https://doi.org/10.48009/2_iis_2008_544-550

AN ANALYSIS OF DISTRIBUTED DATABASE INDEXING METHOD IN REGARD TO PERFORMANCE OF EXTRACT/TRANSFORM/LOAD (ETL) PROCESSES

Dennis C. Guster, St. Cloud State University, dguster@stcloudstate.edu
Paul Safonov, St. Cloud State University, safonov@stcloudstate.edu
Christopher Brown, St. Cloud State University, chrisb@stcloudstate.edu
Brittany Jansen, St. Cloud State University, jabr0601@stcloudstate.edu

ABSTRACT

Data mining and very large databases offer a great deal of promise for business and government alike. Unfortunately, due to the ever changing data needs of the modern organization, as evidenced by the growing number of petabyte data applications, the task to efficiently extract data in a timely manner becomes more difficult. The research described herein presents some encouraging findings in regard to using indexing to improve upon the ETL process. Specifically, a decrease of about 3 times in run time was observed when moving from a flat table to an indexed table function logic. The business value of this decrease may be realized in terms of less wait time when end-user queries are executed.

Keywords: Distributed Databases, Data Mining, Indexing, ETL

INTRODUCTION

Distributed databases and networking technology have made tremendous progress throughout the years; so much, in fact, that it is now possible to store vast amounts of data in a distributed environment [21]. However, being able to access or mine such data successfully and in a timely manner has been and still is complicated. In the past, there have been effective implementations when in a clustered computing environment; unfortunately, however, they have not been universally successful [26]. In fact, both computationally and data intensive problems have been discovered when using distributed data mining in grid environments [24].

Often the bottleneck has been adequate network bandwidth, but the literature suggests data mining can still be effective if proper scheduling techniques are used. One recent study suggested that a two stage scheduling algorithm should be used so that both internal and external constraints can be addressed [17].

Data mining applications tend to more successful on a LAN; however there is a need to be able to support data mining applications in WAN world, particularly across the internet which of course is major source for world wide connectivity. The internet tends to be the preferred means of WAN connectivity because, for some companies absorbing the line costs for a private network to support data mining are out of the question. Further, in order to save costs: both development and training, many companies would like to use their existing network/protocol methodology to provide access to the data to be mined. In the business world this probably is data related to an E-commerce system which means http and a browser. From a design perspective, this may involve a peer to peer network and algorithms that are designed to limit communication overhead [10]. Optimizing scheduling and distribution methodology can have a major impact on performance and there are numerous interrelationships that need to be well thought out [19]. The one of primary concern in this study is the manner in which the database is indexed and how that indexing logic interacts with underlying physical structure upon which that database is stored. The goal in part is not only to improve performance, but do so by distributing the indexing logic across multiple CPUs.

The literature suggests that a well thought out system can produce impressive results. In a study undertaken in 2004 which varied the number of nodes on which the data to be mined was stored upon found increasing the nodes from one to three reduced the time from 23 hours and 18 minutes to 11 hours and 32 minutes. Further, increasing the number of nodes to eight resulted in an even quicker time to solve of 4 hours and 47 minutes [8].

The concept of optimizing access in a distributed database is not new. For example, [9] applied search optimization techniques and determined there were distinct advantages to distributing large databases. As of late distributed databases have been integrated into grid environment which provides plenty of processing for complex/intense applications. Paper [2] delineates the desirable characteristics of such an

environment: support of data indexing at a level finer than the file level, data sharing without a globally shared schema and a peer to peer flavor which is highly scalable. There is also research that specifically addresses the importance of index optimization and the need to perform effective load balancing in the distributed database model [2].

The literature also recognizes the value of distributed databases in supporting intense web applications and states special care needs to be exercised to accommodate the bursty inter-arrival of data requests [4]. This research verifies that parallelism on both a processor and disk IO level can be effective in improving performance.

Although the work to date is encouraging and provides a basic framework there is still a lot to be done. It would be expected that the volume of data and the need to effectively analyze that data will increase in the future. The need to devise reliable and scalable techniques across the internet to support a wide variety of applications will continue to be important [12]. This implies research beyond optimizing the network and splitting the database across multiple nodes. In fact, the fundamental level is how is the database itself searched? Are there logic short cuts that can be invoked that will speed up performance and yet not complicate the hardware configuration? These short cuts are often linked to the manner in which the database is indexed. To test the influence of scaling and index methodology on performance within a web based database/data mining application an experiment was devised and data collected. The next section of this manuscript presents a brief literature review; this is followed by a detailed description of the experiment as well as results. The last section presents a discussion and call for future research.

