sont généralement identifiées par le préfixe xp_

3. Codification des procédures stockée

3.1. Création des procédures stockées

Vous pouvez créer une procédure stockée en utilisant l'instruction Transact-SQL CREATE PROCEDURE. Lisez les informations ci-dessous avant de créer une procédure stockée.

- L'instruction CREATE PROCEDURE ne peut pas s'utiliser conjointement avec d'autres instructions SQL dans un même lot d'instructions.
- L'autorisation de créer des procédures stockées revient par défaut au propriétaire de la base de données, qui peut la transmettre à d'autres utilisateurs.
- Les procédures stockées sont des objets de base de données et leur nom doit respecter les règles gouvernant les identificateurs.
- Vous ne pouvez créer une procédure stockée que dans la base de données en cours.

Pour créer une procédure stockée, vous devez préciser :

- es paramètres d'entrée et de sortie de la procédure ou du lot appelant ;
- les instructions de programmation qui exécutent les opérations dans la base de données, y compris l'appel à d'autres procédures ;
- la valeur d'état renvoyée à la procédure ou au lot appelant pour indiquer la réussite ou l'échec et, dans ce cas, la raison de l'échec.

Syntaxe

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
   { @parameter data_type }
AS sql_statement [ ...n ]
```

Arguments

procedure_name

Nom de la nouvelle procédure stockée. Les noms des procédures doivent respecter les règles applicables aux identificateurs et doivent être uniques dans la base de données et pour son propriétaire.

;number

Nombre entier facultatif utilisé pour regrouper les procédures de même nom afin qu'elles puissent être supprimées ensemble à l'aide d'une seule instruction DROP PROCEDURE. Par exemple, les procédures utilisées avec une application appelée **order** peuvent être nommées **orderproc**;1, **orderproc**;2, etc. L'instruction DROP PROCEDURE **orderproc** abandonne le groupe tout entier.

@parameter

Un paramètre de la procédure. Vous pouvez déclarer un ou plusieurs paramètres dans une instruction CREATE PROCEDURE. La valeur de chaque paramètre déclaré doit être fournie par l'utilisateur lors de l'exécution de la procédure (sauf si vous définissez une valeur par défaut pour le paramètre). Une procédure stockée peut comprendre au maximum 2100 paramètres.

Spécifiez un nom de paramètre en plaçant le signe @ comme premier caractère. Ce nom doit respecter les règles gouvernant les identificateurs. Un paramètre est local à une procédure, vous pouvez donc utiliser le même nom dans d'autres procédures.

data_type

Type de données du paramètre. Tous les types de données y compris les types **text**, **ntext** et **image**, peuvent être utilisés comme paramètre dans une procédure stockée.

AS

Spécifie les actions que peut entreprendre la procédure.

sql_statement

Tout numéro et type d'instructions Transact-SQL à inclure dans la procédure. Certaines limitations s'appliquent.

n

Espace réservé qui indique que plusieurs instructions Transact-SQL peuvent être incluses dans cette procédure.

3.2. Modification des procédures stockées

Pour modifier une procédure stockée :

```
ALTER PROC [ EDURE ] procedure_name [ ; number ]
    { @parameter data_type }
AS
   sql_statement [ ...n ]
```

3.3. Suppression des procédures stockées

Pour supprimer une procédure stockée :

```
Drop proc[edure] procedure_name
```

3.4. Appel d'une procédure stockée

Utilisez l'instruction EXECUTE de Transact-SQL pour exécuter une procédure stockée. L'utilisation du mot clé EXECUTE n'est pas nécessaire à cette exécution si la procédure est la première instruction du lot.

```
Exec [ute] procedure_name @parameter 1= value1,@parameter2 = value2,@parameter3 = value3…
```

Des valeurs de paramètres peuvent être fournies si une procédure stockée a été écrite pour les accepter.

> **Remarque** Si vous entrez des paramètres sous la forme *@Parameter = value*, leur ordre n'a pas d'importance. Vous pouvez aussi omettre les paramètres pour lesquels une valeur par défaut a été définie. Si vous spécifiez un paramètre donné sous la forme *@Parameter = value*, vous devez tous les spécifier de cette façon. Sinon, ils doivent apparaître dans l'ordre indiqué par l'instruction CREATE PROCEDURE.

Lorsque le serveur exécute une procédure stockée, il refuse tous les paramètres qui n'étaient pas insérés dans la liste des paramètres au moment de la création de la procédure. Tout

paramètre qui est passé par référence (en fournissant explicitement son nom) ne sera pas accepté si son nom ne concorde pas.

Bien que vous puissiez omettre des paramètres ayant des valeurs par défaut, seule la liste des paramètres peut être tronquée. Par exemple, si une procédure stockée a cinq paramètres, vous pouvez omettre les deux derniers paramètres, mais pas omettre le quatrième et inclure le cinquième, à moins d'utiliser le format *@parameter = value*.

La valeur par défaut d'un paramètre, si elle a été définie dans la procédure stockée, est utilisée dans les cas suivants :

- si aucune valeur n'est spécifiée pour le paramètre au moment de l'exécution de la procédure ;
- si le mot clé DEFAULT est spécifié comme valeur du paramètre.

Pour exécuter une procédure stockée qui est groupée avec d'autres procédures du même nom, indiquez le numéro d'identification de la procédure à l'intérieur du groupe. Par exemple, pour exécuter la seconde procédure stockée du groupe **my_proc**, spécifiez :

