Introduction

An ontology or a knowledge graph of any appreciable size requires some effort on the part of the consumer before it becomes a useful tool. Knowledge of the subject domain is always helpful, but rarely sufficient as there are many choices to be made in representing domain knowledge as data in a graph. Aligning the shape of the domain knowledge in your mind with the shape of the knowledge represented in the graph is essential. In this article we will discuss some of the features of GraphDB that support such an alignment.

FIBO Overview

The Financial Industry Business Ontology (FIBO) is a conceptual model of the financial industry that has been developed by the Enterprise Data Management Council (EDMC). The EDMC supports an open process for the continuing maintenance and development of FIBO. The goal of FIBO is to provide precise meaning to the data artifacts that describe the business of finance independent of the form the data artifacts may take. FIBO contains the entities and associations that describe the information needed to build, extend and integrate financial business applications. FIBO is specified using RDF(S) and OWL. It is therefore amenable to analysis using SPARQL and OWL inference. In the version used for this article (2020 Q2 Production), it contains:

- 122 namespaces, which represent the module structure;
- 1,542 classes;
- 1,328 concepts;
- 535 predicates.

Since its initial release in 2017, it has grown to include and align with a number of existing standards and has benefited from broad finance industry participation. From its roots in an Excel workbook known as the Semantic Repository, FIBO has grown into a sophisticated ontology based upon RDF and OWL. On its way, the process has produced a number of beneficial side effects including ontology engineering best practices such as textual stability for RDF so that traditional text-based version control systems may be employed, rigorous metadata standards resulting from the close relationship the effort has enjoyed with the Object Management Group (OMG), and rigorous use of OWL's reasoning capabilities. More

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

details can be found here.

The content of FIBO is made available in a variety of vehicles. There is an RDF and OWL ontology, which is the core entity containing the business knowledge that is made available in several serializations including: RDF-XML, Turtle, JSON-LD, and N-Quads/N-Triples. In addition there are:

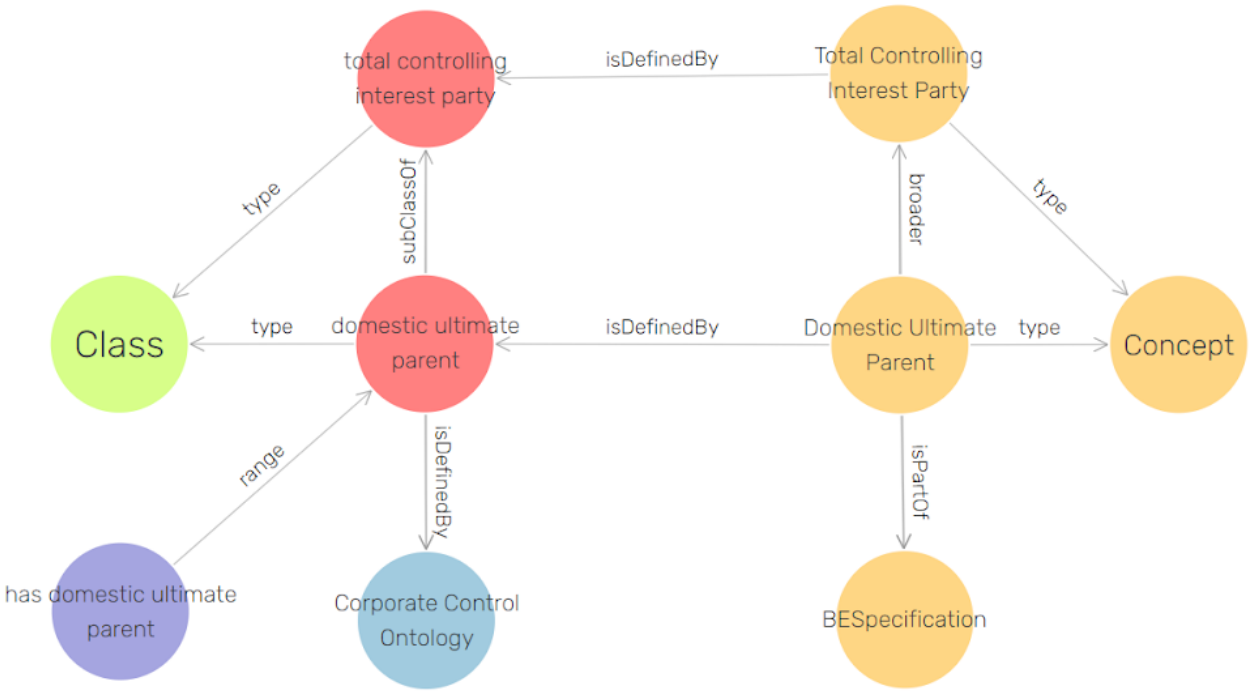
- FIBO Vocabulary SKOS-based taxonomy for use with taxonomy management tools that is available in RDF-XML, Turtle, and JSON-LD serializations.
- FIBO data dictionary that is available as .csv and .xlsx. It contains the operational class in FIBO and their accompanying properties.
- FIBO-DM is an Enterprise Data Model available as SAP PowerDesigner Conceptual and Logical Data Models.

