STEPQ: Spatio-Temporal Engine for Complex Pattern Queries

Dongqing Xiao, Mohamed Eltabakh

Worcester Polytechnic Institute, MA 01604, USA,
{dxiao, meltabakh}@cs.wpi.edu

Abstract. With the increasing complexity and wide diversity of spatio-temporal applications, the query processing requirements over spatio-temporal data go beyond the traditional query types, e.g., range, kNN, and aggregation queries along with their variants. Most applications require support for evaluating powerful spatio-temporal pattern queries (STPQs) that form higher-order correlations and compositions of sequences of events to infer real-world semantics of importance to the targeted application. STPQs can be supported by neither traditional spatio-temporal databases (STDBs) nor by modern complex-event-processing systems (CEP). While the former lack the expressiveness and processing capabilities for handling such complex sequence pattern queries, the later mostly focus on the *Time* dimension as the driving dimension, and hence lack the power of the special-purpose processing technologies established in STDBs over the past decades. In this paper, we propose an efficient and scalable spatio-temporal engine for complex pattern queries (*STEPQ*). STEPQ has several innovative features and ideas that will open the research in the area of integration between spatio-temporal databases and complex event processing.

1 Introduction

The recent advances and wide-spread popularity of mobile devices, wireless cellular phones, and Global Positioning Systems (GPS) have enabled spatio-temporal applications in various domains to continuously monitor and track all objects of interest. Thus with the increasing complexity and wide diversity of spatio-temporal applications, the query and data exploration requirements go beyond the traditional spatio-temporal query types, e.g., *range*, *k nearest-neighbor (kNN)*, and *aggregation* queries [3, 4], to more expressive and semantics-rich spatio-temporal pattern queries (or STPQ) that require higher-order correlation among events. In Table 1, we illustrate several of STPQs from different applications. Evidently, STPQs are prevalent in many applications as they capture real-world semantics that otherwise would have been lost or delegated to the application layer for ad-hoc and inefficient processing. It is not meaningful to assume that a suspicious criminal activity in Q1 or the alert condition for a patient in Q3 depend solely on a single data instance (or even snapshot) of the data stream - rather separate snapshots of instances in the high-speed stream must be trapped at the right moments of time and synchronized to determine the correct match of such a complex STPQ query.

In this paper, we envision the STEPQ system—<u>S</u>patio-<u>T</u>emporal <u>E</u>ngine for complex <u>P</u>attern <u>Q</u>ueries— that addresses the unique challenges of handling STPQs, including: **(1)** They embed powerful semantics not captured by current spatio-temporal query types, **(2)** Unlike traditional query types that can be evaluated on each instance of the database in isolation, STPQs require correlation among spatio-temporal events (both in time and in space) over multiple instances of the database,

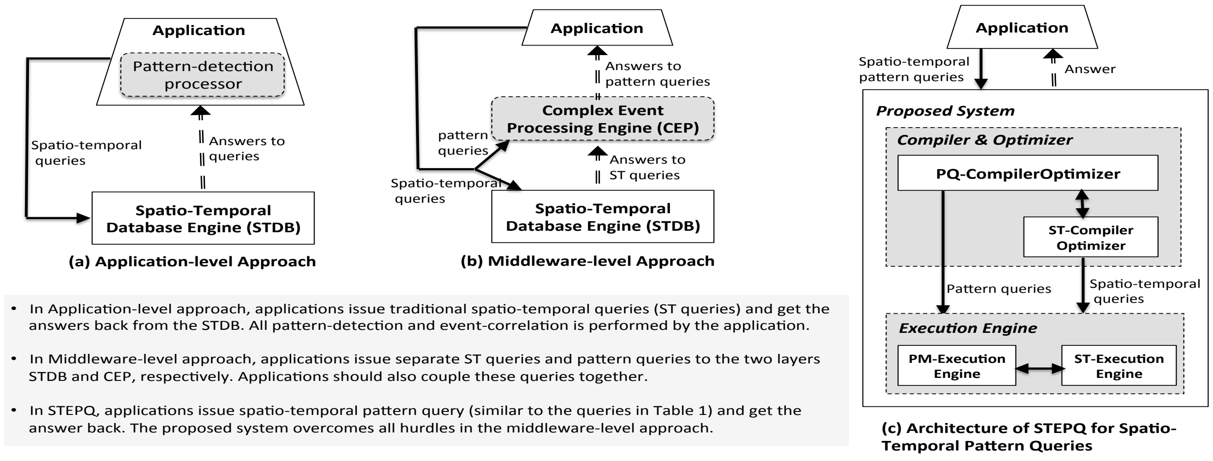
Q1	Report child-abuse criminals who stay in a school area A1 for more than x minutes and then move to a suspicious area A2 within one hour (E.g.,suspicious criminal activity).
Q2	Report cars that stay in my *kNN* over interval T and continuously are getting closer to my moving car.
Q3	Send alert to patient *P*, if she stays in contact (within distance D for at least interval T) with a patient having a transferable disease (E.g., health threat).
Q4	For consecutive areas A1, A2, and A3, report speeding cars (over the speed limit for at least x mins) in A1 and A3 but not in A2 (E.g., testing effect of radar signs over A2 on drivers' behavior).
Q5	Report restaurants located in kNN of two moving cars and getting closer to both cars over interval T (E.g., find common nearby restaurants in direction of moving cars).

