Performance

Improving Microsoft Dynamics CRM 4.0 Performance and Securing Data with Microsoft SQL Server 2008 R2

White Paper

Date: October 2011

Microsoft Dynamics™

Acknowledgements

Initiated by the Microsoft Dynamics CRM *Engineering for Enterprise* (MS CRM E^2) Team, this document was developed with support from across the organization and in direct collaboration with the following:

SQL Server Product Development Team

- Peter Scharlock (*Key Contributor*)
- Sunil Agarwal (*Technical Review*)
- Srini Acharya (*Technical Review*)
- Peter Carlin (*Technical Review*)

UK MCS

Roger Gilchrist (*Technical Review*)

The MS CRM E^2 Team recognizes their efforts in helping to ensure delivery of an accurate and comprehensive technical resource in support of the broader CRM community.

MS CRM E^2 Contributors

Amir Jafri, Program Manager

Jim Toland, Content Project Manager

Feedback

To send comments or suggestions about this document, please click the following link and type your feedback in the message body:
http://go.microsoft.com/fwlink/?LinkId=228479

Important: The subject-line information is used to route your feedback. If you remove or modify the subject line, we may be unable to process your feedback.

Table of Contents

Overview

Project Overview

Microsoft SQL Server 2008 R2 contains a variety of features that, when implemented properly, can improve the performance of a Microsoft Dynamics CRM 4.0 implementation and secure the data within that deployment. These Microsoft SQL Server 2008 R2 features include:

- Compression
- Sparse Columns
- Transparent Data Encryption
- Backup Compression

The MS CRM E2 team, working in conjunction with the Microsoft SQL Server team, recently completed a project that was designed to:

1. Evaluate the new scenarios that these Microsoft SQL Server 2008 R2 features expose
2. Measure the performance impact of implementing these features, both singly and in selected combinations

This paper provides an overview of these Microsoft SQL Server 2008 R2 features, together with benchmark results and recommendations for implementation. This document is organized into three sections, as described in the following table.

Section	Description
Overview of Relevant SQL Server 2008 R2 Features	Provides a high-level description of the SQL Server 2008 R2 features that were included in the testing effort
Details of Performance Analysis	Provides detail about the database, workload, testing process and methodology, and hardware environment for the project
Testing Results and Recommendations	Provides results of the testing effort and recommendations about how and when to use specific SQL Server 2008 R2 features

Overview of Relevant SQL Server 2008 R2 Features

Compression

Microsoft SQL Server 2008 R2 supports ROW and PAGE compression for tables and indexes.

Note: For more information about compressed tables and indexes in SQL Server 2008 R2, on MSDN, see *Creating Compressed Tables and Indexes* at: http://msdn.microsoft.com/en-us/library/cc280449.aspx.

ROW Compression

ROW compression maps a fixed length data type to variable length physical storage to save space used to store the data; basically, it compresses columns in the row. For example, a CHAR(100) column stored in a variable length storage format only uses up the amount of storage defined by the data.

With ROW compression enabled, storing "SQL Server 2008" in the column requires storing only 15 (not the full 100) characters, representing a savings of 85%. Also, with ROW compression enabled, storing zero or null values requires no storage space.

Note: For more information about ROW compression, on MSDN, see *ROW Compression Implementation* at: http://msdn.microsoft.com/en-us/library/cc280576.aspx.

PAGE Compression

A superset of ROW compression, PAGE compression takes into account the redundant data in one or more rows on a page to save space used to store the data. PAGE compression uses column prefixes and a page level dictionary technique for this task. In other words, with both page compression techniques, the storage engine reduces the amount of data that is repeated in the page.

Note: For more information about PAGE compression, on MSDN, see *PAGE Compression Implementation* at:
http://msdn.microsoft.com/en-us/library/cc280464.aspx.

Sparse Columns

Sparse columns are ordinary columns that have an optimized storage for NULL values. Sparse columns reduce the space requirements for NULL values at the cost of greater CPU overhead to retrieve not NULL values. Sparse columns enable applications, such as Windows SharePoint Services, to efficiently store and access a large number of user-defined properties by using SQL Server 2008 R2.

INSERT, UPDATE, and DELETE statements can reference the sparse columns by name. Additionally, you can also view and work with all the sparse columns of a table that are combined into a single XML column. This column is called a column set.

Note: Sparse columns and column sets are defined by using the CREATE TABLE or ALTER TABLE statements. For more information about sparse columns, on MSDN, see *Using Sparse Columns* at:
http://msdn.microsoft.com/en-us/library/cc280604.aspx.

