Blood Bank Donation System

Database Design by

Kevin Clark

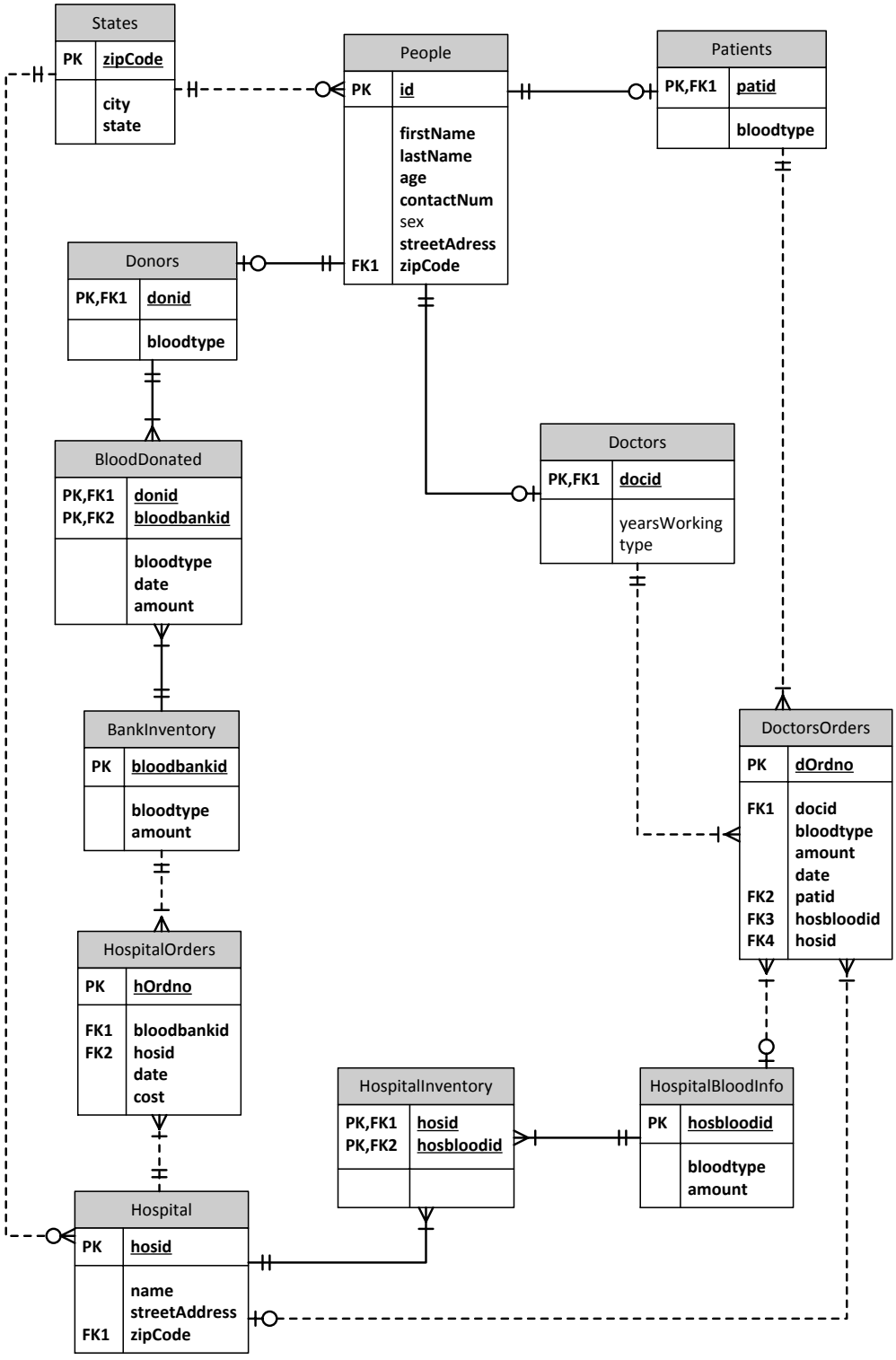
Table of Contents

Executive Summary

This document represents a design and implementation of a database for a blood donation system. This database is a way to see how donor's blood gets from the blood bank that they donated to all the way to the patients in the hospitals. Potential users include doctors, inventory keepers in blood banks and hospitals, and donors who are interested to see how their blood helped a person.

