Chapter 3: T-SQL in Ms. SQL Server

Content

CADT IDT Institute of Digital Technology

1. Declare Variable
2. Operator
3. Control Structure
4. Try Catch
5. Sequence
6. Offset and Fetch
7. Row-Number( )

1. Declare Variable

- What is Variable ?

A Transact-SQL local variable is an object that can hold a single data value of a specific type.

- Syntax: *Declare @Var_Name [as] DataType=Value;*
- Example

```
DECLARE @IVARIABLE INT, @VVARIABLE VARCHAR( 100 ),
@DDATETIME DATETIME
```

A. Set Value to Variable

```
SET @IVARIABLE = 1
SET @VVARIABLE = 'MYVAR'
SET @DDATETIME = GETDATE( )
```

1. Declare Variable

CADT IDT Institute of Digital Technology

B. Display Data

```
SELECT @IVARIABLE IVAR, @VVARIABLE VVAR,
@DDATETIME DDT
GO
```

OR

```
DECLARE @IVARIABLE INT = 1, @VVARIABLE
VARCHAR( 100 ) = 'MYVAR', @DDATETIME DATETIME =
GETDATE( )
SELECT @IVARIABLE IVAR, @VVARIABLE VVAR,
@DDATETIME DDT
GO
```

2. Operator

- Arithmetic Operator

  + Addition

  - Subtraction

  * Multiplication

  / Division

  % Modulus

- Assignment Operator (=)

```
DECLARE @MyCounter INT;
SET @MyCounter = 1;
```

2. Operator

CADT IDT Institute of Digital Technology

- Comparison Operator

=	Equal to
>	Grater than
<	Lower than
>=	Greater than or equal
<=	Lower than or equal
<>	Not Equal
!=	Not Equal (not ISO standard)
!<	Not Lower than(not ISO Standard)
!>	Not Greater than(not ISO Standard)

2. Operator

CADT IDT Institute of Digital Technology

- Compound Operator

+=	add some amount to original value
-=	subtract some amount to original value
*=	multiple some amount to original value
/=	divide some amount to original value
%=	modulus some amount to original value

2. Operator

CADT IDT Institute of Digital Technology

- Logical Operators

AND	TRUE if both Boolean expressions are TRUE.
ANY	TRUE if any one of a set of comparisons are TRUE.
BETWEEN .. and ..	TRUE if the operand is within a range.
IN	TRUE if the operand is equal to one of a list of expressions.
LIKE	TRUE if the operand matches a pattern with Wild Cart Operator
NOT	Reverses the value of any other Boolean operator.
OR	TRUE if either Boolean expression is TRUE.

3. Control Structure

CADT IDT Institute of Digital Technology

1. CASE

- CASE provides a structured method of evaluating a list of options and then returning a single value.
- You can use the CASE statement alone or within a SELECT statement.

Example:

```
SELECT au_fname, au_lname,
   CASE state
      WHEN 'OR' THEN 'Oregon'
      END AS StateName
FROM authors
```

3. Control Structure

CADT IDT Institute of Digital Technology

2. WHILE-BREAK-CONTINUE

- WHILE is a powerful T-SQL control-flow statement. The WHILE statement causes repeated execution of a statement or block of statements while a given condition is true.
- You can specify the optional BREAK and CONTINUE keywords to exit from the while loop or cause the loop to continue.

Example:

```
WHILE @@FETCH_STATUS = 0
BEGIN
   FETCH NEXT FROM Employee_Cursor
END
```

3. Control Structure

CADT IDT Institute of Digital Technology

3. RETURN

RETURN lets you exit from a T-SQL batch or stored procedure. You can specify an optional integer variable with RETURN to pass a status value to the calling procedure, which can evaluate the return code and perform different actions depending on the results of the T-SQL batch or stored procedure.

Example :

- RETURN @return_code

3. Control Structure

CADT IDT Institute of Digital Technology

4. WAITFOR

WAITFOR lets you delay the execution of a T-SQL batch either for a given amount of time (when you specify the DELAY keyword) or until a specified system time (when you specify the TIME keyword).

Example :

- WAITFOR TIME '23:00'
- WAIT FOR DELAY "00:01:00"

3. Control Structure

5. BEGIN-END

BEGIN-END lets you group T-SQL statements and execute multiple statements as a result of an IF test.

