Final Project

Overview:

- The program store and manipulate data of person and files of them. We have stored the personal details and file information in database to be persistent forever.
- Code main divide into 3 parts:
    - Person’s information
    - File’s information
    - Reporting function of person and file
- We have stored the link between 2 nodes as parent-child relationship, partner1-partner2 relationship and if want to be separated then we also have dissolution features in program.
- User can store as many as and any kind of attributes for person and for the file detail.
- There is no restriction over storing the attributes to person and file.
- However, code has some fixed attributes for more depth manipulation and more analytics.
- We also can find the relationship between 2 nodes in database anytime user want in minimum time condition.
- We also can connect files and person detail with each other and get some useful information from that.
- There are many details available for file details which can be gotten with respect of date and chronological order.

Data structure used in program:

- Arraylist
- Hashset
- Array
- Tree based structure
- Basic OOPs concepts

Total functionality of program:

- Find ancestor of node(person)
- Find descendants of node(person)
- Find the relationship between 2 nodes
- Add the (node)person in database
- Add details of person in database
- Add relationship as parent-child, partner1-partner2 and dissolution between persons
- Add file detail in database
- Add file attributes (date/location/quality etc.) in database
- Add people in image/video file
- Add tag in image/video file
- And many more.

For person(node), main functionality is to create the tree-based data structure in database using parent child relationship and can extract many details like ancestors, descendants, and relationship between 2 nodes. For this I've used mainly 3 query to get the most useful info parent-child relationship.

1. **For Descendants:**

```
WITH RECURSIVE descendant AS (
   SELECT  ParentId,
          ChildId,
          0 as level
   FROM parent
   WHERE ParentId = 87

   UNION ALL

   SELECT  p.ChildId,
          p.parentId,
          level + 1
   FROM parent p
JOIN descendant d
ON p.ParentId = d.ParentId
)

SELECT  d.ChildId AS descendant_id,
        p1.ChildId AS ancestor_id,
        d.level
FROM descendant d
JOIN parent p1
ON d.parentId = p1.ChildId
group by d.ParentId
having level  <=2
ORDER BY level;
```

2. For Ancestor:

```
WITH RECURSIVE ancestor AS (
    SELECT  ChildId,
            parentId,
            1 AS level
    FROM parent
    WHERE ChildId = 22

    UNION ALL

    SELECT  per.ChildId,
            per.parentId,
            level + 1
    FROM parent per
JOIN ancestor d
ON per.ChildId = d.parentId
)

SELECT
        a.parentId AS ancestor_id,
        d.level
FROM ancestor d
JOIN parent a
ON d.ChildId = a.ChildId
where level <=5
group by  ancestor_id,level;
```

3. Path from root to each node

```
WITH recursive person1 (PersonId, name, ParentId) AS (
      select PersonId,name,p.ParentId from person
      left join parent as p
      on person.personId = p.ChildId
),
category_path (PersonId, name, path) AS
(
  SELECT PersonId, name , CONCAT(PersonId) AS path
    FROM person1
    WHERE ParentId IS NULL
  UNION ALL
  SELECT c.PersonId, c.name,  CONCAT( c.PersonId, '-', cp.path )
    FROM category_path AS cp JOIN person1 AS c
      ON cp.PersonId = c.ParentId
)
SELECT * FROM category_path
where  personId=84 OR personId= 91
ORDER BY path;
```

References:

[1] "How to Get Descendants of a Parent in SQL," [Online]. Available: https://learnsql.com/blog/hierarchical-data-sql/. [Accessed: 12 12 2021].

[2] "https://www.mysqltutorial.org/mysql-adjacency-list-tree/," [Online]. Available: https://www.mysqltutorial.org/mysql-adjacency-list-tree/. [Accessed: 12 12 2021].

