

# COMP 474 UU,COMP 6741 UU 2204

[Home](#) / [My courses](#) / [COMP-474-2204-UU](#) / 7 February - 13 February / [Lab Session #4](#)

## Lab Session #4

### Introduction

Welcome to lab #4. This week's lab is about learning the [SPARQL query language](#) for knowledge graphs. As always, if you did not finish any of the tasks from the previous week(s), make sure you catch up with any remaining tasks, since some of the new tasks build on previous work.

### Follow-up Lab #3

Here's an [example solution for the FOCU university schema & data](#) from the previous lab. Of course, your solution might look slightly different. If you're confused about any part of it, just ask in the Moodle Discussion forum!

### Task #1: SPARQL 101

Here is a nice [SPARQL cheat sheet](#) (well, more like cheat slides) that are helpful for getting a quick overview (note that we did not discuss all the details of SPARQL in class). To learn SPARQL it's best to experiment with a number of different queries. Many (but not all) public knowledge graphs provide a public SPARQL interface (a so-called *SPARQL endpoint*), for example [this one at DBpedia](#).

Note that these public, open endpoints are typically very restrictive in terms of query result size, execution time, and query memory use, so don't be surprised if you get an error message instead of a result. Sometimes it helps to retry a query or limiting the result size (e.g., with a `LIMIT 50`).

#### Task #1.1: DBpedia

To gain an impression on how powerful graph queries can be, here are two examples that you can try out using DBpedia's public SPARQL endpoint:

1. [Musicians who were born in Berlin:](#)

```
SELECT ?name ?birth ?description ?person
WHERE {
    ?person a dbo:MusicalArtist .
    ?person dbo:birthPlace :Berlin .
    ?person dbo:birthDate ?birth .
    ?person foaf:name ?name .
    ?person rdfs:comment ?description .
    FILTER (LANG(?description) = 'en') .
} ORDER BY ?name
```

2. [Soccer players, who are born in a country with more than 10 million inhabitants, who played as goalkeeper for a club that has a stadium with more than 30.000 seats and where the club's country is different from the player's birth country:](#)

```

SELECT DISTINCT ?soccerplayer ?countryOfBirth ?team ?countryOfTeam ?stadiumcapacity
WHERE {
    ?soccerplayer a dbo:SoccerPlayer ;
        dbo:position|dbp:position <http://dbpedia.org/resource/Goalkeeper\_\(association\_football\)>;
        dbo:birthPlace|dbo:country* ?countryOfBirth ;
        dbo:team ?team .
    ?team dbo:capacity ?stadiumcapacity ;
        dbo:ground ?countryOfTeam .
    ?countryOfBirth a dbo:Country ;
        dbo:populationTotal ?population .
    ?countryOfTeam a dbo:Country .
    FILTER (?countryOfTeam != ?countryOfBirth)
    FILTER (?stadiumcapacity > 30000)
    FILTER (?population > 10000000)
} ORDER BY ?soccerplayer

```

You can see how intelligent assistants like Watson, Siri, Alexa etc. are able to answer so many questions, by querying their knowledge graphs.

Now, try to write your own queries to determine:

1. All *universities located in Canada*, with their city and *optionally* (if it exists) their home page.
2. All people who *studied at Concordia University* (and are listed in DBpedia), together with their description (in English or any other language you prefer). Hint: look for the [Alma Mater](#) predicate and make sure you understand its domain & range.

## Task #2: Your own SPARQL Server

As discussed in the lecture, there are a number of options for setting up your own SPARQL *endpoint*; here, we will use the open source [Apache Fuseki](#) server.

### Task #2.1: Getting started with Fuseki

1. Start by downloading the fuseki binary distribution from <https://jena.apache.org/download/index.cgi>
2. Unpack the archive and make the server script executable:

```

> tar xf apache-jena-fuseki-3.17.0.tar.gz
> cd apache-jena-fuseki-3.17.0
> chmod u+x fuseki-server

