Data Series Management: The Road to Big Sequence Analytics

Themis Palpanas
Paris Descartes University
themis@mi.parisdescartes.fr

ABSTRACT

Massive data series collections are becoming a reality for virtually every scientific and social domain, and have to be processed and analyzed, in order to extract useful knowledge. Current data series management solutions are ad hoc, requiring huge investments in time and effort, and duplication of effort across different teams. Systems like relational databases, Column Stores, and Array Databases are not a suitable solution either, since none of these systems offers native support for data series. Our vision is to design and develop a general-purpose Data Series Management System, able to cope with big data series, that is, very large and fast-changing collections of data series, which can be heterogeneous (i.e., originate from disparate domains and thus exhibit very different characteristics), and which can have uncertainty in their values (e.g., due to inherent errors in the measurements). Just like databases abstracted the relational data management problem and offered a black box solution that is now omnipresent, the proposed system will allow analysts that are not experts in data series management, as well as common users, to tap in the goldmine of the massive and ever-growing data series collections they (already) have.

1. INTRODUCTION

[**Motivation.**] Data series have gathered the attention of the data management community for almost two decades [7, 17, 25]. Data series are one of the most common types of data, and are present in virtually every scientific and social domain: they appear as audio sequences [13], shape and image data [29], financial [24], environmental monitoring [21] and scientific data [11], and they have many diverse applications, such as in health care, astronomy, biology, economics, and others.

Recent advances in sensing, networking, data processing and storage technologies have significantly eased the process of generating and collecting tremendous amounts of data series at extremely high rates and volumes. It is not unusual for applications to involve numbers of sequences in the order of hundreds of millions to billions [1, 3].

[**Data Series.**] A *data series*, or *data sequence*, is an ordered sequence of data points[1] (if the dimension that imposes the ordering of the sequence is time then we talk about time series). Formally, a data series $T = (p_1, ...p_n)$ is defined as a sequence of points $p_i = (v_i, t_i)$, where each point is associated with a value v_i and a time t_i in which this recording was made, and n is the size (or length) of the series.

A key observation is that analysts need to process and analyze a sequence (or subsequence) of values as a single object, rather than the individual points independently, which is what makes the management and analysis of data sequences a hard problem. Note that even though a sequence can be regarded as a point in n-dimensional space, traditional multi-dimensional approaches fail in this case, mainly due to the combination of the following two reasons: (a) the dimensionality is typically very high, i.e., in the order of several hundreds to several thousands, and (b) dimensions are strictly ordered (imposed by the sequence itself) and neighboring values are correlated. Therefore, specialized techniques need to be used, as we discuss in Section 2.

[**Data Series Management Systems.**] There are important reasons why data Series (or Sequence) Management Systems (SMSs) are on the cusp of becoming a focal point for research activity in data management. The solutions that are currently available require custom code and the development of ad hoc systems for various tasks, requiring huge investments in time and effort, and duplication of effort across different teams.

Consider for instance, that for several of their analysis tasks, neuroscientists are currently reducing each of their 3,000 point long sequences to a single number (the global average) in order to be

[1]For the rest of this paper, we are going to use the terms *data series* and *sequence* interchangeably.

able to analyze their huge datasets [1]. Similarly, the increasing number of adults that monitor their health (e.g., a heart rate monitor is producing 9GB of sequence data per month), cannot use the data they produce, which could otherwise enable them to monitor their lifestyles, and (with the assistance of their doctor) identify changes in their health, or the consequences of different dietary or medicinal choices. In astronomy, there are currently available more than 70TB of spectroscopic sequence data from 200 million sky objects, collected by the Sloan Digital Sky Survey [3], allowing scientists to study the universe. These data have to be processed and analyzed, in order to identify patterns, gain insights, detect abnormalities, and extract useful knowledge.

In all the above cases, existing approaches (e.g., based on DBMSs [6], Column Stores [26], or Array Databases [27]) do not provide a viable solution, since they have not been designed for managing and processing sequence data. Therefore, they do not offer a suitable declarative query language, storage model, auxiliary data structures (such as indexes), and optimization mechanism that can support a variety of sequence query workloads (see also Section 3.1) in an efficient manner.

