T-SQL Data Types

www.tsql.info

The Transact SQL language allow you to use various data types like: Numeric (int, numeric, decimal, float), Character Strings (char, varchar), Unicode Character Strings (nchar, nvarchar), Date (date, datetime, datetime2, time) and other data types.

Binary Strings

- Binary
- Varbinary

Character Strings

- Char
- Varchar

Date and Time

- Date
- Datetime
- Datetime2
- Datetimeoffset
- Smalldatetime
- Time

Numerics

- Bigint
- Int
- Smallint
- Tinyint
- Decimal
- Numeric

Unicode Character Strings

- Nchar
- Nvarchar

Other Data Types

- Rowversion
- Uniqueidentifier
- Table

Binary

On Transact SQL language the binary is part of binary strings data types and have fixed length.

The string length must be a value from 1 through 8,000.

Binary syntax:

```
binary [(n)]
```

Binary example:

```
USE model;
GO
DECLARE @myVar BINARY(2);
SET @myVar = 12345678;
SET @myVar = @myVar + 2;
SELECT CAST( @myVar AS INT);
GO
Results
```

Varbinary

On Transact SQL language the binary is part of binary strings data types and have variable length. The string length must be a value from 1 through 8,000.

Varbinary syntax:

```
varbinary [ ( n ) ]
varbinary [ ( max ) ]

Range Storage
2^31-1 bytes (2 GB) 2 Bytes
```

Varbinary example:

```
USE model;
GO
DECLARE @myVar VARBINARY(2);
SET @myVar = 123456789;
SET @myVar = @myVar + 3;
SELECT CAST( @myVar AS INT);
GO

Results
52504
```

Char

On Transact SQL language the char is part of character strings data types and have fixed length.

The string length must be a value from 1 through 8,000.

Char syntax:

```
char[(n)]
```

Char example:

```
USE model;
GO
CREATE TABLE myCharTable ( a char(25) );
GO
INSERT INTO myCharTable VALUES ('abc + def');
GO
SELECT a FROM myCharTable;
GO

Results
abc + def
```

Varchar

On Transact SQL language the varchar is part of character strings data types and have variable length. The string length must be a value from 1 through 8,000.

Varchar syntax:

```
varchar [ ( n ) ]
varchar [ ( max ) ]
```

Varchar example:

```
USE model;
GO
CREATE TABLE varcharTable ( a varchar(10) );
INSERT INTO varcharTable VALUES ('abcdefghij');
SELECT a FROM varcharTable;
GO
    Results
abcdefghij
USE model;
DECLARE @myVar AS varchar(20) = 'abc123';
SELECT @myVar as 'My column', DATALENGTH(@myVar) as 'Length';
GO
 My column
               Length
abc123
             6
```

Date

On Transact SQL language the date is part of date and time data types and define a date on sql server.

Date syntax:

date

Property Value

Default string literal format YYYY-MM-DD

Range 0001-01-01 through 9999-12-31

Length 10

Storage size 3 bytes, fixed Calendar Gregorian

Date example:

USE model;

GO

DECLARE @date date= '08-21-14';

SELECT @date AS 'Date';

GO

Date

2014-08-21

Datetime

On Transact SQL language the datetime is part of date and time data types and define a date that is combined with a time of day with fractional seconds.

Datetime syntax:

datetime

Property Value

Range January 1, 1753, through December 31, 9999

Length 19 minimum - 23 maximum

Storage size 8 bytes Calendar Gregorian

Datetime example:

USE model;

GO

DECLARE @date date= '08-21-14';

DECLARE @datetime datetime = @date;

SELECT @datetime AS 'Datetime', @date AS 'Date';

GO

Datetime Date 2014-08-21 2014-08-00:00:00.000 21

Datetime2

On Transact SQL language the datetime2 is part of date and time data types and is an extension of the datetime type that has a larger date range, a larger default fractional precision.

