CEP, State of the Art

Jiří Kremser

3. května 2011

Outline

Introduction and Motivation

- understanding the system
- layers of an arbitrary system and emergency

Introduction and Motivation

- understanding the system
- layers of an arbitrary system and emergency
- derived abstract events

Introduction and Motivation

- understanding the system
- layers of an arbitrary system and emergency
- derived abstract events
- comparison to data stream mining

Introduction and Motivation

- understanding the system
- layers of an arbitrary system and emergency
- derived abstract events
- comparison to data stream mining
- comparison to time series

Introduction and Motivation

- understanding the system
- layers of an arbitrary system and emergency
- derived abstract events
- comparison to data stream mining
- comparison to time series

Applications

- log analysis
- intelligent buildings

Applications

- log analysis
- intelligent buildings
- smart grid

Applications

- log analysis
- intelligent buildings
- smart grid
- early detection of disease (Medical Ecosystem Personalized Event-Based Surveillance)

Applications

- log analysis
- intelligent buildings
- smart grid
- early detection of disease (Medical Ecosystem Personalized Event-Based Surveillance)
- stock market

Applications

- log analysis
- intelligent buildings
- smart grid
- early detection of disease (Medical Ecosystem Personalized Event-Based Surveillance)
- stock market
- root cause of failure

Applications

- log analysis
- intelligent buildings
- smart grid
- early detection of disease (Medical Ecosystem Personalized Event-Based Surveillance)
- stock market
- root cause of failure
- IDS

Applications

- log analysis
- intelligent buildings
- smart grid
- early detection of disease (Medical Ecosystem Personalized Event-Based Surveillance)
- stock market
- root cause of failure
- IDS
- RFID

Applications

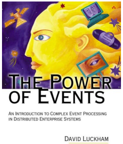
- log analysis
- intelligent buildings
- smart grid
- early detection of disease (Medical Ecosystem Personalized Event-Based Surveillance)
- stock market
- root cause of failure
- IDS
- RFID
- SOA

Applications

- log analysis
- intelligent buildings
- smart grid
- early detection of disease (Medical Ecosystem Personalized Event-Based Surveillance)
- stock market
- root cause of failure
- IDS
- RFID
- SOA

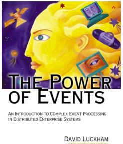
Brief History

- the term was coined by David Luckham (Stanford University)



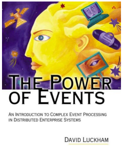
  in [Luc02]
- it is happening since 60's

Brief History

- the term was coined by David Luckham (Stanford University)


in [Luc02]
- it is happening since 60's
- mainly in late 90's

Brief History

- the term was coined by David Luckham (Stanford University)
  

  in [Luc02]
- it is happening since 60's
- mainly in late 90's

Event Informally

- “Event: Anything that happens, or is contemplated as happening.” – D. Luckham
- “Event (also event object, event message, event tuple): An object that represents, en- codes or records an event, generally for the purpose of computer processing.” Event Processing Glossary –[LS08]

Event Informally

- "Event: Anything that happens, or is contemplated as happening." – D. Luckham
- "Event (also event object, event message, event tuple): An object that represents, en- codes or records an event, generally for the purpose of computer processing." Event Processing Glossary –[LS08]
- "An event is a significant state change in the state of the universe. A significant state change is one for which an optimal response by the system is to take an action. An insignificant state change is one for which the system need take no action. An action may be registering information about the event in the enterprise's memory. Insignificant state changes are not registered in memory; they are never 'remembered.' " –[CCC07]

Event Informally

- "Event: Anything that happens, or is contemplated as happening." – D. Luckham
- "Event (also event object, event message, event tuple): An object that represents, en- codes or records an event, generally for the purpose of computer processing." Event Processing Glossary –[LS08]
- "An event is a significant state change in the state of the universe. A significant state change is one for which an optimal response by the system is to take an action. An insignificant state change is one for which the system need take no action. An action may be registering information about the event in the enterprise's memory. Insignificant state changes are not registered in memory; they are never 'remembered.' " –[CCC07]

Event Representation

- n-tuple with timestamp
- XML with pre-defined structure (SOAP)

Event Representation

- n-tuple with timestamp
- XML with pre-defined structure (SOAP)
- POJO (Java object)

Event Representation

- n-tuple with timestamp
- XML with pre-defined structure (SOAP)
- POJO (Java object)
- etc.

