OECEP: Enriching Complex Event Processing with Domain Knowledge from Ontologies

Sebastian Binnewies
Griffith University
Gold Coast Campus
QLD 4222, Australia
s.binnewies@griffith.edu.au

Bela Stantic
Griffith University
Gold Coast Campus
QLD 4222, Australia
b.stantic@griffith.edu.au

ABSTRACT

With the increasing adoption of an event-based perspective in many organizations, the demands for automatic processing of events are becoming more sophisticated. Although complex event processing systems can process events in near real-time, these systems rely heavily upon human domain experts. This becomes an issue in application areas that are rich in specialized domain knowledge and background information, such as clinical environments. We utilize a framework of four techniques to enhance complex event processing with domain knowledge from ontologies to address this issue. We realize this framework in our novel approach of ontology-supported complex event processing, which stands in contrast to related approaches and emphasizes the strengths of current advances in the individual fields of complex event processing and ontologies. Experimental results from the implementation of our approach based on a state-of-the-art system show its feasibility and indicate the direction for future research.

Categories and Subject Descriptors

C.3 [Special-Purpose and Application-Based Systems]: Real-time and embedded systems; I.2.1 [Artificial Intelligence]: Applications and Expert Systems

General Terms

Design, Performance

Keywords

Complex event processing, ontologies, semantic complex event processing

1. INTRODUCTION

Throughout the last decade, the interpretation of data representing state-changing occurrences as *events* gained increasing popularity [7]. This is especially evident in sensor

network applications, cyber-physical systems and the current trend towards an Internet of Things, where data often represents changes to a real-world environment. Many organizations have adopted an event-based perspective for their IT-supported processes and incorporated an event-driven architecture. A core component of these architectures is *complex event processing*. A complex event processing system detects correlated events to generate a *complex event* and triggers responses upon creation of each complex event. Such events can take a multitude of forms, e.g., data from clickstreams or RFID networks, and complex event processing aims to be applicable in a variety of environments. The current situation sees many different approaches to complex event processing in both academic research and the commercial marketplace, as well as the combination of complex event processing with other areas, such as business process management, business activity monitoring and business rule management.

An important part of complex event processing are rules, which reflect the requirements of a specific application domain. As such, rule specification—and, ultimately, the value of decision support—are largely dependent on the amount and quality of knowledge about that domain. This is a critical factor when complex event processing is used in environments that are rich in domain and background knowledge, e.g., in a hospital setting.

In this work, we propose a novel approach to combine complex event processing and domain knowledge from ontologies to address this issue. To realize our approach, we build on a framework of four techniques that enable such a combination [3]. Our modular system interfaces with existing ontologies to integrate the knowledge captured in the ontology into complex event processing. In an experimental evaluation we show the feasibility of our approach on the basis of an existing state-of-the-art complex event processing system.

The remainder of this paper is organized as follows: the next section briefly introduces the background to our research and Section 3 provides an overview of related work. We review the combination framework and present the main contribution of our work in Section 4, followed by a discussion of our experimental evaluation and its results in Section 5. Section 6 then concludes this report.

2. BACKGROUND

Our research bridges the two disciplines complex event processing and knowledge representation in ontologies. The following two sections give a brief overview of the recent advancements in each area.

Figure 1: The complex event processing environment.

2.1 Complex Event Processing

In short, the term complex event processing (CEP) stands for the processing of simple events to generate complex events. The notion of an *event* in CEP is similar to its meaning in everyday language—an event can be any occurrence in an environment. An example would be the information captured in an RFID reading: the time, the location and the object identifier.

A more detailed description of CEP is presented in Figure 1 (based on [7, 15, 22]). A CEP environment consists of three parts—an *event source* (also called event producer), a *CEP engine* and an *event sink* (also called event consumer). Events that occur are passed from the event source to the engine. These events are called *simple events*. The engine then processes events in three steps. First, relevant events are separated from irrelevant events through *event filtering*. Irrelevant events are discarded and only relevant events are passed to the second processing step, *event detection*. In this step, relationships between simple events are discovered, and related events are aggregated to form a *complex event*. Third, *event handling* encompasses all actions ensued by the creation of a new complex event. Such actions include routing complex events to the correct next destination and transforming complex events into the right format for transmission, e.g., a notification or an executable statement. A complex event can be routed to the event sink, which then determines what to do with it. Alternatively or additionally, a complex event can be sent back to the engine as input for further processing.

