Unit 14:
Introduction to Semantics and RDF

MarkLogic

Learning Objectives

- Describe example use cases for semantics in MarkLogic.
- Describe RDF, triples, graphs and IRIs.
- Load triple data.
- Use SPARQL to query triples.

MarkLogic

The Semantic Web

To a computer, the Web is a flat, boring world, devoid of meaning. This is a pity, as in fact documents on the Web describe real objects and imaginary concepts, and give particular relationships between them.

For example, a document might describe a person. The title document to a house describes a house and also the ownership relation with a person.

Adding semantics to the Web involves two things: allowing documents which have information in machine-readable forms, and allowing links to be created with relationship values.

Only when we have this extra level of semantics will we be able to use computer power to help us exploit the information to a greater extent than our own reading.

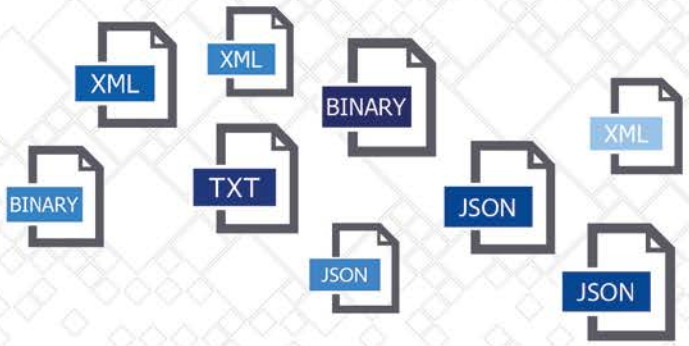
Tim Berners-Lee

MarkLogic

MarkLogic Semantics Architecture

MarkLogic

Better Together

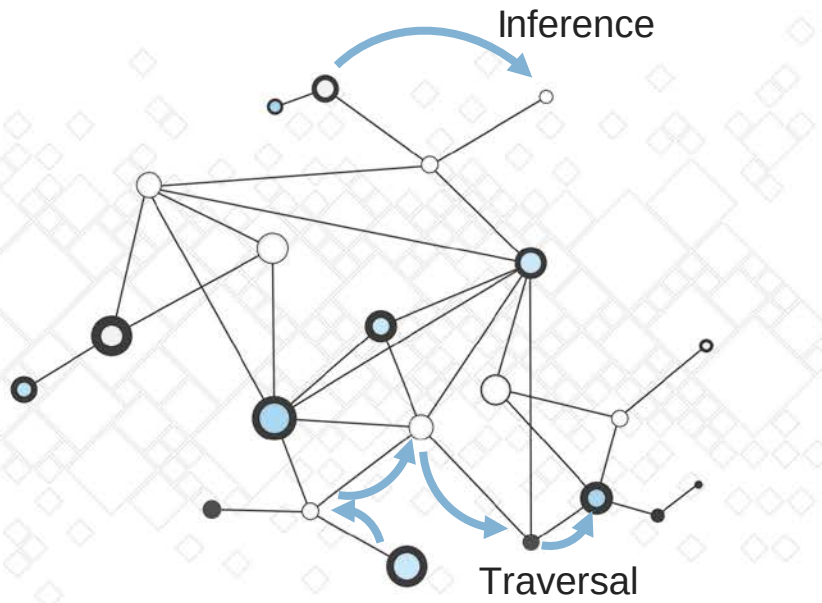


people	
Model: **Person**	
first_name	string
last_name	string
address1_id	integer
address2_id	integer

addresses	
Model: **Address**	
street_number	string
street_name1	string
street_name2	string
city_id	integer

states	
Model: **State**	
name	string
state_id	integer

cities	
Model: **City**	
name	string
state_id	integer



Document Store + Data Store + Triple Store

MarkLogic

Example

- Triples make it easy to do joins

TABLE: BOOKS

ID	TITLE	AUTHOR	PRICE_USD
111	Moby Dick	Herman Melville	9.99

TABLE: AUTHOR

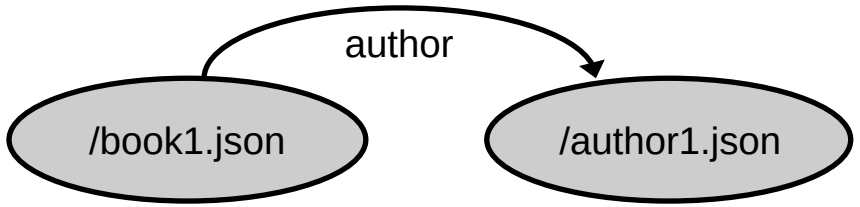
ID	NAME	BIRTHDATE	BIRTHPLACE
999	Herman Melville	01-08-1819	New York City

