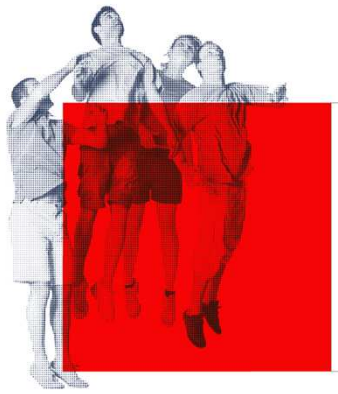
Analyzing SQL Statements with Complex Event Processing (CEP)



Henrik Dittmar
Senior Consultant
henrik.dittmar@trivadis.com

Hamburg, 13.09.2011

trivadis
makes IT easier.

Agenda

- Introduction
  - The History and Objectives of CEP
  - Research Questions
- A Testapplication for Experiments
  - The Class Model
  - Components
    - The NESPER CEP Engine
    - ZQL the SQL Parser
    - ZQL Enhancements
  - Configuration of the CEP Engine
    - Event Structures
    - Events and Actions
    - How to Change the Ruleset
- Analysis and Experiences
- Conclusion

Introduction– The History of CEP

- How does a database firewall work? 
- How to create interfaces without changing application code?
- How to fullfill requirements for legal reporting e.g. in the financial service business?
  - Since aprox 6 years a design/technology called „Complex Event Processing“ (CEP) is used to cope with challenges like this.
  - CEP is about nearly real time analysis of data streams with the help of a rule set. If a rule returns true or a computes a specific values out of the combination of data streams an activitie can be issued.

Introduction– The History of CEP

- Why not use well known database designs with business logic in IF-THEN-ELSE patterns?

- *"By 2011, a new generation of application platforms stemming from the convergence of extreme transaction processing (XTP) technologies will supersede Java EE and .NET as the platform of choice for large-scale, business-critical operational applications."* , Gartner, July 2007

- *“Classical databases or distributed caches are passive data structures that create frozen and slow assets which require explicit querying to make sense of. By turning to ESP/CEP, data flow in real-time through queries - ensuring immediate reactivity, and reduced burden of custom development to make sense of what's going on.”* EsperTech

Introduction – The History of CEP

- CEP can be described as a combination of a time based state machine combined with a rule engine and a trigger for activities.
- *This is a sexy approach because of the descriptive character of rules. Comples rulesets gain flexibility, because rules can be changed online and are totally isolated from other pices of code.*

Definition: Complex Event Processing (CEP)
Complex Event Processing is a Software composition pattern that enables the validation of events with the help of a time driven FIFO Message Stack. Event can be regarded as messages of a certain known structure. For the evaluation or processing of event the pattern asks for a rule engine.

Introduction – The History of CEP

- Event driven „state machines“ are known in informatics for 50 years now.

