Binary RDF for Scalable Publishing, Exchanging and Consumption in the Web of Data

Javier D. Fernández [1,2]
Supervised by: Miguel A. Martínez Prieto [1] and Claudio Gutierrez [2]
[1] Department of Computer Science, University of Valladolid (Spain)
[2] Department of Computer Science, University of Chile (Chile)
jfergar@infor.uva.es

ABSTRACT

The Web of Data is increasingly producing large RDF datasets from diverse fields of knowledge, pushing the Web to a data-to-data cloud. However, traditional RDF representations were inspired by a document-centric view, which results in verbose/redundant data, costly to exchange and post-process. This article discusses an ongoing doctoral thesis addressing efficient formats for publication, exchange and consumption of RDF on a large scale. First, a binary serialization format for RDF, called HDT, is proposed. Then, we focus on compressed rich-functional structures which take part of efficient HDT representation as well as most applications performing on huge RDF datasets.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services-Data sharing

General Terms

Algorithms, Design

1. MOTIVATION. THE PROBLEM

Massive publication efforts have flooded the emerging Web of Data with very large RDF datasets from such diverse fields as bioinformatics, geography or bibliography. This democratization in the creation of semantic data has mainly been driven by the Linked Open Data (LOD) community, which promotes the use of standards (such as RDF and HTTP) to publish such structured data on the Web and to connect it by reusing dereferenceable identifiers between different data sources [13]. More than 30 billion RDF triples are being shared and increasingly linked in the LOD cloud[1], which results in huge interconnected RDF datasets from different providers. Each provider should be responsible for an efficient publication of their datasets in order to mitigate inherent scalability drawbacks on interchange and consumption. However, both providers and consumers are provided with feeble serialization schemes; the mainstream RDF serialization syntaxes (RDF/XML [2] N3 [1], Turtle [4] and RDF/JSON [8]) were not thought up in this "data deluge" era and hence they share a document-centric view, providing human-focused syntaxes, disregarding large data.

In a typical scenario within the current state-of-the-art, efficient interchange of RDF data is limited, at most, to compressing the verbose plain data with universal compression algorithms. The resultant file has no logical structure and there is no agreed way to efficiently publish such data, *i.e.*, to make them (publicly) available for diverse purposes and users. In addition, the data are hardly usable at the time of consumption; the consumer has to decompress the file and, then, to use an appropriate external tool (*e.g.* an RDF store or a visualization software). This diminishes the potential of RDF graphs due to the huge space they take up and the large time required for consumption. Similar problems arise when managing less RDF data but in mobile devices; together with scalability and memory constraints, these devices can face additional transmission costs [23]. This scenario calls for an efficient (binary) RDF serialization format, moving forward RDF syntaxes to a data-centric view.

This paper is organized as follows. Section 2 reviews the state-of-the-art for publishing, exchanging and consuming RDF on a large scale. In Section 3, a binary RDF representation is proposed together with efficient RDF dictionaries and triples structures. The methodology of the on-going thesis is described in Section 4. Current results are cited in Section 5 and we conclude by setting out future work in Section 6.

2. STATE OF THE ART

We can divide the scalability drawbacks of the Web of Data into three correlated processes: publication, exchange and consumption (query) of the information.

Publication. Besides RDF publication with dereferenceable URIs, data providers tend to expose their data as a file to download (RDF dump), or via a SPARQL endpoint, a service which interprets the SPARQL query language [3]. Except for some Linked Data recommendations [21], few works address the publication of RDF on a large scale. The Vocabulary of Interlinked Datasets (VoiD [9]) aims to provide a bridge between data publishers and data users. Publishers make use of the vocabulary to add metadata to their datasets, *e.g.* to point to the associated SPARQL endpoint

[1] http://www4.wiwiss.fu-berlin.de/lodcloud/state/

and RDF dump, to describe the total number of triples and to connect to linked datasets. Thus, consumers can look up this metadata to discover datasets, both in a straightforward way (consuming all the metadata) or through queries (typically SPARQL) which filter certain attributes of the dataset. Semantic Sitemaps [15] extend the traditional Sitemap Protocol for describing RDF data. They include new XML tags so that crawling tools (such as Sindice[2]) can discover and consume the datasets.

