Week 3

RDF: The Turtle Data Format

SPARQL (Basic)

Main Sources:

- https://www.w3.org/TR/turtle/
- http://www.w3.org/TR/rdf-sparql-query/
- Chapter 3 of Semantic Web Primer

Dr. Davoud Mougouei

Lecture Outline

1. **Turtle Syntax of RDF**
2. SPARQL: Querying RDF Documents

Terse RDF Triple Language (Turtle)

The contents are mainly from: https://www.w3.org/TR/turtle/

What is it?

- An alternative to RDF/XML
- A textual syntax for RDF
- Allows an RDF graph to be completely written in a compact and natural text form with abbreviations for common usage patterns and datatypes.
- Provides levels of compatibility with the triple pattern syntax of the SPARQL W3C Recommendation.
- Offers better readability.
- A Turtle document is a textual representations of an RDF graph

Green Goblin and Spiderman!

Turtle representation

```
1.  @base <http://example.org/> .
2.  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3.  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
4.  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
5.  @prefix rel: <http://www.perceive.net/schemas/relationship/> .

6.  <#green-goblin>
7.  rel:enemyOf <#spiderman> ;
8.  a foaf:Person ; # in the context of the Marvel universe foaf:name "Green Goblin" .

9.  <#spiderman>
10. rel:enemyOf <#green-goblin> ;
11. a foaf:Person ; foaf:name "Spiderman", "Человек-паук"@ru .
```

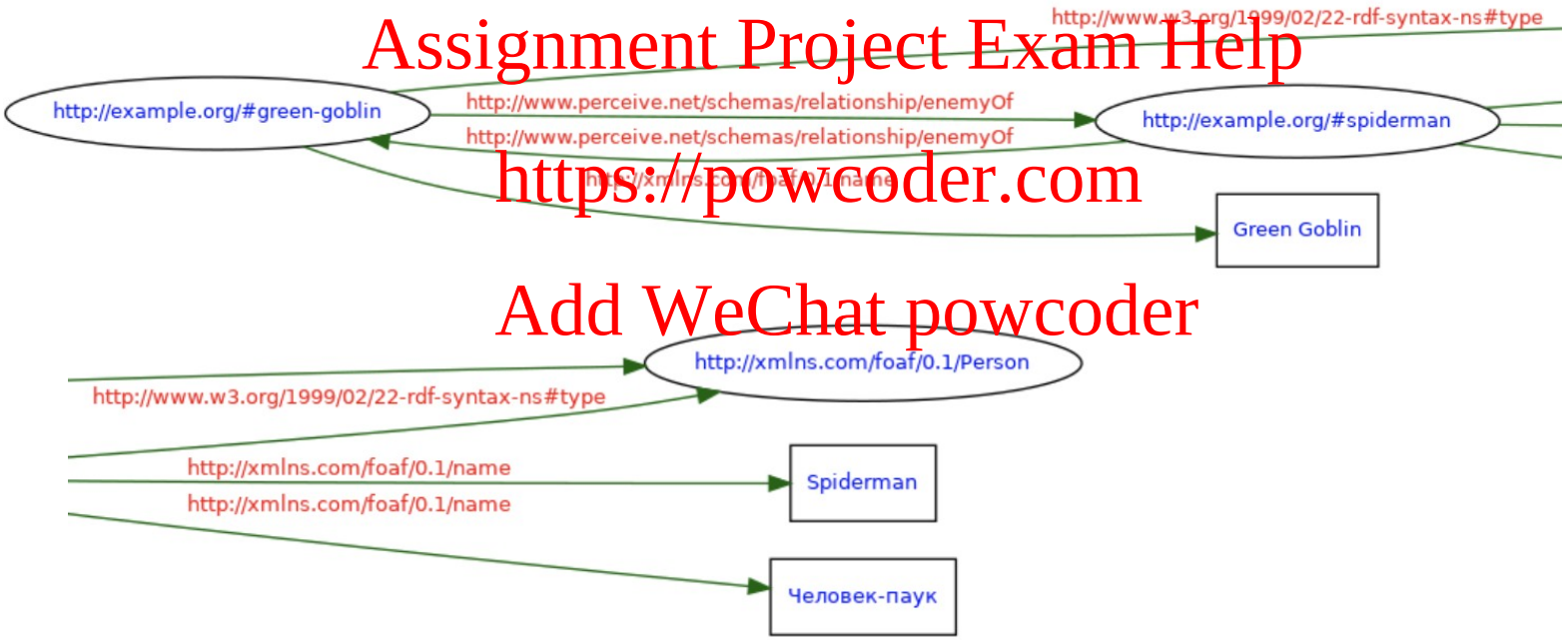

Green Goblin and Spiderman!

RDF/XML representation

```
1.  <?xml version="1.0" encoding="utf-8" ?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.           xmlns:foaf="http://xmlns.com/foaf/0.1/"
4.           xmlns:ns0="http://www.perceive.net/schemas/relationship/">
5.  .org/#green-goblin">
6.     <ns0:enemyOf><foaf:Person rdf:about="http://example
7.       <foaf:Person rdf:about="http://example.org/#spiderman">
8.         <ns0:enemyOf rdf:resource="http://example.org/#green-goblin"/>
9.         <foaf:name>Spiderman</foaf:name>
10.        <foaf:name xml:lang="ru">Человек-паук</foaf:name>
11.      </foaf:Person>
12.    </ns0:enemyOf>
13.    <foaf:name>Green Goblin</foaf:name>
14.  </foaf:Person>
15. </rdf:RDF>
```


Green Goblin and Spiderman!

Graph representation



Green Goblin and Spiderman!

Summary

- This example introduced many of features of the Turtle language: @base and Relative IRIs, @prefix and prefixed names, predicate lists separated by ';', object lists separated by ',', the token a, and literals.

- The Turtle grammar for triples is a subset of the SPARQL 1.1 Query Language grammar for TriplesBlock.

