Data Stream Synopsis Using SaintEtiQ*

Quang-Khai Pham, Noureddine Mouaddib, and Guillaume Raschia

LINA - Polytech'Nantes
ATLAS-GRIM Group
2, rue de la Houssinire, BP 92208
44 322 Nantes cedex 03, France
{Quang-Khai.Pham, Noureddine.Mouaddib, Guillaume.Raschia}@univ-nantes.fr

Abstract. In this paper, a novel approach for building synopses is proposed by using a service and message-oriented architecture. The SaintEtiQ summarization system initially designed for very large stored databases, by its intrinsic features, is capable of dealing with the requirements inherent to the data stream environment. Its incremental maintenance of the output summaries and its scalability allows it to be a serious challenger to existing techniques. The resulting summaries present on the one hand the incoming data in a less precise form but is still on the other hand very informative on the actual content. We expose a novel way of exploiting this semantically rich information for query answering with an approach mid-way between blunt query answering and mid-way between data mining.

1 Introduction

Emerging applications are generating and exploiting data in the form of *data streams*. Such information, as opposed to the traditional way of managing information, is by essence on-line, potentially unlimited and unbounded. Interesting domains of application include network traffic surveillance and administration, financial analysis, sensor data feeds or web applications. The constraints of these application are related to their need for timely answers for decision making purposes. For example, when a broker has to decide whether to buy or sell stocks according to the evolution of different indexes, he needs a final answer within seconds or tens of seconds. When considering the streams, each one corresponding to the real-time evolution of a stock market index, generated by all the stock markets, it is virtually impossible to store the transiting data. Thus, we pinpoint the need for adapted structures for managing this versatile data.

While the data stream domain is relatively new, as opposed to the traditional database paradigm, synopses remain relatively unformalized. Gibbons and Matias define in [6] those, for a class of queries Q, as a function$f(n)$ that provides *(exact or approximate) answers to queries from* Q *that uses* $O(f(n))$ *space for a data set of size* n*, where* $f(n) = O(n^{\epsilon})$ *for some constant* $\epsilon < 1$. The evaluation criteria proposed are thus (i)the coverage of $f(n)$, (ii)the answer quality, (iii)the

* This work supported by the ACI APMD and SemWeb.

H. Legind Larsen et al. (Eds.): FQAS 2006, LNAI 4027, pp. 530–540, 2006.

footprint, (iv)the query time and (v)the computation/update time (the reader is invited to refer to [6] for further details). We retain criteria (i), (ii), (iii) and (v) as a basis to evaluate and position our system comparatively to existing approaches. The challenge that then rises is to find a compromise optimizing these opposing criteria.

Motivating example. A certain number of present applications and needs motivate our research in this direction.

As an example, it is well-known that the only way for car industries to make profit is to reduce their costs. An option is to introduce more and more on-board electronics to replace previously hydraulic and pneumatic systems. Thus, Bosch is working on *motronic* (motorised + electronic) brakes that would be completely independent and would communicate via a wireless connection to a central on-board system. These are monitored in real-time through the sensor feeds they send to the server. Considering the limits of these on-board electronics, it is impracticable to store the data and necessary to provide timely answers.

Formulation of the problem. Recent applications are providing and using more and more input feeds in the form of data streams either structured or not. Due to the host system physical limitations(hard drive access times, computing capabilities, etc...) and the timely answers required, it is not realistic to try to deal with these new inputs with the existing techniques that have been developed for the traditional stored databases and/or data warehouses. On the other hand, even though timely answers are required by such applications, 100% exact answers may not be necessary.

Thus, the main problem for managing and exploiting data streams can partially be answered by providing data structures, called *synopses*, and algorithms to construct and maintain them, in a time and space cost-efficient fashion. These are part of the major criteria retained but these solutions also need to propose answers adapted to the requirements of the application considered, such as the class of queries addressed, and guarantee to a certain extent the quality of the answers provided.

