

Distributed Semantic Query Tool and Query PreProcessor



Janne Saarela, Pasi Tiitinen (Profium)

Philipp Obermeier, Sebastian Dill (FUB)

15/05/09



- Year 2 and reviewers' comments
- Work achieved in year 3
 - Distributed Semantic Query Tool
 - Query PreProcessor

- In year 3 we developed software to
 1. assist developers to make use of Triple Space technology more easily and
 2. optimize query evaluation in a distributed environment
- Following issue was raised at the year 2 review:
 - *While the appropriateness of the [query evaluation] cost model still has to be shown, a well designed cost model will be of value not only in the context of the Tripcom project, but can also provide benefits to other developers of distributed RDF infrastructures.*
 - *-> the cost model has been implemented in the Query PreProcessor*

- Requirements for the tool:
 - must support activities of the Triple Space developers by allowing its users to write valid SPARQL queries.
 - must allow its users to evaluate SPARQL queries against one or many triple space(s).
 - must allow its users to visualize query results in an environment where the query expression can be further edited and evaluated.
 - the constructed query expression and resulting query results must both be storable in standard format in local hard drive for later re-use.
 - the query execution cost may be explained to the user prior to its execution.
 - should show metadata about the Triple Spaces being queried.

Distributed Semantic Query Tool (DSQT)



The screenshot displays the Tripcom query tool interface in a Windows Internet Explorer browser window. The interface is divided into several sections:

