Penn organizational hierarchy - part 1

This is Penn's experience implementing the Grouper organization hierarchy. Note, we used to put the fully qualified org name in the group name, but now we have the group name as the org extension without the ancestor path.

This is the size of our org implementation:

- 27,000 people in orgs at Penn
- 2,200 orgs (700 leaf nodes, 1500 rollup nodes)
- 3,000 org groups (more due to include/exclude lists)
- 500,000 org memberships (there are a lot due to the rollups, and include/exclude lists)

Performance:

- Org loader: 1 minute 20 seconds (after the first load), which equates to 630 groups and 24000 immediate memberships per minute
- Rollup loader: 1 minute 40 seconds (after first load), which equates to 1000 groups per minute
- Center loader: 40 seconds
- Consultant loader (currently one group managed): 4 seconds

General approach

- We have a view of orgs and parent id's, and a view of assignments of org to person
- There are a few loader jobs:
    - One that manages all the leaf node orgs groups (assigns people to org groups)
    - One that manages all the non-leaf rollup nodes (assigns orgs to parent org groups)
    - One that manages Penn "centers" which are our high level organizations (assigns direct center children orgs to center groups)
    - One that manages contractor groups. We dont have contractors assigned in our org structure, so we manage them through attributes in our person database. Then we can manually make other groups next to the org rollup which includes the org rollup and the contractor group
- We will run all these jobs once daily after our payroll jobs run
- Grouper include/exclude automatically creates include and exclude lists which are tacked on to the system of record group. This is useful in loader jobs since the loader system of record list is resynced with the DB query with each run, so manual changes will be undone. Penn actually only needs "include" lists, not exclude at this point (use case is VP's are in a different org than the org they manage, so we want to add them in). So we will have include/exclude groups on the rollup groups, the centers groups, and the contractor groups. We dont think we will need them on the leaf nodes, though if we have a need later, we can add it in. The reason not to do this is performance, it creates a bunch more groups and memberships.

Here are the steps to creating a loader SQL_GROUP_LIST job:

- First of all, you should prefer using views, and simply put simple select from the view in the loader config. It is also easy to tell what the loader is going to do without hunting through the loader configs, and easy to make changes at runtime (though I believe the loader allows runtime query changes as well)
- Note: if you are using include/exclude, then the group names should have the system of record suffix which is configured in the grouper.properties
- Make a view of groups where each row represents a group.
    - There is a col for the group name, display name (optional), description (optional), and security (e.g. readers, viewers, etc: optional)
    - This view will set these attributes of group and auto-create groups which have no members (might be useful for orgs since apps can refer to groups which have no members)
    - Note: if you can give groups a unique suffix in the stem structure, then the job can use the setting "grouperLoaderGroupsLike" which will delete groups which are no longer in the group list
- Make the query which returns the subjectId (and sourceId if not the default loader source), and group name
    - For leaf nodes, this is generally a simple query that assigns people to org groups
    - For rollup nodes, this can be a little complex.
        - First of all, you might union the direct rollup children with the direct rollup leaf nodes
        - The subjectId for groups is the group_id. for Grouper 1.4, you can join to the grouper_attributes table. For 1.5 you can simply join to the grouper_groups table. In both, you can join to grouper_groups_v if you like. What I did for 1.4 is: grouper_attributes ga, grouper_fields gf where gf.NAME = 'name' AND gf.ID = ga.field_id and ga.VALUE = ocrv.MEMBER_GROUP_NAME. I would definitely keep these query in a view.
        - You should also specify the source id for groups: 'g:gsa' as SUBJECT_SOURCE_ID
- Configure the config group for each loader job. I put this next to the top level loaded stem. I generally do this in GSH, though you could also do this in the UI
- Kick off the loader job manually in GSH so you can verify the results without waiting for the cron to run
- Restart your loader so it picks up the new job

Person orgs (leaf nodes)

