

1

Променливи в SQL

Мария Гроздева, 2024г.

Деклариране на променливи

- Име на компания

```
DECLARE @companyName varchar(50)
```

- Дата на започване на курс

```
DECLARE @startDate datetime
```

- Години

```
DECLARE @age int
```

- Цена на нещо

```
DECLARE @cost decimal(5,2)
```

- Можем да декларираме множество променливи „наведнъж“

```
DECLARE @companyName varchar(50), @startDate datetime, @age int, @cost decimal(5,2)
```

Присвояване на стойности на променливи

- ▶ Създаваме променлива и ѝ присвояваме стойност

```
DECLARE @companyName varchar(50)
```

```
SET @companyName = 'Progress'
```

- ▶ Можем алтернативно да използваме запазената дума SELECT

```
DECLARE @companyName varchar(50)
```

```
SELECT @companyName = 'Progress'
```

Инкрементиране/декрементиране стойност на променливи

- ▶ Създаваме брояч

```
DECLARE @counter int = 0
```

- ▶ Изваждаме 1 и добавяме 3

```
SET @counter -= 1
```

```
SET @counter += 3
```

- ▶ Отпечатваме резултата

```
PRINT 'Counter = ' + CAST(@counter AS varchar(10))
```

Живот на променливи

Променливите в SQL „живеят“ само в batch-а, в който са декларирани!

- ▶ Създаваме променлива

```
DECLARE @answer int = 42
```

- ▶ Нов batch

```
GO
```

- ▶ Стойността на променливата?

```
SELECT @answer – Error: Must declare the scalar variable "@answer".
```

УСЛОВНИ КОНСТРУКЦИИ В SQL

6

IF statement

- ▶ Проверка дали дадено условие е истина

IF (condition true)

DO SOMETHING

IF statement (2)

- ▶ Проверка дали дадено условие е истина

IF (condition is true)

BEGIN

DO ONE THING

DO ANOTHER THING

END

IF-ELSE statement

- Проверка дали дадено условие е истина

IF (condition is true)

BEGIN

DO THING A

DO THING B

END

ELSE

BEGIN

DO THING C

DO THING D

DO THING E

END

IF-ELSE statement - пример

► Искаме да проверим дали цветните или черно-белите филми в нашата база Movies са повече

```
DECLARE @monochromeMoviesCount int
```

```
DECLARE @colorfulMoviesCount int
```

```
SET @monochromeMoviesCount = (SELECT COUNT(*) FROM movie WHERE incolor = 'N')
```

```
SET @colorfulMoviesCount = (SELECT COUNT(*) FROM movie WHERE incolor = 'Y')
```

```
IF @colorfulMoviesCount > @monochromeMoviesCount
```

```
    PRINT 'More colorful movies'
```

```
ELSE
```

```
    IF @colorfulMoviesCount < @monochromeMoviesCount
```

```
        PRINT 'More monochrome movies'
```

```
    ELSE
```

```
        PRINT 'Same number of each'
```

CASE-WHEN Statement

- ▶ Отпечатайте с думи дали днешният ден е работен или е уикенд

PRINT

```
CASE DatePart(weekday, GetDate())
```

```
  WHEN 1 THEN 'It"s the weekend!'
```

```
  WHEN 7 THEN 'It"s the weekend!'
```

```
  WHEN 6 THEN 'It"s Friday ...'
```

```
  ELSE 'It"s a weekday ...'
```

```
END
```

ЦИКЛИ В SQL

WHILE loop

- ▶ Декларираме променлива, която ще брой колко пъти се е изпълнило тялото на цикъла

```
DECLARE @counter INT = 0
```

- ▶ Проверяваме дали условието е истина

```
WHILE @counter <= 10
```

- ▶ Ако е, тялото на цикъла се изпълнява

```
BEGIN
```

- ▶ Отпечатваме текущата стойност на брояча

```
PRINT @counter
```

- ▶ Увеличаваме го

```
SET @counter += 1
```

```
END
```

- ▶ Отново се връщаме на реда с условието.

