CSCI 3901 Final Project

Name: Aayushi Rashesh Gandhi
ID: B00890697

Overview
Finding a specific media has become tiresome due to the exponential development in the amount of media on a daily basis. The basic goal of this challenge is to find family relationships between persons in a media and to return a group of photographs of the individual. This problem exclusively considers family relationships and reports them in terms of cousinship and degree of separation.

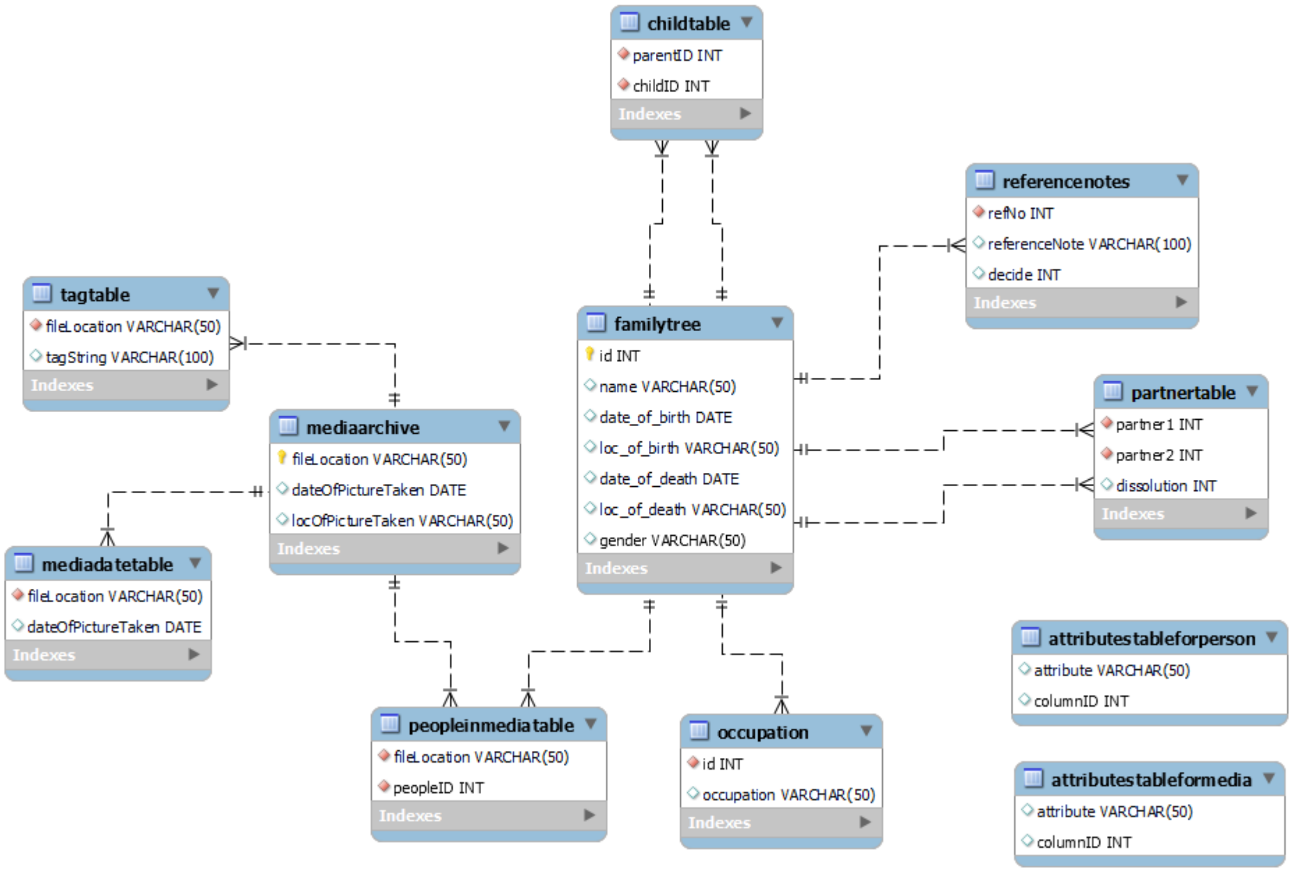
Files and external data
There are 9 files in the program:

