Database Conversion v1.0 - v1.1

Unable to render {include} The included page could not be found.

Grouper Database Conversion

Grouper v1.1 works with a Grouper a v1.0 database just as well as Grouper v1.0 does. However, we found that by reordering the columns in one key table (the grouper_memberships table) significant performance gains occurred across several types of operations. We also reorded the columns in a second table (grouper_members) to explore for additional gains, but found the change only marginal. But the database schema proper - the tables, their columns, indices, and relationships - did not change at all between the v1.0 and v1.1 releases, which is why Grouper v1.1 works with a v1.0 database.

But to reap that performance gain, the columns in a Grouper database must be reordered. To do that we'll use the tools first released in Grouper v1.0 to support a database conversion, should one be needed. Basically, we'll export the entire database (including its metadata) into an XML file, reset the database, then reimport it, following the steps described below.

A database conversion can take a substantial amount of time, depending of course on how large it is. One on demo HSQLDB database containing 628 stems, 640 groups, and 14,447 memberships and non-default privilege assignments, on a laptop it took 1 minute 18 seconds to export, and 3 minutes 57 seconds to re-import, taking on average about 6ms to create each direct or indirect membership and privilege. Your mileage will vary, but it will take time to convert a substantial database.

Conversion steps

This process assumes that you have the Grouper API v1.1 RC3 or later installed and configured to work with your Grouper database, and that you have a shell open on the root of the Grouper API distribution directory.

Note that release candidates 1 and 2 had a bug with the XML export capability that prevented this process from being successful.

1. **ant xml-export -Dcmd="GrouperSystem aFileName"**

The default xml-export properties are correct for doing a complete dump of the database. The filename can refer to a file in the current directory, or it can be a relative or absolute pathname.

2. **Create a new database container**

Do whatever you have to do to setup a fresh database with your RDBMS terchnology.

3. **Setup the SA account used by the Grouper API**

Establish the credential used by the Grouper API for database access in your new database.

4. **Review conf/grouper.hibernate.properties**

Ensure that properties declaring how the Grouper API will connect to your database are correctly declared in the conf/grouper.hibernate.properties file.

4. **ant schemaexport**

5. **ant db-init**

These two steps create the Grouper v1.1 schema in your new database.

6. **ant xml-import -Dcmd="GrouperSystem theSameFileName"**

This re-loads everything into the new database.

An alternative approach

Since this change is only to the column order in two tables, you can consider using SQL to export and suitably re-import the tables. Details will vary by RDBMS type, but this is likely to be far faster than the conversion described above. The two changes are:

- grouper_members: removed 'member_uuid' from position 5 and inserted at position 2
- grouper_memberships: removed 'membership_uuid' from position 2 and inserted at position 10 (the last column)

Unable to render {include} The included page could not be found.