WHILE loop - пример

- За всяка дължина на филм изведете броя на цветните филми

```
DECLARE @minLength INT = (SELECT MIN(length) FROM movie)
```

```
DECLARE @maxLength INT = (SELECT MAX(length) FROM movie)
```

```
DECLARE @currentLengthMoviesCount INT
```

```
WHILE @minLength <= @maxLength
```

```
BEGIN
```

```
    SET @currentLengthMoviesCount =
```

```
        (
```

```
            SELECT COUNT(*)
```

```
            FROM movie
```

```
            WHERE length = @minLength
```

```
        )
```

```
    IF @currentLengthMoviesCount <> 0
```

```
        PRINT
```

```
            'Movies count of length ' + CAST(@minLength AS VARCHAR(3)) + ': ' + CAST(@currentLengthMoviesCount AS VARCHAR(3))
```

```
    SET @minLength += 1
```

```
END
```

WHILE loop – пример (2)

➤ Резултат от изпълнението:

Movies count of length 106: 1

Movies count of length 111: 1

Movies count of length 116: 2

Movies count of length 119: 1

Movies count of length 124: 1

Movies count of length 126: 1

Movies count of length 132: 2

Movies count of length 238: 1

!! Тази задача се решава много по- лесно и ефикасно чрез групиране !!

BREAK Statement

- Прекратява преждевременно изпълнението на цикъл.

```
DECLARE @counter INT = 0
```

```
WHILE 1 = 1
```

```
BEGIN
```

```
    PRINT
```

```
        'Hello from infinite loop, iteration: ' + CAST(@counter AS VARCHAR(3))
```

```
    -- actually the loop is not infinite, will be stopped on the 100th iteration
```

```
    IF @counter = 100
```

```
        BREAK
```

```
    SET @counter += 1
```

```
END
```


Функции в SQL

Scalar-valued Functions

- Задаваме име на функцията

```
CREATE FUNCTION fnFunctionName
```

- Изброяваме аргументите ѝ

```
CREATE FUNCTION fnFunctionName
```

```
(
```

```
    @argumentName AS VARCHAR(MAX) – this parameter is of type VARCHAR
```

```
)
```

- Посочваме типа ѝ на връщане

```
CREATE FUNCTION fnFunctionName
```

```
(
```

```
    @argumentName AS VARCHAR(MAX)
```

```
)
```

```
RETURNS VARCHAR(MAX) – returns a VARCHAR(MAX) value
```

Scalar-valued Functions (2)

- Разписваме тялото ѝ

```
CREATE FUNCTION fnFunctionName  
(  
    @argumentName AS VARCHAR(MAX)  
)  
RETURNS VARCHAR(MAX) – returns a VARCHAR(MAX) value  
AS  
BEGIN  
    -- function body  
END
```

Scalar-valued Functions - пример

- Да се напише функция, която отпечатва подадена като параметър дата „в по- четим формат“.

GO -- The GO command begins a new batch

CREATE FUNCTION fnBetterDate -- CREATE FUNCTION must be the first statement in a batch

(

 @inputDate AS DATETIME

)

RETURNS VARCHAR(MAX)

AS

BEGIN

 RETURN DATENAME(DW, @inputDate) + ' ' +

 DATENAME(D, @inputDate) + ' ' +

 DATENAME(M, @inputDate) + ' ' +

 DATENAME(YY, @inputDate)

END

GO

Scalar-valued Functions – пример (2)

USE ships

```
SELECT name, date, dbo.fnBetterDate(date) AS betterDate  
FROM battles
```

► Результат:

name	date	betterDate
Guadalcanal	1942-11-15 00:00:00.000	Saturday 21 November 1942
North Atlantic	1941-05-25 00:00:00.000	Sunday 25 May 1941
North Cape	1943-12-26 00:00:00.000	Sunday 26 December 1943
Surigao Strait	1944-10-25 00:00:00.000	Monday 2 October 1944

