Exploiting Database Technology for Expressive and Efficient Complex Event Processing

Simon Brodt
Supervisor: François Bry Institute for Informatics
University of Munich
http://www.pms.ifi.lmu.de/
simon.brodt@ifi.lmu.de

ABSTRACT

Recent research [34] shows that, provided a careful design, complex event processing (CEP) on top of a modern DBMS can compete in many cases with specialized DSMS which have been considered superior for a long time [8]. There are mainly two reasons why building complex event processing on top of an existing DBMS is desirable. First event processing profits form sophisticated techniques and algorithms developed during decades of database research. Second new applications are demanding more functionality, particularly the integration of static and stateful data, but also more expressive event queries and the maintenance and later analysis of logs. These new requirements force specialized DSMS to actually re-implement common database functionality.

Starting from that point this doctoral project aims at building an expressive and efficient complex event processing system using a DBMS. For this we first define Temporal Stream Algebra TSA which is an generalization of relational algebra to so-called temporal streams (actually relational algebra is an extremal case of TSA). Second we develop Event-Mill, a system for efficiently evaluating TSA expressions using a database engine. TSA and Event-Mill are designed to meet the requirements from new ambitious applications of complex event recognition like emergency management in large infrastructures. This includes expressive temporal relations, flexible grouping and aggregation, the integration of static and stateful non-event data, the simultaneous use of multiple timestamps and time models, e.g. application and system time, selective logging, subquery sharing, operator reordering, automatic garbage collection, situation dependent query prioritization, bulk-wise, asynchronous and distributed processing, publish-subscribe dissemination and fault resistance.

The doctoral project is part of the international research project EMILI on using CEP for novel forms of emergency management in large infrastructures.

DEBS 2011 Yorktown Heights, NY USA

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory—*semantics*; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages—*Algebraic approaches to semantics, Operational semantics, Program analysis*; H.2.3 [**Database Management**]: Languages—*Query languages*

General Terms

Theory, Algorithms, Languages, Verification

Keywords

Complex Event Processing, CEP, Temporal Stream Algebra, Event-Mill, Incremental Query Evaluation, Temporal Analysis, Optimization

1. MOTIVATION

The continuous and timely analysis of event-streams, often referred to as Complex Event Processing (CEP), has become a widely recognized discipline in the last decade. In the meantime numerous full-fledged data stream management systems (DSMS) [18, 24, 38, 7, 19, 50, 30, 1] tackle the derivation of higher-level knowledge from a stream of low-level events. Most of these systems are based on the assumption that the architecture of existing DBMS is inadequate [8] for an efficient processing of complex events. The simple requirements of early CEP applications [34] and experimental results seem to support this assumption [8]. Therefore specialized DSMS have been designed and built from scratch.

Meanwhile the increasing recognition and acceptance of CEP opens new challenging applications like detecting and managing emergencies in large infrastructures. The requirements derived from these ambitious applications and recent research results [34] strongly suggest that DBMS should be reconsidered as basis for expressive and efficient complex event processing systems.

At the same time CEP still lacks an equally strong formal foundation as traditional database systems find in relational algebra. Present approaches [33, 19, 23] towards a formal foundation for CEP have serious limitations. Complex temporal relations can not be expressed at all or need unnatural and complicated constructions. The simultaneous use of multiple time-stamps, e.g. application- and system-time and/or different time models are not considered at all. Grouping and aggregation capabilities are very restricted.

The proposed evaluation does neither account for an bulk-wise, asynchronous nor distributed processing. Finally the design of formal semantics does not unleash the full potential of optimization like operator-reordering and subquery sharing.

For the above reasons the doctoral project presented here aims for three main objectives:

- Meeting the increased requirements of arising new applications for CEP
- Exploiting existing database technology
- Providing a strong formal foundation for DBMS-based CEP

As mentioned earlier the doctoral project is part of the international research project EMILI funded by the European Commission[1] which is dedicated to exploring the benefits of CEP technology for a proper detection and management in large infrastructures. This is one of the upcoming new and challenging applications for CEP.

