Autonomous Selection of Indexes and Materialized Views in Two Commerical Database Management Systems

Abdur-Rahman El-Sayed
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada N2L 3G1
+1 519 888 4567
aaelsaye@cs.uwaterloo.ca

Hobbe Smit
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada N2L 3G1
+1 519 888 4567
hsmit@cs.uwaterloo.ca

ABSTRACT
The selection of the most beneficial indexes and materialized views (MVs) is a well-known issue in physical database design. The chosen set of indexes and MVs is crucial to the overall performance of the DBMS as it relates to specific workloads. Since this is such an important aspect of physical database design, database administrators (DBAs) spend a great amount of time selecting the best indexes and MVs for the purpose of improving database performance. In order to reduce the total cost of ownership, several DBMSs have tools for automating the physical design structure selection process. Both System B and System A have such tools, including the System B Advisor [15] and the System A Advisor [2]. In this paper, we present an objective assessment and comparison of both tools by defining evaluation metrics, analyzing recommendation algorithms, extensively testing both tools, and by providing a discussion of the results.

1. INTRODUCTION
Database management systems (DBMSs) are becoming more and more important as their applications multiply to include not only banking and internal inventory applications but also real-time and batch-process e-commerce systems. Despite the use of caching, ultimately the bulk of transactions must rely on the DBMS to provide quick and efficient execution of workload queries. Unfortunately, good DBAs are hard to come by and, as a result, they also require large salaries – increasing the total cost of ownership for a DBMS. In reaction to this trend, there has been much work done in recent years with the goal of automating database administration or, at the very least, reducing the load on already over-taxed DBAs. Ground-breaking work on self-managing DBMSs has been accomplished with respect to the recommendation and implementation of the most efficient set of indexes and materialized views – known as physical design structures (PDSs). These structures provide quick access to data. Indeed, formerly, one of the marks of a skilled DBA was the ability to implement the most efficient indexes and MVs for a given database being accessed with a particular workload. Thanks to research done in the last 5 years, there is now less of an onus on DBAs to come up with the most efficient mix of indexes and MVs for a given scenario because many DBMSs now ship with PDS recommendation tools. The two primary DBMSs implementing this technology are System B and System A – both of which have benefited from a lot of research, such as that found in [1,2,3,4,14,15,16].

Unlike previous research on automating the selection of the best PDSs, both System B's and System A's research teams have gone far beyond what could be achieved by strictly static approaches based on the analysis of integrity constraints, simple statistics, and basic schema information [11]. Instead, both teams chose to focus on the analysis of real SQL query workloads for the purpose of creating recommendations tailored to these workloads. Furthermore, they have successfully utilized the cost model of their query optimizers to ensure that PDS recommendations are actually used by the DBMS – unlike previous work, such as [7], which used external cost models. For both the System B and System A teams the goal is to generate the best recommendations given a set of constraints. The problem they face is as shown below:

Problem statement: Given a workload W (a set of SQL statements), a disk space constraint D, a time limit T, find the set of indexes and materialized views recommendations that reduces the workload cost the most within the time limit T, while using no more space than D.

Given the problem statement, it is clear that the problem of recommending the best PDSs is not easy – especially considering the fact that **all** the queries in the workload need to be considered and the fact that an exponential set of PDS combinations must be explored. In light of those facts, we are interested in seeing just how well the recommendation tools found in System A and System B solve the problem stated above.

In order for an objective comparison to be made we designed the appropriate evaluation criteria for assessing the various tests that we executed. Tests comparing System A Advisor and System B Advisor were evaluated using the concepts of: relative workload improvement, improvement estimation accuracy, scalability, workload compression, and constraint conformity. The first two metrics were used primarily for the first test involving a slightly-modified TPC-H workload [13] tested using various space constraints. The third and fourth metrics were each used to evaluate independent tests using other variations of the TPC-H workload. The last metric was used to evaluate how well the recommendation tools conformed to time and space constraints for all the tests executed. The aforementioned tests all involved the recommendation of both indexes and MVs. Another set of tests was run to compare the recommendations generated in indexes-only and MVs-only modes. Other experiments included a test that involved swapping the indexes-only recommendations of both recommenders and then measuring workload improvement – resulting in some surprising results. Finally, we also ran a group of tests against a real-world protein sequence database with two manually-created workloads.

Before we present our results, however, we first provide an overview of both the System A Advisor and System B Advisor in Section 2, along with a brief comparison of these two tools. In Section 3, we discuss the evaluation criteria used for the experimental assessment of both recommenders. Afterwards, we present the experimental setup and test design in Section 4. The experimental evaluation of both tools follows in Section 5. In Section 6, we provide some observations and note the limitations of both recommenders. Finally, we present related work and our conclusion in the last two sections of the paper.

2. OVERVIEW OF ADVISORS

In this section, we present an overview and comparison of both recommendation tools that were evaluated in our experiments. Both tools are generally capable of recommending indexes, materialized views, and database partitions. Our focus is on recommending both indexes and MVs, indexes only, and MVs only. An architectural overview of both recommendation tools is presented in Figures 1 and 2, which are taken from [16] and [1].

2.1 Input Parameters

As shown in Figures 1 and 2, a workload is given to both recommendation tools. The workload given can be a single SQL statement or a file containing several SQL statements. In the case of System B, the workload can also be a list of recently executed SQL statements available from the query cache or from a powerful query management tool included in an extension for System B [9]. For System A, a trace file can be used as a workload; the trace file can be created by running a profiler tool while a workload is being executed against the database. SQL statements can be extracted from events in the trace file [10].

Along with the workload, a space and time constraint is set for both recommendation tools. System B Advisor also allows the user to enable a sampling option, which is used to gather more reliable and accurate estimates for the candidate objects being generated. System A Advisor has sampling turned on by default [3,14].

Once the recommendation process is initialized by the user, each recommendation tool starts by generating candidate MVs and indexes. Having generated the candidate objects, each tool proceeds to select the optimal (or near-optimal) configuration among the generated candidate objects.

2.2 Candidate MVs Generation

Candidate MVs (also known as Materialized Query Tables, or MQTs) are generated differently by the two recommendation tools. System B considers the following techniques for generating MV candidates:

i. Converting queries in a given workload into MVs.

ii. Materialization of existing logical (non-materialized) views in the database.

iii. Use of MQO (Multiple-Query Optimization) to generate candidate MVs. MQO is a sophisticated technique used to find common sub-expressions among multiple queries [17].

On the other hand, System A uses the concept of "interesting table-sets" to generate MV candidates.

