
Grassroots
Joseph Rowell

TABLE OF CONTENTS

The SQL file is fundraise.sql

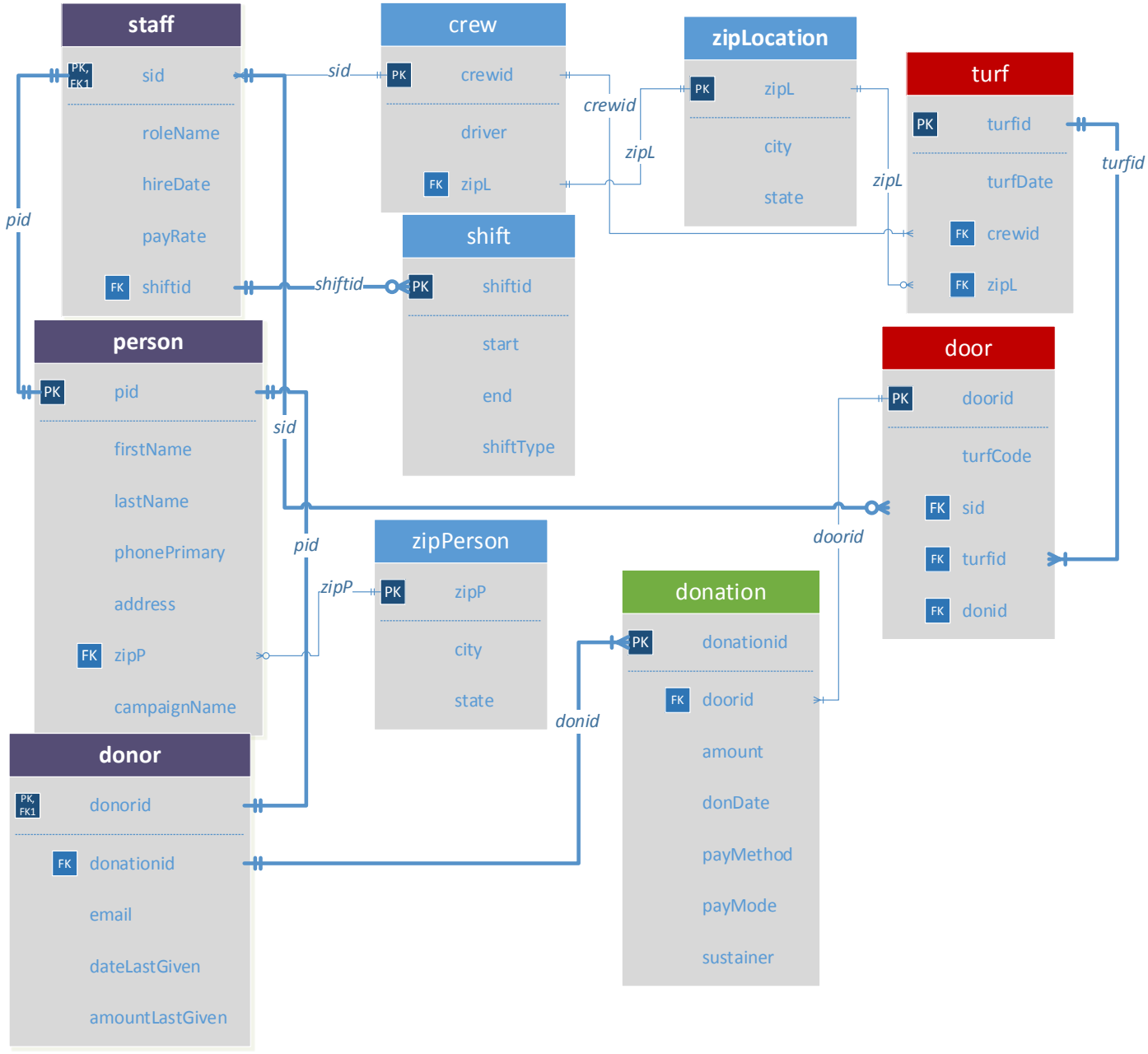
EXECUTIVE SUMMARY

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.



