Lester Lascano CMSC 127 (RS & M)

Roy Doñoz Assignment 2

Paired Kidney Exchange System

General Statement of the Problem

I. *What is the application all about?*

Paired Kidney Exchange System is a web application that allows hospitals to search for a kidney donor match for their patients if their kidney donor is incompatible. This will allow candidate patients with incompatible donor still able to receive a kidney transplant from a willing and matched donor. This project also intends to create a faster hospital-to-hospital transactions swiftly with shared records of patients and donors.

II. *Who are the users?*

The users of Paired Kidney Exchange System are the different hospitals that have patients that need a kidney transplant and in search of a donor because their recipient is not a match.

III. *What is the problem that it's trying to solve?*

This application is intended to solve incompatible kidney-matching of some patients with lesser donor matching time than the traditional manual matching. The application allows hospitals to log in to their respective accounts, add patients and their respective donors to the hospital, which will be added to the national waiting list database, delete patients and their respective donors to the hospital, which will be deleted in the national waiting list database, and update the information of patients and their respective donors to the hospital which will be updated in the national waiting list database.

Database Requirements

I. E/R Diagram of the database requirements

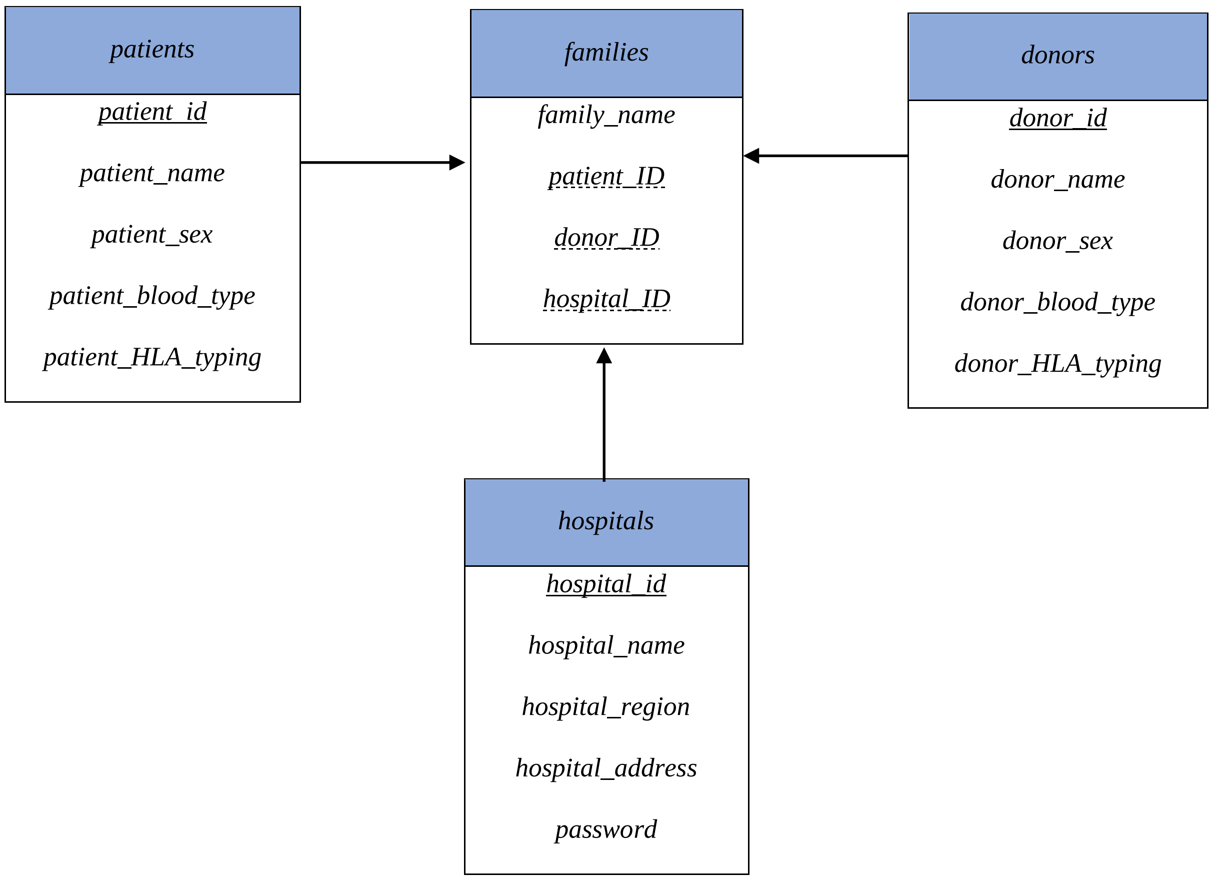


Figure 1. E/R Diagram of the paired_kidney_exchange database

II. Technical description of the idea behind the E/R diagram, relating to the database requirements

The paired kidney exchange database keeps a record of all the families, with its patients and donors. It keeps track which hospitals these families are admitted.

