

# SQL Server for Developers Advanced T-SQL and Troubleshooting

## Module 1: Data Types and System Functions

Bob Beauchemin  
BobB@SQLskills.com



### Overview

- Data Types in SQL Server
- Spatial Data and Indexes
- Filestream Storage and FileTables
- Data Operations in SQL Server
  - Functions and Operators
  - Methods and Properties
- Nullability and Data Integrity
- Schemas
- Contained Databases
- Partitioning



2

© SQLskills, All rights reserved.  
<http://www.SQLskills.com>



## Data Types in SQL Server

- **Numeric series**
  - Integral series
  - Fixed decimal
  - Floating point decimal
- **Character string series**
  - Unicode and Non-Unicode (Fixed or variable length)
  - Large value types - `varchar(max)`, `nvarchar(max)`
  - Text and NText (deprecated)
- **Date and time series**
  - Datetime, Smalldatetime
  - `DateTime2`, `Date`, `Time`, `DateTimeOffset`
- **Binary data**
  - Varbinary
  - `Varbinary(max)`
  - `Varbinary(max)` with Filestream storage
  - IMAGE (deprecated)

## Data Types in SQL Server - 2

- **Special purpose**
  - Money and Smallmoney
  - TABLE and **Strongly Typed TABLE**
  - Uniqueidentifier
  - Rowversion (Timestamp)
  - BIT
  - SQL\_VARIANT
  - XML
  - Type Aliases
- **SQLCLR-based data types**
  - **Geometry**
  - **Geography**
  - **HierarchyID**
  - User-Defined Types

## New Date and Time Data Types

- **SQL Server 2008 extends date/time support**
- **Larger Value Space**
  - Current DATETIME - 1753-9999 Years
  - Current DATETIME - 0.00333 Second Accuracy
  - New Date Types - 0001-9999 Years
  - New Date/Time Types - Variable Precision to 100 nanoseconds
- **Variable Precision Saves Space**
- **Separate Date and Time Saves Space**
- **ANSI Compatible**

## DATETIME2 & DATETIMEOFFSET

- **DATETIME2 Data Type**
  - Extended version of DATETIME
  - New "current datetime" functions
- **DATETIMEOFFSET**
  - Time Zone Offset (From UTCTime) Preserved
  - Not Time Zone Aware - No Daylight Saving Time Support
  - Convert DATETIME, DATETIME2 with TODATETIMEOFFSET
  - SWITCHOFFSET(datetimeoffset, timezone)

## Strongly Typed Tables

- **Table Variable**
  - DECLARE @t table
  - DECLARE @t table (id int, name varchar(20))
- **Table Type = Name + Shape**
  - CREATE TYPE name\_table as table (
    - id int, name varchar(20))
  - DECLARE @t name\_table
- **Useable with variables, not as column in table**
- **Required for table-valued parameters**
  - DECLARE @t name\_table = (1, 'bob')
  - EXECUTE dbo.addnames @t

## Data Operations in SQL Server

- **Arithmetic Functions and Operators**
  - +, -, \*, /, % and +=, -=, ...
- **Character Functions and Operators**
  - LEFT(), RIGHT(), SUBSTRING(), STUFF()...
- **Aggregate Functions**
- **Date and Time Functions**
- **Cast and Convert**
- **Metadata Functions**
- **Methods and Properties on Data Types**
  - WRITE, XML methods
- **Session Variables**
  - @@version, @@error, @@trancount, @@rowcount

## Function Examples

- **Datetime Functions**

(expanded in SQL Server 2008)

```
SELECT getdate() + 1
```

```
SELECT sysdatetime()
```

```
SELECT dateadd(yy, 3, getdate())
```

```
SELECT datepart(iso_week, getdate())
```

```
SELECT datediff(yy, '20000914', '20040101')
```

Be careful with DATEDIFF – Only takes out the boundaries  
requested (year boundary) 2004-2000

```
SELECT CONVERT(char(10), getdate(), 102)
```

```
AS [yyyy.mm.dd]
```

Always use a column header with a style change on date!

