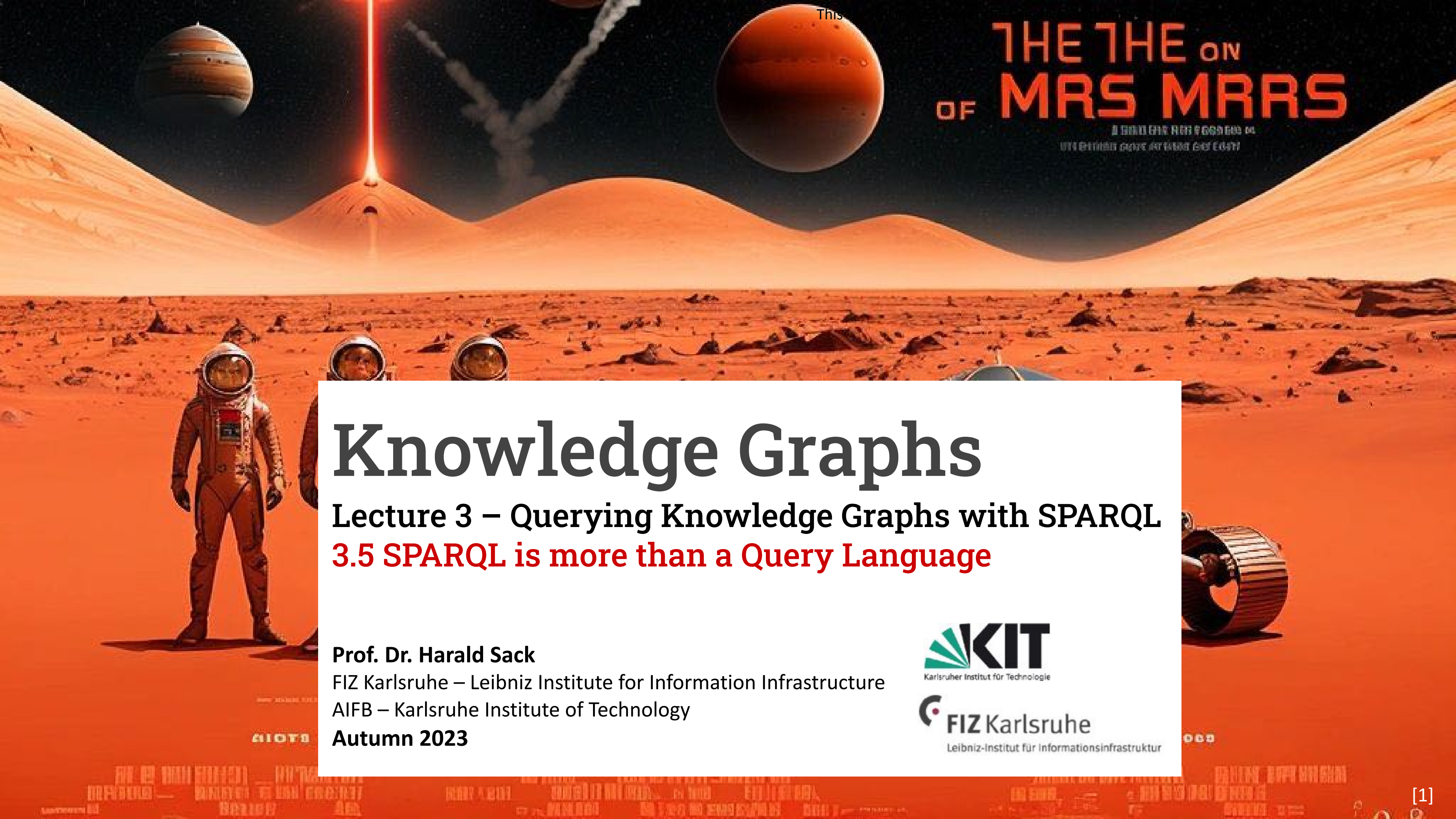
This



Knowledge Graphs

Lecture 3 – Querying Knowledge Graphs with SPARQL

3.5 SPARQL is more than a Query Language

Prof. Dr. Harald Sack
FIZ Karlsruhe – Leibniz Institute for Information Infrastructure
AIFB – Karlsruhe Institute of Technology

