
Grassroots
Joseph Rowell

TABLE OF CONTENTS

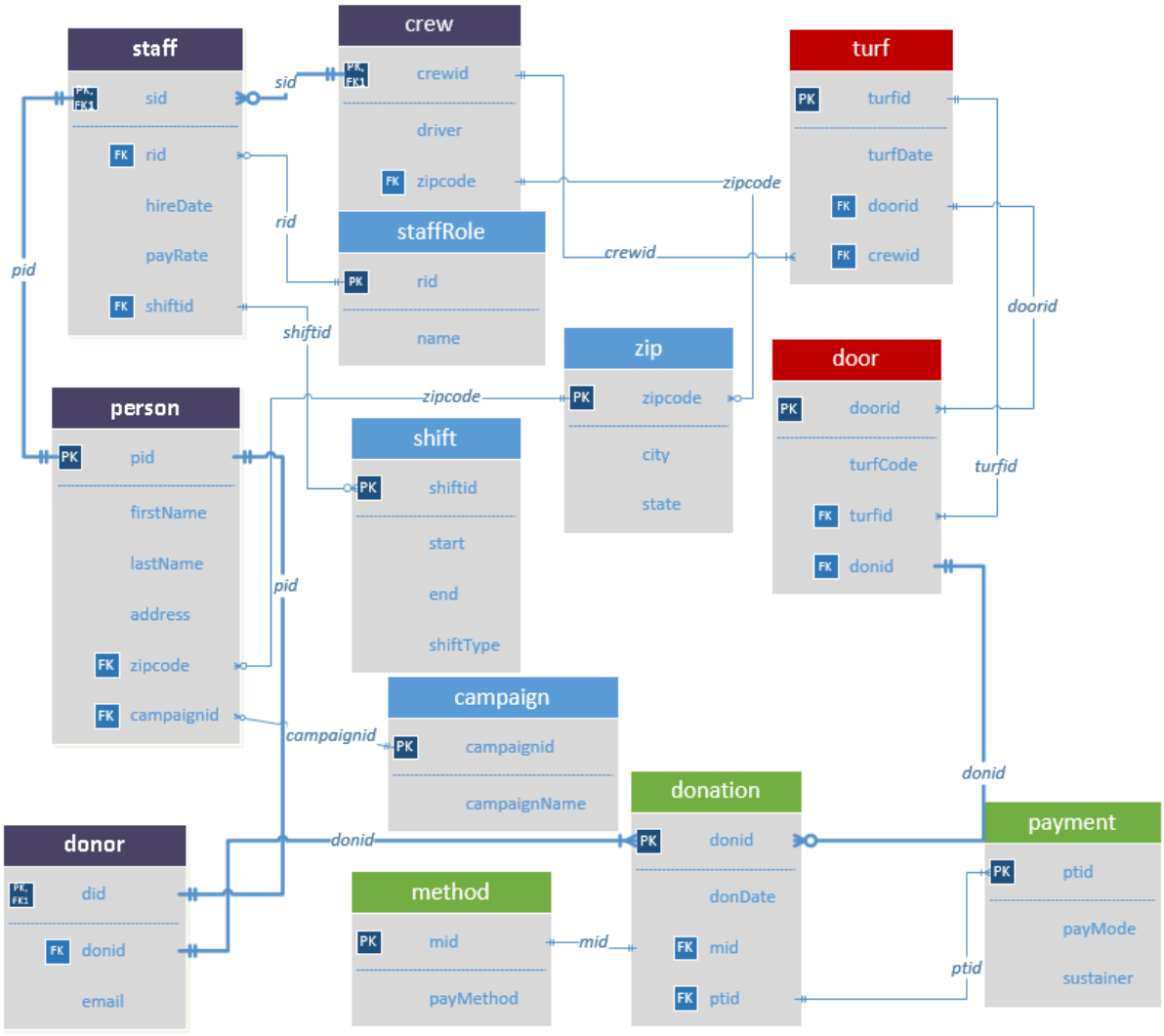
EXECUTIVE SUMMARY

Non-profits need to be able to have a way to manage relations with their best supporters. The ability to have a streamlined database is something that sets non-profits apart from others. While some organizations still use thousands of donor cards and a seemingly endless amount of paperwork, other organizations are leveraging data-driven applications that do away with the mountains of paperwork and overstressed project directors.

