A Path Querying Language for Federation of RDF and Relational Database

Jiahui Zhang, Xiaowang Zhang, and Zhiyong Feng

Tianjin University

Jiahui Zhang, Xiaowang Zhang, and Zhiyong Feng

Tianjin University

zhangjiahui@tju.edu.cn
xiaowangzhang@tju.edu.cn
zyfeng@tju.edu.cn
Outline

- Introduction
- Federated Path Querying Language (FPQ)
- Expressiveness of FPQ
- Experiment and Evaluation
Introduction

**SPARQL**
The standard language for querying RDF data since 2008.

**Navigational Capability**
Versa: using XPath over the XML of RDF graphs
SPARQLeR: adding path variables
CPSPARQL: allowing constraints over regular expressions
nSPARQL: applying nested regular expressions
SPARQL 1.1: appending property paths
Introduction

Q: Go from Rouen to Reims.

- Rouge→Dreux→Paris→Reims

- Rouge→Paris→Reims

Driver

Figure: Geographical map.
Introduction

Q': Go from Dreux to Paris.
Rouen → Dreux → Paris → Reims
Rouen → Paris → Reims → shorter

Table: Taxi-hailing orders.

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Passenger</th>
<th>Driver</th>
<th>Start Point</th>
<th>End Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6:40 a.m.</td>
<td>B</td>
<td>E</td>
<td>Rouen</td>
<td>Reims</td>
</tr>
<tr>
<td>2</td>
<td>6:50 a.m.</td>
<td>C</td>
<td>?</td>
<td>Dreux</td>
<td>Paris</td>
</tr>
</tbody>
</table>

Figure: Geographical map.
Introduction

Figure: Geographical map.

Table: Taxi-hailing orders.

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Passenger</th>
<th>Driver</th>
<th>Start Point</th>
<th>End Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6:40 a.m.</td>
<td>B</td>
<td>E</td>
<td>Rouen</td>
<td>Reims</td>
</tr>
<tr>
<td>2</td>
<td>6:50 a.m.</td>
<td>C</td>
<td>E</td>
<td>Dreux</td>
<td>Paris</td>
</tr>
</tbody>
</table>

- **RDF**: Rouen→Paris→Reims
- **RDF+Relational Database**: Rouen→Dreux→Paris→Reims
Outline

- Introduction
- A Path Querying Language for Federation of RDF and Relational Database
- Expressiveness of FPQ
- Experiment and Evaluation
**Federated Path Querying Language (FPQ)**

\[
q(u, v) := \varphi \wedge \bigwedge_{i=1}^{n} (u_i, e_i, v_i);
\]

where

- \( q \) is the name of FPQ;
- \( \varphi \) is a conjunctive combination of relations;
- each \((u_i, e_i, v_i)\) for \( i \in \{1, \ldots, n\} \) is a NRE triple pattern.
Query:
At a certain time, whether a passenger take a ride?

\[ q(\ ?x, \ ?y) = [R(v) \land R(e)] \land [(\ ?x, exp_1, \ ?y) \land (\ ?x, exp_2, \ ?y) \land (\ ?x, exp_3, \ ?y)] \]

where

- \( R(v) := \text{Position}(\text{Time}, \ ?x, \ ?y, \ ?\text{driverId}) \);
- \( R(e) := \text{Orders}(\text{Time}, \text{lonUp}, \text{latUp}, \ ?x, \ ?y, \ ?\text{driverId}) \);
- \( \text{exp1} : \text{next}^{-1} :: \text{lon} \);
- \( \text{exp2} : \text{next} :: \text{lat} \);
- \( \text{exp3} : \text{next} :: \text{nd} / \text{next} :: \text{ref} / \text{next}^{-1} :: \text{id} \).
Outline

- Introduction
- Federated Path Querying Language (FPQ)
- Expressiveness of FPQ
- Experiment and Evaluation
Expressiveness of FPQ

| : Disjunctive operator;
N : Nesting operator;
∧ : Conjunctive operator;
R : Federated operator;
∨ : Union Operator.

The query evaluation of FCNRPQ:
Data complexity → Polynomial time
Combined complexity → NP-complete time
Outline

- Introduction
- Federation Path Querying Language
- Expressiveness of FPQ
- Experiment and Evaluation
  - Query and Result
  - Extention
Query

- **Query 1:**
  At a certain time, where did the passengers get on vehicles or off?

- **Query 2:**
  At a certain time, did the passengers visit tourist attractions on the map?

- **Query 3:**
  At a certain time, which roads did the passengers go down from the vehicles?
Query

- **Query 1:** At a certain time, where did the passengers get on vehicles or off?
- **Query 2:** At a certain time, did the passengers visit tourist attractions on the map?
- **Query 3:** At a certain time, which roads did the passengers go down from the vehicles?

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Passenger</th>
<th>Driver</th>
<th>Start Point</th>
<th>End Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6:40 a.m.</td>
<td>B</td>
<td>E</td>
<td>Rouen</td>
<td>Reims</td>
</tr>
<tr>
<td>2</td>
<td>6:50 a.m.</td>
<td>C</td>
<td>E</td>
<td>Dreux</td>
<td>Paris</td>
</tr>
</tbody>
</table>

Table: Taxi-hailing orders.
Result

The graph shows the query time for different queries as a function of the number of tuples in the relational database, measured in millions. The x-axis represents the number of tuples, while the y-axis represents the query time in milliseconds. The graph includes four different queries, labeled as Query-1, Query-2, Query-3, and Query. Each query is represented by a different symbol on the graph.
Extention

- Distributed environment to solve the memory overflow
- Streaming data to process the real time query
Q & A

- Jiahui Zhang
  zhangjiahui@tju.edu.cn

- Xiaowang Zhang
  xiaowangzhang@tju.edu.cn

- Zhiyong Feng
  zyfeng@tju.edu.cn

Thank you!