The *families* table contains all the families in the database, each represented by the attribute *family_name*. Each record in the *families* table is a pair of a patient and a donor that are searching for a compatible match in the database, and the hospital where the patient is admitted. The attribute patient_ID is a foreign key which references to the *patient_id* in the *patients* table. The attribute donor_ID is a foreign key which references to the *donor_id* in the *donor* table. The attribute hospital_ID is a foreign key which references to the *hospital_id* in the *hospital* table.

The *patients* table contains all the patients in the database, each uniquely represented by the primary key *patient_ID*. Each record in the *patients* table contains each patient's information with the patient's name represented by the attribute *patient_name*, patient's sex represented by the attribute *patient_sex*, patient's blood type represented by the attribute *patient_blood_type*, and patient's HLA represented by the attribute *patient_HLA_typing*.

The *donors* table contains all the donors in the database, each uniquely represented by the primary key *donor_ID*. Each record in the *donors* table contains each donor's information with the donor's name represented by the attribute *donor _name*, donor's sex represented by the attribute *donor_sex*, donor's blood type represented by the attribute *donor_blood_type*, and donor's HLA represented by the attribute *donor_HLA_typing*.

The *hospitals* table contains all the hospitals in the database, each uniquely represented by the primary key *hospital_ID*. Each record in the *hospitals* table contains each hospital's information with the hospital's name represented by the attribute *hospital _name*, the region where the hospital is located represented by the attribute *hospital_region*, the address of the represented by the attribute *hospital_address*, and the hospital's password represented by the attribute *password*. Each record in the *hospitals* table serves as an account in the system with the *hospital_ID* attribute being the account identifier and *password* attribute being the password.

III. Relations schema

families(family_name, patient_ID, donor_ID, hospital_ID) *patients(<u>patient_id</u>, patient_name, patient _sex, patient _blood_type, patient _HLA_typing)*

donors(<u>donor_id</u>, donor_name, donor_sex, donor_blood_type, donor_HLA_typing)

hospitals(<u>hospital_id</u>, hospital_region, hospital_address, password)

Implementation

I. *SQL Table Definition of the tables*

families table

This table contains the name of the families (family_name) and 3 foreign keys - patient_ID, donor_ID, and hospital_ID. The table serves as a glue that connect the 3 tables – patients table, donors table, and hospitals table.

patients table

This table contains the id of the patients (patient_id) which also serves as the primary key in this table. The table also contains the different information about the patients – patient_name, patient_sex, patient_blood_type, and patient_HLA_typing – that are used to search for a match with patient-donor pairs in the database.

donors table

This table contains the id of the donors (donor_id) which also serves as the primary key in this table. The table also contains the different information about the donors – donor_name, donor_sex, donor_blood_type, and donor_HLA_typing – that are used to search for a match with patient-donor pairs in the database.

hospitals table

This table contains the id of the hospitals (hospital_id) which also serves as the primary key in this table. Each hospital is assigned with a unique hospital_id and a password that are stored in this table. This table also contains other information about the hospital – hospital_name, hospital_region, and hospital_address

II. *Sample Data*

patients and donors table

- [name] The names from the patients and donors table were randomly generated from a Spanish name generator online.
- [sex, blood type] The sex, and blood type of patients and donors were randomly inputted by us.
- [HLA] The HLA of the patients and donors were randomly generated by an online password generator online.

hospitals table

The hospital info for the hospital table was taken from a pdf file from the DOH site.

