Session 6

Module 6

Working with SQL Server Data Types

Module Overview

- Introducing SQL Server Data Types
- Working with Character Data
- Working with Date and Time Data

Lesson 1: Introducing SQL Server Data Types

- SQL Server Data Types
- Numeric Data Types
- Binary String Data Types
- Other Data Types
- Data Type Precedence
- When are Data Types Converted?
- Demonstration: SQL Server Data Types

SQL Server Data Types

- SQL Server associates columns, expressions, variables and parameters with data types
- Data types determine the kind of data that can be held in a column or variable
  - Integers, characters, dates, decimals, binary strings, and so on
- SQL Server supplies built-in data types
- Developers can also define custom data types

SQL Server Data Type Categories	
Exact numeric	Unicode character strings
Approximate numeric	Binary strings
Date and time	Other
Character strings	

Numeric Data Types

- Exact Numeric Data Types

Data Type	Range	Storage (bytes)
tinyint	0 to 255	1
smallint	-32,768 to 32,768	2
int	2^{31} (-2,147,483,648) to 2^{31}-1 (2,147,483,647)	4
bigint	-2^{63} - 2^{63}-1 (+/- 9 quintillion)	8
bit	1, 0 or NULL	1
decimal/numeric	-10^{38} +1 through 10^{38} – 1 when maximum precision is used	5-17
money	-922,337,203,685,477.5808 to 922,337,203,685,477.5807	8
smallmoney	-214,748.3648 to 214,748.3647	4

Binary String Data Types

- Binary string data types

Data Type	Range	Storage (bytes)
binary(n)	1 to 8000 bytes	n bytes
varbinary(n)	1 to 8000 bytes	n bytes + 2
varbinary(max)	1 to 2.1 billion (approx.) bytes	n bytes + 2

- The image data type is also a binary string type but is marked for removal in a future version of SQL Server; varbinary(max) should be used instead

Other Data Types

Data Type	Range	Storage (bytes)	Remarks
xml	0-2 GB	0-2 GB	Stores XML in native hierarchical structure
uniqueidentifier	Auto-generated	16	Globally unique identifier (GUID)
hierarchyid	n/a	Depends on content	Represents position in a hierarchy
rowversion	Auto-generated	8	Previously called timestamp
geometry	0-2 GB	0-2 GB	Shape definitions in Euclidian geometry
geography	0-2 GB	0-2 GB	Shape definitions in round-earth geometry
sql_variant	0-8000 bytes	Depends on content	Can store data of various other data types in the same column
cursor	n/a	n/a	Not a storage datatype—used for cursor operations
table	n/a	n/a	Not a storage data type—used for query operations

Data Type Precedence

- Data type precedence determines which data type will be chosen when expressions of different types are combined
- By default, the data type with the lower precedence is converted to the data type with the higher precedence
- It is important to understand implicit conversions
  - Conversion to a data type of lower precedence must be made explicitly (using CAST or CONVERT functions)
- Example precedence (low to high)
  - CHAR -> VARCHAR -> NVARCHAR -> TINYINT -> INT -> DECIMAL -> TIME -> DATE -> DATETIME2 -> XML
- Not all combinations of data type have a conversion (implicit or explicit)

When are Data Types Converted?

- Data type conversion scenarios
  - When data is moved, compared to or combined with other data
  - During variable assignment
- Implicit conversion
  - When comparing data of one data type to another
  - Transparent to the user

```
WHERE <column of smallint type> = <value of int type>
```

- Explicit conversion
  - Uses CAST or CONVERT functions

```
CAST(unitprice AS INT)
```

Demonstration: SQL Server Data Types

In this demonstration, you will see how to:

- Convert data types

Lesson 2: Working with Character Data

- Character Data Types
- Collation
- String Concatenation
- Character String Functions
- The LIKE Predicate
- Demonstration: Working with Character Data

Character Data Types

- SQL Server supports two kinds of character data as fixed-width or variable-width data:
  - Single-byte: char and varchar
    - One byte stored per character
      - Only 256 possible characters—limits language support
  - Multibyte: nchar and nvarchar
    - Multiple bytes stored per character (usually two bytes, but sometimes up to four)
      - More than 65,000 characters represented—multiple language support
      - Precede character string literals with N (National)
  - text and ntext data types are deprecated, but may still be used in older systems
    - In new development, use varchar(max) and nvarchar(max) instead

Collation

- Collation is a collection of properties for character data
  - Character set
  - Sort order
  - Case sensitivity
  - Accent sensitivity
- When querying, collation awareness is important for comparison
  - Is the database case-sensitive? If so:
    - 'Funk‘ does not equal 'funk‘
    - SELECT * FROM HR.Employee does not equal SELECT * FROM HR.employee
- Add COLLATE clause to control collation comparison

```
SELECT empid, lastname
FROM HR.employees
WHERE lastname COLLATE Latin1_General_CS_AS = N'Funk';
```

String Concatenation

- The + (plus) operator and the CONCAT function can both be used to concatenate strings in SQL 2016
  - Using CONCAT
    - Converts input values to strings and converts NULL to empty string

```
SELECTcustid, city, region, country,
  CONCAT(city, ', ' + region, ', ' + country) AS location
FROM Sales.Customers;
```

  - Using + (plus)
    - No conversion of NULL or data type

```
SELECT empid, lastname, firstname,
firstname + N' ' + lastname AS fullname
FROM HR.Employees;
```

