# Data Types

In SQL Server, each column, local variable, expression, and parameter has a related data type. A data type is an attribute that specifies the type of data that the object can hold: integer data, character data, monetary data, date and time data, binary strings, and so on.

## Data type categories

**Data types in SQL Server are organized into the following categories:**

|  |  |
| --- | --- |
| Exact numerics | Unicode character strings |
| Approximate numerics | Binary strings |
| Date and time | Other data types |
| Character strings |  |

Large value data types: **varchar(max)**, and **nvarchar(max)**

Large object data types: **text**, **ntext**, **image**, **varbinary(max)**, and **xml**

### Exact numerics

|  |  |
| --- | --- |
| [bigint](https://docs.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql) | [numeric](https://docs.microsoft.com/en-us/sql/t-sql/data-types/decimal-and-numeric-transact-sql) |
| [bit](https://docs.microsoft.com/en-us/sql/t-sql/data-types/bit-transact-sql) | [smallint](https://docs.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql) |
| [decimal](https://docs.microsoft.com/en-us/sql/t-sql/data-types/decimal-and-numeric-transact-sql) | [smallmoney](https://docs.microsoft.com/en-us/sql/t-sql/data-types/money-and-smallmoney-transact-sql) |
| [int](https://docs.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql) | [tinyint](https://docs.microsoft.com/en-us/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql) |
| [money](https://docs.microsoft.com/en-us/sql/t-sql/data-types/money-and-smallmoney-transact-sql) |  |

### Approximate numerics

|  |  |
| --- | --- |
| float | real |

### Date and time

|  |  |
| --- | --- |
| [date](https://docs.microsoft.com/en-us/sql/t-sql/data-types/date-transact-sql) | [datetimeoffset](https://docs.microsoft.com/en-us/sql/t-sql/data-types/datetimeoffset-transact-sql) |
| [datetime2](https://docs.microsoft.com/en-us/sql/t-sql/data-types/datetime2-transact-sql) | [smalldatetime](https://docs.microsoft.com/en-us/sql/t-sql/data-types/smalldatetime-transact-sql) |
| [datetime](https://docs.microsoft.com/en-us/sql/t-sql/data-types/datetime-transact-sql) | [time](https://docs.microsoft.com/en-us/sql/t-sql/data-types/time-transact-sql) |

### Character strings

**char varchar text**

### Unicode character strings

**nchar nvarchar ntext**

### Binary strings

**binary varbinary image**

### Other data types

|  |  |
| --- | --- |
| [cursor](https://docs.microsoft.com/en-us/sql/t-sql/data-types/cursor-transact-sql) | [rowversion](https://docs.microsoft.com/en-us/sql/t-sql/data-types/rowversion-transact-sql) |
| [hierarchyid](https://docs.microsoft.com/en-us/sql/t-sql/data-types/hierarchyid-data-type-method-reference) | [uniqueidentifier](https://docs.microsoft.com/en-us/sql/t-sql/data-types/uniqueidentifier-transact-sql) |
| [sql\_variant](https://docs.microsoft.com/en-us/sql/t-sql/data-types/sql-variant-transact-sql) | [xml](https://docs.microsoft.com/en-us/sql/t-sql/xml/xml-transact-sql) |
| [Spatial Geometry Types](https://docs.microsoft.com/en-us/sql/t-sql/spatial-geometry/spatial-types-geometry-transact-sql) | [Spatial Geography Types](https://docs.microsoft.com/en-us/sql/t-sql/spatial-geography/spatial-types-geography) |
| [table](https://docs.microsoft.com/en-us/sql/t-sql/data-types/table-transact-sql) |  |

## Constants

## Data type 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 and 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.
* Explicit conversions use the CAST or CONVERT functions.

## Data type precedence

When an operator combines two expressions of different data types, the rules for data type precedence specify that the data type with the lower precedence is converted to the data type with the higher precedence. If the conversion is not a supported implicit conversion, an error is returned.

