The SPARQL Query Language
The SPARQL Query Language
Agenda

1. Recap

2. SPARQL 1.1 Query Extensions
   - Expressions in Selection and Bindings
   - Aggregates
   - Subqueries
   - Property Paths
   - Negation

3. SPARQL Protocol

4. SPARQL Update

5. SPARQL Service Descriptions

6. Summary
Agenda

1. Recap
2. SPARQL 1.1 Query Extensions
   - Expressions in Selection and Bindings
   - Aggregates
   - Subqueries
   - Property Paths
   - Negation
3. SPARQL Protocol
4. SPARQL Update
5. SPARQL Service Descriptions
6. Summary
Example Pattern

Example

```prefix
PREFIX ex: <http://example.com/>
```

```
{ ?book ex:price ?price
  FILTER (?price < 15)
  OPTIONAL { ?book ex:title ?title }
  { ?book ex:author ex:Shakespeare } UNION
  { ?book ex:author ex:Marlowe } }
```
Translation into SPARQL Algebra

\[
\text{Filter}(\textit{price < 15}, \text{Join(}
\text{LeftJoin(Join(}
\text{Bgp(}\textit{book <http://eg.org/price> ?price})},
\text{Bgp(}\textit{book <http://eg.org/title> ?title}),
\text{true}),
\text{Union(Bgp(}\textit{book <http://eg.org/author>}
\text{<http://eg.org/Shakespeare>}),
\text{Bgp(}\textit{book <http://eg.org/author>}
\text{<http://eg.org/Marlowe>))))}
\]
Simplification of the SPARQL Algebra

Filter(?price < 15,
   Join(  
     LeftJoin(Bgp(?book <http://eg.org/price> ?price),  
     true),  
   Union(Bgp(?book <http://eg.org/author>  
     <http://eg.org/Shakespeare>),  
     Bgp(?book <http://eg.org/author>  
     <http://eg.org/Marlowe>))))
Semantics of the SPARQL Algebra

Operators

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bgp($P$)</td>
<td>match/evaluate pattern $P$</td>
</tr>
<tr>
<td>Join($M_1$, $M_2$)</td>
<td>conjunctive join of solutions $M_1$ and $M_2$</td>
</tr>
<tr>
<td>Union($M_1$, $M_2$)</td>
<td>union of solutions $M_1$ with $M_2$</td>
</tr>
<tr>
<td>LeftJoin($M_1$, $M_2$, $F$)</td>
<td>optional join of $M_1$ with $M_2$ with filter constraint $F$ (true if no filter given)</td>
</tr>
<tr>
<td>Filter($F$, $M$)</td>
<td>filter solutions $M$ with constraint $F$</td>
</tr>
<tr>
<td>$Z$</td>
<td>empty pattern (identity for join)</td>
</tr>
</tbody>
</table>
Agenda

1 Recap

2 SPARQL 1.1 Query Extensions
   • Expressions in Selection and Bindings
   • Aggregates
   • Subqueries
   • Property Paths
   • Negation

3 SPARQL Protocol

4 SPARQL Update

5 SPARQL Service Descriptions

6 Summary
Expressions in the Selection and Bindings

Solutions can be extended by evaluated expressions with \((\text{expression AS}\ ?\text{var})\) used for the assignment:

- In the \text{SELECT} clause
- In the \text{GROUP BY} clause
- Within \text{BIND} in a group graph pattern

Solutions from a group can further be joined with solutions given via \text{VALUES}
Example BIND (without Prefix Declarations)

**Data**

\[
\text{ex:Book ex:title "SPARQL Tutorial" ; ex:price 42 ; ex:discount 10 .}
\]

**Query**

\[
\text{SELECT ?title ?price WHERE}
\]
\[
\text{\{ ?b ex:title ?title ; ex:price ?p ; ex:discount ?r}
\]
\[
\text{BIND ((?p-?r) AS ?price) \}}
\]

**Result**

\[
?\text{title} \rightarrow \text{"SPARQL Tutorial"}, ?\text{price} \rightarrow 32
\]

\[\rightsquigarrow \text{ Algebra: Extend(Bgp(...), ?price, (?p-?r))}\]
Example SELECT Expressions (without Prefix Declarations)

Data

Query
SELECT ?title ((?p-?r) AS ?price) WHERE

Result
?title \mapsto "SPARQL Tutorial", ?price \mapsto 32

\Rightarrow \text{ Algebra: } \text{Extend(Bgp(...), ?price, (?p-?r))}
Example VALUES

Data

ex:Book1 ex:title "SPARQL Tutorial".
ex:Book2 ex:title "SemWeb".

