QUEST SOFTWARE®

Worth the Wait

Why SQL Server 2008 is Great

written by
Kevin Kline,
SQL Server MVP,
Quest Software, Inc.

© Copyright Quest® Software, Inc. 2008. All rights reserved.

This guide contains proprietary information, which is protected by copyright. The software described in this guide is furnished under a software license or nondisclosure agreement. This software may be used or copied only in accordance with the terms of the applicable agreement. No part of this guide may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording for any purpose other than the purchaser's personal use without the written permission of Quest Software, Inc.

WARRANTY

The information contained in this document is subject to change without notice. Quest Software makes no warranty of any kind with respect to this information. QUEST SOFTWARE SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTY OF THE MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Quest Software shall not be liable for any direct, indirect, incidental, consequential, or other damage alleged in connection with the furnishing or use of this information.

TRADEMARKS

All trademarks and registered trademarks used in this guide are property of their respective owners.

World Headquarters
5 Polaris Way
Aliso Viejo, CA 92656
www.quest.com
e-mail: info@quest.com

Please refer to our Web site for regional and international office information.

Updated—January 16, 2008

CONTENTS

INTRODUCTION

SQL Server 2008, scheduled for release in the first half of 2008, is intended to provide a comprehensive data platform for both structured and unstructured data that is more secure, reliable, manageable, and scalable than previous releases. In addition, SQL Server 2008 pushes the envelope with powerful new features for developers, enabling them to create new applications that store and consume all sorts of data on everything from tiny mobile devices to corporate "big iron" servers. SQL Server 2008 also provides a number of new features to turn all of that data into actionable information that enables your users to make informed decisions with new levels of insight and prescience.

SQL Server 2008, like SQL Server 2005, provides a plethora of new features and capabilities, as well as the usual bevy of bug fixes and performance improvements. Microsoft, however, has identified three areas where they're focusing their improvements:

- **Trustworthiness of the platform, its data, and its users**: provide the security, reliability, and scalability that are required by the most demanding business applications
- **Productivity of database administrators and developers**: offer an array of new data-driven management solutions that reduce the time and cost of managing and developing applications
- **Business intelligence**: provide a comprehensive platform for delivering and consuming intelligence when and where your users want it

That's how Microsoft has divided up the main elements of their marketing thrust for SQL Server 2008. However, as a working DBA and developer, my mind is not on the marketing as much as it is on the particulars within SQL Server 2008 that will make my life easier and solve my particular points of pain. With that in mind, this white paper will introduce you to the top ten features and capabilities in SQL Server 2008 that I find to be the most exciting and valuable:

10. SQL Improvements
9. Developer Improvements
8. Reporting Services (RS) Improvements
7. Compression
6. Data Types and Data Storage
5. Security
4. Resource Governor
3. Auditing and Change Tracking
2. Analysis Services (AS) Performance Improvements
1. Multi-Server Management.

10. SQL IMPROVEMENTS

I just can't help it that SQL is so important for me. I am, after all, the author of O'Reilly's *SQL in a Nutshell*, one of the industry's leading reference manuals on SQL. However, if you're a full time DBA or developer, you also need to know SQL well. Two new elements of SQL included in SQL Server 2008 that are most exciting to me are the GROUPING SETS operator and the MERGE statement. Let's talk about each of these new features in more detail.

GROUPING SETS

The GROUPING SETS operator is an extension to the ANSI SQL 2006 standard of the GROUP BY clause that lets you define multiple groupings in the same query. Several other database platforms already support grouping sets. GROUPING SETS enables aggregated groups on several different sets of grouping columns within the same query. This is especially useful when you want to return only a portion of an aggregated result set. GROUPING SETS also lets you select which grouping columns to compare.

The GROUPING SETS operator is best explained by example. Consider the following clauses and the result sets they return:

GROUP BY **CLAUSE**	**RESULT SETS**
`GROUP BY (col_A, col_B, col_C)`	`(col_A, col_B, col_C)`

`GROUP BY GROUPING SETS`
`( (col_A, col_B),`
`(col_A, col_C), (col_C) )`
	`(col_A, col_B)`
	`(col_A, col_C)`
	`(col_C)`

The GROUP BY GROUPING SETS clause lets you aggregate on more than one group in a single query. For each group set, the query returns subtotals with the grouping column marked as NULL. If you've ever used the CUBE and ROLLUP operators, you know they place predefined subtotals into the result set; the GROUPING SETS operator, on the other hand, allows you to control what subtotals to add to the query. The GROUPING SETS operator does not return a grand total.

Here's a query where we chose to subtotal sales data by *year* and *quarter* and separately by *year*:

```
SELECT order_year AS year, order_quarter AS quarter, COUNT (*) AS orders
FROM order_details
WHERE order_year IN (2003, 2004)
GROUP BY GROUPING SETS ( (order_year, order_quarter), (order_year) )
ORDER BY order_year, order_quarter;
```

The results are as follows:

```
year quarter orders
---- ------- ------
2003 NULL    380     -- the total for year 2003
2003 1       87
2003 2       77
2003 3       91
2003 4       125
2004 NULL    268     -- the total for year 2004
2004 1       139
2004 2       119
2004 3       10
```

Another way to think of GROUPING SETS is to consider it to be like a UNION ALL of more than one GROUP BY query that references different parts of the same data. You can tell the database to add subtotals to a grouping set by simply adding in the ROLLUP or CUBE clause according to how you would like subtotaling to occur.

