RDF and SPARQL

SEAT
Massey Universit

Data Model: RDF

Why we need RDF? Semantic Web.

> *"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."*

Specific g

- A descriptive language with standard semanti
  - Make semantics machine-processable an
- Incorporate logical infrastructure to reason a
- W3C proposals: Resource Description Framework (RDF) and its query language SPARQL.

Data Model: RDF

Resource Description Framework (RDF)

RDF is a graph-based model for representing the web as a graph of

- Resources and their properties, using
- Internationalized Resource Identifiers (IRIs) [a] and literal values.
  -
  -

[a] A gener

The graph is a collection of subject-predicate-objec

Data Model: RDF

- RDF Statement : an RDF triple.
- An RDF triple:
  - Subject :IRI or blank node.
  - Predicate : IRI.
  - Object : IRI, blank node or literal.

Data Model: RDF

- RDF Statement : an RDF triple.
- An RDF triple:
  - Subject :IRI or blank node.
  - Predicate : IRI.
  - Object : IRI, blank node or literal.

Data Model: RDF

One can write RDF in different syntaxes:

- RDF/XML: respent RDF data in XML.
- Turtle: a compact presentation of RDF data in RDF triples.

Data Model: RDF

One can write RDF in different syntaxes:

- RDF/XML: respent RDF data in XML.
- Turtle: a compact presentation of RDF data in RDF triples.

Data Model: RDF

One can write RDF in different syntaxes:

- RDF/XML: respent RDF data in XML.
- Turtle: a compact presentation of RDF data in RDF triples


```
<?xml versi
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/
         xmlns:dc="http://purl.org/dc/elem
         xmlns:ex="http://example.org/stuf
  <rdf:Description rdf:about="http://www.w3.or
           dc:title="RDF1.1 XML Syntax">
    <ex:editor>
      <rdf:Description ex:fullName="Dave Beckett">
        <ex:homePage rdf:resource="http://purl.org/net/dajobe/" />
      </rdf:Description>
    </ex:editor>
  </rdf:Description>
</rdf:RDF>
```

Data Model: RDF

RDF document can be presented in different syntaxes.

- RDF/XML: respent RDF data in XML.
- Turtle: a compact presentation of RDF data in RDF triples.


```
@prefix rdf: <http://www.w3.org/1999/02/22-rd
@prefix dc: <http://purl.org/dc/elements/1.1/
@prefix ex: <http://example.org/stuff/1.0/> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax" ;
  ex:editors [
    ex:fullname "Dave Beckett";
    ex:homePage <http://purl.org/net/dajobe/>
  ] .
```

Data Model: RDF

RDF document can be presented in different syntaxes.

- RD
- Tur

Next, we shall mainly use the Turtle syntax to describe a

RDF Sch

RDF Sche
propertie

- usin
- subclasses (sc), subproperties (sp), domain/

Data Model: RDF

RDF Schema

RDF Schema standardizes RDF vocabulary for describing classes and properties

- using object-oriented types + domain/range
- sub

Range an

- A rdfs:domain B
  - A is an instance of the class rdf:Property.
  - B is a instance of the class rdfs:Class.
  - The subjects of triples whose predicate is A are th
- A rdfs:range B
  - A is an instance of the class rdf:Property.
  - B is a instance of the class rdfs:Class.
  - The objects of triples whose predicate is A are the instances of B.

Data Model: RDF

RDF Schema

RDF Schema standardizes RDF vocabulary for describing classes and properties

- using object-oriented types + domain/range
- sub

Data Model: RDF

RDF Schema

RDF Schema standardizes RDF vocabulary for describing classes and properties

- using object-oriented types + domain/range
- sub

The sema

- Sub
  - $(A, sp, B) + (B, sp, C) => (A, sp, C$
  - $(A, sp, B) + (X, A, Y) => (X, B, Y$
- Subclass:
  - $(A, sc, B) + (B, sc, C) => (A, sc, C$
  - $(A, sc, B) + (X, type, A) => (X, type, B)$
- Typing:
  - $(A, dom, B) + (X, A, Y) => (X, type, B)$
  - $(A, range, B) + (X, A, Y) => (Y, type, B)$