Exchange. RDF datasets are exchanged within plain RDF formats. RDF/XML [2] was mainly designed to add small metadata to documents such as web pages. Due to its verbosity, it is good for exchanging data, but only on a small scale. Notation3 (N3 [1]) is a compact and readable alternative. It reduces verbosity with some compacting features such as abbreviations for URI prefixes (and base URI), shorthand for common predicates, square bracket blank node syntax and the use of lists. Turtle [4] is intended to be compatible with Notation3, thus it inherits its compact ability while also adding extra features, *e.g.* abbreviated RDF collections. RDF/JSON [8] resembles Turtle, with the advantage of being coded in a language easier to parse and more widely accepted in the programming world. Although these formats present features to "abbreviate" constructions, they are still dominated by a document-centric and human-readable view.

Universal compressors (*e.g.* gzip) are commonly used over these plain formats in order to reduce their size. RDF/XML is a valid XML format and thus XML interchange formats might be used. For instance, the Efficient XML Interchange Format (EXI [5]) is a compact representation for XML. It is based on efficient encodings of XML event streams using a grammar-driven approach.

Consumption (query). Efficient consumption in any scenario is influenced by two factors: (1) the serialization format, due to the overall data exchange time, and (2) the RDF indexing/querying structure. This latter factor affects the consumption process in different ways: (a) for SPARQL endpoints, the response time depends on the efficiency of the underlying RDF indexes at the publisher. In any case, (b) once the consumer has all the information, the most likely scenario is to post-process the information by indexing the obtained RDF data in order to operate with the graph at the consumer.

Although diverse techniques provide RDF indexes, there are still workloads for scalable indexing and querying optimization [28, 27]. Some of the proposed techniques store RDF in a relational database and perform SPARQL queries through SQL, *e.g.* Virtuoso[3]. A specific technique, called vertical-partitioning, groups triples by predicate and defines a 2-column (S,O) table for each one [28, 7]. A different strategy is followed in RDF-3X [26]; indexes are created for all ordering combinations (SPO, SOP, PSO, POS, OPS, OSP). Although it achieves a global competitive performance, the index replication largely increases spatial requirements. BitMat [12] suggests a compressed bit-matrix structure for storing the graph, whereas other approaches emerge when using distributed nodes with map-reduce operations [30].

[2] `http://sindice.com/`

[3] `http://www.openlinksw.com/dataspace/dav/wiki/Main/VOSRDF`

3. PROPOSED APPROACH

3.1 Binary RDF Representation

The motivation and state-of-the-art call for a binary representation for RDF aimed at reducing the high levels of verbosity/redundancy and weak machine-processable capabilities of the datasets. Our proposal, called `HDT`, succinctly represents the information of an RDF dataset by organizing and representing the RDF graph in terms of three logical components: `Header`, `Dictionary` and `Triples`.

- Header. Although the binary representation is machine-oriented, this component is aimed at gathering important human-friendly metadata of the dataset. Whereas current serialization formats do not provide the means on how to publish metadata along with datasets, HDT makes metadata a first-class citizen. In practice, we propose the Header to itself be an RDF graph containing information about the provenance (provider, publication dates, version), statistics (size, quality, vocabularies), physical organization (subparts, location of files) and other types of information (intellectual property, signatures). The Header, then, serves as an entry point to the information, a mechanism to discover and (even with SPARQL queries) filter the dataset, or parts of it.

- Dictionary. A generic dictionary maps each term used in a dataset to a unique integer ID. RDF stores [26] make use of this simple but effective decision for managing RDF: all triples in the dataset can now be rewritten by replacing long/redundant terms with their corresponding IDs; consequently, the graph structure in triples can be indexed and managed as an integer-stream.

We propose to incorporate the concept of dictionary into the binary RDF representation. Besides *compactness* at exchanging, the consumption performance can also be improved; an advanced dictionary serialization can be quickly parsed to provide basic operations of `locate` (get the corresponding ID from a given string), and `extract` (get a string from a given ID). In addition, it might help in query evaluation and resolution. For instance, FILTER operations in SPARQL restrict the final result by a given condition. This condition usually refers to a regular expression, language or datatype selection which can be evaluated firstly over the Dictionary, delimiting ranges to search in the structure of triples. An advanced dictionary is proposed in Section 3.2.

