Metacat - Bug #5560

Upgrade access control rules in Metacat DB

12/02/2011 09:02 AM - ben leinfelder

Status:	Resolved	**Start date:**	12/02/2011
Priority:	Normal	**Due date:**	
Assignee:	ben leinfelder	**% Done:**	0%
Category:	metacat	**Estimated time:**	0.00 hour
Target version:	2.0.0	**Spent time:**	0.00 hour
Bugzilla-Id:	5560		

Description

Metacat handles a single access control policy for ALL revisions of an object whereas DataONE allows different access control rules for each revision (revisions are not a closely tied to one another in the DataONE model as they are in the Metacat docid.rev approach).
After a lengthy discussion, Matt and I decided the upgrade procedure for Metacat 1.x to 2.x should duplicate access control rules for a document into every revision so that Metacat would have distinct access control policies for every revision -- matching DataONE's model for access control.
This is the general upgrade approach:
1. create GUID mappings in the 'identifier' table for every scope.docid.rev
2. alter xml_access table to include the GUID column.
3. insert xml_access rows for each GUID in the identifier table, copying the xml_access row that exists for that docid (no revision).
4. clean-up xml_access by removing the rows that have no GUID value (the rows we copied from).
5. alter the xml_access table to remove unused columns: accessfileid, ticket and node-based columns.

This means that Metacat 2.0 should only insert and read xml_access records using the GUID. If we are interacting with Metacat using the legacy API, we will first need to look up the GUID from the identifier table. This will ultimately simplify the acces DAO classes.

One question I have is whether the Metacat API should continue to (a) function as it has been where a call to "setaccess" will update access control rules for *every* revision of a document. Another possible policy would be (b) to only update the the given revision if the revision was provided in the docid parameter. If it was simply scope.docid we could update every revision since we wouldn't know which one to specifically update. Or (c) we could update only the latest revision if no docid were provided. Option (a) would effectively look as though Metacat access control handling had not changed from v1.x to v2.x.

History

#1 - 12/02/2011 11:25 AM - ben leinfelder

By removing the 'docid' column from the 'xml_access' table, we introduce a huge amount of refactoring -- the custom EML parser uses the column, the Query spec uses the column, the spatial query cache uses the column -- really anything that needs to check if if can show that the doic is in the system uses docid.
So there are two ways to deal with this refactoring:
1. Continue to code with 'docid' passed around, but join to the identifier table using guid when we need xml_access rows.
2. Change the code to pass around 'guid' for all queries.

#2 - 12/09/2011 02:56 PM - ben leinfelder

Metacat now tracks permissions for each revision of a document/data object.

The upgrade goes like this:
1. Generate GUIDs for every docid+rev in xml_documents (current version)
2. Generate GUIDs for every docid+rev in xml_revisions (old versions, archived versions)
3. Insert xml_access entries for each existing docid (existing rule is duplicated for every revision)
4. Insert special xml_access entries for inline data files we have generated from EML files. These use a special GUID that is not actually tracked outside of the xml_access table.
5. Update accessfileid to use the GUID of the EML file that defines access rules for Metacat-housed data objects.
6. Remove any remaining xml_access rows that do not have a GUID (these are the old existing entries)

#3 - 12/12/2011 01:51 PM - ben leinfelder

Access control JUnit tests are all passing. I would like to test this from a 1.9.5 Metacat installation being upgraded to 2.0.0. A dump from KNB production would be best so that we have an idea of how long it will take to migrate the DB records when running the upgrade SQL.

#4 - 12/16/2011 02:48 PM - ben leinfelder

I'm now also forcing the shared System Metadata map to reload into memory all the system metadata for data objects that an EML doc defines access for -- this ensures we always have the latest changes that are in the DB tables for access control.

#5 - 03/27/2013 02:30 PM - Redmine Admin

Original Bugzilla ID was 5560