Event Representation

- n-tuple with timestamp
- XML with pre-defined structure (SOAP)
- POJO (Java object)
- etc.
- complex event (D. Luckman) = derived event = composite event (active DBs term)

Event Representation

- n-tuple with timestamp
- XML with pre-defined structure (SOAP)
- POJO (Java object)
- etc.
- complex event (D. Luckman) = derived event = composite event (active DBs term)

Time representation

- In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation 'happened before' is therefore only apartial order of events in the system. –Lamport
- A. Luckham's Cause-Time Axiom states that if event A caused event B in system S, then no clock in S gives B an earlier timestamp than it gives A.

Time representation

- In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation 'happened before' is therefore only apartial order of events in the system. –Lamport
- A. Luckham's Cause-Time Axiom states that if event A caused event B in system S, then no clock in S gives B an earlier timestamp than it gives A.
- interval

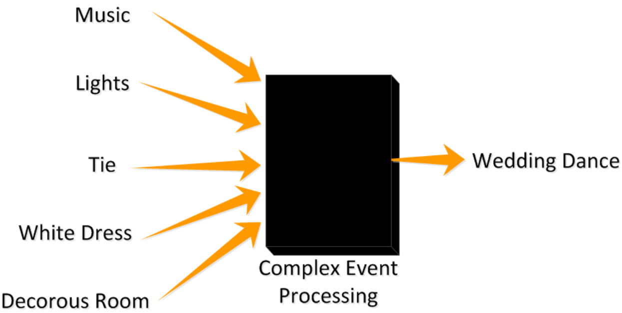
Time representation

- In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation ‘happened before’ is therefore only apartial order of events in the system. –Lamport
- A. Luckham’s Cause-Time Axiom states that if event A caused event B in system S, then no clock in S gives B an earlier timestamp than it gives A.
- interval
- time point

Time representation

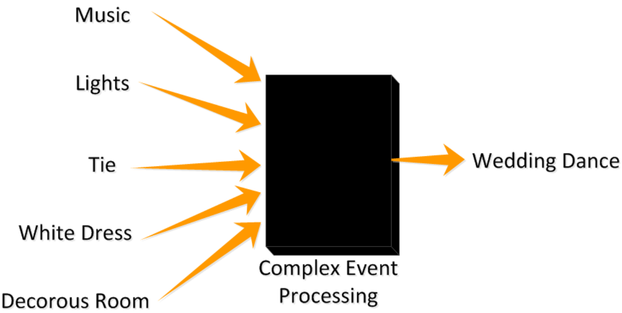
- In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation ‘happened before’ is therefore only apartial order of events in the system. –Lamport
- A. Luckham’s Cause-Time Axiom states that if event A caused event B in system S, then no clock in S gives B an earlier timestamp than it gives A.
- interval
- time point

Example



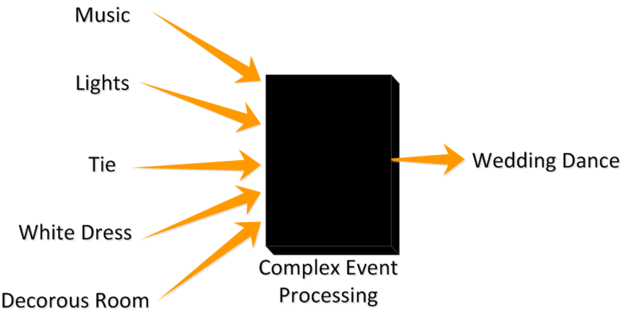
- querying the complex event

Example



- querying the complex event
- CE is a function of its constituent events

Example



- querying the complex event
- CE is a function of its constituent events

Language Classes

- domain specific languages
- composition-operator-based (stress on temporal relations)