In this article we will focus on the FIBO ontology and vocabulary as they are both encoded using RDF and therefore amenable to analysis using SPARQL and OWL reasoning.

A Tale of Two Hierarchies

All of the concepts contained in the FIBO vocabulary are defined by entities in the FIBO ontology. This provides rich contextual information that an application using the vocabulary can make available to the user. For example, fibo-v-be:DomesticUltimateParent is defined by the entity fibo-be-oac-cctl:DomesticUltimateParent.

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



Concepts in FIBO vocabulary participate in a hierarchy defined by the *skos:broader* and *skos:narrower* predicates. In the figure above, Total Controlling Interest Party is a broader concept than Domestic Ultimate Parent. In the vocabulary as published, only *skos:broader* is used. As stated in the SKOS specifications, *"the properties skos:broader and skos:narrower are each other's inverse. Whenever a concept X is broader than another concept Y, then Y is a narrower concept of X according to the SKOS data model"*. Hence, there is an implied *skos:narrower* relation from *fibo-v-be:TotalControllingInterestParty* to *fibo-v-be:DomesticUltimateParent*.

Given the hierarchy in the vocabulary and the hierarchy in the ontology, what are the relationships between the elements of the hierarchy? Does each super-class of *fibo-be-oac-cctl:DomesticUltimateParent* have corresponding concepts in the vocabulary? Do those concepts participate in the *skos:broaderTransitive*/*skos:narrowerTransitive* transitive hierarchy with *fibo-v-be:DomesticUltimateParent*? In the rest of this post we will use GraphDB to answer these questions.

