Group project

BLOOD BANK MANAGEMENT SYSTEM

Gorelsky Roman, Dekkusheva Alina, Pronina Anna, Kulakov Denis

Description:

The designed database is aimed at storing the data about the donors with their medical conditions and prescribed medications, their donations to the specific blood bank, and the staff that was responsible for collecting the samples. The database is built on the PostgreSQL architecture.

Potential users:

- **Medical staff:** access to information about donors, their donations, medical conditions and medications.
- **Donors:** information about their own data and medical history.

Users and their needs:

Medical Staff:

- **Accessing medical data:** Viewing and updating information about donors, their medical conditions, medications and donor history.
- **Treatment Planning and Management:** Access to the data to make decisions about donor treatment and medication management.

Donors and Patients:

- **Accessing their own information:** Viewing their own medical history, donations' information, and survey results.
- **Record and manage data:** Update personal data, health status, medications, date of last donor action.

Functional requirements:

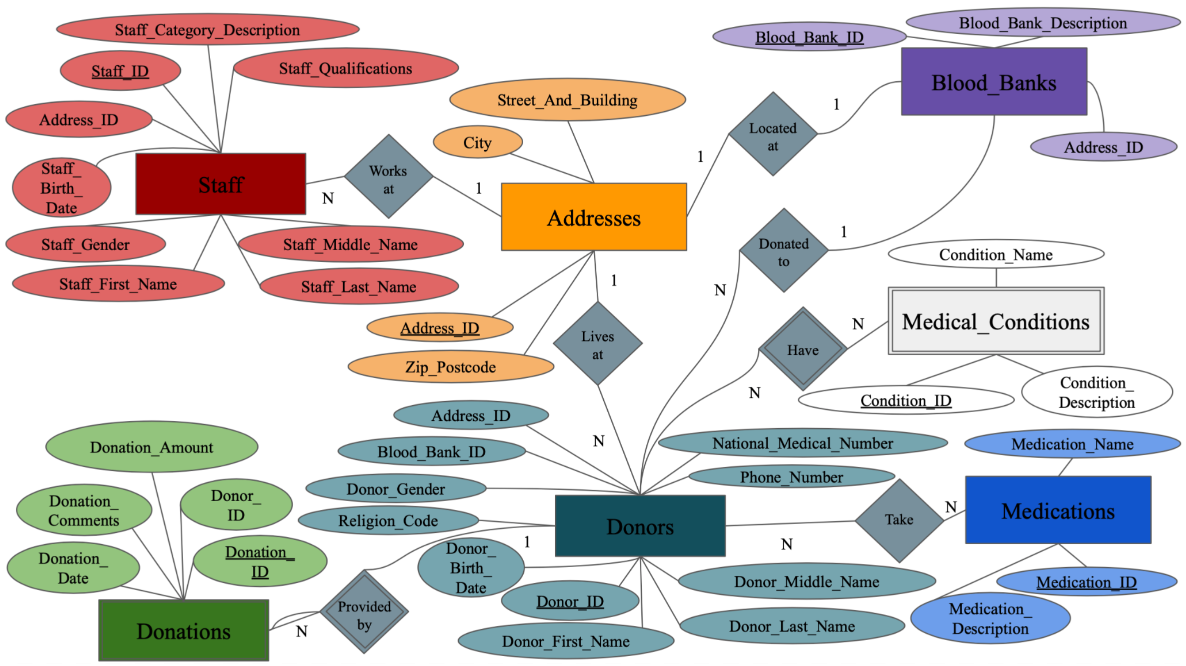
Medical Staff:

- Ability to view and modify the data about the donor.
- Ability to utilize the data to plan the treatment of a patient based on his or her medical conditions, medications, and blood samples.

Donors and Patients:

- Ability to view and modify personal information (telephone number, city of living, etc.) to provide relevant information about themselves.

ER Diagram:



Data Limits:

- All fields containing identifiers (e.g., ***staff_id***, ***donor_id***) must be unique.
- Fields relating to personal information (e.g., ***staff_first_name***, ***last_name***) must not contain empty values.
- Referential integrity must be maintained through foreign key constraints to ensure that all records in linked tables are correct and up-to-date.

