# C:\Users\v-shtoly.REDMOND.000\Pictures\SQL\SQL_logo.png

**Using Star Join and Few-Outer-Row Optimizations to Improve Data Warehousing Queries**

**SQL Server Technical Article**

**Writers:** Dayong Gu (Microsoft), Ashit Gosalia (Microsoft), Wei Liu (NetApp), Vinay Kulkarni (IBM)

**Technical Reviewer:** Susan Price (Microsoft)

**Published:** August 2009; Updated: December 2010

Applies To: SQL Server 2008, SQL Server 2008 R2

**Summary:** In this white paper we discuss two of the new features introduced in SQL Server 2008, Star Join and Few-Outer-Row optimizations. These two features are in SQL Server 2008 R2 as well. We test the performance of SQL Server 2008 on a set of complex data warehouse queries designed to highlight the effect of these two features and observed a significant performance gain over SQL Server 2005 (without these two features). The results observed also apply to SQL Server 2008 R2. On average, about 75 percent of the query execution time has been reduced, compared to SQL Server 2005. We also include data that shows a reduction in the number of rows processed and improved balance in parallel queries, both of which highlight the important role the Star Join and Few Outer-Row features played.

# Introduction

More and more enterprises and organizations rely on data warehouses to make insightful decisions with knowledge obtained from tremendous amounts of data. As large warehouses become more and more widely used, database customers begin to have high expectations that their database system providers will improve the performance of data warehousing by focusing on specific aspects of data warehousing queries. Contrary to online transaction processing (OLTP) applications, there are new challenges in data warehousing applications, such as:

* Large amounts of data being accessed mean long query execution times and high query response times from a customer perspective.
* Smaller maintenance windows are available to keep the data in warehouses up-to-date
* Highly efficient storage solutions are generally required to make sure the points above are handled effectively.

In the Microsoft SQL Server 2008 R2 database software, two optimizations, called *Star Join* and *Few-Outer-Row*, are specifically for data warehousing applications. These two optimizations focus on improving query execution speed and thus reducing the query response time.

To look at how these optimizations improve the performance of data warehousing queries, we selected a large-scale data warehousing database and a set of complex decision support queries that cover the common-use cases in data warehousing applications while accessing large amounts of data. We compared these queries on SQL Server 2008, which has the Star Join and Few-Outer-Row features, to SQL Server 2005, which did not have these features. Our tests showed that using SQL Server 2008 with the Star Join and Few-Outer-Row optimizations reduced the query execution times on average 75 percent compared to executing the same set of queries on SQL Server 2005.

In our experiments, we ran both versions of SQL Server on an IBM System x3950 M2 x64 32-core server connected via multiple 4Gb Fibre Channel (FC) interconnects to NetApp® FAS3070 storage controllers. We chose this testing platform to meet the performance and scalability requirement of the workload of the queries on our large-scale data warehouse.

The remainder of this paper introduces the special aspects and challenges related to data warehousing applications. Additionally, we provide the specifics of the test platforms and methodology used during the tests. Finally, we present the results of our tests and draw some conclusions about the impact of the Star Join and Few-Outer-Row optimizations on data warehouse performance.

# Background

## Data Warehousing Challenges

Many relational data warehouses follow a dimensional modeling design style. The most well-known data warehouse schemas include the so-called star schema and the snowflake schema, which is actually related to or considered to be a variation of star schema.

In dimensional modeling, tables are categorized into two types, fact tables and dimension tables. Fact tables store the transactional data, such as the sales, orders, and inventories in a database designed for retail business. Dimension tables store entity information, such as product, customers, date and time, and so on.

In general, a dimension table is comparatively small, while a fact table can contain tremendous amounts of data. Billions of rows are normal for a fact table in a data warehouse. One problem associated with fact tables is the need for multicolumn composite keys to be repeated for each row in the table. In order to reduce this space cost, a common practice in dimensional modeling is to use smaller surrogate keys to implement the relationship between a fact table and its dimensions. The fact table refers to the surrogate key as opposed to the business key of the dimension table. Therefore, dimension tables are connected to fact tables by these surrogate key relationships. This design schema can significantly reduce the space cost associated with large fact tables. In a star schema, multiple dimension tables are grouped around one fact table. It is also a common practice to spread several dimension tables over multiple levels; this is generally referred to as the snowflake schema.

