Analyse von „Wiki-Daten“: Wikipedia, DBpedia und Wikidata 3

OSD 1

Bettina Berendt

12.1.2024

Wo sind wir, und wo wollen wir hin?

- ✓ Wikipedia kann als Graph betrachtet werden.
- ✓ Wikipedia kann auch als Datenbank betrachtet werden.
- ✓ Ein wichtiger Teil dieser Datenbank ist Dbpedia.
- ✓ Wofür (1)? Ein Anwendungsbeispiel: Diversität in den Medien
- ✓ Dbpedia und Wikidata (Teil 1)
- ✓ Format: RDF, RDFS und z.T. OWL; Linked Open Data
  - ✓ Das kann auch als relationale Datenbank dargestellt werden (bzw. es können Auszüge generiert werden).
- Anfragesprache: SPARQL; mehr Wikidata
- Hausarbeit: Wofür (2)? Selber recherchieren und neue Information hinzufügen.

Fraunhofer
FOKUS

Technische
Universität
Berlin

Grundlagen Digitaler Vernetzung – Linked Data and Semantic Web: Data Management

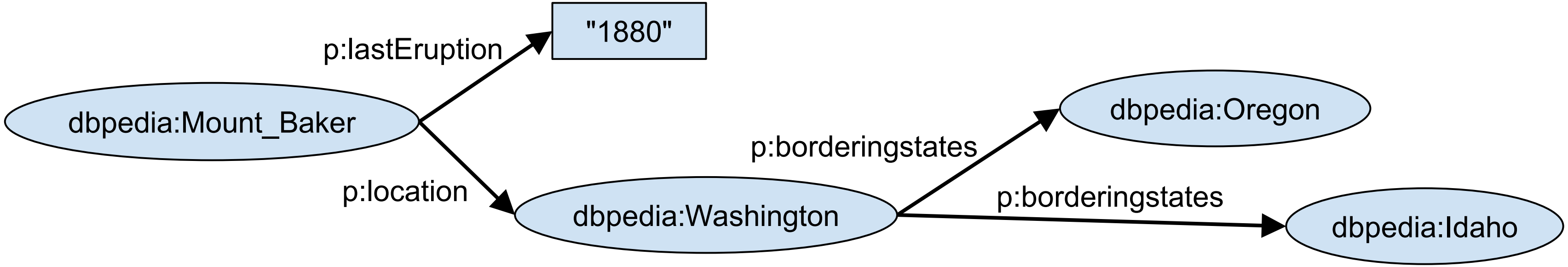
Manfred Hauswirth | Open Distributed Systems | Grundlagen Digitaler Verntzung | SoSe 2022

Open
Distributed
Sytems
ODS

RDF

- Atoms of knowledge are triples (subject, predicate, object)
  - Subject: resources (URI)
  - Predicate: properties (URI)
  - Object: resources (URI) or literals (string, value, number, etc.)
- RDF graph
  - Triples as directed graphs
  - Subjects and objects as vertices
  - Edges labeled by predicate

RDF



```
@prefix dbpedia : <http://dbpedia.org/resource/> .
@prefix p : <http://dbpedia.org/property/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

dbpedia:Mount_Baker          p:lastEruption   "1880"^^xsd:integer ;
                             p:location       dbpedia:Washington .

dbpedia:Washington           p:borderingstates   dbpedia:Oregon ,
                                                 dbpedia:Idaho .
```

RDF-based Data Processing

- RDF and Data Integration
- How to Query RDF(S) – SPARQL
  - Simple & Complex Queries with SPARQL
  - SPARQL Subqueries and Property Paths
  - Data Modification with SPARQL
- SPARQL is more than a query Language
- RDF Databases -> triplestores

RDF and Data Integration

RDF and Data Integration

Books

ID	Author	Title	Publisher	Year
ISBN 978-0140439076	ACD-01	The Sign of the Four	P-01	2001

Authors

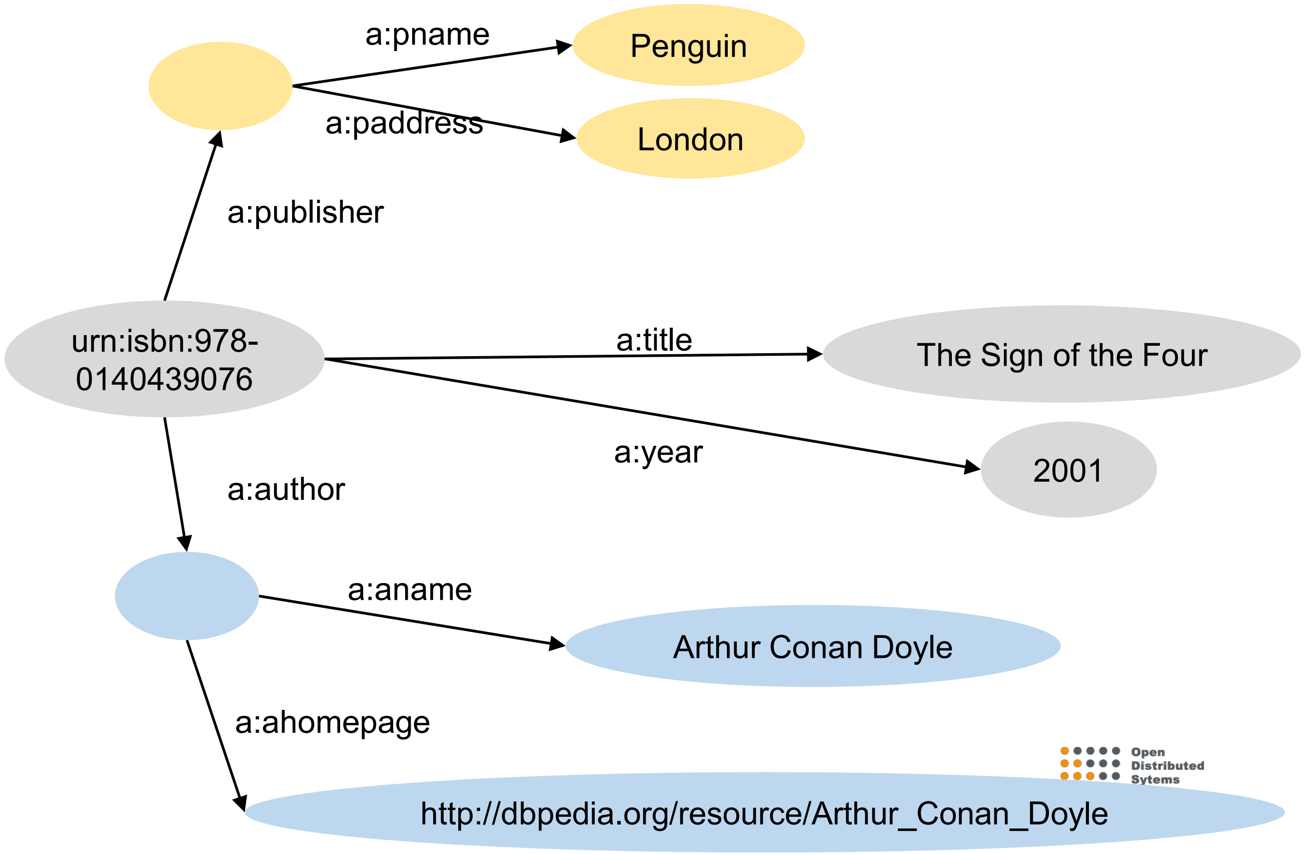
ID	Name	Homepage
ACD-01	Arthur Conan Doyle	http://dbpedia.org/resource/Arthur_Conan_Doyle

