SQL Server Data Types Reference

This sheet provides an easy reference to look up limitations and benefits for each SQL Server data type. There are plenty of sql data types to use in SQL Server. Knowing the limitations and benefit of each sql data type will soon pay off.

Benefits example

Choosing the sql data type ***tinyint*** instead of ***int*** for a "ProductType" column with values ranging from 1 to 10 will save three bytes per record. With 100,000 records you will save 300,000 bytes. That's not much in terms of disc space ("storage is cheap, etc") but ***you'll probably have indexes containing that column*** and if that index takes less memory ***the database engine will process that index much more efficient*** in every "join" and "where" etc.

So, queries will ***perform faster***, ***release locks earlier*** (if any) and ***use less system resources*** (memory and CPU). This will make the whole server perform better as there will be more resources available for other things.

Once learned the sql data types available and spending a few extra minutes when designing your schema will result in faster query execution and an overall better performing database.

The **SQL Data Types** reference sheet

The columns named 8, 9, 10 and 11 indicates SQL Server version data type support where

- 8 = SQL Server 2000
- 9 = SQL Server 2005
- 10 = SQL Server 2008
- 11 = SQL Server 2012

DATATYPE	MIN	MAX	STORAGE	8	9	10	11	TYPE	NOTES
Bigint	-2^63	2^63-1	8 bytes					Exact	
Int	-2,147,483,648	2,147,483,647	4 bytes					Exact	
Smallint	-32,768	32,767	2 bytes					Exact	
Tinyint	0	255	1 bytes					Exact	
Bit	0	1	1 to 8 bit columns in the same table requires a total of 1 byte, 9 to 16 bits = 2 bytes, etc...					Exact	
Decimal	-10^38+1	10^38–1	Precision 1-9 = 5 bytes, precision 10-19 = 9 bytes, precision 20-28 = 13 bytes, precision 29-38 = 17 bytes					Exact	The Decimal and the Numeric data type is exactly the same. Precision is the total number of digits. Scale is the number of decimals. For both the minimum is 1 and the maximum is 38.
Numeric	same as Decimal	same as Decimal	same as Decimal					Exact	
Money	-2^63 / 10000	2^63-1 / 10000	8 bytes					Exact	
Smallmoney	-214,748.3648	214,748.3647	4 bytes					Exact	