We argue that a SMS is necessary in order to enable big sequence analytics, since it will offer the abstractions, tools, and automations needed for achieving this goal. Just like databases abstracted the relational data management problem and offered a black box solution that is now omnipresent, the proposed system will make it feasible for analysts that are not experts in data series management, as well as common users, to tap in the goldmine of the massive and ever-growing data series collections they (already) have.

[**Challenges.**] Several works have studied the problems of sequence management, processing and analysis, focusing on some specific problems in these areas. The results of such studies offer an excellent starting point for our project, but do not constitute a viable solution to our problem.

There is a rich literature on sequence summarization techniques, but the corresponding indexing techniques are rather rigid (i.e., can only answer fixed-length queries), do not deliver in terms of performance what is required for the applications we envision, and most importantly, do not allow for cost-based optimization. We also note the need for a completely new generation of sequence query answering techniques that can efficiently operate on massive data sequence collections without the extremely expensive preprocessing step that current indexes impose. When considering uncertain sequences, we observe that current representation techniques are simplistic (with corresponding side-effects in their effectiveness), and new models and relevant indexing methods are in need.

Designing the right abstractions and data models for a SMS is also challenging. Data sequences could eventually be accommodated in existing systems (such as relational databases, Column Stores, and Array Databases), albeit in an unnatural way, since none of these systems treats sequences as first class citizens. These systems do not offer native support for sequences (e.g., data model, index structures, etc.), thus, any solution built on top of them will suffer in terms of expressive power, usability, and performance.

[**Contributions.**] Our ambitious goal is to design and develop a general-purpose data Series Management System. The contributions we envision can be summarized as follows.

- The system will be able to cope with big data sequences, that is, massive collections of sequences, which can be heterogeneous (i.e., originate from disparate domains and thus exhibit very different characteristics), and which can have uncertainty in their values (e.g., due to inherent errors in the measurements).
- It will efficiently support a wide range of sequence queries and mining operations, by developing ground-breaking techniques for scalable sequence management and analysis, while exploiting the benefits of physical and logical independence.
- A key feature of our solution will be cost-based optimization, which will enable the system to automatically pick the right storage and execution strategies, leading to remarkable improvements in time performance and scalability that would otherwise be untenable.
- In addition, we propose to make advances by leveraging ideas from adaptive systems and distributed computing, two areas that have not been studied in the past in the particular context of data sequences.

[**Organization of this paper.**] We review the related work and existing techniques and approaches in Section 2. In Section 3, we describe the main components of the proposed system, and outline the open research directions. Finally, we conclude in Section 4.

2. STATE OF THE ART

The main objective of the proposed research is to build a general purpose SMS for big sequences. Such a system should include techniques to support

sequences with very different characteristics (originating from disparate domains), as well as uncertain sequences, answer sequence queries fast even for massive sequence collections, and rely on cost-based optimization for query execution.

[Data Series Summarizations.] In order to efficiently process and analyze large volumes of data sequences, we have to operate on summaries (or approximations) of these sequences. Several techniques have been proposed in the literature for the summarization of sequences (e.g., DFT, DCT, DWT, SVD, APCA, PAA, SAX), and it is interesting to note that averaged over many datasets, there is little difference between all the above approaches in terms of accuracy of representation [18] (even though it *is* the case that certain representations favor particular data types, e.g., DFT for star-light-curves, APCA for bursty data, etc.).

[Data Series Indexing.] These summarizations are the basis for the index structures that make possible the efficient execution of sequence queries over very large collections of data sequences. In general, data series indexes operate by pruning the search space based on the summarizations of the data series and corresponding lower bounds, and only use the raw data of the data series in order to filter out the false positives.

Agrawal et al. [4] presented the first data series index, based on the DFT summarization. Various indexes, specific to data series, have also been proposed in the literature [25, 28]. A general observation when using sequence indexes though, is that query answering can degenerate to perform worse that a serial scan. To make things worse, these situations cannot be predicted for the current generation of sequence indexes.

