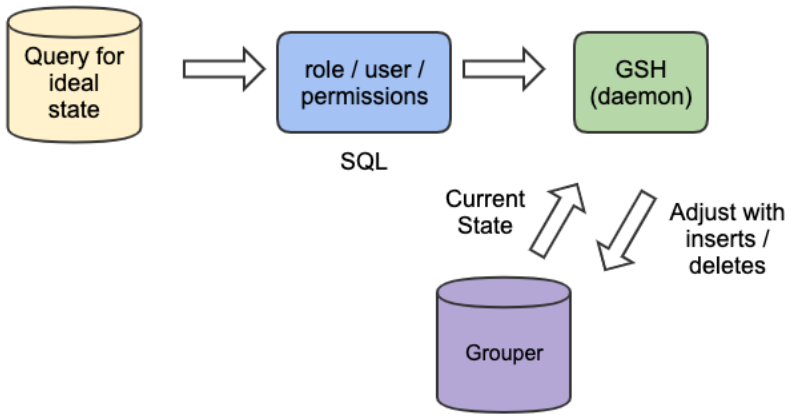
Grouper Loader permissions or attributes POC with GSH

This is not a "loader", this is a script that loads things from a SQL. This is currently a full sync



Standards

1. Make a GSH script that can be compiled and run from java (just add class and main())
2. Code with small blocks that do discrete things (lower cyclic complexity)

Note: we can make some common things library methods to make this easier in future

Make some data

Note, change around the subject source, folders, attributes, etc. Note also this can be destructive, so pay attention and do not delete prod data! 🙂

Table for permissions that simulates a view to pull from (mysql example)

```
CREATE TABLE permission_load (group_name VARCHAR(100), subject_id VARCHAR(100),
       permission_name VARCHAR(100));


ALTER TABLE permission_load
   ADD PRIMARY KEY(group_name, subject_id, permission_name);
```

GSH script creates a bunch of roles and permissions and randomly assigns some data in grouper and database

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import edu.internet2.middleware.grouper.attr.AttributeDef;
import edu.internet2.middleware.grouper.attr.AttributeDefName;
import edu.internet2.middleware.grouper.attr.AttributeDefNameSave;
import edu.internet2.middleware.grouper.attr.AttributeDefSave;
import edu.internet2.middleware.grouper.attr.AttributeDefType;
import edu.internet2.middleware.grouper.attr.AttributeDefValueType;
import edu.internet2.middleware.grouper.group.TypeOfGroup;
import edu.internet2.middleware.grouper.permissions.PermissionAllowed;
import edu.internet2.middleware.grouperClient.collections.MultiKey;
```

```
import edu.internet2.middleware.grouperClient.jdbc.GcDbAccess;
import edu.internet2.middleware.subject.Subject;

boolean deleteOldData = true;
String subjectSource = "jdbc";

