IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 1, Feb-Mar, 2014
ISSN: 2320 - 8791
www.ijreat.org

A Survey on Streaming Data Warehouses

Bolla Saikiran[1], Kolla Morarjee[2]
[1]M.Tech Scholor,Department of Computer Science and Engineering, CMR Institute of Technology
Medchal, Hyderabad, Andhra Pradesh, India

[2]Assistant Professor, Department of Computer Science and Engineering, CMR Institute of Technology
Medchal, Hyderabad, Andhra Pradesh, India

Abstract

Update scheduling in streaming data warehouses, which combine the features of traditional data warehouses and data stream systems. We want to gather the literature related to scalable, updated, real time and dynamic scheduling in streaming warehouses. With this analysis after referring all papers in this area we will came to the best methodology which is suitable for combining streaming and data warehouses by limiting the staleness. Our main objective is to focus on integration of historical data and current data by limiting staleness. Then we want to adopt the efficient integration technique to real business applications where large amount of data is used.

Keywords: *Streaming data warehouse, Data Stream, Data Warehouse.*

1. Introduction

Data mining is the process of analyzing data from different perspectives and summarizing it into useful information that can be used to increase revenue, cuts costs, or both. The goal of a streaming warehouse is to propagate new data across all the relevant tables and views as quickly as possible. Once new data are loaded, the applications and triggers defined on the warehouse can take immediate action. This allows businesses to make decisions in nearly real time, which may lead to increased profits, improved customer satisfaction, and prevention of serious problems that could develop if no action was taken. Real-time scheduling is a well-studied topic with a lengthy literature. a typical hard real time system, jobs must be completed before their deadlines a simple metric to understand and to prove results . In a firm real-time system, jobs can miss their deadlines, and if they do, they are discarded. Streaming warehouse must load all of the data that arrive therefore no updates can be discarded. In a soft real-time system, late jobs are allowed to stay in the system, and the performance metric is lateness which is the difference between the completion times of late jobs and their deadlines. A stream warehouse enables queries that seamlessly range from real-time alerting and diagnostics to long-term data mining.

Data Stream Management Systems (DSMS) support simple analyses on recently arrived data in real time. Streaming warehouses such as Data Depot combine the features of these two systems by maintaining a unified view of current and historical data. The goal of a streaming warehouse is to propagate new data across all the relevant tables and views as quickly as possible. Recent work on streaming warehouses has focused on speeding up the Extract-Transform-Load (ETL) process. A streaming data warehouse maintains two types of tables: base and derived. Each table may be stored partially or wholly on disk. A base table is loaded directly from a data stream. A derived table is a materialized view defined over one or more tables. In practice, warehouse tables are horizontally partitioned by time so that only a small number of recent partitions are affected by updates[14]. While traditional data warehouses are typically refreshed during downtimes, streaming warehouses are updated as new data arrive. The traditional data warehouses are typically refreshed during downtimes, streaming warehouses are updated as new data arrive. The problem with this approach is that new data may arrive on multiple streams, but there is no mechanism for limiting the number of tables that can be updated simultaneously.

Many stream-based applications have sophisticated data processing requirements and real-time performance expectations that need to be met under high-volume, time varying data streams Update Scheduling in Streaming Data Warehouses, which combine the features of traditional data warehouses and data stream systems. In streaming warehouses, whenever new data arrive, it will be push up for appending into the streaming warehouses. In traditional warehouses data will get refreshed during the non-business hours[11]. Data warehouses are typically refreshed in a batch fashion: the updates from data sources are buffered during working hours, then loaded through the Extraction-Transformation-Loading (ETL) process when the warehouse is quiescent. This clean separation between querying and updating is a fundamental assumption of conventional data warehousing applications, and clearly simplifies several aspects of the implementation.

A real-time warehouse scheduler must simultaneously pursue multiple goals. First, the ultimate goal is to ensure that queries on the warehouse see data that is as up-to date as possible. Second, the scheduler must maintain data consistency in the sense that a derived table must be equivalent to running its defining query over the state of its source table. Third, the scheduler must handle heterogeneous task sets, as different streams may have different data rates and inter-arrival times; the main challenge here is to prevent short-lived up-dates from being blocked by long-running updates. Fourth, the scheduler must support multiprocessor machines in order to accommodate high update and query rates.

2. Literature Review

The Literature Review of various scheduling algorithms in streaming data warehouses are discussed below.

