Implement Data Types

Section Objective:

This objective may include but is not limited to: FILESTREAM; spatial, structured, and semi-structured; collations

Data Type Definition:

Objects that contain data have an associated data type that defines the kind of data; for example, character, integer, or binary, the object can contain. The following objects have data types:

- Columns in tables and views.
- Parameters in stored procedures.
- Variables.
- Transact-SQL functions that return one or more data values of a specific data type.
- Stored procedures that have a return code, which always has an **integer** data type.

Assigning a data type to an object defines four attributes of the object:

- The kind of data contained by the object.
- The length or size of the stored value.
- The precision of the number (numeric data types only).
- The scale of the number (numeric data types only).

Data Types Categories:

- Structured
    - Exact Numerics
        - BigInt
        - Bit
        - Decimal
        - Int
        - Money
        - Numeric
        - Smallint
        - Smallmoney
        - Tinyint
    - Approximate Numerics
        - Float
        - Real
    - DateTime
        - Date
        - Datetime2
        - Datetime

        - Datetimeoffset
        - Smalldatetime
        - Time
    - Character Strings
        - Char
        - Text
        - Varchar
    - Unicode Character Strings
        - Nchar
        - Ntext
        - Nvarchar
    - Binary Strings
        - Binary
        - Image
        - Varbinary
- Non-Structured
    - SQL_Variant
    - XML
    - HierarchyID
    - Cursor

Different Type(s) of Data Type(s)

Data Type	Description
BIGINT	**Range:** -2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807) **Storage:** 8 Bytes **Notes:** The **bigint** data type is intended for use when integer values might exceed the range that is supported by the **int** data type. **bigint** fits between **smallmoney** and **int** in the data type precedence chart. Functions return **bigint** only if the parameter expression is a **bigint** data type. SQL Server does not automatically promote other integer data types (**tinyint**, **smallint**, and **int**) to **bigint**.
BINARY	Binary data types of either fixed length or variable length. **binary** [(*n*)] :Fixed-length binary data with a length of *n* bytes, where *n* is a value from 1 through 8,000. The storage size is *n* bytes. **Notes:** When *n* is not specified in a data definition or variable declaration statement, the default length is 1. When *n* is not specified

	with the CAST function, the default length is 30. Use **binary** when the sizes of the column data entries are consistent.
BIT	An integer data type that can take a value of 1, 0, or NULL. The SQL Server Database Engine optimizes storage of **bit** columns. If there are 8 or less **bit** columns in a table, the columns are stored as 1 byte. If there are from 9 up to 16 **bit** columns, the columns are stored as 2 bytes, and so on. The string values TRUE and FALSE can be converted to **bit** values: TRUE is converted to 1 and FALSE is converted to 0.
CHAR	Are character data types of either fixed length or variable length. **char** [(*n*)] Fixed-length, non-Unicode character data with a length of *n* bytes. *n* must be a value from 1 through 8,000. The storage size is *n* bytes. The ISO synonym for **char** is **character**. If you have sites that support multiple languages, consider using the Unicode **nchar** or **nvarchar** data types to minimize character conversion issues. If you use **char,** the recommendation is as follows: • Use **char** when the sizes of the column data entries are consistent.
CLR	
CURSOR	A data type for variables or stored procedure OUTPUT parameters that contain a reference to a cursor. Any variables created with the **cursor** data type are nullable. The operations that can reference variables and parameters having a **cursor** data type are: • The DECLARE *@local_variable* and SET *@local_variable* statements. • The OPEN, FETCH, CLOSE, and DEALLOCATE cursor statements. • Stored procedure output parameters. • The CURSOR_STATUS function. • The **sp_cursor_list**, **sp_describe_cursor**, **sp_describe_cursor_tables**, and **sp_describe_cursor_columns** system stored procedures.
DATE	**Range:**