GROUPING SETS can also be concatenated to concisely generate large combinations of groupings. Concatenated GROUPING SETS yield the cross-product of groupings from each of the sets within a GROUPING SET list. Concatenated GROUPING SETS are compatible with CUBE and ROLLUP. Concatenated GROUPING SETS, since they perform a cross-product of all GROUPING SETS, will generate a very large number of final groupings from even a small number of concatenated groupings. For example, if we extend the earlier table to include concatenated GROUPING SETS, we get the following:

GROUP BY **CLAUSE**	**RESULT SETS**
`GROUP BY (col_A, col_B, col_C)`	`(col_A, col_B, col_C)`
…	…

`GROUP GY GROUPING SETS (col_A, col_B) (col_Y, col_Z)`	`(col_A, col_Y)`
	`(col_A, col_Z)`
	`(col_B, col_Y)`
	`(col_B, col_Z)`

Note that the concatenated GROUPING SETS operator yields a large number of final groupings. You can image how large the result set would be if the concatenated GROUPING SETS contained a large number of groupings! However, the information returned can be very valuable and hard to reproduce without using a stored procedure or other procedural code.

MERGE Statement

The MERGE statement is sometimes called "UPSERT" because it combines the action of multiple DML statements: if the data doesn't exist, then INSERT; if the data does exist, then UPDATE. Because the MERGE statement can combine multiple actions into a single block of IO, you can avoid lots of IO especially in data warehousing situations where lots of data manipulation occurs.

When writing a MERGE statement, you specify the source record set using a SELECT statement and identify the type of data manipulation that will occur by pointing to a specific target table. For example, suppose that we have a nightly process that moves old sales order details from the *order_details* table into the *order_archives* table. Therefore, we want to insert any new *order_detail* records we encounter, and update any existing records with new values if they have changed. With the MERGE statement, you can accomplish this with a single statement rather than with many DML statements, as follows:

```
MERGE Order_Archive AS OA
USING
   (SELECT order_ID,  -- The initial query to find records
    load_date = MIN(CONVERT(VARCHAR(8), GETDATE(), 112)),
    order_total = SUM(order_amount),
    order_count = COUNT(*)
    FROM order_details
    WHERE order_year <= 2008
    GROUP BY order_ID)
AS archive_cte (order_ID, load_date, order_total, order_count)
ON (oa.CustomerID = archive_cte.CustomerID
    AND oa.SalesDate = archive_cte.LoadDate)

WHEN NOT MATCHED THEN
-- The INSERT statement used when no match is found
INSERT (order_ID, order_date, order_amount, order_count,
   create_date, update_date)
VALUES( archive_cte.order_ID, archive_cte.load_date, archive_cte.order_total,
   archive_cte.order_count, GETDATE(), GETDATE())

WHEN MATCHED THEN
-- The UPDATE statement used when a match is found
UPDATE
SET oa.order_amount    = oa.order_amount + archive_cte.order_amount,
    oa.TotalSalesCount = oa.order_count + archive_cte.order_count,
    oa.update_date = GETDATE();
```

Notice that the MERGE statement is broken in to three relatively simple sections:

- The clause *MERGE table_name* specifies the table whose data will be manipulated (the "target" table)
- The *USING* clause specifies the table where the data comes from (the "source" table)
- The ON clause specifies the join conditions between the source and target tables

The WHEN NOT MATCHED clause tells SQL Server the action to perform when no matching record is found in the target table (insert a new record), and the WHEN MATCHED clause specifies what to do if a match is found (update existing values in the *order_archive* table). Note that I could have told SQL Server to DELETE the record instead of updating the values in the record.

As you can tell, this operation can save SQL Server many trips back and forth to cache (or even to disk if the operation is very large) compared to a stored procedure or Transact-SQL batch that performs each separate DML statement serially. This can be especially valuable for data warehouse ETL operations involving millions of records.

9. DEVELOPER IMPROVEMENTS

It's no secret that good developers are always trying to crank out useful code more quickly, whether by making procedural changes to the development cycle, by adopting techniques like SCRUM and Agile, or by coding in more powerful and flexible languages.

However, for many developers the database server represents a sort of intellectual 'firewall' (not in the computer security sense, but in the fire-fighting sense) that is hard to break through. That's because in most IT environments, no matter how good you are at a .NET language like C# or VB.Net, you still have to code in SQL and Transact-SQL. (The Common Language Runtime, which can alleviate the need to code exclusively in Transact-SQL, is not universally adopted in the marketplace.) Well, what if Microsoft could make that coding easier and faster for you?

Microsoft intends to do just that with a couple of new features: Language Integrated Query (LINQ) and table-valued parameters (TVPs). Note that these features might be more tightly associated with Visual Studio 2008, but they are best exploited against SQL Server 2008. Let's talk about them in more detail.

