доц. д-р Цветанка Георгиева-Трифонова

ПРОГРАМИРАНЕ С TRANSACT-SQL

СЪДЪРЖАНИЕ

- Използване на променливи
- Управление на реда на изпълнение
- Извеждане на съобщения
- Функциите @@ERROR и @@ROWCOUNT

ПРОГРАМИРАНЕ С TRANSACT-SQL

- Стандартът SQL се дефинира от:
  - ANSI (*American National Standards Institute – Американски национален институт по стандартизация*)
  - ISO (*International Organization for Standardization – Международна организация по стандартизация*)
- SQL-86, SQL-89, SQL-92, SQL-99, SQL-2003

ПРОГРАМИРАНЕ С TRANSACT-SQL

- Предимства
  - Подобряване на производителността на приложението
  - Изолиране на приложението от промените

ИЗПОЛЗВАНЕ НА ПРОМЕНЛИВИ

- Деклариране на променливи

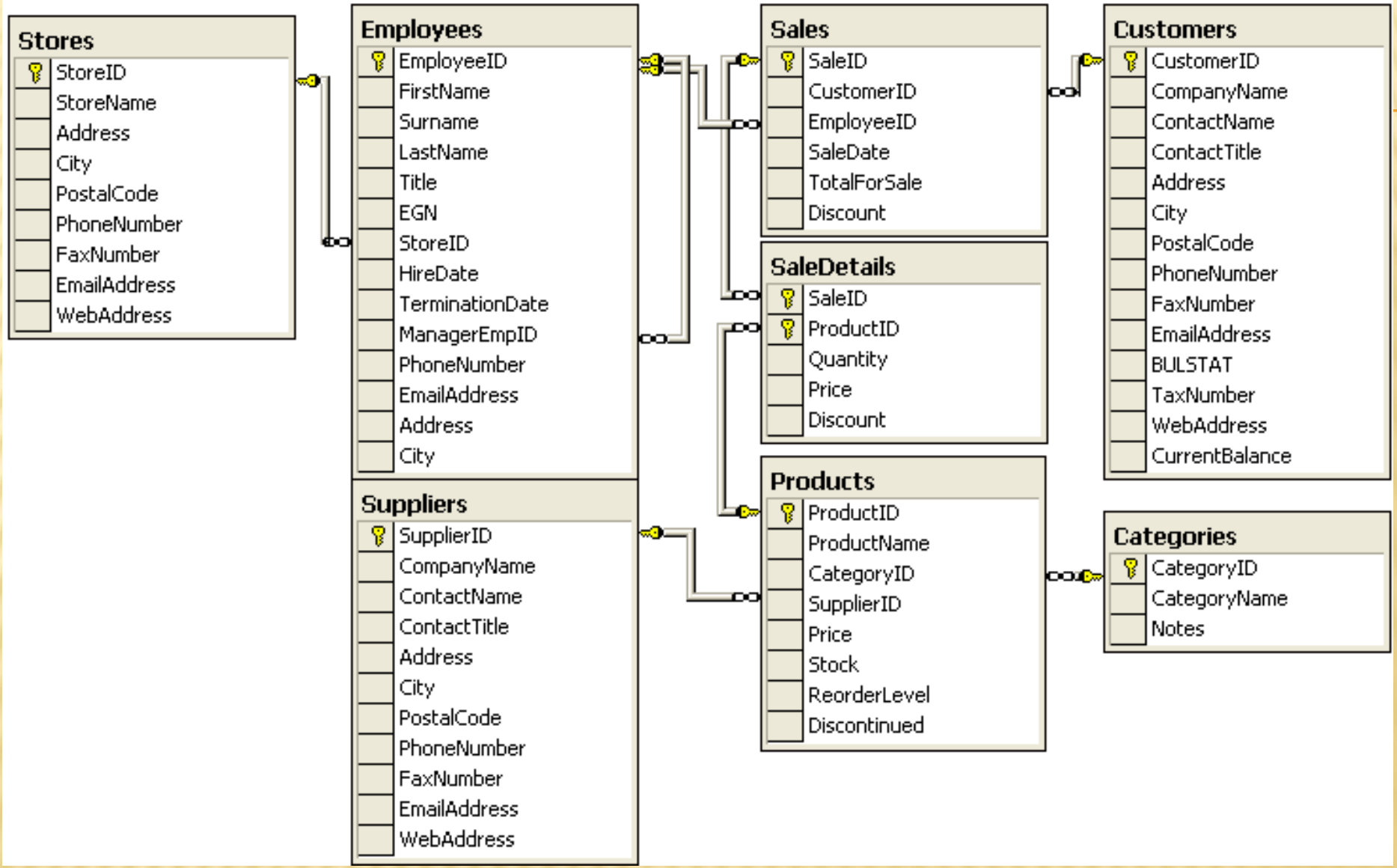
```
DECLARE @var_name datatype
```

- Присвояване на стойност

```
SET @var_name = var_value
```

или

```
SELECT @var_name = var_value
```



База от данни за продажби на продукти

ИЗПОЛЗВАНЕ НА ПРОМЕНЛИВИ – ПРИМЕР

```
DECLARE @min_price money, @max_price money

SELECT @min_price = MIN(price),
       @max_price = MAX(price)
FROM SaleDetails
WHERE ProductID = 1

SELECT @min_price AS MinPrice,
       @max_price AS MaxPrice
```

	MinPrice	MaxPrice
1	1.50	2.00

ИЗПОЛЗВАНЕ НА ПРОМЕНЛИВИ – ПРИМЕР

```
DECLARE @ProductName varchar(40)

SELECT @ProductName = ProductName
FROM Products
WHERE ProductID = 1

SELECT @ProductName

SELECT @ProductName = ProductName
FROM Products
WHERE ProductID = 1898695

SELECT @ProductName -- връща като
```

	(No column name)
1	ябълки

	(No column name)
1	ябълки

ИЗПОЛЗВАНЕ НА ПРОМЕНЛИВИ – ПРИМЕР

```
