From Traversal to Dynamic Federation

Rethinking Link Traversal Query Processing via Subwebs and RDF Data Shapes

Bryan-Elliott Tam

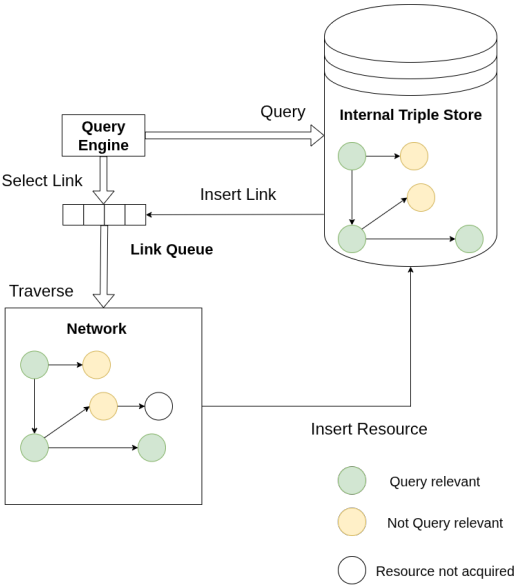
What Are We Trying to Achieve

Query the web of linked data like one unified database

Challenges

- **Decentralization**: No single endpoint, the data scattered across countless servers
- **Scale**: Too large to download completely
- **Data Quality**: Large portions are query-irrelevant or untrusted

Link Traversal Query Processing (LTQP)



Link Traversal Query Processing (LTQP) (ii)

- Challenges of LTQP
  - Performance Issues
    - Slow query execution
    - High network overhead
  - Trust & quality concerns
- Why use LTQP
  - Query unindexed networks
  - **Integrate loosely connected data networks**

Federated Queries And LTQP Similarities

- Involve a federation of *interfaces* (SPARQL, TPF, RDF Files)
- The federation *can be* dynamic
  - Service-Safeness concept for federated queries (Federating queries in SPARQL 1.1: Syntax, semantics and evaluation)
    - Example in the next slide
  - Reachability Criteria (Foundations of Traversal Based Query Execution over Linked Data)
- **Emulation of optimization strategies across querying models *may* be possible**
  - Requires a theoretical foundation

Federated Queries And LTQP Similarities (ii)

```
PREFIX ex: <http://example.org/>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?scientist ?birthPlace
WHERE {
  ?dataset a ex:Dataset ;
           ex:hasService ?outerService .

  SERVICE ?outerService {
    ?resource ex:providesInnerService ?innerService .

    SERVICE ?innerService {
      ?scientist a dbo:Scientist ;
                 dbo:birthPlace ?birthPlace .
    }
  }
}
```

Can FedQPL be a Foundation for LTQP

- Paper: FedQPL: A Language for Logical Query Plans over Heterogeneous Federations of RDF Data Sources (time following definitions and tables are from this paper)

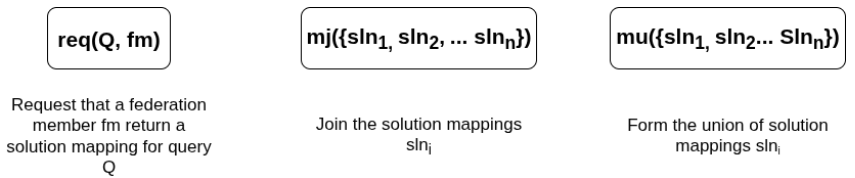


Figure 1: Summary of FedQPL operators as defined in Definition 4

- Notion of *interface*
  - Tied to a federation member
  - Defines which knowledge graph can be queried
  - Defines the query expressivity supported

Can FedQPL be a Foundation for LTQP (ii)