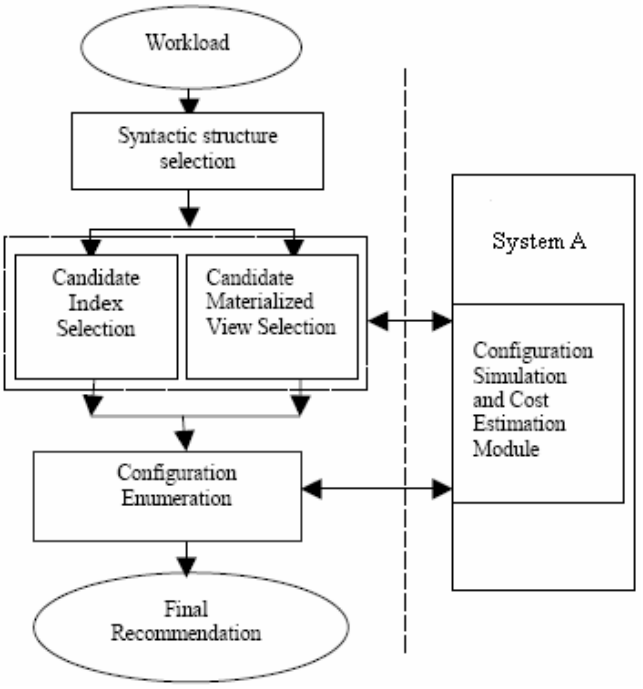

Figure 1. Design Overview of System A Advisor

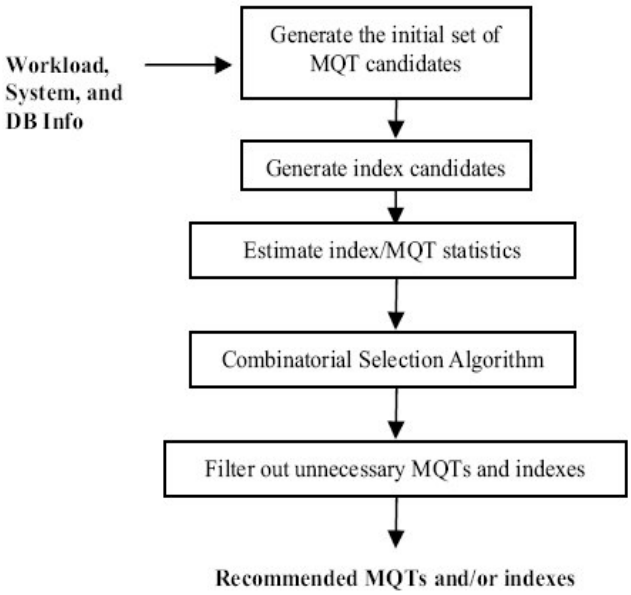

Figure 2. Design Overview of System B Advisor

Basically, a table-set (a subset of tables referenced in a query in the workload.) is considered "interesting" if materializing one or more views on it leads to a decrease in execution time, i.e. workload cost, above a given threshold. After identifying interesting table-sets, the System A Advisor generates a set of candidate MVs for each query that references an "interesting" table-set and then selects the best subset for that particular query. The union of the best subsets produces the final set of candidate MVs. The algorithm then generates an additional set of "merged" MVs that benefit multiple queries [1].

2.3 Candidate Index Generation

For candidate index generation, System B Advisor uses the "Smart column Enumeration for Index Scans" (SAEFIS) enumeration algorithm. The algorithm analyzes the predicates and clauses of both the workload statements and MV candidates to produce candidate indexes. In addition to the SAEFIS algorithm, a different enumeration algorithm, namely the "Brute Force and Ignorance" algorithm, is also used to generate extra candidate indexes. Details about the SAEFIS and BFI algorithms can be found in [16].

On the other hand, System A Advisor first considers all of the "admissible indexes" – that is, indexes that are syntactically relevant for the workload and candidate MVs. Afterwards, the DTA prunes the space of "admissible indexes" using the query-specific-best-configuration algorithm, which is detailed in [3].

2.3 Configuration Enumeration

Having generated candidate MVs and indexes, both tools proceed to the configuration enumeration phase. The System B advisor first estimates the size (cost) and benefit of each candidate object by using the optimizer estimates or, optionally, a sampling technique that provides more reliable and accurate size estimates. Once the benefit and cost for each candidate object is determined, the "Combinatorial Selection Algorithm" is initiated to select those objects with the highest benefit-to-cost ratio, that conforms to the given space constraint. Afterwards, the algorithm enters a swapping stage where it iterates over candidate objects that were not chosen before to find a better candidate set. Finally, the algorithm filters the solution by eliminating candidate indexes and MVs which were not used in the workload.

For the System A Advisor, configuration enumeration is performed by the Greedy(m,k) algorithm, which produces a final configuration with a total of k indexes and MVs. The algorithm starts by picking an optimal configuration of size m, where m is less than or equal to k, through enumerating in an exhaustive manner all of the configurations of size m or less. Afterwards, the algorithm picks the remaining (k - m) structures greedily, until all k physical structures have been chosen, or no further reduction in cost is possible, as described in [1,3].

2.4 Workload Compression

In order to improve the scalability of the recommendations tools, each vendor has incorporated a workload compression module that reduces/compresses the given workload. Both tools have entirely different compression algorithm – as described in this section. System B Advisor uses a simple built-in workload compression module. The module starts by estimating the size of all the queries in the workload and then orders them in a descending order. Afterwards, the module generates a new workload, consisting of the *Top K* queries, where the total cost of those queries is less than or equal to a certain percentage of the total workload cost – as presented in Equation 1.

$$Cost(TopK) \leq X\% * Cost(Workload) \qquad \text{[Equation 1]}$$

The new workload is then used as the input for the System B Advisor. Instead of allocating the responsibility of determining X on the user, the compression module uses 3 levels of compression, HIGH (X=5), MEDIUM (X=25) and LOW (X=60) [15].

On the other hand, System A uses a complex workload compression technique that uses clustering methods from the field of data-mining. The technique uses the concept of a "distance" between queries in a workload to remove similar queries from the workload up to a certain workload cost increase threshold. Further details about System A's compression technique can be found in [5].

3. EVALUATION CRITERIA

In order to facilitate an objective comparison between both recommendation tools we have developed the following evaluation criteria.

3.1 Relative Workload Improvement

A key performance metric for the bulk of our tests was relative workload improvement. The rationale behind choosing *relative* workload improvement over *absolute* workload improvement was that we wanted to ensure a fair comparison between two different DBMSs – each running on a machine with a slightly different CPU speed. In either case, in order to measure workload improvement we had to establish a baseline for each workload used in our tests. Having established a baseline, we derived the relative workload improvement for a given recommendation tool by measuring the overall change in workload execution time before and after the implementation of recommended indexes and MVs. Since we had decided on using *relative* workload improvement, the change in execution time was measured using a percentage value. Comparisons between recommendation tools could then be made using this value.

