Introduction to Semantic Web Databases

Prepared By:

Amgad Madkour
Ph.D. Candidate
Purdue University

http://amgadmadkour.github.io

Last Updated: November 19, 2018

Semantic Web – Motivation

- Represents the next generation of the the world wide web (**Web 3.0**)
- Aims at converting the current web into a **web of data**
- Intended for realizing the **machine-understandable** web
- Allows **combining data** from several applications to arrive at **new information**

What is the Semantic Web ?

- A set of **standards**
- Defines **best practices for sharing data** over the web for use by applications
- Allows defining the **semantics** of data
    - Example:
        - Spouse is a symmetric relations (if A spouse of B then B spouse of A)
        - zip codes are a subset of postal codes
        - “sell” is the opposite of “buy”

Semantic Web – Standardization

- The World Wide Web Consortium (W3C) developed a number of **standards** around the Semantic Web:

1. Data Model (RDF)
2. Query languages (SPARQL)
3. Ontology languages (RDF Schema and OWL variants)

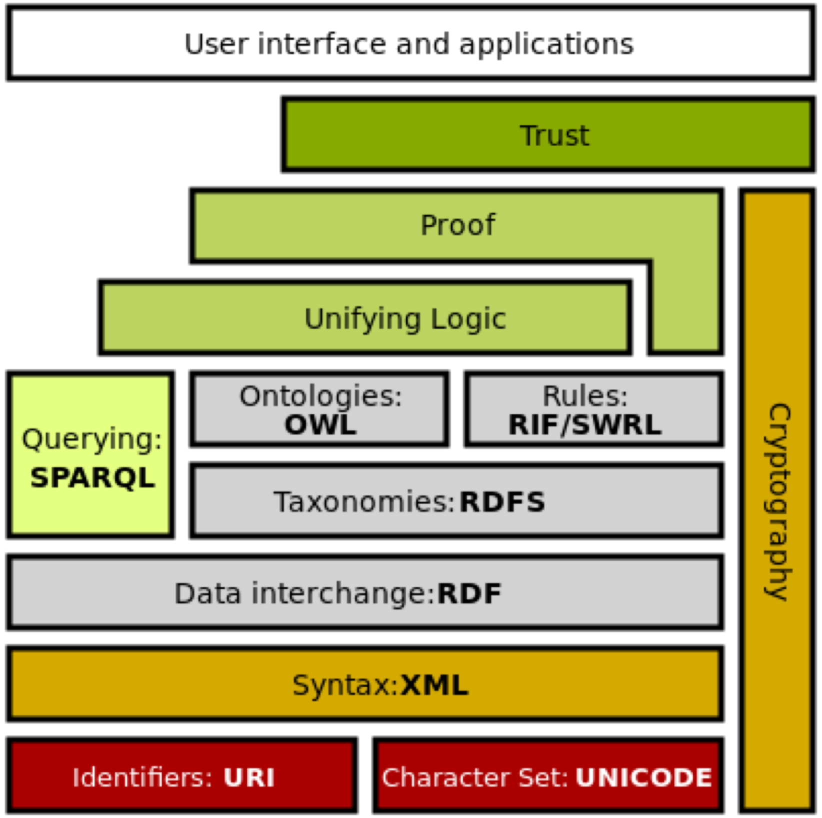
Semantic Web – Use Cases

- Many Semantic Web components (e.g. RDF and SPARQL) are used in various domains:
  - Semantic Search (Google, Microsoft, Amazon)
  - Smart Governments (data.gov.us, data.gov.uk)
  - Pharmaceutical Companies (AstraZeneca)
  - Automation (Siemens)
  - Mass Media (Thomson Reuters)

Semantic Web – Technology Stack

- **Hypertext Web Technologies**
  - **IRI**: Generalization of URI
  - **Unicode**: Language support
  - **XML**: Create documents of structured data

- **Standardized Semantic Web Technologies**
  - **RDF**: Creating statements (triples)
  - **RDFS**: RDF Schema of classes and properties
  - **OWL**: Extends RDFS by adding constructs
  - **SPARQL**: Query RDF-based data
  - **RIF**: Rule interchange format, goes beyond OWL



Resource Description Framework (RDF)

- Is the **standard** for **representing knowledge**
- RDF expresses information as a list of **statements** known as **triples**
- A **triple** consists of:

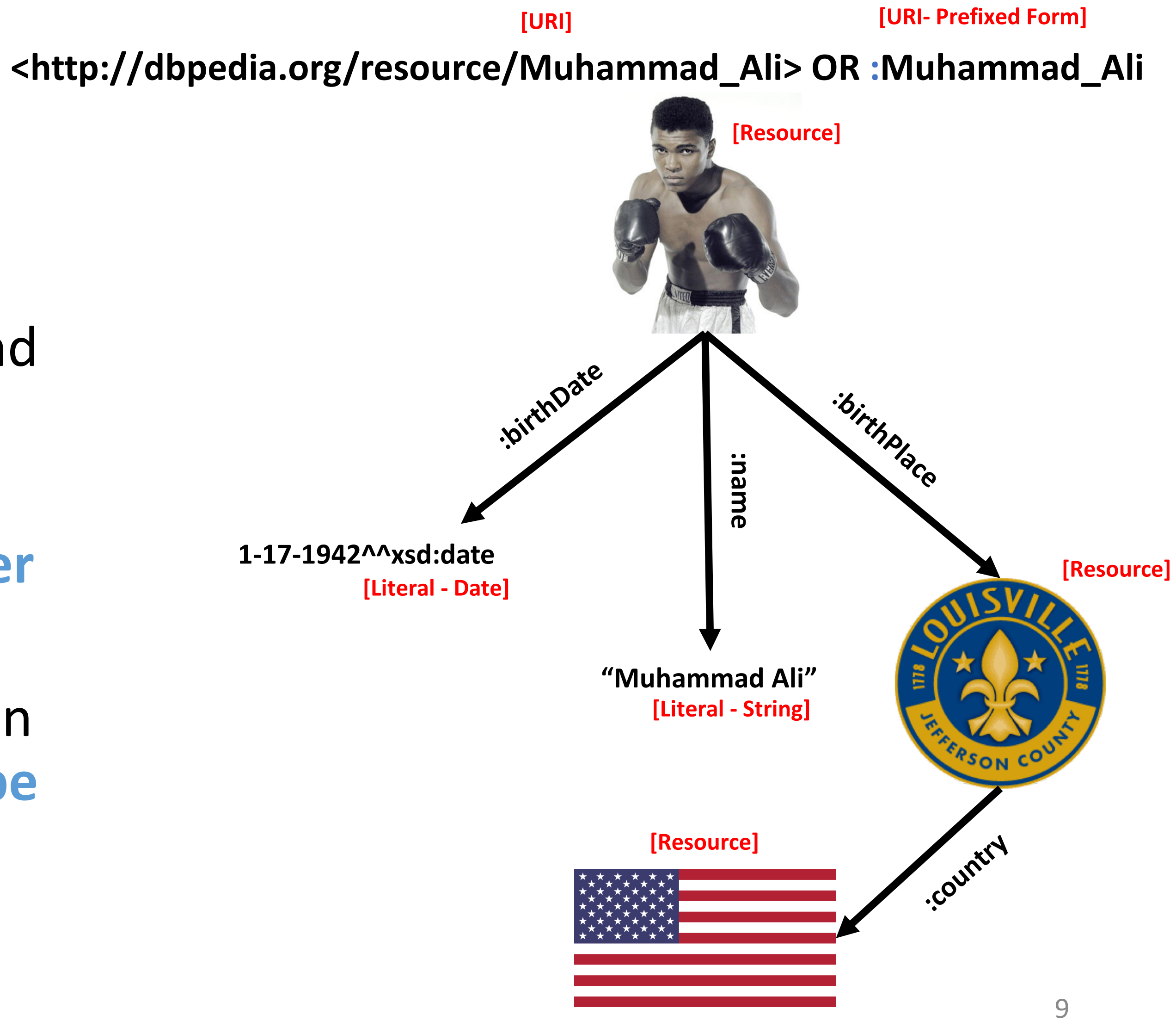
  SUBJECT, **PREDICATE**, and an **OBJECT**

  - **Example**: (“Muhammad Ali”, “isA”, “Boxer”)