Table 1. Examples of spatio-temporal pattern queries (STPQs).

(3) They require a full-fledged query engine equipped not only with efficient event-processing techniques but also with effective spatio-temporal processing capabilities, and **(4)** Unlike state-of-art event-processing techniques (CEP) that have no control over the input stream of events, the STPQs generate these streams of events, and hence crucial optimization strategies can be deployed to control which higher-order events to generate and when. These challenges combined make the state-of-art in complex event processing (CEP), e.g., [8], not applicable since CEP techniques cannot process traditional *range* or *kNN* queries efficiently, also state-of-art in spatio-temporal databases (STDBs), e.g., [2, 7], fall short since they lack the expressiveness power and processing capabilities of handling complex pattern queries.

2 Limitations of State-of-Art Techniques

Conceptually, STPQs can be viewed as two-layered queries where the first layer runs traditional spatio-temporal queries, e.g., *range* and *kNN*, on top of the raw input stream coming from moving objects (we refer to these queries as *base* queries). The second layer runs complex pattern-matching queries on top of the results generated from the base queries. Thus, with the state-of-art technology, there are two possible approaches to support STPQs, namely ***application-level*** and ***middleware-level*** as depicted in Figures 1(a) and (b). In the application-level approach, all of the pattern matching and event correlation is done at the application level to impose the query semantics, which is clearly an ad-hoc and inefficient solution since (1) each application applies its own semantics independently, (2) mobile devices usually have limited power and processing capabilities, and (3) STDBs may send streams of unnecessary results plus the lack of many possible optimizations that could have been performed by the execution engine. The middleware-level approach, which consists of loosely coupled CEP systems, e.g., [8, 1, 5] and STDBs is a more feasible approach. However, it has serious drawbacks and limitations including:
(1) Coupling hurdles: There are several linking problems that emerge between the STDB and CEP layers such as: (a) STDBs deploy incremental evaluation techniques for purposes of efficiency and scalability whereas CEP systems do not handle incremental updates, and (b) The base queries can themselves be moving objects, e.g., Queries Q2, Q3, and Q5 in Table 1, and hence CEP systems need to get as input not only the query answer, but also the query points.
(2) Optimization hurdles: Since STPQs generate both the base queries and the pattern-matching queries, then several optimization opportunities arise that cannot be leveraged with loosely coupled STDB and CEP layers. For example, in Query Q4, the three *range* queries over areas *A1*, *A2*, and *A3* will be concurrently running, although queries over *A2* and *A3* should run only if there is a match in the previous


Application
Pattern-detection processor
Spatio-temporal queries
Answers to queries
Spatio-Temporal Database Engine (STDB)
(a) Application-level Approach
Application
Answers to pattern queries
Complex Event Processing Engine (CEP)
pattern queries
Answers to ST queries
Spatio-temporal queries
Spatio-Temporal Database Engine (STDB)
(b) Middleware-level Approach
Application
Spatio-temporal pattern queries
Answer
Proposed System
Compiler & Optimizer
PQ-CompilerOptimizer
ST-Compiler Optimizer
Pattern queries
Spatio-temporal queries
Execution Engine
PM-Execution Engine
ST-Execution Engine
(c) Architecture of STEPQ for Spatio-Temporal Pattern Queries

- In Application-level approach, applications issue traditional spatio-temporal queries (ST queries) and get the answers back from the STDB. All pattern-detection and event-correlation is performed by the application.
- In Middleware-level approach, applications issue separate ST queries and pattern queries to the two layers STDB and CEP, respectively. Applications should also couple these queries together.
- In STEPQ, applications issue spatio-temporal pattern query (similar to the queries in Table 1) and get the answer back. The proposed system overcomes all hurdles in the middleware-level approach.

Fig. 1. Possible architectures for supporting spatio-temporal pattern queries.

areas.
(3) Synchronization and Transformation hurdles: A STPQ may require not only executing multiple base queries to generate events, but also synchronizing their execution. For example, Query Q5 in Table 1 requires synchronizing the execution of two moving *kNN* queries and then intersecting their results. Such synchronization and transformation over the event streams are not feasible in the middleware-level approach and not even supported by current STDBs.