GrouperSession grouperSession = GrouperSession.startRootSession();
int roleCount = 100;
Map groupMap = new HashMap();
for (int i = 0; i < roleCount; i++) {
  groupMap.put("role" + i,  new GroupSave(grouperSession).assignName("test2:role1").assignTypeOfGroup
(TypeOfGroup.role).assignCreateParentStemsIfNotExist(true).save());
}
for (int i = 0; i < roleCount; i++) {
  groupMap.put("role" + i, new GroupSave(grouperSession).assignName("test2:role" + i).assignTypeOfGroup
(TypeOfGroup.role).assignCreateParentStemsIfNotExist(true).save());
}
Stem test3 = StemFinder.findByName(grouperSession, "test3", false);
if (deleteOldData && test3 != null) {
  Stem.obliterate("test3", false, false, true);
}
if (deleteOldData) {
  new GcDbAccess().sql("delete from permission_load").executeSql();
}
AttributeDef permissionDef = new AttributeDefSave(grouperSession).assignName("test3:permissionDef").
assignCreateParentStemsIfNotExist(true).assignToGroup(true).assignToEffMembership(true).assignAttributeDefType
(AttributeDefType.perm).assignMultiAssignable(false).assignValueType(AttributeDefValueType.marker).save();
permissionDef.getAttributeDefActionDelegate().configureActionList("assign");
int permissionCount = 100;
Map permissionMap = new HashMap();
for (int i = 0; i < permissionCount; i++) {
  permissionMap.put("permission" + i, new AttributeDefNameSave(grouperSession, permissionDef).assignName("test3:
permission" + i).assignCreateParentStemsIfNotExist(true).save());
}
Map subjectMap = new HashMap();
int subjectCount = 10;
for (int i = 0; i < subjectCount; i++) {
  subjectMap.put("test.subject." + i, SubjectFinder.findByIdAndSource("test.subject." + i, subjectSource,
true));
}
int assignmentCount = 100;
List assignments = new ArrayList();
for (int i = 0; i < assignmentCount; i++) {
  Object[] assignment = new Object[3];
  assignments.add(assignment);
}
for (int i = 0; i < assignmentCount; i++) {
  ((Object[]) assignments.get(i))[0] = subjectMap.get("test.subject." + (int) Math.floor(Math.random() *
subjectCount));
}
for (int i = 0; i < assignmentCount; i++) {
  ((Object[]) assignments.get(i))[1] = groupMap.get("role" + (int) Math.floor(Math.random() * roleCount));
}
for (int i = 0; i < assignmentCount; i++) {
  ((Object[]) assignments.get(i))[2] = permissionMap.get("permission" + (int) Math.floor(Math.random() *
permissionCount));
}
for (int i = 0; i < assignmentCount; i++) {
  assignments.set(i, new MultiKey((Object[]) (assignments.get(i))));
}
Set assignmentsSet = new HashSet(assignments);
assignments.clear();
assignments.addAll(assignmentsSet);
for (int i = 0; i < assignmentCount; i++) {
  Subject subject = (Subject) ((MultiKey) assignments.get(i)).getKey(0);
  Group group = (Group) ((MultiKey) assignments.get(i)).getKey(1);
  group.addMember(subject, false);
}
for (int i = 0; i < assignmentCount; i++) {
  if (Math.random() > 0.5) {
    Subject subject = (Subject) ((MultiKey) assignments.get(i)).getKey(0);
```

