Linked Open Data Survey and Development of RDB2RDF Evaluation Framework

Robin Edvinsson

May 13, 2016
Master’s Thesis in Computing Science, 30 ECTS credits
Supervisor at CS-UmU: Anders Broberg
Supervisor at Knowit Norrland: Andreas Hed
Examiner: Henrik Björklund
Abstract

Linked open data is a relatively new way to publish data according to four principles. In this thesis a survey of tasks regarding areas such as SPARQL and publish relational databases as linked open data is surveyed to receive experience and get knowledge of time, complexity, maturity and agility of the different areas.

Data could be stored in different formats and ways, but the most common is that data is stored in relational databases. RDB2RDF is a group of tools used to convert relational databases to RDF and the functionalities of the tools varies. To be able to choose tool to publish relational data with, an evaluation framework is developed in this thesis with the aim to help anyone to choose RDB2RDF tool. This is done through literature studies, interviews with a company, persons and looking at related work. Three tools are evaluated with the framework resulting in two tables and a short summary of information about each tool, which is used to choose tool to use in the survey.

The development of the framework in addition to the survey of tasks indicates that this is an interesting field for the future but that there are some problems due to stability and maturity.
Acknowledgements

I would like to thank knowit Norrland AB for their hospitality and giving me the opportunity to do this thesis. I would also like to give a special thanks to my two supervisors, Andreas Hed at Knowit for all the help, advice and support during this thesis and Anders Broberg at Umeå University for all the help, ideas and feedback especially with the report.
## Contents

1 Introduction .......................................................................................................................... 1  
   1.1 Partner .......................................................................................................................... 2  
   1.2 Problem specification .................................................................................................. 2  
   1.3 Purpose and goals ........................................................................................................ 2  
   1.4 Method .......................................................................................................................... 3  
   1.5 Outline ............................................................................................................................ 4  

2 Background ........................................................................................................................... 5  
   2.1 Linked data .................................................................................................................... 5  
   2.2 Benefits with linked data ............................................................................................... 7  
   2.3 RDF .................................................................................................................................. 8  
   2.4 SPARQL .......................................................................................................................... 8  
   2.5 RDB2RDF ....................................................................................................................... 9  

3 In depth .................................................................................................................................. 10  
   3.1 Background of the evaluation framework ..................................................................... 10  
   3.2 Evaluation framework development .......................................................................... 10  
   3.3 The evaluation framework .............................................................................................. 16  
   3.4 How to use the framework ............................................................................................. 18  
   3.5 Discussion ....................................................................................................................... 19  

4 Evaluation ............................................................................................................................. 20  
   4.1 Which tools to evaluate .................................................................................................... 20  
   4.2 Evaluation of the tools .................................................................................................... 20  
   4.3 Which tool is best ............................................................................................................. 24  
   4.4 Discussion ....................................................................................................................... 24  

5 Publishing data and survey ..................................................................................................... 26  
   5.1 How the survey is carried out ......................................................................................... 26  
   5.2 Properties to document in survey ................................................................................ 26  
   5.3 Survey ............................................................................................................................. 27  
   5.4 General impressions of the area ................................................................................... 33  

6 Conclusions ............................................................................................................................ 35  
   6.1 Conclusion for framework .............................................................................................. 35  
   6.2 Conclusion for survey ..................................................................................................... 35  
   6.3 Restrictions ...................................................................................................................... 36  

7 Future work ............................................................................................................................ 37  

References .................................................................................................................................. 38  

Appendix A ............................................................................................................................... 41
1 Introduction

Today data gets collected from numerous sources such as the use of internet, via sensors or the information given by humans themselves. This data could be about anything such as a city, a person, or a measured value from a sensor and be stored in different ways in different formats.

Knowit Norrland AB thinks that it is going to be more and more interesting to share and publish data in the future and believes that linked open data may be the solution and future for this. But what is this linked open data? It is a way to publish data in a structured way on the web according to four principles. This is a relatively new field but the number of datasets published as linked open data is growing for each day. There are several benefits publishing data this way such as the data gets machine readable and that it becomes easier for people to find other data because of the links. Knowit sees a great potential in the area with a lot of new business models regarding for example internet of things and smart cities. But they are new in the field and that is why they want a survey to get more knowledge about linked open data and the associated techniques and tools.

The survey’s main focus is how to publish relational databases as linked open data. There are tools for this and knowit wants a good motivation of why the specific tool used is chosen. Other questions of interest in the survey surrounds SPARQL which is a W3C recommendation and the query language of linked data, the maturity, support for agile development and the experiences received when working with the different tasks and techniques in the area. This includes tasks such as creating a vocabulary or developing a semantic application using linked data.

This thesis will give answers to questions including things as the tools and techniques maturity and complexity. It will furthermore give knowit more knowledge and a wider insight in the area to be able to decide if there is any potential to offer these kind of services to customers who are collecting or publishing data.

In addition to this survey of tasks, an evaluation framework which will be used to evaluate tools relevant to what W3C calls RDB2RDF [29], these are tools for converting relational databases to resource description framework, is developed and tried out. This evaluation is used to choose and motivate why a specific tool should be used when publishing relational databases as linked open data. It is developed through literature studies, interviews with people and related work. The framework will hopefully also be helpful and used by others who wants to publish their relational databases' data as linked open data. Three tools which are chosen in consultation with knowit are evaluated by the framework developed and one of them are used to publish data in the survey part.
1.1 Partner

This thesis was conducted at Knowit Norrland AB\(^1\) in Umeå which have supplied all equipments and help throughout the thesis. Knowit Norrland AB is a subsidiary to knowit AB which is an IT, Design and Digital and management consult company.

1.2 Problem specification

Knowit thinks that it is going to be more and more interesting to share and publish data in the future and believes that linked open data may be the solution and future for this. There are a number of interesting areas where the possibility to use linked data may give more value to the data. This includes the growing area of internet of things [6], projects surrounding smart cities, open data that Swedish municipalities are publishing, the data collected by the use of internet or to publish machine readable data within a HTML webpage such that search engines could understand the content better.

Knowit has a number of questions regarding linked data that they are interested in and want to know more about. Therefore a survey should be done to get more knowledge in the area to see what possibilities there are to offer customers this and how mature the area is. The main focus is how to publish relational databases as linked open data and give a good motivation of why a specific tool is used to do that. Other areas such as the query language SPARQL and creating vocabularies among others are also of interest.

To be able to give a good motivation of the choice of tool, the first part of this thesis is to develop an evaluation framework for the RDB2RDF tools. Since publishing relational databases as linked data is vital for the expansion of the semantic web, which is explained as “a web of data” [26] and sometimes called web 3.0, this will hopefully not only be helpful in this thesis but also help other people to choose which tool to use when publishing their data.

The second part is to develop, work with and survey different tasks from knowit’s interests. This survey is done to get a wider insight, answers and knowledge of the techniques and tools in the field of linked data. To do this an agile and incremental approach is going to be used when working with the different tasks and information and experiences will be documented and discussed with knowit.

1.3 Purpose and goals

Since there are two different parts of this thesis, there are also two different goals and purposes.

The first goal is to develop an evaluation framework for tools relevant to RDB2RDF that could be used by anyone to evaluate and choose tool for their specific need when publishing relational databases as linked open data. This evaluation framework will be used as motivation why a tool is chosen to knowit in the survey part and hopefully help to make more people publish their relational database data as linked data and help the number of datasets to continue to grow. This framework could also be used in the future to help choosing tool in another project.

\(^1\) http://www.knowit.se/
The second goal is to help knowit get answers to their questions and get more knowledge in the area. By getting to know more about techniques, how well they work, maturity and how things are done this survey should help knowit to make a decision in the area of linked open data. Is it something they should put more effort into, what are the possibilities with linked open data and is it something they can offer their customers. The survey will be made by developing, creating and examine different tasks and should end up in a conclusion surrounding the area.

The work should also give a general knowledge of linked data, how to publish linked data and SPARQL.

1.4 Method

There are three parts of this thesis namely introduction, the development of the framework and the survey part.

1.4.1 Introduction

Introductory work, to get a foundation in the area this part consists mostly of literature studies where readings about linked data in general, what it is, how it is published, the benefits of using it and identifying different tools and frameworks.

To understand the problem and know what has to be done interviews with knowit to identify their goal and purpose of the project and a small study based on that was done. Based on the information received a project plan, time plan and how and what that should be done in the thesis was made.

1.4.2 Evaluation framework

The background of this evaluation framework comes from the vital part to publish relational databases as linked data for the semantic web in combination with knowit’s background, their interests and beliefs for the future of the area. By the knowledge and information from the introductory part and to solve the problem of choosing tool and motivate to knowit the best way to do this was decided to do an in depth study and develop an evaluation framework for tools relevant to what is known as RDB2RDF.

In short this framework was developed by trying to identify all the relevant features, followed up with interviews and discussions with a company, persons with different backgrounds and knowledge and lastly comparing and look at related work to further develop, get aspects, anchor and refine the framework. This was followed by evaluating and comparing three tools, both to test the framework itself and also to choose a tool to publish data with.

1.4.3 Publish data and survey

Identified all relevant tasks around linked data from knowit’s interest and valued their importance. Worked and examined the different tasks in an agile and incremental way. Examples of tasks examined are how to publish relational databases as linked data and how to create a new vocabulary. The experiences, time, complexity, maturity and how well the area supports agile development is documented.
1.5 Outline

The chapters of this thesis:

1. **Introduction**
   A brief introduction of the problem, purpose and goal of this thesis. Explains in short how the thesis is worked out and a little bit about knowit.

2. **Background**
   Introduction to linked open data, RDF, SPARQL and the area of RDB2RDF.

3. **In depth**
   Describes how the evaluation framework is developed and the finished framework is presented and discussed.

4. **Evaluation**
   An evaluation of three tools made with the framework developed in chapter 3. Both the results and personal opinions from using the framework to evaluate the tools is presented.

5. **Publishing data and survey**
   Presents how the survey is made and the result of the each task in terms of experience, time, complexity, maturity and agility.

6. **Conclusion**
   Brings up conclusions and results of the evaluation with the framework and survey.

7. **Future work**
   A brief discussion about future work regarding both the framework and survey.
2 Background

This chapter gives a description of the area of linked open data and its building blocks. The benefits of publishing the data this way and an introduction to RDB2RDF tools and mapping languages are also included. This overview is important to get a basic understanding of linked open data and to understand the rest of the thesis.

2.1 Linked data

There are several expression in the area used today such as linked open data, the acronym LOD and linked data. The data does not have to be open so there could be a difference between linked open data and linked data, but in this thesis we are assuming that the data is open even though the term linked data is used. One other commonly used expression is the semantic web where due to [4] the general opinion about how the terms are related is that the semantic web is built out of linked open data.

In The semantic web article from 2001 [1] Tim Berners Lee talked about the semantic web as an extension to the web, to quote:

“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”

- Quote from Tim Berners Lee’s article [1].

Even though this was written in 2001, the real starting point of linked data was when Tim Berners-Lee in an article [3] from 2006 explained the use of the machine readable linked data and the power of it. In that article he also mentions four rules which is known as the “linked data principles” which are:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards RDF and SPARQL.
4. Include links to other URIs, so that they can discover more things.

The easiest way to understand and get an overview of linked data is to compare it with the web of documents that we all are familiar with today. Instead of documents with links between them, it is data connected with links. Instead of HTML, it is RDF and it is building upon common web techniques such as URI and HTTP. This makes it possible to connect and browse all kinds of data such as persons, places and music [8].

The extension that Tim Berners Lee talks about makes the data on the web browsable by a machine and creates something that could be explained as a global database. He has encouraged everybody to put up their data on the web [2] and come up with how data can be ranked from one to five stars, where to reach four stars the standards RDF and SPARQL should be used and to reach five stars the data should be linked [3].
There are a lot of both legacy data and newly collected data from for example the web which according to [5] in 2007 was backed by 70% of relational databases. To keep the evolution of the semantic web and to make it as powerful as it could be this data must be published as linked open data [22]. One more thing is that internet of things is growing and there are more and more sensors used in the world today collecting a vast amount of data. This means that there are in fact more and more data collected and stored every day, often in relational databases or in some other specific data format.

