Tenzin Tashi

Prof. Hesham A Auda

CSc - 33600 Introduction to Database Systems

March 22, 2021

Assignment 2

Family Relations – II

Consider the Family Entity-Relationship (E-R) diagram[s] discussed in the class.

A. The definition of a brother-in-law in the Cambridge English Dictionary1 is:

a. The husband of a person’s sister,

b. The brother of a person’s wife or husband, or

c. The husband of the sister of a person’s wife or husband

Given the relation **Brothers** that has tuples of the form (*c*, *d*), where *c* is the brother of *d*, the relation **Sisters** that consists of tuples of the form (*g*, *h*), where *g* is the sister of *h*, the relation **Brother-Sister** which has tuples of the form (*e*, *f*), where *e* is the brother and *f* is the sister, and the relation **Husband-Wife** that has tuples of the form (*a*, *b*), where *a* is the husband and *b* is the wife.

a. Describe how you would define the relation **Brother-in-Law** whose tuples have the form (*x*, *y*) with *x* being the brother-in-law of *y*.

As per the Cambridge English Dictionary we can define x being the brother-in-law of y as follows:

a. The husband of a person’s sister,

⇨ First of all, we find the sister of the x person from the **BrotherSisters** relation. Here we will get sister of the person x (sisterId). Then using that Id we will find the husband of the sister (husbandId) from the **Spouses** relation. The husbandId will represent the brother-in-law of the person x.

- ⇨ Similarly, we can find the sister of the x person from the **Sisters** relation where sisterId will be the ID of the sister and childId will be the ID of the person x. Then, we can find the husband of the sisterId i.e., husbandId from the **Spouses** relation who will be the brother-in-law of the person x.

b. The brother of a person’s wife or husband

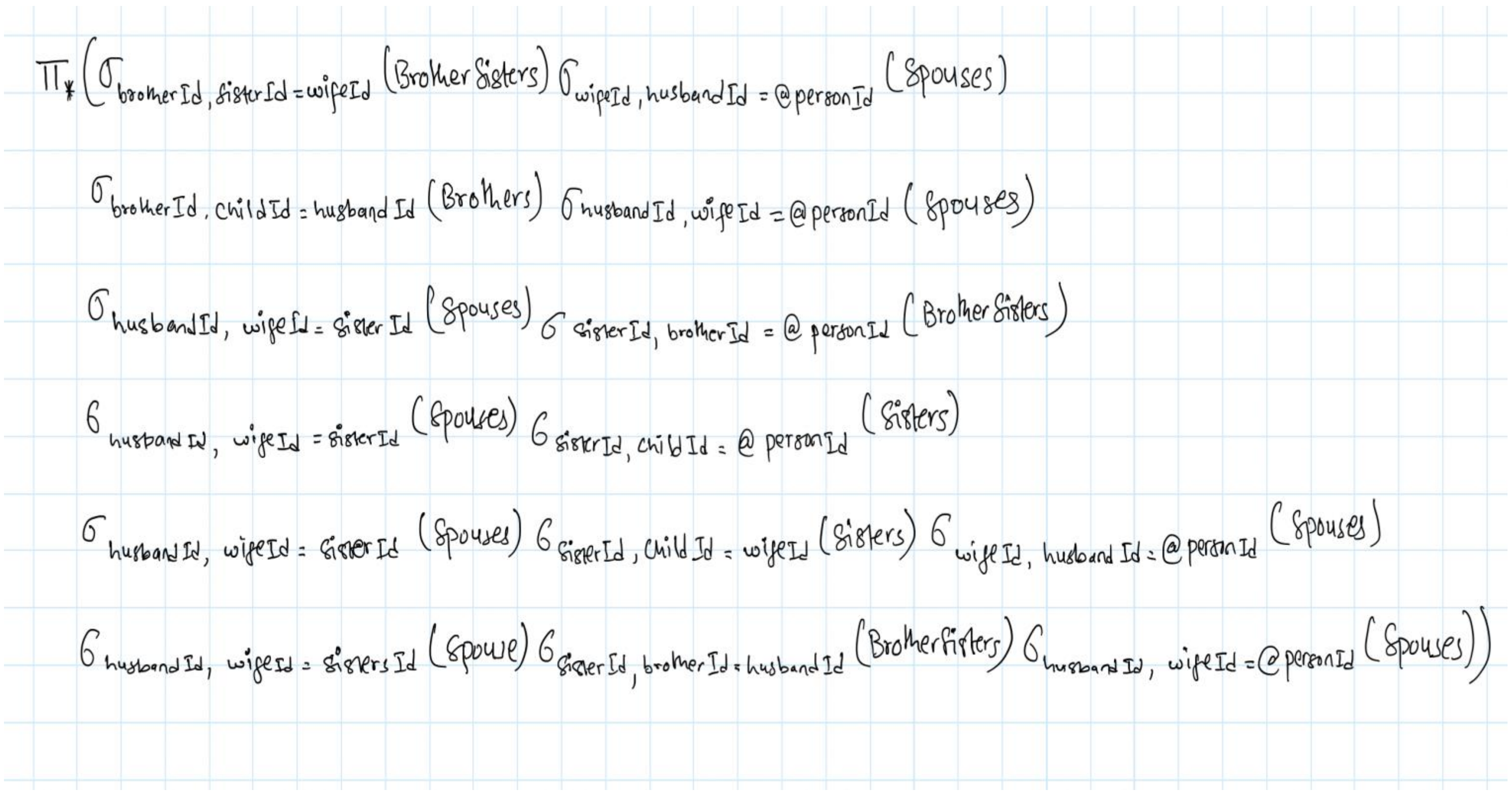
- ⇨ We can find the wifeId of the person x from the **Spouses** relation. Then find the brotherId using the wifeId found earlier from the **BrotherSisters** relation, who will be the brother-in-law of the person x.
- ⇨ Similarly, we can find the husbandId of the person x from the **Spouses** relation. Then find the brother of that person i.e., brotherId from the **Brothers** relation, which will represent the brother-in-law of the person x.

