Member search and sort columns

In order to allow searches for members in a group and sorting of members in a group without having to resolve subjects, we have added additional columns to the grouper_members table.

- name - This would contain subject.getName()
- description - This would contain subject.getDescription().
- sort_string0
- sort_string1
- sort_string2
- sort_string3
- sort_string4
- search_string0
- search_string1
- search_string2
- search_string3
- search_string4

Search strings will allow up to 2K of data and sort strings will allow up to 50 bytes of data. Each sort and search string would be attributes configured in the sources.xml file. They would be configured for each source. Keeping the attributes consistent for each index (for people sources at least) would make the searching/sorting more useful. If there's more than the maximum characters allowed, it will simply be truncated rather than causing an error. Each source will require at least one search string and one sort string, otherwise there will be an error during startup.

Note that since the sources.xml file supports virtual attributes, you also have the option of having multiple attributes within one search index (comma separated) and then just give the user a single search option. The built in searches will only search on one field in each query. Though you can populate multiple search columns if you want certain users to have access to search on additional (e.g. private) attributes and then use the security described below to limit who can search on which fields. To specify the search and sort attributes for the Group Source Adapter (g:gsa) or person sources that are configured in sources.xml, add an init-param for each attribute where the param-name is searchAttribute[0-4] or sortAttribute[0-4] and the param-value is the name of the attribute. The attribute should be obtainable via subject.getAttributeValue(attributeName, false) - Note that for the jdbc sources, the name isn't necessarily the database column name. It needs to be a subject attribute.

```
<init-param>
       <param-name>subjectVirtualAttribute_0_searchAttribute0</param-name>
       <param-value>${subject.name},${subjectUtils.defaultIfBlank(subject.getAttributeValue('LFNAME'),
"")},${subjectUtils.defaultIfBlank(subject.getAttributeValue('LOGINID'), "")},${subjectUtils.defaultIfBlank
(subject.description, "")},${subjectUtils.defaultIfBlank(subject.getAttributeValue('EMAIL'), "")}</param-value>
     </init-param>
     <init-param>
       <param-name>sortAttribute0</param-name>
       <param-value>LFNAME</param-value>
     </init-param>
     <init-param>
       <param-name>sortAttribute1</param-name>
       <param-value>LOGINID</param-value>
     </init-param>
     <init-param>
       <param-name>searchAttribute0</param-name>
       <param-value>searchAttribute0</param-value>
     </init-param>
```

We would allow subjects to have "internal" attributes so that these comma-separated virtual attributes are not included in the Subject API like Subject.getAttributeValue() by default unless a new overloaded method is used. You would be able to specify which attributes are internal attributes in the sources.xml file.

```
<internal-attribute>internalAttribute0</internal-attribute>
<internal-attribute>internalAttribute1</internal-attribute>
<internal-attribute>internalAttribute2</internal-attribute>
```

The sort and search column configuration for the internal and external subject sources are in grouper.properties:

```
# Search and sort strings for internal users
internalSubjects.searchAttribute0.el = ${subject.name},${subject.id}
internalSubjects.sortAttribute0.el = ${subject.name}

...

#search and sort strings added to member objects
externalSubjects.searchAttribute0.el = ${subject.name},${subjectUtils.defaultIfBlank(subject.getAttributeValue
("institution"), "")},${subjectUtils.defaultIfBlank(subject.getAttributeValue("identifier"), "")},${subject.
id},${subjectUtils.defaultIfBlank(subject.getAttributeValue("email"), "")}
externalSubjects.sortAttribute0.el = ${subject.name}
externalSubjects.sortAttribute1.el = ${subjectUtils.defaultIfBlank(subject.getAttributeValue("identifier"), "")}
externalSubjects.sortAttribute2.el = ${subjectUtils.defaultIfBlank(subject.getAttributeValue("institution"),
"")}
```

The data in these new columns would get updated when a subject is resolved by id or identifier or when a new member row is created. Also, group names would get updated when groups are renamed.

The search columns would contain lowercase characters and searches would be substring searches of each word in the string. So a search for "John Doe" on search_string0 would be ... where search_string0 like '%john%' and search_string0 like '%doe%'.

You can restrict users that are allowed to search/sort on each column using groups. The configuration is in grouper.properties. By default, everybody has access.

```
# By default, all users have access to sort using any of the sort strings in the member table and search using
any of the search strings in the member table.
# You can restrict to wheel only or to a certain group.
#security.member.sort.string0.allowOnlyGroup = etc:someGroup
#security.member.sort.string1.allowOnlyGroup = etc:someGroup
#security.member.sort.string2.wheelOnly = true
#security.member.sort.string3.wheelOnly = true
#security.member.sort.string4.wheelOnly = true
#security.member.search.string0.allowOnlyGroup = etc:someGroup
#security.member.search.string1.allowOnlyGroup = etc:someGroup
#security.member.search.string2.wheelOnly = true
#security.member.search.string3.wheelOnly = true
#security.member.search.string4.wheelOnly = true
```

We also have config options to specify the default indexes to use for searching and sorting if one is not specified.

```
###################################
## Member sort and search
###################################

# Attributes of members are kept in the grouper_members table to allow easy sorting and searching (for instance
when listing group members).
# When performing a sort or search and an index is not specified, then a default index will be used as
configured below.  The value is comma-separated,
# so that if the user does not have access to the first index, then next will be tried and so forth.
# Note:  all sources should have attributes configured for all default indexes.
member.search.defaultIndexOrder=0
member.sort.defaultIndexOrder=0
```

Currently, this functionality is available in the admin and lite UIs when displaying the membership list of a group. It is also partially available in the API using Group.getImmediateMembers(Field, Set<Source>, QueryOptions, SortStringEnum, SearchStringEnum, String). For the UIs, using the media. properties configuration file, you can specify if you want to enable or disable member sorting and searching. For sorting, you also have the option to specify if you want to allow users to use the default sort index only or if you want them to be able to choose how they want to sort.

```
#### Member sorting and searching
# Whether to enable member sorting using sort attributes stored in Grouper.
member.sort.enabled=true

# Whether to use default sorting only and not allow users to specify which sort attribute to use.
member.sort.defaultOnly=false

# Whether to enable member searching using search attributes stored in Grouper.
member.search.enabled=true
```

If you're not using default only sorting, you can specify the labels that users would see for each sort index.

```
# If you have enabled member sorting (member.sort.enabled) and disabled default sorting (member.sort.
defaultOnly), be sure to add labels for each default sort string configured in grouper.properties (member.sort.
defaultIndexOrder).
member.sort.string0=Name
member.sort.string1=Login Id
```

So here's one way the data may be stored.

	sort0	sort1	sort2	search0	search1 (Name)
person source	displayName	sn	uid	displayName,uid,ou	uid,ou
group source	displayName	null	null	name,displayName	name,displayName

sort0 = Sort by name
sort1 = Sort by last name
sort2 = Sort by login id
search0 = default search for privileged users
search1 = default search for all other users

Sync Member Attributes

If you make a change to the sort or search strings, you should sync the member attributes. For subjects that are people, you can use USDU. Run the OTHER_JOB_usduDaemon job using the "Daemon jobs" UI page.

For subjects that are groups, you can run the following line using GSH:

```
gsh 0% GrouperSession.startRootSession()
gsh 1% for (String g : HibernateSession.byHqlStatic().createQuery("select uuid from Group").listSet(String.
class)) { subj = SubjectFinder.findByIdAndSource(g, "g:gsa", true); GrouperDAOFactory.getFactory().getMember().
findBySubject(subj).updateMemberAttributes(subj, true); }
```