The Entity Relationship Diagram shows the relationships between all of the tables. It is shown first, followed by the tables, including their SQL code, with example data in them, and their functional dependencies. After the tables, the views, reports, and stored procedure are shown. Finally the security features and potential future enhancements are shown last.

Entity Relationship Diagram



Tables

State

Purpose

This table is used to store valid zip codes and their associated city and state for the people and hospital entities.

Create Statement

```
CREATE TABLE states (
  zipCode          varchar(5) not null,
  city             text not null,
  state            char(2) not null,
 primary key(zipCode)
);
```

Functional Dependencies

zipCode -> city, state

Sample Data

	zipcode character varying(5)	city text	state character(2)
1	12601	Poughkeepsie	NY
2	11731	Huntington	NY
3	07675	Westwood	NJ
4	96826	Honolulu	HI

People

Purpose

This table holds information for everybody that will be in the database including donors, doctors, and patients.

Create Statement

```
CREATE TABLE people (
  pid             char(4) not null,
  firstName       text not null,
  lastName        text not null,
  age             integer not null,
  contactNum      varchar(10) not null,
  sex             char(1),
  streetAddress   text not null,
  zipCode         varchar(5) not null references states(zipCode),
 primary key(pid)
);
```

Functional Dependencies

Pid -> firstName, lastName, age, contactNum, sex, streetAddress, zipCode

Sample Data

	pid character(4)	firstname text	lastname text	age integer	contactnum character varying(10)	sex character(1)	streetaddress text	zipcode character varying(5)
1	dn01	Bill	Nye	45	6514567436	M	123 Science	12601
2	dn02	James	Bond	26	6317894312	M	007 Badass A	12601
3	dn03	Frank	Zappa	28	2015789045	M	84 Guitar St	11731
4	dn04	Olga	Janiak	21	6314593246	F	3 Forest Dr.	11731
5	do01	Bill	Murray	68	8475891245	M	69 Beverly H	96826
6	do02	Frank	Doyle	53	6318952547	M	23 Meatloaf	07675
7	pa01	Sean	Connery	63	7850327490	M	85 Hollywood	96826
8	pa02	Eli	Doris	19	7542368579	M	15 Clare Dr.	11731
9	pa03	Ashley	Delano	20	758203657	F	851 Westwood	07675
10	pa04	Laura	Nillon	33	7589340243	F	3 Marriage S	12601

Donors

Purpose

This table contains the blood type of the donor.

Create Statement

```
CREATE TABLE donors (
  donid           char(4) not null references people(pid),
  bloodtype       text not null CHECK
      (bloodtype in ('A+' , 'A-' ,'B+', 'B-', 'O+', 'O-', 'AB+', 'AB-')),
 primary key(donid)
);
```

Functional Dependencies

Donid -> bloodtype

Sample Data

	donid character(4)	bloodtype text
1	dn01	A-
2	dn02	O+
3	dn03	AB+
4	dn04	B+

Doctors

Purpose

This table contains information on the doctor.

Create Statement

```
CREATE TABLE doctors (
  docid           char(4) not null references people(pid),
  yearsWorking    integer,
  type            text,
 primary key(docid)
);
```

Functional Dependencies

Docid -> yearsWorking, type

Sample Data

	docid character(4)	yearsworking integer	type text
1	do01	22	Surgeon
2	do02	15	Hematologist

Patients

Purpose

This table contains the blood type of the patient.

Create Statement

```
CREATE TABLE patients (
  patid           char(4) not null references people(pid),
  bloodtype       text not null CHECK
      (bloodtype in ('A+' , 'A-' ,'B+', 'B-', 'O+', 'O-', 'AB+', 'AB-')),
 primary key(patid)
);
```

Functional Dependencies

Patid -> bloodtype

Sample Data

	patid character(4)	bloodtype text
1	pa01	AB+
2	pa02	O+
3	pa03	B+
4	pa04	A-

Bank Inventory

Purpose

This table contains information about the blood currently in the blood bank that this database is being used in.