3 STEPQ System: Vision and Challenges

Given the above limitations, it is clear that engineering existing systems to handle STPQs is not the right approach. In the following we envision the architecture of the proposed STEPQ system and the involved challenges. The system consists of two standard layers; compilation/optimization and execution layers as illustrated in Figures 1(c). In the compilation/optimization layer, the *pattern-query compiler & optimizer* (*PQ-CompilerOptimizer*) component, which is the central component of the system, is responsible for compiling and optimizing the entire query. Given a spatio-temporal pattern query, *PQ-CompilerOptimizer* decomposes it into one or more traditional queries (the *base* queries) and pattern-matching queries. The individual base queries are compiled and optimized using an extended *spatio-temporal compiler & optimizer* (*ST-CompilerOptimizer*) that works under the control of the *PQ-CompilerOptimizer*. In contrast, pattern-matching queries are fully compiled by *PQ-CompilerOptimizer*. The base queries will be executed by the extended spatio-temporal execution engine (*ST-ExecutionEngine*), while the pattern-matching queries will be executed by the *pattern-matching execution engine* (*PM-ExecutionEngine*). The continuously generated results from the base queries will drive the progress of the pattern-matching queries. The key characteristics and challenges in STEPQ are (More details and examples can be found in [6]):

• **Leveraging & extending state-of-art in STDBs:** It is crucial to leverage the existing technology in STDBs. This is achieved by the *ST-CompilerOptimizer* and *ST-ExecutionEngine* components that retain all the innovations in STDBs such as continues and incremental evaluation, spatial-aware operators and access methods, and scalable execution. Moreover, base queries will be subject to new optimizations triggered by *PQ-CompilerOptimizer*.

• **Coherent integration between spatio-temporal and pattern-matching techniques:** This is achieved by having a single system with interacting components orchestrated by the *PQ-CompilerOptimizer*. Such integration allows the sharing of base queries across multiple pattern-matching queries, activating/suppressing base queries when needed, and seamless flow between the generated streams from the base queries to the pattern-matching queries.

• **Cross-cutting optimizations:** This is achieved by the *PQ-CompilerOptimizer* component that enables *PM-ExecutionEngine* to provide feedback information to *ST-ExecutionEngine* to control the execution of the base queries depending on the progress of the pattern queries. Cross-cutting optimizations require new communication mechanisms (and feedback loop) between the pattern-matching and base queries to control what events to generate and when.

• **Synchronized Query Processing:** STPQs may require not only executing multiple base queries, but also synchronizing their execution and jointly processing their results. Hence, new execution plans and synchronization strategies need to be integrated in the evaluation of both spatio-temporal and pattern-matching queries.

• **Event Model and Query Language:** New—possibly extensible—query languages and event models are needed to meet the diverse requirements of STPQs. For example, new concepts such as *event sets* need to be introduced to provide logical grouping of events produced from the base queries. The significance of event sets is two-fold. First, each answer set produced from a base query can be pipelined and processed as one unit, and hence further operations, e.g., synchronization and transformation, can be applied on the event sets. Second, event sets provide an efficient mechanism for anticipating when events should occur in the future, and hence they enable continuity/persistency operations.

References

1. R. Adaikkalavan and S. Chakravarthy. SnoopIB: Interval-based event specification and detection for active databases. *TKDE*, 59(1):139–165, 2006.
2. T. Behr and R. H. Guting. Fuzzy Spatial Objects: An Algebra Implementation in SECONDO. In *In Proceedings of the International Conference on Data Engineering, ICDE*, page 1137 1139, 2005.
3. R. Benetis, C. S. Jensen, G. Karciauskas, and S. Saltenis. Nearest Neighbor and Reverse Nearest Neighbor Queries for Moving Objects. In *Proceedings of the International Database Engineering and Applications Symposium, IDEAS*, pages 44–53, 2002.
4. Y. Cai, K. A. Hua, and G. Cao. Processing Range-Monitoring Queries on Heterogeneous Mobile Objects. In *Proceedings of the International Conference on Mobile Data Management, MDM*, 2004.
5. A. Demers, J. Gehrke, and B. Panda. Cayuga: A general purpose event monitoring system. In *CIDR*, pages 412–422, 2007.
6. M. Eltabakh. STEPQ: Extensible Spatio-Temporal Engine for Complex Pattern Queries. Technical Report WPI-CS-TR-13-02.
7. B. Gedik and L. Liu. MobiEyes: Distributed Processing of Continuously Moving Queries on Moving Objects in a Mobile System. In *Proceedings of the International Conference on Extending Database Technology, EDBT*, 2004.
8. E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 407–418, 2006.