*sid on the under the ‘door’ table was changed to sdid.

ZIPPERSON TABLE

This is for matching zip codes to a person. Includes the city and state.

```
CREATE TABLE IF NOT EXISTS zipPerson (
        zipP VARCHAR (15) NOT NULL UNIQUE,
        city VARCHAR (40) NOT NULL,
        state CHAR(3) NOT NULL,
        PRIMARY KEY (zipP)

);
```

Functional Dependencies:

zipcode → city, state

Sample Data:

	zipp character varying(15)	city character varying(40)	state character(3)
1	12603	Poughkeepsie	NY
2	07712	Asbury Park	NJ
3	12561	New Paltz	NY
4	12538	Hyde Park	NY
5	41949	Punksville	PA
6	89421	Los Angeles	CA
7	97255	Mortalville	NY
8	89429	Freedom	NH
9	56412	Ourganac	VT

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,
        zipP VARCHAR(15) NOT NULL REFERENCES zipPerson (zipP),
        campaignName VARCHAR (50) NOT NULL,
        PRIMARY KEY (pid)
);
```

Functional Dependencies:

pid → firstName, lastName, address, phonePrimary, zipP, campaignName

	pid integer	firstname character varying(50)	lastname character varying(50)	zipp character varying(15)	campaignname character varying(50)
1	1	Bob	Dole	12603	hydrofracking
2	2	Bruce	Springsteen	07712	hydrofracking
3	3	Weinberg	Bottomtooth	12603	hydrofracking
4	4	Joe	Smoe	12603	hydrofracking
5	5	Vladamir	Lenin	12538	hydrofracking
6	6	Sid	Vicious	12603	hydrofracking
7	7	Nancy	Spungen	12603	hydrofracking
8	8	D	Boon	89421	hydrofracking
9	9	Exene	Cerveka	89421	hydrofracking
10	10	Laura	Croft	89421	hydrofracking
11	11	Sonia	Blade	97255	Clean Elections
12	12	Johnny	Cage	97255	Clean Elections
13	13	Joe	King	89429	Clean Elections
14	14	Joey	Ramone	89429	Clean Elections
15	15	Lee	Ving	89429	Clean Elections
16	16	Darby	Crash	89429	Free College
17	17	Joan	Jett	56412	Free College
18	18	Mikey	Erg	56412	Free College
19	19	Iggy	Pop	56412	Free College
20	20	Flava	Flav	56412	Free College

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 REFERENCES person (pid),
        hireDate DATE NOT NULL DEFAULT CURRENT_DATE,
        payRate MONEY,
        roleName roleName,
        PRIMARY KEY (sid)
);
```

Functional Dependencies:

sid →hireDate, payRate, roleName

sample data:

	pid inte	firstname character	lastname character vary	address character varying(80)	phoneprimary character(15)	zipp charac	campaignname character varying(	hiredate date	payrate money	rolename rolename
1	1	Bob	Dole	5 dole lane	123-444-4444	12603	hydrofracking	2013-05-08	$10.00	Trainee
2	2	Bruce	Springsteen	20 Asbury Lane	222-867-5309	07712	hydrofracking	2013-05-08	$15.00	Project Director
3	5	Vladamir	Lenin	20 Red Square Drive	555-777-9494	12538	hydrofracking	2013-05-08	$12.00	Field Manager
4	8	D	Boon	48 DoubleNikels Drive	933-555-8888	89421	hydrofracking	2013-05-08	$11.00	Canvasser
5	12	Johnny	Cage	92 Finishhim Way	335-434-5677	97255	Clean Elections	2012-05-07	$12.00	Field Manager

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 zipL (zipLocation),
        PRIMARY KEY (crewid)
);
```

Functional Dependencies:

crewid → driver, zipL

Sample Data:

	crewid integer	driver character varying(50)	zipl character varying(15)
1	2	Bruce Springsteen	12603
2	5	D Boon	12561
3	12	Johnny Cage	12561

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 (
        donorid INTEGER NOT NULL UNIQUE REFERENCES person (pid),
        dateLastGiven DATE NOT NULL,
        amountLastGiven MONEY NOT NULL,
        email VARCHAR(256),
        donationid INT NOT NULL REFERENCES donation (donationid),
        PRIMARY KEY (donorid)
);
```