Publishers

ID	Name	Location
P-01	Penguin	London

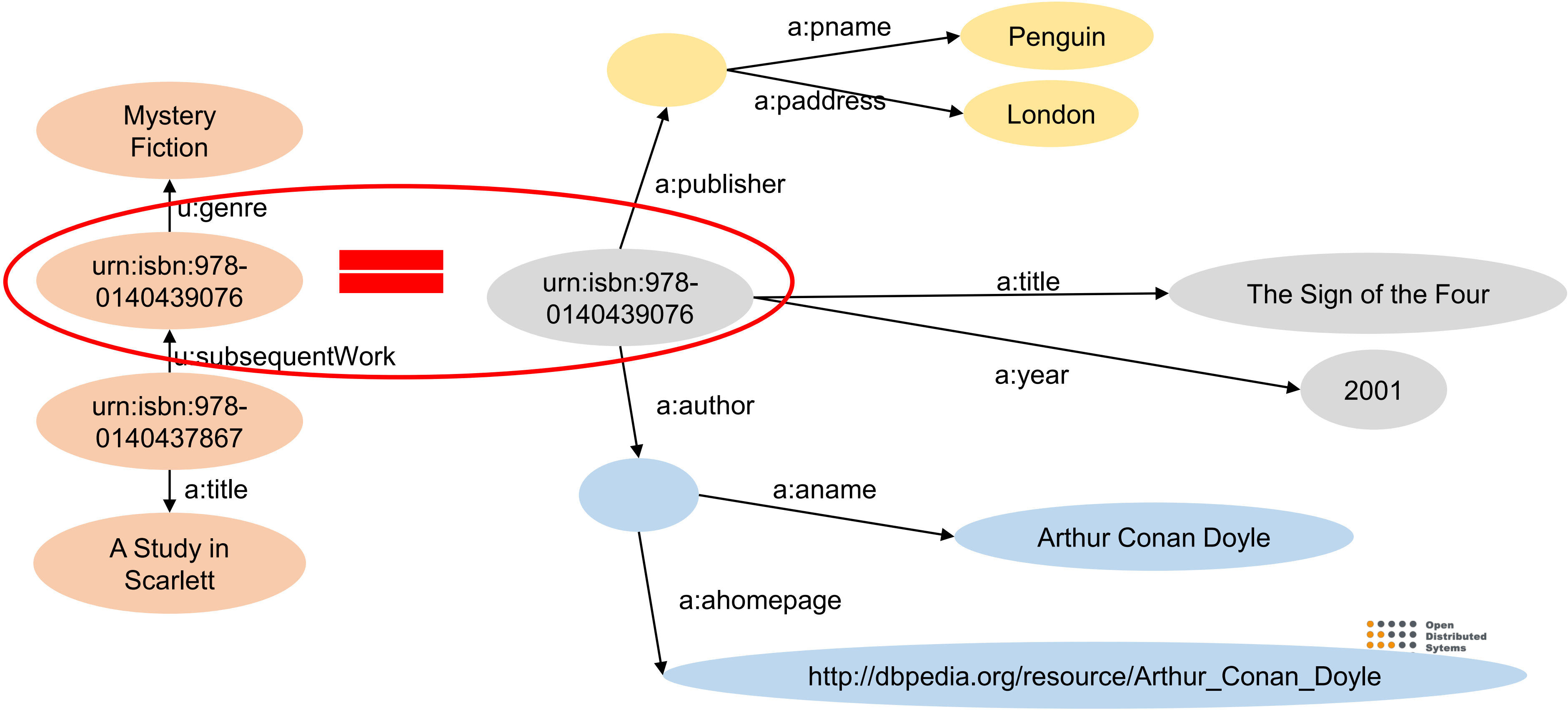
RDF and Data Integration

A simple example: Bibliographic Database



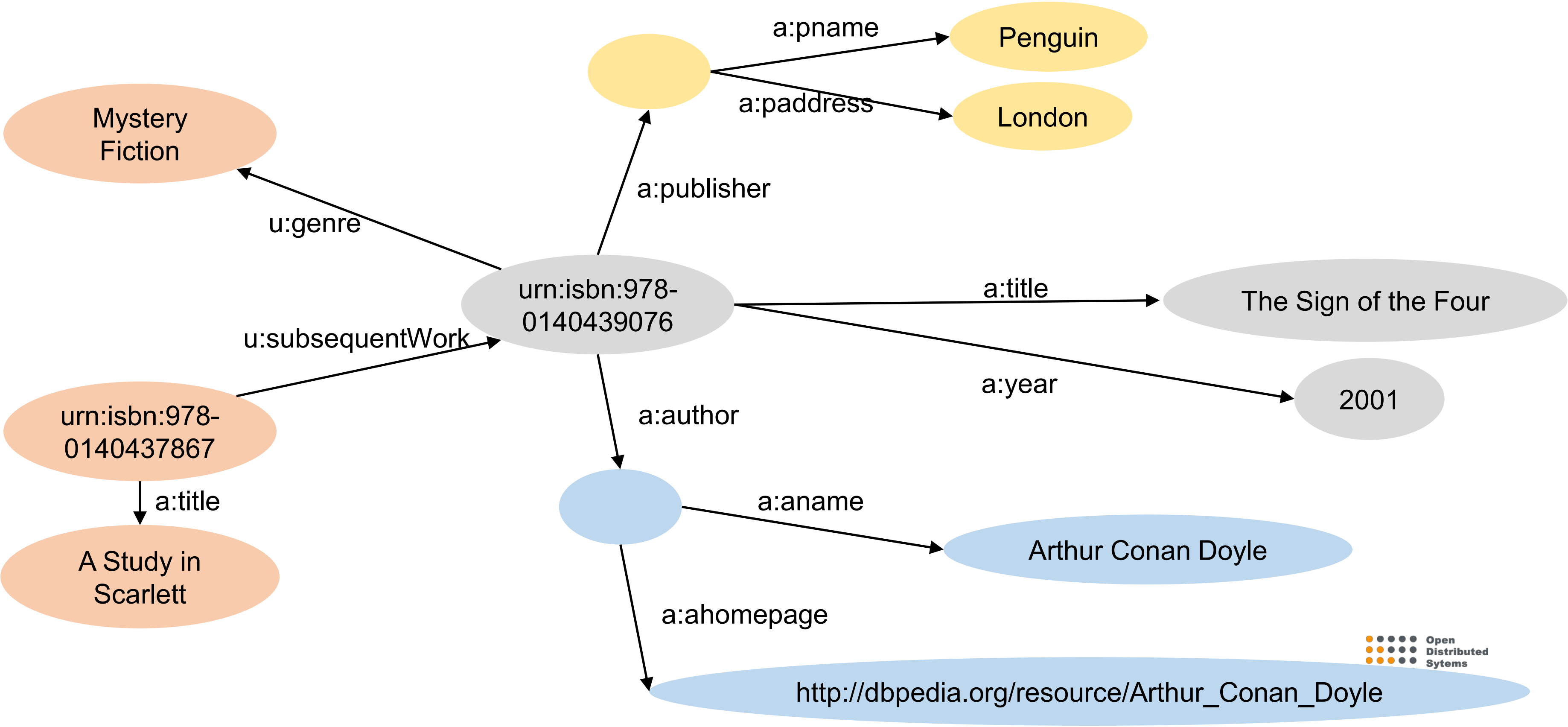
RDF and Data Integration

A simple example: Bibliographic Database



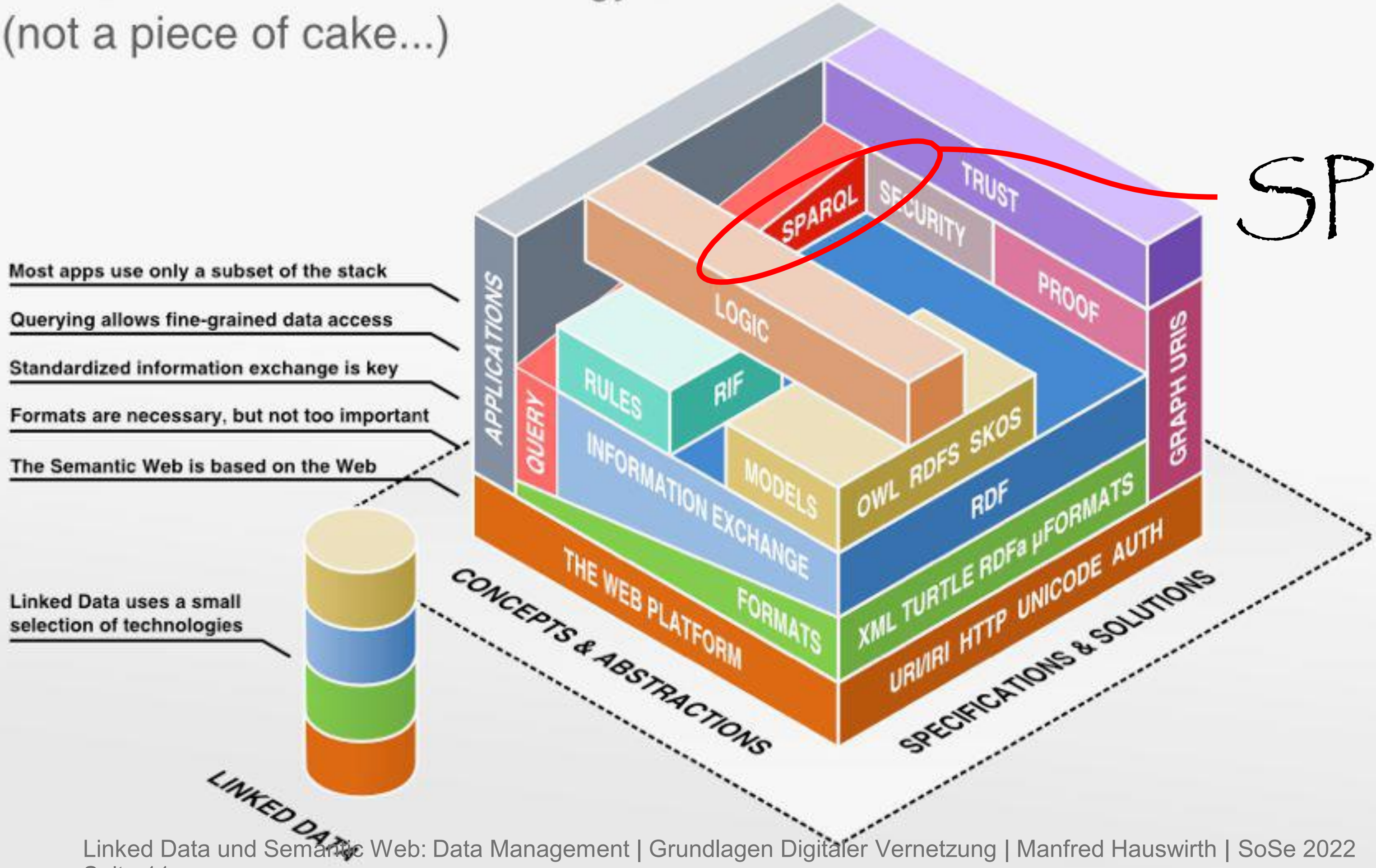
RDF and Data Integration

A simple example: Bibliographic Database



How to query RDF(S)? - SPARQL

The Semantic Web Technology Stack (not a piece of cake...)



Most apps use only a subset of the stack
Querying allows fine-grained data access
Standardized information exchange is key
Formats are necessary, but not too important
The Semantic Web is based on the Web
Linked Data uses a small selection of technologies
LINKED DATA
APPLICATIONS
QUERY
SPARQL
SECURITY
TRUST
PROOF
LOGIC
RULES
RIF
MODELS
OWL RDFS SKOS
INFORMATION EXCHANGE
RDF
FORMATS
XML TURTLE RDFa µFORMATS
THE WEB PLATFORM
URI/IRI HTTP UNICODE AUTH
GRAPH URIS
CONCEPTS & ABSTRACTIONS
SPECIFICATIONS & SOLUTIONS

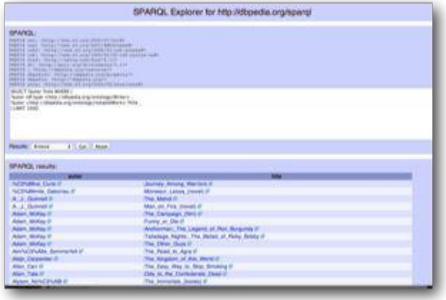
SPARQL

SPARQL - A Query Language for RDF

SPARQL
Protocol Layer
HTTP

Client

Server



SPARQL Explorer for http://dbpedia.org/sparql

SPARQL:

```
[illegible]
SELECT ?author ?title WHERE {
?author rdf:type <http://dbpedia.org/ontology/Writer> .
?author <http://dbpedia.org/ontology/notableWork> ?title .
} LIMIT 1000
```

Results: Browse | Go | Reset

SPARQL results:

author	title
%C3%88ve_Curie	Journey_Among_Warriors
%C3%89mile_Gaboriau	Monsieur_Lecoq_(novel)
A._J._Quinnell	The_Mahdi
A._J._Quinnell	Man_on_Fire_(novel)
Adam_McKay	The_Campaign_(film)
Adam_McKay	Funny_or_Die
Adam_McKay	Anchorman:_The_Legend_of_Ron_Burgundy
Adam_McKay	Talladega_Nights:_The_Ballad_of_Ricky_Bobby
Adam_McKay	The_Other_Guys
Aim%C3%A9e_Sommerfelt	The_Road_to_Agra
Alejo_Carpentier	The_Kingdom_of_this_World
