Procedury składowane

Tworzenie procedury składowanej

```
CREATE PROCEDURE wyb
AS
SELECT Nazwisko, Imie FROM Osoby
GO
```

Wywołanie

```
EXECUTE wyb
EXEC wyb
wyb
```

Tworzenie procedury składowanej

```
CREATE PROCEDURE wyb
        @nazw varchar(40)
AS
SELECT Nazwisko, Imie FROM Osoby
        WHERE  Nazwisko LIKE  @nazw + '%'
GO
```

```
EXEC wyb 'k'
```

Tworzenie procedury składowanej

```
CREATE PROCEDURE wyb
        @nazw varchar(40)='%',
        @im varchar(20)='%'
AS
SELECT Nazwisko, Imie FROM Osoby
        WHERE  Nazwisko LIKE  @nazw + '%'
                AND Imie LIKE  @im + '%'
GO
```

```
EXECUTE wyb
EXEC wyb
wyb
```

```
EXEC wyb 'k', 'j'
EXEC wyb 'k'
EXEC wyb

EXEC wyb @im = 'j'
```

Procedura składowana – przekazująca zmienną

```
CREATE PROCEDURE licz
    @mini real = 0,
    @ile int
AS
SELECT @ile=COUNT(IdOsoby) FROM Osoby
    WHERE  Wzrost > @mini
GO
```

Pomimo poprawnego wykonania na wyjściu nie pojawiają się żadne dane, ponieważ domyślnie zmienna przekazywana jest tylko do procedury

```
DECLARE @ile int
EXEC licz 1.8, @ile
PRINT @ile
```

Procedura składowana – przekazująca zmienną

```
CREATE PROCEDURE licz
    @mini real = 0,
    @ile int  OUTPUT
AS
SELECT @ile=COUNT(IdOsoby) FROM Osoby
    WHERE  Wzrost > @mini
GO
```

OUT jest synonimem OUTPUT

Pomimo poprawnego wykonania na wyjściu nie pojawiają się żadne dane, ponieważ domyślnie zmienna przekazywana jest tylko do procedury

```
DECLARE @ile int
EXEC licz 1.8, @ile OUTPUT
PRINT @ile
```

Wykonanie zapytania danego zmienną

```
DECLARE @k AS Varchar(20), @zap Varchar(200)
SET @k='Imie'
SET @zap='SELECT ' + @k + ' FROM Osoby'
EXEC (@zap)
```

Procedura pozwalająca na wykonanie zapytania definiowanego dynamicznie

```
CREATE PROCEDURE wykonaj
      @pole varchar(30),
      @tabela varchar(30)
AS
DECLARE @zap varchar(max)
SET @zap='SELECT ' + @pole+ ' FROM ' + @tabela
EXEC (@zap)
GO

EXEC wykonaj 'Nazwisko', 'Osoby'
```

Procedury zagnieżdżone

```
DROP PROCEDURE innerproc
DROP PROCEDURE outerproc
GO
CREATE PROCEDURE innerproc AS
SELECT @@NESTLEVEL AS 'Inner Level'
EXEC outerproc
GO
CREATE PROCEDURE outerproc AS
SELECT @@NESTLEVEL AS 'Outer Level'
EXEC innerproc
GO
EXECUTE outerproc
GO
```

```
The module 'innerproc' depends on
the missing object 'outerproc'. The
module will still be created;
however, it cannot run successfully
until the object exists.
Outer Level
-----------
1
(1 row(s) affected)

Inner Level
-----------
2
(1 row(s) affected)

Outer Level
-----------
3
 (1 row(s) affected)
...
Inner Level
-----------
32
(1 row(s) affected)
```

Poziom zagnieżdżenia
Wartość maksymalna 32

```
Msg 217, Level 16, State 1, Procedure innerproc, Line 3
Maximum stored procedure, function, trigger, or view nesting level
exceeded (limit 32).
```

Procedury zagnieżdżone

```
DROP PROCEDURE innerproc
DROP PROCEDURE outerproc
GO
CREATE PROCEDURE innerproc AS
SELECT @@NESTLEVEL AS 'Inner Level'
EXEC('EXEC outerproc')
GO
CREATE PROCEDURE outerproc AS
SELECT @@NESTLEVEL AS 'Outer Level'
EXEC ('EXEC innerproc')
GO
EXECUTE outerproc
GO
```

```
Msg 217, Level 16, State 1, Line 1
Maximum stored procedure, function,
trigger, or view nesting level
exceeded (limit 32).
```

```
Outer Level
-----------
1
(1 row(s) affected)

Inner Level
-----------
3
(1 row(s) affected)

Outer Level
-----------
5
(1 row(s) affected)

Inner Level
-----------
7
(1 row(s) affected)

Outer Level
-----------
9
.....
Inner Level
-----------
31

(1 row(s) affected)
```

Procedura zwracająca wartość przez nazwę

```
DROP PROCEDURE wysocyp
GO
CREATE PROCEDURE wysocyp
@mm real = 0, @ile int OUTPUT
AS
SELECT @ile=COUNT(wzrost) FROM Osoby
WHERE Wzrost >= @mm
IF @ile>0
RETURN 0
ELSE
RETURN 1
GO
DECLARE @a int, @ile int
EXEC @a=wysocyp 3, @ile OUTPUT
PRINT @ile
PRINT @a

EXEC @a=wysocyp 1.5, @ile OUTPUT
PRINT @ile
PRINT @a
```

```
SKUTEK

0
1

12
0
```