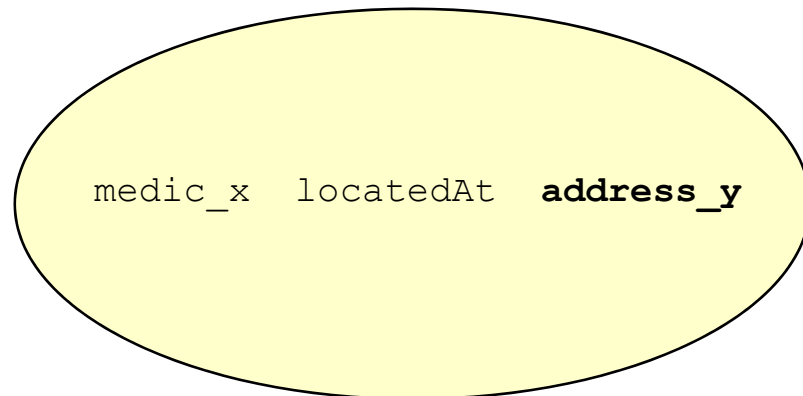
- Spaces:** A tree view showing a hierarchy of spaces. The root space is 'root'. Under 'root', there are three main branches: 'tsc://deri.at', 'tsc://profium.com', and 'tsc://profium.com/medical'. The 'tsc://deri.at' branch is expanded, showing sub-spaces like 'tsc://deri.at/medical' and 'tsc://deri.at/EAI'.
- Select query type:** A section with radio buttons for selecting a query type: SELECT, CONSTRUCT, ASK, and DESCRIBE. A link for [Information about query types](#) is also present.
- Table:** A table with columns 'Name' and 'Prefix'. It contains three rows: 'Dublin Core' with prefix 'dc', 'Foaf' with prefix 'foaf', and 'Atom ontology' with prefix 'atom'. Below the table, there are buttons for 'Lang', 'Type', and 'Add', and a 'Delete' button next to a row containing '?y'.
- Options:** A section with input fields for 'Limit' (set to 20), 'Offset', and 'Distinct' (checkbox). There are buttons for 'Define' (next to 'Order') and 'Get costs'.
- Query:** A text area containing the following SPARQL query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT { ?x foaf:knows ?y }
WHERE { ?x foaf:knows ?y . ?x foaf:firstName john }
LIMIT 20
```
- Define ordering dialog:** A small dialog box titled 'Define ordering' with a 'Field' dropdown, 'Order' radio buttons for 'Ascending' (selected) and 'Descending', and 'Add', 'Ok', and 'Cancel' buttons.

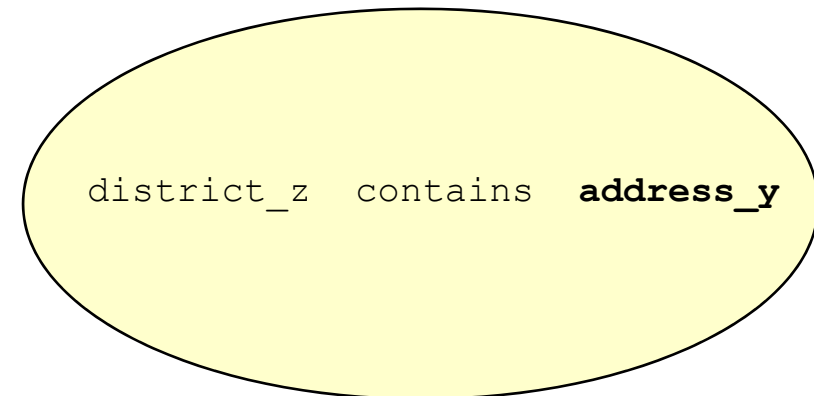
- Thin-client application developed using Google Web Toolkit (GWT)
- A wizard approach for creating a SPARQL query which can be 'explained' and 'executed' much in the same way as in SQL based environments.
- Triple Space specific features allow the user to understand the space structure against which the query is executed.
- DSQT makes use of Query PreProcessor...

- A Triple Space System comprises several machines (kernels) hosting RDF data sets (triple spaces) with SPARQL endpoint (TS API, TS Adapter, SPARQL CONSTRUCT queries as rd templates)
- Related data is often located in different spaces hosted on separated kernels

Space A: medics



Space B: addresses

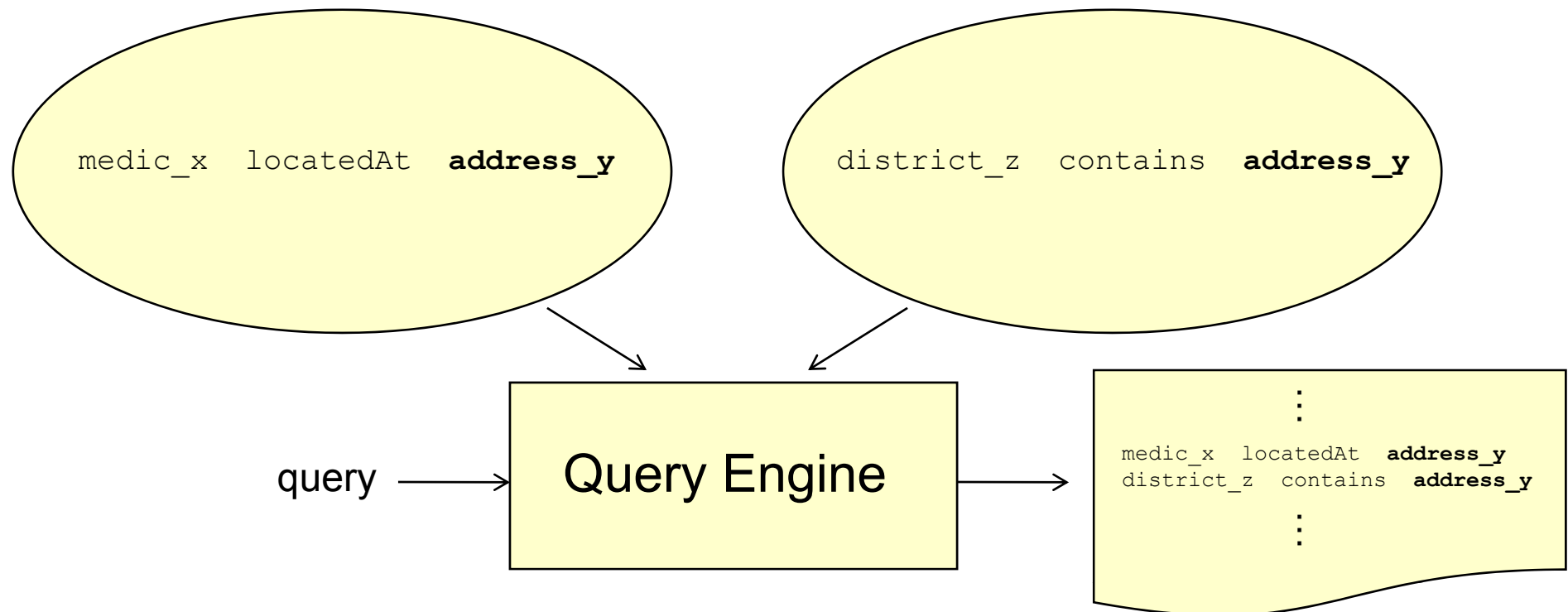


- However: 'rd without space' is evaluated against one space at most

Engine for federated querying upon the integrated RDF data set of multiple RDF data sets with SPARQL endpoint (here spaces)

Space A: medics

Space B: addresses



Query Preprocessor (QPP), a federated query engine for SPARQL:

- provides the means to **distribute the evaluation of a query across multiple RDF data sets with SPARQL endpoint**, optimized for execution time
- **splits query into sub-queries** and sends them to SPARQL endpoints (detected by DHT overlay) for execution.

Significant aspects on **query planning, optimization** and **execution**:

- Data sets **indexed** by DHT overlay adopted from Distribution Manager
- **Cost estimation** based on RDF stats for each data set
- **Intra-query adaptiveness**, i.e., query planning and execution interplay
 - **Planning part**: partitioning of queries (BGPs) into sub-queries controlled by connectivity and existence of adequate data sets (i.e., the ones, which can resolve the searched triple patterns while yielding acceptable costs)
 - **Execution part**: shipping of selected sub-queries to selected data sets for evaluation, return of results to QPP
- **Trade-off of perfect recall against performance**: Choice of single data set with median amount of solutions instead of all adequate data sets; Triple Space System allows non-perfect recall

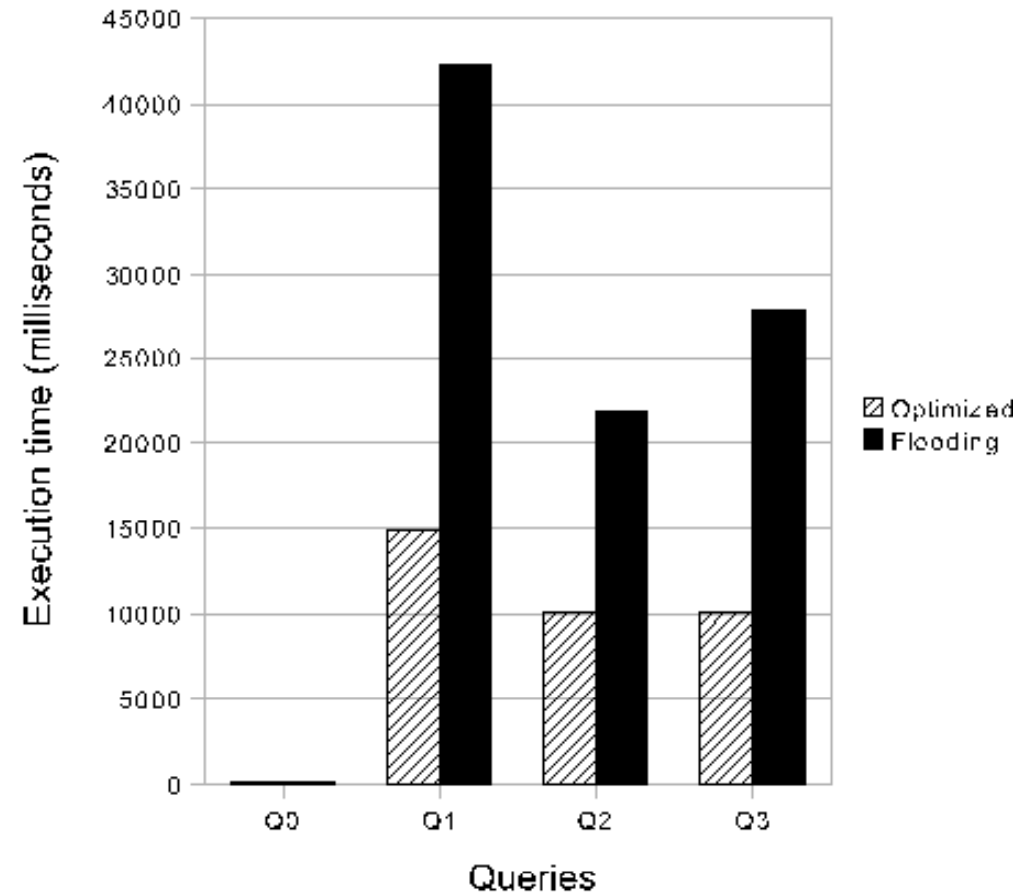
eHealth evaluation scenario:

- Global data set with 13.5 million triples split up into 12 data sets: 3 exclusive data sets for each concept (medics, treatments, drugs, districts) were distributed to 4 machines
- These 4 machines (1 CPU Dual Core, 2GB Ram, Debian Linux) each hosted 3 data sets with a SPARQL endpoint (i.e. 3 instances of Open Link Virtuoso on each machine)
- e.g. test query 1

```
SELECT ?med ?treatment
WHERE { ?med medic:locatedAt ?address .
  district:district_1 district:contains ?address .
  ?med medic:provides ?treatment . }
```

where medic and district objects are only retrievable from different endpoints

- Queries can now be evaluated against the integration of multiple data sets
- Tremendous performance gain in contrast to a naive distributed querying approach (i.e. asking all adequate data sets)
- Performance gain is available outside Triple Space environment for existential queries.



QPP

- provides the means for federated adaptive query processing across SPARQL endpoints and, thus, across triplespaces
- highly optimized for distributed querying scenarios in which only a small fraction, i.e., a few solution mappings, of the perfect recall is sufficient; acts much more efficiently than naive approaches for querying multiple data sources/spaces

Future research on

- further advanced methods for federated querying of SPARQL endpoints