DATATYPE	MIN	MAX	STORAGE	8	9	10	11	TYPE	NOTES
Float	-1.79E + 308	1.79E + 308	4 bytes when precision is less than 25 and 8 bytes when precision is 25 through 53					Approx	Precision is specified from 1 to 53.
Real	-3.40E + 38	3.40E + 38	4 bytes					Approx	Precision is fixed to 7.
Datetime	1753-01-01 00:00:00.000	9999-12-31 23:59:59.997	8 bytes					Datetime	If you are running SQL Server 2008 or later and need milliseconds precision, use datetime2(3) instead to save 1 byte.
Smalldatetime	1900-01-01 00:00	2079-06-06 23:59	4 bytes					Datetime	
Date	0001-01-01	9999-12-31	3 bytes	no	no			Datetime	
Time	00:00:00.0000000	23:59:59.9999999	time(0-2) = 3 bytes, time(3-4) = 4 bytes, time(5-7) = 5 bytes	no	no			Datetime	Specifying the precision is possible. TIME(3) will have milliseconds precision. TIME(7) is the highest and the default precision. Casting values to a lower precision will round the value.
Datetime2	0001-01-01 00:00:00.0000000	9999-12-31 23:59:59.9999999	Presicion 1-2 = 6 bytes precision 3-4 = 7 bytes precision 5-7 = 8 bytes	no	no			Datetime	Combines the date datatype and the time datatype into one. The precision logic is the same as for the time datatype.
Datetimeoffset	0001-01-01 00:00:00.0000000 -14:00	9999-12-31 23:59:59.9999999 +14:00	Presicion 1-2 = 8 bytes precision 3-4 = 9 bytes precision 5-7 = 10 bytes	no	no			Datetime	Is a datetime2 datatype with the UTC offset appended.
Char	0 chars	8000 chars	Defined width					String	Fixed width
Varchar	0 chars	8000 chars	2 bytes + number of chars					String	Variable width
Varchar(max)	0 chars	2^31 chars	2 bytes + number of chars	no				String	Variable width
Text	0 chars	2,147,483,647 chars	4 bytes + number of chars					String	Variable width
Nchar	0 chars	4000 chars	Defined width x 2					Unicode	Fixed width
Nvarchar	0 chars	4000 chars						Unicode	Variable width
Nvarchar(max)	0 chars	2^30 chars		no				Unicode	Variable width
Ntext	0 chars	1,073,741,823 chars						Unicode	Variable width
Binary	0 bytes	8000 bytes						Binary	Fixed width
Varbinary	0 bytes	8000 bytes						Binary	Variable width
Varbinary(max)	0 bytes	2^31 bytes		no				Binary	Variable width
Image	0 bytes	2,147,483,647 bytes						Binary	Variable width. Prefer to use the varbinary(max) type as the image type will be removed in future versions.
Sql_variant								Other	Stores values of various SQL Server-supported data types, except text, ntext, and timestamp.
Timestamp			8 bytes					Other	Stores a database-wide unique number that gets updated every time a row gets updated.
Uniqueidentifier			16 bytes					Other	Stores a globally unique identifier (GUID).
Xml				no				Other	Stores XML data. You can store xml instances in a column or a variable.
Cursor								Other	A reference to a cursor.
Table								Other	Stores a result set for later processing.

A note on precision

Space taken by value entries of the types specifying precision (Float, Decimal, DateTime2 etc) is always the same. It's the column definition that defines how much space each entry takes, not the size of the value itself. So a Decimal(25,5) value of 999.999 takes 13 bytes, not 5 bytes. Even a NULL value will take 13 bytes. The column is fixed-length. Even though this might seem bad there's a performance gain CPU-wise when working with fixed-length data (also remember that index trees contains these values and fixed-length storage requirements).

Another consideration is when summing a precision based value the resulting summed values datatype will be the same (if not casted) as the column definition and an arithmetic overflow might occur. If you know your values will, for instance, range from 0 to 999,99 there's no point from a space perspective to not define it as Decimal(9,2) anyways (the highest 5 byte definition). That way your sum result have more space available and you can perhaps avoid some casting. From a constraining perspective a Decimal(5,2) might be more appropriate, but maybe constraints requirements shouldn't be mixed up with data type decisions (well this is another discussion outside the scope of this article).

My best tip is to **"Define your precision as the highest point before storage requirements increases"**.

Summary

Spend some time studying these sql data types. Correctly used sql data types will improve performance, save storage on disk and reduce backup times. It will also help in providing a consistent structure. Choosing sql data types is indeed an important part of constraining a database schema and communicating intended usage.

Version specific data types reference sheets

The SQL Server data types reference sheet have been extracted into **version specific** sheets. They are found at these locations: SQL Server 2012, SQL Server 2008, SQL Server 2005 and SQL Server 2000.

Do you know exactly every sql data type in SQL Server and their usage and storage requirements?

If not, look them all up in this reference sheet.

WRITTEN BY Max Wikström

Articles

read all »

Application Name for SQL Server Connections

All SQL Server SqlConnection Properties

When to use the SQL Native Client

Network Protocol for SQL Server Connection

Download SQL Server Native Client

SQL Server 2000 Data Types Reference

Q & A

ask question »

Azure SQL DB connection string is not working unless logged in to azure account

Making Constant Data Bases to recall in program.

Connection error IIS10/SQL 2016

SSIS package with ODBC source Teradata Deployment issue

Connect to SQl with gMSA