Language Integrated Query (LINQ)

LINQ allows developers to query SQL Server databases using a programming language like C# or VB.NET instead of SQL statements. (LINQ queries still look very much like SQL to me.) This new feature means you'll get seamless, strongly-typed, set-oriented queries using your favorite .NET language, whether you are running against ADO.Net (using LINQ to SQL), ADO.Net DataSets (using LINQ to DataSets), or ADO.NET Entity Framework (using LINQ to Entities), or via the Entity Data Service Mapping provider. When you use the LINQ to SQL provider, you use LINQ directly on SQL Server 2008 tables and columns.

For example, here are two versions of a standard type-safe LINQ to SQL VB query that retrieves data from a Sales database. The first displays the query in an ASP.NET GridView control:

```
Dim Sales As New SalesDataContext

Dim query = From od IN Sales.Order_Details _
            Where od.category = 'A' _
              AND od.price > 15 _
            Order By od.vendor_ID _
            Select od

GridView1.DataSource = query
GridView1.DataBind()
```

The second does something similar using the LINQ DynamicQuery library:

```
Dim Sales As New SalesDataContext

Dim query = Sales.Order_Details _
            .Where("od.category='A' AND od.price>15") _
            .OrderBy("od.vendor_ID")

GridView1.DataSource = query
GridView1.DataBind()
```

Table-Valued Parameters (TVPs)

While I'm not sure how widely LINQ will be used after the release of SQL Server 2008, I'm quite sure that table-valued parameters will be widely and happily implemented by developers everywhere. Why? Simply that it is one of the most widely requested features by among those tracked by Microsoft.

Table-valued parameters, something very much like an array, solve many challenges for developers, including the following:

- Cleanly encapsulating tabular data that will be exchanged by the client application and the server
- Defining stored procedures with very large numbers of parameters
- Pivoting the scalar parameters of a stored procedure data into rows
- Using scalar parameters as an array in the code and repeatedly acting upon that data as a "row"

TVPs not only solve problems like these, but they also make maintaining code much easier because you can much more easily add new "columns" to the logical "table" created by the TVP. Here's what a simple TVP looks like using Transact-SQL:

```
-- TSQL to CREATE a TABLE TYPE tvp_order_line_details
CREATE TYPE tvp_order_line_details
AS TABLE (
  [order_ID]   int     NOT NULL,
  [product_ID] int     NOT NULL,
  [Qty]        int     NOT NULL)
```

Now, we'll create a stored procedure that uses the TVP we just created, along with another parameter for the customer who placed the order:

```
CREATE PROCEDURE Process_Order (
  @Customer_Id     int,
  @OrderItems      tvp_order_line_details READONLY)
AS
INSERT dbo.order_details
    SELECT o.order_id, o.qty
    FROM @OrderItems AS o
    INNER JOIN dbo.orders AS r ON o.order_id = r.order_id
    WHERE r.in_stock = 0
```

This is a very simple example, but it shows how TVPs make it easy for developers to handle logical “tables” and “rows” of data as parameters. This can be a big win, especially when you’re trying to perform business logic via set-based operations on very large batches of data. The resulting code will also, in my opinion, be cleaner and easier to maintain for both the client- and server-tiers. Application performance is improved as well, especially because you’re able to leverage the power of the SQL query processor in what would otherwise be row-based operations.

8. REPORTING SERVICES (RS) IMPROVEMENTS

SQL Server Reporting Services, which was one of the celebrity features of the later service packs of SQL Server 2000, has grown more mature with each release of the database platform. Its widespread popularity is attributable to its stellar performance and features coupled with zero cost (beyond the license cost for SQL Server).

Dropping the Internet Information Server (IIS) Requirement

One of the most exciting changes for me in SQL Server Reporting Services 2008 is that Internet Information Server (IIS) is no longer a requirement. I'd always found the configuration and installation of IIS to be a burden. Now I don't have to worry about configuring and installing IIS at all! Bloggers have reported other benefits to getting away from IIS, noting that rendering time has been improved and the overall memory load during rendering has been reduced.

Reporting Improvements

RS also has new features that make it useful in more situations and that increase its flexibility and availability to users:

- **Ad-hoc reporting**: Users who are not hard-core developers can quickly and easily build ad-hoc reports of almost any structure and design through the new Report Designer interface
- **Embedded reports**: Users are now able to directly embed their reports into Web portals, dashboards, and business applications. New rendering capabilities allow users to consume reports directly within Microsoft Office Word, as they already could do with Excel. Additional deep integration with Microsoft Office Sharepoint Server 2007 enables organizations to deliver reports to a central Web library or to render reports directly within a Sharepoint dashboard. Sharepoint integrated mode provides integrated shared storage, security, and document access accountability
- **Subscription-based deployment**: RS allows reports to be deployed via both standardized subscriptions (such as automatic delivery via e-mail to a group of users at a predefined time) and data-driven subscriptions, which retrieve subscription properties at run time so multiple users to get the same report, but tailored with the parameters and formatting required by each individual user