Roadmap. The rest of the paper is organized as follows. First, an overview of the existing techniques and of their performances in designing synopses for data streams is presented in section 2. Then, we will discuss the SaintEtiQ approach for building summaries and the novel decision making-oriented paradigm that we propose for exploiting SaintEtiQ summaries in a data stream environment through section 3. Our perspectives and short & long term work will be introduced in section 4. Finally we will conclude this paper in section 5.

2 Related Work

Recent works on data streams have mostly focused on designing synopses, algorithms and/or (adapting) techniques from the traditional database domain for estimating one dimensional data values.

The most basic approach proposed relied on sampling the data stream. This statistical technique used in the stored database context keeps track of a subset of the data values. In a data stream environment, the performances would be appreciable in terms of space and time processing. However, the unknown size of the data set is a critical issue, thus making the approach poorly applicable.

Pretty similar to the sampling approach, load shedding was notably proposed by Babcock & al. in [4]. The idea was to drop sections of unprocessed data from the incoming stream in a two step process: (i)target sampling rates for the queries then (ii)execute the load shedding operation in the most efficient manner possible. This technique suffers from the same issue as the sampling approach; the unknown size of the data set remains a dead end.

Sliding windows focus mainly on recent data. Babcock & al.[3] define a random sample on a window of size n over the stream within which only k items are stored in memory. The authors achieve at best $O(k\ log(n))$ memory usage for their *priority-sample* algorithm developed for timestamp-based windows. The main drawback of this method, even though semantically easy to understand, is giving poor results on queries implying *older* data items.

Other researches have focused on providing methods coping with queries related to frequency moments F_n(see [2] for a more complete definition). The idea is to compute/approximate F_0 the number of distinct values in the sequence, F_1 the length of the sequence, F_2 the self-join size (or Gini's index of homogeneity) or F_∞ the most frequent item's multiplicity. Manku and Motwani proposed in [12] a probabilistic *Sticky Sampling Algorithm* for answering iceberg-like queries[11] (frequency counts exceeding user-defined thresholds). The algorithm computes ϵ-deficient synopses over data streams of singleton items. It was designed to sweep over the data stream and update concise samples[6] which are (item, count) pairs. Cormode and Muthukrishnan[5] proposed on their side an algorithm, based on a divide and conquer process using a dyadic range sum oracle, capable of providing the k *hottest* items for dynamic datasets.

A classic paradigm is representing synopses with histograms. The idea is to capture the data distribution at best with the objective of minimizing the errors while reconstructing the data values. The main approaches include V-Optimal, Equi-Width and End-biased histograms. V-Optimal histograms approximate the distribution of a set a data values $u_1, ..., u_n$ with a function $\hat{u}$ that minimizes the sum of squared error $\Sigma_i(u_i - \hat{u}(i))^2$ (or L^2). One of the most efficient sketching algorithms was proposed by Gilbert & al.[7] where the authors achieved to bound the sketch constructing time and footprint by $poly(B, log(N), log||A||, 1/\epsilon)$ where A is a vector(*buffer* in the case of data streams) of length N and B the number of *buckets* in the histogram. The main drawback of this sketching approach comes from the metric used (L^2) which is not adapted for representing isolated data values.

Equi-Width histograms as reminded by Poosala & al.[14] are meant to partition the data value space into β approximately equal *buckets* by generating quantiles or splitters. Greenwald and Khanna[8] presented a deterministic algorithm capable of computing quantiles in a single-pass and achieving $O(\frac{1}{\epsilon} log(\epsilon N))$

space while guaranteeing ϵN precision. The main weakness of such a method comes from atypical individual data that may be clustered in a same *bucket*.

Manku and Motwani[12] presented randomized and deterministic algorithms for computing *iceberg* queries[11] by using End-Biased histograms. Iceberg queries are computed aggregate functions over an attribute or a set of attributes to find aggregate values above user-defined threshold. The proposed algorithm achieve $O(\frac{1}{\epsilon}\ log(\epsilon N))$ space, where N is the length of the stream and guarantee that any element is undercounted by at most ϵN and that the reported count is not worse than ϵN from the actual count.