The linked open data cloud [7] which shows datasets published as linked open data is continuously growing, see figure 1 and figure 2, and the size of DBpedia$^2$ which you can read about at [10] is a proof of the growing popularity in the area. This shows that many people want to publish their data on the web and help to continue this growth. By using all the data collected and stored in relational databases today and link it together with other data it can become extremely powerful since the possibility to use the data published as a global database. The data itself also gets refined which can lead to great improvements in the information it is giving us. The new techniques and knowledge in the area opens up for new business models and are interesting both for people owning the data the ones who want to consume it.

![Figure 1: Linked open data cloud shows all the datasets published as linked data in March 2009 [7].](image)

$^2$ http://wiki.dbpedia.org/
One of the most famous linked data datasets is the already mentioned DBpedia and as you can see in the linked data cloud, DBpedia is in the center of it. DBpedia is a linked dataset of wikipedia and right now there are 4.58 million things described there [11]. Other big datasets are Geonames\(^3\) which is a geographical database and linkedmdb\(^4\) which is a web database for movies. Some of these datasets are SPARQL endpoints which gives the ability to run SPARQL queries against the data.

There are multiple ways one can publish linked data depending on things like the format and size of the data. In this thesis the focus is on RDB2RDF and then there are two common ways to do this. If the data is not already in RDF, it could be converted to RDF and then stored in a triple store, which is a database designed for storing RDF-triples. This is known as Extract-Transform-Load (ETL). The second is to use a mapping or middleware to virtually expose a relational database so that the data can be queried using SPARQL. Both approaches have their benefits and drawbacks such as consistency problems or performance [8, 12].

### 2.2 Benefits with linked data

By linking one’s data with other data, similar as the web right now is linking documents, complex and new information can be achieved and extracted. It becomes easier to find other data from already found data because of the links and it gets easier for search engines to find and understand data. In addition to this, if the data is open everybody can use the data, which opens up new possibilities doing applications and solutions using the data that the owner has not thought about [18]. Today it exists multiple applications used to consume linked data such as browsers and linked data crawlers [11].

One of the benefits with putting up linked data on the web instead of human readable documents is that machines can do more of the work and thinking for us [9].

By refining the data and making it machine readable, the data gets more powerful than what it otherwise would have been which can be useful both for the ones owning the data and the ones

---

\(^3\) [http://www.geonames.org/](http://www.geonames.org/)

\(^4\) [http://www.linkedmdb.org/](http://www.linkedmdb.org/)
To see the costs and benefits both for the publisher and consumer, the website 5StarData\(^5\) has a list of examples for each star rank.

### 2.3 RDF

RDF is a standard format to interchange data on the web and stands for *Resource Description Framework* and is a W3C recommendation. Linked data is composed by RDF triples which could be explained in a subject - predicate - object manner that together creates a graph. This creates relationships between resources which are URIs, blank nodes or literal values such as integers or strings. These URIs and literals can represent anything in the world such as people or places and the literals could also contain things such as language tags.

There are a few different serialization formats and styles that you can express RDF in where Turtle, RDFa and RDF/XML are popular ones. \([13, 14, 15]\)

### 2.4 SPARQL

SPARQL is a query language for querying RDF data and is also a WC3 recommendation. It is the query language for the semantic web \([16]\) and is similar to SQL but for querying RDF data, which linked data consist of. SPARQL has the ability to query over diverse data sources as opposed to SQL which queries on relations. At \([17]\) you can find a SPARQL query’s general form which is that it consist of five different parts, namely:

PREFIX, namespace prefixes.
SELECT, result set
FROM, Dataset
WHERE query triple pattern
ORDER BY, DISTINCT etc modifiers.

Here you can see a simple example of a SPARQL query asking for 100 persons that have a birthdate and name.

```
PREFIX dbpo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT * WHERE {
    ?person dbpo:birthDate ?dateOfBirth.
    ?person foaf:name ?name
} limit 100
```

To receive data, run the SPARQL queries against a SPARQL endpoint which accepts and returns the result via HTTP [16]. One example is DBpedia’s endpoint where you can try out querying their data.

2.5 RDB2RDF

There exists various tools to help convert, publish and query your data. These tools come in different shapes with different features, some as a java library and some as a whole suite of tools with graphical interfaces. To narrow it down this thesis is going to focus on what W3C calls tools relevant to RDB2RDF where you at [29] can read an overview of RDB2RDF like this:

“RDB2RDF is a collection of two Recommendations to map the content of Relational Databases to RDF. The two languages are the Direct Mapping and R2RML. The mapping languages facilitate the development of several types of products. It could be used to translate relational data into RDF which could be stored in a triple store. This is sometimes called Extract-Transform-Load (ETL). Or it could be used to generate a virtual mapping that could be queried using SPARQL and the SPARQL translated to SQL queries on the underlying relational data. Other products could be layered on top of these capabilities to query and deliver data in different ways as well as to integrate the data with other kinds of information on the Semantic Web.”

- Excerpt from W3C RDB2RDF [29].

In 2012 R2RML, a RDB to RDF mapping language and A Direct mapping of RDB to RDF was presented as W3C recommendations. This as a result of the RDB2RDF working group’s work. These are recommendations that are encouraged to follow, but many tools still have their own language to do this mapping or only fulfills parts of the recommendation. For example the D2RQ mapping language which is a popular, declarative language used by the D2RQ tool to map relational databases to a virtual RDF graph.

2.5.1 RDB2RDF Tools

According to W3C tools relevant to RDB2RDF page there are seven pages or tools which are in the category at the time this is done, namely:

D2RQ  Openlink Virtuoso  R2RML Parser
Db2Triples  SPARQL2XQuery  Spyder
Ultradump

There are more tools which are sometimes built upon the mentioned tools or solving the problem in another way, but to delimit the number of tools in the area the W3C page is used in this thesis. Some of the other tools mentioned are old, not updated on a long time and not used by so many people. These tools also varies a great deal in how they work and the features they offer. It is these kinds of tools the evaluation framework developed in chapter 3 is aimed to evaluate.

---

6 http://dbpedia.org/sparql
7 https://www.w3.org/TR/r2rml/
8 https://www.w3.org/TR/rdb-direct-mapping/
9 https://www.w3.org/2001/sw-rdb2rdf/
10 http://d2rq.org/d2rq-language
11 https://www.w3.org/2001/sw/wiki/Category:RDB2RDF
3 In depth

There exist several different tools for publishing relational database data as linked open data. To be able to choose among these and motivate why a specific tool should be used an evaluation framework for the group of tools called RDB2RDF will be developed and looked into.

In this chapter we go through why and how the RDB2RDF evaluation framework is developed. From the properties identified through literature studies to the anchoring from properties received by discussion with a company and related work. The finished framework and how to use it is also presented.

3.1 Background of the evaluation framework

As brought up earlier a vast amount of data is today stored in relational databases. It is important that this data get published as linked data for the semantic web to evolve so the benefits discussed in 2.2 could be received and the open data could be used.

By publishing data as linked open data, the data gets machine readable so the machines can do more of the work for us. This in combination with that knowit wants to publish relational databases as linked data and get a motivation of why a specific tool should be used to do this in the survey, an evaluation framework for the group of tools relevant to what is called RDB2RDF should be developed.

The motivation that knowit wants is going to be based on the evaluation made with the developed framework.

By developing an evaluation framework and evaluate RDB2RDF tools this will hopefully not only help in this project, but also help other people to choose which tool they should use in just their specific project. Sometimes properties such as economy is very important and another time it may be important that the support is good. To do the framework as good as possible all features and properties that could be relevant have to be found and boiled down to a manageable framework.

3.2 Evaluation framework development

The evaluation framework is developed through incremental refinements and enhancements. Different sources such as literature studies, discussions with a company, interviews with people and related work is looked at to create a framework that suits everybody and is as good as possible.

The aim of the framework is to help people to choose tool and not evaluate technical features or performance of the different tools.

3.2.1 Foundation of the framework

The foundation of the framework consists of all the identified properties that could be relevant in the framework in the first phase. The background of these properties and features comes from reading on the internet, literature studies in the area, small tests of different RDB2RDF tools in combination of
what people in general often want to know in similar cases. Since the tools differs quite a lot there are a broad variety in the features but what features a tool support is one of the framework’s task to show. The different properties identified in this phase, which laid the foundation of the framework can be found in table 1.

<table>
<thead>
<tr>
<th>Documentation</th>
<th>Tutorials of how to use it</th>
<th>How fast is it to develop / start publishing / converting with</th>
</tr>
</thead>
<tbody>
<tr>
<td>Popularity, used by others, number of published datasets</td>
<td>Updates / releases / maintain</td>
<td>Economy, licenses, support, features.</td>
</tr>
<tr>
<td>SPARQL endpoint / query service</td>
<td>RDF dump (ETL)</td>
<td>What kind of RDF serialization are supported</td>
</tr>
<tr>
<td>Plugins, extra tools, support</td>
<td>Ability to tailor, this is develop and modify the tool</td>
<td>Supports different relational databases</td>
</tr>
<tr>
<td>Performance to convert to RDF</td>
<td>Performance SPARQL queries</td>
<td>Server and other things included in the package</td>
</tr>
<tr>
<td>Programming Languages supported</td>
<td>Time to develop</td>
<td>Support WC3 direct mapping</td>
</tr>
<tr>
<td>Support WC3 R2RML</td>
<td>Other extra features</td>
<td>Support, commercial, free or none</td>
</tr>
<tr>
<td>Graphical or CLI or programmatically</td>
<td>Directly usable from, what kind of applications or formats</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1:** Foundation of the framework, the first properties and features identified.

### 3.2.2 What is important for companies?

As a consult company it is often the case that you have to choose tool depending on the customer. In this case of converting data, the tool that is the best might depend on what format the input data is and what output data that is wanted which can vary. Since this is the case one important thing to know when choosing or evaluating tools are what kind of formats and features they support. In this case this includes what different kind of relational databases, programming languages, softwares and RDF formats that are supported.

From a company viewpoint you also must be able to give a price of what a project or specific task will cost the customer, or if it is a product company, how much it will cost the company. To do this, it is important to have a basic knowledge of how long time it will take to learn and use the tool to do the thing that is intended, in this case how long does it take to publish the data from a specific format and maybe also include how long it takes to publish it as linked data. Another thing is how easy and what possibilities there are to maintain the system in the future. Since economy and pricing is the main focus here license costs and other commercial features are of importance. To get to know how easy and fast a tool is to use, tutorials and documentation is relevant, as well as if the tool is a software library, has a good interface and support.

### 3.2.3 Interviews

To get an extra dimension to the framework it is enhanced by the input from an interview with a person involved in especially the open data field.

Properties mentioned in the interview was that if the tool was an ETL or using virtual graph conversion was not of importance for this person. What kind of databases that the tool could use and the possibility to get support was however important while recommendations such as R2RML is
followed was preferred, but not a must. To easily be able to create links to other data was preferred, but functionality was valued more than usability, but a combination of both are the best. Other things mentioned were:

- Performance
- Possibilities to do restores
- Costs
- It should be easy to use and run
- Strong supplier of the tool
- Big and active community

One more source used to enhance the framework before looking at related work is the information received from people with different backgrounds and professions but with the common denominator that they had no experience of linked open data or in the area of RDB2RDF at all. These people are possible future users and got to answer the question:

If you were going to choose between a couple of tools to do something, what would you like to know about to do this?

In this group of people the answers varied and were of different character, but to summarize the answers that were brought up and are relevant and considered in the framework, see below:

- Documentation and the quality of the documentation
- Costs
- Functionality in relation to cost
- Look and feel of the interface
- How easy the tool is to use
- A place to ask questions, like a forum or maillist

3.2.4 Related work

There exists a number of publications, documents and articles in the field of RDB2RDF. Among this work there are benchmarks and tests that compares a specific feature of the tools. Other publications are more technical or have a different aim than the framework developed here, but these are still reviewed both to give a view of what already exists, get information about what is done and ideas from the related work.

A number of these related work brings up descriptions of the tools and also a couple of benefits and drawbacks, which is a good place to get information and ideas of what one should evaluate in a framework of the kind developed in this thesis. Methods and ideas are also looked at and that is why these are reviewed.

The aim here is to find properties to add to the framework and anchor the ones already identified, get ideas on how to do the evaluation and see what already exists and has been done in the area.
Benchmarks
There are benchmarks made on tools, where tests of different SPARQL query times and the differences in performance between virtual mapping against when data are stored in a native triple stores are made. This is important as already noted from the interviews where performance was one property mentioned.

