

# Models for Efficient Semantic Data Storage Demonstrated on Concrete Example of DBpedia

Ivo Lašek and Peter Vojtáš

<sup>1</sup> Czech Technical University in Prague, Faculty of Information Technology, Prague, Czech Republic, [lasekivo@fit.cvut.cz](mailto:lasekivo@fit.cvut.cz)

<sup>2</sup> Charles University in Prague, Faculty of Mathematics and Physics, Prague, Czech Republic, [vojtas@ksi.mff.cuni.cz](mailto:vojtas@ksi.mff.cuni.cz)

**Abstract.** In this paper, we introduce a benchmark to test efficiency of RDF data model for data storage and querying in relation to a concrete dataset. We created Czech DBpedia - a freely available dataset composed of data extracted from Czech Wikipedia. But during creation and querying of this dataset, we faced problems caused by a lack of performance of used RDF storage. We designed metrics to measure efficiency of data storage approaches. Our metric quantifies the impact of data decomposition in RDF triples. Results of our benchmark applied to the dataset of Czech DBpedia are presented.

**Keywords:** Semantic Web, Linked Data, Storage, RDF

## 1 Introduction

DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. In this paper, we introduce a Czech branch of this effort. The main DBpedia development gathers around English DBpedia. However, couple of local clones have emerged lately. To name some of them, we mention Greek, Russian or German local DBpedias. The complete list may be found in [17]. To the best of our knowledge, there has not been any other Czech DBpedia clone, apart from our work. We aim to extract structured information from Czech Wikipedia and provide the data for free use. In this paper, we show some use cases in order to demonstrate what can be such a huge database of machine readable information good for (focused particularly on Czech users).

We describe the data, we have extracted so far. We show basic statistics that can provide also a new point of view on the information available at Czech Wikipedia - not limited to DBpedia.

Described data are accessible via the SPARQL endpoint<sup>3</sup>, so anyone can query them and use them. Also anyone can contribute to the community driven extraction effort via Mappings Wiki [15].

<sup>3</sup> The SPARQL endpoint is accessible on <http://cs.dbpedia.org/sparql>.

## 1.1 The Problem

Though datasets such as DBpedia can be very useful as pointed out in section 4, the adoption of these datasets is often limited to semantic web community. One of the main drawbacks is lack of performance. Data are usually stored as RDF triples. The performance of main RDF stores (or triple stores) increases as indicated by several benchmarks [1, 2]. But ordinary operations that are effectively executed in relational databases remain very demanding in the case of RDF stores.

The main factor that causes RDF stores to be inefficient for certain types of queries is a number of joins the store has to perform in order to evaluate a query. However, in many cases on the web, data is presented in a joined form (e.g. description of one entity and its properties on one page like in a Wikipedia article) and it is split in order to represent it as RDF. We elaborate this process in detail in Section 5. The opened question is, whether this split is necessary. We evaluate this phenomenon and quantify its influence on the dataset performance. Czech DBpedia is used as a representative dataset.

## 1.2 Contributions

- We introduce a comprehensive dataset of machine readable information extracted from Czech Wikipedia.
- The presented dataset covers wide variety of areas and preserves the connection of data to corresponding Wikipedia articles. As such can serve as a data integration hub and a source of common identifiers facilitating data integration of diverse data sets.
- We show possible applications of this dataset. We focus particularly on specific Czech use cases, where Czech DBpedia is worth than the global one.
- We point out the problem of ineffective data representation in case of RDF and quantify its influence on a dataset.
- We provide a SPARQL benchmark for testing data stored in triple stores.

## 1.3 Organization of the Paper

The rest of the paper is organized as follows. Related work is recalled in Section 2. Section 3 introduces the process of our dataset creation and Section 4 describes possible use cases of the dataset. Section 5 describes methods used to store RDF data and introduces our metrics measuring efficiency on a concrete dataset. Experimental results are presented in Section 6. The paper is summarized in Section 7.