Autumn 2023

KIT Karlsruher Institut für Technologie

FIZ Karlsruhe Leibniz-Institut für Informationsinfrastruktur

[1]

Knowledge Graphs

Lecture 3: Querying Knowledge Graphs with SPARQL

3.1 How to Query RDF(S)

Excursion 3: DBpedia Knowledge Graph

Excursion 4: Wikidata Knowledge Graph

3.2 Complex Queries with SPARQL

3.3 More Complex SPARQL Queries

3.4 SPARQL Sub-Select and Property Paths

3.5 SPARQL is more than a Query Language

3.6 Quality Assurance with SHACL Constraints

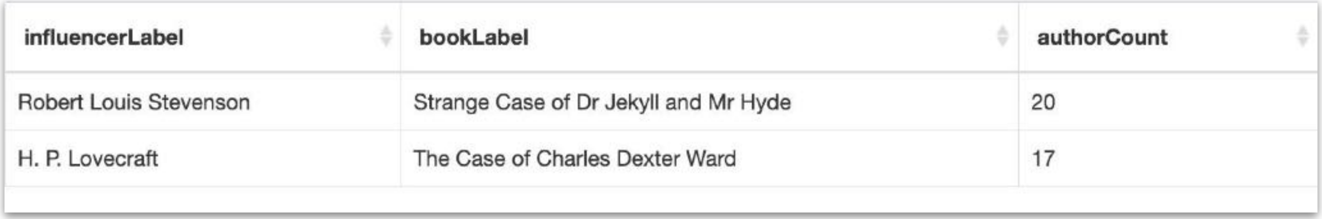
[1]

SPARQL Standards Overview

SPARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage version 1.1
is defined in 11 W3C Recommendation documents:

1. SPARQL 1.1 Overview
2. SPARQL 1.1 Query Language
3. SPARQL 1.1 Update
4. SPARQL1.1 Service Description
5. SPARQL 1.1 Federated Query
6. SPARQL 1.1 Query Results JSON Format
7. SPARQL 1.1 Query Results CSV and TSV Formats
8. SPARQL Query Results XML Format (Second Edition)
9. SPARQL 1.1 Entailment Regimes
10. SPARQL 1.1 Protocol
11. SPARQL 1.1 Graph Store HTTP Protocol

SPARQL Output Formats

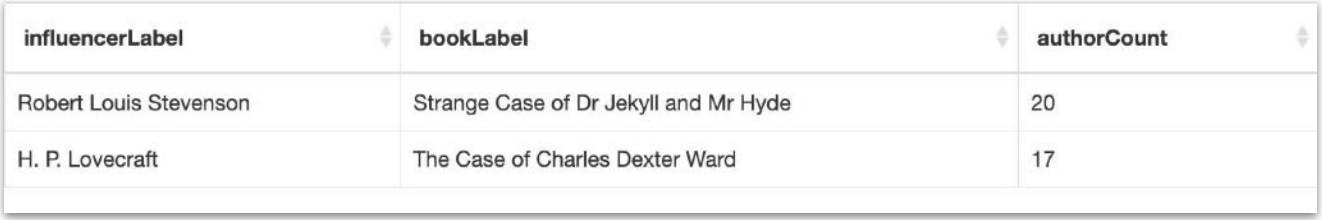


influencerLabel	bookLabel	authorCount
Robert Louis Stevenson	Strange Case of Dr Jekyll and Mr Hyde	20
H. P. Lovecraft	The Case of Charles Dexter Ward	17

- SPARQL Output in CSV based format:

```
influencerLabel,bookLabel,authorCount
“Robert Louis Stevenson”@en,”Strange Case of Dr Jekyll and Mr Hyde”@en,”20”^^<http://www.w3.org/2001/XMLSchema#Integer>
“H. P. Lovecraft”@en,”The Case of Charles Dexter Ward”@en,”17”^^<http://www.w3.org/2001/XMLSchema#Integer>
```

SPARQL Output Formats



influencerLabel	bookLabel	authorCount
Robert Louis Stevenson	Strange Case of Dr Jekyll and Mr Hyde	20
H. P. Lovecraft	The Case of Charles Dexter Ward	17

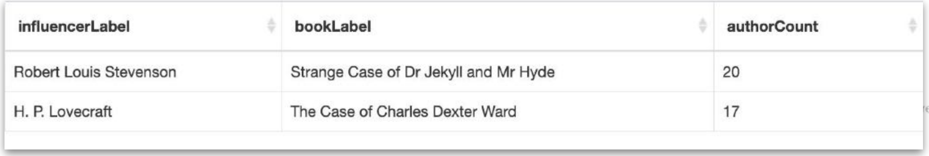
- SPARQL Output as well-formed XML document:

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
 ...
</sparql>
```

- In a `<head>` element all variables of the SPARQL query are listed.

