UNIVERSITY OF WATERLOO
Faculty of Engineering
Nanotechnology Engineering

Upgrading to Microsoft SQL Server 2005

Firmwater Inc.
Toronto, ON

Prepared By:

Rajesh Kumar Swaminathan
1B Nanotechnology
ID #20194189
rajesh@meetrajesh.com

September 13, 2006

Rajesh Kumar Swaminathan

#27-8289 121A St.

Surrey, BC V3W 1G6

September 13, 2006

Dr. Marios Ioannidis, Director

Nanotechnology Engineering

University of Waterloo

Waterloo, Ontario N2L 3B9

Dear Dr. Ioannidis,

This report, entitled, "Upgrading to SQL Server 2005" is my first work term report for the term immediately following 1B spanning the months of May to August 2006. This report was completed during my work term at Firmwater Inc., a company that specializes in creating software for *learning management systems*. The purpose of this report is to investigate what is involved in upgrading our software application's database backend from Microsoft SQL Server 2000 to SQL Server 2005, the latest available general release.

This idea for this topic was specifically suggested by my work term supervisor, Mr. Stefan Leyhane. Stefan was instrumental in paving the direction of this report, and providing the required help and guidance whenever it was sought. He has also been very helpful in answering my many queries and giving me wholesome descriptions of our application's internals so that I constantly kept seeing the "big picture". Since this report is being written at Stefan's request, I have assumed him as my primary audience for the purpose of this report.

My other two colleagues Mr. Jan Vopalensky and Mr. Richard Williams were very encouraging and have provided me, throughout the course of the term, with several hints, bugs, and concerns that have helped me identify issues with our current database version.

Although my topic has nothing to do with Nanotechnology Engineering, I regard it as a great personal asset to be well-versed in something that is totally out of my stated domain. The work

term has been deeply enriching and educational in both the amount of exposure I've gained, as well as the number of new technologies I've been able to learn over a period of just four months. In all, I consider this work term to be a hundred percent successful, despite its irrelevance to nanotechnology.

The topic of this report proved itself to not only be an insightful and exciting topic to research, but a topic that, I hope, will be of immense usefulness to our company and the software product we ship. There are a lot of benefits and outcomes to be discussed in upgrading an application's database server; however, due to the size limitations of this report, I have restricted myself to a select few targeted topics that are of direct relevance to our application's design.

I vouch to the fact that I have received no further help other than what is mentioned above and in the references section in writing this report. I also confirm that this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

Rajesh Swaminathan
20194189

Contributions

At Firmwater Inc., I was a web application developer within a relatively small team of six people: half of these, including myself, were co-op students. Our company specializes in writing software for *learning managements systems* (LMS).

Our primary goal for the four-month term was to push releases for versions 1.6 and 1.7, both of which introduced major features and functionality additions. A few of these were:

- Add multi-lingual support to the corporation's flagship learning management system.
- Enhance the already existing web services API (application programming interface) to enable search functionality between business-to-business (B2B) application clients.
- Customize person fields to allow any number of name-value metadata fields to be collected, stored and validated for a particular client.
- Allow users to self register for courses that are available to them, and handle any payment associated with these courses and relevant options.
- Allow users to self register for new accounts if allowed.
- Add support for location hierarchies and location type hierarchies. This is useful for large organizations that have a number of users spread over several locations around the country.

As a web developer, I was responsible for adding new features to the LMS. I worked in a team of two with a second year co-op student to write code, in the C# programming language, to add features in a seamless manner without breaking or interrupting existing code. I also worked in parallel with a third Waterloo co-op student in-charge of quality assurance, as well as the company's release manager, to ensure releases are brought together in a timely fashion, to the satisfaction of the client.

Although I did have a few long-term responsibilities in terms of active maintenance of the code-base, writing documentation in the project's official wiki, and streamlining the build process, my main responsibility on a day-to-day basis comprised of the following:

- Work toward adding the stated release features to the LMS.
- Identify bugs in the software system and fix them in a timely and intelligent fashion.
- Check with quality assurance (QA) if my fixes are adequate.
- Coordinate with my supervisor and an other co-op student developer to find better and more creative methods of solving common, especially recurring, problems.
- Document my processes and solutions in the company's official wiki, both for our organization, as well as for future co-op students who might build on my additions.

The company's private project management system contains a summary of all action items worked on during the term.

In the broader scheme of things, our application, built on Microsoft technologies, relies heavily on two pieces of software for its working:

1. ASP.*NET*, a core component of the *.NET* framework from Microsoft.
2. Microsoft SQL Server 2000 for data storage and database functionality.