RDF Model

Triple Structure

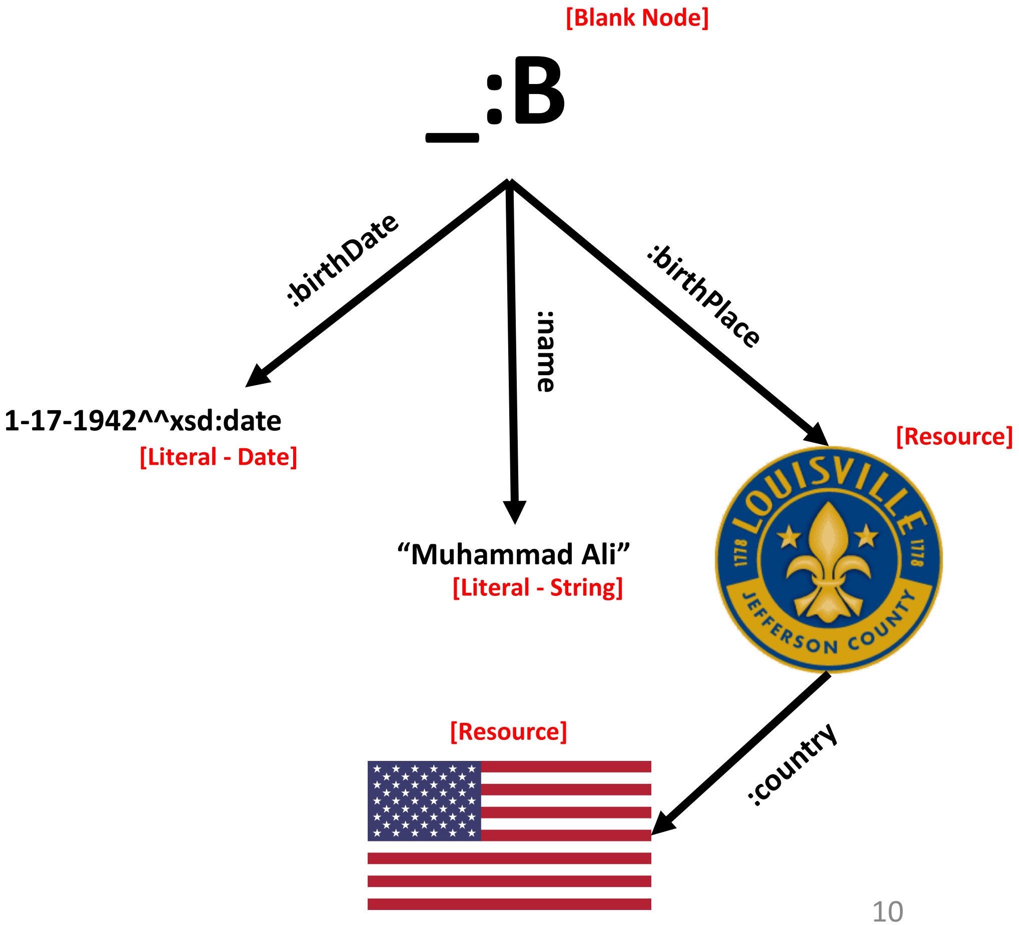
- Subjects, predicates, and objects are represented by **resources** or **literals**
- A resource is represented by a **URI** and denotes a **named thing**
- Literals represent a **string** or a **number**
- Literals representing values other than strings may have an attached **datatype**



RDF Model

Anonymous Resources

- RDF allows one special case of resources where the **URI is not known**
- An **anonymous resource** is represented as having a blank identity or a **blank node/bnode**
- A blank node can only be used as **subject** or **object** of a triple



RDF Model

Namespaces

- URI’s allow defining **distinct** identities to RDF **resources**

- Each RDF dataset provider can define common RDF resources using its own **namespace**
  - **Example**:
    - http://dbpedia.org/resource/Muhammad_Ali
    - http://www.wikipedia.org/Muhammad_Ali

- **URI’s** representing the namespace can be replaced with a **prefix**
  - **Example**:
    - dbp:Muhammad_Ali
    - wiki:Muhammad_Ali

- The namespace can be defined in an RDF document using **@prefix**
  - **Example**:
    - @prefix dbp: http://dbpedia.org/resource/
    - @prefix wiki: http://www.wikipedia.org/

RDF Model

Storing RDF Files

- RDF can be serialized using
  - N-Triple
  - Notation 3/Turtle
  - RDF/XML

- The standardized formats by W3C are **RDF/XML** and **Turtle**

- **Notation 3** is similar to Turtle but includes more enhanced features

- Notation 3 is being developed by **Tim Berners-Lee**

RDF Model

Storing RDF Files - N-Triple Format

<http://dbpedia.org/resource/**Muhammed_Ali**> <http://dbpedia.org/ontology/**birthPlace**> <http://dbpedia.org/resource/**Louisville,_Kentucky**> .
<http://dbpedia.org/resource/**Muhammed_Ali**> <http://dbpedia.org/ontology/**birthDate**> "**1942-01-17**"^^xsd:date .
<http://dbpedia.org/resource/**Muhammed_Ali**> <http://xmlns.com/foaf/0.1/**name**> "**Muhammad Ali**"@en .