```
<head>
    <variable name="influencerLabel"/>
    <variable name="bookLabel"/>
    <variable name="authorCount"/>
</head>
```

SPARQL Output Formats



influencerLabel	bookLabel	authorCount
Robert Louis Stevenson	Strange Case of Dr Jekyll and Mr Hyde	20
H. P. Lovecraft	The Case of Charles Dexter Ward	17

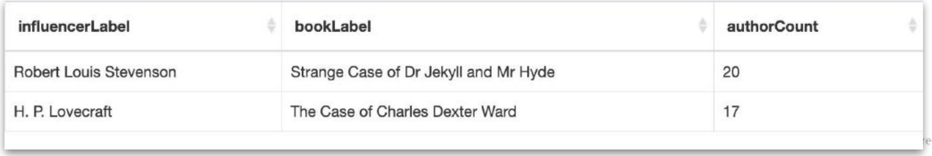
For each SPARQL Query result exists a **`<result>`** element.

```
<?xml version="1.0"?>
<sparql xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="influencerLabel"/>
    ...
  </head>
  <results>
    <result>
      <binding name="influencerLabel"> ... </binding>
      <binding name="bookLabel"> ... </binding>
      <binding name="authorCount"> ... </binding>
    </result>

    <result> ... </result>
    ...
  </results>
</sparql>
```

single SPARQL query result

SPARQL Output Formats



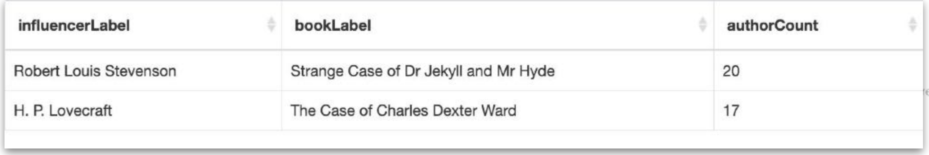
influencerLabel	bookLabel	authorCount
Robert Louis Stevenson	Strange Case of Dr Jekyll and Mr Hyde	20
H. P. Lovecraft	The Case of Charles Dexter Ward	17

Within a **<binding>** element a **<head>** variable is bound to a result.

```
...
<result>
      <binding name="influencerName">
        <literal xml:lang="en">
         Robert Louis Stevenson
        </literal>
      </binding>
      <binding name="bookName">
         <literal xml:lang="en">
         Strange Case of Dr Jekyll and Mr Hyde
        </literal>
      </binding>
      <binding name="authorCount">
        <literal datatype="http://www.w3.org/2001/XMLSchema#integer">
         20
        </literal>
      </binding>
</result>
...
```

variable bound to result

SPARQL Output Formats



influencerLabel	bookLabel	authorCount
Robert Louis Stevenson	Strange Case of Dr Jekyll and Mr Hyde	20
H. P. Lovecraft	The Case of Charles Dexter Ward	17

SPARQL Output as JSON document:

```
{"head":{"vars":["influencerLabel","bookLabel","authorCount"]},
  "results":
    {"bindings":[
       {"influencerLabel":{"xml:lang":"en","type":"literal","value":"Robert Louis Stevenson"
                         },
        "bookLabel":{"xml:lang":"en","type":"literal","value":"Strange Case of Dr Jekyll and Mr Hyde"
                    }
        "authorCount":{"datatype":"http://www.w3.org/2001/XMLSchema#integer","type":"literal","value":"20"
                      },
       },
       {"influencerLabel":{"xml:lang":"en","type":"literal","value":"H. P. Lovecraft"
                         },
         "bookLabel":{"xml:lang":"en","type":"literal","value":"The Case of Charles Dexter Ward"
                     }
         "authorCount":{"datatype":"http://www.w3.org/2001/XMLSchema#integer","type":"literal","value":"17"
                      },
      } ]
   }
}
```

SPARQL Protocol

- Method to query/respond to SPARQL queries via http
- A SPARQL URI consists of 3 parts:

  (1) URL of a SPARQL endpoint (e.g. `http://example.org/sparql`)

  (2) RDF Graph(s) to be queried
  (optional, part of the query string,
  e.g. `named-graph-uri=http://example.org/testrdf.rdf`)

  (3) SPARQL query
  (part of the query string, e.g. `query=SELECT`...)