Late last year, Microsoft introduced a new release of its SQL Server database system, named SQL Server 2005. Although our organization is fairly certain that an upgrade will occur at some point in time, my report focusses on the issues to be considered and the benefits to be gained by upgrading, thereby facilitating the upgrade process and providing increased confidence.

Summary

Many software companies relying on Microsoft-based data solutions are considering an upgrade from SQL Server 2000 to the latest available general release, SQL Server 2005. Microsoft has made it clear, via its various published white-papers, that the new version of their database server packs a number of features and implements a vast number of changes, improvements and bug fixes. In light of this statement, organizations reliant on SQL Server 2000, such as ours, are delighted to upgrade as quickly as we can to the latest version. However, there also remains a concern as to what parts of the application may break as a result of the upgrade.

This paper attempts to analyze the various features in the new version, strategies for upgrade, performance and ease-of-use gains, and the amount of work that will need to be put in before we ensure a successful upgrade. The analysis will look at only those changes that have a direct relevance to Firmwater's learning management system (LMS).

Following an examination of the issues and concerns outlined, the report proceeds to recommend the upgrade. The conclusions section outlines what is to be gained and what is to be taken into account both before and after the upgrade.

Conclusions and Recommendations

Examining the upgrade process reveals that there is very little to lose, but much to gain, by upgrading to SQL Server 2005. Noteworthy benefits are:

- SQL server 2005 management studio for developer productivity (Section 2.1.1)
- SQL prompt plugin enhancements (Section 2.1.2)
- C# stored procedures (Section 2.2)
- Stored procedure optimization of search grids (Section 2.4)
- Reporting services enhancements (Section 2.5)

Issues to be kept in mind when considering an upgrade that relate directly to our application are:

- Unsupported `ORDER BY` table prefixes (Section 3.1)
- Unsupported `GROUP BY` expressions (Section 3.2)
- Issues relating to backwards compatibility (Section 3.3)

Further potential issues may be discovered by browsing through the provided SQL Server Upgrade Advisor's documentation.

A side-by-side upgrade methodology as opposed to an in-place upgrade is recommended. The reasoning behind this is detailed in Section 1.2.

Two weeks before preparing this report, an installation was carried out on a local setup. Based on initial testing, all pages within the application loaded without issues. Basic database operations such as add, edit and delete worked without any indication of error. The Web Service's `Search()` call yielded expected results. No testing was performed on either the reporting services or the assessment engine components.

Table of Contents

List of Figures

List of Tables

1 Introduction

SQL Server, Microsoft's flagship enterprise relational database management system (RDBMS), released a new version of its server in late 2005. This new release sparked a wave of interest among software developers and application vendors.

The primary purpose of this report is:

1. We want to spend time upfront so that we can increase our confidence and reduce any uncertainty and doubt about upgrading from 2000 to 2005.
2. We want to decrease any risk involved in upgrading by addressing all mentioned concerns *before* the upgrade is conducted. This can be done by performing an upgrade advisory and running profiler-generated database trace files through utilities provided by Microsoft.
3. We want suggestions for best practices so that the upgrade can be conducted as smoothly as possible with minimal upgrade downtime or without any loss of information.

1.1 Why Upgrade?

For any database-backed software, the database server in use is a great deal of concern. Microsoft SQL Server, being the data warehouse of our learning management system, is imperatively critical to our application's functionality. It is a core component of our IT infrastructure, and consequently, any room for growth or improvement in this domain is widely sought after. One of Firmwater's key business objectives is to deliver a light-weight application that is simple, fast, and easy to deploy. Microsoft promises up to 35% faster transaction processing, and a statistic like this offers a great deal of promise to improving the overall performance of our application. SQL Server 2005 also promises enhanced developer productivity [7], which is a great win for an organization like ours, where half our work-force consist of co-op students.

SQL Server 2005 is purported to be geared for security, Transact-SQL query performance, full-text catalogs for searching, proactive query execution plan caching, and better management tools to

	SQL Server 2000	SQL Server 2005
RAM	2 GB	No Limit
Failover Clustering	N/A	✓
Online System Changes	N/A	Allowed
Enterprise Manager	✓	N/A
SQL Server Management Studio	N/A	✓
Index Tuning Wizard	✓	N/A
Database Engine Tuning Advisor	N/A	✓
Dynamic Management Views	N/A	✓
Data Encryption and Key Management	N/A	✓
T-SQL Enhancements (advanced exception handling, recursive queries, and support for new data types)	N/A	✓
CLR and .NET Integration	N/A	✓
User-defined Data Types	N/A	✓
Native XML Support	N/A	✓
Database Mail	N/A	✓
Report Builder	N/A	✓
SQL Analytical Functions	N/A	✓
Native Support for Web Services	N/A	✓
HTTP Internet Support	N/A	✓
Custom Rollups	N/A	✓
Calculated Cells	N/A	✓
Actions	N/A	✓
.NET Stored Procedures	N/A	✓

Table 1: SQL Server 20005 Feature Upgrade Matrix (standard editions on 32-bit systems). [1]

improve developer efficiency — all of which are directly relevant to our LMS. SQL Server 2005 also comes built-in with powerful reporting, integration and analysis services that prove to be indispensable when using the database server in different contexts.