URI = /book1.json

```
{
  “title”: “Moby Dick”,
  “author”: “Herman Melville”,
  “price_usd”: “9.99”
}
```

URI = /author1.json

```
{
  “name”: “Herman Melville”,
  “birthdate”: “01-08-1819”,
  “birthplace”: “New York City”
}
```



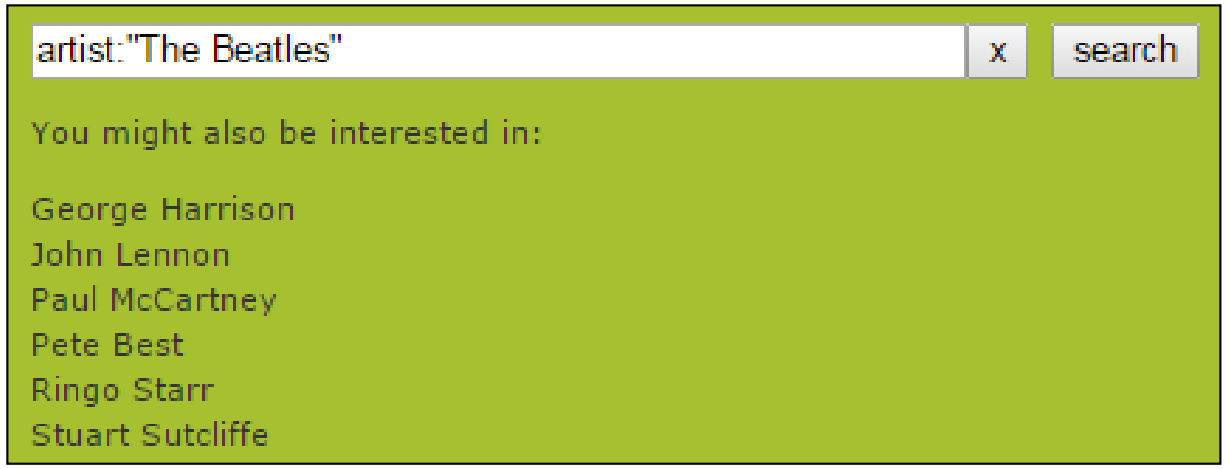
Example Application

- Enhancing search results with semantic data – use facts to add context

MarkLogic

Example Application

- Use triples to establish relationships for search expansion



MarkLogic

Geophoto Home Map Search

Semantic Info

Washington, D.C., United States

Anthem: The Star-Spangled Banner

Currency: USD

Area: 9826675 km2

The United States of America (USA), commonly referred to as the United States (US), America or simply the States, is a federal republic consisting of 50 states and a federal district. The 48 contiguous states and the federal district of Washington, D.C., are in central North America between Canada and Mexico. The state of Alaska is the northwestern part of North America and the state of Hawaii is an archipelago in the mid-Pacific. The country also has five populated and nine unpopulated territories in the Pacific and the Caribbean. At 3.79 million square miles (9.83 million km2) in total and with around 317 million people, the United States is the third or fourth-largest country by total area and third largest by population. It is one of the world's most ethnically diverse and multicultural nations, the product of large-scale immigration from many countries. The geography and climate of the United States is also extremely diverse, and it is home to a wide variety of wildlife.Paleo-indians migrated from Asia to what is now the U.S. mainland around 15,000 years ago, with European colonization beginning in the 16th century. The United States emerged from 13 British

MarkLogic

Basic Terminology

- RDF
  - Resource Description Framework
  - W3C Spec with a defined vocabulary for representing facts
- RDF Triple
  - Each triple represents a single fact
  - Contains a subject, predicate and object
- Graph
  - A set of RDF triple statements or patterns
- IRI
  - International Resource Identifier
  - Uniquely identifies resources in an RDF triple
  - May contain Unicode characters (unlike URIs which are ASCII)