Query

SELECT ?title WHERE {
  ?b ex:title ?title
  VALUES ?b { ex:Book1 }
}

Result

?title ↦ "SPARQL Tutorial"

~~ Bindings are conjunctively joined
Aggregates

- Aggregates allow for grouping of solutions and the computation of values over the groups

Example

```
SELECT (COUNT(?student) AS ?c) ?lecture
WHERE { ?student ex:attends ?lecture }
GROUP BY ?lecture
HAVING ?c > 5
```

- GROUP BY groups the solutions (here into students who attend the same lecture)
- COUNT is an aggregate function that counts the solutions within a group (here the number of students in the lecture)
- HAVING filters aggregated values
Aggregates

- Aggregates allow for grouping of solutions and the computation of values over the groups

Example

```
SELECT (COUNT(?student) AS ?c) ?lecture
WHERE { ?student ex:attends ?lecture }
GROUP BY ?lecture
HAVING ?c > 5
```
Aggregates

- Aggregates allow for grouping of solutions and the computation of values over the groups

**Example**

```sql
SELECT (COUNT(?student) AS ?c) ?lecture
WHERE { ?student ex:attends ?lecture }
GROUP BY ?lecture
HAVING ?c > 5
```

- **GROUP BY** groups the solutions (here into students who attend the same lecture)
- **COUNT** is an aggregate function that counts the solutions within a group (here the number of students in the lecture)
- **HAVING** filters aggregated values
Aggregates in SPARQL 1.1

SPARQL 1.1 supports the following aggregate functions, which are evaluated over the values in a group:

- **COUNT** – counts the solutions
- **MIN** – finds the minimal value
- **MAX** – finds the maximal value
- **SUM** – sums up the values
- **AVG** – computes the average
- **GROUP_CONCAT** – string concatenation, Example: `GROUP_CONCAT(?x ; separator="", "")`
- **SAMPLE** – picks a random value
Exercise Aggregates

**Data**

ex:Paul ex:hasMark 2.0 .
ex:Paul ex:hasMark 3.0 .
ex:Mary ex:hasMark 2.0 .
ex:Peter ex:hasMark 3.5 .

**Query**

SELECT ?student (AVG(?note) as ?avg)
WHERE { ?student ex:hasMark ?note }
GROUP BY ?student
HAVING (?avg > 2.0)
Solution Aggregates
## Solution Aggregates

<table>
<thead>
<tr>
<th>student</th>
<th>avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>ex:Paul</td>
<td>2.5</td>
</tr>
<tr>
<td>ex:Peter</td>
<td>3.5</td>
</tr>
</tbody>
</table>
Subqueries

Query

```
SELECT ?name WHERE {
    ?x foaf:name ?name .
    { SELECT ?x (COUNT(*) AS ?count)
        WHERE { ?x foaf:knows ?y . }
        GROUP BY ?x
        HAVING (?count = 3)
    }
}
```

- Results for the inner query are conjunctively joined with the results of the outer query
Regular Expressions in Patterns

Property Paths are constructed using regular expressions over predicates

- Alternative paths: `?s (ex:p1|ex:p2) ?o`
- Negation of paths: `?s !ex:p ?o`
- Inverse paths: `?s ^ex:p ?o same as ?o ex:p ?s`
- Sequence of paths: `?s ex:p1 / ex:p2 ?o`
- Length zero or one path: `?s ex:p? ?o`
Regular Expressions in Patterns

Property Paths are constructed using regular expressions over predicates

- **Paths with arbitrary length**: $?s \text{ex:p}^+ ?o$, $?s \text{ex:p}^* ?o$
- **Alternative paths**: $?s \text{ex:p}_1 | \text{ex:p}_2 ?o$
- **Negation of paths**: $?s !\text{ex:p} ?o$
- **Inverse paths**: $?s \text{~ex:p} ?o \text{ same as } ?o \text{ex:p} ?s$
- **Sequence of paths**: $?s \text{ex:p}_1 / \text{ex:p}_2 ?o$
- **Length zero or one path**: $?s \text{ex:p}? ?o$

