DigiFam

Bringing Families Together

Abstract

As families grow bigger, houses grow smaller. People don't live with their families and feel disconnected. Enter: DigiFam. DigiFam aims to create a virtual image of your family. It enables you to digitise your family and connect with them. Plan events with your cousins, go to that flea market with your aunt, and how about a surprise anniversary celebration for your mum and dad? DigiFam allows you to create events, and invite family members to them. It also suggests family members to invite, and gives you reminders when a loved one has a special occasion approaching, so that you can plan something special. After the event is over, family members can upload the photos on a shared folder that is automatically created and shared only with the invitees. That's not all. DigiFam also offers a feature to share subscriptions within your family. Netflix won't seem too costly now. Just invite your cousins to pitch in and buy a group subscription. This also extends to family insurances, and other shared resources. It will also suggest events based on interests, and give you targeted suggestions. So if you and your cousin are both into horror movies, the app will suggest you go see Grudge 3 with your cousin. DigiFam will enable you to conveniently connect with your family, turning your phone into your home.

Team Name: DigiFam
Team No. : 65
Team Members: Pankil Kalra, Mohd Naki, Prashant, Smera Goel, Ishan Kapur
Application/domain: Family Database for invitation management and resource sharing.
Stakeholders (minimum 4): Family members, event organisers, subscription-based companies, medical-service providers and insurance companies.

Bonus:

1. For clerks, companies to give access to basic relational queries which they can create using option menu:
    a. choose the table
    b. choose attributes to print
    c. condition on attribute
        i. choose operator
        ii. enter values for the respective operator
    d. run query

    In this you are given different options and you have to select those options and the implemented script will create, run and retrieve results of this query for you. This helps the clerk in executing accurate queries very quickly.
2. Lookup if your loved ones have been diagnosed with COVID-19
3. Create a family tree!

Database Schema
ER Model

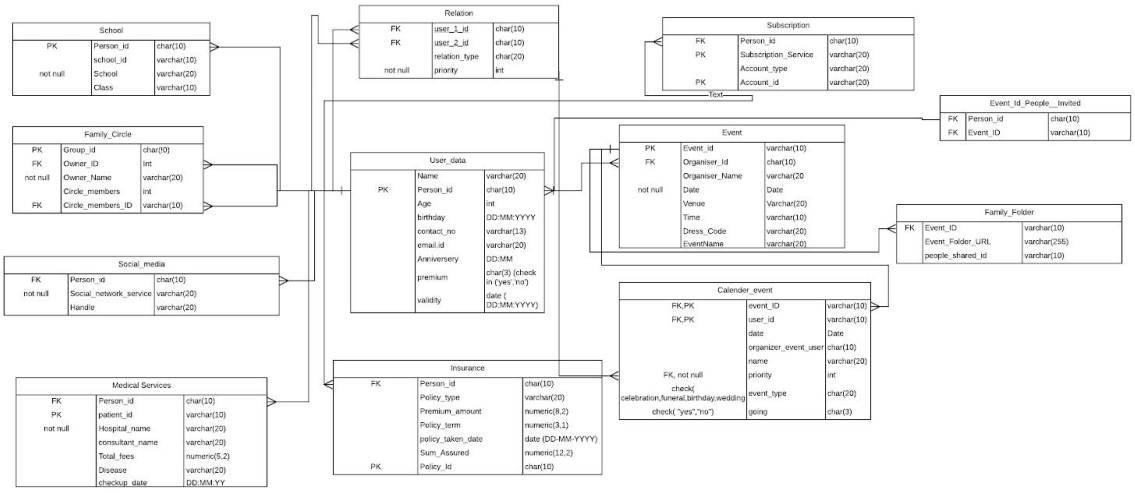
Description:
A networking based application where users can create their own private space by adding their closest contacts. Our project focuses on delivering ease for family and friends to manage any close events like wedding, birthday, anniversary party, other family functions and sharing photos of these events afterwards. Our application also provides resource sharing services, such as sharing subscriptions for netflix, medical insurances etc. and provides targeted suggestions according to the user profile.