MarkLogic

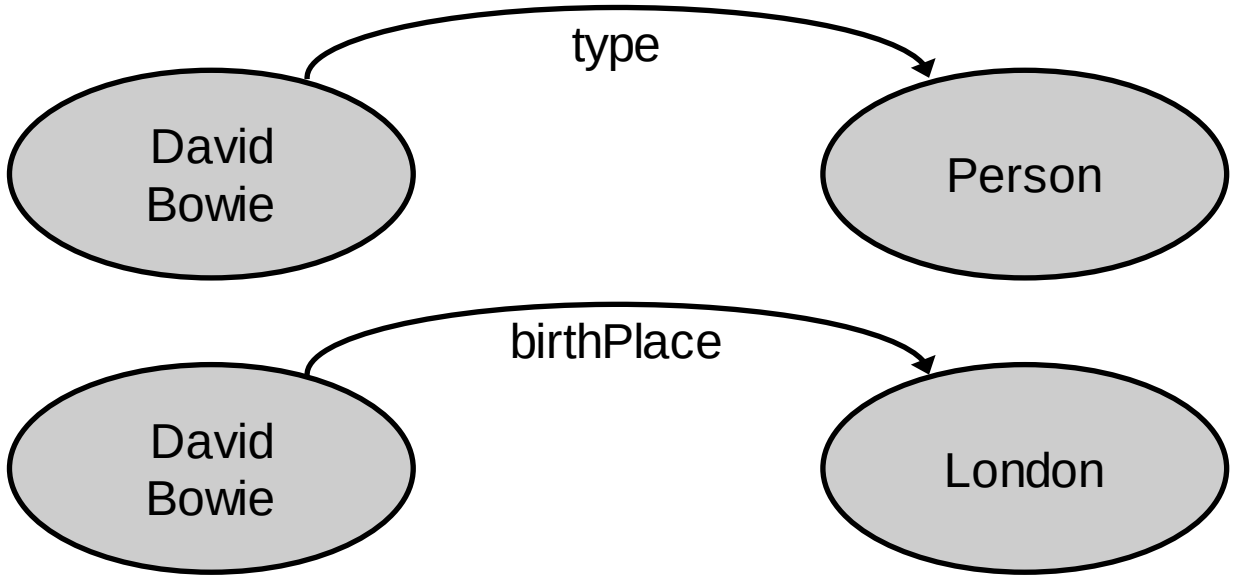
RDF and Triples

- Resource Description Framework (RDF) is a W3C specification with a defined vocabulary.
- The framework is an abstract data model to represent facts, comprising a subject, predicate, and an object as a triple.
  - Subject
    - A representation of a resource. For example a person or an entity.
  - Predicate
    - A representation of a property or characteristics of the subject or of the relationship between the subject and the object.
    - Also known as an arc or edge.
  - Object
    - A property value.
    - May be a typed literal. (xsd:double, xsd:string, xsd:date, etc.)
    - May be the subject of other triples.

MarkLogic

RDF Triples

- Example RDF triples:

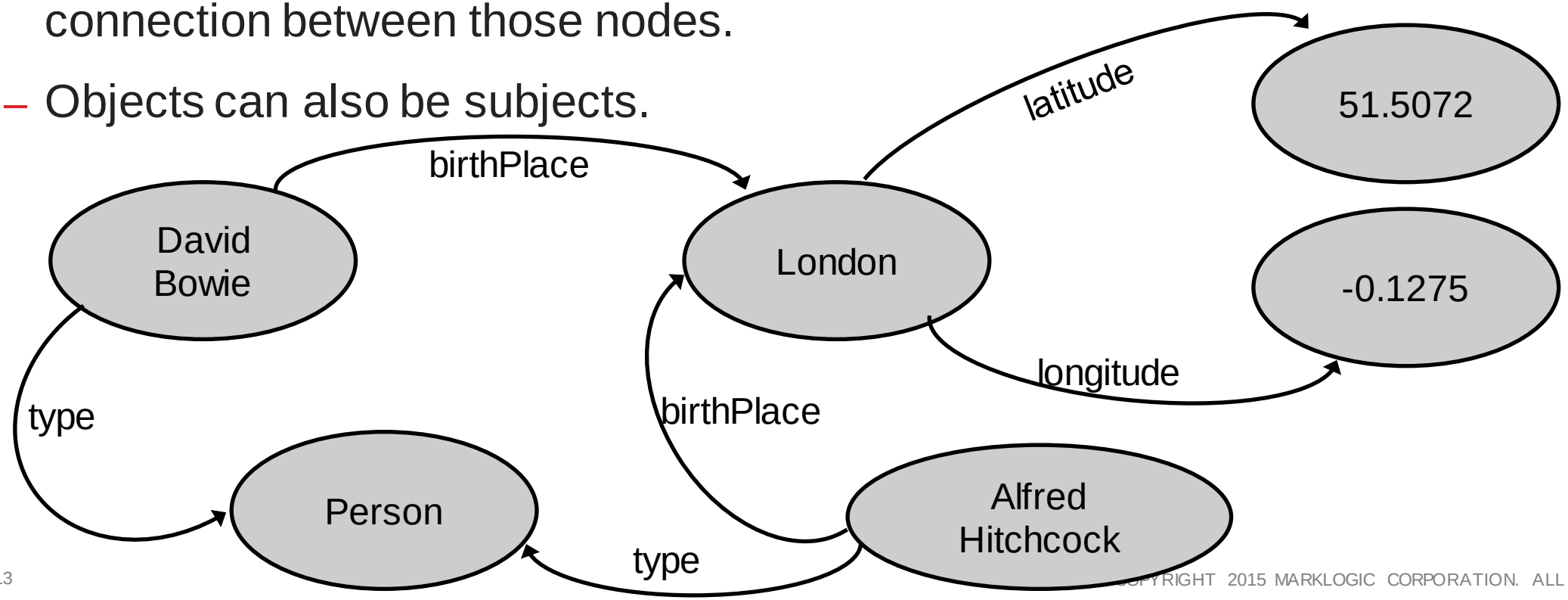


SUBJECT	PREDICATE	OBJECT
David Bowie	type	Person
David Bowie	birthPlace	London

MarkLogic

RDF Graphs

- Graphs
  - A set of RDF triple statements or patterns. In a graph-based RDF model, nodes represent subject or object resources, with the predicate providing the connection between those nodes.
  - Objects can also be subjects.



MarkLogic

IRIs

- Internationalized Resource Identifiers
  - A compact string that are used for uniquely identifying resources in an RDF triple. IRIs may contain characters from the Universal Character Set (Unicode/ISO 10646).
  - Allows for Chinese, Japanese Kanji, etc. characters
- IRI vs. URI
  - URIs (uniform resource identifiers) are limited to ASCII characters

MarkLogic

RDF Triples & IRIs

- As humans we might visualize triples as:

  - In the RDF data model triples are built using IRIs:

Example 1: Object is also a subject, therefore its reflected as IRI	
Subject	`<http://dbpedia.org/resource/David_Bowie>`
Predicate	`<http://dbpedia.org/ontology/birthPlace>`
Object	`<http://dbpedia.org/resource/London>`

Example 2: Object is a typed literal	
Subject	`<http://dbpedia.org/resource/London>`
Predicate	`<http://w3.org/2003/01/geo/wgs84_pos#lat>`
Object	`“51.5072”^^<http://www.w3.org/2001/XMLSchema#float>`

MarkLogic

Triple Data in MarkLogic

- Triples stored as XML documents
- 100 triples per document

```
<?xml version="1.0" encoding="UTF-8"?>
<sem:triples xmlns:sem="http://marklogic.com/semantics">
  <sem:origin>location from where the source data was loaded</sem:origin>
  <sem:triple>
    <sem:subject>http://dbpedia.org/resource/David_Bowie</sem:subject>
    <sem:predicate>http://dbpedia.org/ontology/birthPlace</sem:predicate>
    <sem:object>http://dbpedia.org/resource/London</sem:object>
  </sem:triple>
  ...
  ...
  ...
</sem:triples>
```

MarkLogic

Triple Data in MarkLogic

- Triples may also be embedded in documents

```
<?xml version="1.0" encoding="UTF-8"?>
<article>
  <info>
    <title>Chelsea Wins Match</title>
      <sem:triple>
        <sem:subject>http://example.org/article</sem:subject>
        <sem:predicate>http://example.org/mentions</sem:predicate>
        <sem:object>http://example.org/London</sem:object>
      </sem:triple>
      <sem:triple>
        <sem:subject>http://example.org/article</sem:subject>
        <sem:predicate>http://example.org/sport</sem:predicate>
        <sem:object>http://example.org/Football</sem:object>
      </sem:triple>
...
  </info>
</article>
```