B. Adelberg H. Garcia-Molina B. Kao[17] proposed "Applying Update Streams in a Soft Real-Time Database System". In this paper, they discuss the various properties of updates and views that affect this tradeoff . They also examine, through simulation, four algorithms for scheduling transactions and installing up-dates in a soft real-time database. They also evaluated the algorithms under many other scenarios. For example, they studied the effects of changing the size of the maximum age of objects used in MA, the effects of applying the updates in the update queue in LIFO or FIFO order, and the effects of varying the update arrival rate. They discuss one strategy for processing burst streams — *adaptive, load-aware scheduling* of query operators to minimize resource consumption during times of peak load. They show that the choice of an operator scheduling strategy can have significant impact on the run-time system memory usage. Then they present *Chain scheduling*, an operator scheduling strategy for data stream systems that is near-optimal in minimizing run-time memory usage for any collection of single stream queries involving selections, projections, and foreign-key joins with stored relations. They studied the problem of operator scheduling in data stream systems, with the goal of minimizing memory requirements for buffering tuples. They proposed the Chain scheduling strategy and proved its optimality for the case of single-stream queries with selections, projections, and foreign key joins with static stored relations. Furthermore, they showed that Chain scheduling performs well for other types of queries, including queries with sliding-window joins.

Moustafa A.Hammad et. al.[16], Proposed "Scheduling for shared window joins over data streams". In this paper , they address the shared execution of windowed joins a core operator for CQ systems they show that the strategy used in systems to date has a previously unreported performance flaw that can negatively small windows, then they propose two new execution strategies for shared joins they evaluate the alternatives using both analytical models and implementation in a DBMS the results show that one strategy, called MQT provides the best performance over a range of workload settings. They have described and evaluated three scheduling algorithms that priorities such shared execution to reduce the average response time per query while preserving their original semantics LWO was used previously,SWF and MQT were developed as part of the work described here SWF directly addressed the performance flaw identified for LWO,MQT was motivated by the tradeoffs between LWO and SWF as identified by an analytical study of the two approaches showed that the MQT algorithm provides up to 60% improvement in average response time over the LWO algorithm .The experiments also demonstrated that the benefits of MQT come at the cost of only a small increase in memory overhead.

Don Carney et. al.[15] proposed "Operator Scheduling in a Data Stream Manager". Many stream-based applications have sophisticated data processing requirements and real-time performance expectations that need to be met under high-volume, time-varying data streams. In order to address these challenges, they propose novel operator scheduling approaches that specify which operators to schedule in which order to schedule the operators, and how many tuples to process at each execution step. They also discuss application-aware extensions that make scheduling decisions according to per-application Quality of Service (QoS) specifications. Finally, they present prototype-based experimental results that characterize the efficiency and effectiveness of our approaches under various stream workloads and processing scenarios. This paper presents an experimental investigation of scheduling algorithms for stream data management systems. It demonstrates that the effect of system overheads can have a profound impact on real system performance. They have also discussed exactly how these overheads are affected in a running stream data manager. They also addressed QoS issues and extended our basic algorithms to address application-specific QoS expectations. They described an approximation technique based on bucketing that trades off scheduling quality with scheduling overhead. The overriding message of this paper is that to build a practical data stream management system, one must ensure that scheduler overhead be small relative to useful work. They have provided some interesting results in this direction by focusing on batching techniques.

Shivnathbabu et. al.[13] proposed "Exploiting *k*-Constraints to Reduce Memory Overhead in Continuous Queries Over Data Streams". They present a query processing architecture, called *k-Mon*, that detects useful *k*-constraints automatically and exploits the constraints to reduce run-time state for a wide range of continuous queries. Experimental results showed dramatic state reduction, while only modest computational overhead was incurred for our constraint monitoring and query execution algorithms. In this article they introduced the concept of *k*-constraints: "relaxed" constraints that are more likely to hold in data stream environments than their strict counterparts. They showed empirically that exploiting *k*-constraints can be very effective at reducing the memory requirement for continuous SPJ queries over streams, and that *k*-constraints can be monitored and incorporated into query processing with low computational overhead. Finally, they presented a unified query-processing framework for exploiting *k*-constraints that incorporates our execution and monitoring algorithms.