3.2 Improvement Estimation Accuracy

In order to determine the improvement estimation accuracy of each recommendation tool the final estimate given by the tool was compared with the actual workload improvement measured after implementing recommendations. Since improvement estimates were given in percentage values, deriving the estimation accuracy in terms of percentage error simply involved taking the absolute difference of the relative workload improvement and the estimated improvement. Although this performance metric could be construed as a query optimizer performance metric, we decided that it was nevertheless a good idea to evaluate estimation accuracy since it is integral to the proper functioning of each recommendation tool.

3.3 Scalability of Recommendation Tools

A simple, yet effective, performance metric was used to evaluate the scalability of each recommendation tool; the metric used was the ability to process SQL workloads scaled from 19 queries up to 420 queries. In order to facilitate a proper evaluation of scalability, all these tests were run with unlimited recommendation time. Furthermore, workload compression was disabled so that raw scalability could be tested.

3.4 Workload Compression Evaluation

Since each vendor claims to own a workload compression technique that does not sacrifice the quality of the recommendations, we decided to assess the impact of the workload compression. The metrics used to evaluate workload compression included: a) the decrease in recommendation time, and b) the decrease in the quality of the recommendations. The decrease in recommendation quality was determined by the percentage increase in workload execution time when the post-recommendation time of recommendation *without* compression is subtracted from the post-recommendation time observed for recommendations generated *with* compression.

Unfortunately, we found that System A does not provide means to enforce workload compression. However, System B's advisor allows us to specify the level of workload compression. In order to assess the workload compression algorithms used by both vendors, we designed variations of the TPC-H workloads that include 105, 210, 306, and 420 queries

3.5 Constraint Conformity

Time and space constraints are important parameters for both recommendation tools. We are interested in assessing how well both recommenders conform to those constraints. For an objective assessment, it is important to carefully select those constraints. Further details concerning constraint design and selection are presented in Section 4.2.

4. EXPERIMENTAL SETUP

In this section we present the setup and design of our experiments. The experimental systems are first presented, along with important procedures. Our test design is then described – focusing on how we designed and selected parameter values. The section finishes with a description of the databases and workloads used.

4.1 Environmental Setup

Our experiments were run on two different systems – one for each DBMS. Both systems were Pentium 4 systems running Windows XP Professional with 1 GB of RAM. However, System B was installed on a system having a 2 x 2.8 GHz CPU, while System A ran on a machine having a 2 x 3.2 GHz CPU.

Since both Databases run on a dual CPU machine, they are both capable of exploiting parallelism in query execution. Indeed, we have noticed that SQL Server utilizes parallelism quite well for query execution, especially when compared to System B. Therefore, we have disabled parallelism for both databases in order to put both on the same level in terms of query execution times.

Before executing workloads or running recommendation tools the database table statistics were updated. After recommendations were generated and implemented the database cache and buffer pool were emptied to minimize the factors influencing workload execution time. Having assurance that the queries being executed do not benefit from the query cache and DBMS buffer pool, the workload was executed and the execution time was recorded for comparison against a baseline.

4.2 Test Design

Since each database vendor has its own optimizer and its own way of processing queries, we had to carefully design the experimental tests. Objective evaluation of both recommendation tools required us to properly set the parameters and constraints for both recommendation tools.

As mentioned before, both recommendation tools have various tuning parameters. The primary parameters we used in our experiments imposed space and time constraints and turned sampling on/off. Unfortunately, we could not use these parameters to their full potential due to certain issues. For instance, we noticed that System B's Advisor violates the time constraint in many cases. Clearly this is because it considers the time constraint only at the final swapping stage. We also realized that System A was running on a system with a faster CPU than System B. Both issues could lead to an unfair comparison of recommendation results and time constraint conformity results. Therefore, we have decided to use an unlimited time constraint for all of our experiments.

Although not as problematic as the time constraint parameter, we have opted to compute the space constraint as a percentage of current database size. Our reasoning is illustrated by a simple index space calculation: an index on the TPC-H *lineitem* table consumes 216 MB in System B while it consumes 197 MB in System A. Clearly, it would be unfair to give the same amount of storage space as a constraint since both DBMSs have different storage requirements for the same indexes. The space constraints that we decided on were the following: 30%, 120%, 200%, and 500%. The first constraint was chosen to represent a minimum value that restricted MVs and indexes to a space less than the size of the database. 120% was chosen as the token value above the database size. The other values were chosen to determine how well each recommender performs with large space constraints.

Other than space and time constraints, we decided to set the System B sampling parameter to "ON" for all but one test. This was done in the interest of increasing the objectivity of our testing, since sampling is on by default in SQL Server.

4.3 Databases and Workloads

For our tests we have used two data sets: the often-used TPC-H Benchmark [13] and the PIR Protein Sequence Database [12]. The original data size was 1 GB of data in 8 tables for the TPC-H dataset and 980 MB of data in 16 tables for the PROTEIN dataset.

As for the workloads, for TPC-H we have used variations on the original 22-queries workload which include workloads with 19, 21, 105, 210, 306, and 420 queries. The first workload we used was the 21-query workload (TPCH-1) – which had query 15 removed. We had to remove query 15 because System B Advisor could not handle "CREATE VIEW" statements within the workload. The 19-query workload, referred to as TPCH-2, was created in reaction to test results found for the TPCH-1 workload. In order to provide a more fair comparison we decided to remove queries 17 and 20 from TPCH-1, since their baseline execution times were large when run using SYSTEM B – introducing a bias in favour of SYSTEM B. For the protein database, we have constructed two of our own workloads, using different types of query-families. We developed queries with potentially "hot" indexes, where a "hot" index is defined as an index on a column in a (1) WHERE clause, (2) GROUP BY clause, and/or (3) ORDER BY clause. The PROTEIN-1 workload consisted of 35 queries, while the PROTEIN-2 workload consisted of 30 queries.

5. EXPERIMENTAL EVALUATION

Our experimental results are presented in this section. Here we present results for the TPC-H database comparing workload improvement for various space constraints. Next, indexes-only recommendations for both recommenders are compared with MVs-only recommendations and combined recommendations. After these tests, we swap indexes-only recommendations generated by both recommendation tools and compare the change in improvement over the native recommendation. We then present results for our tests using the PIR protein database. Following our workload improvement results we also evaluate the results found for the workload improvement estimation accuracy, scalability, and workload compression tests.

