Towards Flexible Self-Tuning Data Stream Management Systems

Wieger R. Punter
Supervised by dr. O. Papapetrou
Eindhoven University of Technology
Eindhoven, The Netherlands
w.r.punter@tue.nl

ABSTRACT

The volume and velocity of streaming data are growing exponentially, driven by advancements in sensor technologies and the increasing presence of connected devices. This growth increases the need for efficient data stream management systems. Over the past decades, significant efforts have been made to develop methods for efficient streaming data analytics. While scaling the network resources horizontally and vertically is one approach, another is to scale down the data using synopses. Synopses are a class of approximate data structures for stream summarization, which can be used for aggregate queries. Examples of synopses include samples (e.g., with uniform or reservoir sampling), and sketches (e.g., Count-min and Bloom filters). During my PhD I will work on making an efficient self-tuning data stream management system based on synopses. These self-tuning systems should decrease the time-consuming and complex process of selecting a set of synopses to maintain. Towards this end I will: (a) study the state-of-the-art to identify key gaps in the support for query types by synopses (b) fill major gaps by designing new synopses and (c) develop a policy to continuously tune the optimal set of synopses. So far, my contributions address two novel requirements, by supporting frequency queries with arbitrary many predicates (OmniSketch) and supporting spatial aggregate queries on arbitrary shapes in a grid (SpatialSketch). This workshop will be an excellent opportunity to discuss both the horizon of the proposed system and my planned contributions in detail.

VLDB Workshop Reference Format:
Wieger R. Punter. Towards Flexible Self-Tuning Data Stream Management Systems. VLDB 2024 Workshop: VLDB Ph.D. Workshop.

1 INTRODUCTION

The volume and velocity of streaming data is experiencing unprecedented growth, driven by advancements in sensor technologies and the increasing presence of connected devices. This growth goes hand in hand with the need to process and analyze this streaming data efficiently in real-time. In the past decades, there has been a great effort in developing methods to allow efficient analytics on streaming data. A popular approach is to scale the amount of resources in a network, horizontally and/or vertically. Complementary to this, it is also possible to scale down the data, by using synopses [5, 9].

Synopses are small-space approximate data structures for stream summarization that can subsequently be used to execute aggregate queries. A controllable amount of error is introduced to allow space- and time-efficient analytics. Examples of synopses are Reservoir Sampling [18], Distinct Sampling [8] and sketching techniques as the Bloom Filter [3], Count-min [6] and HyperLogLog [7].

Traditional synopses are typically constructed for answering a single query type. For example, Bloom filters can answer membership queries, whereas Count-min queries can answer frequency queries. Therefore, use of synopses requires prior knowledge about the (expected) future workload on the data stream. In the absence of this knowledge (or limited knowledge), one needs to construct and maintain many different synopses (e.g., as many as can fit in the available RAM or can be processed at line speed) to support a wide range of queries. Furthermore, the allocation of resources to the synopses is also not trivial, as there is an inherent trade-off between accuracy and efficiency for each synopsis. In general, the more resources, e.g. storage, processing time, we allocate to the synopses, the more accurate the query results.

For the practitioner, selecting the set of synopses to maintain is therefore a time-consuming and challenging process. The learning curve of mastering all synopses is steep, and even when you have mastered it, balancing all trade-offs and selecting the set of synopses takes time.

To address the challenge of selecting the set of synopses to maintain, lately we have witnessed the development of many general-use synopses, such as the UnivMon [13] or Elastic sketch [19]. These synopses can be used for answering many query types (e.g., estimating many frequency-moments and heavy hitter queries) with one synopsis. The benefit of these synopses is that the user can select one synopsis for multiple purposes and spend all resources on one structure. However, these general synopses are still not general enough (i.e., many query types require their own synopses and many others cannot even be answered with any synopsis).

There is therefore a need for designing general-use synopses that can be used for answering many query types, as well as data stream management systems that can automatically choose the set of synopses to maintain. My PhD work will focus on addressing these gaps by (1) identifying key gaps in the support for query types by synopses, (2) filling the defined gaps for missing query types and (3) designing selection policies for the set of synopses, to allow self-tuning systems. The results will also be useful outside of data stream management systems, for single applications.