## New System Functions

- **Conversion**

- PARSE, TRY\_PARSE, TRY\_CONVERT, TRY\_CAST

- **String**

- CONCAT, FORMAT

- **Date and time**

- [DateTimeDataType]FROMPARTS
- EOMONTH

- **Logical**

- CHOOSE, IIF

## PARSE

- **Convert nvarchar to choice of data type**
  - Optional locale specification  
`PARSE ( string_value AS data_type [ USING culture ] )`
- **Useful for numeric and date/time types**
  - Otherwise use CAST or CONVERT
  - Other data types return compile-time error
  - Uses doc'd parsing styles and locale specifiers
- **String value cannot be NULL**
  - NULL as literal is compile-time error
    - Runtime error if dynamic SQL
  - Parameter with NULL value – returns NULL
- **Runtime error if string invalid for type**

## TRY\_PARSE and TRY\_CONVERT

- **PARSE and CONVERT return runtime errors**
  - If conversion fails
- **TRY\_PARSE and TRY\_CONVERT/TRY\_CAST**
  - Return NULL if conversion fails
    - Most failed conversions
    - If using invalid style in TRY\_CONVERT
  - Returns compile-time error if conversion not permitted
    - e.g. number to XML
    - Invalid data type in TRY\_PARSE

## CONCAT

- **Varargs function (2->n arguments)**
- **Ignores NULL values**
  - Like CONCAT\_NULL\_YIELDS\_NULL OFF
- **Accepts string and non-string arguments**
  - Converts non-string to string
- **Return type depends on arguments**
  - See chart in Books Online

## Other Caveats

- **FORMAT**
  - Returns NULL if format string invalid
  - Returns error if locale invalid
- **[DateTimeDataType]FROMPARTS**
  - Compile-time error if not correct # of arguments
  - Runtime error if invalid argument
  - Returns NULL if any argument is NULL
- **EOMONTH**
  - Always returns zero for time portion
  - Be careful using it with BETWEEN
- **FROMPARTS/EOMONTH are non-deterministic**
- **CHOOSE return type is SQL\_VARIANT**

## Remoting and New Functions

- **Queries with linked servers allow a part of the query to be executed on the remote side**
  - For performance improvement
- **New functions based on CLR not remoted**
  - PARSE, TRY\_PARSE, FORMAT
- **Other new functions not remoted**
  - If remote side is not SQL Server 2012

## Data Type Properties and Methods

- **SQL Server 2008 types based on SQLCLR**
  - Geometry
  - Geography
  - HierarchyId
- **You can build your own UDT (as of SQL2005)**
- **Data Types exposed as SQLCLR UDTs**
  - Use '.' syntax for properties
  - Use '.' syntax for instance methods
  - Use '::' syntax for static methods
  - Methods and Properties are case-sensitive
- **Input - automatic coercion from varchar**
  - Calls Parse()
- **Output - native (binary) or ToString()**

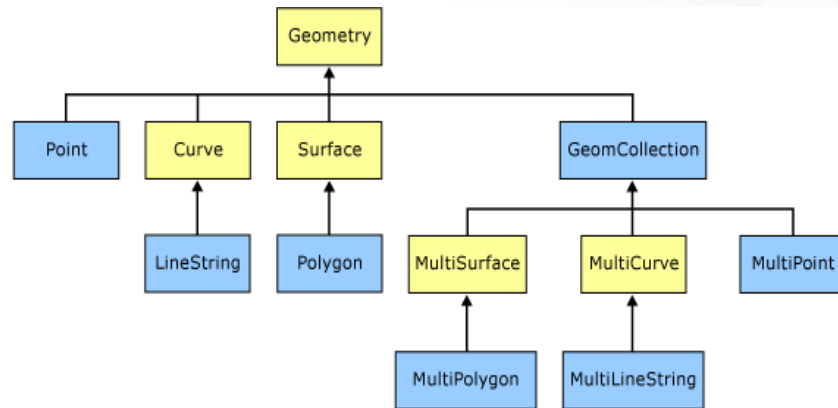
## Spatial Data

- **Spatial data provides answers to location-based queries**
  - Which roads intersect the Microsoft campus?
  - Does my land claim overlap yours?
  - List all of the Italian restaurants within 5 kilometers
- **Spatial data is part of almost every database**
  - If your database includes an address