1. **Constant.java:** To store all the constant values throughout the program. It stores credentials for SQL database connection. Additionally, it stores all the database table names.
2. **SqlConnection.java:** This class makes connections with a SQL database using driver manager.
3. **PersonIdentity.java:** This class is an identifier for an individual in the family tree. It adds a person in the database, records its attributes, records references and notes. Additionally, it stores a parent child relationship, partnering relationship, and dissolution relationship.
4. **FileIdentifier.java:** This class is an identifier for a media in the media archive. It adds a media to the database, and records its attributes. Furthermore, it records a list of people that appear in a particular media and records tags for a particular media.
5. **BiologicalRelation.java:** This class specifies the level of cousinship and degree of removal in a relation. It maps relations between individuals, it finds relations between two persons in terms of level of cousinship and degree of reparation.
6. **Genealogy.java:** This class helps in finding a person, a media, a relation between two individuals, descendents of an individual, ancestors of an individual. Furthermore, it provides all the notes and references for a particular individual, it finds media by tag and location, finds individuals who are present in a set of media, and provides a set of media which contains an individual's immediate children.
7. **TestCases.java:** A test file that contains JUNIT test cases.
8. **SQL queries**: Contains sql queries to create tables
9. **SQL Sample Data:** Contains sample data inserted into the tables

SQL Tables

1. **familyTree:** A family tree table that stores all the individuals and their attributes such as id, name, date of birth, location of birth, date of death, location of death and gender.
2. **occupation:** An occupation table that stores all the multiple occupations of an individual. It stores the ID of an individual and their respective occupation.
3. **referenceNotes**: It stores all the notes and references of an individual in the same order in which they were added.

4. **childTable:** A child table that stores a parent's ID and their respective children's IDs.
5. **partnerTable:** A partner table that stores a person's ID and their respective partner IDs. It contains a dissolution column which contains 0 value when both persons are partnered and contains value 1 when dissolution is performed between two individuals.
6. **mediaArchive:** A media archive table that contains all the media files information such as media file name (in terms of media file location), the date of picture taken and location of picture taken (which includes the location, city, province and country).
7. **peopleInMediaTable:** It stores all the individual's ID that appear in a particular media. It stores file location of a media and the respective individual's ID.
8. **tagTable:** A tag table stores all the tags that are linked to a particular media. It stores the media location and the respective tag.
9. **mediaDateTable:** It stores all the media whose dates fall within the date range. It is a temporary table so as to get the list of media in the ascending chronological order of dates.
10. **attributesTableForPerson:** It stores all the possible keys for attributes of the person. It stores the key of an attribute and the respective column number of the family tree table for that attribute.
11. **attributesTableForMedia:** It stores all the possible keys for attributes of the media. It stores the key of an attribute and the respective column number of the media archive table for that attribute.



Entity-Relation Diagram

Data structures and their relations to each other

Key algorithms and design elements

1. **BiologicalRelation findRelation(PersonIdentity person1, PersonIdentity person2)**
   The degree of cousinship and level of separation will be determined using this method. The family tree is used by the program to find these two variables.

   In order to determine the degree of cousinship, the program will look for a common ancestor for both person1 and person2. To do so, the program keeps track of all of the ancestors of both person 1 and person 2. Further, it finds the common ancestor index and returns the index value of the common ancestor, then it chooses the smaller of the two. After that, it will remove 1 from the value and return it.

   For the level of separation, it identifies the index value of a common ancestor from a list of ancestors, then it will find the difference and return it as the level of separation.

   These two values (cousinship and separation) will be stored in the object of the BiologicalRelation class and will be returned to the user.

2. **Set<PersonIdentity> descendents (PersonIdentity person, Integer generations)**
   This method returns descendents in the family tree who are within “generations” generations of the person.

   Pseudo code:

   ```
   1. Runs loop for "generations" count
       a. Gets all the children of the current person and adds the children in the
          descendent list.
       b. When the temporary parent list is empty it stores the children of only the
          current parent in a temporary parent list.
       c. Takes the first value from the temporary list and makes it as a current
          individual and then removes the first person from the temporary list
   2. Now we have a final descendent list with IDs of persons. A set of objects of class
      PersonIdentity is created.
   3. Stores objects of class PersonIdentity for all the individuals from descendents list
      into a descendents set by getting name of person from the ID and storing it in a
      object
   ```

3. **Set<PersonIdentity> ancestores(PersonIdentity person, Integer generations)**
   This method returns ancestors in the family tree who are within “generations” generations of the person.

