INF3580 – Semantic Technologies – Spring 2011

Lecture 4: The SPARQL Query Language

Kjetil Kjernsmo

15th February 2011

Department of Informatics

University of Oslo

Today's Plan

1. Dagens tips
2. Repetition: RDF
3. Common Vocabularies
4. SPARQL By Example
5. SPARQL Systematically
6. Executing SPARQL Queries
7. More to come!

Outline

Kommandolinje-parser

Fra Redland-biblioteket:

```
rapper data.rdf
```

Dette vil parse og printe N-Triples av det den finner hvis det validerer.

```
rapper -i turtle data.ttl
```

vil gjøre det samme for Turtle.

Installere selv på Debian/Ubuntu:

```
apt-get install raptor-utils
```

Kan også anbefale

```
apt-get install redland-utils rasqal-utils
virtuoso-minimal
```

Outline

Reminder: RDF triples

- The W3C representation of knowledge in the Semantic Web is RDF (Resource Description Framework)

Reminder: RDF triples

- The W3C representation of knowledge in the Semantic Web is RDF (Resource Description Framework)
- RDF talks about *resources* identified by URIs.

Reminder: RDF triples

- The W3C representation of knowledge in the Semantic Web is RDF (Resource Description Framework)
- RDF talks about *resources* identified by URIs.
- In RDF, all knowledge is represented by *triples*

Reminder: RDF triples

- The W3C representation of knowledge in the Semantic Web is RDF (Resource Description Framework)
- RDF talks about *resources* identified by URIs.
- In RDF, all knowledge is represented by *triples*
- A triple consists of *subject*, *predicate*, and *object*

Reminder: RDF triples

- The W3C representation of knowledge in the Semantic Web is RDF (Resource Description Framework)
- RDF talks about *resources* identified by URIs.
- In RDF, all knowledge is represented by *triples*
- A triple consists of *subject*, *predicate*, and *object*
- The subject maybe a resource or a blank node

Reminder: RDF triples

- The W3C representation of knowledge in the Semantic Web is RDF (Resource Description Framework)
- RDF talks about *resources* identified by URIs.
- In RDF, all knowledge is represented by *triples*
- A triple consists of *subject*, *predicate*, and *object*
- The subject maybe a resource or a blank node
- The predicate must be a resource

Reminder: RDF triples

- The W3C representation of knowledge in the Semantic Web is RDF (Resource Description Framework)
- RDF talks about *resources* identified by URIs.
- In RDF, all knowledge is represented by *triples*
- A triple consists of *subject*, *predicate*, and *object*
- The subject maybe a resource or a blank node
- The predicate must be a resource
- The object can be a resource, a blank node, or a literal

Reminder: RDF Literals

- *objects* of triples can also be *literals*

Reminder: RDF Literals

- *objects* of triples can also be *literals*
  - I.e. nodes in an RDF graph can be *resources* or *literals*

Reminder: RDF Literals

- *objects* of triples can also be *literals*
  - I.e. nodes in an RDF graph can be *resources* or *literals*
  - Subjects and predicates of triples can *not* be literals

Reminder: RDF Literals

- *objects* of triples can also be *literals*
  - I.e. nodes in an RDF graph can be *resources* or *literals*
  - Subjects and predicates of triples can *not* be literals
- Literals can be

Reminder: RDF Literals

- *objects* of triples can also be *literals*
  - I.e. nodes in an RDF graph can be *resources* or *literals*
  - Subjects and predicates of triples can *not* be literals
- Literals can be
  - Plain, without language tag:

```
geo:berlin geo:name "Berlin" .
```

Reminder: RDF Literals

- *objects* of triples can also be *literals*
  - I.e. nodes in an RDF graph can be *resources* or *literals*
  - Subjects and predicates of triples can *not* be literals
- Literals can be
  - Plain, without language tag:

    ```
    geo:berlin geo:name "Berlin" .
    ```
  - Plain, with language tag:

    ```
    geo:germany geo:name "Deutschland"@de .
    geo:germany geo:name "Germany"@en .
    ```

Reminder: RDF Literals

- *objects* of triples can also be *literals*
  - I.e. nodes in an RDF graph can be *resources* or *literals*
  - Subjects and predicates of triples can *not* be literals
- Literals can be
  - Plain, without language tag:
    ```
    geo:berlin geo:name "Berlin" .
    ```
  - Plain, with language tag:
    ```
    geo:germany geo:name "Deutschland"@de .
    geo:germany geo:name "Germany"@en .
    ```
  - Typed, with a URI indicating the type:
    ```
    geo:berlin geo:population "3431700"^^xsd:integer .
    ```

Reminder: RDF Blank Nodes

Blank nodes are like resources without a URI

There is a city in Germany called Berlin

```
_:x a geo:City .
_:x geo:containedIn geo:germany .
_:x geo:name "Berlin" .
```

Reminder: RDF Blank Nodes

Blank nodes are like resources without a URI

There is a city in Germany called Berlin

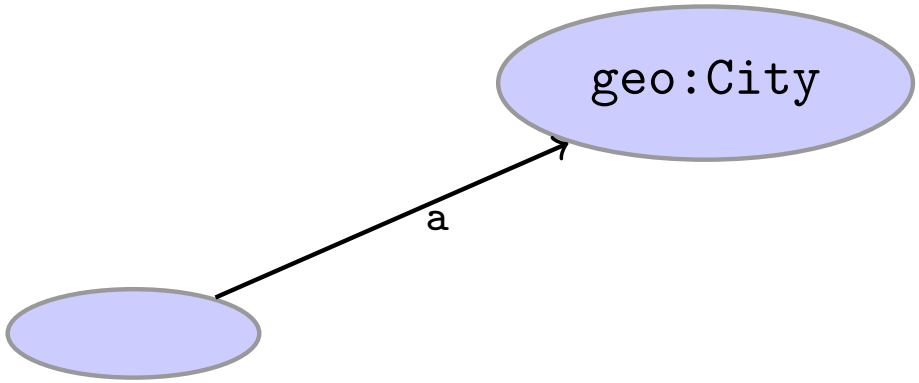
```
_:x a geo:City .
_:x geo:containedIn geo:germany .
_:x geo:name "Berlin" .
```

Reminder: RDF Blank Nodes

Blank nodes are like resources without a URI

There is a city in Germany called Berlin

```
_:x a geo:City .
_:x geo:containedIn geo:germany .
_:x geo:name "Berlin" .
```

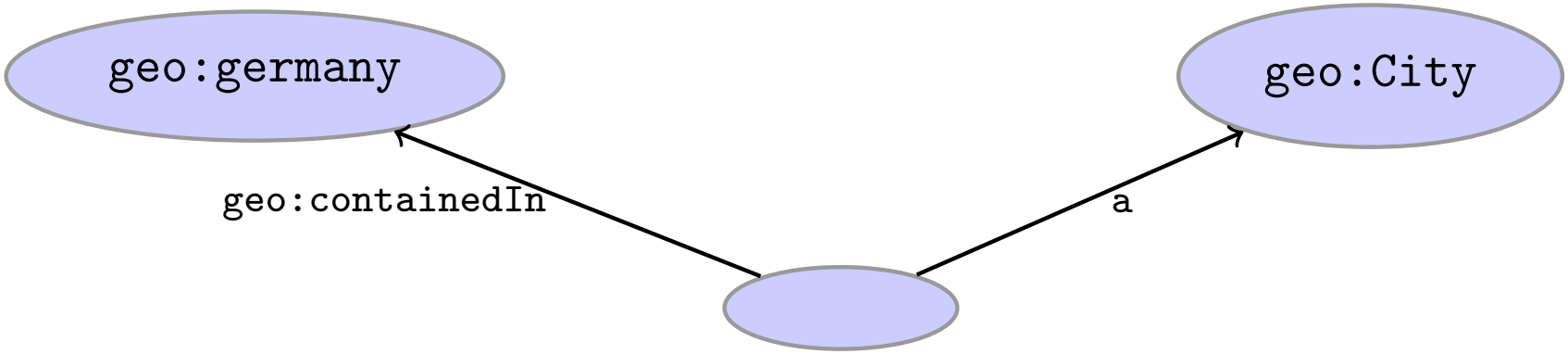


Reminder: RDF Blank Nodes

Blank nodes are like resources without a URI

There is a city in Germany called Berlin

```
_:x a geo:City .
_:x geo:containedIn geo:germany .
_:x geo:name "Berlin" .
```

