Prachi Prakashbhai Raval
B00883324
pr647062@dal.ca

14th December ,2021

CSCI 3901 - FINAL PROJECT

FINAL DOCUMENT:
FAMILY TREE MANAGEMENT, MEDIA ARCHIVE MANAGEMENT AND GENEOLOGY

Index

Prachi Prakashbhai Raval
B00883324
pr647062@dal.ca

14th December,2021

Overview

In the 21st century, digital media like pictures and videos are used by every single individual. It has become very easy to click pictures and store them. This abundance of storage available has led people to store large number of media files and not deleting them. People have been storing large number of media files especially in their smart phones. With large amount of data, it becomes more difficult to find images and videos.

Genealogy is the study of tracing family origins and history. Tracking and charting where you came from and who your ancestors were is an example of genealogy. But it does not have any relation with the media files, pictures and videos that exist currently in a digital format, or archive pictures that are now being digitized.

A genealogist would when asked about the great grandmother of person A would like to get a set of pictures for the individual as well as their name.

This project aims to create a system to link this study of family origin with the digited media of archived pictures along with metadata of the pictures.

Family information and Media information is stored in the database. With the help of recorded data genealogists can answer the following questions:

-how are persons X and Y related (Relation is returned in the form of degree of cousinship and removal)

- show me the references and notes for person X

-list the descendants of person X for Z generations

- list the ancestors of person X for Z generations

- list the pictures of a particular tag within some time range

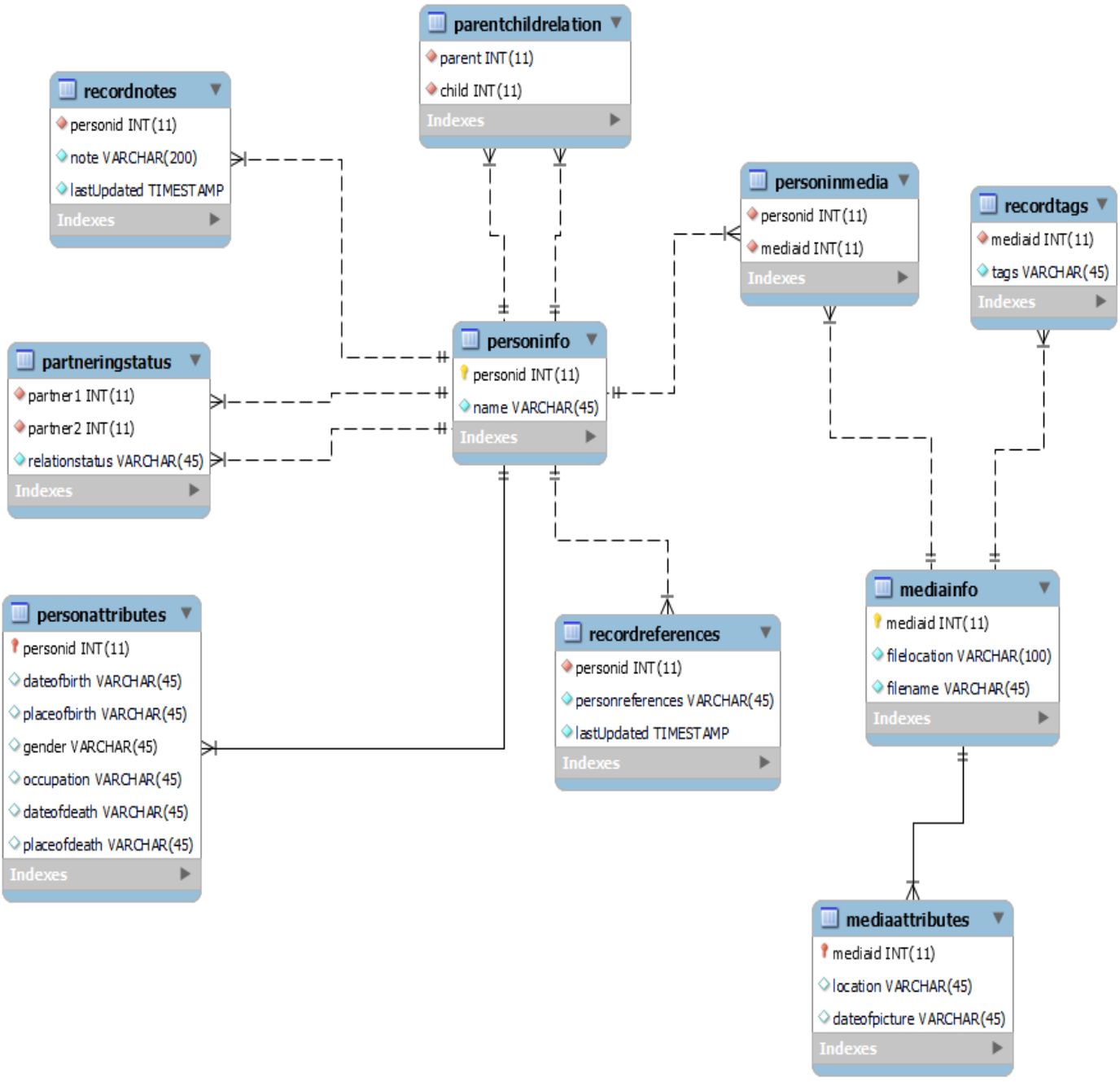
- list the pictures of a particular place within some time range

