الإجرائيات المخزنة

Stored Procedures

الهدف من الجلسة

في نهاية هذه الجلسة سوف نكون قد عرضنا مفهوم الإجرائيات المخزنـة، التعامـل مـع الإجرائيات المخزنة.

سوف نتعرف في هذه الجلسة على:

1. مقدمة.
2. إنشاء إجرائية مخزنة Creating Stored Procedure.
3. تحديد وسطاء إجرائية Specifying Parameters
   a. تحديد الأسماء Specifying a Name
   b. تحديد الأنماط Specifying a Data Type
   c. تحديد اتجاه الوسطاء Specifying the Direction of a Parameter
   d. تحديد القيم الإفتراضية Specifying a default Value
4. برمجة الإجرائيات المخزنة
   a. تضمين الإجرائيات المخزنة Nesting
5. استعادة بيانات من الإجرائيات المخزنة Returning Data From Stored Procedures.
   a. استعادة بيانات باستخدام OUPUT.
   b. استعادة بيانات باستخدام RETURN.
6. تنفيذ الإجرائيات المخزنة Executing.
7. تعديل وإعادة تسمية الإجرائيات المخزنة Modifying and Renaming.
8. إعادة ترجمة الإجرائيات المخزنة Recompiling.
9. استعراض إجرائية مخزنة Viewing.

مقدمة :

TSQL: Transact- SQL هي لغة برمجة تشكل الواجهة الرئيسية بين أي نظام يتعامل مع قواعد البيانات وقاعدة البيانات.

عند استخدام TSQL هناك طريقتين لتخزين وتنفيذ البرامج:

a. **يمكن تخزين البرامج محليا Locally. ومن ثم إنشاء برامج تقوم بإرسال البرامج المخزنة محليا إلى مخدم قواعد البيانات للتنفيذ من ثم معالجة النتائج.**

b. يمكن تخزين البرامج على مخدم قواعد البيانات Stored Procedures، ومن ثم إنشاء برامج تنفذ هذه البرامج المخزنة ومن ثم معالجة النتائج.

ما هي فوائد تخزين الإجرائيات على مخدم قواعد البيانات بدلا من تخزين الإجرائيات بشكل محلي:

1. **تسمح الإجرائيات المخزنة بالبرمجة الكتلية Modular Programming: يتم إنشاء الإجرائية مرة واحدة، تخزن على مخدم قواعد المعطيات، لتيم استدعاؤها عدد من المرات.**
2. **تنفيذ أسرع للإجرائيات Faster Execution: يتم تهجئة Parsing الإجرائية، تحسن الأداء Optimization، وتنفيذ نسخة من الإجرائية موجودة في الذاكرة In Memory بعد تنفيذها لأول مرة.**
3. **تخفيف الضغط على الشبكة Reduce Network Traffics: إذا كان عملية تتطلب تنفيذ مئات من التعليمات على مخدم قواعد البيانات، فإن الإجرائيات المخزنة تساعد بتنفيذها بالإسم.**
4. **يمكن فرض قيود على التنفيذ Execution Security Mechanism: يمكن منح صلاحيات تنفيذ للإجرائيات المخزنة.**

الإجرائيات المخزنة في مخدم قواعد المعطيات تشبه الإجرائيات في لغات البرمجة من حيث:

1. **لها *وسطاء* متحولات دخل parameters، وتعيد مجموعة من القيم على شكل *وسطاء* متحولات خرج Output Parameters إلى البرنامج الذي قام بتنفيذ الإجرائية.**
2. **تحوي على مجموعة من التعليمات Statements التي تقوم بمجموعة من العمليات Operations على قاعدة البيانات، كما يمكن استدعاء إجرائية مخزنة من قبل إجرائية مخزنة أخرى.**
3. **تعيد الإجرائيات المخزنة عند التنفيذ متحول حالة Status Parameter لتعلم البرنامج المُستَدعِي عن نجاح أو فشل العملية.**

إنشاء إجرائية مخزنة Creating Stored Procedure:

يمكن إنشاء إجرائية مخزنة باستخدام التعليمة CREATE PROCEDURE. قبل إنشاء إجرائية مخزنة يجب أخذ بعين الإعتبار النقاط التالية:

1. لا يمكن أن تحوي مجموعة التعلميات BATCH أوامر أخرى في البداية غير التعليمة CREATE PROCEDURE. مثلا مجموعة التعليمات التالية لا تنفذ:

```
USE NORTHWIND
CREATE PROCEDURE TEMP
'>>-> CREATE/ALTER PROCEDURE' must be the first statement in a query batch.
```

بينما الكتابة التالية صحيحة:

```
USE NORTHWIND
go
CREATE PROCEDURE TEMP
```

2. يجب ان يملك صلاحية مالك قاعدة بيانات DBOWNER، كل من يريد أن ينشأ إجرائية مخزنة.
3. يجب تحديد اسم الإجرائية المخزنة. الإجرائية المخزنة هي غرض من أغرض قاعدة البيانات التي يتبع اسمها قواعد تسمية المتحولات.
4. يمكن إنشاء إجرائية مخزنة في قاعدة البيانات الحالية.