Language Classes

- domain specific languages
- composition-operator-based (stress on temporal relations)
- data stream query languages (stress on data)

Language Classes

- domain specific languages
- composition-operator-based (stress on temporal relations)
- data stream query languages (stress on data)
- production rule languages

Language Classes

- domain specific languages
- composition-operator-based (stress on temporal relations)
- data stream query languages (stress on data)
- production rule languages
- hybrids

Language Classes

- domain specific languages
- composition-operator-based (stress on temporal relations)
- data stream query languages (stress on data)
- production rule languages
- hybrids

Language Features

- temporal aspects
- negation queries

Language Features

- temporal aspects
- negation queries
- aggregation

Language Features

- temporal aspects
- negation queries
- aggregation
- correlation

Language Features

- temporal aspects
- negation queries
- aggregation
- correlation
- consumption
  query $(A; B)$ against an event stream a_1, a_2, b_1, b_2 (a_1, a_2 of type A, b_1, b_2 of type B). The event a_1 will be used twice to generate two different complex events, one when b_1 is received and one when b_2 is received. When event b_1 is received, two different complex event are generated at the same time, one for a_1 and one for a_2.

Language Features

- temporal aspects
- negation queries
- aggregation
- correlation
- consumption
  query $(A;B)$ against an event stream a_1, a_2, b_1, b_2 (a_1, a_2 of type A, b_1, b_2 of type B). The event a_1 will be used twice to generate two different complex events, one when b_1 is received and one when b_2 is received. When event b_1 is received, two different complex event are generated at the same time, one for a_1 and one for a_2.

Composition-operator-based Lang.

- “event algebra”
- both time points and intervals

Composition-operator-based Lang.

- "event algebra"
- both time points and intervals
- support for counting

Composition-operator-based Lang.

- “event algebra”
- both time points and intervals
- support for counting

Example

$$((A \wedge B)_{1h}; C)_{2h}$$

Composition-operator-based Lang.

- "event algebra"
- both time points and intervals
- support for counting

Example

$$((A \wedge B)_{1h}; C)_{2h}$$

Composition-operator-based Lang. Pros and Cons

+ temporal aspects
+ negation queries (window problem)

Composition-operator-based Lang. Pros and Cons

+ temporal aspects
+ negation queries (window problem)
− aggregation (bad support, event data is neglected aspect)

Composition-operator-based Lang. Pros and Cons

+ temporal aspects
+ negation queries (window problem)
− aggregation (bad support, event data is neglected aspect)
− correlation

Composition-operator-based Lang. Pros and Cons

+ temporal aspects
+ negation queries (window problem)
− aggregation (bad support, event data is neglected aspect)
− correlation
+ Consumption

Composition-operator-based Lang. Pros and Cons

+ temporal aspects
+ negation queries (window problem)
− aggregation (bad support, event data is neglected aspect)
− correlation
+ Consumption

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]
- Snoop [CKAK94]

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]
- Snoop [CKAK94]
- GEM [MSS97]

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]
- Snoop [CKAK94]
- GEM [MSS97]
- SEL [ZS01]

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]
- Snoop [CKAK94]
- GEM [MSS97]
- SEL [ZS01]
- CEDR [BC06]

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]
- Snoop [CKAK94]
- GEM [MSS97]
- SEL [ZS01]
- CEDR [BC06]
- ruleCore [SB05, MS]

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]
- Snoop [CKAK94]
- GEM [MSS97]
- SEL [ZS01]
- CEDR [BC06]
- ruleCore [SB05, MS]
- SASE Event Language [WDR06]