To begin with, in [19] you can read about the Berlin SPARQL benchmark which has three designed goals:

1. Test storage systems with realistic workloads of use case motivated queries
2. Allow the comparison of storage systems across internal data models
3. Do not require complex reasoning but measure query performance against large amounts of RDF data

Descriptions of the datasets, the specific queries that are used in the benchmark as well as the result can be found in the presentation [19]. Test regarding performance for things such as load times and execution times of the D2R server which is a relational database wrapper against the native triple stores Jena SDB, Sesame and Virtuoso are made.

The conclusions of the benchmark made in [19] was that Virtuoso is getting better with increasing dataset size and that D2R had problem with a specific query and also getting better with increasing dataset.

But [19] is a few years old and results from a newer benchmark can be found in [20] which is a newer Berlin SPARQL benchmark test made with some changes and made on other tools such as BigData, a different version of Virtuoso and TDB. Results of load times and throughput for the different tools are also presented in [20].

A similar benchmark test is done in [21] where they compare their own approach of RDF transformation and test and benchmark native RDF repositories against RDB2RDF transformation. The compared systems include their own METAmorphoses, D2RQ and SquirielRDF. The test made is how long time it takes to produce RDF written to standard output in RDF/XML syntax on queries depending on three groups such as result size, query graph pattern complexity and query condition complexity on a testing dataset. Their main conclusion is that it can be faster to do on the fly data transformations than converting it to a native RDF repository and therefore not necessary to migrate the relational data when publishing.

In addition to these SPARQL and RDF production benchmarks there are tests and research in the field of R2RML, direct mapping and other mapping languages to see how well they perform, the result and quality of the mappings.

Like in the paper [23] where their own Sparqlification Mapping Language (SML) is presented and compared with the R2RML mapping language. They argue that R2RML have some syntactic drawbacks and introduces SML as a human friendly alternative and compares RDB2RDF languages, D2RQ mapping language and Virtuoso RDF view mapping language. Claims due to a survey that SML is easier and lower the learning curve to RDB2RDF mappings compared with R2RML such as that it is easier to read and understand.
These kinds of benchmarks, comparisons of languages, techniques and performance tests are not going to be the aim for the framework developed in this thesis. They are however important when testing new tools to see how they stand against others and when performance is the main thing for deciding which tool to use these kind of studies are the ones that should be made or reviewed. Since there is much work to do a benchmark of tools like this and since it already consists ways and results around this area it is not included in the framework.

However, properties and results surrounding this are taken into account in the development of the framework. This shows that for example R2RML not have to be the best and that things such as mapping language is interesting to have in the framework. It is also interesting to know about RDB2RDF transformation and native triple stores since it is not always necessary to migrate relational data these topics should be included in the evaluation.

Framework and survey
In the paper [22], they survey techniques, tools and applications for RDB to RDF and categorizes and compares approaches with a framework defined in the document. The framework defined consists of six components, namely:

1. Creation of Mappings
2. Mapping Representation and Accessibility of Mappings
3. Mapping Implementation
4. Query Implementation
5. Application Domain
6. Data Integration

In this the tools are divided depending on if they are automatic or domain semantic driven where the tools Virtuoso and D2RQ is of the former. This is a few years old and before the recommendations of R2RML and direct mapping which otherwise feels like a natural divider. Other relevant features they compare are if the tools are an ETL or use a dynamic query driven approach, ways how SPARQL is executed, i.e SPARQL directly or SPARQL rewritten to SQL, is also brought up.

Tools surveyed with the framework are D2RQ, R2O, Virtuoso, Triplify and Dartgrid among others. In the survey there is a big table presented as a comparative view with all the approaches surveyed. The focus in this framework is more on comparing the different tools and implementations than helping to choose one of them from a user perspective.

The survey in [24] is more alike the framework aimed to developed in this thesis. It is a survey of RDB to RDF translation approaches and tools. In the survey they have divided the tools into R2RML compliant and non R2RML tools and it contains very extensive explanations and reviews of 17 tools. They have classified the tools into 3 different axes, namely mapping description language, mapping implementation and data retrieval method and shows that R2RML may not apply to all needs of relational database to RDF translation. Brings up compliance with R2RML test cases and sustainability with versions, releases, company and support. The survey also contains a big table with all the tools and the features supported. Examples of features looked at are mapping language, mapping description, data retrieval and features of the mapping description.

In the article [5] a number of tools are also reviewed and RDB2RDF implementations are discussed.
The descriptions of the tools brings up for instance SPARQL support, relational databases, accessible from Jena, accessible via the web, graphical plugin, ETL tool and mapping languages.

**Result of related work**

Many of these related work documents and articles mentioned [5, 21, 22, 23, 24] are summarizing, using their own frameworks to compare, reviewing and describing a number of different tools. By looking at what the authors have focused on and what are brought up when doing this, enhancements and new properties could be added and the already mentioned properties in the foundation of the framework can be anchored, changed or removed. This gives an understanding of what features and properties research people want to know about, since the things mentioned in the documents are what the authors choose to mention that must be what they think is important. These descriptions and reviews are relevant for this thesis since they are used to get inspiration, look at what properties they have compared and evaluated the tools on.

By putting together and analyzing these documents and articles when describing the tools the framework will be refined. The result of this analyze is an overview of the descriptions, frameworks, what is brought up and is summarized in table 2.

<table>
<thead>
<tr>
<th>Support of for example Jena and Sesame API</th>
<th>SPARQL version supported</th>
<th>Software library / tool suite</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPARQL directly on relational databases or if it is an ETL tool</td>
<td>Relational databases supported</td>
<td>Costs of license / which kind of license</td>
</tr>
<tr>
<td>Support and cost of support</td>
<td>The mapping language, R2RML, direct mapping, D2RQ mapping language</td>
<td>Datasets published / popularity</td>
</tr>
<tr>
<td>GUI and graphical representations.</td>
<td>RDF serializations</td>
<td>Easiness to deploy</td>
</tr>
<tr>
<td>Sustainability with versions, releases, company and support.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2**: Properties and features summary framework from analyzing related work.

Other features that not suits this framework but still are relevant are performance, how a tool scale, what size of data that is suitable and how the mapping is created from keys, tables and columns. These are a bit too technical and sometimes to extensive to test and do not fit in this framework.

One more thing that is taken into account from these related work are the presentation and comparative views of the result in tables. This is something that will be used to present the evaluations result to be able to in an easy way compare the different tools.

To categorize the tools and properties are also common, it makes everything much clearer and that is why the properties in the framework will be grouped into a number of categories.
3.3 The evaluation framework

The aim of the framework is to help to choose RDB2RDF tool when one should publish a relational database as linked data. By combining the three different sources looked at, this is company, interviews with people and looking at related work the properties in the framework has been evolved, anchored and refined. This by identifying common properties mentioned and make sure that the things people would like to know about are in the framework.

Properties mentioned by multiple sources are costs, costs to get features, license and support which therefore have to be in the framework. Documentation is also important for almost all users, but it is clear that different sources have different interests. By evaluating the properties that now are in the framework should suffice for the most users to be able to choose tool from the evaluation. The result of the analyze and the finished framework can be found below in table 3.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra tools support</td>
<td>Relational databases support</td>
</tr>
<tr>
<td>SPARQL endpoint / query service</td>
<td>RDF serialization support</td>
</tr>
<tr>
<td>RDF dump (ETL), virtual mapping or both</td>
<td>Support WC3 direct mapping</td>
</tr>
<tr>
<td>Other prominent extra features or good to knows</td>
<td>Support WC3 R2RML</td>
</tr>
<tr>
<td>Server and other things included in the package</td>
<td>Support of its own language</td>
</tr>
<tr>
<td></td>
<td>SPARQL version</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation and help</td>
<td>Popularity</td>
</tr>
<tr>
<td>Look and feel / impression of the tool</td>
<td>Sustainability</td>
</tr>
<tr>
<td>How to use it</td>
<td>Economy</td>
</tr>
</tbody>
</table>

Table 3: The finished framework with the properties to be evaluated for the tools.

There are of course a number of additional properties that could have been added to the framework, but to make the framework easy to use it is boiled down to a manageable size. Still trying to have all the relevant aspects needed to suit all different users.

3.3.1 How to evaluate

It is almost impossible to avoid properties including subjective opinions when evaluating when developing a framework like this. Because people want to know about these things, it is valued higher to evaluate these properties even though they are subjective than to repel it to get it objective. This does not mean that the result is not useful or possible to use, but it is necessary to discuss what to think about in this and similar kinds of evaluations.

Some of the properties are subjective, some has the possibility to be evaluated on measurable numerical values which would be considered objective, but unfortunately that may not be informative enough. A discussion and a few examples of how to evaluate properties are therefore necessary.
Documentation and help:
This property can be evaluated in numerous ways and give different results depending on what is looked at. Number of hits on Google or StackOverflow, number of pages in a tool's own documentation are all measurable and could be ranked objectively. This is one way that this property could be evaluated. The fact that these RDB2RDF tools are quite different in shape and some contains other functionality than only RDB2RDF and linked data this could though not be the only thing looked at. It might be needed to peel off some of this if the tool contains other functionality that is not relevant to get a fair result.

The quality and correctness are other things relevant in an evaluation, does the documentation help and how easy is it to find what you are looking for? Are there any specific forum to ask questions on or a maillist. Are there any tutorials of how to use the tool or possibility to get support and how fast can one get help. These are things which contains subjective influences and human work but to get a result it must be treated.

Because of this the best way to evaluate this property is to give both one objective measurable rank on numbers of hits on for example StackOverflow and one subjective personal rank where things are peeled off and valued.

Look and feel / impression:
This property must be evaluated subjective and it is hard to know what should be taken into account. Everybody has different knowledge, background and perceive things differently and this property is therefore evaluated on personal opinions of how easy it is to get started, the first impression of the tool and website and then try to set a subjective numerical value on this from the things experienced.

How to use it:
Similar to the look and feel property this is also hard to evaluate since how to get started with the tool and how to use it depends on many things. If it has a graphical interface, CLI or must be used programmatically may be measured that it is often easier to use tools with graphical interfaces. This on the other hand can lead to loss of control and simplify things too much, which means that the rank depends much on who the user is.
If the user is a person who is working in the area or if it is a person using this for the first time, this differs a great deal.

Therefore a subjective rank must be used for this as well. An objective rank would rely on for example if the tool has a graphical interface which would be considered a plus. But since this is not only the interesting part some personal thoughts must be added and thought of depending on who the user is and the rank should therefore take this into account.
Relational databases support:
Some major databases such as MySQL and MS server are almost expected to work by many. The number of databases used could be measured objectively and ranked upon that number. But the thing here is, if there are 15 databases supported, but not the one that you must use, the number of supported databases is not relevant. This leads to the fact that some human work to look if the database one should use is supported or not have to be done and that is the most relevant part. This property is therefore receiving one objective measurable rank and a list of supported databases.

RDF serialization support:
Which RDF serializations are possible to use when doing ETL or receiving data? Could receive rank in number of supported RDF serializations, but the same as in the database case, human work is needed to see what kind that is supported.

Popularity:
Popularity can be defined in multiple ways depending on area. In this case popularity will be evaluated on criterias such as how many datasets that are published with the tool, does any big dataset such as DBpedia use the tool or is there any research on the tool? Possible to get a objective rank depending on for example number of downloads and publications.

Sustainability:
This property comes from the related work part which looks at. When was the latest update? Number of updates? When was it released? Which company or person is the owner and creator of the tool? This could receive two ranks, where the measurable rank can depend on when for example the latest release and what kind of company the tools is backed by and number of updates. While it also could be looked up if needed.

The other properties are straight forward to evaluate, this means that license costs or if the tool supports R2RML or SPARQL will not be discussed here.

3.4 How to use the framework
The evaluation is developed to be used as one of the first steps in a pilot study as help to choose tool and should result in one evaluation document and two comparison tables. To be able to easily compare the tools the comparison tables created should contain the most important information, where one have the numerical ranks and the other all the qualitative information. If more information about a specific property is wanted, the evaluation document should be looked up.

The best result is given if the same user or the person that should work with the tool does the evaluation of all tools because of the subjectiveness involved. One other way is to look at somebody else’s comparison tables and summaries and take a decision from that, but then all the subjective opinions and the ranks are from that person’s viewpoint and impressions.

To see which tool that suits best to use, a specific purpose or project must exists. Let us call this the
context. It is the context that decides which properties that are important and should be taken most accounted for and depending on this different tools may be the best. This is done by looking at the information from the evaluation.

This requires human work since one has to decide the importance of the properties and the information from the evaluation must be gone through to be able to decide which tool that is the best. This means that the result is in the hands of the user of the framework to interpret the comparison of the tools.