## 2 Related Work

The whole philosophy of DBpedia and technical details are provided in [3, 4]. Both papers provide a broad overview of the DBpedia background. In our paper

we focus on a specific Czech environment and describe the dataset based on data from Czech Wikipedia, which might have slightly different characteristics (i.e. more fine grained information about locally specific entities).

Additionally, based on our dataset, we introduce metrics to measure the performance of data storage. There are several benchmarks measuring the performance of RDF data stores using SPARQL queries, similarly to our work. The suite of benchmarks for comparing the performance of SPARQL endpoints - Berlin SPARQL Benchmark [1] - tests the performance of SPARQL engines in prototypical e-commerce scenarios. Another recent benchmark built on top of English DBpedia dataset is DBpedia SPARQL Benchmark [2]. Contrary to Berlin Benchmark, that is composed of artificially designed queries, DBpedia Benchmark uses real world queries of real users. The queries are extracted from query logs of DBpedia SPARQL endpoint.

All the mentioned benchmarks use same metrics to measure the performance. The basic metrics are Queries per Second (QpS), Query Mixes per Hour (QMpH) and Overall Runtime (oaRT). But all these metrics are heavily dependent on a hardware configuration and overall system conditions of a test run. In Section 5.2, we introduce metrics that are more qualitative and quantitative characteristics of the tested dataset according to the used storage approach. Thus the results of our benchmark are completely independent of a concrete hardware configuration.

In order to develop our metrics, we considered various approaches to RDF data storage presented by major triple stores Jena [5], Virtuoso [9], Sesame [7], Oracle [6] and 3store [8]. Additionally, we consider vertical partitioning presented in [10]. These approaches are described and compared in Section 5. A similar problem of transformation of an ontology into a relational database is discussed in [11].

### 3 DBpedia Extraction Framework

In order to obtain raw data from Wikipedia, we use the DBpedia Extraction Framework [12]. This is a module based framework maintained by the international DBpedia team. Wikipedia provides freely accessible dumps of the whole article database [13]. The framework thus downloads recent dumps of all Wikipedia pages covering all topics described on Wikipedia. The pages are downloaded in the source format marked by Wiki markup [14]. These source files are parsed. Data are extracted from parsed pages using various extractors. An extractor is a mapping from a page node to a graph of statements about it.

Various information can be obtained from Wikipedia pages. It is quite easy to get labels of entities and extract their connections by analysis of links between corresponding Wikipedia articles. However, the core of the extraction process is the retrieval of information contained in so called infoboxes. An example of such an infobox in a Wikipedia article is shown in Figure 1.

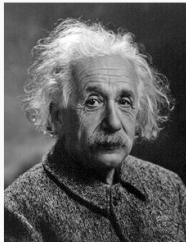
In case of the Czech DBpedia, we use following extractors provided by the extraction framework: Label Extractor (extracts labels of entities from titles of corresponding articles), Geo Extractor (extracts geographic coordinates), Page

## Albert Einstein

**Albert Einstein** (14. března 1879 Ulm, Německo – 18. dubna 1955 Princeton, New Jersey, USA) byl teoretický fyzik, jeden z nejvýznamnějších vědců všech dob. Často je označován za největšího vědce 20. století, případně spolu s Newtonem za nejvýznamnějšího fyzika vůbec. Mezi jeho příspěvky fyzice patří speciální teorie relativity (1905), myšlenka kvantování elektromagnetického pole a vysvětlení fotoefektu (1905), vysvětlení Erovna pohybu (1905) a snad nejvíce obecná teorie relativity (1915), která doposud nejlépe popisuje vesmír ve velkých měřítkách.

Einstein se podílel i na statistické fyzice a kvantové statistice (Boseho-Einsteinova rozdělení), diskusi o interpretaci kvantové mechaniky (diskuse s Bohrem, EPR paradox). S Leó Szilárdem vynalezili nový typ chladničky.