Two interpretations of LTQP

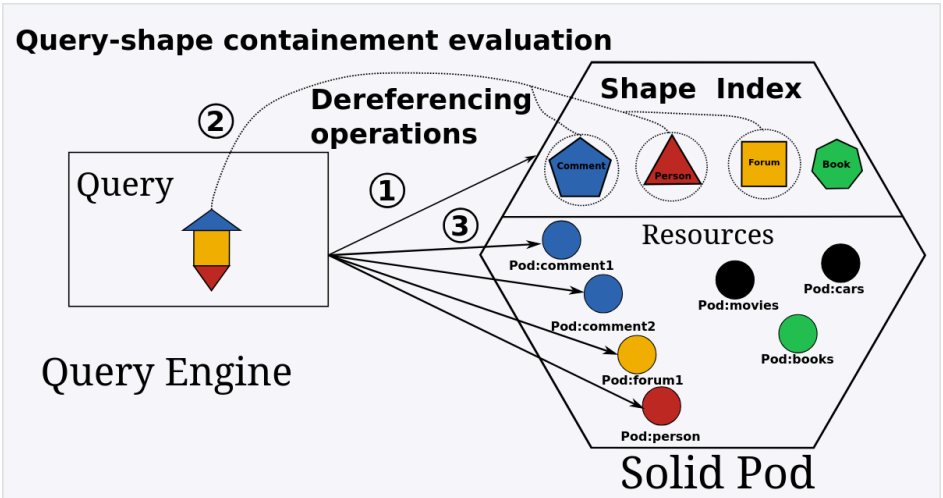
- **Stream-based Internal Querying** (current interpretation without FedQPL)
  - Query the internal triple store with Q in a streaming way
- **Virtual Resource Federation** (proposed interpretation with FedQPL)
  - Query the *virtual* resource (federation member)
  - **Current approach:** "exhaustive source assignment" (Definition 9)
  - **Future approaches:** emulating other source assignment algorithms in the literature

Can FedQPL be a Foundation for LTQP (iii)

Consideration

- Most strategies require statistics about federation members
  - The shape index could provide some of those statistics
  - Already been used to reduce the search space of LTQP
    - Opportunities for Shape-Based Optimization of Link Traversal Queries
    - Journal paper currently under submission
- FedQPL does not consider dynamic federations even for federated queries

Shape Index


Query-shape containement evaluation
Shape Index
Dereferencing operations
②
Query
①
③
Comment
Person
Forum
Book
Resources
Pod:comment1
Pod:movies
Pod:cars
Pod:comment2
Pod:books
Pod:forum1
Pod:person
Query Engine
Solid Pod

FedUp Approach

- Paper: (FedUP: Querying Large-Scale Federations of SPARQL Endpoints)
  - Designed to "[process] SPARQL queries over large federations of SPARQL endpoints"
  - "[O]nly a few combinations of sources may return result"
  - **Similar problem with LTQP**
    - **Previous research** reduce the search space (source selection)
    - **Current research** reduce the non-contributing join ("Result-Aware query plans")

Requirement

- Summary mechanism
  - To compute the "Result-Aware query plans"
- Shape index *could* serve as this summary

Plan

Formalization

- Extend FedQPL to consider dynamic federations
  - Federated queries
  - LTQP
  - Expanding plan
    - *Could* be a simpler particular case of adaptive plan
- Adapt FedUp for LTQP
  - Use the shape index as a summary mechanism
    - *Note:* This may already be addressed through FedQPL extensions, as FedQPL is the foundation of FedUp

Plan (ii)

Implementation

Static File Federation

- Experiment with FedUp algorithm using shape indexes within Comunica

Provenance Information in the Internal Triple Store

- Add sub-web and shape index provenance

Cache Algorithm

- Perform federated query first, then extend results with LTQP

Traversal Integration

- Use FedUp approach during link traversal with adaptive query planning

Plan (iii)

Considerations

- The separate RDF store by resource implementation of Comunica is significantly slower than the one store implementation
  - A refactoring will soon be done to address this issue
  - The one store implementation would require some “hacks” to implement the proposed approach

Evaluation

- We plan to use the SolidBench benchmark
  - Specifically designed for evaluating Link Traversal Query Processing
  - Based on LDBC SNB social network dataset
  - Includes shape index module
- Additional benchmarks or datasets could be interesting
- Evaluation metrics:
  - Query execution time
  - Query planning overhead

Plan (iv)

- First result arrival time
- Termination time (time between the last result and the end of the query)
- Waiting time (cumulative time between the arrival of two results)
- Diefficiency metric
- Ratio of query-relevant joins
- Theoretical metrics about query plan efficiency (to be explored)