LITERATURE REVIEW

Data mining has been successfully used in the industry for several years now. This is largely due to the database systems, which are both well designed and can provide adequate response times. Generally, businesses have utilized data warehousing and data mining to understand customer spending "patterns, associations, [and/] or relationships" [20]. Understanding these relationships and/or patterns is crucial to businesses because it reveals present and future business trends between "'internal' factors such as price, product profitability, or staff skills, and 'external' factors such as economic indicators, competition, and customer demographics" [20]. All of this is done, of course, in hopes that the information disclosed by these patterns will enable users to make higher quality decisions that result in improved profitability [27].

Businesses can employ data warehousing in a number ways to achieve some level of competitive advantage. For example, the banking industry uses data warehousing to make financial decisions and to produce financial statements [18]. In the ultimate attempt at market segmentation, marketers and catalog retailers use data warehousing in an attempt to find market segments of one [13]. Grocery and retail supplies can use data warehousing and data mining to provide evidence for the demand of the products, this information can be very valuable in promoting those products to buyers and managers of retail stores [27]. Insurance companies also employ these technologies to uncover trends within policy type as and location of policy holders [15]. Procedures such as data mining are employed in an attempt to obtain valuable information that can be used in customer relationship management (CRM), business process management, and supply-chain management [26]. The preceding examples provide some insight into existing business uses of data warehousing and data mining and verify that they are well accepted and effective means of analyzing business data for critical decision making.

Aside from the obvious benefits of using data warehousing and data mining, both processes are also resource intensive and great care must be employed in the design of the databases used. Companies employ huge databases and, ultimately, use both methods in order to "draw meaningful conclusions, extract knowledge, and acquire models" from them [28]. In order for this process to occur however, businesses must use a "multidimensional database system" that is supported by sound indexing methods. Sound indexing begins with a well designed system. [3] emphasizes the importance of evaluating a company's data requirements and clean the data and transform it to provide adequate performance. Specifically, [3] states: "companies must consider performance when processing data, because slow running applications waste not only time but money as well." Due to the complexity of very large databases and their associated data mining applications the design process is still not well defined [22]. This situation implies the need for additional research that will provide evidence of what techniques lead to better performance and ease of use. However, there is a basic database design strategy which has proved effective in general data base applications and point two of that suggested technique deals specifically with indexing

methodology [6]. Further, [6] recognizes the value of indexing optimization and the difficulty in getting it right, so he suggests collecting a day or two of data for testing before putting the system into production. [28] state that the ETL process is a prime area of focus in regard to improving performance with in very large databases and feel the way the database is mapped and data extracted can have a major impact. Last, [23] recognize that there is variance in applications and customization of methods may be required which supports the need for research in regard to the potential of indexing optimization in a variety of applications.

EXPERIMENT DESCRIPTION

A fairly complex database was used from a Midwestern University that contained information about academic records. The size of this database was about 4.4GB for the data mart and the staging portion was 8.9GB. It was organized into 154 tables on the data mart level and there were 166 tables in the staging data base. Further, it contained 4,400,768 records. It was loaded into MS SQL server (version 2005 enterprise) housed on a Windows 2003 64bit based system. The database computing unit contained two (dual core) AMD Opteron processors with a clock speed of 2Ghtz and 4GB of memory. The data was stored on a SCSI drive array and the network connection was 100Mbs Ethernet. An experiment was devised in which a series of SQL view inquires were submitted via remote desktop through SQL Server Management Studio. It generated a number of inquiries related to student courses taken during various terms and the grades obtained. These inquiries were run again the database resulting in from 4,055 to 10,624 physical reads taking place depending on the indexing methodology used. The indexing configurations were designed to improve upon the current indexing method (method 1below) which had resulted in the first attempt to improve upon the original configuration where flat tables and a single primary key were the search tools used. Under that configuration it typically took between 7 and 8 minutes to complete the inquiries offered by the script.

Four different indexing configurations were tested:

- The first method featured one scalar function, three table functions and three of the four fields referenced indexes on lookup tables.
- The second method removed the scalar function and used four table value functions.
- The third method was the same as the second except all functions referenced index tables.
- The fourth method was the same as the third except two additional composite indexes on the main tables were included.