The existing data-driven applications do a great job for the most part, but in other areas they are missing out on groundbreaking ideas that would deliver an increased amount of revenue to nonprofits. This is where Grassroots comes in. Grassroots creates a Callback database to store potential givers. There are a few reasons why this is a novel idea: First of all, it saves turf. If you are a non-profit in a certain area, you want to make sure that you don't run out of turf. Second, if your fundraisers are not able to reach strong donors in a given area, new turf could be made to reach them on another date.

The design for this application includes employees, donors past and present, the dates given. It will include a view to show the most effective fundraisers. It will have triggers as well.

This database was designed and tested using PostgreSQL version 1.18.1.


staff
sid
rid
hireDate
payRate
shiftid
crew
crewid
driver
zipcode
staffRole
rid
name
turf
turfid
turfDate
doorid
crewid
zip
zipcode
city
state
door
doorid
turfCode
turfid
donid
person
pid
firstName
lastName
address
zipcode
campaignid
shift
shiftid
start
end
shiftType
campaign
campaignid
campaignName
donation
donid
donDate
mid
ptid
payment
ptid
payMode
sustainer
donor
did
donid
email
method
mid
payMethod
sid
rid
pid
shiftid
zipcode
crewid
doorid
turfid
donid
campaignid
mid
ptid

PERSON TABLE

There are two types of people: Staff and donors. When fundraising, you have people sign their names on a “statement of support” that contains their name, address, phone number and e-mail. Sometimes people at a given door will not put down their phone numbers or e-mails. Also, it is possible that a person can be a donor, as well as staff (there have been cases of donors who later become staff). All while this is going on, all people are part of campaigns. The person table serves as an aggregator for donors and staff.

```
CREATE TABLE IF NOT EXISTS person (
        pid SERIAL NOT NULL UNIQUE,
        firstName VARCHAR(50) NOT NULL,
        lastName VARCHAR(50) NOT NULL,
       address VARCHAR(80) NOT NULL,
       phonePrimary  CHAR(15) NOT NULL,
       FOREIGN KEY (zipcode) REFERENCES zip (zipcode),
       FOREIGN KEY (campaignid) REFERENCES campaign (campaignid),
       PRIMARY KEY (pid)

);
```

Functional Dependencies:

pid → firstName, lastName, address, phonePrimary, zipcode, campaignid

sample data:

insert into

STAFF TABLE

The people responsible for the fundraising. There are 3 common roles in a fundraising organization. These roles are: Canvasser, Field Manager and Project Director.

```
CREATE TABLE IF NOT EXISTS staff (
        sid INTEGER NOT NULL UNIQUE REFERENCES person (pid),
        FOREIGN KEY rid REFERENCES staffRole (rid),
        hireDate NOT NULL DEFAULT_CURRENT TIMESTAMP,
        payRate MONEY,
        FOREIGN KEY (turfid) REFERENCES turf (turfid),
        PRIMARY KEY (pid)

);
```

Functional Dependencies:

staff → sid, rid, hireDate, payRate, turfid

sample data:

CREW TABLE

Sometimes only one person may go out to one turf. This is true in very small canvassing operations. The norm is that somebody (usually a Field Manager) drives out and drops off canvassers at a particular turf. Usually, a Field Manager will only drop off canvassers in a particular zipcode, however, it is entirely possible that turfs could be separated by a zipcode. This is something that should be stressed on the administrative side (only canvass in one zipcode), however it is better to assume that a single turf could multiple zipcodes.

```
CREATE TABLE crew (
        crewid INT NOT NULL UNIQUE,
        driver CONSTRAINT (WHERE driver = staff.sid),
        FOREIGN KEY zipcode REFERENCES zip (zipcode),
        PRIMARY KEY (crewid)
);
```

Functional Dependencies:

crewid → driver, zip

Sample Data:

DONOR TABLE

Donors are the ones that keep non-profits running. They are important to the survival of an organization. No money, no mission. A donor can have multiple donations. Sometimes even two in one year because of things such as special appeals etc. The norm is that a donor will give one time per year for a particular issue.

```
CREATE TABLE IF NOT EXISTS donor (
        did INTEGER NOT NULL UNIQUE REFERENCES person (did),
        email VARCHAR(256),
        FOREIGN KEY (donid) REFERENCES donor (donid),
        PRIMARY KEY (did)

);
```

Functional Dependencies:

did → email, donid

Sample Data:

DONATION

A donor can have many donations and it is theoretically possible that you could have multiple donations at a door.

```
CREATE TABLE IF NOT EXISTS donation (
        donid SERIAL NOT NULL UNIQUE,
        donDate DEFAULT CURRENT DATE NOT NULL,
        FOREIGN KEY (mid) REFERENCES donation (mid),
        FOREIGN KEY (ptid) REFERENCES payment (ptid),
        PRIMARY KEY (did)
);
```

Functional Dependencies:

donid → donDate, mid, ptid

Sample Data:

METHOD

In order to get a better feel for donors, it is wise to list the method in which they contributed. Some people only like to give online, while some people enjoy giving at the door.

```
CREATE TABLE IF NOT EXISTS method (
        mid SERIAL NOT NULL,
        payMethod payMethod,
        PRIMARY KEY (mid)
);
```

Functional Dependencies:

mid → name

Sample data: Door, phone, online,

PAYMENT TABLE

This table shows if donors paid with cash, check or credit, and if they paid in credit if they are a monthly sustainer. If they are a monthly sustainer, how much they have given per month.

```
CREATE TABLE IF NOT EXISTS payType (
        ptid SERIAL NOT NULL,
        payMode payMode,
        CONSTRAINT sustainer CHECK (sustainer = 'Y' or sustainer = 'N'),
        PRIMARY KEY (ptid)
);
```

Functional Dependencies:
ptid → payMode, sustainer

CAMPAIGN TABLE

Denotes the campaigns conducted. Both types of people, canvassers and donors are part of a given campaign.

```
CREATE TABLE IF NOT EXISTS campaign (
        campaignid SERIAL NOT NULL,
        campaignName VARCHAR(75),
        PRIMARY KEY (campaignid)
);
```

Functional Dependencies:

campaignid → campaignName

Sample Data:

1, Special Appeal 2, Hydrofracking 3, Student Debt 4, Economic Justice

ZIP TABLE

This is for matching zipcodes to a city and state.

```
CREATE TABLE zip (
        zipcode VARCHAR (15),
        city VARCHAR (40),
        state TEXT(2),
        PRIMARY KEY (zipcode)
);
```

Functional Dependencies:

zipcode → city, state

Sample Data:

STAFFROLE TABLE

In every canvassing operation, everybody has a role. The roles most well known to a canvassing operation are: canvasser, Field Manager, and Project Director. One role can have many staff. A Project Director could also be a Field Manager and a Canvasser. A Field Manager is also a canvasser.

```
CREATE TABLE IF NOT EXISTS staffRole (
        rid SERIAL NOT NULL UNIQUE,
        roleName roleName,
        PRIMARY KEY (rid)

);
```

Functional Dependencies:

rid → name

SHIFT

When you are in charge of a large canvassing operation, it is important to distinguish trainees from established employees. This way, Project Directors know what types of expectations to set for a particular person. It is also important to know what time an employee came in and left at for payroll/attendance purposes.

```
CREATE TABLE IF NOT EXISTS shift (
        shiftid INT NOT NULL UNIQUE,
        timeIn TIME NOT NULL,
        timeout TIME NOT NULL,
        shiftType shiftType,
        PRIMARY KEY (shiftid)
);
```

Functional Dependencies:

shiftid →timeIn, timeOut

TURF TABLE

An individual is assigned to a turf, but they could also be with a trainer, or in a rare instance, two canvassers will go out and will either split up the turf or canvass the same doors together. A turf contains all the doors of people who given, or potential new givers. A turf has many doors, and there are many turfs in a given crew.

```
CREATE TABLE IF NOT EXISTS turf (
        turfid SERIAL NOT NULL UNIQUE,
        turfDate DEFAULT CURRENT_ DATE NOT NULL,
        FOREIGN KEY (doorid) REFERENCES door (doorid),
        FOREIGN KEY (crewid) REFERENCES turf (turfid),
        PRIMARY KEY (turfid)
);
```

Functional Dependencies:

turfid → turfDate, doorid, crewid

Sample Data:

DOOR

A door is a door whether you interact with the person behind the door or not. A door could theoretically have multiple donations, but it is kind of unlikely (See 'Known Problems' for the solution and workaround for this). One turf has many doors. There is one turf code identifier per door. These are all of the possibilities that happen when you approach a door. They could be either 'NH', 'CB', 'YESNM', 'YESR', 'YESC', 'MV', 'DC', 'X'.

```
CREATE TABLE IF NOT EXISTS door (
        doorid SERIAL NOT NULL UNIQUE,
        turfCode turfCode,
        FOREIGN KEY (turfid) REFERENCES turf (turfid),
        FOREIGN KEY (donid) REFERENCES donation (donid),
        PRIMARY KEY (doorid)
);
++
```

Functional Dependencies:

door → turfCode, turfid, donid

Sample Data:

VIEWS

There are lots of ways that views are leveraged to make this application run better.

