

Stored Procedures

מהן פרוצדורות מאוחסנות?

איך ליצור פרוצדורה כזאת ואיך להשתמש בה (לבצע אותה)?

כיצד לתת קלט ולקבל פלט מפרוצדורה כזו?

מדוע להשתמש בפרוצדורות כאלה ומתי? מה יתרונותיהן ומה חסרונותיהן?

דוגמאות

פרוצדורה מאוחסנת זה בעיקרון אוסף של פקודות בשפה מסוימת (לכל בסיס נתונים יחסי שפה משלו), במקרה של SQL Server זוהי שפת T-SQL

זה כמו פונקציה, או פרוצדורה בשפת תכנות רגילה (Java, C++ וכד') - היא מוגדרת בתוך המנוע של SQL Server, ואפשר לקרוא לה מתוך תוכנית שכתובה נניח ב-C# או VB. אפשר גם לבצע ממסך העבודה של השאילתות.

שפת SQL היא שפה שבעזרתה ניתן לשלוף נתונים ולעדכן בבסיס הנתונים, אולם אינה מיועדת לכתיבת אפליקציה שלמה, עם כללים עסקיים מורכבים. העיסוק עם הנתונים הינו רק חלק מהעיבוד הנדרש. לכן בד"כ השליפה של הנתונים או עידכונם, נעשה מתוך האפליקציה עצמה, בעזרת משפטי SQL אשר 'שתולים' בתוך התוכנית.

למשל: מגיע לקוח חדש לבנק. הפקיד מקליד את נתוני הלקוח למחשב. ישנה אפליקציה (או פונקציה) שיוצרת מספר לקוח חדש על פי לוגיקה מסוימת ועל פי מה שכבר קיים בבסיס הנתונים - כלומר יש כאן גם עיבוד נוסף מלבד קריאת הנתונים.

במקום לשתול פקודות SQL בתוך הקוד בצירוף של עיבוד הנתונים, ניתן לקרוא לפרוצדורה מאוחסנת.

פרוצדורה מאוחסנת יכולה לקבל קלט, כמו למשל מפתח שעבורו רוצים למצוא שורות מתאימות בטבלה כלשהי, ויכולה להחזיר פלט, כמו למשל אוסף שורות שהתאים למפתח. היא יכולה גם לעשות פעולות עידכון, הכנסה או מחיקה של שורות בטבלאות.

פרוצדורה כזו יכולה להכיל מספר משפטי SQL, בנוסף היא יכולה להכיל לוגיקה של תוכנית שתעבד נתונים אלה. פרוצדורה כזו היא אובייקט בתוך בסיס הנתונים ולכן ניתן לתת הרשאות או להגביל מי יכול לקרוא לה.

מתי לא להשתמש בפרוצדורות מאוחסנות

אם האפליקציה אינה נורא מסובכת והקריאות לבסיס הנתונים משתלבות בצורה טבעית עם יתר העיבוד, לעיתים יש יתרון בזה שכל הקוד + הקריאות של ה-SQL נימצאות בקובץ בודד.

אם יש פעולות מורכבות שחוזרות על עצמן במקומות שונים בקוד יתכן שפרוצדורה כזו יכולה להועיל.

נתחיל מהסוף, לבצע פרוצדורה מאוחסנת בשטח העבודה של השאילתה ניתן לכתוב:

```
Execute name-of-proc parms
```

dbo בד"כ מציין: Database Owner (הקידומת של שם הפרוצדורה) זוהי הסכמה (schema) של בסיס הנתונים (משהו כמו Namespace בשפות תכנות). זוהי בד"כ ברירת המחדל אם לא נציין אחרת.

אפשרי גם לכתוב בקיצור:

```
exec name-of-proc parms
```

כדי לבצע, ניבחר את המשפט (כפי שנראה כאן) ונלחץ על Execute!

