AN APPROACH TO AUTOMATICALLY UPDATE THE SPANISH DBPEDIA USING DBPEDIA DATABUS

Author: Oussama Tahiri Alaoui

Tutors: Óscar Corcho and Mariano Rico

Madrid, July 21, 2020
AN APPROACH TO AUTOMATICALLY UPDATE THE SPANISH DBPEDIA USING DBPEDIA DATABUS

Author: Oussama Tahiri Alaoui
Director: Óscar Corcho and Mariano Rico

Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

July 21, 2020
Dedication

In memory of my father, my grandmother and my grandfather (may God bless their souls), no dedication can be eloquent enough to express the depth of my feelings of affection and love I have for them. I will never forget their sincere prayers.

To my parents, for the education they have given me, with all the means and with whatever cost it takes, sacrifices they made for me and for the sense of duty they taught me since my childhood.

To my sisters: Zakia and Rayhana,

To my teachers,

To my friends,

To all those who are close to my heart.
Acknowledgements

First of all, I would like to express my gratitude to God for giving me the courage, the strength and the will to complete this modest project. The entire teaching staff of the Artificial Intelligent Department at the Higher Technical School of Computer Engineering, Universidad Politécnica de Madrid.

Pr. Óscar Corcho and my educational supervisor and my teacher for his kindness and the way he introduced me to the field of Open Data and Knowledge Graphs. Also, his energy and enthusiasm during our courses.

Pr. Mariano Rico my tutor for his technical supervision and monitoring throughout this project, his advice and our discussions that have marked this work, also his motivation and disponibility at all times.

I would like also to thank everyone who has helped me, from near or far.

And thanks to the creators of the Overleaf template used to write this report, licensed as MIT.
The Linked Data based project called DBpedia is publishing big amounts of data from Wikipedia semantically. Like Wikipedia, DBpedia also gives its datasets in different international languages. Spanish being one of these languages and the Spanish DBpedia as the second biggest DBpedia after the English one, collecting information from Wikipedia and serving it based on the user’s requirements. Aimed at keeping its data updated, the DBpedia foundation has created DBpedia Databus, an alternative to DBpedia Live and a way that simplifies working with data. This Master’s final project will be focused on an approach to automatically update the Spanish DBpedia, from DBpedia Databus using the Virtuoso OpenLink Software, and potentially the solutions provided, can help with any other DBpedia chapter to update it’s resources.

Keywords: DBpedia, DBpedia Databus, DBpedia Live, DBpedia foundation, Dataset, Linked Data, Software, Spanish DBpedia, Virtuoso, Wikipedia.
## Glossary of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS</td>
<td>Amazon Web Services</td>
<td>17</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
<td>27</td>
</tr>
<tr>
<td>DBW</td>
<td>DBpedia Wikidata</td>
<td>20</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
<td>47</td>
</tr>
<tr>
<td>IRI</td>
<td>Internationalized Resource Identifier</td>
<td>35</td>
</tr>
<tr>
<td>ISQL</td>
<td>Interactive Structured Query Language</td>
<td>32</td>
</tr>
<tr>
<td>KB</td>
<td>Knowledge Base</td>
<td>1</td>
</tr>
<tr>
<td>KVM</td>
<td>Kernel-based Virtual Machine</td>
<td>26</td>
</tr>
<tr>
<td>LOD</td>
<td>Linked Open Data</td>
<td>2, 5</td>
</tr>
<tr>
<td>OAI</td>
<td>Open Archives Initiative</td>
<td>16</td>
</tr>
<tr>
<td>OAI-PMH</td>
<td>Open Archives Initiative Protocol for Metadata Harvesting</td>
<td>13</td>
</tr>
<tr>
<td>OEG</td>
<td>Ontology Engineering Group</td>
<td>29</td>
</tr>
<tr>
<td>OKG</td>
<td>Open Knowledge Graph</td>
<td>1</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
<td>27, 30</td>
</tr>
<tr>
<td>OWL</td>
<td>Web Ontology Language</td>
<td>7, 16</td>
</tr>
<tr>
<td>RDB</td>
<td>Relational Data Base</td>
<td>16</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
<td>1, 7</td>
</tr>
<tr>
<td>SHACL</td>
<td>Shapes Constraint Language</td>
<td>18</td>
</tr>
<tr>
<td>SKOS</td>
<td>Simple Knowledge Organization System</td>
<td>7, 14</td>
</tr>
</tbody>
</table>
Acronyms

**SQL** Structured Query Language. 34

**TLS** Transport Layer Security. 19

**UPM** Universidad Politécnica de Madrid. 3, 26

**URI** Uniform Resource Identifier. 7

**VAD** Virtuoso Application Distribution. 34, 52

**VM** Virtual Machine. 30, 49

**VPN** Virtual Private Network. 26

**W3C** World Wide Web Consortium. 7
Contents

Dedication i
Acknowledgements ii
Abstract iii
Glossary of Acronyms v
List of Figures x
List of Tables xi
List of Listings xiii

1. Introduction 1
   1.1. Introducing Spanish DBpedia ........................................ 2
   1.2. Problem and solution ................................................... 2
   1.3. Introducing Databus .................................................... 2
   1.4. Project scope .......................................................... 3
   1.5. Document structure ...................................................... 4

2. State of The Art 5
   2.1. Linked Open Data .............................................................. 5
   2.2. Semantic Web Technologies ............................................... 7
   2.2.1. Resource Description Framework ..................................... 7
   2.2.2. RDF serialization formats ........................................... 8
   2.2.3. Ontology ................................................................. 9
   2.2.4. SPARQL Query Language for RDF .................................. 9
   2.2.5. Triplestore .............................................................. 10
   2.3. Introducing DBpedia ........................................................ 10
   2.3.1. The DBpedia Project ................................................... 10
   2.4. DBpedia Live ............................................................... 12
   2.4.1. DBpedia Live system architecture .................................. 13
   2.4.2. Architecture of the Extraction Framework .......................... 14
   2.4.3. DBpedia expension ....................................................... 16

An approach to automatically update the Spanish DBpedia using DBpedia Databus VI
## CONTENTS

2.5. The DBpedia Databus ........................................... 17
  2.5.1. Databus vision ............................................ 17
  2.5.2. Publishing data on the Databus .......................... 17
  2.5.3. How the Databus works at the moment ...................... 17
  2.5.4. Free service and licensing ................................ 18
  2.5.5. Current state and statistics ............................. 18
  2.5.6. Upcoming Developments ................................... 19

2.6. DBpedia and Wikidata .......................................... 20

2.7. Virtuoso ....................................................... 20

2.8. Current state of the Spanish DBpedia ......................... 21
  2.8.1. Current triples number .................................... 21
  2.8.2. Spanish DBpedia Quality .................................. 22
  2.8.3. Current Spanish DBpedia pages interface .................. 24

3. Design .......................................................... 26
  3.1. Development environment .................................... 26
    3.1.1. Virtual Machine Information ............................ 27
    3.1.2. Programming Language .................................. 27
  3.2. Real environment ............................................. 29

4. Development ..................................................... 30
  4.1. Installing and configuring Virtuoso .......................... 30
  4.2. Installing VAD packages ..................................... 34
  4.3. DBpedia interface and Spanish DBpedia data configuration .. 35
  4.4. Cron Job For Scheduled Script Execution .................... 36
  4.5. Updating The Spanish DBpedia ............................... 39
    4.5.1. Downloading the Spanish DBpedia data from the Databus .. 39
    4.5.2. Uncompressing the data files ........................... 40
    4.5.3. Checking the Virtuoso server status ........................ 40
    4.5.4. Bulk Loading the data ................................... 41
    4.5.5. Checking the update process .............................. 41
    4.5.6. SPARQL Query ............................................ 42
    4.5.7. Duration and analysis of the update process .............. 43
  4.6. Modifying The DBpedia Pages Interface ....................... 44

5. Results ........................................................ 49
  5.1. Checking the new loaded Spanish DBpedia data ................. 49
  5.2. Data refresh use case ....................................... 50
  5.3. The updating process log files ................................ 51

6. Conclusion ...................................................... 52

References .......................................................... 54
List of Figures

1.1. Results of the mapping hackathon, from source: Acknowledgement. .......... 4
2.2. The Linked Open Data Cloud in 2011, from source: lod-cloud.net. .......... 6
2.3. Informal graph of the sample triples, from source: RDF 1.1 Primer. .......... 8
2.4. Simple ontology example, from source: Ontology simplest definition. ....... 9
2.5. The Linked Open Data Cloud in 2007, from source: lod-cloud.net. .......... 11
2.6. The Linked Open Data Cloud in 2014, from source: lod-cloud.net. .......... 11
2.7. The Linked Open Data Cloud in 2020, from source: lod-cloud.net. .......... 12
2.8. DBpedia Live system architecture. ........................................ 13
2.9. Overview of DBpedia knowledge extraction framework, from source. ...... 16
2.10. Data and sources loaded on the bus (April 2018), from source. .......... 19
2.11. Querying the number of triples in the Spanish DBpedia endpoint. .......... 22
2.12. The number of triples in the Spanish DBpedia endpoint. ................ 22
2.13. Spanish DBpedia data quality estimation by Luzzu, from source: lod-
     cloud.net. ........................................................................... 23
2.14. The current Spanish DBpedia pages interface. ............................. 24
2.15. The new Spanish DBpedia pages interface. .................................... 25
3.1. The System Architecture ........................................................... 26
3.2. The folder of files needed for the Spanish DBpedia update. ............... 28
3.3. Example of the input to configure cron job each week ..................... 28
3.4. Redirecting the virtual machine Spanish DBpedia to the real environment . 29
4.1. The output after the auto generation command finishes as intended. ....... 31
4.2. The output after the auto generation command finishes as intended. ....... 32
4.3. Creating new Cron job. ............................................................. 38
4.4. The output after the auto generation command finishes as intended. ....... 38
4.5. Replace existing Cron job. ......................................................... 38
4.6. Virtuoso Web User Interface. ..................................................... 45
4.7. Virtuoso Conductor login. ......................................................... 45
4.8. Virtuoso Conductor main page. .................................................. 46
4.9. WebDAV browser. .................................................................... 46
4.10. DBpedia DAV content. .............................................................. 47
LIST OF FIGURES