Allen_Carr	The_Easy_Way_to_Stop_Smoking
Allen_Tate	Ode_to_the_Confederate_Dead
Alyson_No%C3%ABl	The_Immortals_(books)

SPARQL
Query Language

user interface

SPARQL Endpoint

SPARQL - A Query Language for RDF

SPARQL Protocol **a**nd **R**DF **Q**uery **L**anguage is

- a **Query Language** for RDF graph traversal
- a **Protocol Layer**, to use SPARQL via HTTP
- an **Output Format Specification** for SPARQL queries
- W3C Standard (SPARQL 1.1, Mar 2013)
- inspired by SQL

SPARQL - A Query Language for RDF

SPARQL Features:

- **Extraction** of Data as
  - RDF Subgraphs, URIs, Blank Nodes, typed and untyped Literals
  - with aggregate functions, subqueries, complex joins, property paths
- **Exploration** of Data via Query for unknown relations
- **Transformation** of RDF Data from one vocabulary into another
- **Construction** of new RDF Graphs based on RDF Query Graphs
- **Updates of RDF Graphs** as full data manipulation language
- **Logical Entailment** for RDF, RDFS, OWL, and RIF Core entailment.
- **Federated Queries** distributed over different SPARQL endpoints

For Queries we need Variables

SPARQL **Variables** are bound to RDF terms

e.g. **?title, ?author, ?address**

In the same way as in SQL,

- **Query for variables** is performed via **SELECT statement**

  e.g. **SELECT ?title ?author ?published**
- SELECT statement returns Query Results as a **table**

?title	?author	?published
1984	George Orwell	1948
Brave New World	Aldous Huxley	1932
Fahrenheid 451	Ray Bradbury	1953

SPARQL Result

SPARQL - Graph Pattern Matching

- SPARQL is based on **RDF Turtle serialization** and **basic graph pattern matching**.
- A **Graph Pattern** (**Triple Pattern**) is a RDF Triple that can contain variables at any arbitrary place (Subject, Property, Object).