Simple Triples

The simplest triple statement is a sequence of (subject, predicate, object) terms, separated by whitespace and terminated by '.' after each triple.

<http://example.org/#spiderman>
<http://www.perceive.net/schemas/relationship/enemyOf>
<http://example.org/#green-goblin> .

Predicate Lists

- The subject will be referenced by a number of predicates and objects, separated by ';'.

<http://example.org/#spiderman>
<http://www.perceive.net/schemas/relationship/enemyOf>
<http://example.org/#green-goblin> ;
<http://xmlns.com/foaf/0.1/name>
"Spiderman" .

<http://example.org/#spiderman>
<http://www.perceive.net/schemas/relationship/enemyOf>
<http://example.org/#green-goblin> .
<http://example.org/#spiderman>
<http://xmlns.com/foaf/0.1/name>
"Spiderman" .

Object Lists

- A series of objects separated by ',' following a predicate.

```
<http://example.org/#spiderman> <http://xmlns.com/foaf/0.1/name> "Spiderman" ,
"Человек-паук"@ru .

<http://example.org/#spiderman> <http://xmlns.com/foaf/0.1/name> "Spiderman" .
<http://example.org/#spiderman> <http://xmlns.com/foaf/0.1/name> "Человек-
паук"@ru .
```

IRIs

- Define a prefix label for the vocabulary IRI

@prefix somePrefix: <http://www.perceive.net/schemas/relationship/> .
<http://example.org/#green-goblin>
somePrefix:enemyOf
<http://example.org/#spiderman> .

- Define a prefix label using the SPARQL syntax

PREFIX somePrefix: <http://www.perceive.net/schemas/relationship/>
<http://example.org/#green-goblin>
somePrefix:enemyOf
<http://example.org/#spiderman> .

Note!

- The '@prefix' and '@base' directives require a trailing '.' after the IRI, the equalivent 'PREFIX' and 'BASE' must not have a trailing '.' after the IRI part of the directive.

Write a Turtle representation of the following RDF/XML document. Validate your answer here:

http://ttl.summerofcode.be/

Example

RDF/XML

```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.  xmlns:uni="http://www.mydomain.org/unins/">
4.  <rdf:Description rdf:about="http://www.mydomain.org/unins#CIT1111">
5.        <rdf:type rdf:resource="http://www.mydomain.org/unins#course"/>
6.        <uni:courseName>Discrete Maths</uni:courseName>
7.        <uni:isTaughtBy rdf:resource="http://www.mydomain.org/unins#T949318"/>
8.    </rdf:Description>
9.    <rdf:Description rdf:about="http://www.mydomain.org/unins#T949318">
10.       <rdf:type rdf:resource="http://www.mydomain.org/unins#lecturer"/>
11.       <uni:name>David Billington</uni:name>
12.       <uni:title>Associate Professor</uni:title>
13.   </rdf:Description>
14. </rdf:RDF>
```

Example

Turtle

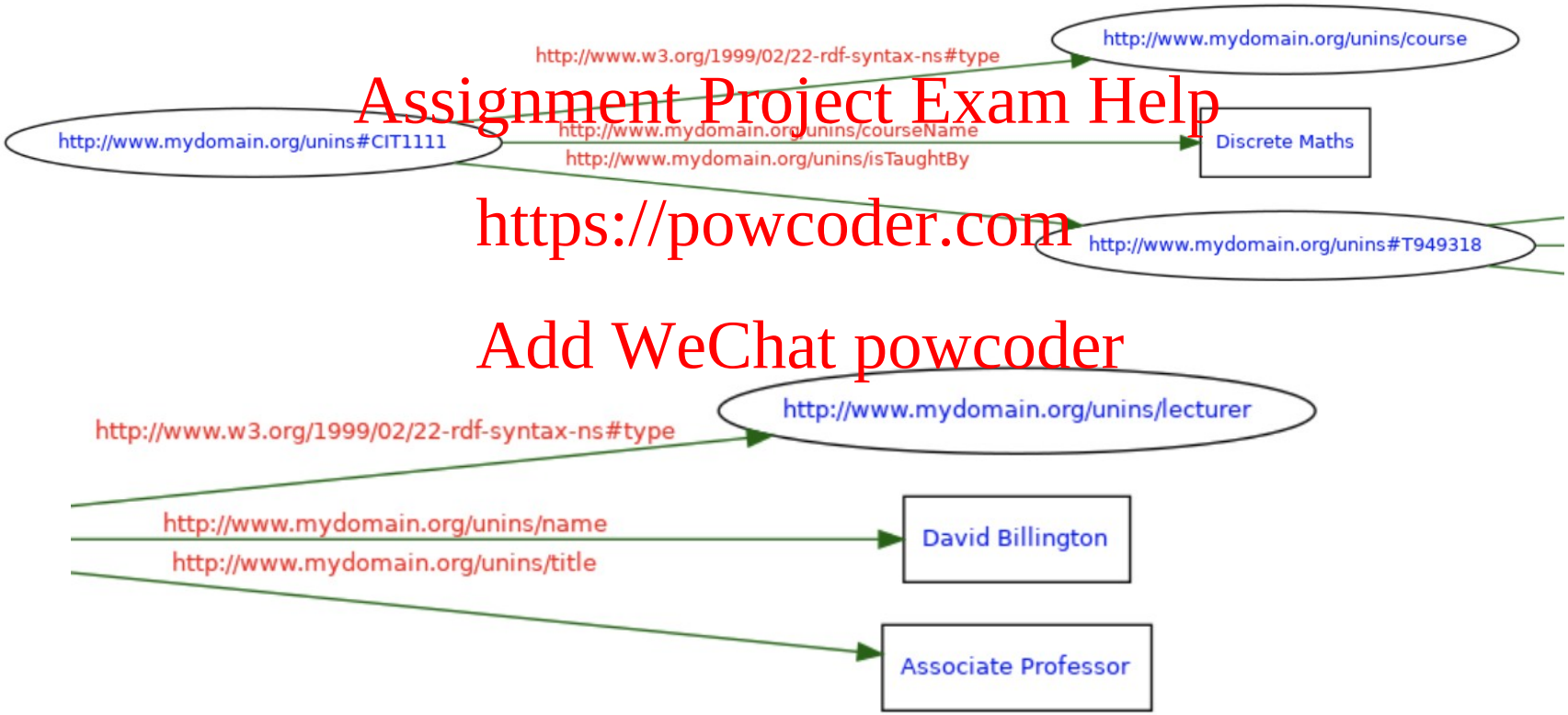
```
1. @prefix uni: <http://www.mydomain.org/unins/> .
2. @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3. @base <http://www.mydomain.org/unins> .

4. <#CIT1111>
5. a uni:course;
6. uni:courseName "Discrete Maths" ;
7. uni:isTaughtBy <#T949318> .
8. <#T949318>
9. a uni:lecturer;
10. uni:name "David Billington" ;
11. uni:title "Associate Professor" .
```