4.11. The HTML code for the DBpedia logo. .................... 47
4.12. The HTML code for modifying the logos. ................... 48
4.13. Checking the modified logos. .......................... 48

5.1. Querying the triples number in real Spanish DBpedia SPARQL endpoint. 49
5.2. No results about novel corona virus before updating data. 50
5.3. Finding results about novel corona virus after updating data. 50
5.4. Log files Generated From The Update Process. ............. 51
List of Tables

2.1. Sample RDF statements. ...................................... 7
3.1. Virtual machine system information .......................... 27
4.1. Compression between normal and parallel performance. 44
Listings

2.1. RDF statement about Harrison Ford ......................................... 8
2.2. SPARQL query to get the spouse of Ford’s child. ...................... 9
2.3. The query for getting the number of triples. ........................... 21
3.1. Commands to successfully run a script file. ............................ 27
3.2. Commands to successfully run the script file for automating the Spanish
      DBpedia updating. ........................................... 28
4.1. Installing essential commands packages (Code in Github) .......... 30
4.2. Downloading Virtuoso Source Code (Code in Github). ............... 31
4.3. Specify compilation settings. ....................................... 31
4.4. Generate Configuration Files (Code in Github). ..................... 31
4.5. Configuring the installation path (Code in Github). ............... 32
4.6. Compilation and Installation (Code in Github) ....................... 32
4.7. Location of the database files (Code in Github). .................... 33
4.8. Location of the database temp files (Code in Github). ............ 33
4.9. Allocated memory (Code in Github). ................................ 33
4.10. Parameters to reduce memory leaking (Code in Github). ........... 33
4.11. Parameter to avoid memory error (Code in Github) ................. 34
4.12. Parameter to avoid memory error (Code in Github) ................. 34
4.13. Enabling the server log access for log files creation (Code in Github). 34
4.15. Installing Needed Packages (Code in Github) ......................... 35
4.16. Associating the registries for the Spanish DBpedia (Code in Github). 35
4.17. Downloading and Installing the DBpedia package (Code in Github). 35
4.18. Cron job creation code to retrieve month (Code in Github). ....... 36
4.19. Cron job creation code to retrieve weekday (Code in Github). .... 36
4.20. Constructing the Cron job command (Code in Github) ............. 37
4.21. Checking if the Cron job already exists (Code in Github) ........ 37
4.22. Executing the Cron job command (Code in Github) ............... 38
4.23. Querying the download URLs and Download the Spanish Data (Code in
      Github). .................................................... 39
4.24. Uncompressing the bzip2 files (Code in Github). .................. 40
4.25. Checking the Virtuoso server status and running in case it’s down (Code in
      Github). .................................................... 40
4.26. Clearing the graph and loading new data (Code in Github) ........ 41
<table>
<thead>
<tr>
<th>Listing</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.27</td>
<td>Creating a checkpoint and querying the data (Code in Github)</td>
<td>42</td>
</tr>
<tr>
<td>4.28</td>
<td>Extract from The SPARQL Query to get Spanish data Downloadurls (Code in Github)</td>
<td>42</td>
</tr>
<tr>
<td>4.29</td>
<td>Parallelizing downloads with wget (Code in Github)</td>
<td>43</td>
</tr>
<tr>
<td>4.30</td>
<td>Parallelizing Bzip2 files unzipping with Parallel command (Code in Github)</td>
<td>43</td>
</tr>
<tr>
<td>4.31</td>
<td>Running multiple Bulk Loaders (Code in Github)</td>
<td>44</td>
</tr>
<tr>
<td>4.32</td>
<td>Calculate and write the execution time in minutes (Code in Github)</td>
<td>44</td>
</tr>
<tr>
<td>4.33</td>
<td>Default logo HTML code (Code in Github)</td>
<td>47</td>
</tr>
<tr>
<td>4.34</td>
<td>HTML code for putting Spanish DBpedia and UPM logos (Code in Github)</td>
<td>47</td>
</tr>
<tr>
<td>A.1</td>
<td>Extract from the 2020.07.10-04.37.01_esdbpedia-update.log file</td>
<td>57</td>
</tr>
<tr>
<td>A.2</td>
<td>Extract from the 2020.07.17-23.58.01_esdbpedia-update.log file</td>
<td>58</td>
</tr>
<tr>
<td>A.3</td>
<td>Extract from the 2020.07.19-00.38.01_esdbpedia-update.log file</td>
<td>60</td>
</tr>
</tbody>
</table>
Introduction

DBpedia [1] is a crowd-sourced community effort to extract structured content from the information created in various Wikimedia projects \(^1\). This structured information resembles an Open Knowledge Graph (OKG) which is available for everyone on the Web. A knowledge graph is a special kind of database which stores knowledge in a machine-readable form and provides means for information to be collected, organised, shared, searched and utilised. Google uses a similar approach to create those knowledge graphs during search. The DBpedia community project extracts structured, multilingual knowledge from Wikipedia [2] and makes it freely available on the Web using Semantic Web [3] and Linked Data [4] technologies. The project extracts knowledge from 111 different language editions of Wikipedia. The largest DBpedia Knowledge Base (KB) which is extracted from the English edition of Wikipedia consists of over 400 million facts that describe 3.7 million things. The DBpedia knowledge bases that are extracted from the other 110 Wikipedia editions together consist of 1.46 billion facts and describe 10 million additional things. The DBpedia project maps Wikipedia infoboxes [5] from 27 different language editions to a single shared ontology consisting of 320 classes and 1,650 properties. The mappings are created via a world-wide crowd-sourcing effort and enable knowledge from the different Wikipedia editions to be combined. The project publishes regular releases of all DBpedia knowledge bases for download and provides SPARQL [6] query access to 14 out of the 111 language editions via a global network of local DBpedia chapters. In addition to the regular releases, the project maintains a live knowledge base which is updated whenever a page in Wikipedia changes. DBpedia sets 27 million Resource Description Framework (RDF) [7] links pointing into over 30 external data sources and thus enables data from these sources to be used together with DBpedia data. Several

\(^1\)https://wikimediafoundation.org/our-work/wikimedia-projects/
hundred data sets on the Web publish RDF links pointing to DBpedia themselves and thus make DBpedia one of the central interlinking hubs in the Linked Open Data (LOD) cloud.

1.1. Introducing Spanish DBpedia

For the Spanish language, the Spanish DBpedia was created in 2012 and currently there are 15 other local DBpedia chapters that were set up in different countries as part of the DBpedia internationalization effort. A 40% of the pages in the Spanish Wikipedia do not have links to the English Wikipedia, which means they are non-canonicalized datasets; therefore, a significant percentage of the information stored in the Spanish DBpedia is not available in the English DBpedia. This fact places the Spanish DBpedia as a valuable and exclusive source of local information. Also, we have to remark that the English DBpedia does not contain all the information stored in the local DBpedias, but only a minimum subset that contain labels and abstracts, specifically.

1.2. Problem and solution

The work presented in this report is about automatizing the process of updating the Spanish DBpedia data, reducing the updating time and making it more frequent. At the moments of writing this report, the Spanish DBpedia doesn’t contain all the available Spanish Data tuples, and the actual update period is about a month which is low, and doesn’t keep the Spanish DBpedia up to date. So the solution that we are presenting in this work is a process to extract all the Spanish data available in the databus and update the Spanish DBpedia each week and in a short period of time possible. To do so we used the Virtuoso OpenLink Software, and developed a bash script that successfully executes this task in a local DBpedia inside a virtual machine, and then move this solution to the real Spanish DBpedia.

1.3. Introducing Databus

Getting the Spanish Data is possible using the DBpedia Databus project, currently in public beta, which aims at providing an end-to-end pipeline easing auto-extraction, metadata-generation and publishing of Linked Data Knowledge Graphs at scale. The Databus platform provides two tools to connect consumers and producers, one for consumers, which is the website and the SPARQL API serve as a user interface to configure data

---

2http://es.dbpedia.org/
3https://wiki.dbpedia.org/Internationalization
4https://www.gnu.org/software/bash/
CHAPTER 1. INTRODUCTION

set retrieval and combination in catalogues, and the other for providers, the Databus Maven plugin 5 enables systematic upload and release of datasets on the bus.

The idea behind using Databus to get our data, is the fact that it’s able to generate metadata about datasets and then upload this metadata, so that anybody can query, download, derive and build applications with this data via the Databus.

1.4. Project scope

The DBpedia project has been generating semantic information from the English Wikipedia. From June 2011 the extraction process extracts information from Wikipedia in 15 of their non English languages. One of these is the Spanish language. The DBpedia internationalization committee has assigned a web site and a SPARQL endpoint [10] to each one of these languages.

The extraction process for the Spanish language has produced so far almost 100 million RDF triples. In the SPARQL endpoint the most relevant triples (70 millions) are available. This work depends on researchers from Universidad Politécnica de Madrid (UPM): Mariano Rico and Oscar Corcho, both of them belonging to the Spanish Thematic Network of Linked Data 6, as well as individuals who dedicate their time and effort to this initiative. This initiative started on a mapping generation hackathon that was celebrated under the auspices of the aforementioned network.

This huge amount of high quality information would not be possible without the effort, specifically, of ‘mappers’, people who manually set the semantic relationships between the DBpedia ontology terms and the infoboxes in the Spanish Wikipedia.

These mappings would not be so productive without the logistical and financial support of the Spanish Thematic Network of Linked Data, which organized in November 2011 a mappings hackaton attended for two days by 15 mappers. The effort paid off, as seen in Figure 1, as it managed to advance 12 positions (from 15 + English) in the ranking of semantic information generated from the information stored in the Spanish Wikipedia infoboxes.

As seen in figure 1.1 where "es" stands for Spanish, currently 80% of the Spanish Wikipedia infoboxes are mapped and, therefore, their information is available as Linked Data.

5http://dev.dbpedia.org/Databus_Derive_Maven_Integration
6http://red.linkeddata.es/
1.5. Document structure

To realize this objective, the rest of this report is structured as follows: chapter 2 describes the state of the art, chapter 3 gives the design used and provides the used methods. The chapter 4 is a detailed description of the script used and the configuration to run it. The chapter 5 gives the results from the testing environment, and than the steps used to move the code into practice in the real environment. Finally concluding with insights on the provided solution and possible improvements.
In this chapter we are going to be introducing LOD, DBpedia and Databus projects in more detail and their structure. Also we’re going to state the problem and the used solution in more.