Performance Improvements

In addition, a variety of performance improvements have greatly improved the power and enterprise-readiness of SQL Server Reporting Services 2008. For example, you can now directly control server behavior with memory management, simplified configuration, on-demand processing, and instance-based rendering. Other significant improvements include:

- **Caching**: Reports can now be cached so that frequently accessed reports that use the same parameter values are available much more quickly. Once a report is processed, a copy is cached so that users who later request the same report can simply view the cached copy; the report is not processed again
- **Snapshots**: You can also request *snapshots* of a report: the report is processed on a schedule and made available for users to view, much like a cached report except that the snapshot is created on schedule, before any user requests it. This feature is especially useful for very complex or time-consuming rendering jobs

I've only touched on the high points of the new performance features in Reporting Services. However, one of the things that I like the most about the improvements in RS is that they simply *work*. You don't have to go in and recode your applications or make difficult changes to your configurations. A simple in-place upgrade will yield immediate and noticeable improvement in the performance of your reports.

7. COMPRESSION

There is an unmistakable trend in the industry for databases to get larger and larger. Companies not only record more information than ever before; they also retain backups longer—sometimes many years longer than they used to. Some progressive organizations are even thinking creatively about how to keep all data online, and seldom if ever taking the data offline for archiving. (That doesn't mean they're not taking backups. I only mean to say that some organizations keep their data live and in use in perpetuity, even while backing it up for safekeeping.)

Microsoft has introduced two new features with SQL Server 2008 that will help organizations more effectively drive their practice of recording more data and keeping it on hand longer: data compression and backup compression.

Data Compression

Data compression enables you to store more data using less disk space. Moreover, data compression improves performance for large I/O-bound workloads like data warehousing, because fewer disk reads and writes are needed when data is compressed. This feature will have a performance impact, so it should be used in selective opportunities like a read-only database.

Data compression works via two strategies: ROW compression and PAGE compression:

- **ROW compression** occurs when SQL Server uses a technique called variable-length storage formatting to reduce the amount of space consumed by the column values in a row of data
- **PAGE compression** works by storing commonly duplicated row values once on the page and then creating pointers the original value from the multiple columns

You use simple DDL commands to enable data compression, as shown the following example:

```
CREATE TABLE my_table1 (c1 INT, c2 CHAR(100) )
WITH (DATA_COMPRESSION = ROW);
CREATE TABLE my_table2 (c1 INT, c2 CHAR(100) )
ON my_partition_scheme
WITH (DATA_COMPRESSION = PAGE ON PARTITIONS (1-4),
DATA_COMPRESSION = NONE ON PARTITION (5) );
ALTER INDEX my_ndx ON foo REBUILD
WITH (DATA_COMPRESSION=PAGE)
```

To estimate the savings data compression might yield, execute the system stored procedure *sp_estimate_data_compression_savings*.

Backup Compression

Backups can take up a lot of space, especially if you need to keep several on hand for a large database. Moreover, processing a large backup or restore can take a long time. There are many sophisticated third-party backup and recovery products available in the market today, but with the Enterprise Edition of SQL Server 2008, backups can be compressed, so less space is consumed on the disk. Interestingly, Microsoft has decided to implement this compression in-process. Of course, the trade-off is that more CPU is consumed during the backups and restores because the CPU must handle the compression algorithms, but the savings in time and disk space can be considerable.

6. DATA TYPES AND DATA STORAGE

Data Types

More and better data type inclusion is always a good feature for a database platform. New data types can offer more logical storage for database designers as well help applications better serve their business owners. SQL Server 2008 includes a number of new data types; some (such as DATE and TIME) have been a part of the ANSI SQL standard for a long time, but are only now making their way into SQL Server. Other data types are new to the industry overall and might not even have been seen as feasible for a relational database ten years ago; these include FILESTREAM and the spatial data types.

Consider these new data types for SQL Server 2008:

- **Date and time:** SQL Server 2008 introduces several new date and time data types that separate date and time types while providing larger data ranges or user-defined precision for the time values:
  - **DATE** stores only the date
  - **TIME** stores only the time
  - **DATETIMEOFFSET** stores datetime, but is time zone aware
  - **DATETIME2** is similar to the pre-existing DATETIME data type, but with larger fractional second precision and year ranges that exceed that of DATETIME
- **FILESTREAM:** The FILESTREAM data type stores very large binary data directly on an NTFS file system, while preserving database control and transactional consistency. For example, FILESTREAM would be just the data type to choose if you wanted to use a database to control a library of video files
- **Spatial data types:** Spatial data types enable your application to be aware of locations within space. The GEOGRAPHY data type uses latitude and longitude coordinates to implement round earth applications of the Earth's surface, while the GEOMETRY data type implements flat earth solutions, such as those associated with interior spaces and the planar surfaces that might go into an interior space. Spatial data types make it easy for application designers to integrate SQL Server applications with a GPS or mapping application, among many other examples
- **HIERARCHYID:** Depicting tree structures in a relational database has always been notoriously difficult. SQL Server 2008's new HIERARCHYID enables database applications to model tree structures more efficiently than ever before. Note that HIERARCHYID is a new system type implemented by a CLR UDT, rather than a data type. HIERARCHYID store values that represent nodes of a hierarchy tree structure and includes some useful built-in methods for managing hierarchy nodes