The last family of sketching techniques uses Haar wavelets as a decomposition function. An error tree compound of the wavelet coefficients is then computed [10]. Considering 8 data items, 8 wavelet coefficients are computed. These coefficients are enough to rebuild the original information. Given a chosen number of coefficients, let's say α, and an error measure m (L^2, maximum absolute error, maximum relative error, etc), research has been focused on selecting the α-best coefficients of the error tree that optimize the error measure m. In the latest work known, Karras and Mamoulis[9] propose one-pass algorithms adapted to the data stream environment (with an incremental construction of the error tree from a data stream) that minimises the maximum-error metrics. They achieve in terms of space complexity at best $O(N)$ and in terms of time complexity their algorithms converge to $O(N\ log^2(N))$ for the maximum absolute error metric and to $O(N\ log^3(N))$ for the maximum relative error metric.

In a Synthetic word. We have presented here a brief overview of existing techniques that answer the problem as we have formulated. Table 1 synthesizes, to our best, these techniques by positioning them according to Gibbons' criteria introduced earlier.

Approach	Coverage	Non–Coverage	Answer Quality	Footprint	Comp. & Update time	Observations
Sampling	Any request	Performs poorly on joins	Approximative & no guarantees	Potentially important	Rapid	Errors due to data distribution (high standard deviation)
Load Shedding	Any request	Performs poorly on joins	Approximative & no guarantees	Potentially important	Rapid	Errors due to data distribution (high standard deviation)
Sliding Windows	Requests on recent data	Requests on historical data	Good on recent data & Poorly on historical data	O(k Log N)	Rapid	
V–Optimal H.	Estimations in query optimizers, Frequent moments	The others	1/e but not adapted for isolated data (L^2 metric used)	Poly(B,Log N,1/e)	Poly(B,Log N,1/e)	
Equi–Width H.	Estimations in query optimizers, Frequent moments	The others	Not worse than eN but relatively poor for data distribution with high standard deviation	O(1/e Log eN)		
End–Biased H.	Aggregation & Iceberg queries	The others	Not worse than eN	O(1/e Log eN)		
Wavelets	Aggregation queries	The others	User defined	Converges to O(N)	Converges to $O(N\ Log^2 N)$ or $O(N\ Log^3 N)$	

Fig. 1. Summary of existing approaches

The most important remark that can be done concerns the domain of application of the different approaches. Most of them focus on providing data structures and algorithms capable of rebuilding (approximately) singleton data values and ensuring maximum control over the error rate.

Our Contribution. We propose to adapt a highly efficient on-line service-oriented summarization service capable of constructing a linguistic summary of very large tables and/or views called SaintEtiQ [1]. SaintEtiQ's summarization system takes a database in any form (tables, views, streams) as input and produces a reduced version of this input through both a rewriting and a generalization process. The resulting table provides tuples with less precision than the original but yet are very informative of the actual content of the database.

Definition 1 (SaintEtiQ Summary). *Let be a first normal form relation* $R(A_1, \ldots, A_n)$ *in the relational database model. The* SaintEtiQ *summarization system constructs a new relation* $R^*(A_1, \ldots, A_n)$*, in which tuples* z *are summaries and attribute values are linguistic labels describing a set of tuples* R_z*, sub-table of* R*. Thus, the* SaintEtiQ *system identifies statements of the form "*Q *tuples of* R *are* $(a_1^1$ *or* $a_1^2 \ldots$ *or* $a_1^{m_1})$ *and* $\ldots$ *and* $(a_2^1 \ldots$ *or* $a_n^{m_n})$*".*

SaintEtiQ was originally designed for summarizing very large databases. It was thus designed as a service-oriented (as a plugable webservice) scalable system with a linear time complexity ($O(n)$, with n the number of tuples to summarize) for the construction and maintenance of the summaries. SaintEtiQ computes and incrementally maintains a hierarchically arranged set of summaries, from the root (the most generic summary) to the leaves (the most specific ones). The reader is invited to refer to [15] for more thorough details. Therefore, we can sum up our contribution in three points.