DECLARE @total money

SELECT
    @total = SUM(sd.price*sd.quantity*(1-sd.discount))
FROM SaleDetails sd
INNER JOIN Sales s ON sd.SaleID = s.SaleID
WHERE DATEDIFF(day, s.SaleDate, GetDate()) = 0

SELECT @total
```

	(No column name)
1	111.65

```
SELECT
    @total = SUM(sd.price*sd.quantity*(1-sd.discount))
FROM SaleDetails sd
INNER JOIN Sales s ON sd.SaleID = s.SaleID
WHERE
   DATEDIFF(day,s.SaleDate,DATEADD(day,1,GetDate()))=0

SELECT @total -- връща като резултат
              -- тъй като няма продажби
```

	(No column name)
1	NULL

УПРАВЛЕНИЕ НА РЕДА НА ИЗПЪЛНЕНИЕ

- условен оператор

```
IF condition
    BEGIN
        statements
    END
ELSE
    BEGIN
        statements
    END
```

УСЛОВЕН ОПЕРАТОР – ПРИМЕР

```
IF EXISTS
        ( SELECT * FROM Customers
          WHERE City = 'Велико Търново'  )
   SELECT COUNT(*) AS CountOfCustomers
   FROM Customers
   WHERE City = 'Велико Търново'
ELSE
   SELECT 'Няма клиенти от избрания град!'
```

ОПЕРАТОР ЗА ЦИКЛИ

```
WHILE condition
    BEGIN
        statements
    END
```

- Пример

```
DECLARE @counter int

SET @counter = 1
WHILE @counter <= 10
   BEGIN
      INSERT INTO Sales
          (CustomerID, EmployeeID, SaleDate)
      VALUES
          (1, 1, DATEADD(mi, @counter, GetDate()))
      SET @counter = @counter + 1
   END
```

ОПЕРАТОР ЗА ИЗХОД ОТ НАЙ-ВЪТРЕШНИЯ ЦИКЪЛ WHILE

BREAK

- Пример

```
DECLARE @counter int

SET @counter = 1
WHILE @counter <= 10
   BEGIN
      SELECT @counter
      SET @counter = @counter + 1
      IF @counter = 7 BREAK
   END
```

- Върнатите като резултатен набор стойности са 1, 2, 3, 4, 5 и 6.