Some of the methods exposed by the UDT include:

- **GETROOT( )**, which returns the root or top-level of the hierarchy type
- **GETDESCENDANT(*child1*, *child2*)**, which returns a child ID between *child1* and *child2*
- **ISDESCENDANT(*child*)**, which returns true when *child* is >= the compared value
- **GETLEVEL( )**, which returns an integer representing the level of the tree or 0 for the root
- **GETANCESTOR(*n*)**, which returns the hierarchy ID representing the *n*th ancestor of the compared value

Data Storage

In addition to the data type additions, SQL Server 2008 includes some nice new capabilities around how it stores data. For example, in earlier versions, Full Text Search catalogs were external to the SQL Server database; SQL Server 2008's integrated Full Text Search makes search for text and relational data easy and seamless.

SQL Server 2008 has also improved the implementation of *sparse columns* (a sparse column is one that frequently stores a great deal of NULL values). In earlier versions of SQL Server, NULL data consumed space. Now, sparse columns consume no physical space and provide a highly efficient way to manage frequently empty columns in a table.

Finally, SQL Server 2008 eliminates the 8KB size limit for User-Defined Types (UDTs). Because UDTs can be dramatically enlarged, their flexibility and power are similarly increased.

5. SECURITY

A lot of people don't realize how secure SQL Server has become in the years following the SQL Slammer internet worm. Since then SQL Server has had only five critical security fixes—Oracle had over 100 in the same time period. That's rather reassuring for me. SQL Server 2008 continues this long run of excruciating attention to security with support for transparent data encryption and extensible key management.

Transparent Data Encryption (TDE)

Transparent data encryption is called "transparent" because the developer or application doesn't have to do anything to seamlessly protect the data. You can protect data not only from outside intruders or unauthorized former employees or consultants; you can also selectively protect data from a DBA who needs to see only part of your employee salary database. Transparent data encryption enables you to encrypt an entire database, data files, or log files, without changing your application. And even if someone were to obtain a backup of your encrypted data, that data would be encrypted and unreadable, which might be especially valuable to organizations who host their applications and backups through a service provider who shouldn't be able to read their data.

While SQL Server 2005 offered column-level and cell-level encryption, SQL Server 2008 offers the complementary feature of encrypting an entire database. The syntax is very easy:

```
ALTER DATABASE my_database_name SET ENCRYPTION [ON | AUTO | OFF]
```

Encryption works by using a special key, called the *database encryption key* (DEK), and it is tracked in the sys.dm_database_encryption_keys DMV.

Extensible Key Management

SQL Server 2005 provided the first comprehensive solution for encryption and key management. SQL Server 2008 builds on that solution by supporting third-party key management and hardware security module (HSM) vendors to register their devices in SQL Server 2008 Enterprise Edition, as well as in the Developer and Evaluation editions. SQL Server 2008's extensible key management leverages the Microsoft Cryptographic API (MSCAPI) provider for encryption and key generation. Encryption keys for data are stored in transient key containers (and can be stored in an HSM such as a card key, adding an extra layer of security by separating the key from the software). They must be exported by the provider before they can be stored in the database, thereby enabling key management that includes both an encryption key hierarchy and key backup to be handled by SQL Server.

4. RESOURCE GOVERNOR

The resource governor is a feature of SQL Server 2008 that many DBAs have coveted for years. With the resource governor, you can specify the resource limits and priorities of the different workloads running on your SQL Server, thereby enabling concurrent workloads to use a consistent amount of system resources and, by extension, provide a consistent and predictable response to end users. And may I say just for the record that this is a really cool feature. (You can get the complete lowdown on the resource governor in the TechNet article *Introducing Resource Governor* at http://technet.microsoft.com/en-us/library/bb895232(SQL.100).aspx.)

The resource governor works by allocating resources to various working pools or groups. There is always a default resource pool. If there are no other workload groups defined for the SQL Server, then all activities both of users and internal process run within the context of the default resource pool. Separate from, but often synonymous with the default group, is the internal group. The internal group includes activities like the lazy writer, ghost record cleanup, and certain trace activities.

A bit of terminology is due. An *instance* can have one or more *resource pools* (but no more than 20). Beneath the resource pools, you can create one or more *groups*. There's no physical limit on the number of groups you can create, but I encourage a conservative approach. A group must be assigned to a single resource pool, but resource pools can have one or more groups.

