THE NETWORK OF GROWERS

In Fall 2017, Arkansas Interfaith Power & Light (IPL) was awarded a $20,000 grant by the Environmental Protection Agency. This funding was granted for the purpose of building a network of gardeners, urban farmers, community garden members and managers, as well as large-scale growers, with the purpose of facilitating the donation of excess produce to IPL, which will then be distributed to 3 local food pantries in low-income neighborhoods.

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

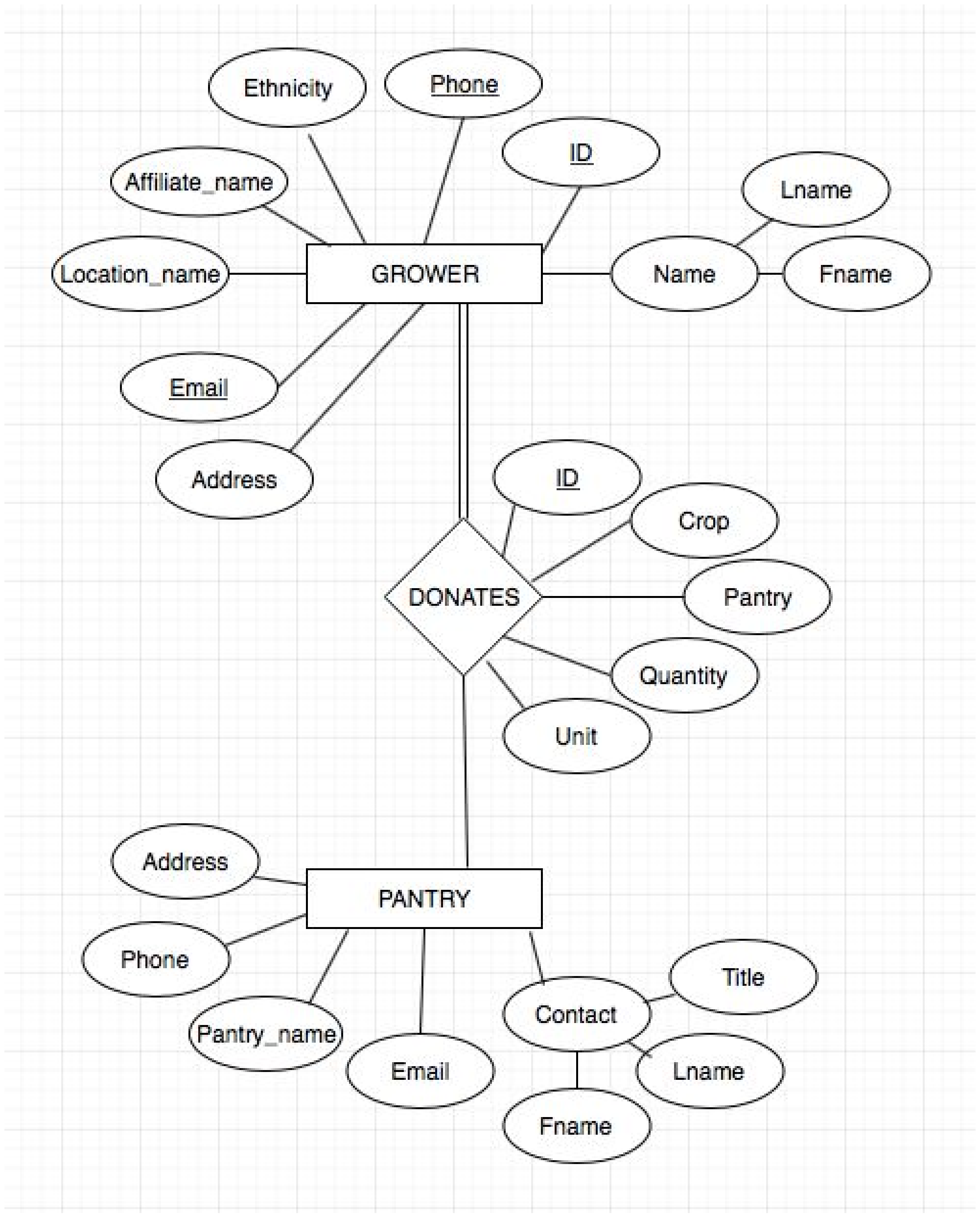
THE SCHEMA

DONATES(id, donationId, crop, quantity, unitOfMeasurement, donationDate, pantry_id) -- donationId is unique, id and pantry_id are foreign keys