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

Loading FIBO in GraphDB

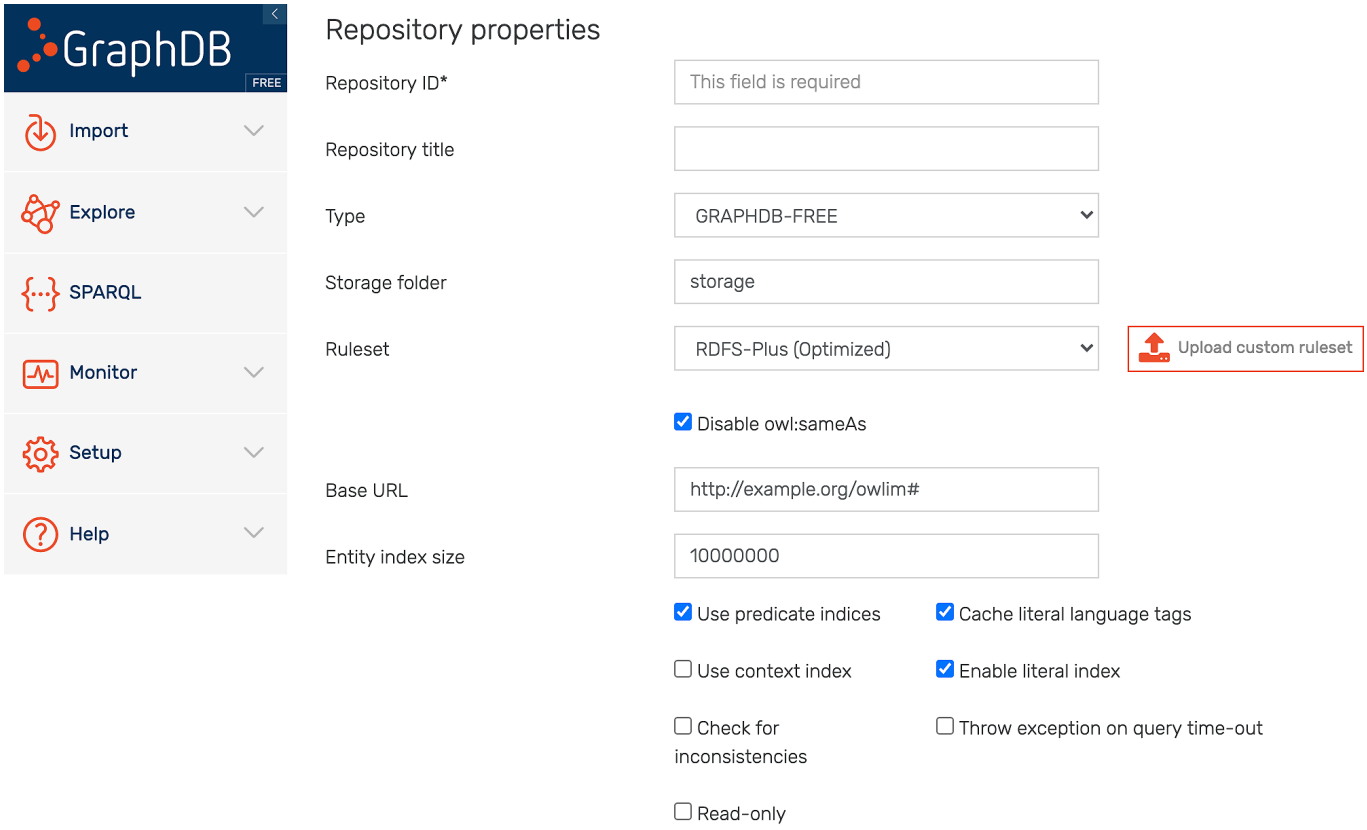
GraphDB is a scalable high performance triple store from Ontotext, formerly known as OWLIM. In its current incarnation, version 9.4.1, it supports RDF 1.1, SPARQL 1.1 and OWL 2 reasoning in addition to a number of product-specific tools for navigation, visualization, analysis and federation. It provides web accessible APIs (including the SPARQL Protocol for end-points) and hence, can be used with any programming language. As you will see in the following section, support for SPARQL 1.1, specification property paths, works hand in glove with OWL 2 reasoning.

The first step in using GraphDB is to create a repository. This is accomplished by navigating to the Setup -> Repositories > Create Repository form. The critical fields in this form are:

- Repository ID, in our case we will enter `FIBO`.
- Ruleset, which in our case should be set to `OWL 2-RL (Optimized)` selected from the menu.
- The “Use context index” box should be checked as FIBO Ontology contains named graphs that reflect the modular structure of the ontology.

The remaining fields on the form can be left with their default values as shown on the following screenshot:

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



The next step is to import the RDF graphs. There are three that are required:

1. From the EDMC FIBO ontology site we need the FIBO Production zipped N-Quads distribution. The specific version used in the article is 2020 Q2 Production.
2. From the EDMC FIBO vocabulary site we need the FIBO-V Production zipped TTL distribution. The specific version used in the article is 2020 Q2 Production.
3. From the W3C we need the SKOS Simple Knowledge Organization System http://www.w3.org/2004/02/skos/core#. The W3C implements HTTP 303 redirect capability. As a result, this URI will produce HTML when referenced in a web browser and RDF when imported into GraphDB.