MarkLogic

Loading Triple Data

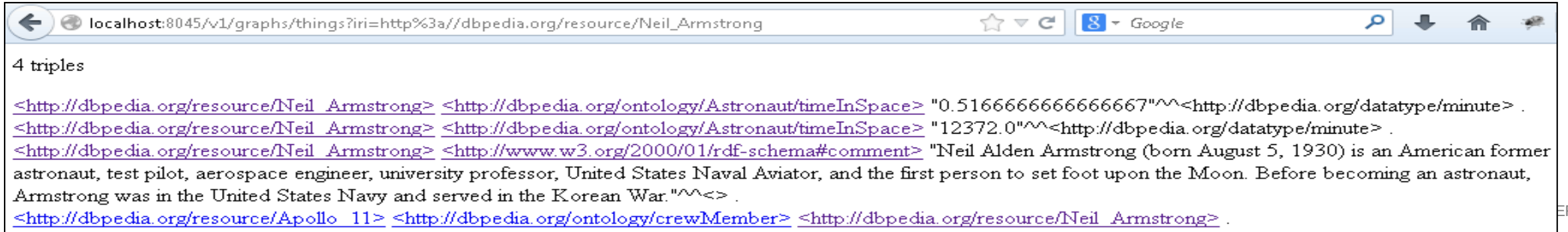
- Load triples embedded in XML documents by loading the document using preferred method of choice
- mlcp provides a fast way to load triples from files containing serialized RDF data
  - Many such data sources exist and are openly available
    - Example: The DBpedia data used in upcoming labs

```
mlcp.bat import -host localhost -port 8046 \
-username admin -password admin \
-input_file_path c:\my-source-location\data.nt \
-mode local \
-input_file_type RDF
```

MarkLogic

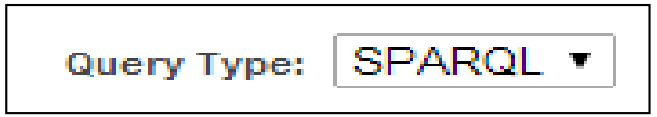
Explore Triple Data

- `/v1/graphs/things`
- REST endpoint enabling you to explore triples in a database
- Good starting point to learn about your data after ingestion
  - Shows a subset of subjects



MarkLogic

SPARQL

- **S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage
  - http://www.w3.org/TR/rdf-sparql-query/
- Use SPARQL in MarkLogic through:
  - Node.js and Java client APIs
  - REST endpoints: `/v1/graphs/sparql`
  - JavaScript APIs: `sem.sparql`
  - XQuery APIs: `sem:sparql()`
  - Query Console: 

MarkLogic

Introduction to SPARQL

- Tell me all the facts you know about a given subject (assume this triple exists)

```
<?xml version="1.0" encoding="UTF-8"?>
<sem:triples xmlns:sem="http://marklogic.com/semantics">
  <sem:triple>
    <sem:subject>http://dbpedia.org/resource/David_Bowie</sem:subject>
    <sem:predicate>http://dbpedia.org/ontology/birthPlace</sem:predicate>
    <sem:object>http://dbpedia.org/resource/London</sem:object>
  </sem:triple>
</sem:triples>
```

- This query:

```
SELECT ?p ?o
WHERE { <http://dbpedia.org/resource/David_Bowie> ?p ?o }
```

- Yields this result:

```
<http://dbpedia.org/ontology/birthPlace>
<http://dbpedia.org/resource/London>
```

MarkLogic

Introduction to SPARQL

- Assume this triple is in the database:

```
<?xml version="1.0" encoding="UTF-8"?>
<sem:triples xmlns:sem="http://marklogic.com/semantics">
  <sem:triple>
    <sem:subject>http://dbpedia.org/resource/David_Bowie</sem:subject>
    <sem:predicate>http://dbpedia.org/ontology/birthPlace</sem:predicate>
    <sem:object>http://dbpedia.org/resource/London</sem:object>
  </sem:triple>
</sem:triples>
```

- What would be the result of this query?:

```
SELECT ?s
WHERE {
        ?s
        <http://dbpedia.org/ontology/birthPlace>
        <http://dbpedia.org/resource/London>
      }
```

