فصل پانزدهم: روال‌های ذخیره شده (Stored Procedure)

روال‌های ذخیره شده یا stored procedure به مجموعه دستورات T-SQL که شامل تعریف متغییرها، عملیات روی آنها و دستورات ورودی و خروجی می‌شود Stored Procedure گفته می‌شود. هر روال ذخیره شده یک شیء است که می‌تواند سیستمی یا غیر سیستمی باشد. روال‌های سیستمی قبلاً تعریف شده‌اند و با ایجاد یک بانک اطلاعاتی جدید به آن بانک اضافه می‌شوند و فقط در T-SQL می‌توانیم آنها را فراخوانی نماییم ولی روال‌های غیر سیستمی توسط خود کاربر تعریف شده و همانند نوع سیستمی می‌توانند بارها و بارها فراخوانی شوند.

در ابتدا شیوه استفاده از stored Procedure را به طور ساده بررسی می‌کنیم و پس از آن به تشریح کامل مطالب می‌پردازیم.

برای ایجاد یک روال ذخیره شده به صورت زیر عمل می‌کنیم:

create procedure نام روال پارامترهای ورودی

as [Output] نوع پارامتر ها

select دستورات

پارامترهای ورودی روال‌ها با علامت @ و نوع داده‌ای تعریف می‌شوند. پارامترهای خروجی بعد از پارامترهای ورودی با علامت @ و نوع داده‌ای و کلمه کلیدی output پس از آن می‌آیند.

مثال زیر استفاده از پارامترهای خروجی را نشان می‌دهد.

```
create procedure  grdesum @sum int output
as
select @sum = sum(grade) from student
```

موقع فراخوانی جمع نمرات به عنوان خروجی برگردانده می‌شود.

مثال زیر استفاده از پارامترهای ورودی را نشان می‌دهد.

```
create   std_info   @name  nvarchar(20) , @family  nvarchar
as
select  std_name , std_family, age  from student
where  std_name = @name  and std_famil=@family
```

در موقع فراخوانی اطلاعات دانش‌آموزانی برگردانده می‌شود که برابر با مقادیر فرستاده شده می‌باشد.

برای فراخوانی procedure بصورت زیر عمل می‌کنیم:

اگر در قطعه کد T-SQL دستور فراخوانی اولین دستور باشد می‌توان فقط نام procedure را نوشت ولی در صورتی که اولین دستور نباشد باید حتماً عبارت exec یا execute را همراه آن نوشت.

فراخوانی روال‌هایی که ورودی دارند به صورت زیر می‌باشد:

exec نام پروسیجر @ نام پارامتر ورودی=مقدار فرستاده شده برای روال

به مثال زیر توجه نمایید:

```
exec    std_info    @name='anis',@family='barmar'
```

برای فراخوانی روال‌ها بدون ورودی به صورت زیر عمل می‌کنیم:

```
exec نام پروسیجر
```