	0001-01-01 through 9999-12-31 January 1, 1 A.D. through December 31, 9999 A.D. **Storage:** 8 bytes fixed **Default value:** 1900-01-01
DATETIME	**Range:** January 1, 1753, through December 31, 9999 00:00:00 through 23:59:59.997 **Storage:** 8 bytes **Default value:** 1900-01-01 00:00:00
DATETIME2	**Definition:** Defines a date that is combined with a time of day that is based on 24-hour clock. **datetime2** can be considered as an extension of the existing **datetime** type that has a larger date range, a larger default fractional precision, and optional user-specified precision. **Date Range:** 0001-01-01 through 9999-12-31 January 1,1 AD through December 31, 9999 AD **Time Range:** 00:00:00 through 23:59:59.9999999 **Storage:** 6 bytes for precisions less than 3; 7 bytes for precisions 3 and 4. All other precisions require 8 bytes. **Syntax:** **datetime2** [(*fractional seconds precision*)]
DATETIMEOFFSET	**Definition:** Defines a date that is combined with a time of a day that has time zone awareness and is based on a 24-hour clock. **Date Range:** 0001-01-01 through 9999-12-31 January 1,1 A.D. through December 31, 9999 A.D. **Time Range:** 00:00:00 through 23:59:59.9999999 **Storage:** 10 bytes, fixed is the default with the default of 100ns fractional second precision.

<table>
<tr><td></td><td>Syntax:
datetimeoffset [(fractional seconds precision)]</td></tr>
<tr><td>DECIMAL</td><td>Definition: Numeric data types that have fixed precision and scale.

Range: - 10^38 +1 through 10^38 - 1.

decimal[(p[, s])] and numeric[(p[, s])]

Fixed precision and scale numbers. When maximum precision is used, valid values are from - 10^38 +1 through 10^38 - 1. The ISO synonyms for decimal are dec and dec(p, s). numeric is functionally equivalent to decimal.

p (precision)

The maximum total number of decimal digits that can be stored, both to the left and to the right of the decimal point. The precision must be a value from 1 through the maximum precision of 38. The default precision is 18.

s (scale)

The maximum number of decimal digits that can be stored to the right of the decimal point. Scale must be a value from 0 through p. Scale can be specified only if precision is specified. The default scale is 0; therefore, 0 <= s <= p. Maximum storage sizes vary, based on the precision.</td></tr>
<tr><td>FLOAT</td><td>Description: Approximate-number data types for use with floating point numeric data. Floating point data is approximate; therefore, not all values in the data type range can be represented exactly.

Range: - 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308

Storage:
<table>
<tr><th>N Value</th><th>Percision</th><th>Storage Size</th></tr>
<tr><td>1-24</td><td>7 digits</td><td>4 bytes</td></tr>
<tr><td>25 - 53</td><td>15 digits</td><td>8 bytes</td></tr>
</table></td></tr>
<tr><td>HIERARCHYID</td><td>Description:

This data type is system-provided. Use hierarchyid as a data type to create tables with a hierarchical structure, or to reference the hierarchical structure of data in another location. Use the hierarchyid functions to query and perform work with hierarchical data by using Transact-SQL.
Hierarchical data is defined as a set of data items that are related to each other by hierarchical relationships. Hierarchical relationships are</td></tr>
</table>