5.1 Workload Improvement

5.1.1 Results using TPCH-1

Our first experiment was to assess the impact of both advisors on improving the workload execution cost. In the first step of experiments, performed with the TPCH-1 workload, we varied the space constraint, and provided an unlimited time constraint for both advisors. The space constraint used for the experiments was: 30%, 120%, 200%, and 500%. The results indicating the actual workload improvement are presented in Figure 3.

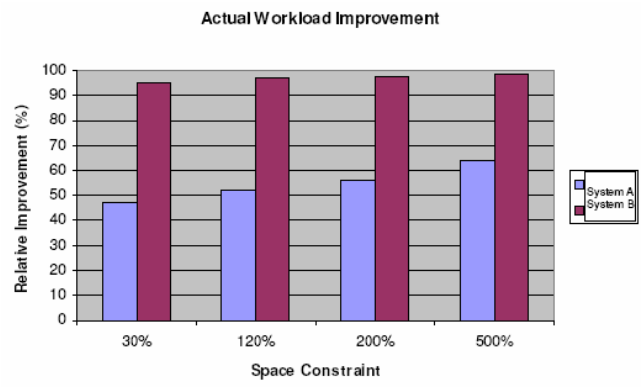

Figure 3. Space Constraint Test Results

As expected, the graph shows that the actual workload increased for both SYSTEM B and SQL Server whenever the space constraint was increased, since there is more space available for extra MVs and indexes. Also observe that the actual improvement for SYSTEM B was significantly higher than for SQL Server – for all of the tests.

5.1.2 Results using TPCH-2

For the next set of experiments, we used TPCH-2 to determine the performance of the recommendation tools for the following recommendation modes: indexes only, MVs only, and both indexes and MVs. We also ran a test to determine the improvement when indexes-only recommendations were swapped. For both tests, the space constraint was fixed at 200%, since we believe such a constraint is reasonable for most databases.

First, we present the baseline execution time for both databases. As can be noted in Figure 6 (on the next page), both SYSTEM B and SQL Server baselines show similar execution time patterns. Having established a more objective baseline we present our first test using TPCH-2. The actual TPCH-2 workload improvement achieved for each of the three recommendation modes is shown in Figure 4, which shows that both tools produced significant workload improvement for the first two recommendation modes. Yet, SYSTEM B had a larger improvement than SQL Server, particularly when only MVs were recommended. Such a result can be partially explained by the fact that SYSTEM B recommended exactly twice as many MVs as SQL Server did (8 vs. 4). Furthermore, after the analysis of the query execution plans, we observed that, for SQL Server, each of the 4 MVs only benefited a single query each. On the other hand, for SYSTEM B, the 8 recommended MVs benefited a total of 10 queries. Finally, for SYSTEM B, we observed a significant reduction in the execution time of queries that exploited MVs. For instance, the execution time for query 6 decreased from 21.574 seconds to 1.297 seconds. However, the most improved query found for SQL Server, query 4, had its execution time decreased from 23.667 seconds to 18.821 seconds. Clearly, SYSTEM B utilized the recommended MVs better than SQL Server did.

Figure 4. Results for Various Recommendation Modes

After running our first batch of TPCH-2 tests we could use the indexes + MVs recommendation result to give an idea of the TPCH-1 baseline bias. By comparing the TPCH-2 and TPCH-1 results for the 200% space constraint we noted that SYSTEM B still has superior improvement. Hence, we decided to keep our TPCH-1 test results.

To further verify our assumption that SYSTEM B's advisor results in a higher workload improvement than System A's DTA, we have decided to swap the indexes-only recommendations and compare the results.

5.1.3 Swapped Recommendations

An important condition in our swapped recommendations test was the requirement that the recommendations were not to exceed the provided disk space constraint. In other words, if SYSTEM B's swapped recommendations consumed over 200% of the SQL Server TPC-H database size, the results would be insignificant, and vice-versa. Fortunately, in our swapped indexes only experiment, the space constraint was not violated. The results of the experiment are shown in Figure 5.

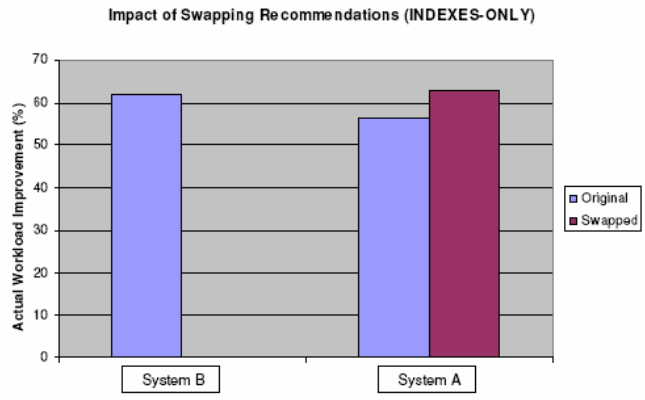

Figure 5. Results found for Swapped Recommendations

Although we could not implement the SQL Server recommendations in SYSTEM B due to reasons given later in this section, we found that after applying the recommendation of SYSTEM B on SQL Server, workload execution time improved

by 6.476% for SQL Server when compared to its original recommendation.

