Grouper SQL database provisioning example from view

Based on sql:

```
SELECT * FROM grouper_memberships_lw_v WHERE group_name LIKE 'test:%'
```

Make a view:

```
CREATE
    VIEW test_sql_sync_v
    AS
(SELECT * FROM grouper_memberships_lw_v WHERE group_name LIKE 'test:%');
```

Make a table to sync to (with no rows):

```
CREATE TABLE test_sql_sync AS SELECT * FROM test_sql_sync_v WHERE 1 != 1
```

Configure the sync in grouper.client.properties

```
# table or view where copying data from, include the schema if needed
grouperClient.syncTable.test_sql_sync.tableFrom = test_sql_sync_v

# table or view where copying data to, include the schema if needed
grouperClient.syncTable.test_sql_sync.tableTo = test_sql_sync

# columns must match in from and to tables, you can specify columns or do all with an asterisk
grouperClient.syncTable.test_sql_sync.columns = *

# if there is a primary key, list it, else list the composite keys.  note, this doesnt
# have to literally be the database prmiary key, just need to be a unique col(s) in table
grouperClient.syncTable.test_sql_sync.primaryKeyColumns = group_id, member_id, list_name

# the grouping column is what is uniquely selected, and then batched through to get data.  Optional.
# for groups this should be the group uuid
grouperClient.syncTable.test_sql_sync.groupingColumn = group_id
```

Add in an other job to run this in grouper-loader.properties

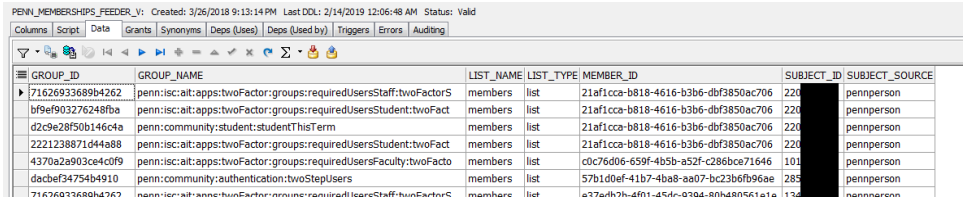
```
# Object Type Job class
otherJob.test_sql_sync.class = edu.internet2.middleware.grouper.app.tableSync.TableSyncOtherJob

# Object Type Job cron
otherJob.test_sql_sync.quartzCron = 0 0 8 * * ?

# this is the key in the grouper.client.properties that represents this job
otherJob.test_sql_sync.grouperClientTableSyncConfigKey = test_sql_sync

# fullSyncFull, fullSyncGroups, fullSyncChangeFlag, incrementalAllColumns, incrementalPrimaryKey
otherJob.test_sql_sync.syncType = fullSyncFull
```

Run the job and see the table and logs

```
finalLog: true, state: done, sync: sqlTableSync, provisionerName: test_sql_sync, syncType: fullSyncFull,
databaseFrom: grouper, tableFrom: test_sql_sync_v, databaseTo: grouper, tableTo: test_sql_sync,
retrieveDataToCount: 0, retrieveDataToMillis: 0, retrieveDataFromCount: 1381, retrieveDataFromMillis: 2239,
deletesCount: 0, deletesMillis: 0, insertsCount: 1381, insertsMillis: 1156, updatesCount: 0, updatesMillis: 0,
queryCount: 11, tookMillis: 3446, took: 0:00:03.446
```

Example: sync memberships to a table in the Grouper database or another database

1. Make a view with the data you want to sync
2. This can be based on some ad hoc groups, folders, or attributes on groups
3. For instance this could be the view



4. This view is defined (oracle) as:

```
/* Formatted on 5/1/2019 11:27:03 AM (QP5 v5.252.13127.32847) */
CREATE OR REPLACE FORCE VIEW PENN_MEMBERSHIPS_FEEDER_V
(
   GROUP_ID, GROUP_NAME,LIST_NAME, LIST_TYPE, MEMBER_ID, SUBJECT_ID, SUBJECT_SOURCE
)
   BEQUEATH DEFINER
AS
   SELECT GROUP_ID,  group_name, list_name, GMLV.LIST_TYPE, GMLV.MEMBER_ID, GMLV.SUBJECT_ID, GMLV.
SUBJECT_SOURCE
     FROM grouper_memberships_lw_v gmlv
    WHERE     (   GMLV.GROUP_NAME IN ('penn:community:student:studentThisTerm',
                                       'penn:community:student:degreePursual:degreePursual_LLM',
                                       ... )
               OR gmlv.group_name LIKE 'penn:isc:ait:apps:atlassian:%')
          AND GMLV.SUBJECT_SOURCE = 'pennperson'
          AND list_type = 'list';
```

5. Configure databases in grouper.client.properties. Note, the "id" of the config group is the part of the config that ties them together, e.g. "pcom" or "awsDev"

```
grouperClient.jdbc.pcom.driver = oracle.jdbc.driver.OracleDriver
grouperClient.jdbc.pcom.url = jdbc:oracle:thin:@(DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST = ...
grouperClient.jdbc.pcom.user = myuser
grouperClient.jdbc.pcom.pass = ******

grouperClient.jdbc.awsDev.driver = org.postgresql.Driver
grouperClient.jdbc.awsDev.url = jdbc:postgresql://penn ...
grouperClient.jdbc.awsDev.user = anotheruser
grouperClient.jdbc.awsDev.pass = ******
```

6. Configure the table sync in grouper.client.properties

   This example settings

```
# grouper client database key where copying data from
# {valueType: "string"}
grouperClient.syncTable.memberships.databaseFrom = pcom

# table or view where copying data from
# {valueType: "string"}
grouperClient.syncTable.memberships.tableFrom = PENN_MEMBERSHIPS_FEEDER_V

# grouper client database key where copying data to
# {valueType: "string"}
grouperClient.syncTable.memberships.databaseTo = awsDev

# table or view where copying data to
# {valueType: "string"}
grouperClient.syncTable.memberships.tableTo = PENN_MEMBERSHIPS_TEMP

# columns must match in from and to tables, you can specify columns or do all with an asterisk
# {valueType: "string"}
grouperClient.syncTable.memberships.columns = *

# if there is a primary key, list it, else list the composite keys
# {valueType: "string"}
grouperClient.syncTable.memberships.primaryKeyColumns = group_id, list_name, member_id

# the grouping column is what is uniquely selected, and then batched through to get data.
# {valueType: "string"}
grouperClient.syncTable.memberships.groupingColumn = group_id
```

7. Setup an other job to run this in grouper-loader.properties

```
# Object Type Job class
# {valueType: "class", mustExtendClass: "edu.internet2.middleware.grouper.app.loader.OtherJobBase",
mustImplementInterface: "org.quartz.Job"}
otherJob.membershipSync.class = edu.internet2.middleware.grouper.app.tableSync.TableSyncOtherJob

# Object Type Job cron
# {valueType: "string"}
otherJob.membershipSync.quartzCron = 0 0/30 * * * ?

# this is the key in the grouper.client.properties that represents this job
otherJob.membershipSync.grouperClientTableSyncConfigKey = memberships

# fullSyncFull, fullSyncGroups, fullSyncChangeFlag, incrementalAllColumns, incrementalPrimaryKey
# {valueType: "string"}
otherJob.membershipSync.syncType = fullSyncFull
```