Random name generator link :

https://blog.reedsy.com/character-name-generator/?fbclid=IwAR1rpBnG_4SOpH8PqWIQYg6goSCtXCpEXkqamRVntspDcXNMvq57n8Gnud0

Random password generator link :

https://randomwordgenerator.com/password.php?fbclid=IwAR0nx_vKS5Xfaw6C0uuwzFVd71jdwyEjKReDapDVBfgST25qP25pgt2N480

Hospital data link :

https://doh.gov.ph/sites/default/files/publications/DOH-Hospitals-Profile_0.pdf?fbclid=IwAR2BBsAZ4G2ZZz-FXaZm7km1CehgczQncRyo2VzFarGtMgIqnBjVawfI3SE

III. *List of sample queries that were used in the application and their explanation*

1. Query for Logging into the website

```
"SELECT *
 FROM hospitals
 WHERE hospitals.hospital_id=$hospital_ID";
```

This is a query that SELECTS ALL FROM the hospital table WHERE the hospital_id is equal to $hospital_ID. $hospital_ID is the ID input during login. This query is used to check if a hospital with the given ID exists in the database, check if the password is correct, and store the hospital details if the hospital exists and the password is correct.

2. Query for inserting a family into the database

2.1 Query for inserting in the families table

```
"INSERT INTO families
 VALUES (:name, :patient_ID, :donor_ID, :hospital_ID)";
```

2.2 Query for inserting in the patients table

```
"INSERT INTO patients (patient_name, patient_sex, patient_blood_type, patient_HLA_typing)
 VALUES (:patient_name, :patient_sex, :patient_blood_type, :patient_HLA_typing)";
```

2.3 Query for inserting in the donors table

```
"INSERT INTO donors (donor_name, donor_sex, donor_blood_type, donor_HLA_typing)
 VALUES (:donor_name, :donor_sex, :donor_blood_type, :donor_HLA_typing)";
```

These are queries that INSERTS the VALUES that the user provided on the form to add a family to families table, patients table, and donors table in the database.

3. Query for deleting a family in the database

```
"DELETE families, patients, donors
 FROM families INNER JOIN patients INNER JOIN donors
 ON families.donor_ID=donors.donor_id AND families.patient_ID=patients.patient_id
 WHERE families.family_name=:name";
```

This query INNER JOIN the 3 tables to select records WHERE families.donor_ID=donors.donor_id AND families.patient_ID=patients.patient_id, then DELETE the records FROM the families, patients, and donors table in the database WHERE families.family_name=name (name is the name of the family that the user want to delete).

4. Query for searching a family from the database

```
"SELECT family_name
 FROM families
 WHERE families.family_name LIKE '%$search%'
 AND families.hospital_ID=$hospital_ID
 ORDER BY family_name";
```

This query SELECT the family_name FROM the families table WHERE families.family_name LIKE '%search%' AND families.hospital_ID=$hospital_ID, then order the result by famile_name. This query is used when a user wants to search a family in the database, the query searches for records where the family_name has the '$search' string inside it and the hospital_id matches the hospital_ID of the logged in hospital.

5. Query for updating a family from the database

```
"UPDATE families INNER JOIN patients INNER JOIN donors
 ON families.donor_ID=donors.donor_id
 AND families.patient_ID=patients.patient_id
 SET families.family_name=:newFamilyName
   , patients.patient_name=:newPatientName
   , patients.patient_sex=:newPatientSex
   , patients.patient_blood_type=:newPatientBloodType
   , patients.patient_HLA_typing=:newPatientHLAtyping
   , donors.donor_name=:newDonorName
   , donors.donor_sex=:newDonorSex
   , donors.donor_blood_type=:newDonorBloodType
   , donors.donor_HLA_typing=:newDonorHLAtyping
 WHERE families.family_name=:name";
```

This query INNER JOIN the 3 tables to select records WHERE families.donor_ID=donors.donor_id AND families.patient_ID=patients.patient_id then updates/SET the records FROM the families, patients, and donors table in the database to the values that the user provided on the form to edit a family WHERE families.family_name=name (name is the name of the family that the user want to update).

6. Query for finding family matches from the database

```
"SELECT *
 FROM families INNER JOIN donors INNER JOIN patients INNER JOIN hospitals
 ON families.donor_ID=donors.donor_id
 AND families.patient_ID=patients.patient_id
 AND families.hospital_ID=hospitals.hospital_id
 WHERE donors.donor_blood_type='O'
 " . donorMatchingQuery($patientHLAtyping)
 . "AND (patients.patient_blood_type='O'
 OR patients.patient_blood_type='A'
 OR patients.patient_blood_type='B'
 OR patients.patient_blood_type='AB')
 " . patientMatchingQuery($donorHLAtyping);
```

This query basically selects all families that is a match to the main family WHERE the main family patient and donor recipient is O blood type.