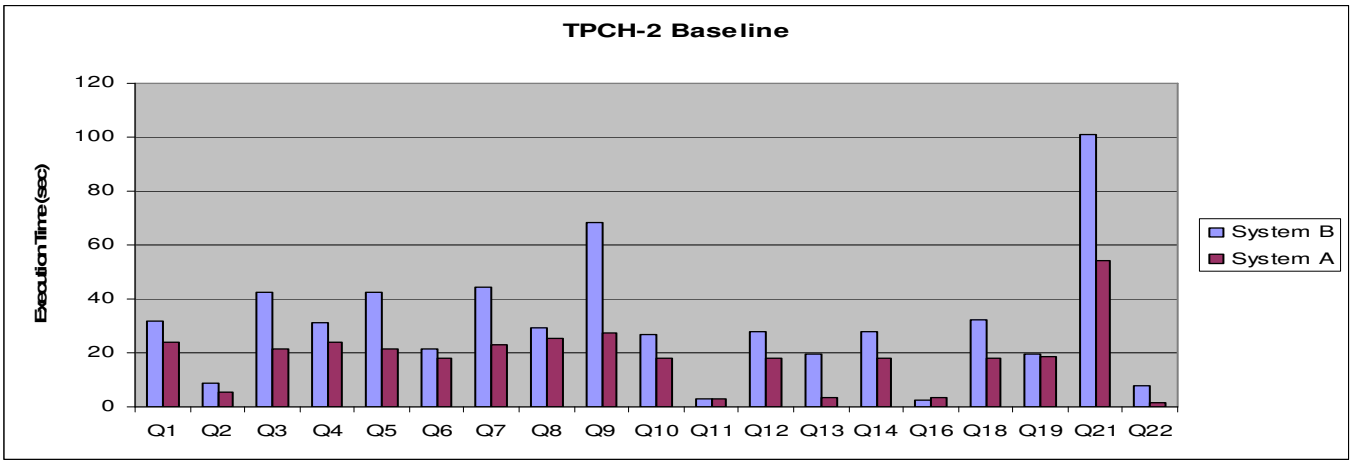

Figure 6. Baseline Query Execution Times for TPCH-2 Workload

As shown in Figure 5, the swapped recommendations for SQL Server improved the actual workload by 62.86%, even higher than both SYSTEM B and SQL Server's original indexes-only improvement, which suggests that SQL Server could have provided better recommendations. We believe there are several reasons for that result. First, we note that SYSTEM B recommended 38 indexes while SQL Server recommended only 13 indexes. The number of indexes recommended by both advisors is shown in Table 1.

Table 1 Indexes Recommended in INDEXES-ONLY Test

Table	# Indexes (SYSTEM B)	# Indexes (SQL Server)
LINEITEM	11	10
ORDERS	7	2
CUSTOMER	5	0
NATION	5	0
SUPPLIER	4	0
PARTSUPP	3	1
PART	2	0
REGION	1	0

As shown in Table 1, SYSTEM B recommended more indexes for each table in the database. Although we do not claim that more indexes implies more workload improvement, there does appear to be a strong correlation between the recommendation of many indexes that are used by multiple queries and the **overall** workload benefit gained from these indexes. Indeed, although some queries had a longer execution time on SQL Server with SYSTEM B recommendations, the total execution time after those recommendations were implemented on SQL Server was less by 22.424 seconds.

The largest improvement was found for query 5, which took 18.408 seconds to execute on SQL Server with its own recommendations and only 5.693 seconds using SYSTEM B recommendations. Query 7 also had an impact on the difference of both actual workload improvements, since it took 10.751 seconds to execute with original recommendations and 6.296 seconds with SYSTEM B recommendations. The other queries had negative or negligible improvement. After analyzing the execution of both queries, we have realized that query 5, when executed with SYSTEM B's recommendations, exploits 6 indexes, on the following tables: NATION, REGION, SUPPLIER, LINEITEM, ORDERS and CUSTOMER. However, when the same query is executed with SQL Server recommendations, it uses one primary (already existing) index and two recommended indexes on the 2 largest tables in the database – namely LINEITEM and ORDERS.

We note that the original execution plan of query 5 uses only one primary index-scan and the rest of the operations are table-scans, whereas the execution plan of query 7 does not exploit any indexes and relies completely on table-scans.

For query 7, when using SQL Server recommendations, the optimizer uses the same two indexes used when query 5 was executed, namely, indexes on the LINEITEM and ORDERS tables. In contrast, when SYSTEM B's recommendations were applied, the optimizer chose 3 different indexes (different than those chosen for query 5) on CUSTOMER, ORDERS and NATION, and the same indexes used in query 5 for LINEITEM and SUPPLIER tables. We believe that the rich set of indexes recommended by SYSTEM B was the main factor in enhancing both query 5 and query 7 when executed by SQL Server, resulting in a 6.476% increase in actual workload improvement.

An important note here is that although the size of both REGION and NATION tables is small compared to other tables, SYSTEM B's indexes on these tables belonged to the set of indexes that improved query 5 and query 7 when executed by System A. Unlike SQL Server, SYSTEM B considers a very huge space of candidate indexes. Furthermore, it does not perform any pruning, whereas SQL Server does. The latter considers "admissible" indexes and further prunes those indexes using the *query-specific-best-configuration* algorithm described in [3]. Since it does no pruning, SYSTEM B's algorithm results in a larger index space, and therefore a potentially better workload improvement. However, not pruning the candidate structures space may have a

negative effect on the scalability of the recommendation tool, as described in Section 5.3.

Given the surprising results for implementing SYSTEM B recommendations on SQL Server, it was unfortunate that we were unable to test the recommendations provided by SQL Server on SYSTEM B. Such a test was unfeasible due to the fact that creating an index with an INCLUDE clause in SYSTEM B requires that the index be UNIQUE [8]. The previous restriction was not imposed by SQL Server, so not all its recommendations were compatible with SYSTEM B.

5.1.4 Results using PROTEIN Workloads

The last of our workload improvement experiments was done against a real-world database and two manually generated workloads. The results of our test are shown below, in Figure 7.

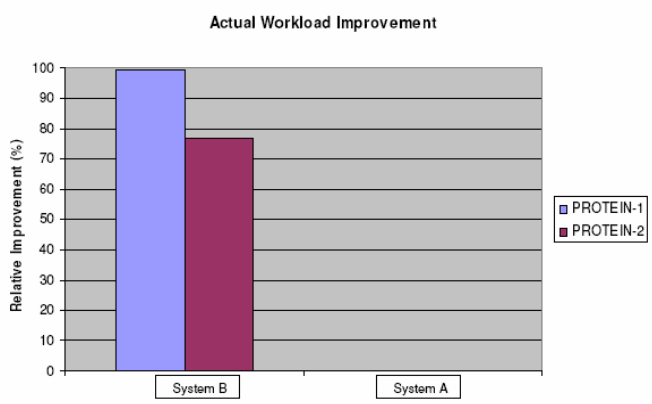

Figure 7. Improvement found for Protein Workloads

It is evident once again that SYSTEM B provided superior results. A partial explanation for such a large difference is that SYSTEM B recommended at least 3 times the number of indexes that SQL Server did and more than twice the number of MVs. However, it is more important to note that the SQL Server baseline execution time was much shorter than SYSTEM B, so a high percentage of improvement would be hard to attain. To get a rough idea as to whether this was the primary factor involved, we considered SYSTEM B's improvement found for only those PROTEIN-2 queries whose SYSTEM B baseline execution time was less than a given threshold. The threshold was set to be the maximum individual query execution time for SQL Server's baseline. As a result, SYSTEM B's improvement dropped to 20.77%. Given that SQL Server's improvement was 12.92%, the new result is comparable to previous workload improvement test results.

5.2 Improvement Estimation Accuracy

In this section, we present a comparison between the estimated and actual workload improvement as returned by both recommenders. We then present the average error of each workload's improvement estimates. The actual and estimated improvements found for both recommenders are shown in Figures 8 and 9 for TPCH-1 and TPCH-2 workloads respectively.

