At a glance:
- More powerful management
- Improved performance and scalability
- Better security and availability
- Changes for developers

SQL Server 2008: what's new?

Randy Dyess

Once again, Microsoft has released a new version of SQL Server that promises to improve life for database administrators. The latest version is SQL Server 2008, and it offers a large variety of new features and functionality that will make the administration a whole lot better.

Database administrators who have been using SQL Server 2005 for any length of time will find many of the same features that they use in their day-to-day jobs, but they'll also soon discover that these familiar tools have been enhanced in SQL Server 2008. The new functionality built on the existing features goes a long way in reducing the number of workarounds or customisations once needed for using various features in complex database environments.

New features in SQL Server 2008 involve a large range of database job roles, making it difficult to categorise them. As I classify features under different topics, I realise that some readers will be wondering why I placed Feature X under Category Y when it clearly belongs under Category Z. It's a matter of perspective, and it's affected by how your company does business.

I do understand that database people often find themselves doing many different job roles, but I will try to break down the new features so they fit into the following

This article is based on a prerelease version of SQL Server 2008. All information herein is subject to change.

generic categories: Management, Scalability, Performance, High Availability, Security, Development and Business Intelligence.

What's new for management?

For database administrators (like me), the additional management functionality makes SQL Server 2008 a very exciting new product. The new policy management, multiple server query capability, configuration servers and data collector/management warehouse offer powerful new abilities for database administrators who are often responsible for managing large and complex database environments with hundreds or thousands of databases on dozens or even hundreds of servers.

The SQL Server 2008 Policy Management feature, which was actually called Declarative Management Framework in the Community Technology Previews (CTPs), allows you to create and execute configuration policies against one or more database servers. With these policies, you can ensure that standard configuration settings are applied and maintained on each of the targeted servers and databases. You can see an example of this feature in **Figure 1**.

Policies are created from a predefined set of facets. Each facet contains a subgroup of SQL Server 2008 configuration settings and other events that you can control. You pair these facets with conditions in order to create a policy. Conditions are the values that are allowed for the properties of a facet, the configuration settings, or other events contained within that facet.

Conditions are also values used for policy filters. Say you want the policy to be executed only against a certain database. In this case, you can create a condition that contains the name of the database and then add this condition to the policy. Now the policy will only apply to that one database. Trust me on this – SQL Server 2008 Policy Management may sound complex, but once you try it you'll realise it's pretty intuitive.

The new Multiple Server Interaction and Configuration Servers capabilities come in handy when you need to execute queries against multiple servers at the same time. You can register servers in your Management Studio and then place those servers together under a grouping. When you need to execute a policy or query against all the servers in the grouping, you simply right-click on the grouping and do so.

As an added benefit, you can configure this feature to return one resultset per server or merge all the resultsets together into one big resultset. You can also specify whether you want the server and database names as part of the results so you can separate the individual results from each server. Being able to store the registered servers on the configuration server rather than in each individual's Management Studio is a big benefit.

Another nice new management feature is the Data Collector. Database administrators often need to collect management data from a large number of servers, and many of these DBAs have created their own custom solution for doing so. The Data Collector is a built-in mechanism that eases the task of collecting management-related data. It allows you to use the SQL Server Agent and SQL Server Integration Services (SSIS) to create a framework that collects and stores your data while providing error handling, auditing, and collection history.

Unlike third-party tools and custom jobs, the Data Collector will be easily understood by most database administrators since it uses SQL Server Agent and SSIS to create a set of jobs and packages to handle the connections, collection, and storage of data (as you can see in **Figure 2**). Once this data is stored in a central location, referred to as the Management Warehouse, it can be viewed and organised through a set of T-SQL statements and SQL Server 2008 Reporting Services reports. This

Policy Management allows you to create and execute configuration policies against one or more database servers

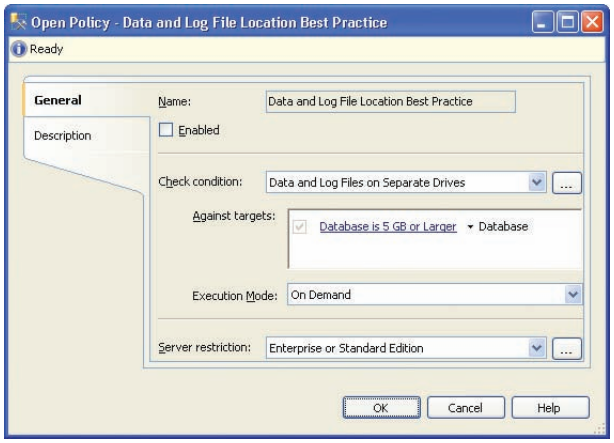

Figure 1 **The Data and Log File Location Best Practice policy**

central data store makes it much easier to analyse and view the overall management metrics of the database environment.