   Pseudo code:

   ```
   1. Runs loop for "generations" count
   ```

        a. Gets parent/s of the current person and adds the parents in the ancestors list.
        b. When the temporary parent list is empty it stores the parent of only the current individual in a temporary parent list.
        c. Takes the first value from the temporary list and makes it as a current individual and then removes the first person from the temporary list
    2. Now we have a final ancestors list with IDs of persons. A set of objects of class PersonIdentity is created.
    3. Stores objects of class PersonIdentity for all the individuals from ancestors list into a ancestors set by getting name of person from the ID and storing it in a object

4. **List<FileIdentifier> findIndividualsMedia(Set<PersonIdentity> people, String startDate, String endDate)**
   This class returns the set of media files that include any of individuals given in the list of people whose dates fall within the date range
    a. If the start date is null then by default 0000-00-00 will be added. And if the end date is null then by default 9999-12-31 will be added.
    b. A query to get media files for particular individuals under a date range is executed and file locations are stored in fileLocationList and dates are stored in dateList.
    c. Inserts the above media files from fileLocationList list with their respective dates from dateList in a table named "mediaDateTable".
    d. A query to sort the table "mediaDateTable" is executed and it gets the media file in ascending chronological order when date is not null and appends the object of class FileIdentifier in listOfMedia list.
    e. A query to sort the table "mediaDateTable" is executed and it gets the media file in ascending chronological order when date is null and appends the object of class FileIdentifier in listOfMedia list.
    f. Finally, the list of objects of class FileIdentifier is returned

5. **List<FileIdentifier> findBiologicalFamilyMedia(PersonIdentity person)**
   Returns the set of media files that include the specified person's immediate children
    a. A query to fetch immediate children of a person is executed.
    b. Stores the immediate children of the person in the children list.
    c. Storing objects of class PersonIdentity in a people list for all the children from the children list.
    d. findIndividualsMedia() method is called and people list with objects of class PersonIdentity is passed as a parameter.

Test Cases

Blackbox test cases are mentioned in Milestone 2 and White Box test cases are mentioned in Milestone 3. Some additional test cases are:

Input Validation Tests

1. addPerson()
    - Null value or empty string passed as an individual name
2. recordAttributes()
    - Negative value passed for date of birth and date of death
    - Empty value passed for date of birth and date of death
    - More than 2 characters passed as date DD or month MM
    - Exactly 4 characters not passed as year YYYY
    - Different date format
    - Date of birth or death passed in future tense
    - Null value or Empty string passed as a gender
    - Undefined value passed as a gender
    - Null value or empty string passed as an occupation
3. recordReference()
    - Null value passed as a reference
    - Empty string passed as a reference
    - Incorrect location passed as a reference
    - Invalid web link passed as a reference
4. recordNote()
    - Null value passed as notes
    - Empty string passed as notes
5. recordChild()
    - Individual passed as a parent does not exist in family tree database
    - Individual passed as a child does not exist in family tree database
    - Same name passed for parent and child
6. recordPartnering()
    - Individual passed as a partner does not exist in family tree database
    - Same name passed for both the partners
7. recordDissolution()
    - Individual passed as a partner does not exist in family tree database
    - Same name passed for both the partners
8. addMediaFile()
    - Null value passed as a media archive
    - Empty string passed as a media archive
    - Incorrect format entered for file location
9. recordMediaAttributes()
    - Negative value passed for date of picture taken
    - Empty value passed for date of picture taken
    - More than 2 characters passed as date DD or month MM
    - Exactly 4 characters not passed as year YYYY
    - Different date format
    - Date of picture taken passed in future tense
    - Null value or empty string passed as location
    - Characters other than alphabets passed as cit name, province or country name

- Null value passed as tags
- Empty string passed as tags
- Null value passed as an individual name seen in the picture
- Empty string passed as an individual name seen in the picture
- Individual name passed that does not exist in family tree database

10. peopleInMedia()
    - Null value passed as an individual name appear in the media file
    - Empty string passed as an individual name appear in the media file
    - Any individual name passed that does not exist in family tree database
11. tagMedia()
    - Null value passed as tags
    - Empty string passed as tags
12. findPerson()
    - Null value passed as an individual name
    - Empty string passed as an individual name
13. findMediaFile()
    - Null value passed as a media name
    - Empty string passed as a media name
14. findName()
    - Null value passed as an object of class PersonIdentity
15. findMediaFile()
    - Null value passed as an object of class FileIdentifier
16. findRelation()
    - Null value passed for object of class PersonIdentity as a person 1
    - Null value passed for object of class PersonIdentity as a person 2
17. descendents()
    - Null value passed for object of class PersonIdentity as a person
    - Negative value passed as generation
    - Null value passed as generation
18. ancestores()
    - Null value passed for object of class PersonIdentity as a person
    - Negative value passed as generation
    - Null value passed as generation
19. notesAndReferences()
    - Null value passed for object of class PersonIdentity as a person
20. findMediaByTag()
    - Null value passed as tag
    - Empty string passed as tag
21. findMediaByLocation()
    - Null value or empty string passed as location
22. findIndividualsMedia()
    - Empty value passed as set of people
    - Empty or null value passed as start date or end date
23. findBiologicalFamilyMedia()