[Relevant Systems.] We note that current relational DBMSs [6], Column Stores [26], and Array Databases [27] could eventually be used to store and process sequences. Nevertheless, they cannot efficiently support complex data mining queries, (that is, queries that treat the entire sequence as a single object, such as sequence similarity queries, clustering, classification, etc.), which require fast distance computations among the sequences in the collection, since they do not natively support any mechanisms for pruning the search space. Consequently, these systems cannot offer optimization functionality for the execution of DM queries, which is a key requirement for a SMS.

3. DATA SERIES MANAGEMENT

In this section, we provide an overview of the necessary components of a SMS, and we discuss the envisioned functionality and open research problems for each one of them.

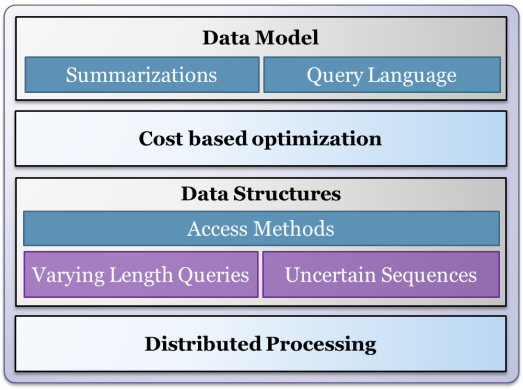

Figure 1: The architecture of a data series management system.

A key element of a SMS is the design of a cost-based optimizer for the execution of sequence queries, with a special focus on complex data mining queries. There is currently no optimizer available for sequence queries, even though it is a necessary component for efficient and scalable processing and analytics. As we discuss next, traditional approaches fail in our setting, and therefore, major breakthroughs are needed in this direction.

The optimizer should depend on and be closely related to the storage and indexing solutions for sequences, two research areas that should also be addressed. The design of the data model should accommodate various sequence summarization techniques, including novel techniques for uncertain sequences, and innovative access methods (i.e., storage and indexing) that will be able to adapt to the user needs (i.e., the query workload). Moreover, particular attention should be paid to optimizations specific to data sequence techniques relevant to modern hardware and distributed environments.

Last, but not least, there is a need for the development of a benchmark specifically designed for data sequences, which as we discuss below is a challenging task in itself. This benchmark will serve to evaluate the state of the art and identify promising ideas and concepts, as well as guide the efforts in the development of new techniques.

In Figure 1, we illustrate the general architecture of a SMS. We elaborate on the individual components of the system in the following sections. We discuss optimization last, since it touches on the rest of the components, and also include a discussion on the need for a data sequence benchmark.

3.1 Data Series Queries

There are various types of data sequence queries that analysts need to perform: (a) simple Selection-Projection-Transformation (SPT) queries, and (b) more complex Data-Mining (DM) queries. Simple SPT queries are those that select sequences and project points based on thresholds, point positions, or specific sequence properties (e.g., above, first 10 points, peaks), or queries that transform sequences using mathematical formulas (e.g., average). An example SPT query could be one that returns the first x points of all the sequences that have at least y points above a threshold. The majority of these queries could be handled (albeit not optimally) by current database management systems, which nevertheless, lack a domain specific query language that would support and facilitate such processing.

DM queries on the other hand are more complex by nature: the processing has to take into consideration the entire sequence, and treat as a single object, therefore being much more complex to process. Examples under this category are: queries by content (range and similarity queries, nearest neighbors), clustering, classification, outlier patterns, frequent sub-sequences, and others. These queries cannot be supported by current data management systems, since they require specialized data structures, algorithms and storage methods in order to be performed efficiently.

Note that the data series datasets and queries may refer to either static, or streaming data. In the case of streaming data series, we are interested in the sub-sequences defined by a sliding window. The same is also true for static data series of very large size (e.g., an electroencephalogram, or a genome sequence), which we divide into sub-sequences using a sliding (or shifting window). The length of these sub-sequences is chosen so that it can contain the patterns of interest.