What's new for scalability?

Over the years, SQL Server database administrators have found their database environments becoming increasingly large. As the size of your database environment increases, you need new methods and tools to achieve the scalability that most enterprises require. SQL Server 2008 has introduced several new features that will help.

SQL Server 2008 has built-in compression that allows you to compress the database files and the transaction log files associated with the compressed database. SQL Server 2005 introduced the ability to compress data on a read-only file or filegroup, but this form of compression simply used the compression ability of Windows NTFS. With SQL Server 2008, you now get both row-level and page-level compression, offering benefits you don't get with compression at the data file level.

Compression at the row and page levels reduces the amount of data space needed, plus it reduces the amount of memory needed since the data remains compressed while in memory. Compressed data in memory results in increased memory utilisation, which benefits the scalability of many systems.

SQL Server 2008 also introduces compression at the backup level. While database backups only back up the active portion of the database, this still represents as many as hundreds of gigabytes or even dozens of terabytes. In database environments that have more than one copy of a multi-terabyte backup file, these backups often take up valuable storage space that could be used more effectively. By allowing database administrators to compress their backup files, SQL Server 2008 frees up some of this space, so it can be used for live data.

Then there's the Resource Governor. This new feature lets you define the amounts of resources that individual or groupings of workloads are allowed to use during execution. With Resource Governor, you can create an environment in which many different workloads coexist on one server without the fear of one or more of those workloads overwhelming the server and reducing the performance of the other workloads.

The benefit of this feature is that you can more effectively use the total amount of resources that are available on your database servers. **Figure 3** shows an example of using the Resource Governor to limit activity on a server.

What's new for performance?

The general performance of databases improves with SQL Server 2008. Thanks to several new features found in SQL Server 2008, you can control and monitor the performance of your databases and the applications that execute against them.

When you have large numbers of transactions performed every second, the locking that normally occurs during these transactions can have a negative impact on the performance of your database applications. SQL Server is designed to reduce the total number of locks a process holds by escalating

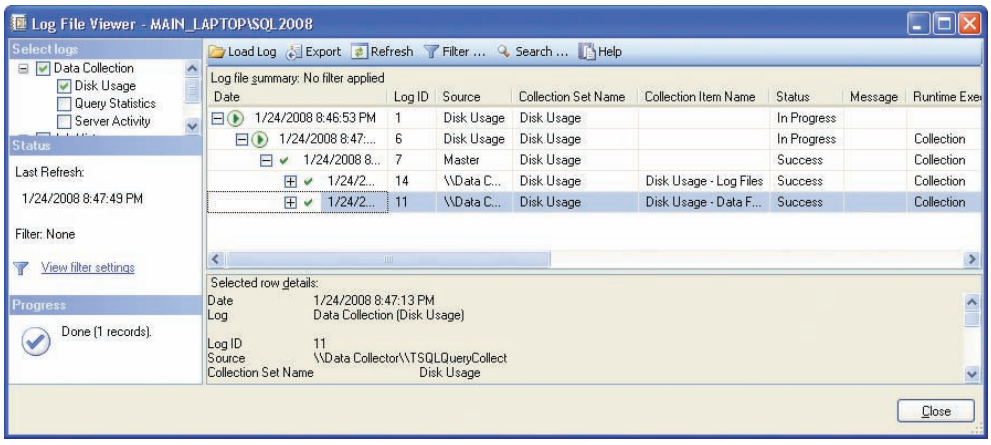

Figure 2 **The Data Collector Disk Usage Log file**

locks from the smaller row-level and page-level locks to large table-level locks. But it's important to understand that this escalation of locks can cause problems. For example, a single transaction can lock an entire table and prevent other transactions from working with that table.

SQL Server 2008 works with the table partitioning mechanism (which was introduced in SQL Server 2005) to allow the SQL Server engine to escalate locks to the partition level before the table level. This intermediary level of locking can dramatically reduce the effects of lock escalation on systems that have to process hundreds and thousands of transactions per second.

SQL Server 2008 offers several new query processor improvements for when the query interacts with partitioned tables. The query optimiser can now perform query seeks against partitions as it would against individual indexes by only working with the partition ID and not the partitioning mechanism at the table level.

What's new for high availability?

As database environments become more complex and databases grow in size, the ability to ensure the availability of those databases becomes increasingly difficult. The familiar mechanisms you have used in the past to achieve high availability are still present in SQL Server 2008. But some of these features have been enhanced in SQL Server 2008 and some new ones have been added.