```
http://example.org/sparql?named-graph-uri=http%3A%2F%2Fexample.org%2Ftestrdf&
query=SELECT+%3Freview_graph+WHERE+%7B%0D%0A++GRAPH+%3Frev
iew_graph+%7B%0D%0A+++++%3Freview+rev%3Arating+10+.%0D%0A++%7D%0D%0A%7D
```

SPARQL Protocol

- Simple SPARQL query

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author  ?work
WHERE {
       ?author rdf:type dbo:Writer ;
               dbo:notableWork ?work .
} LIMIT 100
```

- HTTP Trace of the SPARQL query

```
GET
https://dbpedia.org/sparql?default-graph-uri=https%3A%2F%2Fdbpedia.org&query=PREFIX+rdf%3A+%3Chttp%3A%2F%2Fwww.w3.org%2F1999%2F02%2F22-rdf-syntax-ns%23%3E%0D%0APREFIX+dbo%3A+%3Chttp%3A%2F%2Fdbpedia.org%2Fontology%2F%3E%0D%0ASELECT+%3Fauthor++%3Fwork%0D%0AWHERE+%7B%0D%0A+++++++%3Fauthor+rdf%3Atype+dbo%3AWriter+%3B%0D%0A+++++++++++++++dbo%3AnotableWork+%3Fwork+.%0D%0A%7D+LIMIT+100%0D%0A
Host: dbpedia.org
User-agent: Mozilla/5.0 …
Accept:text/html,application/xhtml+xml,application/xml
```

SPARQL is not only a Query Language

- In addition to SELECT queries, SPARQL allows:
- **ASK**
  - Check whether there is at least one result
  - Result: true or false
  - Result is delivered as XML or JSON

```
ASK
WHERE {
  ?author wdt:P106 wd:Q36180 ; # ?author :occupation :Writer
          wdt:P800 ?book .     #         :notableWork ?book
  ?book wdt:P31 wd:Q571 .       # ?book   :instanceOf :Book
}
```

[2]

- *Example:* Is there an author with a notable work?

query SPARQL endpoint

SPARQL is not only a Query Language

- In addition to `SELECT` queries, SPARQL allows:
- **DESCRIBE**
  - Result: an RDF graph with data about resources
  - Result is RDF/XML or Turtle

```
DESCRIBE ?author ?book
WHERE {
  ?author wdt:P106 wd:Q36180 ; # ?author :occupation :Writer
          wdt:P800 ?book .     #         :notableWork ?book
  ?book wdt:P31 wd:Q571 .       # ?book   :instanceOf :Book
} LIMIT 1000
```

[2]

- *Example:* Show all available data about authors and their notable works?

query SPARQL endpoint

SPARQL is not only a Query Language

- In addition to `SELECT` queries, SPARQL allows:
- **CONSTRUCT**
  - Result: an RDF graph constructed from a template
  - Template: graph pattern with variables from the query pattern
  - Result is RDF/XML or Turtle

```
CONSTRUCT { ?author <http://example.org/hasWritten> ?book .}
WHERE {
  ?author wdt:P106 wd:Q36180 ;
          wdt:P800 ?book .
  ?book wdt:P31 wd:Q571 .
} LIMIT 10
```