Is this statement correct?

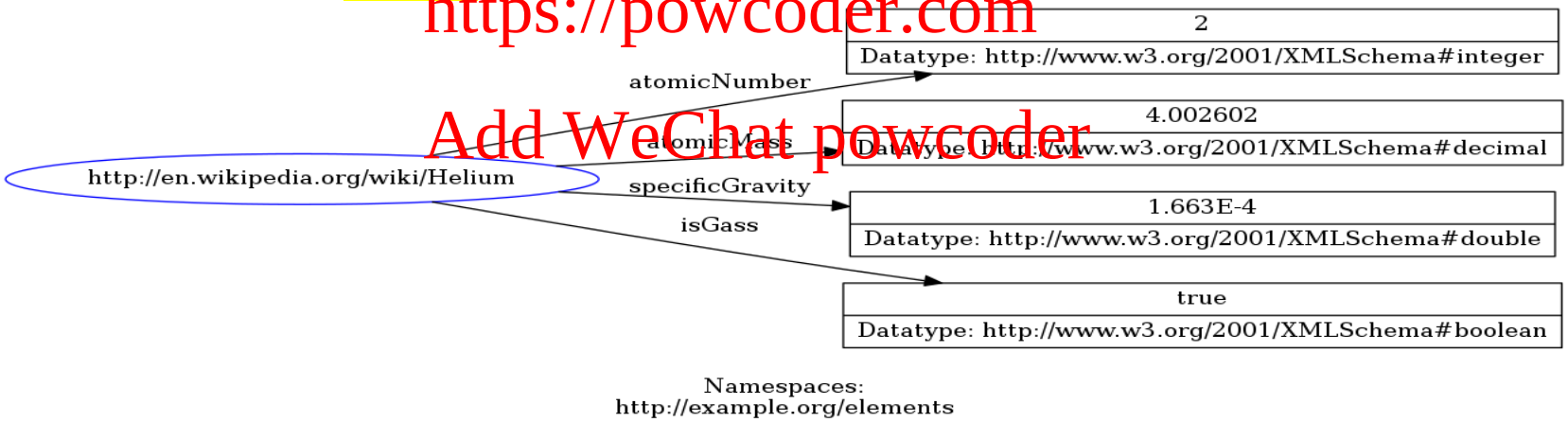
For a given RDF/XML document, there exists only one Turtle representation.

Example
Graph representation



Literals

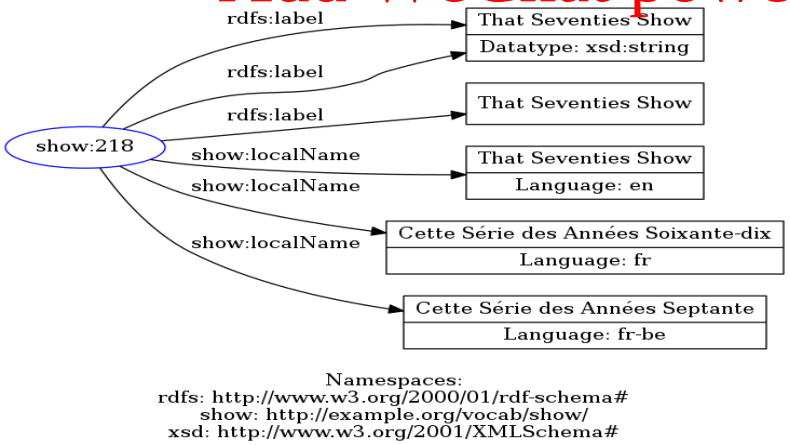
```
@prefix : <http://example.org/elements> .
<http://en.wikipedia.org/wiki/Helium>
    :atomicNumber 2 ; # xsd:integer
    :atomicMass 4.002602 ; # xsd:decimal
    :specificGravity 1.663E-4 ; # xsd:double
    :isGass true . # xsd:boolean
```

http://www.ldf.fi/service/rdf-grapher

Literals

1. @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2. @prefix show: <http://example.org/vocab/show/> .
3. @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
4. show:218 rdfs:label "That Seventies Show"^^xsd:string . # literal with XML Schema string datatype
5. show:218 rdfs:label "That Seventies Show"^^<http://www.w3.org/2001/XMLSchema#string> . # same as above
6. show:218 rdfs:label "That Seventies Show" . # same again
7. show:218 show:localName "That Seventies Show"@en . # literal with a language tag
8. show:218 show:localName 'Cette Série des Années Soixante-dix'@fr . # literal delimited by single quote
9. show:218 show:localName "Cette Série des Années Septante"@fr-be . # literal with a region subtag



http://www.ldf.fi/service/rdf-grapher

Blank Nodes

- RDF blank nodes in Turtle are expressed as _: followed by a blank node label which is a series of name characters.


```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:alice foaf:knows _:bob .
_:bob foaf:knows _:alice .
```