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]
- Snoop [CKAK94]
- GEM [MSS97]
- SEL [ZS01]
- CEDR [BC06]
- ruleCore [SB05, MS]
- SASE Event Language [WDR06]
- XChange [BEP06b]

Composition-operator-based Lang. Instances

- COMPOSE language of the Ode active database [GJS92b, GJS92a, GJS93]
- SAMOS active database [GD93, GD94]
- Snoop [CKAK94]
- GEM [MSS97]
- SEL [ZS01]
- CEDR [BC06]
- ruleCore [SB05, MS]
- SASE Event Language [WDR06]
- XChange [BEP06b]

Formal Semantics

- unambiguous semantics of sequence operator ";"
- informally 7 cases of "before"

Formal Semantics

- unambiguous semantics of sequence operator ";"
- informally 7 cases of "before"
- $O(q, t)$ which is true when there is an answer to the query q with occurrence time t [GA02, AC06]

Formal Semantics

- unambiguous semantics of sequence operator ";"
- informally 7 cases of "before"
- $O(q,t)$ which is true when there is an answer to the query q with occurrence time t [GA02, AC06]

Example

$$O((E_1;E_2),[t_1,t_2]) \Leftrightarrow \exists t \exists t'.t_1 \leq t < t' \leq t_2 \wedge O(E_1,[t_1,t]) \wedge O(E_2,[t',t_2])$$

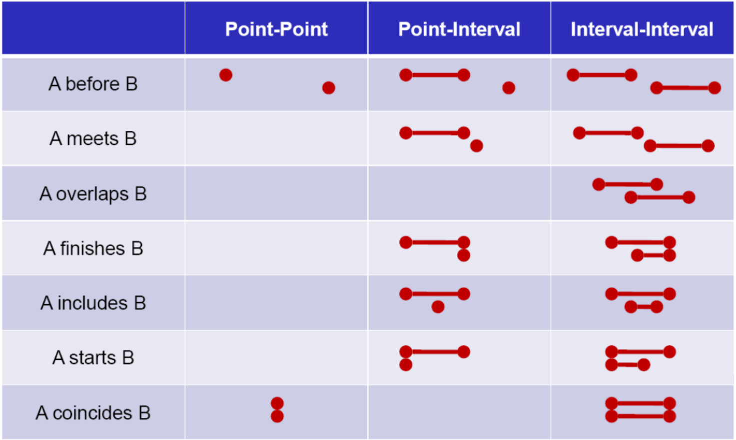
Formal Semantics

- unambiguous semantics of sequence operator ";"
- informally 7 cases of "before"
- $O(q, t)$ which is true when there is an answer to the query q with occurrence time t [GA02, AC06]

Example

$$O((E_1; E_2), [t_1, t_2]) \Leftrightarrow \exists t \exists t'.t_1 \leq t < t' \leq t_2 \wedge O(E_1, [t_1, t]) \wedge O(E_2, [t', t_2])$$

Informal “before”



Data Stream Query Lang.

- time represented only as a time point τ from discrete domain
- event = data tuple with timestamp

Data Stream Query Lang.

- time represented only as a time point τ from discrete domain
- event = data tuple with timestamp
- stream-to-relation operators (e.g., sliding/batching window)
  $S[Range\ 2\ Hours]$, $S[Unbounded]$,
  $S[Range\ 24\ Hours\ Slide\ 1\ Hour]$, $S[Rows\ N]$

Data Stream Query Lang.

- time represented only as a time point τ from discrete domain
- event = data tuple with timestamp
- stream-to-relation operators (e.g., sliding/batching window) $S[Range\ 2\ Hours]$, $S[Unbounded]$, $S[Range\ 24\ Hours\ Slide\ 1\ Hour]$, $S[Rows\ N]$
- relation-to-relation operators (subset of SQL (no nested queries allowed))

Data Stream Query Lang.

