**What’s new in SQL Server 2008?**

[**New Date Datatypes in SQL Server 2008**](http://blogs.techrepublic.com.com/datacenter/?p=443)

As a database architect, one of the features I am most excited about in SQL Server 2008 is the new date related data types. These data types allow for the storage of time-only data, date-only data, a larger range of date values, more time precision, and the ability to store time-zone specific dates. Today I’ll take a look at these great new data types and how they can be used to expand your date storage possibilities.

**DATE**

The new DATE data-type defines a date field without any associated time. In previous versions of SQL Server, storing time-only data was accomplished in one of two ways. First, you could define either a DATETIME or SMALLDATETIME data type and assign the value as a VARCHAR value. Then you would need to handle the time related data in your queries. The other way is to store the date value in a VARCHAR field in the database. Either way, handling date-only data was at best cumbersome. With the new DATE data-type, it intrinsically stores only date values. In the following example I define a DATE data-type and a DATETIME data type and assign today’s date to the variables. As you can see from SELECTing the values from the variables, the DATE variable holds only the date portion whereas the DATETIME field also stores the associated time from the GETDATE() function.

DECLARE @TodaysDate DATE = GETDATE()

DECLARE @TodaysDateTime DATETIME = GETDATE()

SELECT @TodaysDate, @TodaysDateTime

Storing DATE specific data will also make queries easier to write. Instead of using one of the following statements to pull data for a DATETIME column;

SELECT \* FROM SalesHistory

WHERE SaleDate >= ‘4/11/2008' AND SaleDate < ‘4/12/2008'

you can use an equality operator (=) to search only for the specific date, assuming the SaleDateOnly field is defined as a DATE data-type.

SELECT \* FROM SalesHistory

WHERE SaleDateOnly = ‘4/11/2008'

**DATETIME2**

This data type is very simiilar to the datetime data type present in previous versions of SQL Server. The difference is the time precision and the range of dates that DATETIME2 can store. The new DATETIME2 data type can hold a range of dates from 0001-01-01 through 9999-12-31 while DATETIME holds the date range from 1753-01-01 to 9999-12-31. Another benefit of this data type is the precision in which it records time. DATETIME2 is accurate to within 100 nanoseconds whereas DATETIME is accurate to within 3/1000 of a second. The following script compares the values that the DATETIME and the DATETIME2 data-types can hold.

DECLARE @DateTime2 DATETIME2 = GETDATE()

DECLARE @DateTime DATETIME = GETDATE()

SELECT @DateTime, @DateTime2

Notice the additional precision stored in the DATETIME2 variable. This additional data can be of major significance for those applications where time related information is mission-critical.

**TIME**

The TIME data-type stores only the time portion of a date. This is significant in that the time data can be stored seperately and not associated with a specific date. The TIME data-type, like the DATETIME2 data-type is accurate up to 100 nanoseconds. The following script compares the time specific information returned by the TIME data type and that which is returned from a DATETIME data type.

DECLARE @Time TIME = GETDATE()

DECLARE @DateTime DATETIME = GETDATE()

SELECT @Time, @DateTime

**DATETIMEOFFSET**

The new DATETIMEOFFSET data type combines the range and precision of the DATETIME2 data-type along with time-zone awareness based on a 24 hour clock. The time-zone awareness is accomplished by adding or subtracting an hours + minutes value to the date data. In the following example, I define a temporary table that holds two fields. The first field holds the date that is local to my area. The second field, BeijingTime, is an offset date that will hold the time in Beijing, China; which happens to be 12 hours ahead of the time in Louisville, KY.

CREATE TABLE #OffSets

(

LouisvilleTime DATETIME2 DEFAULT(GETDATE()),

BeijingTime AS CONVERT(VARCHAR(50), LouisvilleTime, 121) + '+12:00'

)

INSERT INTO #OffSets(LouisvilleTime)

SELECT GETDATE()

SELECT \* FROM #OffSets

As you can see from the results from the table, the BeijingTime field holds the date from the LouisvilleTime field along with a 12 hour offset to indicate Beijing time. This data-type greatly enhances the ability to create time-zone aware applications.

