Mandatory exercise – INF5100

Introduction

In this mandatory exercise you will use Esper to learn about database systems and complex event processing. Esper is a component for complex event processing (CEP) and event series analysis. By completing this mandatory exercise and the assignments within, you will get a brief introduction to Esper. If you wish to learn more about Esper you can find more information on their webpage: www.espertech.com/esper

Delivery

Who: Alone

When: Hard Deadline! Wednesday, October 5, 2016, 14:00h

Where: https://delivery.ifi.uio.no

What: 1 PDF file that contains all queries, answers to the theoretical questions, table with results and graphs where suitable.

Administrative Issues

The questions are structured in a way that the easier questions come first, and the more difficult questions comes later. In addition, the theoretical questions are meant to prepare you for the upcoming Esper-questions. This means that you should answer the questions in the given order. All questions must be answered in order to pass the assignment. Finally, you are not meant to copy/paste from either the Esper web page or the course material. Instead you should read, understand and write the answers in your own words. Failing to do this properly will result in failing this exercise. Only students that have passed the mandatory exercise are allowed to take part in the examination!

Help (TA): Vivek Kaul, vivekk@ifi.uio.no

How to get started

Download the mandatory exercise from http://www.uio.no/studier/emner/matnat/ifi/INF5100/h16/exercise/ and unzip it. The bundle contains two directories:

Esper-4.9.0/ - contains Esper and its libraries. In order to solve this mandatory exercise, you will need to read the Esper reference documentation. All needed documentation can be found inside this folder:
mandatory_exercise/esper4.9.0/esper/doc/reference/html_single/index.html.

Inf5100/ - contains the following files:

- *Makefile*: Simple Makefile with three commands.
    - make – compiles the exercise
    - make run – runs the queries
    - make clean – removes the Java class files

- *BTC.csv*: The emulated stock exchange, consisting of events (ticks). You are not allowed to modify this file, but you are encouraged to examine the file in order to verify that your queries are correct.

- *INF5100.java*: The code used to generate events and read your queries. By default, the code will loop over all the queries. This can make debugging quite hard. If you want to only test one query, you will need to modify the for-loop on line 28 inside this file so it only reads one query.

- *Query_[1…9].java:* These are the files you should change according to the questions.

- *Tick.java:* The event class. An event consists of the following attributes.
    - symbol - Stock ticker or stock alias. This will be BTC in our simulated stock exchange.
    - timestamp – The date of the event/tick. Our stock exchange sends one tick per day.
    - open - The price of the stock at the beginning of the day.
    - high - The highest price the stock was traded for this day.
    - low - The lowest price the stock was traded for this day.
    - close - The price of the stock at the end of the day.
    - volume - The number of times this stock was bought/sold this day.
    - weightedPrice – A weighted average of the stock price this day.

Question 1 – Text (~0.5 PAGE)

Explain the motivation for DSMS and elaborate on the differences between a traditional DBS and DSMS. In addition, what applications can make use of DSMS?

Question 2 – Query_1.EPL (QUERY + RESULTS)

Find the date of the BTC stock when the closing price is greater than 17 and less than 20.

Question 3 – Text (~0.5 page)

What is a window in DSMS and what are windows used for? In the explanation you should include the difference between Tuple Count Based Windows and Punctuation Based Windows.

Based on your above explanation, what is then the difference between doing aggregated queries (AVG/SUM/MAX/ETC) in standard databases and steam databases?

Question 4 – Query_2.EPL (THOUGHTS + QUERY + RESULTS)

For each non-overlapping 100 tick interval of the BTC stock, find the minimum and maximum weighted price in the interval.

Hint: Chapter 3. Processing Model

Question 5 – Query_3.EPL (THOUGHTS + QUERY + RESULTS)

Extend the above query so that it now also displays the difference between the minimum and maximum weighted price for each interval.

Question 6 – Query_4.EPL (THOUGHTS + QUERY + RESULTS)

Find the total weighted price of all stock tick events in each non-overlapping 100 tick interval of the BTC stock.

Question 7 – Query_5.EPL (THOUGHTS + QUERY + RESULTS)

Find the dates and minimum weighted price for the BTC stock for each, non-overlapping 50 tick interval.

Be aware, this is trickier than anticipated. Double check that the prices correspond to the correct dates!

Question 8 – Text (~0.5 PAGE)

What are patterns and pattern operators? What does the every and followed-by (->) patterns do?

Explain the difference between the following:

- every A -> B
- A -> every B
- every A -> every B
- every (A-> B)

Hint: Chapter 6. EPL Reference: Patterns

Question 9 – Query_6.EPL (THOUGHTS + QUERY + RESULTS)

Find all the start and stop dates with volume, when the weighted price of the BTC stock increases more than 100 times and volume increases by 40%.

Hint: Chapter 5. EPL Reference: Clauses

Question 10 – Text (~0.5 PAGE)

Explain the difference between a centralized DBS and a distributed DBS. Moreover, why do we need distributed DBS? What are the advantages and disadvantages of distributed DBS? Finally, elaborate on the different distributed DBS architectural configurations *(Hint: there are three of them).*

Question 11 – Query_7.EPL (THOUGHTS + QUERY)

We claim that a good buying date for the BTC stock is the date after which the stock has a lower weighted price than a date before it (not necessarily right before), and the volume has increased by 70% as that same earlier date. Vice versa, a good selling date is when the stock has three times increased weighted price than a date before it (not necessarily right before) and a 70% increase in volume. Find the buy dates and closing prices and the sell dates and closing prices to test this hypothesis.

Question 12 – Query_8.EPL (THOUGHTS + QUERY)

We will now try a different algorithm for buying and selling BTC stocks. This algorithm will be based on *trend following*. Trend following is an investment strategy based on technical analysis of market prices, rather than on the fundamental strengths of the companies. We will try to take advantage of the market trends by observing the current direction and using this to decide whether to buy or sell. The strategy is simple: If we see that the weighted price and closing prices are going up over three days, we buy.

Example: If we have an event of three days (not necessarily adjacent days), Monday, Tuesday and Wednesday. If we see that the weighted price has increased from Monday to Tuesday and then again from Tuesday to Wednesday, in addition the closing price on Wednesday is larger than Monday, we buy. And vice versa for selling, if we see that the weighted price has gone down from Monday to Tuesday and then again from Tuesday to Wednesday, in addition the closing price on Wednesday is larger than the closing price on Monday, we sell.

QUESTION 13 – TEXT (~0.5 PAGE)

What are pattern guards? Why cannot the timer:within pattern guard be used in simulated environments? What alternatives exists to control time and dates?

Hint: Chapter 6. EPL Reference: Patterns and Chapter 11. EPL Reference: Date-Time Methods

Question 14 – Query_9.EPL (THOUGHTS + QUERY + RESULTS)

Find the start date, stop date and weighted price when the BTC stock value has decreased by three times within a 15-day period.

Hint: A correct query to this question should yield the following row(1):

EVENT! {StartWeightedPrice=214.67, StopDate=Tue Apr 16 00:00:00 CEST 2013, StopWeightedPrice=65.33, startDate=Tue Apr 09 00:00:00 CEST 2013}

Hint: Chapter 11. EPL Reference: Date-Time Methods

Question 15 – Text (~0.5 PAGE)

What are heterogeneous database system (HDBS), and how does HDBS work? In addition, explain the concepts of the integration layer and explain the issues related to integration.

Question 16 – Text (~0.5 PAGE)

What did you struggle with during this exercise? Was it the provided codes fault, Esper's or your own fault?