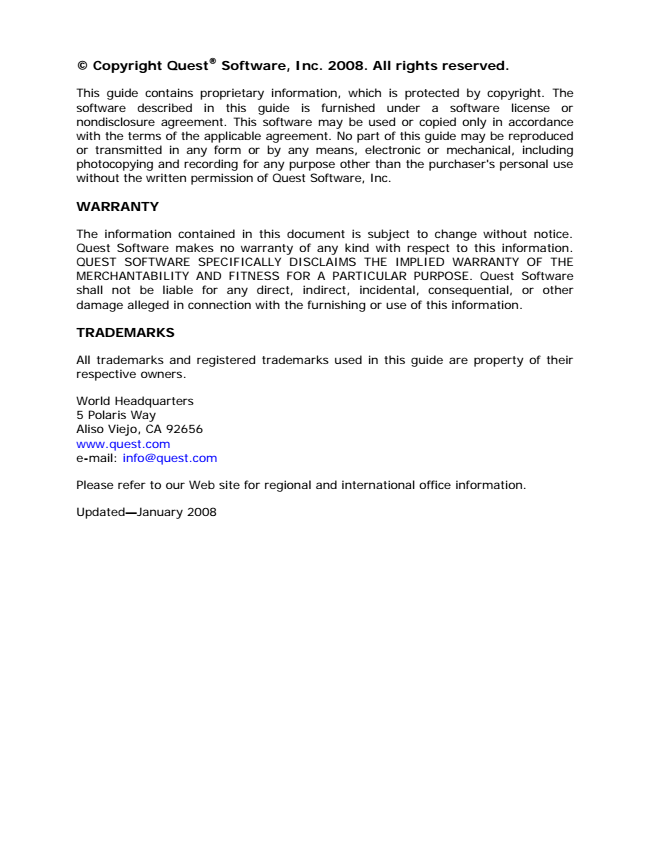


Cool Features for SQL Server 2008

*written by Geoff N. Hiten, SQL Server Expert and Microsoft MVP.*

**

**® © 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.

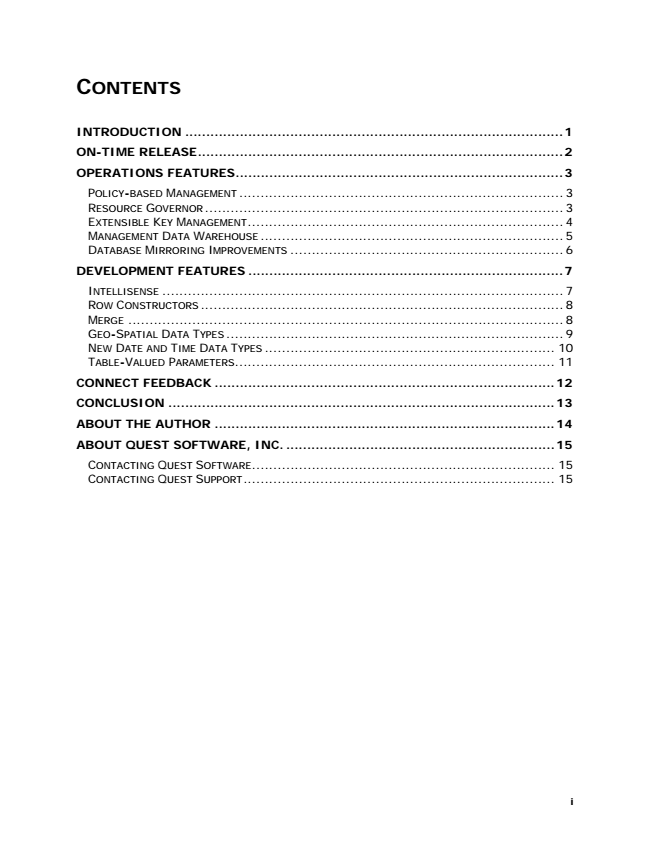
**TRADEMARKS**

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 2008



**CONTENTS**

**INTRODUCTION ..........................................................................................1**

**ON-TIME RELEASE.......................................................................................2**

**OPERATIONS FEATURES..............................................................................3**

POLICY-BASED MANAGEMENT ............................................................................ 3 RESOURCE GOVERNOR.................................................................................... 3 EXTENSIBLE KEY MANAGEMENT.......................................................................... 4 MANAGEMENT DATA WAREHOUSE....................................................................... 5 DATABASE MIRRORING IMPROVEMENTS ................................................................ 6