The first one would be proposing a general framework adaptable to the data stream environment. We ensure a linear time complexity algorithm coupled with a user-definable space usage. Furthermore, the precision of our approach is driven by space usage. Incidentally, we comply with the criteria we have selected from Gibbon's definition. Therefore, we give here a truly interesting compromise by fixing the time complexity factor to an acceptable level and leaving the answer quality and space complexity to the user's need.

The second point is the proposal of a framework capable of dealing with multidimensional data in a user-understandable fashion, which is, to our knowledge, one of its unique features when compared to the existing techniques available for the data stream environment.

Finally but not the least, we propose a novel way of using the synopses built upon the content of the data streams in a datamining-fashioned way through the very informative summaries obtained via SaintEtiQ.

In the following section of this paper, we are going to present the SaintEtiQ process and will position it with more details according to Gibbons' criteria. Finally, we will introduce a novel vision on how data streams can be exploited in their overall form through SaintEtiQ summaries.

3 A SaintEtiQ Approach for Data Streams

3.1 The SaintEtiQ Model

The traditional SaintEtiQ summarization model takes database records as input and gives some kind of knowledge as output. The process is divided into two main steps: the first one consists in rewriting the input *raw* data and the second one is the learning or summarization step.

However, in order to cope with the data stream environment, it is necessary to add a preliminary step: *the data stream bufferization*. Therefore, the SaintEtiQ summarization model for data streams (SEQS MODS) becomes a 3 step process as shown in figure 2.

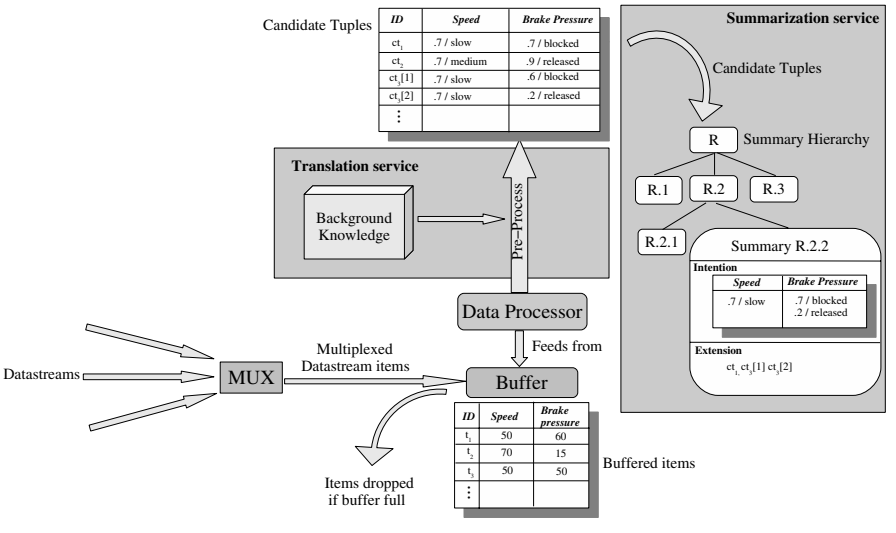

Fig. 2. SEQS MODS 3 step process for brake feeds

The primary stage is focused on receiving the (possibly) multiple streams. Thus, at the input of the system we introduce a multiplexer black-box. Its only purpose is to multiplex the input data into a single stream that will feed a buffer of user-defined size. At this stage of the process, there are many possible scenarii for feeding the buffer. The naive approach is to feed it until it is full then discard the items that do not fit until some space it released. More advanced techniques including *intelligent* sampling or load shedding (this then refers to related work) can be envisaged here. This step is generically illustrated by algorithm 1.

Once the buffer has input data, the *Data Processor*'s job is to retrieve single (or batches of) items and send those to the translation service for the rewriting task.