قالب کلی دستور Create Procedure

```
--SQL Server Stored Procedure Syntax
CREATE { PROC | PROCEDURE } [schema_name.]
procedure_name [ ; number ]
   [ { @parameter [ type_schema_name. ] data_type }
      [ VARYING ] [ = default ] [ OUT | OUTPUT | [READONLY]
   ] [ ,...n ]
[ WITH <procedure_option> [ ,...n ] ]
[ FOR REPLICATION ]
AS { [ BEGIN ] sql_statement [;] [ ...n ] [ END ] }
[;]

<procedure_option> ::=
   [ ENCRYPTION ]
   [ RECOMPILE ]
   [ EXECUTE AS Clause ]
--SQL Server CLR Stored Procedure Syntax
CREATE { PROC | PROCEDURE } [schema_name.]
procedure_name [ ; number ]
   [ { @parameter [ type_schema_name. ] data_type }
      [ = default ] [ OUT | OUTPUT ] [READONLY]
   ] [ ,...n ]
```

```
[ WITH EXECUTE AS Clause ]
AS { EXTERNAL NAME assembly_name.class_name.method_name
}
[;]
--SQL Server Natively Compiled Stored Procedure Syntax
CREATE { PROC | PROCEDURE } [schema_name.]
procedure_name
   [ { @parameter data_type } [ NULL | NOT NULL ] [ = default ] [
OUT | OUTPUT ] [READONLY] ] [ ,... n ]
  WITH NATIVE_COMPILATION, SCHEMABINDING,
EXECUTE AS clause
AS
{
  BEGIN ATOMIC WITH (set_option [ ,... n ] )
sql_statement [;] [ ... n ]
 [ END ]
}
 [;]

<set_option> ::=
   LANGUAGE =  [ N ] 'language'
  | TRANSACTION ISOLATION LEVEL =  { SNAPSHOT |
REPEATABLE READ | SERIALIZABLE }
  | [ DATEFIRST = number ]
  | [ DATEFORMAT = format ]
  | [ DELAYED_DURABILITY = { OFF | ON } ]
-- Windows Azure SQL Database Syntax
CREATE { PROC | PROCEDURE } [schema_name.]
procedure_name
   [ { @parameter [ type_schema_name. ] data_type }
   [ VARYING ] [ = default ] [ OUT | OUTPUT ] [READONLY]
   ] [ ,...n ]
[ WITH <procedure_option> [ ,...n ] ]
AS { <sql_statement> [;][ ...n ] }
[;]
<procedure_option> ::=
   [ RECOMPILE ]
   [ EXECUTE AS Clause ]

<sql_statement> ::=
```

```
{ [ BEGIN ] statements [ END ] }
```

قالب کلی دستور Exec:

```
-- SQL Server Syntax

Execute a stored procedure or function
[ { EXEC | EXECUTE } ]
   {
    [ @return_status = ]
    { module_name [ ;number ] | @module_name_var }
     [ [ @parameter = ] { value
                        | @variable [ OUTPUT ]
                        | [ DEFAULT ]
                        }
     ]
    [ ,...n ]
    [ WITH <execute_option> [ ,...n ] ]
   }
[;]

Execute a character string
{ EXEC | EXECUTE }
   ( { @string_variable | [ N ]'tsql_string' } [ + ...n ] )
   [ AS { LOGIN | USER } = ' name ' ]
[;]

Execute a pass-through command against a linked server
{ EXEC | EXECUTE }
   ( { @string_variable | [ N ] 'command_string [ ? ]' } [ + ...n ]
      [ { , { value | @variable [ OUTPUT ] } } [ ...n ] ]
   )
   [ AS { LOGIN | USER } = ' name ' ]
   [ AT linked_server_name ]
[;]

<execute_option>::=
{
      RECOMPILE
   | { RESULT SETS UNDEFINED }
```

```
   | { RESULT SETS NONE }
   | { RESULT SETS ( <result_sets_definition> [,...n ] ) }
}

<result_sets_definition> ::=
{
   (
      { column_name
        data_type
      [ COLLATE collation_name ]
      [ NULL | NOT NULL ] }
      [,...n ]
   )
   | AS OBJECT
      [ db_name. [ schema_name ]. | schema_name. ]
      {table_name | view_name | table_valued_function_name }
   | AS TYPE [ schema_name.]table_type_name
   | AS FOR XML
}
-- Windows Azure SQL Database

Execute a stored procedure or function
[ { EXEC | EXECUTE } ]
   {
    [ @return_status = ]
    { module_name  | @module_name_var }
     [ [ @parameter = ] { value
                  | @variable [ OUTPUT ]
                  | [ DEFAULT ]
                  }
     ]
    [ ,...n ]
    [ WITH RECOMPILE ]
   }
[;]

Execute a character string
{ EXEC | EXECUTE }
   ( { @string_variable | [ N ]'tsql_string' } [ + ...n ] )
   [ AS {  USER } = ' name ' ]
```