- list the pictures of a given set of people (like a couple) within some time range chronologically

- list the pictures that include all the immediate family members of person X (immediate children)

Prachi Prakashbhai Raval
B00883324
pr647062@dal.ca

14th December,2021



Prachi Prakashbhai Raval
B00883324
pr647062@dal.ca

14th December,2021

Files and External Data:

- login.properties file given to enter information regarding database connection.
- SQL file createTables, provide along with the java file, creates the tables below.
- External Document for test cases.
- 10 Java Files
    - PersonIdentity.java
    - FileIdentifier.java
    - Genealogy.java
    - BiologicalRelation.java
    - PersonInfo.java
    - MediaInfo.java
    - FamilyTree.java
    - DataConnection.java
    - MainUI.java

Note: All the methods in the project can be called by object of Genealogy class, since multiple inheritance is used. Genealogy class object must be created before initializing any part of the program. All the methods in class PersonInfo, MediaInfo, Genealogy can be called using object of Genealogy. Genealogy constructor is used to fetch existing person and media files from database which was inserted in the previous sessions of java and create objects. These objects are stored in the map.

Classes and Methods:

1. PersonIdentity - This class will consist of constructor, getters, and setters for the name and id. PersonIdentity (String name, int personId) Create new object of PersonIdenity type.
2. FileIdentifier – This class consists of constructor to create object if FileIdentifier type and getters and setters for the id and name of the person.
3. BiologicalRelation – This class consists of constructor to create biologicalRelation object and getters and setters for cousinship and removal.
4. PersonInfo – This class consists of methods related to person, used to add information about person in database. It extends class mediainfo.

   Methods :

   - PersonIdentity addPerson(String name) – This method adds new person in database and creates a new object for the same person . This object is stored in Map a where key is the id assigned to the person and value is the object of class PersonIdentity.

- Boolean recordAttributes(PersonIdentity person, Map<String, String> attributes) – This method is used to store information about person like date of birth, place of birth , gender etc. This method can be called more than once if the person does not have all the information in correctly or enters wrong data by mistake. It can update the information. Here the map passed in argument stores key as column name in the attributes table, value of map contains the data to be stored in the row. Attributes of person
    - *dateofbirth*
    - *placeofbirth*
    - *gender*
    - *occupation*
    - *dateofdeath*
    - *placeofdeath*

- Boolean recordReference (PersonIdentity person, String reference) – This method is used to store references for a person. One person can have more than one references.

- Boolean recordNote (PersonIdentity person, String note) – This method stores notes about a person. One person can have more than one notes assigned.

- Boolean recordChild (PersonIdentity parent, PersonIdentity child) – This method is called to add parent and child which are biologically related. One child can have at most two children. Here both parent and child objects must be present in the person table, or in the Map which stores objects of PersonIdentity.

- Boolean recordPartnering(PersonIdentity partner1, PersonIdentity partner2) - This method records partnership between two people. This does not define any biological relationship. If one person is aldredy partnered with someone, partnership cannot be recorded again, unless the person records dissolution with the former partner.

- Boolean recordDissolution(PersonIdentity partner1, PersonIdentity partner2) – This method records dissolution between two people. Dissolutiion can be recorded only if two people are partnered. After dissolution the status is changed from partnered to dissolved.

5. MediaInfo - This class consists of methods related to media files, used to add information about media in database. It also consists a method to record relation media and person. It stores people present in a media file.