A considerable number of commercial and academic CEP systems have been developed to date, and current systems provide solutions to many of the inherent challenges in CEP such as high volumes of input data, unordered event streams, uncertainty of event occurrences, temporal relationships, and processing performance (e.g., SASE [27], TESLA [4], ETALIS [2], TMS-RFID [11]). Although one or more systems exist that stand out from others, different systems have different strengths in addressing these challenges. This suggests that there is no single, silver bullet solution in CEP. In fact, a one-fits-all solution may not be feasible at all, due to the wide range of situations in which CEP is applied.

2.2 Ontologies

An ontology formally models knowledge about a domain and allows reasoning over this knowledge. It consists of a set of classes, properties and relationships between the classes and individuals, which are used to represent the concepts of a domain. Additionally, an ontology contains information about the meaning of these concepts and about the logical conclusions that can be drawn from them. The most common use of ontologies is to standardize the terminology in a domain and to facilitate knowledge sharing. Additionally, if an ontology captures the *formal semantics* of concepts, i.e., if it formally defines the meaning of concepts, the semantics can be accessed and processed by machines. An automated reasoner can then analyze concepts and their meaning and compute inferences.

Ontologies have recently received an increasing interest. Various communities have developed ontologies and many of them are freely accessible through the World Wide Web. The work by the Semantic Web community has led to several standards related to ontologies, such as RDF, OWL, RIF and SPARQL [25]. The life sciences have also adopted the technology of ontology modeling and its benefits, especially in the fields of biology and medicine, as evident in the prominent ontologies SNOMED CT [9], the Gene Ontology [20] and further ontologies from the OBO Foundry [18].

3. RELATED WORK

Although research has been conducted in the fields of ontologies and CEP for years, finding possible synergies has only recently been undertaken. It is an emerging research area that is part of *semantic complex event processing*. The latter term is itself relatively new and refers to CEP that is enhanced with reasoning capabilities. Reasoning over domain knowledge captured in an ontology can be considered a sub-area of semantic CEP and only few studies in this sub-area have been published.

EP-SPARQL [1]—an extension to SPARQL—was recently presented as a first complete solution. EP-SPARQL combines CEP and SPARQL to retrieve the semantics of events captured in RDF serialized knowledge bases. While this may be a solution for event input in the form of RDF triples, data in other formats would have to be converted into RDF triples or graphs. Further investigation is required how this conversion affects CEP performance. In addition, EP-SPARQL queries would require the serialization of OWL concepts into less expressive RDF.

Only few other solutions have been proposed, which are limited in one way or another and differ greatly from our approach. Some implement a working combination of ontologies and CEP [6, 12, 16]. However, each approach is restricted to one specific application domain, respectively. The remaining publications primarily discuss the idea of a solution approach and lack the presentation of a working implementation [17, 19, 23]. Two features are common to all these approaches. Firstly, they create a new ontology for the purpose of CEP. Secondly, they propose a tight integration of ontology and CEP by using a combined rule language and by executing CEP completely through reasoning over ontology concepts. Our approach, on the other hand, aims to utilize existing ontologies and does not restrict itself to a specific domain. It is a lightweight approach that focuses on preserving the strengths of existing CEP systems and

Figure 2: Graph-based representation of ontology concepts with rule rewriting and input rewriting paths.

ontologies, which stands in contrast to a tight integration.

4. ONTOLOGY-ENHANCED COMPLEX EVENT PROCESSING

The framework of *ontology-enhanced complex event processing* (OECEP) consists of four techniques that enable the integration of domain knowledge from ontologies in CEP [3]. We first review the techniques and then present our novel approach for OECEP.

4.1 Enhancement Techniques

To illustrate the four enhancement techniques, consider an airport where a large number of containers with food products arrive daily. At the airport’s loading dock, a routing system decides which containers should be routed to customs for inspection based on their content. We can employ a CEP engine to evaluate container information and trigger a response in the form of a routing decision. However, due to the variety of food products and their ingredients, an impracticably large number of CEP rules would be needed. Assuming that information about the food domain is represented in an ontology, we can use techniques from the OECEP framework to access this information, resulting in a significantly reduced number of rules.