With SQL Server 2005, many administrators started implementing database mirroring to achieve high availability. SQL Server 2008 offers many improvements for the practice of database mirroring. For instance, in the past, database mirroring occasionally had performance issues related to moving transaction log data from the principal to the mirrored databases. In response, SQL Server 2008 now reduces the amount of information that is moved across the network from the principal's transaction log to the mirror's transaction log by compressing the information before sending it to the mirror's transaction log for hardening.

You now have the ability to repair corrupted data pages on the principal. If a principal database suffers corrupt data pages due to errors 823 and 824, the principal can request a fresh copy of those data pages from the mirrored servers. This request of good data pages is an automated process that is transparent to any users who are currently accessing the principal databases.

Another new feature, Hot Add CPU, lets you add additional CPUs to a database server without affecting the availability of the da-

Figure 3 Limit activity with the Resource Governor

```
USE master
go

--Drop function
IF OBJECT_ID('rgclassifier_demo','Function') IS NOT NULL
DROP FUNCTION rgclassifier_demo
go

--Create a classifier function for report group
CREATE FUNCTION rgclassifier_demo() RETURNS SYSNAME
WITH SCHEMABINDING
AS
BEGIN
    DECLARE @group_name AS SYSNAME
  IF (USER_NAME() LIKE '%Launch_Demo%')
         SET @group_name = 'demogroup'
    RETURN @group_name
END
GO

--Drop workload group for anything coming from Management Studio
IF EXISTS (SELECT name FROM sys.resource_governor_workload_groups
  WHERE name = 'demogroup')
BEGIN
  DROP WORKLOAD GROUP demogroup
END
GO

--Create workload group
CREATE WORKLOAD GROUP demogroup
GO

--Register the classifier function with
--Resource Governor
ALTER RESOURCE GOVERNOR WITH (CLASSIFIER_FUNCTION= dbo.rgclassifier_demo)
GO

--Alter the dbogroup workload group to only
--allow 10% of CPU for each workload request
ALTER WORKLOAD GROUP demogroup
WITH (REQUEST_MAX_CPU_TIME_SEC = 10)
GO

--Create a new resource pool and set a maximum CPU limit for all workloads.
IF EXISTS (SELECT name FROM sys.resource_governor_resource_pools
  WHERE name = 'pooldemo')
DROP RESOURCE POOL pooldemo
GO

CREATE RESOURCE POOL pooldemo
WITH (MAX_CPU_PERCENT = 40)
GO

--Configure the workload group so it uses the
--new resource pool.
ALTER WORKLOAD GROUP demogroup
USING pooldemo
GO

--Apply the changes to the Resource Governor
--in-memory configuration.
ALTER RESOURCE GOVERNOR RECONFIGURE
GO
```

tabases residing on that server. However, you should know that Hot Add CPU does have some limitations, as it is only useful when running the 64-bit Itanium-based Windows Server 2008 Enterprise Edition or Datacenter Edition, and it requires the Enterprise Edition of SQL Server 2008.

What's new for security?

SQL Server 2005 introduced data security in the form of data encryption. With SQL Server 2008, encryption is much enhanced through two new features: Extensible Key Management and Transparent Data Encryption.

Extensible Key Management allows for an enhanced structure to safely store the keys used in the encryption infrastructure – not only in the database itself but also outside the database in third-party software modules or with a Hardware Security Module.

Transparent Data Encryption offers improved flexibility for encrypting data by allowing encryption to be a property of the database and not just the result of functions in a line of code. The result is that administrators do not have to perform the large number of changes that are required for their database structure and application code when they perform encryption at the data level. The code in **Figure 4** shows how you can encrypt a database with Transparent Data Encryption.

Figure 4 **Using Transparent Data Encryption**

```
USE master;
GO

--Create a master key
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'YouWillNeedToCreateAStrongPassword';
GO

--Create a certificate to use with TDE
CREATE CERTIFICATE TDECERT WITH SUBJECT = 'TDECert'
GO

--Change to the database to encrypt
USE AdventureWorks
GO

--Create your database master key
CREATE DATABASE ENCRYPTION KEY
WITH ALGORITHM = AES_128 --Use a strong algorithm
ENCRYPTION BY SERVER CERTIFICATE TDECERT
GO

--Alter the database to encrypt it with the
--master database key
ALTER DATABASE AdventureWorks
SET ENCRYPTION ON
GO
```

What's in store for developers?

Database administrators are not the only people that will benefit directly from the changes in SQL Server 2008. There are a number of new features that are designed to help database developers. These range from several new T-SQL enhancements to new components that can help developers create and utilise database queries.

Many database developers are responsible for creating the queries that are needed for returning the data required for their applications. You're probably familiar with the LINQ (Language Integrated Query) tool that enables database developers to issue queries against a database using a Microsoft .NET-based programming language instead of the normal T-SQL statements. Well, SQL Server 2008 enhances LINQ by providing a new LINQ to SQL provider that allows developers to issue LINQ commands directly against SQL Server tables and columns. This will reduce the amount of time it takes to create new data queries.