Example:

```
BEGIN
   SET @ErrorNumber = @@ERROR
   PRINT 'Error encountered'
END
```

3. Control Structure

6. IF-ELSE

The IF statement lets you test a variable's contents and conditionally execute the T-SQL statements that follow, depending on the test's results. When the IF test evaluates to false, the optional ELSE portion of the statement lets an alternative T-SQL statement execute.

Example :

```
IF (@@Error  0)
   ROLLBACK
ELSE
   COMMIT
END
```

3. Control Structure

CADT IDT Institute of Digital Technology

7. GOTO

GOTO is a basic T-SQL control-flow statement. It causes the execution of a T-SQL batch to branch to the label specified in the line with the GOTO statement.

Example:

```
GOTO error_condition;
```

4. Try Catch

The TRY CATCH construct allows you to gracefully handle exceptions in SQL Server. To use the TRY CATCH construct, you first place a group of Transact-SQL statements that could cause an exception in a BEGIN TRY...END TRY block as follows:

```
BEGIN TRY
-- statements that may cause exceptions
END TRY
```

Then you use a BEGIN CATCH...END CATCH block immediately after the TRY block:

```
BEGIN CATCH
   -- statements that handle exception
END CATCH
```

4. Try Catch

CADT IDT Institute of Digital Technology

The CATCH block functions

Inside the CATCH block, you can use the following functions to get the detailed information on the error that occurred:

- ERROR_LINE( ) returns the line number on which the exception occurred.
- ERROR_MESSAGE( ) returns the complete text of the generated error message.
- ERROR_PROCEDURE( ) returns the name of the stored procedure or trigger where the error occurred.
- ERROR_NUMBER( ) returns the number of the error that occurred.
- ERROR_SEVERITY( ) returns the severity level of the error that occurred.
- ERROR_STATE( ) returns the state number of the error that occurred.

4. Try Catch

SQL Server TRY CATCH examples

```
CREATE PROC usp_divide(@a decimal,@b decimal,@c decimal output)
AS
BEGIN
   BEGIN TRY
      SET @c = @a / @b;
   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
END;
```

4. Try Catch

Attempt to divide 10 by zero by calling the usp_divide stored procedure:

```
DECLARE @r2 decimal;
EXEC usp_divide 10, 0, @r2 output;
PRINT @r2;
```

The following picture shows the output:

ErrorNumber	ErrorSeverity	ErrorState	ErrorProcedure	ErrorLine	ErrorMessage
8134	16	1	usp_divide	8	Divide by zero error encountered.

5. Sequence

CADT IDT Institute of Digital Technology

- A sequence object in MS-SQL Server is designated to define and get only integer values, such as int, bigint, smallint, tinyint.
- However, if we want to generate sequence value(s) that are alpha-numeric, then we can define a Stored Procedure that can combine to generate an alpha-numeric combination of sequence values. This blog gives a complete idea of how this can be implemented.

5. Sequence

CADT IDT Institute of Digital Technology

Syntax:

```
CREATE SEQUENCE sequence_name
START With                  -- Start Value of sequence
INCREMENT by                -- Increment of sequences
MINVALUE                    -- Minimum value of Sequence
MAXVALUE                    -- Maximum value of Sequence
CYCLE                       -- Cycle loop of sequence
```

5. Sequence

Example: Create Sequence CountBy1 that start from 1 and increment by 1.

```
CREATE SEQUENCE COUNTBY1
START WITH 1
INCREMENT BY 1 ;
```

You can show information of sequence by using this query.

```
SELECT * FROM sys.sequences WHERE name = 'CountBy1';
```

	name	object_id	principal_id	schema_id	parent_object_id	type	type_desc	create_date	modify_date
1	CountBy1	2107154552	NULL	1	0	SO	SEQUENCE_OBJECT	2014-03-14 21:46:48.040	2014-03-14 21:46:48.040

5. Sequence

- These are default value of sequence

start_value	-9223372036854775808
increment	1
mimimum_value	-9223372036854775808
maximum_value	9223372036854775807
is_cycling	0
is_cached	1
current_value	-9223372036854775808

5. Sequence

Using sequence in SQL Statement.

```
ALTER SEQUENCE COUNTBY1 RESTART WITH 1

DECLARE @MYVAR1 BIGINT = NEXT VALUE FOR COUNTBY1

DECLARE @MYVAR2 BIGINT ;

DECLARE @MYVAR3 BIGINT ;

SET @MYVAR2 = NEXT VALUE FOR COUNTBY1 ;

SELECT @MYVAR3 = NEXT VALUE FOR COUNTBY1 ;

SELECT @MYVAR1 AS MYVAR1, @MYVAR2 AS MYVAR2,
@MYVAR3 AS MYVAR3 ;
```

