Integer IDs on Grouper objects

In Grouper 2.2+ the main Grouper objects in the database (groups, folders, attribute definitions, attribute names) will be assigned unique integers. These integers can be used, for instance, as UNIX GIDs.

The ID's are not used anywhere else in Grouper, so if you need to adjust them feel free, though the max index for an object type should never be more than the last index used in the grouper_table_index table (you can increment that).

Some reserved ID's can be stored in memory so that the table does not need to be accessed for each object creation. By default non-GSH environments will reserve 10 at a time, and GSH will reserve 1 at a time. If Grouper is bounced while ID's are reserved (common case), then those ID's will never be used. Also, if transactions are rolled back, then id's will be wasted. Hence the id's are generally sequential and each id is used, but there will be gaps and they will only be sequential per JVM (even if the reserved size is 1).

The implementation is thread safe per JVM and across JVM's.

You can configure a group whose members are allowed to assign index ID's on create in the grouper.properties. If this config property is blank then anyone can assign index ID's on create

```
# group who can assign id index cols (also, wheel or root is allowed)
grouper.tableIndex.groupWhoCanAssignIdIndex = etc:canAssignIdIndex
```

DDL

The Grouper 2.2 ddl will add large integer columns to the 4 tables: grouper_groups, grouper_stems, grouper_attribute_def, grouper_attribute_def_name. The DDL will also assign ID's to the existing rows. Note, this column is non-null for new deployments of grouper, feel free to add that constraint to upgrades. There is also a grouper_table_index table with 4 rows which keeps track of which if the last index reserved. There is a unique index on the id_index col so no rows can have the same index.

Configuration

Configure this in the grouper.properties

```
idIndex.group.minIndex = 10000
idIndex.stem.minIndex = 10000
idIndex.attributeDef.minIndex = 10000
idIndex.attributeDefName.minIndex = 10000

# verify that table indexes are set and the pointers are ok, incurs a bit of overhead to grouper startup
grouper.tableIndex.verifyOnStartup = true

# in different circumstances, retrieve a different number of IDs at once.
# if it is a system where the JVM is starting and stopping (e.g. GSH), then
# dont reserve that many at once
grouper.tableIndex.reserveIdsGsh = 1
grouper.tableIndex.reserveIdsDefault = 10
grouper.tableIndex.reserveIdsLoader = 10
grouper.tableIndex.reserveIdsWs = 10
grouper.tableIndex.reserveIdsUi = 10
```

Startup

On Grouper startup, two things are checked. If there are rows with null id_indexes, they will be set to id's. If the last reserved index is less than the max index for that object type, it will be updated. If you do not want to perform these checks on startup, set this in the grouper.properties:

```
# verify that table indexes are set and the pointers are ok, incurs a bit of overhead to grouper startup
grouper.tableIndex.verifyOnStartup = true
```

Import / export

ID indexes will export and import (if its a new record and the index isnt in use already), the last reserved index will be incremented appropriately. Note, if you import and there are other running JVMs, you might want to bounce them since they have indexes reserved in memory

API

There are API methods to lookup objects by ID index. e.g. GroupFinder.findByIdIndexSecure(idIndex, exceptionIfNotFound, queryOptions). Note, these methods cache by default, but you can stop the caching with queryOptions. Also there are: StemFinder.findByIdIndexSecure(), AttributeDefFinder.findByIdIndexSecure(), AttributeDefNameFinder.findByIdIndexSecure()

Web services

ID indexes are added to WS for clients 2.2+ (note, for SOAP you need the new 2.2 endpoint). These examples are XML, though you could do the same thing with JSON, XHTML, or SOAP.

You can see the ID index on group objects

```
      <wsGroup>
        <extension>newGroup5</extension>
        <name>aStem:newGroup5</name>
        <idIndex>12345</idIndex>
        ...
      </wsGroup>
```

You can lookup groups to find, or operate on (e.g. add a member to a group)

```
<wsGroupLookup>
  <idIndex>12345</idIndex>
</wsGroupLookup>
```

When creating groups, you can specify the id index (if the user is allowed to do so, and the index is not in use).

```
<wsGroup>
        <name>aStem:newGroup5</name>
        <idIndex>12345</idIndex>
        ...
      </wsGroup>
```

Grouper client

groupSave: lookup a group by idIndex (~~-groupLookupIdIndex), see the idIndex created, and you can optionally specify the id index when creating a group (if allowed) (~~-idIndex)

```
java -jar grouperClient.jar --operation=groupSaveWs --name=a:b:c [--groupLookupIdIndex=12345] [--idIndex=23456]
...
```

addMember: add a member to a group by id index

```
java -jar grouperClient.jar --operation=addMemberWs --groupIdIndex=12345 --subjectIds="test.subject.0,test.
subject.1"
```

getMembers: get members from groups by id index

```
java -jar grouperClient.jar --operation=getMembersWs --groupIdIndexes=123435,23456
```

hasMember: see if has member by id index of group

```
java -jar grouperClient.jar --operation=hasMemberWs --groupIdIndex=123456 --subjectIds="id.test.subject.0,id.
test.subject.1"
```

stemSave: lookup a stem by idIndex, see the idIndex in stems, and optionally you can specify the id index when creating a stem (if allowed)

```
java -jar grouperClient.jar --operation=stemSaveWs --name=aStem:newStem5 --idIndex=12345
```

findGroups: lookup groups by id index

```
java -jar grouperClient.jar --operation=findGroupsWs --groupIdIndexes=12345,23456
```

findStems: lookup stems by id index

```
java -jar grouperClient.jar --operation=findStemsWs --stemIdIndexes=12345,23456
```

getMemberships: constrain the group or stem by id index

getSubjects: constraint the subjects by group id index

getAttributeAssignments: lookup the groups or stems or attribute definitions or attribute names by id index

assignAttributes: lookup the groups or stems or attribute definitions or attribute names by id index

assignAttributeBatch: lookup the groups or stems or attribute definitions or attribute names by id index

getPermissionAssignments: lookup the roles or attribute definitions or attribute names by id index

assignPermissions: lookup the roles or permission names by id index

attributeDefNameSave: lookup the attribute def name by id index, or assign it on inserts (if allowed)

findAttributeDefNames: lookup attribute def names by id index of the attribute definition or name