3.5 Discussion

The vision to develop a framework that could do the evaluation based on factual and measurable properties and be as objective as possible to help people choose tools depending on their needs is hard to achieve but wanted. This to make the evaluation of the tools fair and easy. But the development with all the different viewpoints, what people wanted to know and the area itself makes it very hard to develop this kind of framework. Unfortunately, there is no good way to rank all the properties and give the tool a numeric value or similar that could be compared measurable. This is the case since many of the properties such as R2RML support or SPARQL endpoint is not possible to rank in this manner since they are yes or no properties while some are more qualitative.

A few of the features in the framework are hard to evaluate objectively, since everybody has different knowledge, background and perceive things differently. Some features such as performance is better suited to be tested on its own in a benchmark. To evaluate these kinds of properties could also be very time consuming and not especially important for the big mass of users.

The evaluation should be used as a pilot study to choose tool and that is why it should not be too time consuming using it. Some properties such as maintainability are therefore not added to the framework because that would be a too extensive property to test. Other properties such as documentation and how to use it should therefore be used to give a feeling to these kinds of questions.
4 Evaluation

Three different RDB2RDF tools will be evaluated with the evaluation framework developed in chapter 3. This is done both to test how well the framework works to evaluate RDB2RDF tools with and to find the most suited tool to use in this thesis to publish data.

Since all RDB2RDF tools cannot be tested there is a short description of how the tools evaluated are chosen followed by the result, experiences and opinions received from the evaluation.

4.1 Which tools to evaluate

There are numerous tools possible to use when publishing data. W3C lists seven tools relevant to the area of RDB2RDF and out of these three tools, namely Openlink Virtuoso, Db2triples and D2RQ are going to be evaluated both to test the developed framework and to see which one that should be used in the survey. These tools are chosen since they are popular, have varying functionalities and are open source, where Openlink virtuoso also have a commercial version.

Other tools such as SPARQL2XQuery\textsuperscript{12} aims more against XML datasets and is not the most suited for the survey purpose and neither to test the framework since this would be more as a special case and is therefore not a preferable candidate.

At the time of writing, Spyders\textsuperscript{13} homepage is not up and there is only a few pieces of information on the web of this open source library. Since this is the case it is not relevant to evaluate because there is a big chance that some other tool is better.

The three tools chosen varies in both the way they work and the features they have and are therefore interesting both in the case of how the framework works for different kind of tools and to see which one that suits best to use in the thesis.

4.2 Evaluation of the tools

Each property of the three tools are evaluated by the framework from number of hits on Google to small usage tests. The resulting tables and summaries of the evaluation is presented followed by which tool that is going to be used in the survey and a discussion of the evaluation.

This is a summary created from the most important parts of the evaluation document and tables which can be found in Appendix A.

4.2.1 Virtuoso

Virtuoso is a universal server tool created by OpenLink which includes a lot of different features such as a SPARQL endpoint with SPARQL 1.1 support, possibilities to do both RDF dumps, dynamic

\textsuperscript{12} https://www.w3.org/2001/sw/wiki/SPARQL2XQuery
\textsuperscript{13} https://www.w3.org/2001/sw/wiki/Spyder
mapping and use it as an RDF store. Since it is a universal server it has much functionality that is not relevant to the RDB2RDF area.

Virtuoso is actively maintained with continuous updates and releases. It is a popular tool which is used by for example DBpedia with a nice homepage and interfaces but it is a big tool which takes time to understand.

Virtuoso has an active mailing list and support for both Jena and Sesame. In addition to this the tool has very much own documentation and tutorials in combination with many hits on Google and StackOverflow with varying quality and far from everything is about RDB2RDF. Since Virtuoso have so many different features not everything is relevant for the RDB2RDF area which makes it harder to use for just that purpose. Much of the information, documentation and tutorials are about how to use all the other features which both can be positive and negative depending on what features one need.

There exists both an open source and commercial version where external databases only are supported in the commercial one where the cost varies depending on version, operating system and server. Supports R2RML but also has its own declarative Meta Schema language. In the commercial version there are support for many different databases, otherwise only its own database. RDF support depends on if you want to retrieve data or load data to the triple store, where RDF/XML and N3 among others are supported.

4.2.2 D2RQ

Is an academic open source project published under Apache license which has been very active but its last release was in the middle of 2012. When using the tool a SPARQL endpoint which supports SPARQL version 1.1, a HTML view and a RDF view of the database published is created automatically. D2RQ has free support, a mail list and a lot of Google and StackOverflow hits in combination with its own high quality documentation which is very easy to follow.

D2RQ do not have the most fancy homepage which may be a negative for some users but it still looks serious and is very practical. The tool is focused on converting relational databases and publish it as linked open data. In general very easy to start using and comes in a package with a server which could be used for testing, but Jetty or Tomcat is recommended for production use. The tool should be easy to use with Jena and supports the most major databases such as MySQL and SQL server. It is possible to do both dynamic mapping and RDF dumps on the databases where RDF formats such as Turtle, RDF/XML and N3 are supported.

A great number of datasets are published with it and DBpedia has been published with it before so it is a popular tool which uses its own D2RQ mapping language but also supports direct mapping. Contains a script which can be used to auto generate a mapping file which later can be manually changed.

4.2.3 Db2Triples

Db2Triples is an open source tool that is delivered as a java library developed by the company Antidot. The last release was in December 2013 and it is published under LGPL license.
The overall documentation is not that big, for example there is only 879 hits on Google and 3 hits on StackOverflow when searching on Db2Triples. No public mail list exists but a mail address to Antidot about the program can be found.

Antidot’s homepage is a professional and good looking but it only contains a brief description of the Db2Triples tool. Some dependencies are needed to use the tool such as Sesame and Commons-CLI. It is hard to evaluate the popularity of the tool when it is not possible to publish data with the tool but only convert. It has full support of R2RML and direct mapping, but it has no SPARQL service since it is an ETL tool. The RDF syntaxes support for the materialized graph is Turtle, RDF/XML, N3 and N-Triples.

4.2.4 Evaluation result

As a complement to these text summaries of the tools evaluations, there are two tables created to easily be able to compare the different tools’ properties. Table 4 contains the properties that are based on subjective impressions or numerical values such as number of documentation pages or number of databases supported and some of the properties has both of these ranks. The ranks varies from 1 to 3, where 1 is the best rank.

While table 5 contains the most important information about each property set up in an easy to compare and overviewing table. This to make it easy for a user to choose tool from its context.

To see the whole evaluation document see Appendix A.

<table>
<thead>
<tr>
<th>Documentation and support:</th>
<th>Openlink Virtuoso</th>
<th>D2RQ</th>
<th>Db2Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look and feel / impression of the tool:</td>
<td>F 1, P 1</td>
<td>F 2, P 2</td>
<td>F 3, P 3</td>
</tr>
<tr>
<td>How to use it:</td>
<td>F 2, P 2</td>
<td>F 1, P 1</td>
<td>F 3, P 3</td>
</tr>
<tr>
<td>Popularity:</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Relational database support:</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>RDF serialization support:</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Sustainability:</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 4:** The rank results of the evaluation where 1 is best and 3 is worst. F is the rank received from measurable results and P personal ranks with subjective influences.
Based on the information in the summary evaluation document. A comparison table where the most prominent information is picked out for each property.

<table>
<thead>
<tr>
<th>Property</th>
<th>Openlink Virtuoso</th>
<th>D2RQ</th>
<th>Db2Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational database support:</td>
<td>IBM DB/2, Firebird, Informix, Ingres, Microsoft SQL Server, MySQL, Oracle, PostgreSQL, Progress / OpenEdge, Sybase</td>
<td>Oracle, MySQL, PostgreSQL, SQL Server, HSQLDB, Interbase/Firebird</td>
<td>MySQL, PostgreSQL</td>
</tr>
<tr>
<td>RDF serialization support:</td>
<td>RDF/XML, JSON, N3 N-Triples TURTLE</td>
<td>TURTLE, RDF/XML, RDF/XML-ABBREV, N3 N-TRIPLE</td>
<td>RDF/XML, N3, N-Triples, TURTLE</td>
</tr>
<tr>
<td>Sustainability:</td>
<td>Latest Windows version released 2015-12-09</td>
<td>Latest version 0.8.1 was released 2012-06-22</td>
<td>Last update of the jar was December 2013</td>
</tr>
<tr>
<td>Economy:</td>
<td>Both open source published GNU General Public License Version 2 and commercial, 99.99 USD to 9998 USD</td>
<td>Open source tool published under Apache license</td>
<td>Open source software published under LGPL license</td>
</tr>
<tr>
<td>Extra tools support:</td>
<td>Jena and Sesame support</td>
<td>Jena support</td>
<td>Java library</td>
</tr>
<tr>
<td>Support WC3 direct mapping:</td>
<td></td>
<td>D2RQ mapping language and support for Direct mapping</td>
<td>Supports both R2RML and Direct mapping</td>
</tr>
<tr>
<td>Support WC3 R2RML:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Support of its own language:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPARQL endpoint/query service:</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>SPARQL version:</td>
<td>SPARQL version 1.1</td>
<td>SPARQL version 1.1</td>
<td>-</td>
</tr>
<tr>
<td>RDF dump (ETL), dynamic mapping or both:</td>
<td>Dynamic mapping, some sources claims it is possible to do ETL</td>
<td>Both</td>
<td>ETL</td>
</tr>
<tr>
<td>Server and other things included in the package:</td>
<td>Universal server with web interface and server</td>
<td>Server, recommended to deploy on servlet container i.e Tomcat</td>
<td>-</td>
</tr>
<tr>
<td>Other prominent extra features or good to knows:</td>
<td>Universal server, a lot of different features</td>
<td>Focused on published relational data as linked data</td>
<td>Has been validated by the w3c working group</td>
</tr>
</tbody>
</table>

**Table 5:** The table with the evaluation of each property for the three tools Virtuoso, D2RQ and Db2Triples with information for the properties set up in an easy to compare way. Must be evaluated from the context.
4.3 Which tool is best

One of these three tools are going to be used in the survey and the result of the evaluation is used as motivation to knowit why that tool is chosen. Not all properties are relevant in just this thesis, things like sustainability, RDF serializations support and popularity is of course a plus and evaluated but almost overlooked, but it may be important for another project.

The context for the survey and publish of data could be summarized like this.

Since the focus in this survey is linked open data the tool should preferably be free and open source. There should be a SPARQL endpoint so it is possible to query the data published and SPARQL version 1.1 support is wanted.

There is limited time so it should be easy to get started and preferably easy to use. This involves looking at the ranks of the documentation, support and how to use the tool.

The tool should support MySQL or PostgreSQL so the non measurable property is going to be looked at, the rank on how many different databases that are supported is overlooked. It should be possible to use Jena or Sesame and develop from java. Dynamic mapping is vital but ETL is also wanted since that may be of interest in the survey. It is also a plus if it is easy to deploy it on open source web server such as Tomcat. It should be possible to create links and modify the data and do different test with it.

In this particular context, the most of the important properties are non measurable and therefore table 5 is the most relevant source to look at this time making it more of a look at what kind of properties the tool got more than looking at the numerical ranks. It is possible to see that with this context, one of Openlink Virtuoso or D2RQ is the best suited to use.

When looking at the ranked properties Openlink Virtuoso has overall better ranks, but D2RQ wins the in this case most important how to use property. D2RQ is also best regarding personal rank on documentation and support.

By going property for property in the tables and with the summaries of the tools, it is possible to decide and motivate that D2RQ is the best suited tool to use in this survey. This because it fulfills all the wanted things such as it is a free, open source tool that supports MySQL and PostgreSQL and still is easy to use and aimed to do just these kind of work. What Virtuoso falls on is the must of commercial version to use other databases and that it is a little bit too big and hard to learn using since it is a universal server. It has a lot of documentation but not all documentation is valid and therefore D2RQ is the best. This was discussed and motivated in greater detail with knowit where the tables and summaries played a vital role.
4.4 Discussion

Which tool that was best suited for this survey was chosen by the evaluation done with the framework, but the evaluation of the three tools D2RQ, Openlink Virtuoso and Db2Triples exposed that the framework is not perfect and optimal.

