**Group 4:**

Vũ Minh Hiếu  
Nguyễn Thanh Tùng  
Nguyễn Đức Mạnh  
Đào Tuấn Anh

**Instructor:**

Nguyễn Bá Đạt, PhD

Table of contents

Semantic web (semester i 2018-2019)

[Year]

Table of contents

**1. Problem overview**  
**1.1 Description**  
**1.2 Requirements**

**2. Solution proposal**  
**2.1 Pipeline**  
**2.2 Tools**

**3. Solution details**  
**3.1 Web crawling**   
 **3.1.1 mp3.zing.vn**  
 **3.1.2 nhaccutui.com**  
 **3.1.3 chieu-cao.net**  
 **3.1.4 Storing format**   
 **3.1.5 Note**s   
**3.2 KB construction**  
 **3.2.1 Artist data**  
 **3.2.2 Song data**

**4. Information extraction**  
**4.1 Predicate 1 (birthName), 3 (birthDate) and 11 (image)**  
**4.2 Predicates with respect to songs (composedBy, performedBy and lyric)**  
**4.3 Predicates 12 (linkToWiki)**  
**4.4 Predicate 8 (instrument) and 9 (profession)**  
 **4.4.1 Methodology**  
 **4.4.2 PipelineText extractions**  
 **4.4.3 Information extraction**  
**4.5 Predicate 4 (Height)**  
**4.6 Predicate 2 (birthPlace)**  
 **4.6.1 Methodology**  
 **4.6.2 Building the cities/provinces dictionary**  
 **4.6.3 Indicators**  
**4.7 Predicate 10 (memberOf)**  
 **4.7.1 Challenges**  
 **4.7.2 Our approach**  
**4.8 Handling duplicates**  
 **4.8.1 Challenges**  
 **4.8.2 Methodology**  
 **4.8.3 Notes**

**5. Data Alignment**  
**5.1 Objective**  
**5.2 Merging process**  
 **6.2.1 Merging artists information**  
 **6.2.2 Merging songs information**

**6. Making SPARQL endpoint**  
**6.1 Export the RDF**  
**6.2 Import to Virtuoso**

**7. Summary**

1. **Problem overview**
   1. **Description**

The objective is to build a knowledge base (KB) for Vietnamese music domain using information extracted from some websites (including <http://nhaccuatui.com> & <https://mp3.zing.vn> & <http://chieu-cao.net>). The result is expected to be a SPARQL endpoint.

* 1. **Requirements**

**The predicates to be extracted are:**

* **With respect to an artist:**
* Predicate 1: *birthName* (Person, Text)

The artist’s birth name.

* Predicate 2: *birthPlace* (Person, Location)

The vi.wikipedia page of city where that artist was born

* Predicate 3: *birthDate* (Person, Date/Time)

The artist’s date of birth, with the format being yyyy-mm-dd

* Predicate 4: *Height* (Person, Number)

The artist’s height in centimetres

* Predicate 8: *instrument* (Person, Text)

List of instruments that the artist can play

* Predicate 9: *profession* (Person, Text)

List of the artist’s genres of music

* Predicate 10: *memberOf* (Person, Organization)

Bands that the artist was/is a member of (with correspoding temporal info)

* Predicate 11: *image* (Person, url)

An url to an image of the artist

* Predicate 12: *linkToWiki* (Person, url) (vi.wikipedia.org)

The vi.wikipedia link of the artist

* **With respect to a song:**
* Predicate 5: *composedBy* (Song, Person/Organization)

List of artists who composed/wrote the song

* Predicate 6: *performedBy* (Song, Person/Organization)

