Stream-temporal Querying with Ontologies

Ralf Möller
Christian Neuenstadt
Özgür L. Özçep

Hidest ’15, Dresden
Visualisation & Analysis

Query Formulation

Ontology & Mapping Management

Ontology

Mappings

Query Transformation

Query Planning

Query Execution

Query Execution

Query Execution

streaming data

temporal data

static data

OBDA with streams and temporal data
End-user

Visualisation & Analysis

Query Formulation

Query Execution

IT-expert

Query Formulation

Ontology & Mapping Management

Ontology

Mappings

Query Transformation

Query Planning

Query Execution

OBDA with streams and temporal data

Data models

Std. ontologies...
End-user

Visualisation & Analysis

Query Formulation

Query Transformation

Query Planning

Ontology & Mapping Management

Ontology

Mappings

IT-expert

Data models

Std. ontologies...

STARQL engine

Query Execution

Streaming data

Temporal data

Static data

OBDA with streams and temporal data

Query Formulation

Query Transformation

Query Planning

Ontology & Mapping Management

Ontology

Mappings

STARQL query

STARQL engine

Query Execution

Query Execution

Query Execution

Query Execution

Query Execution

Query Execution

Query Execution
Simplified Schema

CREATE TABLE measurement (Mtimestamp, SID, Mval);
CREATE TABLE sensor (SID, assemblypart, name)

Datasets from Siemens Turbine Diagnostic Center

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Total No. Meas.</th>
<th>Timespan</th>
<th>Sensors</th>
<th>Meas. per Day/Sensor</th>
<th>Total Data Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ds1</td>
<td>82080</td>
<td>3 Days</td>
<td>19</td>
<td>1440</td>
<td>5 MB</td>
</tr>
<tr>
<td>Ds2</td>
<td>210,000</td>
<td>1506 Days</td>
<td>3</td>
<td>46.5</td>
<td>10 MB</td>
</tr>
<tr>
<td>Ds3</td>
<td>515,845,000</td>
<td>1824 Days</td>
<td>204</td>
<td>1386</td>
<td>23,000 MB</td>
</tr>
</tbody>
</table>
Information Need

Starting with time point 00:00 on 1.1.2005, give me every second those temperature sensors whose value grew monotonically in the last 2 seconds.

- Information need for
  - historical data (temporal query) with period of interest
    \(\implies\) needed for reactive diagnosis
  - streamed data (continuous query)
    \(\implies\) needed for proactive diagnosis

- Goal: Use the same query language
  - For historical queries, windows can be processed in parallel with no real-time constraints
Example: Input Stream

Input: One stream $S_{Msmt}$ of measurement assertions.

$$S_{Msmt} = \{ \text{val}(s_0, 90^\circ)\langle 0s \rangle, \text{val}(s_0, 93^\circ)\langle 1s \rangle, \text{val}(s_0, 94^\circ)\langle 2s \rangle, \text{val}(s_0, 92^\circ)\langle 3s \rangle, \text{val}(s_0, 95^\circ)\langle 5s \rangle, \ldots \}$$
Mappings

- Mapping historical data

\[ m_1 : \text{val}(x, y) \langle z \rangle \leftarrow \]

```sql
SELECT f(SID) AS x, Mval AS y, MtimeStamp AS z FROM MEASUREMENT-TABLE
```

- Mapping streams

\[ m_2 : \text{val}(x, y) \langle z \rangle \leftarrow \]

```sql
SELECT Rstream(f(SID) AS x, Mval AS y, MtimeStamp AS z) FROM MEASUREMENT-REL-STREAM
```
PREFIX : <http://www.optique-project.eu/siemens>
CREATE STREAM S_out AS
CONSTRUCT GRAPH NOW { ?s rdf:type MonInc }
FROM STREAM S_Msmt [NOW-2s, NOW]->“1S”^^xsd:duration
   WITH START = "2005-01-01T01:00:00CET”^^xsd:dateTime,
       END = "2005-01-01T02:00:00CET”^^xsd:dateTime
STATIC ABOX <http://www.optique-project.eu/siemens/ABoxstatic>,
   TBOX <http://www.optique-project.eu/siemens/TBox>
USING PULSE WITH
   START = "2005-01-01T00:00:00CET”^^xsd:dateTime,
   FREQUENCY = "1S”^^xsd:duration
WHERE { ?s rdf:type :TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL ?i < ?j IN seq
   FOR ALL ?x,?y:
      THEN ?x <= ?y
PREFIX : <http://www.optique-project.eu/siemens>
CREATE STREAM S_out AS
  CONSTRUCT GRAPH NOW { ?s rdf:type MonInc }
FROM STREAM S_Msmt [NOW-2s, NOW]->"1S"^^xsd:duration
  WITH START = "2005-01-01T01:00:00CET"^^xsd:dateTime,
    END = "2005-01-01T02:00:00CET"^^xsd:dateTime
STATIC ABOX <http://www.optique-project.eu/siemens/ABoxstatic>,
  TBOX <http://www.optique-project.eu/siemens/TBox>
USING PULSE WITH
  START = "2005-01-01T00:00:00CET"^^xsd:dateTime,
  FREQUENCY = "1S"^^xsd:duration
WHERE { ?s rdf:type :TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL ?i < ?j IN seq
  FOR ALL ?x,?y:
    THEN ?x <= ?y
Output Format and Frequency