عند إنشاء إجرائية يجب تحديد النقاط التالية:

1. اسم الإجرائية Procedure Name
2. معاملات الدخل Input Parameters
3. معاملات الخرج Output Parameters
4. القيمة المعادة إلى الإجرائية المستدعية للإجرائية الحالية والتي تدل على فشل أو نجاح الإجرائية. كما يجب تحديد رسالة الخطأ المرافقة للقيمة المعادة.

برمجة الإجرائيات المخزنة

قواعد لبرمجة الإجرائيات المخزنة Rules for Programming Stored Procedures:

- لا يمكن تضمين تعليمات الإنشاء CREATE التالية في إجرائية مخزنة :

CREATE PROCEDURE
CREATE TRIGGER
CREATE VIEW
CREATE RULE
CREATE DEFAULT

غير الذي ذكر في الجدول السابق، يمكن تضمين تعليمة CREATE لأي غرض من أغرض قاعدة بيانات. يمكن استخدام أي غرض تم إنشاؤه ضمن إجرائية مخزنة ما دام الإنشاء يتم قبم الإستخدام.

- **يمكن استخدام الجداول المؤقتة ضمن الإجرائيات. الجدول المؤقت هو جدول يبدأ اسمه بـ #.**
- **إذا تم إنشاء جدول مؤقت ضمن إجرائية فإن هذا الجدول يزول بإنتهاء الإجرائية من التنفيذ.**
- **العدد الأعظمي للوسطاء في إجرائية مخزنة 2100 وسيط.**
- **الحد الأعظمي للمتحولات المحلية ضمن إجرائية محدود فقط بسعة الذاكرة المتوفرة.**
- **الحجم الأعظمي لإجرائية مخزنة 128MB.**

ملاحظة

إن استخدام أغراض في إجرائية تحوي تعليمات INSERT, UPDATE, DELETE, SELECT تسبق أسماء هذه الأغرض بشكل افتراضي بـ DBO حيث DBO هو مالك قاعدة البيانات (DATABASE OWNER) مثال: الجدول TITLES ضمن إجرائية يسبق بـ DBO ليصبح DBO.TITLES.

لنفرض أن المستخدم Mary قام بإنشاء الجدول Marks. فإن الجدول في قاعدة البيانات سيصبح اسمه Mary.Marks. وبالتالي استخدام الجدول Marks محصور بـ Mary.

فإذا حاول المستخدم John تنفيذ إجرائية تحوي تعليمة select من الجدول Marks فإن مخدم قواعد البيانات سوف يبحث عن جدول اسمه John.Marks وبالتالي سوف يرسل مخدم قواعد البيانات رسالة مفادها أن الجدول غير موجود.

تحديد وسطاء إجرائية Specifying Parameters

تتخاطب الإجرائية مع البرنامج الذي يستدعيها عبر مجموعة وسطاء Parameters. حيث تمكن الوسطاء البرنامج من تمرير قيم إلى الإجرائية Input Parameters كما تمكنه من الحصول على قيم من الإجرائية Output Parameters.

تحديد الأسماء Specifying a Name

يجب أن تكون وسطاء الإجرائية وحيدة. (أي أنه لا يمكن أن ننشئ إجرائية تحوي على وسيطين بنفس الإسم). جميع وسطاء الإجرائيات يجب أن تبدأ بـ @. كما أن وسطاء الإجرائيات تتبع في التسمية قواعد تسمية المتحولات (تبدأ بحرف، تحتوي على أحرف أو أرقام أو...).

تحديد الأنماط Specifying a Data Type

يتم تحديد نمط لكل وسيط لإجرائية. تماما كما نعرف نمط اسم حقل في جدول. عند استدعاء إجرائية يجب أن تكون القيم التي تمرر إلى الوسطاء تحترم نمط وحجم الوسيط. مثال إذا كان نمط وسيط هو tinyint فإن الوسيط يجب أن يأخذ قيم طبيعية تخزن على 1 Byte.

