ceur-ws.org/Vol-2399/paper10.pdf

Optimized Spatio-Temporal Data Structures for Hybrid Transactional and Analytical Workloads on Columnar In-Memory Databases

Keven Richly
supervised by Prof. Dr. h.c. Hasso Plattner
Hasso Plattner Institute, University of Potsdam
August-Bebel-Str. 88
Potsdam, Germany
keven.richly@hpi.de

ABSTRACT

Rapid advances in location-acquisition technologies have led to large amounts of trajectory data. This data is the foundation for a broad spectrum of services driven and improved by trajectory data mining. However, for hybrid transactional and analytical workloads, the storing and processing of rapidly accumulated trajectory data is a non-trivial task. In this paper, we propose an approach to optimize the trajectory data management capabilities for relational database systems. Based on the observations that the relational database structure is well-suited to store trajectory data in the sample point format and that the access patterns for trajectory data change over time, we develope a concept for optimized spatio-temporal data structures for in-memory column stores and describe a workload-aware tiered data compression approach. The first evaluations of the approach demonstrate that the observed data access patterns of different real-world use cases are supporting the proposed system architecture.

1. INTRODUCTION

In recent years, rapid advances in location-acquisition technologies have led to large amounts of time-stamped location data. Positioning technologies like GPS-based or communication network-based systems enable the tracking of various moving objects. This data is the foundation for a wide spectrum of applications [6, 10, 9]. A *trajectory* is represented by a series of chronologically ordered sampling points. Each sampling point contains spatial information, which is represented by a multidimensional coordinate in a geographical space, and temporal information, which is represented by a timestamp. Additionally, an object identifier assigns each sampling point to a specific moving object and the corresponding trajectory. Thereby, the duration and sampling rate depends on the application. Based on the characteristics of spatio-temporal trajectory data, there exist four key challenges: the data volume, the high update rate (data velocity), the query latency of analytical queries, and the inherent inaccuracy of the data. For these reasons, it is a nontrivial task to manage and store vast amounts of these data, which are rapidly accumulated. Especially, if we consider hybrid transactional and analytical workloads (so-called HTAP or mixed workloads) on spatio-temporal data, which are challenging concerning space and time complexity.

The most common format to store trajectory data is the sample point format. Here, each observed location is stored as a tuple with the following attributes: trajectory identifier, moving object identifier, multi-dimensional location, and timestamp. In trajectory query processing, we distinguish the four query types: (i) trajectory-based queries, (ii) spatio-temporal range queries, (iii) KNN queries, and (iv) top-k queries [7]. Trajectory-based queries refer to the trajectory of a single moving object and return the entire trajectory, a specific segment of the trajectory, or related information like the length of the trajectory. Relational databases are able to store data in the sample point format and process queries of the four mentioned types. For that reason, it could be promising to investigate this research area. Furthermore, relational data management systems have some further advantages that could be used for trajectory data (e.g., standardized query language, highly optimized data processing capabilities). To process spatio-temporal data efficiently, we have to develop optimized data structures for spatio-temporal data.

Due to the large amount of trajectory data, which have to be stored in various use cases, we also have to evaluate different compression mechanisms for trajectory data to reduce the data footprint. Particularly for in-memory databases, we have to utilize the available memory efficiently. Additionally, in different use cases we can observe that a specific trajectory segment is less accessed over time. Also, the query types and granularity of queries, which access a specific segment change over time. Similar characteristics can be observed in time-series data analysis. For that reason, it is necessary to analyze how we can adopt compression mechanisms for such kind of data access patterns to further reduce the memory consumption and increase the performance by minimizing the number of sample points.

2. RESEARCH ISSUES

In this section, we want to highlight the research aspects, that we derived from the observations mentioned in the previous section. We propose a concept to integrate storage

Proceedings of the VLDB 2019 PhD Workshop, August 26th, 2019. Los Angeles, California. Copyright (C) 2019 for this paper by its authors. Copying permitted for private and academic purposes.

CEUR-WS.org/Vol-2399/paper10.pdf

mechanisms for trajectory data into a relational in-memory database with particular attention to leverage the capabilities of the columnar database layout and the adoption to observed data access patterns. The objective is to reduce the data footprint and to better utilize the available memory bandwidth. We accordingly focus on data structures and compression techniques.

1. Optimized temporal and spatial data structures for columnar in-memory databases: The sample point format, which is used to store trajectory data by the majority of spatio-temporal data management systems [7], is well suited for the table schema of relational database systems. Also, different database systems integrated optimized spatial data types [4]. In different applications, we could observe that the performance of spatio-temporal range queries is strongly dominated by the temporal scan operation. For that reason, it is necessary to develop new concepts, which eliminate this bottleneck. The different types of time awareness that various use cases provide should be considered [7]. Additionally, we should evaluate the spatial functions of database systems with regard to trajectory data management.

