Pogramming in Sql Server

12S2102- Basis Data

Parmonangan R. Togatorop
Fakultas Informatika dan Teknik Elektro
Institut Tenologi Del

10/21/2020

Overview

- Batches
- Control – of – flow statement

Batches

- A group of one or more Transact-SQL statements sent at the same time from an application to SQL Server for execution
- SQL Server compiles the statements of a batch into a single executable unit, called an execution plan
- Example:
  Select * from Employees
  Select * from Products Go
- SQL Server Processes a batch interactively or from a file
- Compiles the statement of a batch into single executable unit (execution plan) and executed one at a time
- If there is syntax error, nothing in the batch will execute
- Certain statement can be combined to create a batch and others cannot combined.

Batches

- A **compile error,** such as a syntax error, prevents the compilation of the execution plan. Therefore, no statements in the batch are executed.
- A **run-time error**, such as an arithmetic overflow or a constraint violation, has one of the following effects:
  - Most run-time errors stop the current statement and the statements that follow it in the batch.
  - Some run-time errors, such as constraint violations, stop only the current statement. All the remaining statements in the batch are executed

Batches (example)

```
CREATE TABLE dbo.t3(a int) ;
INSERT INTO  dbo.t3 VALUES (1) ;
INSERT INTO dbo.t3 VALUES (1,1) ;
INSERT INTO  dbo.t3 VALUES (3) ;
SELECT * FROM dbo.t3;
```

Batches (Example)

- First, the batch is compiled. The CREATE TABLE statement is compiled, but because the table dbo.t3 does not yet exist, the INSERT statements are not compiled.
- Second, the batch starts to execute. The table is created.The first INSERT is compiled and then immediately executed. The table now has one row. Then, the second INSERT statement is compiled. The compilation fails, and the batch is terminated. The SELECT statement returns one row.

Rules for Using Batches

- CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE SCHEMA, CREATE TRIGGER, and CREATE VIEW statements cannot be combined with other statements in a batch. The CREATE statement must start the batch. All other statements that follow in that batch will be interpreted as part of the definition of the first CREATE statement.
- A table cannot be changed and then the new columns referenced in the same batch.
- If an EXECUTE statement is the first statement in a batch, the EXECUTE keyword is not required. The EXECUTE keyword is required if the EXECUTE statement is not the first statement in the batch.

Variables

- Use a variable to store a temporary value.
- You can declare a variable using the DECLARE statement
- Syntax:

DECLARE @variable_name data_type(@ symbol is used by the query processor to identify variables)

Ex:

```
(1)DECLARE @Charge int
(2)DECLARE @myvar char(20)
SET @myvar = 'This is a test'
SELECT @myvar
GO
```