You might create a workload group for quality assurance that looks like this:

```
CREATE WORKLOAD GROUP grp_qa
WITH (GROUP_MAX_REQUESTS = 25,
   IMPORTANCE = LOW,
   REQUEST_MAX_MEMORY_GRANT_PERCENT = 25,
   MAX_DOP = 1   )
```

Assume that we've also created other separate workload groups for the DBA (grp_sa), developers (grp_dev), and so forth. Now we need to assign the workload groups to users or applications connecting to the SQL Server instance through a function like this:

```
CREATE FUNCTION rg_classifier() RETURNS SYSNAME
WITH SCHEMABINDING
AS
BEGIN
    DECLARE @grp_name AS SYSNAME
      IF (SUSER_NAME() = 'sa')
          SET @grp_name = 'grp_sa'
      IF (SUSER_NAME() = 'qa')
          SET @grp_name = 'grp_qa'
      IF (APP_NAME() LIKE '%MANAGEMENT STUDIO%') OR (APP_NAME()
                    LIKE '%QUERY ANALYZER%')
          SET @grp_name = 'grp_adhoc'
      IF (APP_NAME() LIKE '%REPORT SERVER%')
          SET @grp_name = 'grp_rpt'
    RETURN @grp_name
END
```

This function then needs to be applied to the resource governor like this:

```
ALTER RESOURCE GOVERNORE WITH (CLASSIFIER_FUNCTION = dbo.rg_classifier);
```

The next step is to configure which system resources are assigned to each of the workload groups:

```
ALTER WORKLOAD GROUP grp_qa
WITH (REQUEST_MAX_CPU_TIME_SEC = 25);

ALTER WORKLOAD GROUP grp_adhoc
WITH (MAX_CPU_PERCENT = 50);
```

You must finally apply the changes to the resource governor process, which is active and running in memory:

```
ALTER RESOURCE GOVERNOR RECONFIGURE;
```

Note that you should be cautious when altering the resource governor because changes will take effect immediately. You can later change process to run with different resources at any time, immediately affecting all processes already running. However, once a session is bound to a group or pool, it remains bound until it is disconnected.

3. AUDITING AND CHANGE TRACKING

Several important new U.S. laws have a direct impact on how organizations store data and audit the activity against that data. In particular, medical companies must comply with HIPAA and be able to prove their compliance to external auditors. Similarly, publicly traded companies must comply with Sarbanes-Oxley (SOX). SQL Server 2008 provides features that make compliance, data access control, and change tracking much easier.

Audit Object

SQL Server 2008 supports the Audit object, which enables you to capture database activity and store it in a log file or Windows application or security logs. To enable auditing on your SQL Server, you must use the CREATE SERVER AUDIT syntax, which configures the Audit object and determines where its auditing data is written, as shown in the following example:

```
CREATE SERVER AUDIT sox_audit_file
   TO (FILEPATH='\\BOSTON\DATA\AUDIT\');

CREATE SERVER AUDIT sox_application_audit_file
   TO APPLICATION_LOG
   WITH ( QUEUE_DELAY = 300, ON_FAILURE = SHUTDOWN);
```

The *queue_delay* clause enables asynchronous auditing when data is written to a log file, which can boost performance. The *on_failure* clause enables you to specify what SQL Server should if it can no longer write to the audit log; in this example, SQL Server will shut down if auditing cannot continue.

If you specify that the audit data should be written to a file, you do not need to provide a name; SQL Server will provide one for you in the format *audit-name_audit-guid_nn_tstamp.sqlaudit,* where *audit-name* is the name chosen in the CREATE SERVER AUDIT statement, *audit-guid* is a unique identifier for the Audit object, *nn* is the partition number used to partition file sets, and *tstamp* is a timestamp value.

Once the server audit is created, you can add more database-level events to audit using the CREATE DATABASE AUDIT SPECIFICATION statement. For example, the following statement causes the audit we created earlier to record whenever a failed login attempt occurs:

```
CREATE SERVER AUDIT SPECIFICATION failed_login
FOR SERVER AUDIT sox_audit_file
   ADD (failed_login_group);
```

Similarly, if we wanted to know whenever a DDL operation occurs (such as CREATE, ALTER, or DROP statements), plus whenever any INSERT, UPDATE, or DELETE statements are executed against the GeneralLedger schema by the Manager or Admin users, we would specify the following:

```
CREATE DATABASE AUDIT SPECIFICATION gl_audit_specification
FOR SERVER AUDIT sox__application_audit_file
   ADD (DATABASE_OBJECT_CHANGE_GROUP) – captures create/alter/drop
   ADD (INSERT, UPDATE, DELETE        - captures DML by Manager and Admin
      ON Schema::GeneralLedger
      BY Manager, Admin);
```

DDL Triggers

DDL triggers first appeared in SQL Server 2005 and are in many ways similar to a regular INSERT, UPDATE, or DELETE trigger. The primary difference is that the DDL triggers fire after a triggering event such as an ALTER USER statement or a DROP TABLE statement at the database level, or ALTER DATABASE or CREATE LOGIN at the server level. Think of DDL triggers as triggers for the CREATE, ALTER, and DROP statements. SQL Server provides a shorthand with groups. Thus, you could write one trigger for DDL_TABLE_EVENTS to handle CREATE_TABLE, ALTER_TABLE, and DROP TABLE statements. For full details, refer to *SQL Server 2008 Books Online* at http://msdn2.microsoft.com/en-us/library/bb543165(sql.100).aspx.

You can use DDL triggers to prevent certain DDL operations from occurring, or to restrict the activities of particular users. But you can also use this feature to capture and audit DDL activity within a database by writing data to an audit table in the database or server, or to an audit file on the file system.