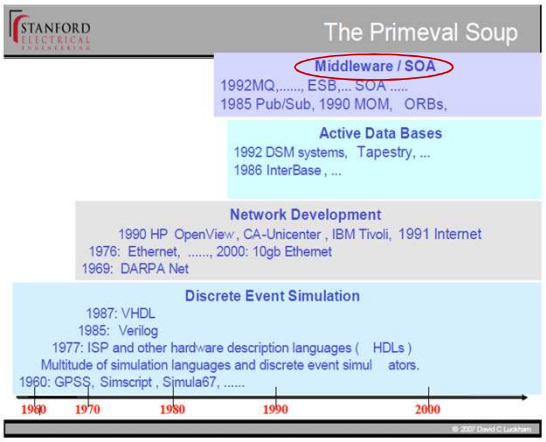


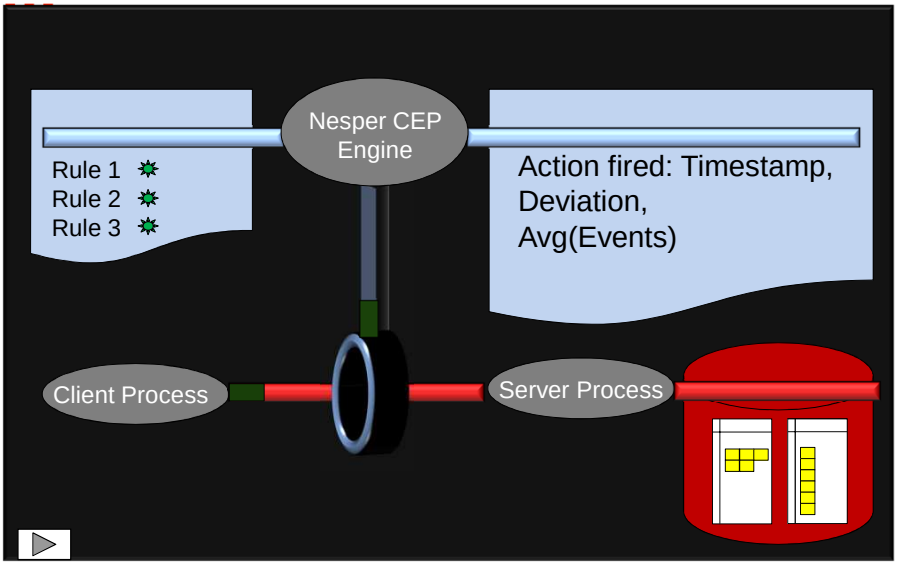
Figure1: Four Major Areas of CEP [David Luckham]

Introduction– The History of CEP

- Now adays CEP has reached the Middleware/Businesslogic level of IT solutions
  - Examples
    - Oracle Database Firewall
    - Individual Software for Financial Risk Control

- **But there may be room for improvement**

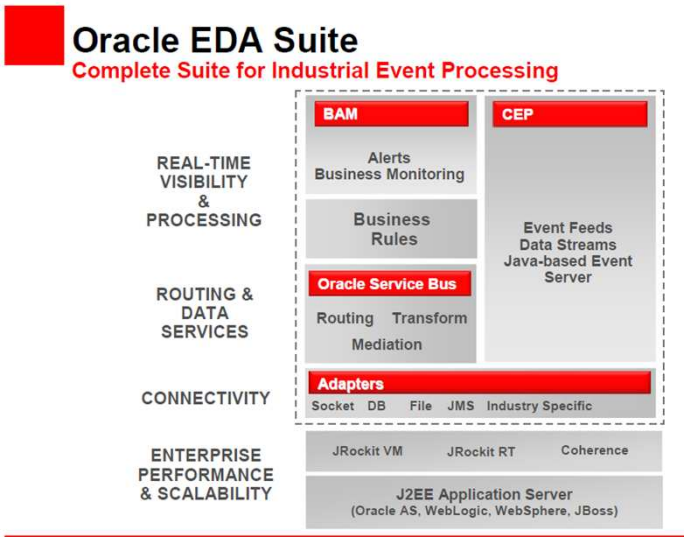
Introduction – The CEP Architecture



Introduction – Research Questions

- To judge the typical missions for CEP design patterns, the stability and flexibility of the technology and their tools the following research questions have been choosen:
  - 1) Which kind of information can be retrived from a stream of SQL Statements (Single Session)?
  - 2) Which steps are necessary to change a rule set and which constraints have to be taken into account?
  - 3) How to distinguish CEP from other rule driven software architectures?

The Oracle Portfolio



Oracle Documentation Links

- Oracle CEP

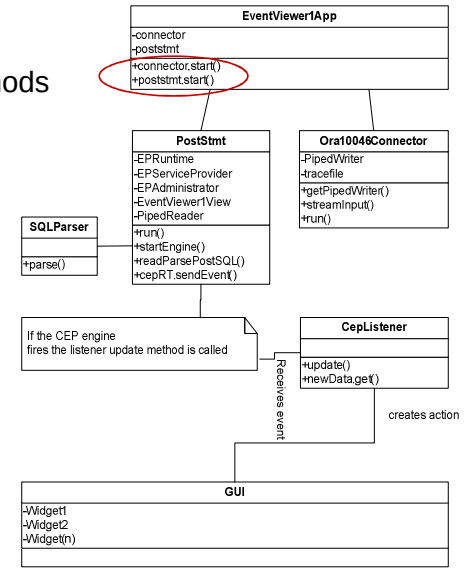
  http://download.oracle.com/docs/cd/E13157_01/wlevs/docs30/get_started/install.html
  App Development Guide
  http://download.oracle.com/docs/cd/E13157_01/wlevs/docs30/create_apps/index.html
  Oracle EPL Guide
  http://download.oracle.com/docs/cd/E13213_01/wlevs/docs30/pdf/epl_guide.pdf