**TIME to try the new data-types**

The new date related data-types included in SQL Server 2008 greatly expand the database developers ability to not only store a wider range of dates and precision for those dates, but also the ability to zone time-zone specific data which can lead to more robust and accurate reporting of time-related data around the world.

[**More New SQL Server 2008 Features**](http://blogs.techrepublic.com.com/datacenter/?p=437)

**Filestream Storage**

As the name may suggest, Filestream enables the storage of unstructured data such as documents and videos.  This feature integrates with the NTFS files system by using the nvarchar(max) data type to store data on the file system.  This feature is great for when you’re currently storing documents inside of your SQL Server system that are larger than 1 MB and fast read access is important.

**Sparse Columns**

Sparse columns are very similar to other types of database columns, except that they are specialized and optimized for null values.  If you have a table that holds a lot of NULL values, first consider your table design.  If your design is solid or cannot be altered, then your table may be a candidate to take advantage of sparse columns.  The rule from Microsoft is to consider using Sparse columns when 20-40 percent of the values in a column will be NULL.

**New Date & Time Data Types**

Prior to SQL Server 2008, there was no way to native data type to store time related data.  Time data was only available as part of the datetime (or smalldatetime) data types.  With SQL Server 2008, TIME is a separate data type, able to hold time values such as ‘23:59:59.9999999′.  Along with the new TIME data type, there are additional data types such as datetime2, which holds additional nanosecond data and the datetimeoffset data type, which can hold datetime data that is timezone aware.

**Spatial Data Types**

Two new data types are included in SQL Server 2008; geometry and geography.  These data types hold information regarding the physical location and shape of geometric objects, which allow for applications to be built that are geographically aware.

**Dependency Reporting**

Reporting dependencies has never been consistent or reliable in SQL Server.  The difficultly is that when objects are added, modified or dropped, special actions must be taken by the database engine to ensure that the dependency chain is correct.  In 2008, new dynamic management views are included keep track of these dependencies so that reliable reporting can be done one these objects.

**Advanced Auditing Features**

SQL Server 2008 includes several new features that facilitate the auditing of data.  These features include a new Auditing feature that allows you to easily create customized audits of database engine events.  Change Data Capture, while not auditing specific, makes data changes typically used for loading a data warehouse easily available in an easily used format.  SQL Server 2008 also includes a Data Collector feature that allows the DBA to gather and compare data that is gathered from several different sources.

**Policy Based Administration**

This is one of my favorite features of SQL Server 2008.  This feature allows a database administrator (DBA) to easily and uniformly set policies across servers that ensure system rules are met.  These policies can be rules such as simplifying administration tasks, preventing unauthorized system changes, making sure code compliance is met, ensuring best practices are met.

**TSQL Intelli-sense**

SQL Server Management Studio 2008 includes Intelli-sense which is a familiar feature in other Visual Studio products.  This feature presents a popup box when typing SQL statements to give options to choose columns from tables that are involved in your queries.

**Central Management Servers**

This feature allows the database administrator (DBA) to administer multiple SQL Servers easily and efficiently.  These servers allow for multi-server queries and for policy based management policies to be executed against multiple servers at the same time.

**Compound Operators**

This form of syntactic sugar is present in many other programming languages such as C++ or C#.  Consider the following TSQL statement.

SET @x = @x + 1

Using compound operators, the above statement can be rewritten in the following manner:

SET @x += 1

This statement, while not groundbreaking, takes TSQL a step forward in the evolution of programming languages.

**Transparent Encryption**

As data becomes more and more sensitive, the ability to keep the data secret becomes more important.  SQL Server 2008 includes the ability to encrypt data stored in the database transparently, which keeps you from having to alter applications to take advantage of the feature.  This feature prevents unauthorized users from reviewing the data even if they have the backup files.