For decision support queries in a snowflake or star schema, the largest workload is usually fact table processing. In most cases, it involves complex join operations between a fact table and a set of dimension tables. In SQL Server 2008 R2, we present two optimizations: *Star Join* and *Few-Outer-Row*, focused on improving join performance in different ways:

* The Star Join optimization significantly reduces the number of rows to be processed during a query. With Star Join, many rows in the fact table that will be eliminate d during the joins with the dimension tables are identified early in query execution. These rows are eliminated from further processing, resulting in significantly reduced processing times.
* The Few-Outer-Row optimization aims to reduce the imbalance in nested loop join execution and improve the level of parallelism achieved during query execution.

The following sections provide more details regarding the Star Join and Few-Outer-Rows optimizations.

## Star Join Optimization

Many data warehousing queries share a common pattern. They select several measures from a fact table, join the fact rows with one or more dimensions along the surrogate keys, and place filter predicates and aggregates on non-key columns of the dimension tables. You can think of a fact table as the star in the center of a solar system with a number of dimension tables orbiting like planets around the star. We refer to this pattern as star pattern. Star Join is a dedicated optimization designed specifically for queries on dimensionally modeled data configured into a star pattern. The first step of the optimization is to detect the star pattern depending on heuristics, such as:

* The largest table that participates in an n-ary join is considered as the fact table.
* To be considered as a fact table, a table must be larger than a specified minimum size.
* All join conditions of each of the binary joins have to be single column equality predicates.
* The joins have to be inner joins, and so on.
* By using these heuristics, the query optimizer of SQL Server 2008 R2 detects the star pattern and identifies the fact table automatically. The query optimizer then builds hash tables for each dimension table that participates. Based on these hash tables, the query optimizer builds bitmap filters to push down the scan on the fact table. These filters effectively eliminate most of the rows that would be removed by later joining actions. As a result, the total number of rows that need to be processed by subsequent operators is greatly reduced, thereby providing a significant performance improvement.

## Few-Outer-Row Optimization

Few-Outer-Row is a specific optimization for nested loop joins. In some data warehousing queries, the outer side of a nested loop join is a parallel scan with a filter. In the case that the qualified data is only a few rows and clustered on the same page, they are selected by a single thread in SQL Server 2005. Consequently, all the work for the nested loop join is done by a single thread. This situation causes an imbalance issue in any system that affords a degree of parallelism (DOP) larger than one, resulting in long query times and high latencies.

SQL Server 2008 R2 detects this pattern of nested loop joins, and it introduces an exchange operator above the outer side of the nested loop join that evenly redistributes the few outer rows among multiple threads. The Few-Outer-Row optimization balances work better and improves scalability significantly.

## NetApp FAS3070

NetApp fabric-attached storage (FAS) systems simplify data management, enabling enterprise customers to reduce costs and complexities, minimize risks, and control change. NetApp FAS systems are versatile storage systems that are often used for storage consolidation.

The FAS3070 addresses the core requirements of the midrange enterprise storage market, delivering a blend of price, performance, and scalability for SQL Server databases and business applications. The compact, modular design provides integrated FC SAN and IP SAN (iSCSI) storage with scalability to over 500 disks. The FAS3070 storage controller supports both FC and SATA disk drives for tiered storage. FAS3070 systems support as many as 32 FC ports or 32 Ethernet ports, including support for 2Gb and 4Gb FC and for 10 Gigabit Ethernet.

The FAS3070 runs the NetApp Data ONTAP® operating system, which is optimized for fast, efficient, and reliable data access and retention. Data ONTAP 7G is designed to simplify common storage provisioning and management operations. Logical unit numbers (LUNs) and volumes created and configured using FlexVol® technology can be dynamically expanded or contracted with a single command. FlexVol volumes also enable thin provisioning, which avoids the cost of overprovisioning and time-consuming reconfiguration. Host-based NetApp SnapDrive® extends this flexible storage provisioning capability to databases and applications. Another Data ONTAP 7G feature, FlexClone®, instantaneously creates cloned LUNs or volumes without requiring additional storage. FlexClone technology is designed to improve the effectiveness and productivity of application and database development and pre-deployment testing.