V roce 1921 byl oceněn Nobelovou cenou za fyziku za „vysvětlení fotoefektu a zaslúhy o teoretickou fyziku“. Obrovským vědeckým úspěchem totiž byly i ostatní tři

|   |   |
|---|---|
| <b>Albert Einstein</b>  |   |
|  |   |
| Albert Einstein   |   |
| <b>Narozen</b>  | 14. března 1879<br>Ulm, Württemberg, Německo  |
| <b>Zemřel</b>   | 18. dubna 1955<br>Princeton, New Jersey, USA  |
| <b>Národnost</b>  | židovská  |
| <b>Obor</b>   | Fyzika  |
| <b>Znamy díky</b>   | Obecná a speciální teorie relativity<br>Objev zákonitosti fotoelektrického jevu<br>$E = mc^2$ |
| <b>Získaná ocenění</b>  | Nobelova cena za fyziku 1921  |

```

{{Infobox Vědec
|jmeno           = Albert Einstein |
|narodnos       = [[Židé|židovská]]
|sirka_boxu     = 30
|obrazek       = Albert_Einstein_Head.jpg |
|popisek       = Albert Einstein |
|motto         = <nowiki>Nikde neleží pravda
                na povrchu.</nowiki>|
|datum_narzeni = [[14. březen|14. března]]
                [[1879]] |
|misto_narzeni = [[Ulm]], [[Württemberg]],
                [[Německo]] |
|datum_umrti   = [[18. duben|18. dubna]]
                [[1955]] |
|misto_umrti   = [[Princeton]], [[New
                Jersey]], [[Spojené státy
                americké|USA]] |
|obor          = [[Fyzika]] |
|znamy_diky    = [[Obecná teorie relativity|
                Obecná]] a [[speciální
                teorie relativity]]<br />
                Objev zákonitosti
                [[fotoelektrický jev|
                fotoelektrického jevu]]<br />
                [[E=mc2|E=mc²; = &nbsp;
                mc&sup2;]] |
|ocneni        = [[Nobelova cena za fyziku]]
                [[1921]] |
}}

```

**Fig. 1.** Infobox example taken from Czech Wikipedia. On the right side, there is a source code of this Wikipedia infobox written using Wiki markup.

Links Extractor (extracts internal links between DBpedia instances from the internal pagelinks between Wikipedia articles), Wiki Page Extractor (extracts links to corresponding articles on Wikipedia), Infobox Extractor (extracts all properties from all infoboxes) and Mapping Extractor (extracts structured data based on hand-generated mappings of Wikipedia infoboxes to the DBpedia ontology).

It is important to note the difference between Infobox Extractor and Mapping Extractor. Consider the source code from Figure 1. The Infobox Extractor extracts this information as it is written in the source code. Property names are not cleaned, there is no consistent ontology for the infobox dataset. Thus generating RDF triples like:

```

<http://cs.dbpedia.org/resource/Albert_Einstein>
  <http://cs.dbpedia.org/property/misto_narozeni>
  <http://cs.dbpedia.org/resource/Ulm> .

```

Unfortunately, infoboxes on Wikipedia are inconsistent and it is usual that same property is described in many different ways. Someone can call the same property `misto_narozeni` (placeOfBirth), whereas someone else might use `puvod` (origin), or `narozeni_misto` (birthPlace).

The answer to these difficulties is the Mapping Extractor which uses hand written rules that map different patterns used in Wikipedia infoboxes to a consistent ontology of DBpedia.

For our dataset we generated rules covering 60 most important entities (e.g. cities, politicians, actors, writers). Totally, there are currently about 109 Infobox

templates on Czech Wikipedia that can be potentially mapped to DBpedia ontology. This means so far we have mapped more than half of them. Mappings can be edited via the Mappings Wiki [15]. As the mapping effort is community driven, everyone can join and help creating and maintaining mapping rules.