- Triples. As stated, the mapping held by the dictionary allows the graph topology to be managed as an integer-stream. These ID-triples are encoded in the triples component, which compactly represent the RDF graph, avoiding the noise produced by long strings. In turn, this is the key component to accessing and querying the RDF graph. RDF indexing (traditionally performed at the time of consumption by an RDF engine) can be moved to the HDT triples component, *i.e.*, an efficient encoding of the triples might provide a succinct index in order to allow some basic queries (*e.g.* SPARQL triple pattern resolution) to be resolved natively on the exchanged component (without the need for decompression nor re-indexing). The Triples component allows diverse configurations and implementations, which can exploit the trade-off between the compression ratio for exchanging and the natively supported operations over the triples. We address a specific triples configuration in Section 3.3.

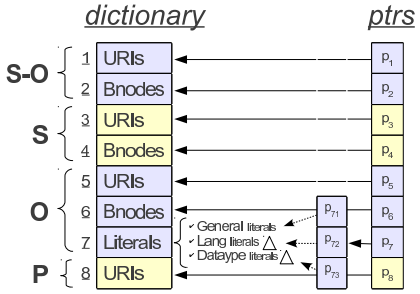

Figure 1: Dictionary organization.

3.2 Dictionary Encoding

An RDF dictionary can be optimized from two correlated perspectives, space and functionality. On the one hand, a dictionary technique which detects and compresses specific *URI, bnode and literal* regularities[4] allows spatial requirements to be optimized. On the other hand, the `locate` and `extract` operations might take into account that the dictionary mapping is not an assignment string-ID over a general set of strings, but a set or terms which are playing a role (subject, predicate, object) in a graph. Literal language and datatypes can be also taken into account for SPARQL filtering. We propose i) a specific organization which optimizes this two perspectives, and ii) the use of compact structures for efficient `locate` and `extract` performance.

Figure 1 illustrates the proposed organization. A role-based partitioning is firstly considered and all terms in the dictionary are organized according to the role they play in the dataset: **Predicates** (`P`) maps all predicates, **Common subjects and objects** (`SO`) organizes all terms which play both subject and object roles in the dataset, **Subjects** (`S`) organizes all subjects which do not play an object role, and vice versa for **Objects** (`O`). An internal subdivision is then performed by attending to the classes (U, B, L) that each partition can store. The `O` partition contains URIs and bnodes, but also an area for literals in which specific representations for *general*, *lang-tagged* and *datatype-tagged* literals are maintained. A very small mapping structure (referred to as *ptrs*) allows for integrating all partitions in a single dictionary. Each cell in the first-level records 1) a pointer to its corresponding sub-dictionary, and 2) the number of terms previously represented. In addition, *ptrs* stores two simple indexes pointing to the beginning of each language and datatype used in the dictionary.

This organization provides a set of interesting features:

- The `SO` partition allows terms playing subject and object roles to be represented once. This decision achieves spatial savings; up to 60% of the strings may be in `SO` as shown in current results [24].
- Due to the limited number of predicates, their individual mapping allows for representing them with fewer bits.
- The hierarchy delimits the ranges for a given language or datatype, allowing string tags to be removed in the final literal representation and filtering operations to be performed on the dictionary.

[4]Note that RDF is typically formalized as follows. Assume infinite, mutually disjoint sets U (RDF URI references), B (Blank nodes), and L (RDF literals). A triple $(s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$ is called an RDF triple [20].

Nevertheless, the most important property of this organization is that it allows to choose the best technique for each partition, *i.e.* to adapt each sub-dictionary in accordance with its features. For instance, URIs and bnodes tend to share long prefixes, whereas literal features are strongly related to the knowledge represented in the dataset. Specific dictionary techniques might be used to represent the terms for each class within each partition. This implies that each dictionary handles its specific mapping, so each term is locally identified within it.

We propose to adapt techniques for string dictionaries [14] to the case of RDF. In particular, we highlight *Front-Coding* compression [31], which takes advantage of the existence of long common prefixes to obtain compact dictionaries; and *self-indexes* [25] which are general compressed indexes suitable to be adapted for representing dictionaries.

