BODS RDF Vocabulary - Detailed proposal, V2

Summary

Our initial proposal focused on traversing the BODS graph data as easily as possible, but it has sacrificed other objectives in the process. As the review from the Open Standard team has pointed, there were missed opportunities in terms of model extensibility and future-proofing the design.

We believe that this new proposal meets these challenges and provides an extensible BODS RDF model which the community can easily contribute to.

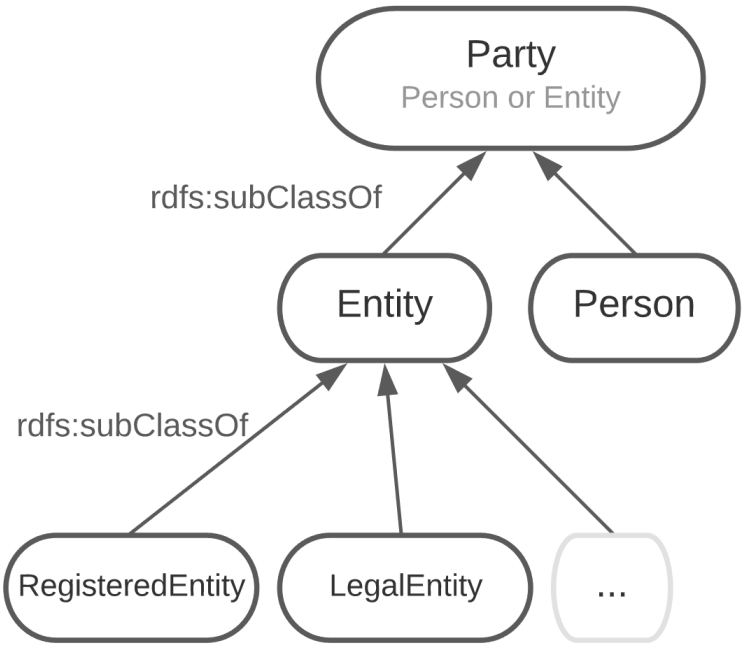
The focus in this new proposal was to materialise the core elements of the BODS schema (statements and interests) into standalone objects. This has inevitably increased complexity on querying and navigating corporate networks, but we believe this to be marginal and to be significantly outweighed by the gained benefits..

RDF Vocabulary

Entities and People

In accordance with the thinking here, we designed the classes `bods:Entity` and `bods:Person` as subclasses `bods:Party`. This is to support the idea that, in an ownership-or-control statement, the interested party can be a person or an entity, while the subject is always an entity.

It's important to note here that we materialise different entity types in explicit vocabulary classes. We believe this is more in line with the RDF design methods than employing an entityType property to reflect the specifics of the object.

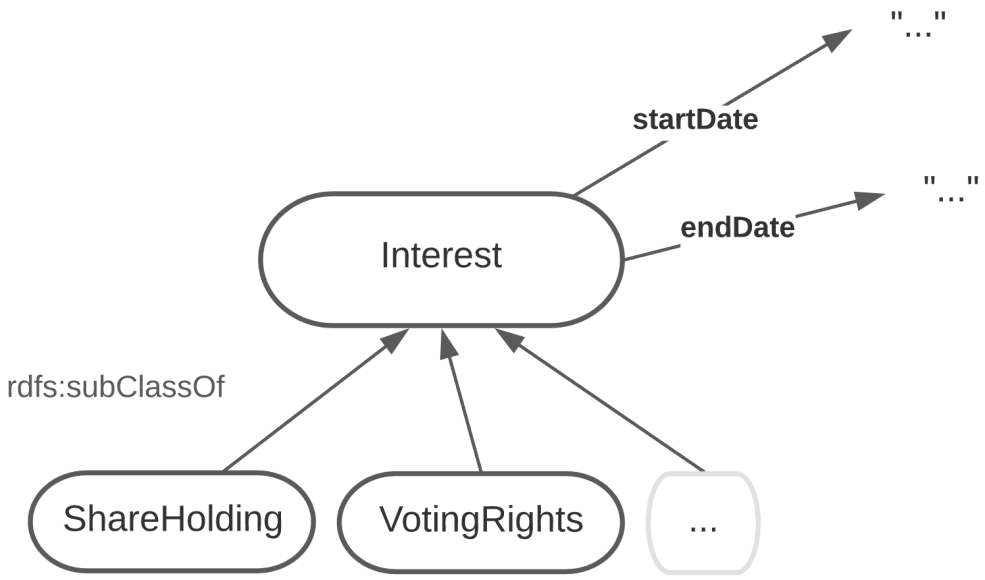


The identity of Parties are defined based on the the statements they were generated from, with the local name of their URIs representing the originating statement ID:

```
bods-res:openownership-register-7189843250441555991
    a bods:Person;
bods-res:openownership-register-11866084424644336529
    a bods:RegisteredEntity;
```