Since the tools are as different as they are the evaluation of some of the properties were unnecessary hard to do. That the tools were very different was already known, but to evaluate them fair were harder than expected. In general it feels like the area of RDB2RDF tools are not mature enough to have a framework like this. When evaluating the different properties, different sources often contains different information which may depend on for example on how old the documentation is. This makes it hard to validate the information and determine what is true and not. The amount of information for Virtuoso and D2RQ was acceptable, but for example Db2Triples is a too small and unknown tool which makes it hard to evaluate properly.

The framework’s primary focus is to evaluate RDB2RDF tools, but when a tool has many other features such as Virtuoso and comparing this with for example the more nished Db2Triples it is impossible to get a fair measurable result when comparing for example documentation the way it is done in this thesis with number of hits and tutorials. Many of the plus points Virtuoso has received on the measurable framework rank, F in table 4, is because of all the other features the tool has and not the features relevant to RDB2RDF. Some information is not even about the tool itself but from other things which makes it hard to evaluate. That is why a personal rank was added to some of the properties to give a more fair RDB2RDF evaluation of the tool and not only use the framework’s measurable results.

To achieve objective and measurable results of each property it would definitely need a more tailored and specified specification of how to evaluate and what to look for when doing this to get good, objective and measurable results. Maybe specify every search phrase and limit what to look at. The context and non measurable properties also leaves a big responsibility to the user doing the comparison right now.

It is in general very hard to be objective, the impression of a homepage, how easy it is doing something and the quality of documentation is all things which are hard to evaluate. There is almost impossible to do something totally objective, there is always a bit of subjective impressions such as look and feel, hard to use and things like that which is very hard to get away from.

Some of the properties were hard to evaluate since it is hard to search for something you do not know exists. The most prominent properties where this occurred were extra tools support, other prominent features or good to knows and what is included in the tool. When you do not know what to search for you will not find it unless you happen to stumble upon it. RDF support was harder to evaluate and not as clear as expected since it differed depending on if it was the syntax in a triple store, an RDF dump or receiving the data which made it hard to compare.

Despite these flaws and problems, the framework definitely helped in choosing tool and a lot of important information is received by evaluating the tools. Knowit was also pleased with the motivation they got from the evaluation which means that the framework’s goal is achieved.
5 Publishing data and survey

The survey’s goal and focus is to, with different tasks and focus from knowit’s interests get answers and knowledge that could help in the future. In this part we are going to use the tool D2RQ that were chosen from the evaluation in chapter 4 to publish data from a relational database. But that is not the only task that will be surveyed, other tasks identified and examined varies from SPARQL queries, publishing the data, develop small applications building upon the linked open data and testing tools and frameworks.

5.1 How the survey is carried out

To do the survey as good as possible a structured and agile work process was used. The first step was to identify all interesting tasks and letting knowit rank and value the importance of them. A coarse time estimation was made from the current knowledge and experience of each task and a backlog was created out of these ranked tasks.

The work has been carried out in an agile and incremental way and it is the tasks in the backlog with the highest priority that is worked with. Meetings with knowit to report progress, how and what to work forward with throughout the survey has also been made.

For each individual task some general comments about the experiences and good to knows received when working with it is documented. Each task has also received a short comment about time and complexity, maturity and agility. In addition to this a broader and more in depth presentation for each task has been made for knowit.

5.2 Properties to document in survey

For each task examined in the survey, the interesting experiences received and to the extent it is possible, a short comment of the properties time and complexity, maturity and agility of each task is documented. A short motivation and explanation of why just these properties are the ones chosen is found below.

Experiences:
The tasks worked with in the survey will receive some general comments about the experiences received. This will include what is regarded good to know or interesting such as possibilities, the future of the area, problems, documentation, help and functionality. This is interesting to know since this is an exploring survey and should give knowledge surrounding the tasks.

Time/complextiy:
Did it take a reasonable time to understand new concepts, develop or work with the task? This is an interesting property if people should start working in the area, to know costs or just to know how much time it takes to do something, especially in a business.
For many tasks time is only relevant the first time one works with it since after that he or she has the
knowledge and time gets irrelevant. That is why a comment surrounding the complexity of the task also is going to be noted to know how complex the task is to do or understand.

**Maturity:**
Linked data is still a new area and everything is not stable and standardized yet. By the knowledge received from the earlier work and development of the framework it is known that much information is old and that there are some problems such as stability of things in the area of linked data. Therefore a comment about maturity of the things surrounding the tasks and stability of for example documentation and help is of interest. This is extra interesting since the survey should give information if this is an area to put more effort into, if it is possible to offer customers and what one can do in the area. If there are problems surrounding this area, it may not be suitable to offer the service yet.

**Agility:**
Today it is very popular to work with agile software development and agile has almost become a buzzword. The most companies are trying to work agile in one way or another instead of the old waterfall way. It has been realized that things changes during longer projects and many applications must be maintained during long times. Customers may want to change or add functionality. This is why this point also is included, where a comment of the possibilities and support to work agile or make changes for the tasks will be made.

In addition to this an overall comment about things that are experienced during this survey will be given.

### 5.3 Survey

The survey mostly consists of smaller tasks which sometimes overlap a bit. In this section the tasks are presented in combination with comments surrounding experience, time, complexity, maturity and agility.

In addition to these tasks, a couple of small semantic applications have also been developed. In these applications, tools, techniques and different parts of the surveyed tasks have been used and put together both to create something useful, get a feeling of possibilities, how good these things work together and complexity building an application. This includes the use of SPARQL, the Jena framework, different API’s, different data and private data published with D2RQ has been used and shown that it is possible to create semantic applications without any problems.

Example of applications developed is an application that visualizes linked data, one that fetches data and shows followable links and literal data from a SPARQL endpoint and one vocabulary application which could be used to find suitable vocabularies when publishing data.

#### 5.3.1 SPARQL survey

SPARQL survey, surrounding things as writing different kinds of SPARQL queries, testing different endpoints, see how long time a query takes and how to write queries from java.
Experience:
SPARQL is similar to SQL which implies that it is a bit tricky to write SPARQL queries, but there exists a lot of help and examples on the web so there is no problem writing and trying queries against endpoints. Simple queries are absolutely no problem writing but if you want to write federated queries, this is a query which asks the query on multiple datasets, or with many specifications in the where clause it is a bit more complicated, similar to more complex SQL queries.

To write and ask queries from java worked fine and was very easy by using the frameworks, where both Sesame and Jena are tested. The possibilities and strength with SPARQL which comes from working against a “global database” and fetch information from many places in a simple way are great, instead of just working against one database. During the survey there have been some performance issues and stabilization problems which seems to be quite common. It is hard to say where the problem lies, if it is the web server, the amount of data, congestion or the SPARQL queries themselves that are creating this problem. This is itself a very big task to examine and lies beyond this thesis.

Time/complexity:
SPARQL is quite similar to SQL and therefore it is no bigger problem to learn for a person with knowledge about SQL. A query could of course be made as complex as you want, but just to use and learn for everyday use there are no problem if you know how the data you query looks like with name and which vocabularies that are used. Since different endpoints supports different features there are sometimes time consuming problems that are hard to understand, this because the query should work but does not because the feature is not supported by just that endpoint.

Maturity:
The endpoints support different features, there seem not to be a guarantee that all features in SPARQL or standards are fulfilled. Different performance where a query can be fast one time and receive a time out the next is a problem and it is hard to know if it depends on stability issues or on the fact that it is heavy workload sometimes. This seems to be a common problem since same information is found on the internet on forums.

To write and execute queries from java using the frameworks Sesame and Jena was very simple and straight forward. There is solid documentation, the functions and possibilities expected and wanted exists. So there is no problem developing a semantic web application from that point of view.

The impression from the SPARQL language itself is that it is mature and worked through so there it is nothing to complain about. When writing queries there exists shortcuts to for example use the same subject and build on that. There are good documentation and in the future when everything else in the area also have been more mature, the future and possibilities looks bright.

Agility:
Regarding agile development, if the vocabularies or data change, then the queries used in an application may not work anymore. But if one has the knowledge about the changes and can rewrite the queries and redeploy the application it should work. But there are risks about this if data or queries get changed.
During the survey it is not found how to write good agile queries from java, such as user written
queries that may change all the time. The problem is that you have to hard code for example vocabulary or parameter names to receive the data, which means that you have to know information about that in advance. There might exist a good solution for this but it is not found at the time of writing.

5.3.2 Vocabulary survey

Vocabulary survey including the status of the vocabulary area, how to create a new vocabulary and look through what the recommendations are. Is there something one has to think about and does it exist any help in the area?

Experience:
The general experience when examine the vocabulary task is that the most in the area is thought of and that it is kind of complete. There are a lot of help, many good and used vocabularies out there and many things such as change and replication are fixed in a thoughtful way. The one thing that would be preferable is a tool to one click publish the vocabulary in a good best practice way.

Time/complexity:
Since there are many popular vocabularies already in use, a lot of help, documentation and examples to look on already exists. To create a vocabulary is easy and there is no problem or complexity doing it. Potential problems could be configuring the server and use a namespace you control when publishing which is a more technical part, but just to create a vocabulary is easy. Apart from this there are no complexity or technical problems in creating a vocabulary.

Maturity:
The whole area of vocabulary is very mature. There are well defined and popular vocabularies such as foaf\[14\] and dublin core\[15\] that one can use. When publishing data there are websites such as the linked open vocabulary website\[16\] that can help to search for classes and terms that could be used to mark up the data. In the case that one has to create its own vocabulary there are plenty of help and documentation of how to do this and what to think about. In addition to this there are a lot of already popular, used and published vocabularies one can look at and learn from. You can also find guidelines and best practises in books and on the internet.

Agility:
Regarding agility people seem to already from the beginning have known that it is going to be copies and replications of already defined properties and classes. To keep good quality the vocabulary created have to be maintained with adding some links between vocabularies. If a property in another vocabulary is found one should add links such as owl:equivalentClass or rdfs:subPropertyOf to mark this. To do changes or adding things to the vocabulary is therefore not a problem but may of course be a bit time consuming because of the maintenance.

\[14\] http://xmlns.com/foaf/spec/
\[15\] http://dublincore.org/documents/dcmi-terms/
\[16\] http://lov.okfn.org/dataset/lov/
5.3.3 Framework survey

Small comparison and test of the two java frameworks Sesame and Jena. Evaluate and try to compare and look at functionality differences.

Experience:
Both of these java frameworks are used to develop semantic web and linked data applications and recommended from different sources. To the extent this task is surveyed there are no bigger differences in functionality discovered between the frameworks except for the OWL API that Jena has. The difference between the frameworks are discussed on forums as well but there are no clear comparison between them.

The implementation tests made with each of them have worked good and the API documentation too. When problems occurred help and answers where slightly easier to find for Jena, which for example has 3806 results on StackOverflow while Sesame only has 1160 which of course makes it more likely to find the answer one is looking for. But both frameworks are used and recommended from different sources when an application should be developed. The opinion right now is that it is a matter of taste which one to use, with a small advantage for Jena since there seems to exist a little more help and the rank received from db-engines [25] which ranks Jena one place higher than Sesame.

Time/complexity:
With a computer science background or knowledge of how to program in java there are no problem using the frameworks. Good tutorials to get started, well documented API’s, documentation and straight forward coding. Could be compared with many of the other frameworks used when developing java applications.

Maturity:
The frameworks have been used for a relatively long time and are maintained and developed by RDF4J which is an Eclipse fork and Jena is an Apache framework so they are backed up by a trustworthy company and organisation respectively. Both are actively maintained and developed which are a plus for the future.

Due to the semantic web wiki, [27, 28] Sesame’s status was mature while Jena had the status of stable in 2012 which is the when the page was updated so this is a couple of years ago.

5.3.4 Publish relational database survey

Publish relational data as linked open data with the D2RQ tool. Refine and test how to do different changes in the mapping such as link to other data, different data formats, add data or information to the mapping and database during runtime.

Experience:
To publish an existing database that is supported by the D2RQ tool is very easy by using the script to generate the mapping and publish it. The problem with this is the quality, there are no links to any external data, it is only the “self created” vocabulary used and only the most necessary things created. It is also possible to create the mapping by hand from the beginning but it is recommended to create the mapping first and then modify it.
There are good documentation of the tool, scripts, examples and mapping language so the most things one want to know can easily be found. Can create a WAR file with Apache Ant to easy deploy it on the web.

The HTML view and SPARQL endpoint included when running the d2r server makes it easy to do trial and error to learn more how everything work such as try to change the mapping file and then see what happens in the linked data view.

It works good to add live data to the database to an already existing table, everything gets updated in real time. This is a benefit of using the mapping instead of materializing the RDF and store it in a triple store, which also is possible. There are however problems when adding a new column or table to the database during runtime.