Individual Role
Smera Goel - **Schema, logical analysis, relations, UI, documentation, python graphics**
Pankil Kalra - **Schema, java/sql integration, ER model, python/sql integration, database refining**
Ishan kapur - **Schema, java/sql integration,creation of networks, ER model, python/sql integration**
Prashant - **Schema Design, java/sql Integration, indexing in database, ER model**
Mohd Naki - **Schema Design,Data Manipulation,Database Creation**

Stakeholder	**Question**
Family Members	1. Contacts of 'n' closest family members 2. Share photos with all the family members who attended a particular family event 3. Invitation to relatives for events/family functions/parties etc. 4. Ask family members to pool in for different subscriptions such as Netflix, gym membership etc. 5. Add different family members from Facebook/through contacts etc.
Subscription-Based Companies	1. Email and contact details of people who the main user shared. 2. Get usability assigned by the main user to each of the contacts to limit (if any) 3. Get details of the user's credit/ debit card. 4. Get details of the user whose subscription has expired.

	5. Get the account id, and using it get the usage stats for each member of the family.
Event Management Companies	1. Get details of people invited and do bookings for them (train, hotels, entry tickets, restaurant reservation etc.) 2. Get age demographic/food preferences etc of people invited to an event 3. Get the contact details of all people invited and send confirmation emails/calls. 4. If children are to be invited for any event, get the contact details of their parents and contact them asking for their consent. 5. If there are gifts to be given at the end of the event, plan them using the age and gender demographics of the people invited to the event. e.g. a child will get a toy whereas an older male would get a tie.
Medical service providers and Insurance Companies	1. Manage and send requests for medical help such as blood donations etc. 2. Keep track of inheritable diseases. 3. Get last health checkup date and contact family members in case checkup is due. 4. Get the detail of their policyholders, what type of policy and their premium amounts, Sum assured amount. 5. Inform the relatives in case of accidents (major or minor according to the specific organization policy).

Database Schema



Indexes

- CREATE UNIQUE INDEX pidx
  ON User_data(person_id);

- CREATE UNIQUE INDEX polidx
  ON Insurance(Policy_id);

- CREATE INDEX priordx
  ON Relation(priority);

- CREATE INDEX schdx
  ON School(school);

- CREATE INDEX clsdx
  ON School(Class);

- CREATE UNIQUE INDEX patdx
  ON Medical_service(patient_id);

- CREATE INDEX deasdx
  ON Medical_service(Disease);

- CREATE UNIQUE INDEX gpdx
  ON Family_Circle(group_id);

- CREATE UNIQUE INDEX accdx
  ON Subscription(Account_id);

- CREATE INDEX servdx
  ON subscription(Subscription_Service);

- CREATE UNIQUE INDEX Evidx
  ON Event(Event_id);

- CREATE INDEX ogidx
  ON Event(Organiser_id);

- CREATE INDEX evitidx
  ON Calender_event(event_type);

- CREATE INDEX goidx
  ON Calender_event(going);

Queries:

- **Find details of all the user who share a Netflix account related to the user “39”**

  $\prod_{V.person\_id}(\sigma_{U.person\_id!=v.person\_id \wedge U.person\_id=”39” \wedge U.subscription\_service = “NETFLIX” \wedge U.account\_id = V.account\_id}(\rho_U(\text{subscription}) X \rho_V(\text{subscription}))$

  Select V.person_id from subscription U inner join subscription V where U.person_id!=V.person_id and U.person_id="39" and U.subscription_service="NETFLIX" and U.account_id = V.account_id;

- **Find all the events organized by the user ID “41” in the year 2020**

  $\sigma_{organizer\_id=”41” \wedge date>=’2020\text{-}01\text{-}01’ \wedge date<=’2020\text{-}12\text{-}31’}(\rho_T(\text{Event}))$

  Select * from event E where E.Organiser_id = "41" and date between '2020-01-01' and '2020-12-31';