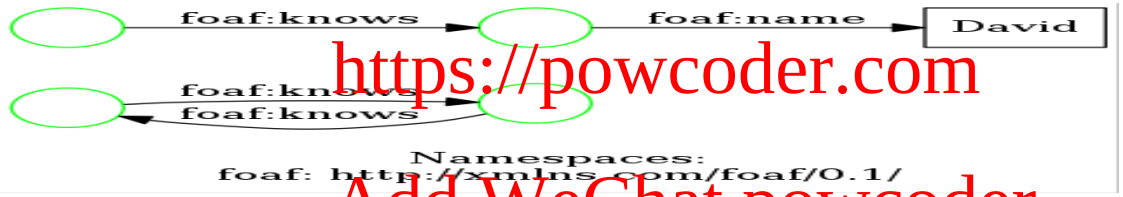


```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
# Someone knows someone else, who has the name “David".
[] foaf:knows [ foaf:name “David" ] .
```

Blank Nodes

Graph Representation

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:alice foaf:knows _:bob .
_:bob foaf:knows _:alice .
```


```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
# Someone knows someone else, who has the name “David".
[] foaf:knows [ foaf:name “David" ] .
```

Nesting Unlabeled Blank Nodes

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
[ foaf:name "Alice" ] foaf:knows
[ foaf:name "Bob" ; foaf:knows [ foaf:name "Eve" ] ;
foaf:mbox <bob@example.com> ] .
```

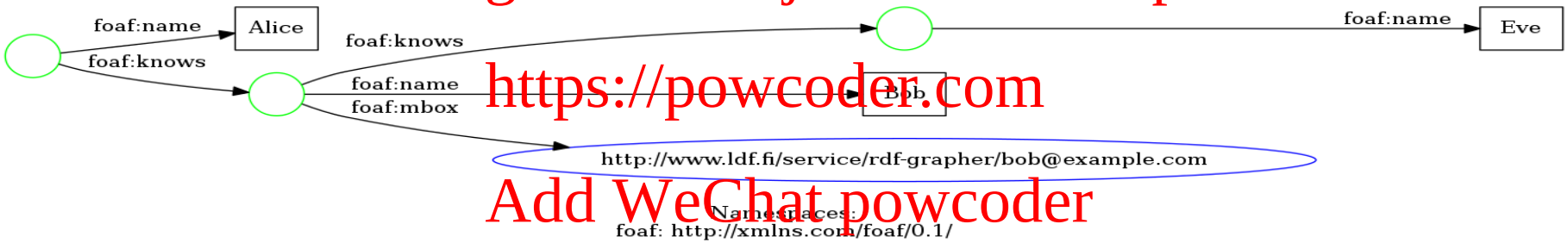


```
_:a <http://xmlns.com/foaf/0.1/name> "Alice" .
_:a <http://xmlns.com/foaf/0.1/knows> _:b .
_:b <http://xmlns.com/foaf/0.1/name> "Bob" .
_:b <http://xmlns.com/foaf/0.1/knows> _:c .
_:c <http://xmlns.com/foaf/0.1/name> "Eve" .
_:b <http://xmlns.com/foaf/0.1/mbox> <bob@example.com> .
```

The Graph representation?

Nesting Unlabeled Blank Nodes

Graph Representation




```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
[ foaf:name "Alice" ] foaf:knows [ foaf:name "Bob" ; foaf:knows
[ foaf:name "Eve" ] ; foaf:mbox <bob@example.com> ] .
```

http://www.ldf.fi/service/rdf-grapher

Collections

```
@prefix : <http://example.org/foo> .
# the object of this triple is the RDF collection blank node
:subject :predicate ( :a :b :c ) .

# an empty collection value - rdf:nil
:subject :predicate2 () .
```

More examples: https://www.w3.org/TR/turtle/

The Graph representation?

Lecture Outline

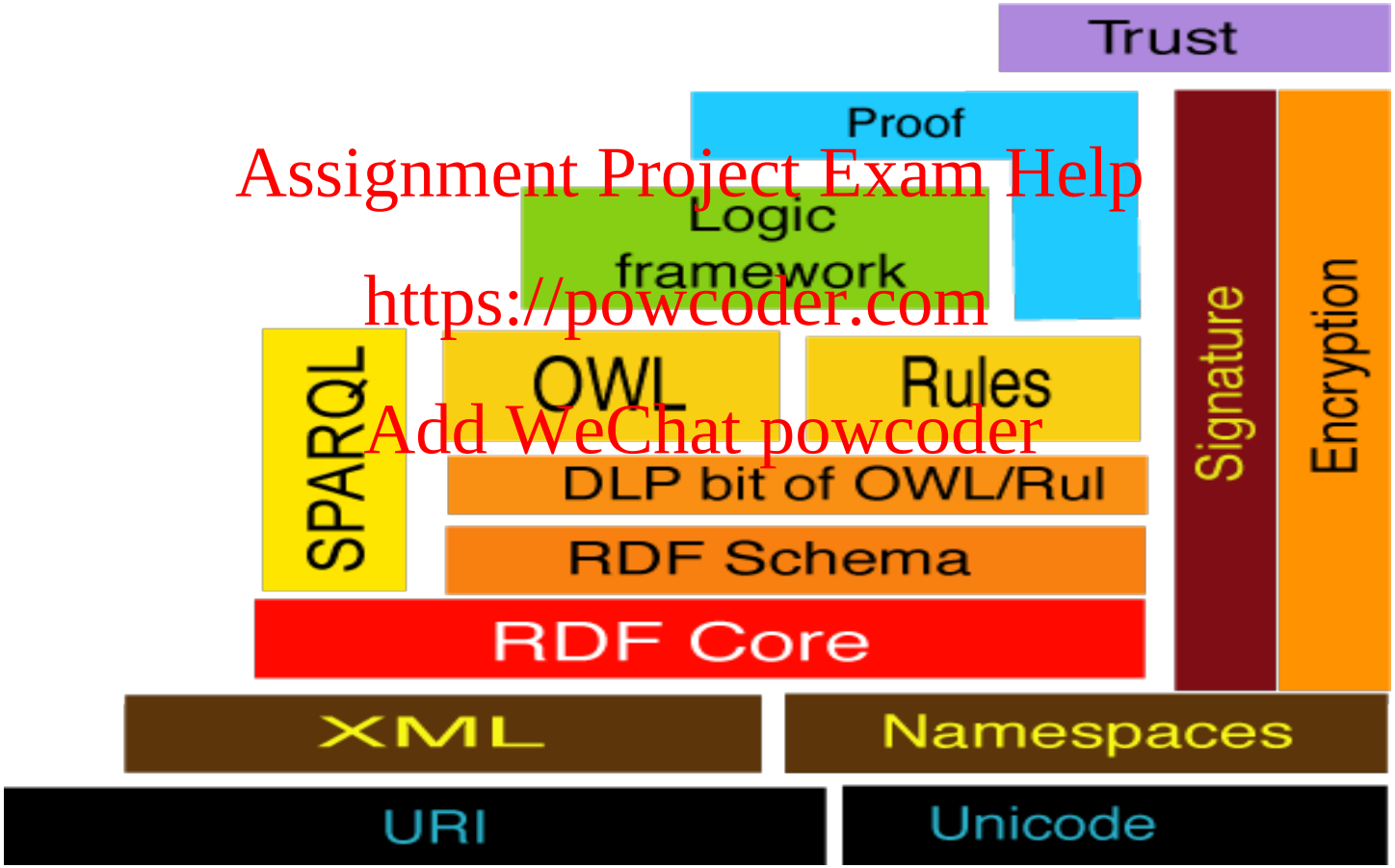
1. Turtle Syntax of RDF
2. **SPARQL: Querying RDF Documents**