**DEVELOPMENT FEATURES ...........................................................................7**

INTELLISENSE .............................................................................................. 7 ROW CONSTRUCTORS..................................................................................... 8 MERGE ...................................................................................................... 8 GEO-SPATIAL DATA TYPES............................................................................... 9 NEW DATE AND TIME DATA TYPES .................................................................... 10 TABLE-VALUED PARAMETERS........................................................................... 11

**CONNECT FEEDBACK .................................................................................12**

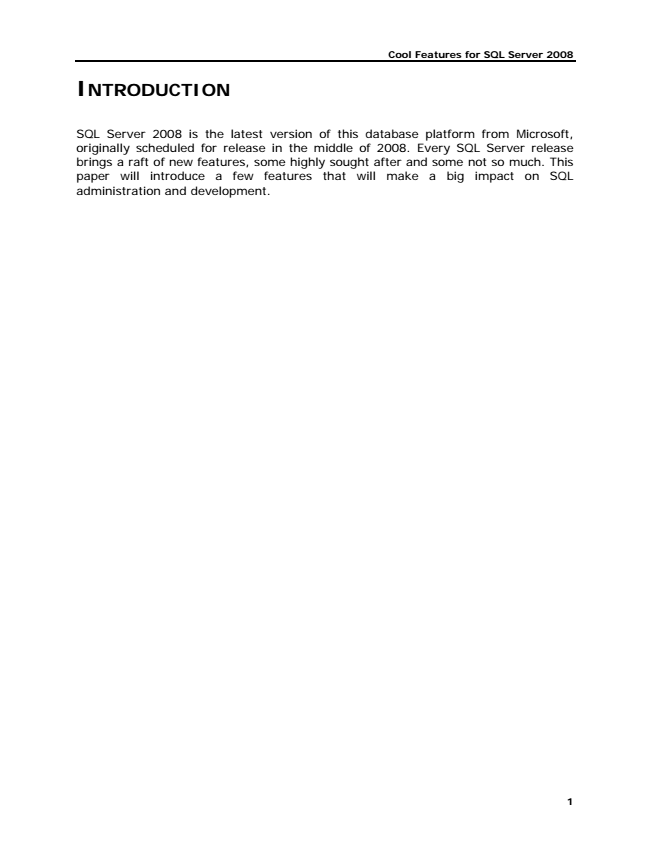
**CONCLUSION ............................................................................................13**

**ABOUT THE AUTHOR .................................................................................14**

**ABOUT QUEST SOFTWARE, INC. ................................................................15**

CONTACTING QUEST SOFTWARE....................................................................... 15 CONTACTING QUEST SUPPORT......................................................................... 15