SQL Server uses the following precedence order for data types:

1. user-defined data types (highest)
2. **sql\_variant**
3. **xml**
4. **datetimeoffset**
5. **datetime2**
6. **datetime**
7. **smalldatetime**
8. **date**
9. **time**
10. **float**
11. **real**
12. **decimal**
13. **money**
14. **smallmoney**
15. **bigint**
16. **int**
17. **smallint**
18. **tinyint**
19. **bit**
20. **ntext**
21. **text**
22. **image**
23. **timestamp**
24. **uniqueidentifier**
25. **nvarchar** (including **nvarchar(max)** )
26. **nchar**
27. **varchar** (including **varchar(max)** )
28. **char**
29. **varbinary** (including **varbinary(max)** )
30. **binary** (lowest)

## 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.
* Converting to bit promotes any nonzero value to 1.

Select Convert(bit,11) O/P : 1

## Cursor :

A data type for variables or stored procedure OUTPUT parameters that contain a reference to a cursor

* Operations OPEN, FETCH, CLOSE, and DEALLOCATE etc. cursor statements has cursor reference data types.

## Date and time types

### Date :

Defines a date in SQL Server.

### DateTime:

Defines a date that is combined with a time of day with fractional seconds that is based on a 24-hour clock.

Ex : 2007-05-08 12:35:29.123

### DateTime2:

[YYYY-MM-DD hh:mm:ss[.nnnnnnn]]

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.

EX: **datetime2 [2007-05-08 12:35:29. 1234567]**

### DateTimeOffset:

[YYYY-MM-DD hh:mm:ss[.nnnnnnn] [+|-]hh:mm ]

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.

DateTimeOffset 2007-05-08 12:35:29.1234567 +12:15

### Smalldatetime:

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.

**NOTE**: Use the **time**, **date**, **datetime2** and **datetimeoffset** data types for new work. These types align with the SQL Standard. They are more portable. **time**, **datetime2** and **datetimeoffset** provide more seconds precision. **datetimeoffset** provides time zone support for globally deployed applications.

### Time:

Defines a time of a day. The time is without time zone awareness and is based on a 24-hour clock.

## Numeric types

### decimal and numeric

* Numeric data types that have fixed precision and scale. Decimal and numeric are synonyms and can be used interchangeably.

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

|  |  |
| --- | --- |
| Precision | Storage bytes |
| 1-9 | 5 |
| 10-19 | 9 |
| 20-28 | 13 |
| 29-38 | 17 |

### float and real

* 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.

### int, bigint, smallint, and tinyint

* Exact-number data types that use integer data. To save space in the database, use the smallest data type that can reliably contain all possible values.

BigInt – 8 Byte Int- 4 Byte SmallInt- 2 Byte tinyint - 1 byte (0-255)

### money and smallmoney

* Data types that represent monetary or currency values.

**Money – 8 bytes Smallmoney – 4 Bytes**

### Rowversion

* 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 **datetime2** data type.

## String and binary

### binary and varbinary

* 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.
* **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.

### char and varchar

* These data types are of either fixed length or variable length.
* **char** [ ( *n* ) ] Fixed-length, non-Unicode string data. *n* defines the string length and must be a value from 1 through 8,000. The storage size is *n* bytes.
* **varchar** [ ( *n* | **max** ) ] Variable-length, non-Unicode string data. *n* defines the string length and can be a value from 1 through 8,000. **max** indicates that the maximum storage size is 2^31-1 bytes (2 GB). The storage size is the actual length of the data entered + 2 bytes.
* When *n* is not specified in a data definition or variable declaration statement, the default length is 1.
* When *n* is not specified when using the CAST and CONVERT functions, the default length is 30.
* If SET ANSI\_PADDING is OFF when either CREATE TABLE or ALTER TABLE is executed, a **char** column that is defined as NULL is handled as **varchar**.

