ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Database Systems

Chapter 5

T-SQL Programming

Session 1:

Introduction, Variables, Control-Flow statements

http://vku.udn.vn/

Outline

1. Introduction to T-SQL Programming
2. Using Variables
3. Control-Flow statements

Introduction to T-SQL Programming

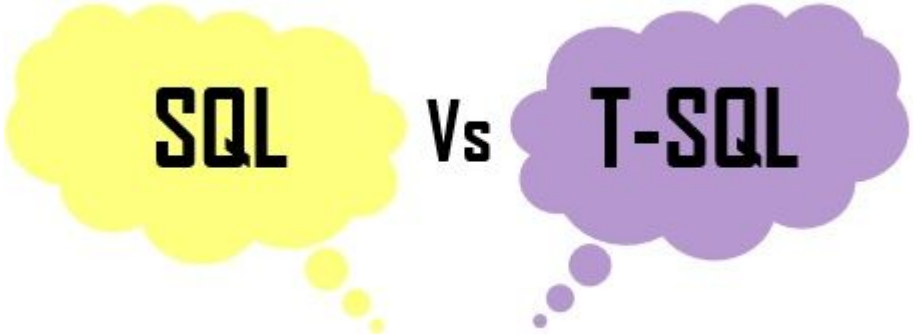
- T-SQL
  - Stands for Transact-SQL
  - It is an extension to SQL
  - It is a procedural language used on both Microsoft SQL Server and Sybase SQL Server systems.
- It is a full-featured programming language that dramatically extends the power of SQL.

Introduction to T-SQL Programming

- The language provides programmers with a broad range of features, including:
  - A rich set of data types, including specialized types for identifiers, timestamps, images, and long text fields
  - Local and global variables
  - Fully programmable server objects like views, triggers, stored procedures, and batch command files
  - Conditional processing
  - Exception and error handling
  - Full transaction control

Introduction to T-SQL Programming

- SQL and T-SQL are the query languages used to manipulate the database and are an important part of the DBMS.



- What are the differences between SQL and T-SQL?

Using Variables

- Variables are the object which acts as a placeholder to a memory location. Variable hold single data value.
- In MS SQL, there are 2 types of variables:
  - Local variable
    - A user declares the local variable.
    - **A local variable starts with @.**
    - Every local variable scope has the restriction to the **current batch** or **procedure** within any given session.
  - Global variable
    - The system maintains the global variable. A user cannot declare them.
    - **The global variable starts with @@**
    - It stores **session related information.**

Using Variables

- Before using any variables, they must be declared variable using DECLARE statement.
- Syntax

```
DECLARE @variable_name [AS] datatype
```

- Example

```
DECLARE @Product_name NVARCHAR(30)
DECLARE @Product_Id int = 7
```

- You can declare multiple variables in one statement, separated by commas

```
DECLARE @Product_name NVARCHAR(30), @Product_Id int
```

Using Variables

- By default, variables are initially set to Null
- You can assign a value to a variable
  - DECLARE statement
  - Using SET
    - Must declare a variable first.
    - Each variable requires a separate SET statement.
  - Using SELECT
    - Must declare a variable first.
    - Can assign a value to multiple variables separated by the comma

Using Variables

- Example to assign value to a variable
  - By DECLARE

```
DECLARE @Product_name NVARCHAR(30) = N'Bút chì', @Product_Id int = 7
PRINT @Product_name
PRINT @Product_Id
```

  - By SET/SELECT

```
DECLARE @Product_name NVARCHAR(30), @Product_Id int
SET @Product_name = N'Bút chì'
SET @Product_Id = 7
PRINT @Product_name + N' với id =' + CAST(@Product_Id AS
NVARCHAR(30))
```

SELECT @Product_name = N'Bút chì', @Product_Id = 7

The contents of Products table

product_id	product_name	brand_id	category_id	model_year	list_price
1	Trek 820 - 2016	9	6	2016	379.99
2	Ritchey Timberwolf Frameset - 2019	5	6	2019	750.00
3	Surly Wednesday Frameset - 2016	8	6	2016	1000.00
4	Trek Fuel EX 8 29 - 2017	9	6	2017	2899.99
5	Heller Shagamaw Frame - 2017	3	6	2017	1320.00
6	Surly Ice Cream Truck Frameset - 2016	8	6	2016	469.99
7	Trek Slash 8 27.5 - 2018	9	6	2018	4000.00
8	Trek Remedy 29 Carbon Frameset - 2016	9	6	2016	1800.00
9	Trek Conduit+ - 2016	9	5	2016	3000.00
10	Surly Straggler - 2020	8	4	2020	1000.00
11	Surly Straggler 650b - 2016	8	4	2016	1700.00
12	Electra Townie Original 21D - 2016	1	3	2016	550.00
13	Electra Cruiser 1 (24-Inch) - 2019	1	3	2019	270.00
14	Electra Girl's Hawaii 1 (16-inch) - 2019	1	3	2019	270.00

Using Variables

- Get value from a subquery by SET/SELECT
  - A subquery must return one value
  - When subquery returns zero row as a result, the variable is assigned NULL value

```
DECLARE @Product_name NVARCHAR(30)
SET @Product_name = (SELECT product_name FROM products WHERE
                     product_id = 1)
PRINT @Product_name
```