Since it is often better to store an image in another place than in the relational database and just store the URL to it instead, it is not tried to publish an image that is stored in the database. There is a downloadMap that could be used for BLOB but it is not tested in this survey.

**Time/complexity:**

Just to publish a relational database as linked data with no quality requirements is really easy and the HTML and RDF views helps to learn more too. This is something that could be done fast if the only requirement is to publish the data. The only knowledge needed is more or less how to use the terminal to change directory and start a script.

To make changes such as changing vocabularies to a more used such as dublin core or FOAF, add comments or language tags are not any problem either since everything is documented and easy to follow. Publish dates, integers and strings are fixed automatically by the tool and script.

The mapping looks complex and hard to understand from the beginning. But with the documentation and when one starts working with it, it is easy to understand.

**Maturity:**

The D2RQ tool does not have any graphical interface to help changing the mapping. It does not support R2RML but there is support for Direct mapping and its own D2RQ mapping language works well and is well documented. DBpedia has been published with it and for example linkedmdb are published with this tool so it is one of the more mature tools in the area. This is relative though, linked data is just a few years old and this tool is not as mature as many other in different areas.

The D2RQ tool has not been updated for a while so the future of the tool is unknown, but it is still used, popular and works well.

**Agility:**

Changing the mapping during runtime is not possible, such as adding a property, static or not, since it is not recognized in the published data which means that you have to republish it to update the changes.

Adding an attribute to a table or a new table to the database also implicate that you have to republish everything to get the change.
Adding a new row to a table in the database, such as adding a country or measured sensor value, is directly shown in the published data, which is great for live and frequently updated data.

To add links for a specific entity in the database was hard and at least during the survey no better way to do this was found. This have to be done manually and there is much work to do it if many links should be added. It has to be done by creating a new table in the database, manually add the data to it, change in the mapping file and republish. So if not another way is found this is best suited for static data where all of this can be added when published even though it implicates a huge amount of work. To add static links which is connected to each entity in a table is however easily done with the mapping file.

5.3.5 RDFa and JSON-LD survey

What is RDFa and JSON-LD, how does it work and how does one write it? Is there any benefits using it or problems? How does the future look like?

**Experience:**
RDFa and JSON-LD is the area in linked data where it feels that it is the greatest possibility to become popular and used. This is a really interesting area which has the possibility to also become commercial. It is an extension to HTML used to add semantic and making the data machine readable to otherwise only unstructured data in an easy way and it is already used. It is a way to publish linked data if the publisher only has the possibility changing in the HTML. But there are also possibilities to use it in other formats such as XML.

By using RDFa or JSON-LD the web pages get more useful since search engines such as Google can use it and understand the data better and build other things such as rich snippets. The major search engines support the schema.org vocabulary. There are discussions surrounding that this field is the future for the search engine world.

It is related to web development which means that it is a big area but probably worth putting some time into since it is an interesting area for the future. Both are W3C recommendations and have the same goal as microformats and microdata but there are differences in how they work. There exists community webpages with information, help and examples.

**Time/complexity:**
Since it is an extension to HTML there is no bigger complexity to start using these kinds of markups. There have been some discussions on forums that microdata is simpler and easier to use then RDFa, but this is subjective and the comparison is often made against RDFa 1.1. There exists a subset of RDFa 1.1 called RDFa lite which due to the documentation is enough for 80 percent of the cases, takes 15 minutes to work through and a day for a web developer to learn. JSON-LD is very similar to JSON which many already is familiar with. RDFa and JSON-LD are W3C recommendations and microdata is only a working group note and therefore overlooked in this survey. Since RDFa is an RDF serialization which is a cornerstone in linked data some people are promoting that.
Maturity:
By looking through documents of RDFa syntax or other recommendations, they are updated in the beginning of 2015. Google are in the process of adding more JSON-LD feature support. The area is fully possible to use but changes and developments could probably be expected. Big search engines such as Google and yahoo understand the schema.org vocabulary and there is a movement where people encourage to use structured data markup.

Big sites such as BBC music is already using RDFa and JSON-LD and since version 7 Drupal support drupal entities in RDFa.

Agility:
There are no problem working agile with these techniques, neither in developing new websites or adding to already existing ones with these kinds of functionalities. There are on the other hand not any special help or support to work agile either, but there should be similar to working with HTML in this matter. Multiple vocabularies could be used to express more complex things.

5.4 General impressions of the area
The experience from the survey of the tasks and the development of the framework is mixed and varies much depending on task.
It is in general very much information, documents and websites that are very old which makes it hard to know what actually is relevant today and many things are superseded. The biggest hype seem to have been for a few years ago and it is very common that links and sites do not work and are removed. But, there are still updates on tools, new answers and questions on forums, there exists published data and many SPARQL endpoints, vocabularies, frameworks and community sites on the web, for example the semantic web forum which one can use. On the other hand there are also some forums and SPARQL endpoints which have been down and not usable during this thesis which indicates that they are not stable.

The parts of linked open data themselves such as RDF, SPARQL or creating vocabularies are not complex or difficult. The hardest thing feels like the publishing part with the content negation, making it readable for both people and machines but sometimes this comes almost free. For people with no or little computer experience this is definitely the biggest problem.

The area is still a bit messy, it would be good and helpful if some real standards were set instead of all the choices such as being able to have 303 URIs or hash URIs, different publishing methods, different mapping languages, different RDF serializations and so on. This could of course be positive to have all of these choices, but it makes it hard to know how to do things and what the right way doing them are. This is a potential problem that the learning curve becomes too steep for many new users which may slow down the evolution of the area. Today it is a lot in movement in many different areas which is good for the future, but it is also making it hard to focus on things.

During this thesis some other tools have also been tried out in a more informal way. The quality and how they work varies, for example the functionality may be good and they work for testing, but may

17 http://www.bbc.co.uk/music
18 http://answers.semanticweb.com/
not work to create a real world project with because of problems with for example concurrent transactions.
There are tools which help to link data sets such as Silk framework which is not used in this thesis but would be interesting to use and may solve some problems.

There are still many persons that have never heard about linked open data and it would be needed to get more known to expand. This is also a problem that could lead to problems making it interesting to customers or convince people using or publishing data since they do not know or see the benefits with it. Today RDFa and JSON-LD feels like the hottest and most interesting area in linked open data and also simple to use. But it would be interesting to do another survey regarding what kind of applications, identify areas and companies that could be interested in using linked open data.
6 Conclusions

This thesis had two goals stated in 1.3 Purpose and goals. An RDB2RDF evaluation framework is developed, tested and used as motivation when choosing tool. The second goal has been achieved via the survey containing publishing of relational data as linked open data, in addition to many other tasks ranked by knowit.

A greater knowledge in the area of linked open data, SPARQL, tools and techniques in the area is also received.

6.1 Conclusion for framework

The goal for this thesis described in 1.3 Purpose and goals to develop a framework to help anyone to choose RDB2RDF tool, do an evaluation of at least two tools and use it as motivation for the choice of tool is achieved. The evaluation framework is developed by taking different persons’ thoughts and sources into account developing it to be suitable for everybody and three different and diverse tools are evaluated to test it in this thesis. The evaluation helped and was used to choose tool and motivate to knowit for the survey and a lot of different information was received hence is the goal achieved.

The development of the framework has also given a deeper knowledge of the area especially of RDB2RDF but also linked open data in general and SPARQL. As discussed earlier in this thesis there may be possibilities to make further refinements to the framework depending on objectively and measurable values. It is hard to avoid this, especially when the properties people would like to know about are subjective such as look and feel.

One drawback of the framework using all of these different sources is that it may not be the best for anybody of the possible users of the framework, but it suits everybody good. This is the case and cost of developing a framework which should work and be relevant for all parts from companies to individuals.

The area of linked open data and especially RDB2RDF do not really feel mature to have a framework like this yet. There are some problems due to documentation and big variations of the tools which makes it hard to do a good and fair evaluation. This could though change in the future if the area gets more popular and mature. Despite this, the evaluation made it able to pick out which of the tools that was most suited to use in this thesis and knowit was pleased with the motivation they got from the evaluation.

6.2 Conclusion for survey

Linked open data is a really exciting field for the future especially with everything surrounding internet of things, smart cities, the fact that we get more and more connected to the web and the data there. I think it is going to grow, but it is still relatively new and the movement is still small. A vital part to get the area to grow is to lower the learning curve by making it easier to use and set more standards to follow. There exists several different techniques and tools right now which is positive,
but on the other hand is that making it hard to focus and know what to start using.

The survey has shown that the different tools and techniques in the area varies very much in for example maturity and agility. Some tasks such as RDFa and JSON-LD feels very hype and right now have a bright future while SPARQL endpoints varies in quality and stability, vocabularies are thought through and mature while there are some problems refining the data when publish a relational database. This means that some areas are more interesting right now than others but it is impossible to know what is going to happen in the future.

Because of all the problems mentioned with for example stabilization problems I would not recommend to start offering commercial services surrounding linked data yet, but maybe offer it just like an additional feature to be active in the area, spread the word and be first and leading in the area. Since I think it is something which is going to keep evolve and expand it is definitely an interesting and fun field to be involved and apart of and it may pay off in the future.
All the tools exists to, for example publish relational databases, but it would be good with more standards and wait to the area is more stable before offering this kind of services as something more than an extra feature.

More people must hear and know about linked open data and the benefits of it. Things such as the SPARQL endpoint stabilization problems, give specific entities links in the a database and more standards are needed to be fixed to really be able to grow and mature. There are more tasks to survey in the field, but due to this thesis it is an interesting field which hopefully will grow and therefore it is good to be active. If the future in the area will include RDB2RDF tools or where the development is going is hard to say, the area is in general not mature enough but it has all the possibilities to be in the future. It is therefore interesting to follow the progress and be a part of it and if possible add linked open data to projects just to spread the word, because it is not particularly complex or time consuming doing that.

6.3 Restrictions

It would of course be preferable to evaluate all the different RDB2RDF tools to get a for now sufficient and whole comparison table to see which tool one should choose. Due to time restrictions this was not possible to do but it is possible to add these extra tools into the evaluation at any time or do new a evaluation of all the tools.

All of the tasks in the survey could of course be examined more in depth to get answers on things such as the variety in SPARQL performance or do a real benchmark test. Create a vocabulary from a real world example and try differences with hash or slash and similar. The survey possibilities are almost infinite, but in this thesis it was prioritized to look on a number of tasks instead of only one or two tasks really in depth which have left some questions open in the tasks.

36
7 Future work

Right now there are only three tools evaluated with the framework, more tools should be evaluated making it possible for a user to look at the evaluations and tables to compare and choose from the survey made if wanted. There are also possibilities to expand or refine the framework even more by making more interviews, look at other sources and viewpoints to maybe make it even better. If a new tool comes out or an update of a current one is made that could be added to the evaluation.

Since there are a number of subjective elements in the framework it is also possible to refine and create a recipe on how to evaluate each property to make it more objective and measurable and make the ranking more specified if that is wanted.

Regarding the survey it is possible to keep working with the tasks that still are in the backlog. All of the tasks may be examined even more in depth or new tasks may be come up with. Create even bigger applications, test different or do more extensive tests on techniques and tools for example against each other such as Jena and Sesame or RDFa and JSON-LD. It is also possible to make a new survey regarding companies and the use of linked data to see the interest from them and talk to them about what kind of applications that could be done in reality with it.
References


Appendix A

Evaluation summary document

Evaluation conducted with the first version of the evaluation framework developed in a master thesis to evaluate and help people to choose RDB2RDF tool to publish relational databases as linked open data. The framework is used to evaluate three tools, namely Openlink Virtuoso, D2RQ and Db2Triples. These three tools distinguish in several ways which tests the framework in a good way to see strengths and weaknesses.
Evaluation framework

The evaluation framework used where each property is evaluated if possible.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extra tools support</td>
<td>Relational databases support</td>
</tr>
<tr>
<td>SPARQL endpoint / query service</td>
<td>RDF serialization support</td>
</tr>
<tr>
<td>RDF dump (ETL), virtual mapping or both</td>
<td>Support WC3 direct mapping</td>
</tr>
<tr>
<td>Other prominent extra features or good to knows</td>
<td>Support WC3 R2RML</td>
</tr>
<tr>
<td>Server and other things included in the package</td>
<td>Support of its own language</td>
</tr>
<tr>
<td></td>
<td>SPARQL version</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation and help</td>
<td>Popularity</td>
</tr>
<tr>
<td>Look and feel / impression of the tool</td>
<td>Sustainability</td>
</tr>
<tr>
<td>How to use it</td>
<td>Economy</td>
</tr>
</tbody>
</table>

How to evaluate

The features and ways to evaluate these may be further developed or personalized in the future by the experience from this evaluation or other ideas. But this is how the properties are evaluated this time and should only be seen as a first proposal of how the properties can be evaluated for each tool. Small differences may occur for the tools or if something is not found the information is left out.