## 4 Use Case Scenarios

In this section, we want to point out the advantages of a local clone of DBpedia compared to the English one.

Similarly to local Wikipedias, local DBpedias usually provide more comprehensive information about specific local entities, like geographical data (smaller Czech cities, mountains, rivers, lakes), data about important persons (Czech politicians, movie directors, writers, etc.).

As such, this dataset may serve as a base for various mashup application or automatic data processing tools. Apart from the range of locally specific data, the language of entries and their direct connection to Czech Wikipedia pages might be an advantage too.

Thanks to tens of manually created mapping rules, Czech DBpedia is a good ontology mapping dictionary as well. It may serve as a central hub providing a set of common identifiers together with basic properties of identified entities.

All machine readable data on DBpedia have a direct connection to corresponding Wikipedia articles, where the information is presented in an unstructured way, usually as a plain text. Thus Czech DBpedia might serve as a testing dataset for various natural language processing and information retrieval approaches focusing on Czech language.

## 5 Efficiency of Data Storage

### 5.1 Data Representation

The RDF data model represents data as statements about resources using a graph connecting resource nodes. An RDF statement has the form of a triple consisting of subject, predicate and object. In the following text, unless otherwise stated, under subject, predicate and object, we understand the appropriate part of an RDF triple.

The majority of RDF data storage solutions including Jena [5], Virtuoso [9], Sesame [7], Oracle [6] and 3store [8] use relational databases to store the data. The general idea is to create one giant triples table with three columns (corresponding to RDF triples: subject, predicate, object) containing one row for each RDF statement. This data representation results in many self-joins on triples table even to execute simple SPARQL queries. Consider following query that returns names of movies directed by Jan Svěrák and filmed in 1996<sup>4</sup>:

<sup>4</sup> Usually, when querying real SPARQL endpoints, queries involve use of identifiers in similar form to URLs. For clarity of presented SPARQL queries, we omit these long identifiers as well as declaration of appropriate prefixes. Instead we use shorter identifiers in our examples (e.g. director, filmed, name).

```

SELECT ?name
WHERE { ?movie director "Jan Svěrák" .
?movie filmed "1996" .
?movie name ?name . }

```

This query results in following SQL query over triples table, invoking three joins:

```

SELECT T3.object
FROM triples_table AS T1,
     triples_table AS T2,
     triples_table AS T3
WHERE T1.subject = T2.subject AND T1.property = 'director'
      AND T1.object = 'Jan Svěrák' AND T2.subject = T3.subject
      AND T2.property = 'filmed' AND T2.object = '1996'
      AND T3.property = 'name'

```

Instead of storing whole identifiers and literals directly in the triples table, Oracle, Virtuoso and Sesame replace strings with integer identifiers, so that the data is normalised in two tables. Whereas Jena takes the trade off - space for speed - and keeps strings directly in the triples table.

In order to minimize the count of inefficient join operations that are invoked by each statement used in SPARQL query, various optimizations were proposed.

Property tables first introduced by authors of Jena [5] optimize data organisation in following way:

- Create clusters of properties that often occur together.
- Based on identified clusters, create tables having properties occurring together as columns.
- The primary key of tables are subjects, in columns are stored objects that are connected with a particular subject by property represented by the column.

A variant of property tables are property-class tables, where clusters are created based on RDF types of subjects.

A similar approach to the property table data structure was introduced in [11] in order to store ontologies in relational databases.

A different approach is vertical partitioning described in [10]. The basic principle is to create a table for each property in the dataset. The table has two columns: subject and object. Each row thus corresponds to an RDF statement in that the particular property (represented by the table) connects subject and object stored in the same row. The performance optimization is achieved by storing data in a column oriented storage.

## 5.2 Metrics

The aim to minimize the impact of joins on the query execution leads us to define metrics that quantify this impact on a particular dataset.