Transparent Data Encryption

Transparent data encryption (TDE) is designed to provide protection for the entire database "at rest" without affecting existing applications. Implementing encryption in a database traditionally involves complicated application changes such as modifying table schemas, removing functionality, and significant performance degradations.

TDE solves these problems by simply encrypting everything while being transparent to the application. Thus, all data types, keys, indexes, and so on can be used to their full potential without sacrificing security or leaking information on the disk. While column or cell-level encryption cannot offer these benefits, two Windows features, Encrypting File System (EFS) and BitLocker Drive Encryption, are often used for the same reasons as TDE; they provide protection on a similar scale and are transparent to the user.

With similar functionalities to that in EFS or BitLocker Drive Encryption technology, TDE also provides the following advantages:

- TDE is *sticky*, so files are encrypted as they are detached or backed up
- A Windows administrator is not required to enable encryption; the task is associated with the database administrator, reducing redundant administrator workload
- All database encryption is managed by SQL Server

Note: For more information about transparent data encryption, see the MSDN SQL Server 2008 Technical Article *Database Encryption in SQL Server 2008 Enterprise Edition* at:
http://msdn.microsoft.com/en-us/library/cc278098.aspx.

Backup Compression

Because a compressed backup is smaller than an uncompressed backup of the same data, compressing a backup typically reduces device I/O and therefore usually increases backup speed significantly.

Important: Creating compressed backups is supported only in SQL Server 2008 Enterprise edition and later versions, but every edition of SQL Server 2008 and later versions can restore a compressed backup. Also, backup compression is not recommended for use with TDE.

Note: For more information, on MSDN, see *Backup Compression (SQL Server)* at http://msdn.microsoft.com/en-us/library/bb964719.aspx.

Details of Performance Analysis

The project involved enabling certain SQL Server features on a Microsoft Dynamics CRM database and then evaluating the impact of those changes by using benchmarks or other means. To capture the data profile of a typical Microsoft Dynamics CRM customer, the database used in the benchmark contained both customer and generated data. To simplify the process of measuring the performance impact, benchmark testing efforts focused on the Accounts table. Similar levels of performance impact are expected across all Microsoft Dynamics CRM entity tables.

Database

The database used in the testing was partially populated with real customer data and partially generated. Additional characteristics of the database tested include:

- 15.5 GB physical size
- 200 users
- Over 2 million Accounts

Workload

The Microsoft Dynamics CRM 4.0 Performance Toolkit was used to execute the benchmark. To generate a 200-concurrent user Account Management application workload, we used the following test cases in the toolkit:

- Create Account
- Find Account
- Update Account
- Delete Account
- Advanced Find on Account

Testing Process and Methodology

Testing evaluated each feature independently, as well as selected features in combination. We used the same methodology in testing each scenario, with the exception of enabling Backup Compression, for which we measured the time and space required for an uncompressed backup versus a compressed backup.

The benchmark testing process and methodology involved:

1. Executing the baseline benchmark
2. Modifying the database for each scenario:
    a. Compression
    b. Sparse Columns
    c. TDE
3. Re-indexing the Microsoft Dynamics CRM database (if necessary)
4. Measuring the impact on table size
5. Executing the benchmark

Hardware Environment

The hardware environment reflected a three-server Microsoft Dynamics CRM deployment and a single test computer, which were configured as shown below.

Microsoft Dynamics CRM 4.0 Server

- Dual Proc Intel Xeon CPU 2.33 GHz
- 16 GB RAM
- Windows 2008 64-bit

SQL Server 2008

- Dual Proc Intel Xeon CPU 2.33 GHz
- 16 GB RAM
- Windows 2008 64-bit

Domain Controller

- Dual Proc Intel Xeon CPU 2.33 GHz
- 16 GB RAM
- Windows 2008 64-bit

Test Computer

- Visual Studio 2005 Test Suite with Microsoft Dynamics CRM 4.0 Performance and Stress Testing Toolkit
- Dual Proc Intel Xeon CPU 2.33 GHz
- 16 GB RAM
- Windows 2008 64-bit

Testing Results and Recommendations

Improving Performance by Using SQL Server 2008 R2 Compression

Columns in Microsoft Dynamics CRM tables are typically sparsely populated, which our analysis of the customer database used in the testing efforts confirmed. Except for certain columns such as Account ID, Name, Phone Number and Email Address, the Account table was sparsely populated. Both row and page compression are very effective on Microsoft Dynamics CRM entity tables, which also contain a lot of binary columns and columns that specify pick list values.