A comparison matrix between the standard editions of SQL Server 2000 and SQL Server 2005 is provided in Table 1. Only those changes that might be relevant to our learning management system are included.

1.1.1 Business Value

Microsoft published statistics based on its pre-release version of SQL Server 2005, also known as Community Technology Preview (CTP). These statistics claim [2] that SQL Server 2005:

- minimizes business disruptions through increased 24/7 availability, scalability and security.
- gains deeper business insight through richer end-user analytics and reporting tools that result in a faster return on investment.
- accelerates the development time of business applications by up to 40% with Visual Studio 2005 and .NET integration, including 50-70% code reduction for most scenarios.
- reduces data management complexity and eases manageability.

It is clear that if these advances can be corroborated through tests done on our own code-base, an upgrade can be unconditionally recommended. Our main goal, if an upgrade is to be conducted, is to improve performance, reduce cost of maintenance, and lower complexity of our code-base.

1.2 Upgrade Methodology

The SQL Server 2005 Technical Reference Guide [1] suggests two possible methods to upgrade:

- An in-place upgrade
- A side-by-side upgrade

An in-place upgrade completely replaces the 2000 instance with a new 2005 instance. This can be advantageous in some scenarios as the data file transfer and server configuration is done automatically by the upgrade script. However, it does not allow a single specific database to be migrated. All databases will have to be migrated.

A side-by-side upgrade (Fig. 1.2), on the other hand, is more flexible as it allows a single database to be run on the new instance. Two instances are created and the legacy 2000 instance is left intact. The drawback is that the data files for the various databases have to be migrated by the administrator manually. The new instance will also have to be configured manually before it can be put to production use.

From a summary of differences between an in-place upgrade and a side-by-side upgrade (Table 2), it is more advantageous, for our circumstances, to conduct a side-by-side upgrade. Reasons for this

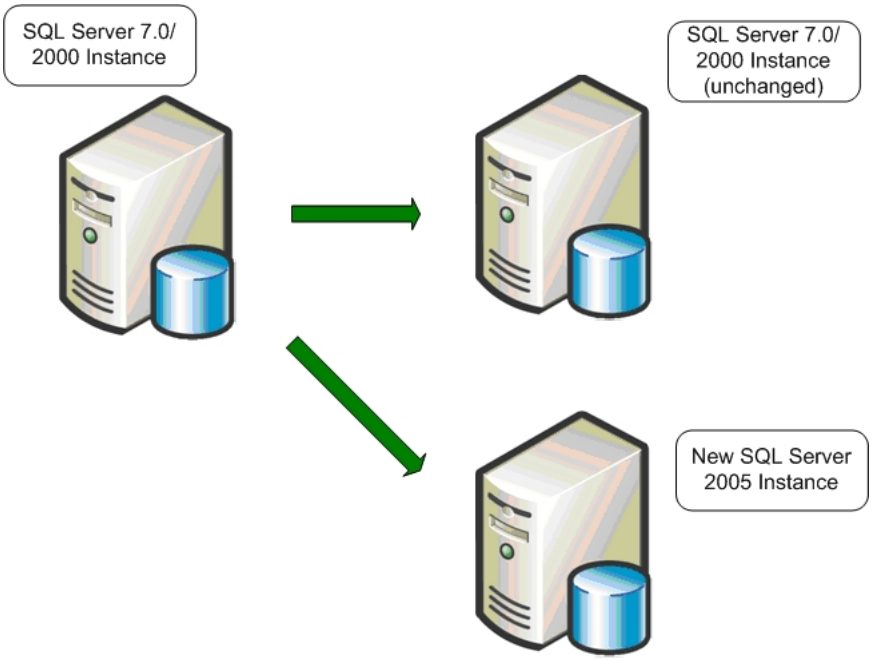

Figure 1: A side-by-side upgrade to a second server leaves the SQL Server 2000 instance untouched. [1]

include:

- Allows us to keep our 2000 instance live as me migrate our data to the new 2005 instance.
- Gain more granular control or what is to be upgraded.
- Conduct tests on the new instance without disturbing the production system.
- Easy to rollback to the old 2000 instance if the migration is not successful or fails certain critical tests that need to be resolved before an upgrade is performed.
- No additional metadata stored as there is when an in-place upgrade is conducted.
- Allows for changes in hardware between the upgrade.