Subjects **Predicates** **Objects**

RDF Model

Storing RDF Files - Notation 3/Turtle Format

```
@prefix dbp: <http://dbpedia.org/resource>  .
@prefix dbo: <http://dbpedia.org/ontology>  .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

dbp:Muhammed_Ali
 dbo:birthPlacedbp:Louisville,_Kentucky ;
 dbo:birthDate  "1942-01-17"^^xsd:date ;
 foaf:name "Muhammad Ali"@en .
```

Representing multiple predicate, object per subject

```
@prefix dbp: <http://dbpedia.org/resource>  .
@prefix dbo: <http://dbpedia.org/ontology>  .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://...w3.org/...22-rdf-syntax-ns#>

dbp:Muhammed_Ali  rdf:type
                          foaf:Person ,
                          dbo:Boxer ,
                          dbo:Agent .
```

Representing multiple objects per predicate of a subject

RDF Model

Data Typing

- Non-URI values are called **literals**
- Literals have a **datatype** assigned to them

```
@prefix dbp: <http://dbpedia.org/resource>  .
@prefix dbo: <http://dbpedia.org/ontology>  .
@prefix dbpr: <http://dbpedia.org/property/> .

dbp:Muhammed_Ali  dbo:birthDate “1942-01-17”^^xsd:date .
dbp:Muhammed_Ali  dbpr:koWins “37”^^xsd:integer .
```

RDF Model
Labeling and Tagging

- RDF Queries can be narrowed down to **literals** tagged in a **particular language**

- One of RDF best practices is to assign a **label** (i.e. rdfs:label) values to resources and **tag** them with a **language**

```
@prefix dbp: <http://dbpedia.org/resource> .
@prefix rdf: <http://...w3.org/...22-rdf-syntax-ns#> .

dbp:Muhammed_Ali  rdf:label
                          “Muhammad Ali”@en ,
                          “モハメド・ア”@ja ,
                          ”محمد علي" @ar .
```

RDF Model

Blank Nodes

- Blank nodes have **no permanent identity**
- Used to **group** together a set of **values**
- Used as a **placeholder** in case other triples need to refer to a blank node grouping

```
@prefix dbp: <http://dbpedia.org/resource>  .
@prefix ex:   <http://example.org/>

dbp:Muhammed_Ali  ex:info _:b1 .

_:b1  ex:firstName “Muhammad” ;
      ex:lastName   “Ali” .
```

RDF Model

Vocabularies

- **Vocabulary** (i.e. new URI’s) can be **created** or **resused**
- Existing vocabularies (e.g. Friend of a Friend - FOAF) are stored using, e.g., **RDF schema (RDFS)**
- The **RDF Vocabulary Description Language** (RDF Schema) allows describing vocabularies
- RDF Schema allows defining **properties** or new **classes** of resources

RDF Model

RDF Schema Properties

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

dc:creator
        rdf:type rdf:Property ;
        rdfs:comment ”Makes a URI"@en-US ;
        rdfs:label "Creator"@en-US .
```

Tip: Another way of specifying rdf:type is using “a”

dc:creator **a** rdf:Property

RDF Model

RDF Schema Class

```
@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:Athlete
        rdf:type rdfs:Class ;
        rdfs:label “Athlete” .

ex:Sport
        a rdfs:Class ;
        rdfs:label “Sport” .
```

RDF Model

RDF Schema Example

```
@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:playsSport
        rdf:type rdf:Property ;
        rdfs:domain ex:Athlete ;
        rdfs:range  ex:Sport .
```

- rdf:domain: If a property is **ex:playSport** in a triple then the ***subject*** is an **ex:Athlete**
- rdf:range: If the property is **ex:playSport** in a triple then the ***object*** is a **ex:Sport**

A query engine can retrieve all resources (e.g. Muhammad Ali) of a specific class (e.g., Athlete) even though there are **no explicit triples** indicating a resource **membership in a class**

Web Ontology Language (OWL)

- A **key technology** for defining **semantics** for RDF data
- OWL extends RDFS to define **ontologies**
- An **ontology** is a **formal definition** of set of vocabulary that define **relationships** between vocabulary **terms** and **class** members
- Ontologies are used to **describe domain knowledge** (e.g. biology) so that users are able to more formally share and understand data
- An ontology defined with OWL is a **collection of triples**

Web Ontology Language (OWL)

Example

```
@prefix ex: <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

ex:opponent
        rdf:type owl:SymmetricProperty ;
        rdfs:comment “Identify someone’s opponent” .