مثال لاستدعاء إجرائية:

عند استدعاء إجرائية يمكن أن تمرر القيم إلى الإجرائية بطريقتين.

مثال لنفرض أننا أنشأنا إجرائية لها الوسطاء التالية بالترتيب : @first, @second, @third. كما أنه اسم الإجرائية هو sum_proc. يمكن تمرير قيم إلى الإجرائية بإحدى الطريقتين التاليتين:

a. Without Specifying Parameter Names:

```
EXECUTE  sum_proc  7, 1, 3
```

b. Specifying Parameters' Names:

```
EXECUTE  sum_proc  @second = 1, @first = 7, @third = 3
```

في الحالة الأولى تكون قيم وسطاء الدخل : @first = 7, @second = 1, @third = 3

تحديد اتجاه الوسطاء Specifying the Direction of a Parameter

يمكن للإجرائيات المخزنة أن تتلقى قيم عبر الوسطاء من قبل البرنامج المستدعي لها.

مثال:

في قاعدة البيانات Pubs، اكتب إجرائية تقوم بطباعة سعر كتاب ما محدد title.

إن تحليل هذا الطلب يتطلب كتابة إجرائية وليكن اسمها get_sales_for_title، لهذه الإجرائية وسيط هو اسم الكتاب @book، وهو وسيط دخل يتم تزويد قيمته من قبل البرنامج المستدعي للإجرائية. وعليه تكون تعلمية إنشاء الإجرائية على الشكل التالي:

```
-- Ex. 01
use pubs
GO
CREATE PROCEDURE get_sales_for_title
 @book varchar(80) -- This is the input parameter.
AS
BEGIN
 SELECT  price
 FROM titles
 WHERE title = @book

 RETURN
END
```

إن الوسيط @title في المثال السابق هو وسيط دخل، لجعل وسيط لإجرائية وسيط خرج يجب إضافة الكلمة OUTPUT إلى تعريف الوسيط، كما سنرى لاحقا.

تحديد القيم الإفتراضية Specifying a default Value

يمكن إنشاء إجرائيات مخزنة بوسطاء اختيارية Optional. يتم ذلك عبر تحديد قيمة افتراضية للوسطاء. يتم ذلك كما في المثال التالي:

مثال: في قاعدة البيانات Pubs، اكتب إجرائية تقوم بإيجاد سعر كتاب ما title. إذا لم يتم تحديد إي كتاب فإن الإجرائية تعيد معلومات سعر واسم ونوع جميع الكتب.

```
-- Ex. 02
CREATE PROCEDURE my_test_out
 @bookName varchar(80) = NULL  -- This is the input parameter.
AS
BEGIN
 IF @bookName IS NULL
 BEGIN
      SELECT Title, type , price
```

```
   FROM titles
   RETURN 12
 END
    SELECT price
    FROM titles
    WHERE title = @bookName
    RETURN 66
END

exec my_test_out 'Life Without Fear' ;
declare @out int;
exec @out = my_test_out 'Life Without Fear' ;
print @out ;
```

استعادة بيانات من الإجرائيات المخزنة Returning Data From Stored Procedures.

تعيد الإجرائيات المخزنة البيانات إلى البرنامج المستدعي بأربع طرق:

- **وسطاء الخرج OUTPUT PARAMETERS.**
- **القيمة المعادة RETURN CODE والذي هي دوما قيمة INTEGRAL.**
- **من أجل كل تعليمة اختيار SELECT مضمنة في إجرائية هناك مجموعة نتائج RESULT SET مقابلة تعبر عن خرج للإجرائية.**
- **مؤشر عام GLOBAL CURSOR يمكن أن يستخدم كخرج لإجرائية.**

استعادة بيانات باستخدام OUPUT.

مثال: يظهر المثال التالي إجرائية بوسيط دخل ووسيط خرج. المتحول الأول هو book@ وهو وسيط دخل، الوسيط الثاني هو وسيط خرج sales@ الذي يمثل سعر الكتاب book @:

ملا حظة: لا يسمح باضافة حقول أخرى في الاستعلام الذي يسند قيمة لوسيط خرج

```
-- ex 04

use pubs
GO
DROP PROCEDURE out_in
GO
```

```
CREATE PROCEDURE out_in
              @book varchar(80),@sales int output
AS
BEGIN
 SELECT
     @sales = price
 FROM
     titles
 WHERE
     title = @book
 RETURN
END

declare @outtt int;
 -- exec out_in @book= 'Life Without Fear' ,@sales = @outtt output
exec out_in  'Life Without Fear' , @outtt output
print @outtt ;
```

استعادة بيانات باستخدام RETURN.

