Data Series Management: The Next Challenge (Keynote Talk)

Themis Palpanas
Paris Descartes University
Paris, France
Email: themis@mi.parisdescartes.fr

***Abstract*—There is an increasingly pressing need, by several applications in diverse domains, for developing techniques able to index and mine very large collections of sequences, or data series. Examples of such applications come from biology, astronomy, entomology, the web, and other domains. It is not unusual for these applications to involve numbers of data series in the order of hundreds of millions to billions, which are often times not analyzed in their full detail due to their sheer size. In this study, we describe recent efforts in designing techniques for indexing and mining truly massive collections of data series that will enable scientists to easily analyze their data. We argue that the main bottleneck in mining such massive datasets is the time taken to build the index. Therefore, we discuss solutions to this problem, including novel techniques that adaptively create data series indexes, allowing users to correctly answer queries before the indexing task is finished. Finally, we present our vision for the future of the very promising data series management research.**

I. Introduction

[Motivation.] Data series have gathered the attention of the data management community for almost two decades [40], [10], [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 [18], shape and image data [44], financial [39], telecommunications [32], [27], environmental monitoring [36] and scientific data [16], [1], 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], [2].

[Data Series.] A *data series*, or *data sequence*, is an ordered sequence of data points[1]. 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. If the dimension that imposes the ordering of the sequence is time then we talk about *time series*, though, a series can also be defined over other measures (e.g., angle in radial profiles in astronomy, mass in mass spectroscopy, position in genome sequences, etc.).

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 length (or 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.

[Need for Data Series Indexing.] There are two main types of data series 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). DM queries on the other hand, treat an entire sequence as a single object, and are therefore 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.

In this context, the nearest neighbor operation is of paramount importance, since it forms the basis of virtually every DM query. However, nearest neighbor queries across a large collection of data series are challenging, because data series collections grow very large in practice [11], [35], and existing data management solutions [7], [41], [42] cannot efficiently support them. Thus, methods for answering nearest neighbor queries rely on two main techniques: data summarization and indexing. Data series summarization is used to reduce the dimensionality of the data series [20], [34], [22], [4], [19], [12], [24], and then indexes are built on top of these summarizations [34], [40], [5], [38], [43].

[Outlook.] We argue that a general-purpose data series management system is necessary in order to enable big sequence analytics, since it will offer the abstractions, tools, and automations needed for achieving this goal. It should be able to efficiently support a wide range of sequence queries and mining operations at a scalable fashion, while exploiting the benefits of physical and logical independence, and it should support cost-based optimization, which will enable the system to automatically pick the right storage and execution strategies

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

for answering different queries. 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.

II. The Current State of Affairs

In this section, we briefly describe and comment on the some of the most prominent efforts in the areas of managing and indexing data series collections[2].

A. Using Existing Data Management Systems

Even existing approaches based on DBMSs [7], Column Stores [41], or Array Databases [42] do not provide a viable solution, since they have not been designed for managing and processing sequence data, and do not treat sequences as first class citizens. Note that 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.

These systems 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 in an efficient manner. Therefore, any solution built on top of them will suffer in terms of expressive power, usability, and performance.

B. Scaling Up

Even though recent studies have shown that in certain cases sequential scans can be performed very efficiently [35], such techniques do not bring benefit to the general case of querying a mixed database of several data series (rather than a single, long series). Therefore, indexing is required in order to efficiently support data exploration tasks, which involve ad-hoc queries, i.e., the query workload is not known in advance. A large set of indexing methods have been proposed for the different data series summarization methods, including traditional multidimensional [15], [34], [8], [21] and specialized [40], [5], [19], [38], [43] indexes.

Indexing can significantly reduce the time to answer DM queries. Yet, as the data series collections grow in size, the operation of indexing these collections can itself become the bottleneck in the entire process [10], [11], [46]. As an answer to this problem, iSAX2.0 [10] and iSAX2+ [11] were proposed, which are the first data series indexes that inherently support bulk loading, and thus aimed to minimize the index building time. More recently, the ADS+ index [46], [47] was developed, which is the first data series index than can start answering queries correctly before the entire index has been built, by adaptively building and growing only the parts of the index that are needed for answering the queries. These techniques considerably shrink the data-to-query gap, allowing users to start answering queries much faster than any previous approach. Nevertheless, many interesting problems are still open. For example, how we can efficiently support exact queries with ADS+, how we can reduce the large variance in exact query answering times, and how the iSAX2+ and ADS+ indexes are best parallelized.