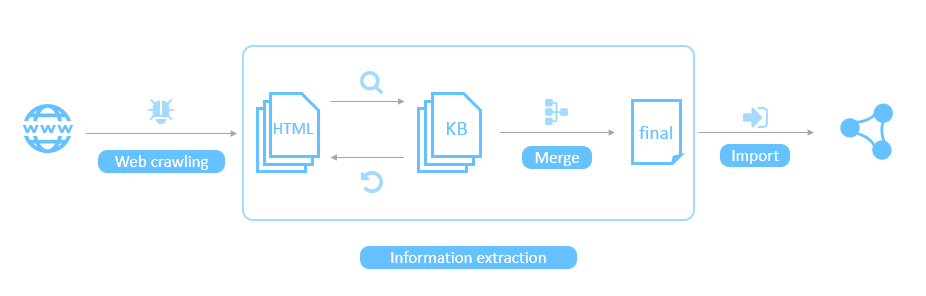
List of artists who performed the song

* Predicate 7: *lyric* (Song, Text)

The lyric of the song

1. **Solution proposal**
   1. **Pipeline**

* Crawl necessary pages and store them for later processing
* Extract information from each site
* Combine all the extracted information (information alignment)
* Import to a SPARQL endpoint



* 1. **Tools**
* Programming language: python 3
* Web scraping: scrapy and some python built-in modules
* Html parsing: BeautifulSoup4
* NLP: underthesea
* Database: Virtuaso

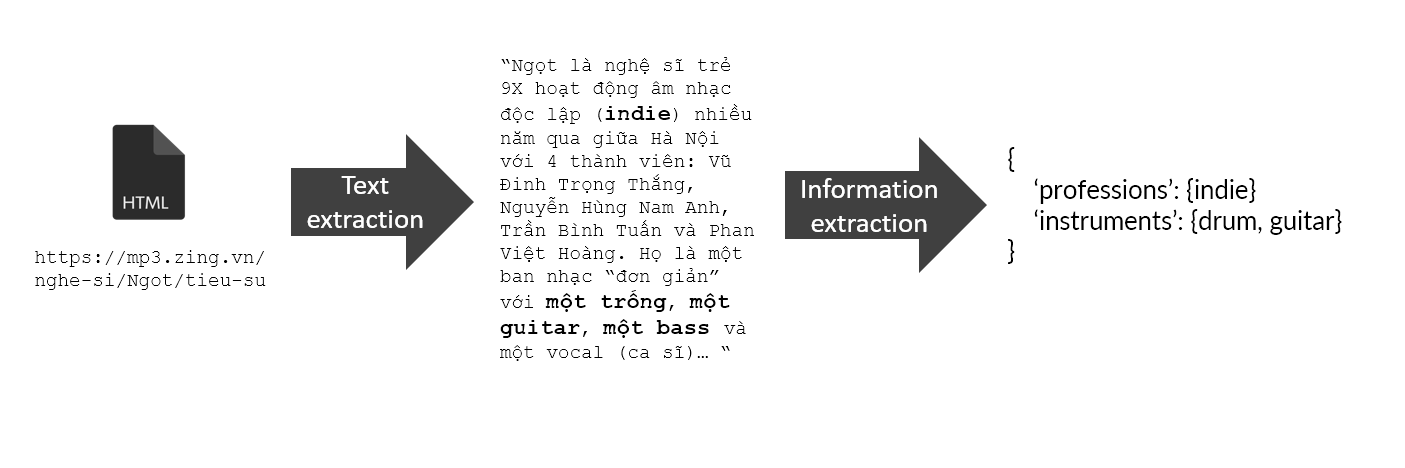
1. **Solution details**
   1. **Web crawling**
      1. **mp3.zing.vn**