**i**

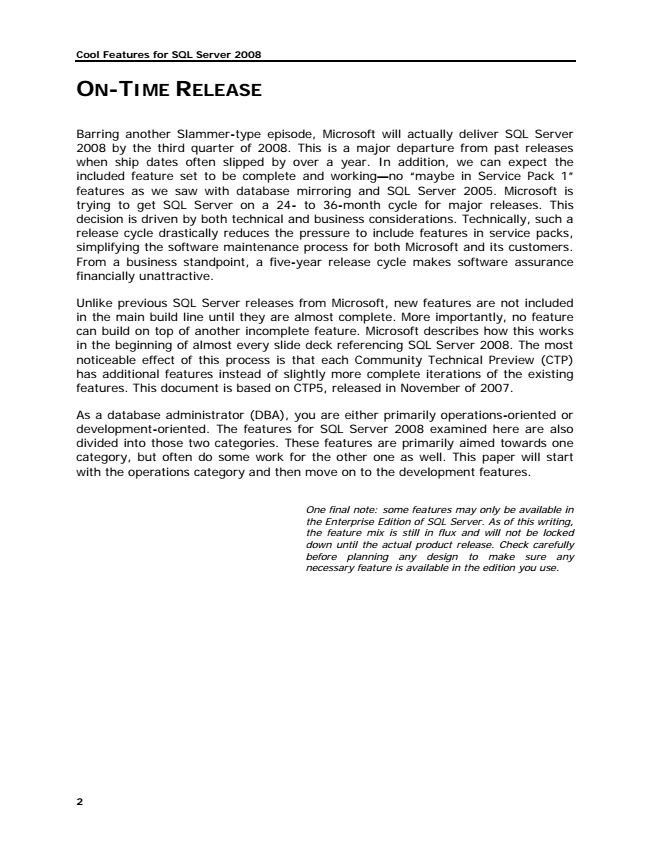
****

**Cool Features for SQL Server 2008**

**INTRODUCTION**

SQL Server 2008 is the latest version of this database platform from Microsoft, originally scheduled for release in the middle of 2008. Every SQL Server release brings a raft of new features, some highly sought after and some not so much. This paper will introduce a few features that will make a big impact on SQL administration and development.

**1**

****

**Cool Features for SQL Server 2008**

**ON-TIME RELEASE**

Barring another Slammer-type episode, Microsoft will actually deliver SQL Server 2008 by the third quarter of 2008. This is a major departure from past releases when ship dates often slipped by over a year. In addition, we can expect the included feature set to be complete and working—no “maybe in Service Pack 1” features as we saw with database mirroring and SQL Server 2005. Microsoft is trying to get SQL Server on a 24- to 36-month cycle for major releases. This decision is driven by both technical and business considerations. Technically, such a release cycle drastically reduces the pressure to include features in service packs, simplifying the software maintenance process for both Microsoft and its customers. From a business standpoint, a five-year release cycle makes software assurance financially unattractive.

Unlike previous SQL Server releases from Microsoft, new features are not included in the main build line until they are almost complete. More importantly, no feature can build on top of another incomplete feature. Microsoft describes how this works in the beginning of almost every slide deck referencing SQL Server 2008. The most noticeable effect of this process is that each Community Technical Preview (CTP) has additional features instead of slightly more complete iterations of the existing features. This document is based on CTP5, released in November of 2007.

As a database administrator (DBA), you are either primarily operations-oriented or development-oriented. The features for SQL Server 2008 examined here are also divided into those two categories. These features are primarily aimed towards one category, but often do some work for the other one as well. This paper will start with the operations category and then move on to the development features.

*One final note: some features may only be available in the Enterprise Edition of SQL Server. As of this writing, the feature mix is still in flux and will not be locked down until the actual product release. Check carefully before planning any design to make sure any necessary feature is available in the edition you use.*

**2**

****

**Cool Features for SQL Server 2008**

**OPERATIONS FEATURES**

**Policy-based Management**

Properly speaking, the Declarative Management Framework (DMF) is not a feature, it is a scenario. In Microsoft-language, a scenario is a group of individual features that offer a complete, interrelated set of functionality. There are several individual features that combine into the new DMF. The concept is quite simple and easy to describe. DBAs define policies such as the recovery model for read-only databases, guest log-in security, or auto-shrink for databases. They then point the policy to one or more servers and choose the policy type to implement. The goal of this scenario is to make managing a group of servers no more complex than managing a single server.

As with any new technology, there are new terms to learn. Microsoft breaks policies into targets, facets and conditions. Targets are the easiest term to understand. A target is any database object. These objects start at an instance and continue all the way down to a single database element such as an index or a view that can have a policy applied to it. Different policies may require different targets. As an example, a recovery model policy only would apply to database targets.