Documentation and help:

Is evaluated by looking at number of hits on Google, Youtube and StackOverflow when searching on the tools name and other phrases connected with the tool. Does the tool have its own documentation on its homepage or similar.
Are there any specific forum to ask questions on or a maillist.
Are there any tutorials of how to use the tool or possibility to get support.
Receives a measurable numeric rank by the number of the number of hits.
Receives a personal rank by human work and subjective impressions of quality and similar.

Look and feel / impression of the tool:

To rate this a small look up and hands on test is made. This to see how easy it is to get started with the tool and what kind of first impression the tool gives.
What kind of website and homepage does the tool have are looked up.
Is it easy to download, does it look good, what is the first impression when start using the tool is also considered.

These different tests and look ups adds up to a subjective rank depending on personal opinions.
How to use it:
To rate this a small look up and hands on test is made.
How should one do the setup and start using the tool?
Has it a graphical interface, CLI or must it be used programmatically?
How is it started and how to use the tool.
Receives a subjective rank depending on this.

Relational databases support:
List all the databases supported by the tool.
Objective rank on number supported databases.
List all supported databases in the comparison table.

RDF serialization support:
Which RDF serializations are possible to use when doing ETL or receiving data?
One objective rank depending on the number of different serializations supported.
List all supported serializations in the comparison table.

Popularity:
This is evaluated by trying to see how many datasets that are published with the tool.
Does any big dataset such as DBpedia use the tool? Is there any research where the tool is of interest, look up by the UB library search. If it is possible, see how many downloads the tool has. By this try to give an objective rank of how popular and used the tool is.

Receives a rank from the result of how many or big datasets that are published with the tool and how much research and similar that is made.

Sustainability:
From the related work part that features:
When was the latest update? Number of updates? When was it released? Which company or person is the owner and creator of the tool? This could receive two ranks, where the measurable rank can depend on when the latest release and what kind of company the tool is backed by. While it also could be used as a look up if needed.

Economy:
Open source or license cost? Commercial or free support? Pay to get additional features?

Extra tools support:
Possibilities or support to use API e.g Jena or Sesame?

Support WC3 direct mapping: Support WC3 R2RML: Support of its’ own language:
What kind of mapping language does the tool use? R2RML or its own?
SPARQL endpoint / query service:
Does the tool include a query service or SPARQL endpoint so one can query the data directly?

SPARQL version:
If it supports SPARQL, which version is supported, 1.0 or 1.1?

RDF dump (ETL), virtual mapping or both:
How does the tool do the mapping? RDF dump (static data materialisation), virtual mapping (dynamic mapping) or both?

Server and other things included in the package or have to develop by yourself:
How to deploy the data, is there a server included or do you have to use or develop something by yourself or use another tool to do this?

Other prominent extra features or good to knows:
Something extra or special one should know about the tool? Any benefits or drawbacks worth mentioning.
Openlink Virtuoso

Openlink Virtuoso is a universal server with plenty of functionality where one is to publish relational data as linked open data. There are both an open source and commercial version of the tool. Virtuoso creates a SPARQL endpoint, supports R2RML and is popular, but since its has many functionalities it is a bit harder to learn and use.

**Documentation and support:**

Number of hits when searching on Google with the search phrases:
- Openlink virtuoso gives 331 000 hits.
- Virtuoso linked data gives 353 000.
- Virtuoso universal server gives 145 000 hits.
- Openlink virtuoso tutorial 15 200 hits.

By just Google the phrase Virtuoso gives 19 500 000, this since there are a lot of other things with the name so this have to be overlooked.

Number of hits when searching on phrases on Youtube:
- Openlink virtuoso 52 hits.
- Virtuoso linked data 363 hits.
- Openlink virtuoso tutorial gives 86 hits.

*Far from all are hits are relevant or about the tool virtuoso at all.*

Searching on phrases on StackOverflow to get question and answers:
- Virtuoso 788 results.
- Openlink virtuoso 74 results.

Virtuoso has four different mailing lists hosted by SourceForge. The most relevant is the one for the users which is for user support and are very active with 7671 results when the evaluation is made. It started in 2006 and the last post was in 2016.

There are some documentation at the github page about packages, how to install, disk space, how to run and similar information about the tool.

Some of the documentation is only relevant for the open source version and some for the commercial version.

Virtuoso has their own tutorials, there are a number of tutorials on their homepage but not all are about the area of RDB2RDF. Some are about for example SQL, XML and data replication and it is hard to estimate how much that are relevant in the area. But the fact that there are tutorials is a plus. *Have to be considered that not everything with Virtuoso are about RDB2RDF.*

Virtuoso’s own documentation is massive, it is almost too much since it gets very hard to embrace and use. Some documentation is old and not up to date so it does not work which only leads to confusion. But by searching through it and put some effort searching in it the most cases or the things you would like to have help with are documented. The biggest question is if the documentation is valid or if it is out of date for the current version. One more thing is that not everything is about RDB2RDF which is the focus of this evaluation.
Links:
Virtuoso has four different mail lists:
List of mailist: http://virtuoso.openlinksw.com/dataspace/doc/day/wiki/Main/VOSMailingLists
User mailist: https://sourceforge.net/p/virtuoso/mailman/virtuoso-users/
Github page: https://github.com/openlink/virtuoso-opensource
Virtuosos own tutorials: http://demo.openlinksw.com/tutorial/
Virtuoso documentation: http://docs.openlinksw.com/virtuoso/

Grade:
Framework 1
Personal 2

Look and feel / impression of the tool:
The first impression when entering the homepage is that it looks and gives a very professional impression, what most people would expect. It looks serious, has nice colours, images, layout and links to everything you can want.

The tool itself, the Openlink Virtuoso conductor has a nice webpage interface when using it after starting it through the terminal. So the first impression, look and feel of the tool is good and the most users would definitely have an impression that this is a tool that they would like to use. The problem is that the tool is so massive that it may scare many users with all the different possibilities. The impression is even though the nice interface that it is complicated and takes time to learn. There are many different tabs, packages and settings which may scare away people and may be complicated to learn for many users.

Since there are two different versions, one commercial and one open source it can get confusing with not supported features in some of the tabs which drags down the grade of the impression.

Links:
Virtuoso homepage: http://virtuoso.openlinksw.com/

Grade:
Framework 1
Personal 1

How to use it:
No problems to start it from the terminal to get it to run and have the conductor up and running when finding documentation of how to do it. There is no graphical interface but since it is so easy to start it is not needed.

Unnecessary hard to start using the tool, had to install some packages, follow documentation which was out of date, logging in and more. Managed to deploy a toy example after a while but ran into some trouble due to documentation that was out of date. Hard to know what the different packages and cartridges do so you get a feeling that it is complicated and it is hard to understand how everything works. Since I only tested the open source version I do not know how hard it is to deploy an external database.
Should be possible to use Jena and Sesame provider to use it programatically.

Had problem with getting it to run, followed some out of date documentation, download packages and more to get a toy example to run.

Grade:
Framework 2
Personal 2

**Relational database support**
Due to documentation do Openlink Virtuoso support
IBM DB/2
Firebird
Informix
Ingres
Microsoft SQL Server
MySQL
Oracle
PostgreSQL
Progress / OpenEdge
Sybase

It is only possible to connect to external databases with the commercial version. In the open source version it is only Virtuoso’s own database that is supported.

Grade:
Framework 1

**RDF serialization support:**
RDF data may be retrieved in:
RDF/XML, JSON or N3 syntaxes.

It is possible to load N3 / N-Triples and RDF/XML RDF Data, TURTLE serializations into Virtuoso triple stores.

Grade:
Framework 1

**Popularity:**
DBpedia which is the biggest dataset right now is published with Openlink Virtuoso. Other datasets published with Virtuoso is due to [https://www.w3.org/wiki/SparqlEndpoints](https://www.w3.org/wiki/SparqlEndpoints) BBC programmes and music and data.gov among others.

The number of downloads at SourceForge are 183 the week the day the evaluation is done. It is also possible to download it on other places and there are commercial licenses too so this is not 100% representative. Unknown how many commercial licenses there are.
Searching on Openlink Virtuoso on UB library search:
- Openlink Virtuoso gives 162 hits on articles, conference articles, thesis.
- Virtuoso linked data gives 39 hits.

Links:
Source of SPARQL endpoints: https://www.w3.org/wiki/SparqlEndpoints
How many downloads at: https://sourceforge.net/projects/virtuoso/
Ub search: http://www.ub.umu.se/sok
Github: https://github.com/openlink/virtuoso-opensource

Grade:
Framework 1

Sustainability:
Actively maintained by the Openlink Software company with continuous releases and updates. The latest version for Windows was released 2015-12-09. It has been around for a long time, on github a version 4.5.0 was released on 11 april 2006. The Virtuoso project was born in 1998.

Links:
Source: http://www.openlinksw.com/press/
Github release: https://github.com/openlink/virtuoso-opensource/releases?after=v4.5.4

Grade:
Framework 1

Economy:
It exists both an open source and a commercial version of the tool. The license of the open source is a GNU General Public License Version 2.

There are different costs and multiple choices for licenses depending on version, operating systems, server, desktop and more. The price is ranging from 99,99 USD to 9998 USD. It is only possible to connect external databases with the commercial version.

Technical support to all users but users with license may purchase support contract to get higher priority and better help.

Links:
Virtuoso licensing offers: http://virtuoso.openlinksw.com/offers/
Info about support: http://support.openlinksw.com/supportweb/

Extra tools support:
There are possibilities to at least use Jena and Sesame. Do not know how well they work or how complicated they are to use but the providers exist.
The tool contains VAD packages, Sponger middleware, cartridges and other things that is used in the
tool.

Links:
Link to Jena provider: http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtJenaProvider
Link to Sesame provider: http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtSesame2Provider

Support WC3 direct mapping: Support WC3 R2RML: Support of its’ own language:
Due to the information found there are support for R2RML but not Direct mapping.

Virtuoso has its own declarative Meta schema language, which can be further read about by following
the link below. Meta Schema language is often refereed to as “linked data views”.

Links:
Read more about Meta schema language http://virtuoso.openlinksw.com/whitepapers/relational
%20rdf%20views%20mapping.html

SPARQL endpoint / query service:
Yes SPARQL endpoint / includes SPARQL support.

SPARQL version:
SPARQL version 1.1

RDF dump (ETL), dynamic mapping or both:
Main focus is to create dynamic mapping that reflects changes on data (linked data views). Due to the
sources looked at it is not Virtuosos main focus to do data materialisation but that it is possible.

Server and other things included in the package or have to develop by yourself:
Virtuoso Universal server, includes many different features such as the Conductor web interface. It is
a universal server and a web server is included in the package.

Links:
Homepage: http://virtuoso.openlinksw.com/

Other prominent extra features or good to knows:
The Universal server contains web application server with SOAP and Restful. RDF store, XML
features, data management and document file services. Look at the homepage to see more information
about this.

SPASQL (SPARQL inside SQL functionality).
D2RQ

D2RQ is an open source academic platform to publish relational databases as linked data. Uses its own D2RQ mapping language to do the mapping, integrates seamlessly with Apache Jena and contains a script to create RDF dumps. Popular tool with good documentation.

Documentation and support:
Number of hits when searching on Google with the search phrases:
- D2rq gives 306 000 hits.
- D2rq server gives 15 900 hits.
- D2rq linked data gives 27 100.
- D2r linked data 70 100.
- D2r server gives 75 700.
- D2rq tutorial 8530.

Number of hits when searching on phrases on Youtube:
- d2rq 107 hits,
- d2rq linked data 28 hits.
- d2rq tutorial 28 hits.
Far from all are relevant or about the tool D2RQ at all.

Searching on phrases on StackOverflow to get question and answers:
- D2rq 75 results
- d2r 156 results.

D2RQ has a user mail list on sourceforge which is active and used. It started in 2005 and last post was in 2015. There are a bunch of questions and answers there, when the evaluation was done it was 2153.

