

SQLskills Immersion Event

IETS: Advanced Transact-SQL Programming

Module 1: Data Types and System Functions

Bob Beauchemin
BobB@SQLskills.com



Overview

- Data Types in SQL Server
- Spatial and HierarchyID Types
- Data Operations in SQL Server
 - Functions and Operators
 - Methods and Properties
- Filestream and FileTable
- Uniqueness and Data Integrity
- Deployment
 - Objects
 - Schemas
 - Contained Databases
 - Synonyms
 - Partitioning



2

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



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

Data Types in SQL Server (1)

- **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**
 - Traditional DATETIME - 1753-9999 Years
 - Traditional 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 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()**

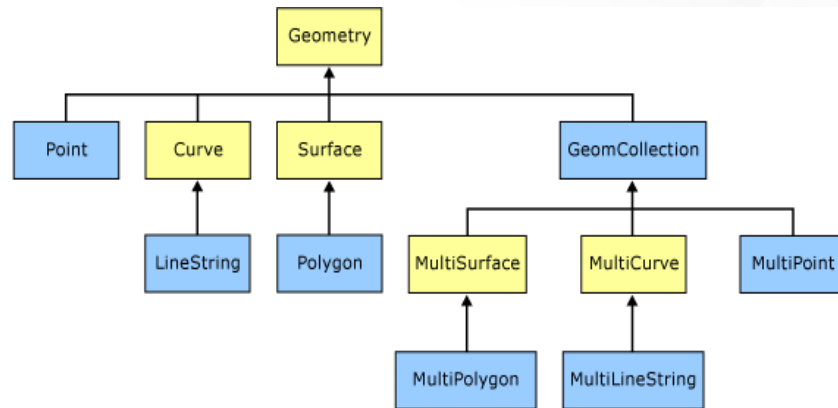
Spatial Data

- **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

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**

Data Operations in SQL Server

- **Arithmetic Functions and Operators**
 - +, -, *, /, % and +=, -=, ...
 - Compound assignment operators added in 2008
- **Character Functions and Operators**
 - LEFT(), RIGHT(), SUBSTRING(), STUFF()...
- **Aggregate Functions**
- **Cast and Convert, Null-handling functions**
- **Date and Time Functions**
- **Metadata Functions**
- **Methods and Properties on Data Types**
 - WRITE, XML methods
- **Session Variables**
 - @@version, @@error, @@trancount, @@rowcount

Aggregates

- **SQL Server supports a superset of ANSI standard aggregate functions**
 - ANSI standard aggregates include COUNT, AVG, MIN, MAX, SUM
 - Some (few) additional built-in aggregates
 - User-defined aggregate functions using .NET
 - When using most aggregate functions, rows with NULL values are thrown away
 - Count of a column is the count of not NULL values and count(*) is a row count. If the column does not allow null values then count(column) will be as efficient as count(*).
 - COALESCE, and ISNULL can provide defaults for NULL values

ANSI and SQL Server-Specific Functions

- **ISNULL, COALESCE, and NULLIF**
 - COALESCE is an ANSI standard
 - Varargs function to replace NULL with non-NULL value
 - ISNULL is SQL Server specific
 - Two arguments – replace NULL with non-NULL value
 - Slightly different than COALESCE with type handling
 - Slightly faster than COALESCE
 - NULLIF
 - Replaces specific value with NULL value
- **CAST and CONVERT**
 - CAST is an ANSI standard
 - CONVERT is SQL Server specific
 - Has third operand for datatype-specific refinements
 - More powerful than CAST

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 SQL 2012 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 documented parsing styles and locale specifications
- **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

New Functions - 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

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)

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

UniquelIdentifier

- **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

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)
```

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)
 - Entity – row-wise constraints
 - 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 on UPDATE or DELETE**
 - Cascade
 - Set Null
 - Set Default
 - No Action

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

Database Objects (1)

- **Tables**
 - Unit of storage and query
- **Indexes**
 - Materialized form of subset of columns
 - Special indexes are tables that include “derived columns”
 - Columnstore index change storage unit from row/page to column/segment
- **Constraints**
 - Nullability and multiple constraint types
- **Views**
 - Virtual table, definition is kept in database
- **Schemas**
 - Container for objects
- **Synonym**
 - Alternate name for object

Database Objects (2)

- **Stored Procedures**
 - Modules to encapsulate multiple statements for use cases
 - Can provide security
 - Can validate parameters
 - Can be used as a transaction boundary
- **User-Defined Functions**
 - Inline Table-Valued – view with parameters
 - Multi-statement Table-Valued – procedure that returns table variable
 - Scalar – For use in SQL statements – computed columns
- **Trigger**
 - For automatic response with DML actions, DDL, logins

SQL 2014 – Memory Optimized Tables

- **Durable and non-durable**
- **Durable tables are (re)-loaded at startup time**
 - Must have enough memory
- **Non-durable are not persisted or logged**
 - For ETL staging or temp table replacement
 - Strongly-typed table variables supported
- **Hash and range indexes**
- **Limitations**
 - In table definitions
 - Some T-SQL constructs not supported
 - Not supported with all SQL Server features

Schemas

- **SQL Server allows multiple schemas**
 - Schemas exist independent of users
- **Schemas can be owned by roles as well as users**
 - 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?
- **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
 - Linked server calls incur overhead
 - Delegation issues if instances on separate machines
 - New EXECUTE AT query hint gives more control

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
- **Partitioning can be implemented with views or tables, but**
 - Table partitioning built-in to SQL Server 2005+
 - 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?