* Follow all links starting from <https://mp3.zing.vn/the-loai-nghe-si>, only crawl those that ends with ‘/tieu-su’
* Crawl all the song list pages in every biography page (e.g. <https://mp3.zing.vn/nghe-si/JustaTee/bai-hat>, <https://mp3.zing.vn/nghe-si/JustaTee/bai-hat?&page=4>)
* Crawl all the song pages listed in every song list page (e.g. <https://mp3.zing.vn/bai-hat/Trong-Rong-JustaTee/ZWZBZBEC.html>)
* Ignore links that contain the symbol “#”
* Might contain some 404 pages
  + 1. **nhaccutui.com**
* Follow and crawl all artist biography pages starting from <https://www.nhaccuatui.com/nghe-si.html>
* Crawl all the song list pages in every biography page (e.g. <https://www.nhaccuatui.com/nghe-si-Rhymastic.bai-hat.html>, <https://www.nhaccuatui.com/nghe-si-Rhymastic.bai-hat.17.html>)
* Crawl all “Official” song pages listed in every song list page (e.g. <https://www.nhaccuatui.com/bai-hat/treasure-rhymastic.6oKIssEnzPa5.html>)
  + 1. **chieu-cao.net**
* Follow and crawl all pages starting from <https://chieu-cao.net/category/ca-sy>
* Result are store in a folder structure with the relative path of a .html file is the route of the corresponding URL (e.g. the page <https://chieu-cao.net/thong-tin-tieu-su-tien-cookie.html> can be found at …/chieu-cao.net/thong-tin-tieu-su-tien-cookie.html
* 4557 pages
  + 1. **Storing format**
* **For mp3.zing.vn and nhaccuatui.com**:
* Store list of URLs in .csv files
* Store html text in to .txt files, at most **10000** pages per file.
* Before every html content, there is a line contains the url of the content and a line contains the delimiter string (“##### NEW FILE #####”)
* Html pages are stored in the same order as the order of their corresponding url in .csv files
* Artist biography pages are stored in files begin with “tieu\_su\_”, song list pages are stored in files begin with “song\_list\_”, song pages are stored in files begin with “songs\_”
* **mp3.zing.vn** has **588861** pages which are stored in **3** “tieu\_su\_” files, **4** “song\_list\_” files and **52** “songs\_” files with the total size of **58.5 GB** approximately
* **nhaccuatui.com** has **150176** pages which are stored in **1** “tieu\_su\_” files, **3** “song\_list\_” files and **15** “songs\_” files with the total size of **26.1 GB** approximately
* **For chieu-cao.net:**

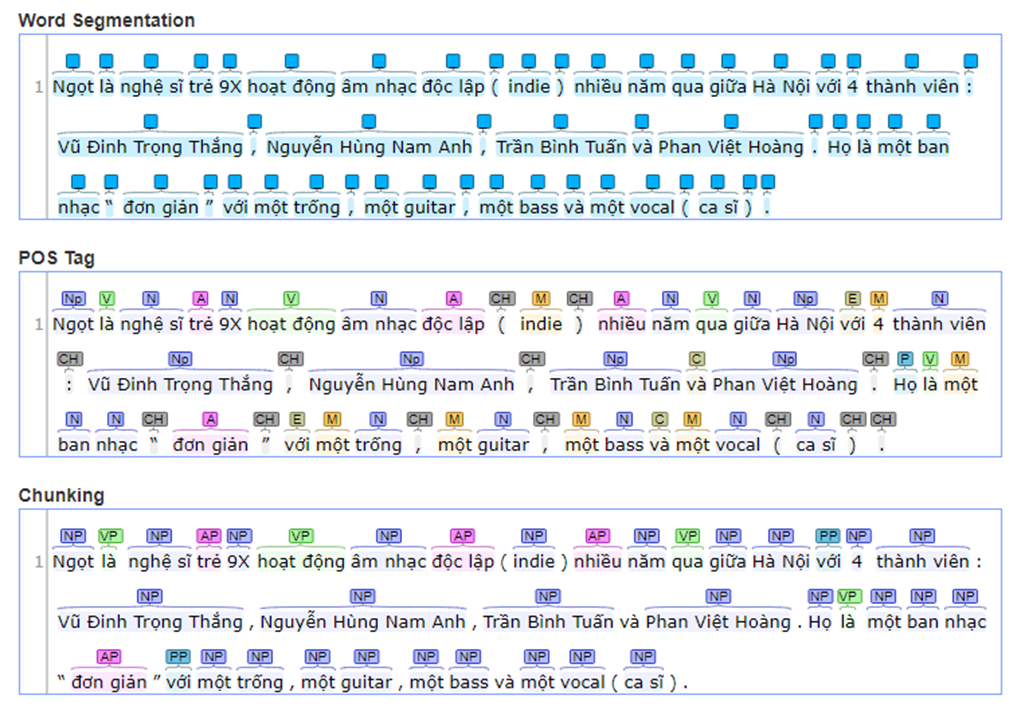
Html files are stored in a directory tree similar to that of the website