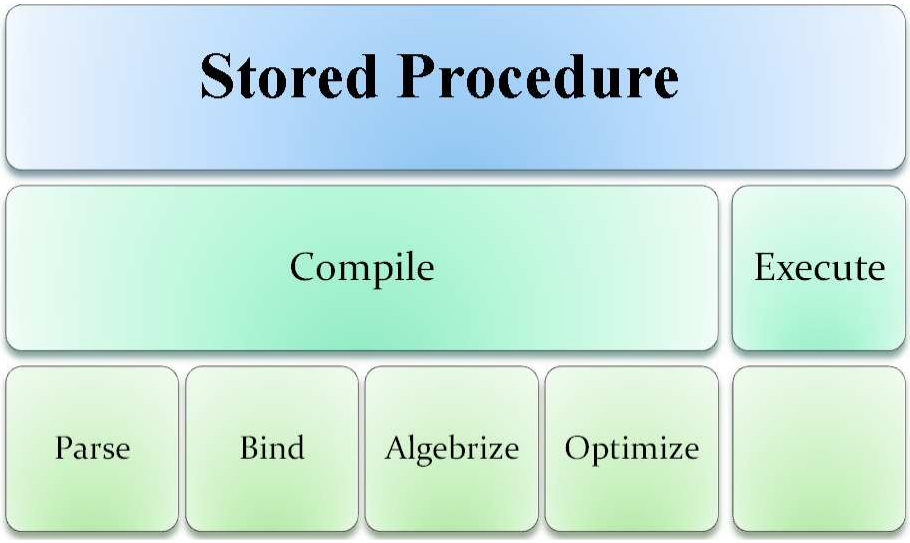
```
[;]

<execute_option>::=
{
      RECOMPILE
   | { RESULT SETS UNDEFINED }
   | { RESULT SETS NONE }
   | { RESULT SETS ( <result_sets_definition> [,...n ] ) }
}

<result_sets_definition> ::=
{
   (
       { column_name
         data_type
       [ COLLATE collation_name ]
       [ NULL | NOT NULL ] }
       [,...n ]
   )
   | AS OBJECT
      [ db_name. [ schema_name ]. | schema_name. ]
      {table_name | view_name | table_valued_function_name }
   | AS TYPE [ schema_name.]table_type_name
   | AS FOR XML
```

مراحل ساخت Stored Procedure

مراحل ساخت Stored Procedure در شکل ۱۵-۱ نشان داده شده است.



شکل (۱۵-۱): مراحل ساخت Stored Procedure

Parse

در این مرحله query از نظر ساختار کلی و syntax کنترل می‌شود.

Bind

کنترل‌هایی از این قبیل کنترل‌های زیر انجام می‌شود:

- موارد مقابل from آیا واقعاً table هستند؟
- کنترل object ها و نحوه استفاده از آنها مانند بود و نبود فیلدها در مقابل select و where یا اینکه جلوی from حتماً نام table یا view باشد.
- کنترل وجود فیلدهای مورد استفاده در جداول

Algebrize

در این مرحله نتیجه نهایی Query مشخص می‌شود و query optimizer می‌فهمد که هدف از انجام این query چیست.

Optimize

در این مرحله estimate ها به دست می‌آید.

باید بدانیم مرحله compile خود زمان‌گیر بوده و SQL به ازای هر Query که کاربر می‌نویسد plan آن را در حافظه ذخیره می‌کند. در Stored Procedure ما مرحله compile را با یک بار اجرای Stored Procedure خواهیم داشت و در بقیه موارد فقط execution داریم.