ОПЕРАТОР ЗА РЕСТАРТИРАНЕ НА ЦИКЪЛА WHILE

CONTINUE

- Пример

```
DECLARE @counter int

SET @counter = 0
WHILE @counter < 10
    BEGIN
       SET @counter = @counter + 1
       IF @counter = 7 CONTINUE
       SELECT @counter
    END
```

- Върнатите като резултатен набор стойности са 1, 2, 3, 4, 5, 6, 8, 9 и 10.

ОПЕРАТОР ЗА БЕЗУСЛОВЕН ПРЕХОД

…

```
GOTO label_name
```

…

```
label_name:
```

…

ОПЕРАТОР ЗА БЕЗУСЛОВЕН ИЗХОД

`RETURN [n]`

- използва се в съхранена процедура или тригер;
- цялото число *n*
  - установява се като изходен статус, който да бъде присвоен на променлива при изпълняването на съхранената процедура;
  - ако се пропусне, RETURN връща 0.

ОПЕРАТОР ЗА УСТАНОВЯВАНЕ НА ВРЕМЕ ЗА ИЗПЪЛНЕНИЕ НА КОНСТРУКЦИЯТА

WAITFOR

- Времето може да бъде:
  - интервал (до 24 часа), тогава операторът има вида:

    WAITFOR DELAY '*интервал за изчакване*'
  - определен час от деня, тогава операторът има вида:

    WAITFOR TIME '*час*'
- Примери

```
WAITFOR DELAY '5:20'
WAITFOR TIME '14:00'
DECLARE @Morning datetime
  SET @Morning = '12:06:30'
  WAITFOR TIME @Morning
  SELECT @Morning
```

ОПЕРАТОР ЗА ОБРАБОТКА НА ГРЕШКИ

```
BEGIN TRY
      statements
END TRY
BEGIN CATCH
      statements
END CATCH
```

- Ако възникне грешка в TRY блока, изпълнението преминава към групата от конструкции, заградени в CATCH блока.

ОПЕРАТОР ЗА ОБРАБОТКА НА ГРЕШКИ

- Системни функции за получаване на информация за грешката, предизвикала изпълнението на CATCH блока:
  - ERROR_NUMBER() – номера на грешката;
  - ERROR_SEVERITY() – степента на строгост на грешката;
  - ERROR_STATE() – кода на състоянието, свързан с грешката;
  - ERROR_PROCEDURE() – наименованието на съхранената процедура или тригера, в който е възникнала грешката;
  - ERROR_LINE() – номера на реда, в който е възникнала грешката;
  - ERROR_MESSAGE() – пълния текст на съобщението на грешката. Текстът включва стойностите, получени от всички параметри за заместване.

ОПЕРАТОР ЗА ОБРАБОТКА НА ГРЕШКИ – ПРИМЕР

- **Пример** Нека в таблицата *SaleDetails* съществува ред със стойност 1 в *ProductID*.

```
BEGIN TRY
    -- Генерира грешка за нарушаване на ограничение външен ключ.
    DELETE FROM Products
    WHERE ProductID = 1
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() AS ErrorNumber,
           ERROR_SEVERITY() AS ErrorSeverity,
           ERROR_STATE() AS ErrorState,
           ERROR_PROCEDURE() AS ErrorProcedure,
           ERROR_LINE() AS ErrorLine,
           ERROR_MESSAGE() AS ErrorMessage
END CATCH
```

ОПЕРАТОР ЗА ОБРАБОТКА НА ГРЕШКИ – ПРИМЕР

- Резултатът е:

ErrorNumber	ErrorSeverity	ErrorState	ErrorProcedure	ErrorLine	ErrorMessage
547	16	0	NULL	3	The DELETE statement conflicted with the REFERENCE constraint "FK_ProductInformation_Product". The conflict occurred in database "MyDatabase", table "dbo.ProductInformation", column 'ProductID'.