## SQL Server 2008 and Spatial Data

- **SQL Server supports two spatial data types**
  - GEOMETRY - flat earth model
  - GEOGRAPHY - round earth model
- **Both types support all of the instanciable OGC types**
- **Supports two dimensional, vector data**
  - X and Y or Lat and Long members
  - Z member - elevation (user-defined semantics)
  - M member - measure (user-defined semantics)

## OGC Hierarchy of Spatial Types



## Useful Methods/Properties

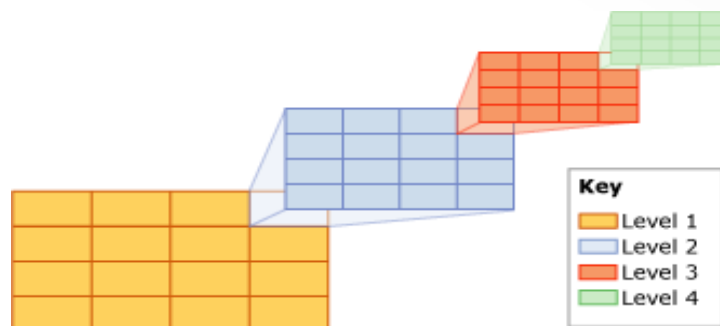
- **Input/Output with WKT or GML format**
- **Descriptive**
  - STArea
  - STLength
  - STCentroid
- **Relation between two instances**
  - STIntersects
  - STDistance
- **Collections**
  - STGeometryN
  - STPointN

## Spatial Indexes

- **SQL Server Spatial Indexes Based on B-Trees**
  - Uses tessellation to tile 2D to linear
  - Divides space into grids (uses Hilbert algorithm)
  - Two gridding algorithms
    - GEOMETRY\_GRID and GEOGRAPHY\_GRID
- **Meant as a first level of row elimination**
  - Can produce false positives
  - Never false negatives

## Spatial Index Grid Hierarchy

- **Specify four grid levels**
  - Low (4x4), medium (8x8), or high (16x16) density
  - Specify bounding box (Geometry\_Grid only)



## SQL Server Spatial 2012

- **New Spatial Types**
  - CircularString, CompoundCurve, CurvePolygon
  - FullGlobe
  - Geography bigger than hemisphere
- **New SRID – Earth Unit Sphere - 104001**
- **Better precision for constructed types**
- **New Spatial Methods**
  - For curves, geography, and more
- **Spatial Aggregates**
- **New Spatial Index Type – AUTO GRID**
- **Query optimizations and new hints**

## Filestream Storage

- **Storing large binary objects in databases is suboptimal**
  - Large objects take buffers in database memory
  - Updating large objects cause database fragmentation
    - In file system however, "update" is delete and insert
    - "Before image" in an update is not deleted immediately
- **Storing all related data in a database adds**
  - Transactional consistency
  - Integrated, point-in-time backup and restore
  - Single storage and query vehicle

## Filestream Implementation

- **A filegroup for filestream storage is declared using DDL**
  - Filestream storage is tied to a database
- **The filegroup is mapped to a directory**
  - Must be NTFS file system
  - Caution: Files deletable from file system if you have appropriate permissions
- **VARBINARY(MAX) columns can be defined with FILESTREAM attribute**
  - Table must also have UNIQUEIDENTIFIER column
  - Filestream storage not available for other large types
- **Data is stored in the file system**

## Programming with Filestreams

- **Filestream columns available with SQL methods**
  - If SQL is used, indistinguishable from varbinary(max)
- **Filestream can be stored and accessed using file IO**
  - PathName function gets a symbolic path name
  - Acquire context with
    - GET\_FILESTREAM\_TRANSACTION\_CONTEXT
  - Use SqlFileStream type or OpenSqlFileStream to get a handle based on
    - File Name
    - Required Access
    - Access Options
    - Filestream Transaction Context