دستور dbcc FreeProcCache باعث می‌شود تا plan های ذخیره شده در cache خالی شود.

```
set statistics on
select * from master.dbo.syscacheobjects
```

دستور بالا باعث می‌شود موارد داخل cache به ما نشان داده شود. به مثال زیر درباره دستور dbcc FreeProcCache و دستور بالا توجه نمایید:

```
Set statisticstimeon
dbcc FreeProcCache
exec spC_City 'madrid'
exec spC_City 'london'
create proc spC_Country_City
@city nvarchar(50),
@country nvarchar(50)
as
select * from customers
        where city=@city and country=@country
go
exec spC_Country_City 'madrid','spain'

select * from master.dbo.syscacheobjects
```

پارامترها

Stored Procedure می‌تواند مقادیر default برای پارامتر ها داشته باشد و باید بعد از پارامترها بعد از علامت = مقدار default را به کار ببریم، در نهایت Stored Procedure را بدون پارامتر اجرا می‌نماییم. به مثال زیر در این مورد توجه نمایید.

```
Alter proc spC_Country_City
@city nvarchar(50)='london',
@country nvarchar(50)='uk'
as
select * from customers
        where city=@city and country=@country
go
exec spC_Country_City
```

همان طور که در ابتدای فصل گفته شد، چنانچه بعد از تعریف پارامتر ها output بنویسیم به عنوان پارامتر خروجی خواهد بود.

برای دریافت خروجی پارامترها در متغیر باید متغیرها را با declare تعریف کرده و در هنگام exec باید در هنگام اجرای proc بعد از بکار بردن نام متغیرها باید output را استفاده نماییم.

```
exec SP1 6, 8, @V1 output, @V2 output
```

در stored procedure می‌توان یک مقدار را از کوئری دریافت کرده و به عنوان یک result برگرداند. به این منظور باید بعد از select مقدار parameter را قبل از نام فیلد بنویسیم.

```
select @param = count(orderid) from orders
```

در مواقعی که result بیش از یک رکورد باشد به ازای هر رکورد یک بار عمل جایگزینی انجام می‌شود و در نهایت نتیجه آخرین رکورد به عنوان خروجی بر می‌گردد. به مثال‌های زیر توجه نمایید.

```
Alter proc spSUM_Mult
@p1 int,
@p2 int
as
select @p1+@p1 as [SUM],@p1*@p2 as [Mult]
go
exec spSUM_Mult 6,9
alter proc spSUM_Mult
@p1 int,
@p2 int,
@pSUM intoutput,
@pMult intoutput
as
--set @pSUM=@p1+@p1
--set @pMult=@p1*@p2

select @pSUM=@p1+@p2 , @pMULT=@p1*@p2
go
declare @v1 int, @v2 int
exec spSUM_Mult 3,5,@v1 output,@v2 output
select @v1,@v2
create proc northwind.dbo.sp_helpconstraint
as
select null
create proc TestRepl.dbo.a
as
select null
exec northwind.dbo.sp_helpconstraint
create proc sp_a
as
```

```
select 'northwind'
go
exec sp_a
create proc spOC
@cid char(5),
@oc int output
as
select @oc=count(orderid)from orders
        where customerid=@cid
go
declare @oc int
exec spOC  'alfki',@oc output
select @oc
```

در exec کردن یک stored procedure چنانچه پارامترها را همراه با نامشان به stored procedure پاس دهیم، دیگر نیاز نیست همه پارامتر ها را پاس کنیم یا حتی می‌توان پارامتر ها را به صورت جابجا نیز به آنها پاس کرد. به مثال‌های زیر توجه نمایید.

```
exec @p1 = 1, @p3 = NULL
```

مثلاً در اینجا @P2 پاس نشد.

```
exec @p2 = 'A', @p1 = NULL
create non clustered index orders(orderid)
create procedure sp1
@oID
as
select * from orders where ordered >= @oID
go
exec sp 99995
create proc sp1
as
@p1 int=0,
@p2 int=null,
@p3 int
as
....
go
exec sp1 @p3=88,@p1=10
```

Return Value

return value مقداری است برای تشخیص این که خطایی رخ داده یا نه و یا اینکه عملیات با موفقیت انجام پذیرفته است یا خیر.

عدد صفر به منزله انجام موفقیت آمیز کار تلقی می‌شود و هر عددی غیر از صفر یعنی خطایی رخ داده است.

```
CREATE PROCEDURE procedure_name
AS
if …
        return 1;
```

در stored procedure هرگاه به دستور return برسیم stored procedure خاتمه می‌یابد.