ОПЕРАТОР ЗА ОБРАБОТКА НА ГРЕШКИ – ПРИМЕР

- **Пример** TRY … CATCH конструкцията не обработва грешки при компилиране (като например синтактичните грешки):

```
BEGIN TRY
    SELECT * FROM NonExistentTable
END TRY
BEGIN CATCH
     SELECT ERROR_NUMBER() AS ErrorNumber,
            ERROR_MESSAGE() AS ErrorMessage
END CATCH
```

- Резултатът е системното съобщение за грешка, която възниква при използване на несъществуваща таблица:

```
Msg 208, Level 16, State 1, Line 2
Invalid object name 'NonExistentTable'.
```

ОПЕРАТОР ЗА ОБРАБОТКА НА ГРЕШКИ – ПРИМЕР

```
BEGIN TRY
    DECLARE @sql varchar(2000)
    SET @sql = 'SELECT * FROM NonExistentTable'
    EXEC (@sql)
END TRY
BEGIN CATCH
     SELECT ERROR_NUMBER() AS ErrorNumber,
            ERROR_MESSAGE() AS ErrorMessage;
END CATCH
```

- Резултатът от изпълнението на последния код е:

```
ErrorNumber      ErrorMessage
-----------------------------------------------------------------------------
208              Invalid object name 'NonExistentTable'.
```

ИЗВЕЖДАНЕ НА СЪОБЩЕНИЯ

- PRINT
  - визуализира символните низове до 8000 символа.
- Примери

```
PRINT 'низ-съобщение'
PRINT 'Today is ' +
      CONVERT(char(30), GetDate())
```

ИЗВЕЖДАНЕ НА СЪОБЩЕНИЯ

```
RAISERROR( {msg_num OR msg_str},
           severity_num, state
          [, argument1 [,… ]] )
[WITH option [,… ]]
```

- *msg_num* – номер на грешката; числата 0÷50000 се използват от SQL Server и за потребителя са тези над 50000. Максималната възможна стойност е 2^{31}-1;
- *msg_str* – низ за описание на грешката (до 2047 символа);
- *severity_num* – степен на строгост на грешката (0÷25);
- *state* – число от 1 до 127, обозначаващо състоянието на системата в момента на възникване на грешката;
- опции:
  - LOG – записва кода на грешката в *log* файл за грешки на SQL Server;
  - SETERROR – задава стойност на @@ERROR с посочения номер на грешката.

ИЗВЕЖДАНЕ НА СЪОБЩЕНИЯ

- параметри за заместване
  - знакът % показва, че следващият го знак е позицията, която трябва да се заеме от заместващата стойност, както и типа данни на заместващата стойност:
    - d или i – цяло число със знак;
    - o – осмично число без знак;
    - s – символен низ;
    - u – цяло число без знак;
    - x или X – шестнадесетично число без знак.
- Примери:

```
RAISERROR ('Message', 1, 1)
RAISERROR ('Error', 16, 1)
RAISERROR (50001, 1, 1) WITH SETERROR
RAISERROR('Идентификаторът на клиента е %d.', 11,
  1, 87)
```