The project EMILI focuses on three use-cases namely underground metro systems, airports and power grids. A detailed description is available in [48, 49]. One main lesson learned from these use-cases is that considering the context of an event is essential for a good situation assessment and appropriate reactions. The context is given by static data describing the surveyed infrastructure, e.g. its topology, the placement of sensors or possible escape routes, and stateful data either describing the state of some physical component, e.g. the ventilation regime or the availability of some staircase, or an abstract state of the emergency management system e.g. the current operation mode. This implies that static and stateful date have to be tightly integrated into the reasoning on complex events.

A simple example adopted from the metro use-case might illustrate this. Consider a burning train stopping inside a metro station. Unfortunately the fire is close to one staircase which therefore is likely to quickly fill with smoke and should not be used for evacuation. For correctly assessing the situation and choosing an appropriate reaction the the system needs to combine the alarm events from smoke and temperature sensors with static data about the location of sensors and staircases. In this way the system can conclude that one staircase is close to the fire, thus will be blocked shortly and should not be used for evacuation. Consequently the the evacuation initiated by the system directs people to other staircases/exits. Furthermore the system changes to emergency/fire mode as reaction to the first series of fire detections. This change in operation mode, i.e. this change in stateful data, affects the way how further events reporting high temperature or smoke alarms near to the known fire location are treated. Instead of raising further fire alarms and confronting human operators with an avalanche of useless re-indications of the same fire, the new events are not reported at all or are presented as detected spread of the existing fire. In this way operators are only provided with relevant information and can concentrate on the evacuation process.

Further requirements derived from the EMILI use-cases are expressive temporal relations, the simultaneous use of multiple time-stamps and strong formal foundations. Particularly the latter is important in case of emergency management as security experts will hardly accept an approximate knowledge of "what the system will do".

[1]grant agreement number 242438

Interestingly other use-cases which are completely independent from emergency management apparently have similar requirements. This holds for the auction use-case examined in the QONCEPT project[2] of Olga Poppe [42] as well as for general business process applications [15]. Furthermore it is in accordance with the requirements analysis in [47].

The need for an tight integration of static and stateful data is one of the main motivations for considering a DBMS as basis for expressive complex event processing. Another motivation is encouraging recent research [34] which shows that, provided a careful design, a modern DBMS can be turned into an efficient CEP system. In this way sophisticated techniques and algorithms from database research and tuned implementations of these techniques and algorithms can be exploited for CEP.

As formal semantics are important for CEP in general and emergency management in particular and exploiting database research is already on the table, it seems to be a natural idea to use a generalisation of relational algebra as formal foundation for CEP. This idea is also backed by the observation that despite some fundamental differences, CEP and traditional databases share strong similarities with respect to the basic intention and structure of query expressions. This similarity already inspired a number of attempts to transfer relational algebra to CEP. While keeping the basic idea, compared to previous approaches we choose quite a different way for bringing relational algebra to CEP. Instead of extending or adopting relational algebra we intend to generalize relational algebra towards so-called temporal streams. This is done in such a way that no new (basic) operators are needed and the characteristics of these operators, e.g. with respect to reordering, are fully preserved. By this well known optimization techniques and algorithms from database theory are directly applicable. The algebra resulting from the generalization of relational algebra is called Temporal Stream Algebra (TSA).

Finally we propose Event-Mill, an evaluation approach which on the one hand is closely backed by the theory on incremental evaluation of TSA and on the other hand is particularly suited for an implementation on top of an existing data-base system. The main idea is to continuously execute incremental TSA-queries. The incremental TSA-queries perform a bulk-wise processing and their execution is not driven by single events (or events at all) but by response time constraints which require reevaluation at a certain frequency. Note that this is significantly different from RETE-based evaluation in production rule systems, from the RETE-like evaluation of CERA [21, 23] and from automaton-based approaches [19]. Moreover Event-Mill enables asynchronous and distributed processing and situation dependent query prioritization.

2. STATE OF THE DOCTORAL PROJECT

2.1 Investigation of Emergency Management Use Cases

The requirements analysis for the three EMILI use-cases, namely public transport systems, airports and power-grids has been completed in joint-work with Steffen Hausmann

[2]funded by the Deutsche Forschungsgemeinschaft (DFG) under reference number BR 2355/1-1

[12, 11]. The derived requirements can be divided into two categories: Requirements concerning the expressiveness of queries and those towards query evaluation.

Requirements to the expressiveness of Queries:.