MarkLogic

Introduction to SPARQL

- Assume this triple is in the database:

```
<?xml version="1.0" encoding="UTF-8"?>
<sem:triples xmlns:sem="http://marklogic.com/semantics">
  <sem:triple>
    <sem:subject>http://dbpedia.org/resource/David_Bowie</sem:subject>
    <sem:predicate>http://dbpedia.org/ontology/birthPlace</sem:predicate>
    <sem:object>http://dbpedia.org/resource/London</sem:object>
  </sem:triple>
</sem:triples>
```

- Prefixes make unwieldy IRIs easy to use:

```
PREFIX db: <http://dbpedia.org/resource/>
PREFIX onto: <http://dbpedia.org/ontology/>

SELECT ?s
WHERE { ?s onto:birthPlace db:London }
```

MarkLogic

Introduction to SPARQL

- CURIEs
  - Compact URIs
  - Standard prefixes for commonly used vocabularies

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX db: <http://dbpedia.org/resource/>
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

MarkLogic

Introduction to SPARQL

- Assume your database contains triples from many sources
- What if you want to constrain which triples you query?
- One approach is to specify collections when ingesting triples:

```
xquery version "1.0-ml";

import module namespace sem = "http://marklogic.com/semantics" at "/MarkLogic/semantics.xqy";

sem:rdf-insert(
  sem:triple(sem:iri("http://dbpedia.org/resource/John_Lennon"),
             sem:iri("http://dbpedia.org/ontology/birthPlace"),
             sem:iri("http://dbpedia.org/resource/Liverpool")),
(), (), "music-graph", (), ())
,
sem:rdf-insert(
  sem:triple(sem:iri("http://dbpedia.org/resource/Charles_Dickens"),
             sem:iri("http://dbpedia.org/ontology/birthPlace"),
             sem:iri("http://dbpedia.org/resource/Landport")),
(), (), "literature-graph", (), ())
```

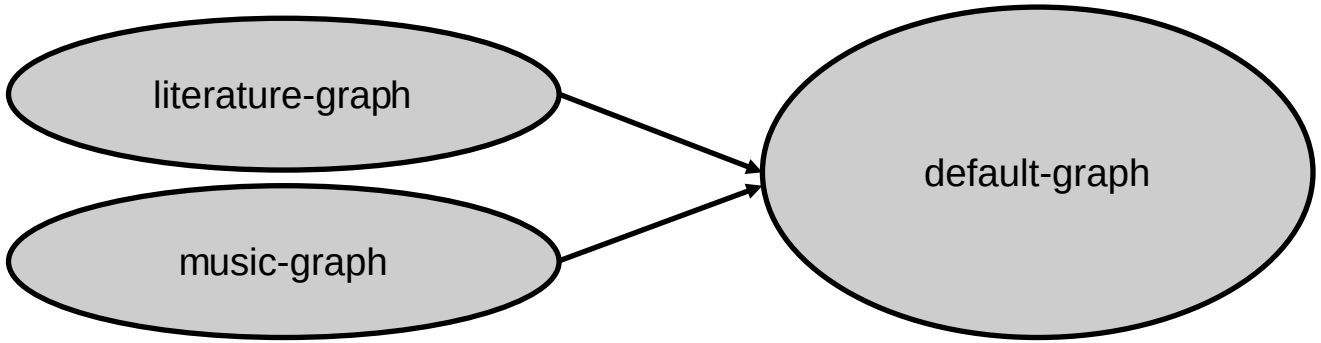
MarkLogic

Introduction to SPARQL

- Use FROM to constrain a SPARQL query
- This creates a union of the graphs specified in the FROM clause

```
PREFIX db: <http://dbpedia.org/resource/>
PREFIX onto: <http://dbpedia.org/ontology/>

SELECT ?s
FROM <music-graph>
FROM <literature-graph>
WHERE { ?s onto:birthPlace db:Liverpool }
```



MarkLogic

Introduction to SPARQL

- For queries with multiple triple patterns, end each triple with “.”
- The period is like AND in SQL

```
PREFIX db: <http://dbpedia.org/resource/>
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX dc:<http://purl.org/dc/elements/1.1/>

SELECT *
WHERE {
        ?s db:birthPlace db:Athens .
        ?s db:birthPlace db:Greece .
        ?s dc:description ?o
      }
```