Data Model: RDF

prefix	Namespace IRI	RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary

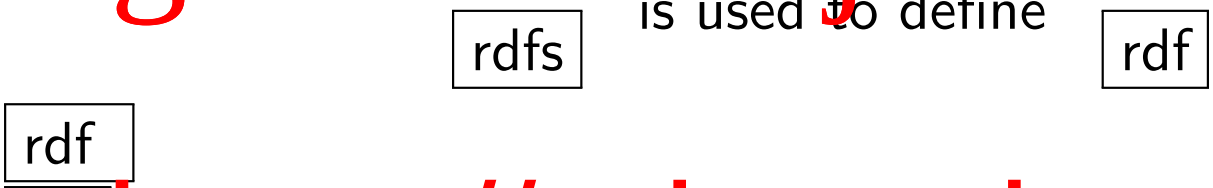
rdfs

Data Model: RDF

prefix	Namespace IRI	RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary

Data Model: RDF

prefix	Namespace IRI	RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary

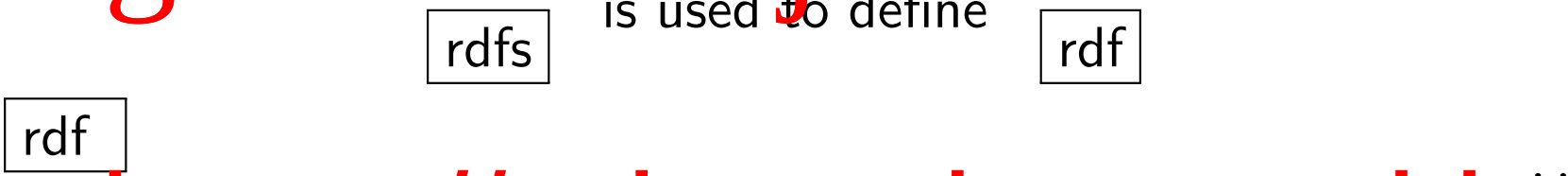


rdf

rdf

Data Model: RDF

prefix	Namespace IRI	RDF vocabulary
rdfs	`http://www.w3.org/2000/01/rdf-schema#`	The RDF Schema vocabulary
rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`	The RDF built-in vocabulary



rdf

rdf

...

Data Model: RDF

- Namespace URI: the common substring of the URIs in an RDF vocabulary.

prefix	Namespace IRI	RDF vocabulary
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The RDF built-in vocabulary
rdfs	http://www.w3.org/2000/01/rdf-schema#	The RDF Schema vocabulary
xsd	http://www.w3.org/2001/XMLSchema#	The RDF-compatible XSD types
dc		adata Element S
eg		ned locally.

Data Model: RDF

- Namespace URI: the common substring of the URIs in an RDF vocabulary.

prefix	Namespace IRI	RDF vocabulary
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The RDF built-in vocabulary
rdfs	http://www.w3.org/2000/01/rdf-schema#	The RDF Schema vocabulary
xsd	http://www.w3.org/2001/XMLSchema#	The RDF-compatible XSD types
dc		adata Element S
eg		ned locally.

Data Model: RDF

Ontology = ontos(being) + logos(word).

- An ontology is a systematic explanation of existence (400BC).
- Gruber (1998)
  - Formal: machine readable
  - Explicit: specifications of concepts, properties, functions, axioms are

Data Model: RDF

Ontology = ontos(being) + logos(word).

- An ontology is a systematic explanation of existence (400BC).
- Gruber (1998)
  - Formal: machine readable
  - Explicit: specifications of concepts, properties, functions, axioms are

- https://eduassistpro.github.io/fct/

Example

Querying RDF: SPARQL

An SQL query retrieves data from a database under the relational model.

```
SELECT name                    data needed
FROM student                   data source
WHERE age > 20                 data constraint
```


SPARQ

- a rec
  SP
- a graph-matching query language for RDF gra
- recommended by W3C
- Like SQL, SPARQL consists of three compone
  - select: the entities to be returned
  - from: the data source (RDF graph)
  - where: the pattern to be matched against the RDF graph
  - optional prologue: namespace

Querying RDF: SPARQL

Example: find 10 films with the keyword of "Paddington" in thier labels.

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix movie: <http://data.linkedmdb.org/resource/movie/>

select dist
  { ?film a m
  { ?film a d
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
}
limit 10
```

Try this on the SPARQL endPoint of dbpedia

https://dbpedia.org/sparql.

Query Language: SPARQL

The WHERE clause specifies the graph pattern which

- is an RDF graph
- invo
- has t
- can b

Semantic: for a pattern P and an RDF graph the set of matchings of P, that are, subgraphs of

Triple pattern

Example: A triple pattern can have one/more variables.

- ?fil
- ?fil

Note: a is the


```
select ?film where {
   ?film a dbpedia-owl:Film
} limit 10
```

https://dbpedia.org/sparql.

Query Language: SPARQL

A larger pattern is assembled with triple patterns.

The queries below are equivalent.