- time represented only as a time point τ from discrete domain
- event = data tuple with timestamp
- stream-to-relation operators (e.g., sliding/batching window) $S[Range\ 2\ Hours]$, $S[Unbounded]$, $S[Range\ 24\ Hours\ Slide\ 1\ Hour]$, $S[Rows\ N]$
- relation-to-relation operators (subset of SQL (no nested queries allowed))
- relation-to-stream operators Istream, Dstream, and Rstream

Data Stream Query Lang.

- time represented only as a time point τ from discrete domain
- event = data tuple with timestamp
- stream-to-relation operators (e.g., sliding/batching window)
  $S[Range\ 2\ Hours]$, $S[Unbounded]$,
  $S[Range\ 24\ Hours\ Slide\ 1\ Hour]$, $S[Rows\ N]$
- relation-to-relation operators (subset of SQL (no nested queries allowed))
- relation-to-stream operators Istream, Dstream, and Rstream

Data Stream Query Lang. Example

Example

SELECT Istream (payment, count(id))
FROM O [**Range** 24 **Hours**]
GROUP BY O.payment

Number of daily payments grouped by payment type.

Data Stream Query Lang. Pros and Cons

- temporal aspects (poorly supported, notion of window)
- negation queries (window problem, but possible)

Data Stream Query Lang. Pros and Cons

- – temporal aspects (poorly supported, notion of window)
- – negation queries (window problem, but possible)
- + aggregation (very good support)

Data Stream Query Lang. Pros and Cons

- – temporal aspects (poorly supported, notion of window)
- – negation queries (window problem, but possible)
- + aggregation (very good support)
- – correlation

Data Stream Query Lang. Pros and Cons

- − temporal aspects (poorly supported, notion of window)
- − negation queries (window problem, but possible)
- + aggregation (very good support)
- − correlation
- − consumption (only one implementation [KLG07] similar to petri nets)

Data Stream Query Lang. Pros and Cons

- – temporal aspects (poorly supported, notion of window)
- – negation queries (window problem, but possible)
- + aggregation (very good support)
- – correlation
- – consumption (only one implementation [KLG07] similar to petri nets)

Data Stream Query Lang. Instances

- Continuous Query Language (CQL) used in STREAM systems [ABW06]
- EPL (Esper [Esp], Oracle Fusion Middleware [Ora], BEA WebLogic® Event Server 2.0)

Data Stream Query Lang. Instances

- Continuous Query Language (CQL) used in STREAM systems [ABW06]
- EPL (Esper [Esp], Oracle Fusion Middleware [Ora], BEA WebLogic® Event Server 2.0)
- Coral8 [MV07]

Data Stream Query Lang. Instances

- Continuous Query Language (CQL) used in STREAM systems [ABW06]
- EPL (Esper [Esp], Oracle Fusion Middleware [Ora], BEA WebLogic® Event Server 2.0)
- Coral8 [MV07]
- StreamSQL (StreamBase) Simple support of causality

Data Stream Query Lang. Instances

- Continuous Query Language (CQL) used in STREAM systems [ABW06]
- EPL (Esper [Esp], Oracle Fusion Middleware [Ora], BEA WebLogic® Event Server 2.0)
- Coral8 [MV07]
- StreamSQL (StreamBase) Simple support of causality

Production Rule Languages

- hosting programming language
- WHEN condition THEN action

Production Rule Languages

- hosting programming language
- WHEN condition THEN action
- Rete algorithm for pattern matching

Production Rule Languages

- hosting programming language
- WHEN condition THEN action
- Rete algorithm for pattern matching
- not suitable for Event processing networks

Production Rule Languages

- hosting programming language
- WHEN condition THEN action
- Rete algorithm for pattern matching
- not suitable for Event processing networks
- relation-to-stream operators Istream, Dstream, and Rstream

Production Rule Languages

- hosting programming language
- WHEN condition THEN action
- Rete algorithm for pattern matching
- not suitable for Event processing networks
- relation-to-stream operators Istream, Dstream, and Rstream
- OPS5 [For81]