c. The husband of the sister of a person’s wife or husband

- ⇨ For this we need to first find the husbandId of the person x from the **Spouses** relation, then using this Id we will find the sisterId from the **BrotherSisters** relation, and finally find the husbandId of the sisterId from the **Spouses** relation, who will be the brother-in-law of the person x.
- ⇨ Similarly, we will find the wifeId of the person x from the **Spouses** relation, then using the wifeId we’ll find the sisterId from the **Sisters** relation, and finally find the husbandId of the sisterId from the **Spouses** relation who will represent the brother-in-law of the person x.

b. Give appropriate relational algebra and SQL expressions that return the relation **brother-in-Law**.

Relational Algebra:



$$\Pi_{*}(\sigma_{brotherId,\ sisterId = wifeId}(BrotherSisters)\ \sigma_{wifeId,\ husbandId = @personId}(Spouses)$$

$$\sigma_{brotherId,\ childId = husbandId}(Brothers)\ \sigma_{husbandId,\ wifeId = @personId}(Spouses)$$

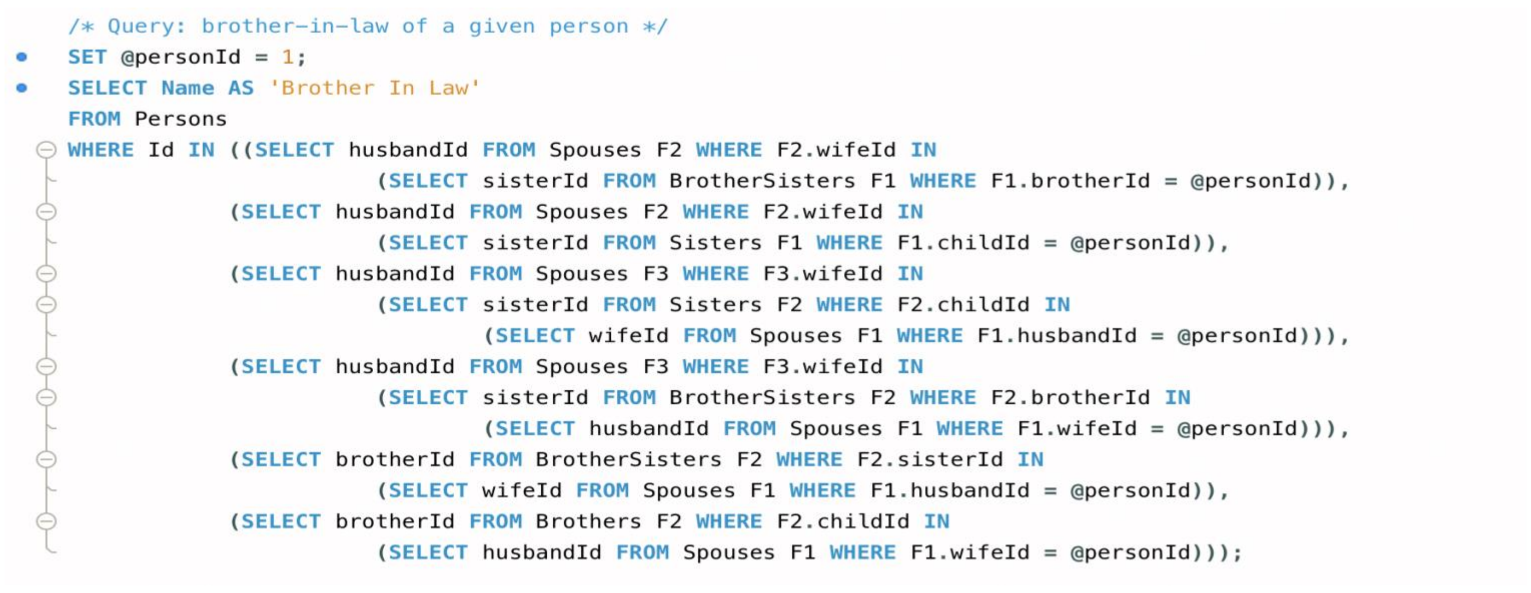
$$\sigma_{husbandId,\ wifeId = sisterId}(Spouses)\ \sigma_{sisterId,\ brotherId = @personId}(BrotherSisters)$$