The rewriting step allows the system to rewrite the input stream items in order to be processed by the mining algorithm. This translation step gives birth to candidate tuples, which are different representations of a single stream item,

Algorithm 1. Data stream bufferization process

while (Data stream has input items) or (process not terminated by user) **do**
  Receive data from data stream s into $sdata$
  Apply policy for inserting $sdata$ into $buffer$
  %% It can be load shedding, sampling or simply first-come, first-served %%
end while

in accordance with a Background Knowledge(BK). BK is made up of fuzzy partitions defined over attribute domains. Each class of a partition is also labelled with a linguistic descriptor which is provided by the user or a domain expert. For example, the fuzzy label "Slow" could belong to a partition built over the domain of the attribute "Speed".

Thus, once a *raw* stream item is rewritten, for reducing processing time, the candidate tuple with the highest membership scores is selected among those generated. It is then considered by a machine learning algorithm and is first classified into the existing summary hierarchy to find the best matching summary following a top-down approach. At each node, the hierarchy is modified to incorporate this new instance through operations that can create or delete hierarchical child nodes. Decisions are taken based on local optimization criteria called the Partition Quality (PQ) which tries to minimize the length of the intentional description of the summary. This intentional description is made of a fuzzy set of domain descriptors on each attribute associated with a possibility measure.

In a data stream environment, due to the inherent limitations in terms of data rate and volume, a strategy needs to be held in order to be able to answer aggregation queries on the streams. Thus, in each node of the hierarchy, tables or histograms recording the contribution of each candidate tuple to each fuzzy linguistic label may be maintained and updated. As an example, a summary node compound of the attributes `(Speed, Brake Pressure, Temperature)` may be associated with table 1.

Table 1. Attribute contribution table

Too fast	Released	Cool
5	4	6

The result is a set of summaries which each describe a subset of the data stream items coupled with quantitative information recorded in histograms. These summaries are hierarchically organized so that the root summary describes the overall data stream, and the leaf summaries describe a very precise subset of the stream. This part of the process is presented in the generic algorithm 2.

The scalability issues are inherent to the data stream paradigm; thus, SaintEtiQ is intrinsicly able to cope with the memory consumption and the time complexity factors in order to handle massive datasets. SaintEtiQ process time

Algorithm 2. SEQS MODS process

while Input data streams have items **do**
  Read data stream input from *buffer* into *sdata*
  Rewrite *sdata* in the translation service
  Retrieve cooked data into *cookedSData*
  Classify *cookedSData* in summary hierarchy through the summarization service
end while

complexity was designed to be linear ($0(n)$ with n the number of candidate tuples). In terms of space cost, it is obvious that the more precise the BK, the more voluminous the summaries will be. Let us denote S the average size in kilo-bytes of a summary or cooked data. In the average-case assumption, there are $\sum_{k=0}^{d} B^k = (B^{d+1} - 1)/(B - 1)$ nodes in a B-arity tree with d, the average depth of the hierarchy. Thus, the average space requirement is given by: $c_m = S \cdot \frac{B^{d+1}-1}{B-1}$. Based on real tests, $S = 3kB$ gives a rough estimation of the space required for each summary. However, the system was built in a way that it is possible to maintain only parts of the hierarchy in main memory and store the less accessed ones on disk.

Having said this, we are now able to position SaintEtiQ according to Gibbons' criteria.

Coverage. When constructing the output hierarchy it is currently possible to maintain a table/histogram of the contribution of each data stream item to each fuzzy linguistic label. This quantitative information is the key for answering aggregate, frequent moment or iceberg queries and potentially any other type of request, thus allowing the system to position itself on the same tracks as the previously presented techniques. However, if we consider joins, our systems is restricted to providing size estimations of the different attributes involved in the join, as it does not keep references to the stream items.

Answer Quality. As said earlier, the precision of the summaries evolve with the precision of the BK provided. The more accurate the BK, the more precise the summaries and as a result the more voluminous they are. It then becomes obvious that the quality of the answers provided depends mostly on the compromise on the quality of the input BK.