Process	In-place Upgrade	Side-by-Side Upgrade
Number of resulting instances	One only	Two
Number of servers involved	One	One or more
Data file transfer	Automatic	Manual
Server instance configuration	Automatic	Manual
Supporting utility	SQL Server Setup	Various migration and data-transfer methods

Table 2: Summary Characteristics of an In-Place Upgrade and a Side-by-Side Upgrade. [1]

2 Features, Benefits and Advantages

2.1 Ease of Use

The following observations were made after using the developer edition of SQL Server 2005 for about a week.

2.1.1 SQL Server 2005 Management Studio

SQL Server 2005 Management Studio (Fig. 2.1.1) is brand new software that helps developers and database administrators alike to perform mundane tasks like query execution, backup and restore, instance management, report creation, etc. in a single organized place.

Previously, there were two programs to manage and work with the database: Enterprise Manager and Query Analyzer. Management Studio integrates the functionality of both softwares in one place, so developers need not have two programs running at the same time where one would suffice.

Furthermore, the user interface within Management Studio is more similar to that of Visual Studio's; this makes it more convenient for developers. The Studio allows SQL scripts and stored procedures to be checked-out directly from SourceSafe. The Studio provides exact line-number references to execution errors in stored procedures: something that was not available previously using Query Analyzer. The Studio also provides the ability to split the window both vertically and horizontally, enabling developers to look at two files simultaneously.

In all, SQL Server 2005 Management Studio seems to be a well-designed tool geared toward im-

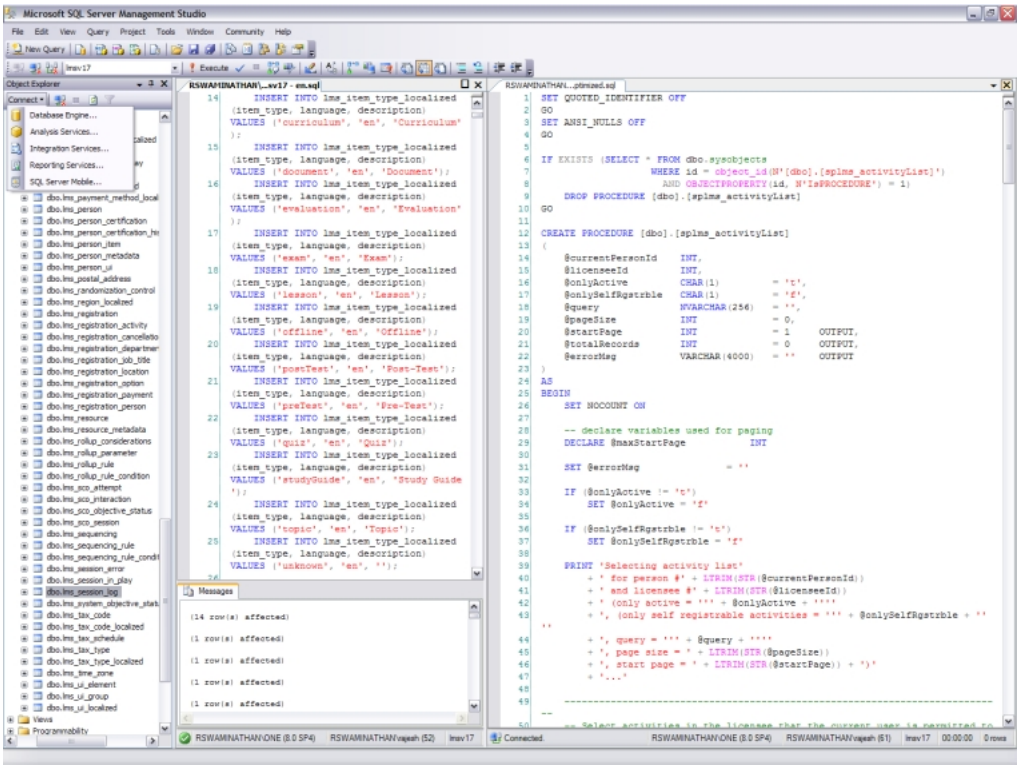

Figure 2: SQL Server 2005 Management Studio

proving developer efficiency and reducing development time.

2.1.2 SQL Prompt Plugin

SQL Prompt, an intellisense-equipped auto complete tool that all our developers are equipped with, works much better on a 2005 instance. Helpful features include:

- Significantly increased speed and accuracy
- Keystrokes much closer to Visual Studio
- Fewer prompts for authentication
- Improved JOIN support
- Support for table name auto completion. This did not work previously on a 2000 instance.
- More intelligent support for column and table *aliases*. Automatic join conditions are provided

based on the table alias (and not the table name itself), and `ORDER BY` clauses get auto completed using any column aliases defined earlier in the SQL statement.