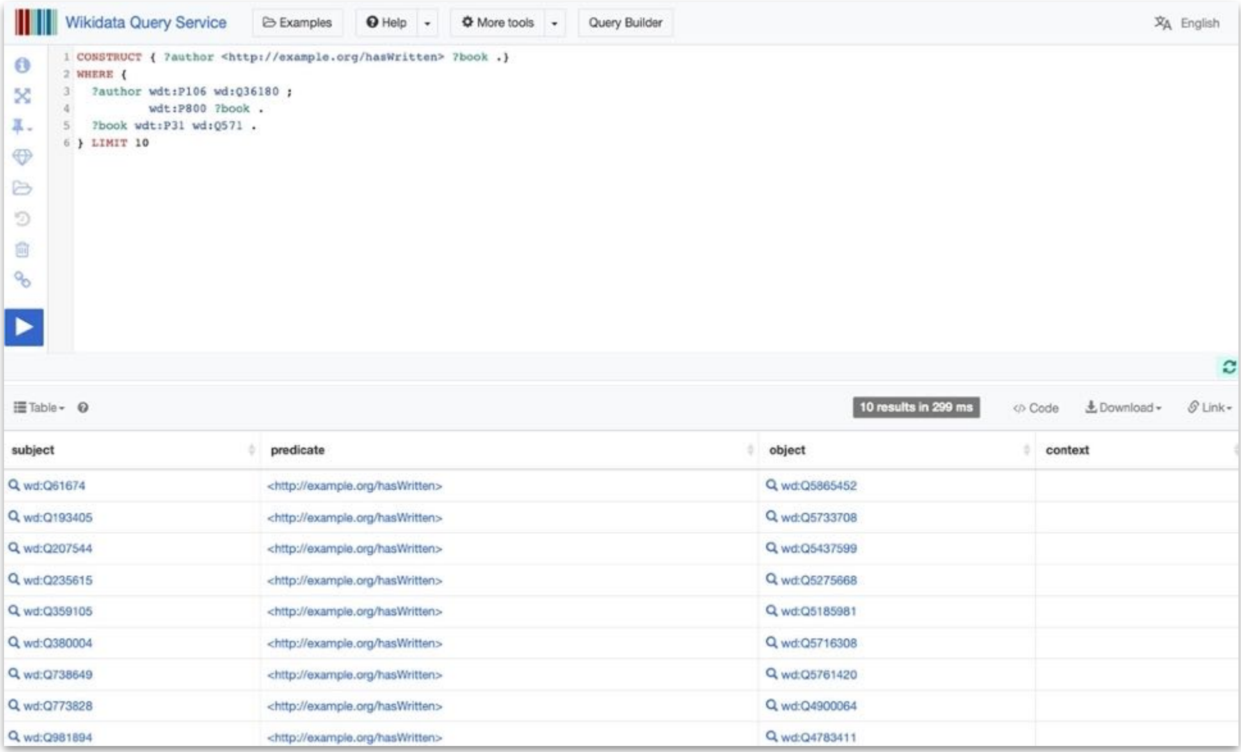
- *Example:* Create new RDF triples for authors and their notable works?

[2]

query SPARQL endpoint

SPARQL is not only a Query Language

Example: Create new RDF triples for authors and their notable works?



```
CONSTRUCT { ?author <http://example.org/hasWritten> ?book .}
WHERE {
  ?author wdt:P106 wd:Q36180 ;
          wdt:P800 ?book .
  ?book wdt:P31 wd:Q571 .
} LIMIT 10
```

subject	predicate	object	context
wd:Q61674	<http://example.org/hasWritten>	wd:Q5865452	
wd:Q193405	<http://example.org/hasWritten>	wd:Q5733708	
wd:Q207544	<http://example.org/hasWritten>	wd:Q5437599	
wd:Q235615	<http://example.org/hasWritten>	wd:Q5275668	
wd:Q359105	<http://example.org/hasWritten>	wd:Q5185981	
wd:Q380004	<http://example.org/hasWritten>	wd:Q5716308	
wd:Q738649	<http://example.org/hasWritten>	wd:Q5761420	
wd:Q773828	<http://example.org/hasWritten>	wd:Q4900064	
wd:Q981894	<http://example.org/hasWritten>	wd:Q4783411	