$$\sigma_{husbandId,\ wifeId = sisterId}(Spouses)\ \sigma_{sisterId,\ childId = @personId}(Sisters)$$

$$\sigma_{husbandId,\ wifeId = sisterId}(Spouses)\ \sigma_{sisterId,\ childId = wifeId}(Sisters)\ \sigma_{wifeId,\ husbandId = @personId}(Spouses)$$

$$\sigma_{husbandId,\ wifeId = sistersId}(Spouse)\ \sigma_{sisterId,\ brotherId = husbandId}(BrotherSisters)\ \sigma_{husbandId,\ wifeId = @personId}(Spouses))$$

SQL Expressions:



```sql
/* Query: brother-in-law of a given person */
SET @personId = 1;
SELECT Name AS 'Brother In Law'
FROM Persons
WHERE Id IN ((SELECT husbandId FROM Spouses F2 WHERE F2.wifeId IN
                        (SELECT sisterId FROM BrotherSisters F1 WHERE F1.brotherId = @personId)),
            (SELECT husbandId FROM Spouses F2 WHERE F2.wifeId IN
                        (SELECT sisterId FROM Sisters F1 WHERE F1.childId = @personId)),
            (SELECT husbandId FROM Spouses F3 WHERE F3.wifeId IN
                        (SELECT sisterId FROM Sisters F2 WHERE F2.childId IN
                                (SELECT wifeId FROM Spouses F1 WHERE F1.husbandId = @personId))),
            (SELECT husbandId FROM Spouses F3 WHERE F3.wifeId IN
                        (SELECT sisterId FROM BrotherSisters F2 WHERE F2.brotherId IN
                                (SELECT husbandId FROM Spouses F1 WHERE F1.wifeId = @personId))),
            (SELECT brotherId FROM BrotherSisters F2 WHERE F2.sisterId IN
                        (SELECT wifeId FROM Spouses F1 WHERE F1.husbandId = @personId)),
            (SELECT brotherId FROM Brothers F2 WHERE F2.childId IN
                        (SELECT husbandId FROM Spouses F1 WHERE F1.wifeId = @personId)));
```

B. Design and implement a Java application that connects to the database. The application employs the following class inheritance hierarchy:

Person;
Child *is_a* **Person**;
GrandParent *is_a* **Person**;
BrotherInLaw *is_a* **Person**;
Nephew *is_a* **Child**.

Source Code:

Persons Class

```java
public  class Persons implements SQLFamily {
	protected int Id;
	protected String Name;
	protected String Sex;

	ConnectSQLServer JavaToSQL = new ConnectSQLServer();
	Connection conn = JavaToSQL.SQLConnection();

  /**
   * default constructor
   */
	public Persons(){
		this.Name = null;
		this.Sex = null;
	}

  /**
   * overloaded constructor
   * @param personid
   * @param personName
   * @param personSex
   */
	public Persons(int personid, String personName, String personSex){
		this.Id = personid;
		this.Name = personName;
		this.Sex = personSex;
	}

  /**
   * Setter for the Id field
   * @param Id the Id to set
   */
	public void setId(int Id) {
		this.Id = Id;
	}
```

```
/**
 * Setter for the Name field
 * @param Name the Name to set
 */
    public void setName(String Name) {
        this.Name = Name;
    }

/**
 * Setter for the Sex field
 * @param sex the Sex to set
 */
    public void setSex(String sex) {
        this.Sex = sex;
    }

/**
 * insert person object into the Person Table in the database
 */
    public void setPerson(){
        try {
            if (this.conn != null && this.Name != null && this.Sex != null) {
                String query = " insert into Persons (Id, Name, Sex)"
                + " values (?, ?, ?)";
                // create the mysql insert preparedstatement
                PreparedStatement preparedStmt =
conn.prepareStatement(query);
                preparedStmt.setInt (1, this.Id);
                preparedStmt.setString(2, this.Name);
                preparedStmt.setString(3, this.Sex);
                // execute the preparedstatement
                preparedStmt.execute();
                System.out.println("New Person Added to Database Successfully");
            }else if(this.Name == null || this.Sex == null){
                System.out.println("Error!!!\nAll fields (Id, Name, Sex) are required");
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

/**
 * Find a person info using Id
 * @param personId
 */
    public void getPerson(int personId){
        try {
             if (conn != null) {
                String sql = "SELECT * FROM Persons WHERE Id=?";
                PreparedStatement statement = conn.prepareStatement(sql);
                statement.setString(1, Integer.toString(personId));
                ResultSet result = statement.executeQuery();
                while (result.next()){
                    int pid = result.getInt(1);
                    String name = result.getString(2);
                    String sex = result.getString(3);

                    String output = "Data Found!!!\nId: %s\nName: %s \nSex: %s\n";
                    System.out.println(String.format(output,
Integer.toString(pid), name, sex));
                }
            }


        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

 /**
  *
  * @return formated String for a Persons Table
  */
    @Override
    public String toString(){
        String output = "";
        try{
            // 1. Display the legend
            System.out.println(String.format("%-5s \t %-15s \t %.5s\n", "Id",
"Name", "Sex"));
            // 2. Create a statement
            Statement myStmt = conn.createStatement();
            // 3. Execute SQL query
            ResultSet myRs = myStmt.executeQuery("select * from Persons");
            // 4. Process the result set
            while (myRs.next()) {
                output += String.format("%-5d \t %-15s \t %.5s\n",
myRs.getInt("Id"), myRs.getString("Name"), myRs.getString("Sex"));

            }
        }catch(SQLException ex){
            ex.printStackTrace();
        }
        return output;
    }

 /**
  * Method to find a record on a database using person Id
  * @param personId
  */
```

```
    @Override
    public void Select(int personId) {
        try {
            if (conn != null) {
                String sql = "SELECT * FROM Persons WHERE Id=?";

                PreparedStatement statement = conn.prepareStatement(sql);
                statement.setString(1, Integer.toString(personId));
                ResultSet result = statement.executeQuery();
                while (result.next()){
                    String pid = result.getString(1);
                    String name = result.getString(2);
                    String sex = result.getString(3);
                    System.out.println(String.format("%-5d \t %-15s \t %.5s\n",
pid, name, sex));
                }
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

  /**
   * Method to update a record on a database
   * @param personId
   * @param Name
   * @param Sex
   */
    @Override
    public void Update(int personId, String Name, String Sex) {
        try {
        if (conn != null) {
            String sql = "UPDATE Persons SET Name=?, Sex=? WHERE Id=?";

            PreparedStatement statement = conn.prepareStatement(sql);
            statement.setString(1, Name);
            statement.setString(2, Sex);
            statement.setString(3, Integer.toString(personId));

            int rowsUpdated = statement.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("An existing user was updated successfully!");
            }
        }
        }catch (SQLException ex) {
            ex.printStackTrace();
        } finally {
                try {
                        if (conn != null && !conn.isClosed()) {
                            //conn.close();
                        }
                    }
                catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

  /**
   * Method to delete a record from database
   * @param personId
   */
     @Override
     public void Delete(int personId) {
         try {
         if (conn != null) {
             String sql = "DELETE FROM Persons WHERE Id=?";
             PreparedStatement statement = conn.prepareStatement(sql);
             statement.setString(1, Integer.toString(personId));

             int rowsDeleted = statement.executeUpdate();
             if (rowsDeleted > 0) {
                 System.out.println("A user was deleted successfully!");
             }
         }
         } catch (SQLException ex) {
             ex.printStackTrace();
         } finally {
                 try {
                         if (conn != null && !conn.isClosed()) {
                             //conn.close();
                         }
                     }
                 catch (SQLException ex) {
                 ex.printStackTrace();
             }
         }
     }
}
```

Child Class

```
public  class Child extends Persons{

  /**
   * Query to find the child of a given couple
   * @param spouse1Id
   * @param spouse2Id
   * @return query of child
   */
      public ResultSet executeQuery(int spouse1Id, int spouse2Id){
            try {
                  if (conn != null) {
                    Statement statement = conn.createStatement();
                    String query = "SELECT ID, Name AS 'Child', sex\n"+
                                   "FROM Persons\n"+
                                   "WHERE Id IN  (SELECT personId\n"+
                                   "From Family\n"+
                         "WHERE fatherId ="+spouse1Id+" AND motherId ="+spouse2Id+
                                                  " OR "
                +"fatherId =" + spouse2Id + " AND motherId =" + spouse1Id + ")";
                     ResultSet result = statement.executeQuery(query);
                     return result;
                  }
            } catch (SQLException ex) {
                  ex.printStackTrace();
            }
            return null;
      }

  /**
   * Method to print the child query
   * @param result
   */
      public void print(ResultSet result)
   {
            // print children of a given couple
            try {
                  if (result != null) {
                        System.out.println(String.format("%-15s \t %.5s\n", "Child",
"Sex"));
                        while (result.next()) {
                              System.out.println(String.format("%-15s \t %.5s",
result.getString(2), result.getString(3)));
                        }
                  }
            } catch (SQLException ex) {
                  ex.printStackTrace();
            }
      }
}
```