Functional Dependencies:

donorid → dateLastGiven, amountLastGiven, email, donationid

Sample Data:

	donorid integer	datelastgiven date	amountlastgiven money	email character varying(256)	donationid integer
1	3	2012-05-15	$300.00	toorichformyowngood	1
2	4	2011-08-14	$3.00	eatatsmoes@gmail.co:	2
3	6	2010-05-17	$66.00	vicious4life@anonma	3
4	7	2012-05-15	$67.00	nancyh8sfracking@ea	4
5	9	2011-05-01	$66.00	xcervenkax@xmail.co:	5
6	21	2011-05-01	$365.00	frackfracking@frack	6
7	14	2011-05-10	$6.00	heyholetsgo@aol.com	7
8	11	2010-05-05	$100.00	bladelady44@mkombat	8
9	13	2011-05-25	$12.00		9
10	15	2011-05-25	$62.00		10

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 (
        donationid SERIAL NOT NULL UNIQUE,
        donDate DATE NOT NULL DEFAULT CURRENT_DATE,
        amount MONEY NOT NULL,
        payMethod payMethod,
        payMode payMode,
        sustainer BOOLEAN NOT NULL,
        PRIMARY KEY (donationid)
);
```

Functional Dependencies:

donationid → donDate, amount, payMethod, payMode, sustainer

Sample Data:

	donationid integer	dondate date	amount money	paymethod paymethod	paymode paymode	sustainer boolean
1	1	2013-05	$365.00	Check	Door	f
2	2	2013-05	$36.00	Cash	Door	f
3	3	2013-05	$1.00	Check	Door	f
4	4	2013-05	$68.00	Check	Door	f
5	5	2013-05	$68.00	Credit	Door	f
6	6	2013-05	$365.00	Check	Door	t

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 integer	timein time without time zone	timeout time without time zone	shifttype shifttype
1	1	02:00:00	09:30:00	Training
2	2	12:00:00	09:30:00	Full
3	3	01:00:00	09:30:00	Full
4	4	01:00:00	09:30:00	Full
5	5	12:00:00	09:30:00	Full

shiftid → timeIn, timeOut, shiftType

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 (crewid) REFERENCES crew (crewid),
        PRIMARY KEY (turfid)
);
```

Functional Dependencies:

turfid → turfDate, crewid

Sample Data:

	turfid integer	turfdate date	crewid integer
1	1	2013-05-10	2
2	2	2013-05-10	12
3	3	2011-05-10	5
4	4	2013-05-10	5
5	5	2011-05-25	12

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', or 'X'.

On the frontend that I made for this database, here are the codes for comments:

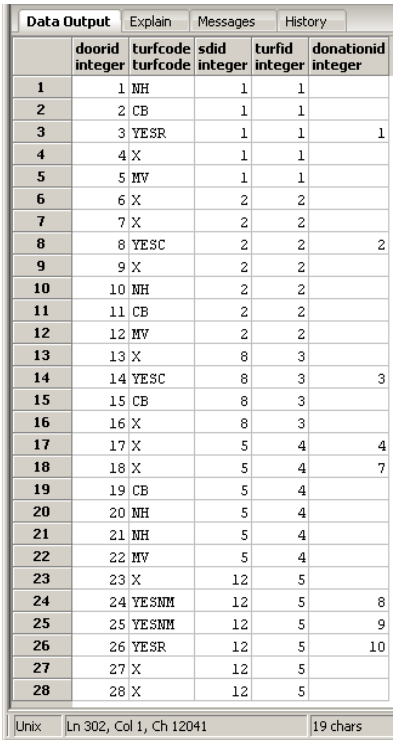
Codes for Comments:
NH = Not Home
CB = Come Back (specify reason, time later that night or another date)
Yes-R = Yes, renewed membership (write amount in Result box)
Yes-NM = Yes, became a new member (write amount in Result box)
Yes-C = Yes, gave a contribution (write amount in Result box)
X = Not interested, or refused to contribute
DC = Deceased

```
CREATE TABLE IF NOT EXISTS door (
        doorid SERIAL NOT NULL UNIQUE,
        turfCode turfCode,
        sdid INT NOT NULL REFERENCES staff(sid),
        turfid INT NOT NULL REFERENCES turf (turfid),
        donationid INT REFERENCES donation (donationid),
        PRIMARY KEY (doorid)
);
```