(Graph) Triple Pattern = Turtle + Variables

- Example:

Look for countries and their capitals:

```
?country dbo:capital ?capital .
```

- A **Basic Graph Pattern** (**BGP**) is a set of Triple Pattern

SPARQL - Graph Pattern Matching

Triple Pattern

```
?country dbo:capital ?capital .
```

RDF Graph

```
dbpedia:Venezuela rdf:type dbo:Country .
dbpedia:Venezuela dbo:capital dbpedia:Caracas .
dbpedia:Venezuela dbprop:language “Spanish“ .
dbpedia:Germany rdf:type dbo:Country .
dbpedia:Germany dbo:capital “Berlin“ .
dbpedia:Germany dbp:language “German“ .
...
```

SPARQL - Complex Query Patterns

- SPARQL Graph Pattern can be combined to form **complex (conjunctive) queries** for

RDF graph traversal

- *Find countries, their capitals, and their population count:*

```
        ?country dbo:capital ?capital .
        ?country dbo:population ?population .
```

- *Given a URI, find the name of a person and his spouse:*

```
dbpedia:Joe_Weider dbpedia:name ?name ;
                   dbpedia:spouse ?spouse .
?spouse dbpedia:name ?spouse_name .
```

SPARQL - General Query Format

find all writers and the titles of their notable works:

```
PREFIX :     <http://dbpedia.org/resource/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo:  <http://dbpedia.org/ontology/>

SELECT ?author_name ?title

FROM <http://dbpedia.org/>

WHERE {
     ?author rdf:type dbo:Writer .
     ?author rdfs:label ?author_name .
     ?author dbo:notableWork ?work .
     ?work rdfs:label ?title .
}
```

specifies namespaces

specifies output variables

specifies graph to be queried

specifies graph pattern to be matched

SPARQL - General Query Format

search all writers and the titles of their notable works **ordered by** authors in ascending order and **limit** the results to the first 10 results starting the list at **offset** 10 position:

```
PREFIX :     <http://dbpedia.org/resource/>
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo:  <http://dbpedia.org/ontology/>

SELECT ?author_name ?title

FROM <http://dbpedia.org/>

WHERE {
      ?author rdf:type dbo:Writer .
      ?author rdfs:label ?author_name .
      ?author dbo:notableWork ?work .
      ?work rdfs:label ?title .
}
ORDER BY ASC (?author_name)
LIMIT 10
OFFSET 10
```

solution sequence modifiers

query SPARQL endpoint

SPARQL - Filter Constraints

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author rdfs:label ?author_name .
       ?author dbo:notableWork ?work .
       ?work dbo:numberOfPages ?pages .
       FILTER (?pages > 500) .
       ?work rdfs:label ?title .
} LIMIT 100
```

specifies constraints for the result

- `FILTER` expressions contain operators and functions
- `FILTER` can NOT assign/create new values

query SPARQL endpoint

SPARQL - Filter Constraints

Example: Filter results only for English labels

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title ?pages
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author rdfs:label ?author_name .
       FILTER (LANG(?author_name)="en").
       ?author dbo:notableWork ?work .
       ?work dbo:numberOfPages ?pages
       FILTER (?pages > 500) .
       ?work rdfs:label ?title .
       FILTER (LANG(?title)="en").
} LIMIT 100
```

query SPARQL endpoint

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

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:  <http://dbpedia.org/ontology/>

ASK
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
       }
```

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

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:  <http://dbpedia.org/ontology/>

DESCRIBE ?author ?work
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work .
} LIMIT 10
```

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

```
PREFIX rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo:  <http://dbpedia.org/ontology/>

CONSTRUCT { ?author <http://example.org/hasWritten> ?work .}
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author dbo:notableWork ?work
       .
} LIMIT 10
```

query SPARQL endpoint

More SPARQL Operators

Logical connectives && and || for xsd:boolean

Comparison operators =, !=, <, >, <=, and >= for numeric datatypes,
xsd:dateTime, xsd:string, and xsd:boolean

Comparison operators = and != for other datatypes

Arithmetic operators +, -, *, and / for numeric datatypes

and in addition:

- `REGEX(String,Pattern)` or `REGEX(String,Pattern,Flags)`
- `sameTERM(A,B)`
- `langMATCHES(A,B)`

SPARQL - Filter Constraints

Example: Book titles that contain the word “love”

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?title
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author rdfs:label ?author_name
       FILTER (LANG(?author_name)="en") .
       ?author dbo:notableWork ?work .
       ?work rdfs:label ?title .
       FILTER (LANG(?title)="en")
       FILTER REGEX (?title, "love", "i") .
} LIMIT 100
```

string

regular expression

flags

learn more about regular expressions at http://regexone.com/

query SPARQL endpoint

SPARQL - Filter Constraints

Example: Retrieve also the German book title, if available

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?author_name ?en_title ?de_title
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       ?author rdfs:label ?author_name
       FILTER (LANG(?author_name)="en").
       ?author dbo:notableWork ?work .
       ?work rdfs:label ?en_title .
       FILTER (LANG(?en_title)="en") .
       OPTIONAL {?work rdfs:label ?de_title
                 FILTER (LANG(?de_title)="de") . }
} LIMIT 100
```

optional constraint

- The keyword **OPTIONAL** selects optional elements from the RDF graph
- Complies to a Left Outer Join

query SPARQL endpoint

SPARQL - Negation

Example: Retrieve authors that don‘t have an entry for “notable work”

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
       OPTIONAL {?author dbo:notableWork ?work . }
       FILTER (!BOUND(?work))  .
} LIMIT 100
```

no variable binding

Negation in SPARQL complies to “NOT EXISTS” in SQL

query SPARQL endpoint

SPARQL - Negation (2)

Example: Retrieve authors that don‘t have an entry for “notable work”

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {

    ?author rdf:type dbo:Writer
    FILTER NOT EXISTS {?author dbo:notableWork ?work .}

} LIMIT 100
```

filter query for result existence

SPARQL 1.1 also provides `FILTER` expressions `EXISTS` and `NOT EXISTS`.

SPARQL - Negation (3)

Example: Retrieve authors that don‘t have an entry for “notable work”

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?author
FROM <http://dbpedia.org/>
WHERE {
    ?author rdf:type dbo:Writer
    MINUS {?author dbo:notableWork ?work .}
} LIMIT 100
```

remove from query results

Filtering of the query results by removing possible results with `MINUS`.

Differences to `NOT EXISTS`:

- `MINUS` changes the graph pattern
- query results are dependent of position of `MINUS`

SPARQL - Aggregate Functions

Example: How many authors are there in DBpedia?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT COUNT(?author) AS ?num
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer .
}
```

aggregate function

query SPARQL endpoint

SPARQL - Aggregate Functions

Example: How many distinct authors are there in DBpedia who have entries for notable works?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT COUNT(DISTINCT(?author)) AS ?num

