Project Milestone 3

Name: Hardee Rameshchandra Garala **Std. ID**: B00869195

Faculty: Prof. Mike McAllister **Date**: 26th Nov 2021

Objective

To document the idea for implementation of project by describing the data structures used to store the essential data, design of code and algorithms which by using the database accomplish the desired results.

Data Structures

I am using tree data structure to store all person. I am using 2 arraylists. One of type PersonIdentity to store all the people. One of type FileIdentifier to store all objects of file. I have created 6 tables:

1. Contains id, and name of the person.
2. Contains references and id of the person.
3. Contains notes and id of the person. Will be implementing table similarly for files details.
4. Contains id of a person and id of person's child
5. Contains tag

Code design and key algorithms

When addPerson() method is called I store the name in the person table. I have set the id to auto increment and primary key, so when a new name is added a unique id is created. I retrieve this id from database and create an object using this id and name and return it.

I have made a different table to record references. Where I store reference_id and person_reference everytime recordReference method is called. Reference_id is foreign key for the id of the person table so that multiple values for same id are stored in different line and data of same id can be mapped with the person in order to find for other methods. Similarly implemented for recordNote() method by creating a different table to store notes.

For recordAttributes() method I am giving the user instruction to provide the key of the data in specific format. As I have made columns for all the attributes listed in assignment in person table. So that by comparing the key the value can be stored in respective column. For an unknown column exception will be thrown.

I have created a table to store children through recordChild() method. This table contains 2 columns, id of parent and id of child. Id of parent is a foreign key as there can be many children of a person. Similarly, for partnering and dissolution. After, adding a child in the database, I retrieve id and place it into the family tree accordingly.

File table contains id of the file which is unique, location of file is also unique and name of the file. For recordMediaAttributes() method I am listing out the keys for the user to use so that the keys remain unique. If a user adds different key then throws an error.

For tagMedia method a new table is created where one column is of media id and other is tags. One media id can have multiple tags and here media id is unique.

In findPerson method the passed name is checked in the database if it is repeating then gives error. If unique then retrieves the id and create a new object from this data and returns that object. Similarly for findMediaFile() method. In findName() method as the object is passed the name can be obtained from the object.

To find relation between two person using findRelation() method the id of both person can be obtained form database and this id can be checked in tree. From this constructed key degree of cousinship and level of removal can be calculated as per below algo [1].

1. Find number of ancestors X have from the designed tree.
2. Find number of ancestors Y have from the designed tree.
3. Take Number of ancestors of X – Number of ancestors of Y
    - If this number is greater than 0: It means that X has more ancestors, so move to the parent that is the difference of Number of ancestors of X and Number of ancestors of Y away from X. Then move to the next parent one step at a time for both X and Y and compare parents. At some point if there is a common ancestor that will be found and we can say if there is a relation and calculate degree of cousinship and degree of removal.
    - If number is less than 0: It means that Y has more ancestors, so perform the vice versa steps of above.
    - If number is 0: It means that both are at same level in a tree and there is no need for any node to move up to its parent and we can start comparing parents right away and move up till we find the common node.

4. For all the above 3 conditions if there is no common ancestor, then it can be said that X and Y are not related.

BiologicalRelation class will contain id of 2 persons and cousinship between them and level of removal.

For the person whose descendants are to be found we traverse through the tree in breadth first search manner. Initially we assume generation as 0. If the person with PersonIdentity identifier is found we take all of the nodes connected with it and add it to the set and increment the generation value. If it equals to the given number of generations then we return the set otherwise we continue the loop. And check for children of all the people added in this step of set. Once the loop breaks we return the created Set of PersonIdentity. Similarly we can find ancestors.

Reference:

[1] H. Garala, *“Project milestone 1”*.