The goal was to determine the potential speed up of varying the indexing method. The logic being that disk delay is an integral part of the delay a client will experience while making a remote inquiry to a database server. While optimizing the network and distributing the database across multiple nodes can have significant effects on the improvement of response time a fundamental first step in this optimization equation is a well thought out indexing strategy. Often in accessing data from a large database, disk or IO delay receives prime consideration as it should. However, indexing strategy can focus on preloading some of the data into memory (as determined by the SQL optimizer) which will provide superior performance when compared to strategies that don't preplan and read directly from disk record by record. In this case we attempted to anticipate access patterns and preloaded tables into memory (and reduce the amount of data to be read) and used indexes, composite keys and functions to optimize the search process. Further, the computing system used had a total of four cores available so we tried to direct the SQL Server optimizer to maximize parallelism.

The *first method* built upon the strategy above by using a scalar function to pass the fields to a function and return the primary key. This further allows us to reduce joins; hence it is still a table scan using a lookup on a non-indexed field. The three table functions allowed us to return the tables to memory and pass a select statement which allowed an inline implicit join thereby reducing the reads required by the hard drive(s). Indexing on three of the four fields was included as a short cut to find the record. This logic reduces sort manipulation because the search doesn't have to traverse the table for every lookup, thus reducing number of reads. All of this is related to the goal of obtaining an arrangement of table functions that will permit parallelism and with this method two of the four cores were used.

The *second method* basically converted the scalar function into a table function. The goal was to get even more information into memory and improve upon the parallelism. Now all indexing is being called by a function, but there is still one field not indexed. The *third method* provided the indexing for the fourth field. The *fourth method* attempted to determine the effectiveness of providing some of the search logic on the key level through the use of composite keys. Because tables related to student

courses and the terms taken needed to be joined resulting in a data set that would contain information for all students during all terms for all course ever recorded it was thought a composite key might provide a short cut. The composite keys used were: term, course, and student ID. In methods 2-4 parallelism was obtained across all four cores.

RESULTS

The results are depicted below in Table 1. To ascertain whether there was consistency within each method a series of three trials were run per method. The variance was only a few seconds per trial so the table values reported is that of the median values obtained. Generally speaking the results confirm that a performance gain can be attained by varying the indexing method. However, it was hoped that performance would improve as additional complex methods were added. The results indicate a nice performance gain from method 1 to 2, but slight degradation appears from method 2 to 3 and method 3 to 4.

First, there is a very nice improvement from method 1 to method 2(2:48 down from 5:45). Further if one considers this 2:48 value in regard to the original flat table method that was in the 7 to 8 minute range the true effectiveness of indexing comes into perspective. One then might suspect that methods 3 and 4 may have some value in a larger database with more complexity, but the overhead their additional logic places on the system is counterproductive in this scenario.

Table 1: Performance Data for the 4 Different Indexing Methods

Indexing Method	**1**	**2**	**3**	**4**
Run time (mm:ss)	5:45	2:48	2:56	3:02
CPU time (megaticks)	353	12	12	12
Physical reads	4,055	10,624	10,515	10,514

It is interesting to note the different ratio of CPU time to physical reads. Method one takes way more CPU time (perhaps loading information into memory as the result of the scalar) but only requires about half of the physical reads. Although the goal is generally to minimize disk traffic doing so at the cost of 30 times the CPU time while only reducing the reads in half is not effective in this case. Interestingly, the same basic ratios are obtained for methods 2-4 which probably indicates that the conversion of the scalar to a table function was the most effective action in the strategy used herein. The CPU usage patterns are also interesting. Figure 1 provides the information for method one while Figure 2 provides the information for method two. Figures are not provided for methods 3 and 4 because they are analogous in CPU to physical read ratios to method 2 and look very similar. In both figures the Y axis is usage percent and the X axis is time.