Results - Row Compression

Benchmark testing indicated that row compression results in significant improvement in application performance as well as a substantial savings in space requirements.

Measure	Baseline	w/ Row Compression	Percent Change
Data (KB)	2733672	1949464	***-28.69***
Index (KB)	1899000	1903088	*0.22*
Avg. resp. time (sec)	1.3775	0.90	***-35.03***
SQL Util. (%)	21.575	16.80	***-22.13***
Web Util. (%)	2.135	2.01	***-5.74***

Results - Page Compression

Page compression proved to be the best option if space is at a premium. While application performance was roughly the same with or without page compression enabled, implementing page compression resulted in a reduction in space requirements by over 80%!

Measure	Baseline	w/ Page Compression	Percent Change
Data (KB)	2733672	507072	***-81.45***
Index (KB)	1899000	1905552	*0.35*
Avg. resp. time (sec)	1.3775	1.43	*3.81*
SQL Util. (%)	21.575	19.44	***-9.90***
Web Util. (%)	2.135	2.03	***-4.92***

Recommendations

When considering the use of SQL Server 2008 R2 Compression to improve the performance of a Microsoft Dynamics CRM 4.0 implementation, keep in mind the following recommendations:

1. Identify the largest tables in the Microsoft Dynamics CRM database and consult SQL Server best practices to pick the best candidates for compression.

   Note: Microsoft Dynamics CRM tables (such as metadata tables) that are frequently accessed or continuously updated may not be good candidates for compression. An example is the PrincipleObjectAccess table which manages the privileges for the Microsoft Dynamics CRM system. In these cases, the performance impact of managing these compressed tables may outweigh the space savings from compression.

2. Estimate savings for each table by using the following stored procedure *sp_estimate_data_compression_savings*

3. Enable Page Compression on tables with mostly static data.
4. Enable Row Compression on entity tables.
5. Verify performance impact.

Important: Compressing every table in the database may put a lot of load on the computer running SQL Server and compression should be limited to large tables as per SQL Server best practices.

Note: For additional information about compression strategies in SQL Server 2008, on the MSDN Developer Blog, as part of the information related to the SQL Server Storage Engine, see the entry *Compression Strategies* at: http://blogs.msdn.com/sqlserverstorageengine/archive/2008/01/27/compression-strategies.aspx.

Improving Performance by Using SQL Server 2008 R2 Sparse Columns

As mentioned previously, columns in Microsoft Dynamics CRM tables typically are sparsely populated. The sparse columns feature in SQL 2008 is perfectly suited for Microsoft Dynamics CRM tables because it reduces the space required to store data in user-specified columns. In addition, by setting the columns as sparse, Microsoft Dynamics CRM administrators can optimize access to frequently accessed tables in which certain columns are rarely accessed and include all or mostly NULL values.

Benchmark testing on the Account table demonstrated significant improvements in space requirements and application performance. Using the guidance provided by the SQL Server best practices that are documented in the topic *Using Sparse Columns* in the *Designing and Implementing Semistructured Storage (Database Engine)* MSDN article, we marked 36 of the 83 columns in the AccountBase table as sparse.

Note: SQL Server CPU utilization would have improved even more if the server running SQL Server had been memory limited. The computer running SQL Server in the benchmark testing had 16 GB of RAM and the database was 15.5 GB, so therefore over the course of testing the entire Account table was resident in memory.

Results

Measure	Baseline	w/ Sparse Columns	Percent Change
Data (KB)	2733672	2244160	***-17.91***
Index (KB)	1899000	2086696	*9.88*
Avg. resp. time (sec)	1.3775	1.03	***-25.23***
SQL Util. (%)	21.575	19.64	***-8.97***
Web Util. (%)	2.135	2.05	***-4.22***

Recommendations

Designating a column as sparse is only useful when the column contains mostly NULL values; consider using sparse columns when the space saved is at least 20 percent to 40 percent to strike a balance between space savings and additional CPU overhead.

As you work to improve Microsoft Dynamics CRM 4.0 performance by using SQL Server 2008 R2 Sparse Columns, keep in mind the following recommendations:

1. Identify large tables in the Microsoft Dynamics CRM database.
2. Consult SQL Server best practices documented in the artice mentioned previously to select the columns to be marked as sparse; typically, use sparse storage for columns in which most values are NULL.

   Note: Marking non-sparse columns as sparse will significantly increase the amount of space needed to store the data.
3. Mark the columns as sparse and rebuild the indexes on the table.
4. Verify performance impact.