Mohammad Hossein et. al.[12] Proposed "Scheduling to Minimize Staleness and Stretch in Real-Time Data Warehouses". they study scheduling algorithms for loading data feeds into real time data warehouses, which are used in applications such as IP network monitoring, online canonical trading, and credit card fraud detection. Their first objective is to schedule the updates on one or more processors in a way that minimizes the total staleness .In order to ensure fairness, our second objective is to limit the maximum \stretch", which they denote (roughly) as the ratio between the duration of time an update waits till it is finished being processed, and the length of the update. They prove that any online non preemptive algorithm, no processor of which is ever voluntarily idle, incurs a staleness at most a constant factor larger than an obvious lower bound on total staleness They give a constant-stretch algorithm, provided that the processors are succulently fast, for the quasi periodic model, in which tables can be clustered into a few groups such that the update frequencies within each group vary by at most a constant factor. Finally, they show that our constant-stretch algorithm is also constant-competitive(subject to the same proviso on processor speed) in the quasi periodic model with respect to total weighted staleness, where tables are assigned weights that receive their priorities. In this paper, they studied the complexity of scheduling data-loading jobs to minimize the staleness of a real time stream warehouse. They proved that any on-line non preemptive algorithm that is never voluntarily idle achieves a constant competitive ratio with respect to the total staleness of all tables in the warehouse, provided that the processors are succulently fast. They also showed that stretch and weighted staleness can be bounded under certain conditions on the processor speed and on the arrival times of new data.

Lukasz Golab et. al. [10] proposed "Scalable Scheduling of Updates in Streaming Data Warehouses". In this paper they model the streaming warehouse update problem as a scheduling problem, where jobs correspond to processes that load new data into tables, and whose objective is to minimize data staleness over time and also present a suite of update scheduling algorithms and extensive simulation experiments to map out factors which affect their performance. The goal of a streaming warehouse is to propagate new data across all the relevant tables and views as quickly as possible.

L. Gladis Flower1 et. al.[9] proposed" Updating the jobs in Streaming Data Warehouse using scheduling framework". In this paper, they motivated, formalized, and solved the problem of non preemptively scheduling updates in a real time streaming warehouse. They proposed the notion of average staleness as a scheduling metric and presented scheduling algorithms designed to handle the complex environment of a streaming data warehouse. Then they proposed a scheduling framework that assigns jobs to processing tracks and uses basic algorithms to schedule jobs within a track. The main feature of our framework is the ability to reserve resources for short jobs that often correspond to important frequently refreshed tables, while avoiding the inefficiencies associated with partitioned scheduling techniques. They had implemented some of the proposed algorithms in the Data Depot streaming warehouse, which is currently used for several very large warehousing projects within AT&T. As future work, they plan to extend our framework with new basic algorithms.

P. Urmila et. al.[8] proposed "Scheduling of Updates in Data Warehouses". In this paper they develop a theory of temporal consistency for stream warehouses that allows for multiple consistency levels. they model the streaming warehouse update problem as scheduling problem, where jobs correspond to processes that load new data into tables, and whose objective is to minimize data staleness over time. Unlike traditional data sets, stream data flow in and out of a computer system *continuously* and with varying update rates. They are *temporally ordered, fast changing, massive, and potentially infinite*. To discover knowledge or patterns from data streams, it is necessary to develop single-scan, on-line, multilevel, multidimensional stream processing and analysis methods. There has also been work on supporting various warehouse maintenance policies, such as immediate, deferred and periodic There has been little work on choosing, of all the tables that are now out-of-date due to the arrival of new data, which one should update next. This is exactly the problem they study in this paper. In addition to understanding data semantics and query results, another use for consistency is to minimize the number of base table and view updates in a warehouse. Averages staleness as scheduling metric and presented scheduling algorithms designed to handle complex environment of a streaming data warehouse. they then proposed a scheduling framework that assigns jobs to processing tracks and also uses the basic algorithms to schedule jobs within a same.

Dattatray G.Modani,Vinod S.Badgujar,,A.Simhadribabu,[7] Proposed "Rapid Scalable Scheduling for Updates in Streaming Data Warehouses". This paper includes a streaming data warehouse update problem as a scheduling problem where jobs correspond to the process that load new data into tables and the objective is to minimize data staleness over time. They proposed scheduling framework that handles the complications encountered by a stream warehouse: view hierarchies and priorities, data consistency, inability to pre-empt updates, heterogeneity of update jobs caused by different inter arrival times and data volumes among different sources and transient overload. They proposed the notion of average staleness as scheduling metric and presented scheduling algorithms designed to handle the complex environment of a streaming data warehouse. Then they proposed a scheduling framework that assigns jobs to processing tracks and uses basic algorithms to schedule jobs within a track. The main feature of framework is the ability to reserve resources for short jobs that often correspond to important frequently refreshed tables, while avoiding the inefficiencies associated with partitioned scheduling techniques.