2. Compression techniques for trajectory data: Zhang et al. [13] evaluated the compression ratio and quality of various trajectory simplification algorithms. Columnar in-memory databases and databases, in general, use additional data compression techniques apart form delta encoding (e.g., dictionary or run-length encoding). For that reason, it is necessary to evaluate the effect of these compression techniques in the context of columnar in-memory databases and analyze the potentials of compression mechanisms that combine traditional database compression approaches with specialized trajectory simplification algorithms.

3. Workload-aware tiered data compression: The structures of various trajectory data management systems have a simar blueprint (see Figure 2). In this blueprint, the trajectory data is stored in data partitions. In general, the systems do not distinguish the different data partitions and treat all equally. In various applications and especially in systems, which analysze times series, we could observe that the data access pattern for a database entry changed over time [1, 5]. In time series data management, an approach is to increase the compression ratio for old values [5]. Considering the massive amounts of trajectory data, which are tracked by various companies (e.g., transportation network companies), a tiered compression approach could reduce the data footprint reasonably.

3. RELATED WORK

As shown in Figure 1, there are several systems, which focus on the storage, indexing, and compression of trajectory data. Various systems were primarily designed to store spatial data points but they are also used for trajectory data. With the increased availability of spatio-temporal data management system were built for different system infrastructures, which leverage the specific characteristics of trajectory data. The optimization focus of the developed system strongly depends on the addressed use cases: (i) scalability, (ii) footprint reduction, (iii) elasticity, (iv) efficiency, or (v) query performance [7].

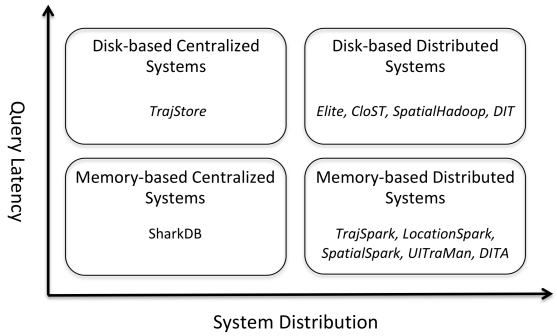

Figure 1: A classification model for trajectory data management systems

Although the various systems are designed for different infrastructures and optimized for different use cases, similarities in the general structure can be determined and summarized in a general blueprint. Most trajectory data management systems are optimized for spatio-temporal range queries or k-nearest neighbor queries. They use a hybrid partitioning approach to efficiently skip data partitions during the query execution. To archive spatio-temporal data locality the data is partitioned by the temporal dimension first and the spatial dimension second. The trajectory of a moving object is not stored together in one partition. The systems divide a trajectory into different trajectory segments and distribute them on the data partitions on the basis of the partitioning scheme. Inside a data partition, the observed locations are often stored in a sample point data format, whereby the sample points are ordered by the trajectory identifier to increase the performance of trajectory-based queries. To compress the data all systems use delta compression. Additionally, a wide range of systems applies further compression techniques for all data partitions. To optimize the query performance, the trajectory data management systems use a layered index structure. As displayed in Figure 2, the structure consists of a three-level hybrid index. The first layer is a temporal index structure, which divides the temporal dimensions into different time intervals. There are different implementations for this index (e.g., skip-list, range index). For each time interval in the first layer, there exists a spatial index. In most cases, the spatial index is represented by a traditional KD-tree, quad-tree or oct-tree. The last layer is normally a tree structure (e.g., b+ tree), which index the different spatio-temporal partitions.

There are also several systems, which already integrate data structures to store spatial data and spatio-temporal data in relational database systems [4, 12]. Integrating trajectory data management into relational database systems has the advantages that there is a standardized query language, the spatio-temporal data can be combined with other data (e.g., business data), and it is possible to leverage the highly optimized data processing capabilities of these systems.

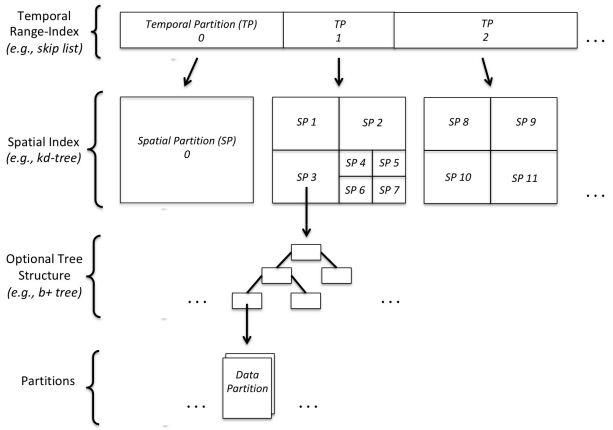

Figure 2: General blueprint of trajectory management systems

4. RESERACH PLAN

In this section, we describe the approach we propose to optimize the capabilities of relational databases to store and process trajectory data. Furthermore, we explain our planned evaluation of the concept.