```
select
where {
   {?fil
   {?film rdfs:label ?label }
}
```

```
}
```


Semantics: $[P_1 \text{ AND } P_2]_G = [P_1]_G \bowtie [P_2]$

Query Language: SPARQL

Alternative Graphs: Union


```
select ?film
where {
   {?fi
   {?fi
}
```


Return the films that EITHER satisfy the pattern of

- ?film a movie:Film or
- ?film rdfs:label ?label

Semantic: $[P1 \mathit{UNION} P2] = [P1] \cup [P2]$.

Query Language: SPARQL

Optional pattern:


```
select ?film ?label
where {
  {?fi
  {?fi
}
```


Return the films that satisfy the pattern of

- ?film a movie:Film

and return its labels if there are any.

Semantics: $[P_1 \text{ OPT } P_2]_G = [P_1]_G \mathbin{⟕} [P_2]_G$

Query Language: SPARQL

Filter clause: a boolean expression consists of

- RDF-model related operators
  - is Literal(?aNode)
  - isURI(?aNode)
  - ...
- Nu
- Lite


```
select
  ?fil
  filter regex( str(?label), "Paddingto
}
```


- xsd:boolean REGEX (string literal text, simple literal pattern, flags)
- flags
  - i: case-insensitive

See `https://www.w3.org/TR/xpath-functions/#regex-syntax` for more information.

Query Language: SPARQL

Filter clause: a boolean expression consists of

- RDF-model related operators
  - isLiteral(?aNode)
  - isURI(?aNode)
  -
- Nu
- Lite


```
select ?film where  {
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddingto
}
```

Semantics: $[P \text{ FILTER } R]_G = \{\mu \in [P]_G \mid \mu \models R\}$

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.


```
{ P1
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery


```
{ { P1
```

- Gro


```
}
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.

- Gro
- Opt


```
{ { P1

}
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.

- Gro
- Opt
- Nesting


```
{ { P1

}
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.

- Gro
- Opt
- Nesting
- Union of patterns


```
{ { P1

}
UNION
{ P9 }
```

Query Language: SPARQL

The pattern of a SPARQL query can be complicated.

Let $P1$ to $P9$ be triple patterns and R be a regular expression in XQuery.

- Gro
- Opt
- Nesting
- Union of patterns
- Filtering


```
{ { P1

}
UNION
{ P9
  FILTER ( R ) }
```

Query Language: SPARQL

Let $P1$ and $P9$ be triple patterns and R be a regular expression in XQuery.

- Graph patterns: full parenthesized algebra

original SPARQL syntax	algebraic syntax
`{ P1 P2 }`	
`{ P1 OPT`	
`{ P1 } UNION { P2 }`	
`{ P1 FILTER ( R ) }`	

Query Language: SPARQL

Semantics of SPARQL

Given an RDF graph G.

Let t be a t

Definiti

$$
\begin{aligned}
[t]_G &= \\
[P_1 \text{ AND } P_2]_G &= \\
[P_1 \text{ UNION } P_2]_G &= \\
[P_1 \text{ OPT } P_2]_G &=
\end{aligned}
$$

Semantics of SPARQL

Given an RDF graph G.

Let t be a t

Definiti

$$
\begin{array}{lcl}
[t]_G & = & \text{the subgrap} \\
[P_1 \text{ AND } P_2]_G & = & [P_1]_G \bowtie [P \\
[P_1 \text{ UNION } P_2]_G & = & [P_1]_G \cup [P \\
[P_1 \text{ OPT } P_2]_G & = & [P_1]_G \mathbin{⟕} [
\end{array}
$$

Query Language: SPARQL

Solution Modifiers.

- ORDER BY
- LIMIT
- DISTINCT
- PROJECTION
- OF
  seq
- RE


```
prefix rdfs: <http://www.w3.org/2000/01/rdf-s
prefix dbpedia-owl: <http://dbpedia.org/ontolog
prefix movie: <http://data.linkedmdb.org/reso

select distinct ?film where {
  { ?film a movie:Film } union
  { ?film a dbpedia-owl:Film }
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
}
order by ?film
limit 10
```

Query Language: SPARQL

SPARQL can be used to:

- sele
- con
- test
- describe a resource with an RDF graph

Query Language: SPARQL

Example: find 10 films with the keyword of "Paddington" in their labels.

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix db
prefix mo

select dist
  { ?film a m                    }
  { ?film a dbpedia-owl:Film }
  ?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
}
limit 10
```

Query Language: SPARQL

Example of AND , UNION : find 10 films either with the keyword of "Paddington" in their labels or acted by Tom Cruise.

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix mo

select dist
  { ?film a m
  { ?film a dbpedia-owl:Film }
  {?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
  {?film dbo:starring ?actor
  filter regex( str(?actor), "Tom_Cruise", "i")
}
limit 10
```

Query Language: SPARQL

Example of OPT : find 10 films with the keyword of "Paddington" in their labels and show if they are acted by Hugh Grant.

```
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix dbpedia-owl: <http://dbpedia.org/ontology/>
prefix mo

select dist
  { ?film a m
  { ?film a dbpedia-owl:Film }
  {?film rdfs:label ?label .
  filter regex( str(?label), "Paddington", "i")
  {?film dbo:starring ?actor
  filter regex( str(?actor), "Hugh_Grant", "i")
}
limit 10
```

- Data Model: RDF
- Que

Resourc

- Prefixes: `http://dbpedia.org/spa`
- DBpedia: `http://dbpedia.org/sp`
- W3C: `https://www.w3.org/TR/rdf`

References