Create Statement

```
CREATE TABLE bankinventory (
  bloodbankid     char(4) not null,
  bloodtype       text not null CHECK
      (bloodtype in ('A+' , 'A-' ,'B+', 'B-', 'O+', 'O-', 'AB+', 'AB-')),
  amount          integer not null,
 primary key(bloodbankid)
);
```

Functional Dependencies

Bloodbankid -> bloodtype, amount

Sample Data

	bloodbankid character(4)	bloodtype text	amount integer
1	b001	O+	12
2	b002	A+	10
3	b003	B-	10
4	b004	AB-	9
5	b005	O-	13
6	b006	AB+	12
7	b007	O+	10
8	b008	A-	10
9	b009	B+	10

Blood Donated

Purpose

This table contains information about the blood donated from the patient to the blood bank.

Create Statement

```
CREATE TABLE blooddonated (
  donid           char(4) not null references donors(donid),
  bloodbankid     char(4) not null references bankinventory(bloodbankid),
  bloodtype       text not null CHECK
      (bloodtype in ('A+' , 'A-' ,'B+', 'B-', 'O+', 'O-', 'AB+', 'AB-')),
  amount          integer not null,
  date            date not null,
 primary key(donid, bloodbankid)
);
```

Functional Dependencies

Donid, bloodbankid -> bloodtype, amount, date

Sample Data

	donid character(4)	bloodbankid character(4)	bloodtype text	amount integer	date date
1	dn01	b008	A-	10	2014-04-19
2	dn02	b007	O+	13	2014-03-29
3	dn03	b006	AB+	12	2014-02-21
4	dn04	b009	B+	10	2014-04-20

Hospital

Purpose

This table contains information about the different hospitals that order blood from this blood bank.

Create Statement

```
CREATE TABLE hospital (
  hosid                 char(4) not null,
  name                  text not null,
  streetAddress   text not null,
  zipCode         varchar(5) not null references states(zipCode),
 primary key(hosid)
);
```

Functional Dependencies

Hosed -> name, streetAddress, zipCode

Sample Data

	hosid character(4)	name text	streetaddress text	zipcode character varying(5)
1	h001	St. Francis	108 North Rd.	12601
2	h002	Huntington Hospital	5 Main St.	11731
3	h003	Hackensack Hospital	16 Cherry Ln.	07675
4	h004	Straub Clinic	84 Volcano Ave.	96826

Hospital Orders

Purpose

This table contains information about the orders of blood that the hospital asks the blood bank for.

Create Statement

```
CREATE TABLE hospitalorders (
  hordno          char(4) not null,
  bloodbankid     char(4) not null references bankinventory(bloodbankid),
  hosid           char(4) not null references hospital(hosid),
  date            date not null,
  cost            numeric(10,2),
 primary key(hordno)
);
```

Functional Dependencies

Hordno -> bloodbankid, hosid, date, cost

Sample Data

	hordno character(4)	bloodbankid character(4)	hosid character(4)	date date	cost numeric(10,2)
1	o001	b006	h001	2014-02-22	635.00
2	o002	b007	h002	2014-02-22	635.00
3	o003	b008	h003	2014-02-22	635.00
4	o004	b009	h004	2014-04-21	784.00

Hospital Blood Information

Purpose

This table contains information about what blood all of the hospitals have.

Create Statement

```
CREATE TABLE hospitalbloodinfo (
  hosbloodid      char(4) not null,
  bloodtype       text not null CHECK
      (bloodtype in ('A+' , 'A-' ,'B+', 'B-', 'O+', 'O-', 'AB+', 'AB-')),
  amount          integer not null,
 primary key(hosbloodid)
);
```

Functional Dependencies

Hosbloodid -> bloodtype, amount

Sample Data

	hosbloodid character(4)	bloodtype text	amount integer
1	h001	AB+	12
2	h002	O+	1
3	h003	A-	10
4	h004	B+	10

Hospital Inventory

Purpose

This table contains information about what hospitals contain what blood.