FROM <http://dbpedia.org/>
WHERE {
    ?author rdf:type dbo:Writer ;
          dbo:notableWork ?work .
}
```

aggregate function

query SPARQL endpoint

SPARQL - Aggregate Functions

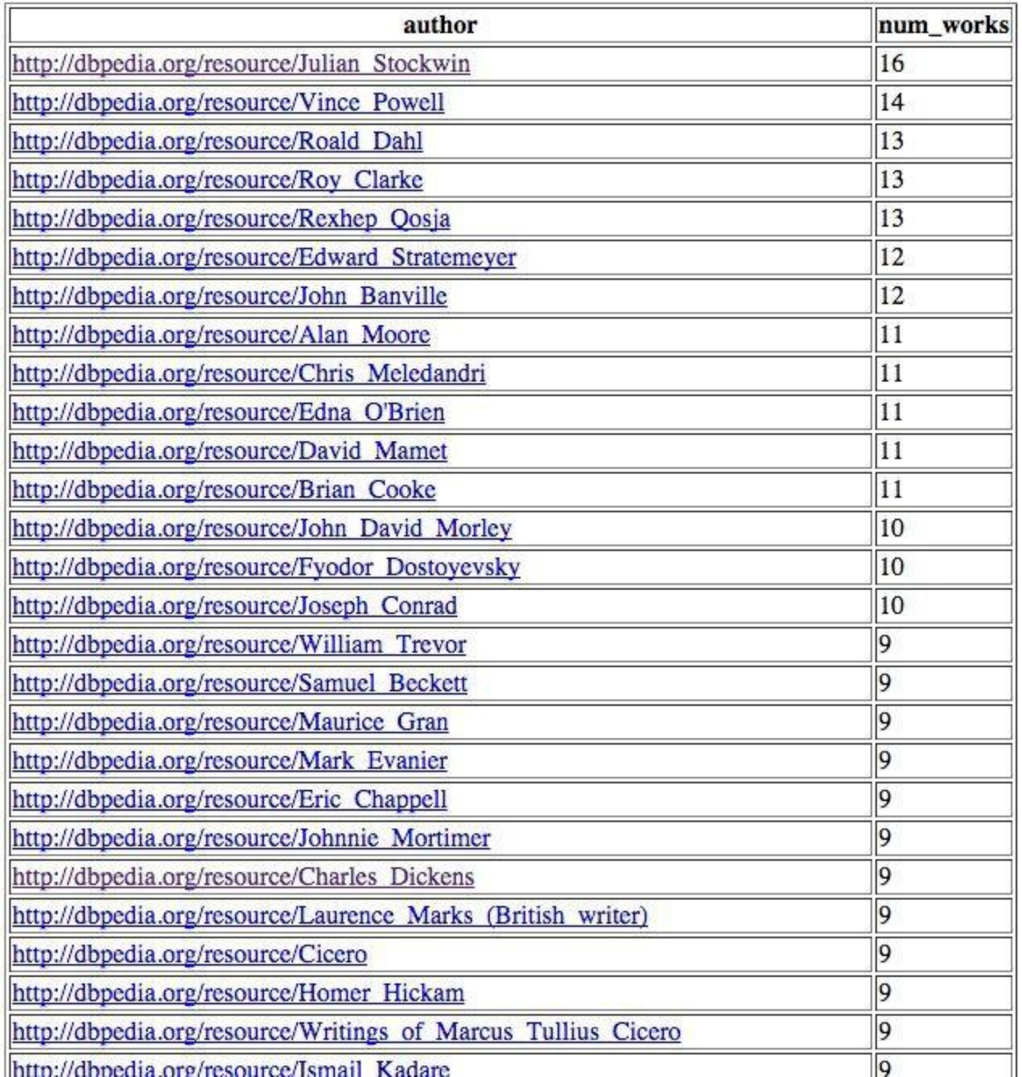
Example: Which author wrote how many notable works?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?author COUNT(?work) AS ?num_works
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer
       ;       dbo:notableWork ?work .
} GROUP BY ?author
ORDER BY DESC (?num_works)
```

aggregate function

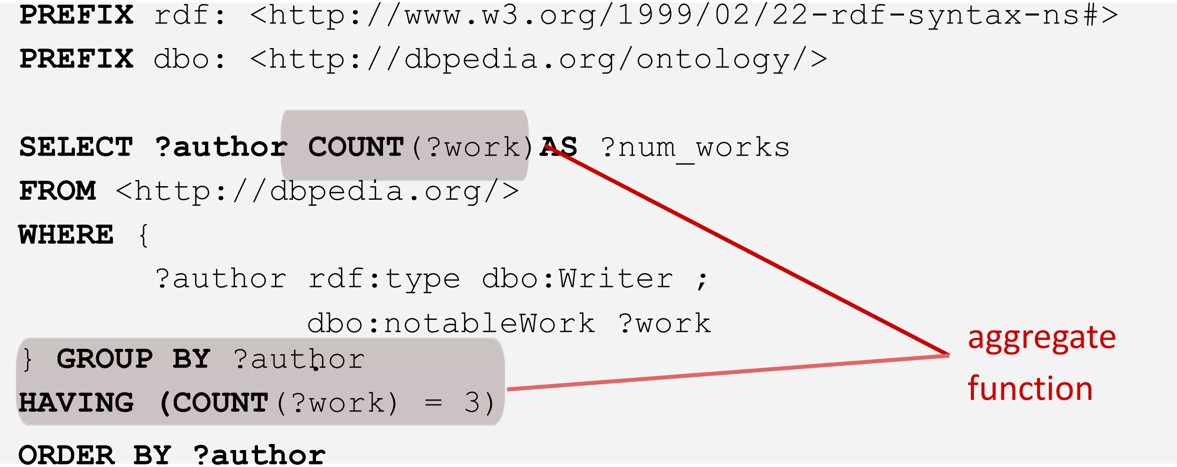
query SPARQL endpoint



author	num_works
http://dbpedia.org/resource/Julian_Stockwin	16
http://dbpedia.org/resource/Vince_Powell	14
http://dbpedia.org/resource/Roald_Dahl	13
http://dbpedia.org/resource/Roy_Clarke	13
http://dbpedia.org/resource/Rexhep_Qosja	13
http://dbpedia.org/resource/Edward_Stratemeyer	12
http://dbpedia.org/resource/John_Banville	12
http://dbpedia.org/resource/Alan_Moore	11
http://dbpedia.org/resource/Chris_Meledandri	11
http://dbpedia.org/resource/Edna_O'Brien	11
http://dbpedia.org/resource/David_Mamet	11
http://dbpedia.org/resource/Brian_Cooke	11
http://dbpedia.org/resource/John_David_Morley	10
http://dbpedia.org/resource/Fyodor_Dostoyevsky	10
http://dbpedia.org/resource/Joseph_Conrad	10
http://dbpedia.org/resource/William_Trevor	9
http://dbpedia.org/resource/Samuel_Beckett	9
http://dbpedia.org/resource/Maurice_Gran	9
http://dbpedia.org/resource/Mark_Evanier	9
http://dbpedia.org/resource/Eric_Chappell	9
http://dbpedia.org/resource/Johnnie_Mortimer	9
http://dbpedia.org/resource/Charles_Dickens	9
http://dbpedia.org/resource/Laurence_Marks_(British_writer)	9
http://dbpedia.org/resource/Cicero	9
http://dbpedia.org/resource/Homer_Hickam	9
http://dbpedia.org/resource/Writings_of_Marcus_Tullius_Cicero	9
http://dbpedia.org/resource/Ismail_Kadare	9

SPARQL - Aggregate Functions

Example: Which author wrote exactly 3 notable works (according to DBpedia)?

