From Traversal to Dynamic Federation

Rethinking Link Traversal Query Processing via Subwebs and RDF Data Shapes

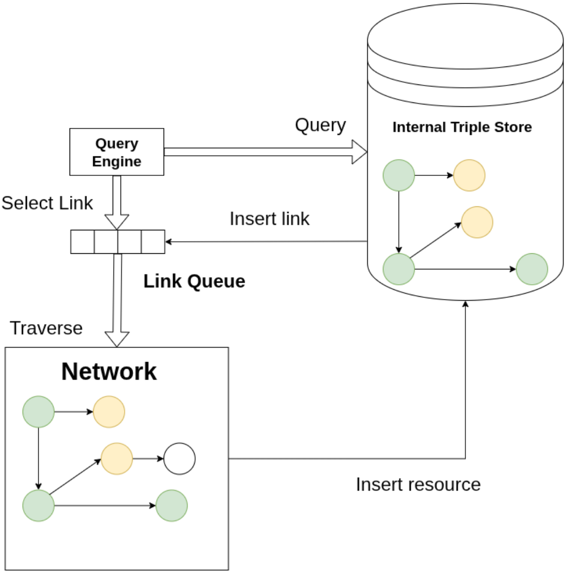
Bryan-Elliott Tam

Why Link Traversal Query Processing (LTQP)

- Challenges of LTQP
  - Performance Issues
    - Slow query execution
    - High network cost (many HTTP requests)
  - Difficult to determine result trustworthiness and quality

- Why use LTQP
  - Querying of unindexed networks
  - **Facilitate integration across multiple indexed data networks that are only loosely connected**

LTQP


Query
Internal Triple Store
Query Engine
Select Link
Insert link
Link Queue
Traverse
Network
Insert resource

Federated Queries and LTQP

- Both involve a federation of *interfaces* (SPARQL, TPF, Files)
- Both can involve dynamic federation with a finite number of member
  - Service-Safeness concept for federated queries (Federating queries in SPARQL 1.1: Syntax, semantics and evaluation)
  - Reachability Criteria (Foundations of Traversal Based Query Execution over Linked Data)
- We could potentially apply optimization strategies from the federated query world to LTQP