My first two works have contributed to this by significantly reducing the amount of synopses needed to support the following

Proceedings of the VLDB Endowment. ISSN 2150-8097.

query types: frequency queries with multiple predicates and spatial aggregations on arbitrary shapes. For both, the complexity of the problem lies in that it should be possible to specify the predicates and the arbitrary shapes at query time.

Roadmap. The remainder of this document is structured as follows. In Section 2, a brief background on mentioned synopses is provided. Section 3 discusses my current contributions in more detail. The envisioned system is presented in Section 4. Finally, my research plan is presented in Section 5.

2 RELATED WORK

This section covers a selection of synopses that are mentioned in the rest of this document. It also highlights sketches that answer multiple query types with one sketch. A thorough survey of synopses can be found in [5].

Synopses are small-space approximate data structures for stream summarization that can subsequently be used to execute aggregate queries. They can be categorized in Histograms, Wavelets, Samples and Sketches. We focus on the latter two. In general, samples are data structures that often support more than one query type, while sketches are more specific [5]. Sketches perform better than samples on some query types as distinct count and join size estimation, and are also more robust against deletes in the stream [5]. We will now briefly discuss some widely-used sampling and sketching methods and their purpose.

Reservoir sampling [18] is a uniform sampling method for streams. It is constant in memory-usage and can be used for multiple queries, such as selectivity. Distinct sampling [8] can be used to estimate the number of distinct elements. Min-wise hashing [2, 16] is a method to estimate the cardinality of the intersection of multiple sets, based on a summary that is computed independently for each set.

The Count-min sketch [6] summarizes the distribution of data streams and support frequency and inner product queries. Bloom Filters [3] support membership queries. The HyperLogLog sketch [7] supports distinct count. Misra-Gries answers heavy hitter queries. Recently, sketches were developed to answer multiple query types with one synopses. Examples are UnivMon [13], elastic sketch [19] and Panakos [20]. These sketches usually focus on estimating multiple frequency moments and heavy hitter queries.

3 CURRENT RESULTS

I started my PhD in December 2022. My focus up to now was on understanding the key functionality of synopses, and designing synopses for addressing specific requirements. This section covers my contributions so far. The OmniSketch is published in VLDB 2024 [17]. I also contributed to the SpatialSketch [12], which is under submission.

3.1 Omnisketch

My first work, OmniSketch [17], can efficiently answer frequency queries with an arbitrary number of predicates.

```
SELECT COUNT (*) FROM stream
WHERE attr1 = x AND attr2 > y AND ... attrA < z
```

At query time, the user can specify the subset of attributes in the stream they wish to filter, and the specific predicate values.

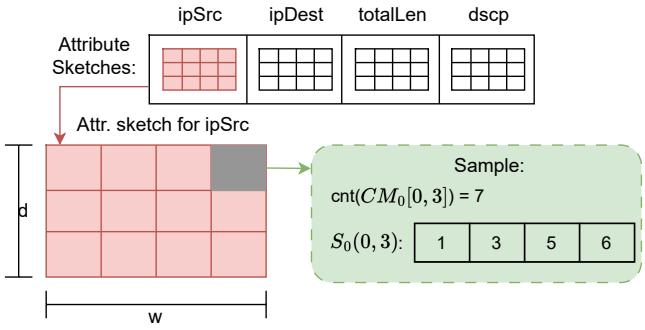

Figure 1: OmniSketch Data Structure

Existing methods, [4, 6, 19], either require knowing the subsets of attributes in advance and building a synopsis for each combination (inflexible), or maintaining a synopsis for all possible subsets, which scales exponentially with the number of attributes in the stream (inefficient). Hydra, a sketch developed by Manousis et al. [14] is a single synopses that can answer this query type, even though it is not specifically designed for it. However, it only shifts the exponential factor from the number of synopses to the insertion time and accuracy. When dealing with high-velocity streams, this remains too inefficient. Therefore, we developed the OmniSketch, for which the space and insertion time requirements scales linearly with the amount of attributes in the stream, instead of exponentially. This comes at the cost of a small but manageable increase in the query time.