:Muhammad_Ali
        ex:opponent :Joe_Frazier
```

- **:Muhammad_Ali** is now known to have an opponent **:Joe_Frazier**
- No triples for **:Joe_Frazier** are required to be defined for **ex:opponent** relation

Linked Data

- RDF allows interlinking datasets either on the **data level** or the **query level**
- **On the data level**: RDF dataset creators can provide “**sameAs**” dataset that interlinks the same resources across datasets
- **On the query level:** The query engine can be used to merge results from multiple sources

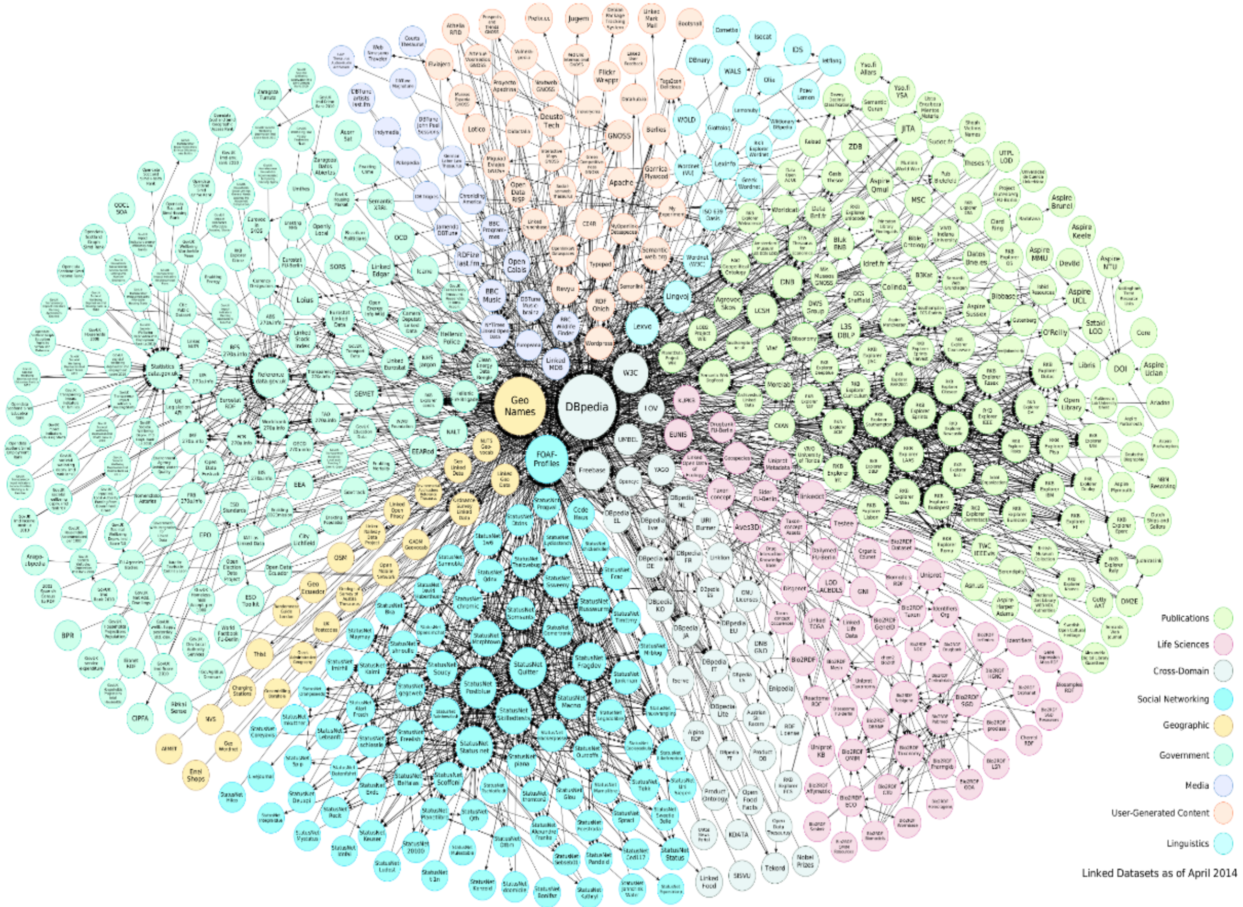


Figure: Linked RDF Data Cloud , containing thousands of datasets

Linked Data

Principles

- Use **URIs** as **names for things**
- Use **HTTP URIs** so that people can **look up** those names