דוגמה ליצירת פרוצדורה מאוחסנת פשוטה. בד"כ ישנם מספר משפטי הקדמה של השמת ערכים למשתנים סביבתיים כללים שיראו בערך כך

```
USE dbname
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

ולאחר מכן יצירת הפרוצדורה עצמה:

```
CREATE PROCEDURE dbo.getaircrafts
  @aid_given smallint - - parameter given when invoking the proc
AS BEGIN

  SELECT *
  FROM dbo.Aircraft
  WHERE aid > @aid_given - - referencing the parameter
  ORDER BY aid
```

```
END
```

כאשר נבצע את פקודה הנ"ל ונקבל:

```
command completed successfully
```

הוספנו פרוצדורה מאוחסנת בשם:

```
dbo.getaircrafts
```

ניתן לראות את שמה בתיקיה שניקראת: StoredProcedures שנימצאת תחת: Programmability
ה- BEGIN וה- END מתחמים בלוק של קוד, כמו { } בשפות אחרות.

עוד דרך ליצור פרוצדורה מאוחסנת - ע"י קליק ימני על התיקיה: StoredProcedures ובחירת:
new stored procedure

נקבל תבנית כללית של פרוצדורה.

כדי לקחת תבנית של פרוצדורה מסוג מסוים, אפשר לבחור מהתפריט: view ואז: Template Explorer.
זה יגרום לרשימת תיקיות להיפתח. תחת התיקיה: Stored Procedure תהיינה מספר תבניות. קליק כפול על מי מהם, יפתח את התבנית בשטח העבודה.

כדי לשנות פרוצדורה קיימת: קליק ימני + modify - זה מייצר משפט של ALTER PROCEDURE
מהקיימת. ואז אפשר לשנות את גוף הפרוצדורה. ביצוע הפקודה ישמור על הפרוצדורה החדשה (בשם
הישן)

תחת כל תיקיה של פרוצדורה יהיו גם **הפרמטרים** שהיא מקבל והערכים שהיא מחזירה.
צורת הגדרת הפרמטרים נעשית בעזרת **קידומת של @**, בדוגמה:

```
@aid_given smallint
```

גם שם וגם סוג של הפרמטר. וכעת נראה שינוי הפרוצדורה שהגדרנו קודם (בחירת 2 עמודות במקום
הכל)

```
ALTER PROCEDURE dbo.getaircrafts
  @aid_given smallint
AS BEGIN
```

```
  SELECT aid, aname
  FROM dbo.Aircraft
    WHERE aid > @aid_given
  ORDER BY aid
```

```
END
```

כעת, אם נריץ את הפרוצדורה המעודכנת, ניצטרך לספק פרמטר, אחרת נקבל הודעת שגיאה. לכן נכתוב:
exec dbo.getaircrafts 300

וכך נקבל מטוסים בעלי תו זיהוי גדול מ-300.

כעת נוסיף הגדרת משתנים וקצת לוגיקה לפרוצדורה (נגדיר חדשה) זה נעשה בעזרת משפטי DECLARE של טווח הטיסה. הפרוצדורה מקבלת פרמטר של קוד זיהוי המטוס (כמו קודם), והפעם בוחרת רק מטוס אחד על פי הקוד מהפרמטר, ושולפת את טווח הטיסה של מטוס זה, נגדיר גם משתנה של הודעה לפלט. לאחר קבלת טווח הטיסה הלוגיקה בעזרת משפט תנאי, קובעת מה להדפיס לגבי המטוס.

הגדרנו גם משתנה שישמש לפלט מהפרוצדורה ע"י השמת המילה **OUTPUT** לאחריו (@message)

```
CREATE PROCEDURE dbo.flying_range
  @aid_given smallint ,
  @message varchar(30) OUTPUT - -declared in the parameters area
```

```
AS
```

```
  DECLARE @cruising smallint
  SELECT @cruising = cruisingrange   השמת ערך מטבלה למשתנה שהוגדר
  FROM dbo.Aircraft
  WHERE aid = @aid_given
```