## FileTable

- **FileTable is a fixed format table (SQL 2012)**
  - Uses filestream storage and container
  - Includes file system properties as columns
  - HierarchyID columns presents data as "synthesized" hierarchical file system share
- **Available as file system share or T-SQL table**
- **Can be maintained through the share directly**
  - Some columns maintainable through T-SQL
  - Preserves filename and suffix
- **Allows non-transactional access through share**

## FileTable Creation

- **Enable filestream feature**
- **ALTER database to**
  - Allow non-transactional access
  - Specify directory name for share
- **CREATE filetable**
  - Specify filetable\_directory (share subdirectory)

## Nullability - Missing Values

- No specific value required at INSERT
- NULL values DO NOT mean empty space (stored separately from the column data)
- Working with NULLs
  - Accessing columns which allow NULL values can cause inconsistencies when developers/users are not aware of them
  - Math with NULL values can produce interesting results (? – NULL = NULL)
  - ANSI session settings can affect results sets when accessing columns that allow nulls
- NULL values v. DEFAULT value

## NULL Values

```
SELECT t.title, t.price
FROM pubs.dbo.titles AS t
WHERE t.price IS NULL
```

```
SELECT t.title, t.price
FROM pubs.dbo.titles AS t
WHERE t.price IS NOT NULL
```

```
SELECT t.title, ISNULL(t.price, 0)
FROM pubs.dbo.titles AS t
```

- NULL Values must be evaluated using IS or IS NOT
- Do NOT use column = NULL when ANSI\_NULLS is ON
- Do NOT turn ANSI\_NULLS off as it can have a negative impact on performance
- Replacing a NULL Value with a “real” value can be easy with ISNULL function

## NULL and comparison

- **NULLs almost never compare equal, but...**
  - NULLs sort together in ORDER BY
  - NULLs form a single group in GROUP BY
  - Two NULL values return one DISTINCT row
- **Functions and operators return NULL when any parameter is NULL**
  - Except concat when concat\_null\_yields\_null off
- **Mutators (like VARCHAR(max).Write) cause errors when invoked on NULL values**

```
DECLARE @str VARCHAR(max)
-- fails, mutator on NULL value
SET @str.WRITE(@newchunk, @offset, @remove)
```

## Coding in Relational Database

- **Relational data is based on set theory**
  - No support in RDBMS for multi-valued data
  - Each value should represent something unique
  - No two rows should be identical
  - The order of rows and columns is irrelevant to the meaning of the data
- **Set-based solutions are almost always BETTER than cursor-based solutions**
  - The key is thinking in sets

## Structuring Data For Queries

- Columns should use appropriate data types
- This makes for straightforward queries with less data coercion
- Choosing the right structure is crucial for best performance
  - Data Type
  - Primary Key Choice
  - Data Placement
  - Indexes

## Uniqueness

- Not all data is intrinsically unique
- To enforce uniqueness
  - Identity property
  - Uniqueidentifier data type
  - Sequence object (SQL Server 2012)
- Neither one guarantees uniqueness
  - Use in conjunction with constraints

## Identity vs. Sequence

- **Sequence is a database object**
  - Can be used with multiple columns
  - Not auto-generated
    - Can use "NEXT VALUE FOR" as DEFAULT
    - Can be generated on server or client
    - Series of sequence numbers can be reserved
- **Identity is a column property**
  - Value generated during insert statement
  - Cannot be updated
- **Both useable with same data types**
  - Integral series and decimal with zero scale

## UniqueIdentifier

- **GUID-based data type**
  - 16 bytes – represented as hex string
- **Required for some features (ROWGUIDCOL)**
  - Merge replication
  - Filestream
- **Can be auto-generated using NEWID() or NEWSEQUENTIALID()**
  - Using NEWID() always causes fragmentation

## Data Integrity Affects Query

- **Good design makes a database**
  - Easier to query
    - JOINS more straightforward
    - Less need for outer joins
  - Queries are more performant
  - Easier to maintain
  - Less subject to update anomalies
- **Data Integrity consists of**
  - Domain integrity (columns and values)
  - Entity integrity (rows)
  - Relational integrity (between/among entities)

## Foreign Key References

- **Foreign Key enforces domain and referential integrity**
  - Domain – restricts set of legal values to set in referenced table/column(s)
  - Referential – enforces relationship between the tables (FK must reference PK or UK)
    - INSERTs/UPDATEs in the referencing table must supply a value which exists
    - DELETEs in the referenced table are not allowed when row is referenced
- **Foreign Key reference can cascade, set null, or set to default**
  - ON UPDATE – clean, easy, consistent
  - ON DELETE – Too easy, beware!