[2]We do not discuss here problems related to data mining and analysis. Nevertheless, we argue that in most cases, the correct data management techniques can lead to significant time efficiency benefits for the mining and analysis algorithms.

C. Scaling Out

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 [23], [26], 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 [3].

Gorilla is a recent effort on building a distributed, in-memory sequence database, coupled with a long-term storage solution using HBase [33]. This system has several desirable properties: dedicated storage manager, compression support, efficient update mechanism, high availability, and very good scalability characteristics. The focus of the system though, is on answering simple SPT queries, and extending it to cover DM queries as well, is an open problem.

III. The Next Challenge

There are important reasons why work on data series management systems constitutes an exciting and promising research direction: (a) the techniques and tools that are available are rather fragmented, each one addressing only specific and narrow needs; and (b) 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. Existing approaches for managing and analyzing very large data series collections fall short of the expectations and requirements of modern applications in this domain.

As a result, the few expert analysts need to invest heavily in the development of customized tools for processing their datasets in order to identify patterns, gain insights, detect abnormalities, and extract useful knowledge, while the many analysts that are not experts are simply not able to process their data in their full detail (for example, neuroscientists are summarizing functional magnetic resonance imaging sequences of length 3000 with asingle value, i.e., the global mean, because they cannot process the data in their full detail [1]).

A. A Sequence Management System (SMS)

In the following, we present our vision for a general-purpose Sequence Management System (SMS), along with the corresponding challenges. Figure 1 illustrates the general architecture of a SMS.

First, a sequence data model should be developed, able to effectively describe collections of sequences, and to allow

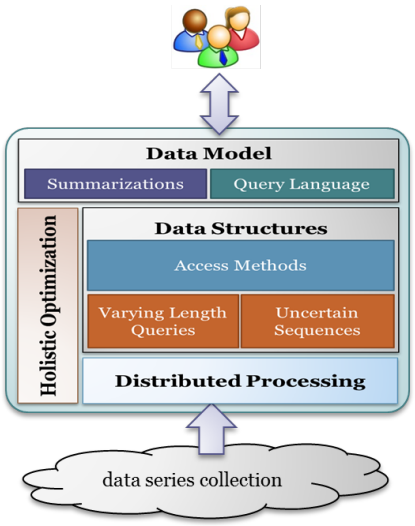

Fig. 1. The architecture of a data series management system.

us to do operations on them (e.g., 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). 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.

In terms of scale out, 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. The critical point here is that timely communications among workers play a crucial role in reducing the amount of total work done. Therefore, there is need for more research in this area, possible taking into consideration new paradigms as well [9].

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. 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 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 [7], 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 in this direction.

We also propose to extend these techniques along two orthogonal dimensions: supporting queries of varying length, and uncertain sequences. 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 [17], 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. It is also true that in several cases the values of data sequences are uncertain (e.g., because of errors introduced by the measurement devices). There exist promising studies on modeling and analyzing uncertain sequences [6], [45], [37], but more work is needed in order to improve the quality and time performance [13]. 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 [14]. Nevertheless, there are still important scalability issues to be overcome in order for such techniques to be used with large sequence collections.

Finally, 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. A recent work takes the first step in this direction: it shows that the amount of effort employed by data series indexes can be consistently captured across different indexing approaches, using implementation-invariant measures [48].

IV. Conclusions

In this study, we focused on the problem of data series management[3]. We discussed the state-of-the-art data series indexing approaches that can cope with the data deluge, including the first to support bulk loading, iSAX 2.0 and iSAX2+, and the first adaptive indexing approach, ADS+.

Furthermore, we observed that 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. Therefore, we articulate the necessity for rigorous work on data series management systems, which involves several challenging and exciting research directions.

Acknowledgements
I would like to thank my collaborators (in alphabetical order):

[3] A more detailed analysis of the topics discussed in this paper can be found in our previous works [31], [21], [30], [10], [13], [11], [46], [14], [48], [47], [28], [29].