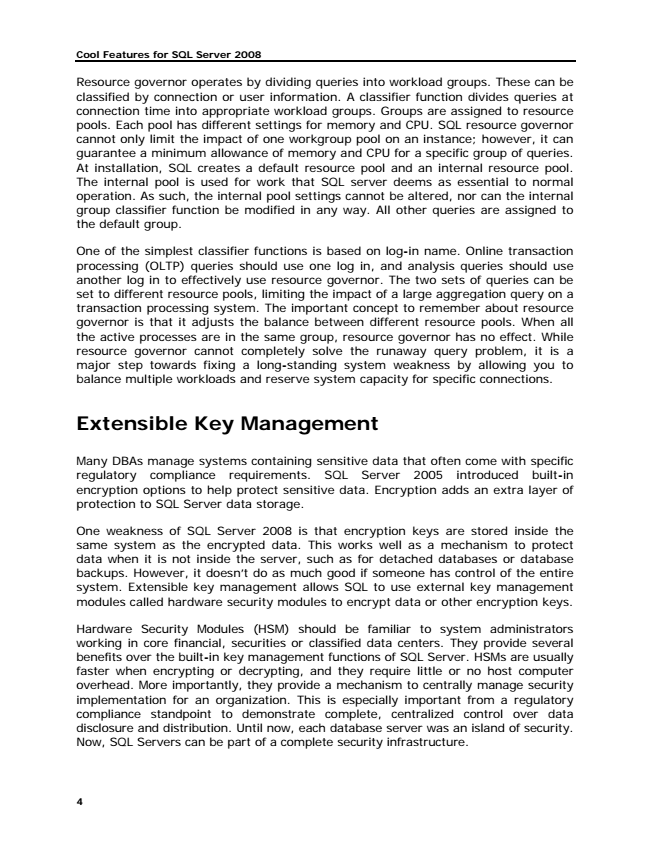
A facet is possibly the most difficult term to describe. A facet is a set of behaviors and characteristics for a particular target that can be controlled by a policy. Facets expose these characteristics as Boolean results called conditions. Put together, a facet set to a particular condition is determined by a policy. The real complexity of the entire DMF scenario is modeling behaviors and conditions into consistently controllable facets.

Policies can be run automatically or on demand. On-demand policies can be run either to check a server’s current configuration or to set a server to match the policy. Policies can be grouped into categories for easier management. Automatic policies have three implementation options. One mode will prevent any changes from the set policy. A second mode will allow changes, but log out-of-compliance items. The third and final mode runs a scheduled check and logs any out-of- compliance items.

**Resource Governor**

SQL Server has always had one major critical weakness: a single, badly formed query can consume all system resources, rendering the system unresponsive for normal operations. Running an analysis-type query on a transaction processing system often generates major end-user complaints due to excessive CPU or memory consumption. SQL Server 2008 introduces a resource governor to limit the impact of a class of queries on a particular system.

**3**

****

**Cool Features for SQL Server 2008**

Resource governor operates by dividing queries into workload groups. These can be classified by connection or user information. A classifier function divides queries at connection time into appropriate workload groups. Groups are assigned to resource pools. Each pool has different settings for memory and CPU. SQL resource governor cannot only limit the impact of one workgroup pool on an instance; however, it can guarantee a minimum allowance of memory and CPU for a specific group of queries. At installation, SQL creates a default resource pool and an internal resource pool. The internal pool is used for work that SQL server deems as essential to normal operation. As such, the internal pool settings cannot be altered, nor can the internal group classifier function be modified in any way. All other queries are assigned to the default group.

One of the simplest classifier functions is based on log-in name. Online transaction processing (OLTP) queries should use one log in, and analysis queries should use another log in to effectively use resource governor. The two sets of queries can be set to different resource pools, limiting the impact of a large aggregation query on a transaction processing system. The important concept to remember about resource governor is that it adjusts the balance between different resource pools. When all the active processes are in the same group, resource governor has no effect. While resource governor cannot completely solve the runaway query problem, it is a major step towards fixing a long-standing system weakness by allowing you to balance multiple workloads and reserve system capacity for specific connections.