We first INNER JOIN the 3 tables to select records WHERE families.donor_ID=donors.donor_id AND families.patient_ID=patients.patient_id AND families.hospital_ID=hospitals.hospital_id.

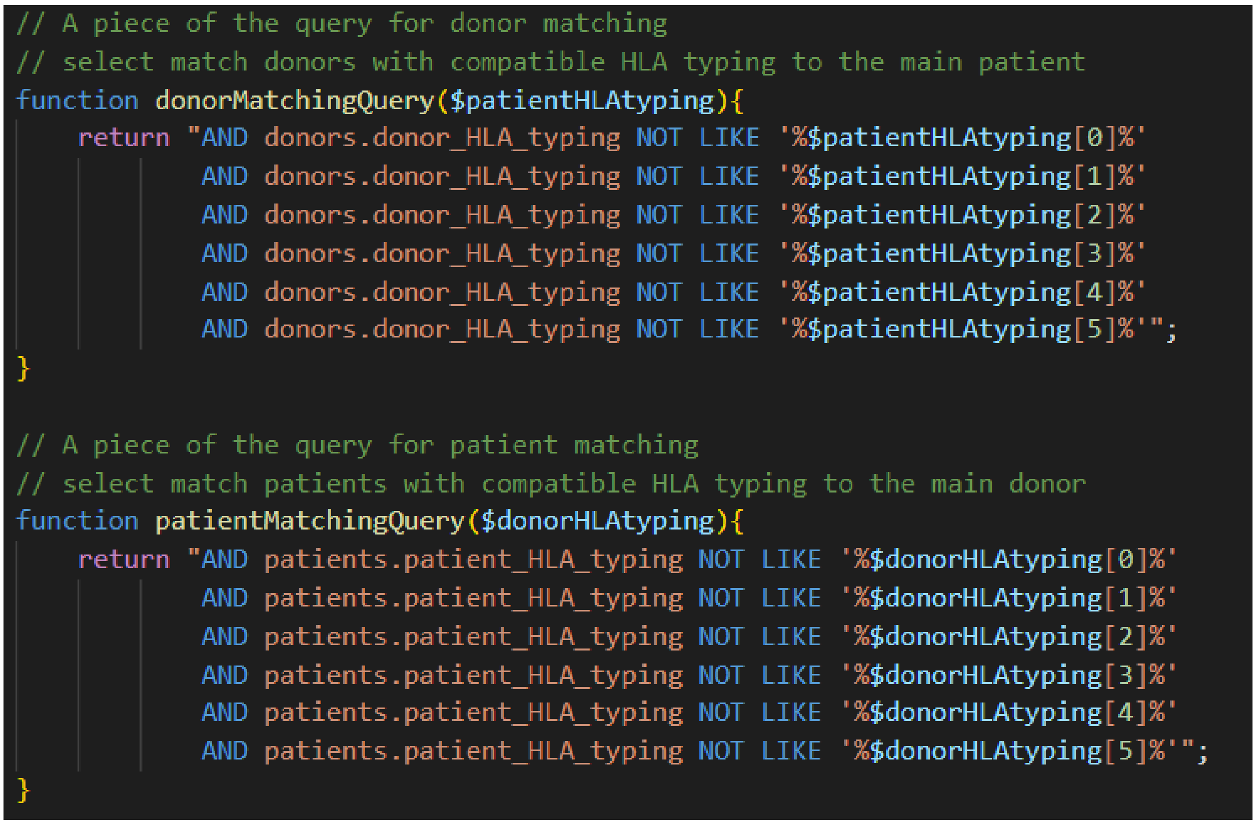
WHERE donors.donor_blood_type=’O’ clause is there because only O blood type donors are compatible with O blood type patients.