* + 1. **Notes**
* All data can be found at <https://drive.google.com/drive/u/1/folders/1SxL5OSiffIjonv_uUGWZYThHRjSzlJgo?fbclid=IwAR27v7FJaSiwHrIYimzASt9SlEJiY7ohOB4tBaid2TdEt1Q4k3CuygrqTiI>
* Please read the README file (Vietnamese) for more information
  1. **KB construction**
     1. **Artist data**
* Artist data is store in 2 python dictionary objects, 1 for mp3.zing.vn and 1 for nhacuatui.com, let’s call them kb objects.
* The kb’s keys are the URLs of every artist biography collected after the process described in section 3.1
* The kb’s values are dictionary objects containing:
  + ‘name’: string – the name of the artist
  + ‘city’: string – URL of the vi.wikipedia page about the city where the artist was born
  + ‘country’: string – URL of the vi.wikipedia page about the country where the artist comes from
  + ‘birth\_name’: string – the real name of the artist
  + ‘dob’: string of date in dd/mm/yyyy format – the date of birth of the artist
  + ‘height’: int – the height of the artist in cm
  + ‘image’: string – URL to a image of the artist
  + ‘instruments’: set of strings – instruments that the artist can play
  + ‘genres: set of strings – the artist’s genres
  + ‘member\_of’: set of strings – name of bands that the artist was and/or is a member of (could include temporal info later)
  + ‘wiki’: string – URL of the vi.wikikipedia page about the artist
    1. **Song data**
* Stores in 2 kb objects like artist data
* The kb’s keys are the URLs of every song collected after the process described in section 3.1
* The kb’s values are dictionary objects containing:
  + ‘name’: string – the name of the song
  + ‘composed\_by’: set of strings - URLs of the biography pages of all the artists that compose the song
  + ‘performed\_by’: set of strings – URLs of the biography pages of all the artists that perform the song
  + ‘lyric’: string – the lyric of the song

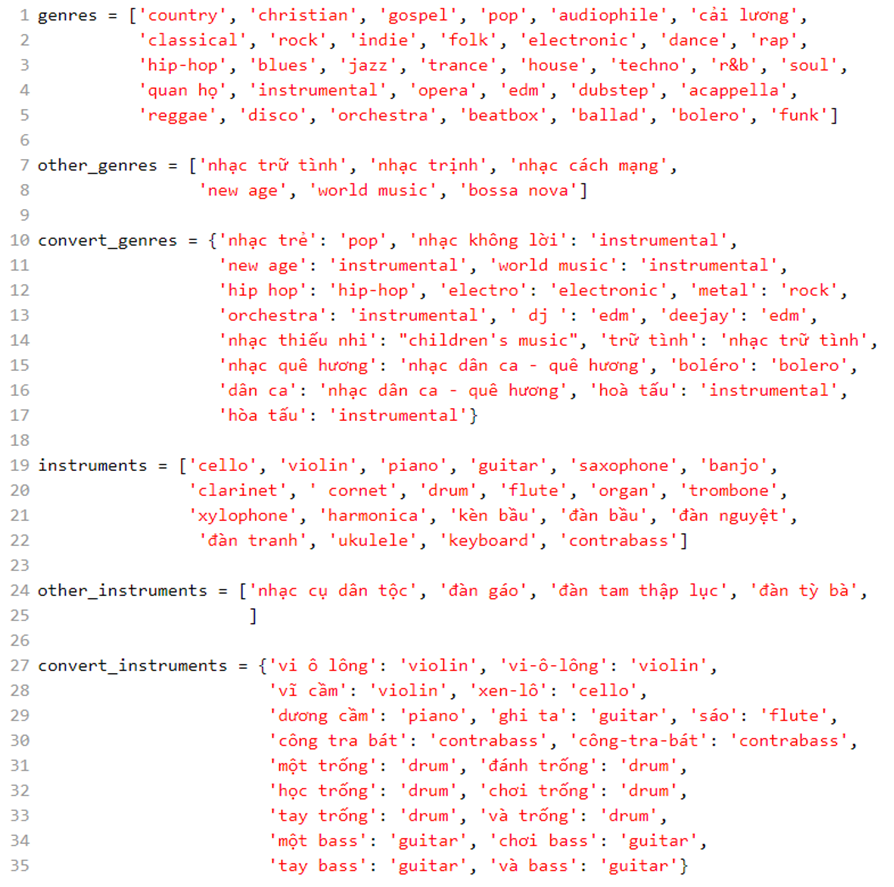
1. **Information extraction**
   1. **Predicate 1 (birthName), 3 (birthDate) and 11 (image)**