In the following text, we use the term **shallow properties** to label properties such that there is at least one resource appearing with them in an object position that does not appear in a subject position in another statement.

Contrary, by **deep properties** we mean properties such that there is at least one resource appearing with them in an object position that appears in a subject position in at least one another statement.

**Joinability** measures an average count of shallow properties on a subject. If shallow properties were stored in one table (as columns of the table), there would not be a need to join the data at all. Shallow properties result in an avoidable subject-subject join in that case. Joinability is defined as follows:

$$j = \frac{c_s}{|S|}$$

Where  $c_s$  is the count of statements, where a property plays a role of a shallow property and  $|S|$  is the count of distinct subjects in the dataset.

**Linkability** is an average count of deep properties on a subject. The occurrence of deep properties may result in an object-subject join, if we query for properties of an object as well. Linkability is defined in the following way:

$$l = \frac{c_d}{|S|}$$

Where  $c_d$  is the count of statements, where a property plays a role of a deep property and  $|S|$  is the count of distinct subjects in the dataset.

Finally, we define **Weighted Joinability** as:

$$w = \frac{j}{l}$$

Another interesting characteristic of a dataset is an average **indegree** of an object and **outdegree** of a subject. Especially outdegree of a subject indicates, how many joins could be potentially saved, if all the properties of a particular subject were stored in one row. Note that sum of an average joinability and linkability gives an average outdegree of subjects in a dataset.

## 6 Evaluation

### 6.1 Data Characteristics

For the evaluation, we used the dataset of Czech DBpedia. It is smaller than the dataset of English DBpedia, so it is feasible to run some more demanding queries that are difficult to run in reasonable time on the English one. However, still Czech DBpedia represents a comprehensive dataset, with similar characteristics to the English one. In this section, we provide some basic characteristics of data that can be extracted from Czech Wikipedia and compare it to the English Wikipedia.

Czech DBpedia contains totally 12 402 513 RDF statements. About 251 877 RDF statements are extracted based on our hand written mappings, whereas the English DBpedia composes of about 17,5 million RDF statements based on mappings (according to the last dataset release report [16]).

Data is extracted from 220 000 articles on Czech Wikipedia. English Wikipedia has about 3 800 000 articles which is more than 17 times bigger dataset.

There are 704 760 distinct subjects that have some property in the current Czech dataset.

Additionally, we counted entities. Under entities, we mean real world concepts. Usually, each entity corresponds to a Wikipedia article that describes it. Thus total count of entities should correspond to 220 000 Wikipedia articles. But in reality the count of identified entities is heavily dependent on a coverage of hand written mapping rules. Entities have commonly a type. Counts of most common entities compared to the English dataset are provided in Table 1. It is remarkable that in case of cities, the count of entities in Czech dataset is closer to the English dataset than in other cases (it is more than 34% of the count of cities in English dataset). This points to the fact that information about cities are well elaborated on Czech Wikipedia and also good mappings are provided to transform it to DBpedia. The English DBpedia presents actually much more countries than their real count in the world. This fact is caused by a vague definition of a country on Wikipedia. For example a self-governing British Overseas Territory Falkland Islands is considered to be a country as well.

**Table 1.** Comparison of Czech and English DBpedia. Count of entities of certain types. In the column Size Comparison the sizes of both datasets are compared in percent.

| Entity Type | Count of Entities |                 | Size Comparison |
|-------------|-------------------|-----------------|-----------------|
|             | Czech DBpedia     | English DBpedia |                 |
| Person      | 8478              | 416079          | 2,0%            |
| Company     | 964               | 40132           | 2,4%            |
| Country     | 287               | 2531            | 11,3%           |
| City        | 4730              | 13790           | 34,3%           |