Create Statement

```
CREATE TABLE hospitalinventory (
  hosid           char(4) not null references hospital(hosid),
  hosbloodid      char(4) not null references hospitalbloodinfo(hosbloodid),
 primary key(hosid, hosbloodid)
);
```

Functional Dependencies

None

Sample Data

	hosid character(4)	hosbloodid character(4)
1	h001	h004
2	h002	h003
3	h003	h002
4	h004	h001

Doctor's Orders

Purpose

This table shows information of the doctor's orders of blood from the hospital inventory to the patient.

Create Statement

```
CREATE TABLE doctorsorders (
  dordno          char(4) not null,
  hosbloodid      char(4) not null references hospitalbloodinfo(hosbloodid),
  docid           char(4) not null references doctors(docid),
  bloodtype text not null CHECK
      (bloodtype in ('A+' , 'A-' ,'B+', 'B-', 'O+', 'O-', 'AB+', 'AB-')),
  amount          integer not null,
  date            date not null,
  hosid           char(4) not null references hospital(hosid),
  patid           char(4) not null references patients(patid),
 primary key(dordno)
);
```

Functional Dependencies

Dordno -> hosbloodid, docid, bloodtype, amount, date, hosid, patid

Sample Data

	dordno character(4)	hosbloodid character(4)	docid character(4)	bloodtype text	amount integer	date date	hosid character(4)	patid character(4)
1	o001	h001	do02	AB+	12	2014-04-02	h001	pa01
2	o002	h002	do02	O+	12	2014-04-06	h002	pa02
3	o003	h004	do01	A-	10	2014-04-09	h004	pa04
4	o004	h003	do02	A-	12	2014-04-15	h003	pa03

Views

Patients Doctors

This view matches up patients with their doctors and no other data about the blood ordered for the patient.

Create Statement

```
CREATE VIEW patients_doctors as
Select pat.lastName AS "patient_lastname",
       pat.firstName AS "patient_firstname",
       doc.lastName AS "doctor_lastname",
       doc.firstName AS "doctor_firstname"
From people pat, people doc ,patients p, doctors d, doctorsorders doco
Where pat.pid = p.patid
  and doc.pid = d.docid
  and d.docid = doco.docid
  and p.patid = doco.patid
Order by pat.lastName
```

Sample Data

	patient_lastname text	patient_firstname text	doctor_lastname text	doctor_firstname text
1	Connery	Sean	Doyle	Frank
2	Delano	Ashley	Doyle	Frank
3	Doris	Eli	Doyle	Frank
4	Nillon	Laura	Murray	Bill

Donors to Patients

This view shows the name of the patient who got the donor's blood. It shows both the donor and patients name.

Create Statement

```
CREATE VIEW donors_to_patients as
Select don.lastName AS "donor_lastname",
       don.firstName AS "donor_firstname",
       pat.lastName AS "patient_lastname",
       pat.firstName AS "patient_firstname"
From people don, people pat, donors d, patients p, blooddonated bd,
bankinventory bi, hospitalorders ho, hospital h, hospitalinventory hi,
hospitalbloodinfo hbi, doctorsorders doco
Where don.pid = d.donid
  and pat.pid = p.patid
  and bd.donid = d.donid
  and bi.bloodbankid = bd.bloodbankid
  and ho.bloodbankid = bi.bloodbankid
  and h.hosid = ho.hosid
  and hi.hosid = h.hosid
  and hbi.hosbloodid = hi.hosbloodid
  and doco.hosbloodid = hbi.hosbloodid
  and doco.patid = p.patid
Order by don.lastName
```

Sample Data

	donor_lastname text	donor_firstname text	patient_lastname text	patient_firstname text
1	Bond	James	Delano	Ashley
2	Janiak	Olga	Connery	Sean
3	Nye	Bill	Doris	Eli
4	Zappa	Frank	Nillon	Laura

Change in ID View

This view shows the difference between the blood bank id number and the hospital id number for identifying blood.

