OBDA for Temporal Querying and Streams

Christian Neuenstadt, Ralf Möller, Özgür L. Özçep
Institute of Information Systems (ifis)
University of Lübeck

22. September 2015

HiDeSt '15

Streaming in OPTIQUE

- EU funded FP7 project
- Task: simplify big data acces for engineers
- Goal: establish OBDA for industrial applications
- Two industrial stakeholders STATOIL and SIEMENS
- UseCase: Querying of time and streamed data

Streaming with Starql

- New sparql-like stream-temporal language STARQL
- Flexible Framework for use in context with ABDEO, OBDA or simple RDF-Streams
- Support of SPARQL 1.0+ in the OBDA usecase
- Building sequences of ABoxes
- Possibility for different window operators

Starql Stream-Operators

S2R:
- Timebased window input, based on window parameter
- Sequence operator builds sequence of ABoxes

R2R:
- Where-Clause for access on static data
- Time based Having-Clause for access on time sequences

R2S:
- Timeintervall based stream output (Pulse)
- RStream principle
- Binding tuples or graph triple

A simple Starql Example 1/2

```
CREATE STREAM S_out AS
CONSTRUCT  GRAPH NOW { ?s rdf:type MonInc }
FROM STREAM S_Msmt [NOW−"1S"^^xsd:duration,
    NOW]−>"1S"^^xsd:duration
SEQUENCE BY StdSeq AS seq
```

Sliding window view in the measurement scenario

Timestamp	Sensor	Value
00:00	sens1	90
00:01	sens1	91
00:01	sens1	91
:	:	:
:	:	:

WindowID	*ABoxID*	*Timestamp*	Sensor	Value
1	1	00:00	sens1	90
1	2	00:01	sens1	91
2	1	00:01	sens1	91
:	:	:	:	:
:	:	:	:	:

A simple Starql Example 2/2

HAVING FORALL ?i,?j **IN** seq,?x,?y:
IF (**GRAPH** ?i { ?s :val ?x } **AND GRAPH** ?j { ?s :val ?y }
AND ?i < ?j)
THEN ?x <= ?y

OBDA: Range Restriction for Starql Having-Clause

- Transformation from one query language into another requires domain independence $\rightarrow$ All variables have to be range restricted
- Example for non restricted variable x: '$x > 1$'
- Can be restricted by: '$x > 1$ **AND** GRAPH *?i* { *?s :val ?x* }'
- Starql requires the Having-Clause in Safe Range Form, although not necessary each subformula has to be range restricted.
- Transformation steps Starql-Having $\rightarrow$ SRNF $\rightarrow$ RANF $\rightarrow$ Algebra

Starql Example Parse Tree

–**HAVING**–
––**FORALL**[i, j][?y, ?x]
––––**IF**
––––––**AND**
––––––––**AND**
––––––––––**GRAPH** { :TC255 :hasVal ?x . }<i>
––––––––––**GRAPH** { :TC255 :hasVal ?y . }<j>
––––––––i < j
––––**THEN**
––––––_x <= _y

SRNF for Starql-HavingClause

- Variable Substitution: $x \to x_1$, $x \to x_2$
- Remove universal quantifier: $\forall \vec{x} \psi \to \neg \exists \vec{x} \neg \psi$
- Remove implications: $\psi \to \xi \to \neg \psi \vee \xi$
- Push down negations: only $\neg \exists \psi$ or $\neg \mathbf{a}$ allowed
- Flattening:
  - $\psi_1 \wedge (\psi_2 \wedge \psi_3) \to \psi_1 \wedge \psi_2 \wedge \psi_3$
  - $\psi_1 \vee (\psi_2 \vee \psi_3) \to \psi_1 \vee \psi_2 \vee \psi_3$
  - $\exists \vec{x} \exists \vec{y} \to \exists \vec{x} \vec{y}$

Variable Substitution

```
--FORALL[i, j][?y, ?x]
----IF
------AND
--------AND
----------GRAPH { :S1 :hasVal ?x . }<i>
----------GRAPH { :S1 :hasVal ?y . }<j>
--------i < j
----THEN
------_x <= _y
```

Remove universal quantifier

NOT
––**EXISTS**[i, j][?y, ?x]
––––**NOT**
––––––**IF**
––––––––**AND**
––––––––––**AND**
––––––––––––**GRAPH** { :S1 :hasVal ?x . }<i>
––––––––––––**GRAPH** { :S1 :hasVal ?y . }<j>
––––––––––i < j
––––––**THEN**
––––––––_x <= _y

Remove implications

NOT
––**EXISTS**[i, j][?y, ?x]
––––**NOT**
––––––**OR**
––––––––**NOT**
––––––––––**AND**
––––––––––––**AND**
––––––––––––––**GRAPH** { :S1 :hasVal ?x . }<i>
––––––––––––––**GRAPH** { :S1 :hasVal ?y . }<j>
––––––––––––i < j
––––––––_x <= _y

Push down negations

NOT
−−**EXISTS**[i, j][?y, ?x]
−−−−**AND**
−−−−−−**AND**
−−−−−−−−**AND**
−−−−−−−−−−**GRAPH** { :S1 :hasVal ?x . }<i>
−−−−−−−−−−**GRAPH** { :S1 :hasVal ?y . }<j>
−−−−−−−−i < j
−−−−−−−_x > _y