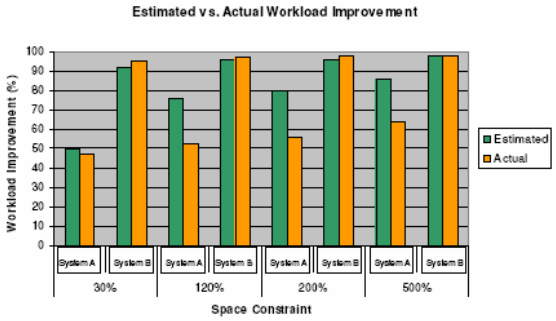

Figure 8. TPCH-1 Improvement Estimates

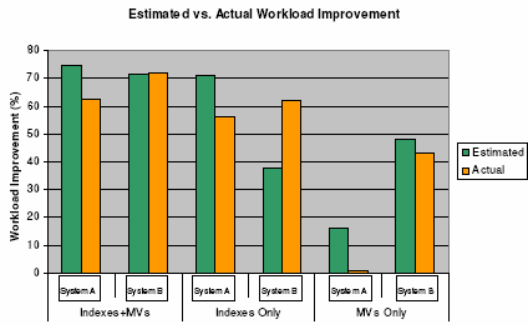

Figure 9. TPCH-2 Improvement Estimates

As can be seen for both workloads, in most cases SYSTEM B's estimates were a lot more accurate than those of SQL Server. In addition, for TPCH-1, SYSTEM B *never* over-estimated the actual workload improvement, while SQL Server *always* over-estimated the actual improvement.

Using the Figure 8 as a baseline, it is clear that for the MVs-only recommendation (shown in Figure 9) SYSTEM B breaks its trend of never over-estimating improvement and provides a somewhat liberal estimate. We observe that the only case where SQL Server provides a better estimate than SYSTEM B is shown in Figure 9 for the indexes-only test. Using the results shown in the previous figures we derived the average estimation error, which is displayed in Figure 10.

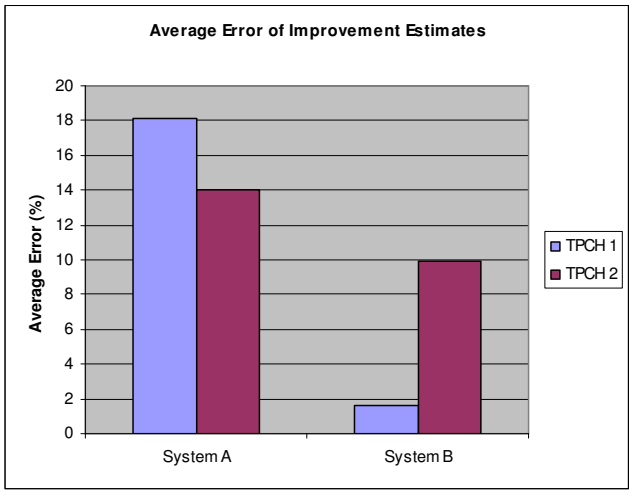

Figure 10. Average Improvement Estimation Error

Clearly, the overall estimation accuracy for both workloads was better for SYSTEM B. Note, however, that estimation accuracy is highly dependent on the optimizer. Nevertheless, we present these results as a part of our broad assessment of both recommendation tools.

5.3 Scalability Test

For the scalability test, we wanted to test the "raw" scalability of both advisors, that is, without the benefit of workload compression. We disabled workload compression for SYSTEM B using the "-k OFF" parameter for System Badvis command line. Unfortunately, System A does not provide means to enable/disable workload compression, although [2] indicates that the module is incorporated into System A. Nevertheless, there is much evidence leading us to suspect that the workload compression module is not installed, or is not enabled. First, in contrast to SYSTEM B workload compression, there is no application documentation on any available workload compression parameters for the DTA. Second, we have observed that when the DTA is executed with a large workload, it takes a very long time compared to System B Advisor with workload compression – almost 6 hours with a 210-query workload. Since SYSTEM B took just over 4 hours to tune the same workload **without compression**, this is compelling evidence for our aforementioned conclusion.

Table 2 Scalability Test Results

# of Queries	SYSTEM B Advisor (no comp.)	SYSTEM B Advisor (med. comp.)	System A Advisor
19	pass	pass	pass
22	pass	pass	pass
105	pass	pass	pass
210	pass	pass	pass
306	pass (no sampling)	pass	pass
420	**fail**	pass	pass

For our scalability tests the following workloads have been provided for both recommenders: TPCH-1, TPCH-2, TPCH-3, TPCH-4, TPCH-5 and TPCH-6, consisting of 19, 21, 105, 210, 306 and 410 queries respectively. For these tests we are simply interested whether or not the recommendation task successfully completed with recommendations. The results of the tests are presented in Table 2, which indicates that SQL Server was able to cope with all the workloads, and provided recommendations for each test.

On the other hand, SYSTEM B was not able to provide recommendations for the 420-query workload (TPCH-6) when compression was disabled, unlike when compression was set to MEDIUM. The SYSTEM B advisor consumed all of the available disk space and exited with a critical error after evaluating 137,670 configurations. This unusual result can be explained by considering that the System B Advisor selection algorithm injects a huge amount of virtual indexes and materialized views into the database, generates their statistics, and then computes the execution cost of every workload query [16].

As a result, the free disk space on which SYSTEM B was installed, which was approximately 18 GB was consumed completely by the System B Advisor. We therefore conclude that the SYSTEM B advisor does not scale with large workloads because it does not substantially prune the space of candidate indexes and MVs. On the other hand, it appears that System A is able to deal with large workloads quite well, assuming that workload compression was not incorporated into the Beta version we were using.

5.4 Workload Compression Evaluation

In this experiment, we wanted to assess the impact of workload compression on both the advisor execution time and the drop in quality of the recommendations produced. Unfortunately, we could not assess SQL Server's workload compression module due to the reasons noted earlier. Therefore, only SYSTEM B workload compression module is assessed.

As mentioned earlier, SYSTEM B has 4 options for workload compression: OFF, LOW, MEDIUM and HIGH. We have decided to use MEDIUM and HIGH for assessing the impact of workload compression on the advisor execution time, using a workload consisting of 105 queries (TPCH-3). Figure 11 shows the impact of the workload compression on the advisor execution time when the TPCH-3 workload was used.