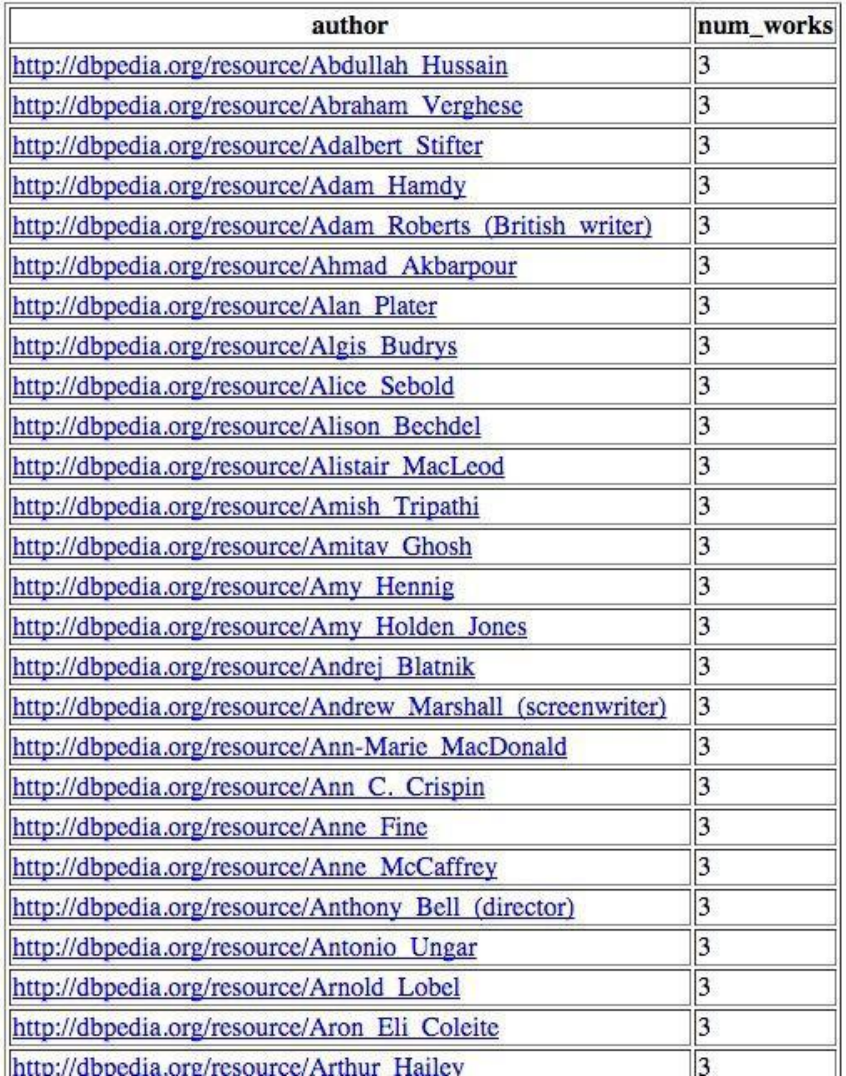


```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?author COUNT(?work)AS ?num_works
FROM <http://dbpedia.org/>
WHERE {
       ?author rdf:type dbo:Writer ;
               dbo:notableWork ?work
} GROUP BY ?author
HAVING (COUNT(?work) = 3)
ORDER BY ?author
```

aggregate function

query SPARQL endpoint



author	num_works
http://dbpedia.org/resource/Abdullah_Hussain	3
http://dbpedia.org/resource/Abraham_Verghese	3
http://dbpedia.org/resource/Adalbert_Stifter	3
http://dbpedia.org/resource/Adam_Hamdy	3
http://dbpedia.org/resource/Adam_Roberts_(British_writer)	3
http://dbpedia.org/resource/Ahmad_Akbarpour	3
http://dbpedia.org/resource/Alan_Plater	3
http://dbpedia.org/resource/Algis_Budrys	3
http://dbpedia.org/resource/Alice_Sebold	3
http://dbpedia.org/resource/Alison_Bechdel	3
http://dbpedia.org/resource/Alistair_MacLeod	3
http://dbpedia.org/resource/Amish_Tripathi	3
http://dbpedia.org/resource/Amitav_Ghosh	3
http://dbpedia.org/resource/Amy_Hennig	3
http://dbpedia.org/resource/Amy_Holden_Jones	3
http://dbpedia.org/resource/Andrej_Blatnik	3
http://dbpedia.org/resource/Andrew_Marshall_(screenwriter)	3
http://dbpedia.org/resource/Ann-Marie_MacDonald	3
http://dbpedia.org/resource/Ann_C._Crispin	3
http://dbpedia.org/resource/Anne_Fine	3
http://dbpedia.org/resource/Anne_McCaffrey	3
http://dbpedia.org/resource/Anthony_Bell_(director)	3
http://dbpedia.org/resource/Antonio_Ungar	3
http://dbpedia.org/resource/Arnold_Lobel	3
http://dbpedia.org/resource/Aron_Eli_Coleite	3
http://dbpedia.org/resource/Arthur_Hailey	3

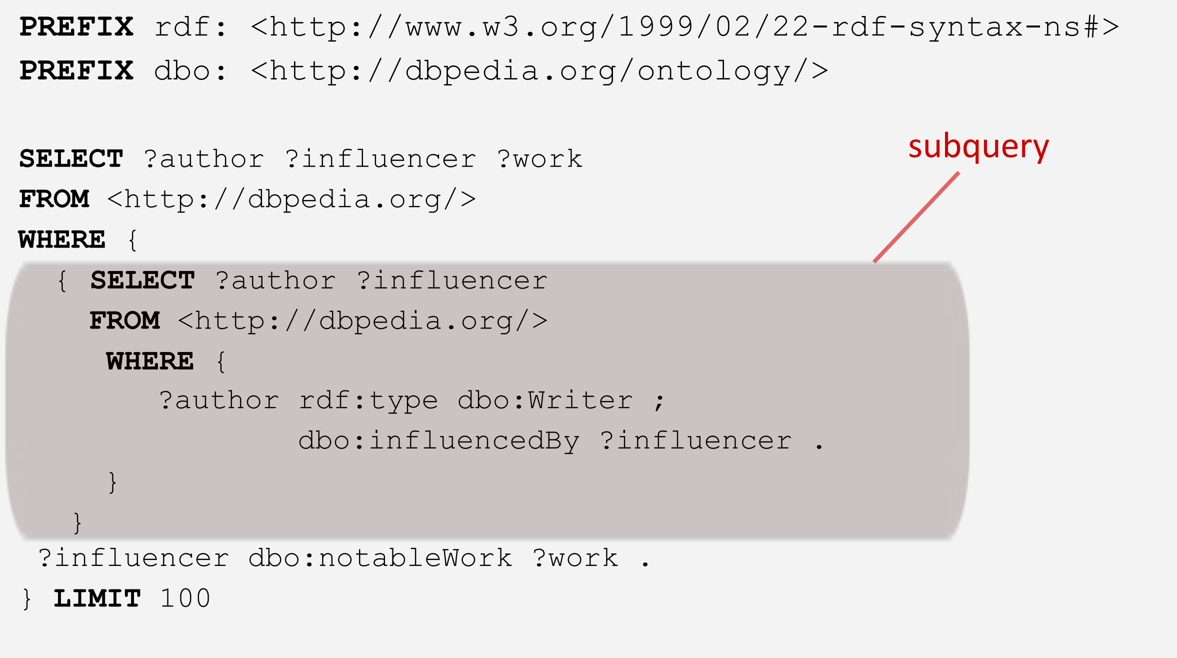
SPARQL - Aggregate Functions

SPARQL 1.1 provides more aggregate functions

- `SUM`
- `AVG`
- `MIN`
- `MAX`
- `SAMPLE` - „pick" one non-deterministically
- `GROUP_CONCAT` - concatenate values with a designated string separator