- When someone looks up a URI, **provide** useful **information**, using the standards (RDF*, SPARQL)
- Include **links** to **other URIs** so that they can discover more things

By: **Tim Berners-Lee**

SPARQL Query Language

Overview

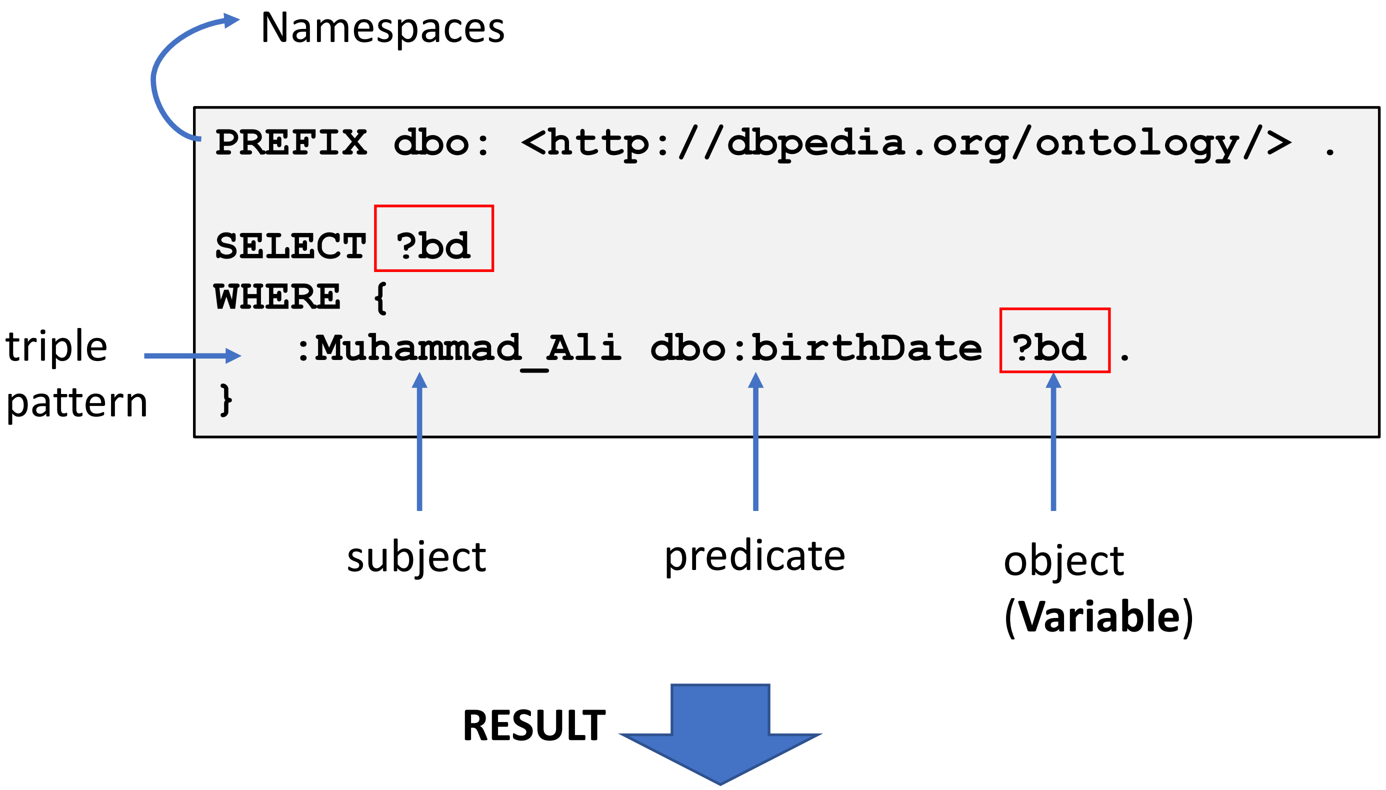
- SPARQL (pronounced "sparkle”) is an acronym for **SPARQL Protocol and RDF Query Language**
- SPARQL is an RDF/semantic query language for databases that store RDF data
- SPARQL query can consist of **triple patterns**, **conjunctions**, **disjunctions**, and **optional patterns**

SPARQL Query Language

Triple Pattern

Query: Get the birth date of Muhammad Ali

- The conditions of a SPARQL query is specified using **triple patterns**
- Triple patterns are similar to RDF triples but contain **variables**
- Variables add **flexibility** to the triple patterns matching



