ontotext

Claiming that RDF triplestores are typically used for offline analytics suggests unfamiliarity with their most popular use cases. Triplestores are often applied in very dynamic operational database setups such as metadata-based content management at the world's largest media and publishers such as BBC, FT, Wiley, Elsevier, Oxford University Press and DK.

As new approaches to data management are gaining popularity, we start seeing more texts that compare the different NoSQL and particularly graph database engines. A recent example is "Graph Databases for Beginners: Other Graph Data Technologies".

While such comparisons do a great job helping developers understand "how stuff works", sometimes they tend to be imprecise when authors comment engines beyond their core area of expertise.

The above-mentioned post makes statements about triplestores (also called semantic graph databases) like these:

> However, triple stores are not "native" graph databases because they don't support index-free adjacency, nor are their storage engines optimized for storing property graphs.

and

> … the most common use case for triplestores is offline analytics rather than for online transactions.

And so my 20+ years of piled up expertise urge me to comment.

To provide a bit of background, let's start with:

- Triplestores are graph database engines that, unlike engines based on property graphs, implement a set of comprehensive, vendor-independent standards: RDF (the data model), RDFS and OWL (schema languages), and SPARQL (query language).

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

- Triplestores work with globally unique identifiers – together with few other features, which makes them very suitable for integration of data – be it the thousands Linked Open Data (LOD) datasets or proprietary data.

Index-free Adjacency in Triplestores

Now we'll dive deeper into the theory of data representation and indexing, so if you want to understand how these are actually implemented, don't skip this section.

Indeed, traversal from one node of the graph to another is not the most typical operation for triplestores, so, many triplestores do not provide efficient support for it out of the box. Still, the leading triplestores can be configured so that such operations are efficiently supported.

The leading triplestores can be configured to efficiently support graph-traversal. Share on X

To understand how triplestores work, I will provide a quick intro to the most typical designs.

Most triplestores have some sort of dictionaries, which assign each node in the graph an integer number as an internal identifier. Technically, they map the actual entity identifiers (URIs such as *"http://company.com/data/person.101"*) and literals (such as *"Frank Lampard"* and *"2015-02-28T23:39:07Z"^^xsd:dateTime*) to an integer number unique for the database instance.

The most popular such index is PSO (Predicate Subject Object), where triples are ordered first by their predicate (the type of the relationship), then by subject (the end node) and, finally, by the object (the start node). Each of the elements of the triple is represented in such an index by its internal integer IDs for efficiency purposes.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

Predicate	Subject	Object
has_number	Jason Hernandez	2
has_number	Kwame Watson-Siriboe	3
has_number	David Villa	7
has_number	Frank Lampard	8
has_number	Mix Diskerud	9
has_number	Ned Grabavoy	10
has_position	Jason Hernandez	Defender
has_position	Kwame Watson-Siriboe	Defender
has_position	David Villa	Forward
has_position	Frank Lampard	Midfielder
has_position	Mix Diskerud	Midfielder
has_position	Ned Grabavoy	Midfielder
plays_for	Jason Hernandez	NYC FC
plays_for	Kwame Watson-Siriboe	NYC FC
plays_for	David Villa	NYC FC
plays_for	Frank Lampard	NYC FC
plays_for	Mix Diskerud	NYC FC
plays_for	Ned Grabavoy	NYC FC

The PSO index handles efficiently queries where the predicate and the subject are known, e.g.:

```
SELECT ?team WHERE { :Frank_Lampard :plays_for ?team}
```

and even when only the predicate is known:

```
SELECT ?who ?team WHERE { ?who ::plays_for ?team}
```

Usually, triplestores maintain several such indices, to be able to efficiently deal with different

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

triple patterns. The concrete indices to be used are easy to configure in accordance with the typical loads and the performance requirements for the database instance.

Note that triplestores do not “store” triples for the sake of storing them. Indexing triples in PSO and other similar indices is also the way to store them. Each triple is stored in each of the indices, which is not a problem because its internal representation by integer IDs is sufficiently compact.

To support efficiently graph traversal from one node into another, a triplestore needs to be configured to enable its subject-object-predicate (SOP) index. With this index switched on, a triplestore becomes a de-facto “index-free adjacency” engine. One can consider the SOP index being “the storage” of the graph database.