FAS hardware design and the Data ONTAP operating system are tightly integrated to provide resilient system operation and high data availability. FAS systems incorporate redundant and hot-swappable components and patented double-parity RAID-DP®. NetApp RAID-DP, a high-performance implementation of RAID 6, is designed to provide superior data protection with negligible impact on performance. The NetApp Snapshot™ technology provides up to 255 data-in-place, point-in-time images per LUN or file system, available for near-instantaneous file-level or full data set recovery. Host-based SnapManager® software integrates Snapshot management with applications, providing consistent backup images and application-level recovery in minutes. SnapMirror® uses Snapshot copies to provide incremental block-level synchronous and asynchronous replication; SnapVault® uses it for block-level incremental backups to another system. Together, these SnapSuite™ products can help in delivering the high application-level availability that enterprises require for 24×7 operation.

## IBM System x3950 M2

The IBM System x3950 M2 server is based on eX4, the 4th generation of the proven IBM Enterprise X-Architecture® chipset. The eX4 chipset, including the memory controller, provides the x3950 M2 server with:

* Ability to scale-up on processor and memory based on a modular “pay-as-you-grow” design.
* High performance as demonstrated by leadership TPC benchmark results.
* Mainframe-like reliability in an x86 environment.

IBM System x3950 M2 key features include:

* True 2–to–16-socket scalability up to 64 cores with the current operating systems from Microsoft
* Quad-Core Intel® Xeon® MP 7300 and six-core MP 7400 Series processors
* Up to 1 terabyte of registered DIMM memory for better workload density and less power consumption than servers using fully buffered DIMM technology
* IBM Memory ProteXion™ with redundant bit-steering, which offers high memory resiliency
* Fourth-generation snoop filter
* IBM Predictive Failure Analysis®, not just on hard disks and memory but also on processors, power supplies, fans, and voltage regulator modules
* Memory latency that is very low

# Experimental Environment

## Server and Storage Configuration

The IBM x3950 M2 consists of two physically separate chassis connected with two scalability port cables. There is an IBM ScaleXpander key in each chassis, which permits them to logically operate as a single SMP server. The total system has eight Intel quad core Xeon X7350 processors and 128GB memory. Six of the seven PCI-E slots in each chassis were populated with Emulex LP111002 and QLogic QLE2462 HBAs.

|  |
| --- |
| **IBM x3950 M2 Server Details** |
| Processors | 8 Intel Xeon X7350 2.93 GHz (Quad core) |
| Cores | 32 |
| Front-side bus frequency | 1,066 MHz |
| Memory | 128 GB (64 2 GB DIMMs) |
| PCI | 6 Emulex LP111002-M4 dual-port 4Gb/sec HBAs; 6 QLogic QLE2462 dual-port 4Gb/sec HBAs |

**Table 1:** Summary of IBM x3950 M2 configuration

NetApp storage controllers were used for SQL Server 2008 Decision Support Systems (DSS) databases and transaction logs. Figure 1 shows twelve NetApp FAS3070 storage controllers directly connected to the IBM x3950 M2 server via 6 Emulex LP111002-M4 Dual-port 4Gb/sec HBAs and 6 QLogic QLA2642 Dual-port 4Gb/sec HBAs. The result was two 4Gb/sec FC links connected from each FAS3070 storage controller to the IBM x3950 M2 server.

Also shown in Figure 1 are 8 LUNs (250 GB each) per FAS3070 storage controller, for a total of 96 LUNs mounted to the IBM x3950 M2 server for DSS databases and transaction logs.

The FAS3070 storage controllers ran version 7.3 of Data ONTAP. Each FAS3070 was configured with forty-two 300 GB 15K RPM FC disks.



**Figure 1:** Server and storage connection via 4 Gb/sec Fibre Channel interconnects

**Note:** The FAS3070 storage systems described in this paper are used by the Microsoft SQL Server Performance Engineering Team for multiple parallel projects, and they were configured for maximum testing flexibility rather than sized for this particular test. As a result, the tested configuration can provide more I/O throughput and storage capacity than was used during this test. This design allowed the storage to be provided in such a way that no bottlenecks would be associated with the storage system or the interconnects to that storage. This means that more controllers and more disks were provided than would otherwise have been necessary to support the load during the test.