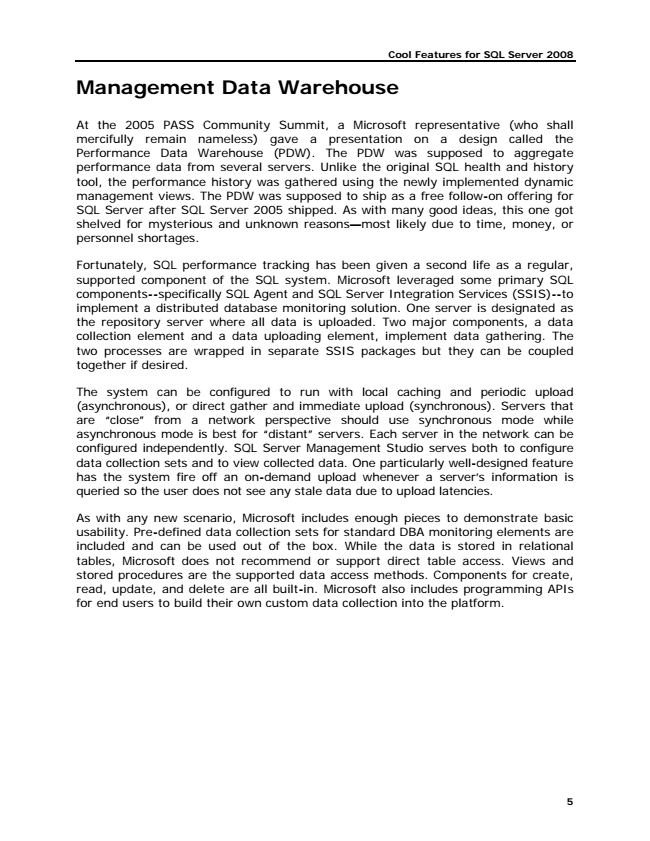
**Extensible Key Management**

Many DBAs manage systems containing sensitive data that often come with specific regulatory compliance requirements. SQL Server 2005 introduced built-in encryption options to help protect sensitive data. Encryption adds an extra layer of protection to SQL Server data storage.

One weakness of SQL Server 2008 is that encryption keys are stored inside the same system as the encrypted data. This works well as a mechanism to protect data when it is not inside the server, such as for detached databases or database backups. However, it doesn’t do as much good if someone has control of the entire system. Extensible key management allows SQL to use external key management modules called hardware security modules to encrypt data or other encryption keys.

Hardware Security Modules (HSM) should be familiar to system administrators working in core financial, securities or classified data centers. They provide several benefits over the built-in key management functions of SQL Server. HSMs are usually faster when encrypting or decrypting, and they require little or no host computer overhead. More importantly, they provide a mechanism to centrally manage security implementation for an organization. This is especially important from a regulatory compliance standpoint to demonstrate complete, centralized control over data disclosure and distribution. Until now, each database server was an island of security. Now, SQL Servers can be part of a complete security infrastructure.