In transact SQL: local variable and global variable

Local variable : user define them and must begin with : @ sign

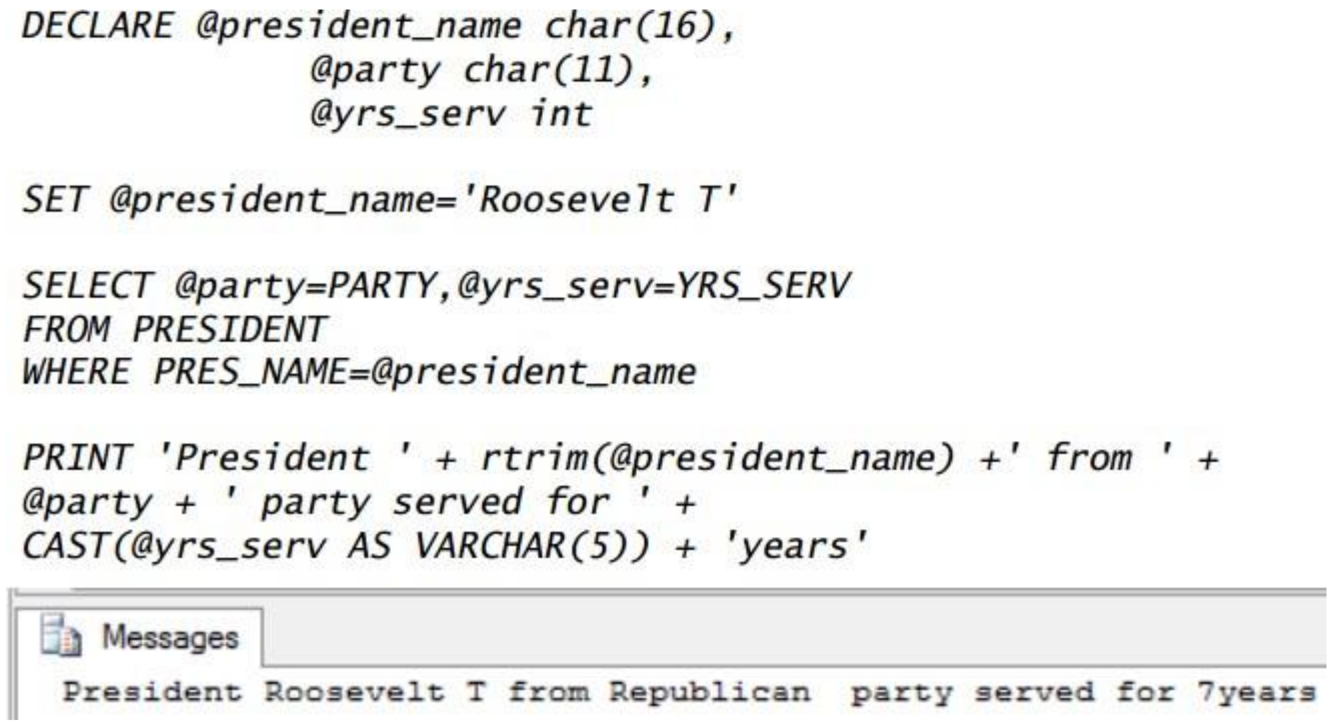
Global variable : declare by the server and have two @ sign (@@)preceding their names.

Standard Identifiers

- Standard identifiers can contain from one to 128 characters, including letters, symbols (_, @, or #), numbers. No embedded spaces are allowed in standard identifiers.
  Rules for using identifiers include:
  - The first character must be an alphabetic character of a-z or A-Z.
  - After the first character, identifiers can include letters, numerals, or the @, $, #, or _.
  - Identifier names starting with a symbol have special uses:
    - An identifier beginning with the @ symbol denotes a local variable or parameter.
    - An identifier beginning with a pound sign (#) denotes a temporary table or procedure.
    - An identifier beginning with a double pound sign(##) denotes a global temporary object.

Local Variables

- User-defined with **DECLARE** Statement
- Assigned Values with **SET**or **Select** Statement



```
DECLARE @president_name char(16),
              @party char(11),
              @yrs_serv int

SET @president_name='Roosevelt T'

SELECT @party=PARTY,@yrs_serv=YRS_SERV
FROM PRESIDENT
WHERE PRES_NAME=@president_name

PRINT 'President ' + rtrim(@president_name) +' from ' +
@party + ' party served for ' +
CAST(@yrs_serv AS VARCHAR(5)) + 'years'
```

Messages

```
President Roosevelt T from Republican  party served for 7years
```

Global variable

- @@version
- @@servername
- @@spid
- @@procid
- @@error
- @@rowcount
- @@connections, etc…

CAST and CONVERT

- Converts an expression of one data type to another
  - CAST (expression AS data_type [(length)])

```
SELECT 9.5AS Original, CAST(9.5AS int)AS in_integer,
CAST(9.5AS decimal(6,4))AS in_decimal;
```

  - CONVERT (data_type [(length)] ,expression [,style])

```
SELECT 9.5AS Original, CONVERT(int, 9.5)AS in_integer,
CONVERT(decimal(6,4), 9.5)AS in_decimal;
```

Printing Message

- Using PRINT statement to display a userdefined message or content variable on the screen
  - Ex: DECLARE @MyName char(50)
    SELECT @MyName = ‘Coomar Chris’
    PRINT @MyName

Comment entry

- Use comment entries in the batches to write description code.
  - Example:
  - /*……………comment………………..*/
  - --comment