Three high-level concepts must be representable namely events, temporary objects and actions. Temporary objects can be used to represent static and stateful data. They be created and terminated. Temporary objects are visible from the beginning of their lifetime and can exist for an arbitrary time until they are terminated. In contrast to that, events are visible only at the end of their lifetime, i.e. the end of an event is known as soon as the event becomes visible.

Expressive temporal relations. Complex combinations of temporal relations like before or within 2 minutes are needed. For example it should be possible to express all $2^1 3$ temporal relations of Allen's interval algebra [6]. Sliding, tumbling or other time-windows of a fixed size applied to the inputs of a query are not suitable for representing temporal relations.

Multiple time-stamps. All use-cases need at least three different time-stamps in parallel. The first two can be identified with application- and system-time. The third time is mainly due to simulated events used for predictions on the future development of an emergency. For simulated events there is a difference between the time where the underlying data is available (applications / system-time) and the future time for which the simulated event makes an prediction. (For example simulated events are needed for determining whether some staircase will remain free of smoke, i.e. will be safe, for the full duration of an evacuation and thus can be used for evacuation.)

Flexible grouping and aggregation. We need some mechanism to somehow turn grouping and aggregation as reaction to some (possibly complex) event, to apply the full power of reasoning for selecting the events for aggregation and to finally stop grouping and aggregation as reaction to another event. For example one might want to count the number of persons leaving the station starting with the first fire-alarm and stopping with the arrival of the fire brigade.

Requirements to query evaluation.

Robustness against peeks in the event-load. In emergency management, peeks typically result from beginning emergencies when more and more sensors start firing alarm events. Thus peaks in the event load appear at the same time where the fastest response-time is needed.

Situation dependent query prioritization. During an emergency those queries which are most important for the emergency should be processed with the highest priority as to ensure short response times. Less important queries could be delayed, be evaluated less frequently or be completely ignored.

Automatic garbage collection. Garbage collection is essential for preserving a high system performance. Indeed events that are not further relevant should be discarded as soon as possible to free memory and to avoid unnecessary computations. However defining a correct garbage collection strategy explicitly, e.g. choosing the right time windows, is not feasible for the security experts writing the emergency management rules. Furthermore explicit garbage collection massively reduces maintainability of the CEP-program. Therefore we strive for an automatic garbage collection where relevance conditions are statically derived from temporal relations specified in the query and the constraints on temporal relations that are part of the schema.

2.2 Temporal Stream Algebra – TSA

TSA aims to be a generalization of relational algebra [2, 25]. The design of TSA follows a number of working assumptions: TSA should have the same set of minimal and orthogonal operators as relational algebra namely selection σ, projection $\dot{\pi}$, imbedding ι, cross product $\otimes$, set difference $\setminus$, grouping γ and union $\dot{\cup}$, where ι serves for computing new attributes from existing ones and is typically part of the projection operator. TSA should fully preserve the properties of the operators particularly with regards to operator permutations. TSA must be capable to simultaneously manage multiple timestamps and time models (even continuous ones). TSA must enable the formulation of complex temporal relations on multiple events. TSA makes minimum assumptions on the semantic of timestamps as to allow a maximum flexibility for higher-level languages building on TSA.

The definition TSA bases on three main ideas

- The concept of Temporal Streams
- The integration of constraints on temporal relations between timestamp attributes into the schema of an relation
- The restriction of operator applications according to the constraints on temporal relations between timestamp attributes

Temporal Streams. Basically a temporal stream is a potentially infinite relation with designated timestamp attributes, that always has a finite past. This means that when restricting all timestamp attributes of the temporal stream with an upper bound, then the resulting prefix of the stream, i.e. the "past", has only finite size. This just reflects the fact that up to any point in time every CEP-system will only receive a finite amount of data. The only restriction for the domain of timestamp attributes is that it must be totally ordered, especially temporal domains do not have to be discrete ($\mathbb{Z}$) but can be continuous ($\mathbb{Q}, \mathbb{R}$).

Temporal streams serve as common algebraic representation for the three high-level concepts events, temporary objects and actions mentioned in Section 2.1. All three concepts are represented as tuples of a temporal stream. At that point we exploit the fact that TSA does not make assumptions about the semantics of time-stamps as the semantics of time-stamps is likely to be different in the three cases events, stateful objects and actions.