Auto-completion is a great feature that helps developers save a lot of time while trying out queries on Management Studio, especially when there are a lot of tables involved with very long column names.

2.2 Stored Procedures in C#

SQL Server 2005 bundles enhancements to Transact-SQL (T-SQL) and the common language runtime (CLR). [3] This enables developers to write functions and stored procedures in C#, which is a much more versatile and powerful language than T-SQL. The components can later be compiled as stand-alone assemblies and used as SQL Server functions or stored procedures. This enables us to push more complex business logic into stored procedures (such as recursion, for example[1]) and have them execute quickly and more efficiently. Doing so allows us to exploit any idle process time on the database server, and alleviate stress on the web server.

2.3 Encryption

SQL Server 2005 allows the internal file-system to be encrypted. This was not an option on SQL Server 2000. However, turning this feature on in a 2005 instance will require consideration of the implications on performance during the encrypt/decrypt process.

2.4 Performance

The search grid problem involving pagination of a result set is a good example to demonstrate performance gains of stored procedures that are used extensively throughout our application. The currently implemented solution is extremely complicated, slow, and highly sensitive to even minor changes.

[1] Even if recursion is supported by T-SQL, there may be a limit to the number of *levels of recursion allowed.*

On other popular relational database management systems such as MySQL and PostgreSQL, the pagination problem is solved quite simply by allowing two parameters to be passed to the `LIMIT` clause. A `LIMIT` clause with just one argument is identical to the `TOP` x clause on SQL Server, allowing only the top x results to be returned. However, the `LIMIT` clause on MySQL/PostgreSql systems is much more versatile: if provided with two arguments, say x and y, it will return exactly y records *starting from the* x^{th} *record*. The `LIMIT` clause is unfortunately not a Transact-SQL standard and is hence not available on SQL Server.

An investigation provided insight that SQL Server 2005 introduces the new `ROW_NUMBER()` summary function to solve the pagination problem quite intuitively. This built-in function computes, after executing the query, the row number of the record in the result set. This enables us to grab say, 10 records from say, the 2nd *page*, where each page consists of 10 records. This can be achieved by asking for all rows whose *row number* is between 11 and 20 and so on. The field calculated by `ROW_NUMBER()` can be treated like any other field, so this allows us to use variables to specify what the row number must be between as the following example demonstrates.

Say our query looked like this:

```
SELECT col1, ROW_NUMBER() OVER (ORDER BY col1) AS rank FROM t1
```

This would return the entire result set with the second field indicating the row number of that record. So in reality, the `ROW_NUMBER()` function acts as an indexer that provides a counter as to which record we're currently running on. In this example, I have aliased the output returned by `ROW_NUMBER()` with `rank`. The `ROW_NUMBER()` function also accepts as mandatory attribute, the `ORDER BY` clause associated with the query.

Let us suppose a result set contains 100 records. Let us also suppose our application requires each *page* in the result set to contain 20 records to yield 5 pages in total. We can restrict the result set to the 3rd page (records 41 to 60) by simply specifying a `WHERE` clause like so:

```
WHERE rank BETWEEN 41 AND 60
```

We already have two variables @pageSize and @startPage defined within our search grid stored

procedures. @pageSize is the number of records per page, while @startPage is the current page number we're on.

Making use of these variables, pagination simply boils down to

```
WHERE rank BETWEEN (@startPage * @pageSize) + 1 AND @pageSize * (@startPage + 1)
```

assuming @startPage starts with zero.

An important note to keep in mind is that since the `ROW_NUMBER()` function is not computed until the query is executed, we can not include it in the `SELECT` clause and then immediately reference it in a following `WHERE` clause. This is because all of the `WHERE` clauses need to be evaluated *before* calculating the `ROW_NUMBER` for a record. This is to say that the additional cells returned by `ROW_NUMBER()` are *calculated cells*, and do not get calculated until very late in the query execution stack.

We can work around this by wrapping our query as a sub-query and including the final `WHERE` clause that restricts the record set to a specific page in the outer `SELECT` like so:

```
SELECT col1, rank FROM
( SELECT col1, ROW_NUMBER() OVER (ORDER BY col1) AS rank FROM t1 ) i
WHERE rank BETWEEN ...
```

The stored procedure `splms_activityList.sql` retrieves all courses within the system for a given client for use in a pagination-enabled search grid. Taking advantage of the `ROW_NUMBER()` function enabled us to remove around 200 lines of code from this stored procedure that were in use to manually mimic pagination.