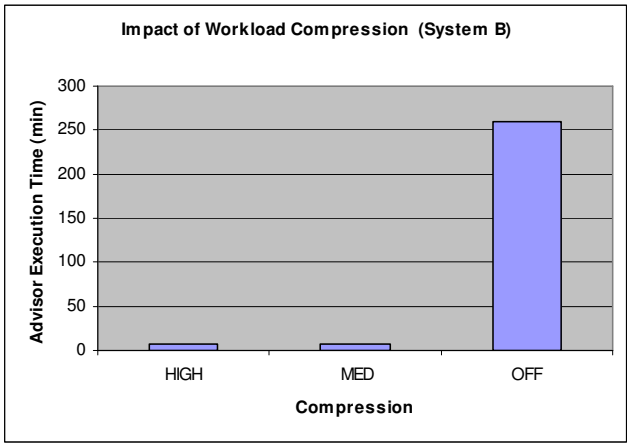

Figure 11. SYSTEM B Workload Compression Results

As displayed in the above graph, the original advisor execution time (with no compression) was approximately 4.3 hours. When workload compression was enabled with MEDIUM and HIGH, the execution time dropped to 6.3 and 7.3 minutes respectively. Clearly, the reduction of execution time was very impressive.

Since MEDIUM is the default level of compression, we have decided to test the impact of workload compression on the drop of recommendation quality in using that level. Essentially, we ran the advisor using the TPCH-3 workload, MEDIUM compression, unlimited time, and a 200% space constraint. We then implemented the recommendations, and compared the actual workload improvement with the actual workload improvement achieved by the recommendations when workload compression was disabled (OFF). The results are show in Figure 12.

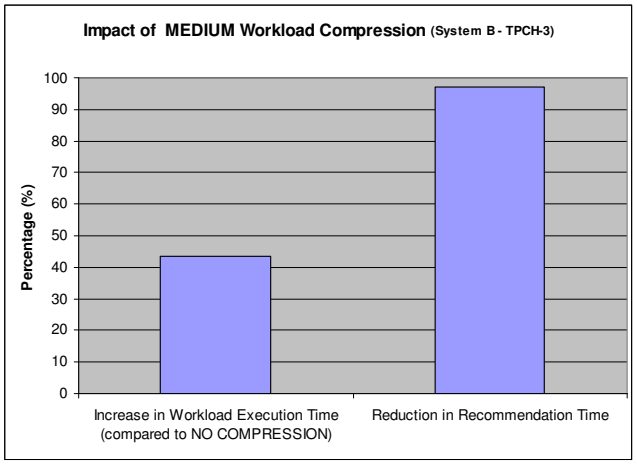

Figure 12. Impact of Workload Compression on SYSTEM B

As shown in Figure 12, SYSTEM B's workload compression technique significantly reduces the advisor execution time – up to 97%. On the other hand, we noticed that the workload execution time increased by 43% compared to the workload execution time when "no compression" recommendations were implemented. Our rationale of why such a dramatic trade-off occurred is based on the simplistic approach of SYSTEM B's compression algorithm. The algorithm, as explained earlier, picks up the top *K* queries where the total cost of those queries is less than or equal to a certain percentage of the workload cost. As a result, in many cases where almost of all the queries in the workload have a high cost, as is the case for TPC-H queries, many queries in the workload are pruned without considering the fact that they also have high execution costs. As a result, the drop in quality in some cases may be unsatisfactory. We believe SYSTEM B's algorithm is geared towards pruning and not compression, even though the authors claim that their approach compresses workloads and is comparable with SQL Server's approach, as stated in [15]. To further illustrate the weakness of SYSTEM B's workload compression approach, we present the following scenario: Consider a workload, consisting of 50 queries, where the cost of the any query is determined using the following equation, where *I* is the query number:

$$F(I) = 5000 - 20 * I \qquad \text{[Equation 2]}$$

Using Equation 2, the first query's cost is 5000, the next query costs 4980, the next costs 4960, and so on. Therefore, the total workload cost = 5000 * 50 – 20 * [50 * (50 + 1) / 2] = 224,500 **timerons** (a SYSTEM B metric). Assuming medium compression is enabled, the algorithm will pick the top *K* queries where their total cost is less than or equal to 25% of total workload cost (i.e., 56,125 timerons). Given the aforementioned threshold, only 11 top *K* queries will be chosen since the total cost of those queries is less than or equal to 56,125 timerons. The rest of the 39 queries will be pruned without any consideration. We argue that it is common to have a workload consisting of high cost queries where the difference between the costs of all queries is not very significant. In such a scenario, SYSTEM B's compression algorithm prunes a large number of queries without any consideration, and produces unsatisfactory results.

Even though we did not test SQL Server's workload compression module, which based on a complex technique that relies on data-mining concepts [5], we strongly believe it is more efficient than SYSTEM B's approach, since it takes into account the similarities of the queries in the workload. Not to mention, it has been shown in [5] that the SQL Server workload compression module may actually lead to higher-quality recommendations than those produced using an uncompressed workload.

6. OBSERVATIONS AND LIMITATIONS

While experimenting with both SYSTEM B and SQL Server advisors, we have observed several anomalies and limitations within both advisors. In this section, we discuss and analyze them in detail.

6.1 "Suffered Queries"

After implementing the recommendations produced by both advisors, we have noticed that some of the workload queries have "suffered". That is, their execution time took longer **after** applying the recommendations of the advisor. For example, when using the TPCH-1 workload on SYSTEM B, and running the advisor with unlimited time and a space constraint of 500%, we noticed that query 2 had a longer execution time **after** applying the recommendations; the query took 6.531 seconds before the recommendations were applied and 12.046 seconds after, a 5.515 second increase in execution time. It should be noted that this issue is not the caused by the advisor, but rather by the optimizer. By analyzing the optimizer's execution plan for query 2 before and after the recommendations, we have noticed that all of the table-scan operations were swapped by an unclustered index-scan when the recommendations were applied. Such a swapping resulted in increased execution cost, since for that particular query, most of the rows of the referenced tables were fetched (having a high selectivity) and the use of an unclustered index-scan adds an overheard to the total I/O cost. It is important to note the optimizer depends heavily on the current statistics, which we updated at every stage of the recommendation and testing process.

6.2 SYSTEM B "Indexes Only" Mode

We have noticed that, upon giving the TPCH-2 workload as an input for the SYSTEM B advisor and running it in "indexes only" mode, the advisor does not generate any recommendations when given a space constraint of up to 3 GB, even though it evaluates 3082 solutions. Yet, when a space constraint of 3100 MB was given, the advisor recommended 38 indexes with an actual workload improvement of 61.95%. The reason behind these unusual results was clear after we queried the ADVISE_WORKLOAD table and discovered that the COST_BEFORE and COST_AFTER columns, which represent the estimated cost for a query before and after implementing recommendations, for all 19 queries were the same. What is not clear is how the addition of only 100 MB to the space constraint had the impact of positively changing the COST_AFTER value for 17 out of 19 queries, resulting in an estimated improvement of 37.86%.