```
EXECUTE my_proc;2
```

Pour faire appel à une procédure stockée on utilise le mot *call*

{**call** *procedure_name(@parameter….)*}
{**call** "DataBase"."Owner"."Procedure_name" }

3.5. Définition des paramètres, traitement des erreurs

les procédures ne seraient pas intéressantes si on ne pouvait pas spécifier de paramètres. Heureusement, il est très facile d'écrire une procédure stockée paramétrable.
Pour déclarer un paramètre, il suffit donc de le spécifier dans l'entête de la procédure en lui indiquant :

- son nom : @Parameter (n'oubliez pas le @), qui sera utilisable comme une variable dans la procédure stockée.
- Un type de donnée, choisi parmi les types SQL ou les types utilisateurs.
- Une valeur par défaut optionnelle.
- Une direction, en mettant OUTPUT derrière le nom d'un paramètre.

3.4.1 Paramètres optionnels

Vous pouvez spécifier pour chaque paramètre une valeur par défaut.

3.4.2 Direction des paramètres

Vous pouvez également définir des paramètres de sortie. Ces paramètres sont modifiés dans la procédure stockée puis retournés à l'appelant.

3.6. Utilisation de NOCOUNT, EXISTS

NOCOUNT :

Empêche le message indiquant le nombre de lignes affectées par une instruction Transact-SQL d'être renvoyé en tant que résultat.

Syntaxe

SET NOCOUNT {ON | OFF}

Notes

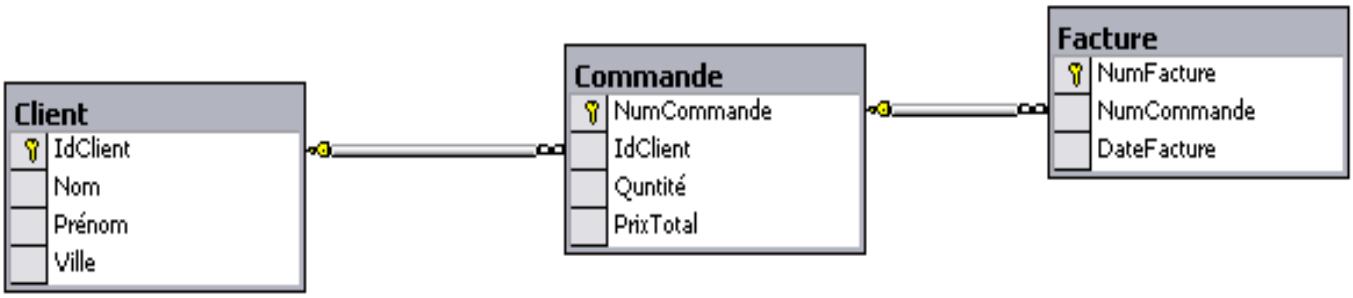
Si SET NOCOUNT est activée (ON), le chiffre indiquant le nombre de lignes affectées par une instruction Transact-SQL n'est pas renvoyé. Si la valeur de SET NOCOUNT est définie sur OFF, ce chiffre est renvoyé.

EXISTS :

Précise une sous-requête pour déterminer l'existence ou non de lignes.

****

Dans les exemples et les exercices qui suivent, on travaille sur la base de données «GestionCommande »:



4. Exemples

```
----Création d'une procédure stockée simple
Create procedure PS;1
as
select * from Client
where IdClient>1650
Exec PS;1
--------------------------------
```

```
----Création d'une procédure avec un seul paramètre
create proc myprocedure
@d datetime
as
select * from Facture where DateFacture=@d
exec myprocedure '08/09/2002'

create proc PS;2
(@Id int)
as
select IdClient,Nom,Prénom from Client where IdClient=@Id
Exec PS;2 @Id=1668
--------------------------------
```