SPARQL - Subqueries

Example: Select all authors, by whom they are influenced and all the influencers’ notable works



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

SELECT ?author ?influencer ?work
FROM <http://dbpedia.org/>
WHERE {
  { SELECT ?author ?influencer
    FROM <http://dbpedia.org/>
     WHERE {
        ?author rdf:type dbo:Writer ;
                dbo:influencedBy ?influencer .
     }
  }
 ?influencer dbo:notableWork ?work .
} LIMIT 100
```

subquery

Subqueries are a way to embed SPARQL queries within other queries

result is achieved by first evaluating the inner query

query SPARQL endpoint

SPARQL - Property Paths

A **property path** is a possible route through an RDF graph between two graph nodes.

trivial case: property path of length 1, i.e. a triple pattern

alternatives: match one or both possibilities

```
{ :book1 dc:title|rdfs:label ?displayString }
```

sequence: property path of length >1

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows/foaf:knows/foaf:name ?name . }
```

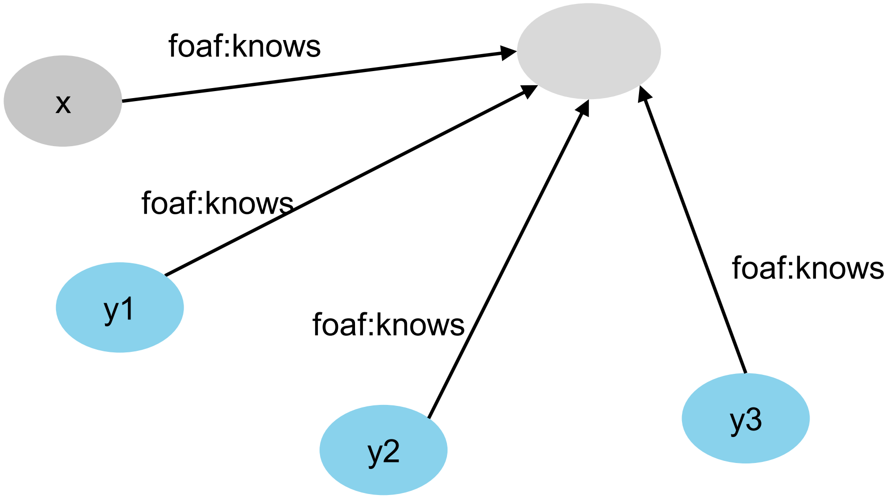
inverse property paths: reversing the direction of the triple

```
{ ?x foaf:mbox <mailto:alice@example> }
                     =
{ <mailto:alice@example> ^foaf:mbox ?x }
```

SPARQL - Property Paths

inverse path sequences

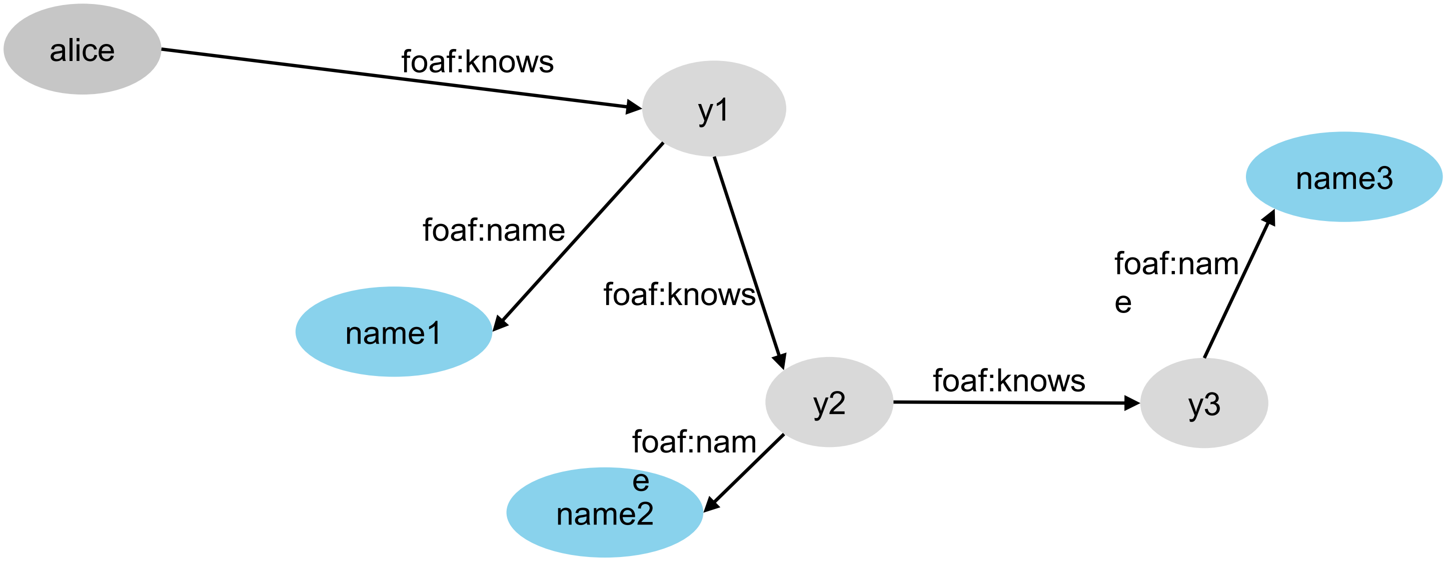
```
{ ?x foaf:knows/^foaf:knows ?y
     FILTER (?x != ?y) . }
```



SPARQL - Property Paths

arbitrary length match

```
{ ?x foaf:mbox <mailto:alice@example> .
  ?x foaf:knows+/foaf:name ?name . }
```



Wo sind wir, und wo wollen wir hin?

- ✓ Wikipedia kann als Graph betrachtet werden.
- ✓ Wikipedia kann auch als Datenbank betrachtet werden.
- ✓ Ein wichtiger Teil dieser Datenbank ist Dbpedia.
- ✓ Wofür (1)? Ein Anwendungsbeispiel: Diversität in den Medien
- ✓ Dbpedia und Wikidata (Teil 1)
- ✓ Format: RDF, RDFS und z.T. OWL; Linked Open Data
  - ✓ Das kann auch als relationale Datenbank dargestellt werden (bzw. es können Auszüge generiert werden).
- Anfragesprache: SPARQL; mehr Wikidata
- Hausarbeit: Wofür (2)? Selber recherchieren und neue Information hinzufügen.

Anmerkung: Wie DBpedia und Wikidata 2 Visionen des Semantic Web und der Linked Open Data umsetzen

- Beide werden als LOD zur Verfügung gestellt und sind somit nachnutzbar u.a. für Software.
- Aber die Konstruktionsprinzipien entsprechen eigentlich zwei Generationen von Tim Berners-Lees Idee, wie man ein maschinenlesbares Web schaffen kann:
  - Menschen/Ontology Engineers → menschenlesbare Seiten (Wikipedia) → maschinenverarbeitbare Daten (DBpedia)
    - ~ Semantic Web (2001)
  - Menschen/Ontology Engineers → maschinenverarbeitbare Daten (Wikidata) ← Existierende Daten(banken)
    - ~ Linked Open Data (2006)

Bonus / Ausblick

- “Aber wozu brauchen wir das alles, wenn wir GPT3 haben?”
- Mehr hierzu im Master-Kurs “Ethics, data, and networked AI” ☺

Wie extrahiert man soziale Graphen?

Gehen wir zurück zum Beispiel “Game of Thrones”.