Federated Queries and LTQP (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 .
    }
  }
}
```

FedQPL

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

Definition 4. A **FedQPL expression** is an expression φ that can be constructed from the following grammar, in which *req*, *tpAdd*, *bgpAdd*, *join*, *union*, *mj*, *mu*, (, and) are terminal symbols, ρ is an expression in the request language L_{req} of some interface, *fm* is a federation member, tp is a triple pattern, B is a BGP, and Φ is a nonempty set of FedQPL expressions.

$$\varphi ::= \; \mathit{req}^{\rho}_{\mathit{fm}} \;\mid\; \mathit{tpAdd}^{tp}_{\mathit{fm}}(\varphi) \;\mid\; \mathit{bgpAdd}^{B}_{\mathit{fm}}(\varphi) \;\mid\; \mathit{join}(\varphi,\varphi) \;\mid\; \mathit{union}(\varphi,\varphi) \;\mid\; \mathit{mj}\,\Phi \;\mid\; \mathit{mu}\,\Phi$$

Before we present the formal semantics of FedQPL expressions, we provide an intuition of the different operators of the language.

Definition 1. An **RDF data access interface** (**interface**) I is a tuple $(L_{\text{req}}, \varrho)$ where L_{req} denotes a language and ϱ is a function that maps every pair (ρ, G), consisting of an expression ρ in L_{req} and an RDF graph G, to a set of solution mappings.

LTQP Query Plan Using the FedQPL Model

Two interpretations of LTQP:

- **Stream-based Internal Querying** (current interpretations)
  - Query the internal triple store with Q in a streaming way
- **Virtual Resource Federation**
  - Query the *virtual* resource (federation member)
  - **Current approach** Engine performs “exhaustive source assignment” (Definition 9)
  - Enables investigation of source assignment strategies
  - Most strategies require statistics about federation members
    - The shape index could provide some of those statistics

Definition 9. Let $F = \{fm_1, fm_2, \dots, fm_m\}$ be a federation that is triple pattern accessible, and let $B = \{tp_1, tp_2, \dots, tp_n\}$ be a BGP. The **exhaustive source assignment** for B over F is the following source assignment.

$$mj\left\{\ mu\{req_{fm_1}^{tp_1}, \dots, req_{fm_m}^{tp_1}\}\ , \dots,\ mu\{req_{fm_1}^{tp_n}, \dots, req_{fm_m}^{tp_n}\}\ \right\}$$

LTQP Query Plan Using the FedQPL Model (ii)

federation engine	source assignment	sa-cost
FedX [26]	$mj\left\{ req_{fm_{\text{drb}}}^{\{tp_1, tp_2\}}, req_{fm_{\text{kegg}}}^{\{tp_3, tp_4\}}, mu\{req_{fm_{\text{drb}}}^{tp_5}, req_{fm_{\text{kegg}}}^{tp_5}, req_{fm_{\text{dbp}}}^{tp_5}\} \right\}$	5
SemaGrow [7]	$mj\left\{ req_{fm_{\text{drb}}}^{\{tp_1, tp_2\}}, req_{fm_{\text{kegg}}}^{tp_3}, mu\{req_{fm_{\text{kegg}}}^{tp_4}, req_{fm_{\text{chebi}}}^{tp_4}\}, mu\{req_{fm_{\text{kegg}}}^{tp_5}, req_{fm_{\text{chebi}}}^{tp_5}\} \right\}$	6
CostFed [23]	$mj\left\{ req_{fm_{\text{drb}}}^{\{tp_1, tp_2\}}, req_{fm_{\text{kegg}}}^{\{tp_3, tp_4, tp_5\}} \right\}$	2

Table 1: Source assignments for FedBench query LS6 by different federation engines.

FedUp Approach

- **FedUp Framework** (FedUP: Querying Large-Scale Federations of SPARQL Endpoints)
  - Designed to "process SPARQL queries over large federations of SPARQL endpoints"
  - Shows similar analogy with LTQP approach

PROBLEM 1 (BGP-AWARE SOURCE SELECTION [20]). *Let Q be a SPARQL query and F a federation, find the minimal set of federation members $R(tp) \subseteq F$ for each triple pattern $tp \in Q$ where $\forall (G, I) \in R(tp), \exists \mu \in [\![Q]\!]_F$ such that $\mu(tp) \in G$.*

Example 3 (Unions-over-joins logical plans). Using the results of $S6$ over F_1, an alternative to $S6_j$ is the unions-over-joins FedQPL expression $S6_u$ depicted in Figure 3b:

$$S6_u = mu\,\{mj\,\{req_{D_1}^{tp1}, req_{D_1}^{tp2}, req_{D_2}^{tp3}, req_{D_2}^{tp4}\},\; mj\,\{req_{D_3}^{tp1}, req_{D_3}^{tp2}, req_{D_4}^{tp3}, req_{D_4}^{tp4}\}\}$$

FedUp Approach (ii)

- **Key Requirements**
  - Necessitates a summary mechanism
  - Shape index *could* serve as this summary

FedUP introduces a vicious cycle: its source selection requires query results, and query results require computing source selection. To tackle this issue, we execute Algorithm 1 and $sols(\varphi)$ on a tiny quotient summary [4, 5] of the federation.

Definition 3.2 (Quotient RDF summary [5]). Given an RDF graph G and an RDF node equivalence relation ψ, the summary of G by ψ, which is an RDF graph denoted $\psi(G)$, is the quotient of G by ψ.

Plan

Formalization

- Make FedQPL Dynamic
  - Data source discovery in federated queries
  - LTQP reachability integration
    - Expanding plan vs adaptative plan
- Describe the FedUp algorithm with the shape index
  - Describe algorithm with shape index integration

Implementation

Static File Federation

- Experiment with Fedup algorithm using shape indexes inside of Comunica

Provenance Information in the Internal Triple Store

- Add subweb provenance to triples
- Store subweb shape index information in engine

Plan (ii)

Cache Algorithm

- Perform federated query first, then extend results with LTQP

Traversal integration

- Use Fedup approach during link traversal with adaptative query planning