- Lets make a function which strips out special chars:

```
CREATE OR REPLACE PACKAGE AUTHZADM.authzadm_pkg
AS

   FUNCTION remove_special_chars (the_input varchar2)
      RETURN varchar2;

END authzadm_pkg;
/

CREATE OR REPLACE PACKAGE BODY AUTHZADM.authzadm_pkg
AS

   FUNCTION remove_special_chars (the_input varchar2)
      RETURN varchar2
   AS
     the_string varchar2(4000);
   BEGIN
     --take out anything not alphanumeric, space, underscore, or dash
     the_string := REGEXP_REPLACE(the_input, '[Penn organizational hierarchy - part 1^a-zA-Z0-9 _\-]', '');
     the_string := trim(the_string);
     return the_string;
   END;

END authzadm_pkg;
/
```

* Implement the view of orgs. Note, in the view you can easily use unions in the sql to end the hierarchy at a different node. At first we were going to do this, then we decided against it. However, you will see that we do filter out certain branches and nodes. Also note we shorten the names of some top level nodes.

```
CREATE OR REPLACE VIEW ORG_LIST_V
(ORG_NAME, ORG_DISPLAY_NAME, ORG_DESCRIPTION, PARENT_ID, PAYROLL_FLAG,
 CENTER_CODE, CENTER_NAME, center_code_assign)
AS
select trim(organization_code) org_name,
decode(organization_code,      'UNIV',      'Penn',
    'NOTU',     'Not Acad',
    'TOPU',     'Top',
    authzadm_pkg.remove_special_chars(nvl(oc.description, oc.ORG_SHORT_NAME))) org_display_name,
authzadm_pkg.remove_special_chars(nvl(oc.description, oc.ORG_SHORT_NAME)) org_description,
trim(parent_org_code)  parent_id,
oc.PAYROLL_FLAG,
authzadm_pkg.remove_special_chars(oc.CENTER_CODE) center_code, authzadm_pkg.remove_special_chars(oc.
CENTER_NAME) center_name,
/* if the parent is in the same center, dont list it, only list it if the parent is in a different center */
(select authzadm_pkg.remove_special_chars(oc.CENTER_CODE) from diradmin.dir_org_codes oc2
where oc.PARENT_ORG_CODE = oc2.ORGANIZATION_CODE and oc.CENTER_CODE != oc2.CENTER_CODE ) as center_code_assign
from diradmin.dir_org_codes oc where oc.ENABLED = 'Y'
and oc.organization_code not in ( 'DEAD', 'BUD5', 'RADA', 'T', 'CAOP')
and oc.PARENT_ORG_CODE not in ( 'DEAD', 'BUD5', 'RADA', 'T', 'CAOP')
 and oc.description is not null
```

- This shows the following data (2200 rows)

```
ORG_NAME ORG_DISPLAY_NAME                       ORG_DESCRIPTION                      PARENT_ID
PAYROLL_FLAG CENTER_CODE CENTER_NAME
78YY     ACP Other Parent                       ACP Other Parent                     78XX
N            78          Audit Compliance and Privacy
9985     AG-Center for School Study Councils    AG-Center for School Study Councils  AG32
N            99          External Organizations (Agency Funds)
4602     AG-Institute on Aging                  AG-Institute on Aging                IAGE
Y            40          School of Medicine
IAGE     AG-Institute on Aging Parent           AG-Institute on Aging Parent         SOMI
N            40          School of Medicine
```

- Here is the view which assigns people to orgs. This view makes sure the person has at least one active job in that org (doesnt have to be the primary job)

```
CREATE OR REPLACE VIEW ORG_ASSIGN_V
(ORG_CODE, PENN_ID)
AS
Select distinct p.job_org_code org_code, a.v_penn_id penn_id
from comadmin.ssn4_affiliation_view a, comadmin.pennpay_appointment_v p
Where a.v_penn_id = p.penn_id And a.v_source = 'PENNPAY' and a.V_ACTIVE_CODE = 'A'
and p.JOB_ACTIVE_CODE = 'A'
```