Production Rule Languages

- hosting programming language
- WHEN condition THEN action
- Rete algorithm for pattern matching
- not suitable for Event processing networks
- relation-to-stream operators Istream, Dstream, and Rstream
- OPS5 [For81]
- Drools (also called JBoss Rules) [JBo]

Production Rule Languages

- hosting programming language
- WHEN condition THEN action
- Rete algorithm for pattern matching
- not suitable for Event processing networks
- relation-to-stream operators Istream, Dstream, and Rstream
- OPS5 [For81]
- Drools (also called JBoss Rules) [JBo]
- RuleML standard

Production Rule Languages

- hosting programming language
- WHEN condition THEN action
- Rete algorithm for pattern matching
- not suitable for Event processing networks
- relation-to-stream operators Istream, Dstream, and Rstream
- OPS5 [For81]
- Drools (also called JBoss Rules) [JBo]
- RuleML standard

Production Rule Languages Pros and Cons

– temporal aspects (no built-in support, only way is to treat timestamp as regular data attribute)
– negation queries (low-level implementation)

Production Rule Languages Pros and Cons

- temporal aspects (no built-in support, only way is to treat timestamp as regular data attribute)
- negation queries (low-level implementation)
- aggregation (low-level implementation)

Production Rule Languages Pros and Cons

- temporal aspects (no built-in support, only way is to treat timestamp as regular data attribute)
- negation queries (low-level implementation)
- aggregation (low-level implementation)
- correlation (low-level implementation)

Production Rule Languages Pros and Cons

- temporal aspects (no built-in support, only way is to treat timestamp as regular data attribute)
- negation queries (low-level implementation)
- aggregation (low-level implementation)
- correlation (low-level implementation)
- consumption (low-level implementation)

Production Rule Languages Pros and Cons

- temporal aspects (no built-in support, only way is to treat timestamp as regular data attribute)
- negation queries (low-level implementation)
- aggregation (low-level implementation)
- correlation (low-level implementation)
- consumption (low-level implementation)

EPA EPN EDA BI

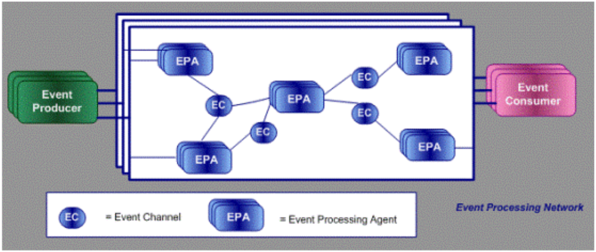
- EPA $\sim$ process representing query
- EPN