3.2 Data Model

As we discussed earlier, neither the relational model nor the array model can adequately capture the characteristics of sequences. In the case of relational data, there are various options available for translating sequences into relations and each one of them has significant limitations. On the other hand, in Array Databases we lack the expressive power to define collections of sequences, and are restricted to defining large multi-dimensional matrices that encode both sequence and meta-data on an equal basis, which hinders efficiency.

An ideal sequence model should instead be able to effectively describe collections of sequences and allow us to do operations on them. It should allow us for example to select sequences based on meta-data or based on their values, project them as complete sequences or sub-sequences and join them in a variety of ways for computing calculations. At the same time such a model should intuitively allow for both intra-sequence and inter-sequence aggregations, and be compatible with different sequence summarization methods. Finally, the corresponding query language could be based on previous works [15, 22], suitably extended to deal with data series as single objects, as well as with DM queries.

3.3 Data Structures

A large collection of access methods has been proposed in the literature, able to evaluate different queries under various settings, including both indexes and scan-based methods. A recently proposed representation technique, SAX, has several desired properties, including good approximation quality, small memory footprint, and fast bitwise operations [25]. However, the indexing structure proposed for SAX in certain cases exhibits degenerate performance (i.e., worse than a serial scan), which is not desirable in practice. Initial work in this area is encouraging [7,8], with iSAX2+ demonstrating scalability to dataset sizes 2-3 orders of magnitude more than the current state of the art.

[Adaptive Indexes.] Nevertheless, new techniques are necessary in order to further improve scalability, and more importantly, to significantly boost time performance, in order to match the requirements of modern applications. Our experience with massive sequence collections [8] shows that indexing itself can become a bottleneck in a data sequence analysis task: for example, it takes more than 24 hours to build a state-of-the-art index over a dataset of 1 billion sequences in a modern server machine. It is imperative therefore, to develop techniques that interactively and adaptively build parts of the index, focusing on the data necessary to answer the queries, and making the data available for querying (almost) immediately. Preliminary results in this area are very promising, demonstrating a 7-fold improvement in the time to prepare an index on 1 billion data series and answer 100,000 *approximate* queries [31].

Other promising directions should also be explored, such as methods that rely on fast scans of the data [14, 19]. These directions can provide viable alternatives to the indexes discussed above, and in several situations can be the access method of choice. This is especially true given the data management trend on large-scale parallelization, the usage of compres-

sion, multi-cores, SIMD architectures and the exploitation of available GPUs [20].

We also propose to extend these techniques along two orthogonal dimensions: supporting queries of varying length, and uncertain sequences.

[Varying Length Queries.] Existing techniques only consider collections of data series with the same length, leading to indexes that can answer queries of a fixed (predefined) length. As a result, new access methods that also consider varying length queries have to be developed. Contrary to previous approaches [12], we argue that the information already captured by certain data sequence indexes can be exploited, and is possible to develop new varying-length query answering techniques on top of this.

[Uncertain Sequences.] In several cases, data sequences can be uncertain, that is, the raw data have an inherent uncertainty in their values (e.g., because of errors introduced by the measurement devices), and integrate the solutions to the proposed system. There exist promising studies on modeling and analyzing uncertain sequences [5, 23, 30], but more work is needed in order to improve the quality and time performance [9]. A promising direction in this respect is the modeling of uncertain sequences with possible world semantics based on full-joint distributions, which can retain the correlation information among neighboring points [10]. Nevertheless, there are still important scalability issues to be overcome in order for such techniques to be used with large sequence collections.

3.4 Distributed Processing

During the last years there has been a lot of research on MapReduce systems, where various methods have been proposed to support the indexing of large multidimensional data [16], where an index is distributed among several compute nodes. Nevertheless, up to this point work on sequential data query processing using MapReduce has mainly concentrated on efficiently performing parallel scans of the complete dataset, while all indexing-related studies only consider read-only operations. Even though various approaches have been proposed for speeding up iterative algorithms, none of the proposed models is a suitable match for the algorithms and techniques we need, where timely communications among workers play a crucial role in reducing the amount of total work done. Therefore, there is need for more work in this area, taking into consideration new paradigms as well [2].