## 6.2 Measurements and Benchmarks

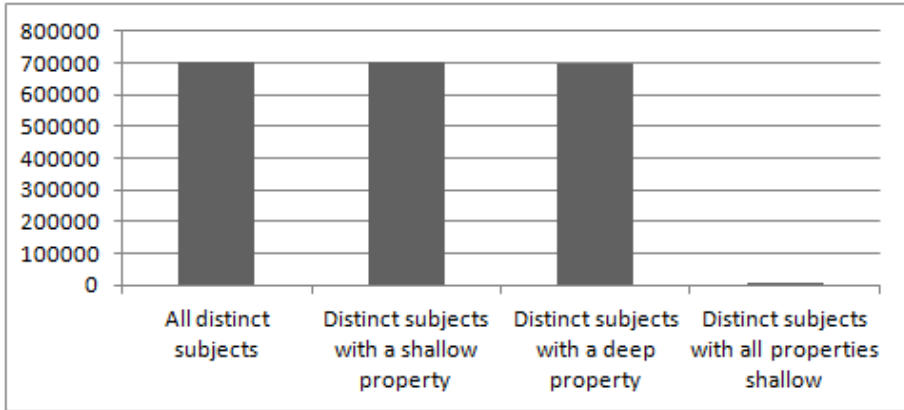
In this section, we provide some characteristics, we measured on the dataset of Czech DBpedia with respect to metrics introduced in Section 5.2. Due to the lack of space in this paper, we present the actual benchmark on a separate web page<sup>5</sup>.

In Figure 2, we compare count of distinct subjects with at least one shallow property and at least one deep property. We can see that almost all subjects

<sup>5</sup> The whole benchmark used to obtain presented results composes of SPARQL queries and basic scripts to parse results of these queries. It is available for download at <http://research.i-lasek.cz/semantic-web/joinability-benchmark/>



have a shallow property as well as a deep property. There were less than 6 000 subjects having all properties shallow. These were in most cases operational and meta data information rather than real entities such as cities, persons etc.



**Fig. 2.** Counts of distinct subjects with at least one shallow property and at least one deep property compared to the total count of distinct subjects in the dataset. Additionally subjects with all properties shallow are displayed.

In Figure 3, we further investigated overall count of distinct shallow properties and distinct deep properties. It is remarkable that both sets have a common intersection. According to the results, almost all properties are in some cases shallow. While some of them play a role of a deep property as well. This is legal, because a property might connect a subject with an object, that is not further described in the dataset, while in another case another connected object is a subject in another statement. Also sometimes, data is messy and same property connects subject with a literal value and at the same time connects it with an object with further properties. 1 697 out of 6 714 properties played a role of deep properties in at least one statement.

In Figure 4, we compare counts of objects that do not play a role of a subject in any other statement (result in a shallow connecting property), with objects that play a role of subjects in at least one another statement (result in a deep connecting property).

Afterwards, we measured counts of non distinct shallow and deep properties and counted Linkability and Joinability. For Czech DBpedia, these are:  $j = 8,92$  and  $l = 8,67$ .

Finally, we evaluated average outdegree of subjects (for Czech DBpedia it is approximately 17,6) and average indegree of objects (approximately 5,0). For the quantification of join impact, the outdegree of subjects is especially important. This means that, while querying DBpedia for all properties of an entity, a query results in average in almost 18 self-joins, if we consider the triples table approach