5. Sequence

- Call Sequence Used in Table and Generate Auto Number with Letters in SQL Server.
- Create Table

```
CREATE TABLE MyTable
(IDColumn nvarchar(25) PRIMARY KEY,name varchar(25)) ;
```

- Create Sequence CounterSeq

```
CREATE SEQUENCE CounterSeq
    AS int
    START WITH 1
    INCREMENT BY 1 ;
```

5. Sequence

- Modify the table created above by adding Auto Number to the Fields IDColumn.

```
ALTER TABLE MyTable
    ADD
        DEFAULT N'AdvWorks_' +
        CAST( NEXT VALUE FOR CounterSeq AS NVARCHAR( 20 ) )
        FOR IDColumn;
```

- Insert data into Table MyTable

```
INSERT MyTable ( name ) VALUES ( 'Larry' ) ;
```

6. Offset and Fetch

- To extract data from the Database and display on the GridView to the Current Page.
- OFFSET contains minutes to determine the starting point of the Record we want to display.
- FECTH has minutes to determine the number of Records we need to present.

Syntax:

```
SELECT * FROM TableName ORDER BY ID
OFFSET Start ROWS
FETCH NEXT Count ROWS ONLY;
```

Example:

```
SELECT * FROM tblBodyCompares ORDER BY BodyID
OFFSET 0 ROWS   FETCH NEXT 15 ROWS ONLY;
```

6. Offset and Fetch

- Below we create two variables for assigning values to OFFSET and Fetch Next.

```
DECLARE @OffsetRows tinyint = 0, @FetchRows tinyint = 20;
SELECT TransactionID, ProductID, TransactionDate, Quantity,
ActualCost
FROM TransactionHistory
ORDER BY TransactionDate DESC
OFFSET @OffsetRows ROWS
FETCH NEXT @FetchRows ROWS ONLY;
```

7. Row-Number()

- The ROW_NUMBER( ) is a window function that assigns a sequential integer to each row within the partition of a result set. The row number starts with 1 for the first row in each partition.
- The following shows the syntax of the ROW_NUMBER( ) function:

```
ROW_NUMBER( ) OVER (
    [PARTITION BY partition_expression, ... ]
    ORDER BY sort_expression [ASC | DESC], ...
)
```

7. Row-Number()

Using SQL Server ROW_NUMBER( ) function over a result set example

- The following statement uses the ROW_NUMBER( ) to assign each customer row a sequential number:

```
SELECT ROW_NUMBER( ) OVER (ORDER BY first_name)
row_num, first_name, last_name, city
FROM sales.customers;
```

row_num	first_name	last_name	city
1	Aaron	Knapp	Yonkers
2	Abbey	Pugh	Forest Hills
3	Abby	Gamble	Amityville
4	Abram	Copeland	Harlingen
5	Adam	Henderson	Los Banos
6	Adam	Thornton	Central Islip

7. Row-Number()

Using SQL Server ROW_NUMBER( ) over partitions example

- The following example uses the ROW_NUMBER( ) function to assign a sequential integer to each customer. It resets the number when the city changes:

```
SELECT first_name, last_name, city,
   ROW_NUMBER( ) OVER (PARTITION BY city
      ORDER BY first_name) row_num
FROM sales.customers
```

first_name	last_name	city	row_num
Douglass	Blankenship	Albany	1
Mi	Gray	Albany	2
Priscilla	Wilkins	Albany	3
Andria	Rivers	Amarillo	1
Delaine	Estes	Amarillo	2
Jonell	Rivas	Amarillo	3
Luis	Tyler	Amarillo	4

Question for review

CADT IDT Institute of Digital Technology

1. What is Variable in T-SQL ?
2. How to declare variable in T-SQL ?
3. What is Control Structure ?
4. What is Try Catch ? Sequence ? Offset and Fetch ?
5. What is Row-Number( ) ?

Homework

Please write code T-SQL for:

1. Convert from one digit: 0 - 9 to khmer number: ០ - ៩
2. Convert from one digit: 0 - 9 to khmer word: សូន្យ - ប្រាំបួន
3. Convert from tow digits: 10 - 90 to khmer word: ដប់ - កៅសិប
4. Convert from three plus digits: 100-1000000 to khmer word: រយ - លាន.
5. Convert from number to khmer number for example: 2022 to khmer number: ២០២២.