GROWERS(id, name, phone, email, address, affiliate_name, location_name) -- Id is unique

PANTRY(pantry_id, phone, contact, email, pantry_name, address) -- pantry_id and phone are unique

THE DESIGN


Ethnicity
Phone
ID
Affiliate_name
Lname
Location_name
GROWER
Name
Fname
Email
Address
ID
Crop
DONATES
Pantry
Quantity
Unit
Address
PANTRY
Phone
Title
Contact
Pantry_name
Email
Lname
Fname

THE CODE — CREATING AND POPULATING THE TABLES

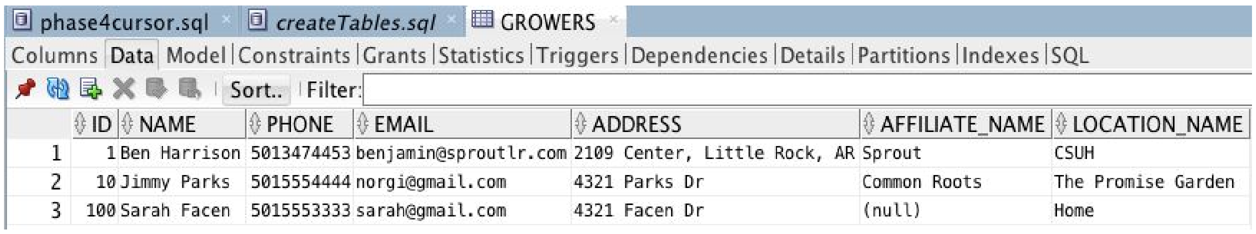
1. This code generates and populates the GROWERS table.

```
create table growers (
   id number not null unique,
   name varchar2(30) not null,
   phone number not null,
   email varchar2(50),
   address varchar2(50),
   affiliate_name varchar2(50),
   location_name varchar2(50),
   FOREIGN KEY (id) REFERENCES donates(id)
);

insert into growers
values (1,'Ben Harrison',5013474453,'benjamin@sproutlr.com','2109 Center,
   Little Rock, AR','Sprout','CSUH');

insert into growers
values (10,'Jimmy Parks',5015554444,'norgi@gmail.com','4321 Parks Dr',
   'Common Roots','The Promise Garden');

insert into growers
values (100,'Sarah Facen',5015553333,'sarah@gmail.com','4321 Facen Dr',null,
   'Home');
```



	ID	NAME	PHONE	EMAIL	ADDRESS	AFFILIATE_NAME	LOCATION_NAME
1	1	Ben Harrison	5013474453	benjamin@sproutlr.com	2109 Center, Little Rock, AR	Sprout	CSUH
2	10	Jimmy Parks	5015554444	norgi@gmail.com	4321 Parks Dr	Common Roots	The Promise Garden
3	100	Sarah Facen	5015553333	sarah@gmail.com	4321 Facen Dr	(null)	Home