Dattatray G.Modani,A.Simhadribabu,Jayant D.Bokefode,[6] Proposed "A Novel Approach For Updates In Streaming Data Warehouses By Scalable Scheduling **".**This paper includes a streaming data warehouse update problem as a scheduling problem where jobs correspond to the process that load new data into tables and the objective is to minimize data staleness over time. They proposed scheduling framework that handles the complications encountered by a stream warehouse. The need for on-line warehouse refreshment introduces several challenges in the implementation of data warehouse transformations, with respect to their Execution time and their overhead to the warehouse processes. The problem with this approach is that new data may arrive on multiple streams, but there is no mechanism for limiting the number of tables that can be updated simultaneously**.** The formalized and solved the problem of no preemptively scheduling updates in a real-time streaming warehouse. They proposed the notion of average staleness as scheduling metric and presented scheduling algorithms designed to handle the complex environment of a streaming data warehouse. Then they proposed a scheduling framework that assigns jobs to processing tracks and uses basic algorithms to schedule jobs within a track. The main feature of framework is the ability to reserve resources for short jobs that often correspond to important frequently refreshed tables, while avoiding the inefficiencies associated with partitioned scheduling techniques.

Sathya.T [5] proposed "Updates of Scheduling in Streaming Warehouses". A stream warehouse is a Data Stream Management System (DSMS) that stores a very long history, e.g. years or decades or equivalently a data warehouse that is continuously loaded. Which are the combining features of data stream system and traditional data warehouse So, in each time the update process to be done in DWH(data warehouse).This is the main problem of scheduling. To overcome this problem to propose update scheduling technique is used. Which is used to handle the multiple problems of stream warehouse? Particularly the staleness problem to be minimized with the help of update scheduling. Which is efficiently used to update the jobs by different arrival times? The objective is to schedule the updates on one or more processors in a way that minimizes the total staleness. In traditional applications, warehouses are updated periodically (e.g., every night or once a week) and data analysis is done off-line. In contrast, real time warehouses also known as active warehouses continually load incoming data feeds to support time critical analyses. For instance, an Internet Service Provider (ISP) may collect streams of network configuration and performance data generated by remote sources in nearly real time. The goal of a streaming warehouse is to propagate new data across all the relevant tables and views as quickly as possible. They proposed scheduling algorithm is used to handle complex streaming data warehouse. This algorithm issued to assign the jobs to processing tracks and basic algorithms are used to schedule jobs within the track.

Mr.S.S.Boopathy, Mr.K.M.Subramanian [4] proposed "Improved Scheduling and Minimized Updates in Data Warehouses". In the proposed work, Particles Swarm Optimization (PSO) algorithm is used. It is a computational method that can optimizes the problem by iteratively trying to improve the possible solutions with regard to a given measure of quality. In their setting, the arrival of a set of new data releases an *update* that seeks to append the data to the corresponding table. They analytically derive *a priori guarantees* on the quality of data in the warehouse expressed as upper bounds on *table staleness,* which is the discrepancy between the current time and the maximum timestamp of a record uploaded in the table so far. Their main innovation is the multi track Proportional algorithm for scheduling the large and heterogeneous job sets encountered by a streaming warehouse.

S.M Subhani1, G.Srinivas Reddy [3] proposed "Dynamic Updates on Streaming of Data ware Houses Explore Tradeoffs". In this paper they will extend this traditional approach to new granularity algorithms for accessing efficient updates multiple tables together. they intend to explore the tradeoffs between update efficiency and minimizing staleness in the context. Their main innovation is the Multi track Proportional algorithm for scheduling the large and heterogeneous job sets encountered by a streaming warehouse additionally, they propose an update chopping to deal with transient overload. They proposed the notion of average staleness as scheduling metric and presented scheduling algorithms designed to handle the complex environment of a streaming data warehouse. Then proposed a scheduling framework that assigns jobs to processing tracks and uses basic algorithms to schedule jobs within a track.