4.1 Approach

The foundation of the proposed approach is the chunk concept implemented by the relational in-memory research database *Hyrise* [3]. As displayed in Figure 3, a database table is divided into chunks, which are temporally ordered horizontal partitions of a certain size. A chunk contains fragments of all columns of a database table, which are called segments. Besides efficient multiprocessing and chunk pruning, this structure enables the creation of auxiliary data structures (e.g., indices) on a per-chunk basis. Additionally, it provides the possibility to change the order criteria between different chunks (e.g., trajectory identifier, location) and enables the application of varying trajectory simplification algorithms. We can apply the same concept to the encodings of segments. For that reason, some segments of a column can be unencoded, others dictionary-encoded, and further segments can be encoded with additional compression techniques [2].

4.1.1 Optimized temporal and spatial data structures for columnar in-memory databases

These capabilities enable the workload-aware selection of compression techniques. Furthermore, it is possible to adopt the ordering of data tuples per chunk. For example, if we assume a workload that consists of mainly trajectory-based queries on recent trajectories and in contrast mainly spatio-temproal range queries on past trajectories it would be beneficial to order the newer chunks by trajectory identifier and the older ones by location (see Figure 3). Also, space-filling curves might be used to improve the scan performance of spatial filter queries. Moreover, this approach allows adoption to changes in the data distribution or the workload. Zhang et al. [14] also address this topic in *TrajSpark* by using a time-decay model to monitor the data distribution and adopt the used indexing schema correspondingly. In our proposed system, these changes lead to a new optimal ordering of data tuples in chunks or compression technique.

Additionally, we developed an approach to store timestamps more efficiently in columnar databases to address the previously mentioned problem that the temporal scan performance often dominates the execution time of trajectory database queries. For that reason, we propose a separate columns approach. The values for year, month, day, hour, minute, second, and fractional second are each stored in a separate column. This approach has some advantages over traditional timestamp concepts. Since there are only several possible values for the different columns, dictionary encoding is effective due to saturated and small dictionaries. Another aspect of reducing the data footprint is that in one chunk only a subset of the different columns (e.g., minutes or seconds) is modified. For that reason, we could minimize the memory consumption of the unmodified columns by applying run-length encoding. Also, we could improve the bandwidth utilization for temporal scan operations. A disadvantage is that for between queries, it is necessary to scan eventually multiple columns.

4.1.2 Workload-aware tiered data compression

Based on the observation that the data access patterns for a trajectory segment change over time, we propose a tiered data compression approach. To select a suitable compression configuration, detailed knowledge about the data characteristics and the workload are required. Most current databases use simple heuristics to choose a compression scheme for a given column [2]. However, we propose a heuristic that selects a compression configuration, which defines the applied compression technique for columns on a per-chunk basis.

Depending on the workload, the determined configuration defines for each chunk, if the spatio-temporal data should be stored uncompressed, lossless compressed with a specific compression algorithm or simplified based on a trajectory simplification algorithm. Additionally, it should also specify the compression ratio of the simplification algorithm per chunk. In general, we expect that recent chunks are uncompressed or only compressed with lossless compression techniques. In contrast, the compression ratio increases with the age of a chunk (see Figure 3). Besides reducing the memory footprint, the selected configuration has also impact on the query performance as well as the accuracy of the results. We see potential to optimize the compression schema for a given memory budget and error-based application constraints (e.g., accuracy metrics [11]). It might be preferable to lose performance or accuracy for less often accessed chunks to leverage the gained space to apply only lightweight compression methods on frequently accessed chunks.

4.2 Planned Evaluation

For the evaluations of our approach, we plan to use two real-world use cases. The first one is in the sports sector and includes positional information of soccer players during professional soccer games [8]. The data is collected with specialized camera systems, which track around 3.5 million data points per game. By analyzing this data, we could observe data access patterns that support our approach [6]. In the post-processing mainly the data of the last game is analyzed. Additionally, algorithms are used to detect and classify specific game events (e.g., passes) on the fine-grained trajectory