يمكن لإجرائية مخزنة أن تعيد إلى البرنامج المستدعى قيمة INTEGRAL باستخدام تعليمة RETURN. إن القيمة المعادة باستخدام هذه التعلمية تخزن في متحول يسند إليه تنفيذ الإجرائية:

مثال: في المثال السابق للحصول على القيمة الراجعة عن التعليمة RETURN نعرف متحول عام من نمط int نسند إليه تنفيذ الإجرائية :

```
DECLARE @RES INT
EXEC @RES = GET_SALES_FOR_TITLE …….
```

تنفيذ الإجرائيات المخزنة Executing.

كما هو واضح من الأمثلة السابقة، فإنه لتنفيذ إجرائة مخزنة نستخدم التعليمة EXECUTE أو EXEC. في الحالة التي يكون فيها تنفيذ الإجرائية هو أو ل تعليمة فإنه لا حاجة لإستخدام كلمة EXECUTE.

مثال:

لتنفيذ الإجرائية المعرفة في المثال EX 04: (بالطبع هناك طرق أخرى لتنفيذ إجرائية)

- نعرف متحول لكل متحول دخل للإجرائية
- نعرف متحول لكل متحول خرج للإجرائية
- نعرف متحول نسند إليه القيمة الراجعة من تنفيذ الإجرائية .
- نسند قيم لمتحولات الدخل.
- ننفذ الإجرائية

```
-- EX 05
DECLARE @price_out INT
DECLARE @title VARCHAR(30)
DECLARE @res INT

SET @title = 'Sushi, Anyone?'
EXEC @res = out_in @title, @price_out OUTPUT
IF @ytd_sales IS NULL
   PRINT 'null'
ELSE
PRINT ' sales for ' + @title + CONVERT(VARCHAR(6), @price_out)

PRINT ' the result of return is ' + convert(varchar(6), @res)
```

تعديل وإعادة تسمية الإجرائيات المخزنة Modifying and Renaming.

لحذف إجرائية مخزنة نستخدم التعليمة DROP.

مثال:

```
-- EX 06
DROP PROCEDURE get_sales_for_title
```

لإعادة تسمية إجرائية تحذف الإجرائية ثم تنشأ من جديد. هذا الأمر يؤدي إلى حذف جميع الصلاحيات التي أعطيت للإجرائية.

يمكن استخدام التعليمة ALTER PROCEDURE لتعديل إجرائية دون الحاجة إلى استخدام الأمر DROP PROCEDURE.

إذا أردت أن تشفر الإجرائية التي قمت بإنشائها، بحيث لا يستطيع أحد رؤية محتوى الإجرائية فإن ذلك ممكنا عبر استخدام التعليمة WITH ENCRYPTION.

مثال:

```
-- EX 03
USE NORTHWIND
GO
CREATE PROCEDURE MyProc WITH ENCRYPTION
AS
BEGIN
 SELECT * FROM EMPLOYEES
END
```

تضمين الإجرائيات المخزنة Nesting

التضمين هنا بمعنى الإستدعاء. (كنا قد ذكرنا أنه لا يمكن أن نستخدم تعليمة CREATE PROCEDURE ضمن إجرائية) . يمكن أن يصل عمق الإستدعاء لـ 32 مستوى.

هناك متحول عام @@NESTLEVEL يحوي في كل لحظة مستوى التضمين الحالي. إذا تجاوز مستوى التضمين 32 ، فإن مخدم قواعد البيانات يقوم بإقاف سلسلة الإستدعاءات ويعطي رسالة خطأ.

إعادة ترجمة الإجرائيات المخزنة Recompiling.

يمكن إعادة ترجمة الإجرائيات المخزنة بالتعليمة التالية: sp_recompile.

كما يمكن تحديد with recompile عند إنشاء الإجرائية. الأمر الذي يعني أن الإجرائية سوف يعاد ترجمتها في كل مرة تنفذ.

```
-- ex 07
sp_recompile get_sales_for_title

CREATE PROCEDURE test WITH RECOMPILE
AS
```

```
BEGIN
 RETURN 0
END
```

استعراض إجرائية مخزنة Viewing.

لعرض إجرائية مخزنة نستخدم التعليمة التالية: sp_helptext.

مثال:

```
-- ex 08
sp_helptext get_sales_for_title
```

انتهى