Functional dependencies for each table:

Staff:

Staff_ID ⇒ Staff_First_Name, Staff_Middle_Name, Staff_Last_Name, Staff_Gender, Staff_Birth_Date, Staff_Qualifications, Staff_Category_Description

Addresses:

Address_ID ⇒ Street_And_Building, City, Zip_Postcode

Zip_Postcode ⇒ Street_And_Building, City

Blood_Banks:

Blood_Bank_ID ⇒ Blood_Bank_Description, Address_ID

Donations:

Donation_ID ⇒ Donor_ID, Donation_Date, Donation_Amount, Donation_Comments

Donors:

Donor_ID ⇒ Address_ID, Blood_Bank_ID, Donor_First_Name, Donor_Middle_Name, Donor_Last_Name, Donor_Gender, Religion_Code, Donor_Birth_Date, National_Medical_Number, Phone_Number

Medical_Conditions:

Condition_ID ⇒ Condition_Name, Condition_Description

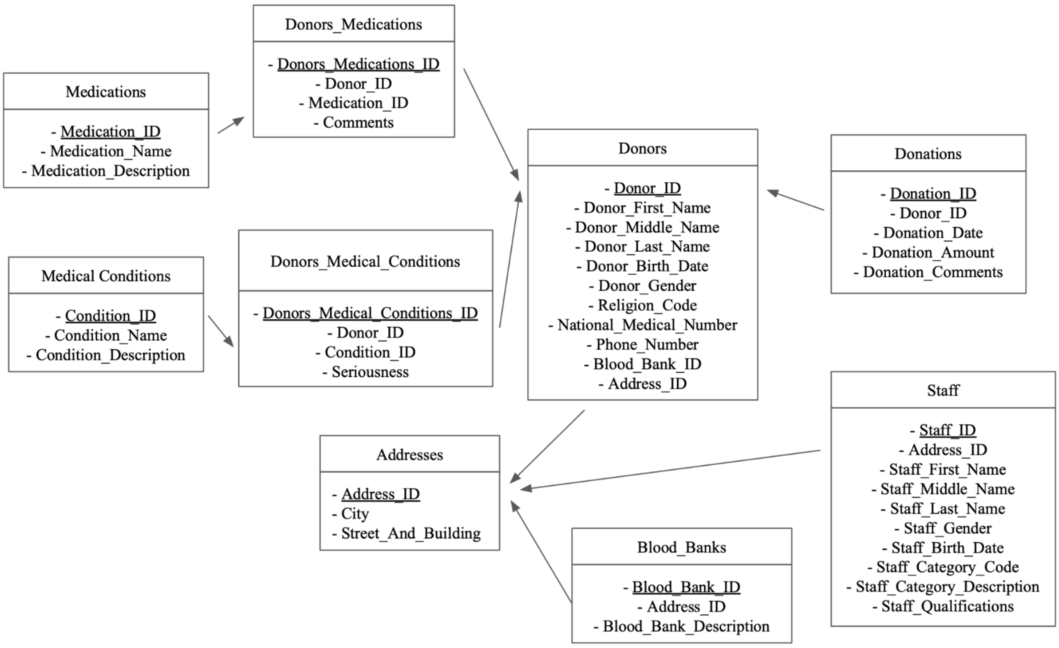
Medications:

Medication_ID ⇒ Medication_Name, Medication_Description

BCNF:

Without normalization such problems like data duplication can appear which leads to increased query completion complexity. By looking at the functional dependencies it can be concluded that the only point for which it doesn't hold is the table "Addresses". It was decided to remove this field from the table as it can be replaced by the other two fields without loss of generality. The relational diagram which corresponds to BCNF conditions is presented in the next chapter.

Relational diagram:



Code:

```
CREATE TABLE staff (

   staff_id serial PRIMARY KEY,

   address_id integer NOT NULL,

   blood_bank_id integer NOT NULL,

   gender_mfu char(1),

   staff_job_title text,

   staff_first_name text NOT NULL,

   staff_middle_name text,

   staff_last_name text NOT NULL,

   staff_qualifications text,

   staff_birth_date date,

   other_staff_details text

);
```

```sql
CREATE TABLE addresses (

   address_id serial PRIMARY KEY,

   Street_And_Building text,

   city text NOT NULL,

   zip_postcode text NOT NULL,

   state_province_county text,

   country text NOT NULL,

   other_address_details text

);


CREATE TABLE blood_banks (

   blood_bank_id serial PRIMARY KEY,

   address_id integer REFERENCES addresses(address_id),

   blood_bank_details text

);



CREATE TABLE donors (

   donor_id serial PRIMARY KEY,

   address_id integer REFERENCES addresses(address_id),

   blood_bank_id integer REFERENCES
blood_banks(blood_bank_id),

   national_medical_number text,

   gender_mfu char(1),

   date_of_birth date NOT NULL,
```

```
  first_name text NOT NULL,
  middle_name text,
  last_name text NOT NULL,
  phone_number text,
  cell_mobile_phone text,
  other_details text
);

CREATE TABLE medications (
  medication_code serial PRIMARY KEY,
  medication_name text NOT NULL,
  medication_description text
);

CREATE TABLE medical_conditions (
  condition_code serial PRIMARY KEY,
  condition_name text NOT NULL,
  condition_description text
);

CREATE TABLE donors_medications (
  donor_medications_id serial PRIMARY KEY,
  donor_id integer REFERENCES donors(donor_id),
  medication_code integer REFERENCES
medications(medication_code),
  comments text
```

```
);


CREATE TABLE donors_medical_conditions (
   donors_medical_conditions_id serial PRIMARY KEY,
   donor_id integer REFERENCES donors(donor_id),
   condition_code integer REFERENCES
medical_conditions(condition_code),
   seriousness text,
   donation_amount integer
);


CREATE TABLE donations (
   donation_id serial PRIMARY KEY,
   donor_id integer REFERENCES donors(donor_id),
   donation_date date NOT NULL,
   donation_details text,
   comments text
);
```