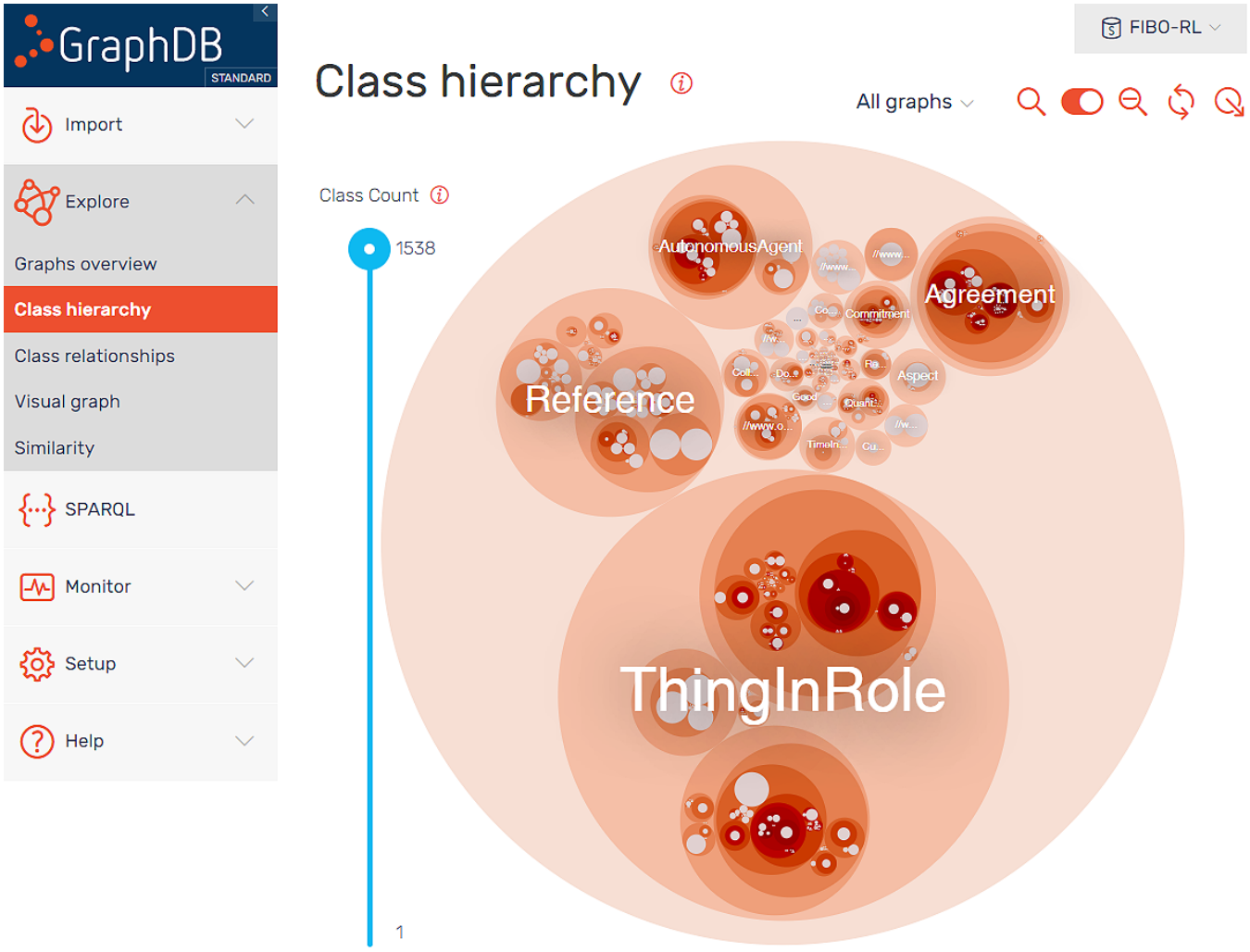
For the sake of efficiency, it is better to download the FIBO components to a local disk, preferably into the GraphDB import directory while SKOS can be easily downloaded over the internet.

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

Importing the vocabulary from disk takes one second, while importing the ontology takes one minute and ten seconds. Importing SKOS over the internet takes two seconds. In terms of performance, it should be noted that it is during import processing that inference is performed. The vocabulary is based upon SKOS, which makes very light use of the structuring elements that generate new triples and so the vocabulary and the SKOS graph are imported quickly. The ontology takes a good deal longer due to the sophisticated use of OWL employed in its construction. As a result, the 106,187 explicit statements from the imported graphs result in 405,493 inferred statements for a grand total of 511,680 statements. It should be noted that all of the inferred statements are created in the default graph, which also contains all of the statements from the named graphs [1].