Characters from Game of Thrones

```
SELECT ?character ?name_of_character
WHERE
{
   ?character wdt:P31 wd:Q20086263. # instance of, Game of Thrones character
   ?character wdt:P1559 ?name_of_character. # name in native language

   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" . }
}
```

https://w.wiki/6B3F

Fathers

```
SELECT ?character1 ?character2 ?name_of_character1 ?name_of_character2
WHERE
{
    ?character1 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
    ?character2 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
    ?character1 wdt:P1559 ?name_of_character1. # p: name in native language
    ?character2 wdt:P1559 ?name_of_character2. # p: name in native language
    ?character1 wdt:P22 ?character2. # father
    SERVICE wikibase:label {
        bd:serviceParam wikibase:language "en" .
    }
}
```

https://w.wiki/6B3M

Parents

```
SELECT ?character1 ?character2 ?name_of_character1 ?name_of_character2
WHERE
{
   ?character1 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
   ?character2 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
   ?character1 wdt:P1559 ?name_of_character1. # p: name in native language
   ?character2 wdt:P1559 ?name_of_character2. # p: name in native language
   {SELECT ?character1 ?character2
   WHERE
   {
    {?character1 wdt:P22 ?character2.} # p: father
    UNION
    {?character1 wdt:P25 ?character2.} # p: mother
   } }
   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" . }
}
   ORDER BY ?character1
```

https://w.wiki/6B3Z

Parents (2)

```
SELECT ?character1 ?character2 ?name_of_character1 ?name_of_character2
WHERE
{
  ?character1 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
  ?character2 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
  ?character1 wdt:P1559 ?name_of_character1. # p: name in native language
  ?character2 wdt:P1559 ?name_of_character2. # p: name in native language
  ?character1 ?relationship ?character2.
  FILTER (?relationship=wdt:P22 || ?relationship=wdt:P25) # father or mother
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "en" .
  }
}
  ORDER BY ?character1
```

https://w.wiki/6B3c

Various family relationships

```
SELECT ?character1Label ?character2Label ?verbindungLabel
WHERE
{
   ?character1 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
   ?character2 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
   ?character1 ?verbindungt ?character2.
   FILTER (?verbindungt=wdt:P22 || ?verbindungt=wdt:P25 ||
?verbindungt=wdt:P3373 || ?verbindungt=wdt:P451 || ?verbindungt=wdt:P1038) #
all family-type relationships
   ?verbindung wikibase:directClaim ?verbindungt.

   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" .  }
}
ORDER BY ?character1
```

https://w.wiki/6B4n

How to get from an entity to its label

```
SELECT ?character1Label ?character2Label ?verbindungLabel
WHERE
{
   ?character1 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
   ?character2 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
   ?character1 ?verbindungt ?character2.
   FILTER (?verbindungt=wdt:P22 || ?verbindungt=wdt:P25 || ?verbindungt=wdt:P3373 || ?verbindungt=wdt:P451 || ?verbindungt=wdt:P1038) # all family-type relationships
   ?verbindung wikibase:directClaim ?verbindungt.

   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" .  }
}
ORDER BY ?character1
```

For each entity variable you introduce, you get its label in WD ‘for free’ (just add “Label” to its name). PS: same with “Description”.

Set the language for labels and descriptions here.

How to get from a property to its label

```
SELECT ?character1Label ?character2Label ?verbindungLabel
WHERE
{
   ?character1 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
   ?character2 wdt:P31 wd:Q20086263. # p: instance of, o: Game of Thrones character
   ?character1 ?verbindungt ?character2.
   FILTER (?verbindungt=wdt:P22 || ?verbindungt=wdt:P25 ||
?verbindungt=wdt:P3373 || ?verbindungt=wdt:P451 || ?verbindungt=wdt:P1038) #
all family-type relationships
   ?verbindung wikibase:directClaim ?verbindungt.

   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" .  }
}
ORDER BY ?character1
```

For each property (variable), first derive its pseudo-entity. Then you can access the label of this entity.

Set the language for labels and descriptions here.

Again, you can get the description of this entity analogously

Result, slightly post-processed, for Gephi

1. Via Wikidata Query Services’s Download option, download a CSV.
2. Adjust the labels in the first row analogously to the other Game of Thrones dataset.
3. Result: see ISIS
4. Run analyses and visualise ☺

Frage

- Die Daten sind offensichtlich unterschiedlich von denen, die wir im Teil zu “Social Network Analysis” benutzt haben.
- Warum?
- Können Sie hier Biases (im Sinne des ersten Teils der Vorlesung) ausmachen? Welche?

Einen Schritt zurück: Was verstehen wir hier unter “soziale Daten”?

- In diesem Kurs betrachten wir v.a. soziale Graphen und sozial erzeugte Daten und fassen diese als “soziale Daten” zusammen.
- Warum?
- Menschen sind Teil sozialer Netzwerke.
- Soziale Graphen sind formale Modelle solcher Netzwerke.
- Soziale Graphen entstehen, grob gesprochen, in zweierlei Weise:
  - Menschen beschreiben sich oder andere mit dem Ziel, solche Daten zu erzeugen (z.B. als Wissens-Ressourcen)
  - Menschen tun Dinge mit anderen Zielen (z.B. mit ihren Freunden kommunizieren), dabei fallen Daten incl. soziale Graphen an (“exhaust data”)
  - Evt. gibt es Zwischenschritte (z.B. Textverarbeitung)
- In allen Fällen können die Werkzeuge und Umgebungen, die dabei benutzt werden (Wikipedia-Editor und -Policies, Instagram-Interface, …), einen Einfluss auf die erzeugten Daten haben.
- Nicht alle kollaborativ erzeugten Daten sind soziale Graphen (z.B. Wikidata über chemische Elemente), aber auch diese Daten sind sozial erzeugte Daten.

Wo sind wir, und wo wollen wir hin?

- ✓ Wikipedia kann als Graph betrachtet werden.
- ✓ Wikipedia kann auch als Datenbank betrachtet werden.
- ✓ Ein wichtiger Teil dieser Datenbank ist Dbpedia.
- ✓ Wofür (1)? Ein Anwendungsbeispiel: Diversität in den Medien
- ✓ Dbpedia und Wikidata (Teil 1)
- ✓ Format: RDF, RDFS und z.T. OWL; Linked Open Data
  - ✓ Das kann auch als relationale Datenbank dargestellt werden (bzw. es können Auszüge generiert werden).
- Anfragesprache: SPARQL; mehr Wikidata
- Hausarbeit: Wofür (2)? Selber recherchieren und neue Information hinzufügen.

SPARQL: Zum Üben (Hausarbeit)

1. All Actors from Game of Thrones
2. Nationality distribution of GoT Actors
3. Actors whose birthdays are today
4. Number of actors per 1 million inhabitant of countries in the EU
5. Most successful actors from every country in the world
   - (Hint: look at *award* s)

1. Queries, die sich aus Ihren Überlegungen vom 14.12. zu “wie stelle ich eine möglichst diverse Gruppe von Schauspielern zusammen” ergeben.

Wo sind wir, und wo wollen wir hin?

- ✓ Wikipedia kann als Graph betrachtet werden.
- ✓ Wikipedia kann auch als Datenbank betrachtet werden.
- ✓ Ein wichtiger Teil dieser Datenbank ist Dbpedia.
- ✓ Wofür (1)? Ein Anwendungsbeispiel: Diversität in den Medien
- ✓ Dbpedia und Wikidata (Teil 1)
- ✓ Format: RDF, RDFS und z.T. OWL; Linked Open Data
  - ✓ Das kann auch als relationale Datenbank dargestellt werden (bzw. es können Auszüge generiert werden).
- Anfragesprache: SPARQL; mehr Wikidata
- Bonus: Wofür (2)? Selber recherchieren und neue Information hinzufügen.