- This will give the following data (cleansed). The penn_id is the subject_id of the person

```
ORG_CODE    PENN_ID
0007        12345678
8105        12345679
```

- Make a view about the person org metadata

```
CREATE OR REPLACE FORCE VIEW ORG_LOADER_PERSON_META_V
(
   GROUP_NAME,
   GROUP_DISPLAY_NAME,
   READERS,
   VIEWERS,
   ORG_ID
)
AS
     SELECT DISTINCT
               'penn:community:employee:org:'
            || OLV.ORG_NAME
            || ':'
            || OLV.ORG_NAME
            || '_personorg'
               AS group_name,
               'penn:community:employee:org:'
            || OLV.ORG_NAME
            || ' - '
            || olv.ORG_DISPLAY_NAME
            || ':'
            || OLV.ORG_NAME
            || ' - '
            || olv.ORG_DISPLAY_NAME
               AS group_display_name,
            'penn:community:employee:orgSecurity:orgReaders' AS readers,
            'penn:community:employee:orgSecurity:orgViewers' AS viewers,
            OLV.ORG_NAME AS org_id
       FROM org_list_v olv
      WHERE olv.PAYROLL_FLAG = 'Y'
   ORDER BY 2;
```

* This data looks like this

```
penn:community:employee:org:0001:0001_personorg penn:community:employee:org:0001 - General University:0001 -
General University                                 penn:community:employee:orgSecurity:
orgReaders penn:community:employee:orgSecurity:orgViewers 0001
penn:community:employee:org:0007:0007_personorg penn:community:employee:org:0007 - General University EB Pool:
0007 - General University EB Pool                            penn:community:employee:orgSecurity:
orgReaders penn:community:employee:orgSecurity:orgViewers 0007
penn:community:employee:org:0030:0030_personorg penn:community:employee:org:0030 - Learning Alliance for Higher
Education LLC:0030 - Learning Alliance for Higher Education LLC penn:community:employee:orgSecurity:orgReaders
penn:community:employee:orgSecurity:orgViewers 0030
penn:community:employee:org:0101:0101_personorg penn:community:employee:org:0101 - Anthropology:0101 -
Anthropology                                                   penn:community:employee:orgSecurity:
orgReaders penn:community:employee:orgSecurity:orgViewers 0101
penn:community:employee:org:0102:0102_personorg penn:community:employee:org:0102 - Asian and Middle Eastern
```

```
Studies:0102 - Asian and Middle Eastern Studies                    penn:community:employee:orgSecurity:
orgReaders penn:community:employee:orgSecurity:orgViewers 0102
```

* Penn has two types of orgs: orgs that hold people, and orgs that dont (they hold other orgs). So I will create them separately, here is a view of orgs that hold people, that the loader will use. Note: these will not be include/exclude since we only need that on the higher level groups (rollups). Note, each group here should end in _personorg so we can know which groups are managed by this loader process.

```
/* Formatted on 3/10/2013 12:47:28 PM (QP5 v5.163.1008.3004) */
CREATE OR REPLACE FORCE VIEW ORG_LOADER_PERSON_V
(
   GROUP_NAME,
   SUBJECT_ID
)
AS
   SELECT DISTINCT olpmv.GROUP_NAME, oav.PENN_ID AS subject_id
     FROM ORG_LOADER_PERSON_META_V olpmv, org_assign_v oav
    WHERE olpmv.org_id = oav.ORG_CODE;
COMMENT ON TABLE ORG_LOADER_PERSON_V IS 'view which the loader uses to load people to payroll orgs';
```

