DataONE API - Task #8529

Add field to Solr index that includes the obsolescence chain

2018-03-29 22:36 - Peter Slaughter

Status:	New	**Start date:**	2018-03-29
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
Story Points:			

Description

When a new PID is added to the Solr index, populate a field that contains the entire obsolescence chain for that PID. This will greatly reduce the amount of time clients have to spend on obtaining this information.

For each PID that precedes the new PID in the obs. chain, update it's Solr document to include the new, complete obs. chain.

The new Solr field that contains the obs. chain should be a multivalued field, if the order of PIDs in the field can be guaranteed. In this scenario, the firsts PID is the lowest in the obs. chain, the lasts pid in the field is the mosts recent, un-obsoleted PID.

If the order cannot be guaranteed in a multi-valued field, then a simple text field should be used, with a reasonable value as a delimiter, one that is not valid for PIDs, such as a blank.

This task is related to https://redmine.dataone.org/issues/8528

Related issues:

Related to DataONE API - Task #8528: Add MNRead.getVersions, CNRead.getVersions	**New**	**2018-03-29**

History

#1 - 2018-03-30 04:11 - Rob Nahf

- Related to Task #8528: Add MNRead.getVersions, CNRead.getVersions added

#2 - 2018-03-30 04:33 - Rob Nahf

Performance on these updates should be evaluated, especially if we are implementing a chain identifier in CN/MN Read. In that case it would be easier to expose the chain identifier, and query for it as we currently do for seriesId. Appropriate queries would make light work of ordering the items retrieved client side.

Since each update changes 2 records using two index tasks, the number of updates would be 2n, unless we get away from the "something in the system metadata changed" approach of creating one-size-fits-all, "overwrite from the systemMetadata" tasks (which might be a good idea anyway - how many updates would happen if you updated the access policy on the whole chain? n^2, it would seem).

Also, more labor intensive would be updating a Resource Map, because it reindexes all of its members, which would then cascade down to updates on every version of all members of the package (twice). The potential for locking issues would seem to rise.

#3 - 2018-03-30 16:28 - Rob Nahf

a simpler implementation would be the creation of a dedicated index for chains. When a new PID needs to be added, just look up on the pid field using the obsoletes value, then append a new value if it's not already there. (d1.updates create 2 index tasks)

The record would simply be structured with an id field, and a pid field. The id would correspond to the chain-identifier, and could be opaque.

Querying would be something like q=pid:myPidOfInterest, and would return the entire record with all pids in the chain.

Ordering of elements for the CN would be more complicated than for the MN, because we couldn't rely on order of insert due to out of order

synchronizations. Reindexing would also be troublesome on both MN and CN.

#4 - 2018-03-30 19:45 - Rob Nahf

The order of values in multi-valued fields is guaranteed, but the order of fields in a record is not.

```
Yeah, Solr has a weaker guarantee.
Order is guaranteed to be maintained for values in a multi-valued field.
Order of different fields is not maintained.

-Yonik
http://lucidworks.com
```

we will need to be careful to either insert values in the proper order, or develop the capability to remove all and redo the list if we need to insert in the middle. Or we could also create multivalued obsoletes and obsoletedBy fields to allow the client (or a separate processor) to figure out the definitive order.

#5 - 2018-03-30 20:13 - Dave Vieglais

This seems like a reasonable solution - create a separate core for identifiers. Given two fields:

```
pid required multivalued text
sid optional single      text
```

It would be trivial to:

a) get the obsolescence chain given a pid or a sid: q=pid:id_to_find OR sid:id_to_find

b) determine if an identifier is a pid or a sid by examining the matching record

Reindexing could be driven from the content in postgres. Adding a new entry to the chain is a trivial update.

If choosing this route then it may be worth considering other types of common relationships that could be included in the core.

Rob Nahf wrote:

> a simpler implementation would be the creation of a dedicated index for chains. When a new PID needs to be added, just look up on the pid field using the obsoletes value, then append a new value if it's not already there. (d1.updates create 2 index tasks)
>
> The record would simply be structured with an id field, and a pid field. The id would correspond to the chain-identifier, and could be opaque.
>
> Querying would be something like q=pid:myPidOfInterest, and would return the entire record with all pids in the chain.
>
> Ordering of elements for the CN would be more complicated than for the MN, because we couldn't rely on order of insert due to out of order synchronizations. Reindexing would also be troublesome on both MN and CN.

#6 - 2018-04-10 05:21 - Rob Nahf

Don't forget to look into graph queries possible in later versions of Solr

https://lucene.apache.org/solr/guide/7_3/other-parsers.html

and perhaps, but not as likely to be useful:

https://lucene.apache.org/solr/guide/7_3/graph-traversal.html