Let's take again the example of join queries. On the one hand, in the extreme case where the BK is very roughly defined, a large number of tuples may be selected for the cartesian product operation. Therefore, the selectivity power of the summary is very poor as many tuples will not participate in the cartesian product. On the other hand, if the data distribution can be estimated accurately on certain attributes, the tuples selected for the join operation are probabilistically better chosen, which means that most selected tuples will participate in the operation; thus the selectivity power of the summary, and the underlying histograms, may be dramatically increased.

This example pinpoints how important the definition of the BK influences the quality of the answers provided by the SaintEtiQ summaries. Therefore,

an interesting axis of work may be oriented towards dynamically tuning the BK according to the distribution of the dataset, to the evaluation of the current performances and to the user's needs for a more data-oriented approach.

Footprint. As the computation and update time of a summary were designed to be linear, the footprint of the summaries is a major factor on which compromises must be done. Detailed BKs induce high memory consumption.

However, each node of the hierarchy was designed as an autonomous agent that can be serialized as a small binary stream. A *cache manager* can discard or resurrect *older* summaries to/from the disk providing disk accesses are acceptable for the system. Therefore, the footprint of the summaries is also a tunable factor according to the user's needs in terms of memory capabilities/savings.

Computation & Update time. The SaintEtiQ process was designed to be linear, $O(n)$, where n is the number of candidate tuples.

3.2 Novel Field of Application for Data Stream Synopses

The summaries resulting from processing multidimensional input data through SaintEtiQ convey semantically richer information than any other *traditional* data stream synopsis. This unique feature allows us to open the path to advanced applications such as identifying the topology of a group within the data stream. Thus, SaintEtiQ synopses are potentially geared towards a data mining-oriented usage. It seems interesting to associate decision making actions with (candidate tuple) pattern matching rules.

Let us take a different and maybe more challenging toy example, this time in the financial analysis environment. Suppose a broker needs to advice a customer for the investment into company A(CA). What he'll like to do is to sell both stocks of CA and other stocks to his customer. However, the latter would only buy if there are relative *guarantees* on the profitability of the transaction. Thus, imagine the following scenario: Company B (CB) wants to perform a takeover bid on CA - but CA is against. Therefore, in order to delay and/or avoid this operation, CA starts a takeover on Company C(CC). Incoming feeds would be in the form `(actor, target, price, number of stocks)`.

When processing these input stream *items*, as CA invests massively into CC, many candidate tuples in the form `(actor, target, cheap, plenty)` may be generated. An agent specially designed could then detect this tendency of the market. Speculatively speaking, stocks of CC are then susceptible to become more valuable in the short term. Thus, when considering this example, SEQS MODS summaries become very handful. The broker, within the first transactions, may be able to identify the tendency, which is those massive acquisitions of CC by CA at a *cheap* price (relatively to the probable future short-term value). He can then advice his customer to quickly invest with *high confidence* into CC stocks. Identifying these kind of topology is obviously a crucial parameter for his decision making job.

This toy example highlights the many possibilities offered in terms of application by SaintEtiQ summaries. They provide a rich background for advanced techniques, including mining or profiling, on data streams.

4 Perspectives and Future Work

Up till now, we be have based our talk on the experience and results that we have acquired in a traditionally stored database background. What we can at least guarantee with our current experiments on the current prototype is our ability to manage data streams with data rates at least equal to the system's current input capacity on stored databases. For the time being, our work is focused on optimizing the prototype so as to squeeze out the time consuming tasks entirely related to the stored database domain. Many optimization works, especially the implementation of the heuristics(see [15]), the cache management or the distribution of the process are currently on track. Once these are integrated into the system, we believe the summarization task will be dramatically sped up.