Comment

- In Line Comments

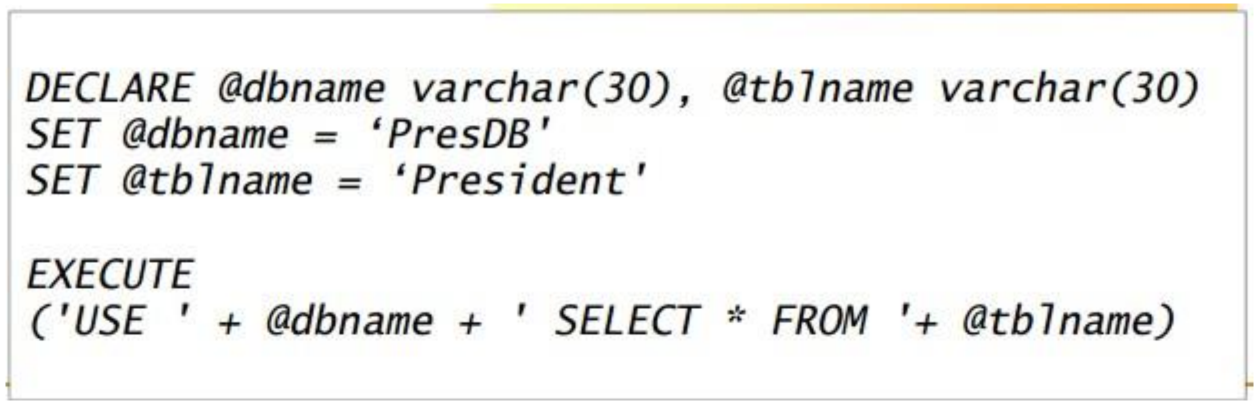
```
SELECT ProductName,
(UnitsInStock + UnitsOnOrder) AS Max-- Calculates inventory
,SupplierID
FROM Products
```

- Block Comments

```
/*
** This code retrieves all rows of the products table
** and displays the unit price, the unit price increased
** by 10 percent, and thename of the product.
*/
SELECT UnitPrice, (UnitPrice * 1.1), ProductName
FROM Products
```

Dynamically Constructing Statements

- Use EXECUTE with String Literals and Variables
- Use When You Must Assign Value of Variable at Execution Time
- Any Variables and Temporary Tables Last Only During Execution

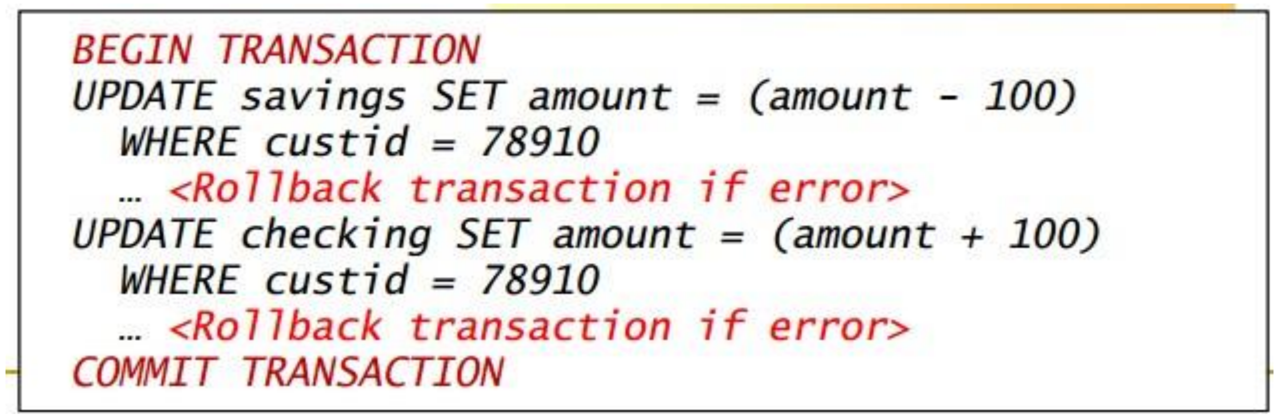


```
DECLARE @dbname varchar(30), @tblname varchar(30)
SET @dbname = ‘PresDB'
SET @tblname = ‘President'

EXECUTE
('USE ' + @dbname + ' SELECT * FROM '+ @tblname)
```

Using Transactions

- Processed Like a Batch
- Data Integrity Is Guaranteed
- Changes to the Database Are Either Applied Together or Rolled Back



```
BEGIN TRANSACTION
UPDATE savings SET amount = (amount - 100)
  WHERE custid = 78910
  … <Rollback transaction if error>
UPDATE checking SET amount = (amount + 100)
  WHERE custid = 78910
  … <Rollback transaction if error>
COMMIT TRANSACTION
```