GrandParent Class

```
public class GrandParent extends Persons{
 /**
  * Query to find the grandparents of a given person
  * @param pId
  * @return
  */
    public ResultSet executeQuery(int pId) {
     try {
         if (conn != null) {
         String query = "SELECT Name As 'Child',\n"+
                   "(SELECT Name FROM Persons WHERE Id IN\n"+
                   "(SELECT fatherId FROM Family F2 WHERE F2.personId IN\n"+
                   "(SELECT fatherId FROM Family F1 WHERE F1.personId = "+pId+" ))),\n"+
                   "(SELECT Name FROM Persons WHERE Id IN\n"+
                   "(SELECT motherId FROM Family F2 WHERE F2.personId IN\n"+
                   "(SELECT fatherId FROM Family F1 WHERE F1.personId = "+pId+" ))),\n"+
                   "(SELECT Name FROM Persons WHERE Id IN\n"+
                   "(SELECT fatherId FROM Family F2 WHERE F2.personId IN\n"+
                   "(SELECT motherId FROM Family F1 WHERE F1.personId = "+pId+" ))),\n"+
                   "(SELECT Name FROM Persons WHERE Id IN\n"+
                   "(SELECT motherId FROM Family F2 WHERE F2.personId IN\n"+
                   "(SELECT motherId FROM Family F1 WHERE F1.personId = "+pId+" )))\n"+
                   "FROM Persons\n"+
                   "WHERE Id = "+pId;
          PreparedStatement statement = conn.prepareStatement(query);
          ResultSet result = statement.executeQuery(query);
          return result; }
         } catch (SQLException ex) {
            ex.printStackTrace();}
         return null; }
 /**
  * Method to print the query of grandparents
  * @param result
  */
public void print(ResultSet result){
try {
if (result != null) {
System.out.println(String.format("%-20s \t %-20s \t %-20s \t %-20s \t %-20s ",
"Child", "Paternal Grandfather", "Paternal Grandmother", "Maternal Grandfather", "Maternal
Grandmother" ));
while (result.next()) {
 System.out.println(String.format("%-20s \t %-20s \t %-20s \t %-20s \t %-20s ",
result.getString(1),result.getString(2),result.getString(3),result.getStri
ng(4),result.getString(5))); }}
         } catch (SQLException ex) {
            ex.printStackTrace();}
    }
}
```

Brother In Law Class

```
public class BrotherInLaw extends Persons{
 /**
  * Query to find the brother in law of a given person
  * @param personId
  * @return the query of brother in law
  */
public ResultSet executeQuery(int personId) {
   try {
      if (conn != null) {
      String query = "SELECT Name AS 'Brother In Law'\n" +
              "FROM Persons\n" +
              "WHERE Id IN ((SELECT husbandId FROM Spouses F2 WHERE F2.wifeId IN\n" +
              "(SELECT sisterId FROM BrotherSisters F1 WHERE F1.brotherId = "+personId+")),\n" +
              "(SELECT husbandId FROM Spouses F2 WHERE F2.wifeId IN\n" +
              "(SELECT sisterId FROM Sisters F1 WHERE F1.childId = "+personId+")),\n" +
              "(SELECT husbandId FROM Spouses F3 WHERE F3.wifeId IN\n" +
              "(SELECT sisterId FROM Sisters F2 WHERE F2.childId IN\n" +
              "(SELECT wifeId FROM Spouses F1 WHERE F1.husbandId = "+personId+"))),\n" +
              "(SELECT husbandId FROM Spouses F3 WHERE F3.wifeId IN\n" +
              "(SELECT sisterId FROM BrotherSisters F2 WHERE F2.brotherId IN\n" +
              "(SELECT husbandId FROM Spouses F1 WHERE F1.wifeId = "+personId+"))),\n" +
              "(SELECT brotherId FROM BrotherSisters F2 WHERE F2.sisterId IN\n" +
              "(SELECT wifeId FROM Spouses F1 WHERE F1.husbandId = "+personId+")),\n" +
              "(SELECT brotherId FROM Brothers F2 WHERE F2.childId IN\n" +
              "(SELECT husbandId FROM Spouses F1 WHERE F1.wifeId = "+personId+")))";
       PreparedStatement statement = conn.prepareStatement(query);
       ResultSet result = statement.executeQuery(query);
       return result;}
          } catch (SQLException ex) {
             ex.printStackTrace();}
          return null; }
 /**
  * Method to print the query of brother in law
  * @param result
  */
    public void print(ResultSet result) {
          // print brother in law of a given person
          try {
           if (result != null) {
              System.out.println(String.format("%-20s ", "Brother In Law"));
              while (result.next()) {
              System.out.println(String.format("%-15s", result.getString(1)));
                   }
              }
          } catch (SQLException ex) {
             ex.printStackTrace();
          }
    }
}
```