```
----Création d'une procédure avec deux ou plusieurs paramètres
create proc PS;3
(@n varchar(20),
@p varchar(20))
as
select * from Client where Nom=@n and Prénom=@p
Exec PS;3 'Alaoui','Ahmed'
--------------------------------
```

```
----Procédure qui a un paramètre optionnel
Create proc PS;4
(@Id int =null)
As
If @Id is null
        Begin
                Select * from Client
        End
Else Begin
                Select * from Client where IdClient=@Id
        End
Go
exec PS;4 @Id=1925
exec PS;4 1668
exec PS;4
exec PS;4 @Id=default
--------------------------------
```

```
----On crée une procédure stockée porte le même nom d'une procédure stockée système
"sp_depends"
create procedure sp_depends
@n int
as
select * from Commande where NumCommande=@n
exec sp_depends 'Client'
--------------------------------
```

```
----Procédure Stockée temporaire globale
create procedure ##ProcGlobale
as
select * from Client where IdClient=1668
exec sp_executesql ##ProcGlobale
---------------------------------
```

```
----Procédure Stockée temporaire locale
create procedure #ProcLocale
as
select distinct * from Commande
exec sp_executesql #ProcLocale
--------------------------------
```

```
----Appel d'une procédure stockée
{call "GestionCommande"."dbo"."PS;1"}
{call PS;1}
{call PS;3 ('alaoui','ahmed')}
{call myprocedure ('08/09/2002')}
--------------------------------
```

```
-------Procédure avec plusieurs commandes SQL
create proc procédureM
as
begin
select * from Client
end
begin
insert into Client values(3334,'Tourabi','Amina','Casa')
end
-----appel de la procédure
{call procédureM}
-------------------------------------
```

```
------Modification d'une Procédure stockée
alter proc procédureM
as
select * from Commande
-------------------------------------
```

```
-------Suppression d'une Procédure stockée
drop proc procédureM
-------------------------------------
```

```
--------Gestion des erreurs-----------
```

```
Create procedure pp
@aa int
as
if @aa>0
select * from client where IdClient=@aa
else
print'Attention le IdClient que vous avez entrer n"est pas correct!'
Exec pp -55
-------------------------------------
```

5. Exercices

Exercice 1 :

Créer une procédure stockée qui affiche les clients dont la quantité commande est supérieur à 75 et les factures sont réalisées entre 2003 et 2004

Corrigé :

```
create proc E;1
as
select * from Client where exists
(select * from Commande
where IdClient=Client.IdClient and Quntité>75 and exists
(select * from Facture
where datefacture between '01/01/2003'and '31/12/2004'and
NumCommande=Commande.NumCommande ))
exec E;1
```

Exercice 2 :

Créer une procédure stockée qui retourne la somme des prix à payer par tous les clients en utilisant un paramètre de sortie.

Corrigé :

```
create proc E;2
@somme money output
as

        select @somme = sum(PrixTotal)
          from Commande
if @somme < 1000
          print 'La société va fermer ses portes.'
else
          ---SELECT 'La société a réalisé ' + convert (varchar(20), @somme)+' F' as PrixTotal
select @somme as SommeRéalisé
go
declare @P money
```

```
exec E;2  @P output
```

Exercice3 :

Créer une procédure qui affiche les noms et les prénoms des clients dont le nom commence par ‘Al’ en utilisant un cursor qui permet d’extraire les lignes ligne par ligne

Corrigé :

```
create proc ProcCursor
as
DECLARE Cur CURSOR FOR
SELECT Nom, Prénom FROM Client where Nom like 'Al%'
OPEN Cur
FETCH NEXT FROM Cur
WHILE @@FETCH_STATUS = 0
BEGIN
insert into client values(2056,'toto','titi','Mars')
   FETCH NEXT FROM Cur
END
CLOSE Cur
drop proc ProcCursor
exec ProcCursor
```

Exercice 4 :

Créer une procédure qui exécute la procédure précédente

Corrigé :

```
create proc ProcAppelante
@P1 varchar(10)
as
exec @P1
---------
exec ProcAppelante 'ProcCursor'
```

Exercice 5 :

Créer la procédure stockée qui compte le nombre de commandes d’un client et affiche le résultat

Corrigé :

```
create proc prc
(@id int,
@var int output)
as
```

```
        select @var = count(NumCommande)
          from Commande where IdClient=@id
group by IdClient

drop proc prc
declare @P int
Exec prc 1578,@P output
Select @p as NbreCommandes
```