[**Four cool new features slated for SQL Server 2008**](http://blogs.techrepublic.com.com/datacenter/?p=148)

[SQL Server 2008](http://www.microsoft.com/sql/prodinfo/futureversion/default.mspx) (code named Katmai) is scheduled to launch in February of next year, and the code isn’t slated to be released to manufacturing until Q2 2008. But database gurus don’t have to wait to get a sense of what’s to come because Microsoft recently released SQL Server 2008 to the community for preview.

Some of the goals of the Katmai [Community Technology Preview](https://connect.microsoft.com/SQLServer/content/content.aspx?ContentID=5395) (CTP) 1 release and future CTP releases include easing the manageability of the database, more security availability, more high availability options, and scalability. These types of overall enhancements are typically included in every version of SQL Server, which is why this article skips over such high-level features.

This is an overview of four features in SQL Server 2008 that I think are neat and will be important to DBAs. (Of course, there’s no guarantee that Microsoft will include these features in the final release.) For a comprehensive feature list, go to Microsoft’s [SQL Server 2008 Product Overview page](http://www.microsoft.com/sql/techinfo/whitepapers/sql2008Overview.mspx).

**Table variable parameters**

For a long time, I have wished that developers could pass a table variable as a parameter to stored procedures. With the advent of SQL Server 2005, [XML variables can be passed into procedures and parsed out with XQUERY](http://articles.techrepublic.com.com/5100-9592_11-6140404.html) inside the procedure to form a table. However, with the ability to pass a table variable in SQL Server 2008, developers will not have to be fully XQuery literate to pass a table of data into or out from a stored procedure. In my opinion, this will be one of the more integral tools for the developer in SQL Server 2008.

**MERGE statement**

The idea behind the MERGE statement is that the developer can construct TSQL [data-manipulation language](http://articles.techrepublic.com.com/5100-9592_11-6074046.html) (DML) statements in which INSERT, UPDATE, or DELETE can occur in the same statement, based on different search conditions. I think this idea is very cool. The ability to complete multiple statements within one statement could potentially lead to less coding and increased performance.

In addition to this statement, another great feature has been added to the INSERT statement. In SQL Server 2008, the developer can issue multiple rows to be inserted without using a SELECT statement as the INSERT statement source. Instead, the VALUE clause of the INSERT statement can be used to specify sets of values separated by parentheses and commas.

**Declarative Management Framework**

The Declarative Management Framework gives DBAs a central location to manage database server policies for server management, object management, and security. I presume that some type application will be deployed with client tools that use this framework to manage your database server policies. Because it is a framework, you should be able to develop your own database policies through custom .NET code.

The potential for this type of tool is huge. It could possibly save DBAs a huge amount of time administering enterprise databases.

**Change Data Capture**

Change Data Capture makes database auditing easier to create and maintain. While DML auditing has been available via triggers in SQL Server for a few versions, and DDL auditing has been available since SQL Server 2005, the solutions had to be custom written by a developer with reasonably intimate details of the database’s underpinnings. With Change Data Capture, system stored procedures are used to mark which types of objects you want to audit, and the stored procedures take care of how the auditing occurs.

While setting up these auditing procedures is typically a one-time deal, business requirements change over time; it is usually easier to use system stored procedures to handle changes, plus it’s less error prone than changing your custom auditing code. Unfortunately, when you use Change Data Capture, you will lose the auditing flexibility that you currently have with your own auditing solution.

**Here are 10 new features:**

10.  **Plug-in model for SSMS (SQL Server Management Studio)**.   SSMS 2005 also had a plug-in model, but it was not published, so the few developers that braved that environment were flying blind.  Apparently for 2008, the plug-in model will be published and a thousand add-ins will bloom.

9.  **Inline variable assignment.**  I often wondered why, as a language, SQL languishes behind the times.  I mean, it has barely any modern syntactic sugar.  Well, in this version, they are at least scratching the the tip of the iceberg.