2. ANALYSIS SERVICES (AS) PERFORMANCE IMPROVEMENTS

SQL Server 2008 Analysis Services have gotten a lot faster, so if you are one of the many users who make heavy use of Analysis Services, you're going to be very happy with SQL Server 2008. The following are a few of the features that improve the performance of Analysis Services:

- **Block computations**: Block computations provide big performance gains by eliminating unnecessary aggregation calculations, such as NULL values in an aggregation. (It's called "block mode" because it evaluates data by blocks rather than the previous "cell-by-cell mode" of early versions of AS.) Some bloggers have reported anecdotal testing results that show calculations being 50-60% faster than the same calculations on SQL Server 2005. Cube calculation times are likewise much faster, so users can increase the depth of their hierarchies, perform more complex calculations, or simply accomplish the same work as before more quickly.

  Since block mode computations are driven by proper MDX code, see the full list of MDC commands that work with or invalid block computations in the white paper *Performance Improvements for MDX in AS 2008*, available at http://msdn2.microsoft.com/en-us/library/bb934106(SQL.100).aspx.

- **Writeback:** Writeback is the ability to change the structure of a dimension even as you analyze it. Efficient writeback is critical to high-performance what-if scenario modeling. SQL Server 2008 includes a new MOLAP-enabled writeback feature that removes the need to query ROLAP partitions. The new writeback technique means that fewer round trips are needed between the client and the OLAP partition to complete the analytical modeling. That is, writeback occurs faster in SQL Server 2008, so that when changes are made to a model, they are then written back to the database, which in turn refreshes the cube in a more timely manner. Users get better and faster writeback scenarios without giving up any of the advantages of a traditional OLAP engine for aggregating data.

  Note that writeback has been in SQL Server Analysis Services for a while, but the improvements in SQL Server 2008 are drastic enough that you will not be able to use writebacks created using earlier versions of SQL Server with the 2008 release; you'll need to recreate all of your writebacks when you upgrade.

- **Star join query optimizations**: By default, the SQL Server query engine is very efficient at choosing a query plan, even when dealing with millions of rows of data for a single query. One technique SQL Server uses to improve star schema joins is through a *bitmap filter*, a miniature model of a set of values from a table in one part of the operator tree to filter rows out of a second table in another part of the tree. By making sure to join only to *qualified* records in the second table when analyzing records in the first table, the bitmap filter significantly reduces the amount of data that needs to be processed. SQL Server 2005 allowed the use of bitmap filtering after optimization, but SQL Server 2008 also allows a bitmap filter to be dynamically applied during query plan generation. When the bitmap filter is applied, non-qualifying rows in the fact table are immediately eliminated, so the query is enormously faster. Read more about the intricacies of this new feature in the TechNet article *Optimizing Data Warehouse Query Performance Through Bitmap Filtering* at http://technet.microsoft.com/en-us/library/bb522541(SQL.100).aspx.

  Other improvements to SQL Server 2008 Analysis Services provide enhanced analytical capabilities and support more complex computations and aggregations. And less prominent features offer useful incremental improvements in the overall performance of Analysis Services. Without a doubt, SQL Server 2008 Analysis Services will be the fastest and most powerful release yet.

1. MULTI-SERVER MANAGEMENT

Multi-server management is a topic near and dear to my heart because I struggled mightily with it for many years of my career. And I see a strong trend for DBAs and developers to be able to deploy large numbers of SQL servers throughout their IT environments and then to adequately support and maintain them. On the one hand, many shops still work under the policy of "one application, one server," which results in rampant and sometimes entirely unnecessary server proliferation. On the other hand, even in shops where server proliferation is tightly controlled, the enterprise often simply needs a lot of SQL servers but cannot afford a lot of DBAs to maintain them. Consequently, most enterprise SQL Server DBAs have to support a LOT of SQL servers in their shops.

With previous versions of SQL Server, I (and others like me) would automate as much work as possible through clever use of SQL Server Agent jobs and scripts. For example, one DBA I know gets new servers to administer so frequently that he can't keep up with getting them on backup rotations. So he built a script and a job to look for new servers in his shop and put all of the databases on any new server onto a backup schedule. SQL Server 2008 makes managing multiple servers much easier, in part through two improvements, streamlined installation and Policy-based Management.

Streamlined Installation

Microsoft completely overhauled the process of installation, setup, and configuration of SQL Server. Whereas earlier versions forced you to perform all three activities at once, SQL Server 2008 separates the installation of the physical bits on the hardware from the configuration of SQL Server. This means that a centralized IT group or a third-party provider can give sites a recommended installation configuration (contained in a file) without also having to install SQL Server. Once a local administrator is able to install SQL Server, the configuration file tells SQL Server how to set up and configure itself without user intervention.

Policy-Based Management (PBM)

You might have heard of Policy-based Management by its early CTP name, the Declarative Management Framework (DMF). It doesn't sound terribly exciting by either moniker, but it is. It's an entirely new way to manage SQL Server, both conceptually and operationally.