- **Get a list of all students that are currently in 12th standard and go to ‘Basilicata University Potenza’ school.**

  $\pi_{U.person\_id,U.Name}(\sigma_{U.person\_id = S.person\_id \wedge S.Class = ‘12’ \wedge S.School = ‘Basilicata\ University\ Potenza’}(\rho_U(\text{User\_data}) \times \rho_S(\text{School})))$

  SELECT U.Person_id, U.Name
  FROM User_data as U, School as S
  WHERE U.person_id = S.person_id AND S.Class = "12" AND S.School = "Basilicata University Potenza";

- **Get list of all Event Folder URLs of all the events organised by 'Quinn Didsbury' in the year '2020'**

  $\prod_{v.event\_id,v.event\_folder\_URL}(\sigma_{u.event\_id =v.event\_id \wedge u.organizer\_name = Quinn\ Didsbury” \wedge u.date>’31\text{-}12\text{-}2019’ \wedge u.date< ‘01\text{-}01\text{-}2021’}(\rho_U(\textbf{event}) X \rho_V(\textbf{family\_folder}))$

  Select v.event_id,v.event_folder_URL from event u inner join family_folder v where u.event_id = v.event_id and u.organiser_name="Quinn Didsbury" and u.date > '2019-12-31' and u.date < '2021-01-01';

- **Get list of people diagnosed with disease 'Dengue' on '5th Dec 2019'**

$\pi_{U.person\_id,\ U.Name}(\sigma_{U.person\_id = M.person\_id \wedge M.Disease = \text{‘Dengue’’} \wedge M.checkup\_date = \text{‘2020-12-05’}}$
$(\rho_U(User\_data) \times \rho_M(Medical\ Service)))$

```
SELECT U.person_id, U.Name
FROM user_data as U, medical_sercives as M
WHERE U.person_id = M.person_id AND
M.Disease = "Dengue" AND M.checkup_date = '2020-12-05';
```

- **Get list of all the guest who are attend the 'Wedding' event in the year 2020**
  $\pi_{C.user\_id,\ U.Name}(\sigma_{C.event\_type = \text{‘Wedding’} \wedge C.going = 1 \wedge C.user\_id = U.person\_id \wedge YEAR(C.date) = \text{‘2020’}}$
  $(\rho_C(Calender\_evet) \times \rho_U(User\_data)))$

```
SELECT C.user_id,U.Name
FROM Calender_event as C, user_data as U
WHERE C.event_type = 'Wedding' AND C.going = '1' AND
C.user_id =U.person_id AND YEAR(C.date) = '2020';
```

- **Get the total amount spent on medical services for user_id =”5” in the years 2019-2020.**
  $\pi_{SUM(total\_fees)}(\sigma_{Person\_id = \text{‘5’} \wedge\ checkup\_date >= \text{‘01-01-2019’} \wedge checkup\_date <= \text{’31-12-2020’}}$ (Medical Services))

```
SELECT SUM(total_fees)
from medical_sercives
where Person_id = "5" and checkup_date between '2019-01-01' and
'2020-12-31';
```

- **Get info of all the events organised by the user with Person_id = ‘25’ ever on new years day.**
  $\pi_*(\sigma_{Organiser\_id = \text{‘25’’} \wedge\ month(date) = \text{‘01’} \wedge day(date) = \text{’01’}}$ (Event))

```
SELECT * from event where Organiser_Id = "25" and month(date) = '01' and
day(date) = '01';
```

- **Find all the name of organisers who organised an event on date 2nd april 2020**

  $\pi_{Organiser\_name}(\sigma_{date="2020-04-02"}(Event))$

  ```
  SELECT E.Organiser_name
  FROM Event as E
  WHERE E.date = '2020-04-02';
  ```

- **Get the list of diseases user was diagnosed with in the year "2020"**

  $\pi_{Disease}(\sigma_{year(date) = "2020'}(Medical\_Services))$

  ```
  SELECT distinct M.Disease
  FROM medical_sercives as M
  Where year(checkup_date) = '2020';
  ```