- Property paths are, where possible, translated into standard SPARQL constructs
- Some new operators are still necessary
Property Path Example

Query 1

PREFIX ...
SELECT ?xName WHERE {
  ?x rdf:type foaf:Person .
  ?x foaf:name ?xName
}

Query 2

PREFIX ...
SELECT ?s WHERE {
  ?s rdf:type ?type .
  ?type rdfs:subClassOf* ex:SomeClass .
}
Negation in Queries

- Two forms of negation with conceptual and small semantic differences
  1. Test non-matches for a pattern
  2. Removal of matching patterns

1. Filter

```sql
SELECT ?x WHERE {
  ?x rdf:type foaf:Person .
  FILTER NOT EXISTS { ?x foaf:name ?name }
}
```

2. Minus

```sql
SELECT ?x WHERE {
  ?x rdf:type foaf:Person .
  MINUS { ?x foaf:name ?name }
}
```
Evaluation of Negation via Filter

Data

_:x rdf:type foaf:Person .
_:x foaf:name "Peter" .
_:y rdf:type foaf:Person .

Query Pattern

\{ ?x rdf:type foaf:Person .
FILTER NOT EXISTS \{ ?x foaf:name ?name \} \}

1. \[[ \text{Bgp(1. Pattern)} \]\]
\[ \mu_1: ?x \mapsto _:x, \mu_2: ?x \mapsto _:y \]

2. For each solution, we instantiate the second pattern
   - Solution is removed if the instantiated pattern matches \( \mu_1 \)
   - otherwise we keep the solution \( \mu_2 \)
Evaluation of Negation via Minus

Data

_:x rdf:type foaf:Person .
_:x foaf:name "Peter" .
_:y rdf:type foaf:Person .

Query Pattern

{ ?x rdf:type foaf:Person .
  MINUS { ?x foaf:name ?name } }

\[ \text{Bgp(1. Pattern)} \]_G: \( \Omega_1 = \{ \mu_1: ?x \mapsto _:x, \mu_2: ?x \mapsto _:y \} \)
\[ \text{Bgp(2. Pattern)} \]_G: \( \Omega_2 = \{ \mu_3: ?x \mapsto _:x, \?name \mapsto "Peter" \} \)
\[ \text{Minus(}\Omega_1, \Omega_2) \]_G: \( \Omega = \{ \mu | \mu \in \Omega_1 \text{ and } \forall \mu' \in \Omega_2 : \mu \text{ and } \mu' \text{ incompatible or } \text{dom}(\mu) \cap \text{dom}(\mu') = \emptyset \} \)

\( \mu_1 \notin \Omega: \mu_1 \text{ compatible with } \mu_3 \text{ and non-disjoint domains} \)
\( \mu_2 \in \Omega: \mu_2 \text{ incompatible with } \mu_3 \)
Differences Minus and Filter Negation

Data
ex:a ex:b ex:c.

Query Pattern
{ ?s ?p ?o FILTER NOT EXISTS { ?x ?y ?z } }  
  • Filter pattern matches always (variables disjoint) \(\iff\) every solution is removed

Query Pattern
{ ?s ?p ?o MINUS { ?x ?y ?z } }  
  • Minus does not remove any solutions since the domain of the solutions is disjoint
 Agenda

1 Recap
2 SPARQL 1.1 Query Extensions
   - Expressions in Selection and Bindings
   - Aggregates
   - Subqueries
   - Property Paths
   - Negation
3 SPARQL Protocol
4 SPARQL Update
5 SPARQL Service Descriptions
6 Summary
SPARQL Protocol

- Specifies how queries can be sent to a SPARQL endpoint in the Web and how results are returned
- Specifies how errors are communicated
- Query
  - **GET** Query etc. is part of the URL:
    http://server/endpoint1?query=...
  - **POST** Query is in the body of the HTTP request, e.g., via an HTML form
- Update
  - **http://server/endpoint2?update=...**
  - **POST** with content-type application/sparql-update
  - **POST** via HTML form
- Query and Update are separate services
Graph Store HTTP Protocol