برای دریافت return value باید به صورت زیر عمل کنیم:

```
declare @x int;
exec @x = sp …, …
print @x
```

به مثال زیر در مورد return value توجه نمایید.

```
alter  procedure CustomerOrders
        @customerId char(5)
as
        if not exists(select*from customers where CustomerId = @customerId)
                return 0;
        select * from Orders where CustomerId = @customerId
        return 1
go
declare @x int;
exec @x = CustomerOrders 'alfki'

if @x=0
        print 'invalid customer'
exec sp_HelpText 'CustomerOrders'
```

امنیت

برای محافظت کردن از یک stored procedure باید قبل از as عبارت with encryption را استفاده نمود، در غیر این صورت می‌توان با دستور sp_HelpText SPName، source آن stored procedure را دید. به مثال زیر در این مورد توجه نمایید.

```
Alter procedure CustomerOrders
        @customerId char(5)
--with encryption
as
        if not exists(select*from customers where CustomerId = @customerId)
                return 0;
```

```
        select * from Orders where CustomerId = @customerId
        return 1
go
declare @x int;
exec @x = CustomerOrders 'alfki'

if @x=0
        print 'invalid customer'

Exec sp_HelpText'CustomerOrders'
```

With recompile

به Stored Procedure زیر توجه کنید.

```
create proc sp1
as
@p1 int=0,
@p2 int=null,
@p3 int
as
....
go
exec sp1 @p3=88,@p1=10
```

در اولین اجرای sp1 یک plan در cache ایجاد می‌شود و چون look up در leaf ها cost خوبی در این exec دارد بنابراین plan در cache ثبت می‌شود.

در اجرای exec sp1 20 که تعداد رکوردهای زیادی باید look up زده شود، یک cost بالا استفاده می‌شود. در این حالت باید plan جدید در cache ایجاد شود. (در حالت دو scan جدول بهتر است.)

```
Create proc sp1
@p1 int
with recompile
as
....
go
```

نکته ۱: عبارت with recompile باعث می‌شود تا plan جدید در cache ایجاد شود ولی مشکل این که Stored Procedure همیشه recompile شود آن است که این Stored Procedure در این موقع lock می‌شود و بقیه نمی‌توانند از آن استفاده کنند.

نکته ۲: راه حل دوم آن است که ورودی Stored Procedure را به دو دسته

۱. typical مقادیر عادی
۲. untypical مقادیر غیرعادی

تقسیم کنیم و هر گاه بخواهیم Stored Procedure را با یک untypical value، exec کنیم، بعد از exec … with recompile استفاده نماییم. در این حالت plan جدید در cache، override نمی‌شود و plan قبلی باقی می‌ماند.

چنانچه چند select در Stored Procedure داشته باشیم برای اینکه هر query جدا recompile شود، بعد از دستور select ، option (recompile) استفاده می‌شود.

```
Create proc sp1
@p1 int
as
select * from customers --where ...

select * from orders --where ...
        option(recompile)
--....
go
```

چند کار باعث نامعتبر شدن plan ها در cache می‌شود که در زیر لیست شده‌اند.

- dbcc freeproccache: باعث می‌شود تا تمام plan ها خالی شود.
- exec sp_recompile 'sp_name': برای recompile شدن علامت می‌خورد.
- exec sp_recompile 'table_name': تمام stored procedure های استفاده کننده برای recompile شدن علامت می‌خورند.
- روی یک جدول ایندکس بگذاریم.
- راه‌اندازی مجدد سرویس SQL

مثال‌های متنوع در مورد رویه‌های ذخیره شده

Create Procedure Example:

Basic Syntax

A. Creating a simple Transact-SQL procedure

```
IF OBJECT_ID ( 'HumanResources.uspGetAllEmployees', 'P' ) IS NOT
NULL
   DROP PROCEDURE HumanResources.uspGetAllEmployees;
GO
CREATE PROCEDURE HumanResources.uspGetAllEmployees
```