SPARQL Implementations

Andy Seaborne

SPARQL Implementations

1. ARQ – Complete, general purpose query system
   - SPARQL Parser and serializer
   - SPARQL Algebra
   - SPARQL Execution
   - Results handling
2. SDB – Specialised ARQ extension
   - SPARQL to SQL rewriter

ARQ
http://jena.sf.net/ARQ

Execution Issues

- Transformations:
  - Query String => Algebra expression
  - Algebra => Execution Plan
  - Execution plan => solutions
  - Solutions => query results form
- Stream-based
  - Transformation possible for majority of queries
  - Multi-sets as iterators : ToList is a no-op

Linearization

- Where possible execution is by one stage extending/removing results of previous stage
- Indexing = Substitution
- Exceptions:
  - Nested optionals with locally free variables
    - Non "well-designed" patterns
  - Join/LeftJoin : Out of scope variables in filters
  - These are done bottom-up for correctness


```
PREFIX :    <http://example/>
SELECT *
{ :x1 :p ?v .
  OPTIONAL {
      :x3 :q ?w .
      OPTIONAL { :x2 :p ?v } } }
```

```
PREFIX :    <http://example/>
SELECT *
{ :x1 :p ?v .
  { :x2 :q ?w . FILTER( ?v + ?w < 5 ) }
}
```

SDB

http://jena.svn.sf.net/viewvc/jena/SDB/

SDB Table layouts

- Layout 1
  - Single triple table, RDF terms encoded into the entries
  - c.f. Jena's RDB layout
  - Optimal for fine-grain API use
- Layout 2
  - Triples table ; Quads table ; Nodes table
  - Better for plain SPARQL queries
  - Id and hash forms
- Layout 2+
  - Cached partial queries
  - Inference support
  - (values)

Layout2 / hash variant / single graph

```
CREATE TABLE Triples (
    s BIGINT NOT NULL,
    p BIGINT NOT NULL,
    o BIGINT NOT NULL,
    PRIMARY KEY (s, p, o)
    )

CREATE INDEX PredObj ON Triples (p, o)
CREATE INDEX ObjSubj ON Triples (o, s)


CREATE TABLE Node (
   hash BIGINT NOT NULL,
   lex TEXT NOT NULL,
   lang varchar NOT NULL default '',
   datatype varchar(200) NOT NULL default '',
   type integer NOT NULL default '0',
   PRIMARY KEY (hash)
   )

CREATE UNIQUE INDEX Hash ON Nodes (hash)
```


Example 1

```
PREFIX person: <http://example/person/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?nick
WHERE
{ ?x foaf:name "Fred" .
  ?x foaf:nick ?nick . }
```

```
SELECT                                  -- V_1=?nick
  R_1.lex AS V_1_lex, R_1.datatype AS V_1_datatype,
  R_1.lang AS V_1_lang, R_1.type AS V_1_type
FROM
    Triples AS T_1                      -- ?x foaf:name "Fred"
  INNER JOIN
    Triples AS T_2                      -- ?x foaf:nick ?nick
  ON ( T_1.p = -2290624521842110797     -- Const: <http://xmlns.com/foaf/0.1/name>
   AND T_1.o = 6622531991636827042      -- Const: "Fred"
   AND T_2.p = 5173304175992580252      -- Const: <http://xmlns.com/foaf/0.1/nick>
   AND T_1.s = T_2.s                    -- Join var: ?x
   )
  LEFT OUTER JOIN
    Nodes AS R_1                        -- Var: ?nick
  ON ( T_2.o = R_1.hash )
```

Example 2

```
PREFIX :    <http://example/>
SELECT *
{ :x1 :p ?v .
  OPTIONAL {
      :x3 :q ?w .
      OPTIONAL { :x2 :p ?v } } }
```

```
(leftjoin
  (BGP [triple :x1 :p ?v])
  (leftjoin
    (BGP [triple :x3 :q ?w])
    (BGP [triple :x2 :p ?v])))
```