   Methods:

   - FileIdentifier addMediaFile(String fileLocation)- This method takes unique file location as input and stores the file location and name in the table . Also adds the object created for the media file to the map.

   - Boolean recordMediaAttributes(FileIdentifier fileIdentifier, Map<String, String> attributes) – This method records attributes for media files. This method can be called more than once if the person does not have all the information in correctly or enters wrong data by mistake. They can update the information. Here the map passed in argument stores key as column name in the attributes table , value of map contains the data to be stored in the row.
     Attributes for media files:
     - *dateofpicture*
     - *location*

   - Boolean peopleInMedia(FileIdentifier fileIdentifier, List<PersonIdentity> people) – This method is used to record the relationship between media and person. It records the people present in media file. Person id which is the primary key in table personinfo and mediaid which is the primary key in the media info table both are have many to many relationship. One media file can have multiple people in it, also one person can be in multiple media files.

   - Boolean tagMedia(FileIdentifier, fileIdentifier, String tag) – This method is used to assign tags for a media file

6. Genealogy- This class is used to find useful information about the people and media files. It reports information like

   Methods :

   - PersonIdentity findPerson(String name) – This method returns PersonIdentity object by taking input person name. If there are multiple objects present with same name, this method throws an Illegal argument exception.

- FileIdentifier findMediaFile(String name) – This method returns fileidentifier object by taking input file name. If there are multiple objects with same name it throws exception.

- String findName(PersonIdentity id) – This method takes input as PersonIdentity and returns name of the object..

- String findMediaFile(FileIdentifier fileId) – This method takes input as FileIdentifier object and returns it's name.

- BiologicalRelation findRelation(PersonIdentity person1, PesonIdentity person2)- This method finds how person1 and person2 are biologically related. It then returns an object of BiologicalRelation which has two integers cousinship and removal.

- Set<PersonIdentity> descendents(PersonIdentity person, Integer generations) This method finds descendents in the family tree who are within generations of the person.

- Set<PersonIdentity> ancestores(PersonIdentity person, Integer generations) This method finds ancestors in the family tree who are within generations of the person.

- List<String> notesAndReferences(PersonIdentity person)
  This method returns all the notes and references on the individual, returned in the same order in which they were added to the family tree.

- Set<FileIdentifier> findMediaByTag(String tag , String startDate, String endDate)
  Return the set of media files linked to the given tag whose dates fall within the date range. Null values for the dates indicate no restrictions on the dates.

- Set<FileIdentifier> findMediaByLocation(String location, String startDate, String endDate)
  Return the set of media files linked to the given location whose dates fall within the date range. Null values for the dates indicate no restrictions on the dates.

- List<FileIdentifier> findIndividualsMedia(Set<PersonIdentity> people, String startDate, String endDate)

Return the set of media files that include any of individuals given in the list of people whose dates fall within the date range. Null values for the dates indicate no restrictions on the dates. Return the files in ascending chronological order (breaking ties by the ascending order of the file names).

- List<FileIdentifier> findBiologicalFamilyMedia(PersonIdentity person)
  Return the set of media files that include the specified person's immediate children.Return the files in ascending chronological order (breaking ties by the ascending order of the file names).

7. DataConnection – This class contains method to form JDBC database connection with sql, using user name password and database name from the login.properties file.

8. TreeSize- This method is called before implementing any methods of class Family Tree. This method creates enough memory to store all parent child relation in array.

9. FamilyTree – This class is used to find ancestors, descendants, and Biological Relationship in terms of cousinship and removal.

   Methods:

   - void createMemoryForNodes() – This method creates memory for all the arrays according to the number of parent child records.
   - void readParentChildTable() – This method reads the parent child rows one by one from the database and then call the method addParentChild.
   - void addParenChild(int a, int b) – This method adds the parent child relation between person 1 and person 1 into different data structures to later find out genealogy reports.
   - BiologicalRelation findBiologicalRelation(int person1, int person2) -This method finds the biological realation between person 1 and person 2. It checks three conditions to find the lowest common ancestor and then find removal and cousinship.

     ```
     - List<Integer> person1LCA(List<Integer> call) {
     - List<Integer> person2LCA(List<Integer> call)
     ```

     These two methods find all the ancestors of person 1 and person 2. Then later in findBiologicalRelation method we compare both the list and find Lowest Common Ancestor (LCA).