- Add the config group. Note, there are no org members yet (1=0), so I can inspect the grouperorgs_hierarchical table

```
[appadmin@lorenzo bin]$ ./gsh.sh
Type help() for instructions
gsh 0% GSH_DEBUG=true
true
gsh 1% grouperSession = GrouperSession.startRootSession();
edu.internet2.middleware.grouper.GrouperSession:
9702ff7015a84c019ae58ce6ae950115,'GrouperSystem','application'
gsh 2% stem = StemFinder.findByName(grouperSession, "penn:community:employee");
stem: name='penn:community:employee' displayName='penn:community:employee' uuid='3cb63130-03e1-4b60-8f01-
1454ee3c9588'
gsh 4% group = addGroup("penn:community:employee", "orgConfig", "orgConfig");
group: name='penn:community:employee:orgConfig' displayName='penn:community:employee:orgConfig'
uuid='f36a72d38053405d997ee8fc6eb66ff4'
gsh 5% groupAddType("penn:community:employee:orgConfig", "grouperLoader");
true
gsh 6% setGroupAttr("penn:community:employee:orgConfig", "grouperLoaderDbName", "grouper");
true
gsh 7% setGroupAttr("penn:community:employee:orgConfig", "grouperLoaderQuartzCron", "0 46 6 * * ? ");
true
gsh 8% setGroupAttr("penn:community:employee:orgConfig", "grouperLoaderQuery", "select group_name,
subject_id from org_loader_person_v where 1=0");
true
gsh 9% setGroupAttr("penn:community:employee:orgConfig", "grouperLoaderScheduleType", "CRON");
true
gsh 10% setGroupAttr("penn:community:employee:orgConfig", "grouperLoaderType", "SQL_GROUP_LIST");
true
```

* Run the job

```
[appadmin@lorenzo bin]$ ./gsh.sh
Type help() for instructions
gsh 0% grouperSession = GrouperSession.startRootSession();
edu.internet2.middleware.grouper.GrouperSession:
6e1432f7de314aeca2f927f939f1a5be,'GrouperSystem','application'
gsh 1% group = GroupFinder.findByName(grouperSession, "penn:community:employee:orgConfig");
group: name='penn:community:employee:orgConfig' displayName='penn:community:employee:orgConfig'
uuid='f36a72d38053405d997ee8fc6eb66ff4'
gsh 2% loaderRunOneJob(group);
loader ran successfully, inserted 0 memberships, deleted 0 memberships, total membership count: 0
```

~~* Inspect the grouperorgs_hierarchy table. Note, there were some problems, so we adding the function to strip bad chars and trim the data... also adjust the org_loader_person_v (so the names and everything are ok). Here is what the org_loader_person_v looks like (person data scrubbed)~~