2.1. Linked Open Data

LOD is becoming increasingly important in the fields of state-of-the-art information and data management. It is already being used by many well-known organisations, products and services to create portals, platforms, internet-based services and applications. Despite that, we are still in a situation today where many organizations do not understand the idea of publishing data on the web, let alone why data on the web should be linked. The evolution of the web can be seen as follows:
The Linked Data paradigm has evolved as a powerful enabler for the transition of the current document-oriented Web into a Web of interlinked Data and, ultimately, into the Semantic Web. The term Linked Data here refers to a set of best practices for publishing and connecting structured data on the Web. These best practices have been adopted by an increasing number of data providers over the past three years, leading to the creation of a global data space that contains many billions of assertions – the Web of Linked Data [4]. The so called LOD cloud covers more than an estimated 50 billion facts from many different domains like geography, media, biology, chemistry, economy, energy, etc. The data is of varying quality and most of it can also be re-used for commercial purposes. Below is a version of the LOD Cloud diagram of 2011:

Figure 2.1: Evolution of the Web Data, from source: [11, p. 23].

Figure 2.2: The Linked Open Data Cloud in 2011, from source: lod-cloud.net.
All of the different ways to publish information on the web are based on the idea that there is an audience out there that will make use of the published information, even if we are not sure who exactly it is and how they will use it.

2.2. Semantic Web Technologies

There are quite a few different definitions of the concept Semantic Web. Tim Berners-Lee, one of the inventors of the Semantic Web idea, defines it as "The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation" (Berners-Lee et al., 2001). According to the World Wide Web Consortium (W3C) the concept can be defined as "Semantic Web is the idea of having data on the Web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration, and reuse of data across various applications." (W3C Semantic Web Activity, 2009). In order to achieve this vision several technologies were developed and standardized. These include the RDF, Uniform Resource Identifiers (URIs) as well as vocabulary, schema and ontology [12] languages built on top of them such as RDF-Schema, Web Ontology Language (OWL) and Simple Knowledge Organization System (SKOS). More details about Semantic Web Technologies can be found in (Yu, 2007) [13].

2.2.1. Resource Description Framework

RDF is a knowledge representation language for describing arbitrary resources such as Web resources, entities and general concepts. RDF is the main building block of the Semantic Web, and it is for the Semantic Web what HTML is for the conventional Web. RDF has several basic elements, namely resources (also called entities), properties (also predicates or roles), and statements (also triples or facts). Resources comprise any concrete or abstract entities, e.g. a book or a person. Every resource is identified by a URI which is guaranteed to be globally unique.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbpedia:Harrison_Ford</td>
<td>dbp:dateOfBirth</td>
<td>&quot;July 13, 1942&quot;</td>
</tr>
<tr>
<td>dbpedia:Harrison_Ford</td>
<td>dbp:occupation</td>
<td>&quot;Actor, aviator, environmental, activist&quot;</td>
</tr>
</tbody>
</table>

Table 2.1: Sample RDF statements.

For example http://dbpedia.org/resource/Harrison_Ford. Properties are specific resources which can be used to describe attributes or relations of other resources. For instance, the property http://dbpedia.org/ontology/Person/height identifies the height of a human being. Statements are used to describe properties of resources in the format subject, predicate, object. According to the RDF data model all three components of
a triple can be resources (i.e. URIs), subject and object can also be blank nodes (i.e. unnamed entities) and objects can also be literals consisting of a data value and optionally an associated a datatype or language tag. For example:

**Code 2.1: RDF statement about Harrison Ford**

```plaintext
1  <http://dbpedia.org/resource/Harrison_Ford>
2  <http://dbpedia.org/property/birthPlace>
```

This RDF statement simply says "The subject identified by http://dbpedia.org/resource/Harrison_Ford has the property identified by http://dbpedia.org/property/birthPlace, whose value is equal to "Chicago, Illinois, U.S". Table 2.1 shows more RDF statements about Harrison Ford. The URIs are abbreviated with namespace prefixes.

### 2.2.2. RDF serialization formats

There are several formats for serializing RDF data. A simple text format is the N-Triples format (Grant and Beckett, 2004). Each RDF triple is written in a separate line and terminated by a period ".". Typically files with N-Triples have the ".nt" extension. Figure 2.3 shows some sample triples encoded in N-Triples format. There are also a number of other RDF serialization formats such as the XML serialization RDF/XML, the text serialization formats N3 and Turtle, which are very similar to the N-Triples format and at last the RDFa serialisation, which allows to integrate RDF into HTML.

![Informe de un ejemplo de los triples](image)

**Figure 2.3: Informal graph of the sample triples, from source: RDF 1.1 Primer.**
2.2.3. Ontology

The W3C defines an ontology as follows: "An ontology defines the terms used to describe and represent an area of knowledge." 1. Ontologies are used by people, databases, and applications that need to share domain information. A domain can be a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc.

Figure 2.4: Simple ontology example, from source: Ontology simplest definition.

In the ontology Figure 2.4, there is one entity sub-type, or category of object, which is a person. The person can take one of two different roles in a parentship relation with another person entity: a parent or child role. The parentship relation has a single property associated with it (a date of birth), while the person entity has a number of properties, such as names, age and gender.

2.2.4. SPARQL Query Language for RDF

SPARQL stands for SPARQL Protocol and RDF Query Language. It is used to ask queries against RDF graphs. A SPARQL processor finds sets of triples in the RDF graph that match to the required pattern [14]. The results of SPARQL queries can be result sets or RDF graphs. For instance, an example for the query "Who is the spouse of Harrison Ford’s child?” is shown in Figure 2.2.

Code 2.2: SPARQL query to get the spouse of Ford’s child.