Data Structure. The data structure of the sketch is shown in Figure 1 for an example stream of form $(rid, ipSrc, ipDest, totalLen, dscp)$. We maintain an attribute sketch per attribute that is similar to the Count-min sketch. Every attribute sketch contains d rows and w columns. In each cell of the sketch, the min-wise sample method described by Pagh et al. in 2014 [16] is maintained. The records are inserted to each of the $|A|$ attribute sketches. As example, at the attribute sketch for $ipSrc$, d hash functions of form $h_j(r_{ipSrc}) -> [1 \dots w]$ are executed to find the corresponding cells at each row j. At each corresponding cell, the rid is inserted into the min-wise sample. At query time, the corresponding cells are found by hashing the predicate values at the relevant attribute sketches. The intersection size of all samples is computed and used to scale the answer result. When the parameters are set correctly, the error is guaranteed to be lower than ϵN, with high probability. N denotes the size of the stream up until querying. The mathematical proof is described in detail in the paper and the technical report.

The empirical results of the sketch show OmniSketch outperforms the state-of-the-art (in our experiments, by more than 2 orders of magnitude in throughput) while still providing highly accurate estimates. This method only works for the insertion-only streaming model. In the future, we plan to extend this work by supporting deletes in the streaming model as well.

3.2 SpatialSketch

In the second project I contributed to, synopses for summarizing spatial data streams [12], we considered the challenge of computing

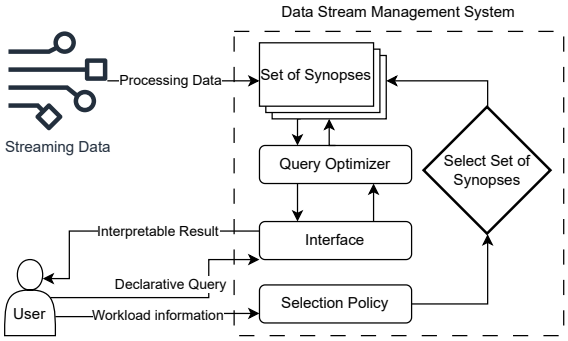

Figure 2: Design of Data Stream Management System

aggregate statistics for spatial ranges over data streams, where the spatial ranges is specified at query time. We introduced two novel sketches, SpatialSketch and DynSketch, which can support different types of aggregates (e.g., frequency estimation, L2 norm, membership queries) by incorporating the functionality of other nested sketches like Count-min sketches and Bloom filters.

Data Structure. The data structure of SpatialSketch is based on dyadic ranges [10]. The sketch has a layered structure of grids. At the base layer, the grid has the maximum spatial resolution, specified by the user. At the top layer, there is a grid with one cell, covering the entire spatial region. Each cell in the grids contains a nested sketch, e.g. the Count-min. At insertion time, a record is inserted at each layer to the nested sketch at the cell based on the spatial location of the record. At query time, a partition algorithm partitions the shape of arbitrary form first into rectangles, and then into dyadic intervals inside those rectangles. The corresponding intervals are queried and the result is reduced with the appropriate function, based on the nested sketch. It also allows merging of the nested sketches at the queried intervals.

DynSketch is a dynamic version of the SpatialSketch that dynamically drops certain grids and layers when dealing with memory constraints. Importantly, the two sketches are backed by a formal analysis that provides accuracy guarantees for diverse aggregates, and for different classes of nested sketches. Through an extensive experimental evaluation with both real and synthetic data streams, we demonstrated that the proposed sketches outperform the competitors (both exact and approximate) in terms of functionality, efficiency, and accuracy, within identical memory constraints.

4 VISION

In this section, I present an envisioned system to enable the widespread use of synopses. This vision places the contributions during my PhD in the right context, as my work aims to make this system possible. The system extends beyond the scope of my PhD.

The envisioned system is inspired by Chapter 6 in the survey by Cormode, Garofalakis & Haas [5] and the system described by Ioannidis [11]. The survey Chapter describes future work directions, such as the implementation of synopses in data stream management systems, and open problems and issues that constitute roadblocks for the wide acceptance of approximate query processing. The system described by Ioannidis discusses problems for approximations in database systems, not streaming systems. Even though there is a focus on a different data model than streaming data, the problems remain relevant, as they concern approximations. The motivation for my system is to be able to answer user queries in an approximate fashion, without users having to undergo a steep learning curve.