The stored procedure was to be analyzed both under the 2000 as well as the 2005 instances without optimization first, with and without a search query. Following this, the above optimization was to be applied to the stored procedure and re-analyzed under the 2005 instance with and without a search query.

Unfortunately, due to time constraints, the benchmark could be not performed by the end of the

work term. However, the code needed to benchmark the old and new stored procedures is included in Appendix A. This might provide an opportunity for a new co-op student to investigate the performance gains.

2.5 Reporting Services

We already use Reporting Services to provide metrics for students and administrators. In SQL Server 2005, the entire reporting infrastructure has been revamped due to a large number of complaints from users.

The new reporting capabilities in SQL Server 2005 promise [1]:

- Faster report development.
- More flexible interaction.
- Timely information availability and alerting.

Reporting Services 2005 includes an improved report builder, aimed at simplifying report generation. The new version allows the report designer to add multi-value as well as hierarchical parameters, so that category-based boundaries such as licensee privileges, location levels, etc. can be easily represented and selected within a report. Furthermore, when data grids are displayed on the Web, they can be sorted by any column. This is not currently automated in Reporting Services 2000.

The greatest advantage with reporting in the 2005 version is the built-in access to Analysis Services 2005. This means that each report generated has access to the analysis services. Hard-coded query statements in SQL Server 2000 Reporting Services can now be built dynamically, and on-the-fly. This connectivity allows "filtering, pivoting and highlighting" within reports. [2]

3 Issues

3.1 ORDER BY Prefixes

Microsoft's upgrade preparedness utility tool, described in Section 3.4, was run against a copy of our development database (GLACIER). The following issue with the `ORDER BY` clause was discovered.

> Column aliases in an `ORDER BY` clause cannot be prefixed by the table alias.

This functionality, although available in SQL Server 2000 does not parse under 2005. A possible reason for this is because column aliases can reference several tables simultaneously and hence it makes no sense to include a table reference within the `ORDER BY` clause.

For example

```
SELECT t1.col1 AS somecol FROM table1 AS t1 ORDER BY t1.somecol

SELECT (t1.col1 + t1.col2) AS somecol FROM table1 AS t1 ORDER BY t1.somecol

SELECT (t1.col1 + t1.col2) AS somecol FROM table1 AS t1 ORDER BY t1.somecol

SELECT (t1.col1 + t2.col2) AS somecol FROM table1 AS t1 JOIN table2
AS t2 ON (join_condition) ORDER BY t1.somecol
```

are all acceptable in the 2000 instance, but not under a 2005 instance. The database engine does not match `t1.somecol` in the `ORDER BY` clause to a valid column in the table named `table1` (aliased `t1`).

Upgrade Advisor identified this problem to occur in five different objects (stored procedures) within our application at the time of writing:

1. splms_departmentList.sql
2. splms_jobTitleList.sql
3. splms_locationList.sql
4. splms_reportStartsCompletes.sql

5. splms_trainingRoot.sql

However, none of these stored procedures return errors when attempted to run under a SQL Server 2005 instance. The help page associated with this error documents the following exception.

Exception

If the prefixed column alias specified in the `ORDER BY` clause happens to be a valid column name in the specified table, the query executes without error. In the above example, if `somecol` was also a column of `table1` (aliased `t1`), the query would execute without error. These kind of ambiguities are hard to track.

Another semantic error that might occur is illustrated in the following example: [8]

> For example, the column alias `id` specified in the following statement is a valid column name in the `sysobjects` table. In SQL Server 2000, when the statement executes, the `CAST` operation is performed after the result set is sorted. This means the name column is used in the sort operation. In SQL Server 2005, the `CAST` operation occurs before the sort operation. This means the `id` column in the table is used in the sort operation and returns the result set in an unexpected order.

```
SELECT CAST (o.name AS char(128)) AS id
FROM sysobjects AS o
ORDER BY o.id;
```

Corrective Action

This ambiguity can be corrected by modifying all queries that use column aliases prefixed by table aliases in the `ORDER BY` clause in one of the two following ways:

- Remove the prefix in the `ORDER BY` clause.

- Replace the column alias with the column name.

For example, both of the following queries execute without error in SQL Server 2005:

```
SELECT t1.col1 AS somecol FROM table1 AS t1 ORDER BY somecol
SELECT t1.col1 AS somecol FROM table1 AS t1 ORDER BY t1.col1
```

3.2 GROUP BY Expressions

This problem was identified during development [4], and is a change worthy of keeping in mind while designing queries using aggregate grouping using the GROUP BY clause.