Interests

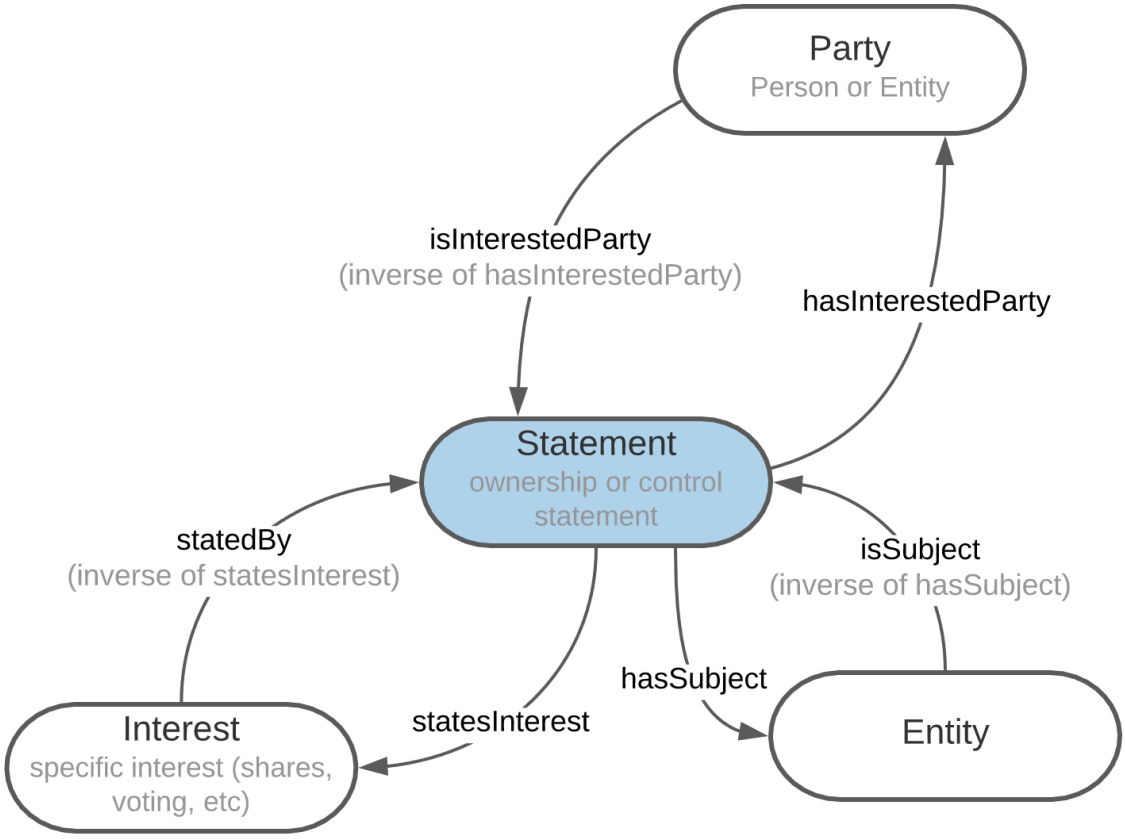
Interests are now modelled as objects in their own right, which allows us to easily expand the metadata we store against them (startDate, endDate, etc) with minimal effort. Similar to entities, we chose class inheritance to define specific types of interests.



Statements

The most significant shift from the previous design is the fact that statements are now defined as objects of their own, with properties and metadata.

In this context, a statement object will have a `bods:hasInterestedParty` (and a `bods:isInterestedParty` inverse) to reference the interested party. Similarly, an entity which is a subject for a statement is referenced with a `bods:isSubject`. Lastly, interest objects are linked explicitly to the statement using `bods:statesInterest` (not our favourite naming so far, so we welcome suggestions for this).