4.1.1 Rule Rewriting

In ‘standard’ CEP, each condition of a rule has to be explicitly specified to create a complex event. Using rule rewriting, we can relax some conditions while still achieving the same outcome. The CEP engine initiates the rule rewriting process by passing a relaxed condition and rewriting instruction to the ontology interface. The interface includes a reasoner that computes according to the instruction all inferences related to the condition. The interface passes the inferred conditions back to the engine in order to substitute the initial, relaxed condition.

In the airport example, a container needs to be inspected if it contains some kind of *fish*. Each species of *fish* would have to be specified as a condition in ‘standard’ CEP so that all containers with some kind of fish are found. Rule rewriting allows a relaxation of conditions and we simply have to

Figure 3: Example processes of rule rewriting and input rewriting.

declare *fish* as the only condition. The engine passes this condition to the ontology interface, along with the instruction to infer all instances of *fish* and its subclasses. The engine receives the retrieved instances, substitutes them for the initial condition *fish* and evaluates container information against the new conditions. Figures 2 and 3 illustrate the inference path and rule rewriting process for this example, respectively.

4.1.2 Input Rewriting

Input rewriting enables us to rewrite an attribute value of an input event. Whenever the engine receives a new event, it sends the value of the attribute under consideration to the ontology interface, along with the rewriting instruction. The reasoner derives all relevant inferences and passes them back to the engine in order to fill the attribute with the new values. In our example, an input event is the arrival of a container at the loading dock. Such an event has a *Contents* attribute, possibly populated by the value *WhiteBass*. Using input rewriting, this value is passed to the ontology interface to realize all classes this instance belongs to. The *Contents* attribute is then filled with the inferred classes. Consequently, the engine performs its processing steps over the rewritten event. The inference path and input rewriting process are displayed in Figures 2 and 3, respectively.

4.1.3 Output Rewriting

Output rewriting is similar to input rewriting. In output rewriting, though, the attribute value of a complex event is rewritten. Upon detection of a complex event, the value of one of its attributes is passed to the reasoner, which performs the same steps as in input rewriting. The engine disseminates the rewritten complex event to the event sink or uses it as input for further processing. In the airport example, whenever we detect a container that holds some fish, we create a complex event to route it to customs. Assuming the food ontology stores information about foods that need refrigeration, this output could be rewritten into a complex event routing the container to the temperature-controlled storage at customs.

4.1.4 Incorporating Domain Knowledge

Apart from using ontology information for rewriting, do-

main knowledge from an ontology can be used in additional ways for CEP. By using information about concepts and their relationships, we can define rule conditions that cannot be expressed using ordinary CEP rules. Incorporating domain knowledge into CEP can be realized in different forms. For example, it is possible to specify the condition that the attribute value X of an event A has to participate in at most one relationship φ to the attribute value Y of another event B . The CEP engine has the ability to compare attribute values, however, we require an ontology reasoner to perform the evaluation of the relationship φ . The extent to which we can include domain knowledge obviously depends on the information that is actually captured in the ontology.

If our food ontology models a relationship *hasSeed* between certain foods and their seeds, e.g., stone, single seed, multiple seeds, then it is possible to specify a condition that checks whether a food product contains seeds. In this case, the reasoner receives *Content* values and computes whether at least one relationship *hasSeed* exists. It returns true or false, and the engine proceeds to evaluate the event based on this outcome.

Conceptually, rule rewriting, input rewriting and output rewriting are a form of incorporating domain knowledge. Any combination of rule rewriting, input rewriting and output rewriting, as well as incorporating domain knowledge, can be used to specify rule conditions.

4.2 A Lightweight OECEP Approach

In Section 3, the existing approaches to combine ontologies and complex event processing were presented. Most of them use one or more techniques of the OECEP framework by tightly integrating the ontology with the CEP engine. They specify a common language for ontology modeling and CEP rule specification to enable a full integration of ontology reasoning into each processing stage. In addition, events are often modeled as concepts in the domain ontology or in a separate event ontology. As a result, reasoning over an ontology is conducted for every event pattern rule. We label this sort of approach *ontology-integrated complex event processing* (OICEP).