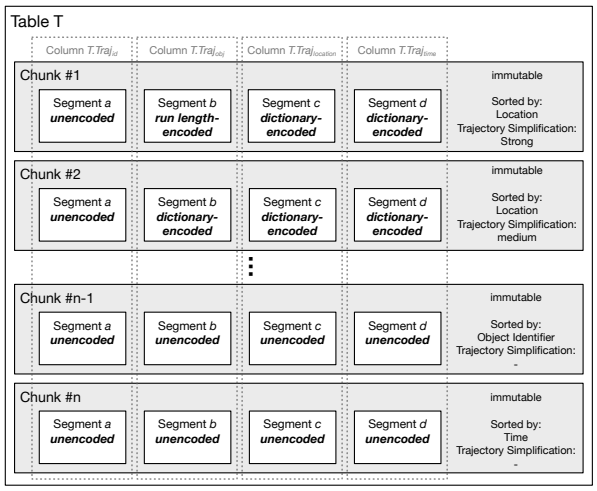

Figure 3: Depiction of the storage layout for an exemplary table T with n chunks and four trajectory relevant attributes

data. The trajectory data of other games of a season is not analyzed in that depth. All analyses on data of past seasons are typically only on an event-based level. The second one is a dataset of a transportation network company, which includes several million driver positions per hour. Due to the amount of the collected data, it is only possible to perform analysis on compressed data or subsets of the data. For various use cases (e.g., driver-passenger assignment) only recent data is analyzed. Also, older trajectories might not represent the current traffic situation.

5. CONCLUSIONS

In this paper, we have presented our approach to store trajectory data in a relational database system. Based on our observation that data points of trajectories are less frequent accessed over time, we developed a workload-aware tired data compression approach to reduce the data footprint and better utilize the available memory bandwidth. Additionally, we presented an optimized data layout for temporal data in the context of in-memory column stores. The first evaluations of the approach demonstrate that the observed data access patterns of different real-world use cases are supporting the proposed system architecture. Upcoming tasks include the further analysis of trajectory compression techniques for columnar databases, the implementation of workload-aware metrics to select compression techniques for chunks, assessing the impact of different configurations and an extensive evaluation with real-world systems.

6. REFERENCES

[1] Reducing the footprint of main memory htap systems: Removing, compressing, tiering, and ignoring data. In *PhD Workshop at VLDB*, volume 2175 of *CEUR Workshop Proceedings*. CEUR-WS.org, 8 2018.

[2] M. Boissier and M. Jendruk. Workload-driven and robust selection of compression schemes for column stores. In *22nd International Conference on Extending Database Technology (EDBT)*, pages 674–677, 3 2019.

[3] M. Dreseler, J. Kossmann, M. Boissier, S. Klauck, M. Uflacker, and H. Plattner. Hyrise re-engineered: An extensible database system for research in relational in-memory data management. In *22nd International Conference on Extending Database Technology (EDBT)*, pages 313–324, 3 2019.

[4] V. Pandey, A. Kipf, D. Vorona, T. Mühlbauer, T. Neumann, and A. Kemper. High-performance geospatial analytics in hyperspace. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2145–2148. ACM, 2016.

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

[6] K. Richly. Leveraging spatio-temporal soccer data to define a graphical query language for game recordings. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 3456–3463, 2018.

[7] K. Richly. A survey on trajectory data management for hybrid transactional and analytical workloads. In *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pages 562–569, 2018.

[8] K. Richly, M. Bothe, T. Rohloff, and C. Schwarz. Recognizing compound events in spatio-temporal football data. In *International Conference on Internet of Things and Big Data (IoTBD)*, 4 2016.

[9] K. Richly, F. Moritz, and C. Schwarz. Utilizing artificial neural networks to detect compound events in spatio-temporal soccer data. In *SIGKDD17 Workshop on Mining and Learning from Time Series (MiLeTS)*, 2017.

[10] K. Richly, R. Teusner, A. Immer, F. Windheuser, and L. Wolf. Optimizing routes of public transportation systems by analyzing the data of taxi rides. In *Proceedings of the 1st International ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics, UrbanGIS@SIGSPATIAL 2015, Bellevue, WA, USA, November 3-6, 2015*, pages 70–76, 2015.

[11] P. Sun, S. Xia, G. Yuan, and D. Li. An overview of moving object trajectory compression algorithms. *Mathematical Problems in Engineering*, 2016, 2016.

[12] H. Wang, K. Zheng, J. Xu, B. Zheng, X. Zhou, and S. Sadiq. Sharkdb: An in-memory column-oriented trajectory storage. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 1409–1418. ACM, 2014.

[13] D. Zhang, M. Ding, D. Yang, Y. Liu, J. Fan, and H. T. Shen. Trajectory simplification: an experimental study and quality analysis. *Proceedings of the VLDB Endowment*, 11(9):934–946, 2018.

[14] Z. Zhang, C. Jin, J. Mao, X. Yang, and A. Zhou. Trajspark: A scalable and efficient in-memory management system for big trajectory data. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, pages 11–26. Springer, 2017.