EPA EPN EDA BI

- EPA $\sim$ process representing query
- EPN
- EDA

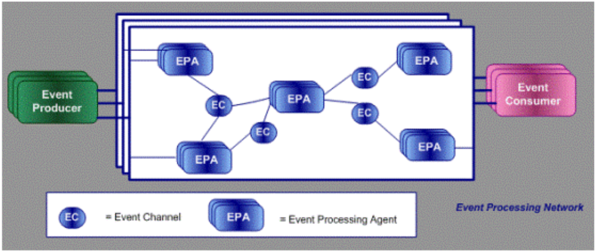
EPA EPN EDA BI

- EPA $\sim$ process representing query
- EPN
- EDA
- BI

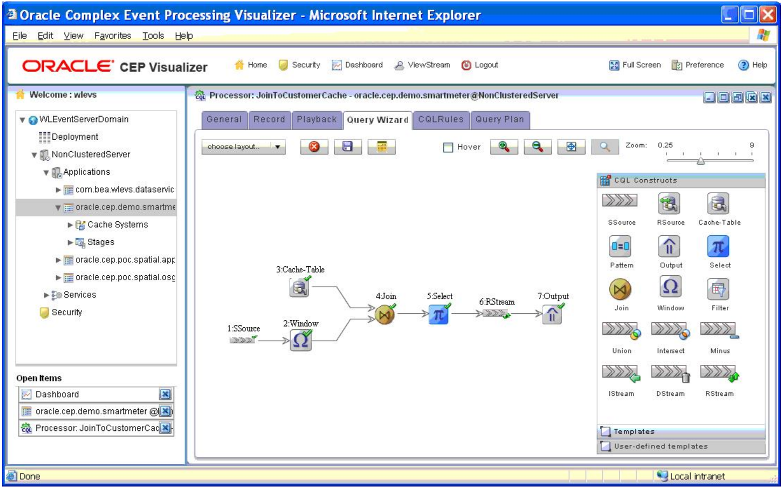


EPA EPN EDA BI

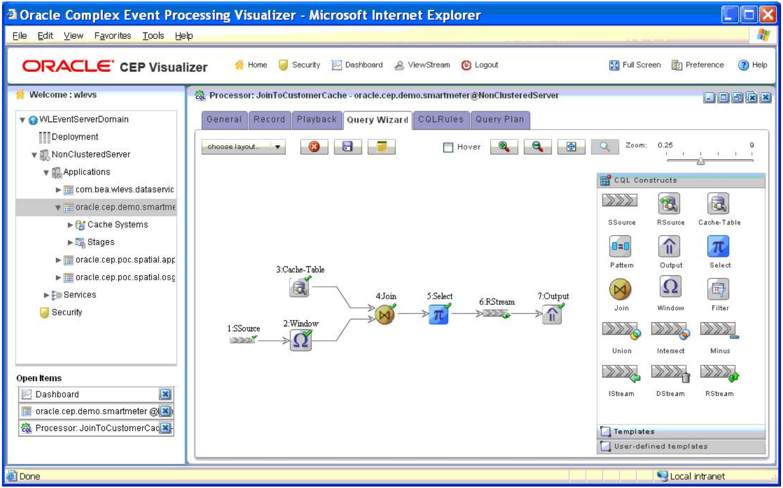
- EPA $\sim$ process representing query
- EPN
- EDA
- BI



Example of Oracle Solution



Example of Oracle Solution



Future Work

- unexpected behaviour detection
- causality between events [LMH02]

Future Work

- unexpected behaviour detection
- causality between events [LMH02]
- spatial relations in query language

Future Work

- unexpected behaviour detection
- causality between events [LMH02]
- spatial relations in query language
- uncertainity (errors in measurements $\Rightarrow$ more sensors and then average or outlier detections)

Future Work

- unexpected behaviour detection
- causality between events [LMH02]
- spatial relations in query language
- uncertainity (errors in measurements $\Rightarrow$ more sensors and then average or outlier detections)

Event Mining

- a priori queries $\sim$ top-down design
- unknown patterns? $\sim$ outlier detection

Event Mining

- a priori queries $\sim$ top-down design
- unknown patterns? $\sim$ outlier detection
- ML-outlier = isolated (strange) data point X CEP-outlier = strange combination of events

Event Mining

- a priori queries $\sim$ top-down design
- unknown patterns? $\sim$ outlier detection
- ML-outlier = isolated (strange) data point X CEP-outlier = strange combination of events
- online clustering for identifying related events

Event Mining

- a priori queries $\sim$ top-down design
- unknown patterns? $\sim$ outlier detection
- ML-outlier = isolated (strange) data point X CEP-outlier = strange combination of events
- online clustering for identifying related events
- associative rules may help to causality detection

Event Mining

- a priori queries $\sim$ top-down design
- unknown patterns? $\sim$ outlier detection
- ML-outlier $=$ isolated (strange) data point X CEP-outlier $=$ strange combination of events
- online clustering for identifying related events
- associative rules may help to causality detection

Conclusion

- promising technology
- language classes

Conclusion

- promising technology
- language classes
- from top-down to bottom-up approach

Conclusion

- promising technology
- language classes
- from top-down to bottom-up approach
- incorporate ML algorithms

Conclusion

- promising technology
- language classes
- from top-down to bottom-up approach
- incorporate ML algorithms

[Luc02] David C. Luckham. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems.* Addison-Wesley, 2002.