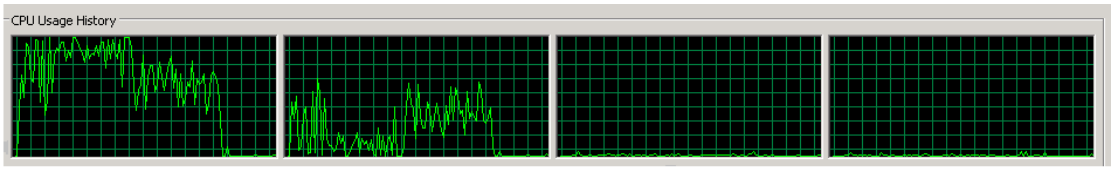

Figure 1. CPU Usage Method 1

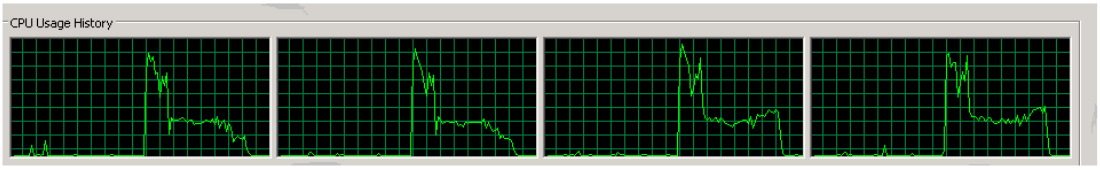

Figure 2. CPU Usage Method 2

A quick glance at the figures reveals that method one is only able to take advantage of two CPU (cores) while method two is able to utilize all four. Further, particularly on the first CPU method one exceeds the 80% threshold level on numerous occasions. Basic queuing theory warns that in such situations response time tends to deteriate exponentially. Hence this processor bound situation when compared to the CPU graphs for method two which helps explain the difference in performance. Further, Figure 2 provided insight into why additional performance gains were not realized in methods 3 and 4. An analysis of the basic shape on each CPU shows a spike and then a plateau. The spike indicates the CPU overhead for loading the tables, once this preplanning logic is in place the actual records still need to be extracted from disk. The plateau indicates that the system is no longer processor bound, but now disk IO bound. Once that happens the disk becomes the bottle neck and traditional improvements in the search logic will have minimal effect. This appears to have happened herein and in fact the additional logic overhead made methods 3 and 4 slightly less efficient than method two because of the added CPU cycles needed. In other words the improved logic could not overcome the disk IO bottleneck. One might suspect a more efficient hard drive schema would change this situation and methods 3 and 4 might yield performance beyond what was observed in method 2.

BUSINESS VALUE OF IMPROVED EFFICIENCY

In most cases, whether from the consumer, business, or government perspective of online activity (e.g. query retrieval, e-commerce, or other use), users will only wait so long to obtain results before giving up. Therefore, it is imperative to take response time into consideration when working in a distributed environment. In fact, because response time is so important in e-commerce, programmers have created indices that track these times [16]. There are a number of combinations and methods of indices that can be employed and can be used to distribute a data warehouse.

Data mining is often used in decision support systems and, as a result, when the data is updated must always be taken into consideration. For instance, traditionally financial data is updated on a period ending basis (e.g. month, quarter, or fiscal year end). If the system is user friendly, in addition to being accurate, the business value of data mining may be increased. The frequency which the database is updated or modified can have a significant impact on how best to distribute the database [11]. Further, there is not one specific method of partitioning that is best under all circumstances, therefore, it is crucial to consider that due to various types of and uses for distributed databases customization may be needed [11]. Currently, potential business value of data mining is incredible. If the efficiency of retrieving information can be increased, researchers are one step closer to making the integration of multiple data locations as seamless as possible.

The findings presented herein are encouraging in the sense that indexing method can actually improve efficiency. That efficiency is a critical foundation because, as [25] asserts, there is addition overhead involved when for security, privacy, or other reasons there needs to be various logical views of the database to reduce "database inference." Also, the traditional paradigm of single physical data stores may not serve modern data analysis needs, especially in the world of multi-terabyte and petabyte data warehouses [14]. As data warehouses continue to evolve, there will no doubt be more challenges on the horizon. Fortunately, from the business and end-user perspectives, if access time is kept at reasonable levels, the physical location as well as the size and complexity of the data warehouse will be of little concern. Sound indexing can help lessen that concern.

DISCUSSION AND CONCLUSIONS

The results definitely indicate that performance advantages can be obtained when indexing is applied. Given the size and complexity of many databases in use today the one used herein while not towards the top of sophistication was not trivial either. While the results were successful and reduced the run time in half the goal of obtaining better performance when moving to each successive method was not realized. However, the data was very useful in analyzing why the methods 3 and 4 did not provide the desired results. That data indicated that it was very important to be aware of limitations on the hardware level. For example, method 1 may have performed better if additional CPU cycles were available, but probably only up to the point where disk IO became an issue and the run times observed in the other methods would probably impose the same limitation. It was also clear no matter how well thought out the indexing was that without improved disk hardware that the ~2:45 barrier would not be improved upon.