```
    Group group = (Group) ((MultiKey) assignments.get(i)).getKey(1);
    AttributeDefName permission = (AttributeDefName) ((MultiKey) assignments.get(i)).getKey(2);
    new GcDbAccess().sql("insert into permission_load (group_name, subject_id, permission_name) values (?, ?,
?)").addBindVar(group.getName()).addBindVar(subject.getId()).addBindVar(permission.getName()).executeSql();
  }
}
for (int i = 0; i < assignmentCount; i++) {
  if (Math.random() > 0.5) {
    Subject subject = (Subject) ((MultiKey) assignments.get(i)).getKey(0);
    Group group = (Group) ((MultiKey) assignments.get(i)).getKey(1);
    AttributeDefName permission = (AttributeDefName) ((MultiKey) assignments.get(i)).getKey(2);
    group.getPermissionRoleDelegate().assignSubjectRolePermission(permission, subject,PermissionAllowed.
ALLOWED);
  }
}
```

Sync data from database to grouper

GSH script can be scheduled via script daemon

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;

import edu.internet2.middleware.grouper.app.loader.OtherJobScript;
import edu.internet2.middleware.grouper.attr.AttributeDefName;
import edu.internet2.middleware.grouper.attr.finder.AttributeDefNameFinder;
import edu.internet2.middleware.grouper.permissions.PermissionAllowed;
import edu.internet2.middleware.grouperClient.collections.MultiKey;
import edu.internet2.middleware.grouperClient.jdbc.GcDbAccess;
import edu.internet2.middleware.subject.Subject;


// run as root
GrouperSession grouperSession = GrouperSession.startRootSession();
String sourceId = "jdbc";

// get results from target
List resultsFromTarget = new GcDbAccess().sql("select distinct group_name, subject_id, permission_name from
permission_load").selectList(Object[].class);
for (int i=0;i<resultsFromTarget.size();i++) { resultsFromTarget.set(i, new MultiKey((Object[])
resultsFromTarget.get(i))); }

if (OtherJobScript.retrieveFromThreadLocal() != null) {
  OtherJobScript.retrieveFromThreadLocal().getOtherJobInput().getHib3GrouperLoaderLog().addTotalCount
(resultsFromTarget.size());
}

// get results from grouper
List resultsFromGrouper = new GcDbAccess().sql("select gaaev.group_name, gaaev.subject_id, gaaev.
attribute_def_name_name from grouper_attr_asn_efmship_v gaaev where gaaev.subject_source_id = 'jdbc' and gaaev.
attribute_def_name_name like 'test3:%' and gaaev.group_name like 'test2:%' and gaaev.name_of_attribute_def =
'test3:permissionDef'").selectList(Object[].class);
for (int i=0;i<resultsFromGrouper.size();i++) { resultsFromGrouper.set(i, new MultiKey((Object[])
resultsFromGrouper.get(i))); }

// list of inserts
List inserts = new ArrayList(resultsFromTarget);
inserts.removeAll(resultsFromGrouper);

// list of deletes
List deletes = new ArrayList(resultsFromGrouper);
deletes.removeAll(resultsFromTarget);
```

```
// all objects to deal with
List allOperations = new ArrayList(inserts);
allOperations.addAll(deletes);

// consolidate groups
Set groupNameSet = new HashSet();
for (int i=0;i<allOperations.size();i++) {
  MultiKey operation = (MultiKey)allOperations.get(i);
  String groupName = (String)operation.getKey(0);
  groupNameSet.add(groupName);
}

// consolidate subjects
Set subjectIdSet = new HashSet();
for (int i=0;i<allOperations.size();i++) {
  MultiKey operation = (MultiKey)allOperations.get(i);
  String subjectId = (String)operation.getKey(1);
  subjectIdSet.add(subjectId);
}

// consolidate permissions
Set permissionNameSet = new HashSet();
for (int i=0;i<allOperations.size();i++) {
  MultiKey operation = (MultiKey)allOperations.get(i);
  String permissionName = (String)operation.getKey(2);
  permissionNameSet.add(permissionName);
}

// lookup groups
Map groupMap = new HashMap();
List groupNameList = new ArrayList(groupNameSet);
for (int i=0;i<groupNameList.size();i++) {
  String groupName = (String)groupNameList.get(i);
  Group group = GroupFinder.findByName(grouperSession, groupName, false);
  if (group != null) {
    groupMap.put(groupName, group);
  } else {
    if (OtherJobScript.retrieveFromThreadLocal() != null) {