**4**

****

**Cool Features for SQL Server 2008**

**Management Data Warehouse**

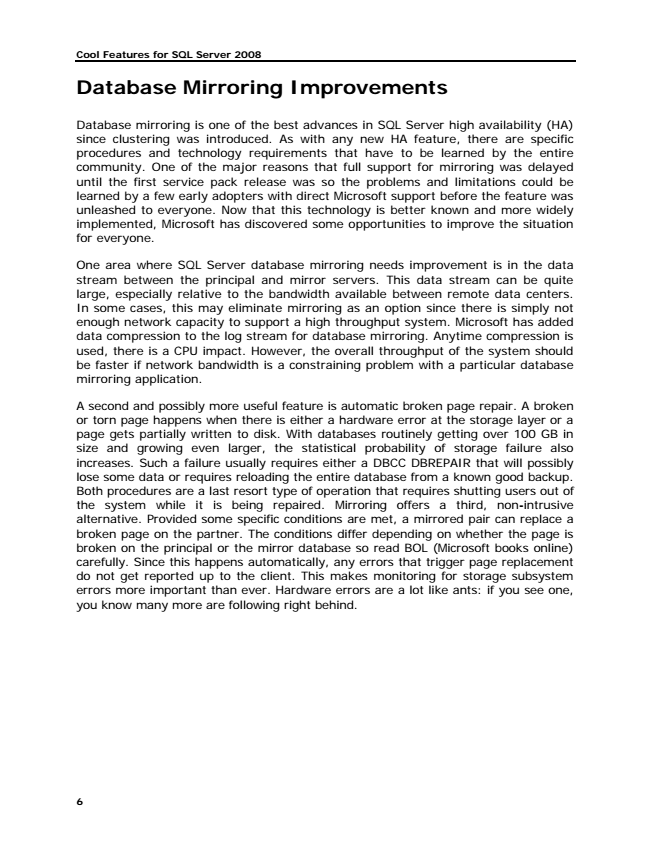
At the 2005 PASS Community Summit, a Microsoft representative (who shall mercifully remain nameless) gave a presentation on a design called the Performance Data Warehouse (PDW). The PDW was supposed to aggregate performance data from several servers. Unlike the original SQL health and history tool, the performance history was gathered using the newly implemented dynamic management views. The PDW was supposed to ship as a free follow-on offering for SQL Server after SQL Server 2005 shipped. As with many good ideas, this one got shelved for mysterious and unknown reasons—most likely due to time, money, or personnel shortages.

Fortunately, SQL performance tracking has been given a second life as a regular, supported component of the SQL system. Microsoft leveraged some primary SQL components--specifically SQL Agent and SQL Server Integration Services (SSIS)--to implement a distributed database monitoring solution. One server is designated as the repository server where all data is uploaded. Two major components, a data collection element and a data uploading element, implement data gathering. The two processes are wrapped in separate SSIS packages but they can be coupled together if desired.

The system can be configured to run with local caching and periodic upload (asynchronous), or direct gather and immediate upload (synchronous). Servers that are “close” from a network perspective should use synchronous mode while asynchronous mode is best for “distant” servers. Each server in the network can be configured independently. SQL Server Management Studio serves both to configure data collection sets and to view collected data. One particularly well-designed feature has the system fire off an on-demand upload whenever a server’s information is queried so the user does not see any stale data due to upload latencies.

As with any new scenario, Microsoft includes enough pieces to demonstrate basic usability. Pre-defined data collection sets for standard DBA monitoring elements are included and can be used out of the box. While the data is stored in relational tables, Microsoft does not recommend or support direct table access. Views and stored procedures are the supported data access methods. Components for create, read, update, and delete are all built-in. Microsoft also includes programming APIs for end users to build their own custom data collection into the platform.

