Ordering

RDF, as a graph, is inherently unordered, and this can lead to difficulty when forms of description that presuppose ordering are translated into it. The Fedora community is trying several methods for constructing order in a graph. Here are some examples:

- PCDM Ordering

The choice of how to represent ordering is up to you. In the following discussion we'll consider an ordered list of authors for an academic research paper. Generally the authors of these papers care about the order they are cited, and so in this scenario we must retain their ordering.

First a caveat: one might be tempted to use a structure that relies on RDF "blank nodes", such as most tools generate by default from the Collection notation in RDF Turtle. As mentioned in Common metadata design patterns, blank nodes are not well-defined in the repository context and should generally not be used in Fedora. So here's an example of what to avoid:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
<>
  dc:title "Important Academic Research Paper" ;
  # Avoid the following
  dc:creator (<http://example.com/author/Quinn>  #no
              <http://example.com/author/Alice>  #no
              <http://example.com/author/Bob>) . #no
```

That example creates a bunch of blank nodes, which then get mangled going into Fedora. Let's not do that.

Here's a very simple alternative formation that gets the job done. We use hash-URIs instead of blank nodes.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
<>
  dc:title "Test title" ;
  dc:creator <#author_1>, <#author_2>, <#author_3> .
<#author_1> owl:sameAs <http://example.com/author/Quinn> .
<#author_2> owl:sameAs <http://example.com/author/Alice> .
<#author_3> owl:sameAs <http://example.com/author/Bob> .
```

In addition to being Fedora-friendly, this formulation is arguably better from a representational standpoint: anybody iterating the triples can query for the paper by author without having to jump through rdf:list hijinx. Of course, those consumers that care about the ordering would need to parse the number out of the hash part of each URI and sort by that. For plenty of people that's good enough.

What if you wanted to represent ordering as a separate number, rather than parsing the hash-URIs? Here's a slightly more complex version that accomplishes that; here, each hash URIs can be anything you want, and the ordering is stored as a separate triple:

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix schema: <https://schema.org/> .
<>
  dc:title "Test title" ;
  dc:creator <#xyz>, <#abc>, <#123> .
<#xyz> owl:sameAs <http://example.com/author/Quinn>; schema:Order 1 .
<#abc> owl:sameAs <http://example.com/author/Alice>; schema:Order 2 .
<#123> owl:sameAs <http://example.com/author/Bob>  ; schema:Order 3 .
```

If you prefer, you could go with a PCDM-style proxy ordering approach. This generates the most triples of any approach we've considered, but for some tool chains it makes good sense.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix iana: <http://www.iana.org/assignments/relation/> .
@prefix ore: <http://www.openarchives.org/ore/1.0/datamodel#> .
<>
  dc:title "Test title" ;
  dc:creator <#creators> .
<#creators> a ore:Aggregation;
            iana:first <#xyz>;
            iana:last <#123> .
<#xyz> ore:proxyFor <http://example.com/author/Quinn>;
       ore:proxyIn <#creators>;
       iana:next <#abc> .
<#abc> ore:proxyFor <http://example.com/author/Alice>;
       ore:proxyIn <#creators>;
       iana:prev <#xyz> ;
       iana:next <#123> .
<#123> ore:proxyFor <http://example.com/author/Bob>;
       ore:proxyIn <#creators>;
       iana:prev <#abc> .
```

As you can see there are many ways to do this, depending on how you want to model your metadata. Let's consider one final version in which we model authorship as an event which records not just the author involved but also the date of their involvement.

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix ex: <http://example.com/relations/> .
<>
  dc:title "Test title" ;
  dc:creator <#authA>, <#authB>, <#authC> .
<#authA> a ex:Authorship;
         ex:occurredOn "2016-05-15";
         ex:authorInvolved <http://example.com/author/Quinn>;
         ex:followingAuthor <#authB> .
<#authB> a ex:Authorship;
         ex:occurredOn "2016-05-15";
         ex:authorInvolved <http://example.com/author/Alice>;
         ex:followingAuthor <#authC> .
<#authC> a ex:Authorship;
         ex:occurredOn "2016-05-15";
         ex:authorInvolved <http://example.com/author/Bob> .
```