A. Sreeja et. al [2]. proposed "Load Balancer Scheduling Over Streaming Data in Federated Databases". This paper includes a streaming data warehouse update problem as a scheduling problem where jobs correspond to the process that load new data into tables and the objective is to minimize data staleness over time. They proposed scheduling framework that handles the complications encountered by a stream warehouse: view hierarchies and priorities, data consistency, inability to pre-empt updates, heterogeneity of update jobs caused by different inter arrival times and data volumes among different sources and transient overload. The formalized and solved the problem of no preemptively scheduling updates in a real-time streaming warehouse. They proposed the notion of average staleness as scheduling metric and presented scheduling algorithms designed to handle the complex environment of a streaming data warehouse. Then they proposed a scheduling framework that assigns jobs to processing tracks and uses basic algorithms to schedule jobs within a track. The main feature of framework is the ability to reserve resources for short jobs that often correspond to important frequently refreshed Tables, while avoiding the inefficiencies associated with partitioned scheduling techniques.

S. M. Subhani1, M. Nagendramma proposed [1] "Minimize Staleness and Stretch in Streaming Data Warehouses". They study scheduling algorithms for loading data feeds into real time data warehouses, which are used in applications such as IP network monitoring, online financial trading, and credit card fraud detection. They discuss update scheduling in streaming data warehouses, which combine the features of traditional data warehouses and data stream systems. While traditional data warehouses are typically refreshed during downtimes, streaming warehouses are updated as new data arrive. In this paper they develop a theory of temporal consistency for stream warehouses that allows for multiple consistency levels. They model the streaming warehouse update problem as a scheduling problem, where jobs correspond to processes that load new data into tables , and whose objective is to minimize data staleness over time. they studied the complexity of scheduling data loading jobs to minimize the staleness of a real time stream warehouse. they proved that any on-line non-preemptive algorithm that is never voluntarily idle achieves a constant competitive ratio with respect to the total staleness of all tables in the warehouse, provided that the processors are sufficiently fast. They projected the notion of averages staleness as a scheduling metric and presented scheduling algorithms designed to handle complex environment of a streaming data warehouse. Then they proposed a scheduling framework that assigns jobs to processing tracks and also uses the basic algorithms to schedule jobs within a same.
Shivnath Babu and Jennifer Widom[18] proposed "Exploiting *k*-Constraints to Reduce Memory Overhead in Continuous Queries over Data Streams". Then they present a query execution algorithm that takes constraints over streams into account in

order to reduce memory overhead. If input streams do not adhere to constraints within the specified adherence parameters, our algorithm automatically degrades gracefully to provide continuous approximate answers. They have implemented our approach in a test bed continuous query processor and preliminary experimental results are reported. In this paper they have demonstrated the effectiveness of incorporating constraints into a data stream query processor in order to reduce the memory overhead required for continuous queries. They introduced the important concept of constraints, which are likely to hold in data stream environments even when strict constraints do not hold. they showed empirically that constraints are very effective at reducing the memory requirement in a wide variety of SPJ queries. They showed that precision of query results degrades gracefully in cases when constraints may be compromised.

Neoclis Polyzotis et. al.[19] Proposed "Supporting Streaming Updates in an Active Data Warehouse". Active Data Warehousing has emerged as an alternative to conventional warehousing practices in order to meet the high demand of applications for up-to-date information. The need for on-line warehouse refreshment introduces several challenges in the implementation of data warehouse transformations, with respect to their execution time and their overhead to the warehouse processes. In this paper, they focus on a frequently encountered operation in this context, namely, the join of a fast stream S of source updates with a disk-based relation R, under the constraint of limited memory. They proposed a specialized join algorithm, termed mesh join (MESHJOIN), that compensates for the difference in the access cost of the two join inputs by (a) relying entirely on fast sequential scans of R, and (b)sharing the I/O cost of accessing R across multiple tulles of S. They present an experimental study that validates the performance of MESHJOIN on synthetic and real-life data. Our results verify the scalability of MESHJOIN to fast streams and large relations, and demonstrate its numerous advantages over existing join algorithms .In this paper, they have considered an operation that is commonly encountered in the context of active data warehousing: the join between a fast stream of source updates Sand a disk-based relation R under the constraint of limited memory. they have proposed the *mesh join* (MESHJOIN), a novel join operator that operates under minimum assumptions for the stream and the relation. They have developed a systematic cost model and tuning methodology that accurately associates memory consumption with the incoming stream rate. Finally, they have validated our proposal through an experimental study that has demonstrated its scalability to fast streams and large relations under limited main memory.