Stateful data represented using temporary objects implies the need for recursion as the new state frequently depends on the preceding one. However it seems to be sufficient to allow a restricted form of recursive TSA-expressions called temporal hierarchical TSA-expressions where cycles have to make temporal progress. [3] In other words recursion in the same time-point is not allowed (as it is not for relational algebra) and the past may not depend on the future. However recursion to the future is allowed. In this way termination for

[3]The term temporal hierarchical is chosen as the required temporal progress for cycles on the level of temporal streams implies that dependencies on the data level are cycle-free, i.e. hierarchical.

each time-point and each finite stream prefix is guaranteed. Without time bound temporal hierarchical TSA-expressions are Turing-complete, though.

An interesting property of temporal streams is the fact that in the absence of timestamp attributes, a temporal stream actually describes a normal (finite) database relation. When using only such relations, the operators of TSA behave exactly as usual relational algebra operators. Thus TSA is a clean generalization of relational algebra. Obviously this enables a very smooth integration of static data.

Constraints on temporal relations. Constraints are used to provide information on temporal relations at the schema level. The information is implicitly collected when a selection on temporal relations is applied or when new relative time-stamps are computed. The information is stored in the constraints and is propagated to subsequent TSA operators in this way.

The information on temporal relations is mainly used for three purposes: First for determining the correctness of an TSA-expression. This is immediately related to the restrictions of operator applications described in the following. Second for deriving wait conditions which are needed for the evaluation of grouping, aggregation and negation. Wait conditions ensure that the processing of an tuple is delayed until all other tuples relevant for the processing, e.g. the tuples in the same group, have arrived, too. For deriving relevance conditions. Relevance conditions are used to reduce the size of the input to an incremental TSA-expression. The worst-case over all relevance conditions of some temporal stream defines the relevance conditions needed for automatic garbage collection.[4]

Restriction of operator applications. The applications of operators is restricted for two reasons: First to ensure that grouping, aggregation and negation can be evaluated and second for achieving answer-closedness. Answer-closedness in this case means that applying an operator to an temporal stream results in a temporal stream again. The other property which is important for the evaluation of grouping, aggregation and negation is called temporal preservation. The temporal preservation property guarantees that the computation of a finite prefix of the result stream of an TSA-expression needs only finite prefixes of the input streams of the expression and the size of the needed prefixes can be statically determined. The necessary restriction mainly affects the projection and the grouping operator. Crudely spoken the temporal relations must imply that the values of the remaining timestamp attributes upper bound[5] the values of the discarded timestamp attributes.

2.3 Event-Mill

Event-Mill is a natural and straight-forward approach for implementing the theory of incremental evaluation of TSA-expressions. The basic idea is that like in a turning mill all incremental TSA-Expressions are continuously evaluated. With each evaluation round all events that have arrived since the last round are processed at once, i.e. a bulk-wise processing is performed (see Figure 1). Event-Mill can easily be implemented on an existing database system as incremental TSA-Expression can be read as normal relation algebra expression.

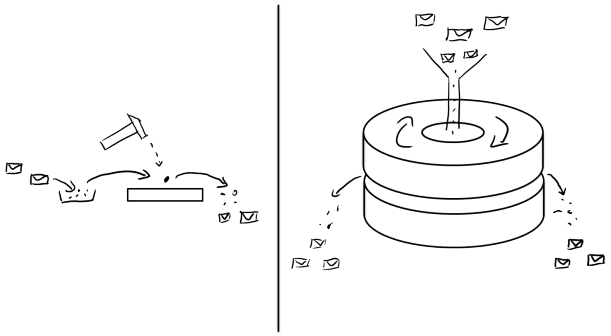

Figure 1: In contrast to traditional approches Event-Mill processes several tuples at a time

We expect that Event-Mill fullfils the requirements formulated in Section 2.1. Due to the bulk-wise processing should be robust against peaks in the event load. A processing strategy reacting on single events is not likely to fulfill this requirement. Furthermore it allows situation dependent query prioritization basically by running incomplete rounds. Thus Event-Mill is apparently well-suited for the requirements of emergency management but probably also for those of many other applications. However Event-Mill might not be the optimal choice if an application demands for a minimum response time on the arrival of a single event.

3. RELATED WORK

Temporal Algebras [51, 36, 40] are used in so-called Temporal Databases which are one of the (many) ancestral fields of CEP. However Temporal Algebras do neither account for the stream aspect of CEP nor for the incremental evaluation needed to the potentially infinite relations representing event streams.