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

SELECT ?spouse WHERE {
}
```

1https://www.w3.org/2003/08/owlfaq
2.2.5. Triplestore

A triplestore is a software program capable of storing and indexing RDF data, in order to enable querying this data efficiently. Most triplestores support the SPARQL query language for querying RDF data. Virtuoso (Erling and Mikhailov, 2007), Sesame (Broekstra et al., 2002), and BigOWLIM (Bishop et al., 2011) are typical examples of triplestores. DBpedia is using Virtuoso as the underlying triplestore, and for this work we have used virtuoso as well.

2.3. Introducing DBpedia

At the time of writing this report, Wikipedia is the 13th most popular website\(^2\), the most widely used encyclopedia, and one of the finest examples of truly collaboratively created content. As of June 2020, Wikipedia articles have been created in 310 languages, with 300 official Wikipedia additions which range in size from a couple of hundred articles up to 6 million articles (English edition)\(^3\). The DBpedia community project extracts structured, multilingual knowledge from Wikipedia and makes it freely available on the Web using Semantic Web and Linked Data technologies\(^4\).

2.3.1. The DBpedia Project

The DBpedia project\(^1\) builds a large scale, multilingual knowledge base by extracting structured data from Wikipedia editions in 125 languages. This knowledge base can be used to answer expressive queries such as finding all rivers that flow into the Rhine and are longer than 100 miles. Being multilingual and covering a wide range of topics, the DBpedia knowledge base is also useful within further application domains such as data integration, named entity recognition, topic detection, and document ranking. The DBpedia knowledge base is also widely used as a test environment in the research community and numerous applications, algorithms and tools have been built around or applied to DBpedia. DBpedia is Wikipedia content represented as a Semantic Web\(^2\) of Linked Data\(^16\). Since it covers a wide variety of topics and sets RDF links pointing into various external data sources, many Linked Data publishers have decided to set RDF links pointing to DBpedia from their data sets. Thus, DBpedia has developed into a central interlinking\(^17\) hub in the Web of Linked Data and has been a key factor for the success of the Linked Open Data\(^18\) initiative\(^4\).

The diagrams figure 2.5, figure 2.6 and figure 2.7 give an overview of how the DBpedia data set is interlinked with increased number of other data sources:

\(^2\)https://www.alexa.com/topsites
\(^3\)https://meta.wikimedia.org/wiki/List_of_Wikipedias
\(^4\)https://lod-cloud.net/
 CHAPTER 2. STATE OF THE ART

Figure 2.5: The Linked Open Data Cloud in 2007, from source: lod-cloud.net.

Figure 2.6: The Linked Open Data Cloud in 2014, from source: lod-cloud.net.
As we can see in the cloud diagram, some of the data providers have become well-known and established as popular linking hubs in the web of data. Prominent examples are DBpedia, a community effort to extract structured information from Wikipedia and geonames that provides RDF descriptions of millions of geographical locations worldwide.

The original DBpedia representation was generated from a static dump of Wikipedia content, in a process that took roughly 6 months from Wikipedia dump to DBpedia publication. To update DBpedia, new Wikipedia dumps have been taken periodically (roughly every 6-12 months) since then and processed in the same way. DBpedia content has thus always been 6-18 months behind updates applied to Wikipedia content.

2.4. DBpedia Live

As the use of DBpedia, and the dynamism of Wikipedia content, have increased, the need for DBpedia to update constantly by processing the Wikipedia "firehose" changelog became apparent. DBpedia Live is the current fruit of that effort. DBpedia Live retrieves all edits on Wikipedia immediately, extracts all information and transfers them into an
CHAPTER 2. STATE OF THE ART

online SPARQL database for querying. The extraction framework [19] can handle up to 1 million edits per day on an average server and in addition the Virtuoso [20] Database handles the loading for online querying.

2.4.1. DBpedia Live system architecture

The main components of the DBpedia-Live system are as follows:

- Local Wikipedia: which is an installed local copy of Wikipedia that is kept in synchronization with Wikipedia. The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) enables an application to get a continuous stream of updates from a wiki. OAI-PMH is also used to feed updates into the DBpedia-Live Extraction Manager.

- MappingWiki: DBpedia mappings can be found at http://mappings.dbpedia.org, which is itself a wiki. OAI-PMH is used to get a stream of updates in DBpedia mappings. Basically, a change of mapping affects several Wikipedia pages, which should all be reprocessed.

- DBpedia Live Extraction Manager: This component is the actual DBpedia-Live extraction framework. When there is a page that should be processed, the framework applies the extractors to it. After processing a page, the newly extracted RDF statements are inserted into the backend data store (the Quad Store functionality of the Virtuoso Universal Server), where they replace the old RDF statements. The newly extracted RDF is also written to a compressed N-Triples file. Mirrors of DBpedia-Live, as well as other applications that should always be in synchronization with the DBpedia-Live endpoint, can download those change set files and feed them into their own RDF data stores [21].

The new Java-based live-extraction framework is deployed on a server hosted by OpenLink Software. It has a SPARQL endpoint, also operated by OpenLink Software,

In addition to the migration to Java, the new DBpedia Live framework has the following features:

- **Abstract extraction:** The abstract of a Wikipedia article contains the first few paragraphs of that article. The new framework has the ability to cleanly extract the abstract of an article.

- **Mapping-affected pages:** Upon a change in mapping, the pages affected by that mapping should be reprocessed and their RDF descriptions should be updated to reflect that change.

- **Updating unmodified pages:** Sometimes a change in the system occurs, e.g. a change in the implementation of an extractor. This change can affect many pages even if they are not modified. In DBpedia-Live, we use a low-priority queue for such changes, such that the updates will eventually appear in DBpedia-Live, but recent Wikipedia updates are processed first.

- **Publication of change sets:** Upon modification, old RDF statements are replaced with updated statements. The added and/or deleted statements are also written to N-Triples files and then compressed. Any client application or DBpedia-Live mirror can download the files and integrate (and, hence, update) a local copy of DBpedia. This enables that application to stay in synchronization with our version of DBpedia-Live.

- **Development of synchronization tool:** The synchronization tool enables a DBpedia-Live mirror to stay in synchronization with our live endpoint. It downloads the changeset files sequentially, decompresses them, and integrates them with another DBpedia-Live mirror. In addition to the infobox extraction process, the framework has currently 19 extractors which process the following types of Wikipedia content: Labels, Abstracts, Interlanguage links, Images, Redirects, Disambiguation, External links, Page links, Homepages, Geo-coordinates, Person data, PND, SKOS categories, Page ID, Revision ID, Category label, Article categories, Mappings and Infobox.

If the DBpedia-live extraction stops, it will be resumed at the last point at which it stopped. According to statistics, roughly 1.4 Wikipedia articles are modified per second, which works out to about 84 articles per minute. With current resources, DBpedia-Live can process about 105 pages per minute, on average. Which means that the speed of processing of DBpedia-Live cope with the speed of data-stream5.

### 2.4.2. Architecture of the Extraction Framework

While Wikipedia articles consist mostly of free text, they also contain various types of structured information, such as infobox templates, categorisation information, images,

---

5https://wiki.dbpedia.org/online-access/DBpediaLive
geo coordinates, links to external Web pages and other Wikipedia articles, disambiguation information, redirects and crosslanguage links. The DBpedia extraction framework extracts these different kinds of information and turns them into RDF data. All entities in DBpedia are assigned a unique URI of the form http://dbpedia.org/resource/Name, where Name is taken from the URL of the source Wikipedia article, which has the form http://en.wikipedia.org/wiki/Name.

The type of wiki contents that are most valuable for the DBpedia extraction are Wikipedia infoboxes. Infoboxes contain attribute value pairs and are used to display an article’s most relevant facts as a table at the top right-hand side of the corresponding Wikipedia page. Wikipedia’s infobox template system has evolved over time without central coordination. Therefore, there is a lack of uniformity of infoboxes. Different templates use different names for the same attribute (e.g. birthplace and placeofbirth). While the first version of our infobox extractor used a generic method to turn property value pairs into triples and hence struggled with the different names of attributes, our new mapping-based extractor aims to solve that problem by introducing a central DBpedia ontology and mappings between templates and the ontology.

This ontology was created by manually arranging the 350 most commonly used infobox templates within the English edition of Wikipedia into a subsumption hierarchy consisting of 170 classes and then mapping 2300 attributes from within these templates to 720 ontology properties. The property mappings define fine-grained rules on how to parse infobox values and define target datatypes, which help the parsers to process values.

The figure 2.9 gives an overview of the DBpedia knowledge extraction framework main components [22]. The main components of the framework are: PageCollections which are an abstraction of local or remote sources of Wikipedia articles, Destinations that store or serialize extracted RDF triples, Extractors which turn a specific type of wiki markup into triples, Parsers which support the extractors by determining datatypes, conversion values between different units and splitting markup into lists. ExtractionJobs group a page collection, extractions and a destination into a workflow. The core of the framework is the Extraction Manager which manages the process of passing Wikipedia articles to the extractors and delivers their output to the destination.
CHAPTER 2. STATE OF THE ART

The DBpedia live extraction framework extracts RDF data from articles in the English Wikipedia1 with a delay of only a few seconds after they are edited. In brief, the extraction proceeds as follows: When an article is modified, the live extraction framework polls its latest revision via Wikipedia’s non public Open Archives Initiative (OAI) Repository. After that, a set of extractors are run which generate RDF output for this Wikipedia page. Finally, this data is written into a publicly accessible triple store where it can be accessed via Linked Data and SPARQL [23].

There are different approaches for managing triples in the context of the DBpedia live extraction: The first based on OWL [13] 2 annotations, the second using domain specific queries, the third using individual graphs and the fourth being assisted by an Relational Data Base (RDB) [23].

2.4.3. DBpedia expension

Nowadays, only 1.7M Wikipedia pages are deeply classified in the DBpedia ontology, although the English Wikipedia contains almost 4M pages, showing a clear problem of coverage. In other languages like Spanish this coverage is even lower. To increase the coverage of DBpedia in different languages, the major problems that was solved concern the high number of classes involved in the DBpedia ontology and the lack of coverage for some classes in certain languages. The solution was by extend the population of the classes for the different languages by connecting the corresponding Wikipedia pages through cross-language links. Then, training a supervised classifier using this extended set as training data. This approach can add more than 1M new entities to DBpedia.

Figure 2.9: Overview of DBpedia knowledge extraction framework, from source.

An approach to automatically update the Spanish DBpedia using DBpedia Databus
with high precision (90%) and recall (50%). The resulting resource is available through a SPARQL endpoint and a downloadable package [24].

2.5. The DBpedia Databus

The DBpedia Databus [25] is a platform to capture invested effort by data consumers who needed better data quality in order to use the data and give improvements back to the data source and other consumers. DBpedia Databus enables us to build an automated DBpedia-style extraction, mapping and testing for any data that we need ⁶.

2.5.1. Databus vision

It’s an effort that allows exchanging, curating and accessing data between multiple stakeholders. Any data entering the bus will be versioned, cleaned, mapped, linked and its licenses and provenance tracked. Hosting in multiple formats will be provided to access the data either as dump download or as API ⁷.

2.5.2. Publishing data on the Databus

Databus is used to publish data on the web as data loaded on the bus will be highly visible, available and queryable. We should think of it as a service that offers, besides visibility and web download, a Linked Data interface, SPARQL-endpoint, Lookup with autocomplete or other means of availability (like Amazon Web Services (AWS) or Docker images). Any distribution that will be available will funnel feedback and collaboration opportunities to improve datasets and internal data quality. As well as receiving an enriched dataset, which is connected and complemented with any other available data.

2.5.3. How the Databus works at the moment

Since the integration of data is easy with the Databus, many additional datasets have been integrated and loaded alongside DBpedia for the world to query. Popular datasets are ICD10 (medical data) and organizations and people. The databus is still in an initial state, but it has already loaded 10 datasets (6 from DBpedia, 4 external) on the bus using these phases:

- Acquisition: data is downloaded from the source and logged in.

⁶https://wiki.dbpedia.org/blog/one-billion-derived-knowledge-graphs
CHAPTER 2. STATE OF THE ART

- Conversion: data is converted to N-Triples and cleaned (Syntax parsing, datatype validation, and Shapes Constraint Language (SHACL)).

- Mapping: the vocabulary is mapped on the DBpedia Ontology and converted (this was being done for Wikipedia’s Infoboxes and Wikidata [26], but now it is done for other datasets as well).

- Linking: Links are mainly collected from the sources, cleaned and enriched.

- IDying: All entities found are given a new Databus ID for tracking.

- Clustering: ID’s are merged onto clusters using one of the Databus ID’s as cluster representative.

- Data Comparison: Each dataset is compared with all other datasets. We have an algorithm that decides on the best value, but the main goal here is transparency, i.e. to see which data value was chosen and how it compares to the other sources.

- A main knowledge graph fused from all the sources, i.e. a transparent aggregate.

- For each source, a local fused version called the “Databus Complement” is being produced. This is a major feedback mechanism for all data providers, where they can see what data they are missing, what data differs in other sources and what links are available for their IDs.

- The possibility to compare all data via a web service.

2.5.4. Free service and licensing

The DBpedia Association is providing everything that they can afford for free. In the past it was providing everything for free, but it wasn’t a healthy model, as DBpedia can’t neither maintain quality properly nor grow. Also, maintaining the Databus is a lot of work and servers incurring is high cost.

On the Databus everything is provided “As is” without any guarantees or warranty and improvements can be done by the volunteer community. Final databases are licensed under ODC-By. This covers our work on recomposition of data. Each fact is individually licensed, e.g. Wikipedia abstracts are CC-BY-SA, some are CC-BY-NC, some are copyrighted. This means that data is available for research, informational and educational purposes. For any professional use of the data, it’s recommended to contact the providers so they can guarantee that legal matters are handled correctly. Otherwise, professional use is at own risk.

2.5.5. Current state and statistics

The Databus data is available at http://downloads.dbpedia.org/databus/ ordered into three main folders (To note that the file and folder structure are still subject to change):

An approach to automatically update the Spanish DBpedia using DBpedia Databus
CHAPTER 2. STATE OF THE ART

- Data: the data that is loaded on the Databus at the moment.
- Global: a folder that contains provenance data and the mappings to the new IDs.
- Fusion: the output of the Databus.

2.5.6. Upcoming Developments

DBpedia and the Databus future goal is to will Linked Data into a networked dataeconomy, to do so a lot of upcoming development need to take place, and some of the are:

- Include more existing data from DBpedia.
- Renew all DBpedia releases in a separate fashion.
- Load all data in the comparison tool.
- Load all data into a SPARQL endpoint.
- Create a simple open source software that let’s everybody push data on the Databus in an automated way.
- The ability to build your own data inventory and merchandise your data via Linked Data or via secure named graphs in the DBpedia SPARQL Endpoint (WebID + Transport Layer Security (TLS) + OpenLink’s Virtuoso database).
- The possibility to offer your Linked Data tools, services, products.
- Incubate new research into products.

Figure 2.10: Data and sources loaded on the bus (April 2018), from source.
2.6. DBpedia and Wikidata

DBpedia is one of the first and most prominent nodes of the Linked Open Data cloud. It provides structured data for more than 100 Wikipedia language editions as well as Wikimedia Commons, has a mature ontology and a stable and thorough Linked Data publishing life cycle. Wikidata has recently emerged as a user curated source for structured information which is included in Wikipedia.

In this section we compare between DBpedia and Wikidata, and we describe the integration of Wikidata into the DBpedia Data Stack. People are not used to the currently very evolving Wikidata schema, while DBpedia has a relatively stable and commonly used ontology. As a result, with the DBpedia Wikidata (DBW) dataset can be queried with the same queries that are used with DBpedia [27].

DBpedia uses human-readable Wikipedia article identifiers to create IRIs for concepts in each Wikipedia language edition and uses RDF and Named Graphs as its original data model. Wikidata on the other hand uses language independent numeric identifiers and developed its own data model, which provides better means for capturing provenance information. The multilingual DBpedia ontology, organizes the extracted data and integrates the different language editions while Wikidata is rather schemaless, providing only simple templates and attribute recommendations. All DBpedia data is extracted from Wikipedia and Wikipedia authors thus unconsciously also curate the DBpedia knowledge base. Wikidata on the other hand has its own data curation interface, which is also based on the MediaWiki framework 8. DBpedia publishes a number of datasets for each language edition in a number of Linked Data ways, including datasets dumps, dereferenceable resources and SPARQL endpoints. While DBpedia covers a very large share of Wikipedia at the expense of partially reduced quality, Wikidata covers a significantly smaller share, but due to the manual curation with higher quality and provenance information. As a result of this complementarity, aligning both efforts in a loosely coupled way would render a number of benefits for users. Wikidata would be better integrated into the network of Linked Open Datasets and Linked Data aware users had a coherent way to access Wikidata and DBpedia data. Applications and use cases have more options for choosing the right balance between coverage and quality [28].

2.7. Virtuoso

OpenLink Virtuoso was first developed as a row-wise transaction oriented RDBMS with SQL federation. It was subsequently re-targeted as an RDF graph store with built-in SPARQL and inference. The largest Virtuoso applications are in the RDF space, with terabytes of RDF triples that usually do not fit in RAM. The excellent space efficiency of column-wise compression was the greatest incentive for the column store transition. Additionally, this makes Virtuoso an option for relational analytics also. Finally, combining a schema-less data model with analytics performance is attractive

8https://www.mediawiki.org/wiki/MediaWiki
for data integration in places with high schema volatility. Virtuoso has a shared nothing cluster capability for scale-out. This is mostly used for large RDF deployments. The cluster capability is largely independent of the column-store aspect but is mentioned here because this has influenced some of the column store design choices [20]. Experience with Virtuoso has encountered most of the known issues of RDF storage and has shown that without overwhelmingly large modifications, a relational engine can be molded to efficiently support RDF. This has also resulted in generic features which benefit the relational side of the product as well. The reason is that RDF makes relatively greater demands on a DBMS than relational applications dealing with the same data [9].

2.8. Current state of the Spanish DBpedia

The DBpedia project [1] has been generating semantic information from the English Wikipedia in many languages, one of them is the Spanish language. The DBpedia internationalization committee has assigned a web site and a SPARQL endpoint to each one of these languages including the Spanish language. For the Spanish language, the Spanish DBpedia\(^9\) (esDBpedia) was created in 2012. The Spanish Wikipedia do not have links to the English Wikipedia, therefore, a significant percentage of the information stored in the Spanish DBpedia is not available in the English DBpedia. This makes the Spanish DBpedia a valuable and exclusive source of local information [29].

2.8.1. Current triples number

The extraction process for the Spanish language has produced so far almost 100 million RDF triples. Using the SPARQL endpoint we can check the most relevant triples number that is available using the following SPARQL request:

Code 2.3: The query for getting the number of triples.