Nephew Class

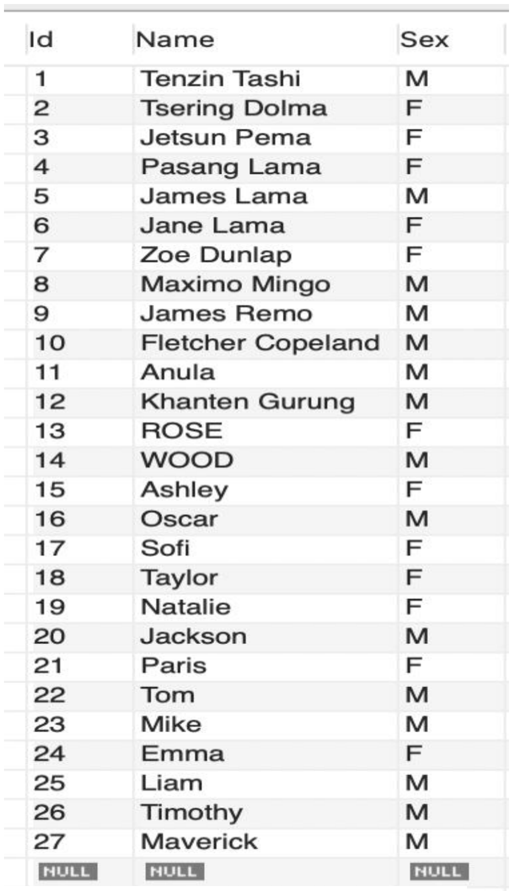
```
public class Nephew extends Child{
 /**
  *
  * @param personId
  * @return
  */
 public ResultSet executeQuery(int personId){
  try {
   if (conn != null) {
    String sql = "SELECT Id, Name AS 'Nephew', Sex\n" +
    "FROM Persons\n" +
    "WHERE Id IN ((SELECT personId\n" +
    "FROM Family F1\n" +
    "WHERE F1.fatherId IN (SELECT brotherId \n" +
    "FROM Brothers F2 \n" +
    "WHERE F2.childId = "+personId+" and Sex = 'M'))\n" +
    "union\n" +
    "(SELECT personId \n" +
    "FROM Family F1 \n" +
    "WHERE F1.fatherId IN (SELECT brotherId \n" +
    "FROM BrotherSisters F2 \n" +
    "WHERE F2.sisterId = "+personId+" and Sex = 'M'))\n" +
    "union\n" +
    "(SELECT personId \n" +
    "FROM Family F1 \n" +
    "WHERE F1.motherId IN (SELECT sisterId \n" +
    "FROM BrotherSisters F2 \n" +
    "WHERE F2.brotherId = "+personId+" and Sex = 'M'))\n" +
    "union\n" +
    "(SELECT personId \n" +
    "FROM Family F1 \n" +
    "WHERE F1.motherId IN (SELECT sisterId \n" +
    "FROM Sisters F2 \n" +
    "WHERE F2.childId = "+personId+" and Sex = 'M'))\n" +
    "union\n" +
    "(SELECT personId\n" +
    "FROM Family F1\n" +
    "WHERE F1.fatherId In (SELECT husbandId FROM Spouses F2 WHERE F2.wifeId IN\n" +
    "(SELECT sisterId FROM BrotherSisters F1 WHERE F1.brotherId = "+personId+")))\n" +
    "union\n" +
    "(SELECT personId\n" +
    "FROM Family F1\n" +
    "WHERE F1.fatherId In (SELECT husbandId FROM Spouses F2 WHERE F2.wifeId IN\n" +
    "(SELECT sisterId FROM Sisters F1 WHERE F1.childId = "+personId+")))\n" +
    "union\n" +
    "(SELECT personId\n" +
    "FROM Family F1\n" +
        "WHERE F1.fatherId In (SELECT husbandId FROM Spouses F3 WHERE F3.wifeId IN\n"  +
        "(SELECT sisterId FROM Sisters F2 WHERE F2.childId IN\n"  +
        "(SELECT wifeId FROM Spouses F1 WHERE F1.husbandId = "+personId+"))))\n"  +
        "union\n"  +
        "(SELECT personId\n"  +
        "FROM Family F1\n"  +
        "WHERE F1.fatherId In (SELECT husbandId FROM Spouses F3 WHERE F3.wifeId IN\n"  +
        "(SELECT sisterId FROM BrotherSisters F2 WHERE F2.brotherId IN\n"  +
        "(SELECT husbandId FROM Spouses F1 WHERE F1.wifeId = "+personId+"))))\n"  +
        "union\n"  +
        "(SELECT personId\n"  +
        "FROM Family F1\n"  +
        "WHERE F1.fatherId In (SELECT brotherId FROM BrotherSisters F2 WHERE F2.sisterId IN\n"  +
        "(SELECT wifeId FROM Spouses F1 WHERE F1.husbandId = "+personId+")))\n"  +
        "union\n"  +
        "(SELECT personId\n"  +
        "FROM Family F1\n"  +
        "WHERE F1.fatherId In (SELECT brotherId FROM Brothers F2 WHERE F2.childId IN\n"  +
        "(SELECT husbandId FROM Spouses F1 WHERE F1.wifeId = "+personId+"))))";
    PreparedStatement statement = conn.prepareStatement(sql);
    ResultSet result = statement.executeQuery();
    return result;
   }
 } catch (SQLException ex) {
            ex.printStackTrace();
       }
         return null;
    }

    @Override
public void print(ResultSet m_result){
// print nephew of a given person
try {
 if (m_result != null) {
   System.out.println(String.format("%-15s \t %.5s\n", "Nephew", "Sex"));
   while (m_result.next()){
     System.out.println(String.format("%-15s \t %.5s\n", m_result.getString(2),
m_result.getString(3)));
                 } }
         }catch (SQLException ex) {
             ex.printStackTrace();
         } finally {
                 try {
                         if (m_result != null && !m_result.isClosed()) {
                             m_result.close();
                         }}
                 catch (SQLException ex) {
             }}
    }
}
```

Interface SQLFamily:

Interface **SQLFamily** is implemented by the superclass **Person**, and includes method signatures, static methods, and/or default methods appropriate for the execution of the DML statements and SQL queries associated with the database.