When developing against databases, developers use higher-level objects that they map to individual database tables and columns. These objects, also known as entities, represent the data needed for database applications and, therefore, the developer doesn't need to understand the actual storage structure of the data and schema of the database. The new ADO.NET Entity Framework now allows developers to create database queries using these entities. The abstracting of the underlying database structure allows developers to be more productive.

SQL Server 2008 offers many different enhancements to T-SQL that allow database developers to be more efficient. One example is the new MERGE statement, which allows the developer to check for the existence of data before trying to insert the data. This check prior to performing the INSERT statement allows the data to be updated. No longer is it necessary to create complex joins in order to update data that exists and to insert data that does not already exist, all during a single statement.

In addition, separating time and date data from the combined date/time data type has been made easier. SQL Server 2008 introduces two separate data types to handle date and

time data. Different data types will translate to improved performance for many queries since there will no longer be a need to perform an operation on the data before it can be used in the query.

When creating newer database structures, database developers often find themselves stretching the structure of databases in order to implement mapping applications. SQL Server 2008 helps to address this issue with new spatial data types. The two spatial data types, GEOGRAPHY and GEOMETRY, allow developers to store location-specific data directly into the database without having to break those data elements down into formats that fit other standard data types. The code in **Figure 5** is an example of a simple spatial table.

Figure 5 **A simple spatial table**

```
IF OBJECT_ID ( 'Demo_SpatialTable', 'Table' ) IS NOT NULL
    DROP TABLE Demo_SpatialTable
GO

--Create table to hold spatial data
CREATE TABLE Demo_SpatialTable
    ( SpatialID int IDENTITY (1,1),
    SpatialInputCol geography,
    SpatialOutputCol AS SpatialInputCol.STAsText() )
GO

--Insert data into table
INSERT INTO Demo_SpatialTable (SpatialInputCol)
VALUES (geography::STGeomFromText('LINESTRING(47.656 -122.360, 47.656 -122.343)', 4326));

INSERT INTO Demo_SpatialTable (SpatialInputCol)
VALUES (geography::STGeomFromText('POLYGON((47.653 -122.358, 47.649 -122.348, 47.658 -
122.348, 47.658 -122.358, 47.653 -122.358))', 4326));
GO

--View data to see that data has been converted and stored in col2
SELECT * FROM Demo_SpatialTable
```

A very common issue for database developers in the past was how to store and utilise large binary objects such as documents and media files. The method typically used was to store the files outside of the database and just store a pointer in the database to the external file. With this method, however, when you move the file, you must also remember to update the pointer.

SQL Server 2008 handles this issue with the new FILESTREAM data type. With this data type, files can still be stored outside of the database, but the data is considered part of the database for transactional consistency. This allows for the use of common file operations while still maintaining the performance and security benefits of the database.

What about business intelligence?

Increased use of SQL Server over the last few years has been driven in large part by the adoption of business intelligence strategies. Business intelligence capabilities are not new to SQL Server, but SQL Server 2008 does bring some new features to the table.

For example, when data is stored in data warehouses, space is often wasted due to NULL values. Columns that store NULL values take up the space of the largest allowed data size defined in the column. This means that a column with thousands of NULL values can actually consume many MB of space without actually storing any data.

SQL Server 2008 introduces sparse columns, which allows NULL values to be stored without taking up any physical space on the disk. Because sparse columns do not consume actual space, tables that contain sparse columns can actually exceed the 1,024 column limit.

SQL Server 2008 also introduces a new mechanism, Change Data Capture, for managing incremental changes that need to be loaded into the data warehouse. This captures and places changed data into a set of change tables. Capturing updated, deleted and inserted data in an easily consumed storage schema allows for the incremental loading of data warehouses from those tables – as opposed to having to build custom insert statements that try to figure out the changes made to existing rows of data before updating the data warehouse.

Wrapping up

This is just a quick overview of what SQL Server 2008 has in store. It will bring a broad set of new features and updates to existing features that will improve life for both database administrators and database developers. Ultimately, it will offer much improved performance and scalability for today's ever-demanding databases. ■

Randy Dyess *is a mentor with Solid Quality Mentors. He specialises in SQL Server OLTP systems. Randy is the author of* TransactSQL Language Reference Guide, *coauthor of* MCTS Self-Paced Training Kit: Microsoft SQL Server 2005 Implementation and Maintenance (Exam 70-431), *and numerous magazine and newsletter articles. Randy is also the Director of Programs for the North Texas SQL Server Users Group and is a SQL Server MVP. Visit his blog at http://blogs.solidq.com/EN/rdyess/default.aspx.*