ARQ - A SPARQL Processor for Jena

Mehdi Allahyari
Shima Dastgheib

CSCI 8370 Advanced Database Systems
University of Georgia, Spring 2012

ARQ - Application API

- ARQ is a query engine for Jena that supports the SPARQL
- The application API is in the package com.hp.hpl.jena.query.
- Other packages contain various parts of the system (execution engine, parsers, testing etc)
- Most applications will only need to use the main package.
- Only applications wishing to programmatically build queries or modify the behavior of the query engine need to use the others packages directly.

Key Classes ARQ Application API

- The package com.hp.hpl.jena.query is the main application package.
- Query
  - a class that represents the application query
  - It is a container for all the details of the query.
  - Objects of class Query are normally created by calling one of the methods of QueryFactory methods which provide access to the various parsers.

Key Classes ARQ Application API

- QueryExecution
  - represents one execution of a query.
- QueryExecutionFactory
  - a place to get QueryExecution instances
- DatasetFactory
  - a place to make datasets, including making a DataSource (an updatable Dataset)

Key Classes ARQ Application API

- For SELECT queries:
  - QuerySolution - A single solution to the query
  - ResultSet - All the QuerySolutions. An iterator.
  - ResultSetFormatter - turn a ResultSet into various forms; into text, into an RDF graph (Model, in Jena terminology) or as plain XML

SELECT queries

- The basic steps in making a SELECT query are outlined in the example below:
- A query is created from a string using the QueryFactory.
- The query and model or RDF dataset to be queried are then passed to QueryExecutionFactory to produce an instance of a query execution.
- Result are handled in a loop and finally the query execution is closed.

SELECT queries

```
import com.hp.hpl.jena.query.* ;

Model model= ... ;

String queryString= " .... " ;

Query query= QueryFactory.create(queryString) ;

QueryExecutionqexec= QueryExecutionFactory.create(query, model) ;

try {ResultSetresults = qexec.execSelect() ;

for ( ; results.hasNext() ; )

{QuerySolutionsoln= results.nextSolution() ;

RDFNodex = soln.get("varName") ; // Get a result variable by name.

Resource r = soln.getResource("VarR") ; // Get a result variable -must be a resource

Literal l = soln.getLiteral("VarL") ; // Get a result variable -must be a literal

}}finally { qexec.close() ; }
```

ARQ - Extending Query Execution

- There are mechanisms that can be used to extend and modify query execution within ARQ.
- Through these mechanisms, ARQ can be used to query different graph implementations and to provide different query evaluation and optimization strategies for particular circumstances.
- These mechanisms are used by SDB and TDB
- ARQ can be extended in various ways to incorporate custom code into a query.

ARQ - Extending Query Execution

- Custom filter functions and property functions provide ways to add application specific code.
- free text search capabilities, using Apache Lucene, are provided via a property function.
- ARQ can be extended at the basic graph matching or algebra level.

ARQ Query Processing

- sequence of actions performed by ARQ to perform a query:
  - Parsing
  - algebra generation
  - execution building
  - high-level optimization
  - low-level optimization
  - Evaluation
- not usual to modify the parsing step from the parse tree to the algebra form. It is a fixed algorithm defined by the SPARQL standard.

ARQ Query Processing

- Extensions can modify the algebra form by transforming it from one algebra expression to another, including introducing new operators.
- **Parsing:**
  - turns a query string into a Query object.
  - class **Query** represents the abstract syntax tree (AST) for the query and provides methods to create the AST, primarily for use by the parser.
  - query object also provides methods to serialize the query to a string.

ARQ Query Processing

- Because this is the AST, the string produced is very close to the original query with the same syntactic elements, but without comments, and formatted with a whitespace for readability.
- The query object can be used many times.
- not modified once created
- not modified by query execution

ARQ Query Processing

- **Algebra generation:**
  - ARQ generates the SPARQL algebra expression for the query.
  - An algorithm in the SPARQL specification for translating a SPARQL query string, as held in a Query object into a SPARQL algebra expression.

ARQ Query Processing

For example, the query:

```
PREFIX foaf: http://xmlns.com/foaf/0.1/
SELECT ?name ?mbox ?nick
WHERE { ?x foaf:name ?name ;
    foaf:mbox ?mbox .
    OPTIONAL { ?x foaf:nick ?nick }
    }
```

Becomes:

```
(prefix ((foaf: <http://xmlns.com/foaf/0.1/>))
    (project (?name ?mbox ?nick)
        (leftjoin
            (bgp
                (triple ?x foaf:name ?name)
                (triple ?x foaf:mbox ?mbox)
            )
            (bgp (triple ?x foaf:nick ?nick)
            )
    )))
```

- http://sparql.org/query-validator.html

ARQ Query Processing

- **High-Level Optimization and Transformations:**
  - collection of transformations that can be applied to the algebra like replacing equality filters with a more efficient graph pattern.
  - query processor for a custom storage layout can choose which optimizations are appropriate and can also provide its own algebra transformations.
  - A transform is code that converts an algebra operation into other algebra operations using the Transformer class:
    - Op op = ... ;
    - Transform someTransform = ... ;
    - op = Transformer.transform(someTransform, op) ;

ARQ Query Processing

- The Transformer class applies the transform to each operation in the algebra expression tree.
- Transform is an interface, with one method signature for each operation type, returning a replacement for the operator instance it is called on.
- Transformations proceed from the bottom of the expression tree to the top

ARQ Query Processing

- **Low-Level Optimization and Evaluation:**
  - The step of evaluating a query is the process of executing the algebra expression, as modified by any transformations applied, to yield a stream of pattern solutions.
  - Low-level optimizations include choosing the order in which to evaluate basic graph patterns.
  - Low-level optimization can be carried out dynamically as part of evaluation.
  - Internally, ARQ uses iterators extensively.
  - Evaluating an algebra expression produces a iterator of query solutions.

References

- http://openjena.org/ARQ/arq-query-eval.html
- http://incubator.apache.org/jena/documentation/query/app_api.html
- http://openjena.org/ARQ/algebra.html
- http://jena.sourceforge.net/ARQ/extension.html