## Enforcing Data Integrity

- **Declarative Data Integrity**
  - Criteria defined in object definitions
  - SQL Server enforces automatically
  - Implement by using constraints, defaults, and rules
  - Used by optimizer
- **Procedural Data Integrity**
  - Criteria defined in code
  - Script enforces
  - Implement by using triggers and stored procedures

## Schemas

- **SQL Server allows multiple schemas**
  - Schemas exist independent of users
- **Schemas can be owned by roles, approles**
  - Eases management when personnel changes
- **Schema owner is default object owner**

## Users and Schemas

- **Each user has a default schema**
  - if default schema not specified, it is dbo
- **Application role can have default schema**
- **Roles do not have default schemas**
  - Windows Groups do in SQL Server 2012
- **User may not have create rights in default schema**
  - May have create rights in other schema
  - 2-part name must be used to create in non-default

## Instances, Databases, or Schemas

- **Applications can be partitioned by schema**
  - Schemas provide an alternative to having everything owned by dbo
  - Need 2-part names
  - Avoids cross database ownership chains
  - Is every file on the file system owned by admin?
  - SQLCLR code is shared by assembly owner
- **Applications can be partitioned by database**
  - Need 3-part names
  - No default cross database ownership chaining
  - TRUSTWORTHY permission for off-database
  - Not 100% separate – logins still shared
- **Separate instances provide full security**
  - Need 4-part names
  - Do not share logins
  - Overhead when linked server calls
  - Delegation issues if on separate machines
  - New EXECUTE AT query hint

## Contained Databases

- **Meant to separate application (in-database) and system (in instance) objects and settings**
  - Makes databases more portable
- **Introduced in SQL Server 2012**
  - Partially contained databases
  - Database-level authorization
  - Standard collation handling
    - Database collation and Catalog collation
    - Vs. Database collation, Instance, TempDB collation
  - DMV and XEvent point out uncontained usage

## Synonyms

- **Synonym is an alternate name for object**
  - Can have synonyms on 2,3,4 part names
  - Synonyms live in schemas
  - 1-part synonym cannot be used for UDF
- **Synonyms cannot be**
  - Used for full-text DML
  - Referenced from a linked server
  - Referenced from schema-bound objects
    - Views and functions with schemabinding
    - Check constraints, defaults, rules, computed columns

## Partitioning

- **Why Partition?**
  - Resource Contention Limitations
  - Varying Access patterns
  - Maintenance Restrictions
  - Availability Requirements
  - To remove resource blocking or minimize maintenance costs
  - But:
    - Partitioning does not automatically mean Distributed Partitioned Views (DPVs)
    - DPVs are a form of scale-out partitioning

## Partitioning Strategies

- **Vertical Partitioning**
  - Multiple tables – “subsets” of columns
  - Different column sets
    - Separate sets based on usage patterns
    - Primary key is redundant, possibly other columns
- **Horizontal Range Partitioning**
  - Multiple tables – all columns
  - Different row sets
    - Separate sets based on date range
    - Separate sets based on lists – also “range” partitions

## Partitioning Steps

- **Create filegroups**
  - Add files to filegroups
- **Create partition function**
  - Can ONLY include ONE column of a table
  - Always include entire domain of possible values
  - Always define n-1 boundary points for n partitions
- **Create partition scheme**
  - Maps partitions to filegroups
- **Define tables, indexes that use partition scheme**

## Review

- **SQL Server has a full complement of data types**
  - Additional date and time types
  - Hybrid and relational and file system
    - FileTable with Filestream storage
  - Spatial and XML data
- **Using the right data type**
  - Optimizes Storage and Query
- **Use schemas to keep ownership under control**
- **Partition for different access patterns**

# Questions?