Control-Of-Flow Language

- Control the flow of execution of SQL statement
- Typically used when statements have to be conditionally or repeatedly executed
- They are : IF…ELSE statement, BEGIN…END block, CASE statement, WHILE statement

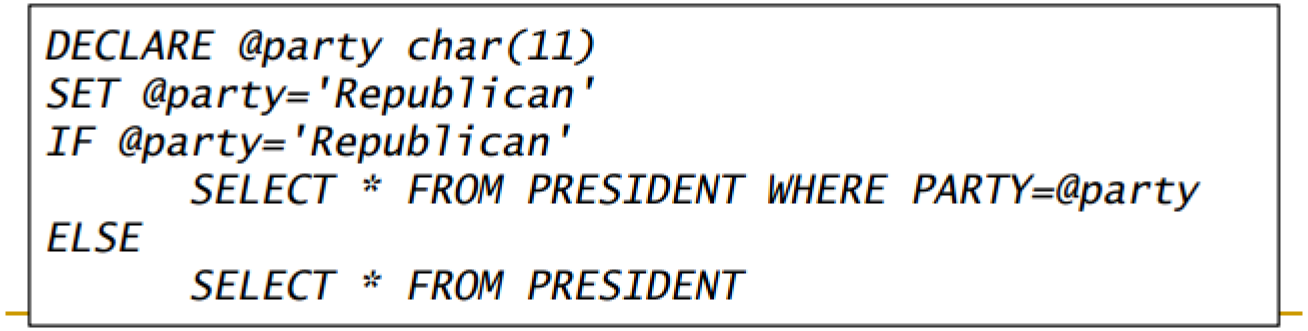
IF…ELSE statement

- Used to execute SQL statement conditionally
- Syntax: IF boolean expression

  {sql statement/statement block}

  [ELSE boolean expression

  {sql statement/statement block}]

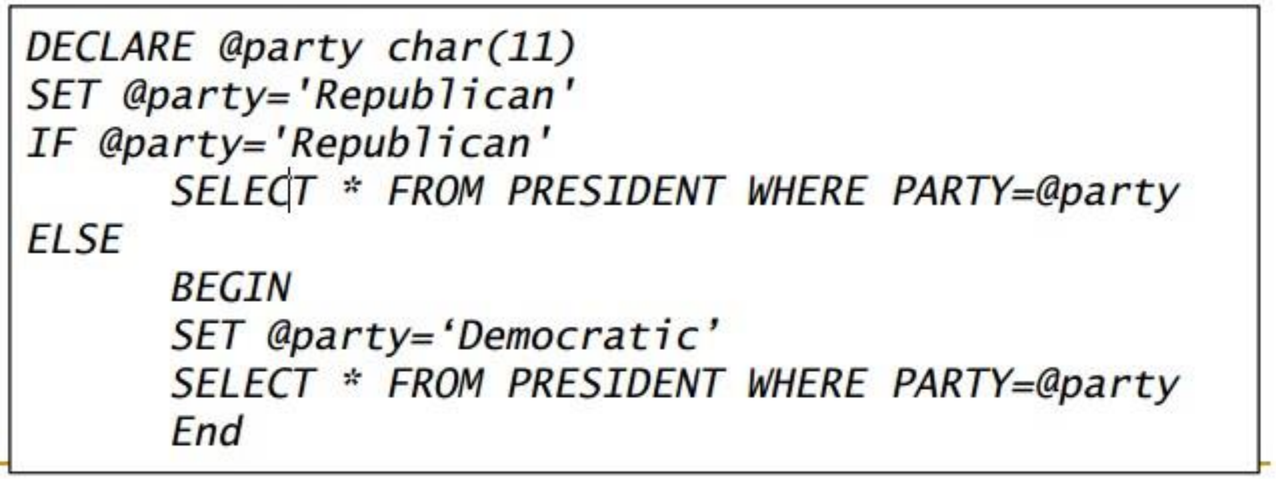


```
DECLARE @party char(11)
SET @party='Republican'
IF @party='Republican'
      SELECT * FROM PRESIDENT WHERE PARTY=@party
ELSE
      SELECT * FROM PRESIDENT
```

BEGIN…END block

- If there are multiple T-SQL statement, then these must be enclosed within BEGIN and END key words.
- Syntax :

```
BEGIN
{SQL_statement/statement_block}
End
```



```
DECLARE @party char(11)
SET @party='Republican'
IF @party='Republican'
      SELECT * FROM PRESIDENT WHERE PARTY=@party
ELSE
      BEGIN
      SET @party='Democratic'
      SELECT * FROM PRESIDENT WHERE PARTY=@party
      End
```