PBM is a policy-based system for managing one or more instances of SQL Server 2008. This is a very important shift because most DBAs organize their work around tasks, such as creating backups or performing consistency checks, not policies. PBM enables you to implement a "monitor by exceptions" style of working in which you need to get involved only when an exception to the rules of good operations occurs. For example, you can use PBM to create policies that manage entities on the server (such as the instance of SQL Server, databases, and other SQL Server objects), as well as the things you want to restrict (such as failed logins), the things you'd like to enforce (like naming standards), and the things you'd like to monitor (like blocked processes). And you can even scope your policies to both the entire server and a specific database. Wow!

PBM also allows some really neat automation. Anyone in the PolicyAdministratorRole can create and apply policies. Under PBM, you can run policies on demand or use one of these automated execution modes:

- Changes are attempted and, when out of compliance, are prevented. PBM uses DDL triggers to prevent policy violations
- **Changes are attempted and, when out of compliance, are allowed but logged**. PBM uses event notification to evaluate a policy when a relevant change occurs
- **On schedule, log out-of-compliance**. PBM uses a SQL Server Agent job to periodically evaluate a policy on all the servers where it's applied

You can create a policy using Management Studio. Once you've created a policy, you can export it to an XML file and then import it, individually or by group, into other SQL Servers.

Here are some examples of things you can configure in a policy:

- Impose the surface area configuration settings of one instance onto another
- Create and enforce a naming convention policy
- Enable, disable, or assign values using SP_CONFIGURE
- Set the service account to a privileged user
- Place a database in read-only mode
- Create a schedule to run DBCC statements

And because the policies are easy to automate and easy to apply to many servers at once, you can use PBM to control most all aspects of your SQL Server environment—as long as the servers are running SQL Server 2008.

CONCLUSION

SQL Server 2008 has so many new features and capabilities that I can't possibly list them all here. Instead, explained in detail the features that I'm most excited about. Take the time to get to know why these features are so useful and then explore on your own. You may find other new features in SQL Server 2008 that are even more exciting for your particular situations.

To review, my top ten favorite features in SQL Server 2008 are:

10. The SQL improvements offered by GROUPING SETS and the MERGE statement
9. The developer-centric improvements of LINQ and table-valued parameters
8. Reporting Services improvements: dropping the requirement for IIS; powerful new features for ad-hoc reporting, embedded reports, and subscription-based deployment; and performance improvements like improved caching and snapshot reports
7. The data and backup compression features in SQL Server 2008 Enterprise Edition
6. The new spatial, FILESTREAM, HIERARCHYID, and date and time data types
5. The security improvements of transparent data encryption and extensible key management
4. The oh-so-sweet resource governor
3. The audit and change tracking abilities of audit objects and DDL triggers
2. The big gains in Analysis Services performance with things like block computations and writeback improvements
1. The multi-server management capabilities of Policy-based Management and streamlined installation.

About the Author

Kevin Kline is the technical strategy manager for SQL Server solutions at Quest Software, a leading provider of award-winning tools for database management and application monitoring on the SQL Server platform. Kevin is the president of the international Professional Association for SQL Server (PASS; www.sqlpass.org). He has been a Microsoft SQL Server MVP since 2004. Kevin is the lead author of *SQL in a Nutshell* (http://www.oreilly.com/catalog/sqlnut2/), as well as a co-author of both *Professional SQL Server 2005 Database Design and Optimization* (http://www.apress.com/book/bookDisplay.html?bID=10005) and *Database Benchmarking* (http://www.rampant-books.com/book_2007_1_database_benchmarking.htm).

Kevin writes monthly columns for *Database Trends and Applications* and *SQL Server Magazine*. He also maintains blogs at SQLBlog.com and SQLMag.com. Kevin is a top-rated speaker, appearing at international conferences such as Microsoft TechEd, the PASS Community Summit, Microsoft IT Forum, DevTeach, and SQL Connections. When he's not pulling out his hair over work, he loves to spend time with his four kids and in his flower and vegetable gardens.

ABOUT QUEST SOFTWARE, INC.

Quest Software, Inc. delivers innovative products that help organizations get more performance and productivity from their applications, databases and Windows infrastructure. Through a deep expertise in IT operations and a continued focus on what works best, Quest helps more than 50,000 customers worldwide meet higher expectations for enterprise IT. Quest Software can be found in offices around the globe and at www.quest.com.

Contacting Quest Software

Phone:	949.754.8000 (United States and Canada)
Email:	info@quest.com
Mail:	Quest Software, Inc. World Headquarters 5 Polaris Way Aliso Viejo, CA 92656 USA
Web site	www.quest.com

Please refer to our Web site for regional and international office information.

Contacting Quest Support

Quest Support is available to customers who have a trial version of a Quest product or who have purchased a commercial version and have a valid maintenance contract. Quest Support provides around the clock coverage with SupportLink, our web self-service. Visit SupportLink at http://support.quest.com

From SupportLink, you can do the following:

- Quickly find thousands of solutions (Knowledgebase articles/documents).
- Download patches and upgrades.
- Seek help from a Support engineer.
- Log and update your case, and check its status.

View the ***Global Support Guide*** for a detailed explanation of support programs, online services, contact information, and policy and procedures. The guide is available at: http://support.quest.com/pdfs/Global Support Guide.pdf