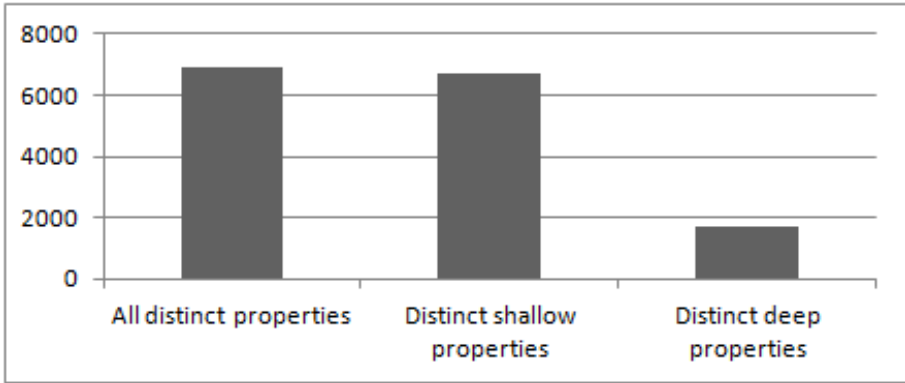


Fig. 3. Counts of distinct shallow properties and distinct deep properties compared to overall count of distinct properties.

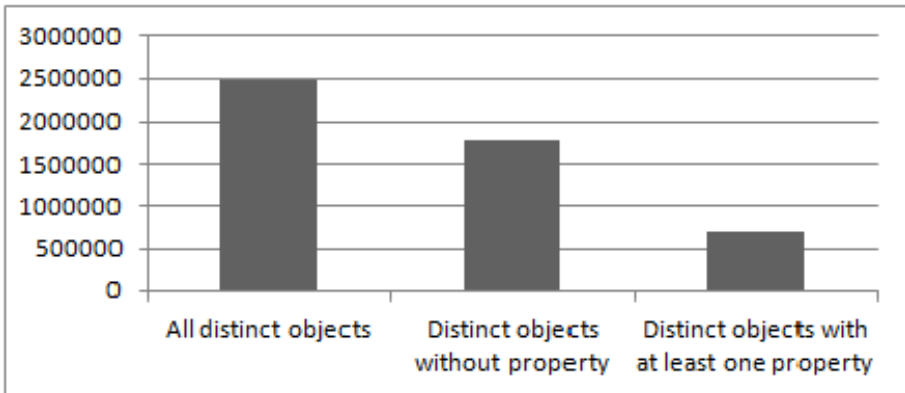
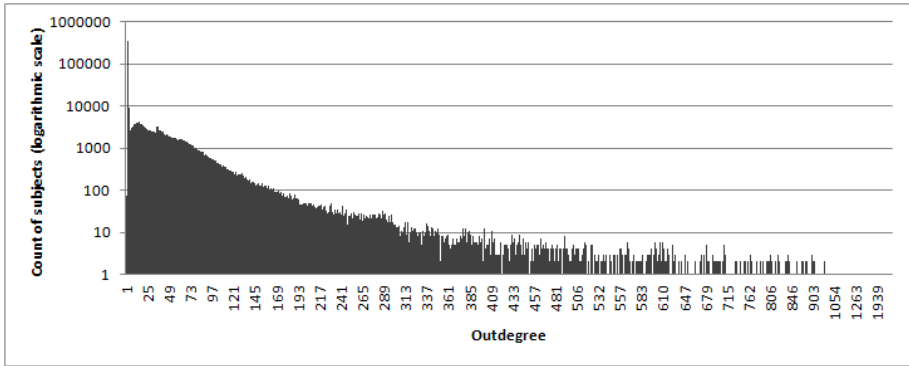


Fig. 4. Counts of distinct objects that have at least one property in the dataset (are in the role of a subject in another triple) and distinct objects that do not have any property within the given dataset.

described in Section 5.1. The distribution of subjects outdegree is displayed in Figure 5.



**Fig. 5.** Subjects outdegree distribution. Counts of occurrences are displayed in a logarithmic scale.

## 7 Conclusion

We introduced Czech DBpedia and briefly described the process of creation of this dataset. The main idea was introduced: On the web, huge amount of data is presented in a joined form (e.g. infobox tables on Wikipedia). However, often this data is split in order to present it as RDF. Afterwards it has to be joined again, while querying RDF data storage, which is a demanding operation. We designed metrics to measure efficiency of data storage approaches, according to the join demands. We introduced a benchmark to test efficiency of RDF data model for data storage and querying in relation to a concrete dataset. Finally, we presented results of this benchmark applied to a dataset of Czech DBpedia. The benchmark is publicly available on the web, so anyone can use it to test any dataset accessible via a SPARQL endpoint.