3. Conclusion

This paper presents a survey on various streaming data warehouses Scheduling Algorithms that were proposed by earlier researchers for the development in the field of Data Warehouses. Various algorithms and methodologies discussed above will help in developing efficient and effective update scheduling algorithm in streaming data warehouses.

References

[1] S.M.Subani.M.Nagendramma," *Minimize Staleness and stretch in Streaming Data Warehouses"*, IJIR, Vol.2, Issue 9, September 2013.
[2] A. Sreeja, I.V.Sailakshmiharitha, N.Bhaskar, "*Load Balancer Scheduling Over Streaming Data in Federated Databases*" IJRET, Vol.2, Issue 8, August2013.
[3]S.M.Subhani, G.Srinivas Reddy, "*Dynamic Updates on Streaming of Data warehouses Explore Tradeoffs*", IJDCST, Vol.1, Issue5, August 2013.
[4] Mr.S.S.Boopathy, Mr.K.M.Subraman,"*Improved Scheduling and Minimized Updates in Data Warehouses",* IJADMC Vol.1, Issue2, August2013
[5] Satya T,"*Updates of Scheduling in Streaming Warehouses*", ASARIC May 2013
[6]Dattatray G.Modani,A.Simhadribabu,Jayant D.Bokefode, "*A Novel Approach For Updates In Streaming Data Warehouses By Scalable Scheduling* ",IJERT,Vol 2,Issue 7,July 2013.
[7] Dattatray G.Modani,Vinod S.Badgujar,A.Simhadribabu, "*Rapid Scalable Scheduling for Updates in Streaming Data Warehouses*",IJIET,Vol 2,Issue 3,June 2013.
[8] P. Urmila ,K.Siva Rama Krishnan,P.Raja Prakash Rao, "*Scheduling of Updates in Data Warehouses*",IJACMS,vol 3,Issue 3,September 2012.
[9]. L. Gladis Flower1, C.Saravanan," *Updating the jobs in Streaming Data Warehouse using scheduling framework*", IJAEA,Vol 1,Issue 2,2012.
[10]Lukas Golab, Theodore Johnson, and Vladislav Shkapenyuk,"*Scalable Scheduling of Updates in Streaming Data Warehouses*",IEEE Transactions On KDE,vol.24,No.6,June 2012.
[11] M.H. Bateni, L. Golab, M.T. Hajiaghayi, and H. Karloff, "*Scheduling to Minimize Staleness and Stretch in Real-time Data Warehouses*," Proc. 21st Ann. Symp. Parallelism in Algorithms and Architectures (SPAA), pp. 29-38, 2009.
[12] Mohammad Hossein et. al, "*Scheduling to Minimize Staleness and Stretch in Real-Time Data Warehouses*",ACM, August 2009.
[13] S. Babu, U. Srivastava, and J. Widom, "*Exploiting K-constraints to Reduce Memory Overhead in Continuous Queries over Data Streams,*" ACM Trans. Database Systems, vol. 29, no. 3, pp. 545- 580, 2004.
[14] B. Babcock, S. Babu, M. Datar, and R. Motwani, "*Chain: Operator Scheduling for Memory Minimization in Data Stream Systems*," Proc. ACM SIGMOD Int'l Conf. Management of Data, /pp. 253-264, 2003.
[15] D. Carney, U. Cetintemel, A. Rasin, S. Zdonik, M. Cherniack, and M. Stonebreaker, "*Operator Scheduling in a Data Stream Manager*," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB), pp. 838- 849, 2003.
[16] Moustafa A.Hammad et. al., "*Scheduling for shared window joins over data streams*", VLDB Conference 2003.
[17] B. Adelberg, H. Garcia-Molina, and B. Kao, "*Applying Update Streams in a Soft Real-Time Database System*," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 245-256, 1995.
[18] Shivnath Babu and Jennifer Widom,"*Exploiting k-Constraints to Reduce Memory Overhead in Continuous Queries over Data Streams*",Stanford University.
[19] Neoclis Polyzotis et. al. , "*Supporting Streaming Updates in an Active Data Warehouse*".