\[
\begin{align*}
S_{Msmt} & = \{ \text{val}(s_0, 90^\circ)\langle 0s \rangle, \\
& \quad \text{val}(s_0, 93^\circ)\langle 1s \rangle, \\
& \quad \text{val}(s_0, 94^\circ)\langle 2s \rangle, \\
& \quad \text{val}(s_0, 92^\circ)\langle 3s \rangle, \\
& \quad \text{val}(s_0, 95^\circ)\langle 5s \rangle, \\
& \quad \ldots \}\}
\end{align*}
\]

\[
\begin{align*}
S_{out} & = \{ \text{MonInc}(s_0)\langle 0s \rangle, \\
& \quad \text{MonInc}(s_0)\langle 1s \rangle, \\
& \quad \text{MonInc}(s_0)\langle 2s \rangle, \\
& \quad \text{MonInc}(s_0)\langle 5s \rangle, \\
& \quad \ldots \}\}
\end{align*}
\]
PREFIX : <http://www.optique-project.eu/siemens>

CREATE STREAM S_out AS
CONSTRUCT GRAPH NOW { ?s rdf:type MonInc }
FROM STREAM S_Msmt [NOW-2s, NOW]->"1S"^^xsd:duration
WITH START = "2005-01-01T01:00:00CET"^^xsd:dateTime,
END = "2005-01-01T02:00:00CET"^^xsd:dateTime
STATIC ABOX <http://www.optique-project.eu/siemens/ABoxstatic>,
TBOX <http://www.optique-project.eu/siemens/TBox>
USING PULSE WITH
START = "2005-01-01T00:00:00CET"^^xsd:dateTime,
FREQUENCY = "1S"^^xsd:duration
WHERE { ?s rdf:type :TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL ?i < ?j IN seq
FOR ALL ?x,?y:
  THEN ?x <= ?y
**Window Semantics**

- $S_{Msmt} [NOW-2s,NOW] \rightarrow 1s$: stream of temporal ABoxes
- Sliding movement as in CQL but with timestamp preservation

<table>
<thead>
<tr>
<th>Time</th>
<th>Window contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0s</td>
<td>$val(s_0, 90^\circ)\langle 0s \rangle$</td>
</tr>
<tr>
<td>1s</td>
<td>$val(s_0, 90^\circ)\langle 0s \rangle, \ val(s_0, 93^\circ)\langle 1s \rangle$</td>
</tr>
<tr>
<td>2s</td>
<td>$val(s_0, 90^\circ)\langle 0s \rangle, \ val(s_0, 93^\circ)\langle 1s \rangle, \ val(s_0, 94^\circ)\langle 2s \rangle$</td>
</tr>
<tr>
<td>3s</td>
<td>$val(s_0, 93^\circ)\langle 1s \rangle, \ val(s_0, 94^\circ)\langle 2s \rangle, \ val(s_0, 92^\circ)\langle 3s \rangle$</td>
</tr>
<tr>
<td>4s</td>
<td>$val(s_0, 94^\circ)\langle 2s \rangle, \ val(s_0, 92^\circ)\langle 3s \rangle$</td>
</tr>
<tr>
<td>5s</td>
<td>$val(s_0, 92^\circ)\langle 3s \rangle, \ val(s_0, 95^\circ)\langle 5s \rangle$</td>
</tr>
</tbody>
</table>
## Timestamps to Sequences