Промяна на дефиницията на вече създадена функция

ALTER FUNCTION fnBetterDate -- Must be in a separate batch!!! Missing here in order to fit the snippet in the slide

```
(
    @inputDate AS DATETIME
)
RETURNS VARCHAR(MAX)
AS
BEGIN
    RETURN DATENAME(DW, @inputDate) + ' ' +
        DATENAME(D, @inputDate) +
        CASE
            WHEN DAY(@inputDate) IN (1, 21, 31) THEN 'st'
            WHEN DAY(@inputDate) IN (2, 22) THEN 'nd'
            WHEN DAY(@inputDate) IN (3, 23) THEN 'rd'
            ELSE 'th'
        END + ' ' +
        DATENAME(M, @inputDate) + ' ' +
        DATENAME(YY, @inputDate)
END
```

Промяна на дефиницията на вече създадена функция (2)

USE ships

```
SELECT name, date, dbo.fnBetterDate(date) AS bestDate  
FROM battles
```

► Резултат:

name	date	bestDate
Guadalcanal	1942-11-15 00:00:00.000	Saturday 21st November 1942
North Atlantic	1941-05-25 00:00:00.000	Sunday 25th May 1941
North Cape	1943-12-26 00:00:00.000	Sunday 26th December 1943
Surigao Strait 1	944-10-25 00:00:00.000	Monday 2nd October 1944

Изтриване на функция

DROP FUNCTION fnBetterDate

ДОМАШНО

- ▶ Функция, която приема пълно име на човек (две имена, разделени с интервал) и връща само първото име (можете да използвате атрибута **name** от таблицата **moviestar**). Помислете за случая, когато измежду подадените за вход данни имате хора, които са записани само с едно име. Вашата функция трябва да работи коректно и за тях.

Временни таблици в SQL

Създаване на временна таблица – първи начин

```
SELECT title, year  
INTO #longMovies -- temporary table is created  
FROM movie  
WHERE length > 120
```

► Преглеждаме всички записи от временната таблица

```
SELECT *  
FROM #longMovies  
ORDER BY length DESC
```

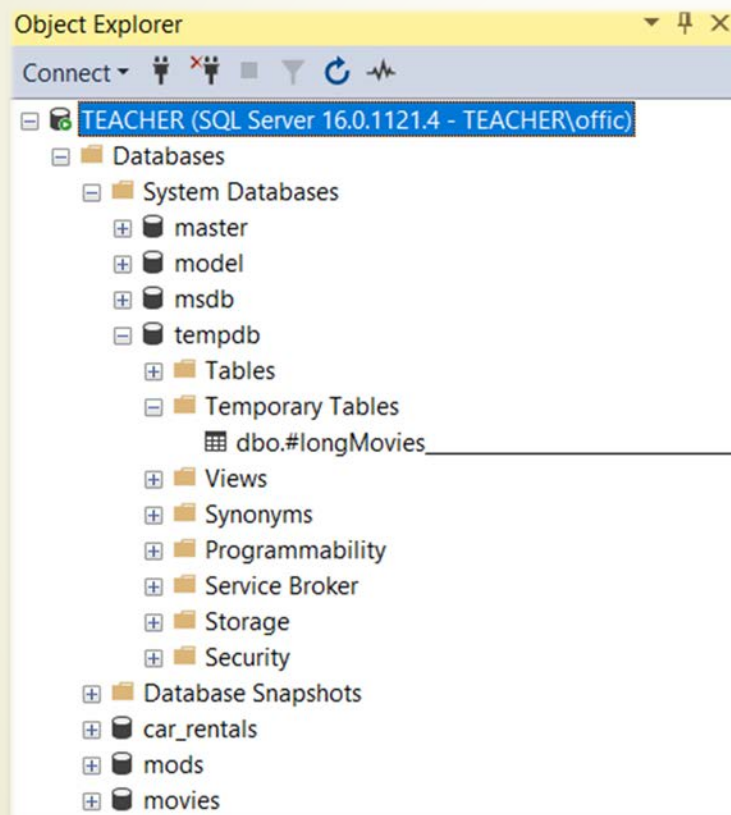
Създаване на временна таблица – втори начин

```
CREATE TABLE #longMovies  
(  
    title VARCHAR(255),  
    year INT,  
    length INT  
)
```

```
-- insert some records into the table
```

```
SELECT *  
FROM #longMovies  
ORDER BY length DESC
```

Къде се съхраняват временните таблици?