* These are easiest
* Utilize the fact that information is embedded in a specific html tag
* Information extraction can be done easily by selecting the appropriate html tag
* “birthDate” however, needs further processing, as it appears in various format
  + From all the date extracted (numbers are approximate): 5% are invalid, 50% have missing component(s), 1% have month in text (March 26, 1996), 1% with “.” as component delimiter (26.3.1996), 43% with “/“ (26/3/1996)
  + Assume that day always come before month
  + If it only has 1 component, we will assume that component is the year
  + If it has 2 components, based on the value to decide whether day or year is missing
  + Convert date to YYYY-MM-DD format, using those notation as padding in case of missing components
  1. **Predicates with respect to songs (composedBy, performedBy and lyric)**
* The approach is similar to that of extracting predicate 1, 3 and 11.
* Performers are stored as URLs to artists’ biography pages for later data alignment
  1. **Predicates 12 (linkToWiki)**
* Using the provided infoxboxalldump file, which contains over 1 million pieces of html
* List all artists’ name and provinces/cities in Vietnam, select all inforbox items whose entity’s name is a substring of any member of the aforementioned list
* Store the result in a kb object with entity’s name as the primary key
  1. **Predicate 8 (instrument) and 9 (profession)**
     1. **Methodology**
* Extract biography section from html as text
* Scan the text for occurrences of instruments and genres
  + 1. **Pipeline**
    2. **Text extractions**

**Using css selectors to extract biography contents**

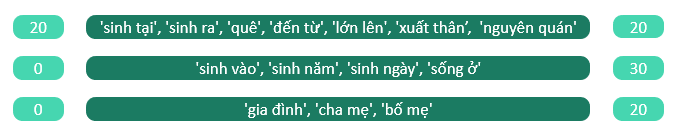
* **nhaccuatui.com:** ‘#divDescription’
* **mp3.zing.vn:** requires more than 1 selector due to inconsistent html layout
  + 'body > div.wrapper-page > div.wrap-body.group.page-artist-all.page-artist.container > div.wrap-2-col > div.wrap-content > div > div.row > div > div' (majority)
  + body > div.wrapper-page > div.full-banner > div.wrap-body.group.page-artist-all.page-artist.container > div.wrap-2-col > div.wrap-content > div > div.row > div > div'
  + 'body > div.wrapper-page > div > div.container > div.wrap-body.group.page-artist-all.page-artist.container > div.wrap-2-col > div.wrap-content'
    1. **Information extraction**
* **First attempt:** Apply pos tagging and dependency parsing
  + Word segmentation alone is computationally expensive
  + Impractical, cannot scale (having 20000+ samples)

*An example of using NLP method*

* **Second attempt:** Build a lookup dictionary manually
  + Time-consuming
  + Cannot cover all cases
  + Need to standardize different terms that have the same meaning
  + Could include false cases
  + Safe choice, a good starting point