Functional Dependencies:

door → turfCode, sdid, turfid, donationid

Sample Data:



Data Output | Explain | Messages | History

	doorid integer	turfcode turfcode	sdid integer	turfid integer	donationid integer
1	1	NH	1	1	
2	2	CB	1	1	
3	3	YESR	1	1	1
4	4	X	1	1	
5	5	MV	1	1	
6	6	X	2	2	
7	7	X	2	2	
8	8	YESC	2	2	2
9	9	X	2	2	
10	10	NH	2	2	
11	11	CB	2	2	
12	12	MV	2	2	
13	13	X	8	3	
14	14	YESC	8	3	3
15	15	CB	8	3	
16	16	X	8	3	
17	17	X	5	4	4
18	18	X	5	4	7
19	19	CB	5	4	
20	20	NH	5	4	
21	21	NH	5	4	
22	22	MV	5	4	
23	23	X	12	5	
24	24	YESNM	12	5	8
25	25	YESNM	12	5	9
26	26	YESR	12	5	10
27	27	X	12	5	
28	28	X	12	5	

Unix | Ln 302, Col 1, Ch 12041 | 19 chars

VIEWS

There are lots of ways that views are leveraged to make this application run better. Creating views is very crucial to find out awesome pieces of important information.

Nightly donors:

```
CREATE VIEW tonightsDonors AS
select distinct firstName, lastName, address, zipPerson.city, phonePrimary, amount from person
        INNER JOIN zipPerson
        on zipPerson.zipP = person.zipP
        INNER JOIN donor
        on person.pid = donor.donorid
        INNER JOIN donation
        on donor.donationid = donation.donationid
        order by city asc;
```

	firstname character varying(50)	lastname character varying(50)	address character varying(80)	city character varying(40)	phoneprimary character(15)	amount money
1	Joe	King	88 Webelo Way	Freedom	343-564-4322	$12.00
2	Joey	Ramone	53rd and 3rd	Freedom	350-435-5364	$365.00
3	Lee	Ving	44 Fear Place	Freedom	335-333-1122	$84.00
4	Exene	Cerveka	80 LosAngeles Blvd	Los Angeles	333-111-3435	$68.00
5	Sonia	Blade	94 Finishhim Way	Mortalville	924-345-5335	$120.00
6	Doctor	Freedom	2014 Nofracking Way	Poughkeepsie	951-753-4682	$365.00
7	Joe	Smoe	1 Main Street	Poughkeepsie	123-456-7890	$36.00
8	Nancy	Spungen	76 Anarchy Plaza	Poughkeepsie	333-707-7070	$67.00
9	Sid	Vicious	77 Anarchy Plaza	Poughkeepsie	111-222-1234	$1.00
10	Weinberg	Bottomtooth	1 Private Drive	Poughkeepsie	666-888-9999	$365.00

Nightly totals by payment method:

```
DROP VIEW IF EXISTS payMethodTotals;

CREATE VIEW payMethodTotals AS

SELECT sum(amount), payMethod as nightlyTotals
FROM donation where payMethod = 'Check'
GROUP BY payMethod union

SELECT sum(amount), payMethod as cashTotal
FROM donation where payMethod = 'Cash'
GROUP BY payMethod union

SELECT sum(amount), payMethod as bitcoinTotal
FROM donation where payMethod = 'Bitcoin'
GROUP BY payMethod union

SELECT sum(amount), payMethod as creditTotal
FROM donation where payMethod = 'Credit'
GROUP BY payMethod union

SELECT sum(amount), payMethod as debitTotal
FROM donation where payMethod = 'Debit'
GROUP BY payMethod;
```

	sum money	nightlytotals paymethod
1	$48.00	Cash
2	$120.00	Bitcoin
3	$152.00	Credit
4	$1,398.00	Check