Have a page at github with some documentation about license, ant builds, how to run, deploy and some issues and links.

Short but still enough and solid documentation, easy to navigate and use. Overall straight forward documentation which really helps when looking at it.

Links:
Subscribe to maillist at: d2rq-map-devel@lists.sourceforge.net
Maillist https://sourceforge.net/p/d2rq-map/mailman/d2rq-map-devel/
Github: https://github.com/d2rq/d2rq
Homepage with documentation: http://d2rq.org/
Links to other resources: http://d2rq.org/resources

Grade:
Framework 2
Personal 1
**Look and feel / impression of the tool:**
Kind of simple homepage, nothing fancy and selling, but it still looks serious. Have all the information needed, links to other places and it is very easy and straight forward to use and find information. No problem with downloading the tool.

By following the documentation it is easy to get started, do the mapping, create RDF dump or whatever wanted. May look scary for people not used to using the terminal when setting up and start the program but since it is very straight forward from the documentation which is solid there is no problem.

Fast to publish linked data, get results fast with an easy to use website. Is started from the terminal but it is no problems doing it.

When running the tool there is a webpage with a HTML view, RDF view and SPARQL endpoint of the database data. These are maybe not that beautiful but very practical and is a good place to start at. The webpage also contains some information.

**Links:**
Homepage and place to download: [http://d2rq.org/](http://d2rq.org/)

**Grade:**
Framework 2
Personal 2

**How to use it:**
Need to manually change and write in the mapping file, feels scary at first but are no bigger problem when reading documentation of the mapping language where everything is documented. No graphical interface exists.

Should be possible using it programmatically from Jena with no bigger problems and there is documentation for it.

**Links:**
D2RQ mapping language: [http://d2rq.org/d2rq-language](http://d2rq.org/d2rq-language)
D2RQ jena: [http://d2rq.org/jena](http://d2rq.org/jena)

**Grade:**
Framework 1
Personal 1
Relational database support
Due to documentation do D2RQ support
Oracle
MySQL
PostgreSQL
SQL Server
HSQldb
Interbase/Firebird

Other databases may or may not work and they are interested in reports about this.

Grade:
Framework 2

RDF serialization support:
Supported syntaxes when doing an RDF dump are:
TURTLE, RDF/XML, RDF/XML-ABBREV, N3, and N-TRIPLE (the default). N-TRIPLE works best for large databases.

The Generated mapping file is done in TURTLE.

Links:
Source: http://d2rq.org/generate-mapping

Grade:
Framework 1

Popularity:
First version of DBpedia was published with D2RQ. Other big datasets such as Linked movie database and due to https://www.w3.org/wiki/SparqlEndpoints there are a bunch of other datasets such as DrugBank and GeneOntology database that are also published with D2RQ.

The number of downloads at SourceForge the week this is done is 8. Downloads of the tool are done at other places too so this is not 100 % representative.

Searching on D2RQ and d2r to get number of publications and articles in Ub library search.
- D2Rq gives 20 hits on ub.
- D2r server 104.
- D2rq linked data 15 hits.

Links:
Source of SPARQL endpoints: https://www.w3.org/wiki/SparqlEndpoints
How many downloads at: https://sourceforge.net/projects/d2rq-map
Ub search: http://www.ub.umu.se/sok
Grade:
Framework 2

**Sustainability:**
Has been very active but right now the latest version 0.8.1 was released 2012-06-22 which was a while ago. Is an academic project open source. Was first released 2004-06-22 with v0.1 and it has a very active community.

Maintainers, Richard Cyganiak
Project founder, Chris Bizer
A number of contributors and contributing organisations.

Links:
Release notes: https://github.com/d2rq/d2rq/blob/master/RELEASE_NOTES.md

Grade:
Framework 3

**Economy:**
Free open source tool, published under the Apache license. It is possible to contact the development crew through the issue tracker on github if needed. Free support is possible to get.

**Extra tools support:**
Works with Jena but do usually only work with the Jena version that it is shipped with. Had a Sesame support but that is removed since version v0.8.8.

**Support WC3 direct mapping:**
Uses its own D2RQ mapping language, which is similar to R2RML but has some distinctions. The latest version v0.8.1 has support for Direct mapping.

**SPARQL endpoint / query service:**
Yes SPARQL endpoint that you can query the relational database through.

**SPARQL version:**
SPARQL version 1.1

**RDF dump (ETL), virtual mapping or both:**
Both, dynamic virtual mapping through its own D2RQ language that reflects changes on underlying data, contains a script which creates an RDF dump that can be loaded into a triple store.

**Server and other things included in the package or have to develop by yourself:**
Contains a server that can be used to test with but it is recommended to deploy the d2r server on a servlet container like tomcat or jetty server for production use.
Other prominent extra features or good to knows:
Focused on publish the relational data as linked data. Access non-RDF data through Jena.
Db2Triples
An RDB2RDF tool, packaged as an open source Java library developed by Antidot. Supports PostgreSQL and MySQL and the standards R2RML and direct mapping. Is an ETL tool with brief documentation.

Documentation and support:
Number of hits when searching on Google with the search phrases:
- db2triples gives 879 hits
- db2triples linked data gives 933
- db2triples tutorial gives 216
- db2triples rdb2rdf gives 351
- antidot db2triples gives 29

Number of hits when searching on phrases on Youtube:
- db2triples gives 0 hits
- antidot db2triples 0 hits

Searching on phrases on StackOverflow to get question and answers:
- db2triples 3 hits
- antidot db2triples 0 hits

Db2triples does not have any public maillist.

Has a github page for the tool where there are some documentation about how to get and execute the jar, dependencies needed and contacts to a mail. (opensource@antidot.net) Not as much information as many would want to have to start using the tool.

Just a small presentation of the Db2Triples on Antidot’s website, “contact the mailadress” to get more information so there is no clear documentation or tutorials.

Links:
Github page: https://github.com/antidot/db2triples
Antidot homepage: http://www.antidot.net/en/
Antidot db2triple page: http://www.antidot.net/en/ecosystem/db2triples/

Grade:
Framework 3
Personal 3
Look and feel / impression of the tool:
The company Antidot itself has a very nice and professional website and homepage. But the db2triples is just a tiny part of the page and can be found under Ecosystem/opensource/db2triples and only contains a brief description of the tool. Otherwise it is the github page that is the place to look for information for the tool.

Links:
Antidot homepage: http://www.antidot.net/en/
Antidot db2triple page: http://www.antidot.net/en/ecosystem/db2triples/
Github page: https://github.com/antidot/db2triples

Grade:
Framework 3
Personal 3

How to use it:
It is delivered as a java library, some dependencies such as Sesame, Commons-logging and Commons-CLI are needed. You can launch it from the command line or use maven to build your own jar. Overall very complicated tool to use and programming experience is needed.

Grade:
Framework 3
Personal 3

Relational databases support:
Supported databases:
MySQL and PostgreSQL

Grade:
Framework 3

RDF serialization support:
Supported syntaxes for the materialized graphs created by Db2Triples are:
RDF/XML, N3, N-triples and Turtle.

Grade:
Framework 2

Popularity:
Db2Triples has served as benchmark and been involved in the semantic web. No SPARQL endpoint or possibilities to publish data without support tool makes it hard to evaluate the popularity when it is “only” a RDF2RDF converter tool. The number of hits on internet and lack of information is though a guideline that it is not that commonly used.

Searching in UB on the phrases:
- Db2triples gives 1 hit.
- Antidot db2triples gives 0.

Links:
Ub search [http://www.ub.umu.se/sok](http://www.ub.umu.se/sok)

Grade:
Personal 3
Framework 3

**Sustainability:**
Github seem to been active since September 2011 and the tool is developed by Antidot.
Not sure if Antidot continuously updates or maintains the tool. Last update of the jar was in December 2013. On github there are five releases, the first was in 2011.

Grade:
Framework 2

**Economy:**
Open source software published under LGPL license.

**Extra tools support:**
Due to github readme is the following needed dependencies. Db2Triples is a java library.
OpenRdf Sesame > 2.6.x
Commons-cli > 1.2
Commons-logging > 1.1.1

**Support WC3 direct mapping:** Support WC3 R2RML : Support of its’ own language:
Supports both R2RML and direct mapping.

**SPARQL endpoint / query service:**
No

**SPARQL version:**
No SPARQL, must load the data into an external triple store or develop your own.

**RDF dump (ETL), virtual mapping or both:**
ETL, extracts data that must be loaded into a triple store.

**Server and other things included in the package or have to develop by yourself:**
Is a java library.

**Other prominent extra features or good to knows:**
Has been validated by the w3c working group
The comparison tables

<table>
<thead>
<tr>
<th></th>
<th>Openlink Virtuoso</th>
<th>D2RQ</th>
<th>Db2Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Documentation and support:</strong></td>
<td>F 1, P 2</td>
<td>F 2, P 1</td>
<td>F 3, P 3</td>
</tr>
<tr>
<td><strong>Look and feel / impression of the tool:</strong></td>
<td>F 1, P 1</td>
<td>F 2, P 2</td>
<td>F 3, P 3</td>
</tr>
<tr>
<td><strong>How to use it:</strong></td>
<td>F 2, P 2</td>
<td>F 1, P 1</td>
<td>F 3, P 3</td>
</tr>
<tr>
<td><strong>Popularity:</strong></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td><strong>Relational database support:</strong></td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td><strong>RDF serialization support:</strong></td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>Sustainability:</strong></td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

**Table 1:** The rank results of the evaluation, based on number of hits, some subjective impressions and number of databases and formats supported. F is the measurable and P personal rank.
<table>
<thead>
<tr>
<th></th>
<th>Openlink Virtuoso</th>
<th>D2RQ</th>
<th>Db2Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Relational database support:</strong></td>
<td>IBM DB/2, Firebird,Informix,Ingres ,Microsoft SQL Server,MySQL, Oracle ,PostgreSQL ,Progress / OpenEdge Sybase</td>
<td>Oracle ,MySQL PostgreSQL , SQL Server HSQldb , Interbase/Firebird</td>
<td>MySQL, PostgreSQL</td>
</tr>
<tr>
<td><strong>RDF serialization support:</strong></td>
<td>RDF/XML, JSON, N3 N-Triples TURTLE</td>
<td>TURTLE,RDF/XML, RDF/XML-ABBREV, N3 N-TRIPLE</td>
<td>RDF/XML, N3, N-triples, TURTLE</td>
</tr>
<tr>
<td><strong>Sustainability:</strong></td>
<td>Latest windows version released 2015-12-09</td>
<td>Latest version 0.8.1 was released 2012-06-22</td>
<td>Last update of the jar was December 2013</td>
</tr>
<tr>
<td><strong>Economy:</strong></td>
<td>Both open source published GNU General Public License Version 2 and commercial , 99.99 USD to 9998 USD</td>
<td>Open source tool published under Apache license</td>
<td>Open source software published under LGPL license</td>
</tr>
<tr>
<td><strong>Extra tools support:</strong></td>
<td>Jena and Sesame support</td>
<td>Jena support</td>
<td>Java library</td>
</tr>
<tr>
<td><strong>Support WC3 direct mapping:</strong></td>
<td><strong>Support WC3 R2RML:</strong> R2RML and its own Meta schema language</td>
<td>D2RQ mapping language and support for Direct mapping</td>
<td>Supports both R2RML and direct mapping.</td>
</tr>
<tr>
<td><strong>SPARQL endpoint / query service:</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>SPARQL version:</strong></td>
<td>SPARQL version 1.1</td>
<td>SPARQL version 1.1</td>
<td>-</td>
</tr>
<tr>
<td><strong>RDF dump (ETL), dynamic mapping or both:</strong></td>
<td>Dynamic mapping, some sources claims it is possible to do ETL</td>
<td>Both</td>
<td>ETL</td>
</tr>
<tr>
<td><strong>Server and other things included in the package</strong></td>
<td>Universal server with web interface and server</td>
<td>Server, recommended to deploy on servlet container i.e Tomcat</td>
<td>-</td>
</tr>
<tr>
<td><strong>Other prominent extra features or good to knows:</strong></td>
<td>Universal server, a lot of different features</td>
<td>Focused on published relational data as linked data</td>
<td>Has been validated by the w3c working group</td>
</tr>
</tbody>
</table>

**Table 2:** The table with the evaluation of each property for the three tools Virtuoso, D2RQ and Db2Triples with information about non measurable properties. Must be evaluated from the context and used as a look up table.