To further investigate this issue, we ran the advisor in "indexes-only" mode and supplied it with the PROTEIN-1 workload and a small space constraint of 500 MB. Unlike the TPCH-2 workload results, the advisor recommended 7 indexes that would improve the workload by 1.55%. We conclude that the complex structure of the TPC-H queries may have caused the advisor not to recommend any indexes within a space constraint of 3GB. Nevertheless, we admit that such a "no recommendation" result is

not convincing, and that it bears further investigation and analysis with the help of System B's support team.

6.3 System B's Sampling vs. No Sampling

The authors of [16] claimed that with sampling enabled, the System B advisor is capable of obtaining more accurate statistics and, thus, higher quality recommendations. In light of the above statement, we assessed the impact of enabling sampling in terms of the advisor execution time and the actual workload improvement. We note that sampling is used only for obtaining statistics for candidate MVs, but not statistics for candidate indexes. We ran the System B advisor in "Indexes and MVs" mode, with an unlimited time constraint, a 200% space constraint, and the TPCH-2 workload. The results are presented in Table 3:

Table 3 Sampling Parameter Test Results

Sampling	# of Indexes	# of MVs	Estimated Improvement	Actual Improvement	Advisor Execution Time (minutes)
ON	38	8	71.46 %	72.033 %	36.02
OFF	32	6	57.54 %	54.085 %	2.53

From the results, we note that the number of recommended indexes and MVs increased when sampling was enabled. Indexes increased from 32 to 38 and MVs from 6 to 8. The addition of those new indexes and MVs resulted in increasing both the estimated and actual workload improvement, verifying the claim presented in [16]. We also note that when sampling was enabled, more accurate statistics lead to an improvement estimate that was very close to the actual improvement. In contrast, when sampling was disabled, the improvement estimate was inaccurate and overestimated the actual improvement.

Despite the advantages obtained when sampling is enabled, one can notice its trade-off in the significant increase of the advisor execution time. In this particular recommendation experiment, the advisor execution time was increased by 1344% – a significant increase. It is also important to note that a time constraint of 5 minutes given to the advisor with sampling enabled and all other parameters remaining as previously mentioned, would have been violated easily. This violation is due to the nature of the advisor's selection algorithm, which only takes into account the time constraint at the final swapping stage.

7. RELATED WORK

As far as we are aware, there has not been any direct comparison between two autonomous recommenders in the literature of database systems. However, the authors of [6] designed a benchmark for autonomic configuration recommenders, which can be used to compare the quality of recommendations produced by one or more recommenders on the same database system. These authors note that the question of comparing two autonomous recommenders is not an objective of their work and should be addressed separately. Hence, we have focused on comparing two different recommenders running on two different database systems.

In addition, the authors of [6] proposed a new metric, namely "cumulative frequency", for assessing workload recommendations. We argue that such a metric is not effective in comparing recommenders on different database systems, due to its high dependence on the execution order of the queries in the workload.

8. CONCLUSION

After a broad and extensive experimental evaluation of two recommendation tools capable of producing integrated recommendations containing both indexes and MVs, it is clear that both the System A Advisor and System B Advisor are very useful for database administrators. They allow for significant improvements to be made to complex real and synthetic workloads. Nevertheless, after presenting what we hope is an objective assessment, we note that the System B Advisor was better than the System A Advisor in two key areas: relative workload improvement and improvement estimation accuracy. Compared to System B, the System A Advisor was better in terms of constraint conformity, scalability to large workloads, and, in our algorithmic analysis, workload compression.

Future work in this area should include an investigation of the impact of having UPDATE queries in the workload, and how their associated negative impact affects the relative workload improvement and the overall quality of the recommendations.

9. REFERENCES

[1] Agrawal S., Chaudhuri S. and Narasayya V. Automated Selection of Materialized Views and Indexes for SQL Databases. *Proceedings of the 26th International Conference on Very Large Databases* (VLDB00), Cairo, Egypt, 2000.

[2] Agrawal, S., Chaudhuri, S., Kollár, L., Marathe, A. P., Narasayya, V. R., and Syamala, M. Database Tuning Advisor for Microsoft SQL Server 2005. *Proceedings of the 30th International Conference on Very Large Databases* (VLDB04), Toronto, Canada, 2004.

[3] Chaudhuri, S. and Narasayya, V.R. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. *Proceedings of the 30th International Conference on Very Large Databases* (VLDB97), Athens, Greece, 1997.

[4] Chaudhuri, S. and Narasayya, V.R AutoAdmin 'What-if' Index Analysis Utility. *Proceedings of the ACM SIGMOD*, 1998.

[5] Chaudhuri, S., Gupta, A. K., and Narasayya, V. K. Compressing SQL Workloads. *Proceedings of the ACM SIGMOD*, 2002.

[6] Consens, M., Barbosa, D., Teisanu, A., and Mignet, L. Goals and Benchmarks for Autonomic Configuration Recommenders. *Proceedings of the ACM SIGMOD*, 2005.

[7] Hobbs, L. and England, K., *Rdb/VMS A Comprehensive Guide*, Digital Press, 1991.

[8] IBM DB2 Information Center. http://publib.boulder.ibm.com/infocenter/SystemBhelp/index.jsp

[9] IBM DB2 Query Patroller, http://www.ibm.com/software/data/System B/querypatroller/.

[10] Microsoft Developer Network Library. http://msdn.microsoft.com/library/default.asp

[11] Peter, C. and Gurry, M., *ORACLE Performance Tuning,* O'Reilly & Associates, Inc. 1993.

[12] PIR International Protein Sequence Database. http://pir.georgetown.edu/pirwww/search/textpsd.shtml

[13] TPC Benchmark H. Decisions Support. http://www.tpc.org/

[14] Valentin, G., Zuliani, M., Zilio, D. C., Lohman, G. M., and Skelley, A. DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes. *ICDE* 2000

[15] Zilio, D. C., Rao, J., Lightstone, S., Lohman, G. M., Storm, A., Garcia-Arellano, C., Fadden, S. DB2 Design Advisor: Integrated Automatic Physical Database Design. *VLDB* 2004.

[16] Zilio, D. C., Zuzarte, C., Lightstone, S., Ma, W., Lohman, G. M., Cochrane, R., Pirahesh, H., Colby, L. S., Gryz, J., Alton, E., Liang, D., and Valentin, G. Recommending Materialized Views and Indexes with IBM DB2 Design Advisor. *IEEE International Conference on Autonomic Computing* (ICAC) 2004.

[17] Lehner, W., Cochrane, B., Pirahesh, H., and Zaharioudakis, M. Applying Mass Query Optimization to Speed up Automatic Summary Table Refresh. *ICDE* 2001.