Data ONTAP 7G introduced storage virtualization features that can simplify storage provisioning and improve storage use, flexibility, and manageability. Two of the key concepts are aggregates and FlexVol volumes. An aggregate is a logical container for pools of physical disks that are organized into one or more RAID-DP groups. An aggregate is the logical layer that decouples volumes from the underlying physical storage. A FlexVol volume is a logical entity that is separated from the physical storage that contains the associated data. One or more FlexVol volumes can reside within an aggregate. A FlexVol volume can grow or shrink in size, constrained only by the hard limits of the aggregate size or the soft limits set when the volume was first created. Each FlexVol volume leverages the performance of all of the disks in the aggregate.



**Figure 2:** FAS 3070 configuration

Figure 2 shows a detailed configuration of one of the FAS3070 storage controllers. Note that all 12 controllers are configured in identical fashion.

A total of 40 of the 42 disks within each controller were used to form 2 aggregates, each using 20 disks. The remaining 2 disks were left as hot swappable spares. In Figure 3, 2 disks within each aggregate are darker gray than the others. These 2 disks are the 2 parity disks within the RAID-DP group, which protect against double disk failure.

One FlexVol volume was then created within each aggregate, resulting in 2 volumes per controller. Note also that residing in aggr0 is an additional volume, vol0, which is needed by Data ONTAP for controller management.

Within each FlexVol volume, 4 LUNs (250 GB) were created. Thus 4 LUNs per aggregate and 8 LUNs per controller (96 LUNs total on all 12 controllers) were created and then mapped to the IBM x3950 M2 server for storing DSS databases and transaction logs.

Figure 2 also illustrates that 4 LUNs residing within an aggregate were physically connected to the IBM x3950 M2 server using one 4 Gb/sec FC port. Therefore 2 ports per controller were used in this test environment. Table 2 summarizes the storage configuration information.

|  |  |
| --- | --- |
| Storage Entity | Quantity |
| NetApp FAS3070 controllers | 12 |
| 300GB 15K 4Gb/sec FC disks | 504 (42 per controller) |
| Aggregates | 24 (2 per controller) |
| FlexVol volumes | 24 (2 per controller, 1 per aggregate) |
| LUNs | 96 (8 per controller, 4 per aggregate, 4 per volume) |
| 4Gb/sec FC links | 24 (2 per controller, 1 per aggregate) |

**Table 2:** Summary of NetApp storage used in the 1-terabyte data warehouse

## SQL Server Database

On the same testing environment, we ran a set of typical data warehousing queries on SQL Server 2005 and SQL Server 2008 respectively. For each SQL Server version, we ran the same set of queries in three different modes:

* Continuous run
In this mode, we ran the queries continuously in a sequence. After the completion of the previous query execution, we began the next query immediately. In this mode, the performance of a query was sometimes impacted by a previous query to some extent. However, we believe this best simulated the real life database application.
* Cold run
In this mode, before running a query, we cleaned up all the cached data in SQL Server and thus provided a clean environment for the next query as if it were the first query running on SQL Server. In this case, SQL Server read all the data required by the query from the storage system. There was some cold-start penalty, and the I/O subsystem played an important role for the overall performance.
* Warm run
In this mode, each query was run twice. After the first run of a query, we began a second run immediately. We took the data of the second run as warm run data. In the warm run, all cacheable data for the query had already been brought into the system by the first run and had not been polluted by other queries. In other words, the query was executed after the database caches had been warmed up. In this mode, to a large extent, the impact of I/O subsystem on overall performance was reduced.

We believe these three query execution modes provide good coverage over most the common cases in database applications and running the queries multiple times provides a more realistic measure of performance compared to running only a single instance of each query.

## Testing Queries

On the same testing environment, we ran a set of typical data warehousing queries on SQL Server 2005 and SQL Server 2008 respectively. For each SQL Server version, we ran the same set of queries in three different modes:

For the tests, we ran eleven typical decision support queries over a 1-terabyte size database to evaluate the impact of the Star Join and Few-Outer-Row features on performance. We chose these queries for their complexity, diversity, and relationship to data warehousing applications. All of them involve a large amount of data processing. Descriptions of the queries are as follows:

**Query A**

This query lists a set of attributes of the products, such as brand, quantity, price, coupon used, and so on bought by a specific category of customers during a specific time period.