Several questions arise in the context of OICEP. Firstly, it is not clear from the existing body of work which event information is stored in the (event) ontology. When an engine receives a high-volume input stream, it cannot be feasible to model each event of the stream as a concept in the ontology. Yet, events that form an intermediate result towards a complex event have to be modeled in the ontology so that these are included in the reasoning step when future events are evaluated. Additionally, it is not clear how event information is converted into the ontology data format, e.g. RDF, and how this conversion affects performance. Finally, the question arises where the line between the domain ontology and the event ontology is drawn.

Here, we introduce the notion of *ontology-supported complex event processing* (OSCEP). In contrast to OICEP, OSCEP does not model events in an ontology. It refers to the ontology only when this is required by the event or rule to be evaluated. OSCEP is a strictly modular and minimalistic approach. It aims to preserve the inherent characteristics of both CEP and ontologies. It thereby acknowledges the advancements in both areas and tries to leverage their strengths for a combined solution.

Figure 4 illustrates the process architecture in an OSCEP

Figure 4: OSCEP process architecture.

system. It shows how the CEP algorithms for event filtering, detection and handling as well as the ontology data remain autonomous of rewriting processes. Rewriting processes are only triggered between processing steps. The ontology serves as a vocabulary for rewriting and is accessed through a query or inference engine. Input rewriting is applied to input events after filtering out irrelevant events. Output rewriting is used to rewrite a complex event or an intermediate result, which is an event or a combination of events that partially satisfies a CEP rule. Rule rewriting is applied directly to the rulebase when new rules are added and, as such, is also independent of the CEP mechanism.

Since OSCEP keeps ontologies and CEP separate, the questions relating to OICEP from above do not apply. In fact, OSCEP is designed as an approach on the opposite side of OICEP in the spectrum of OECEP solutions. It remains to be seen if these two approaches can complement each other in different application settings, or if one is clearly superior to the other. A thorough comparison between OICEP and OSCEP in regards to functionality and performance will form the basis for our future research.

5. EXPERIMENTAL EVALUATION

To demonstrate the viability of OSCEP using our enrichment techniques, we conducted an experiment based on the CEP system SASE. The following two sections briefly introduce SASE and explain the experiment design. Section 5.3 then discusses the experiment results.

5.1 CEP System: SASE

We have selected SASE as the basis for our experiment for two reasons. Firstly, SASE has received much attention in the academic CEP community (e.g., [5, 13, 14]). Secondly, SASE has been improved continually by the developers and the open source code for SASE 1.0 is now publicly available [21].

Initially, SASE was introduced to handle high-volume event streams [26]. It consists of two components—an event language and an event processing mechanism that is based on a query plan. While the SASE language is primarily based on other event languages, it includes several additions. It offers expression of value-based and negation constraints and the specification of a sequence of simple events in a certain order. Additionally, a rule can include time interval constraints, or

sliding windows, in which all primitive events must occur. The semantics of five rule operators are formally defined by a translation into logical operators and quantifiers. A query plan serves as the foundation for SASE’s event processing mechanism, which utilizes automata for detecting and aggregating simple events.

A critical extension to SASE is SASE+, which allows event pattern rules to include Kleene star operators [8]. Since rule evaluation is carried out using an automata model, this makes it possible to capture the permutations and concatenations of events that will (not) advance the automaton. The latest extension to SASE [27] addresses the problem of evaluating event pattern queries when the timestamps of events are imprecise. The authors propose a *temporal uncertainty model* that includes an uncertainty time interval, which consists of a lower and upper bound for an event, and an optional probability function, which denotes the likelihood of occurrence for each element in the uncertainty interval. The version used for our experiment is SASE 1.0, which includes the Kleene star operator, but does not allow queries over events with imprecise timestamps.

5.2 Experiment Design

Table 1 shows the design variables of our experiment. We chose the OWL API as an interaction mechanism and HermiT as the ontology reasoner. This choice is based on results from our initial testings [3], which showed that the OWL API can outperform SPARQL for input rewriting, even though its expressiveness is higher. Using OWL API functions, we created a module that integrates into the SASE system and that enables data exchange between SASE and HermiT. We further selected the Food Ontology [24] as the test ontology. The Food Ontology models knowledge about food in 138 named classes, 206 individuals, 17 properties and 356 axioms. We utilize this ontology to test input rewriting, as this is the technique that is most dependent on the number of events to be processed. In contrast, rule rewriting is independent of the number of events to be processed since it is only executed per rule and not per event. In our experiment, we randomly assigned each input event one of four individual names, i.e., ‘Tuna’, ‘Crab’, ‘Cheese’ or ‘Nuts’. Our module passes these individuals to the reasoner, which computes all superclasses of these individuals. It then rewrites the event and evaluates the condition if one of these superclasses equals ‘Seafood’.