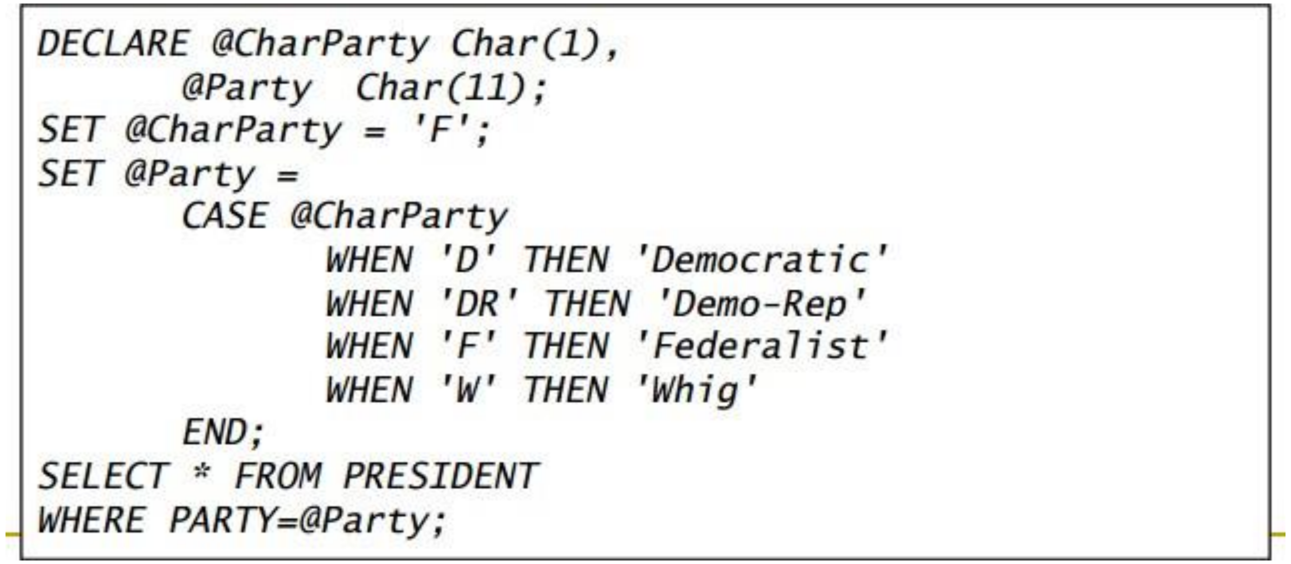
CASE construct

- For situations where several conditions need to be evaluated
- Evaluates a list of condition and returns one of various possible result
- Syntax :

```
CASE
 WHEN boolean_expression THEN expression 1
  [WHEN boolean_expression THEN  expression][....]
  [ELSE expression]
END
```

CASE

For some example, read sql book online or
CASE (Transact-SQL).pdf



```
DECLARE @CharParty Char(1),
        @Party  Char(11);
SET @CharParty = 'F';
SET @Party =
        CASE @CharParty
                WHEN 'D' THEN 'Democratic'
                WHEN 'DR' THEN 'Demo-Rep'
                WHEN 'F' THEN 'Federalist'
                WHEN 'W' THEN 'Whig'
        END;
SELECT * FROM PRESIDENT
WHERE PARTY=@Party;
```

WHILE

- Allow a set of T-SQL Statements to execute repeteadly as long as the given condition holds true.
- Syntax:

```
WHILE boolean_expression
{sql_statement/statement_block}
[BREAK]
{sql_statement/statement_block}
[CONTINUE]
```

WHILE

```
DECLARE @intFlag INT
SET @intFlag = 1
WHILE (@intFlag <=5)
BEGIN
    PRINT @intFlag
    SET @intFlag = @intFlag + 1
END
```

```
WHILE (SELECT AVG(ListPrice) FROM Production.Product) < $300
BEGIN
   UPDATE Production.Product
      SET ListPrice = ListPrice * 2
   SELECT MAX(ListPrice) FROM Production.Product
   IF (SELECT MAX(ListPrice) FROM Production.Product) > $500
      BREAK
   ELSE
      CONTINUE
END
```

Thank you