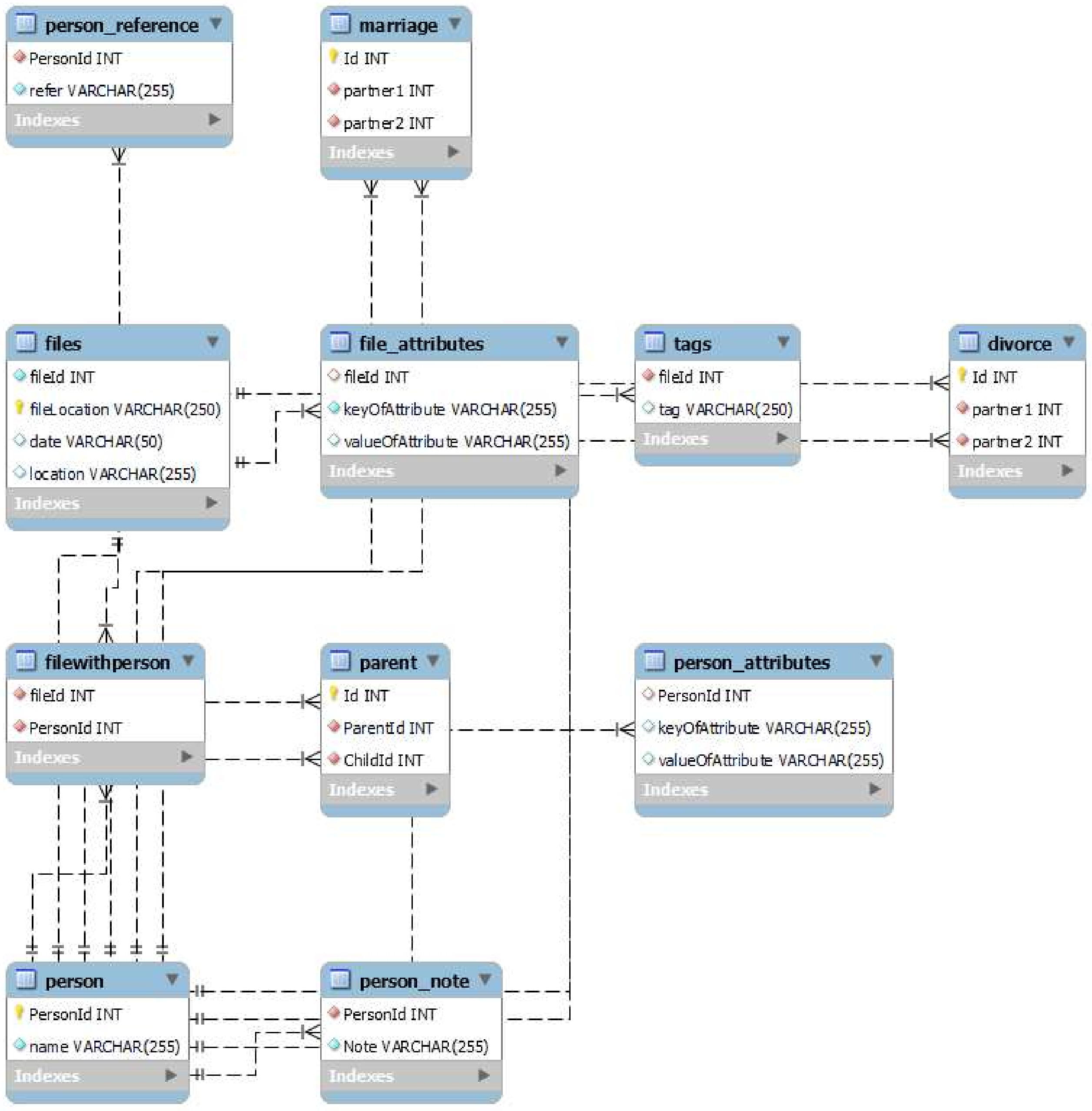
[3] "Storing trees in databases," [Online]. Available: https://makandracards.com/makandra/45275-storing-trees-in-databases. [Accessed: 12 12 2021].

[4] "Recursive Query Throwdown in MySQL 8 - Bill Karwin," [Online]. Available: https://www.youtube.com/watch?v=M4O0YQGTxjM. [Accessed: 12 12 2021].

Limitation:

- Date should be in YYYY-MM-DD, YYYY-MM, YYYY format only.
- Can not remove parent child relationship once it will be established
- Cannot store the blob content of image/video in database
- Only store the file name of images/videos
- Cannot store some portion of images and video
- Cannot throw an option for selecting the from multiple options (persons and files are which has same name)
- With GUI, we can provide the more efficient way to operate the functionalities
- Prefix attributes are not available for person and file so that no extra and accurate manipulation can be done

Database Design:



- I have stored the person in person table with primary key of personId and from that table many table has reference to its foreign key.
- personAttribbutes, parent, marriage, dissolution, person_notes, person_reference, filewithperson relate to person table with personId.
- fileAttributtes, filewithperson, tag with files table in reference of fileId as primary key in files table in fileId.

Key algorithms:

Find LCA (Lowest Common Ancestor) of X and Y and relationship of X and Y:

1. Find path from X to its most root node
2. Find path from Y to its most root node
3. Iterate paths from root to X / Y until the node is not matched
4. Get LCA using step 3
5. Count level (generation) using LCA to last node in each path and store it in Nx
6. Count level (generation) using LCA to last node in each path and store it in Ny
7. Find relationship between x and y using min{nx - ny} – 1 cousins |nx - ny| removed

List the ancestors of person X for Z generations:

1. Find parent of X in db using parent child relationship table
2. Increment count to 1
3. Add into array list
4. Repeat step 1,2 and 3until count reaches to Z

List the descendents of person X for Z generations:

1. Find child of X in db using parent child relationship table
2. Increment count to 1
3. Add into array list
4. Recursice call to child of X
5. Repeat step 1,2,3 and 4 until count reaches to Z

Test case:

- when we add relationship in dissolution in which relationship is already exist in partnering.
- Add dynamic attributes in personattributes and file attributes
- In findBiologicalFamilyMedia, get and sort record which has same date and name with or without minor modification
- Pass all kind of dates in function parameter
    - YYYY
    - YYYY-MM-DD
    - YYYY-DD
    - YYYY-MM
    - YYYY/MM
    - DD-MM-YY
    - DD-MM-YYYY

    And many more possible solution
- Get ancestor and descendants of degree 1 and 0 in person relationship within database.
- Pass some null or empty string in list while using parameter in some method.
- Find relationship path while structure has more than 1 root node and from one node, we can go to the multiple direction in database structure.
- Pass some dynamic attributes for person and files and then change it with other value in same key attributes