Groups

Users have requested the creation of 'groups' since they have found sharing using spaces alone to be cumbersome. If a user wants to share a space with 'students in 2 classes' this would require manually adding each student to each space, whereas the creation of a 'group' would allow the creation of a group once with name of class, and then it would be easy to add people to multiple spaces. This would also make removal easier. A group could be called 'ECE 200 Fall 2019' and during the semester, they are given access to a space, and at the end of the semester, that group is removed.

This would also help prevent the problem where users want to grant access to a space to a group of people but forget to add certain individuals. A professor (A) can create a group of their graduate students. Another professor (B) gives that group access to a space. If the first professor gets a new graduate student, they are added to "A Group" and since B gave "A Group" access the new student can see everything in the space.

The following describes how permissions and the storing of space and roles might be changed due to this feature.

Currently all the spaceandrole data is stored in the user object in the field spaceandrole. If this field is removed some new object would have to be created to store this data. It will need to easily work with how permissions are already checked in api.Permissions.

Right now, a permission is checked by providing an [objectId, objectType, userid, Permission.] For the spacedIds of the object, it is checked whether the Role of the user in that space has the correct permission. If so, the permission is true. Currently, a user can only have one role in a space, but if a user can belong to groups, they might have different roles.

What would we need to check permissions? If we have a spaceId, we would then need not just a single role, but a list of Roles. So now, for each space, we need an object or table that can store the following :

1. id of space
2. a list of objects that would contain the following:
2.1 - a Role
2.2 - users with that Role
2.3 - groups with that role

This could be made more general by

1. id of object
2. type of object (space, dataset, collection)
3. subobject
3.1 Role
3.2 Users with that role
3.3 Groups with that role

Achieving this with one object would have

1. id of object
2. type of object
3. Role
4. Users with that role
5. Groups with that role

The 'Role' can also be replaced with the name of the Role if permissions will not be explicitly stored.

if the last case is used, then when we find all matches of that object type that contain the id of object, the id of user, a the id of a group containing the user. We return all the Roles that result from that query, and then we return the list of permissions for that Role.

FilteredQuery in all MongoDBService implementations would also need to be changed. Right now the UserSpaceAndRole stored in the User object is used to check permissions to determine 'okspaces' and such. It would have to be changed to find all objects of type that have

id of space
user in the userlist
a group of user in grouplist

return Roles, from Roles, return Permissions
check against permission in query.