SPARQL

Querying RDF Documents

The contents are mainly from:
http://www.w3.org/TR/rdf-sparql-query/

A Semantic Web Layer Stack



Query Forms

- SPARQL has four query forms. These query forms use the solutions from pattern matching to form result sets or RDF graphs.
- SELECT Returns all, or a subset of, the variables bound in a query pattern match.
- CONSTRUCT Returns an RDF graph constructed by substituting variables in a set of triple templates.
- ASK Returns a boolean indicating whether a query pattern matches or not.
- DESCRIBE Returns an RDF graph that describes the resources found.

- https://www.youtube.com/watch?time_continue=4&v=L_eB7Z84M4c&feature=emb_logo

Conventions

Namespaces

The following slides assume the following namespace prefix bindings:

Prefix	IRI
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
xsd:	http://www.w3.org/2001/XMLSchema#
fn:	http://www.w3.org/2005/xpath-functions#

Conventions

Data Descriptions

- We use the Turtle [TURTLE] data format to show each triple explicitly. Turtle allows IRIs to be abbreviated with prefixes:


```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix: <http://example.org/book/> . :book1 dc:title "SPARQL Tutorial" .
```

SPARQL Basics

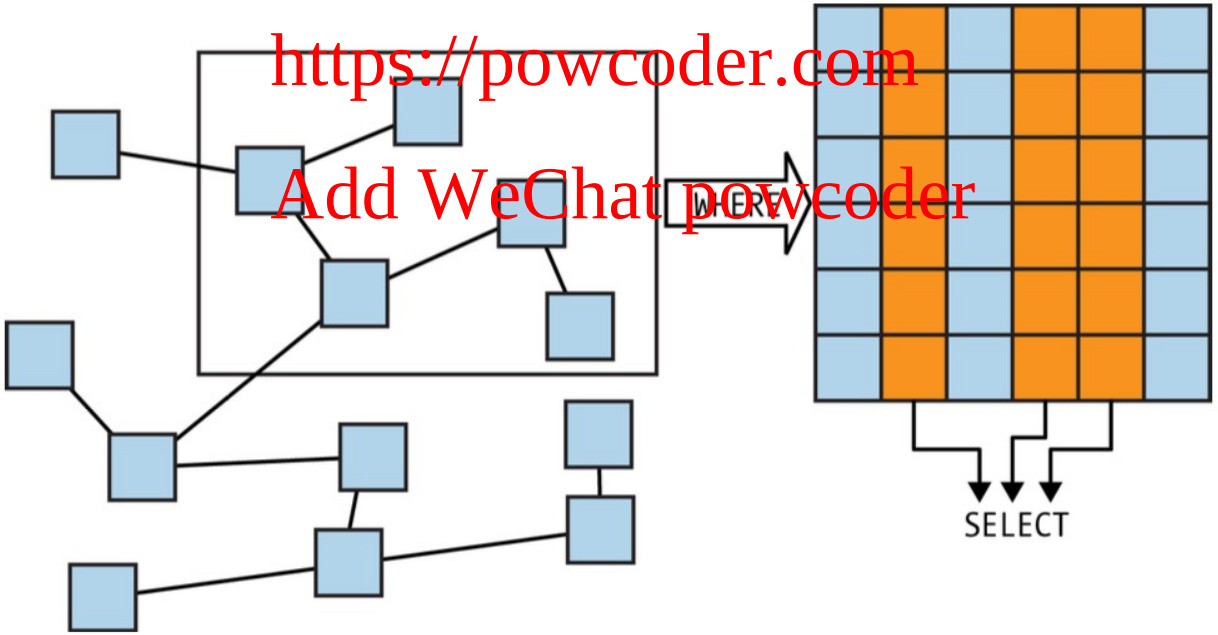
- ? or $ for variables
- SELECT for selection, WHERE for conditions
- WHERE typically combines multiple conditions, over multiple subjects
  - In SQL this would be a JOIN
  - without the headache you get by using JOIN

Query Forms (Select)

- The query consists of two parts:
- The SELECT clause identifies the variables to appear in the query results, and
- The WHERE clause provides the basic graph pattern to match against the data graph.