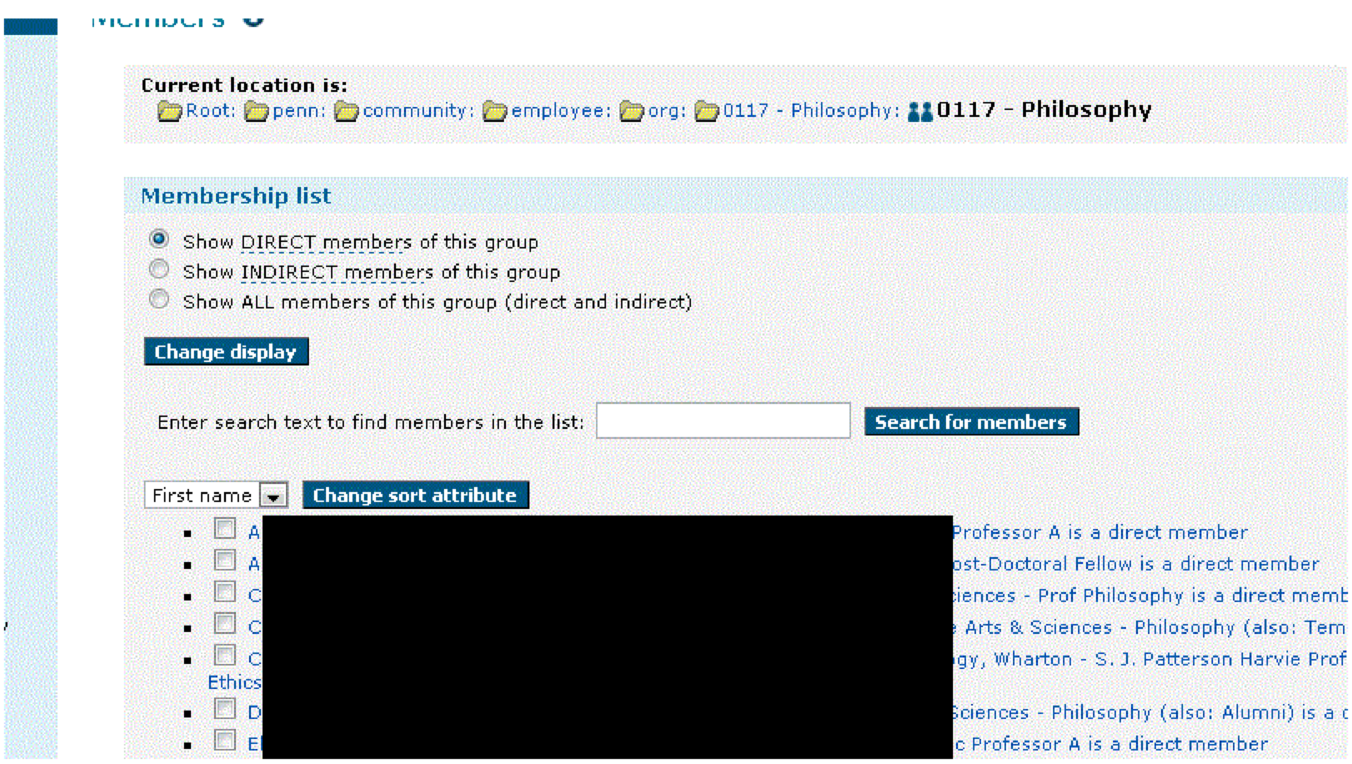
```
GROUP_NAME                                                                SUBJECT_ID
penn:community:employee:org:TOPU:UNIV:USCH:51XX:DPDN:5188:5188_personorg       12345678
penn:community:employee:org:TOPU:UNIV:USTU:85XX:CRSC:ASPP:8508:8508_personorg   12345679
```

```
penn:community:employee:org:TOPU:UNIV:USCH:36XX:APPC:3604:3604_personorg        12345680
penn:community:employee:org:TOPU:UNIV:USCH:02XX:GRAD:GRAO:0315:0315_personorg   12345681
```

* Add the group query, and fix the member query (take out 1=0)

```
gsh 5% setGroupAttr("penn:community:employee:orgConfig", "grouperLoaderQuery", "select group_name,
subject_id from org_loader_person_v");
true
gsh 6% setGroupAttr("penn:community:employee:orgConfig", "grouperLoaderGroupQuery", "select olpmv.
GROUP_NAME as group_name, olpmv.GROUP_DISPLAY_NAME as group_display_name, olpmv.READERS, olpmv.VIEWERS,
olpmv.ORG_ID from ORG_LOADER_PERSON_META_V olpmv");
true
gsh 7% setGroupAttr("penn:community:employee:orgConfig", "grouperLoaderGroupsLike", "penn:community:
employee:org:%_personorg");
true
gsh 8% loaderRunOneJob(group);
```

- This created 736 org groups with 33k members



Current location is:
Root: penn: community: employee: org: 0117 - Philosophy: **0117 - Philosophy**

Membership list

[✓ Show DIRECT members of this group]
Show INDIRECT members of this group
Show ALL members of this group (direct and indirect)

Change display

Enter search text to find members in the list: Search for members

First name Change sort attribute

- ☐ A [redacted] Professor A is a direct member
- ☐ A ost-Doctoral Fellow is a direct member
- ☐ C iences - Prof Philosophy is a direct memb
- ☐ C Arts & Sciences - Philosophy (also: Tem
- ☐ C gy, Wharton - S. J. Patterson Harvie Prof
  Ethics
- ☐ D Sciences - Philosophy (also: Alumni) is a c
- ☐ E c Professor A is a direct member

Penn organizational hierarchy - part 2