- Null value passed for object of class PersonIdentity as a person

Boundary Cases Tests

1. addPerson()
    - Single character passed as an individual name
    - Too many characters passed as an individual name
2. recordAttributes()
    - Present date passed as date of birth or death
    - Single character passed as an occupation
    - Too many characters passed as an occupation
3. recordReference()
    - Single source passed as a reference
    - Too many sources passed as a reference
4. recordNote()
    - Single note passed for an individual
    - Too many notes passed for an individual
5. recordChild()
    - A parent has a single child
    - A parent has multiple children
6. addMediaFile()
    - Single character passed as a media file name
    - Too many characters passed as a media file name
7. recordMediaAttributes()
    - Single character passed as a city name
    - Too many characters passed as a city name
8. tagMedia()
    - Single tag passed for an individual
    - Too many tags passed for an individual

Control Flow Tests

1. findPerson()
    - Locating a person that does not exist in family tree database
2. findMediaFile()
    - Locating a media that does not exist in media archive
3. findName()
    - Returning id for individuals with same name
4. findRelation()
    - Finding relations when two individuals are at same generations back
    - Finding relations when X individual is smaller generations back than Y individual
    - Finding relations when X individual is greater generations back than Y individual
    - Finding relations when two individuals are not biologically connected
5. descendents()
    - When no descendents are available for a person in the family tree
    - When only 1 descendent is available for a person in the family tree

    - When many descendents are available for a person in the family tree

6. ancestors()
    - When no ancestors are available for a person in the family tree
    - When only 1 ancestor is available for a person in the family tree
    - When many ancestors are available for a person in the family tree
7. findMediaByTag()
    - When no media file is linked to a given tag
    - When only 1 media file is linked to a given tag
    - When many media files are linked to a given tag
8. findMediaByLocation()
    - When no media file is linked to a given location
    - When only 1 media file is linked to a given location
    - When many media files are linked to a given location
9. findIndividualsMedia()
    - When no media file is available for any of the individuals
    - When only 1 media file is available for any of the individuals
    - When many media files are available for many of the individuals
10. findBiologicalFamilyMedia()
    - When no immediate child is available for the specified person
    - When only 1 immediate child is available for the specified person
    - When many immediate children are available for the specified person

Data Flow Tests

1. descendents()
    - When 0 value is passed as generations
    - When value of generation is less than the length of family tree
    - When value of generation is greater than the length of the family tree
2. ancestores()
    - When 0 value is passed as generations
    - When value of generation is less than the length of family tree
    - When value of generation is greater than the length of the family tree
3. recordPartnering()
    - Partnering with an already partnered individual
    - Partnering two already divorced individuals
4. recordDissolution()
    - Divorcing an already divorced individual
    - Divorcing two already partnered individuals
5. findMediaByTag()
    - Finding media by tags whose dates fall within the date range
    - Null value or empty string passed as start date
    - Null value or empty string passed as end date
    - Only year passed as start date or end date
    - Only year and month passed as start date or end date
    - Wrong date format passed as start date or end date

6. findMediaByLocation()
    - Finding media by location whose dates fall within the date range
    - Null value or empty string passed as start date
    - Null value or empty string passed as end date
    - Only year passed as start date or end date
    - Only year and month passed as start date or end date
    - Wrong date format passed as start date or end date
7. findIndividualsMedia()
    - Finding media by individuals whose dates fall within the date range
    - Null value or empty string passed as start date
    - Null value or empty string passed as end date
    - Only year passed as start date or end date
    - Only year and month passed as start date or end date
    - Wrong date format passed as start date or end date

Assumption

1. If the user enters only the “year” value for the date, then the program appends 0 for month and day (YYYY-00-00). Similarly, if the user enters only the “year and month” value for the date, then the program appends 0 for day (YYYY-MM-00). If only month and day are passed then it is not accepted.
2. For method recordAttributes(): If any unacceptable attribute value is passed then that value will not be stored in the database and rest all acceptable attributes will be successfully stored and the method will return true value. The program prints all the attributes that are not recorded in the database just to let the user know. If all the attributes are unacceptable then only the method returns false value.