*Nobody paid in debit, so it did not show up

Code counts:
This is a very important aspect of a fundraising organization: The turf codes. They tell how effective canvassers and campaigns are.

```
DROP VIEW IF EXISTS codeCounts

CREATE VIEW codeCounts AS
SELECT count(*) tCount, turfCode from door as nhCount
where turfCode = 'NH'
GROUP BY turfCode UNION

SELECT count(*) tCount, turfCode from door as cbCount
where turfCode = 'CB'
GROUP BY turfCode UNION

SELECT count(*) tCount, turfCode from door as yrCount
where turfCode = 'YESR'
GROUP BY turfCode UNION

SELECT count(*) tCount, turfCode from door as ynCount
where turfCode = 'YESNM'
GROUP BY turfCode UNION

SELECT count(*) tCount, turfCode from door as ycCount
where turfCode = 'YESC'
GROUP BY turfCode UNION

SELECT count(*) tCount, turfCode from door as MVCount
where turfCode = 'MV'
GROUP BY turfCode UNION

SELECT count(*) tCount, turfCode from door as xCount
where turfCode = 'X'
GROUP BY turfCode UNION

SELECT count(*) tCount, turfCode from door as dcCount
where turfCode = 'DC'
GROUP BY turfCode;
```

	turfcode bigint	turfcode turfcode
1	4	NH
2	4	CB
3	2	YESR
4	2	YESNM
5	3	MV
6	2	YESC
7	11	X

*Nobody died, which is great.

STORED PROCEDURES

When you are a nonprofit looking for the best fundraising talent, the best way to measure their success is to see how they are performing with their contact rate. Below is a function to calculate the contact rate for a given canvasser.

```
CREATE FUNCTION getContactRate(pid int) RETURNS decimal AS $$
DECLARE
contact integer := (SELECT count(*) FROM door

WHERE sdid = pid AND turfCode = 'YESR' OR 'YESC' OR 'YESNM');
numHomes integer := (SELECT count(*) FROM door
WHERE sdid = pid);
ratio decimal := 0;

BEGIN
ratio := (SELECT CAST (contacts AS decimal(2))/(SELECT CAST (numHomes AS decimal(2))));
return (trunc (ratio, 3));
END;
$$ LANGUAGE plpgsql;
```

TRIGGERS

```
CREATE TRIGGER contactRate
  After Insert
  ON door
  FOR EACH ROW
  EXECUTE PROCEDURE getContactRate(pid int);
```

After every insert, the contact rate will be calculated, and the correct output will be specified on a dynamic PHP variable once the full design of this application is completed.

SECURITY

Canvassers, Field Managers, Project Coordinators, Database Administrator

--Canvasser:

```
create role canvasser;
GRANT SELECT, UPDATE, INSERT
ON door, donation
TO canvasser;
```

--Field Manager

```
create role fieldManager;
GRANT SELECT, UPDATE, DELETE, INSERT
ON door, donation, crew;
GRANT SELECT, UPDATE, INSERT, DELETE
ON turf, zipLocation
TO fieldManager;
```

--Project Coordinator

```
create role projectCoordinator;
GRANT SELECT, UPDATE, DELETE, INSERT
ON door, donation, crew, staff, shift, zipPerson, donor
TO projectCoordinator;
```