Composition Operator Approaches are characterized by the use of composition operators, particularly the sequence operator, for building complex events and expressing temporal relations. The classification comes from [22] where those approaches are classified as "composition operator languages". The underlying semantics (if such) typically uses the term Event Algebra, however they have little in common with relational algebra. The following systems and approaches belong to this category: Amit [5], ruleCore [50, 39], CAYUGA [19] and [28, 26, 27, 17, 3, 4, 35, 54, 9, 53, 20, 13, 14, 44, 37, 29, 16, 10, 46, 45, 41].

Datastream Approaches typically use an SQL-like query language and are classified as "data stream query languages" in [22]. Due to their SQL-like query they are relatively close to relational algebra. Some of them even have an formal semantics which is an adoption relational algebra [33]. This way of adoption is however completely different than the one chosen for TSA. The following systems fall into this category: PIPES / Logical Stream Algebra [32, 33], TelegraphCQ [18, 43], Aurora / Borealis / StreamBase / StreamSQL [52, 30, 1], CQL / STREAM [7], Esper [24], Coral8 [38] and DataCell[6] [31, 34]

[4]Note that the quality of temporal analysis affects the quality of wait- and relevance conditions and therefore might have a strong impact on the performance of the evaluation.

[5] In the case of grouping, the time-stamp attributes within the grouping attributes.

[6] Note that from the point of evaluation the DataCell approach has some familiarity to Event-Mill. It utilizes bulk

CERA is the Complex Event Relational Algebra of Michael Eckert [21, 23] proposed as operational semantics for the logic event query language XChange^{EQ}. TSA picks up many of the ideas of CERA but somehow makes a restart as to achieve minimal and orthogonal operators which are better suited for optimizations. Furthermore TSA provides more expressive temporal relations, a significantly more flexible grouping a better temporal analysis resulting in an improved garbage collection and an more evolved (particularly asynchronous) incremental evaluation.

4. CURRENT AND FUTURE WORK

The definition of TSA is almost completed but the asynchronous variant and the details of temporal analysis still need some work. The Event-Mill approach is completely designed and a prototype is on the way. As soon as the prototype is available both TSA and Event-Mill should be evaluated experimentally. Furthermore a theoretic analysis of their computational complexity would be interesting. As we expect that TSA preserves the nice properties of relational algebra particularly with respect to operator permutations the possibilities of applying optimizations algorithms known from relational algebra like sub-query sharing operator reordering should be examined. Further optimizations could also be examined. The QONCEPT project [42] of Olga Poppe aims at that direction.

5. REFERENCES

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. B. Zdonik. Aurora: A new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] R. Adaikkalavan and S. Chakravarthy. Formalization and detection of events using interval-based semantics. In *Proc. Int. Conf. on Management of Data (COMAD)*, pages 58–69. Computer Society of India, 2005.

[4] R. Adaikkalavan and S. Chakravarthy. SnoopIB: Interval-based event specification and detection for active databases. *Data and Knowledge Engineering*, 1(59):139–165, 2006.

[5] A. Adi and O. Etzion. Amit — the situation manager. *The VLDB Journal*, 13(2):177–203, 2004.

[6] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[7] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: Semantic foundations and query execution. *The VLDB Journal*, 15(2):121–142, 2006.

[8] A. Arasu, M. Cherniack, E. F. Galvez, D. Maier, A. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts. Linear road: A stream data management benchmark. In *Proc. Int. Conf. on Very Large Databases*, pages 480–491. Morgan Kaufmann, 2004.

[9] R. S. Barga and H. Caituiro-Monge. Event correlation and pattern detection in CEDR. In *Proc. Int. Workshop Reactivity on the Web*, volume 4254 of *LNCS*, pages 919–930. Springer, 2006.

[10] M. Bernauer, G. Kappel, and G. Kramler. Composite events for XML. In *Proc. Int. Conf. on World Wide Web*, pages 175–183. ACM, 2004.

[11] S. Brodt, S. Hausmann, and F. Bry. Deliverable D4.2: Reactive rules for emergency management, 2010.

[12] S. Brodt, S. Hausmann, F. Bry, O. Poppe, and M. Eckert. Deliverable D4.1: A survey on IT-techniques for a dynamic emergency management in large infrastructures, 2010.