Embedded SQL

1) **Find 5 people closest to you based on the priority.**

```
stmt = conn.createStatement();
String sql11 = "SELECT v.name from relation r left join user_data v on
r.user_2_id = v.person_id where  r.user_1_id =\""+get_id()+"\" ORDER by
priority LIMIT 5;";
rs=stmt.executeQuery(sql11);
System.out.println("Answer for query :\n"+ sql11+"\n");
while(rs.next())
{
        System.out.println(rs.getString(1));
 }
```

2) **Get a list of all insurance that are expiring in the next 6 months or expired.**

```
stmt = conn.createStatement();
String sql12 = "SELECT * from insurance I where
DATE_ADD(I.policy_taken_date, INTERVAL I.policy_term
YEAR)<=DATE_ADD('"+get_date()+"',INTERVAL 6 MONTH) ;";
// get_date() returns the current date as string in SQL format
rs=stmt.executeQuery(sql12);
System.out.println("Answer for query :\n"+ sql12+"\n");
while(rs.next()) {
System.out.println(rs.getString(1)+"  "+rs.getString(2)+"  "+rs.getString(3)+"
"+rs.getString(4)+"  "+rs.getString(5)+"  "+rs.getString(6)+"  "+rs.getString(7));
}
```

3) **Find all the users with a birthday today and wish them a happy birthday.**

```
stmt = conn.createStatement();
String sql13 = "SELECT person_id from user_data U where day(U.birthday)
=day('"+ get_date() + "') AND month(U.birthday) = month('"+ get_date() +
"')";
  // get_date() returns the current date as string in SQL format
rs=stmt.executeQuery(sql13);
System.out.println("Answer for query :\n"+ sql13+"\n");
System.out.println("happy birthday");
while(rs.next()) {

   System.out.println(rs.getString(1));
}
```

4) **Delete subscription entry for service and accoun_id stored in variable as service and handle respectively**

```
String sql14 = "DELETE from Subscription WHERE Subscription_Service = '" +
service + "' AND Account_id = \"" + handle + "\"";

int r=stmt.executeUpdate(sql14);
System.out.println("Answer for query :\n"+ sql14+"\n");
if(r==1){

   System.out.println("success");
}
```

5) **Update contact number for person with his person_id and new contact info stored in the variables person_id and number**

```
String sql15 = "UPDATE User_Data set contact_no = \"" + (number) + "\"
WHERE Person_id = \"" + person_id+ "\"" ;

r=stmt.executeUpdate(sql15);
System.out.println("Answer for query :\n"+ sql15+"\n");
if(r==1){
   System.out.println("success");
}
```

6) **Add an list of person with person_id stored in a arraylist for a particular event**

```
for(int i=0;i<persons.size();i++)
{

        sql16 = "Insert into event_id_people_invited values(\"" +event_id+ "\",
                \"" +persons.get(i)+ "\")";
        r=stmt.executeUpdate(sql16);
        System.out.println("Answer for query :\n"+ sql16+"\n");
        if(r==1){
           System.out.println("success");
        }
 }
```

7) **Count the number of birthdays coming up in this month of people whose priority is 4 or higher (means 1,2,3,4).**

```
Statement stmt = con.createStatement();
String sql17 = "SELECT count(*) from Calender_Event WHERE event_type =
\"Birthday\" AND Priority <= 4 AND MONTH('"+get_date()+"')= MONTH(date);";

rs=stmt.executeQuery(sql17);
System.out.println("Answer for query :\n"+ sql17+"\n");
while(rs.next()) {
   System.out.println(rs.getString(1));
}
```