**Warning** : Each non-null varchar(max) or nvarchar(max) column requires 24 bytes of additional fixed allocation which counts against the 8,060 byte row limit during a sort operation. This can create an implicit limit to the number of non-null varchar(max) or nvarchar(max) columns that can be created in a table. No special error is provided when the table is created (beyond the usual warning that the maximum row size exceeds the allowed maximum of 8060 bytes) or at the time of data insertion. This large row size can cause errors (such as error 512) during some normal operations, such as a clustered index key update, or sorts of the full column set, which users cannot anticipate until performing an operation.

### nchar and nvarchar

* Character data types that are either fixed-length, **nchar**, or variable-length, **nvarchar**, Unicode data and use the UNICODE UCS-2 character set.
* **nchar** [ ( n ) ]

Fixed-length Unicode string data. *n* defines the string length and must be a value from 1 through 4,000. The storage size is two times *n* bytes. When the collation code page uses double-byte characters, the storage size is still *n* bytes. Depending on the string, the storage size of *n* bytes can be less than the value specified for *n*.

* **nvarchar** [ ( n | **max** ) ]

Variable-length Unicode string data. *n* defines the string length and can be a value from 1 through 4,000. **Max** indicates that the maximum storage size is 2^31-1 characters (2 GB). The storage size, in bytes, is two times the actual length of data entered + 2 bytes.

* SET ANSI\_PADDING is always ON for **nchar** and **nvarchar**. SET ANSI\_PADDING OFF does not apply to the **nchar** or **nvarchar** data types.

**Warning**:

Each non-null **varchar(max)** or **nvarchar(max)** column requires 24 bytes of additional fixed allocation which counts against the 8,060 byte row limit during a sort operation. This can create an implicit limit to the number of non-null **varchar(max)** or **nvarchar(max)** columns that can be created in a table. No special error is provided when the table is created (beyond the usual warning that the maximum row size exceeds the allowed maximum of 8060 bytes) or at the time of data insertion. This large row size can cause errors (such as error 512) during some normal operations, such as a clustered index key update, or

sorts of the full column set, which users cannot anticipate until performing an operation.

### ntext, text, and image

* 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.
* **IMPORTANT! ntext**, **text**, and **image** data types will be removed in a future version of SQL Server. Avoid using these data types in new development work, and plan to modify applications that currently use them. Use nvarchar(max), varchar(max), and varbinary(max) instead.
* The following functions can be used with **ntext**, **text**, or **image** data.

DATALENGTH PATINDEX SUBSTRING TEXTPTR TEXTVALID

* The following statements can be used with **ntext**, **text**, or **image** data.

READTEXT SET TEXTSIZE UPDATETEXT WRITETEXT

## Spatial geography

The geography spatial data type, **geography**, is implemented as a .NET common language runtime (CLR) data type in SQL Server. This type represents data in a round-earth coordinate system. The SQL Server **geography** data type stores ellipsoidal (round-earth) data, such as GPS latitude and longitude coordinates.

## Spatial geometry

* The planar spatial data type, **geometry**, is implemented as a common language runtime (CLR) data type in SQLServer. This type represents data in a Euclidean (flat) coordinate system.
* SQL Server supports a set of methods for the **geometry** spatial data type. These methods include methods on **geometry** that are defined by the Open Geospatial Consortium (OGC) standard and a set of Microsoft extensions to that standard.

## sql\_variant

* A data type that stores values of various SQL Server-supported data types.
* **sql\_variant** cannot be used in CONTAINSTABLE and FREETEXTTABLE.

## Table

## Uniqueidentifier

Is a 16-byte GUID.

A column or local variable of **uniqueidentifier** data type can be initialized to a value in the following ways:

* By using the NEWID or NEWSEQUENTIALID functions.
* By converting from a string constant in the form *xxxxxxxx*-*xxxx*-*xxxx*-*xxxx*-*xxxxxxxxxxxx*, in which each *x* is a hexadecimal digit in the range 0-9 or a-f. For example, 6F9619FF-8B86-D011-B42D-00C04FC964FF is a valid **uniqueidentifier** value.
* Comparison operators can be used with **uniqueidentifier** values

## XML

Is the data type that stores XML data. You can store **xml** instances in a column, or a variable of **xml** type.