One can get an overview of the ontology using the Class hierarchy diagram (from the Explore menu of the GraphDB Workbench), which represents the sub-classes as circles nested in their super-classes:

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



Reasoning in GraphDB

The OWL 2 Web Ontology Language Profiles (Second Edition) specification supports a number of different reasoning regimes. GraphDB provides implementations for few of these language profiles. In GraphDB the language profile is determined by a ruleset and it is specified at the repository level. The reasoner (a proprietary rule engine) is invoked whenever triples are added to the repository, either via SPARQL INSERT operation or by importing graphs. The effect of any of the rulesets, other than Empty (no inference), is to cause additional implicit triples to be materialized and stored, beyond the explicitly inserted ones. A unique feature of

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

GraphDB is that upon commit of SPARQL DELETE operation, the inferred statements, which are no longer inferreable, would be retracted. The nature of the triples created is dependent upon the content of the repository and the rules specified in the selected ruleset. For example, if the definition of two predicates indicates that they are inverse to each other then when one of them is encountered in a triple, a triple with the corresponding inverse property will be created.

To work with FIBO we chose the OWL 2 RL language profile, which is aimed at applications that require scalable reasoning without sacrificing too much expressive power. We experimented loading FIBO with the RDF Schema (RDFS) ruleset, which is much simpler and provides only support for rdfs:subPropertyOf, rdfs:subClassOf, rdfs:domain and rdfs:range. As expected, loading data with RDFS reasoning was much quicker: it took less than a second to load the FIBO ontology NQ file, versus more than a minute with OWL 2 RL. It also inferred only 170,804 implicit statements (more than 2 times less than OWL 2 RL). The drawback is that with RDFS we omit some important inference. For instance, the following query (executed against the RDFS repository) will extract those subclass relationships, which appear in OWL 2 RL repository, but are missing in RDFS:

```
SELECT * WHERE {
  { SERVICE <repository:FIBO-RL> {
      ?sub_class rdfs:subClassOf ?super_class
      FILTER(?sub_class != ?super_class)
      FILTER(?super_class != owl:Thing)
      FILTER(contains(str(?sub_class),'fibo')
          && contains(str(?super_class),'fibo'))
  }  }
  FILTER NOT EXISTS {
          ?sub_class rdfs:subClassOf ?super_class
  }
}
```

Here we use the so-called internal federation of GraphDB, which allows for efficient querying of data across repositories of one and the same database instance. Here follow the subclass relationships, which are missing in the repository with RDFS reasoning:

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

sub_class	super_class
fibo-fnd-qt-qtu:Rate	fibo-fnd-utl-alx:Ratio
fibo-fnd-qt-qtu:Rate	fibo-fnd-utl-alx:Expression
fibo-fnd-qt-qtu:Rate	fibo-fnd-utl-alx:StatisticalMeasure
fibo-fnd-utl-alx:Ratio	fibo-fnd-qt-qtu:Quantity
fibo-fnd-utl-alx:Ratio	fibo-fnd-qt-qtu:Rate
fibo-fnd-oac-oac:OwnershipAndControl	fibo-fnd-oac-ctl:Control
fibo-fnd-oac-oac:OwnershipAndControl	fibo-fnd-pty-pty:Situation
fibo-fnd-oac-oac:OwnershipAndControl	fibo-fnd-oac-own:Ownership
fibo-fnd-agr-ctr:TransferableContract	fibo-fnd-agr-ctr:WrittenContract
fibo-be-oac-cpty:MajorityControllingParty	fibo-be-oac-cctl:Affiliate
fibo-be-oac-cctl:ControlledCompany	fibo-be-oac-cctl:Affiliate
fibo-der-rtd-irswp:FixedFloatSingleCurrencyInterestRateSwap	fibo-der-rtd-irswp:SingleCurrencyInterestRateSwap
fibo-der-rtd-irswp:FloatFloatCrossCurrencyInterestRateSwap	fibo-der-rtd-irswp:CrossCurrencyInterestRateSwap
fibo-der-rtd-irswp:FloatFloatSingleCurrencyInterestRateSwap	fibo-der-rtd-irswp:SingleCurrencyInterestRateSwap