	where one item of data is the parent of another item. Hierarchical data is common in databases. Examples include the following: • An organizational structure • A file system • A set of tasks in a project • A taxonomy of language terms • A graph of links between Web pages
IMAGE	**Size:** Variable-length binary data from 0 through 2^31-1 (2,147,483,647) bytes. **Note:** This data types will be removed in a future version of Microsoft SQL Server. Avoid using this data type in new development work, and plan to modify applications that currently use it. Use varbinary(MAX) instead.
INT	**Range:** -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647) **Storage:** 4 Bytes The **int** data type is the primary integer data type in SQL Server.
MONEY	**Range:** –922,337,203,685,477.5808 to 922,337,203,685,477.5807 **Storage:** 8 bytes
NCHAR	**Description:** Character data types that are fixed-length, **nchar**, Unicode data and use the UNICODE UCS-2 character set. **nchar** [(n)] Fixed-length Unicode character data of *n* characters. *n* must be a value from 1 through 4,000. The storage size is two times *n* bytes. The ISO synonyms for **nchar** are **national char** and **national character**.
NTEXT	**Range:** Variable-length Unicode data with a maximum length of 2^30 - 1 (1,073,741,823) characters. Storage size, in bytes, is two times the number of characters entered. The ISO synonym for **ntext** is **national text**. **Note:** This data type will be removed in a future version of Microsoft SQL Server. Avoid using this data type in new development work, and plan to modify applications that currently use them. Use nvarchar(max) instead. Fixed and variable-length data types for storing large non-Unicode and Unicode character and binary data. Unicode data uses the UNICODE UCS-2 character set.
NUMERIC	Numeric data types that have fixed precision and scale.

<table>
<tr><td></td><td>decimal[(p[, s])] and numeric[(p[, s])]
Fixed precision and scale numbers. When maximum precision is used, valid values are from - 10^38 +1 through 10^38 - 1. The ISO synonyms for decimal are dec and dec(p, s). numeric is functionally equivalent to decimal.

p (precision)
The maximum total number of decimal digits that can be stored, both to the left and to the right of the decimal point. The precision must be a value from 1 through the maximum precision of 38. The default precision is 18.

s (scale)
The maximum number of decimal digits that can be stored to the right of the decimal point. Scale must be a value from 0 through p. Scale can be specified only if precision is specified. The default scale is 0; therefore, 0 <= s <= p. Maximum storage sizes vary, based on the precision.

<table>
<tr><th>Precision</th><th>Storage bytes</th></tr>
<tr><td>1 - 9</td><td>5</td></tr>
<tr><td>10 - 19</td><td>9</td></tr>
<tr><td>20 - 28</td><td>13</td></tr>
<tr><td>29 - 38</td><td>17</td></tr>
</table>
</td></tr>
<tr><td>NVARCHAR</td><td>Description: Character data types that are variable-length, nvarchar, Unicode data and use the UNICODE UCS-2 character set.

nvarchar [(n | max)]
Variable-length Unicode character data. n can be a value from 1 through 4,000. max indicates that the maximum storage size is 2^31-1 bytes. The storage size, in bytes, is two times the number of characters entered + 2 bytes. The data entered can be 0 characters in length. The ISO synonyms for nvarchar are national char varying and national character varying.</td></tr>
<tr><td>REAL</td><td>Description: Approximate-number data types for use with floating point numeric data. Floating point data is approximate; therefore, not all values in the data type range can be represented exactly.

Range: - 3.40E + 38 to -1.18E - 38, 0 and 1.18E - 38 to 3.40E + 38

Storage: 4 bytes</td></tr>
<tr><td>ROWVERSION</td><td>Is a data type that exposes automatically generated, unique binary numbers within a database. rowversion is generally used as a mechanism for version-stamping table rows. The storage size is 8 bytes. The rowversion data type is just an incrementing number and does not preserve a date or a time. To record a date or time, use a</td></tr>
</table>