Figure 2 presents my envisioned system, with the following workflow:

(1) **Data:** The user connects static and/or streaming data sources to the system.
(2) **Workload information:** The user specifies any information they have on the type of future workload. This information can change after initialization.
(3) **Selecting Set of Synopses:** The system maintains a set of synopses to support the specified workload, but allowing flexibility to execute other workloads as well. The size of the set of synopses is limited by either the user or the system. As workload information can change, the system continuously optimizes the set of synopses and the resource allocation within.
(4) **Processing Data:** The data is inserted in the corresponding (sub)set of synopses.
(5) **Querying Synopses:** The user can execute approximate queries on the data, by using a declarative language, possibly CQL [1]. The system will optimize the running time and accuracy of the query plan based on available synopses and static tables.
(6) **Interpretable Results:** The user is aided in interpreting the approximate results by the system.

This workflow imposes many research challenges, such as (relation to workflow in parentheses):

(2 & 3) How should the workload information be asked from the user?
(2,3&5) How to handle missing/incorrect workload information?
(3) How to select the optimal set of synopses based on available information?
(3) How to allocate resources within the set?
(3) How to dynamically maintain the set if the available information about the future query needs changes, without needing significant redesigns?
(6) How should results be presented to aid the user?

5 RESEARCH PLAN

As my PhD is time-constrained, I will contribute to the needs defined in the introduction: designing general-use synopses that can be used for answering many query types and developing a policy to select the optimal set of synopses given the available information. My research plan contains three steps: (1) identifying key gaps in the support for query types by synopses, (2) filling the defined gaps for missing query types and (3) designing selection policies for the set of synopses.

Identifying the key missing functionalities. There is a need for an overview of the state-of-the-art on synopses. With the overview,

we can identify the state of the art for each query type, the relationship between different synopses, as well as unsupported query types. The most complete survey to date is the one by Cormode, Garofalakis and Haas [5]. However, this is from 2012 and in the last few years there were many additional key results.

Filling the gap for missing query types. The identified missing functionalities will be used to focus on the right problems and to design synopses for the missing query types. This process can also entail generalizing a synopsis to support multiple query types. So far, gaps are defined for (a) dealing with frequency queries with arbitrary many predicates in the scenario of turnstile streams and (b) supporting query types, other than frequency, with arbitrary many predicates. We will now discuss both.

Currently, when dealing with frequency queries with arbitrary many predicates, there is only support for insertion-only streams, by using the OmniSketch. Deletes are a problem for the min-wise sampling method used in each cell of the attribute sketches. It is easy to delete an element from a sample, but when this is done, another element from the past stream should have been inside the sample. When inserting an element from the future stream, the sampling properties break. It is not known which of the two elements should have been in the sample: the past or the future element. In this project, we aim to solve this by either improving the current sampling method or using a more robust method.

Besides frequency queries in the case of arbitrary many predicates, there are many other query types. We aim to support query types for other frequency moments, such as count distinct or L2-Norm. An example query would be:

```
SELECT COUNT DISTINCT (*) FROM stream
WHERE attr1 = x AND attr2 > y AND ... attrA < z
```

Or estimating the size of queries with inner joins and predicates:

```
SELECT COUNT(*) FROM s1, s2, s3
INNER JOIN s2 ON s1.s2id = s2.id
INNER JOIN s2 ON s1.s3id = s3.id
WHERE s1.attr2 = "x" AND s3.attr6 = "z"
```

New research is needed for extending OmniSketch to support these types. Possible directions can be found in the recent developments on sketches supporting more than one frequency norm [13, 19, 20].

Selection policy for set of synopses. When gaps on query types are filled, it is time to work towards the described system. An essential component will be the policy for continuously selecting the set of synopses and allocating resources. This policy must consider user-provided information on the query types that need support and the desired flexibility for other types of queries. The research challenge lies in structuring this information and choosing the synopses. A starting point are techniques as Taster, a self-tuning, elastic and online approximate query processing engine designed by Olma et al. in 2019 [15].

ACKNOWLEDGMENTS

This work was partially funded by the European Commission under the STELAR (HORIZON-EUROPE - Grant No. 101070122) project.