*The genres and instruments dictionary*

* 1. **Predicate 4 (Height)**
* **Design a regular expression**: *“1\s{0,2}m\s{0,2}\d{1,2}|1\d\d\s{0,2}cm|1[\.,]\d{1,2}\s{0,2}m|1\d\d[,\.]\d?\s{0,2}cm”*
* **This covers 4 cases:**
  + *1\s{0,2}m\s{0,2}\d{1,2}* – height strings that appear in metres with “m” in between (“1 m 74”, “1m6”)
  + *1\d\d\s{0,2}cm* – height strings that appear in centimetres with “cm” at the end (“174 cm”, “166cm”)
  + *1[\.,]\d{1,2}\s{0,2}m* – height strings that appear in metres with decimal notation and “m” at the end (“1.74m”, “1,66 m”)
  + *1\d\d[,\.]\d?\s{0,2}cm* – height strings that appear in centimetres with decimal notation and “cm” at the end (“174,5 cm”, “166,5cm”)
* **There are 14 cases that more than 1 instance are extracted, those are handled manually**
  1. **Predicate 2 (birthPlace)**
     1. **Methodology**
* Collect a dictionary of cities/provinces in Vietnam
* Use sets of indicators with hierarchy to locate the information
* Split the text into sentences and then extract the first one that contains any of the indicators
* Find the first occurrence of a city/province within a window around the indicator
  + 1. **Building the cities/provinces dictionary**
* Contains **104** cities, provinces (and their metropolis)
* If a province and its metropolis have the same name, keep the province only.
* Add some alternative names (Sài gòn, TP HCM)
  + 1. **Indicators**
* There are 3 levels of indicator sets
* Search each one with an appropriate window, if nothing is found then use the next level
* If none is found afterward, one might take the first city/province occurrence in the entire text. We decided not to do these as we believed no information is better than incorrect information



*Sets of indicators with corresponding windows*

* 1. **Predicate 10 (memberOf)**
     1. **Challenges**
* This we considered the hardest one since it involves heavy natural language processing
* Hard to extract temporal information
  + 1. **Our approach**
* List all band Wikipedia links which are extracted in predicate 12
* Parse those Wikipedia pages for list of members and their corresponding temporal information
* Handle manually missing cases
* Generate dummy artist entities for unknown band members
  1. **Handling duplicates**
     1. **Challenges**
* There are many artists with the same name from mp3.zing.vn
* Need to disambiguate them so the data can later be merge across sites
* The process cannot be done systematically
  + 1. **Methodology**
* Remove artists that have no song
* If 2 records refer to the same artists, remove the one that has less information (or less song)
* If 2 artists have same name but different real names, pick the less popular one (or have less song) and modify their record as follow: add “ == [real\_name]” to the field ‘name’ where [real\_name] is that artist real name.
* If 2 artists have same name, same real name, but come from different countries, pick the less popular one (or have less song) and modify their record as follow: add “ >> [country]” to the field ‘name’ where [country] is the country where that artist comes from.
* If 2 artists have same name, same real name, come from the same country, but have different major genres pick the less popular one (or have less song) and modify their record as follow: add “ => [genre]” to the field ‘name’ where [genre] is the major genre of that artist.
  + 1. **Notes**

During this process, many other information is found (height, dob). That information has been added manually into the kb object and should not be overwrote.