Query Forms (Select)

- SPARQL query's WHERE clause says "pull this data out of the dataset," and the SELECT part names which parts of that pulled data you actually want to see.



Learning SPARQL, 2nd Edition

Writing a Simple Query

- Find the title of a book from the given data graph.

Data:

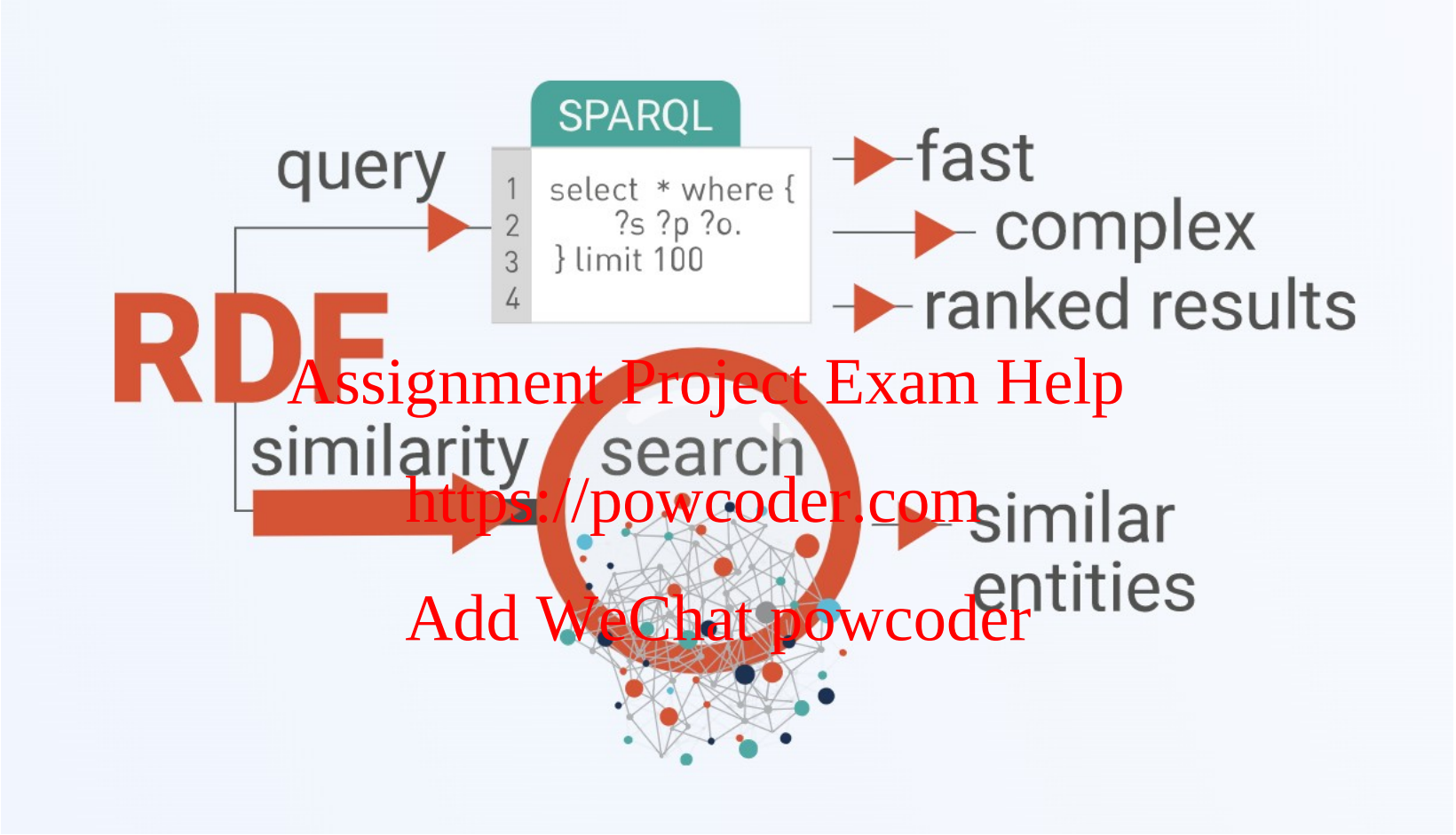
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .

Query:

SELECT ?title WHERE { <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title.}

Result:

"SPARQL Tutorial"



https://www.ontotext.com/products/graphdb/graphdb-free/?hsCtaTracking=394693e6-2c2a-4b0f-8dcd-a5c322997796%7C7148c241-badf-4c36-8449-faf12dca6d1f

Multiple Matches

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Johnny Lee Outlaw" .
_:a foaf:mbox <mailto:jlow@example.com> .
_:b foaf:name "Peter Goodguy" .
_:b foaf:mbox <mailto:peter@example.org> .
_:c foaf:mbox <mailto:carol@example.org> .
```


Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox WHERE {?x foaf:name ?name . ?x foaf:mbox ?mbox}
```

Result:

name	mbox
"Johnny Lee Outlaw"	<mailto:jlow@example.com>
"Peter Goodguy"	<mailto:peter@example.org>

Matching RDF Literals

Matching Literals with Language Tags

Data:

```
@prefix dt: <http://example.org/datatype#> .
@prefix ns: <http://example.org/ns#> .
@prefix : <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
:x ns:p "cat"@en .
:y ns:p "42"^^xsd:integer .
:z ns:p "abc"^^dt:specialDatatype .
```


Query:

```
SELECT ?v WHERE { ?v ?p "cat"}
```

Result:

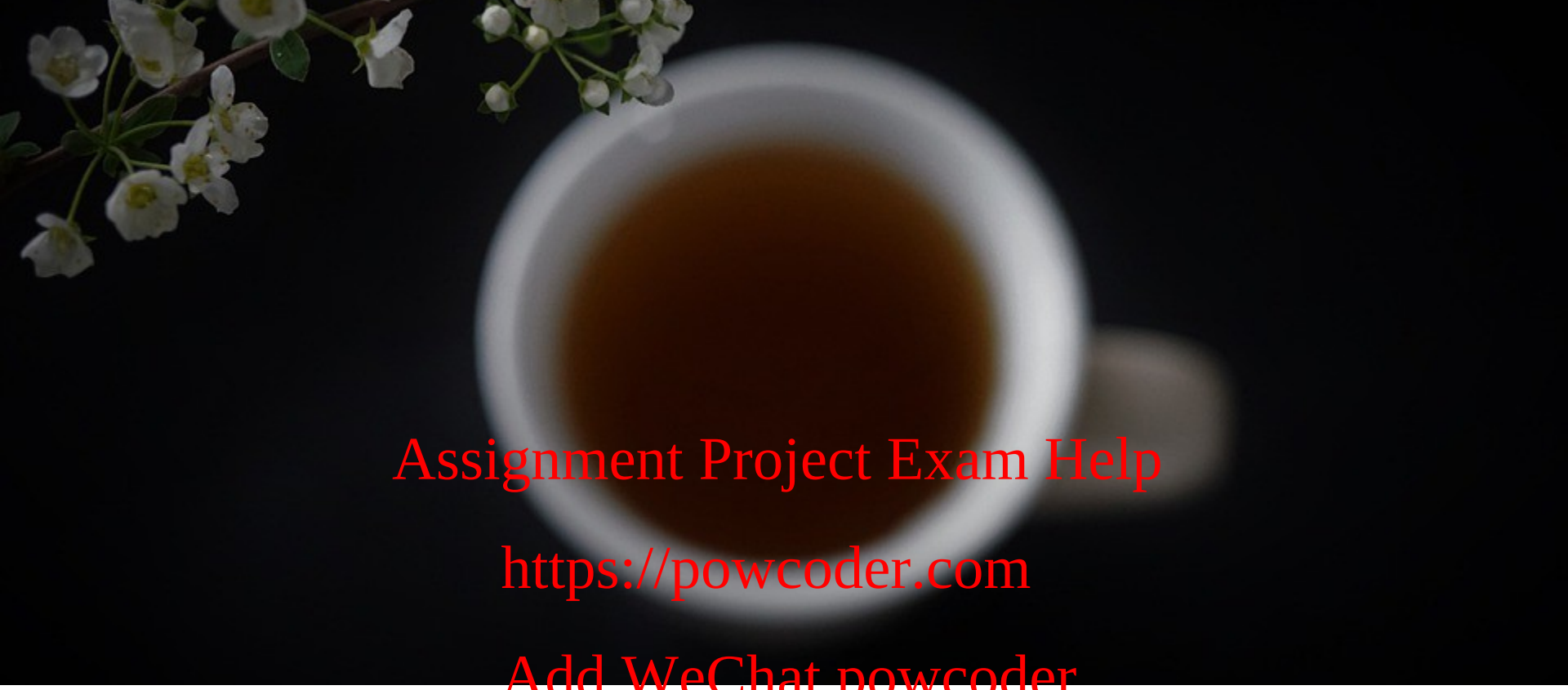
???

Query:

```
SELECT ?v WHERE { ?v ?p "cat"@en}
```

Result:

v
<http://example.org/ns#x>



- Take a break ☺

Matching RDF Literals

Matching Literals with Numeric Types

Data:

```
@prefix dt: <http://example.org/datatype#> .
@prefix ns: <http://example.org/ns#> .
@prefix : <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
:x ns:p "cat"@en .
:y ns:p "42"^^xsd:integer .
:z ns:p "abc"^^dt:specialDatatype .
```


Query:

```
SELECT ?v WHERE {?v ?p 42}
```

Result:

v
<http://example.org/ns#y>

Matching RDF Literals

Matching Literals with Arbitrary Datatypes

Data:

```
@prefix dt: <http://example.org/datatype#> .
@prefix ns: <http://example.org/ns#> .
@prefix : <http://example.org/ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
:x ns:p "cat"@en .
:y ns:p "42"^^xsd:integer .
:z ns:p "abc"^^dt:specialDatatype .
```


Query:

```
SELECT ?v WHERE {?v ?p "abc"^^<http://example.org/datatype#specialDatatype>}
```

Result:

v
<http://example.org/ns#z>

Blank Node Labels in Query Results

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
 _:a foaf:name "Alice" .
_:b foaf:name "Bob" .
```


Query:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?x ?name WHERE { ?x foaf:name ?name }
```

Result:

x	name
_:c	"Alice"
_:d	"Bob"

RDF Term Constraints

- Graph pattern matching produces a solution sequence, where each solution has a set of bindings of variables to RDF terms. SPARQL FILTERs restrict solutions to those for which the filter expression evaluates to TRUE.
- SPARQL FILTER functions like regex can test RDF literals. regex matches only plain literals with no language tag.
- https://www.w3.org/TR/xpath-functions/#regex-syntax

RDF Term Constraints

Restricting the Values of Strings

Data:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```


Query:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title WHERE {?x dc:title ?title FILTER regex(?title, "^SPA.*L$", “i”) }
```

Result:

title
"SPARQL Tutorial"

RDF Term Constraints

Restricting Numeric Values

Data:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```


Query:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price WHERE { ?x ns:price ?price . FILTER (?price < 30.5) . ?x dc:title ?title . }
```

Result:

title	price
"The Semantic Web"	23

Different Queries Same Result

Query 1:

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT  $title WHERE   {ns:book2  dc:title  $title}
```


Query 2:

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <http://example.org/ns#>
SELECT  $title WHERE   {:book2  dc:title  $title}
```


Query 3:


```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
SELECT  $title WHERE {<http://example.org/ns#book2>  dc:title  $title}
```

Query 4:

```
PREFIX  dc: <http://purl.org/dc/elements/1.1/>
BASE  <http://example.org/ns#>
SELECT  $title WHERE {<#book2>  dc:title  $title}
```

Predicate-Object Lists

- Triple patterns with a common subject can be written so that the subject is only written once and is used for more than one triple pattern using ";"

Pattern	Simplified Pattern
?x foaf:name ?name . ?x foaf:mbox ?mbox .	?x foaf:name ?name ; foaf:mbox ?mbox .