```

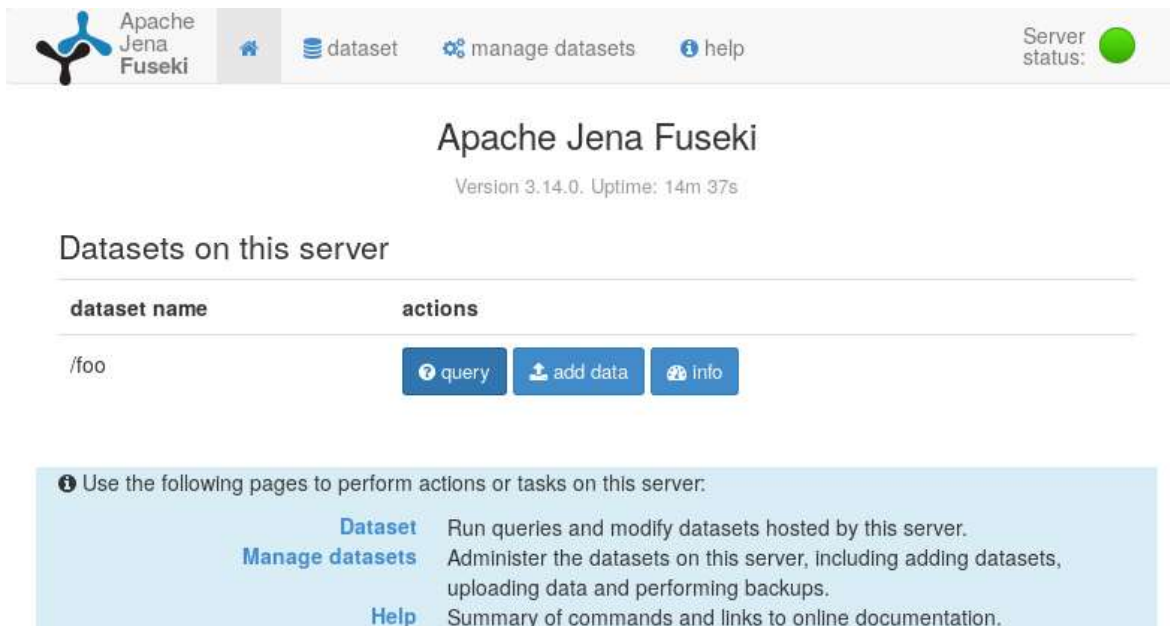
3. Now you can run the server (note that you must have a JDK installed):

```

> ./fuseki-server
[2020-02-09 09:55:15] Server      INFO  Apache Jena Fuseki 3.17.0
[2020-02-09 09:55:15] Config     INFO  FUSEKI_HOME=/home/rene/fuseki/apache-jena-fuseki-3.17.0/.
[2020-02-09 09:55:15] Config     INFO  FUSEKI_BASE=/home/rene/fuseki/apache-jena-fuseki-3.17.0/run
[2020-02-09 09:55:15] Config     INFO  Shiro file: file:///home/rene/fuseki/apache-jena-fuseki-3.17.0/run/shiro.ini
[2020-02-09 09:55:17] Server      INFO  Started 2020/02/09 09:55:17 EST on port 3030

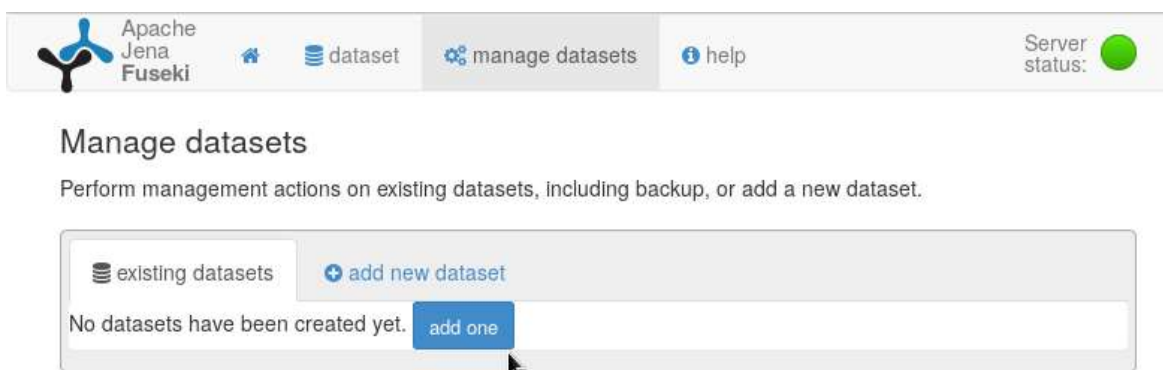
```

As you can see, it starts a local server on port 3030, so open <http://localhost:3030/> in your browser. You should see the main page:



The screenshot shows the Apache Jena Fuseki dashboard. At the top, there is a navigation bar with the Apache Jena Fuseki logo, a home icon, a 'dataset' link, a 'manage datasets' link, and a 'help' link. On the right, the 'Server status' is indicated by a green circle. Below the navigation bar, the title 'Apache Jena Fuseki' is displayed, followed by the version 'Version 3.14.0' and uptime 'Uptime: 14m 37s'. The main section is titled 'Datasets on this server'. It contains a table with two columns: 'dataset name' and 'actions'. The first row shows the dataset name '/foo' and three action buttons: 'query', 'add data', and 'info'. Below the table, there is a light blue box with an information icon and text: 'Use the following pages to perform actions or tasks on this server:'. This box contains three links: 'Dataset' (Run queries and modify datasets hosted by this server.), 'Manage datasets' (Administer the datasets on this server, including adding datasets, uploading data and performing backups.), and 'Help' (Summary of commands and links to online documentation.).