TRIGGERS

STORED PROCEDURES

SECURITY

Canvassers, Field Managers, Project Coordinators, Database Administrator

Canvassers:

GRANT SELECT, UPDATE

Database Administrator:

GRANTALL PRIVELEGES ON ALL TABLES IN SCHEMA public TO dbAdmin;

KNOWN PROBLEMS

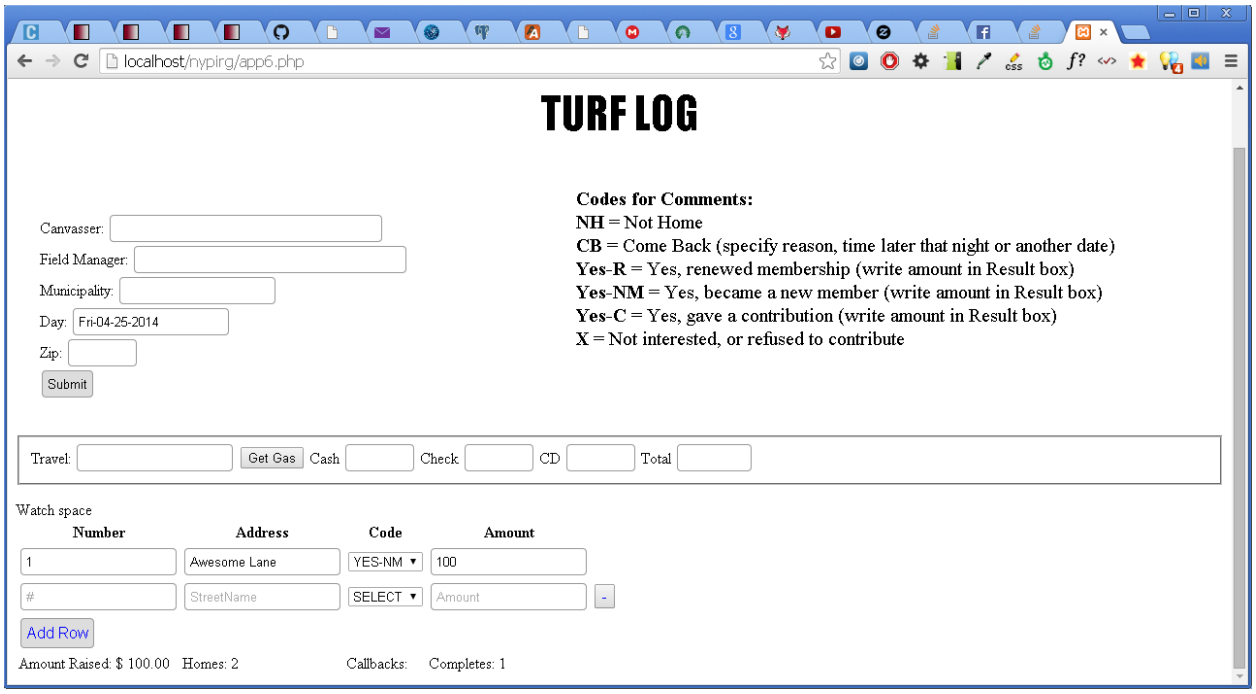
One door could possibly have many donations. The reason why this is a problem is because each row counts as a door. This anomaly does not happen very often but there are two easy possible workarounds. You could count the two donations as 1 person (ask before you do this), or you could put the other donation on the next row. This would mess up the door count by one, but you could at the end of the night delete a row marked as “NH”. By doing that, it’s as if you never knocked on that door that was “Not Home”. The total door count is usually expected to be off by one or two anyways because of canvasser miscounts etc. You definitely do NOT want to miscount actual donors/donations.

FUTURE

Since this project is a particular interest of mine, I will be improving it in the future. I do have a barely working (not working) frontend to insert the data in. I will be fixing it up and improving the Javascript. Once I have the frontend acting nicely and doing what I want it to do, I’m going to implement some really important features.

Election database: Organizations like Working Families do both fundraising and election work, whether the election work is gathering signatures, or getting people out to the polls for election day. There needs to be a mode where you can switch over to an election based database.

Fully functional Frontend: This is what I have so far:



I spent over 30 hours on this. I kept getting weird PostgreSQL errors, but I did my best.