جامعة طرابلس
كلية تقنية المعلومات

Advanced Databases قواعد البيانات المتقدمة

ITSE312

د. عبدالسلام منصور الشريف

a.abdoessalam@uot.edu.ly

المحاضرة السابعة – اللغة الإجرائية

Procedural Language

Contents

- Programming Database
  - Variables
  - Data Conversion
  - Procedural Language

Procedural Language - Variables

- Variables are declared in the body of a batch or procedure with the DECLARE statement and are assigned values by using either a SET or SELECT (Not ANSI) statement.
- After declaration, all variables are initialized as NULL, unless a value is provided as part of the declaration.

```
DECLARE  @local_variable [AS] data_type [ = value ]
```

Procedural Language - Variables

```
declare @hello as nvarchar(200)=N'Hello World'
select 'Your message is =', @hello;
```

```
declare @age as int;
set @age = 45;
select 'Your age is =',@age
```

```
declare @mark as decimal(5,2);
select @mark = 50.53822345;
select 'Your mark is =', @mark;
```

```
declare @now as datetime2 = getdate();
print  @now;
```

Procedural Language – Conversion

- Data types can be converted in the following scenarios:
  - When data from one object is moved to, compared with, or combined with data from another object, the data may have to be converted from the data type of one object to the data type of the other.
  - When data from a Transact-SQL result column, return code, or output parameter is moved into a program variable, the data must be converted from the SQL Server system data type to the data type of the variable.

Implicit vs Explicit Conversion

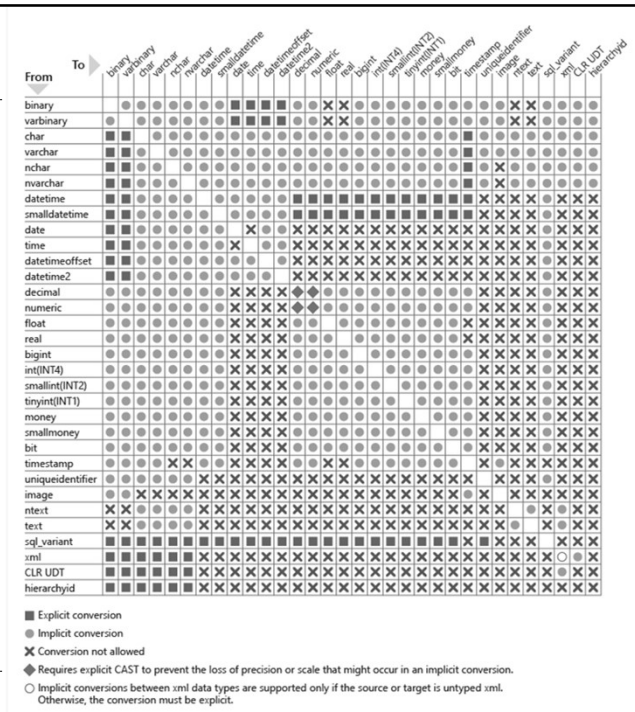
- Data types can be converted either implicitly or explicitly.
  - Implicit conversions are not visible to the user. SQL Server automatically converts the data from one data type to another. For example, when a **smallint** is compared to an int, the **smallint** is implicitly converted to **int** before the comparison proceeds.
  - GETDATE() implicitly converts to date style 0. SYSDATETIME() implicitly converts to date style 21.
  - Explicit conversions use the CAST function.
  - The CAST function convert a value (a local variable, a column, or another expression) from one data type to another.

Implicit Conversion

```
declare @string VARCHAR(10);
set @string = 1; -- implicitly converted to string
select @string + ' is a string.'
```

```
declare @notastring INT;
set @notastring = '1'; -- implicitly converted to int
select @notastring + '1'; -- implicitly converted to int
```

```
declare @intstring INT;
set @intstring = '1'; -- implicitly converted to int
select @intstring + ' is not a string.'; -- can't convert
```



Procedural Language – Cast

- This function converts an expression of one data type to another.

```
CAST ( expression AS data_type [ ( length ) ] )
```

expression: any valid expression.

data_type: the target data type.

length: an optional integer that specifies the length of the target data type, for data types that allow a user specified length. The default value is 30.

Procedural Language – Cast

- Explicit conversion, cast will truncate or round numbers according to the target data type.

```
SELECT  CAST(10.6496 AS INT) as trunc1,
        CAST(-10.6496 AS INT) as trunc2,
        CAST(10.6496 AS NUMERIC) as round1,
        CAST(-10.6496 AS NUMERIC) as round2;
```

```
trunc1  trunc2  round1  round2
10      -10     11      -11
```

Procedural Language – IF…ELSE

- Imposes conditions on the execution of a Transact-SQL statement. The Transact-SQL statement that follows an IF keyword and its condition is executed if the condition is satisfied: the Boolean expression returns TRUE.
- The optional ELSE keyword introduces another Transact-SQL statement that is executed when the IF condition is not satisfied: the Boolean expression returns FALSE.

```
IF Boolean_expression
     { sql_statement | statement_block }
[ ELSE
     { sql_statement | statement_block } ]
```

```
IF DATENAME(weekday, GETDATE()) IN (N'Saturday', N'Sunday')
       SELECT 'Weekend';
ELSE
       SELECT 'Weekday';
```

Procedural Language – IF…ELSE

- The condition could be a SQL statement.

```
IF
  (SELECT COUNT(*) FROM [dbo].[Students] WHERE Enrolled = 1) > 5
       PRINT 'There are more than 5 Students enrolled.'
ELSE
       PRINT 'There are 5 or less students enrolled.' ;
```

Procedural Language – CASE

- Evaluates a list of conditions and returns one of multiple possible result expressions.
- The searched CASE expression evaluates a set of Boolean expressions to determine the result.
- It supports an optional ELSE argument.
- CASE can be used in any statement or clause that allows a valid expression. For example, you can use CASE in statements such as SELECT, UPDATE, DELETE and SET, and in clauses such as select_list, IN, WHERE, ORDER BY, and HAVING.

```
CASE
     WHEN Boolean_expression THEN result_expression
     [ ...n ]
     [ ELSE else_result_expression ]
END
```

Procedural Language – CASE

- List students' Enrollment status.

```
SELECT Id,Name,
  CASE
    WHEN Enrolled = 0 THEN N'Not Enrolled'
    WHEN Enrolled = 1 THEN N'Enrolled'
    ELSE N'Unknown'
  END
FROM [dbo].[Students]
```

Procedural Language – BEGIN … END

- Encloses a series of Transact-SQL statements so that a group of Transact-SQL statements can be executed. BEGIN and END are control-of-flow language keywords.

```
BEGIN
    { sql_statement | statement_block }
END
```

```
IF DATENAME(weekday, GETDATE()) IN (N'Saturday', N'Sunday’)
BEGIN
       SELECT 'Weekend’;
END
```