[LS08] David Luckham and Roy Schulte. *Event processing glossary.* http://complexevents.com/?p=361, May 2008.

[LMH02] Liang, F., Ma, S., and Hellerstein. *Discovering Fully Dependent Patterns.* SIAMDM, 2002.

[CCC07] K. Mani Chandy, Michel Charpentier, and Agostino Capponi. *Towards a theory of events.* In Proc. Int. Conf. on Distributed Event-Based Systems, pages 180–187. ACM, 2007.

[GJS92b] Narain H. Gehani, H.V. Jagadish, and Oded Shmueli. *Event specification in an active object-oriented database.* In Proc. Int. ACM Conf. on Management of Data (SIGMOD), pages 81–90. ACM, 1992.

[GJS92a] Narain H. Gehani, H. V. Jagadish, and Oded Shmueli. *Composite event specification in active databases: Model & implementation.* In Proc. Int. Conf. on Very Large Data Bases, pages 327–338. Morgan Kaufmann, 1992.

[GJS93] Narain H. Gehani, H. V. Jagadish, and Oded Shmueli. *Compose: A system for composite specification and detection.* In Advanced Database Systems, volume 759 of LNCS, pages 3–15. Springer, 1993.

[GD94] Stella Gatziu and Klaus 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.

[GD93] Stella Gatziu and Klaus R. Dittrich. *Events in an active object-oriented database system.* In Proc. Int. Workshop on Rules in Database Systems, pages 23–39. Springer, 1993.

[CKAK94] Sharma Chakravarthy, V. Krishnaprasad, Eman 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.

[MSS97] Masoud Mansouri-Samani and Morris Sloman. *GEM: A generalized event monitoring language for distributed systems.* Distributed Systems Engineering, 4(2):96–108, 1997.

[ZS01] Dong Zhu and Adarshpal 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.

[BC06] Roger S. Barga and Hillary 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.

[SB05] Marco Seirio and Mikael 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.

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

[WDR06] Eugene Wu, Yanlei Diao, and Shariq Rizvi. *High-performance Complex Event Processing over streams.* In Proc. Int. ACM Conf. on Management of Data (SIGMOD), pages 407–418. ACM, 2006.

[BEP06b] Francois Bry, Michael Eckert, and Paula-Lavinia Patranjan. *Reactivity on the Web: Paradigms and applications of the language XChange.* J. of Web Engineering, 5(1):3 – 24, 2006.

[GA02] Antony Galton and Juan Carlos Augusto. *Two approaches to event definition.* In Proc. Int. Conf. on Database and Expert Systems Applications, volume 2453 of LNCS, pages 547–556. Springer, 2002.

[AC06] Raman Adaikkalavan and Sharma Chakravarthy. *SnoopIB: Interval-based event specification and detection for active databases.* Data and Knowledge Engineering, 1(59):139–165, 2006.

[ABW06] Arvind Arasu, Shivnath Babu, and Jennifer Widom. *The CQL continuous query language: Semantic foundations and query execution.* The VLDB Journal, 15(2):121 – 142, 2006.

[Esp] EsperTech Inc. *Event stream intelligence: Esper NEsper.* http://esper.codehaus.org.

[MV07] John Morrell and Stevan D. Vidich. *Complex Event Processing with Coral8.* White Paper., 2007.

[KLG07] Martin Kersten, Erietta Liarou, and Romulo Goncalves. *A query language for a data refinery cell.* In Proc. Int. Workshop on Event-Driven Architecture, Processing and Systems, 2007.

[For81] Charles Forgy. *OPS5 user's manual.* Technical Report CMU-CS-81-135, Carnegie Mellon University, 1981.

[JBo] *JBoss.org. Drools.* http://www.jboss.org/drools.

Thank you for your attention

Q&A