Datetime2 syntax:

datetime2

Property Value

Default string literal YYYY-MM-DD hh:mm:ss[.fractional

format seconds]

Range 0001-01-01 through 9999-12-31 Length 19 minimum - 27 maximum

Storage size 8 bytes
Calendar Gregorian

Datetime2 example:

USE model;

GO

DECLARE @datetime2 datetime2 = '08-21-14 10:09:30.123';

SELECT @datetime2 AS 'Datetime2';

GO

Datetime2

Datetimeoffset

On Transact SQL language the datetimeoffset is part of date and time data types and define a date that is combined with a time of a day that has time zone awareness.

Datetimeoffset syntax:

datetimeoffset [(fractional seconds precision)]

Property Value

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

format $[\{+|-\}hh:mm]$

Range 0001-01-01 - 9999-12-31 Length 26 minimum - 34 maximum

Storage size 10 bytes

Datetimeoffset example:

USE model;

GO

DECLARE @datetimeoffset datetimeoffset(3) = $'2014-08-21\ 10:49:32.1234+10:0'$;

DECLARE @datetime2 datetime2(3)=@datetimeoffset;

SELECT @datetimeoffset AS 'Datetimeoffset', @datetime2 AS 'Datetime2';

GO

Datetimeoffset

Datetime2

2014-08-21 10:49:32.123 +10:00 2014-08-21 10:49:32.123

Smalldatetime

On Transact SQL language the smalldatetime is part of date and time data types and define a date that is combined with a time of day.

Smalldatetime syntax:

smalldatetime

Property Value

Range 1900-01-01 through 2079-06-06

Length 19 maximum
Storage size 4 bytes
Calendar Gregorian

Smalldatetime example:

USE model;

GO

DECLARE @smalldatetime smalldatetime = '2014-08-21 11:03:17';

DECLARE @date date = @smalldatetime

SELECT @smalldatetime AS 'Smalldatetime', @date AS 'Date';

GO

Smalldatetime Date 2014-08-21 11:03:00 2014-08-21

Time

On Transact SQL language the time is part of date and time data types and define a time of a day.

Time syntax:

time

Property Value

Default string literal format hh:mm:ss[.nnnnnnn]

Range 00:00:00.0000000 - 23:59:59.9999999

Length 8 minimum - 16 maximum

Storage size 5 bytes

Time example:

```
USE model;
GO
DECLARE @time time = '08-21-14 10:21:12.123';
SELECT @time AS 'Time';
GO
```

Time

10:21:12.1230000

Bigint

On Transact SQL language the bigint is an numeric data type. On this page you can read about max value and find an simple example.

Bigint syntax:

Range	Storage
-2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807)	8 Bytes

Bigint example:

```
USE model;
GO
CREATE TABLE test_table ( a bigint );
GO
INSERT INTO test_table VALUES (9223372036854775807);
GO
SELECT a FROM test_table;
GO
```

Results

9223372036854775807

Int

On Transact SQL language the int is an numeric data type. The int data type is the primary integer data type in SQL Server.

Int syntax:

Range Storage

-2³1 (-2,147,483,648) to 2³1-1 (2,147,483,647) 4 Bytes

Int example:

USE model;

GO

CREATE TABLE myIntTable (b int);

GO

INSERT INTO myIntTable VALUES (214483647);

GO

SELECT b FROM myIntTable;

GO

Results

214483647

Smallint

On Transact SQL language the smallint is an numeric data type. Here you can read about max value and find an simple example.

Smallint syntax:

Range Storage -2^15 (-32,768) to 2^15-1 (32,767) 2 Bytes

Smallint example:

USE model;

GO

CREATE TABLE mySmallintTable (c smallint);

GO

INSERT INTO mySmallintTable VALUES (32767);

GC

SELECT c FROM mySmallintTable;

GO

Results

32767

Tinyint

On Transact SQL language the tinyint is an numeric data type. The maximum value of tinyint data type is 255.