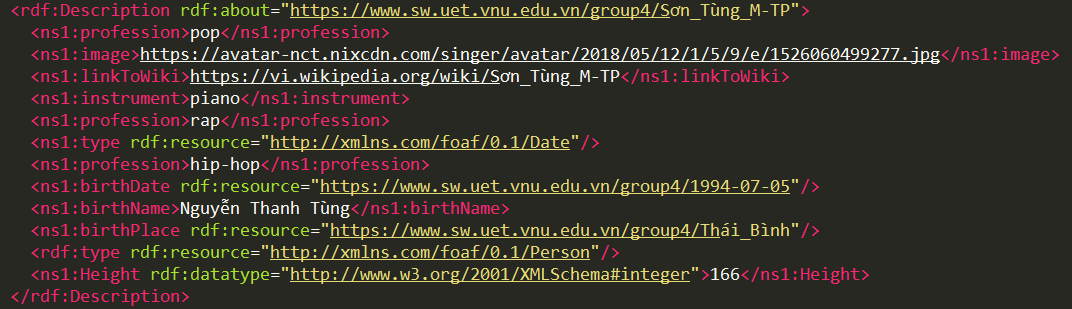
1. **Data Alignment**
   1. **Objective**

**Combines all KB files which collected from different websites and combine them to make a final KB**

* 1. **Merging process**
     1. **Merging artists information**
* **Merge mp3.zing.vn and nhacuatui.com**
  + Remove all non-Vietnamese artists
  + Change primary key to artist name
  + Manually skim through all records in the nhacuatui kb which are not in the mp3.zing.vn kb (abount **200** instances, practical do to manually)
  + Merge 2 kb by the primary key (artist name), if one of these information: birthName, birthDate, birthPlace is different, add them to an error list for later inspection
  + Add 2 new entry to the kb: “zing\_url” and “nct\_url”
  + The result is a kb object called artists.final.kb
* **Merge artists.final.kb with Wikipedia and chieu-cao.net**
  + Manually skim through all records in artists.final.kb which are not in the Wikipedia and chieu-cao.net kb (abount **50** instances, practical do to manually)
  + Merge 2 kb by the primary key (artist name), if Height or birthPlace is different, add them to an error list for later inspection
    1. **Merging songs information**
* Convert all performers (stored as URLs) to the corresponding name using artists.final.kb as the lookup table, remove those who are missing from the lookup table
* Remove all songs that have no performers
* Change primary key to song name
* Merge 2 kb by the primary key (song name), combine lists of composers and lists of performers

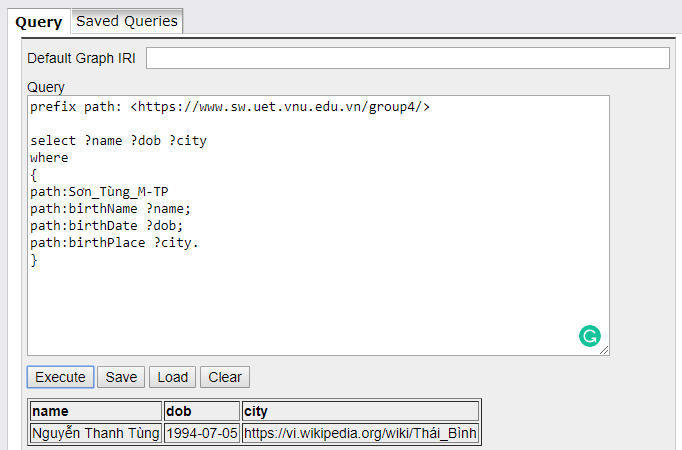
1. **Making SPARQL endpoint**
   1. **Export the RDF**

* Using the python rdflib library to generate a single xml file
* The namespace is <https://www.sw.uet.vnu.edu.vn/group4/>
* Artists, bands and songs are RDF resources, other entity are literals