ИЗВЕЖДАНЕ НА СЪОБЩЕНИЯ – ПРИМЕР

```
DECLARE @s decimal(10, 3), @sn varchar(40),
        @st varchar(10)

SELECT @sn = ProductName, @s = Stock
FROM Products
WHERE ProductID = 1

SET @st = LTRIM(STR(@s, 10, 3))

IF @sn IS NOT NULL
   RAISERROR ('The %s''s stock is %s.', 11, 1, @sn, @st)
ELSE RAISERROR('The product doesn''t exist.', 16, 1)
```

- Резултатът е:

```
Server: Msg 50000, Level 11, State 1, Line 10
The product_name's stock is 318.000.
```

ФУНКЦИЯ @@ERROR

- **@@ERROR** – номера на грешката, генерирана от последната конструкция в текущата конекция. Стойността на функцията е 0, ако не е генерирана грешка.
- Пример

```
DECLARE @error int

INSERT INTO SaleDetails
     (SaleID, ProductID, Price, Quantity, Discount)
VALUES (999999, 11, 10.00, 10, 0)
/*  @@ERROR се установява на кода на грешката, получена от
    изпълнението на тази конструкция.  */

--  Кодът на грешката се съхранява в променлива.
SELECT @error = @@ERROR

PRINT 'The value of @error is ' +
       CONVERT(varchar(10), @error)
```

ФУНКЦИЯ @@ROWCOUNT

- @@ROWCOUNT – броя на избраните или засегнатите редове от последната конструкция в текущата конекция.
- Пример

```
SELECT * FROM Customers
SELECT @@ROWCOUNT
SELECT @@ROWCOUNT
```

ПРИМЕРИ

- Да се определи резултата от изпълнението на следния код:

```
DECLARE @string varchar(20)

SET @string = 'SearchDatabase.com'

SELECT LEFT(@string, 6) AS FirstSix,
       RIGHT(@string, 4) AS LastFour
```

	FirstSix	LastFour
1	Search	.com

ПРИМЕРИ

- Да се определи резултата от изпълнението на следния код:

```
DECLARE @string varchar(25)

SELECT @string = '     SearchDatabase.com    '

SELECT LTRIM(@string) AS NoLeadingSpaces,
       RTRIM(@string) AS NoTrailingSpaces
```

	NoLeadingSpaces	NoTrailingSpaces
1	SearchDatabase.com	SearchDatabase.com

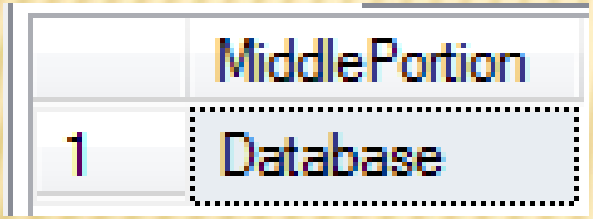
ПРИМЕРИ

- Да се определи резултата от изпълнението на следния код:

```
DECLARE @string varchar(20)

SELECT @string = 'SearchDatabase.com'

SELECT SUBSTRING(@string, 7, 8) AS MiddlePortion
```



ПРИМЕРИ

- Да се определи резултата от изпълнението на следния код:

```
DECLARE @string varchar(20)

SELECT @string = 'SearchDatabase.com'

SELECT SUBSTRING(@string, CHARINDEX('d', @string),
       LEN(@string) -
       ( CHARINDEX('d', @string) - 1 ) -
         CHARINDEX('.', REVERSE(@string) ) )
       AS MiddlePortion
```

	MiddlePortion
1	Database

ЗАДАЧИ

- *1*. Да се напише код за извеждане на съобщение за доставната цена на продукт с идентификатор 10.
- *2*. Да се напише код, чрез който да се определи броя на думите в символен низ, съдържащ думи, разделени със запетая, например '*sqlserver,html,xml,access,adp*'.
- *3*. Да се напише код, чрез който да се отделят думите в символен низ, съдържащ разделител запетая, например '*sqlserver,html,xml,access,adp*'.
- *4*. Да се напише код за извеждане на символен низ по такъв начин, че всички думи в него да започват с главни букви.