The identity of an RDF statement object is defined as the ID of the BODS JSON statement that it originates from. As argued earlier, we believe there will be a need to dereference BODS JSON data in certain circumstances, and this is a crucial element to support that. To this purpose, we supply the original statement ID as both the local part of the URI, as well as an explicit property for the object:

```
bods-res:openownership-register-10163665973112164634
  bods:statementId "openownership-register-10163665973112164634" ;
```

Query performance on network traversal

As discussed earlier, materialising the statement and interest objects has an impact on query complexity when navigating corporate structure and ownership. For example, a query to determine the ultimate parent entity (an important use case for consumers of this data), would be defined using property paths like this:

```
SELECT ?ultimateParent ?ultimateParentName
WHERE  {
    ?ultimateParent (bods:isInterestedParty/bods:hasSubject)+
bods-res:openownership-register-13061757995887162872 .
...
```

The complexity of this query is impractically high for most applications, and this increases when performing other similar queries aimed at identifying specifics within an entity's corporate network.

Custom inference rules can be designed to infer a `bods:hasInterestIn` for every (`bods:Party`, `bods:Entity`) pair involved in a `bods:Statement`. Ideally, we'd be able to define such reasoning at an ontology level, but we're not aware of any tools within the RDF standard specs that would achieve this. Different RDF storage platforms provide solutions for this, but all of them are vendor specific so we believe we should avoid this approach.

What seems a reasonable compromise is to deliver this inference as part of the dataset rather than the model, as this can easily be achieved with a simple INSERT query:

```
...

INSERT {
    ?interestedParty bods:hasInterestIn ?subject .
}
WHERE {
    ?statement bods:hasInterestedParty ?interestedParty .
    ?statement bods:hasSubject ?subject .
}
```

Other considerations

Unknown interested parties

The previous model proposal had the limitation of being unable to capture statements where these parties are unknown. With the interest object now materialised there are a series of options, but we believed that blank nodes reflect this notion most accurately:

```
bods-res:openownership-register-17585097645624593557 a bods:Statement;
  bods:hasInterestedParty _:node1epiv0l9ix2;
  bods:hasSubject bods-res:openownership-register-13494875149591033933;
  bods:statementId "openownership-register-17585097645624593557" .
```

Identity of interests

Interest objects are derived from ownership-or-control statements and currently have no identifiable context outside that in the BODS schema. It's then a challenge to define the URIs for these objects outside that scope. While there are several options that can be discussed (from hashing of context to random ID generation) we couldn't find very strong arguments for any of these.

It could be argued that these identifiers should be reproducible. That is, the interest URIs are consistent when running the same RDF conversion process over the same BODS dataset (which, as per initial proposal, remains the single source of truth).

While there are counter-arguments to that, we think that keeping this consistency was, at the moment, favourable to not having it. So the interim solution that seemed to make sense was to generate these URIs based on their index in the `interests` JSON array of the containing statement:

```
bods-res:openownership-register-10163665973112164634
  a                   bods:Statement ;
  bods:statesInterest bods-res:openownership-register-10163665973112164634_0 ;
  bods:statesInterest bods-res:openownership-register-10163665973112164634_1 ;
  bods:statesInterest bods-res:openownership-register-10163665973112164634_2 ;
  bods:statesInterest bods-res:openownership-register-10163665973112164634_3 .
```

De-referencing RDF data for statements, parties, interests

The ability to dereference objects like statements or interests could be key for certain use cases. However, this is a matter of operations and infrastructure which go significantly beyond the scope of modelling an RDF vocabulary. Any organisation undertaking this would become - at least to a degree - a de facto BODS data provider. While we'd love to be part of this work, unfortunately none of this is attainable at this stage for the Blue Anvil team and we believe it would be a struggle to see this delivered in practice without some form of investment.

Generating the RDF vocabulary automatically

It's important to note that the vocabulary is assembled from two distinctive but related components:

- A set of definitions which capture the top level objects, relationships and constraints.
- Details which are version-specific in different BODS JSON schema versions. These are specific classes like entities or interests, and are derived automatically from BODS schema code lists, which usually vary between releases.

This allows us to prescribe the general functionality of the model, but fully automate the specifics, and so we can easily generate a complete BODS RDF model from a reference JSON schema version. It should also make it easier for the consumer to reason about the reference model they require as there isn't a need to maintain a version mapping between the JSON and RDF schemas.

Below are complete definitions for each of the BODS JSON schema versions currently available:

- https://github.com/cosmin-marginean/kbods/blob/849f1171267daf9caf333a8fac54543796453b78/kbods-rdf/src/main/resources/vocabulary/bods-vocabulary-0.1.0.ttl which corresponds to http://standard.openownership.org/en/0.1.0/schema/
- https://github.com/cosmin-marginean/kbods/blob/849f1171267daf9caf333a8fac54543796453b78/kbods-rdf/src/main/resources/vocabulary/bods-vocabulary-0.2.0.ttl which corresponds to http://standard.openownership.org/en/0.2.0/schema/