**Optique**

<table>
<thead>
<tr>
<th>Time</th>
<th>Window contents before sequencing</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>5s</td>
<td>(\text{val}(s_0, 92^\circ)\langle3s\rangle, \text{val}(s_0, 95^\circ)\langle5s\rangle)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Window contents after standard sequencing</th>
<th>SEQ1</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>5s</td>
<td>({\text{val}(s_0, 92^\circ)}\langle0\rangle, {\text{val}(s_0, 95^\circ)}\langle1\rangle)</td>
<td>({0,1})</td>
</tr>
</tbody>
</table>

- Timestamped assertions are grouped to ABoxes (RDF graphs) with state index
- Information on timestamps and on their distance gets lost
- For transformation to SQL: the timestamp-index-window id has to be stored
  \(\Rightarrow\) window(\(\text{Index}\), \(\text{Timestamp}\), \(\text{WID}\))
PREFIX : <http://www.optique-project.eu/siemens>
CREATE STREAM S_out AS
CONSTRUCT GRAPH NOW { ?s rdf:type MonInc }
FROM STREAM S_Msmt [NOW-2s, NOW]->"1S"^^xsd:duration
   WITH START = "2005-01-01T00:00:00CET"^^xsd:dateTime,
      END = "2005-01-01T02:00:00CET"^^xsd:dateTime
STATIC ABOX <http://www.optique-project.eu/siemens/ABoxstatic>,
TBOX <http://www.optique-project.eu/siemens/TBox>
USING PULSE WITH
   START = "2005-01-01T00:00:00CET"^^xsd:dateTime,
   FREQUENCY = "1S"^^xsd:duration
WHERE { ?s rdf:type :TempSens }
SEQUENCE BY StdSeq AS seq
HAVING FORALL ?i < ?j IN seq
   FOR ALL ?x,?y:
      THEN ?x <= ?y
Testing the Conditions

$S_{Msmt} = \{ \text{val}(s_0, 90^\circ C)\langle 0s \rangle, \text{val}(s_0, 93^\circ C)\langle 1s \rangle, \text{val}(s_0, 94^\circ C)\langle 2s \rangle, \text{val}(s_0, 92^\circ C)\langle 3s \rangle, \text{val}(s_0, 95^\circ C)\langle 5s \rangle \ldots \}$

$S_{out} = \{ \text{MonInc}(s_0)\langle 0s \rangle, \text{MonInc}(s_0)\langle 1s \rangle, \text{MonInc}(s_0)\langle 2s \rangle, \text{MonInc}(s_0)\langle 5s \rangle \ldots \}$

$\text{MonInc(s0)}<\text{NOW}>? \quad \text{yes} \quad \text{yes} \quad \text{yes} \quad \text{no} \quad \text{no} \quad \text{yes}$

<table>
<thead>
<tr>
<th>Time/sec</th>
<th>Temp/$^\circ C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>90.0</td>
</tr>
<tr>
<td>1.0</td>
<td>91.0</td>
</tr>
<tr>
<td>2.0</td>
<td>92.0</td>
</tr>
<tr>
<td>3.0</td>
<td>93.0</td>
</tr>
<tr>
<td>4.0</td>
<td>94.0</td>
</tr>
<tr>
<td>5.0</td>
<td>95.0</td>
</tr>
</tbody>
</table>
STARQL Engine: Architecture

- Query (String)
- STARQL Grammar
- Normalization Rules
- SQL Query and Views
- Parser
- Query (Data Structure)
- Normalizer
- Query (Normal Form)
- Datalog Transformer
- Datalog Program
- Optimizer
- Optimized Datalog Program
- SQL Transformer
- Datalog Transformation Rules
- Optimization Rules
- Mapping Rules

- SQL Transformation Rules

Optique
Tested feasibility of engine

Datasets of increasing size (5MB, 10MB, 23,000MB)

Two queries: monotonic increase (quadratic) and threshold query (linear)

Two different approaches

- preprocessing of window table (temporal querying)
- incremental generation of window table (temporal and stream)

Results

Incremental approach more feasible; but preprocessing step useful for caching.
• EU 7th framework program

• Two big data use cases from industrial partners
  • STATOIL SAS: Querying data on wellbore related DBs
  • SIEMENS: Querying sensor and event data from (gas) turbines

• Optique platform: OBDA + User Friendliness + Scalable Rewriting + Elastic Clouds + Real-Time Processing

• For more information: http://www.optique-project.eu/

• Demos and Optique assets to be found at fluidOps http://appcenter.fluidops.net
One can build stream-temporal OBDA engines that
1. have a query language with neat formal semantics,
2. use standard backend systems (such as PostgreSQL)
3. and provide good performance (under optimized transformation)