4. Before you can do anything else, you must create a *Dataset*, so select the first open above:



The screenshot shows the 'Manage datasets' page in the Apache Jena Fuseki interface. The navigation bar is the same as in the previous screenshot, but the 'manage datasets' link is now active. The title 'Manage datasets' is displayed, followed by the instruction: 'Perform management actions on existing datasets, including backup, or add a new dataset.' Below this, there is a section with two tabs: 'existing datasets' and 'add new dataset'. The 'existing datasets' tab is selected, showing the message 'No datasets have been created yet.' and a blue button labeled 'add one'.

Give your dataset a name and use the "In-memory" option (this means your triples will not be stored persistently, but this is fine for some first experiments).

5. You should now see your new dataset under "existing datasets" and can start to upload triples. Use your university (FOAF/FOCU) triples from two weeks ago (or any other triple file you have):

The screenshot shows the Apache Jena Fuseki web interface. At the top, there is a navigation bar with the Fuseki logo, a home icon, a 'dataset' tab, a 'manage datasets' icon, and a 'help' icon. On the right, the 'Server status' is indicated by a green circle. Below the navigation bar, a 'Dataset:' dropdown menu is set to '/foo'. The main content area has tabs for 'query', 'upload files', 'edit', and 'info'. The 'upload files' tab is active, showing the 'Upload files' section. It includes instructions: 'Load data into the default graph of the currently selected dataset, or the given named graph. You may upload any RDF format, such as Turtle, RDF/XML or TRIG.' Below this, there is a 'Destination graph name' field with the text 'Leave blank for default graph'. Under 'Files to upload', there are two buttons: '+ select files...' and 'upload all'. A file named 'focu-example.ttl' (1.3kb) is listed with the result 'success. 29 triples'. A green progress bar is at the bottom of the upload section.

6. Now you are ready to send SPARQL queries to your server: go to the *query* tab and start querying your data:

The screenshot shows the Apache Jena Fuseki web interface with the 'query' tab active. The 'Dataset:' dropdown is still set to '/foo'. The 'SPARQL query' section has the instruction: 'To try out some SPARQL queries against the selected dataset, enter your query here.' Below this, there are 'EXAMPLE QUERIES' buttons for 'Selection of triples' and 'Selection of classes'. A 'PREFIXES' section shows buttons for 'rdf', 'rdfs', 'owl', and 'xsd'. The 'SPARQL ENDPOINT' field contains 'http://localhost:3030/foo/sparql'. The 'CONTENT TYPE (SELECT)' dropdown is set to 'JSON', and the 'CONTENT TYPE (GRAPH)' dropdown is set to 'Turtle'. At the bottom, a text area contains a SPARQL query: 

```
1
2
3 SELECT ?subject ?predicate ?object
4 WHERE {
5   ?subject ?predicate ?object
6 }
7 LIMIT 25
```

 To the right of the query area are icons for sharing, full screen, and execution.

## Task #2.2: Learning SPARQL with Fuseki

With your own SPARQL server, you can now experiment with queries without relying on an external server. Go through the [SPARQL Tutorial](#) available on the Apache Jena, which comes with datasets and query exercises.

## Task #3: SPARQL with Python

This task builds on the code you've previously developed with RDFlib.

### Task #3.1: RDFlib

Continue the RDFlib introduction at [Querying with SPARQL](#).

### Task #3.2: Query University Data

Load some of the triples you created in the lab last week (FOAF/FOCU) into your program and query it using SPARQL. Write queries to:

1. List all students, sorted by age
2. Find all predicates and objects for a given student, searching by first name (e.g., "Joe")
3. Print a count of all students by university

### Task #3.3: Smart University Agent v1.1

Modify the code for your intelligent university agent from the previous week to use SPARQL queries, rather than manually traversing the graph.

That's all for this lab!

Last modified: Wednesday, 10 February 2021, 5:37 PM

[◀ Worksheet #04](#)

Jump to...

[Lecture Slides #06 ▶](#)

You are logged in as Raj Mistry (Log out)

COMP-474-2204-UU

Academic Integrity

Academic Assessment Tool

English (en)

Deutsch (de)

English (en)

Español - Internacional (es)

Français (Canada) (fr\_ca)

Français (fr)

Italiano (it)

العربية (ar)