```
PREFIX dbo: <http://dbpedia.org/ontology/> .

SELECT ?bd
WHERE {
   :Muhammad_Ali dbo:birthDate ?bd .
}
```

bd
1942-01-17

SPARQL Query Language

Multiple Triple Patterns

Query: Get names of all Boxers

```
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 ?name
WHERE {
  ?uri rdf:type     dbo:Boxer.
  ?uri rdfs:label   ?name .
}
```

Two triple patterns
joined by ?uri variable

Results include labels in multiple languages as they all match the query triple patterns

name
"Muhammad Ali"@en
”محمد علي" @ar
"モハメド・アリ"@ja
"Mike Tyson"@en
”مايك تايسون" @ar
…

RESULT

SPARQL Query Language

FILTER

<u>Query:</u> Get names of all Boxers in English

```
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 ?name
WHERE {
  ?uri rdf:type    dbo:Boxer.
  ?uri rdfs:label  ?name .
  FILTER ( lang(?name) = ‘en’)
}
```

Results are filtered based on the language tag assigned to the label

name
"Muhammad Ali"@en
"Mike Tyson"@en
…

RESULT

SPARQL Query Language

OPTIONAL

Query: Get names of all Boxers and show nicknames if exists

```
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/> .
PREFIX foaf: <http://xmlns.com/foaf/0.1/> .

SELECT ?resource ?label ?nickname
WHERE {
  ?resource rdf:type dbo:Boxer .
  ?resource rdfs:label ?lbl .
  OPTIONAL { ?resource foaf:nick ?nickname . }
  FILTER(lang(?lbl) = 'en')
}
```

lbl	nickname
"Lennox Lewis"@en	"The Lion"@en
"Mike Tyson"@en	"Iron"@en
"Mike Tyson"@en	"Kid Dynamite"@en
"Barbados Joe Walcott"@en	"Barbados Demon"@en
"Chris Arreola"@en	"The Nightmare"@en
"Giulian Ilie"@en	"The Dentist"@en
…	…

RESULT

Note: The order of the OPTIONAL graph patterns matters in case multiple OPTIONAL patterns exist

SPARQL Query Language

MINUS

Query: Get names of all Boxers that do not have a nickname

```
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/> .
PREFIX foaf: <http://xmlns.com/foaf/0.1/> .

SELECT ?label
WHERE {
  ?resource rdf:type dbo:Boxer .
  ?resource rdfs:label ?lbl .
  MINUS { ?resource foaf:nick ?nickname . }
}
```

lbl
"Franciszek Szymura"@en
"Victor McLaglen"@en
"Anders Petersen (boxer)"@en
"Dick Turpin (boxer)"@en
"Edward Flynn (boxer)"@en
"Frederick Wedge"@en
…

RESULT

SPARQL Query Language

Property Paths – Alternative Paths (|)

<u>Query:</u> Get name or titles of Muhammad Ali

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
PREFIX dbp: <http://dbpedia.org/property/> .
PREFIX : <http://dbpedia.org/resource/> .

SELECT ?var
WHERE {
   :Muhammad_Ali (dbp:title | rdfs:label) ?var .
}
```

title
"Muhammad Ali"@en
"WBA heavyweight champion"^^rdf:langString
"WBC heavyweight champion"^^rdf:langString
"Lineal heavyweight champion"^^rdf:langString
"NABF heavyweight champion"^^rdf:langString
"The Ring heavyweight champion"^^rdf:langString
"Undisputed heavyweight champion"^^rdf:langString

RESULT

SPARQL Query Language

Property Paths – Using Regular Expression

Query: Get all heavy weight champions ***before*** Muhammad Ali

```
PREFIX dbp: <http://dbpedia.org/property/> .

SELECT ?champions
WHERE {
   ?champions dbp:before+ :Muhammad_Ali .
}
```

champions
:John_Tate_(boxer)
:Leon_Spinks
:Jimmy_Ellis_(boxer)

RESULT

Recursively get all Boxing Heavy-weight Champions ***before*** Muhammad Ali

+ → One or more

* → Zero or more

SPARQL Query Language

Property Paths – Using Defined Paths

<u>Query:</u> Get all heavy weight champions before Muhammad Ali that are ***two links away***

```
PREFIX dbp: <http://dbpedia.org/property/>

SELECT ?s
WHERE {
   :Muhammad_Ali dbp:before/dbp:before ?s .
}
```

champions
:Floyd_Patterson

RESULT

SPARQL Query Language

Property Paths – Regular Expression

Query: Get all heavy weight champions ***before*** Muhammad Ali

```
PREFIX dbp: <http://dbpedia.org/property/> .

SELECT ?champions
WHERE {
   ?champions dbp:before+ :Muhammad_Ali .
}
```

champions
:John_Tate_(boxer)
:Leon_Spinks
:Jimmy_Ellis_(boxer)