Quoting from the MSDN transact-SQL reference for GROUP BY [5]:

> When GROUP BY is specified, either each column in any non-aggregate expression in the select list should be included in the GROUP BY list, or the GROUP BY expression must exactly match the select list expression.

This is to say that if we have an arbitrarily complex expression involving multiple columns in the SELECT clause, one of the two must occur:

- each of the columns used in the SELECT clause must be included in the GROUP BY clause separated by commas. The order does not matter.
- The identical select list expression must be included in the GROUP BY clause.

SQL Server 2005 allows a very long expression to be split up into multiple expressions, and any of these expression can be in the GROUP BY clause. This would mean that the entire SELECT expression need not occur within the GROUP BY clause — a requirement in SQL Server 2000.

3.3 Backwards Compatibility

Backwards compatibility, fortunately, is not an issue for us as we host our own application. We do not have clients who use our application but host it on their own database server. Consequently,

changing our code to suit SQL Server 2005 standards will not affect any installation, as all installation instances will be upgraded simultaneously. We therefore do not have to coerce a client into using the latest version of SQL Server if they want to avail themselves of the latest application changes.

3.4 Upgrade Advisor

Microsoft SQL Server 2005 Upgrade Advisor [6] is a tool that helps isolate potential issues and provides guidelines on where to begin while conducting post-upgrade testing. It also provides a checklist of changed features and functionality that need to be considered if our application use them. Upgrade Advisor uses a rules-based system, so it simply scans the 2000 instance, and optionally, any trace files generated by SQL Profiler, stored procedures, functions, and maintenance scripts to determine issues to fix before upgrading. Further, constantly updated rules are available online as organizations can choose to share their results with Microsoft.

Unfortunately, running the upgrade advisor—as seen in Figure 3—against a copy of the current development database yielded only one useful issue.[2] This was described in Section 3.1.

The Upgrade Advisor additionally points to a help file[3] that recommends that indexes are dropped and recreated after an upgrade if the data is migrated manually using a backup-and-restore procedure. This is because some of the indexes might be disabled after the upgrade process.

[2]This is a good indication that our stored procedures, functions and maintenance scripts are cleanly written and forwards-compatible.

[3]The help file contains upgrade issues that cannot be detected or the detection of the issue would result in too many false-positive results.

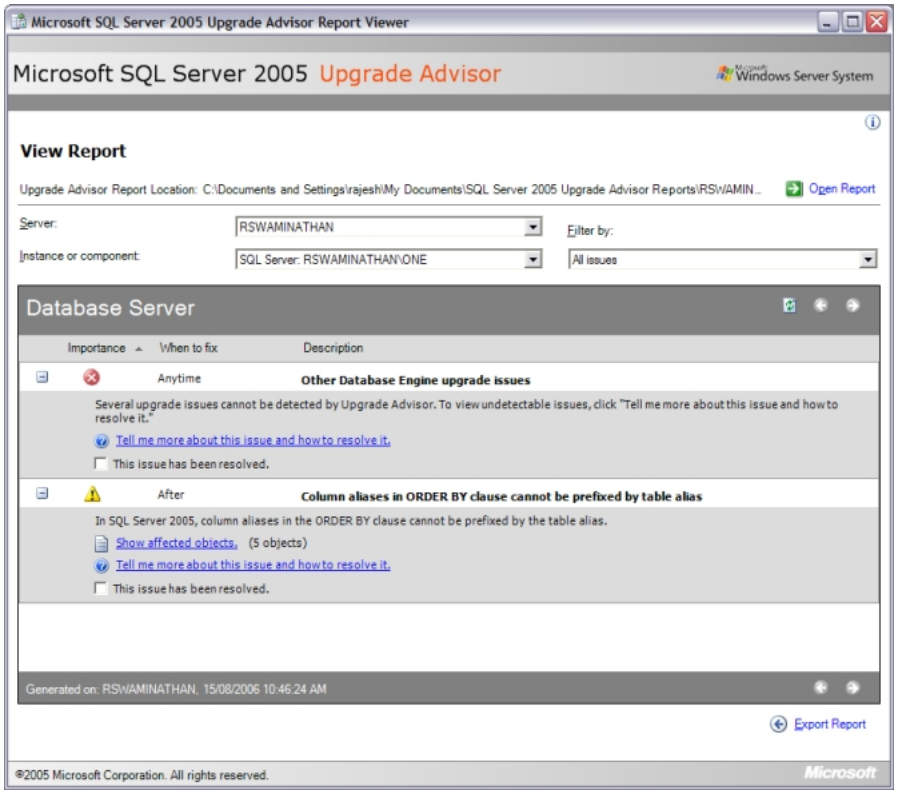

Figure 3: Upgrade Advisor displays unresolved issues with the current 2000 instance.