```sql
SELECT (count(*) AS ?Triples)
FROM <http://es.dbpedia.org>
WHERE
{ ?s ?p ?o }
```

\(^9\)http://es.dbpedia.org/
2.8.2. Spanish DBpedia Quality

data extracted from semi-structured sources, such as DBpedia, often contains inconsistencies as well as misrepresented and incomplete information. However, in the case of DBpedia, the data quality is perfectly sufficient for enriching Web search with facts or suggestions about topics of general interest, such as entertainment topics. For developing a medical application, on the other hand, the quality of DBpedia is probably insufficient, since data is extracted from a semi-structured source created in a crowd sourcing effort (i.e. Wikipedia). It should be noted that even the traditional document-oriented Web has content of varying quality but is still commonly perceived to be extremely useful. Consequently, a key challenge is to determine the quality of datasets published on the Web and make this quality information explicit. Assessing data quality in a scalable way is a challenge in Linked Data as the underlying data stems from many autonomous, evolving, and increasingly large data sources [30].
Luzzu is a Quality Assessment Framework for Linked Open Datasets, based on the Dataset Quality Ontology, allowing users to define their own quality metrics. Luzzu is an integrated platform that:

- Assesses Linked Data quality using a library of generic and user-provided domain specific quality metrics in a scalable manner;
- provides queryable quality metadata on the assessed datasets;
- Assembles detailed quality reports on assessed datasets.

The figure 2.13, show the data quality estimation radar graph generated by Luzzu framework:

![Radar Graph](image)

Figure 2.13: Spanish DBpedia data quality estimation by Luzzu, from source: lod-cloud.net.

As we can see in the figure, the Spanish DBpedia has a relatively good quality aspects compared to the LOD latest average. The human-readable labelling metric of classes, properties and entities, is the percentage of entities having an rdfs:label or rdfs:comment or boolean (e.g. whether or not a SPARQL endpoint is accessible) \(^{10}\). The undefined classes and properties metric also is good as well. But in the Basic Provenance () and Ontology Hijacking (Hijacking occurs when an ontology makes reference to terms T, properties P or objects O from another namespace K, where that namespace K does not really have any definitions for T, P and O \([31]\).) metrics it’s scoring very low.

\(^{10}\)[https://www.w3.org/2013/dwbp/wiki/QualityQuestionnaire]
A related study to the Spanish DBpedia quality [29], shows that 97.93% of the properties used in the Spanish DBpedia are auto-generated properties (not using the mappings in the DBpedia extraction process). Properties suffer from conciseness quality issues due to several causes such as inconsistent capitalization (857 properties), inconsistent usage of accents (1,252 properties), slashes in infobox labels (107 properties), spelling mistakes, etc. Also other inconsistencies such as properties simultaneously being object and datatype properties (3,380 properties) or wrong domain/range values (2,821 properties). Syntactic validity problems were found in 3,675 properties.

2.8.3. Current Spanish DBpedia pages interface

The current Spanish DBpedia interface is an old one as you can see in figure 2.14, and a brand new DBpedia interface is already available, which being used by DBpedia an some other DBpedia chapters. We are able to see the interface via a link to some resource, for example the resource for "Español"  

![Figure 2.14: The current Spanish DBpedia pages interface.](image)

In this work we provides a way to change this interface to the new one that we see in figure 2.15.

An approach to automatically update the Spanish DBpedia using DBpedia Databus
In this chapter we are showing the big picture of how we solved the Spanish DBpedia update problem and the used approaches.

3.1. Development environment

We used the UPM Virtual Private Network (VPN) Remote Access Service that allows us to access certain university resources from outside the university network. In our case access a remote Kernel-based Virtual Machine (KVM).

Below is the architecture of the system running in the UPM server and how complex it is. Starting with my local host which was the testing environment, and than moving the code to practice:
3.1.1. Virtual Machine Information

<table>
<thead>
<tr>
<th>Type</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>System:</td>
<td>Host: esdbpedia Kernel: 4.15.0-109-generic 64 bits</td>
</tr>
<tr>
<td></td>
<td>Console: Ubuntu 18.04.4 LTS</td>
</tr>
<tr>
<td>CPU(s):</td>
<td>8 Single core Intel Xeon E312xx cache: 131072 KB</td>
</tr>
<tr>
<td>RAM:</td>
<td>Size: 40GB</td>
</tr>
<tr>
<td>Graphics:</td>
<td>Card: Red Hat QXL paravirtual graphic card</td>
</tr>
<tr>
<td>Network:</td>
<td>Card: Red Hat Virtio network device driver: virtio-pci</td>
</tr>
<tr>
<td>Drives:</td>
<td>HDD Total Size: 268.4GB</td>
</tr>
</tbody>
</table>

Table 3.1: Virtual machine system information

3.1.2. Programming Language

Bash scripting is an extremely useful and powerful part of system administration and development. Bash is a Unix shell, which is a Command Line Interface (CLI) for interacting with an Operating System (OS). Any command that you can run from the command line can be used in a bash script. Scripts are used to run a series of commands. Bash is available by default on Linux [32] and macOS operating systems [33].

In our solution there are certain tasks, we need to repeat over and over again by running commands. Shell scripting comes handy and we can create a script with commands we want to run or execute frequently. We can schedule it through cron. We can give permissions to it so that only those who are supposed to run can run it. If someone doesn’t know a set of commands but we want him to run those commands, put them in a shell script and give him command to copy paste in prompt and execute.

Mainly the script that we wrote (.sh files) can be opened or run in the terminal on Linux or Unix-like system. The .sh file is nothing but the shell script to manage the Spanish DBpedia updating by performing different tasks and executing different commands. The easiest way to run .sh shell script in Linux or UNIX is to type the following commands. We open the terminal (your shell prompt) and type the commands:

Code 3.1: Commands to successfully run a script file.