Изтриване на временни таблици

- ▶ Временните таблици в SQL се изтриват автоматично след затваряне на сесията, която ги е създадала.
- ▶ Можем ръчно да ги изтрием с
`DROP TABLE #longMovies`

Променливи-таблицы в SQL

31

Създаване

```
DECLARE @longMovies TABLE  
(  
    title VARCHAR(255),  
    year INT,  
    length INT  
)
```

```
INSERT INTO @longMovies  
SELECT title, year, length  
FROM movie  
WHERE length > 120
```


Предимства на променливите-таблици пред временните таблици

- ▶ На следния линк можете да прочете повече за съответните предимства/недостатъци:

[Comparing Table Variables with Temporary Tables](#)

- ▶ Съществено предимство на променливите-таблици пред временните е, че те не се съхраняват в базата и се създават при всяко изпълнение на заявка. Това означава, че, за разлика от временните таблици, при променливите не се интересуваме дали вече съществуват и няма да получаваме грешки, ако се опитаме да изпълним повторно създаващата ги заявка, да ги променим и т.н.

Функции в SQL (2)

Table-valued Functions: in-line-table-valued functions

➤ Синтаксис

```
CREATE FUNCTION fnNameOfFunction  
(  
    @param1 datatype,  
    @param2 datatype, ...  
)  
RETURNS TABLE  
AS  
RETURN  
    -- only one select statement allowed  
    SELECT ...
```

In-line-table-valued functions - пример

- Да се напише функция, която дава информация за корабите, пуснати на вода между конкретни години, подадени като параметри.

```
GO
CREATE FUNCTION fnShipsLaunchedByYear
(
    @minYear INT, @maxYear INT
)
RETURNS TABLE
AS
RETURN
    SELECT name, launched
    FROM ships
    WHERE launched BETWEEN @minYear AND @maxYear
GO
```

In-line-table-valued functions– пример (2)

```
SELECT * FROM dbo.fnShipsLaunchedByYear(1940, 1950)
```

► Результат:

name	launched
Missouri	1944
Wisconsin	1944
Yamashiro	1947

Table-valued Functions: multi-statement table-valued functions (**MSTVF**)

■ Синтаксис

```
CREATE FUNCTION fnNameOfFunction
(
    @param1 datatype, @param2 datatype, ...
)
RETURNS @TableName TABLE
(
    Column1 datatype, Column2 datatype, ...
)
AS
BEGIN
    -- typically insert rows into this table
    -- eventually, return the results
    RETURN
END
```

Multi-statement table-valued functions

- пример

- Да се напише функция, която дава информация за цените на компютрите и лаптопите, по-скъпи от някаква цена, подадена като параметър.

```
CREATE FUNCTION fnExpensivePCsAndLaptops (  
    @price INT  
)  
RETURNS @t TABLE (  
    product VARCHAR(10), code INT, model VARCHAR(4), price FLOAT  
)  
AS  
BEGIN  
    INSERT INTO @t  
    SELECT 'pc', code, model, price FROM pc WHERE price > @price  
    INSERT INTO @t  
    SELECT 'laptop', code, model, price FROM laptop WHERE price > @price  
  
    RETURN  
END
```


Multi-statement table-valued functions

- пример (2)

```
SELECT * FROM dbo.fnExpensivePCsAndLaptops(800)
```

➤ Результат:

Product	code	model	price
pc	2	1121	850
pc	4	1121	850
pc	5	1121	850
pc	6	1233	950
pc	11	1233	980
laptop	2	1321	970
laptop	3	1750	1200
laptop	4	1298	1050
laptop	5	1752	1150
laptop	6	1298	950