Example 2

```
SELECT                                  -- V_1=?v V_2=?w
  R_1.lex AS V_1_lex, R_1.datatype AS V_1_datatype, R_1.lang AS V_1_lang, R_1.type AS V_1_type,
  R_2.lex AS V_2_lex, R_2.datatype AS V_2_datatype, R_2.lang AS V_2_lang, R_2.type AS V_2_type
FROM
    ( SELECT *
      FROM Triples AS T_1               -- :x1 :p ?v
      WHERE ( T_1.s = -7272111352983262523 -- Const: <http://example/x1>
         AND T_1.p = 2004134117598721274 -- Const: <http://example/p>
        )
    ) AS T_1
  LEFT OUTER JOIN
    (   ( SELECT *
          FROM Triples AS T_2           -- :x3 :q ?w
          WHERE ( T_2.s = 4693521611208290624 -- Const: <http://example/x3>
             AND T_2.p = -4884978200120352820 -- Const: <http://example/q>
             )
        ) AS T_2
      LEFT OUTER JOIN
        Triples AS T_3                  -- :x2 :p ?v
      ON ( T_3.s = -6898947185675171362  -- Const: <http://example/x2>
       AND T_3.p = 2004134117598721274   -- Const: <http://example/p>
       )
    )
  ON ( ( ( T_3.o IS NULL ) OR ( T_1.o = T_3.o ) ) -- Join var: ?v
   )
  LEFT OUTER JOIN
    Nodes AS R_1                        -- Var: ?v
  ON ( T_1.o = R_1.hash )
  LEFT OUTER JOIN
    Nodes AS R_2                        -- Var: ?w
  ON ( T_2.o = R_2.hash )
```

Example 3

```
PREFIX :    <http://example/>

SELECT *
{
    ?x :p ?v .
    OPTIONAL { ?x :p ?a }
    OPTIONAL { ?x :q ?a }
}
```

```
(leftjoin
  (leftjoin
    (BGP [triple ?x :p ?v])
    (BGP [triple ?x :p ?a])
  )
  (BGP [triple ?x :q ?a])
)
```


Example 3

```
SELECT                                  -- V_1=?x V_2=?v V_3=?a
  R_1.lex AS V_1_lex, R_1.datatype AS V_1_datatype, R_1.lang AS V_1_lang, R_1.type AS V_1_type,
  R_2.lex AS V_2_lex, R_2.datatype AS V_2_datatype, R_2.lang AS V_2_lang, R_2.type AS V_2_type,
  R_3.lex AS V_3_lex, R_3.datatype AS V_3_datatype, R_3.lang AS V_3_lang, R_3.type AS V_3_type
FROM
    ( SELECT COALESCE(T_2.o, T_3.o) AS VC_1, T_1.o AS VC_2, T_1.s AS VC_3
        FROM
          ( SELECT *
            FROM Triples AS T_1          -- ?x :p ?v
            WHERE ( T_1.p = 2004134117598721274 -- Const: <http://example/p>
                  )
          ) AS T_1
        LEFT OUTER JOIN
          Triples AS T_2                 -- ?x :p ?a
        ON ( T_2.p = 2004134117598721274 -- Const: <http://example/p>
         AND T_1.s = T_2.s               -- Join var: ?x
         )
        LEFT OUTER JOIN
          Triples AS T_3                 -- ?x :q ?a
        ON ( T_3.p = -4884978200120352820 -- Const: <http://example/q>
         AND T_1.s = T_3.s               -- Join var: ?x
         AND ( ( T_2.o IS NULL ) OR ( T_2.o = T_3.o ) ) -- Join var: ?a
         )
      ) AS M_1                          -- ?a as M_1.VC_1, ?v as M_1.VC_2, ?x as M_1.VC_3
    LEFT OUTER JOIN
      Nodes AS R_1                      -- Var: ?x
    ON ( M_1.VC_3 = R_1.hash )
    LEFT OUTER JOIN
      Nodes AS R_2                      -- Var: ?v
    ON ( M_1.VC_2 = R_2.hash )
    LEFT OUTER JOIN
      Nodes AS R_3                      -- Var: ?a
    ON ( M_1.VC_1 = R_3.hash )
```