Agenda

- Introduction
  - The History and Objectives of CEP
  - Research Questions
- A Testapplication For Experiments
  - The Class Model
  - Components
    - The NESPER CEP Engine
    - ZQL the SQL Parser
    - ZQL Enhancements
  - Configuration of the CEP Engine
    - Event Structures
    - Events and Actions
    - How to Change the Ruleset
- Analysis and Experiences
- Conclusion

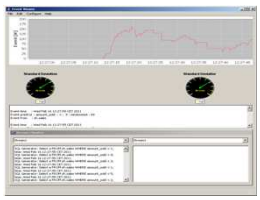
A Testapplication

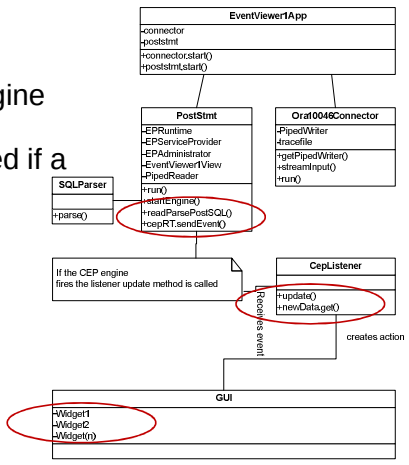
- Class model and calling methods
- Start postStmt() and connecetor() as seperate threads
- Threads are comunicting via a Pipe



A Testapplication

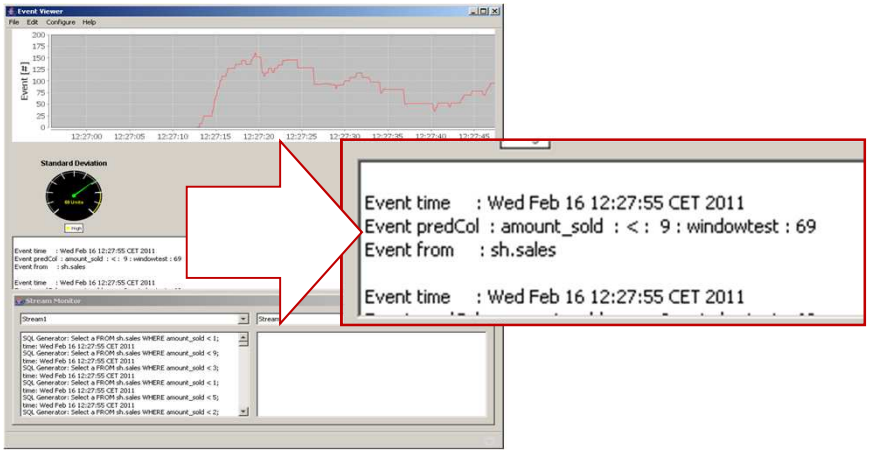
- Class model and calling methods
- Read every stmt from the pipe
- sendEvent() to send it to the engine
- Listener.update() methode is fired if a rule returns TRUE
- The GUI shows recent events





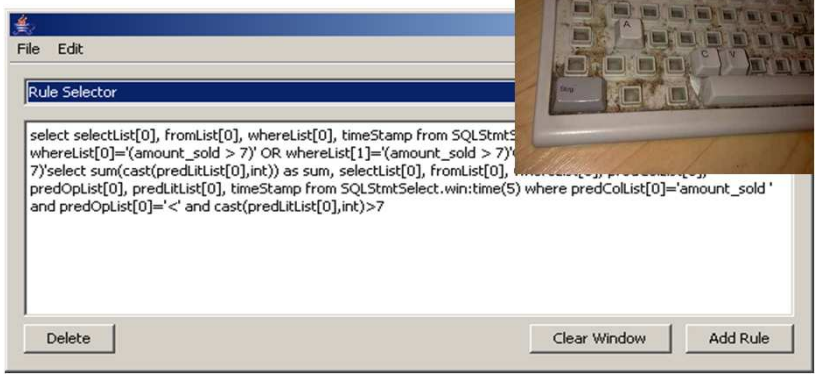
A Testapplication

- The GUI



A Testapplication

- A very simple rule editor



- At the moment we are missing a graphical freeware rule editor

Agenda

- Introduction
  - The History and Objectives of CEP
  - Research Questions
- A Testapplication for Experiments
  - The Class Model
  - Components
    - The NESPER CEP Engine
    - ZQL the SQL Parser
    - ZQL Enhancements
  - Configuration of the CEP Engine
    - Event Structures
    - Events and Actions
    - How to Change the Ruleset
- Analysis and Experiences
- Evaluation Results