Flattening

NOT
––**EXISTS**[i, j][?y, ?x]
––––**AND**
––––––**GRAPH** { :S1 :hasVal ?x . }<i>
––––––**GRAPH** { :S1 :hasVal ?y . }<j>
––––––i < j
––––––_x > _y

From SRNF to RANF

- "A SRNF formula ψ is in relational algebra normal form (RANF) if each subformula of ψ is self contained"
- An occurrence of a subformula ψ in ϕ is *self-contained* if its root is $\wedge$ or if
  1. $\psi = \psi_1 \vee ... \vee \psi_n$ and $rr(\psi) = rr(\psi_1) = ... = rr(\psi_n) = free(\psi)$
  2. $\psi = \exists \vec{x} \psi_1$ and $rr(\psi) = free(\psi_1)$
  3. $\psi = \neg \psi_1$ and $rr(\psi) = free(\psi_1)$
- $rr(\psi)$: set of all range restricted variables of ψ
  $free(\psi)$: set of all unbound variables of ψ

Algorithm: Transformation into Algebra 1/2

Input: a formula ψ in modified RANF
Output: an algebra query E_ψ equivalent to ψ

$$
\begin{array}{rcl}
R(\tilde{e}) & \longrightarrow & \pi_{wid,aid,x_1,\ldots,x_n}(R) \\
x = a & \longrightarrow & \{\langle x : a \rangle\} \\
\psi \wedge \xi & \longrightarrow & \textit{see next slide} \\
\neg\psi & \longrightarrow & \{\langle\rangle\} - E_\psi \\
 & & \text{(if } \neg\psi \text{ does not have “and” as parent)} \\
\psi_1 \vee \ldots \vee \psi_n & \longrightarrow & E_\psi \cup \ldots \cup E_\psi \\
\exists x_1,\ldots,x_n \psi(x_1,\ldots,x_n,\, y_1,\ldots,y_m) & \longrightarrow & \pi_{wid,y_1,\ldots,y_n}(E_\psi)
\end{array}
$$

Algorithm: Transformation into Algebra 2/2

$\psi \wedge \xi \quad \longrightarrow \quad$ if ξ is $x = x$, then E_ψ

if ξ is $x = y$, (with x, y distinct) then

$\sigma_{x=y}(E_\psi)$, if $\{x, y\} \subseteq \mathit{free}(\psi)$

$\sigma_{x=y}(E_\psi \bowtie \delta_{x \to y} E_\psi)$, if $x \in \mathit{free}(\psi)$ and $y \notin \mathit{free}(\psi)$

$\sigma_{x=y}(E_\psi \bowtie \delta_{y \to x} E_\psi)$, if $y \in \mathit{free}(\psi)$ and $x \notin \mathit{free}(\psi)$

if ξ is $x \neq y$, then $\sigma_{x \neq y}(E_\psi)$

if $\xi = \neg \xi'$, then

$E_\psi - (E_\psi \bowtie E_{\xi'})$, if $\mathit{free}(\xi') \subset \mathit{free}(\psi)$

$E_\psi - E_{\xi'}$, if $\mathit{free}(\xi') = \mathit{free}(\psi)$

otherwise, $E_\psi \bowtie E_\xi$

Transformation Example

$$\neg\exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (1)$$

$$\{\langle\rangle\} - \exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (2)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (3)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(R(:Sens1, x, i) \wedge S(:Sens1, y, j)))) \quad (4)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(\pi_{wid,aid,s,x}(R) \bowtie \pi_{wid,aid,s,y}(S)))) \quad (5)$$

Transformation Example

$$\neg\exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (1)$$

$$\{\langle\rangle\} - \exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (2)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (3)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(R(:Sens1, x, i) \wedge S(:Sens1, y, j)))) \quad (4)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(\pi_{wid,aid,s,x}(R) \bowtie \pi_{wid,aid,s,y}(S)))) \quad (5)$$

Transformation Example

$$\neg\exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (1)$$

$$\{\langle\rangle\} - \exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (2)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (3)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(R(:Sens1, x, i) \wedge S(:Sens1, y, j)))) \quad (4)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(\pi_{wid,aid,s,x}(R) \bowtie \pi_{wid,aid,s,y}(S)))) \quad (5)$$

Transformation Example

$$\neg\exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (1)$$

$$\{\langle\rangle\} - \exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (2)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (3)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(R(:Sens1, x, i) \wedge S(:Sens1, y, j)))) \quad (4)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(\pi_{wid,aid,s,x}(R) \bowtie \pi_{wid,aid,s,y}(S)))) \quad (5)$$

Transformation Example

$$\neg\exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (1)$$

$$\{\langle\rangle\} - \exists i, j, x, y(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (2)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(R(:Sens1, x, i) \wedge S(:Sens1, y, j) \wedge i < j \wedge x > y) \quad (3)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(R(:Sens1, x, i) \wedge S(:Sens1, y, j)))) \quad (4)$$

$$\{\langle\rangle\} - \pi_{wid,aid,s}(\sigma_{x>y}(\sigma_{i<j}(\pi_{wid,aid,s,x}(R) \bowtie \pi_{wid,aid,s,y}(S)))) \quad (5)$$

Demo of a Starql Query in Postgresql

- Formulating a STARQL Query
- Transforming Starql into SQL
- Running the query
- View the results