```bash
1 # Set execute permission on your script:
2 chmod +x script-name-here.sh
3 # To run your script, enter:
4 ./script-name-here.sh
```

An approach to automatically update the Spanish DBpedia using DBpedia Databus
CHAPTER 3. DESIGN

5
6    # Another option is as follows to execute shell script:
7    sh script-name-here.sh
8    # OR
9    bash script-name-here.sh

So to run our script, we first get the files from Github \(^1\), change directory to the downloaded folder which looks like figure 3.2:

![Figure 3.2: The folder of files needed for the Spanish DBpedia update.](image)

and then execute the bash commands on the dbpedia-cron.sh file. These steps are shown in the following code:

Code 3.2: Commands to successfully run the script file for automating the Spanish DBpedia updating.

1   sudo git clone https://github.com/oussama-talaoui/updating-esdbpedia.git
2   cd updating-esdbpedia
3   sudo bash dbpedia-cron.sh

The terminal will prompt an output to configure the cron job, we follow the instructions and we configure the frequency of running the cron job that will handle the update process for us periodically. For example for updating the Spanish DBpedia each week at Saturday night we introduce the following:

![Figure 3.3: Example of the input to configure cron job each week](image)

\(^1\)https://github.com/oussama-talaoui/updating-esdbpedia
3.2. Real environment

The DBpedia dbpedia.org makes the redirection of es.dbpedia.org (a sub domain of DBpedia) to one of the Ontology Engineering Group (OEG) machines where the Spanish DBpedia is located. So, to put the results of the machine that we worked on in this project, and which has the new Spanish DBpedia data we do another redirection from the OEG machine to it.

Figure 3.4: Redirecting the virtual machine Spanish DBpedia to the real environment
This chapter is a more detailed section of the report, that shows the used scripts, and explains the different script sections. Configuring virtuoso and configuring DBpedia loading process.

4.1. Installing and configuring Virtuoso

The Virtual Machine (VM) the we worked on has Ubuntu 18.04 OS which is based on the long-term supported Linux release series since it is easy to use, and stable. To do all the installation we used the Ubuntu terminal. Before starting the Virtuoso installing process we have to install the needed packages first, these packages are essential for the commands that we are using in the terminal. To install or to make sure that the packages are installed we execute the following code in the terminal:

Code 4.1: Installing essential commands packages (Code in Github).

```
1 sudo apt-get update
2 sudo apt-get install build-essential module-assistant
3 sudo apt-get install git
4 sudo apt-get install autoconf autogen
5 sudo apt-get install libtool bison flex gawk gperf
6 sudo apt-get install libssl1.0.0 libssl-dev
7 sudo apt-get install curl
8 sudo apt-get install moreutils
9 sudo apt-get install parallel
10 sudo apt-get install libssl1.0-dev
11 sudo apt-get install libssl1.0-dev
```
We can execute the command (line 12 of Code 4.1), and we enter the root password so we won’t have to use `sudo` command every time we execute a command. After making sure all the needed packages are installed we start downloading the latest stable release of the virtuoso-opensource 7.x branch and we create new local branch named "7":


```
1  git clone git://github.com/openlink/virtuoso-opensource.git
2  cd virtuoso-opensource
3  git checkout --track remotes/origin/stable/7
```

We set the compiler flags. (e.g. `export CFLAGS="-O2 -m64"` for "typical" 64 bit linux machines:

Code 4.3: Specify compilation settings.

```
1  export CFLAGS="-O2 -m64"
```

We generate the configuration files:

Code 4.4: Generate Configuration Files (Code in Github).

```
1  ./autogen.sh
```

When the command complete correctly the final output will be the following:

![Configuration output](image)

Figure 4.1: The output after the auto generation command finishes as intended.
We configure the installation path, different options are possible, but for our case we specify Virtuoso to install in /usr/local/virtuoso using the option `prefix` (line 1 of Code 4.5), the option `program-transform-name` (line 1 of Code 4.5) is to rename the virtuoso Interactive Structured Query Language (ISQL) command interface as `isql-v` to avoid a clash with other commands:

Code 4.5: Configuring the installation path (Code in Github).

```bash
./configure --prefix=/usr/local/virtuoso --program-transform-name="s/isql/isql-v/"
```

When the configuration finishes correctly we get the following summary:

![Virtuoso Open Source Edition (Column Store) 7.2.6 configuration summary](image)

An approach to automatically update the Spanish DBpedia using DBpedia Databus
Virtuoso must be installed properly by now, we just need to configure it by modifying in the virtuoso.ini file which will be located in the following path '/usr/local/virtuoso/var/db'. We edit the following entries in the file if they are different:

**Code 4.7: Location of the database files (Code in Github).**

```
[Database]
DatabaseFile = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso.db
ErrorLogFile = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso.log
LockFile = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso.lck
TransactionFile = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso.trx
xa_persistent_file = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso.pxa
...

[TempDatabase]
DatabaseFile = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso-temp.db
TransactionFile = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso-temp.trx
```

**Code 4.8: Location of the database temp files (Code in Github).**

```
...
[TempDatabase]
DatabaseFile = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso-temp.db
TransactionFile = /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso-temp.trx
...
```

We adjust the parameters NumberOfBuffers and MaxDirtyBuffers in the virtuoso.ini file according to the recommendations in comments. Tuning these parameters will help with performance when running with large data sets. Since we used a 40GB RAM machine we modified the following:

**Code 4.9: Allocated memory (Code in Github).**

```
...
NumberOfBuffers = 3060000
MaxDirtyBuffers = 2250000
...
```

Each buffer caches a 8K page of data and occupies approximately 8700 bytes of memory. It’s suggested to set this value to 65% of ram for a db only server so if you have 36 GB of ram: $36 \times 1000^3 \times 0.65/8700 = 2689655$ default is 2000 which will use 16 MB ram. Also, we make sure to remove the white-space when we uncomment existing lines.

We set both parameters, ThreadCleanupInterval and ResourcesCleanupInterval to 1 in order to reduce memory leaking:

**Code 4.10: Parameters to reduce memory leaking (Code in Github).**

```
[Parameters]
...
ThreadCleanupInterval = 1
ThreadThreshold = 10
ResourcesCleanupInterval = 1
...
```
CHAPTER 4. DEVELOPMENT

To avoid memory errors, we make sure the values for the parameters NumberOfBuffers and MaxCheckpointRemap are not set with the same values. It’s recommended to set MaxCheckpointRemap to 1/4th of NumberOfBuffers:

Code 4.11: Parameter to avoid memory error (Code in Github).

```plaintext
... 
[Database]
...
MaxCheckpointRemap = 765000
...
[TempDatabase]
...
MaxCheckpointRemap = 765000
...
```

We increase the MaxQueryMem that controls the maximum amount of memory that can be used across the server process for large vectors:

Code 4.12: Parameter to avoid memory error (Code in Github).

```plaintext
... 
MaxQueryMem = 4G ; memory allocated to query processor...
```

Enabling the server log access will permit creating log files with the name for [date as DDMMYYYY].log when running the virtuoso server. To do so we un-commented the following line:

Code 4.13: Enabling the server log access for log files creation (Code in Github).

```plaintext
HTTPLogFile = logs/http.log
```

4.2. Installing VAD packages

A Virtuoso Application Distribution (VAD) package contains all required Virtuoso components, which would constitute an application or hosted solution, within a single distributable file. The VAD packages can be installed from a file with the use of an Structured Query Language (SQL) function, and the function can be executed via the Virtuoso ISQL utility which allows users to execute queries and scripts in Virtuoso. To use ISQL we first need to run the Virtuoso server using the following command:


```plaintext
sudo virtuoso-t -f -c /usr/local/virtuoso/var/lib/virtuoso.ini
```

We download the VAD packages using wget command to the path "/usr/local/virtuoso/share/virtuoso/vad" which is by default the allowed path for Virtuoso to access, and it is specified in the virtuoso.ini file by setting "DirsAllowed" parameter. The Conductor VAD package exits in the allowed path so we don’t have to download it. The default login parameters for Virtuoso database administrator are, username "dba" and password "dba". The default port for SQL data access via ISQL is port 1111:

An approach to automatically update the Spanish DBpedia using DBpedia Databus
4.3. DBpedia interface and Spanish DBpedia data configuration

The DBpedia VAD package includes various things, like the DBPedia pages template, which are not part of the DBpedia data set nor the base Virtuoso install. We open ISQL with the command `isql-v` than we execute the lines of command to associate the Spanish DBpedia values to their names in the Database registry:

```
1
2
3
4
5
6
7
8
9
10
11
```

It’s recommended to execute the lines one by one to make sure all the code lines are executed correctly. Spanish DBpedia must support Internationalized Resource Identifiers (IRIs) because of the nature of some Spanish characters, we enable that for the DBpedia interface giving "on" as a value for "dbp_decode_iri" parameter (line 2 of Code 4.16).

We download the DBpedia .vad file which is the DBpedia Virtuoso plugin and we install it same as previous vad packages.

```
1
2
```
4.4. Cron Job For Scheduled Script Execution

Cron jobs are very useful Linux tool aimed at saving you time by scheduling tasks within your machine. A programmed cron task will execute commands within our script by the minute, day, week or month parameters that we specify. We scheduled it to run the script of updating the Spanish DBpedia automatically on a period that we choose. It’s a repetitive task of updating that becomes easy when incorporating a cron job. While there are numerous ways to run a cron task, we will be using the crontab option that is inherent within Ubuntu to set up our job. We created the following code so it will be easier for normal users to create the cron job by following simple instructions, also it’s able to check if a job already exists and delete it if we want.

The script retrieves the month first, it should be a number containing digits from 0 to 9 and between 1 and 12:

```bash
Code 4.18: Cron job creation code to retrieve month (Code in Github).
1   echo "Creating a cron job for updating Spanish DBpedia."
2   read -p "Enter month (1 to 12 from January to December, empty or * for every month): " month
3   while [[ ! $month =~ ^[0-9]+$ || ! $month -ge 1 || ! $month -le 12 ]]
4     do
5       if [[ $month == '' || $month == '*' ]]
6         then
7           month='*
8           break
9       fi
10       printf 'Please enter a valid month number: '
11       read month
12   done
```

Then it retrieves the day of the week, it should be a number containing digits from 0 to 9 and between 0 and 6. The 0 is for Sunday and 6 for Saturday:

```bash
Code 4.19: Cron job creation code to retrieve weekday (Code in Github).
1   read -p "Enter weekday (0 to 6 from Sunday to Saturday, empty or * for every weekday): " weekday
2   while [[ ! $weekday =~ ^[0-9]+$ || ! $weekday -ge 0 || ! $weekday -le 6 ]]
3     do
4       if [[ $weekday == '' || $weekday == '*' ]]
5         then
6           weekday='*
7           break
8       fi
9       printf 'Please enter a valid weekday: '
10      read weekday
11   done
```

Same logic for retrieving the rest of the cron job parameters (month day, hour and minute), we can leave a parameter empty by leaving the input empty or putting ‘*’. 

An approach to automatically update the Spanish DBpedia using DBpedia Databus
CHAPTER 4. DEVELOPMENT

Afterwards the script constructs the cron job command with the input data and assigns time stamps to couple of output files that will have the log for all the executed commands from the updating script. There is two types of log files, one for the commands outputs called timestamp_esdbpedia-update.log and the other for execution error code called timestamp_code-error.log that contains the error code after the script executing, 0 in case there is no execution error:

Code 4.20: Constructing the Cron job command (Code in Github).