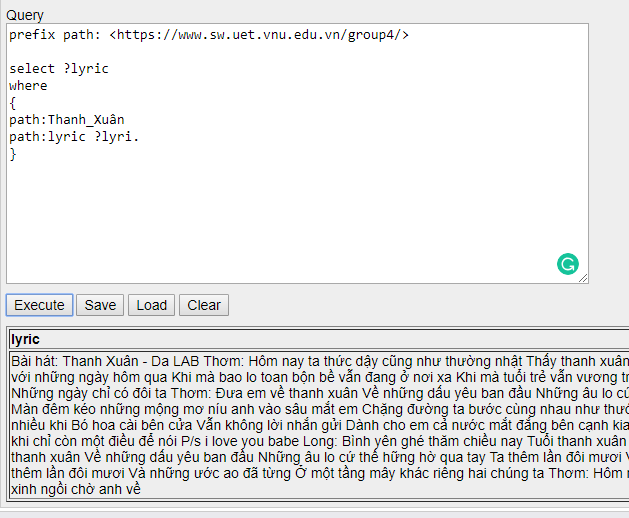


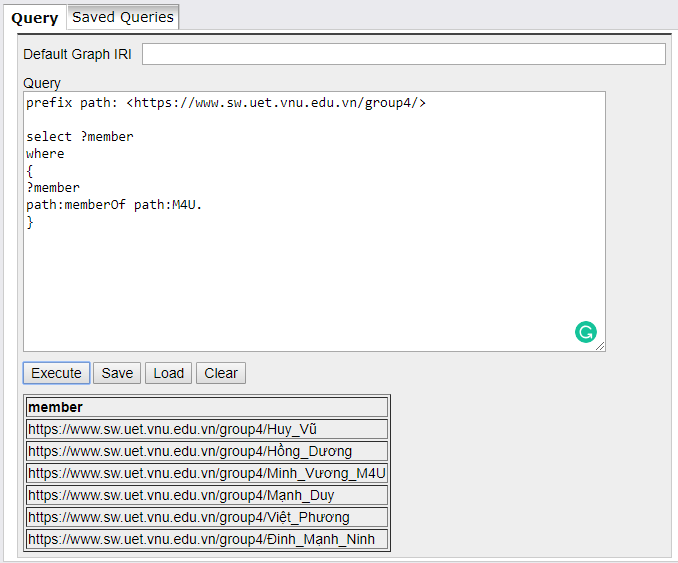
*Example of the RDF file content*

* 1. **Import to Virtuoso**
* The RDF file is imported through the Virtuoso web interface which can be found at <http://35.236.182.171:9001>
* An endpoint is generated as <https://www.sw.uet.vnu.edu.vn/group4/> and can be query using SPRQL
* Here are some query examples:

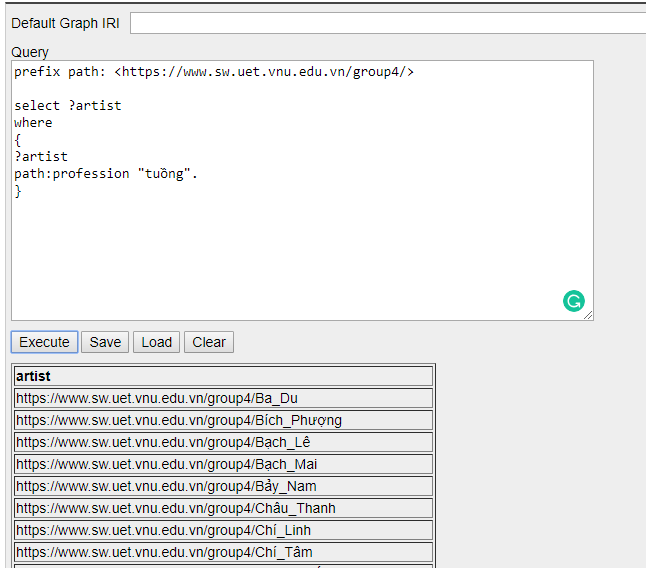


*Query information of Sơn Tùng M-TP*

*Query the lyric of the song called Thanh Xuân*



*Query all member of M4U*



*Query all tuồng artist*

1. **Summary**

* **The project took place during the last 6 weeks of the course**
* **Skills and knowledge acquired:**
  + Semantic web and RDF terminology
  + Information extraction method: regex, NLP, dictionary lookup
  + SPRQL
  + Collaboration and communication
* **Possible improvements:**
  + Advanced machine learning methods could be leveraged to improve information extraction accuracy
  + More domains could be used to fill missing information
  + Many manual processes can be done systematically