RESULT

Recursively (+) get all Boxing Heavy-weight Champions
before Muhammad Ali

SPARQL Query Language

Property Paths – Negation

Query: Get all heavy weight champions that Muhammad Ali is ***not before*** them

```
PREFIX dbp: <http://dbpedia.org/property/> .

SELECT ?champions
WHERE {
   :Muhammad_Ali ^dbp:before ?champions .
}
```

champions
:John_Tate_(boxer)
:Leon_Spinks
:Jimmy_Ellis_(boxer)

RESULT

Switching the subject & object and **negating** the predicate achieves the same result as previous query

SPARQL Query Language

DISTINCT - Eliminating Redundant Output

Query: Get all **unique** predicates/relations for the Muhammed Ali

```
PREFIX : <http://dbpedia.org/resource/> .

SELECT DISTINCT ?predicate
WHERE {
   :Muhammad_Ali ?predicate ?o .
}
```

predicate
rdf:type
rdfs:label
rdfs:comment
rdfs:seeAlso
…

RESULT

SPARQL Query Language

UNION

<u>Query:</u> Get the champion before and after Muhammed Ali

```
PREFIX dbp: <http://dbpedia.org/property/> .
PREFIX : <http://dbpedia.org/resource/> .

SELECT ?champion
WHERE {
   {?champion dbp:before :Muhammad_Ali .}
   UNION
   {?champion dbp:after  :Muhammad_Ali .}
}
```

champion
:John_Tate_(boxer)
:Leon_Spinks
:Jimmy_Ellis_(boxer)
:Ernie_Terrell
:Joe_Frazier
:Sonny_Liston
:Antonio_Rebollo

RESULT

SPARQL Query Language

FILTER on Condition - regexp

<u>Query:</u> Get matches of Muhammed Ali that contain the word “Undisputed”

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX : <http://dbpedia.org/resource/>

SELECT ?title
WHERE {
   :Muhammad_Ali dbp:title ?title .
   FILTER(regex(?title, ‘Undisputed’ , ‘i’))
}
```

title
"Undisputed heavyweight champion"^^rdf:langString
:List_of_undisputed_boxing_champions

RESULT

Filter the results by the word ‘Undisputed’ in a case insensitive fashion (‘i’)

SPARQL Query Language

FILTER on Condition - isURI

Query: Get matches of Muhammed Ali that contain the word “Undisputed” and is **not a URI**

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX : <http://dbpedia.org/resource/>

SELECT ?title
WHERE {
   :Muhammad_Ali dbp:title ?title .
   FILTER(regex(?title, ‘Undisputed’ , ‘i’))
   FILTER(!(isURI(?title)))
}
```

title
"Undisputed heavyweight champion"^^rdf:langString

RESULT

SPARQL Query Language

LIMIT and OFFSET

Query: Get two titles after the second returned title of Muhammed Ali

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX : <http://dbpedia.org/resource/>

SELECT ?title
WHERE {
   :Muhammad_Ali dbp:title ?title .
    FILTER(!(isURI(?title)))
}
OFFSET 1
LIMIT 2
```

title
"WBC heavyweight champion"^^rdf:langString
"Lineal heavyweight champion"^^rdf:langString

RESULT

Skip the first result and limit to 2 following result

SPARQL Query Language

ORDER BY – Sorting Results

Query: Get all sorted titles of Muhammed Ali

```
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX : <http://dbpedia.org/resource/>

SELECT ?title
WHERE {
   :Muhammad_Ali dbp:title ?title .
}
ORDER BY(?title)
```

ORDER BY DESC(?title) can also be used to sort results in a descending order

title
"Lineal heavyweight champion"^^rdf:langString
"NABF heavyweight champion"^^rdf:langString
"The Ring heavyweight champion"^^rdf:langString
"Undisputed heavyweight champion"^^rdf:langString
"WBA heavyweight champion"^^rdf:langString
"WBC heavyweight champion"^^rdf:langString

RESULT

Semantic Web: Case Study

- Solid (**So**cial **Li**nked **D**ata) is a web **decentralization** project led by Tim Berners-Lee
- The objective of Solid is to create **true data ownership** and **improved privacy**
- **Applications** and **data** are **separate**, allowing users to store personal data where they want
- A user stores personal data in "**pods**" (personal online data stores)
- Applications are **authenticated** by Solid and are given access to pods based on the application permission

References

- **Learning SPARQL** , Second Edition
- **RDF Basic Concepts**
  - https://jena.apache.org/documentation/rdf/index.html
- **RDF Tutorial**
  - https://jena.apache.org/tutorials/rdf_api.html
- **SPARQL Tutorial**
  - https://jena.apache.org/tutorials/sparql.html
- **SPARQL Recommendation (W3C)**
  - https://www.w3.org/TR/sparql11-query/