We varied the size of the input stream between 1000, 10,000 and 100,000 events and executed twenty runs each. To compare the results, we processed the same stream sizes over the same number of runs and based on the same event evaluation rule with standard SASE. All tests were carried out on a Mac with OS X Version 10.6.8, Intel Core 2 Duo 2.2GHz and 2GB of RAM. Programming code was written in Java using Java 1.6.0.

5.3 Results and Analysis

Figure 5 shows the average processing time per event in microseconds for three different input stream volumes, each evaluated with and without rewriting. As can be seen from the chart, the results are mixed across the variations, ranging from a total of 144 μ s (1000 input events) to 77 μ s (10,000 and 100,000 input events) for input rewriting and from a total of 27 μ s (1000 input events) to 7 μ s (10,000 input events) and 5 μ s (100,000 input events) for standard

Table 1: Experiment Design Variables

Processing Methods	Standard SASE / SASE with Input Rewriting
Input Size	1000 Events / 10,000 Events / 100,000 Events
Ontology (for Rewriting)	Food Ontology (OWL DL <i>SHOIN</i>)
Num. of Individuals (for Rewriting)	4
Interaction Mechanism	OWL API 3.2.2 with HermiT 1.3.3

Figure 5: Performance results for input rewriting and comparison to standard processing.

processing. We conjecture that this difference is caused by one-time overhead costs in both rewriting and processing that are amortized sufficiently from at least 10,000 input events.

When comparing the pure processing time with and without rewriting for each stream size, the chart shows only minimal differences. In our experiment, the event value containing an individual is rewritten into one superclass only. Therefore, the workload of the pure rule evaluation process should not increase. This is supported by these test results. Furthermore, it is evident from the chart that rewriting contributes significantly to the total processing time. It makes up 84% of the total processing for 1000 input events and 94% for 10,000 and 100,000 input events. Further investigation is required how to improve the performance. Thorough comparisons with rule rewriting, with SPARQL as the interaction mechanism and with other ontologies need to be conducted.

Even though the scope of our experiment is limited, we obtained some interesting initial results. We incorporated input rewriting into an existing CEP system, from which we conjecture that the remaining techniques—rule rewriting, output rewriting and incorporating domain knowledge—can be incorporated in a similar manner. However, even for realistic stream sizes of 10,000 events and above, input rewriting makes up a significant amount of total processing time.

Further optimization is thus required. In addition, we can safely assume that accessing a more complex ontology than the Food Ontology will increase processing time. Future research should therefore investigate how to speed up the reasoning process, possibly by considering only modules of an ontology.

6. CONCLUSION

Our study addressed the limitation of rule accuracy in existing CEP systems when used in information-rich environments. We proposed a new approach that effectively uses domain knowledge from ontologies to overcome this limitation.

More specifically, this study makes the following contributions to the field:

- We introduced the concept of ontology-supported complex event processing based on a framework of four enhancement techniques for CEP systems. This modular approach interfaces with the ontology between processing steps, thus allowing us to leverage the strengths of existing processing methods.
- We implemented our approach on the basis of a state-of-the-art CEP system and utilized an existing ontology. The experimental results show the feasibility of our approach and indicate the direction for further research.

Future work will aim to access modules of SNOMED CT with the help of the classifier Snorocket [10] to support CEP in e-Health environments.

