An Introduction to Complex Event Processing

Jarle Søberg, INF5100
Overview

• Motivation
• Events
• Languages
• Hierarchies
Motivation I

• DSMS technology
  • A data stream of tuples
    • A tuple consists of attributes
      • Values
      • Timestamps
    • In-memory real-time processing
  • Perform projections and aggregations
    • Give me all tuples from company X
    • Give me the average packet size in the network from the last five minutes
    • Tell me if the temperature is lower than 15 degrees
  • Uses windowing for blocking operators
  • Based on SQL
Motivation II

• Beyond projections and aggregation: We want to know more about the data stream!
  • When does patterns occur?
    • Tell me when the stock prize from company X increases by 10% for ten minutes before decreasing
  • Order of consecutive events
    • Only tell me when the goods have been taken out of the shop without being purchased in the counter
    • How long does it take from a person falls until he gets help?
  • Concurrency
    • Tell me when the cooker is on while the person has left the home
Complex event processing (CEP)

• Our interpretation:

• CEP is still not a well defined application domain

© Jarle Søberg, 2010
Example Applications

• Algorithmic stock trading
• Credit card fraud detection
• Business activity monitoring
• Security monitoring
• Automated home care
• …
Complex Events

• An event is *something that happens* (*generic definition*)
  • An object with attributes, e.g. timestamps
  • Signifies an activity
  • Might be relative to other events

• A complex event is an aggregation of related events
  • Consecutive
    • Orders events: A happened before B
    • An event can have one, two, or several timestamps
  • Concurrent
    • For instance, A happens while B happens
Ordered Events

• A simple approach is to model order by using a DAG (directed acyclic graph)

Bank transaction:

A network client (with real-time requirements):
Event Adaption

• A CEP system should manage to observe and act on the data stream
Patterns and Rules

• A template that matches certain sets of events
  • The sets you want to find

• Shows consecutive dependencies, timing, data parameters, and context
  • Example: All orders from customers at the regular price that have led to the customer requesting a reduced price in response to a discount announcement

• Event pattern rules can be reactive
  • A trigger; the event pattern
  • An action; the event that is created when the trigger matches

• Sequential rules
• Parallel rules

© Jarle Søberg, 2010
A Declarative Approach

• Just as in DSMSs
  • Describe what should happen
  • Perform aggregations on the data
• Describe what we want, not how to get it
• SQL-like syntax is common
  SELECT attributes
  FROM event streams
  WHERE attribute values…

• In addition, the systems have syntax for describing patterns
  • E.g. in Esper
## Example in STRAW-EPL

<table>
<thead>
<tr>
<th>Element</th>
<th>Declarations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>Node N1, Node N2, Data D, Bit B, Time T1, Time T2, Time T3, Time T4;</td>
</tr>
<tr>
<td>Event types</td>
<td>Send(Node N1, Node N2, Data D, Bit B, Time T1),</td>
</tr>
<tr>
<td></td>
<td>Receive(Node N1, Node N2, Data D, Bit B, Time T1),</td>
</tr>
<tr>
<td></td>
<td>Ack(Node N1, Node N2, Bit B, Time T1),</td>
</tr>
<tr>
<td></td>
<td>RecAck(Node N1, Node N2, Bit B, Time T1),</td>
</tr>
<tr>
<td></td>
<td>Warning(Node N1, Node N2, Time T1, Time T2);</td>
</tr>
<tr>
<td>Relational operators</td>
<td>→ (order)</td>
</tr>
<tr>
<td>Pattern</td>
<td>Send(N1, N2, D, B, T1) →Receive(N2, N1, D,B, T2) →Ack(N2, N1, B, T3) →RecAck(N1,N2,B,T4);</td>
</tr>
<tr>
<td>Context test</td>
<td>T4 – T1 ≥1 hour</td>
</tr>
<tr>
<td>Action</td>
<td><strong>Create</strong> Warning(N1,N2,T1,T4)</td>
</tr>
</tbody>
</table>
Example in STRAW-EPL

Alternative events
Patterns and Constraints

• The *never* constraint contains
  • A temporal operator *never*
  • A pattern

• Checks if a pattern never happens in the system
  • Identify inconsistent behavior
  • Example:
    • *Never(Confirm(Price, T1) and Deny(Price, T2)) with context test T1 ≤ T2*
Event Hierarchies

• A complex event is an aggregation of *members*
  • Events at different times
  • Events at widely separate components of the system

• Examples
  • A CPU instruction, Add(X, Y)
  • Customer C’s order has been fulfilled
  • An intrusion has been attempted on subnet N
  • Message M has been routed to client C based upon C’s message content filter
  • Message M is routed to client C depending upon the context it was received
Event Hierarchies II

- A sequence of levels of activities
  - Abstractions
  - Each level is a complex event, i.e., an aggregation of lower levels
- Define a set of event aggregation rules for each level:
CEP Systems

- Research:
  - YFilter++
  - Cayuga
  - SASE+
  - CommonSens
  - ...
- Open source:
  - Esper
- Commercial:
  - Aleri
  - StreamBase
  - System S
  - And more (see complexevents.com)!

© Jarle Søberg, 2010