Arvados - Bug #6076

[API] walk api server installations and ensure modified_at for collections is unique + ensure modified_at is enforced to be unique at the api level.

05/19/2015 05:09 PM - Ward Vandewege

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Arvados Future Sprints		

Description

Until this is done, datamanager needs to ask for groups of collections with overlap. That is not the most efficient, but it is also potentially problematic in light of #5834 which dynamically adjust the result set based on result size (and only guarantees a minimum of 1 result).

Related issues:

Related to Arvados - Story #3408: [Keep] Implement Production Data Manager	**Resolved**	**07/29/2014**

History

#1 - 05/19/2015 05:30 PM - Ward Vandewege

- Subject changed from [API] walk api server installations and ensure modified_at for collections is unique to [API] walk api server installations and ensure modified_at for collections is unique + ensure modified_at is unique at the api level.

- Description updated

#2 - 05/19/2015 05:31 PM - Ward Vandewege

- Subject changed from [API] walk api server installations and ensure modified_at for collections is unique + ensure modified_at is unique at the api level. to [API] walk api server installations and ensure modified_at for collections is unique + ensure modified_at is enforced to be unique at the api level.

#3 - 05/20/2015 06:23 PM - Brett Smith

Isn't this easy enough to handle at the client level with filters like modified_at >= current_working_modified_time, uuid not in [list of UUIDs I've already seen with that modified time]?

#4 - 06/09/2015 07:01 PM - Tom Clegg

The goal here is to provide a safe and easy way for a client to get the next page of results. Note-3 is safe but not easy. Ensuring modified_at uniqueness helps make the "modified_at >= X" method safe, but even that is a bit awkward.

In order for "sort by modified_at" to be completely safe, we must guarantee at the database level that modified_at is not only unique, but greater than all other rows' modified_at values at the time of insert/update. Currently, modified_at is assigned by Rails model validations and (AFAIK) there is nothing in place to prevent updates from getting reordered between "assign modified_at" and "insert/update database record". *I think it's a good idea to enforce this constraint.*

An even better solution would be for the server to provide a "next page token" which the client can send with its next request. This way the "next page" logic can sit on the server side instead of the client side, and the client doesn't need to know exactly which ordering guarantees are enforced. It can stick to "if the server doesn't give me a safe next-page link, I can't work."

#5 - 08/11/2015 02:15 PM - Brett Smith

- Target version changed from Bug Triage to Arvados Future Sprints