Compact Representation of Large RDF Data Sets for Publishing and Exchange [⋆] [⋆⋆]

Javier D. Fernández[1], Miguel A. Martínez-Prieto[1,2], and Claudio Gutierrez[2]

[1] Department of Computer Science, Universidad de Valladolid (Spain)
{jfergar,migumar2}@infor.uva.es
[2] Department of Computer Science, Universidad de Chile (Chile)
cgutierr@dcc.uchile.cl

The current "Web of Data" produces increasingly huge RDF data sets. However, traditional RDF representations stay influenced by the *old* document-centric perspective of the Web, containing high levels of redundancy and verbose syntaxes. This leads to fuzzy publications, inefficient management, complex processing and lack of scalability.

We introduced a new representation format (*Header-Dictionary-Triples*: `HDT`[3]) that modularizes the data and uses the skewed structure of big RDF graphs [2] to achieve large spatial savings. It is based on three main components, described as follows.

Header. This component includes logical and physical metadata describing the RDF data set. It serves as an entrance point to the information on the data set. The Header is downloaded or queried online by the consumer. For instance, the consumer can see a detailed summary of the published data set, which might be (i) distributed in several chunks, (ii) duplicated in different formats or (iii) duplicated under different versions.

We consider the Header as an RDF graph. This allows expressing metadata about the data set with an RDF syntax, which can be used through well-known mechanisms, such as SPARQL Endpoints. We provide an extension of VoiD[4] for HDT Headers[5].

Dictionary. The Dictionary component assigns a unique ID to each element in the data set. This way, the dictionary contributes to the goal of compactness. We provide a basic implementation considering four subsets; (i) shared subjects-objects, (ii) unique subjects, (iii) unique object and (iv) predicates, as shown in Figure 1.

Triples. This component transforms a stream of strings into a stream of IDs, by means of the dictionary. In addition to its compact ability, it allows to access and query the RDF graph. We provide three implementations for Triples encoding, as shown in Figure 1. **Plain Triples** is the most naive approach in which only the ID substitution is performed. **Compact Triples** implies a triple sorting by subject and the creation of predicate and object adjacency lists. The first stream of *Predicates* corresponds to the

[⋆] In Proc. of 9th *International Semantic Web Conference (ISWC 2010)*, LNCS 6496, p. 193–208, Springer-Verlag, 2010. Impact metrics: ERA rank B (Data Format), 22% acceptance rate.

[⋆⋆] Partially funded by MICINN (TIN2009-14009-C02-02), ICDB (Grant ICM P05-001-F), and Fondecyt 1090565. The first author is granted by the JCyL (Spain) and the ESF.

[3] `HDT` is a W3C Member Submission: `http://www.w3.org/Submission/2011/03/`

[4] `http://www.w3.org/2001/sw/interest/void/`

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

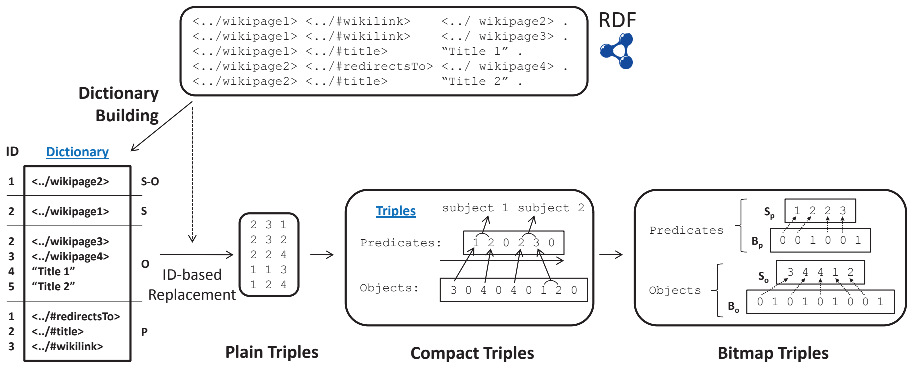

Fig. 1. Incremental representation of an RDF data set with HDT

Data set	Triples (millions)	Size (MB)	HDT		Universal Compressors		
			Plain	Compress	gzip	bzip2	ppmdi
Billion Triples	106.9	15081.74	*31.87%*	**3.91%**	9.54%	6.83%	5.32%
Uniprot RDF	79.2	7083.22	*14.33%*	**3.23%**	8.71%	5.04%	3.99%
Wikipedia	47	6882.20	*6.62%*	**2.22%**	6.97%	5.11%	4.10%

Table 1. Compression results.

lists of predicates associated with subjects, maintaining the implicit grouping order. The end of a list of predicates implies a change of subject, marked with a separator, *e.g.* the non-assigned zero ID. The second stream (*Objects*) groups the lists of objects for each pair (s, p). **Bitmap Triples** implementation extracts the auxiliary zero values embed in each stream. These values are kept in two bitmaps in which each `1` value marks the end of the corresponding adjacency list. The bitsequences can support `rank/select` operations in constant time [1], which constitutes the basis for accessing the graph.

HDT with Bitmap Triples (referred to as `Plain-HDT`) is even more compressible. `HDT-Compress` makes specific decisions: (1) We take advantage of repeated prefixes using a predictive high-order compressor such as PPM to encode the dictionary. (2) We take advantage of the power-law distribution of the streams with a `Huffman` code.

Table 1 compares HDT with respect to three well-known universal compressors within an heterogeneous corpora. `HDT-Compress` achieves the most effective results with ratios between $2-4\%$. In turn, `HDT-Compress` also outperforms universal compressors by improving the best results, achieved on `ppmdi`, of between $20-45\%$.

HDT provides clean publication and efficient exchange by approaching a more compact representation format. We are currently exploring new dictionary and triples implementations leading to efficient query resolution.

References

1. D. Clark. *Compact PAT trees*. PhD thesis, University of Waterloo, 1996.
2. 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.