IN5040
Assignment

Thomas Plagemann

Important information

- Deadline: Monday September 28, 2026, 23:59h

- Deliverable:
  - What: a 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
  - Where: https://devilry.ifi.uio.no

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 system 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
- Documentation: http://esper.espertech.com/release-8.2.0/reference-esper
- Try online: https://esper-epl-tryout.appspot.com/epltryout/mainform.html

How to learn syntax and features?

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

2. Basic Concepts 9
2.1. Introduction 9
2.2. Basic Select 10
2.3. Basic Aggregation 11
2.4. Basic Filter 12
2.5. Basic Filter and Aggregation 13
2.6. Basic Data Window 14
2.7. Basic Data Window and Aggregation 15
2.8. Basic Filter, Data Window and Aggregation 17
2.9. Basic Where-Clause 17
2.10. Basic Time Window and Aggregation 19
2.11. Basic Partitioned Statement 20
2.12. Basic Output-Rate-Limited Statement 21
2.13. Basic Partitioned and Output-Rate-Limited Statement 22
2.14. Basic Named Windows and Tables 23
2.14.1. Named Windows 23
2.14.2. Tables 25
2.15. Basic Aggregated Statement Types 27
2.15.1. Un-Aggregated and Un-Grouped 27
2.15.2. Fully Aggregated and Un-Grouped 27
2.15.3. Aggregated and Un-Grouped 28
2.15.4. Fully Aggregated and Grouped 28
2.15.5. Aggregated and Grouped 28
2.16. Basic Match-Recognize Patterns 29
2.17. Basic EPL Patterns 29

http://esper.espertech.com/release-8.2.0/reference-esper/pdf/esper_reference.pdf

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
- query_1.epl, query_2.epl, …..query_8.epl
- lib folder

Compilation

- make all:
  - cleans java class files
  - compiles java files

- make runX: runs the java files and the query_X.epl, 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

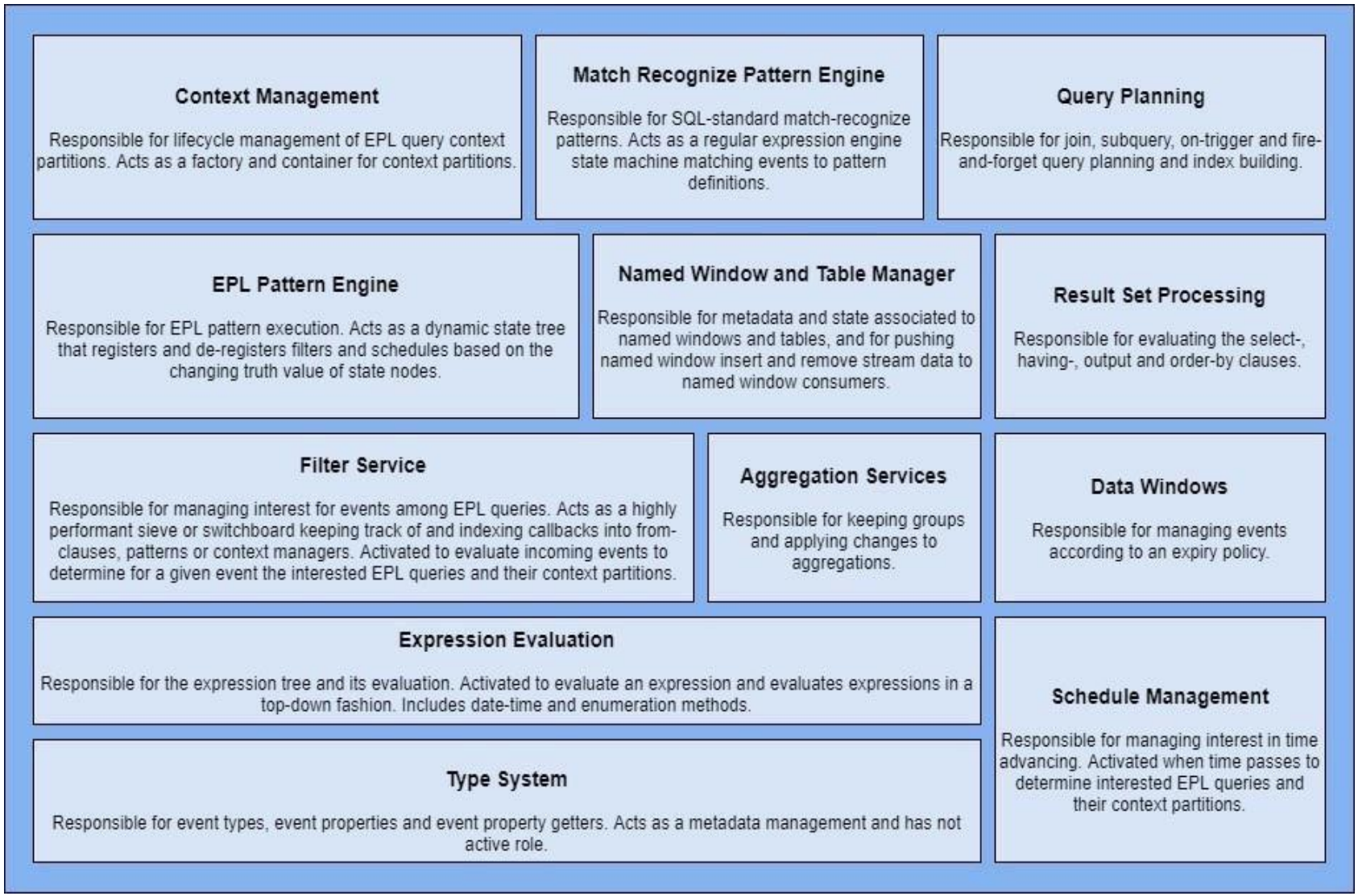
- Timestamp
- weatherStation
- stationName
- averageTemperature
- minimumTemperature
- maximumTemperature
- averageWindSpeed
- precipitation
- weather

