

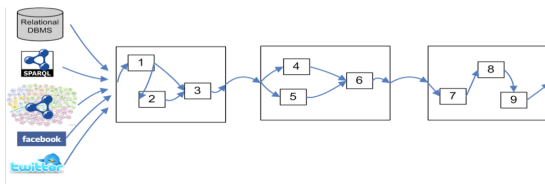
The RDFGears Project

Jan Hidders

WISer Day 2012

June 12, 2012

Goal of RDFGears Project



Goal: building a language and system for RDF transformations

- XML: XSLT & XQuery, RDF: ?? & SPARQL
- Visual workflows, easy to use, extensible
 - ▶ Inspired by scientific workflow systems: Taverna, Kepler, Triana, Galaxy, ...
- Well-founded and well-understood language allowing research into:
 - ▶ Scalability, optimization, provenance, ...
- Applications: (RDF) Data
 - ▶ transformation
 - ▶ integration and enrichment
 - ▶ publication

Workflow Languages for RDF transformation

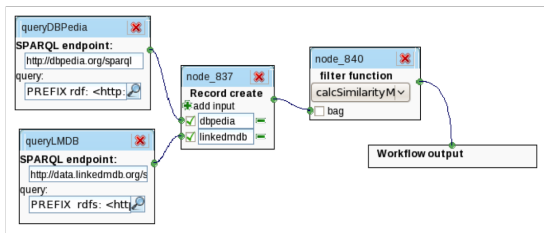
What is a suitable (graphical) workflow language?

- Expressive power, as graph transformation language
- Understandability and Usability
- Versatility and Extendability
- Optimizability

Our solution:

- Start from Nested Relational Algebra with a graphical notation
- Add basic types: Literal, URL, RDF-Graph
- Use SPARQL for (1) retrieval and (2) graph transformation/creation

The RDFGears Workflow Language



- A **SPARQL endpoint** processor produces a bag of tuples.
 - ▶ Missing variables in binding are represented by null values.
- A **Record create** processor creates a record.
 - ▶ Input ports can be marked as “iterating” which means it consumes a bag rather than a single value and iterates over that bag
 - ▶ If multiple ports are marked, cartesian product is computed.
- A **Filter** processor selects elements from a bag based on a user-defined function or earlier created workflow.

Nested Relational Calculus/Algebra operators

The screenshot shows the RDF Gears tool interface in Mozilla Firefox. The main window displays a workflow diagram with several operators connected by arrows. The operators include:

- queryDBPedia**: SPARQL endpoint: `http://localhost:2020/s`, query: `PREFIX rdf: <http://`
- queryLMB**: SPARQL endpoint: `http://localhost:2020/s`, query: `fs:label ?label. }}`
- nestResults**: Group, bag_of_records, group by: mov
- matchMovies**: function: findBestMatch, mov_group, dir_group_bag
- verifyAccept**: categorizer, bag, add category, accept, verify

The workflow diagram shows data flow from the query operators through the nestResults and matchMovies operators, then through the verifyAccept operator, and finally to an output node labeled "ow output".

The SPARQL query in the bottom panel is:

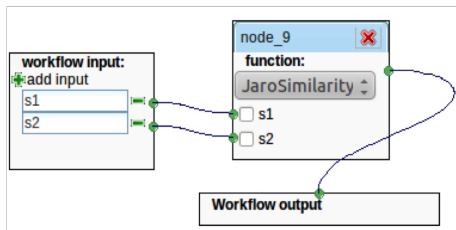
```

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX movie: <http://data.linkedmdb.org/resource/movie/>
SELECT DISTINCT ?dir ?label ?dir_name
FROM <file:Data/lmdb.n3>
WHERE {
  ?dir rdf:type movie:director .
  OPTIONAL {
    ?dir movie:director_name ?dir_name .
  }
  OPTIONAL {

```

- Usual operators from NRC/NRA available, plus short-hands.

Workflows published as web services



- All user-defined workflows can be published as web service.
- <http://wis.ewi.tudelft.nl/rdfgears/myworkflow?s1=Good&s2=Goed>
- Output is RDF graph or xml-encoded nested relational value.

Ongoing and Future Research Topics

- Streaming main-memory implementation.
- Typing.
- Disk-based implementation.
- Query optimization by graph rewriting
- Distributed query optimization
 - ▶ Leverage NRA/NRC rewriting rules
 - ▶ Eliminate and optimize SPARQL queries
 - ▶ Allow virtual SPARQL endpoint publishing of workflow results
- MAPREDUCE mapping of RDFGears Workflow Language
 - ▶ In general interesting question for NRC/NRA
 - ▶ What is a good mapping? What is a complete mapping?
 - ▶ Also related to already existing similar languages: PIG, HIVE
- Provenance functionality
 - ▶ Differences between NRA and NRC?
 - ▶ Why provenance. Why-not provenance.