	datetime2 data type.
SMALLDATETIME	**Description:** Defines a date that is combined with a time of day. The time is based on a 24-hour day, with seconds always zero (:00) and without fractional seconds. **Date Range:** 1900-01-01 through 2079-06-06 January 1, 1900, through June 6, 2079 **Time Range:** 00:00:00 through 23:59:59 2007-05-09 23:59:59 will round to 2007-05-10 00:00:00 **Storage:** 4 bytes fixed **Syntax:** **smalldatetime**
SMALLINT	**Range:** -2^15 (-32,768) to 2^15-1 (32,767) **Storage:** 2 Bytes
SMALLMONEY	**Description:** Data types that represent monetary or currency values. **Range:** - 214,748.3648 to 214,748.3647 **Storage:** 4 Bytes
SQL_VARIANT	**Description:** The **sql_variant** data type operates similarly to the **variant** data type in Microsoft Visual Basic. **sql_variant** enables a single column, parameter, or variable to store data values of different data types. For example, one **sql_variant** column can hold **int**, **decimal**, **char**, **binary**, and **nchar** values. Each instance of a **sql_variant** column records the data value and the metadata information. This includes the base data type, maximum size, scale, precision, and collation.
TABLE	**Description:** Is a special data type that can be used to store a result set for processing at a later time. The **table** data type is primarily used for temporary storage of a set of rows returned as the result set of a table-valued function.
TEXT	**Description:** Variable-length non-Unicode data in the code page of the server and with a maximum length of 2^31-1 (2,147,483,647) characters. When the server code page uses double-byte characters, the storage is still 2,147,483,647 bytes. Depending on the character string, the storage size may be less than 2,147,483,647 bytes.

TIME	**Description:** Defines a date that is combined with a time of day. The time is based on a 24-hour day, with seconds always zero (:00) and without fractional seconds. **Time Range:** 00:00:00.0000000 through 23:59:59.9999999 **Storage:** 3 to 5 bytes **Syntax:** Time
TIMESTAMP	**Description:** Timestamp is the wrong name, quite confusing in fact. It has nothing to do with time. Microsoft will rename it rowversion in the future. Rowversion is the synonym for timestamp in SQL Server 2005 and SQL Server 2008. **Storage:** It is an 8 bytes unique binary key within the database.
TINYINT	**Range:** 0 to 255 **Storage:** 1 Byte
VARBINARY	**varbinary** [(*n* | **max**)] Variable-length binary data. *n* can be a value from 1 through 8,000. **max** indicates that the maximum storage size is 2^31-1 bytes. The storage size is the actual length of the data entered + 2 bytes. The data that is entered can be 0 bytes in length. The ANSI SQL synonym for **varbinary** is **binary varying**. **Note(s):** When *n* is not specified in a data definition or variable declaration statement, the default length is 1. When *n* is not specified with the CAST function, the default length is 30. Use **varbinary** when the sizes of the column data entries vary considerably. Use **varbinary(max)** when the column data entries exceed 8,000 bytes.
VARCHAR	Are character data types of either fixed length or variable length. **varchar** [(*n* | **max**)] Variable-length, non-Unicode character data. *n* can be a value from 1 through 8,000. **max** indicates that the maximum storage size is 2^31-1 bytes. The storage size is the actual length of data entered + 2 bytes. The data entered can be 0 characters in

	length. The ISO synonyms for **varchar** are **char varying** or **character varying**. If you have sites that support multiple languages, consider using the Unicode **nchar** or **nvarchar** data types to minimize character conversion issues. If **varchar**, the recommendation is as follows: • Use **varchar** when the sizes of the column data entries vary considerably. • Use **varchar(max)** when the sizes of the column data entries vary considerably, and the size might exceed 8,000 bytes.
UNIQUEIDENTIFIER	**Description:** The **uniqueidentifier** data type stores 16-byte binary values that operate as globally unique identifiers (GUIDs). A GUID is a unique binary number; no other computer in the world will generate a duplicate of that GUID value. The main use for a GUID is for assigning an identifier that must be unique in a network that has many computers at many sites.
XML	**Description:** Is the data type that stores XML data. You can store **xml** instances in a column, or a variable of **xml** type.