Components

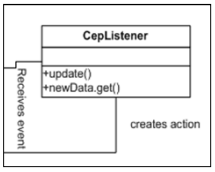
- Which components are neccessary to create a simple CEP solution for SQL?
  - A CEP engine
  - A SQL Parser
  - Some surounding code pieces
- The CEP Engine NESPER from EsperTech was choosen because:
  - It‘s freeware
  - Rules are coded in the language EPL, which is similar to SQL
- ZQL was choosen because:
  - It‘s freeware
  - It provides a collection of very easy to use methods to parse SQL

Components

- Which components are neccessary to create a simple CEP solution for SQL?
  - A CEP engine
  - A SQL Parser
  - Some surounding code pieces
- ZQL is not able to pares the details of a WHERE clause. Thefore it was neccesary to implement some enhancements to reach out for predicates, operators, literals and bind variables within.

Components

- The Nesper CEP Engine
  - The engine is provided from NesperTech. A company founded 2004 near NY.
  - The engine is freeware. Surrounding tools have to be licenced:
    - HA Enviroment
    - Rule Editor
    - etc.
- The very heart of the CEP Engine is:
  - The rule language
  - The rule engine
  - The event handler (called listener)



The Nesper CEP Engine - Rules

- A simple rule, noted in EPL:

```
SELECT *, timeStamp
FROM       SQLStmtSelect
WHERE
          whereList[0]=(amount_sold > 7)
          OR whereList[1]=(cust_id = 100)
          OR whereList[2]=(cust_name = 'King')
```

In three clauses SELECT, FROM, WHERE the return values (*, timestamp) can be discovered, as well as the data source SQLStmtSelect and filter predicates. The data source represents the stream of SQL statements flowing by.

So ist all like quering a database? No!

The Nesper CEP Engine - Rules

- The CEP specific difference hides behind the data stream. It's structure is called „Event Type" in the NESPER semantic.
- The SELECT clause allows single row functions
- The FROM clause defines time windows to aggregate events
- … and combinations of both clauses to use prebuild or self created aggregate functions
  - avg([all|distinct] *expression*)
  - stddev([all|distinct] *expression*)
  - median([all|distinct] *expression*)
  - Min, max, sum, count, etc.
- WHERE clause for filtering

The Nesper CEP Engine - Rules

- A more complex rule, noted in EPL:

```
SELECT   sum(cast(predLitList[0],int)) as sum,
         selectList[0], fromList[0], whereList[0],
         predColList[0], predOpList[0], predLitList[0], timeStamp
FROM
         SQLStmtSelect.win:time(5)
WHERE
          predColList[0]='amount_sold ' and predOpList[0]='<' and cast(predLitList[0],int)>7
```

The rule defines a time window. The predicates are evaluating the mean average over the last 5 seconds.

- Additional clauses and key words to reinsert events in an event or to modify data within an event (INSERT) are neglated in this introduction

The Nesper CEP Engine – Rule Propagation

- Create rules for the engine

```
EPStatement cepStatement1 = cepAdm.createEPL( [String] )
```

- Define the Event Structure

```
cepConfig.addEventType("SQLStmtSelect",SQLStreamSelect.class.getName());
```

Agenda

- Introduction
  - The History and Objectives of CEP
  - Research Questions
- A Testapplication for Experiments
  - The Class Model
  - Components
    - The NESPER CEP Engine
    - ZQL the SQL Parser
    - ZQL Enhancements
  - Configuration of the CEP Engine
    - Event Structures
    - Events and Actions
    - How to Change the Ruleset
- Analysis and Experiences
- Conclusion

Configuration Of The CEP Engine

- To configure and start the engine the follwing steps are neccessary

1. Start the CEP Engine:
   Configuration cepConfig = new Configuration();
2. Definition of the Event Typs:
   cepConfig.addEventType("SQL", SQLStreamSelect.class.getName());
3. Provide a rule set to the engine:
   EPStatement cepStatement1 = cepAdm.createEPL("select […]“);