     - If person1 is ancestors of person 2
     - If person2 is ancestor of person 1

- If person1 and person2 are neither ancestor nor descendant of each other. Based on this information the relation is found between two people.
  If no lowest common ancestor is found between two people they are not related. If no relation is found between two people this method returns null.

- List<Integer> getAncestors(int node, int level) - This method is used for find ancestors upto certain generations, generations are defined as levels. This method find ancestors by each level using bf search . Recursive function is called to check if any more ancestors exist.
- List<Integer> recurAns(List<Integer> call) -This method is called in recursion for finding next level of ancestors.
  Similarly there are two methods as above used to find the descendants of a person till certain generations.

10. MainUI – This class has a menu based User interface for testing purposes.

Database Tables Information

PERSON

1. personinfo – This table contains name of person and identifier id for person, which is a primary key and auto increment, and it act's as a unique value assigned to a person to find information about the person.

2. personattributes – This table contains attributes of a person, where again personid is used as foreign key. Attributes can be updated if they are entered wrong or if some attribute value is added later when found. Column names for attributes are:
   - ***dateofbirth***
   - ***placeofbirth***
   - ***gender***
   - ***occupation***
   - ***dateofdeath***
   - ***placeofdeath***

3. recordnotes – This table is used to record notes about a person. Each person have more than one note. Notes are identified using personid. personid and notes have many to many relationship. Note date type is varchar(200) . Length

set for note about a person. Record note also has a timestamp column which is later used while reporting notes and references of a person in the order they were entered in the data base.

4. recordreferences – This table is similar to the recordnotes table, but references are smaller values such as a website name or location, which can be stores in varchar(45).

5. parentchildrelation- This table stores relation of parent and child in respective columns. According to the problem statement, one child can have at max two parents, thus before inserting parent child in the table, it is checked that child can have at max two parents.

6. partneringstatus - This table is used to store partnering and dissolution status .If two people are partnered the status shows "partnered" . It is updated to "dissolved" , after recording dissolution. If the partnering or dissolution between two people has happened more than once. It is recorded in the table as a new row. This keeps a track of relation status between two people.

MEDIA

1. mediainfo – This table contains information media such as medianame, medialocation, mediaid. Mediaid is used as a unique identifier to easily access media files.

2. mediaattributes – This table contains attributes of a media, where again mediaid is used as foreign key. Attributes can be updated if they are entered wrong or if some attribute value is added later when found. Column names for attributes are:
    - ***dateofpicture***
    - ***location***

3. recordtags – This table is used to record tags for a media file. Each media file can have more than one tag. Mediaid and tags have many to many relationships.

4. personinmedia- This table stores relation of person and media in respective columns. Each media is identified by mediaid and can have more than one person present in it. Person is identified by personid. There is a many to many relationship between mediaid and personid. Both columns in this table are foreign keys.

Prachi Prakashbhai Raval
B00883324
pr647062@dal.ca

14th December,2021

Data Structures:

- These two maps are used to store objects for person and fileidentifier, along with the id which corresponds to that person in the database. Key contains the id while value contains the object of PersonIdentity and FileIdentifier.

  Map<Integer, PersonIdentity> personIdentityMap = new HashMap<>();
  Map<Integer, FileIdentifier> fileIdentifierMap = new HashMap<>();
- Other data structures used to build dynamic tree with multiple roots and find Biological Relation , Ancestors , Descendants

  List<Integer>[] tree = new ArrayList[2*size of parent child relation]
  Memory occupied by the above data structure is not optimized. The size of the tree can be made equal to required size. For now to avoid any exceptions I have assumed the size to be 2 times size of number of people available.

  To find ancestors and descendants separate ArrayList array is used to store each parent's child and each child's parents.

- Other data structures used are ArrayList and Map to store levels or generation of a person.

Assumptions

- Format of date must be either yyyy-mm-dd or yyyy-mm.
- Once parent child relation is recorded, partnering cannot be done between ancestors and descendants.
- All string values must be within varchar (45) except for notes of a person which is varchar (200)
- Two people cannot record dissolution if they are not partnered before.
- Media file location is unique which contains file name.
  Format of mediafile location example:
  C:\Users\prach\OneDrive\Documents\\filename.png

Limitations

- Efficiency for both memory and speed are not as required, because everytime genealogy functions are called, entries from the database are fetch and put into data structures to find the output.