Create Statement

```
CREATE VIEW change_in_id as
Select hi.hosbloodid AS "hospital_id",
       bi.bloodbankid AS "blood_bank_id"
From bankinventory bi, hospitalorders ho, hospital h, hospitalinventory hi
Where bi.bloodbankid = ho.bloodbankid
  and h.hosid = ho.hosid
  and hi.hosid = h.hosid
Order by hi.hosbloodid
```

Sample Data

	hospital_id character(4)	blood_bank_id character(4)
1	h001	b009
2	h002	b008
3	h003	b007
4	h004	b006

Reports

Difference between Donation and Ordering Dates

It is important to know how long the blood has been sitting around in the blood bank before a hospital orders for it.

Create Statement

```
Select bd.date - ho.date AS "days_sitting_in_bank", bd.date AS
"donation_date", ho.date AS "order_date"
From blooddonated bd, hospitalorders ho , bankinventory bi
Where bd.bloodbankid = bi.bloodbankid
  and ho.bloodbankid = bi.bloodbankid
```

Sample Data

	days_sitting_in_bank integer	donation_date date	order_date date
1	56	2014-04-19	2014-02-22
2	35	2014-03-29	2014-02-22

Difference between Doctor Order Date and Hospital Order Date

Again it is important to know how long blood has been sitting around in the hospital inventory before a doctor orders it.

Create Statement

```
Select doco.date - ho.date AS "days_sitting_in_hospital", ho.date AS
"hos_order_date", doco.date AS "doc_order_date"

From hospitalorders ho, hospital h, hospitalinventory hi, hospitalbloodinfo
hbi, doctorsorders doco

Where ho.hosid = h.hosid

  and hi.hosid = h.hosid

  and h.hosid = hbi.hosbloodid

  and doco.hosbloodid = hbi.hosbloodid
```

Sample Data

	days_sitting_in_hospital integer	hos_order_date date	doc_order_date date
1	39	2014-02-22	2014-04-02
2	43	2014-02-22	2014-04-06
3	52	2014-02-22	2014-04-15

Stored Procedures

Patients_in_hospitals()

Lists the patient's first and last names and what hospital they are in.

Create Statement

```
CREATE or REPLACE function patients_in_hospitals(text, REFCURSOR) returns
refcursor as
$$
DECLARE
  pat_lastName          text        := $1;
  --pat_firstName       text        := $2;
  resultset             REFCURSOR   := $2;
BEGIN
  open resultset for
    Select h.name AS "Hospital Name", p.lastName AS "Patient Last Name",
p.firstName AS "Patient First Name"
    from hospital h, doctorsorders doco, people p, patients pat
    where p.lastName = pat_lastName
      AND h.hosid = doco.hosid
      AND doco.patid = pat.patid
      AND pat.patid = p.pid
   ORDER BY h.name, p.lastName;
  return resultset;
END;
$$
language plpgsql;
```

Sample Data

```
Select patients_in_hospitals('Connery', 'results');
Fetch all from results;
```

	Hospital Name text	Patient Last Name text	Patient First Name text
1	St. Francis	Connery	Sean

Security

There would be two types of users for this database.

1. The administrator who can change, update, and maintain the database.

```
CREATE ROLE admin
GRANT SELECT, INSERT, UPDATE, ALTER
ON ALL TABLES IN SCHEMA PUBLIC
TO admin
```

2. The public user who can see the database and perform queries.

```
CREATE ROLE public
GRANT SELECT
ON ALL TABLES IN SCHEMA PUBLIC
TO public
```

Implementation Notes / Known Problems / Future Enhancements

The implementation went well with a couple of issues. The main issue was how exactly to decide on an id. The solution was to just use the first letter of whatever was being described and three numbers after it. This could pose a problem with a large amount of data as eventually there will be > 2 numbers needed to identify an item. This could be fixed with a future enhancement.

A known problem is that if administrator entering the information in accidentally mixes up the blood type between two tables (even if it is the same blood) then it will still be accepted. A future enhancement could be to create a stored procedure to fix this issue.

This database is specifically made for one blood bank that sends its blood out to many hospitals. A future enhancement that could be is to make it so that if the blood bank decides to open up another branch, they would have to specify which blood bank has which blood.