WHERE patients.patient_blood_type=’O’

AND patients.patient_blood_type=’A’

AND patients.patient_blood_type=’B’

AND patients.patient_blood_type='AB' clause is there because O, A, B, and AB blood type donors are compatible with O blood type patients.



```
// A piece of the query for donor matching
// select match donors with compatible HLA typing to the main patient
function donorMatchingQuery($patientHLAtyping){
    return "AND donors.donor_HLA_typing NOT LIKE '%$patientHLAtyping[0]%'
            AND donors.donor_HLA_typing NOT LIKE '%$patientHLAtyping[1]%'
            AND donors.donor_HLA_typing NOT LIKE '%$patientHLAtyping[2]%'
            AND donors.donor_HLA_typing NOT LIKE '%$patientHLAtyping[3]%'
            AND donors.donor_HLA_typing NOT LIKE '%$patientHLAtyping[4]%'
            AND donors.donor_HLA_typing NOT LIKE '%$patientHLAtyping[5]%'";
}

// A piece of the query for patient matching
// select match patients with compatible HLA typing to the main donor
function patientMatchingQuery($donorHLAtyping){
    return "AND patients.patient_HLA_typing NOT LIKE '%$donorHLAtyping[0]%'
            AND patients.patient_HLA_typing NOT LIKE '%$donorHLAtyping[1]%'
            AND patients.patient_HLA_typing NOT LIKE '%$donorHLAtyping[2]%'
            AND patients.patient_HLA_typing NOT LIKE '%$donorHLAtyping[3]%'
            AND patients.patient_HLA_typing NOT LIKE '%$donorHLAtyping[4]%'
            AND patients.patient_HLA_typing NOT LIKE '%$donorHLAtyping[5]%'";
}
```

The *donorMatchingQuery()* and *patientMatchingQuery()* functions are there to only select records of patient-donor pairings with no matching HLA.

IV. *A short discussion of the development of the front-end component (GUI) of the application*

The GUI for this web application *Paired Kidney Exchange System* was made easy and convenient for the users. The target users are different hospitals that have patients in need of a kidney transplant and in search of a donor because their recipient is not a match. This will allow faster tracking of available donors for the hospital's kidney patients.

- The login page was direct and simple by asking for the hospital's credentials such as the Hospital ID and its password.

- The Homepage contains the different functions of the app, structured in HTML and designed in CSS for better visual presentation.
- The buttons direct the user to its desired page of the application and displays the necessary information required and/or acquired by the system.

Overall, this web application was made user-friendly and at the same time making sure that the forms for acquiring and displaying data were not compromised.

Conclusion and Reflection

In conclusion, working using the three programming languages (HTML, CSS, and PHP) was indeed a challenge. Structuring the web application using HTML and designing it using CSS was lesser of a task than the efforts in creating the database, acquiring dummy information, and creating the queries in PHP. The knowledge gained and skills developed in creating this project will prove useful in more future projects to be created.

References

Character Name Generator. (n.d.). Retrieved from blog.reedsy.com: https://blog.reedsy.com/character-name-generator/?fbclid=IwAR1rpBnG_4SOpH8PqWIQYg6goSCtXCpEXkqamRVntspDcXNMvq57n8Gnud0

Department of Health - Health Facility Development Bureau. (2020, October 30). *DOH Hospital Profile.* Retrieved from doh.gov.ph: https://doh.gov.ph/sites/default/files/publications/DOH-Hospitals-Profile_0.pdf?fbclid=IwAR2BBsAZ4G2ZZz-FXaZm7km1CehgczQncRyo2VzFarGtMgIqnBjVawfI3SE

Random Password Generator. (n.d.). Retrieved from https://randomwordgenerator.com/: https://randomwordgenerator.com/password.php?fbclid=IwAR0nx_vKS5Xfaw6C0uuwzFVd71jdwyEjKReDapDVBfgST25qP25pgt2N480

UC Davis Health. (n.d.). *Living Kidney Donation Matching and Compatibility*. Retrieved from health.ucdavis.edu: https://health.ucdavis.edu/transplant/livingkidneydonation/matching-and-compatibility.html?fbclid=IwAR3RWhBYDxogpuIIrZOPldWEXA2RW3iF7lWqjXakH-xvaXGrwUtgeSe-KwE

University of Michigan Health. (n.d.). *Paired Kidney Exhange*. Retrieved from uofmhealth.org: ttps://www.uofmhealth.org/conditions-treatments/transplant/paired-kidney-exchange?fbclid=IwAR0gLY-w-2tJ8D_qvfGoQyEFTAo9dxuh81--atT1RKiUEDd4w_Ifkapi3IA#:~:text=A%20paired%20kidney%20exchange%2C%20also,live%20donor%20transplants%20would%20occur