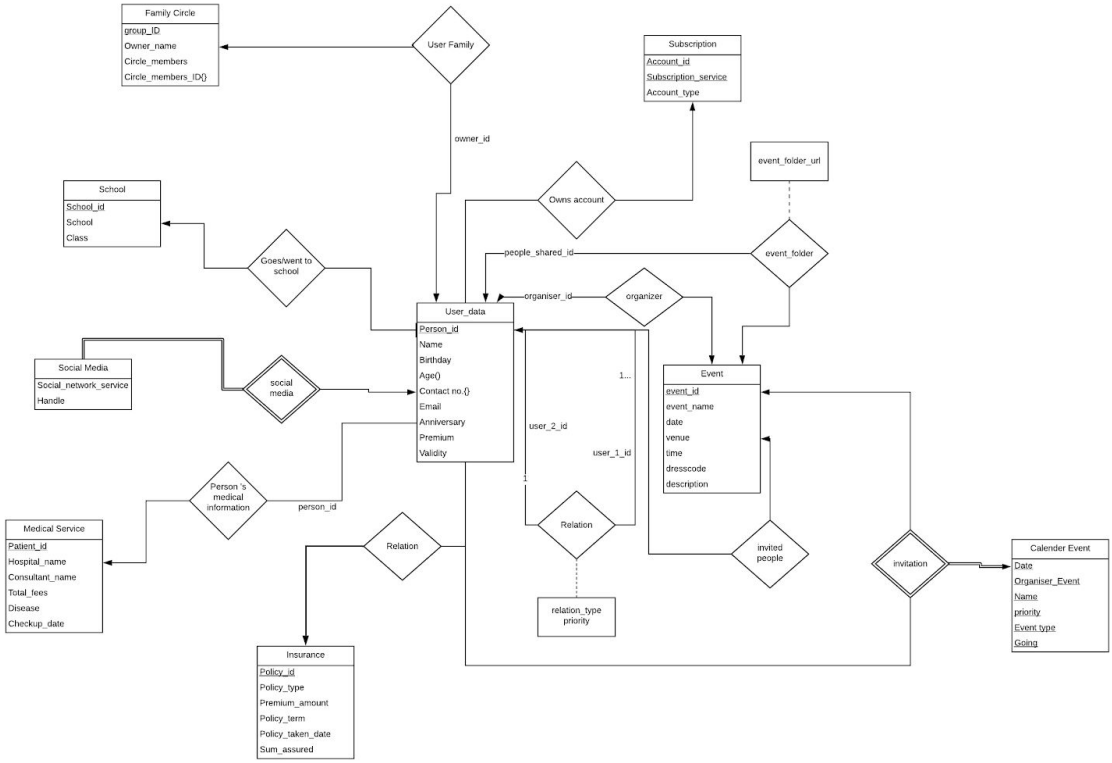
8) **Update the user account to premium with validity of 1 year from now for the user with id '5'**

```
Statement stmt = con.createStatement();

String sql18 = "UPDATE User_data set premium = \"1\" , validity =
DATE_ADD('"+get_date()+"', INTERVAL 1 YEAR) WHERE Person_id =\"5\"";

r=stmt.executeUpdate(sql18);
System.out.println("Answer for query :\n"+ sql18+"\n");
 if(r==1)
{
   System.out.println("success");
}
```

ER Model


Family Circle
group_ID
Owner_name
Circle_members
Circle_members_ID{}
User Family
owner_id
Subscription
Account_id
Subscription_service
Account_type
event_folder_url
School
School_id
School
Class
Owns account
Goes/went to school
people_shared_id
event_folder
organiser_id
organizer
User_data
Person_id
Name
Birthday
Age()
Contact no.{}
Email
Anniversary
Premium
Validity
Social Media
Social_network_service
Handle
social media
1...
user_2_id
user_1_id
1
Event
event_id
event_name
date
venue
time
dresscode
description
Person 's medical information
person_id
Relation
invited people
invitation
Calender Event
Date
Organiser_Event
Name
priority
Event type
Going
Medical Service
Patient_id
Hospital_name
Consultant_name
Total_fees
Disease
Checkup_date
Relation
relation_type priority
Insurance
Policy_id
Policy_type
Premium_amount
Policy_term
Policy_taken_date
Sum_assured