Too often database design is done with little or no consideration to matching the system to the hardware it will run on. Certainly, there is a tendency to assume that hardware is so cheap that it is cost

effective to upgrade it after the fact, rather than spend personnel time to analyze where the bottleneck actually is and there is fine line here. Often with relatively simple systems the hardware upgrade is tenable. However, in complex systems the probability of success lessens. In this case adding CPU would probably do little in regard to breaking the 2:45 second barrier. That being said the methodology used herein, while not the ultimate in sophistication does provide useful decision making data regarding system upgrades while not requiring massive amount of personnel time to generate.

In meeting the goal of speeding up database access for web accessible databases and data mining applications the speed up observed would provide a respectable foundation for a client/server system. This speedup is important, particularly in a growing systems because added delays often propagate exponentially as delays on the server application, network and client software level are added to the transaction.

FUTURE RESEARCH

Because the intensity of this experiment was relatively moderate, it would be interesting to replicate the basic model logic on a larger and more sophisticated database. However, future research most logically should center on proving the premise that more efficient disk hardware would provide better performance across all methods used. In addition to simply looking at single disks or arrays with better performance on the hardware level it would be appropriate to look at the concept of distributing the database across multiple nodes (computers). Research in this area has shown that the distribution method can make a difference [7]. Specifically, a load balancing algorithm in which the individual units were queried to determine the unit with lowest CPU utilization improved performance (reduced delay) by an approximate factor of five times. However, the characteristics of the client traffic in [7] were different from the data herein.

In [7] the inquiries were typical inquiries designed to call a single record from a relatively small database. The process limitation was more likely to be CPU than disk. In this study, a larger database was used with intense inquiries resulting in the disk being the limiting factor. It would be interesting to replicate the indexing portion of this study this and combine it with multiple nodes and load balancing in an effort to reduce the disk IO bottleneck observed herein.

REFERENCES

1. Abdallah, M & Le, H. C. (2005). Scalable range query processing for large scale distributed database applications. *Proceedings of Parallel and Distributed Computing and Systems*, Phoenix, AZ, Nov. 14-16, 2005. (p. 466). Anaheim: ACTA Press.
2. Abdallah, M. & Temal, L. (2004). GriDB: A scalable distributed database sharing system for grid environments. *Proceedings of Advances in Computer Science and Technology*, St. Thomas, Virgin Islands, Nov. 22-24. (p. 431). Anaheim: ACTA Press.
3. Abramson, C. (2006). Take your ETL processes to the next level utilizing p-ETL. *What Works* [Online]. Retrieved September 10, 2007, from http://www.tdwi.org/Publications/WhatWorks/display.aspx?id=7973.
4. Awan, I. & Younas, M. (2004). Analytical modeling of priority commits protocol for reliable web applications. *Proceedings of the ACM Symposium on Applied Computing*, Nicosia, Cyprus, March 14-17. (pp. 313-317). New York: ACM.
5. Bala, J., Baik, S., Hadjarian, A., Gogia, B. K., & Manthrone, C. (2002). Application of a distributed data mining approach to network intrusion detection. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: part 3*, Bologna, Italy, July 15-19. (pp. 1419-1420). New York: ACM Press.
6. Bostrup, T. (2003). Database performance philosophy. *15 Seconds* [Online]. Retrieved March 14, 2008, from http://www.15seconds.com/issue/040115.htm.
7. Brown, C., Guster, D., & Krzenski, S. (2007). Can distributed databases provide an effective means of speeding up web access times? *Journal of Information Technology Management, XVIII*: 1-15.
8. Cannataro, M., Pugliese, A., & Trunfio, P. (2002). Distributed data mining on grids: Services, tools, and applications. *IEEE Transactions on Systems, Man, and Cybernetics 34*(6), 2451-2465.
9. Ceri, S. & Pelagatti, G. (1982). Allocation of operations in distributed database access. *Transactions on Computers, 31*(2), 119-129.
10. Datta, S., Bhaduri, K., Giannella, C., Wolff, R., & Kargupta, H. (2006). Distributed data mining in peer-to-peer networks. *IEEE Internet Computing, 10*(4): 18-26.