Tinyint syntax:

```
Range Storage 0 to 255 1 Byte
```

Tinyint example:

```
USE model;
GO
CREATE TABLE myTinyintTable ( d tinyint );
GO
INSERT INTO myTinyintTable VALUES (255);
GO
SELECT d FROM myTinyintTable;
GO
```

Results

255

Decimal

On Transact SQL language the decimal is the same like numeric data types and have fixed precision and scale.

Decimal syntax:

Precision	Storage
1 - 9	5 Bytes
10-19	9 Bytes

```
20-2829-3813 Bytes17 Bytes
```

Decimal example:

```
USE model;
GO
CREATE TABLE myDecTable ( b decimal (7,2) );
GO
INSERT INTO myDecTable VALUES (234);
GO
SELECT b FROM myDecTable;
GO

Results
234.00
```

Numeric

On Transact SQL language the numeric data types that have fixed precision and scale.

Numeric syntax:

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

Numeric example:

```
USE model;
GO
CREATE TABLE myNumTable ( a numeric (12,6) );
GO
INSERT INTO myNumTable VALUES (777.123);
GO
SELECT a FROM myNumTable;
GO
```

Nchar

On Transact SQL language the nchar is part of unicode character strings data types and have fixed length. The string length must be a value from 1 through 4,000.

Nchar syntax:

```
nchar [(n)]
```

Nchar example:

```
USE model;
GO
CREATE TABLE myNcharTable ( a nchar(20) );
GO
INSERT INTO myNcharTable VALUES ('This is an example');
GO
SELECT a FROM myNcharTable;
GO
```

Results

This is an example

Nvarchar

On Transact SQL language the nvarchar is part of unicode character strings data types and have variable length. The string length must be a value from 1 through 4,000.

Nvarchar syntax:

```
nvarchar [ ( n ) ]
nvarchar [ ( max ) ]

Range Storage
2^31-1 bytes (2 GB) 2 Bytes
```

Nvarchar example:

```
USE model;
GO
CREATE TABLE nvarcharTable ( a nvarchar(25) );
GO
INSERT INTO nvarcharTable VALUES ('This is an example');
GO
SELECT a FROM nvarcharTable;
GO
```

Results

This is an example

Rowversion

On Transact SQL language the rowversion is a data type that generate automatically unique binary numbers within a database. The storage size is 8 bytes.

Rowversion syntax:

rowversion

Rowversion example:

```
USE model;
GO
CREATE TABLE myTest (id int PRIMARY KEY, name char(20), test_column rowversion);
GO
INSERT INTO myTest (id, name) VALUES (1, 'test_1');
GO
INSERT INTO myTest (id, name) VALUES (2, 'test_2');
GO
SELECT * FROM myTest;
GO

id name test_column

1 test_1 0x0000000000000FA1
2 test 2 0x0000000000000000A2
```

Uniqueidentifier

On Transact SQL language the unique identifier is a character type for the purposes of conversion from a character expression.

Uniqueidentifier syntax:

uniqueidentifier

Uniqueidentifier example:

```
USE model;
GO
DECLARE @id_var uniqueidentifier = NEWID();
SELECT @id_var AS 'Result';
GO
```

Result

4BCF38AD-BB98-42CA-82D3-97DBE081EB4B

Table

On Transact SQL language the table variable is a special data type that can be used to store a result set for processing at a later time. Table variables are automatically will be emptied at the end of the function or stored procedure in which they are defined.

Table syntax:

table

Table example:

```
USE model;
```

GO

DECLARE @TableVar table(id int NOT NULL, OldValue char(2), NewValue char(2)); UPDATE my_table SET b='c'

OUTPUT INSERTED.a, DELETED.b, INSERTED.b INTO @TableVar; SELECT id, OldValue, NewValue FROM @TableVar; GO

id OldValue NewValue

1 a c c 2 b c

Resources:

www.tsql.info/data-types/data-types.php