3.5 Cost based optimization

As we discussed above, there can be multiple different execution strategies for answering the same query, including the various choices of serial scans, indexes, and processing methods (e.g., parallelization, GPU, etc.). The challenge in choosing the right execution strategy is to estimate the amount of data that such a query will need to access before executing it. For example, a fast parallel SIMD-enabled scan on compressed data might be a better option than the use of a non-optimized index when SIMD instructions are available, but not a better choice when such instructions are not available. All these characteristics have to be exploited by the cost-based optimization models, and considered in a way that is transparent to the user. This problem becomes even more challenging when complex queries involving several operators need to be executed (e.g., consider an analysis task that combines a series of SPT operators as a pre-processing step, and then applies a DM operator).

While in traditional relational databases there are simple and efficient ways in order to estimate query selectivity [6], this is not the case for sequence similarity queries that lie in the heart of most sequence mining algorithms. The challenges in this context arise from the combination of the very high dimensional and sequential nature (i.e., the inherent correlations among neighboring values) of these data.

Up to this point, no efficient methods have been proposed to solve this problem, and ground-breaking work needs to be done. We believe that a promising direction is to carefully study the hardness of a query: being able to control the effort needed to answer a query can be the right step stone for solving the inverse problem, that of estimating the effort it will take to answer a query, before executing it.

3.6 Data Series Benchmarking

Despite the rich literature on methods for indexing and answering similarity queries on data sequences, we note the absence of any related benchmarks. We argue for the need of fair benchmarks that can stress-test sequence processing techniques in a controlled way and to pre-defined levels of query hardness. Such benchmarks will be designed to capture differences in the quality of summarization methods, indexes and storage methods, when working in *combination*, which is what makes the design of such a benchmark a challenging task. Our ongoing work constitutes the first solution towards this directions: it hows that the amount of effort employed by data series indexes can be consistently captured across different indexing approaches, using implementation-invariant measures [32].

4. CONCLUSIONS

Even though data series are a very common data type, there is currently no system that can inherently accommodate, manage, and support complex analytics for this type of data. In this paper, we argue for the special nature of the sequences data type, and articulate the necessity for rigorous work on data series management systems. We propose a SMS that will employ a data model specialized to sequences. The system will be distributed by design, and consider the large volume of sequences, their heterogeneity (in terms of properties and characteristics), and possible uncertainty in their values. Finally, the system will support cost-based optimization, thus, leading to the desired scalability for big sequence analytics.

Acknowledgements

I would like to thank my collaborators, A. Camerra, M. Dallachiesa, J. Gehrke, S. Idreos, I. Ilyas, E. Keogh, M. Linardi, Y. Lou, K. Mirylenka, B. Nushi, and J. Shieh. Special thanks go to K. Zoumpatianos, who has been the driving force behind several of the ideas discussed in this paper.

References

[1] Adhd-200. `http://fcon_1000.projects.nitrc.org/indi/adhd200/`, 2011.

[2] Orleans: Distributed virtual actors for programmability and scalability. MSR-TR-2014-41, 2014.

[3] Sloan digital sky survey. `https://www.sdss3.org/dr10/data_access/volume.php`, 2015.

[4] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, 1993.

[5] J. Aßfalg, H. Kriegel, P. Kröger, and M. Renz. Probabilistic similarity search for uncertain time series. In *SSDBM*, 2009.

[6] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. Gray, P. P. Griffiths, W. F. K. III, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. System R: relational approach to database management. *TODS*, 1(2):97–137, 1976.

[7] A. Camerra, T. Palpanas, J. Shieh, and E. Keogh. iSAX 2.0: Indexing and mining one billion time series. In *ICDM*, 2010.

[8] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. J. Keogh. Beyond one billion time series: indexing and mining very large time series collections with isax2+. *KAIS*, 39(1):123–151, 2014.

[9] M. Dallachiesa, B. Nushi, K. Mirylenka, and T. Palpanas. Uncertain time-series similarity: Return to the basics. *PVLDB*, 5(11):1662–1673, 2012.