The second axis for our future work is to determine in the resulting summary hierarchy whether it is more interesting to keep the whole hierarchy structure or prune of it. Actually, as said earlier, the root of the hierarchy is the most general synopsis and the leaves are the most specific ones. It may not be interesting to conserve the root nor all of the leaves. Our attention is then directed towards finding a compromise between what needs to be kept and what doesn't. Obviously, this has an important impact on the summary maintaining process and the hence the performance of the overall architecture. This domain needs more thorough investigation.

Finally, we expect to take advantage of all the experience acquired here to adapt and optimize existing distributed architectures such as the WS-CatalogNet [13] or support applications as *"Advance Resource Reservation"*[16].

5 Conclusion

In this paper, we presented the integration of an on-line linguistic summarization process into a data stream environment. This is facilitated by the intrinsic feature of the SaintEtiQ system. By the system's advantages such as being an on-line service, as its scalability, as its reduced and controlled footprint and linear time complexity, SaintEtiQ is by essence ready for dealing with high data rate and voluminous data streams.

The summaries obtained through the process are furthermore semantically rich enough to open a new field of application on data streams. We are now able, through the same process, to summarize and mine the streams. It is now possible in a Decision Making paradigm to exploit the resulting summaries to explore and find profiles for example either in financial analysis or network surveillance and administration.

Immediate work is focused on optimizing and distributing the process for better performances in terms of stream processing.

At a medium term, we will investigate the way SaintEtiQ summaries may be pruned in order to improve the summarization process.

At last, further developments include the use of summaries to help query E-Community E-Catalogs or use their structure for data source selectivity which are domains potentially exposed to the data stream model. It would be, for instance, promising to implement such a service on an application level on mobile routers(MR) as described in [16] in order to help the MRs to optimize their network traffic.

References

1. SaintEtiQ. `http://www.simulation.fr/seq`.
2. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. of ACM Symposium on Principles of Database Systems 2002*.
3. B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. *2002 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002)*, 2002.
4. B. Babcock, M. Datar, and R. Motwani. Load shedding techniques for data stream systems. *20th International Conference on Data Engineering ICDE 2004*, 2004.
5. G. Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Proc. of ACM Principles of Database Systems (PODS 2003)*, pages 296–306, 2003.
6. P. B. Gibbons and Y. Matias. Synopsis data structures for massive data sets. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1998.
7. A. C. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Fast, small-space algorithms for approximate histogram maintenance. *STOC'O2*, 2002.
8. M. Greenwald and S. Khanna. Space-efficient online computation of quantile summaries. *ACM SIGMOD 2001*, 2001.
9. P. Karras and N. Mamoulis. One-pass wavelet synopses for maximum-error metrics. In *Proc. of 31st VLDB Conference 2005*, 2005.
10. Y. Matias, J. S. Vitter, and M. Wang. Wavelet-based histograms for selectivity estimation. In *Proc. of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD 1998)*, pages 448–459, June 1998.
11. R. Motwani, M. Fang, N. Shivakumar, H. Garcia-Molina, and J. D. Ullman. Computing iceberg queries efficiently. In *Proc. of the 24th VLDB Conference 1998*, pages 299–310, 1998.
12. R. Motwani and G. S. Manku. Approximate frequency counts over data streams. In *Proc. of the 28th VLDB Conference 2002*, 2002.
13. H.-Y. Paik, B. Benatallah, K. Baïna, F. Toumani, C. Rey, A. Rutkowska, and B. Harianto. Ws-catalognet: An infrastructure for creating, peering, and querying e-catalog communities. In *Proc. of the 30th VLDB Conference 2004*, 2004.
14. V. Poosala, Y. E. Ioannisdis, P. J. Haas, and E. J. Shekita. Improved histograms for selectivity estimation of range predicates. *22nd VLDB Conference 1996*, 1996.
15. R. Saint-Paul, G. Raschia, and N. Mouaddib. General purpose dataset summarization. *31st VLDB Conference 2005*, 2005.
16. A. Sun, M. Hassan, M. B. I. Hassan, P. Pham, and B. Benatallah. Fast and scalable access to advance resource reservation data in future mobile networks. *IEEE ICC*, 2006.