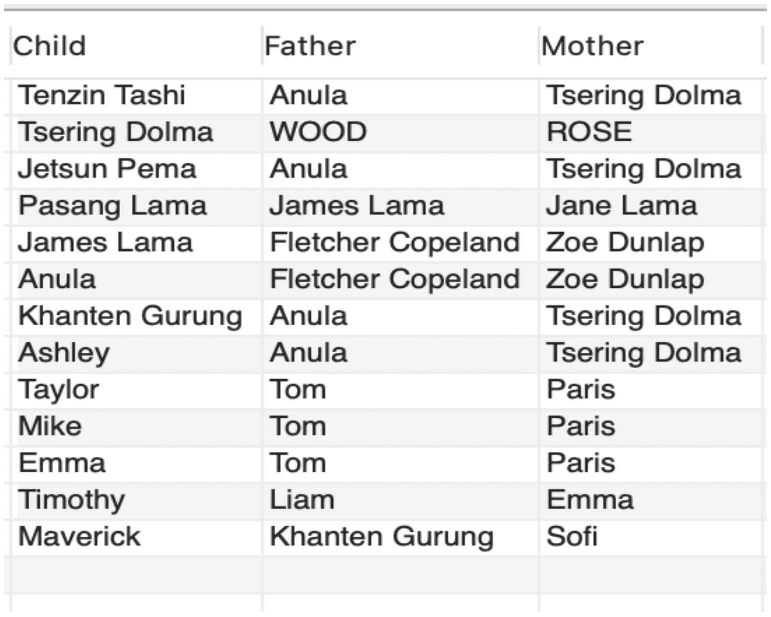
```
public  interface SQLFamily {
        public void Select (int personId);
        public void Update (int personId, String Name, String Sex);
        public void Delete (int personId);
}
```

Output



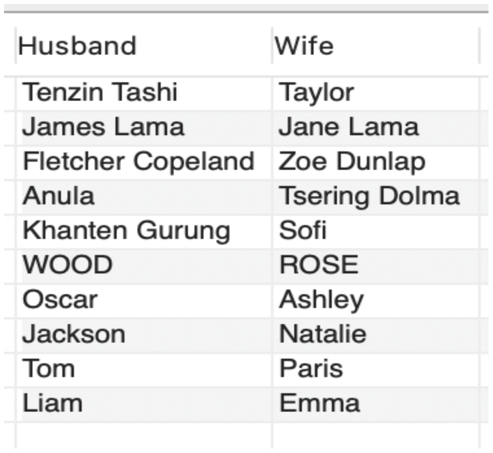
Id	Name	Sex
1	Tenzin Tashi	M
2	Tsering Dolma	F
3	Jetsun Pema	F
4	Pasang Lama	F
5	James Lama	M
6	Jane Lama	F
7	Zoe Dunlap	F
8	Maximo Mingo	M
9	James Remo	M
10	Fletcher Copeland	M
11	Anula	M
12	Khanten Gurung	M
13	ROSE	F
14	WOOD	M
15	Ashley	F
16	Oscar	M
17	Sofi	F
18	Taylor	F
19	Natalie	F
20	Jackson	M
21	Paris	F
22	Tom	M
23	Mike	M
24	Emma	F
25	Liam	M
26	Timothy	M
27	Maverick	M
NULL	NULL	NULL

Figure 1 Persons Table



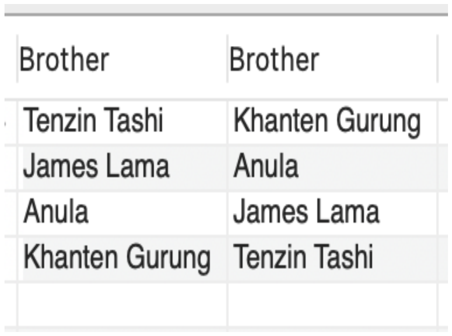
Child	Father	Mother
Tenzin Tashi	Anula	Tsering Dolma
Tsering Dolma	WOOD	ROSE
Jetsun Pema	Anula	Tsering Dolma
Pasang Lama	James Lama	Jane Lama
James Lama	Fletcher Copeland	Zoe Dunlap
Anula	Fletcher Copeland	Zoe Dunlap
Khanten Gurung	Anula	Tsering Dolma
Ashley	Anula	Tsering Dolma
Taylor	Tom	Paris
Mike	Tom	Paris
Emma	Tom	Paris
Timothy	Liam	Emma
Maverick	Khanten Gurung	Sofi