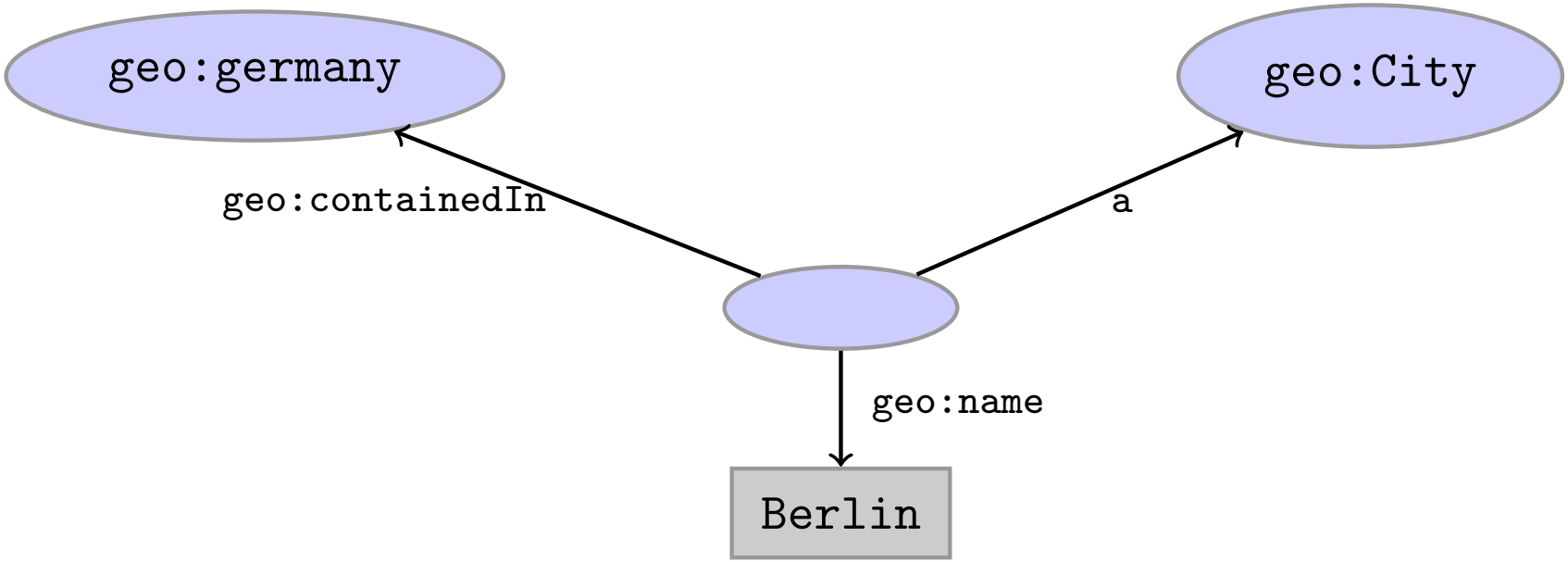


Reminder: RDF Blank Nodes

Blank nodes are like resources without a URI

There is a city in Germany called Berlin

```
_:x a geo:City .
_:x geo:containedIn geo:germany .
_:x geo:name "Berlin" .
```



Reminder: Jena

- Jena is a semantic web programming framework

Reminder: Jena

- Jena is a semantic web programming framework
- API has interfaces `Resource`, `Property`, `Literal`, `Statement`, `Model`

Reminder: Jena

- Jena is a semantic web programming framework
- API has interfaces `Resource`, `Property`, `Literal`, `Statement`, `Model`
- Need to create a `Model` first, using `ModelFactory` or `ModelMaker`.

Reminder: Jena

- Jena is a semantic web programming framework
- API has interfaces `Resource`, `Property`, `Literal`, `Statement`, `Model`
- Need to create a `Model` first, using `ModelFactory` or `ModelMaker`.
- Different kinds of models have different backing storage (memory, files, RDB)

Reminder: Jena

- Jena is a semantic web programming framework
- API has interfaces `Resource`, `Property`, `Literal`, `Statement`, `Model`
- Need to create a `Model` first, using `ModelFactory` or `ModelMaker`.
- Different kinds of models have different backing storage (memory, files, RDB)
- `Statements` and `Resources` point back to the model they belong to

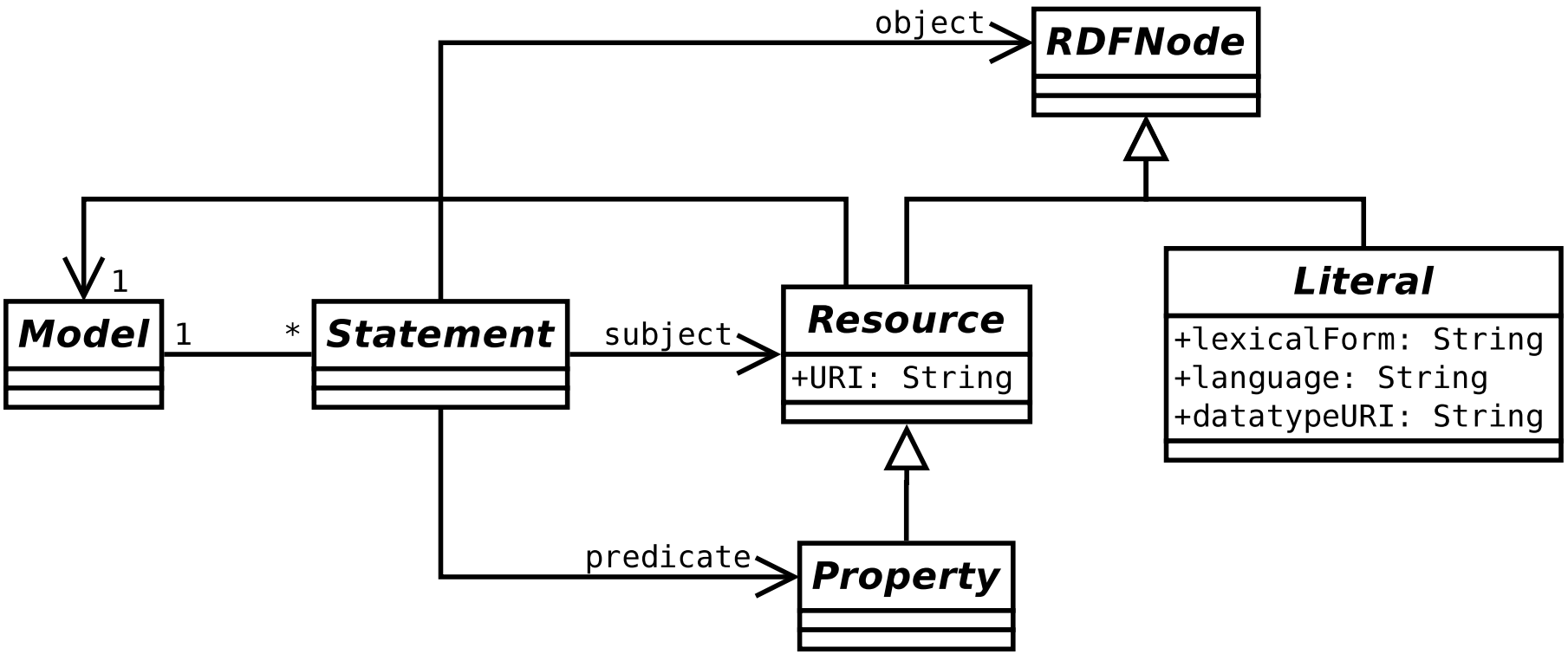
Reminder: Jena

- Jena is a semantic web programming framework
- API has interfaces `Resource`, `Property`, `Literal`, `Statement`, `Model`
- Need to create a `Model` first, using `ModelFactory` or `ModelMaker`.
- Different kinds of models have different backing storage (memory, files, RDB)
- `Statements` and `Resources` point back to the model they belong to
- Retrieval of information via methods in `Model` and `Resource`

Reminder: Jena

- Jena is a semantic web programming framework
- API has interfaces `Resource`, `Property`, `Literal`, `Statement`, `Model`
- Need to create a `Model` first, using `ModelFactory` or `ModelMaker`.
- Different kinds of models have different backing storage (memory, files, RDB)
- `Statements` and `Resources` point back to the model they belong to
- Retrieval of information via methods in `Model` and `Resource`
- Simple pattern matching with `null` as wildcard possible

Reminder: Jena



Outline

The RDF Vocabulary

- Prefix `rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`

The RDF Vocabulary

- Prefix `rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- (needs to be declared like all others!)

The RDF Vocabulary

- Prefix `rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- (needs to be declared like all others!)
- Important elements:

The RDF Vocabulary

- Prefix `rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- (needs to be declared like all others!)
- Important elements:

  type links a resource to a type (can be abbreviated).

