Name - Nidhi Praveen Jain **Data Store – Hbase**

I have made few changes to my schema in Part 2 of the assignment and have reverted few of those changes. So, I would like to give a overview of the final schema.

BG’s conceptual model can be represented using three tables depicted below. Members table has 5 column families namely MemberAuthDetails, Details, confFrnds, pendFrnds and resIds.

MemberAuthDetails contains authentication data like username, password and emailId. Details column family contains all the other data. ConfFrnds is a list of all the confirmed friends of that user whereas pendFrnds is the list of all the people who have sent a friend request to that user but is still not accepted by the user. resIds is the list of all the resources created by that user.

Resources table has 2 column families namely ResourceDetails and ManipulationDetails. ResourceDetails contains all the data of the resource when it is created. ManipulationDetails contains data of comments on the resource. Initially, manipulationDetails column family will not hold any data.

Friends table has 1 column family namely FriendDetails. The row key of friends table is the concatenation of inviterId+inviteeId. FriendDetails column family contains data of both the users and the status of friendship between them.

HBase logical data model for BG:

Table members:

Rowkey	MemberAuthDetails			Details										confFrnds	pendFrnds	resIds
userId	email	uname	pwd	fname	lname	address	tel	dob	jdate	ldate	gender	pic	tpic			

Table resources:

Rowkey	resDetails					manipulationDetails						
rId	creatorId	walluserId	type	body	doc	mid	creatorId	rId	modifierId	timestamp	type	content

Table friends:

Rowkey	frndDetails		
userId1+userID2	userId1	userID2	status

JSON representation:

```
Members : {
        “userid” : {
                “MemberauthDetails” : {
                        “Email” : “”
                        “UName” : “”
                        “Pwd” : “”
                        }
```

```
            }

                    “MemberDetails” : {
                            “fName” : “”
                            “lName” : “”
                            “Address” : “”
                            “tel” : “”
                            “gender” : “”
                            “DOB” : “”
                            “jDate” : “”
                            “lDate” : “”
                            “pic” : “”
                            “tpic” : “”
                            }
                    “confFrnds” : []
                    “pendFrnds” : []
                    “resIds” : []
                    }
            }

    Resources : {
            “Rid” : {
                    "ResourceDetails” : {
                            “creatorid” : “”
                            “walluserId” : “”
                            “type” : “”
                            “body” : “”
                            “doc” : “”
                            }
                    “ManipulationDetails” : {
                            “mId” : “”
                            “creatorId” : “”
                            “rId” : “”
                            “timestamp” : “”
                            “modifierId” : “”
                            “type” : “”
                            “content” : “”
                            }
                    }
            }

    Friends : {
            “UserId1+UserIId2” : {
                    “FriendsDetails” : {
                            “userId1” : “”
                            “userId2” : “”
                            “status” : “”
                            }
                    }
            }
```

The data stored within each of these cells is in the form [Timestamp]:[column value]

Actions:

1. Viewprofile: We need to get the profile details for the specified userId(rowkey of members table). For the particular userId, use the 'get' data model operation to get the count of pending friends, confirmed friends and resource count from the members table using confFrnds, pendFrnds and resIds column families. We also have to get the other member details like first name, last name, DOB, joining date, leaving date, telephone number, images from the details column family.

2. listFriends : We have to get the data from the confFrnds column family of the members table. Then, map this data to the members table to get the remaining attributes of the user.

3. viewFriendRequest : We have to get the data from the pendFrnds column family of the members table. Then, map this data to the members table to get the remaining attributes of the user.

4. acceptFriend – When a user sends a request to another user, the entry is 'put' to the friends table where FriendsDetails:userId1 is the sender userId and FriendsDetails:userId2 is the receivers userId. The FriendsDetails:status of of this entry is pending initially which needs to be changed to confirmed when the user2 accepts the friend request.
   Also, initailly userId1 will be in the pendFrnds columnFamily of userId2's member row. We have to delete it from pendFrnds and insert a confirmed friend row in the confFrnd column family of both the userIds.

5. rejectFriend – When a user sends a request to another user, the entry is 'put' to the friends table where FriendsDetails:userId1 is the sender userId and FriendsDetails:userId2 is the receivers userId. The FriendsDetails:status of of this entry is pending initially which needs to be changed to 'rejected' if the user2 rejects the friend request.
   Also, initailly userId1 will be in the pendFrnds columnFamily of userId2's member row. We have to delete it from pendFrnds.

6. inviteFriend – When a user sends an invitation to another user, 'put' an entry in the friends table where the inviter is FriendsDetails:userId1 and the reciever of the request is FriendsDetails:userId2. The FriendsDetails:staus of this entry will be 'invited'.
   Also, we put an entry of userId1 in the pendFrnds column family in members table of userId2.

7. thawFriendship - When a user sends a request to another user, the entry is 'put' to the friends table where FriendsDetails:userId1 is the sender userId and FriendsDetails:userId2 is the receivers userId. The FriendsDetails:status of of this entry is pending initially which is then changed to confirmed and the needs to be changed to 'thawed' if a user deletes user2 from its friends list.
   Also, we have to remove the entry of each user user from the other user's confFrnds column family in the members table.

8. viewTopKResources – From the resource table, get the top K resources i.e. get the details of first K resources based on timestamp for a particular userId. We have to get details of the k most recent resources based on resources:timestamp. Compare userId, with ResourceDetails:creatorId in the resources table and get the resourceId(rowkey of resource table),

ResourceDetails:walluserId, ResourceDetails:type, ResourceDetails:body, ResourceDetails:doc of the resource.

9. ViewCommentOnResource – For a particular resourceId(rowkey of resource table), get the manipulation details like ManipulationDetails:manipulationId, ManipulationDetails:modifierId and ManipulationDetails:content from the resource table where ManipulationDetails:type = comments.

10. PostCommentonResource – For a particular resourceId(rowkey of resource table), 'put' an entry for ManipulationDetails:type = comment and content as the comment to be posted for that resourse in the manipulation details of the resource.

11. delCommentonResource – For a particular resourceId(rowkey of resource table), 'delete' entry from the manipulation details where ManipulationDetails:type=comment and the user who is trying to delete the comment should be the ManipulationDetails:modifierId.

Data Store Benchmarking Survey

a. How many man hours did you spend modifying your schema? 4

b. How many hours did you spend understanding BG’s 11 actions, including their functionality and requirements? 15

c. How many man hours did you spend on writing code for BG’s 11 actions? 10-12

d. How many man hours did you spend getting the BG’s command line interface running and understanding how to use it? 2

e. How many man hours did you spend testing your code written for BG’s 11 actions? 4

f. How many man hours did you spend understanding the parameters required to execute the benchmarking phase of BG? 2

g. How many hours did you spend understanding the output of BG’s benchmarking phase? 2

h. What were the resources (e.g. bgbenchmark.org, in-class tutorial, BG paper, BG slides, google forum, TA’s help) you used for doing this part of the homework and which one was more useful? bgbenchmark.org, google forum, TA’s help. Bgbecnchmark.org was most useful.