11. Diehr, G., Saharia, A. N., & Chao, D. (1990). Maintaining remote decision support databases. *Journal of Management Information Systems*, *7*(2): 111-138.
12. Donachy, P., Rasch, J., Harmer, T. J., Bearder, S., Perrot, R., & Beckett, M. (2003, September). Grid enabled distributed data mining and conversion of unstructured data. *Proceedings of UK e-Science All Hands Meeting.* Proceedings of UK e-Science All Hands Meeting, Nottingham, U.K.
13. Forcht, K., & Cochran, K. (1999). Using Data Mining and Data Warehousing Techniques. *Industrial Management and Data Systems, 9* (5): *189-196.*
14. Foster, I., & Grossman, R. L. (2003). Data Integration in a Bandwidth-Rich World. *Communications of the ACM*, *46*(11): 50-57.
15. Inmon, W. (1996). *Building the Data Warehouse, Second Edition*. New York, NY: John Wiley & Sons.
16. KEYNOTE.COM. (n.d.). *Performance management: Performance indices*. [Online]. Retrieved September 10, 2007, from http://www.keynote.com/solutions/performance_indices/ecommerce/ecommerce.html.
17. Lao, P., Lu, K., Shi, Z., & He, Q. (2007). Distributed data mining in grid computing environments. *Future Generation Computer Systems*, *23*(1): 84-91.
18. Ma, C., Chou, D., & Yen, D. (2000). Data warehousing, technology assessment and management. *Industrial Management and Data Systems*, *100*(3): 125-134.
19. Orlando, S., Palmerini, P., Perego, R., & Silvestri, F. (2002). Scheduling high performance data mining tasks on a data grid environment. *Proceedings of Europar Conference of Parallel Processing*, London, U.K., August 27-30 (pp. 375 – 384). Springer-Verlag.
20. Palace, B. (1996). *Data mining*. [Online]. Retrieved March 14, 2007, from http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/index.htm
21. Parthasarathy, S., Ghoting, A., & Otey, M. E. (2006). A chapter in data streams: Models and algorithms. *A Survey of Distributed Mining of Data Streams*. Berlin: Springer.
22. Rizzi, S., Abello, A., Lechtenborger, J., & Trujillo, J. (2006). Research in data warehouse modeling and design: Dead or alive. *Proceedings of the 9th ACM International Workshop on Data Warehousing and OLAP*. Arlington, VA, Nov. 10 (pp. 3-10). New York: ACM Press.
23. Simitsis, A., Vassiliadis, P. & Sellis, T. (2005). Optimizing ETL processes in data warehouses. *Proceedings of the 21st International Conference of Data engineering*., Washington, DC: IEEE Computer Society, April 5-8 (pp. 564-575)
24. Talia, D. (2006). Grid-based distributed data mining systems. Algorithms and services. *Proceeding Workshop HPDM 2006 at the SIAM Data Mining Conference*, Bethesda, MD, April 20-22. Philadelphia: Society for Industrial and Applied Mathematics.
25. Tracy, J., Liwu Chang, J., & Moskowitz, I. S. (2003). An agent-based approach to inference prevention in distributed database systems international. *Journal on Artificial Intelligence Tools, 12* (3): 297-313.
26. Viswanathan, M., Yang, Y., & Whangbo, T. K. (2005). Lecture notes in computer science. *Distributed Data Mining on Clusters with Bayesian Mixture Modeling* 3613: 1207-1216.
27. Wells, J., & Hess, T. (2002). Understanding decision-making in data warehousing and related decision support systems: An explanatory study of a customer relationship management application. *Information Resources Management Journal*: 16-32.
28. Wenlong L., Li, E. Jaleel, A., Jiulong, S., Yurong C., Qigang, W., Iyer, R., Illikkal, R., Yimin, Z., Dong L., Liao, M., Wei, W. & Du, J. (2007). Understanding the memory performance of data-mining workloads on small, medium, and large-scale CMPs using hardware-software co-simulation. *Performance Analysis of Systems & Software*, San Jose, CA, April 25-27, 2007, (pp. 35 – 43). Santa Clara: Intel Corporation.
29. Yubin, B., Jie, S., Fanglin, L., Daling, W., & Ge, Y. (2007). Study and implementation of a new SQL-based ETL approach. *Wuhan University Journal of Natural Science*, *12*(5): 804-808.