Fundamentally, this is because the definitions of the concept include constructs, which an RDFS reasoner cannot deal with. As a simple example, classes Rate and Ratio are defined to be equivalent with the following statement

```
fibo-fnd-qt-qtu:Rate owl:equivalentClass fibo-fnd-utl-alx:Ratio
```

In all language profiles of OWL 2 this implies that they are sub-classes of one another, but this is not part of the RDF Schema semantics.

Augmenting Reasoning with Property Paths

Property Paths are a feature of SPARQL that provide the capability to span the nodes in an RDF graph beyond the basic triple. A triple is the simplest property path available in SPARQL.

In the FIBO ontology the class hierarchy is represented by a transitive predicate, *rdfs:subClassOf*. The FIBO vocabulary concept hierarchy has concepts in an intransitive predicate, skos:broader. This means that the class hierarchy of the ontology, one that corresponds to the most common notion of class hierarchies in programming languages such as JAVA, PYTHON and C++, and specification languages such as UML, is being mapped to a

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

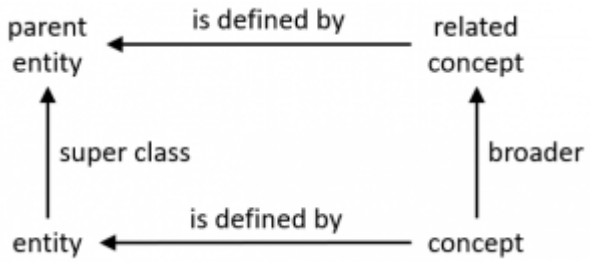
predicate which, when the specified semantics is strictly adhered to means less than the authors intended. The mapping is lossy as the intended intension is not accurately represented.

The consumer of the vocabulary that suffers as a result, is one who only requires the vocabulary, for the purposes of providing a broader context for their own domain specific vocabulary. This user must distinguish between the structure of their vocabulary and the structure of FIBO's vocabulary. Because the hierarchy implicitly mirrors the transitive class structure of the ontology, integration at any point requires that the external vocabulary become committed to/entangled with the implicit FIBO hierarchy.

Because the intention of the FIBO ontology's hierarchy actually maps to *skos:broaderTransitive* and *skos:narrower_transitive* predicates, one can ameliorate the mapping issue by the use of inference. By using a GraphDB repository with the OWL 2 RL (Optimized) ruleset, all of the necessary triples for a clean interface are created. At the top of the SKOS semantic property hierarchy is *skos:semanticRelation*. This predicate provides the structure necessary for visual representation and navigation of the vocabulary's contents. The s*kos:broader* and *skos:narrower* triples are available for low ontological commitment to FIBO. The *skos:broader_transitive* and *skos:narrower_transitive* are available for high ontological commitment to FIBO. The choice is left to the consumer and can be actualized piecemeal or by policy at the consumer's discretion using SPARQL.

A concrete example of the utility of property paths and inference is a lint check of the two hierarchies. By definition, every vocabulary concept is defined by an entity in the ontology. What entities participate in the class hierarchies of the mapped concepts that don't have related vocabulary entries themselves?

Structural Integrity Constraints



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

Analyzing the Graphs with SPARQL

This query finds classes that are connected to a concept trough sub-classes in the class hierarchy, but lack a connection to it through the concept hierarchy of the FIBO vocabulary.

Lint Query

```
SELECT DISTINCT ?parentEntity  where {
    ?concept a skos:Concept ;
             rdfs:isDefinedBy ?entity .
    # Every concept is defined by an entity
    ?entity rdfs:subClassOf ?parentEntity .
    # Exclude restrictions
    FILTER(ISIRI(?parentEntity))
    # Only consider resources in the FIBO namespaces
    FILTER(CONTAINS(str(?parentEntity),'fibo'))
    FILTER NOT EXISTS {
        # Find where there is no semantic relation
        # between concept and related concept
        ?relatedConcept rdfs:isDefinedBy ?parentEntity .
        # Consider the entire set of
        # related concepts in the hierarchy
        ?concept (skos:semanticRelation)+ ?relatedConcept
    }
}
```

The use of property paths with *skos:semanticRelation* provides one half of the round trip between the vocabulary and the ontology, *rdfs:subClassOf* provides the other.