**Query B**

This query summarizes the price of the Internet-based purchases of a specific set of products by customers in a specific group of geographic areas during a specific time period.

**Query C**

This query collects the information of returned items in several fixed length time periods for all stores.

**Query D**

This query selects the top items in each quarter of a specific year belonging to a specific set of categories.

**Query E**

This query computes the total revenue from a specific set of products, and categorizes the revenue in $50 segments.

**Query F**

This query summarizes the monthly sales achievement made by a manager on a specific set of products.

**Query G**

This query identifies customers who have a strongly preferred way of shopping.

**Query H**

This query obtains the sale records of a specific store in each half hour period.

**Query I**

This query shows the per-month sale record for the best-selling products in specific categories.

**Query J**

This query gets the sale record of a specific store in a specific time slot where the density of stores in the same area reaches a given level.

**Query K**

This query selects the record of top warehouses where the ship date and sale date are within several fixed-length time periods in a specific year.

These eleven queries are either qualified for applying Star Join or Few-Outer-Row, or both. We show the details in Table 3. For the queries qualified for Star Join or Few-Outer-Row, we put an asterisk in the cell on the corresponding line.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Feature | Qry A | Qry B | Qry C | Qry D | Qry E | Qry F | Qry G | Qry H | Qry I | Qry J | Qry K |
| Star Join | \* | \* |  | \* | \* | \* | \* | \* | \* | \* | \* |
| Few –Outer Row | \* | \* | \* | \* | \* | \* |  |  | \* |  |  |

**Table 3:** Query/feature matrix

### Result Discussion

## Overall Performance

We normalized the execution time of each query in continuous, cold, and warm run in SQL Server 2008 to that of SQL Server 2005, as demonstrated in Figure 3. In this graph, a value of 10% indicates that SQL Server 2008 takes only 10 percent of the execution time of SQL Server 2005 to execute the same query. We obtain significant improvements on all queries for all three of the execution modes:

* SQL Server 2008 reduces execution time at least 40 percent for all the queries.
* On six queries, SQL Server 2008 reduces the execution time by 80 percent.
* On four queries, A, C, D, and F, SQL Server 2008 reduces the execution time by 90 percent, which means a tenfold performance improvement.
* On average, SQL Server 2008 reduces the execution time by about 75 percent compared to SQL Server 2005.

Overall, we obtained a significant performance gain over SQL Server 2005, which has no Star Join or Few-Outer-Row optimizations. Moreover, to measure the impact that these two new features had in improving the overall performance, we collected statistical data on the number of rows and the total balance level in query execution. These two sets of data show the actual impact of the two features clearly. We discuss them in detail in the coming sections.

The following figure shows the execution time of continuous, cold, and warm runs of 11 queries of SQL Server 2008 normalized to continuous, cold, and warm runs of the queries on SQL Server 2005.



**Figure 3:** Normalized execution time

In this figure, we assume the execution time in SQL Server 2005 is 100 percent. A value such as “5.4%” means, on that query, in that execution mode, the execution time of SQL Server 2008 is only 5.4 percent of that of SQL Server 2005, which is an almost twentyfold improvement.

## Star Join Result Discussion

We detected the Star Join pattern on 10 out of 11 queries. The lone exception was Query C. The Star Join optimization is designed to improve the performance by reducing the number of rows processed in query execution. To prove that the Star Join optimization has actually been performed and really reduces the number of rows, we normalized the sum of the number of rows for each node in a SQL Server 2008 plan to that of SQL Server 2005 for all ten queries with the Star Join pattern. The result is illustrated in Figure 4.

A great portion of rows that need to be processed has been reduced in most queries:

* On 6 queries, we reduced the number of rows to less than 10 percent of the same measure in SQL Server 2005!
* We also achieved a significant row reduction ratio on queries G, I, and K. At least 40 percent of the rows were reduced.
* Only on Query B, we did not successfully reduce the number of rows (obviously).

These results confirm that Star Join optimization does play an important role in the performance gain of SQL Server 2008 over SQL Server 2005. A large number of rows were reduced by utilizing Star Join optimization in the majority of the queries used for this test.

The following figure shows the number of rows processed in SQL Server 2008 for the queries with Star Join patterns, normalized to that of SQL Server 2005.