Data base requests:

Add a new donor (Create):

```
INSERT INTO staff (
   address_id,    blood_bank_id,    gender_mfu,
staff_job_title,    staff_first_name,    staff_middle_name,
   staff_last_name,    staff_qualifications,
staff_birth_date,    other_staff_details
```

```
) VALUES (
   1,     1,     'M',     'Nurse',     'John',     'A.',     'Doe',
'RN',     '1980-05-15',     'No allergies'
);
```

Record a new donation:

```
INSERT INTO donations (
   donor_id,     donation_date,     donation_details,
comments
) VALUES (
   1,     '2023-12-27',     '500ml blood',     'No
complications'
);
```

Update a donor's phone number:

```
UPDATE donors
SET phone_number = '555-4221'
WHERE donor_id = 1;
```

Get all staff members:

```
SELECT *
FROM staff;
```

	staff_id	address_id	blood_bank_id	staff_category_code	gender_mfu	staff_job_title	staff_first_name	staff_middle_name	staf
1	9	9	5	34	F	Nurse	Jessica	<null>	Smith
2	10	6	1	19	F	Nurse	Emily	<null>	Smith
3	11	7	5	46	U	Nurse	Sarah	<null>	Smith
4	12	2	3	42	U	Doctor	Sarah	<null>	Garcia
5	13	5	4	29	F	HR Specialist	Olivia	<null>	Brown
6	14	4	2	19	F	Nurse	Daniel	<null>	Martine
7	15	5	4	22	U	Maintenance	Robert	<null>	Smith
8	16	6	4	22	F	Doctor	Olivia	<null>	Martine

Get all donations made on specific date:

```
SELECT *
FROM donations
WHERE donation_date = '2023-01-20';
```

	donation_id	donor_id	donation_date	donation_details	comments
1	69	80	2023-01-20	450ml blood donation	No adverse reactions
2	70	81	2023-01-20	450ml blood donation	Felt dizzy afterwards

Possibility for indexes:

Due to the fact that the ***"Donations"*** table has an attribute ***"Donation_Amount"*** it is highly probable that if this attribute is made as an index, the overall complexity for queries related to this table will improve. Below you can see how this hypothesis can be tested:

1. Run the following query to find out the complexity of finding dates of donations that have at least 20 ml.

```
EXPLAIN ANALYSE SELECT donation_id, donation_date,
donation_amount
FROM donations
WHERE donation_amount > 20
```

2. Make an index out of ***"donation_amount"***.

```
CREATE INDEX donation_amount_index ON
donations(donation_amount)
```

3. Run the code from step 1 to see the improvements in the overall query complexity.

Conclusion:

The development of the database provided the hands-on-experience of how complex structures comply with the ideas of not only serving the purpose of creation but also being efficient in time and memory complexity. Also, one of the most important chapters is the compliance of the designed structure to the norms of BCNF which ensures data consistency.

Presented database was mounted and tested on the "Render" platform the usage of which allowed to show the results of queries. Below you can find the information to get access to the database.

Warning*: The following database is not the final version which was presented in the work. It was used only for testing the queries and getting experience on how to make a database non-local for team collaboration.*

Hostname: dpg-cltf2421hbls73ee14d0-a

Port: 5432

Database: blood_bank

Username: denis

Password: NvJqoqh0jIWAm1oOk4DFdooXwhWZC7Nk

External Database URL:
postgres://denis:NvJqoqh0jIWAm1oOk4DFdooXwhWZC7Nk@dpg-cltf2421hbls73ee14d0-a.frankfurt-postgres.render.com/blood_bank