3.3 Triples Encoding

As stated, several Triples encodings are feasible with different trade-offs between the compression ratio (exchanging) and some natively supported operations over the triples (consumption). We propose an intuitive technique called Bitmap Triples (Figure 2). Starting from ID-triples (triples after ID replacement according to the dictionary), predicate and object adjacency lists can be created. These lists draw tree-shaped structures containing the subject ID in the root, the predicate IDs in the middle level, and the object IDs in the leaves. Bitmap Triples implements a compact mechanism which allows subjects to be implicitly represented by considering that the $i-th$ tree draws the adjacency list related to the $i-th$ subject. The integer sequences, $\mathcal{S}_p$ and $\mathcal{S}_o$, are used for storing respectively the predicate and the object IDs within the adjacency lists. Two additional bitsequences: $\mathcal{B}_p$ and $\mathcal{B}_o$ (storing list cardinalities) are used for delimitation purposes.

Enhanced bitmaps [19], supporting `rank/select` operations in constant time, are required for accessing the graph. These two operations enable graph structure traversing and allow some SPARQL triple pattern queries to be performed. For instance, the presented approach efficiently and natively resolves the triple patterns (S, P, O), $(S, P, ?O)$, $(S, ?P, ?O)$, and $(S, ?P, O)$.

4. METHODOLOGY

We have devised an incremental cycle for our research, grounded in five major subjects.

RDF structure in theory and practice. Skewed power-law distributions present in RDF data [16] should establish the basis of any compact RDF solution. However, little work has been done to understand the RDF essence before researching or applying this data model [29, 22]. In this thesis, we firstly propose a detailed study of the most important trends to get a global understanding of the real structure of RDF networks. The main objective is to isolate common features in order to achieve an objective characterization of real-world RDF data. This can lead to better dataset designs, as well as efficient RDF data structures, indexes and compressors.

Binary RDF Specification. The main objective is to design, analyze, develop and evaluate a binary RDF format with the following minimum requirements. For publishing, i) the format should rely on a clear scheme, providing a stan-

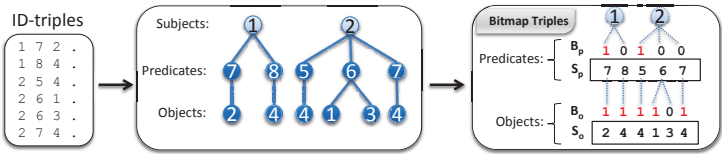

Figure 2: Description of Bitmap Triples.

dard way to add provenance and other metadata (such as internal structure and statistics or a summary of the content). For exchanging, ii) the representation should be based on compact structures, thus minimizing redundancy and saving transmission costs. Finally, in order to improve large RDF consumption, iii) the format should allow basic data operations (such as simple lookup operations) to be performed natively on the compact representation.

Succinct Dictionaries. RDF dictionary design firstly requires an empirical study characterizing the main features in real-world RDF datasets, *e.g.* the growth rate w.r.t. the number of triples, and the size of each dictionary subset (URIs, bnodes, literals). As *string dictionaries* are the natural precedent of RDF dictionaries, a comparison of these techniques should be performed, focusing on compressed dictionary techniques [14]. Finally, specific configurable techniques for RDF dictionaries must be proposed. The aim is to achieve highly-compressed RDF dictionaries with very efficient performance. Real-world and micro-benchmark evaluation must be considered.

Triples Indexes. Different approaches for triples indexes need to be proposed, studied and evaluated on real-world scenarios. Bitmap Triples achieves an interesting trade-off between compression and basic searching without the need for decompression. However, it cannot provide efficient full SPARQL resolution by itself. We plan to work on two aspects: i) there is much research on the application of compressed indexes [10] to RDF triples which can be serialized and included in the binary representation. ii) We also consider the exchange of a smaller index, but not a fully functional one for advanced querying, and to quickly construct a complementary index at the consumer. This might fulfill the lack of such indexes as Bitmap Triples, with no additional network overhead. Whereas, in the first case, efficiency is measured as a compressed ratio, in the latter, performance must be evaluated as a limited in-memory store.

Practical deployment. Both for Dictionary and Triples structures, there is a need to study the binary cost, *i.e.* the cost of binary compression, exchanging and consumption with respect to plain exchanging and consumption. Furthermore, we should evaluate the impact of the studied RDF structure features on both the binary representation and the final consumption/querying.

5. RESULTS

To date, several results have been achieved. The HDT format has recently been accepted as a **W3C Member Submission** [6], highlighting the relevancy of "efficient interchange of RDF graphs". The Submission specifies a vocabulary for the metadata of the Header, extending the VoiD vocabulary to the particularities of binary RDF. An open-source HDT implementation has also been developed[5].