The RDF Vocabulary

- Prefix `rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- (needs to be declared like all others!)
- Important elements:

type	links a resource to a type (can be abbreviated).
Resource	type of all resources

The RDF Vocabulary

- Prefix `rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- (needs to be declared like all others!)
- Important elements:

type	links a resource to a type (can be abbreviated).
Resource	type of all resources
Property	type of all properties

The RDF Vocabulary

- Prefix `rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>`
- (needs to be declared like all others!)
- Important elements:

type	links a resource to a type (can be abbreviated).
Resource	type of all resources
Property	type of all properties

- Examples:

```
geo:berlin rdf:type rdf:Resource .
geo:containedIn a rdf:Property .
rdf:type a rdf:Property .
```

Friend Of A Friend

- People, personal information, friends, see `http://www.foaf-project.org/`

Friend Of A Friend

- People, personal information, friends, see `http://www.foaf-project.org/`
- Prefix `foaf:<http://xmlns.com/foaf/0.1/>`

Friend Of A Friend

- People, personal information, friends, see `http://www.foaf-project.org/`
- Prefix `foaf:<http://xmlns.com/foaf/0.1/>`
- Important elements:

Friend Of A Friend

- People, personal information, friends, see `http://www.foaf-project.org/`
- Prefix `foaf:<http://xmlns.com/foaf/0.1/>`
- Important elements:

  Person a person, alive, dead, real, imaginary

Friend Of A Friend

- People, personal information, friends, see `http://www.foaf-project.org/`
- Prefix `foaf:<http://xmlns.com/foaf/0.1/>`
- Important elements:

  Person a person, alive, dead, real, imaginary

  name name of a person (also firstName, familyName)

Friend Of A Friend

- People, personal information, friends, see `http://www.foaf-project.org/`
- Prefix `foaf:<http://xmlns.com/foaf/0.1/>`
- Important elements:

Person	a person, alive, dead, real, imaginary
name	name of a person (also firstName, familyName)
mbox	mailbox URL of a person

Friend Of A Friend

- People, personal information, friends, see `http://www.foaf-project.org/`
- Prefix `foaf:<http://xmlns.com/foaf/0.1/>`
- Important elements:

Person	a person, alive, dead, real, imaginary
name	name of a person (also firstName, familyName)
mbox	mailbox URL of a person
knows	a person knows another

Friend Of A Friend

- People, personal information, friends, see `http://www.foaf-project.org/`
- Prefix `foaf:<http://xmlns.com/foaf/0.1/>`
- Important elements:

Person	a person, alive, dead, real, imaginary
name	name of a person (also firstName, familyName)
mbox	mailbox URL of a person
knows	a person knows another

- Examples:

```
<http://heim.ifi.uio.no/martingi/foaf#me>
    a foaf:Person ;
    foaf:name "Martin Giese" ;
    foaf:mbox <mailto:martingi@ifi.uio.no> ;
    foaf:knows <http://.../martige/foaf#me> .
```

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.
- Prefix `dct:<http://purl.org/dc/terms/>`

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.
- Prefix `dct:<http://purl.org/dc/terms/>`
- Use this instead of legacy `dc:`.

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.
- Prefix `dct:<http://purl.org/dc/terms/>`
- Use this instead of legacy `dc:`.
- Important elements in `dct:`

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.
- Prefix `dct:<http://purl.org/dc/terms/>`
- Use this instead of legacy `dc:`.
- Important elements in `dct`:

  creator a document's main author

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.
- Prefix `dct:<http://purl.org/dc/terms/>`
- Use this instead of legacy `dc:`.
- Important elements in `dct`:

  creator a document's main author

  created the creation date

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.
- Prefix `dct:<http://purl.org/dc/terms/>`
- Use this instead of legacy `dc:`.
- Important elements in `dct`:

creator	a document's main author
created	the creation date
description	a natural language description

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.
- Prefix `dct:<http://purl.org/dc/terms/>`
- Use this instead of legacy `dc:`.
- Important elements in `dct`:

creator	a document's main author
created	the creation date
description	a natural language description
replaces	another document superseded by this

Dublin Core

- Metadata for documents, see `http://dublincore.org/`.
- Prefix `dct:<http://purl.org/dc/terms/>`
- Use this instead of legacy `dc:`.
- Important elements in `dct`:

creator	a document's main author
created	the creation date
description	a natural language description
replaces	another document superseded by this

- Examples:

```
<http://heim.ifi.uio.no/martingi/>
    dct:creator <http://.../foaf#me> ;
    dct:created "2007-08-01" ;
    dct:description "Martin Giese's homepage"@en ;
    dct:replaces <http://my.old.homepage/> .
```

Outline

SPARQL

- SPARQL Protocol And RDF Query Language

SPARQL

- SPARQL Protocol And RDF Query Language
- Documentation:

SPARQL

- SPARQL Protocol And RDF Query Language
- Documentation:

  Queries `http://www.w3.org/TR/rdf-sparql-query/`
  Language for submitting "graph pattern" queries

SPARQL

- SPARQL Protocol And RDF Query Language
- Documentation:

  Queries `http://www.w3.org/TR/rdf-sparql-query/`
  Language for submitting "graph pattern" queries

  Protocol `http://www.w3.org/TR/rdf-sparql-protocol/`
  Protocol to submit queries to a server ("endpoint")

SPARQL

- SPARQL Protocol And RDF Query Language
- Documentation:

Queries	`http://www.w3.org/TR/rdf-sparql-query/` Language for submitting "graph pattern" queries
Protocol	`http://www.w3.org/TR/rdf-sparql-protocol/` Protocol to submit queries to a server ("endpoint")
Results	`http://www.w3.org/TR/rdf-sparql-XMLres/` XML format in which results are returned

SPARQL

- SPARQL Protocol And RDF Query Language
- Documentation:

Queries	`http://www.w3.org/TR/rdf-sparql-query/` Language for submitting "graph pattern" queries
Protocol	`http://www.w3.org/TR/rdf-sparql-protocol/` Protocol to submit queries to a server ("endpoint")
Results	`http://www.w3.org/TR/rdf-sparql-XMLres/` XML format in which results are returned

- Try it out:

SPARQL

- SPARQL Protocol And RDF Query Language
- Documentation:

  Queries `http://www.w3.org/TR/rdf-sparql-query/`
  Language for submitting "graph pattern" queries

  Protocol `http://www.w3.org/TR/rdf-sparql-protocol/`
  Protocol to submit queries to a server ("endpoint")

  Results `http://www.w3.org/TR/rdf-sparql-XMLres/`
  XML format in which results are returned

- Try it out:

  DBLP `http://dblp.l3s.de/d2r/snorql/`

SPARQL

- SPARQL Protocol And RDF Query Language
- Documentation:

Queries	`http://www.w3.org/TR/rdf-sparql-query/`
	Language for submitting "graph pattern" queries
Protocol	`http://www.w3.org/TR/rdf-sparql-protocol/`
	Protocol to submit queries to a server ("endpoint")
Results	`http://www.w3.org/TR/rdf-sparql-XMLres/`
	XML format in which results are returned

- Try it out:

DBLP	`http://dblp.l3s.de/d2r/snorql/`
DBpedia	`http://dbpedia.org/sparql`

SPARQL

- SPARQL Protocol And RDF Query Language
- Documentation:

Queries	`http://www.w3.org/TR/rdf-sparql-query/`
	Language for submitting "graph pattern" queries
Protocol	`http://www.w3.org/TR/rdf-sparql-protocol/`
	Protocol to submit queries to a server ("endpoint")
Results	`http://www.w3.org/TR/rdf-sparql-XMLres/`
	XML format in which results are returned

- Try it out:

DBLP	`http://dblp.l3s.de/d2r/snorql/`
DBpedia	`http://dbpedia.org/sparql`
DBtunes	`http://dbtune.org/musicbrainz/`

Simple Examples

- DBLP contains computer science publications

Simple Examples

- DBLP contains computer science publications
- vocabulary of RDF version:

Simple Examples

- DBLP contains computer science publications
- vocabulary of RDF version:
  - author of a document: `dc:creator`

Simple Examples

- DBLP contains computer science publications
- vocabulary of RDF version:
  - author of a document: `dc:creator`
  - title of a document: `dc:title`

Simple Examples

- DBLP contains computer science publications
- vocabulary of RDF version:
  - author of a document: `dc:creator`
  - title of a document: `dc:title`
  - name of a person: `foaf:name`

Simple Examples

- DBLP contains computer science publications
- vocabulary of RDF version:
  - author of a document: `dc:creator`
  - title of a document: `dc:title`
  - name of a person: `foaf:name`

People called "Martin Giese"

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?mg WHERE {
    ?mg foaf:name "Martin Giese" .
}
```

Simple Examples

- DBLP contains computer science publications
- vocabulary of RDF version:
  - author of a document: `dc:creator`
  - title of a document: `dc:title`
  - name of a person: `foaf:name`

People called "Martin Giese"

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT ?mg WHERE {
    ?mg foaf:name "Martin Giese" .
}
```

Answer:

?mg
<http://dblp.l3s.de/d2r/resource/authors/Martin_Giese>

Simple Examples (cont.)