      OtherJobScript.retrieveFromThreadLocal().getOtherJobInput().getHib3GrouperLoaderLog().appendJobMessage("
Group not found: " + groupName + "! ");
      }
  }
}

// lookup subjects
Map subjectMap = new HashMap();
List subjectIdList = new ArrayList(subjectIdSet);
for (int i=0;i<subjectIdList.size();i++) {
  String subjectId = (String)subjectIdList.get(i);
  Subject subject = SubjectFinder.findByIdAndSource(subjectId, "jdbc", false);
  if (subject != null) {
    subjectMap.put(subjectId, subject);
  } else {
    if (OtherJobScript.retrieveFromThreadLocal() != null) {
      OtherJobScript.retrieveFromThreadLocal().getOtherJobInput().getHib3GrouperLoaderLog().appendJobMessage("
Subject not found: " + subjectId + "! ");
      OtherJobScript.retrieveFromThreadLocal().getOtherJobInput().getHib3GrouperLoaderLog().
addUnresolvableSubjectCount(1);
    }
  }
}

// lookup permissions
Map permissionMap = new HashMap();
List permissiionNameList = new ArrayList(permissionNameSet);
for (int i=0;i<permissiionNameList.size();i++) {
  String permissionName = (String)permissiionNameList.get(i);
  AttributeDefName permission = AttributeDefNameFinder.findByName(permissionName, false);
  if (permission != null) {
    permissionMap.put(permissionName, permission);
```

```
  } else {
    if (OtherJobScript.retrieveFromThreadLocal() != null) {
      OtherJobScript.retrieveFromThreadLocal().getOtherJobInput().getHib3GrouperLoaderLog().appendJobMessage("
Permission not found: " + permissionName + "! ");
    }
  }
}

// convert insert objects
List insertsMultiKey = new ArrayList();
for (int i=0;i<inserts.size();i++) {
  MultiKey insert = (MultiKey)inserts.get(i);
  Group group = (Group)groupMap.get((String)insert.getKey(0));
  if (group == null) {
    continue;
  }
  Subject subject = (Subject)subjectMap.get((String)insert.getKey(1));
  if (subject == null) {
    continue;
  }
  if (!group.hasMember(subject)) {
    if (OtherJobScript.retrieveFromThreadLocal() != null) {
      OtherJobScript.retrieveFromThreadLocal().getOtherJobInput().getHib3GrouperLoaderLog().appendJobMessage("
" + group.getName() + " does not have member " + subject.getId() + "!!! ");
    }
    continue;
  }
  AttributeDefName permission = (AttributeDefName)permissionMap.get((String)insert.getKey(2));
  if (permission == null) {
    continue;
  }
  insertsMultiKey.add(new MultiKey(group, subject, permission));
}

// convert delete objects
List deletesMultiKey = new ArrayList();
for (int i=0;i<deletes.size();i++) {
  MultiKey delete = (MultiKey)deletes.get(i);
  Group group = (Group)groupMap.get((String)delete.getKey(0));
  if (group == null) {
    continue;
  }
  Subject subject = (Subject)subjectMap.get((String)delete.getKey(1));
  if (subject == null) {
    continue;
  }
  AttributeDefName permission = (AttributeDefName)permissionMap.get((String)delete.getKey(2));
  if (permission == null) {
    continue;
  }
  deletesMultiKey.add(new MultiKey(group, subject, permission));
}

// do the inserts
for (int i=0;i<insertsMultiKey.size();i++) {
  Group group = (Group)((MultiKey)insertsMultiKey.get(i)).getKey(0);
  Subject subject = (Subject)((MultiKey)insertsMultiKey.get(i)).getKey(1);
  AttributeDefName permission = (AttributeDefName)((MultiKey)insertsMultiKey.get(i)).getKey(2);
  group.getPermissionRoleDelegate().assignSubjectRolePermission(permission, subject, PermissionAllowed.ALLOWED);
  if (OtherJobScript.retrieveFromThreadLocal() != null) {
    OtherJobScript.retrieveFromThreadLocal().getOtherJobInput().getHib3GrouperLoaderLog().addInsertCount(1);
  }
}

// do the deletes
for (int i=0;i<deletesMultiKey.size();i++) {
  Group group = (Group)((MultiKey)deletesMultiKey.get(i)).getKey(0);
  Subject subject = (Subject)((MultiKey)deletesMultiKey.get(i)).getKey(1);
  AttributeDefName permission = (AttributeDefName)((MultiKey)deletesMultiKey.get(i)).getKey(2);
  group.getPermissionRoleDelegate().removeSubjectRolePermission(permission, subject);
  if (OtherJobScript.retrieveFromThreadLocal() != null) {
```

```
    OtherJobScript.retrieveFromThreadLocal().getOtherJobInput().getHib3GrouperLoaderLog().addDeleteCount(1);
  }
}
```