There are multiple deployments of triplestores that are tuned this way and do support efficient graph traversal.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

Subject	Object	Predicate
David Villa	7	has_number
David Villa	Forward	has_position
David Villa	NYC FC	plays_for
Frank Lampard	8	has_number
Frank Lampard	Midfielder	has_position
Frank Lampard	NYC FC	plays_for
Jason Hernandez	2	has_number
Jason Hernandez	Defender	has_position
Jason Hernandez	NYC FC	plays_for
Kwame Watson-Siriboe	3	has_number
Kwame Watson-Siriboe	Defender	has_position
Kwame Watson-Siriboe	NYC FC	plays_for
Mix Diskerud	9	has_number
Mix Diskerud	Midfielder	has_position
Mix Diskerud	NYC FC	plays_for
Ned Grabavoy	10	has_number
Ned Grabavoy	Midfielder	has_position
Ned Grabavoy	NYC FC	plays_for

Triplestores as Dynamic Operational Databases

Triplestores are often used for dynamic operational databases. Plenty of such applications can be found in publishing and media, where triplestores are used for dynamic management of content, based on rich metadata descriptions.

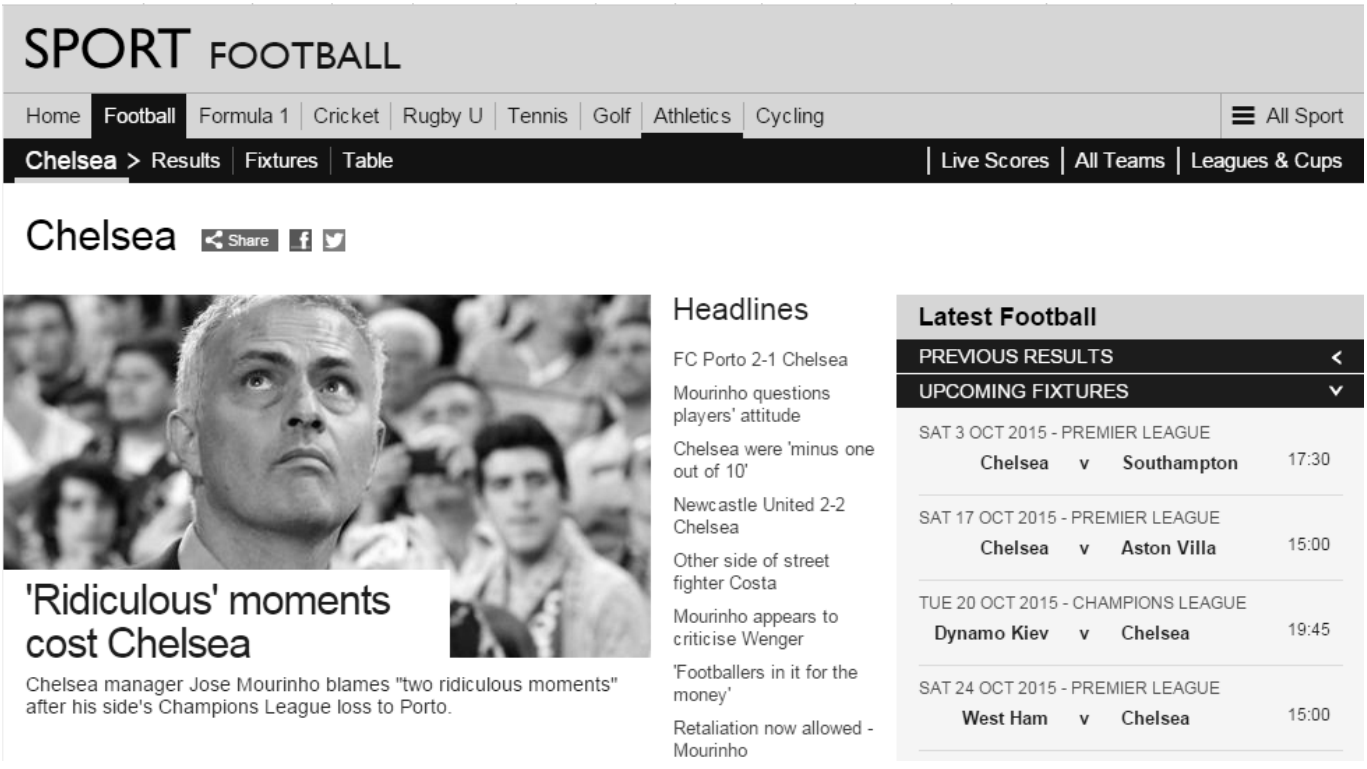
This usage pattern is known as dynamic semantic publishing and it is embodied in the LDBC Semantic Publishing Benchmark (SPB). In SPB news, images and other “creative works” are described with metadata: namely, Dublin Core-like attributes and links to entities and

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

concepts that are most relevant to them. Entities and concepts are described as reference data: a huge Knowledge Graph derived from Linked Open Data datasets such as DBpedia, GeoNames and others.