Instead of:

DECLARE @myVar int

SET @myVar = 5

you can do it in one line:

DECLARE @myVar int = 5

Sweet.

8.  **C like math syntax**.  SET @i += 5.  Enough said.  They finally let a C# developer on the SQL team.

7.  **Auditing.**  It's a 10 dollar word for storing changes to your data for later review, debugging or in response to regulatory laws.  It's a thankless and a mundane task and no one is ever excited by the prospect of writing triggers to handle it.  SQL Server 2008 introduces automatic auditing, so we can now check one thing off our to do list.

6.  **Compression**.  You may think that this feature is a waste of time, but it's not what it sounds like.  The release will offer row-level and page-level compression.  The compression mostly takes place on the metadata.  For instance, page compression will store common data for affected rows in a single place.

The metadata storage for variable length fields is going to be completely crazy: they are pushing things into bits (instead of bytes).  For instance, length of the varchar will be stored in 3 bits.

Anyway, I don't really care about space savings - storage is cheap.  What I do care about is that the feature promised (key word here "promises") to reduce I/O and RAM utilization, while increasing CPU utilization.  Every single performance problem I ever dealt with had to do with I/O overloading.  Will see how this plays out.  I am skeptical until I see some real world production benchmarks.

5.  **Filtered Indexes.**  This is another feature that sounds great - will have to see how it plays out.  Anyway, it allows you to create an index while specifying what rows are not to be in the index.  For example, index all rows where Status != null.  Theoretically, it'll get rid of all the dead weight in the index, allowing for faster queries.

4.  **Resource governor.**  All I can say is FINALLY.  Sybase has had it since version 12 (that's last millennium, people).  Basically it allows the DBA to specify how much resources (e.g. CPU/RAM) each user is entitled to.  At the very least, it'll prevent people, with sparse SQL knowledge from shooting off a query with a Cartesian product and bringing down the box.

Actually Sybase is still ahead of MS on this feature.  Its ASE server allows you to prioritize one user over another - a feature that I found immensely useful.

3.  **Plan freezing**.  This is a solution to my personal pet peeve. Sometimes SQL Server decides to change its plan on you (in response to data changes, etc...).  If you've achieved your optimal query plan, now you can stick with it.  Yeah, I know, hints are evil, but there are situations when you want to take a hammer to SQL Server - well, this is the chill pill.

2.  **Processing of delimited strings.**   This is awesome and I could have used this feature...well, always.  Currently, we pass in delimited strings in the following manner:

exec sp\_MySproc 'murphy,35;galen,31;samuels,27;colton,42'

Then the stored proc needs to parse the string into a usable form - a mindless task.

In 2008, Microsoft introduced Table Value Parameters (TVP).

CREATE TYPE PeepsType AS TABLE (Name varchar(20), Age int)

DECLARE @myPeeps PeepsType

INSERT @myPeeps SELECT 'murphy', 35

INSERT @myPeeps SELECT 'galen', 31

INSERT @myPeeps SELECT 'samuels', 27

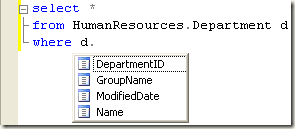
INSERT @myPeeps SELECT 'colton', 42

exec sp\_MySproc2 @myPeeps

And the sproc would look like this:

CREATE PROCEDURE sp\_MySproc2(@myPeeps PeepsType READONLY) ...

The advantage here is that you can treat the Table Type as a regular table, use it in joins, etc.  Say goodbye to all those string parsing routines.

1. **Intellisense** in the SQL Server Management Studio (SSMS).  This has been previously possible in SQL Server 2000 and 2005 with use of 3rd party add-ins like SQL Prompt ($195).  But these tools are a horrible hack at best (e.g. they hook into the editor window and try to interpret what the application is doing).

Built-in intellisense is huge - it means new people can easily learn the database schema as they go.