Alessandro Camerra, Michele Dallachiesa, Johannes Gehrke, Stratos Idreos, Eamonn Keogh, Michele Linardi, and Yin Lou. Special thanks go to Kostas 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] Sloan digital sky survey. https://www.sdss3.org/dr10/data_access/volume.php, 2015.

[3] T. G. Addair, D. A. Dodge, W. R. Walter, and S. D. Ruppert. Large-scale seismic signal analysis with hadoop. *Computers & Geosciences*, 66:145–154, 2014.

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

[5] I. Assent, R. Krieger, F. Afschari, and T. Seidl. The ts-tree: Efficient time series search and retrieval. In *EDBT*, 2008.

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

[7] 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.

[8] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-tree: An index structure for high-dimensional data. In *VLDB*, pages 28–39, 1996.

[9] P. Bernstein, S. Bykov, A. Geller, G. Kliot, and J. Thelin. Orleans: Distributed virtual actors for programmability and scalability. MSR-TR-2014-41, 2014.

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

[11] 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.

[12] K.-P. Chan and A.-C. Fu. Efficient time series matching by wavelets. In *ICDE*, 1999.

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

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

[15] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, 1984.

[16] 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.

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

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

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

[20] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *KAIS*, 3(3):263–286, 2000.

[21] E. J. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle. Indexing large human-motion databases. In *VLDB*, pages 780–791, 2004.

[22] C.-S. Li, P. Yu, and V. Castelli. Hierarchyscan: a hierarchical similarity search algorithm for databases of long sequences. In *ICDE*, 1996.

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

[24] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD*, 2003.

[25] 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.

[26] P. Lu, G. Chen, B. C. Ooi, H. T. Vo, and S. Wu. Scalagist: Scalable generalized search trees for mapreduce systems [innovative systems paper]. *PVLDB*, 7(14):1797–1808, 2014.

[27] K. Mirylenka, V. Christophides, T. Palpanas, I. Pefkianakis, and M. May. Characterizing home device usage from wireless traffic time series. In *EDBT*, pages 551–562, 2016.

[28] T. Palpanas. Data series management: The road to big sequence analytics. *SIGMOD Rec.*, 44(2):47–52, 2015.

[29] T. Palpanas. Big sequence management: A glimpse of the past, the present, and the future. In *SOFSEM 2016: Theory and Practice of Computer Science - 42nd International Conference on Current Trends in Theory and Practice of Computer Science, Harrachov, Czech Republic, January 23-28, 2016, Proceedings*, pages 63–80, 2016.

[30] T. Palpanas, M. Vlachos, E. J. Keogh, and D. Gunopulos. Streaming time series summarization using user-defined amnesic functions. *IEEE Trans. Knowl. Data Eng.*, 20(7):992–1006, 2008.

[31] T. Palpanas, M. Vlachos, E. J. Keogh, D. Gunopulos, and W. Truppel. Online amnesic approximation of streaming time series. In *ICDE*, pages 339–349, 2004.

[32] P. Paraskevopoulos, T.-C. Dinh, Z. Dashdorj, T. Palpanas, and L. Serafini. Identification and characterization of human behavior patterns from mobile phone data. In *D4D Challenge session, NetMob*, 2013.

[33] T. Pelkonen, S. Franklin, P. Cavallaro, Q. Huang, J. Meza, J. Teller, and K. Veeraraghavan. Gorilla: A fast, scalable, in-memory time series database. *PVLDB*, 8(12):1816–1827, 2015.

[34] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In *SIGMOD*, 1997.

[35] T. Rakthanmanon, B. J. L. Campana, A. Mueen, G. 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.

[36] 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.

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

[38] P. Schäfer and M. Högqvist. Sfa: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In *EDBT*, 2012.

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

[40] J. Shieh and E. J. Keogh. *i*sax: indexing and mining terabyte sized time series. In *KDD*, pages 623–631, 2008.

[41] 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.

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

[43] 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.

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

[45] 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.

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

[47] K. Zoumpatianos, S. Idreos, and T. Palpanas. RINSE: interactive data series exploration with ADS+. *PVLDB*, 8(12):1912–1923, 2015.

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