Both the metadata and the reference data are stored in a triplestore that is accessed by two types of clients (agents):

- Aggregation agents retrieve information on specific subjects. For instance, at the Sports section of the website of BBC, each topic web page (e.g., the one for Chelsea) is dynamically generated by several SPARQL queries to the underlying triplestore;
- Editorial agents are constantly making changes to the database, either inserting metadata for the newly coming content or updating the reference data (e.g., the number of goals scored by Frank Lampard this season).



In dynamic semantic publishing scenarios, triplestores typically handle hundreds of read

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

queries per second, while in parallel processing tens of update transactions per second for Knowledge Graphs that contain hundreds of millions of… Share on X

These are real statistics from mission-critical deployments like the triplestore behind BBC SPORT – a very dynamic operational database backing the website 24×7 since the year 2012.

How To Evaluate A Graph Database

The Linked Data Benchmark Council (LDBC) is an industry consortium governing and developing TPC-like benchmarks for graph databases and triplestores. Its members include leading vendors in both fields, e.g., IBM, ORACLE, Neo Technologies, OpenLink Software, Ontotext and others. At the LDBC's website, one can find benchmarks and benchmark results alongside blog posts on related subjects and event announcements.

The Dynamic Semantic Publishing (DSP) application pattern was invented and first implemented by BBC's team for their website for the FIFA World Cup 2010. A great blog post describing this project was published by Jem Rayfield. One can read about the dynamic semantic publishing use case and the Semantic Publishing Benchmark in the blog post that I wrote earlier this year.

On the technology side in DSP, the graph database engine needs to interplay closely with text-mining technology used for automated metadata generation. Particularly when Linked Open Data is used for text analytics and tagging purposes. This is something that triplestores have proven to do very well – not a surprise given that LOD comes in RDF.

In April this year, Philip Howard from Bloor Research completed a report on the graph database market. One can download the full version of the "Graph and RDF databases 2015" report. Philip also refers to Property Graphs as "operational graph databases" and considers triplestores incapable of "index-free adjacency", reflecting historical trends and attitudes.

The summary about RDF databases there is as follows:

> Often semantically focused… for use in operational environments but have inferencing capabilities. Require indexes even in transactional environments.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

> Often ACID compliance.

Back in January, Robin Bloor, published "The Graph Database and the RDF Database" that provides a number of good insights about the differences and commonalities between RDF databases and Property Graph databases:

> Where the RDF databases really score is when you want to do set processing (a la SQL) at the same time that you want to do graph processing. Consider a query such as "*Who are the biggest influencers on Twitter over the past six months?*"

Both the RDF and property graph databases would handle such a query and return the same results quickly. But if you ask the very different question: *"Which influencers have had the same pattern of influence on Twitter over the last six months?",* you are asking for both graph processing and set processing at the same time to get to the answer, and the RDF databases do both well. Not only that, but this is an area of analytics, which was virtually untapped until recently because there was no software that could easily do it.

Final Words

The popularity of graph databases is growing based on good track record of projects where these engines delivered to the expectations.

That's true for all types of graph database technology: property graphs, RDF, and other graph analytics. People start paying more attention to the differences between the different graph database standards in order to choose the one most appropriate for their application.

In this post, I provided some insights on how triplestores work and how they can support graph-traversal efficiently, although "index-free adjacency" is not central to their design. I also presented the "dynamic semantic publishing" pattern – a typical use case where triplestores are used as a dynamic operational database.

I also provided references to recent posts and reports that touch the RDF vs. Property Graphs subject. The later excel in graph analytics, although triplestores can do this, too.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

I will summarize the advantages of RDF-based graph engines as follows:

- Support for comprehensive vendor-independent standards
- Good for data integration (both Linked Open Data and other)
- Inference support
- Can do graph processing and set processing at the same time
- Text-analysis using Linked Open Data

Want to learn more about graph databases like Ontotext's GraphDB?

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com