[13] F. Bry, M. Eckert, and P.-L. Pătrânjan. Querying composite events for reactivity on the Web. In *Proc. Int. Workshop on XML Research and Applications*, volume 3842 of *LNCS*, pages 38–47. Springer, 2006.

[14] F. Bry, M. Eckert, and P.-L. Pătrânjan. Reactivity on the Web: Paradigms and applications of the language XChange. *J. of Web Engineering*, 5(1):3–24, 2006.

[15] F. Bry, M. Eckert, P.-L. Pătrânjan, and I. Romanenko. Realizing business processes with eca rules: Benefits, challenges, limits. In *Proceedings of 4th Workshop on Principles and Practice of Semantic Web Reasoning, Budva, Montenegro (10th–11th June 2006)*, LNCS, 2006.

[16] J. Carlson and B. Lisper. An event detection algebra for reactive systems. In *Proc. ACM Int. Conf. On Embedded Software*, pages 147–154. ACM, 2004.

[17] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S.-K. Kim. Composite events for active databases: Semantics, contexts and detection. In *Proc. Int. Conf. on Very Large Data Bases*, pages 606–617. Morgan Kaufmann, 1994.

[18] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. A. Shah. Telegraphcq: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.

[19] A. J. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. M. White. Cayuga: A general purpose event monitoring system. In *CIDR*, pages 412–422, 2007.

[20] M. Eckert. Reactivity on the Web: Event Queries and Composite Event Detection in XChange. Master's thesis (Diplomarbeit), Institute for Informatics, University of Munich, 2005.

[21] M. Eckert. *Complex Event Processing with XChange^{EQ}: Language Design, Formal Semantics and Incremental Evaluation for Querying Events*. PhD thesis, Institute for Informatics, University of Munich, 2008.

[22] M. Eckert, F. Bry, S. Brodt, O. Poppe, and S. Hausmann. A CEP Babelfish: Languages for Complex Event Processing and Querying Surveyed. In S. Helmer, A. Poulovassilis, and F. Xhafa, editors, *Reasoning in Event-based Distributed Systems*, volume 347 of *Studies in Computational Intelligence*, chapter 3. Springer, 1 edition, 2011.

[23] M. Eckert, F. Bry, S. Brodt, O. Poppe, and S. Hausmann. Two Semantics for CEP, no Double Talk: Complex Event Relational Algebra (CERA) and its Application to XChange^{EQ}. In *Reasoning in Event-based Distributed Systems*, volume 347 of

processing and is implemented using the DBMS MonetDB.

Studies in Computational Intelligence, chapter 4. Springer, 1 edition, 2011.

[24] EsperTech Inc. Event stream intelligence: Esper & NEsper. `http://esper.codehaus.org`.

[25] H. Garcia-Molina, J. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2001.

[26] S. Gatziu and K. R. Dittrich. Events in an active object-oriented database system. In *Proc. Int. Workshop on Rules in Database Systems*, pages 23–39. Springer, 1993.

[27] S. Gatziu and K. R. Dittrich. Detecting composite events in active database systems using petri nets. In *Proc. Int. Workshop on Research Issues in Data Engineering: Active Database Systems*, pages 2–9. IEEE, 1994.

[28] N. H. Gehani, H. V. Jagadish, and O. Shmueli. Compose: A system for composite specification and detection. In *Advanced Database Systems*, LNCS, pages 3–15. Springer, 1993.

[29] A. Hinze and A. Voisard. A parameterized algebra for event notification services. In *Proc. Int. Symp. on Temporal Representation and Reasoning*, pages 61–65. IEEE, 2002.

[30] N. Jain, S. Mishra, A. Srinivasan, J. Gehrke, J. Widom, H. Balakrishnan, U. Çetintemel, M. Cherniack, R. Tibbetts, and S. B. Zdonik. Towards a streaming sql standard. *PVLDB*, 1(2):1379–1390, 2008.

[31] M. Kersten, E. Liarou, and R. Goncalves. A query language for a data refinery cell. In *Proc. Int. Workshop on Event-Driven Architecture, Processing and Systems*, 2007.

[32] J. Krämer and B. Seeger. Pipes: a public infrastructure for processing and exploring streams. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 925–926, New York, NY, USA, 2004. ACM.