**5**

****

**Cool Features for SQL Server 2008**

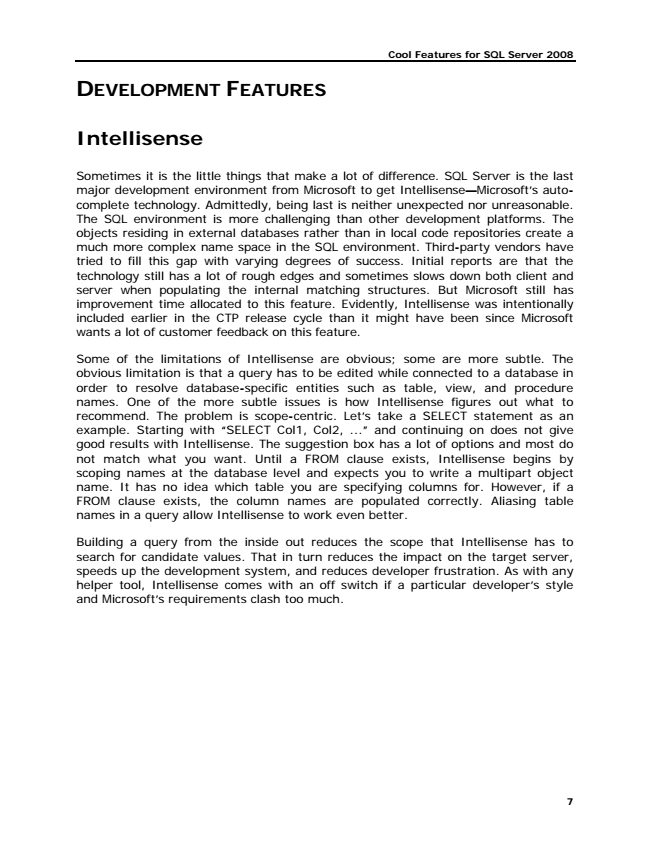
**Database Mirroring Improvements**

Database mirroring is one of the best advances in SQL Server high availability (HA) since clustering was introduced. As with any new HA feature, there are specific procedures and technology requirements that have to be learned by the entire community. One of the major reasons that full support for mirroring was delayed until the first service pack release was so the problems and limitations could be learned by a few early adopters with direct Microsoft support before the feature was unleashed to everyone. Now that this technology is better known and more widely implemented, Microsoft has discovered some opportunities to improve the situation for everyone.

One area where SQL Server database mirroring needs improvement is in the data stream between the principal and mirror servers. This data stream can be quite large, especially relative to the bandwidth available between remote data centers. In some cases, this may eliminate mirroring as an option since there is simply not enough network capacity to support a high throughput system. Microsoft has added data compression to the log stream for database mirroring. Anytime compression is used, there is a CPU impact. However, the overall throughput of the system should be faster if network bandwidth is a constraining problem with a particular database mirroring application.

A second and possibly more useful feature is automatic broken page repair. A broken or torn page happens when there is either a hardware error at the storage layer or a page gets partially written to disk. With databases routinely getting over 100 GB in size and growing even larger, the statistical probability of storage failure also increases. Such a failure usually requires either a DBCC DBREPAIR that will possibly lose some data or requires reloading the entire database from a known good backup. Both procedures are a last resort type of operation that requires shutting users out of the system while it is being repaired. Mirroring offers a third, non-intrusive alternative. Provided some specific conditions are met, a mirrored pair can replace a broken page on the partner. The conditions differ depending on whether the page is broken on the principal or the mirror database so read BOL (Microsoft books online) carefully. Since this happens automatically, any errors that trigger page replacement do not get reported up to the client. This makes monitoring for storage subsystem errors more important than ever. Hardware errors are a lot like ants: if you see one, you know many more are following right behind.

**6**

****

**Cool Features for SQL Server 2008**

**DEVELOPMENT FEATURES**

**Intellisense**

Sometimes it is the little things that make a lot of difference. SQL Server is the last major development environment from Microsoft to get Intellisense—Microsoft’s auto- complete technology. Admittedly, being last is neither unexpected nor unreasonable. The SQL environment is more challenging than other development platforms. The objects residing in external databases rather than in local code repositories create a much more complex name space in the SQL environment. Third-party vendors have tried to fill this gap with varying degrees of success. Initial reports are that the technology still has a lot of rough edges and sometimes slows down both client and server when populating the internal matching structures. But Microsoft still has improvement time allocated to this feature. Evidently, Intellisense was intentionally included earlier in the CTP release cycle than it might have been since Microsoft wants a lot of customer feedback on this feature.

Some of the limitations of Intellisense are obvious; some are more subtle. The obvious limitation is that a query has to be edited while connected to a database in order to resolve database-specific entities such as table, view, and procedure names. One of the more subtle issues is how Intellisense figures out what to recommend. The problem is scope-centric. Let’s take a SELECT statement as an example. Starting with “SELECT Col1, Col2, ...” and continuing on does not give good results with Intellisense. The suggestion box has a lot of options and most do not match what you want. Until a FROM clause exists, Intellisense begins by scoping names at the database level and expects you to write a multipart object name. It has no idea which table you are specifying columns for. However, if a FROM clause exists, the column names are populated correctly. Aliasing table names in a query allow Intellisense to work even better.

Building a query from the inside out reduces the scope that Intellisense has to search for candidate values. That in turn reduces the impact on the target server, speeds up the development system, and reduces developer frustration. As with any helper tool, Intellisense comes with an off switch if a particular developer’s style and Microsoft’s requirements clash too much.

**7**