7. REFERENCES

- [1] D. Anicic, P. Fodor, S. Rudolph, and N. Stojanovic. EP-SPARQL: A unified language for event processing and stream reasoning. In *WWW 2011*, pages 635–644. ACM, 2011.
- [2] D. Anicic, P. Fodor, S. Rudolph, R. Stühmer, N. Stojanovic, and R. Studer. A rule-based language for complex event processing and reasoning. In *RR 2010*, volume 6333 of *LNCS*, pages 42–57. Springer Berlin / Heidelberg, 2010.
- [3] S. Binnewies and B. Stantic. Introducing knowledge-enrichment techniques for complex event processing. In *ICIEIS 2011*, volume 253 of *CCIS*, pages 228–242. Springer Berlin Heidelberg, 2011.
- [4] G. Cugola and A. Margara. TESLA: A formally defined event specification language. In *DEBS 2010*, pages 50–61. ACM, 2010.
- [5] L. Ding, S. Chen, E. Rundensteiner, J. Tatemura, W. Hsiung, and K. Candan. Runtime semantic query optimization for event stream processing. In *ICDE 2008*, pages 676–685. IEEE, 2008.
- [6] J. Dunkel. On complex event processing for sensor networks. In *ISADS 2009*, pages 1–6. IEEE, 2009.
- [7] O. Etzion and P. Niblett. *Event Processing in Action*. Manning, Greenwich, CT, 2011.
- [8] D. Gyllstrom, J. Agrawal, Y. Diao, and N. Immerman. On supporting Kleene closure over event streams. In *ICDE 2008*, pages 1391–1393. IEEE, 2008.
- [9] International Health Terminology Standards Development Organisation. SNOMED CT. <http://www.ihtsdo.org/snomed-ct/>, 2012.
- [10] M. Lawley and C. Bousquet. Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In *Australasian Ontology Workshop 2010: Advances in Ontologies*, pages 45–50. ACS, 2010.
- [11] X. Li, J. Liu, Q. Sheng, S. Zeadally, and W. Zhong. TMS-RFID: Temporal management of large-scale RFID applications. *Information Systems Frontiers*, pages 1–20, 2009.
- [12] D. Liu, C. Pedrinaci, and J. Domingue. A framework for feeding Linked Data to complex event processing engines. In *The 1st International Workshop on Consuming Linked Data (COLD 2010) at ISWC 2010*, 2010.
- [13] H. Liu, Q. Chen, and Z. Li. Optimization techniques for RFID complex event processing. *Journal of Computer Science and Technology*, 24(4):723–733, 2009.
- [14] Y. Mei and S. Madden. ZStream: A cost-based query processor for adaptively detecting composite events. In *SIGMOD 2009*, pages 193–206. ACM, 2009.
- [15] A. Paschke and P. Vincent. A reference architecture for event processing. In *DEBS 2009*, pages 25:1–25:4, 2009.
- [16] H. Paulheim. Efficient semantic event processing: Lessons learned in user interface integration. In *The Semantic Web: Research and Applications*, volume 6089 of *LNCS*, pages 60–74. Springer, 2010.
- [17] K. Schmidt, D. Anicic, and R. Stühmer. Event-driven reactivity: A survey and requirements analysis. In *SBPM 2008*, pages 72–86, 2008.
- [18] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. Goldberg, K. Eilbeck, A. Ireland, C. Mungall, et al. The OBO foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25:1251–1255, 2007.
- [19] K. Teymourian. Enabling knowledge-based complex event processing. In *EDBT 2010 Workshops*, pages 1–7. ACM, 2010.
- [20] The Gene Ontology Consortium. The Gene Ontology. <http://www.geneontology.org/>, 2012.
- [21] University of Massachusetts Amherst. SASE – Code Release. <http://avid.cs.umass.edu/sase/index.php>, 2012.
- [22] F. Wang, S. Liu, and P. Liu. Complex RFID event processing. *The VLDB Journal*, 18(4):913–931, 2009.
- [23] L. W. M. Wienhofen and P. J. Toussaint. Enriching events to support hospital care. In *MDS 2010*, pages 26–30. ACM, 2010.
- [24] World Wide Web Consortium. OWL Web Ontology Language: Guide. <http://www.w3.org/TR/2004/REC-owl-guide-20040210/food.rdf>, 2004.
- [25] World Wide Web Consortium. Semantic Web Standards. http://www.w3.org/2001/sw/wiki/Main_Page, 2012.
- [26] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *SIGMOD 2006*, pages 407–418. ACM, 2006.
- [27] H. Zhang, Y. Diao, and N. Immerman. Recognizing patterns in streams with imprecise timestamps. *Proceedings of the VLDB Endowment*, 3(1):244–255, 2010.