storage footprint ratio to raw source data

3 posts by 2 authors

Bo Ferri Apr 27

Hi all,

we try to do some MDM in our open-source project [1,2]. Currently, we do this with the help of Neo4j as datahub for all the content [3]. I'm really wondering a bit about the storage footprint that will be created, when we load the content into the database. Right now, we can process arbitrary XML and CSV data. We transform the raw content into a neutral data model [4] that is based on top of RDF (without losing any information from the original source). The footprint ratio is somehow exploding in the database. Here, are some excerpts:

raw = 30MB
db = 2261MB
index* = 353MB

nodes= ~1.4M
relationships= ~1.9M
properties= ~7.3M

db/raw = 75x
index/raw = 11.76x
index/db = 15,6%

Some reasons are that we add various metadata that have its origin in the neutral data model (where we make use of URIs etc.) and some metadata for versioning information (two int values at each relationship). However, generally the ratio looks really high. Can you explain this somehow?

Thanks a lot in advance.

Cheers,

Bo

*) index size is also include in db site

[1] http://dswarm.org
[2] https://github.com/dswarm
[3] https://github.com/dswarm/dswarm-graph-neo4j
[4] https://github.com/dswarm/dswarm-documentation/wiki/Graph-Data-Model

Bo Ferri Apr 27

Okay,

I think the main increase comes from the transformation of the original data (CSV or XML) to our neutral data format (GDM). I did further testing (also with more data) and got a ratio for GDM/raw of ~11x. Whereby, the ratio DB/GDM is 5.4x. We try to implement a URI pre and post processing to reduce the size in the graph db.

Cheers,

Bo
- show quoted text -

Michael Hunger Apr 28

Hi Bob,

I think most of the increase is due to the RDF model.

I suggest you revisit the model so that attributes that belong to an entity are really stored as properties of that entity (node or relationship) and only semantic links between objects are created as relationships.

For many attributes the total normalization of the RDF model is not needed. OTOH it also provides a better read and update performance in Neo4j if you're closer to the property graph model.

That said for attribute where it makes sense to treat them as separate nodes, feel free to do so, just not as a general rule.

HTH

Cheers, Michael

- show quoted text -
- show quoted text -
--