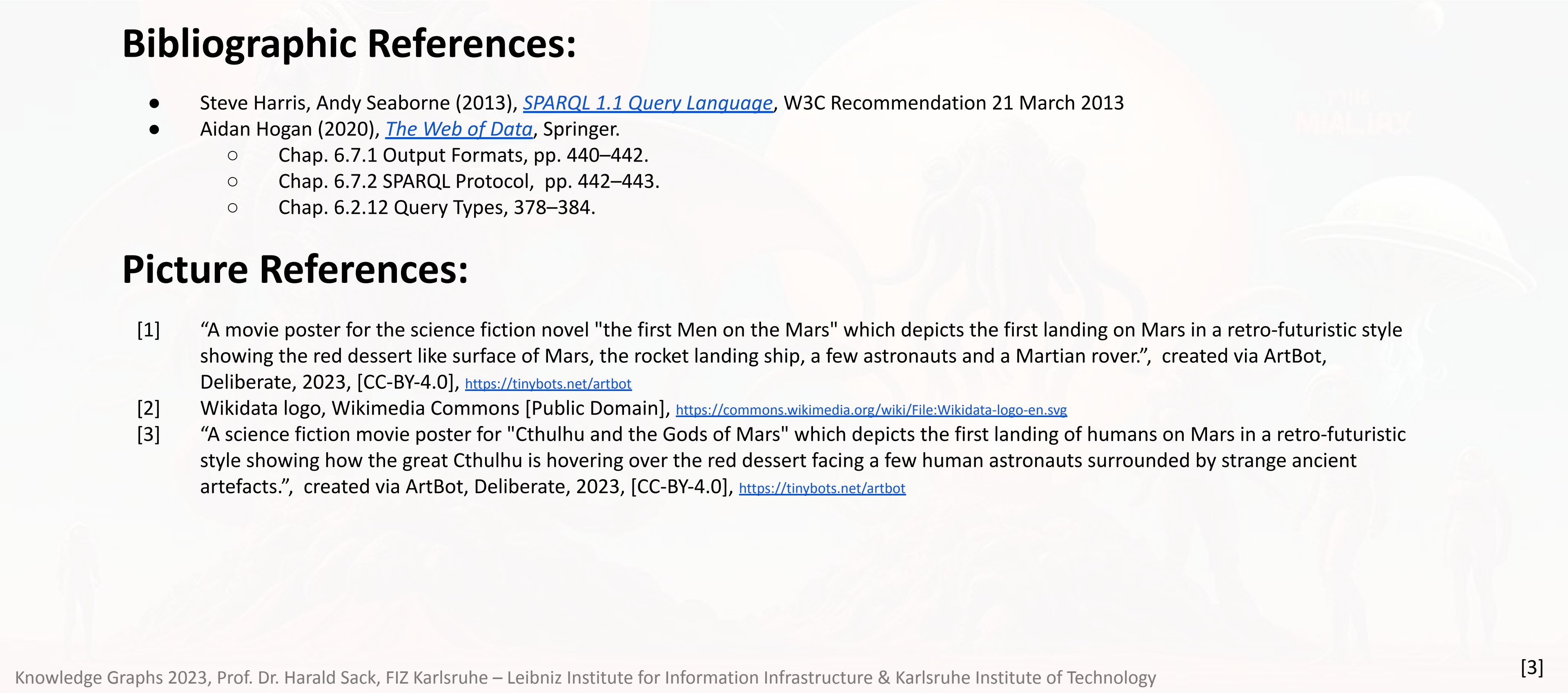
[2]

query SPARQL endpoint

Next Lecture...

Quality Assurance with SHACL Constraints

[3]



Bibliographic References:

- Steve Harris, Andy Seaborne (2013), *SPARQL 1.1 Query Language*, W3C Recommendation 21 March 2013
- Aidan Hogan (2020), *The Web of Data*, Springer.
  - Chap. 6.7.1 Output Formats, pp. 440–442.
  - Chap. 6.7.2 SPARQL Protocol, pp. 442–443.
  - Chap. 6.2.12 Query Types, 378–384.

Picture References:

[1] “A movie poster for the science fiction novel "the first Men on the Mars" which depicts the first landing on Mars in a retro-futuristic style showing the red dessert like surface of Mars, the rocket landing ship, a few astronauts and a Martian rover.”, created via ArtBot, Deliberate, 2023, [CC-BY-4.0], https://tinybots.net/artbot

[2] Wikidata logo, Wikimedia Commons [Public Domain], https://commons.wikimedia.org/wiki/File:Wikidata-logo-en.svg

[3] “A science fiction movie poster for "Cthulhu and the Gods of Mars" which depicts the first landing of humans on Mars in a retro-futuristic style showing how the great Cthulhu is hovering over the red dessert facing a few human astronauts surrounded by strange ancient artefacts.”, created via ArtBot, Deliberate, 2023, [CC-BY-4.0], https://tinybots.net/artbot

[3]