Publications by people called "Martin Giese"

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
SELECT ?pub WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
}
```

Simple Examples (cont.)

Publications by people called "Martin Giese"

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
SELECT ?pub WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
}
```

Answer:

?pub
<http://dblp.l3s.de/d2r/resource/publications/conf/cade/Giese01>
<http://dblp.l3s.de/d2r/resource/publications/conf/cade/BeckertGHKRSS07>
<http://dblp.l3s.de/d2r/resource/publications/conf/fase/AhrendtBBGHHMMS02>
<http://dblp.l3s.de/d2r/resource/publications/conf/jelia/AhrendtBBGHHMS00>
<http://dblp.l3s.de/d2r/resource/publications/conf/lpar/Giese06>
...

Simple Examples (cont.)

Titles of publications by people called "Martin Giese"

```
SELECT ?title WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
    ?pub dc:title ?title .
}
```

Simple Examples (cont.)

Titles of publications by people called "Martin Giese"

```
SELECT ?title WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
    ?pub dc:title ?title .
}
```

Answer:

?title
"Incremental Closure of Free Variable Tableaux."^^xsd:string
"The KeY system 1.0 (Deduction Component)."^^xsd:string
"The KeY System: Integrating Object-Oriented Design and Formal Methods."^^xsd:string
"The KeY Approach: Integrating Object Oriented Design and Formal Verification."^^xsd:string
"Saturation Up to Redundancy for Tableau and Sequent Calculi."^^xsd:string
...

Simple Examples (cont.)

Names of people who have published with "Martin Giese"

```
SELECT ?name WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
    ?pub dc:creator ?other .
    ?other foaf:name ?name.
}
```

Simple Examples (cont.)

Names of people who have published with "Martin Giese"

```
SELECT ?name WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
    ?pub dc:creator ?other .
    ?other foaf:name ?name.
}
```

Answer:

?name
"Martin Giese"
"Bernhard Beckert"
"Martin Giese"
"Reiner Hähnle"
"Vladimir Klebanov"
...

Simple Examples (cont.)

Names of people who have published with "Martin Giese"

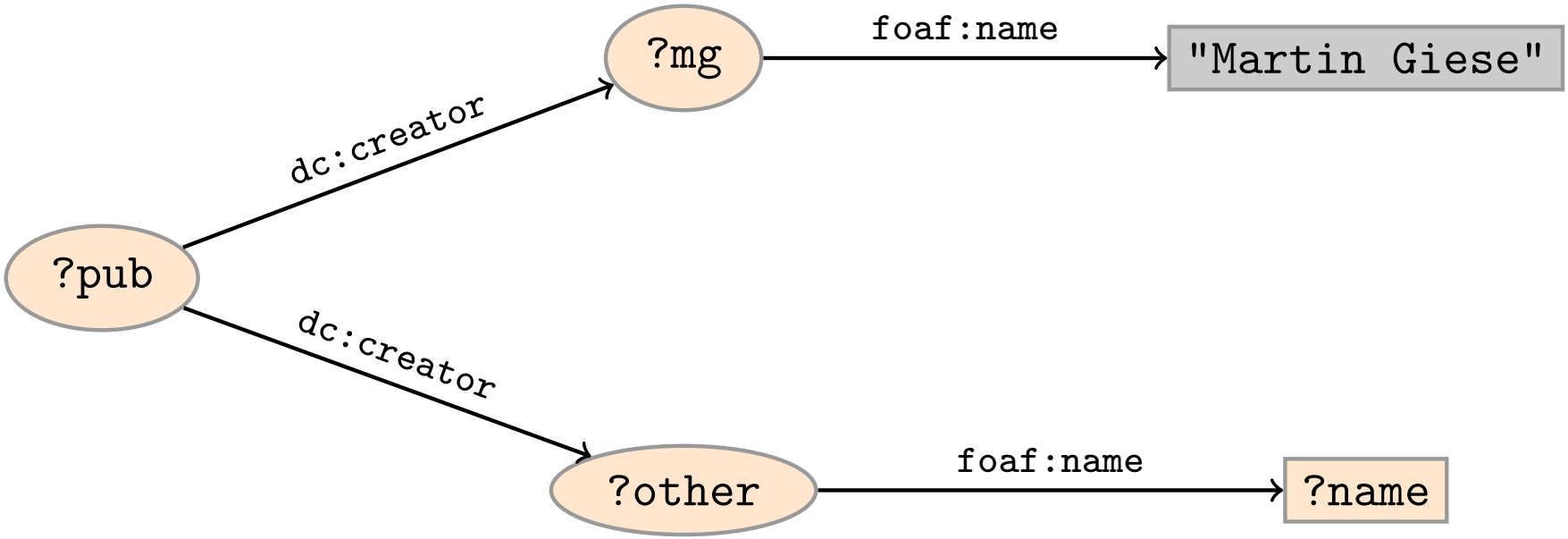
```
SELECT DISTINCT ?name WHERE {
    ?mg foaf:name "Martin Giese" .
    ?pub dc:creator ?mg .
    ?pub dc:creator ?other .
    ?other foaf:name ?name.
}
```

Answer:

?name
"Martin Giese"
"Bernhard Beckert"
"Reiner Hähnle"
"Vladimir Klebanov"
"Philipp Rümmer"
...

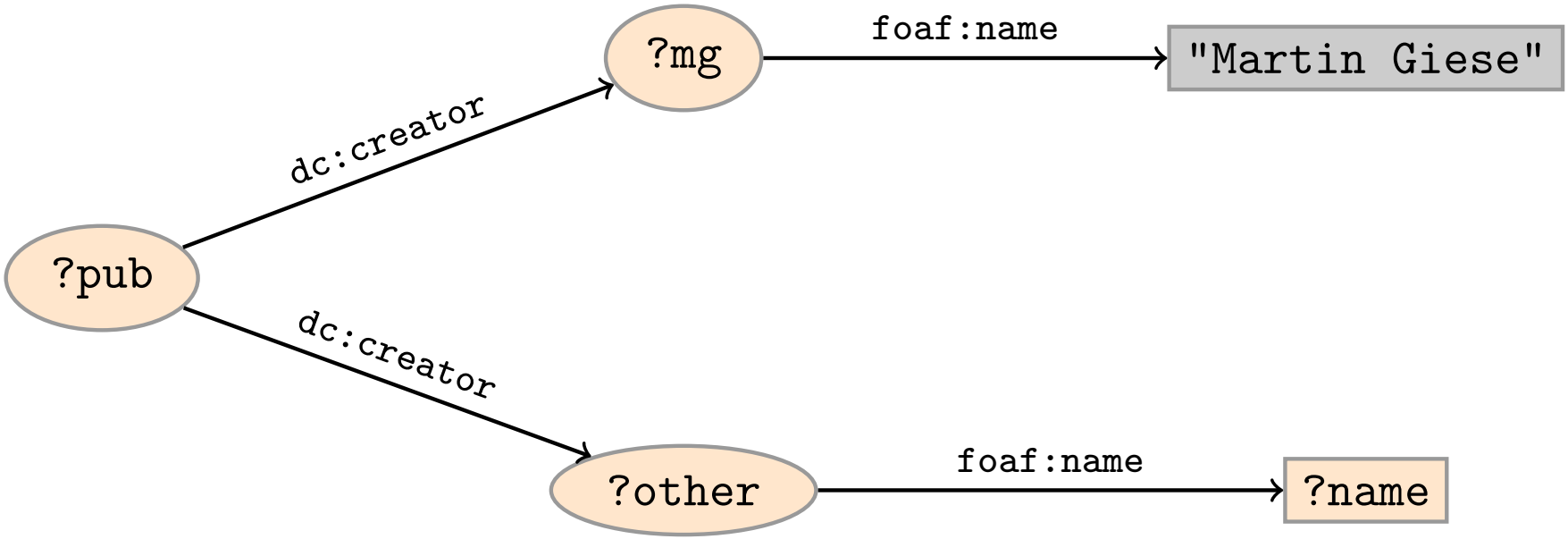
Graph Patterns

The previous SPARQL query as a graph:



Graph Patterns

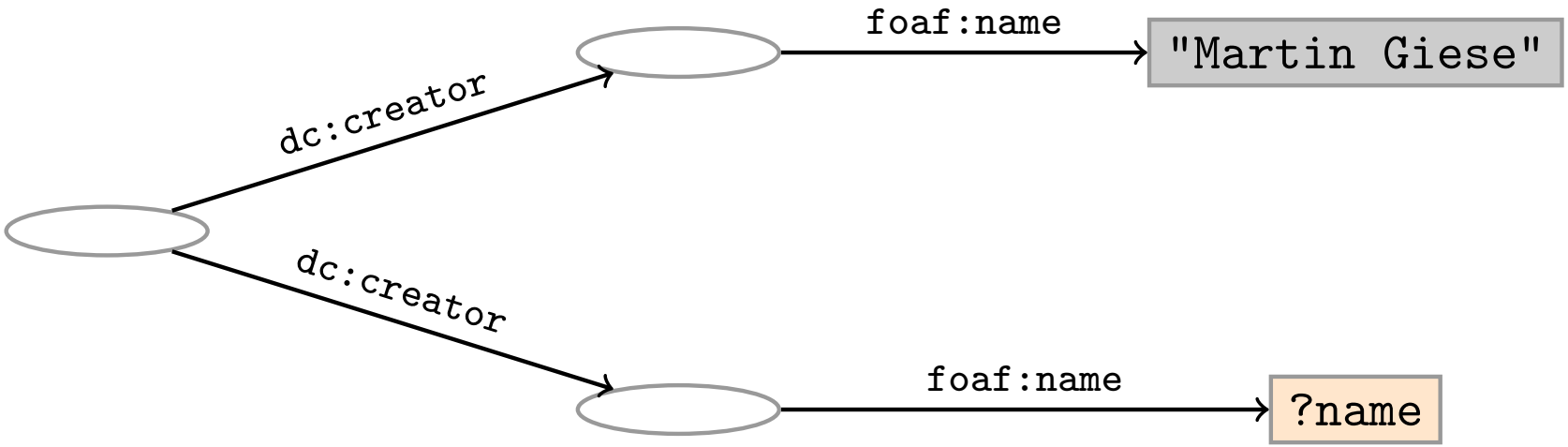
The previous SPARQL query as a graph:



Assign values to variables to make this a sub-graph of the RDF graph!

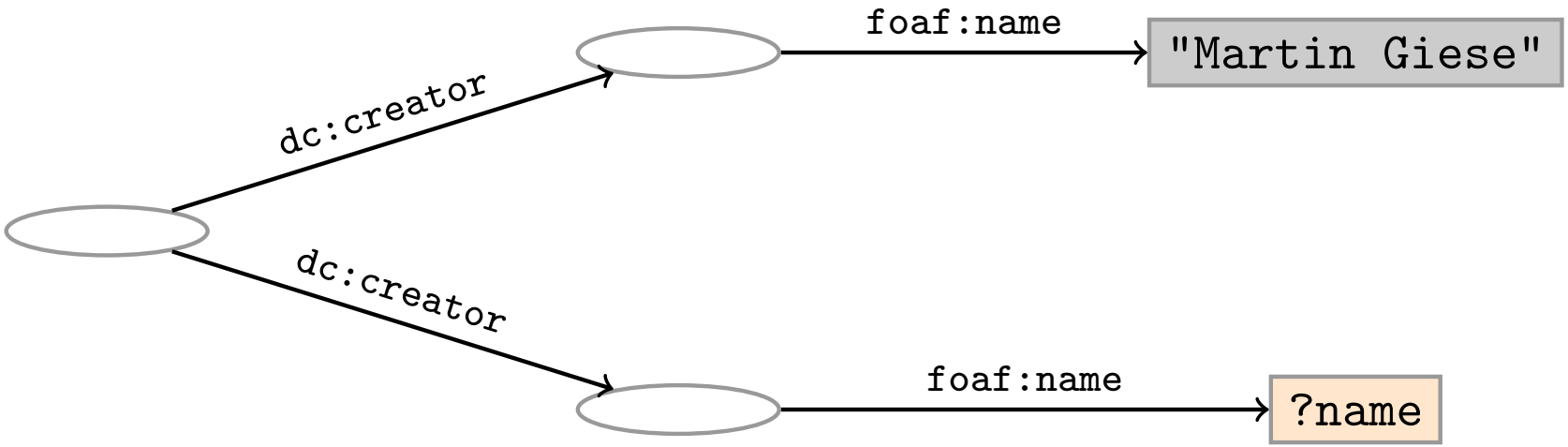
Graph with blank nodes

Variables not SELECTed can equivalently be blank:



Graph with blank nodes

Variables not SELECTed can equivalently be blank:



Assign values to variables and blank nodes to make this a sub-graph of the RDF graph!

SPARQL Query with blank nodes

Names of people who have published with "Martin Giese"

```
SELECT DISTINCT ?name WHERE {
    _:mg foaf:name "Martin Giese" .
    _:pub dc:creator _:mg .
    _:pub dc:creator _:other .
    _:other foaf:name ?name.
}
```

SPARQL Query with blank nodes

Names of people who have published with "Martin Giese"

```
SELECT DISTINCT ?name WHERE {
    _:mg foaf:name "Martin Giese" .
    _:pub dc:creator _:mg .
    _:pub dc:creator _:other .
    _:other foaf:name ?name.
}
```

The same with blank node syntax

```
SELECT DISTINCT ?name WHERE {
    _:pub dc:creator [foaf:name "Martin Giese"] .
    _:pub dc:creator _:other .
    _:other foaf:name ?name.
}
```

SPARQL Query with blank nodes

Names of people who have published with "Martin Giese"

```
SELECT DISTINCT ?name WHERE {
    _:mg foaf:name "Martin Giese" .
    _:pub dc:creator _:mg .
    _:pub dc:creator _:other .
    _:other foaf:name ?name.
}
```

The same with blank node syntax

```
SELECT DISTINCT ?name WHERE {
    _:pub dc:creator [foaf:name "Martin Giese"] .
    _:pub dc:creator [foaf:name ?name] .
}
```

SPARQL Query with blank nodes

Names of people who have published with "Martin Giese"

```
SELECT DISTINCT ?name WHERE {
    _:mg foaf:name "Martin Giese" .
    _:pub dc:creator _:mg .
    _:pub dc:creator _:other .
    _:other foaf:name ?name.
}
```

The same with blank node syntax

```
SELECT DISTINCT ?name WHERE {
    [ dc:creator [foaf:name "Martin Giese"] ;
      dc:creator [foaf:name ?name]
    ]
}
```

SPARQL Query with blank nodes

Names of people who have published with "Martin Giese"

```
SELECT DISTINCT ?name WHERE {
    _:mg foaf:name "Martin Giese" .
    _:pub dc:creator _:mg .
    _:pub dc:creator _:other .
    _:other foaf:name ?name.
}
```

The same with blank node syntax

```
SELECT DISTINCT ?name WHERE {
    [ dc:creator [foaf:name "Martin Giese"] ,
                 [foaf:name ?name]
    ]
}
```

Outline

Basic Graph Patterns

- A *Basic Graph Pattern* is a set of triple patterns.

Basic Graph Patterns

- A *Basic Graph Pattern* is a set of triple patterns.
- e.g.

```
?mg foaf:name "Martin Giese" .
_:pub dc:creator ?mg .
_:pub dc:creator ?other .
```

Basic Graph Patterns

- A *Basic Graph Pattern* is a set of triple patterns.
- e.g.

```
?mg foaf:name "Martin Giese" .
_:pub dc:creator ?mg .
_:pub dc:creator ?other .
```

- Scope of blank node labels is the basic graph pattern

Basic Graph Patterns

- A *Basic Graph Pattern* is a set of triple patterns.
- e.g.

```
?mg foaf:name "Martin Giese" .
_:pub dc:creator ?mg .
_:pub dc:creator ?other .
```

- Scope of blank node labels is the basic graph pattern
- Matching is defined via *entailment*, see next lecture

Basic Graph Patterns

- A *Basic Graph Pattern* is a set of triple patterns.
- e.g.

```
?mg foaf:name "Martin Giese" .
_:pub dc:creator ?mg .
_:pub dc:creator ?other .
```

- Scope of blank node labels is the basic graph pattern
- Matching is defined via *entailment*, see next lecture
- Basically: A match is a function that maps

Basic Graph Patterns

- A *Basic Graph Pattern* is a set of triple patterns.
- e.g.

```
?mg foaf:name "Martin Giese" .
_:pub dc:creator ?mg .
_:pub dc:creator ?other .
```

- Scope of blank node labels is the basic graph pattern
- Matching is defined via *entailment*, see next lecture
- Basically: A match is a function that maps
  - every variable and every blank node in the pattern

Basic Graph Patterns

- A *Basic Graph Pattern* is a set of triple patterns.
- e.g.

```
?mg foaf:name "Martin Giese" .
_:pub dc:creator ?mg .
_:pub dc:creator ?other .
```

- Scope of blank node labels is the basic graph pattern
- Matching is defined via *entailment*, see next lecture
- Basically: A match is a function that maps
  - every variable and every blank node in the pattern
  - to a resource, a blank node, or a literal in the RDF graph (an "RDF term")

Group Graph Patterns

- Group several patterns with { and }.