--Database Administrator:

```
Create role dbAdmin;
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.

Another thing that was a pain was having a turfDate and a donation date. I did this because it is possible that a canvasser could have zero donations at the end of the day. I would have fixed this if I wasn’t already fairly far into the project.

One thing that probably violates normalization is the ‘driver’ field in the ‘crew’ table. I should have just simplified it to make sure that only Field Managers can be drivers, but from my experience, a canvasser can sometimes also be a driver. I tried to put a check constraint in to make sure that only anybody in staff can be a driver, but for some reason that did not work. Maybe I should have tried to put the driver field in at a later date. I kinda wish I didn’t add it now because I’m making Codd roll in his grave.

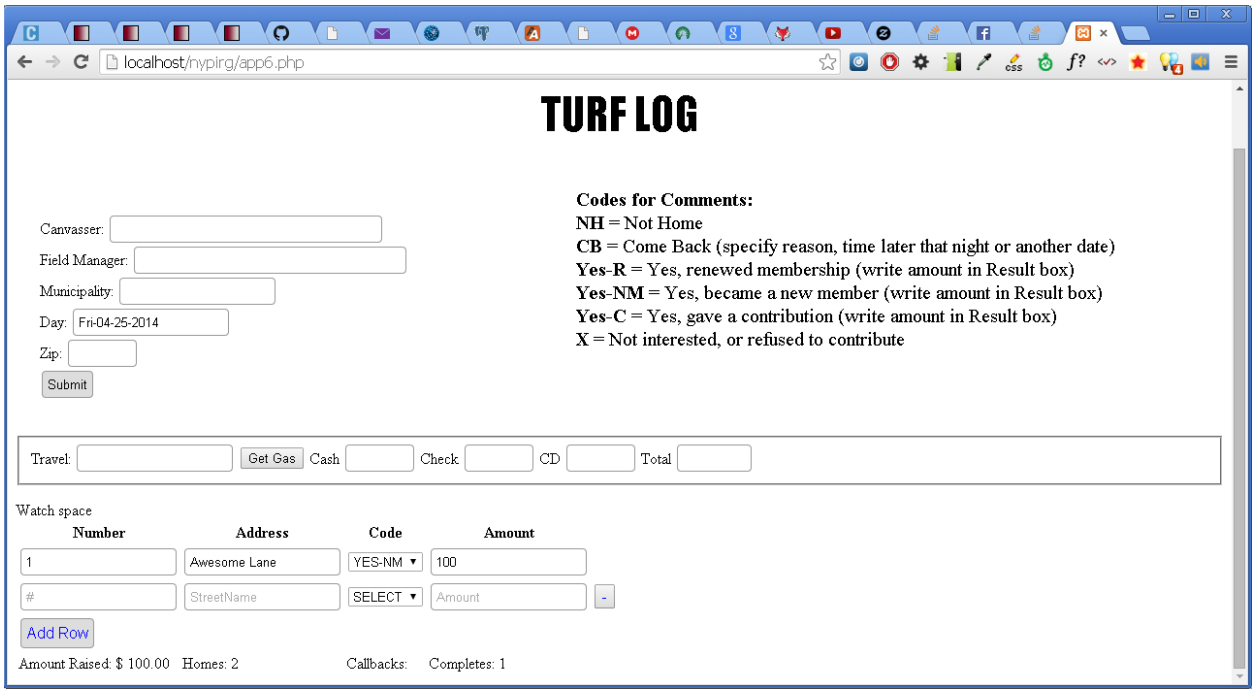
The views that I made are for an entire night(s), and not a single individual’s payMethods or turfCodes.

FUTURE

Since this project is a particular interest of mine, I will be improving it in the future. I do have a barely 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. I'm going to incorporate PHP with PostgreSQL to have a fully functional system.

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:



This is built mostly using jQuery. I want to totally redesign this using AngularJS.

Improved donation history: Some organizations have it where you can see each year that they have donated previously. So I could do like donation2Years, donation3Years to mark in the past each year they gave, but this would add a lot of NULL values. I could implement a running total of all donations from 1 person.

More Views: Even more ways to make sense of the various data that will be helpful to a canvassing organization.

More Stored Procedures/Triggers: I will make a stored procedure that makes it so that all new donations (whether they come from new or past donors) update in the 'amountLastGiven' field.

Integration with Google Maps: Lost canvassers can be a real problem to organizations, with the help of GPS mapping, this will become less of an issue.

More sample data: With a variety of sample data, I'll be able to try out a variety of scenerios.