Configuration Of The CEP Engine

- The event type
  - The structure of an event is defind by one class per event

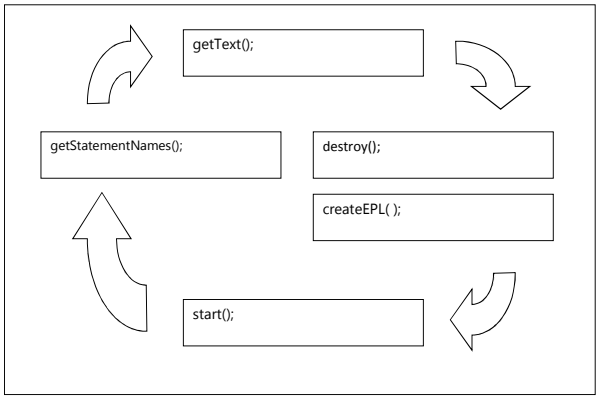
```
public class SQLStreamSelect {
      String[]    selectList             = new String[10];
      String[]    fromList               = new String[10];
      String[]    whereList              = new String[10];
      String[]    predColList            = new String[256];
      String[]    predOpList             = new String[256];
      String[]    predLitList            = new String[256];
      String[]    predLogOpList          = new String[256];
      // The predicate list delivers:
      // pPredList[i][0]:= column Name pPredList[i][1] := Operator
pPredList[i][2]   := Literal
pPredList[i][3]   := Logical operator

String[][]        predList               = new String[256][4];
      Date                               timeStamp;
```


Configuration Of The CEP Engine

- Has the rule engine validated a rule to be true for a certain event, the listener method is called.
- The listener acts as the API for actions which should start for a certain event.

Configuration Of The CEP Engine

- The rule management cycle
  - Destroy, create and start can be issued at runtime



Agenda

- Introduction
  - The History and Objectives of CEP
  - Research Questions
- A Testapplication for Experiments
  - The Class Model
  - Components
    - The NESPER CEP Engine
    - ZQL the SQL Parser
    - ZQL Enhancements
  - Configuration of the CEP Engine
    - Event Structures
    - Events and Actions
    - How to Change the Ruleset
- Analysis and Experiences
- Conclusion

Analysis And Experience

- Q1: Which kind of information can be retrived from a stream of SQL Statements (Single Session)?

- The critical success factor is the complexity of the data structure transmitted with the help SQL statements
  - Data may be splitted into several SQL statements
  - Within a logical unit of work a rollback may interrupt and rollback information
  - Data may be persisted in different RDBMS tables: One business obejct -> many rows in different tables.

- If the information goes one by one with the SQL statement, things are easy:
  - Defining the event type
  - Evaluating the data from SELECT, FROM, WHERE of a single statement

Analysis And Experience

- Q2: Which steps are necessary to change a rule set and which constraints have to be taken into account?

- Changing rules is as easy as pie, compared to changing IF-THEN-ELSE codings. Rules can be changed at runtime and activated at a certain point in time. The example of a banking application supports this experience. The business unit is used to design their own rules for credit risk assesment. The ruleset contains about 1000 rules, a single risk computation is done in about 1,5 s.

Analysis And Experience

- Q3: How to distinguish CEP from other rule driven software architectures?

➢ The difference between well known rule engine architectures is the time based message stack. All events are correlated to a time line. This can be done in nearly real time, because data doesn't need to be persisited in a database for further queuering.

➢ This methode can be used to build security proxies to filter SQL streams (DB Firewalls). On the other hand information wich is spread over different SQL statements or logical units of work are difficult to interprete.

➢ How a business object can be mapped to a group of SQL statements is a question which has to be investigated in further assignments. A case by case analysis seems to be necessary.

Analysis And Experience

➢ How a business object can be mapped to a group of SQL statements is a question which has to be investigated in further assignments. A case by case analysis seems to be necessary.

Analysis And Experience

- Experience

➢ To write an Oracle connector, grepping the 10046 trace, is an easy task

➢ An TNS proxy is the better choice, because the overhead for tracing can be avoided

Agenda

- Introduction
  - The History and Objectives of CEP
  - Research Questions
- A Testapplication for Experiments
  - The Class Model
  - Components
    - The NESPER CEP Engine
    - ZQL the SQL Parser
    - ZQL Enhancements
  - Configuration of the CEP Engine
    - Event Structures
    - Events and Actions
    - How to Change the Ruleset
- Analysis And Experiences
- Conclusion

Conclusion

- Complex Event Processing provides a complete design pattern to analyze streams of data. Architects are challenged by understanding and modeling the data structure of the event (Event Type) and to work out ideas how data streams could be correlated beneficial.
- Creating rule sets is easy, if you are used think in descriptive language pattern. Business people normally do!
- The CEP technology is mature enough to be used in products like risk assessment tools.
- No graphical rule editors have been found to create rule or to derive rules from process models.

Thanks for your participation