Group Graph Patterns

- Group several patterns with `{` and `}`.
- A group containing *one* basic graph pattern:

```
{
  _:pub dc:creator ?mg .
  _:pub dc:creator ?other .
}
```

Group Graph Patterns

- Group several patterns with `{` and `}`.
- A group containing *one* basic graph pattern:

```
{
  _:pub dc:creator ?mg .
  _:pub dc:creator ?other .
}
```

- A group containing two groups:

```
{
  { _:pub1 dc:creator ?mg . }
  { _:pub2 dc:creator ?other . }
}
```

Group Graph Patterns

- Group several patterns with `{` and `}`.
- A group containing *one* basic graph pattern:

```
{
  _:pub dc:creator ?mg .
  _:pub dc:creator ?other .
}
```

- A group containing two groups:

```
{
  { _:pub1 dc:creator ?mg . }
  { _:pub2 dc:creator ?other . }
}
```

- Note: Same name for two different blank nodes not allowed!

Group Graph Patterns

- Group several patterns with `{` and `}`.
- A group containing *one* basic graph pattern:

```
{
  _:pub dc:creator ?mg .
  _:pub dc:creator ?other .
}
```

- A group containing two groups:

```
{
  { _:pub1 dc:creator ?mg . }
  { _:pub2 dc:creator ?other . }
}
```

- Note: Same name for two different blank nodes not allowed!
- Match is a function from variables to RDF terms

Group Graph Patterns

- Group several patterns with `{` and `}`.
- A group containing *one* basic graph pattern:

```
{
  _:pub dc:creator ?mg .
  _:pub dc:creator ?other .
}
```

- A group containing two groups:

```
{
  { _:pub1 dc:creator ?mg . }
  { _:pub2 dc:creator ?other . }
}
```

- Note: Same name for two different blank nodes not allowed!
- Match is a function from variables to RDF terms
- Need to match all the patterns in the group.

Filters

- Groups may include *constraints* or *filters*

Filters

- Groups may include *constraints* or *filters*
- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:population ?pop .
  FILTER (?pop > 1000000)
}
```

Filters

- Groups may include *constraints* or *filters*
- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:population ?pop .
  FILTER (?pop > 1000000)
}
```

- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:abstract ?abs .
  FILTER (lang(?abs) = "no")
}
```

Filters

- Groups may include *constraints* or *filters*
- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:population ?pop .
  FILTER (?pop > 1000000)
}
```

- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:abstract ?abs .
  FILTER (lang(?abs) = "no")
}
```

- Numerical functions, string operations, reg. exp. matching, etc.

Filters

- Groups may include *constraints* or *filters*
- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:population ?pop .
  FILTER (?pop > 1000000)
}
```

- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:abstract ?abs .
  FILTER (lang(?abs) = "no")
}
```

- Numerical functions, string operations, reg. exp. matching, etc.
- Reduces matches of surrounding group to those where filter applies

Optional Patterns

- A match can leave some variables *unbound*

Optional Patterns

- A match can leave some variables *unbound*
- A *partial* function from variables to RDF terms

Optional Patterns

- A match can leave some variables *unbound*
- A *partial* function from variables to RDF terms
- Groups may include *optional parts*

Optional Patterns

- A match can leave some variables *unbound*
- A *partial* function from variables to RDF terms
- Groups may include *optional parts*
- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:population ?pop .
  OPTIONAL {
    ?x dbpprop:abstract ?abs .
    FILTER (lang(?abs) = "no")
  }
}
```

Optional Patterns

- A match can leave some variables *unbound*
- A *partial* function from variables to RDF terms
- Groups may include *optional parts*
- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:population ?pop .
  OPTIONAL {
    ?x dbpprop:abstract ?abs .
    FILTER (lang(?abs) = "no")
  }
}
```

- `?x` and `?pop` bound in every match, `?abs` bound if there is a Norwegian abstract

Optional Patterns

- A match can leave some variables *unbound*
- A *partial* function from variables to RDF terms
- Groups may include *optional parts*
- E.g.

```
{
  ?x a dbpedia-owl:Place ;
     dbpprop:population ?pop .
  OPTIONAL {
    ?x dbpprop:abstract ?abs .
    FILTER (lang(?abs) = "no")
  }
}
```

- `?x` and `?pop` bound in every match, `?abs` bound if there is a Norwegian abstract
- Groups can contain several optional parts, evaluated separately

Matching Alternatives

- A UNION pattern matches if any of some alternatives matches

Matching Alternatives

- A UNION pattern matches if any of some alternatives matches
- E.g.

```
{
  { ?book dc:creator ?author ;
          dc:created ?date . }
  UNION
  { ?book foaf:maker ?author . }
  UNION
  { ?author foaf:made ?book . }
}
```

Matching Alternatives

- A UNION pattern matches if any of some alternatives matches
- E.g.

```
{
  { ?book dc:creator ?author ;
          dc:created ?date . }
  UNION
  { ?book foaf:maker ?author . }
  UNION
  { ?author foaf:made ?book . }
}
```

- Variables in matches union of variables in sub-patterns

Matching Alternatives

- A UNION pattern matches if any of some alternatives matches
- E.g.

```
{
  { ?book dc:creator ?author ;
          dc:created ?date . }
  UNION
  { ?book foaf:maker ?author . }
  UNION
  { ?author foaf:made ?book . }
}
```

- Variables in matches union of variables in sub-patterns
- Match of one pattern leaves rest of variables unbound

Four Types of Queries

SELECT Compute table of bindings for variables

```
SELECT ?a ?b WHERE {
  [ dc:creator ?a ;
    dc:creator ?b ]
}
```

Four Types of Queries

SELECT Compute table of bindings for variables

```
SELECT ?a ?b WHERE {
  [ dc:creator ?a ;
    dc:creator ?b ]
}
```

CONSTRUCT Use bindings to construct a new RDF graph

```
CONSTRUCT {
  ?a foaf:knows ?b .
} WHERE {
  [ dc:creator ?a ;
    dc:creator ?b ]
}
```

Four Types of Queries

SELECT Compute table of bindings for variables

```
SELECT ?a ?b WHERE {
  [ dc:creator ?a ;
    dc:creator ?b ]
}
```

CONSTRUCT Use bindings to construct a new RDF graph

```
CONSTRUCT {
  ?a foaf:knows ?b .
} WHERE {
  [ dc:creator ?a ;
    dc:creator ?b ]
}
```

ASK Answer (yes/no) whether there is ≥ 1 match

Four Types of Queries

SELECT Compute table of bindings for variables

```
SELECT ?a ?b WHERE {
  [ dc:creator ?a ;
    dc:creator ?b ]
}
```

CONSTRUCT Use bindings to construct a new RDF graph

```
CONSTRUCT {
  ?a foaf:knows ?b .
} WHERE {
  [ dc:creator ?a ;
    dc:creator ?b ]
}
```

ASK Answer (yes/no) whether there is ≥ 1 match

DESCRIBE Answer available information about matching resources

Solution Modifiers

- Patterns generate an unordered collection of solutions

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)
- *Sequence modifiers* can modify the solution sequence:

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)
- *Sequence modifiers* can modify the solution sequence:
  - Order

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)
- *Sequence modifiers* can modify the solution sequence:
  - Order
  - Projection

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)
- *Sequence modifiers* can modify the solution sequence:
  - Order
  - Projection
  - Distinct

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)
- *Sequence modifiers* can modify the solution sequence:
  - Order
  - Projection
  - Distinct
  - Reduce

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)
- *Sequence modifiers* can modify the solution sequence:
  - Order
  - Projection
  - Distinct
  - Reduce
  - Offset

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)
- *Sequence modifiers* can modify the solution sequence:
  - Order
  - Projection
  - Distinct
  - Reduce
  - Offset
  - Limit

Solution Modifiers

- Patterns generate an unordered collection of solutions
- Each solution is a partial function from variables to RDF terms
- SELECT treats solutions as a sequence (solution sequence)
- *Sequence modifiers* can modify the solution sequence:
  - Order
  - Projection
  - Distinct
  - Reduce
  - Offset
  - Limit
- Applied in this order.

ORDER BY

- Used to sort the solution sequence in a given way:

ORDER BY

- Used to sort the solution sequence in a given way:
- `SELECT ... WHERE ... ORDER BY ...`

ORDER BY

- Used to sort the solution sequence in a given way:
- `SELECT ... WHERE ... ORDER BY ...`
- E.g.

