**Sharing Rules Enhancement**

Client needs a way to extend sharing rule limits.  Currently the limit is 300 rules with up to 50 criteria based rules, if available for the object.

Currently groups have 3-4 users and client is looking to have up to 3000+ users which would mean that we would need 750+ rules for each group

In order to make this work like standard Salesforce:

1. Create a custom object called group member tracker
   1. It has reference user id and group id
   2. We need a custom object that has group name and group id
   3. We run a batch file (maybe once a day) to automatically add opportunities to the group
   4. Also we can update the sharing when the member is first added to the group

Implementation:

**Obstacles:**

1. GroupMembers and Groups are not triggerable.  This means that if a user is added or removed from a group, there is not a way to trigger sharing rules through apex.  If a group is removed, we have no way of knowing which opportunityshare records to update.
2. The way sharing rules is being used is if a user is a member of a group then all opportunities that they own are shared with that group.  The sharing rules indicates if the records are read only or read/write. So we need to track:
   1. The user id
   2. The opportunities associated with the user: opportunity object
   3. The groups associated with the user: groupmember object
   4. The opportunities shared with the group: opportunityshare object (OpportunityID, UserOrGroupID, OpportunityAccessLevel)
   5. If the owner of the opportunity changes: opportunity object
3. Issue 2 continued:  The problem is we are looking at three SOQL queries.  Total number of SOQL queries available is 50,000 but we can only do 10,000 DML statements.  This means that if we have over 10,000 records to insert, update, upsert or delete we would have to break it into separate Apex transactions.

**Possible Solutions:**

1. Obstacle 1:  We can run a scheduled batch that counts the number of groupmembers.  Groupmember records will comprise of a groupId and a UserId. When a member is removed from the group, the record is deleted.  This batch could run every hour or maybe even with shorter intervals depending on how active the ORG with regards to batches. I simply don’t know the level of activity of this ORG.  When a count difference is detected, we can go through and update sharing for opportunities based on owner. Once a count change is detected we would have to determine the missing member and then remove sharing rules for those opportunities.
2. Obstacle 2: In order to reduce SOQL queries we can have an sobject structure that tracks Groups, Users, and Opportunities.  Relationship is like this:

 Diagram 1:

 Opportunity-<GroupDef-<userid

1. Diagram 1 object definitions:
   1. Opportunity is opportunity ids.  One opportunity can belong to multiple groups.
   2. GroupDef contains groupID for each user.  With flag indicating readonly or read/write.  Example groupID:E would be GroupID with Editing right (which is read/write).  A group definition can have multiple users
   3. UserID are the users that belong to the group definition.

Examples:

Share record with users in a group:

opportunityshare opshare = new opportunityshare(userOrGroupId='00G4P0000052hLC', OpportunityId='0064P00000mT12KQAS', OpportunityAccessLevel='Read');

insert opshare;

Additional Info:

1. OpportunityShare object records update when a user is removed from the group or the group is deleted

Testing:

Based on my testing when we change the opportunity owner, the sharing record is automatically removed.  This means that if the new owner is in the shared group, we have to account for that.

I tested 1000 groups and was able to add a sharing record for each.

Testing Scenario:

1. New Opportunity: When an opportunity is created, check the User ID and find groups associated with that UserId and share the opportunity with that group
2. Owner ID changes: If the owner id is changed, get the new user id, check the groups that that user id is associated with and make sure that the opportunity id is shared with those groups
3. User is added to group.  This is handled by SF
4. User is removed from group. Handled

First Prototype Test:

1. Adding 100 opportunities resulted in 100001 error with two group members. Why?

Remember:

1. Add help page with instructions and video