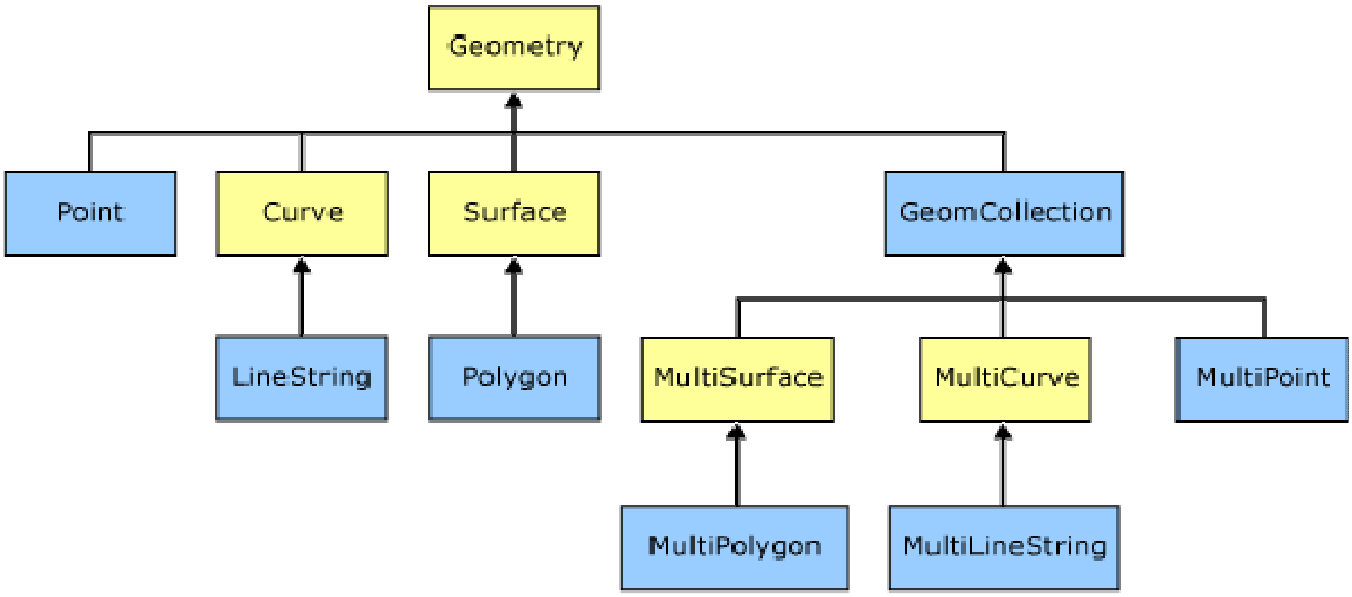
Spatial

There are two types of spatial data. The **geometry** data type supports planar, or Euclidean (flat-earth), data. The **geometry** data type conforms to the Open Geospatial Consortium (OGC) Simple Features for SQL Specification version 1.1.0.

In addition, SQL Server supports the **geography** data type, which stores ellipsoidal (round-earth) data, such as GPS latitude and longitude coordinates.

The **geometry** and **geography** Data Types support eleven spatial data objects, or instance types. However, only seven of these instance types are *instantiable*; you can create and work with these instances (or instantiate them) in a database. These instances derive certain properties from their parent data types that distinguish them as **Points**, **LineStrings**, **Polygons**, or as multiple **geometry** or **geography** instances in a **GeometryCollection**.

The figure below depicts the **geometry** hierarchy upon which the **geometry** and **geography** data types are based. The instantiable types of **geometry** and **geography** are indicated in blue.



Filestream

SQL Server 2008 introduces the FILESTREAM storage attribute for binary (BLOB) data stored in a varbinary(max) column. SQL Server has always provided the capability to store binary data, but working with it has required special handling. Unstructured data, such as text documents, images and video, is often stored outside of the database, making it difficult to manage.

Collations

Collation precedence, also known as collation coercion rules, determines the following:

- The collation of the final result of an expression that is evaluated to a character string.
- The collation that is used by collation-sensitive operators that use character string inputs but do not return a character string, such as LIKE and IN.

The collation precedence rules apply only to the character string data types: **char**, **varchar**, **text**, **nchar**, **nvarchar**, and **ntext**. Objects that have other data types do not participate in collation evaluations.