The plus sign (+) after *skos:semanticRelation* indicates that this predicate is to be used as a path that connects the subject and object of the path by one or more matches of *rdfs:subClassOf*. There are a variety of operations that can be applied in property paths, a discussion of which is outside the scope of this article. The interested reader should consult SPARQL 1.1 Query language W3C Recommendation 21 March 2013.

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

Now that we have a repository containing the explicit and implicit RDF statements for the FIBO ontology and the FIBO vocabulary, we can execute the lint query, which produces a list of classes that do not meet the above presented integrity check: super-classes should be linked to concepts that are related in the vocabulary to concepts linked to the corresponding sub-classes.

parentEntity
fibo-der-drc-swp:SwapLifecycleEventIdentifier
fibo-fbc-fct-bc:BusinessCenterCodeScheme
fibo-fbc-fct-breg:RegistrationAuthorityCode
fibo-fbc-fct-fse:FinancialServiceProviderIdentifierScheme
fibo-fbc-fct-rga:RegulationIdentificationScheme
fibo-fbc-fct-rga:RegulationIdentifier
fibo-fbc-fct-usjrga:FederalReserveDistrictIdentifier
fibo-fbc-fi-fi:SecuritiesTransactionIdentifier
fibo-fnd-arr-arr:CollectionConstituent
fibo-fnd-arr-arr:DatedCollectionConstituent
fibo-fnd-arr-arr:DatedStructuredCollection
fibo-fnd-arr-arr:Scheme
fibo-fnd-arr-arr:StructuredCollection
fibo-fnd-dt-fd:CombinedDateTime
fibo-fnd-gao-gl:Goal
fibo-fnd-law-lcap:LicenseIdentifier
fibo-fnd-oac-ctl:Control
fibo-fnd-oac-oac:OwnershipAndControl
fibo-fnd-oac-own:Ownership
fibo-fnd-pas-pas:ProductIdentifier
fibo-fnd-plc-adr:RegionSpecificIdentifier
fibo-fnd-plc-loc:County
fibo-fnd-plc-loc:FederalCapitalArea
fibo-fnd-plc-loc:FederalState

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

fibo-fnd-plc-loc:Parcel
fibo-fnd-plc-uspsa:DeliveryAddressCodeSet
fibo-fnd-plc-uspsa:DeliveryPointCodeSet
fibo-fnd-plc-uspsa:ZipCodeScheme
fibo-fnd-plc-vrt:NotionalPlace
fibo-fnd-pty-pty:Situation
fibo-fnd-rel-rel:Reference
fibo-fnd-utl-alx:StatisticalAreaIdentifier
fibo-sec-sec-iss:SecurityOfferingDistributionType

Conclusion

FIBO is an example of very sophisticated ontology engineering performed by people with broad financial industry knowledge and by people with deep knowledge of ontologies and how to manage them. It is not something where one can simply ‘read the code’. To gain maximum utility from FIBO it is necessary to employ a sophisticated and easy to use tool such as GraphDB so that the wealth of knowledge it contains can best be employed in your specific use case. We have demonstrated that, as expected, OWL2 RL is a better choice than RDFS for reasoning over FIBO. Combined with reasoning, property paths are able to detect some structural issues. This demonstrates that such techniques provide useful tools for quality assuring large, complex, ontologies and knowledge graphs.

This is the first introductory post of a series that will demonstrate how graph database engines and semantic technology can be used to deal with ontologies and data in the financial services sector.

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

This performance was achieved on a MacBook Pro with 3GB of memory and a 2.6 GHz 6-Core Intel Core i7. It should also be noted that in its default configuration GraphDB Standard Edition (SE) uses all available cores. This is not a problem in a single user laptop but can create contention when these actions are performed on a large multiuser system.

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