REFERENCES

[1] Arvind Arasu, Shivnath Babu, and Jennifer Widom. 2006. The CQL continuous query language: semantic foundations and query execution. *VLDB J.* 15, 2 (2006), 121–142. https://doi.org/10.1007/S00778-004-0147-Z

[2] Andrei Z. Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of SEQUENCES 1997, Positano, Amalfitan Coast, Salerno, Italy, June 11-13, 1997, Proceedings*. IEEE, 21–29. https://doi.org/10.1109/SEQUEN.1997.666900

[3] Andrei Z. Broder and Michael Mitzenmacher. 2003. Survey: Network Applications of Bloom Filters: A Survey. *Internet Math.* 1, 4 (2003), 485–509. https://doi.org/10.1080/15427951.2004.10129096

[4] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2004. Finding frequent items in data streams. *Theor. Comput. Sci.* 312, 1 (2004), 3–15. https://doi.org/10.1016/S0304-3975(03)00400-6

[5] Graham Cormode, Minos N. Garofalakis, Peter J. Haas, and Chris Jermaine. 2012. Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches. *Found. Trends Databases* 4, 1-3 (2012), 1–294. https://doi.org/10.1561/1900000004

[6] Graham Cormode and S. Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms* 55, 1 (2005), 58–75. https://doi.org/10.1016/J.JALGOR.2003.12.001

[7] Philippe Flajolet, Eric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. HyperLogLog: The analysis of a near-optimal cardinality estimation algorithm. *Discrete Mathematics & Theoretical Computer Science* (03 2007), 137–156. https://doi.org/10.46298/dmtcs.3545

[8] Phillip B. Gibbons. 2001. Distinct Sampling for Highly-Accurate Answers to Distinct Values Queries and Event Reports. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*. Morgan Kaufmann, 541–550. http://www.vldb.org/conf/2001/P541.pdf

[9] Phillip B. Gibbons and Yossi Matias. 1999. Synopsis Data Structures for Massive Data Sets. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland, USA*. ACM/SIAM, 909–910. http://dl.acm.org/citation.cfm?id=314500.315083

[10] Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. 2002. How to Summarize the Universe: Dynamic Maintenance of Quantiles. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*. Morgan Kaufmann, 454–465. https://doi.org/10.1016/B978-155860869-6/50047-0

[11] Yannis E. Ioannidis. 2003. Approximations in Database Systems. In *Database Theory - ICDT 2003, 9th International Conference, Siena, Italy, January 8-10, 2003, Proceedings (Lecture Notes in Computer Science)*, Vol. 2572. Springer, 16–30. https://doi.org/10.1007/3-540-36285-1_2

[12] Jacco Kiezebrink, Wieger R. Punter, Odysseas Papapetrou, and Kevin Verbeek. 2023. *Synopses for Aggregating Arbitrary Regions in Spatial Data*. Technical Report. TU Eindhoven. https://github.com/JaccoKiezebrink/SpatialSketch

[13] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. 2016. One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon. In *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016*. ACM, 101–114. https://doi.org/10.1145/2934872.2934906

[14] Antonis Manousis, Zhuo Cheng, Ran Ben Basat, Zaoxing Liu, and Vyas Sekar. 2022. Enabling Efficient and General Subpopulation Analytics in Multidimensional Data Streams. *Proc. VLDB Endow.* 15, 11 (2022), 3249–3262. https://doi.org/10.14778/3551793.3551867

[15] Matthaios Olma, Odysseas Papapetrou, Raja Appuswamy, and Anastasia Ailamaki. 2019. Taster: Self-Tuning, Elastic and Online Approximate Query Processing. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 482–493. https://doi.org/10.1109/ICDE.2019.00050

[16] Rasmus Pagh, Morten Stöckel, and David P. Woodruff. 2014. Is min-wise hashing optimal for summarizing set intersection?. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*. ACM, 109–120. https://doi.org/10.1145/2594538.2594554

[17] Wieger R. Punter, Odysseas Papapetrou, and Minos N. Garofalakis. 2023. OmniSketch: Efficient Multi-Dimensional High-Velocity Stream Analytics with Arbitrary Predicates. *Proc. VLDB Endow.* 17, 3 (2023), 319–331. https://doi.org/10.14778/3632093.3632098

[18] Jeffrey Scott Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Softw.* 11, 1 (1985), 37–57. https://doi.org/10.1145/3147.3165

[19] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. 2018. Elastic sketch: adaptive and fast network-wide measurements. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*. ACM, 561–575. https://doi.org/10.1145/3230543.3230544

[20] Fuheng Zhao, Punnal Ismail Khan, Divyakant Agrawal, Amr El Abbadi, Arpit Gupta, and Zaoxing Liu. 2023. Panakos: Chasing the Tails for Multidimensional Data Streams. *Proc. VLDB Endow.* 16, 6 (2023), 1291–1304. https://doi.org/10.14778/3583140.3583147