Funkcja zwracająca skalar

```
CREATE FUNCTION wysocy (@mm decimal (3,2) = 0)
RETURNS int AS
BEGIN
DECLARE @ile int
SELECT @ile=COUNT(wzrost) FROM Osoby
WHERE Wzrost >= @mm
RETURN @ile
END
```

Konieczne jest użycie nazwy kwalifikowanej, w przeciwnym przypadku funkcja nie jest wykrywana

Msg 195, Level 15, State 10, Line 2
'wysocy' is not a recognized built-in function name.

WYWOŁANIE

```
DECLARE @a int
SET @a=dbo.wysocy(1.5)
SELECT @a
```

Odwołanie do wartości domyślnej odbywa się przez zastosowanie słowa kluczowego default

```
DECLARE @a int
SET @a=dbo.wysocy(default)
SELECT @a
```

Może zostać wykorzystana w zapytaniu wybierającym

```
SELECT Nazwisko, Wzrost, dbo.wysocy(Wzrost) AS Ile
FROM Osoby
```

Zwracanie w instrukcji warunkowej

```
CREATE FUNCTION spr
(@kto varchar(15))RETURNS int
AS
BEGIN
IF EXISTS (SELECT * FROM Osoby WHERE Nazwisko
    LIKE @kto + '%')
RETURN 1;
ELSE
RETURN 0;
END;
```

Msg 455, Level 16, State 2, Procedure spr, Line 8

The last statement included within a function must be a return statement.

Zwracanie w instrukcji warunkowej

```
CREATE FUNCTION spr
(@kto varchar(15))RETURNS int
AS
BEGIN
IF EXISTS (SELECT * FROM Osoby WHERE Nazwisko LIKE @kto + '%')
RETURN 1;
ELSE
RETURN 0;
RETURN 0;
END;
```

Po dodaniu RETURN działa

Zwracanie w instrukcji warunkowej

```
DROP FUNCTION spr
go
CREATE FUNCTION spr
(@kto varchar(15))RETURNS int
AS
BEGIN
DECLARE @ok int
IF EXISTS (SELECT * FROM Osoby WHERE Nazwisko LIKE @kto + '%')
SET @ok=1
ELSE
SET @ok =0
RETURN @ok
END
```

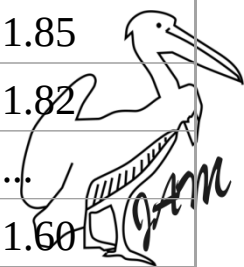
Tak bardziej elegancko

Funkcja zwracająca rekordy (tabelę) – stara postać

```
DROP FUNCTION wysocy_table
GO
CREATE FUNCTION wysocy_table (@mm decimal (3,2) = 0)
RETURNS
@Wysocy  TABLE          (Nazwisko varchar(15),
                         Wzrost decimal(3,2))
 AS
BEGIN
INSERT INTO @Wysocy
SELECT Nazwisko, Wzrost FROM Osoby
WHERE Wzrost >= @mm  ORDER BY Wzrost DESC
RETURN
END
GO

SELECT * FROM dbo.wysocy_table(1.5)
```

Nazwisko	Wzrost
Nowicki	1.93
Nowak	1.93
Majewski	1.87
Szewczyk	1.85
Kowalski	1.82
...	...
Nowak	1.60



Funkcja zwracająca rekordy (tabelę) – nowa postać

```
CREATE FUNCTION wysocyt (@minimum real = 0)
RETURNS TABLE
AS
RETURN
(
        SELECT Nazwisko, Imie, Wzrost
        FROM Osoby WHERE Wzrost >=@minimum
)
GO
SELECT * FROM wysocyt (1.5)
SELECT * FROM wysocyt (default)
```

Nie można stosować klauzuli ORDER BY w definicji zestawu rekordów

Nie jest potrzebne w wywołaniu funkcji zwracającej tabelę stosowanie nazw kwalifikowanych (dbo.wysocyt) wystarczy podać nazwę, ale w skalarnych pozostało, nawet jeśli określono synonim (do takiego synonimu też nazwa kwalifikowana)

Złączenie z funkcją zwracającą rekordy (tabelę)

```
DROP FUNCTION prac
GO
CREATE FUNCTION prac (@dzial int)
RETURNS TABLE
AS
RETURN
(SELECT Nazwisko, Imie  FROM Osoby WHERE IdDzialu = @dzial)
GO

SELECT Nazwa, Nazwisko FROM Dzialy
CROSS APPLY dbo.prac(IdDzialu);
SELECT Nazwa, Nazwisko FROM Dzialy
OUTER APPLY  dbo.prac(IdDzialu);
```

Prefiks dbo opcjonalnie

OUTER daje możliwość wyświetlenia działów, w których nie ma pracowników (analogia LEFT JOIN)

Następnie Procedury wyzwalane – Triggery