[5] `http://www.rdfhdt.org/`

The logical Header-Dictionary-Triples organization was studied in [18], together with a specific algorithm which allows to `check&find` triple patterns on top of the Bitmap Triples representation. Latest results [24] involve the compression of RDF Dictionaries. We propose a specific combination of *Front-Coding* compression and *self-indexes* over the dictionary organization presented in Section 3.2. This achieves highly-compressed dictionaries supporting `locate` and `extract` at the level of microseconds. Table 1 gives a brief revision of the current achievements of HDT. We compare the HDT effectiveness with respect to some universal compressors directly applied on RDF datasets[6] serialized in N3. It is worth noting that our results (shown in column HDT) are obtained by combining the dictionary encoding from [24] and the Bitmap Triples representation from [18]. As can be seen, HDT always achieves the best compression ratios and allows dictionary management and triples querying to be directly performed without previous decompression. These on-going results can be tested through an end-user prototype [11] which implements this HDT configuration. It consumes RDF in HDT format and takes advantage of its machine-friendly features to offer a 3D visualization of the dataset and to support basic queries.

Last but not least, we showed in [17] that big RDF datasets are highly compressible due to the skewed structure of RDF graphs and the inherent verbosity of URIs and textual RDF syntaxes. This empirical analysis set the basic foundations of the work above. We have also started an on-going work of RDF structure characterization, proposing a set of initial metrics [18].

6. CONCLUSIONS AND FUTURE WORK

Web of Data suffer from diverse scalability problems when moving to an RDF data-intense processing era. Traditional verbose RDF formats remain one of the main bottlenecks for exchanging and post-processing. The on-going thesis addresses these problems by i) studying the underlying RDF structure essence, ii) proposing a novel RDF binary format (HDT) and iii) giving compact RDF dictionaries and triples structures which can be both efficiently serialized for exchanging and queried without decompression at the time of consumption.

Following the detailed methodology, our on-going work is firstly focused on getting a global understanding of the real structure of RDF networks. This study and characterization is intended to become a theoretical framework leading future researches. HDT format should also be strengthened to fulfill uncovered needs, such as a clear definition for RDF streaming. An additional line of future work focuses on evolving the compressed dictionary and triples to support full SPARQL at consumption, including filtering at the dictionary level. The support of dynamic operations of in-

[6] wikipedia (`http://labs.systemone.at`), dbtune and uniprot (`http://km.aifb.kit.edu/projects/btc-2010`) and dbpedia (`http://wiki.dbpedia.org`).

Dataset	Triples (millions)	Size (GB)	Compression (MB)		
			gzip	bzip2	HDT
wikipedia	47.0	6.88	491.04	360.01	**230.48**
dbtune	58.9	9.34	924.85	630.28	**462.31**
uniprot	72.5	9.11	1233.25	739.76	**481.34**
dbpedia-en	232.5	33.12	3513.58	2645.36	**2176.54**

Table 1: Compression results of HDT for several datasets.

serting, deleting, and updating binary RDF is also essential for efficient dynamic consumption.

Acknowledges

This work is funded by MICINN (TIN2009-14009-C02-02), the Regional Government of Castilla y León (Spain), Erasmus Mundus and the European Social Fund.

7. REFERENCES

[1] *Notation3*. W3C Design Issues. 1998. http://www.w3.org/DesignIssues/Notation3.

[2] *RDF/XML Syntax Specification (Revised)*. W3C Recommendation. 2004. http://www.w3.org/TR/rdf-syntax-grammar/.

[3] *SPARQL Query Language for RDF*. W3C Recommendation. 2008. http://www.w3.org/TR/rdf-sparql-query/.

[4] *Turtle - Terse RDF Triple Language*. W3C Team Submission. 2008. http://www.w3.org/TeamSubmission/turtle/.

[5] *Efficient XML Interchange (EXI) Format 1.0*. W3C Candidate Recommendation. 2009. http://www.w3.org/TR/2009/CR-exi-20091208/.

[6] *Binary RDF Representation for Publication and Exchange (HDT)*. W3C Member Submission. 2011. http://www.w3.org/Submission/2011/03/.