Query Structure

```
SELECT *
FROM jfk
```

```
SELECT *
FROM san_francisco
```

Esper Architecture



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

```
select a.custId, sum(a.price + b.price)
from pattern [every a=ServiceOrder ->
                    b=ProductOrder(custId = a.custId)
              where timer:within(1 min)]#win:time(2 hour)
where a.name in ('Repair', b.name)
group by a.custId
having sum(a.price + b.price) > 100
```

Window

- Choose correct window when writing EPL queries!
- Examples of windows/views:

Window	Syntax	Description
Time window	win:time(time period)	Sliding time window
Externally-timed window	win:ext_timed(timestamp 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	win:ext_timed_batch(*timestamp_expression*, *time_period* [,*optional_reference_point*])	Tumbling window, based on the millisecond time value supplied by an expression

Example: https://esper-epl-tryout.appspot.com/epltryout/mainform.html

- EPL Statements
- **EPL Module Text**
- Enter EPL Here:

```
create schema ProductOrder(custId string, name
string, price double);
create schema ServiceOrder(custId string, name
string, price double);

@Name('Out') select a.custId, sum(a.price +
b.price) from pattern [every a=ServiceOrder ->
b=ProductOrder(custId =a.custId) where
timer:within(1 min)]#win:time(20 min)
where a.name in ('Repair', b.name) group by
a.custId having sum(a.price + b.price) > 100;
```

Time And Event Sequence

Beginning Of Time

Provide a timestamp to start at:
2001-01-01 08:00:00.000

```
ServiceOrder={custId='customer1', name='Repair',
price=60.5}
t=t.plus(5 seconds)
ProductOrder={custId='customer1', name='12345',
price=53}
t=t.plus(60 seconds)
ServiceOrder={custId='customer2', name='6789',
price=90}
t=t.plus(25 seconds)
ProductOrder={custId='customer2', name='6789',
price=60}
t=t.plus(120 seconds)
ServiceOrder={custId='customer1', name='Repair',
price=60.5}
t=t.plus(10 seconds)
ProductOrder={custId='customer1', name='7654',
price=0.5}
```

Scenario Results

All Output Events

```
At: 2001-01-01 08:00:05.000
Statement: Out
Insert
Out-output={a.custId='customer1',
sum(a.price+b.price)=113.5}

At: 2001-01-01 08:01:30.000
Statement: Out
Insert
Out-output={a.custId='customer2',
sum(a.price+b.price)=150.0}

At: 2001-01-01 08:03:40.000
Statement: Out
Insert
Out-output={a.custId='customer1',
sum(a.price+b.price)=174.5}
```