Assignment 1

IN5040

TA: Espen Volnes

Email: espenvol@ifi.uio.no

Important information

- Deadline: Tuesday 6 October 11:59
- Deliverable: PDF file containing:
 - -> Query and results for questions: 1 -3 4 6 7 8
 - -> Query, results and a short written answer for question 2
 - -> Short written answer for question 5

Overview

In this assignment we simulate a meteorological institute which receives summarised daily weather recordings from two weather stations

→ Period: 2014 - 2018

Weather stations: JFK airport - New York

San Francisco Airport

Purpose of the Assignment

To gain an understanding of some of the basic techniques used for analysing stream data.

 For this task, we will be using Esper, which is a software for Complex event processing (CEP) and Stream processing

Esper

- Language, compiler and runtime for DSM & CEP
- Basic version is a library that can be imported in Java or .NET
- Event Processing Language (EPL) enables rich expressions over events and time
- We will be using version 8.2.0
- http://esper.espertech.com/release-8.2.0/reference-esper
 - Look at page 3 points 2.1 2.10 and 2.17 for some of the most
 - important points for the assignment

How to learn syntax and features?

- 1. Documentation
 - Focus on the basic concepts to begin with
- 2. Esper notebook
 - https://www.esperonline.net/
- 3. Esper EPL online
 - https://esper-epl-tryout.appspot.com/epltryout/index.html

	Pagia Companta	
2. 1	Basic Concepts	
	2.1. Introduction	
	2.2. Basic Select	1
	2.3. Basic Aggregation	1
	2.4. Basic Filter	1
	2.5. Basic Filter and Aggregation	1
	2.6. Basic Data Window	
	2.7. Basic Data Window and Aggregation	1
	2.8. Basic Filter, Data Window and Aggregation	
	2.9. Basic Where-Clause	1
	2.10. Basic Time Window and Aggregation	1
	2.11. Basic Partitioned Statement	2
	2.12. Basic Output-Rate-Limited Statement	2
	2.13. Basic Partitioned and Output-Rate-Limited Statement	2
	2.14. Basic Named Windows and Tables	2
	2.14.1. Named Windows	2
	2.14.2. Tables	2
	2.15. Basic Aggregated Statement Types	2
	2.15.1. Un-Aggregated and Un-Grouped	2
	2.15.2. Fully Aggregated and Un-Grouped	2
	2.15.3. Aggregated and Un-Grouped	2
	2.15.4. Fully Aggregated and Grouped	2
	2.15.5. Aggregated and Grouped	2
	2.16. Basic Match-Recognize Patterns	2
	2.17. Basic EPL Patterns	2

Getting Started

- Download assignment1.tar.gz from the IN5040 course page
- Extract:

tar -xzvf assignment1.tar.gz

Assignment 1 contents

- Assignment1.pdf
- Assignment1.java
- WeatherTuple.java
- Makefile
- jfk.csv
- san_francisco.csv
- query1 8.epl
- → lib folder

Compilation

make all:

Cleans java class files Compiles java files

make runX

Runs the java files and the queryX, where X is the query number.

Assignment1.java

- You don't have to make changes here, but it's useful to understand what's happening
- What the program does
 - creates runtime system and deploys the selected query
 - Reads from the two data set files
 - Sets up two threads, one for each data set
 - The threads sleep between each processed event
 - You can comment out the sleep invocation in the java program to speed up the execution, but don't do it for the last query (query 8)
 - Reason is that query 8 looks at both data streams

Structure of a WeatherTuple

- timestamp
- weatherStation
- stationName
- averageTemperature
- minimumTemperature
- maximumTemperature
- averageWindSpeed
- precipitation
- weather

Query Structure

→ SELECT * FROM jfk

SELECT * FROM san_francisco

Esper architecture

Context Management

Responsible for lifecycle management of EPL query context partitions. Acts as a factory and container for context partitions.

Match Recognize Pattern Engine

Responsible for SQL-standard match-recognize patterns. Acts as a regular expression engine state machine matching events to pattern definitions.

Query Planning

Responsible for join, subquery, on-trigger and fireand-forget query planning and index building.

EPL Pattern Engine

Responsible for EPL pattern execution. Acts as a dynamic state tree that registers and de-registers filters and schedules based on the changing truth value of state nodes.

Named Window and Table Manager

Responsible for metadata and state associated to named windows and tables, and for pushing named window insert and remove stream data to named window consumers.

Result Set Processing

Responsible for evaluating the select-, having-, output and order-by clauses.

Filter Service

Responsible for managing interest for events among EPL queries. Acts as a highly performant sieve or switchboard keeping track of and indexing callbacks into from-clauses, patterns or context managers. Activated to evaluate incoming events to determine for a given event the interested EPL queries and their context partitions.

Aggregation Services

Responsible for keeping groups and applying changes to aggregations.

Data Windows

Responsible for managing events according to an expiry policy.

Expression Evaluation

Responsible for the expression tree and its evaluation. Activated to evaluate an expression and evaluates expressions in a top-down fashion. Includes date-time and enumeration methods.

Type System

Responsible for event types, event properties and event property getters. Acts as a metadata management and has not active role.

Schedule Management

Responsible for managing interest in time advancing. Activated when time passes to determine interested EPL queries and their context partitions.

EPL Syntax

• Create schema

```
create schema SchemaName(field_name1 type, field_name2 type)
```

• Data Stream Management:

```
select ID as sensorId, sum(countTags) as numTagsPerSensor
from AutoIdRFIDExample#win:time(60 seconds)
where Observation[0].Command = 'READ_PALLET_TAGS_ONLY'
group by ID
```

• CEP:

Window

- Called a view in Esper
- Choose correct window when writing EPL queries!
- Examples of windows/views:

View	Syntax	Description
Time window	win:time(time period)	Sliding time window
Externally-timed window	win:ext_timed(timesta mp expression, time period)	Sliding time window, based on the millisecond time value supplied by an expression
Time batch window	win:time_batch(time period[,optional reference point] [, flow control])	Tumbling window that batches events and releases them every specified time interval, with flow control options
Externally-timed batch window	<pre>win:ext_timed_batch(timestamp</pre>	Tumbling window, based on the millisecond time value supplied by an expression