[7] D. Abadi, A. Marcus, S. Madden, and K. Hollenbach. SW-Store: a vertically partitioned DBMS for Semantic Web data management. *The VLDB Journal*, 18:385–406, 2009.

[8] K. Alexander. RDF in JSON: A Specification for serialising RDF in JSON. In *SFSW*, 2008.

[9] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets-On the Design and Usage of voiD, the'Vocabulary of Interlinked Datasets'. In *LDOW at WWW*, 2009.

[10] S. Álvarez García, N. Brisaboa, J. Fernández, and M. Martínez-Prieto. Compressed k2-Triples for Full-In-Memory RDF Engines. In *AMCIS, paper 350*, 2011.

[11] M. Arias, J. Fernández, M. Martínez-Prieto, and C. Gutierrez. HDT-it: Storing, Sharing and Visualizing Huge RDF Datasets. In *ISWC*, 2011.

[12] M. Atre, V. Chaoji, M. Zaki, and J. Hendler. Matrix "Bit" loaded: a scalable lightweight join query processor for RDF data. In *WWW*, pages 41–50, 2010.

[13] C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked Data On the Web (LDOW2008). In *WWW*, pages 1265–1266, 2008.

[14] N. Brisaboa, R. Cánovas, F. Claude, M. A. Martínez-Prieto, and G. Navarro. Compressed String Dictionaries. In *SEA*, pages 136–147, 2011.

[15] R. Cyganiak, H. Stenzhorn, R. Delbru, S. Decker, and G. Tummarello. Semantic sitemaps: Efficient and flexible access to datasets on the semantic web. In *ESWC*, pages 690–704. Springer-Verlag, 2008.

[16] L. Ding and T. Finin. Characterizing the Semantic Web on the Web. In *RISC*, pages 242–257, 2006.

[17] J. Fernández, C. Gutierrez, and M. Martínez-Prieto. RDF compression: basic approaches. In *WWW*, pages 1091–1092, 2010.

[18] J. Fernández, M. Martínez-Prieto, and C. Gutierrez. Compact Representation of Large RDF Data Sets for Publishing and Exchange. In *ISWC*, pages 193–208, 2010.

[19] R. González, S. Grabowski, V. Mäkinen, and G. Navarro. Practical Implementation of Rank and Select Queries. In *WEA*, pages 27–38, 2005.

[20] C. Gutierrez, C. Hurtado, A. Mendelzon, and J. Perez. Foundations of semantic web databases. *J COMPUT SYST SCI*, 77:520–541, 2011.

[21] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 2011.

[22] W. Hu, J. Chen, H. Zhang, and Y. Qu. How Matchable Are Four Thousand Ontologies on the Semantic Web. In *ESWC*, pages 290–304, 2011.

[23] D. Le-Phuoc, J. X. Parreira, V. Reynolds, and M. Hauswirth. RDF On the Go : An RDF Storage and Query Processor for Mobile Devices. In *ISWC*, 2010. Available at http://iswc2010.semanticweb.org/pdf/503.pdf.

[24] M. Martínez-Prieto, J. Fernández, and R. Cánovas. Compression of RDF Dictionaries. In *SAC*, pages 1841–1848, 2012.

[25] G. Navarro and V. Mäkinen. Compressed Full-Text Indexes. *ACM Computing Surveys*, 39(1):art. 2, 2007.

[26] T. Neumann and G. Weikum. The RDF-3X engine for scalable management of RDF data. *The VLDB Journal*, 19, 2010.

[27] M. Schmidt, M. Meier, and G. Lausen. Foundations of SPARQL query optimization. In *ICDT*, 2010.

[28] L. Sidirourgos, R. Goncalves, M. Kersten, N. Nes, and S. Manegold. Column-Store Support for RDF Data Management: not all Swans are White. *VLDB Endowment*, 1(2):1553–1563, 2008.

[29] Y. Theoharis, Y. Tzitzikas, D. Kotzinos, and V. Christophides. On Graph Features of Semantic Web Schemas. *IEEE Trans. on Know. and Data Engineering*, 20(5):692–702, 2008.

[30] J. Urbani, J. Maassen, and H. Bal. Massive Semantic Web data compression with MapReduce. In *HPDC 2010*, pages 795–802, 2010.

[31] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes : Compressing and Indexing Documents and Images*. Morgan Kaufmann, 1999.