Figure 2 Family Table



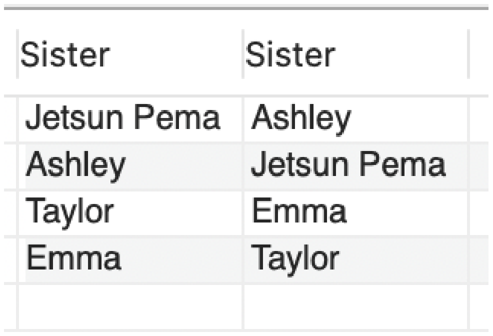
Husband	Wife
Tenzin Tashi	Taylor
James Lama	Jane Lama
Fletcher Copeland	Zoe Dunlap
Anula	Tsering Dolma
Khanten Gurung	Sofi
WOOD	ROSE
Oscar	Ashley
Jackson	Natalie
Tom	Paris
Liam	Emma

Figure 3 Spouses Table



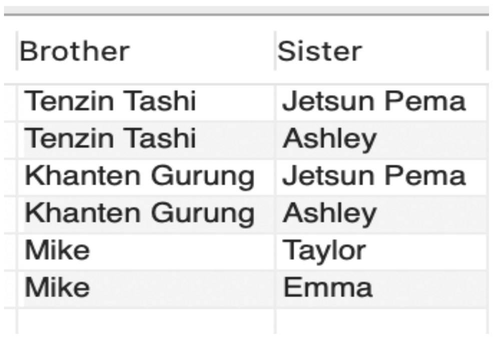
Brother	Brother
Tenzin Tashi	Khanten Gurung
James Lama	Anula
Anula	James Lama
Khanten Gurung	Tenzin Tashi

Figure 4 Brothers Table



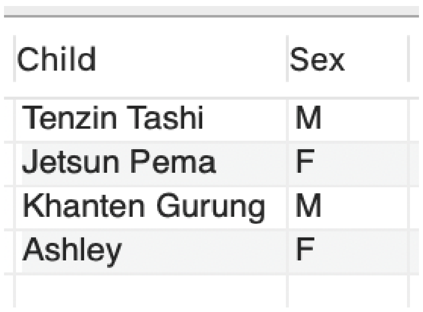
Sister	Sister
Jetsun Pema	Ashley
Ashley	Jetsun Pema
Taylor	Emma
Emma	Taylor

Figure 5 Sisters Table



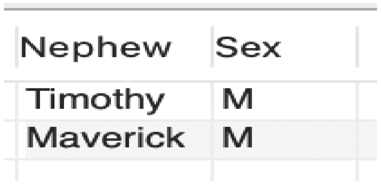
Brother	Sister
Tenzin Tashi	Jetsun Pema
Tenzin Tashi	Ashley
Khanten Gurung	Jetsun Pema
Khanten Gurung	Ashley
Mike	Taylor
Mike	Emma

Figure 6 BrotherSisters Table



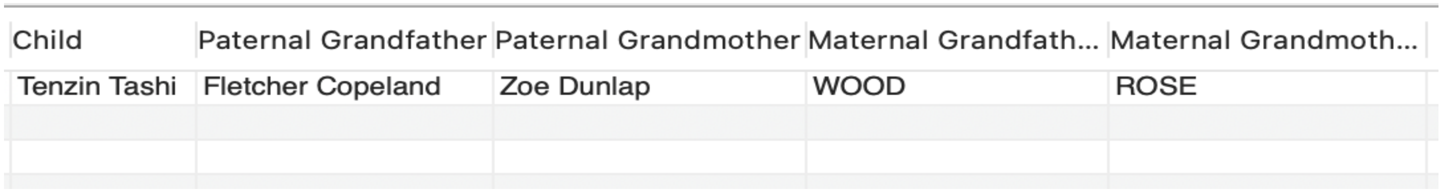
Child	Sex
Tenzin Tashi	M
Jetsun Pema	F
Khanten Gurung	M
Ashley	F

Figure 7 Child Query of a given Couple



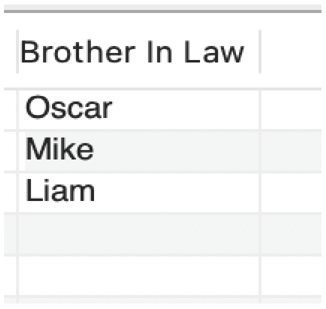
Nephew	Sex
Timothy	M
Maverick	M

Figure 8 Nephew Query of a given person



Child	Paternal Grandfather	Paternal Grandmother	Maternal Grandfath...	Maternal Grandmoth...
Tenzin Tashi	Fletcher Copeland	Zoe Dunlap	WOOD	ROSE

Figure 9 GrandParents Query



Brother In Law
Oscar
Mike
Liam

Figure 10 Brother-in-law Query

Clarify in the report how **Brothers**, **Sisters**, **Brother-Sister** and **Husband-Wife** relations are accounted for in your java application.

The **Brothers**, **Sisters**, **Brother-Sisters**, and **Husband-Wife** (Couples) relations are all accounted in the Brother-in-law class to find the brother-in-law of a given person.