[10] M. Dallachiesa, T. Palpanas, and I. F. Ilyas. Top-k nearest neighbor search in uncertain data series. *PVLDB*, 8(1):13–24, 2014.

[11] P. Huijse, P. A. Estévez, P. Protopapas, J. C. Principe, and P. Zegers. Computational intelligence challenges and applications on large-scale astronomical time series databases. *IEEE Comp. Int. Mag.*, 9(3):27–39, 2014.

[12] S. Kadiyala and N. Shiri. A compact multi-resolution index for variable length queries in time series databases. *KAIS*, 15(2):131–147, 2008.

[13] K. Kashino, G. Smith, and H. Murase. Time-series active search for quick retrieval of audio and video. In *ICASSP*, 1999.

[14] S. Kashyap and P. Karras. Scalable knn search on vertically stored time series. In *KDD*, 2011.

[15] A. Lerner and D. Shasha. Aquery: Query language for ordered data, optimization techniques, and experiments. In *VLDB*, 2003.

[16] H. Liao, J. Han, and J. Fang. Multi-dimensional index on hadoop distributed file system. In *NAS*, 2010.

[17] J. Lin, R. Khade, and Y. Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *J. Intell. Inf. Syst.*, 39(2), 2012.

[18] T. Palpanas, M. Vlachos, E. J. Keogh, and D. Gunopulos. Streaming time series summarization using user-defined amnesic functions. *TKDE*, 20(7), 2008.

[19] T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. E. A. P. A. Batista, M. B. Westover, Q. Zhu, J. Zakaria, and E. J. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *KDD*, 2012.

[20] V. Raman, G. K. Attaluri, R. Barber, N. Chainani, D. Kalmuk, V. KulandaiSamy, J. Leenstra, S. Lightstone, S. Liu, G. M. Lohman, T. Malkemus, R. Müller, I. Pandis, B. Schiefer, D. Sharpe, R. Sidle, A. J. Storm, and L. Zhang. DB2 with BLU acceleration: So much more than just a column store. *PVLDB*, 6(11):1080–1091, 2013.

[21] U. Raza, A. Camerra, A. L. Murphy, T. Palpanas, and G. P. Picco. Practical data prediction for real-world wireless sensor networks. *IEEE Trans. Knowl. Data Eng.*, accepted for publication, 2015.

[22] R. Sadri, C. Zaniolo, A. M. Zarkesh, and J. Adibi. A sequential pattern query language for supporting instant data mining for e-services. In *VLDB*, 2001.

[23] S. R. Sarangi and K. Murthy. DUST: a generalized notion of similarity between uncertain time series. In *KDD*, 2010.

[24] D. Shasha. Tuning time series queries in finance: Case studies and recommendations. *IEEE Data Eng. Bull.*, 22(2):40–46, 1999.

[25] J. Shieh and E. Keogh. iSAX: Indexing and Mining Terabyte Sized Time Series. KDD, 2008.

[26] M. Stonebraker, D. J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S. Madden, E. J. O'Neil, P. E. O'Neil, A. Rasin, N. Tran, and S. B. Zdonik. C-store: A column-oriented DBMS. In *VLDB*, 2005.

[27] M. Stonebraker, P. Brown, A. Poliakov, and S. Raman. The architecture of scidb. In *SSDBM*, 2011.

[28] Y. Wang, P. Wang, J. Pei, W. Wang, and S. Huang. A data-adaptive and dynamic segmentation index for whole matching on time series. *PVLDB*, 6(10):793–804, 2013.

[29] L. Ye and E. J. Keogh. Time series shapelets: a new primitive for data mining. In *KDD*, 2009.

[30] M. Yeh, K. Wu, P. S. Yu, and M. Chen. PROUD: a probabilistic approach to processing similarity queries over uncertain data streams. In *EDBT*, 2009.

[31] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. In *SIGMOD*, 2014.

[32] K. Zoumpatianos, Y. Lou, T. Palpanas, and J. Gehrke. Query workloads for data series indexes. In *KDD*, 2015.