MarkLogic

Introduction to SPARQL

- For triple patterns that share the same subject, use ";"

```
PREFIX db: <http://dbpedia.org/resource/>
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX dc:<http://purl.org/dc/elements/1.1/>

SELECT *
WHERE {
        ?s db:birthPlace db:Athens ;
           db:birthPlace db:Greece ;
           dc:description ?o
      }
```

MarkLogic

Introduction to SPARQL

- OPTIONAL keyword

```
#Give me the name of subjects that have a first and last name and #email.  But if they don't
have an email, I still want their name.

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

SELECT ?fn ?ln
WHERE{
        ?x foaf:givenName ?fn .
        ?x foaf:surname ?ln .
        OPTIONAL{?x foaf:email ?mb .}
    }
```

MarkLogic

Introduction to SPARQL

- `UNION` keyword
- OR logic

```
# find people who are Authors OR Novelists and their date of birth

prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix foaf: <http://xmlns.com/foaf/0.1/>
prefix dc: <http://purl.org/dc/elements/1.1/>
PPREFIX onto: <http://dbpedia.org/ontology/>

SELECT  ?person ?desc ?date
WHERE { ?person rdf:type foaf:Person .
      ?person dc:description ?desc .
      ?person onto:birthDate ?date .

      { ?person dc:description "Novelist" . }
UNION
      { ?person  dc:description "Author" . }
      }
```

MarkLogic

Introduction to SPARQL

- Ways to limit query results:
  - `FILTER, DISTINCT, LIMIT`
- What if a subject was described as a “Factory Worker”?

```
#This example uses FILTER to show only the subjects who were born in #London and whose
description contains the substring actor.
#The “I” flag indicates case insensitive pattern matching.

PREFIX db: <http://dbpedia.org/resource/>
PREFIX onto: <http://dbpedia.org/ontology/>
PREFIX dc:<http://purl.org/dc/elements/1.1/>

SELECT ?s ?desc
WHERE {
        ?s onto:birthPlace db:London .
        ?s dc:description ?desc .
FILTER (regex (?desc, "actor", "i"))
      }
```

MarkLogic

SPARQL and Node.js

- Must be a user with a role that has read permissions on the triples and the **sem:sparql** execute privilege.

```
var marklogic = require("marklogic");
var dbConn = require("./connections.js");
var mlAdmin = marklogic.createDatabaseClient(dbConn.mlAdmin);

var query = [
  "PREFIX db: <http://dbpedia.org/resource/>",
  "SELECT * ",
  "WHERE { db:John_Lennon ?p ?o }"
];

mlAdmin.graphs.sparql("application/sparql-results+json", query.join("\n")
).result(function (response) {
  console.log(JSON.stringify(response, null, 2));
}, function(error) {
  console.log(JSON.stringify(error, null, 2));
});
```

MarkLogic

Additional Resources

- Using MarkLogic Semantics
  - Free 1 day course from MarkLogic University focused entirely on Semantics
- mlu.marklogic.com/ondemand
  - Free short video tutorials on various topics, including Semantics
- docs.marklogic.com
  - Semantics Developer Guide
- http://www.w3.org/TR/rdf-sparql-query/

Labs: Unit 14

Exercise 1: Configure the Triple Index
Exercise 2: Load Triple Data
Exercise 3: Explore Triple Data
Exercise 4: Query Triple Data with SPARQL
Exercise 5: Query Triple Data from a Node.js App

Unit Review Question 1:

When loaded in MarkLogic, triples are modeled as:

1. JSON
2. XML
3. N-Triples
4. Quads

Unit Review Question 1:

When loaded in MarkLogic, triples are modeled as:

1. JSON
2. **XML**
3. N-Triples
4. Quads

Unit Review Question 2:

In order to run SPARQL queries, you must be a user with a role that has which execute privilege:

1. sem:update
2. sem:triples
3. sem:sparql
4. No special privilege required.

Unit Review Question 2:

In order to run SPARQL queries, you must be a user with a role that has which execute privilege:

1. sem:update
2. sem:triples
3. **sem:sparql**
4. No special privilege required.

Unit Review Question 3:

To load triples using mlcp, set the input_type=

1. XML
2. Aggregates
3. TRIPLES
4. RDF

Unit Review Question 3:

To load triples using mlcp, set the input_type=

1. XML
2. Aggregates
3. TRIPLES
4. **RDF**