### 7.1 Future Work

In our future work, we plan to design an efficient RDF repository. Based on our current findings, we intend to use property tables. Our modification is that we use a variant of the algorithm for creating concepts (in the sense of Conceptual lattices (see [18]) on properties (columns of tables). Our optimisation is driven by requirement to minimize number of joins in most frequent expected queries (conjunctive queries looking for subject with conjunction of conditions on property values). Secondary optimization is devoted to minimization of number of NULL values (which occurs when a subject does not have some property).

**Acknowledgments.** This work has been supported by the grant of Czech Technical University in Prague (SGS12/093/OHK3/1T/18) and by the grant GACR P202/10/0761.

## References

1. Bizer, C., Schultz, A.: Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints. In *International Journal On Semantic Web and Information Systems*, 2009.
2. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo: DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In *The International Semantic Web Conference – ISWC 2011*, Springer Berlin / Heidelberg, 2011, 7031, Pages 454–469.
3. Bizer, Ch., Lehmann, J., Kobilarov, G., Auer, S., Becker, Ch., Cyganiak, R., Hellmann, S.: DBpedia - A crystallization point for the Web of Data. In *Web Semantics: Science, Services and Agents on the World Wide Web*, Volume 7, Issue 3, September 2009, Pages 154–165, ISSN 1570-8268.
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data *The Semantic Web*. Springer Berlin / Heidelberg, 2007, 4825, Pages 722–735.
5. Wilkinson, K., Sayers, C., Kuno, H., Reynolds, D.: Efficient RDF Storage and Retrieval in Jena2. In *SWDB*, Pages 131–150, 2003.
6. Chong, E. I., Das, S., Eadon, G., Srinivasan, J.: An Efficient SQL-based RDF Querying Scheme. In *VLDB*, Pages 1216–1227, 2005.
7. Broekstra, J., Kampman, A., Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *ISWC*, Pages 54–68, 2002.
8. Harris, S., Gibbins, N.: 3store: Efficient bulk RDF storage. In *In Proc. of PSSS 03*, Pages 1–15, 2003.
9. Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS. In *Networked Knowledge - Networked Media*, Studies in Computational Intelligence, Springer Berlin / Heidelberg, isbn: 978-3-642-02183-1, 2009, Pages 7–24.
10. Abadi, D. J., Marcus, A., Data, B.: Scalable semantic web data management using vertical partitioning. In *VLDB 2007*, Pages 411–422, 2007.
11. Pokorný, J., Pribolová, J., Vojtáš, P.: *Ontology Engineering Relationally*. In *DATESO 2009*, Špindlerův Mlýn, Czech Republic, 2009, Pages 44–55.
12. The DBpedia Information Extraction Framework, <http://wiki.dbpedia.org/Documentation>
13. Wikipedia Database Download [http://en.wikipedia.org/wiki/Wikipedia:Database\\_download](http://en.wikipedia.org/wiki/Wikipedia:Database_download)
14. Wiki markup [http://en.wikipedia.org/wiki/Help:Wiki\\_markup](http://en.wikipedia.org/wiki/Help:Wiki_markup)
15. Czech Mappings on DBpedia Mappings Wiki <http://mappings.dbpedia.org/index.php?title=Special%3AAllPages&from=&to=&namespace=224>
16. Dataset Release Report <http://blog.dbpedia.org/category/dataset-releases/>
17. DBpedia Internationalization Committee <http://dbpedia.org/Internationalization>
18. Ganter, B., Stumme, G., Wille, R., eds. (2005): *Formal Concept Analysis: Foundations and Applications*. In *Lecture Notes in Artificial Intelligence*, no. 3626, Springer-Verlag, ISBN 3-540-27891-5.