4 Concluding Summary

In all, SQL Server 2005 is definitely an advantageous upgrade that will help take our application's as well as our developers' abilities to the next level. SQL Server 2005's integrated tools, faster performance, more reliability and availability, and an easy upgrade make for some compelling reasons to recommend an upgrade.

Some of the mentionable benefits of upgrading to SQL Server 2005 are:

- SQL server 2005 management studio for developer productivity
- SQL prompt plugin enhancements
- C# stored procedures
- Stored procedure optimization of search grids
- Reporting services enhancements

Certain issues need to be kept in mind while considering an upgrade:

- Unsupported `ORDER BY` table prefixes
- Unsupported `GROUP BY` expressions
- Issues relating to backwards compatibility

Further potential issues may be discovered by browsing through the provided SQL Server Upgrade Advisor's documentation.

Due to restrictions on the size of this report, a consideration of the cost of upgrade—in terms of time and money—was not possible.

A `splms_activityList` Benchmark SQL Script

```
1  DECLARE @startDT DATETIME
2  DECLARE @endDT   DATETIME
3
4  DECLARE @startReads INT
5  DECLARE @endReads   INT
6  DECLARE @firstRead  INT
7  DECLARE @secondRead INT
8  DECLARE @totalReads INT
9
10 DECLARE @firstMS  INT
11 DECLARE @secondMS INT
12 DECLARE @totalMS  INT
13
14 -- without search query
15 EXEC sp_recompile splms_activityList
16 SET @startReads = @@TOTAL_READ
17 SET @startDT = GETDATE();
18 EXEC splms_activityList 6065, 57
19 SET @endDT = GETDATE();
20 SET @endReads = @@TOTAL_READ
21
22 SET @firstMS = DATEDIFF(ms, @startDT, @endDT)
23 SET @firstRead = @endReads - @startReads
24
25 EXEC sp_recompile splms_activityList
26 SET @startReads = @@TOTAL_READ
27 SET @startDT = GETDATE();
28
29 -- with search query
30 EXEC splms_activityList 6065,
```

```
                        57,
                        DEFAULT,
                        DEFAULT,
                        DEFAULT,
                        DEFAULT,
                        DEFAULT,
                        DEFAULT,
                        'hello',
                        10,
                        1

SET @endDT = GETDATE();
SET @endReads = @@TOTAL_READ

SET @secondMS = DATEDIFF(ms, @startDT, @endDT)
SET @secondRead = @endReads - @startReads

SET @totalMS    = @firstMS   + @secondMS
SET @totalReads = @firstRead + @secondRead

-- stat display
SELECT '@S.No', '@Property', '@Duration', '@Reads'
 UNION SELECT '1', 'Start to Tick1', CAST(@firstMS  AS CHAR), CAST(@firstRead AS CHAR)
 UNION SELECT '2', 'Tick1 to End',   CAST(@secondMS AS CHAR), CAST(@secondRead AS CHAR)
 UNION SELECT '3', '------------',   '',                      ''
 UNION SELECT '4', 'TOTAL',          CAST(@totalMS  AS CHAR), CAST(@totalReads AS CHAR)
```

References

[1] R. Dyess, D. Fackler, M. Hotek et al. "Upgrade Technical Resource Guide." *Microsoft Publishing*. Jun 2006.

[2] The Business Value of Upgrading to SQL Server 2005. *Navigant Consulting, Inc.* December 2005.

[3] Why Upgrade to SQL Server 2005 Business Intelligence: A Technical Overview. *Microsoft Publishing*. Nov 2005.

[4] Personal conversation with Stefan Leyhane, upon execution error of a stored procedure on a 2000 instance but not on a 2005 instance. August 14 2006.

[5] Microsoft Developer Documentation for GROUP BY (Transact-SQL). *Microsoft Developer Network*. http://msdn2.microsoft.com/en-us/library/ms177673.aspx

[6] SQL Server 2005 Upgrade Advisor. Upgrade Advisor analyzes instances of SQL Server 2000 to help developers prepare for upgrades to SQL Server 2005. http://www.microsoft.com/downloads/details.aspx?familyid=1470E86B-7E05-4322-A677-95AB44F12D75.

[7] SQL Server 2005 Upgrade Handbook. Douglas McDowell, Erik Veerman, and Michael Otey. *Solid Quality Learning*. November 2005. http://www.microsoft.com/technet/prodtechnol/sql/2005/sqlupgrd.mspx

[8] SQL Server 2005 Upgrade Advisor Compiled Help File (CHM). The help document comes along with the installation of Upgrade Advisor.