- Application protocol for distributed updating and fetching of RDF graph content via HTTP
  - IRIs identify a graph in a graph store
  - GET to receive the graph content
  - PUT to send a query that modifies a graph
  - DELETE to delete a graph
  - POST to merge submitted RDF data into an existing graph
Agenda

1 Recap

2 SPARQL 1.1 Query Extensions
   - Expressions in Selection and Bindings
   - Aggregates
   - Subqueries
   - Property Paths
   - Negation

3 SPARQL Protocol

4 SPARQL Update

5 SPARQL Service Descriptions

6 Summary
SPARQL Update

- For manipulation of graphs or graph content
- Based on the idea of a graph store (Quads)
  - Addition and removal of graphs
  - Addition and removal of triples in a graphs
- `LOAD`, `DROP`, `CREATE`
- `INSERT`, `DELETE` for data/triples
- No transactions, a query can consist of several atomic parts

Example Query

```sparql
DELETE { ?person foaf:givenName "Bill" } 
INSERT { ?person foaf:givenName "William" } 
WHERE {
  ?person a foaf:Person .
  ?person foaf:givenName "Bill"
}
```
Agenda

1. Recap
2. SPARQL 1.1 Query Extensions
   - Expressions in Selection and Bindings
   - Aggregates
   - Subqueries
   - Property Paths
   - Negation
3. SPARQL Protocol
4. SPARQL Update
5. SPARQL Service Descriptions
6. Summary
Service Descriptions

- Method and vocabulary for describing SPARQL endpoints
- Client/User can request information about the SPARQL service, e.g.,
  - supported extension functions,
  - used data set or
  - supported inference mechanisms
HTTP Request

GET /sparql/ HTTP/1.1
Host: www.example.org
Accept: text/turtle

Possible Response (beginning)

HTTP/1.1 200 OK
Date: Fri, 09 Oct 2009 17:31:12 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.4 DAV/1.0.3
Connection: close
Content-Type: text/turtle

@prefix sd: <http://www.w3.org/ns/sparql-service-description#> .
@prefix ent: <http://www.w3.org/ns/entailment/> .
@prefix prof: <http://www.w3.org/ns/owl-profile/> .
...

TU Dresden  Foundations of Semantic Web Technologies  Folie 37 von 42
Possible Response (continued)

```xml

[] a sd:Service ;
   sd:endpoint <http://ex.org/sparql/> ;
   sd:supportedLanguage sd:SPARQL11Query ;
   sd:resultFormat
       <http://www.w3.org/ns/formats/RDF_XML>,
       <http://www.w3.org/ns/formats/Turtle> ;
   sd:extensionFunction <http://ex.org/Distance> ;
   sd:feature sd:DereferencesURIs ;
   sd:defaultEntailmentRegime ent:RDFS ;
```
Agenda

1 Recap

2 SPARQL 1.1 Query Extensions
   - Expressions in Selection and Bindings
   - Aggregates
   - Subqueries
   - Property Paths
   - Negation

3 SPARQL Protocol

4 SPARQL Update

5 SPARQL Service Descriptions

6 Summary
Summary

- We have learned about the main SPARQL 1.1 extensions
- SPARQL 1.1 is a recommendation since March 2013
- SPARQL UPDATE allows for modifying graphs
- Protocol specifies the client server communication
- Service Descriptions describe a SPARQL service (machine readable)
- Further result formats: JSON, CVS, TSV (not covered)

Outlook:
- Entailment Regimes: SPARQL with inferred results
Public SPARQL Endpoints

DBPedia  structured Wikipedia Data (> 100 million triples):
http://dbpedia.org/sparql

DBTune  14 billion RDF triple about music
http://dbtune.org/jamendo/store/user/query

CKAN  Dataset repository with SPARQL service
http://semantic.ckan.net/
http://semantic.ckan.net/snorql/

Linked Movie Database  http://data.linkedmdb.org/ and
http://data.linkedmdb.org/sparql

SPARQL Editor  with examples about space data http://api.talis.com/
stores/space/items/tutorial/spared.html

Semantic Web Dog Food  Information about authors, publications and
conferences  http://data.semanticweb.org/snorql