Character String Functions

- Common functions that modify character strings

Function	Syntax	Remarks
SUBSTRING	SUBSTRING (expression , start , length)	Returns part of an expression.
LEFT, RIGHT	LEFT (expression , integer_value) RIGHT (expression , integer_value)	LEFT returns left part of string up to integer_value. RIGHT returns right part of string up to integer value.
LEN, DATALENGTH	LEN (string_expression) DATALENGTH (expression)	LEN returns the number of characters in string_expression, excluding trailing spaces. DATALENGTH returns the number of bytes used.
CHARINDEX	CHARINDEX (expressionToFind, expressionToSearch)	Searches expressionToSearch for expressionToFind and returns its start position if found.
REPLACE	REPLACE (string_expression , string_pattern , string_replacement)	Replaces all occurrences of string_pattern in string_expression with string_replacement.
UPPER, LOWER	UPPER (character_expression) LOWER (character_expression)	UPPER converts all characters in a string to uppercase. LOWER converts all characters in a string to lowercase.

The LIKE Predicate

- The LIKE predicate can be used to check a character string for a match with a pattern
- Patterns are expressed with symbols
  - % (Percent) represents a string of any length
  - _ (Underscore) represents a single character
  - [<List of characters>] represents a single character within the supplied list
  - [<Character> - <character>] represents a single character within the specified range
  - [^<Character list or range>] represents a single character not in the specified list or range
  - ESCAPE Character allows you to search for characters that would otherwise be treated as part of a pattern - %, _, [, and])

```sql
SELECT categoryid, categoryname, description
FROM Production.Categories
WHERE description LIKE 'Sweet%';
```

Demonstration: Working with Character Data

In this demonstration, you will see how to:

- Manipulate character data

Lesson 3: Working with Date and Time Data

- Date and Time Data Types
- Entering Date and Time Data Types Using Strings
- Working Separately with Date and Time
- Querying Date and Time Values
- Date and Time Functions
- Demonstration: Working with Date and Time Data

Date and Time Data Types

- Older versions of SQL Server support only datetime and smalldatetime data types
- SQL Server 2008 introduced date, time, datetime2 and datetimeoffset data types
- SQL Server 2012 added further functionality for working with date and time data types

Data Type	Storage (bytes)	Date Range (Gregorian Calendar)	Accuracy	Recommended Entry Format
datetime	8	January 1, 1753 to December 31, 9999	Rounded to increments of .000, .003, or .007 seconds	YYYYMMDD hh:mm:ss[.mmm]
smalldatetime	4	January 1, 1900 to June 6, 2079	1 minute	YYYYMMDD hh:mm:ss[.mmm]
datetime2	6 to 8	January 1, 0001 to December 31, 9999	100 nanoseconds	YYYYMMDD hh:mm:ss[.nnnnnnn]
date	3	January 1, 0001 to December 31, 9999	1 day	YYYY-MM-DD
time	3 to 5	n/a – time only	100 nanoseconds	hh:mm:ss[.nnnnnnn]
datetimeoffset	8 to 10	January 1, 0001 to December 31, 9999	100 nanoseconds	YYYY-MM-DDThh:mm:ss[.nnnnnnn][{+|-}hh:mm]

Entering Date and Time Data Types Using Strings

- SQL Server doesn't offer a means to enter a date or time value as a literal value
  - Dates and times are entered as character literals and converted explicitly or implicitly
    - For example, char converted to datetime due to precedence
  - Formats are language-dependent, and can cause confusion
- Best practices:
  - Use character strings to express date and time values
  - Use language-neutral formats

```sql
SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHERE orderdate = '20070825';
```

Working Separately with Date and Time

- datetime, smalldatetime, datetime2, and datetimeoffset include both date and time data
- If only date is specified, time set to midnight (all zeros)

```
DECLARE @DateOnly AS datetime2 = '20160112';
SELECT @DateOnly AS Result;
```

- If only time is specified, date set to base date (January 1, 1900)

```
DECLARE @time AS time = '12:34:56';
SELECT CAST(@time AS datetime2) AS Result;
```

Querying Date and Time Values

- Date values converted from character literals often omit time
  - Queries written with equality operator for date will match midnight

```
SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHERE orderdate= '20070825';
```

- If time values are stored, queries need to account for time past midnight on a date
  - Use range filters instead of equality

```
SELECT orderid, custid, empid, orderdate
FROM Sales.Orders
WHEREorderdate >= '20070825'
ANDorderdate <  '20070826';
```

Date and Time Functions

- To get system date and time values
  - For example, GETDATE, GETUTCDATE, SYSDATETIME
- To get date and time parts
  - For example, DATENAME, DATEPART
- To get date and time values from their parts
  - For example, DATETIME2FROMPARTS, DATEFROMPARTS
- To get date and time difference
  - For example, DATEDIFF, DATEDIFF_BIG
- To modify date and time values
  - For example, DATEADD, EOMONTH
- To validate date and time values
  - For example, ISDATE

Demonstration: Working with Date and Time Data

In this demonstration, you will see how to:

- Query date and time values

Lab: Working with SQL Server 2016 Data Types

- Exercise 1: Writing Queries That Return Date and Time Data
- Exercise 2: Writing Queries That Use Date and Time Functions
- Exercise 3: Writing Queries That Return Character Data
- Exercise 4: Writing Queries That Use Character Functions

Lab Scenario

You are an Adventure Works business analyst who will be writing reports using corporate databases stored in SQL Server 2016. You have been given a set of business requirements for data and you will write T-SQL queries to retrieve the specified data from the databases. You will need to retrieve and convert character, and date and time data into various formats.

Module Review and Takeaways

- Review Question(s)