You can replace SET by SELECT

Using Variables

- Using variables in a query

```
DECLARE @Product_id INT
SET @Product_id = 1
SELECT product_name, model_year, list_price
FROM products
WHERE product_id = @Product_id
ORDER BY product_name
```

- Result

product_name	model_year	list_price
Trek 820 - 2016	2016	379.99

Using Variables

- Using variables in a query
  - Storing query result in a variable

```
DECLARE @product_count int
SET @product_count = (
    SELECT
        COUNT(*)
    FROM
        products
)
SELECT @product_count AS 'Number of Products'
```

- Result

Number of Products
14

Using Variables

- Using variables in a query
  - Selecting a record into variables

```
DECLARE @product_name VARCHAR(MAX), @list_price DECIMAL(10,2)
SELECT @product_name = product_name,@list_price = list_price
FROM products
WHERE product_id = 1
SELECT @product_name AS product_name,
       @list_price AS list_price
```

  - Result

product_name	list_price
Trek 820 - 2016	379.99

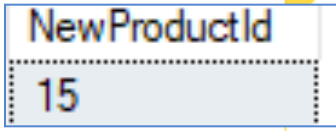
Using Global variables

- Global variables are pre-defined system functions. Their names begin with an @@ prefix
- Some common global variables
  - @@IDENTITY
  - @@ERROR
  - @@ROWCOUNT
  - @@TOTAL_ERRORS
  - @@SERVERNAME

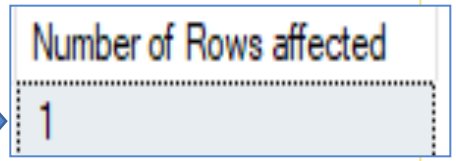
Using Global variables example

- Example
  - @@IDENTITY is used to get the last value inserted into an IDENTITY column by an insert statement.
  - Example: get the last inserted product_id which is the identity column in previous product table

```
INSERT INTO products
VALUES('Electra - 2020', 1, 3, 2020, 2000)
GO
SELECT @@IDENTITY AS NewProductId
```




```
SELECT @@rowcount as 'Number of Rows affected'
```



Control-Flow statements

- IF/ IF…ELSE statement
- CASE statement
- WHILE statement

IF/IF…ELSE statement

- IF statement

```
IF boolean_expression
      BEGIN statement_block
      END
```

- IF…ELSE statement

```
IF Boolean_expression
      BEGIN Statement block
      END
ELSE
      BEGIN Statement block
      END
```

IF/IF…ELSE statement

- Example

```
DECLARE @product_count int
SET @product_count = (SELECT COUNT(*) FROM products
                WHERE list_price >1000)
IF @product_count > 0
   BEGIN
        PRINT 'The products have price are greater than 100'
        SELECT product_id, product_name, list_price
        FROM products
        WHERE list_price >1000
   END
ELSE
   BEGIN
        PRINT 'There is no product that has price is less than
               or equal to 1000'
   END
```

CASE statement

- Syntax

```
CASE
     WHEN condition1 THEN result1
     WHEN condition2 THEN result2
     WHEN conditionN THEN resultN
     ELSE result
END
```

- The CASE statement always goes in the SELECT clause
- CASE must include the following components: WHEN, THEN, and END. ELSE is an optional component.

CASE statement

- Example

```
SELECT product_id, list_price,
CASE
    WHEN list_price > 1000 THEN 'The price is greater than 1000'
    WHEN list_price = 1000 THEN 'The price is 1000'
    ELSE 'The price is under 1000'
END AS PriceText
FROM products
WHERE model_year = 2016
```

product_id	list_price	PriceText
1	379.99	The price is under 1000
3	1000.00	The price is 1000
6	469.99	The price is under 1000
8	1800.00	The price is greater than 1000
9	3000.00	The price is greater than 1000
11	1700.00	The price is greater than 1000
12	550.00	The price is under 1000

WHILE statement

- Syntax

```
WHILE Boolean_expression
BEGIN
      statement_block
END
```

- In WHILE, you can use
  - BREAK statement to exit from the WHILE LOOP
  - CONTINUE to restart the WHILE LOOP from the beginning

WHILE statement

- Example

```
DECLARE @Counter INT , @MaxId INT,
        @ProductName NVARCHAR(100)
SELECT @Counter = min(product_id) , @MaxId = max(product_id)
FROM products
WHERE model_year= 2016
WHILE(@Counter IS NOT NULL
      AND @Counter <= @MaxId)
BEGIN
   SELECT @ProductName = product_name
   FROM products WHERE product_id = @Counter

   PRINT CONVERT(VARCHAR(MAX),@Counter) + '. product name is '
+ @ProductName
   SET @Counter  = @Counter  + 1
END
```

WHILE statement

- Result

```
1. product name is Trek 820 - 2016
2. product name is Ritchey Timberwolf Frameset - 2019
3. product name is Surly Wednesday Frameset - 2016
4. product name is Trek Fuel EX 8 29 - 2017
5. product name is Heller Shagamaw Frame - 2017
6. product name is Surly Ice Cream Truck Frameset - 2016
7. product name is Trek Slash 8 27.5 - 2018
8. product name is Trek Remedy 29 Carbon Frameset - 2016
9. product name is Trek Conduit+ - 2016
10. product name is Surly Straggler - 2020
11. product name is Surly Straggler 650b - 2016
12. product name is Electra Townie Original 21D - 2016
```

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Thank You !