**Figure 4:** Normalized number of rows

In this figure, we assume the number of rows processed on SQL Server 2005 is 100 percent. A value such as “1.07%” means, on that query, the number of rows that need to be processed on SQL Server 2008 is only 1/100 of that on SQL Server 2005.

## Few-Outer Row Result Discussion

We detected the Few-Outer-Row pattern on 7 out of these 11 queries. Few-Outer-Row improves the performance by balancing the work done by parallel threads and hence making full use of the available parallelism of the test platform. To understand whether the Few-Outer-Row feature has been performed and how much impact it can have on query execution, we needed a way to measure the overall balance of query execution.

For this purpose, we defined a metric named the Imbalance Score (IB) to quantitatively evaluate the level of balance in query execution. The definition of IB is as follows:

$$IB=\frac{\sum\_{i=1}^{n}Max\left(Actual rows on each Thread on NODE i\right)}{\sum\_{i=1}^{n}Average Number of Rows on each Thread on NODE i}$$

In layman’s terms, for each node in a query execution plan, we first measure the maximum and average of the number of rows processed by each thread. (All of our experiments used DOP=32. We compute the average by dividing the total number of rows on a node by 32.) We then compute the sum of the maximum number of rows for all nodes, which gives us an estimate of the length of the longest path in the plan. We also compute the sum of the average of number of rows for each node, which gives us an estimate of the length of the longest path in the ideal case. Therefore, IB is a ratio of the actual overall balance level of the whole plan to the ideal case for the same plan. A higher IB means greater imbalance issues in the query execution. We then normalize the IB for SQL Server 2008 plans to that of SQL Server 2005 for all the queries that have Few-Outer-Row patterns.

As shown in Figure 5, we reduce the imbalance on all the queries significantly. In the worst case, Query E, we reduce the IB about 35 percent. For all other queries, we reduce the IB to less than 16 percent of SQL Server 2005, which indicates that the Few-Outer-Row feature significantly increases the balance during query execution. Query C is one of the queries where we achieved a significant reduction in IB ratio. Recall that the Star Join optimization is not applied on this query so the performance gain is primarily coming from a result of the Few-Outer-Row optimization.

The following figure shows the measure of imbalance (IB) in SQL Server 2008 for queries with the Few Outer Row pattern, normalized to that of SQL Server 2005.



**Figure 5:** Normalized SQL Server 2008 IB

In this figure, we assume the imbalance (IB) measure on SQL Server 2005 is 100 percent. A value such as “5.00%” means, on that query, the value of the imbalance measure on SQL Server 2008 is only about 1/20 of that on SQL Server 2005.

# Conclusion

In SQL Server 2008, and SQL Server 2008 R2 by extension, the Star Join and Few-Outer-Row features were introduced to improve performance of data warehousing applications. Our experimental results show a great performance improvement in SQL Server 2008 / SQL Server 2008 R2 over SQL Server 2005 on a set of complex yet typical data warehousing queries. We also show two sets of data to demonstrate the effectiveness of both the new features in query execution.

Star Join significantly reduces the number of rows processed, while Few-Outer-Row effectively solves the imbalance issue in query execution. Together they contribute to the large overall performance improvement.

The selection of the NetApp FAS3070 storage system and the IBM x3950 M2 server provides us with an appropriate environment for the workload and reduce both I/O and CPU-bound bias, which is essential for the reliability of the performance result.

**For more information:**

<http://www.microsoft.com/sqlserver/>: SQL Server Web site

SQL Server TechCenter: <http://technet.microsoft.com/en-us/sqlserver/ff398089>

SQL Server DevCenter: <http://msdn.microsoft.com/en-us/sqlserver/>

Did this paper help you? Please give us your feedback. Tell us on a scale of 1 (poor) to 5 (excellent), how would you rate this paper and why have you given it this rating? For example:

* Are you rating it high due to having good examples, excellent screen shots, clear writing, or another reason?
* Are you rating it low due to poor examples, fuzzy screen shots, or unclear writing?

This feedback will help us improve the quality of white papers we release.

Send feedback.

# Acknowledgements

The authors wish to thank IBM and NetApp for providing the hardware equipment that was essential for this work to be possible. We would like to thank Christopher Lemmons and Richard Preston for their careful proofreading and outstanding suggestions.