```
SELECT ?country ?city ?pop WHERE {
  ?city geo:containedIn ?country ;
        geo:population ?pop .
} ORDER BY ?country DESC(?pop)
```

ORDER BY

- Used to sort the solution sequence in a given way:
- `SELECT ... WHERE ... ORDER BY ...`
- E.g.

```
SELECT ?country ?city ?pop WHERE {
  ?city geo:containedIn ?country ;
        geo:population ?pop .
} ORDER BY ?country DESC(?pop)
```

- standard defines sorting conventions for literals, URIs, etc.

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution
  - Done with `SELECT ?x ?y WHERE {?x ?y ?z...}`

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution
  - Done with `SELECT ?x ?y WHERE {?x ?y ?z...}`
- DISTINCT eliminates duplicate solutions:

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution
  - Done with `SELECT ?x ?y WHERE {?x ?y ?z...}`
- DISTINCT eliminates duplicate solutions:
  - Done with `SELECT DISTINCT ?x ?y WHERE {?x ?y ?z...}`

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution
  - Done with `SELECT ?x ?y WHERE {?x ?y ?z...}`
- DISTINCT eliminates duplicate solutions:
  - Done with `SELECT DISTINCT ?x ?y WHERE {?x ?y ?z...}`
  - A solution is duplicate if it assigns the same RDF terms to all variables as another solution.

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution
  - Done with `SELECT ?x ?y WHERE {?x ?y ?z...}`
- DISTINCT eliminates duplicate solutions:
  - Done with `SELECT DISTINCT ?x ?y WHERE {?x ?y ?z...}`
  - A solution is duplicate if it assigns the same RDF terms to all variables as another solution.
- REDUCE *allows* to remove some or all duplicate solutions

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution
  - Done with `SELECT ?x ?y WHERE {?x ?y ?z...}`
- DISTINCT eliminates duplicate solutions:
  - Done with `SELECT DISTINCT ?x ?y WHERE {?x ?y ?z...}`
  - A solution is duplicate if it assigns the same RDF terms to all variables as another solution.
- REDUCE *allows* to remove some or all duplicate solutions
  - Done with `SELECT REDUCED ?x ?y WHERE {?x ?y ?z...}`

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution
  - Done with `SELECT ?x ?y WHERE {?x ?y ?z...}`
- DISTINCT eliminates duplicate solutions:
  - Done with `SELECT DISTINCT ?x ?y WHERE {?x ?y ?z...}`
  - A solution is duplicate if it assigns the same RDF terms to all variables as another solution.
- REDUCE *allows* to remove some or all duplicate solutions
  - Done with `SELECT REDUCED ?x ?y WHERE {?x ?y ?z...}`
  - Can be expensive to find and remove all duplicates

Projection, DISTINCT, REDUCED

- Projection means that only some variables are part of the solution
  - Done with `SELECT ?x ?y WHERE {?x ?y ?z...}`
- DISTINCT eliminates duplicate solutions:
  - Done with `SELECT DISTINCT ?x ?y WHERE {?x ?y ?z...}`
  - A solution is duplicate if it assigns the same RDF terms to all variables as another solution.
- REDUCE *allows* to remove some or all duplicate solutions
  - Done with `SELECT REDUCED ?x ?y WHERE {?x ?y ?z...}`
  - Can be expensive to find and remove all duplicates
  - Leaves amount of removal to implementation

OFFSET and LIMIT

- Useful for paging through a large set of solutions

OFFSET and LIMIT

- Useful for paging through a large set of solutions
- ...but not useful for implementing paging in applications.

OFFSET and LIMIT

- Useful for paging through a large set of solutions
- ...but not useful for implementing paging in applications.
- Can compute solutions number 51 to 60

OFFSET and LIMIT

- Useful for paging through a large set of solutions
- ...but not useful for implementing paging in applications.
- Can compute solutions number 51 to 60
- Done with

```
SELECT ... WHERE {...} ORDER BY ...
LIMIT 10 OFFSET 50
```

OFFSET and LIMIT

- Useful for paging through a large set of solutions
- ...but not useful for implementing paging in applications.
- Can compute solutions number 51 to 60
- Done with
  ```
  SELECT ... WHERE {...} ORDER BY ...
  LIMIT 10 OFFSET 50
  ```
- LIMIT and OFFSET can be used separately

OFFSET and LIMIT

- Useful for paging through a large set of solutions
- ...but not useful for implementing paging in applications.
- Can compute solutions number 51 to 60
- Done with
  `SELECT ... WHERE {...} ORDER BY ...`
  `LIMIT 10 OFFSET 50`
- LIMIT and OFFSET can be used separately
- `OFFSET` not meaningful without `ORDER BY`.

Outline

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  `http://openjena.org/ARQ/javadoc/index.html`

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  `http://openjena.org/ARQ/javadoc/index.html`

- Main classes in package `com.hp.hpl.jena.query`

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  `http://openjena.org/ARQ/javadoc/index.html`

- Main classes in package `com.hp.hpl.jena.query`
  - `Query` a SPARQL query

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  `http://openjena.org/ARQ/javadoc/index.html`

- Main classes in package `com.hp.hpl.jena.query`
  - `Query` a SPARQL query
  - `QueryFactory` for creating queries in various ways

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  `http://openjena.org/ARQ/javadoc/index.html`

- Main classes in package `com.hp.hpl.jena.query`
  - `Query` a SPARQL query
  - `QueryFactory` for creating queries in various ways
  - `QueryExecution` for the execution state of a query

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  `http://openjena.org/ARQ/javadoc/index.html`

- Main classes in package `com.hp.hpl.jena.query`
  - `Query` a SPARQL query
  - `QueryFactory` for creating queries in various ways
  - `QueryExecution` for the execution state of a query
  - `QueryExecutionFactory` for creating query executions

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  `http://openjena.org/ARQ/javadoc/index.html`

- Main classes in package `com.hp.hpl.jena.query`
  - `Query` a SPARQL query
  - `QueryFactory` for creating queries in various ways
  - `QueryExecution` for the execution state of a query
  - `QueryExecutionFactory` for creating query executions
  - `ResultSet` for results of a `SELECT`

SPARQL in Jena

- SPARQL functionality bundled with Jena has separate Javadocs:

  `http://openjena.org/ARQ/javadoc/index.html`

- Main classes in package `com.hp.hpl.jena.query`
  - `Query` a SPARQL query
  - `QueryFactory` for creating queries in various ways
  - `QueryExecution` for the execution state of a query
  - `QueryExecutionFactory` for creating query executions
  - `ResultSet` for results of a `SELECT`
- `CONSTRUCT` and `DESCRIBE` return `Models`, `ASK` a Java boolean.

Constructing a Query and a QueryExecution

- Query objects are usually constructed by parsing:

```
String qStr =
      "PREFIX foaf:  <" + foafNS + ">"
    + "SELECT ?a ?b WHERE {"
    + "  ?a foaf:knows ?b ."
    + "} ORDER BY ?a ?b";
Query q = QueryFactory.create(qStr);
```

Constructing a Query and a QueryExecution

- Query objects are usually constructed by parsing:

```
String qStr =
        "PREFIX foaf:  <" + foafNS + ">"
      + "SELECT ?a ?b WHERE {"
      + "  ?a foaf:knows ?b ."
      + "} ORDER BY ?a ?b";
Query q = QueryFactory.create(qStr);
```

- Programming interface deprecated and badly documented

Constructing a Query and a QueryExecution

- Query objects are usually constructed by parsing:

```
String qStr =
        "PREFIX foaf:  <" + foafNS + ">"
      + "SELECT ?a ?b WHERE {"
      + "  ?a foaf:knows ?b ."
      + "} ORDER BY ?a ?b";
Query q = QueryFactory.create(qStr);
```

- Programming interface deprecated and badly documented
- A Query can be used several times, on multiple models

Constructing a Query and a QueryExecution

- Query objects are usually constructed by parsing:

```
String qStr =
        "PREFIX foaf:  <" + foafNS + ">"
      + "SELECT ?a ?b WHERE {"
      + "  ?a foaf:knows ?b ."
      + "} ORDER BY ?a ?b";
Query q = QueryFactory.create(qStr);
```

- Programming interface deprecated and badly documented
- A Query can be used several times, on multiple models
- For each execution, a new QueryExecution is needed

Constructing a Query and a QueryExecution

- Query objects are usually constructed by parsing:

```
String qStr =
        "PREFIX foaf:  <" + foafNS + ">"
      + "SELECT ?a ?b WHERE {"
      + "  ?a foaf:knows ?b ."
      + "} ORDER BY ?a ?b";
Query q = QueryFactory.create(qStr);
```

- Programming interface deprecated and badly documented
- A Query can be used several times, on multiple models
- For each execution, a new QueryExecution is needed
- To produce a QueryExecution for a given Query and Model:

```
QueryExecution qe =
    QueryExecutionFactory.create(q,model);
```

Executing a Query

- `QueryExecution` contains methods to execute different kinds of queries (SELECT, CONSTRUCT, etc.)

Executing a Query

- `QueryExecution` contains methods to execute different kinds of queries (SELECT, CONSTRUCT, etc.)
- E.g. for a SELECT query:

```
ResultSet res = qe.execSelect();
```

Executing a Query

- `QueryExecution` contains methods to execute different kinds of queries (SELECT, CONSTRUCT, etc.)
- E.g. for a SELECT query:

```
ResultSet res = qe.execSelect();
```

- `ResultSet` is a sub-interface of `Iterator<QuerySolution>`

Executing a Query

- `QueryExecution` contains methods to execute different kinds of queries (SELECT, CONSTRUCT, etc.)
- E.g. for a SELECT query:

```
ResultSet res = qe.execSelect();
```

- `ResultSet` is a sub-interface of `Iterator<QuerySolution>`
- Also has methods to get list of variables

Executing a Query

- `QueryExecution` contains methods to execute different kinds of queries (SELECT, CONSTRUCT, etc.)
- E.g. for a SELECT query:

```
ResultSet res = qe.execSelect();
```

- `ResultSet` is a sub-interface of `Iterator<QuerySolution>`
- Also has methods to get list of variables
- `QuerySolution` has methods to get list of variables, value of single variables, etc.

Executing a Query

- `QueryExecution` contains methods to execute different kinds of queries (SELECT, CONSTRUCT, etc.)
- E.g. for a SELECT query:

```
ResultSet res = qe.execSelect();
```

- `ResultSet` is a sub-interface of `Iterator<QuerySolution>`
- Also has methods to get list of variables
- `QuerySolution` has methods to get list of variables, value of single variables, etc.
- Important to call `close()` on query executions when no longer needed.

Example: SPARQL in Jena

```
String qStr = "SELECT ?a ?b ...";
Query q = QueryFactory.create(qStr);

QueryExecution qe =
   QueryExecutionFactory.create(q,model);

try {
   res = qe.execSelect();
   while( res.hasNext()) {
      QuerySolution soln = res.next();
      RDFNode a = soln.get("?a");
      RDFNode b = soln.get("?b");
      System.out.println(""+a+" knows "+b);
   }
} finally {
   qe.close();
}
```

SPARQL on the 'Net

- Many sites (DBLP, dbpedia, dbtunes,...) publish *SPARQL endpoints*

SPARQL on the 'Net

- Many sites (DBLP, dbpedia, dbtunes,...) publish *SPARQL endpoints*
- I.e. SPARQL queries can be submitted to a database server that sends back the results

SPARQL on the 'Net

- Many sites (DBLP, dbpedia, dbtunes,...) publish *SPARQL endpoints*
- I.e. SPARQL queries can be submitted to a database server that sends back the results
- Uses HTTP to submit URL-encoded queries to server

```
GET /sparql/?query=...  HTTP/1.1
```

SPARQL on the 'Net

- Many sites (DBLP, dbpedia, dbtunes,...) publish *SPARQL endpoints*
- I.e. SPARQL queries can be submitted to a database server that sends back the results
- Uses HTTP to submit URL-encoded queries to server
  `GET /sparql/?query=... HTTP/1.1`
- Actually defined via W3C Web Services, see

  `http://www.w3.org/TR/rdf-sparql-protocol/`

SPARQL on the 'Net

- Many sites (DBLP, dbpedia, dbtunes,...) publish *SPARQL endpoints*
- I.e. SPARQL queries can be submitted to a database server that sends back the results
- Uses HTTP to submit URL-encoded queries to server
  `GET /sparql/?query=... HTTP/1.1`
- Actually defined via W3C Web Services, see

  `http://www.w3.org/TR/rdf-sparql-protocol/`
- For `SELECT` queries you get a XML or JSON result set, see

  `http://www.w3.org/TR/rdf-sparql-XMLres/`
  `http://www.w3.org/TR/rdf-sparql-json-res/`

SPARQL on the 'Net

- Many sites (DBLP, dbpedia, dbtunes,...) publish *SPARQL endpoints*
- I.e. SPARQL queries can be submitted to a database server that sends back the results
- Uses HTTP to submit URL-encoded queries to server
  `GET /sparql/?query=... HTTP/1.1`
- Actually defined via W3C Web Services, see

  `http://www.w3.org/TR/rdf-sparql-protocol/`
- For `SELECT` queries you get a XML or JSON result set, see

  `http://www.w3.org/TR/rdf-sparql-XMLres/`
  `http://www.w3.org/TR/rdf-sparql-json-res/`
- Nothing you would want to do manually!

Remote SPARQL with Jena

- Jena can send SPARQL queries to a remote endpoint!

Remote SPARQL with Jena

- Jena can send SPARQL queries to a remote endpoint!
- Use one of the `sparqlService` in `QueryExecutionFactory`

Remote SPARQL with Jena

- Jena can send SPARQL queries to a remote endpoint!
- Use one of the `sparqlService` in `QueryExecutionFactory`
- E.g.

```
String endpoint = "http://dblp.l3s.de/d2r/sparql";
String qStr = "SELECT ?a ?b ...";
Query q = QueryFactory.create(qStr);

QueryExecution qe =
   QueryExecutionFactory.sparqlService(endpoint,q);

try {
   res = qe.execSelect();
   ...
} finally {
   qe.close();
}
```

Summary

- SPARQL is a W3C-standardised query language for RDF graphs

Summary

- SPARQL is a W3C-standardised query language for RDF graphs
- It is built about "graph patterns"

Summary

- SPARQL is a W3C-standardised query language for RDF graphs
- It is built about "graph patterns"
- Only queries compatible with "open world assumption"

Summary

- SPARQL is a W3C-standardised query language for RDF graphs
- It is built about "graph patterns"
- Only queries compatible with "open world assumption"
- Comes with a protocol to communicate with "endpoints"

Summary

- SPARQL is a W3C-standardised query language for RDF graphs
- It is built about "graph patterns"
- Only queries compatible with "open world assumption"
- Comes with a protocol to communicate with "endpoints"
- Can be conveniently used with Jena and tens of other systems.

Outline

RDF Datasets

- SPARQL contains a mechanism for named RDF graphs
- Collections of named graphs are called "RDF datasets"
- Syntax for declaring named graphs in SPARQL
- Syntax for matching graph patterns in a given graph

SPARQL 1.1 in the works

- Updates (add delete triples)

SPARQL 1.1 in the works

- Updates (add delete triples)
- Service Descriptions

SPARQL 1.1 in the works

- Updates (add delete triples)
- Service Descriptions
- Basic Federated query

SPARQL 1.1 in the works

- Updates (add delete triples)
- Service Descriptions
- Basic Federated query
- Subqueries.

SPARQL 1.1 in the works

- Updates (add delete triples)
- Service Descriptions
- Basic Federated query
- Subqueries.
- Property paths (to shorten common queries)

SPARQL 1.1 in the works

- Updates (add delete triples)
- Service Descriptions
- Basic Federated query
- Subqueries.
- Property paths (to shorten common queries)
- Aggregate functions (count, sum, average,. . .)

SPARQL 1.1 in the works

- Updates (add delete triples)
- Service Descriptions
- Basic Federated query
- Subqueries.
- Property paths (to shorten common queries)
- Aggregate functions (count, sum, average,. . .)
- Negation, set difference, i.e. something is *not* in a graph

SPARQL 1.1 in the works

- Updates (add delete triples)
- Service Descriptions
- Basic Federated query
- Subqueries.
- Property paths (to shorten common queries)
- Aggregate functions (count, sum, average,. . .)
- Negation, set difference, i.e. something is *not* in a graph
- Entailment