Securing Microsoft Dynamics CRM Data by using SQL Server 2008 R2 Transparent Data Encryption

TDE impacts how the data is stored on disk. The test database was approximately 15.5 GB and the computer running SQL Server was configured with 16 GB of RAM.

Results

After the data is decrypted and loaded into memory, application performance was the same as the performance without encryption. To measure the actual impact of TDE, we flushed the SQL Server buffer pool and data cache before every run to ensure that data is loaded from disk. Results demonstrate that with TDE enabled, first-run application performance experienced a slight (less than 10%) degradation in performance. As expected, SQL Server CPU utilization was also higher as a result of the overhead associated with decrypting the data on disk during the initial run. With the data already decrypted and loaded into memory, subsequent runs yielded results similar to the baseline figures.

Measure	Baseline	w/ Trans. Data Encrypt.	Percent Change
Data (KB)	2733672	*2733928*	*0.01*
Index (KB)	1899000	*1900616*	*0.09*
Avg. resp. time (sec)	1.3775	*1.50*	*8.71*
SQL Util. (%)	21.575	*24.13*	*11.82*
Web Util. (%)	2.135	*2.09*	***-2.34***

Note: We verified that the database files were not readable via a hex editor after encryption.

Recommendations

For business scenarios that require a level of protection for the entire database at rest, consider enabling TDE, which test results confirm will have a minimal effect on the performance of existing applications.

Performing and Maintaining Backups More Efficiently by Using SQL Server 2008 R2 Backup Compression

As expected, results confirmed that enabling Backup Compression yields significant savings in the time required to perform backups and in the space required for storing those backups.

Results

Measure	Uncompressed	Compressed	Percent Change
Backup Time (min)	9:03:09	2:38:51	***-70.75***
Backup Size (Kb)	12299686	2791721	***-77.03***

Note: The size of the database used in this test was 15.5 gigabytes.

Recommendations

Enable backup compression to increase the efficiency of performing and maintaining backups. However, keep in mind that compression can significantly increase CPU usage and that the additional overhead might adversely impact concurrent operations. As a result, when using backup compression, be sure to verify that the overall system performs at desired levels. Additionally, consider creating low-priority compressed backups in a session whose CPU usage is limited by Resource Governor.

Note: For more information, on MSDN, see *How to: Use Resource Governor to Limit CPU Usage by Backup Compression (Transact-SQL)* at http://msdn.microsoft.com/en-us/library/cc280384.aspx.

Summary

In summary, the results of our testing efforts demonstrate that the SQL 2008 R2 features we evaluated can significantly improve the application performance of both SQL Server 2008 R2 and Microsoft Dynamics CRM 4.0 while resulting in space savings. While our testing focused exclusively on the Accounts entity, we would expect similar results across all Microsoft Dynamics CRM entities.

In addition, deploying these features does not require invasive infrastructure changes, and the features are easily enabled and disabled.

Based on the project results, it is recommended that customers use the information contained in this document, together with best practices defined in the SQL Server 2008 R2 documentation, to take full advantage of the potential efficiencies in performance and space utilization that these features can offer.

Important: Be sure to verify the performance gains of making specific changes to the configuration of your Microsoft Dynamics CRM implementation.

Appendix A: Resources

The following resources contain additional information about improving Microsoft Dynamics CRM performance and securing data with by using selected features in Microsoft SQL Server 2008 R2.

- *Microsoft SQL Server Books Online*
  http://msdn.microsoft.com/en-us/library/ms130214.aspx
- *Creating Compressed Tables and Indexes*
  http://msdn.microsoft.com/en-us/library/cc280449.aspx
- *ROW Compression Implementation*
  http://msdn.microsoft.com/en-us/library/cc280576.aspx
- *PAGE Compression Implementation*
  http://msdn.microsoft.com/en-us/library/cc280464.aspx
- *Filtered Index Design Guidelines*
  http://msdn.microsoft.com/en-us/library/cc280372.aspx
- *Designing and Implementing Semistructured Storage (Database Engine)*
  http://msdn.microsoft.com/en-us/library/bb522446.aspx
- *Using Sparse Columns*
  http://msdn.microsoft.com/en-us/library/cc280604.aspx
- *Database Encryption in SQL Server 2008 Enterprise Edition*
  http://msdn.microsoft.com/en-us/library/cc278098.aspx
- *Backup Compression (SQL Server)*
  http://msdn.microsoft.com/en-us/library/bb964719.aspx
- *How to Use Resource Governor to Limit CPU Usage by Backup Compression (Transact-SQL)*
  http://msdn.microsoft.com/en-us/library/cc280384.aspx