```bash
dir=$(pwd)
timestamp='$(date "+%Y.%m.%d-%H.%M.%S")'
mkdir -p logs
cronjob="$minute $hour $day $month $weekday "$dir"/update-script.sh | ts '[\%Y-\%m-\%d \%H:\%M:\%S]' >> "$dir"/logs/"$timestamp"_esdbpedia-update.log 2>&1; echo $? | ts '[\%Y-\%m-\%d \%H:\%M:\%S]' >> "$dir"/logs/"$timestamp"_code-error.log"
```

Backslashes used in Code 4.20 are to escape % character which is a new line in ts command.

The cron script than checks of the job already exists in the cron list, and it prompts option related to the situation:

Code 4.21: Checking if the Cron job already exists (Code in Github).

```bash
cronexists='echo Cron job for this file already exists!
read -p "Do want to remove it (y/n)? " -n 1 -r
if [[ ! $REPLY =~ ^[Yy]$ ]]; then
  exit 1
else
  crontab -r
fi
read -p "Do want to create this new cron job (y/n)? " -n 1 -r
if [[ ! $REPLY =~ ^[Yy]$ ]]; then
  exit 1
else
  { echo "$cronjob"; } | crontab -
fi
''
```

An approach to automatically update the Spanish DBpedia using DBpedia Databus
Before executing the cron job command we must mark the 'update-script.sh' file as executable:

```
| chmod +x update-script.sh
```

An approach to automatically update the Spanish DBpedia using DBpedia Databus.
4.5. Updating The Spanish DBpedia

The script file that is responsible for updating the Spanish DBpedia is ‘update-script.sh’ which has the commands to take care of all the updating steps from downloading the DBpedia data files, to un-compressing, clearing the name space and bulk loading the new data and at the end checking if the process executed well by querying the Spanish DBpedia data.

4.5.1. Downloading the Spanish DBpedia data from the Databus

Through the Databus SPARQL API available via the URL: https://databus.dbpedia.org/repo/sparql using a POST request because the sparql query is long (add the used spark query to the appendices).

We post request to the Spanish sparql end point using content in the sparql code in the txt file and write the list of downloadable files to a variable as list after we remove double quotes "" from downloadurls. We then remove any previous existing data files if there are any, and we start downloading the data using wget while attaching the timestamp to each file.

We remove unneeded additional files that come with the download.

Code 4.23: Querying the download URLs and Download the Spanish Data (Code in Github).

```bash
QUERY=$(</home/zinner/Desktop/request.txt)
#
# post request to a sparql end point using content in txt file and write the result to
# variable as list
QUERY=$(<"$DIR"/request.txt)
#
# retrieve downloadurls with sparql query
DOWNLOADURLS="curl -X POST --data-urlencode query="$QUERY" --data-urlencode format="text/tab-separated-values" "https://databus.dbpedia.org/repo/sparql"
#
# remove double quotes " from downloadurls, because of wget scheme missing
DOWNLOADURLS="echo $DOWNLOADURLS| sed 's/\"/\'/g'
#
# remove previous data files
echo -n 'Deleting files... '
rmdir -f /usr/local/virtuoso/share/virtuoso/"$DATAPATH"/* & PID=$!
```
4.5.2. Uncompressing the data files

The un-compressing process of the downloaded .bz2 files is done using a parallel executing to increase un-compression speed, and after it’s done all the .bz2 files are deleted to free space.

Code 4.24: Uncompressing the bzip2 files (Code in Github).

```
printf "Uncompressing .bz2 files..."
ls /usr/local/virtuoso/share/virtuoso/"$DATAPATH"/*.bz2 | parallel bunzip2 & PID=$!
```

4.5.3. Checking the Virtuoso server status

To execute the rest of the command which requires that the Virtuoso server is running, the script checks the server port. If the Virtuoso server is not running the script runs it. But first, the script removes the virtuoso.lck file if it does already exist because some times it causes problem trying to run the Virtuoso server:

Code 4.25: Checking the Virtuoso server status and running in case it’s down (Code in Github).

```
rm -f /usr/local/virtuoso/var/lib/virtuoso/db/virtuoso.lck
```

An approach to automatically update the Spanish DBpedia using DBpedia Databus
4.5.4. Bulk Loading the data

To bulk load the data the virtuoso we must first create a file for the graph name inside the files directory. The file is called "global.graph" and it does contain the graph name inside. We then clear the graph in virtuoso if it does already exists and we clean the load-list table.

It’s way faster to launch multiple bulk loaders depending on the number of cores that our machine has, so the script get that number and divides it by 2.5 for the recommended amount of bulk loaders that can run simultaneously.

Code 4.26: Clearing the graph and loading new data (Code in Github).

```
# Create graph name file
echo "http://es.dbpedia.org" > /usr/local/virtuoso/share/virtuoso/vad/global.graph

# Clear the graph and the load list table, to load new data
# Large graphs can be cleared by changing the transaction log mode to autocommit on each operation, deleting the graph(s) or triples. This is easily done using the Virtuoso log_enable function, with the settings log_enable(3,1).

isql-v 1111 dba dba exec="log_enable(3,1);"

isql-v 1111 dba dba exec="DELETE FROM rdf_quad WHERE g = iri_to_id ('http://es.dbpedia.org');"
# isql-v 1111 dba dba exec="SPARQL CLEAR GRAPH <http://es.dbpedia.org>;;

isql-v 1111 dba dba exec="DELETE FROM DB.DBA.load_list;"

# Getting number of cores in the machine and running multiple Loaders using no cores / 2.5, to optimally parallelize the data load.

noperation=`echo $(nproc) / 2.5 | bc`
echo $noperation

for (( c=1; c<=$noperation; c++ ))
    do
        isql-v 1111 dba dba exec="rdf_loader_run();" &
    done

wait
```

4.5.5. Checking the update process

At the end of the loading process we have checkpoint command to commit all transactions to the database and create a checkpoint. Other commands to check some of
the data loaded into virtuoso and the number of tuples that we successfully loaded.

Code 4.27: Creating a checkpoint and querying the data (Code in Github).

```sql
isql-v 1111 dba dba exec="checkpoint;"
# Showing the time taken to load the datasets to our graph
isql-v 1111 dba dba exec="select min(ll_started) as start, max(ll_done) as finish,
→ datediff('second', min(ll_started), max(ll_done)) as delta from load_list where
→ ll_graph like 'http://es.dbpedia.org';"
```

```sql
isql-v 1111 dba dba exec="SPARQL SELECT (COUNT(*) as ?Triples) FROM <http://es.dbpedia
→ .org> WHERE { ?s ?p ?o };
```

```sql
isql-v 1111 dba dba exec="SPARQL SELECT * FROM <http://es.dbpedia.org> WHERE {
→ ?s ?p ?o } LIMIT 100 ;"
```

4.5.6. SPARQL Query

We can customize SPARQL queries over the Databus SPARQL API, to be able to query exactly the download links for the Spanish data that we need. Databus pages \(^1\) serve as landing pages to show documentation and browser-clickable download links as well as create collections including the SPARQL query generator at the bottom of each page \(^2\). By creating an account on the databus we can easily generate SPARQL queries, check our query summary and view the downloadable data. Also we made sure to get the latest data available.

We specify that the compression type should be bzip2 for all the downloadables, and that the RDF data is stored in the turtle (.ttl) format.

Code 4.28: Extract from The SPARQL Query to get Spanish data Downloadurls (Code in Github).

```sql
PREFIX dcat: <http://www.w3.org/ns/dcat#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dataid-cv: <http://dataid.dbpedia.org/ns/cv#>
PREFIX dataid: <http://dataid.dbpedia.org/ns/core#>

SELECT DISTINCT ?file WHERE {
  {
    # Get all files
    SELECT DISTINCT ?file WHERE {
      }
      }
  }
}
```

\(^1\)https://databus.dbpedia.org/
\(^2\)http://dev.dbpedia.org/Download_DBpedia
4.5.7. Duration and analysis of the update process

Virtuoso is clearly the fastest free triple store available [34], so using a powerful machine loading and deleting data from it is going to be fast. But the whole data updating process has other operations that take time, which are downloading the data and uncompressing. The downloading speed depends on the internet connection, but to optimize it we download compressed files and then we do uncompression locally. In our machine the script download 1.9GB of bzip2 files in 3 minutes. We also parallelized downloads with wget by sending the wget process to the background using the &-operator:\(^3\):

Code 4.29: Parallelizing downloads with wget (Code in Github).

```bash
for url in $DOWNLOADURLS; do
  echo "Downloading" $url
current_time=$(date "+%Y.%m.%d-%H.%M.%S")
filename=$(basename "$url")
  wget -q -c $url -O /usr/local/virtuoso/share/virtuoso/$DATAPATH/$current_time.$filename &
done
```

Each call to wget is forked to the background and runs asynchronously in its own separate sub-shell. For uncompression the GNU parallel command is well suited to this type of thing, it uses as many cores available in our machine, keeping each core busy with unzipping a file, until they are all done.

Code 4.30: Parallelizing Bzip2 files unzipping with Parallel command (Code in Github).