```
  IF @cruising > 2000
    BEGIN
      SET @message = 'long distance'   דוגמה להשמת ערכים באופן ידני
    END
  ELSE
    BEGIN
      SET @message = 'short distance'
    END
```

לשם סיכום, השמת ערך למשתנים שהוגדרו בפרוצדורה נעשית או על ידי קריאת הערכים מטבלה (@cruising = cruisingrange), או על ידי פקודת: SET.

השימוש במשתנה פלט של הפרוצדורה

לשם כך, לפני הרצתה, נגדיר משתנה מתאים שיכיל את הפלט, ונכתוב את הקוד הבא (בשטח העבודה, מחוץ לפרוצדורה)

```
DECLARE @out1 varchar(30)
exec dbo.flying_range 305 , @out1 OUTPUT
PRINT 'Output : ' + @out1
```

מה שיודפס יהיה:

Output: long distance

כעת נריץ עם פרמטר של מספר מטוס: 309 וניראה מה נקבל.

ערך מוחזר מפרוצדורה - זהו ערך בודד, משמש לעיתים להחזר קוד שגיאה או מידע על תוצאת הפעלת הפרוצדורה, לדוגמה, אם נירצה להחזיר ערך של כמה שורות ניקראו בשאלתה בתוך הפרוצדורה, נשתמש במשתנה סרבר מיוחד, שלאחר שאילתה מעודכן לכמה שורות הוחזרו (@@ROWCOUNT):

```
DECLARE @RowCout INT
```

```
.....query...
```

```
SET @RowCount = @@ROWCOUNT  
RETURN @RowCount
```

כדי להשתמש בערך המוחזר מהפרוצדורה, נצטרך גם להגדיר משתנה מחוצה לה, שאליו תיכנס תוצאת הביצוע של הפרוצדורה, ואז גם נצטרך לעשות פעולת SELECT עבור ערך זה (גם מחוץ לפרוצדורה) כלהלן:

```
DECLARE @rows INT  
exec @rows = dbo.numaircrafts  
SELECT @rows 'Number of rows returned'
```

ביצוע 3 שורות אלה גם יתן את תוצאת הפרוצדורה וגם תחת הכותרת: Number of rows returned יתן את מספר השורות שחזרו.

CURSOR

מה זה? כיצד לנהל אותו בתוך פרוצדורה?

לעיתים יש צורך בתוכנית לבצע עיבוד של כל שורה שחוזרת כתוצאה משאלתה. תוצאת השאלתה ניקראת: Result set. לשם כך נוצר מבנה שניקרא קרסור (בכל סוג של SQL ישנו תחביר משלו), שאליו ניקראת טבלת התוצאה, ויש מנגנון שנותן לנו אפשרות לעבד את השורות אחת לאחת.

ניראה דוגמה עם פרוצדורה שעובדת על בסיס הנתונים של חברות התעופה.

בעיקרון הטכניקה היא להגדיר את ה-cursor כטבלת תוצאה של שאלתה, להביא ממנו שורה בכל פעם, עד סיום כל השורות, ועבור כל שורה לבצע את העיבוד רצוי. בסיום סגירת cursor.

```

USE [database1]
GO
/***** Object: StoredProcedure [dbo].[cursor_sample]  Script Date: 10/5/2015 4:37:36 PM *****/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =====
-- Author:          <Jacob Shutzman>
-- =====
CREATE PROCEDURE [dbo].[cursor_sample]
    -- Add the parameters for the stored procedure here
    @aid_input smallint
AS
BEGIN
    declare @cruising smallint
    declare @aname varchar(30)
    declare db_cursor cursor for
        select dbo.Aircraft.cruisingrange, dbo.Aircraft.aname
        from dbo.Aircraft
        where aid >@aid_input
        order by Aircraft.aid;
    open db_cursor
        fetch next from db_cursor into @cruising, @aname
        while @@FETCH_STATUS = 0 --gets -1 at end of cursor result
        begin
            print 'Plane: ' + @aname + replicate(' ', 20-len(@aname)) + ' can cruise: ' +
                str(@cruising) + ' miles'
            fetch next from db_cursor into @cruising, @aname
        end
        close db_cursor
        deallocate db_cursor
END

```