Object Lists

- If triple patterns share both subject and predicate, the objects may be separated by ","

Pattern	Simplified Pattern
?x foaf:nick "Alice" . ?x foaf:nick "Alice_" .	?x foaf:nick "Alice" , "Alice_" .

Simplify the following SPARQL query pattern.

Pattern

```
?x foaf:name ?name .
?x foaf:nick "Alice" .
?x foaf:nick "Alice_" .
```

Predicate-Object List Combined with Object Lists

Pattern

```
?x foaf:name ?name .
?x foaf:nick "Alice" .
?x foaf:nick "Alice_" .
```

1

Simplified Pattern

```
?x foaf:name ?name ;
foaf:nick "Alice" ;
foaf:nick "Alice_" .
```


2

Simplified Pattern

```
?x foaf:name ?name ;
foaf:nick "Alice", "Alice_" ;
```

Optional Pattern Matching

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
_:a rdf:type foaf:Person .
_:a foaf:name "Alice" .
_:a foaf:mbox <mailto:alice@example.com> .
_:a foaf:mbox <mailto:alice@work.example> .
_:b rdf:type foaf:Person .
_:b foaf:name "Bob" .
```


Query :

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

Result:

name	mbox
"Alice"	<mailto:alice@example.com>
"Alice"	<mailto:alice@work.example>
"Bob"	

Modify the query in the previous slide to remove null values from the result.

Optional Pattern Matching

Constraints

Data:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .
:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```


Query :

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX ns: <http://example.org/ns#>
SELECT ?title ?price WHERE {?x dc:title ?title . OPTIONAL { ?x ns:price ?price . FILTER (?price < 30) } }
```

Result:

title	price
"SPARQL Tutorial"	
"The Semantic Web"	23

Multiple Optional Patterns

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:homepage <http://work.example.org/alice/> .
_:b foaf:name "Bob" .
_:b foaf:mbox <mailto:bob@work.example> .
```


Query :


```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox ?hpage WHERE { ?x foaf:name ?name . OPTIONAL { ?x foaf:mbox ?mbox } .
OPTIONAL { ?x foaf:homepage ?hpage } }
```


Result:

name	mbox	hpage
"Alice"		<http://work.example.org/alice/>
"Bob"	<mailto:bob@work.example>	

Matching Alternatives

Data:

```
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
_:a dc10:title "SPARQL Query Language Tutorial" .
_:a dc10:creator "Alice" .
_:b dc11:title "SPARQL Protocol Tutorial" .
_:b dc11:creator "Bob" .
_:c dc10:title "SPARQL" .
_:c dc11:title "SPARQL (updated)" .
```


Query :

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?title WHERE { { ?book dc10:title ?title } UNION { ?book dc11:title ?title } }
```

Result:

title
"SPARQL Protocol Tutorial"
"SPARQL"
"SPARQL (updated)"
"SPARQL Query Language Tutorial"

Matching Alternatives

Data:

```
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
_:a dc10:title "SPARQL Query Language Tutorial" .
_:a dc10:creator "Alice" .
_:b dc11:title "SPARQL Protocol Tutorial" .
_:b dc11:creator "Bob" .
_:c dc10:title "SPARQL" .
_:c dc11:title "SPARQL (updated)" .
```


Query :

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?x ?y WHERE { { ?book dc10:title ?x } UNION { ?book dc11:title ?y } }
```

Result:

x	y
	"SPARQL (updated)"
	"SPARQL Protocol Tutorial"
"SPARQL"	
"SPARQL Query Language Tutorial"	

Matching Alternatives

Data:

```
@prefix dc10: <http://purl.org/dc/elements/1.0/> .
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
_:a dc10:title "SPARQL Query Language Tutorial" .
_:a dc10:creator "Alice" .
_:b dc11:title "SPARQL Protocol Tutorial" .
_:b dc11:creator "Bob" .
_:c dc10:title "SPARQL" .
_:c dc11:title "SPARQL (updated)" .
```


Query :

```
PREFIX dc10: <http://purl.org/dc/elements/1.0/>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
SELECT ?title ?author WHERE { { ?book dc10:title ?title . ?book dc10:creator ?author } UNION { ?
book dc11:title ?title . ?book dc11:creator ?author } }
```

Result:

author	title
"Alice"	"SPARQL Protocol Tutorial"
"Bob"	"SPARQL Query Language Tutorial"

Modify the query in the previous slide to include all titles with or without authors.

Write a query for the following.

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:knows _:b .
_:a foaf:knows _:c .
_:b foaf:name "Bob" .
_:c foaf:name "Clare" .
_:c foaf:nick "CT" .
```


Query :

???

Result:

nameX	nameY	nickY
"Alice"	"Bob"	
"Alice"	"Clare"	"CT"

Data:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
_:a foaf:name "Alice" .
_:a foaf:knows _:b .
_:a foaf:knows _:c .
_:b foaf:name "Bob" .
_:c foaf:name "Clare" .
_:c foaf:nick "CT" .
```


Query :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?nameX ?nameY ?nickY WHERE {
?x foaf:knows ?y ; foaf:name ?nameX . ?y foaf:name ?nameY . OPTIONAL { ?y foaf:nick
?nickY } }
```

Result:

nameX	nameY	nickY
"Alice"	"Bob"	
"Alice"	"Clare"	"CT"