```bash
printf "Uncompressing .bz2 files..."
ls /usr/local/virtuoso/share/virtuoso/$DATAPATH/*.bz2 | parallel bunzip2 & wait
echo 'Done!'
```

\(^3\)https://www.baeldung.com/linux/wget-parallel-downloading
Because of this the uncompression takes 5 minutes and the output data from it is 20GB.

<table>
<thead>
<tr>
<th></th>
<th>Unzipping time (min)</th>
<th>Download time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>Parallel</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4.1: Compression between normal and parallel performance.

Other operations that take time during the uploading process are clearing the existing graph, Bulk loading the data to Virtuoso and creating checkpoint. Clearing the graph takes about 4 hours and we are working to optimize it. Concerning the Bulk loading and since we are working on a multi-core machine, we run the loading function multiple times depending on the number of cores we have, it is recommended a maximum of number of cores devised by 2.5 to optimally parallelize the data load and hence maximize load speed. In the script it’s implemented as the following:

Code 4.31: Running multiple Bulk Loaders (Code in Github).

```bash
noperation=`echo $(nproc) / 2.5 | bc`
echo $noperation
echo "Running multiple Loaders:"
for (( c=1; c<=$noperation; c++ ))
do
   isql-v 1111 dba dba exec="rdf_loader_run();" &
done
wait
```

In this way, and using our machine which has 8 cores we end up devising the normal time by 3 and the loading process takes 40 minutes other than 2 hours. The checkpoint command to commit all transactions to the database takes about 1 minute.

Along the execution of our script we calculate the time taken to execute it from start to finish in minutes, and displays it at the end of the log each update log file.

Code 4.32: Calculate and write the execution time in minutes (Code in Github).

```bash
#--------------------- Start of the script ---------------------
start=`date +%s`
... 
end=`date +%s`
runtime=$((end-start))
echo Running the script took $((runtime / 60)) minutes.
```

4.6. Modifying The DBpedia Pages Interface

While the Virtuoso server is running, we go to the Virtuoso Web User Interface using the URL http://localhost:8890 in our browser:
We select the Conductor link in the left frame and login as the dba user since the default user is dba with password dba, or we use our credentials if we changed the default credentials:
Figure 4.8: Virtuoso Conductor main page.

We go to VAD -> dbpedia:

Figure 4.9: WebDAV browser.

We click on the edition icon corresponding to the description.vsp file to edit it:
We replace the following Hypertext Markup Language (HTML) code with our code:

Code 4.33: Default logo HTML code (Code in Github).

```html
<a class="navbar-brand" href="<?=registry_get('dbp_imprint') ?>" title="About DBpedia" style="color: #2c5078">
    <img class="img-responsive" src="/statics/images/dbpedia_logo_land_120.png" alt="About DBpedia" style="display: inline-block; margin-top: -12px"/>
</a>
```

An approach to automatically update the Spanish DBpedia using DBpedia Databus.
Finally we click on save, and the modification will be applied:

![Image of saving modification]

**Figure 4.12:** The HTML code for modifying the logos.

We can check by searching for a resource or a DBpedia page:

![Image of modified logos]

**Figure 4.13:** Checking the modified logos.

Using the same procedure we can modify the HTML component of the DBpedia pages as we like, by modifying in the description.vsp file.
5

Results

This chapter shows some results from the Spanish DBpedia updating process.

5.1. Checking the new loaded Spanish DBpedia data

In the first upload where we used the the DBpedia 2019-08-30 release to get the downloadurls, we were able to upload 128,629,035 triples (line 32 of Code A.1). The SPARQL query that we used to get the number of triples is line 4 of Code 4.27.

After changing the SPARQL query in order to get the latest version releases, we got an upload of 133,181,970 triples (line 61 of Code A.2). The SPARQL query that we used to get a limit of 100 triples is line 5 of Code 4.27.

The latest number of triples that was uploaded is 133,541,420 triples and the VM Virtuoso server port was successfully redirected to a real environment having the SPARQL endpoint \(^1\) where we queried the triples number figure 5.1.

\(^1\)http://testdbpedia.linkeddata.es/sparql

Figure 5.1: Querying the triples number in real Spanish DBpedia SPARQL endpoint.
CHAPTER 5. RESULTS

5.2. Data refresh use case

To make sure that our end point has some new data that’s wasn’t available before, we looked for a new Spanish resource called "Pandemia de enfermedad por coronavirus de 2019-2020" \(^2\) that was added in January of 2020:

As we can see from the captures, the resource was added in the latest version, and

\(^2\)https://es.wikipedia.org/wiki/Pandemia_de_enfermedad_por_coronavirus_de_2019-2020

Figure 5.2: No results about novel corona virus before updating data.

Figure 5.3: Finding results about novel corona virus after updating data.
wasn’t available in the first one. So we can be sure that the upload process gets new Spanish data.

5.3. The updating process log files

Let’s say the script was running for a few months, to know if it was running properly at that time, the system administrators should check the generated log files. Each update generates two log files with timestamp indication the time and date when the script started executing.

![Log files Generated From The Update Process.](image)

Figure 5.4: Log files Generated From The Update Process.

Each line inside the log files is indicated with a timestamp, which makes it easy to find any issues that occurred at any time during that period. View and extract of the log file A.1.
Conclusion

Keeping the Spanish DBpedia up to date provides valuable data that is not found in any other DBpedia. In this work we provided an automated approach to the updating process problem, by augmenting the frequency of updates and minimizing the time.

In this work we presented in detail the Spanish DBpedia updating approach, while we made all the process resources open and available in GitHub for usage. The process consisted of, firstly, installing and configuring Virtuoso server in the used machine. Showing each step and how we can execute it without problems, starting from the needed Linux packages, to configuring Virtuoso parameters for an optimal performance, going through downloading and installing needed VAD packages for a better DBpedia pages interface experience. Secondly, we explained the bash script that we created, and how it’s able to get downloadurls of the latest Spanish data available in the Databus. To then uncompressing the downloaded files and uploading them into the graph inside Virtuoso server. We used the recommended multiple loading process for minimizing the data uploading, as well as the recommended way of deleting the graph, which takes place before every new updating execution. Many improvements can be done to the bash script files, like using more variable and making it more user friendly. The query that’s responsible for getting the Spanish downloadurls from Databus can be improved as well to get more data and with better quality.

The result of this project can be a valuable source for the community or individuals, since all the work and code was made available and accessible to anyone who is interested in implementing this process whether for testing or for other DBpedias pages that doesn’t already implement one.

As future work we must perform an analysis of the Spanish DBpedia instances. Such
analysis will shed light on some quality issues, that might be fixed already if they were common with other DBpedias, or must be looked into if they were specific to the Spanish DBpedia. Furthermore, solutions to these issues must be automated using adequate tools.
References


REFERENCES


REFERENCES


Appendices


```plaintext
[2020-07-10 04:37:02] Deleting files... Done!
[2020-07-10 04:37:02] Downloading file
categories/2019.08.30/categories_lang=es_articles.ttl.bz2
categories/2019.08.30/categories_lang=es_labels.ttl.bz2
categories/2019.08.30/categories_lang=es_skos.ttl.bz2
...
[2020-07-10 05:00:11] Connected to OpenLink Virtuoso
[2020-07-10 05:00:11] Driver: 07.20.3229 OpenLink Virtuoso ODBC Driver
[2020-07-10 05:00:11] start finish delta
[2020-07-10 05:00:11] TIMESTAMP TIMESTAMP INTEGER
[2020-07-10 05:00:11] 2020.7.10 4:46.57 964127000 2020.7.10 4:59.1 20222000 724
[2020-07-10 05:00:11] 1 Rows. -- 2 msec.
[2020-07-10 05:00:11] Connected to OpenLink Virtuoso
[2020-07-10 05:00:11] Driver: 07.20.3229 OpenLink Virtuoso ODBC Driver
[2020-07-10 05:00:11] Type HELP; for help and EXIT; to exit.
```

An approach to automatically update the Spanish DBpedia using DBpedia Databus

57
APPENDIX A. APPENDICES


---

Deleting files... Done!

---

Running the script took 24 minutes.
An approach to automatically update the Spanish DBpedia using DBpedia Databus
An approach to automatically update the Spanish DBpedia using DBpedia Databus


```
[2020-07-19 00:38:03] Downloading http://akswnc7.informatik.uni-leipzig.de/dav/vehnem/
  replaced-iris/long-abstracts/2020.02.01/long-abstracts_lang=es_uris=en.ttl.bz2
[2020-07-19 00:38:03] Downloading http://akswnc7.informatik.uni-leipzig.de/dav/vehnem/
  replaced-iris/sameAs/2020.02.01/sameAs_lang=es_uris=en.ttl.bz2
[2020-07-19 00:38:03] Downloading http://akswnc7.informatik.uni-leipzig.de/dav/vehnem/
  replaced-iris/labels/2020.02.01/labels_lang=es_uris=en.ttl.bz2
[2020-07-19 00:38:03] Downloading http://akswnc7.informatik.uni-leipzig.de/dav/vehnem/
  replaced-iris/short-abstracts/2020.02.01/short-abstracts_lang=es_uris=en.ttl.bz2
[2020-07-19 00:41:27] Removing unneeded files.
[2020-07-19 00:51:48] Uncompressing .bz2 files...Done!
[2020-07-19 00:51:49] tcp 0 0 0.0.0.0:1111          0.0.0.0:* LISTEN 15875/virtuoso-t
[2020-07-19 00:51:49] tcp 0 0 0.0.0.0:8890          0.0.0.0:* LISTEN 15875/virtuoso-t
[2020-07-19 00:51:49] server running
```
An approach to automatically update the Spanish DBpedia using DBpedia Databus
<table>
<thead>
<tr>
<th>start</th>
<th>finish</th>
<th>delta</th>
</tr>
</thead>
</table>

1 Rows. -- 52 msec.

<table>
<thead>
<tr>
<th>start</th>
<th>finish</th>
<th>delta</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020.7.19 5:1.99</td>
<td>2020.7.19 5:1.12</td>
<td>831</td>
</tr>
</tbody>
</table>

133181970

1 Rows. -- 538 msec.

Running the script took 270 minutes.