[33] J. Krämer and B. Seeger. Semantics and implementation of continuous sliding window queries over data streams. *ACM Trans. Database Syst.*, 34:4:1–4:49, April 2009.

[34] E. Liarou, R. Goncalves, and S. Idreos. Exploiting the power of relational databases for efficient stream processing. In *Int. Conf. on Extending Database Technology (EDBT)*, volume 360, pages 323–334. ACM, 2009.

[35] M. Mansouri-Samani and M. Sloman. GEM: A generalized event monitoring language for distributed systems. *Distributed Systems Engineering*, 4(2):96–108, 1997.

[36] L. E. McKenzie, Jr. and R. T. Snodgrass. Evaluation of relational algebras incorporating the time dimension in databases. *ACM Comput. Surv.*, 23:501–543, December 1991.

[37] D. Moreto and M. Endler. Evaluating composite events using shared trees. *IEE Proceedings — Software*, 148(1):1–10, 2001.

[38] J. Morrell and S. D. Vidich. Complex Event Processing with Coral8. White Paper. `http://www.coral8.com/system/files/assets/pdf/Complex_Event_Processing_with_Coral8.pdf`, 2007.

[39] MS Analog Software. ruleCore(R) Complex Event Processing (CEP) Server. `http://www.rulecore.com`.

[40] G. Ozsoyoglu and R. T. Snodgrass. Temporal and real-time databases: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 7:513–532, August 1995.

[41] N. W. Paton, editor. *Active Rules in Database Systems*. Springer, 1998.

[42] O. Poppe. Qoncept – semantic query optimisation in complex event processing technologies. `http://www.pms.ifi.lmu.de/qoncept/`.

[43] F. Reiss, K. Stockinger, K. Wu, A. Shoshani, and J. M. Hellerstein. Enabling real-time querying of live and historical stream data. In *SSDBM*, page 28, 2007.

[44] C. Roncancio. Toward duration-based, constrained and dynamic event types. In *Proc. Int. Workshop on Active, Real-Time, and Temporal Database Systems*, volume 1553 of *LNCS*, pages 176–193. Springer, 1997.

[45] C. Sánchez, S. Sankaranarayanan, H. Sipma, T. Zhang, D. L. Dill, and Z. Manna. Event correlation: Language and semantics. In *Proc. Int. Conf. on Embedded Software*, volume 2855 of *LNCS*, pages 323–339. Springer, 2003.

[46] C. Sánchez, M. Slanina, H. B. Sipma, and Z. Manna. Expressive completeness of an event-pattern reactive programming language. In *Int. Conf. on Formal Techniques for Networked and Distributed Systems*, volume 3731 of *LNCS*, pages 529–532. Springer, 2005.

[47] K.-U. Schmidt, D. Anicic, and R. Stühmer. Event-driven reactivity: A survey and requirements analysis. In *SBPM2008: 3rd Int. Workshop on Semantic Business Process Management in Conjunction with the 5th European Semantic Web Conf. (ESWC'08)*. CEUR Workshop Proceedings, 2008.

[48] N. Seifert and M. Bettelini. Deliverable D3.1: Use cases requirements analysis and specification (main report), 2010.

[49] N. Seifert, M. Bettelini, and S. Rigert. Deliverable D3.2: Concrete use case models, 2011.

[50] M. Seiriö and M. Berndtsson. Design and implementation of an ECA rule markup language. In *Proc. Int. Conf. on Rules and Rule Markup Languages for the Semantic Web*, volume 3791 of *LNCS*, pages 98–112. Springer, 2005.

[51] G. Slivinskas, C. S. Jensen, S. Member, R. T. Snodgrass, and S. Member. A foundation for conventional and temporal query optimization addressing duplicates and ordering. *IEEE TKDE*, 13:2001, 2001.

[52] StreamBase Systems. *StreamSQL Guide*, 2011. `http://streambase.com/developers/docs/latest/streamsql/index.html`.

[53] E. Wu, Y. Diao, and S. Rizvi. High-performance Complex Event Processing over streams. In *Proc. Int. ACM Conf. on Management of Data (SIGMOD)*, pages 407–418. ACM, 2006.

[54] D. Zhu and A. S. Sethi. SEL, a new event pattern specification language for event correlation. In *Proc. Int. Conf. on Computer Communications and Networks*, pages 586–589. IEEE, 2001.