2. This code generates and populates the PANTRY table.

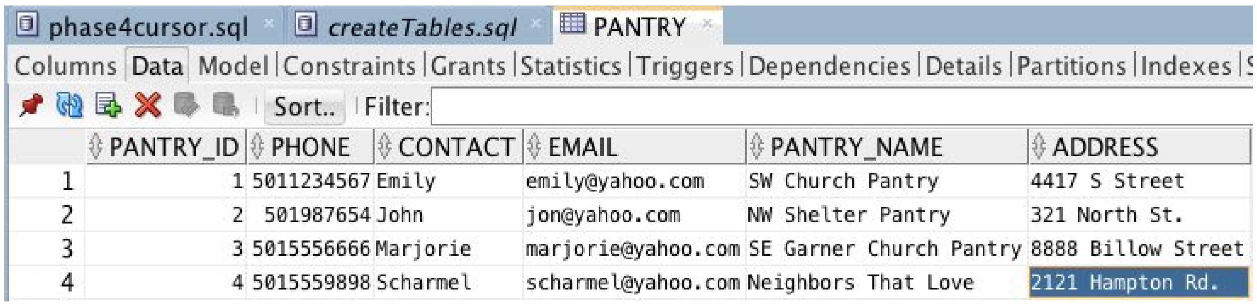
```
create table pantry (
    pantry_id number not null unique,
    phone number not null unique,
    contact varchar2(50),
    email varchar2(50),
    pantry_name varchar2(50) not null,
    address varchar2(50)
    FOREIGN KEY (pantry_id) REFERENCES donates(pantry_id)
);

insert into pantry
values (1,5011234567,'Emily','emily@yahoo.com','SW Church Pantry',
    '4417 S Street');

insert into pantry
values (2,501987654,'John','john@yahoo.com','NW Shelter Pantry',
    '321 North St.');

insert into pantry
values (3,5015556666,'Marjorie','marjorie@yahoo.com','SE Garner Church Pantry',
    '8888 Billow Street');

insert into pantry
values (4,5015559898,'Brittany','brittany@gmail.com','Neighbors That Love',
    '2121 Hampton Rd.');
```

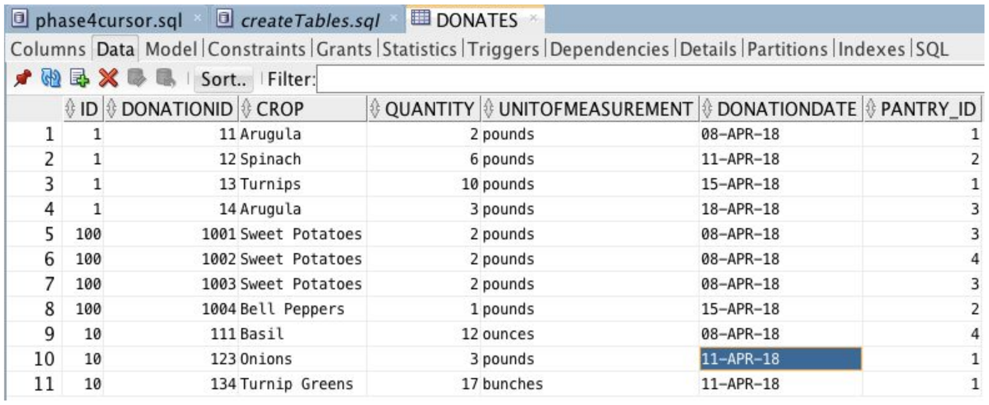


	PANTRY_ID	PHONE	CONTACT	EMAIL	PANTRY_NAME	ADDRESS
1	1	5011234567	Emily	emily@yahoo.com	SW Church Pantry	4417 S Street
2	2	501987654	John	jon@yahoo.com	NW Shelter Pantry	321 North St.
3	3	5015556666	Marjorie	marjorie@yahoo.com	SE Garner Church Pantry	8888 Billow Street
4	4	5015559898	Scharmel	scharmel@yahoo.com	Neighbors That Love	2121 Hampton Rd.

3. This code generates and populates the DONATES table.

```
create table donates (
   id number not null,
   donationId number not null unique,
   crop varchar2(30) not null,
   quantity number not null,
   unitOfMeasurement varchar2(10) not null,
   donationDate date not null,
   pantryId number,
   FOREIGN KEY (pantry_id) REFERENCES pantry(pantry_id)
);

insert into donates
values (1,11,'Arugula',2,'pounds',08-APR-18,1);
insert into donates
values (1,12,'Spinach',6,'pounds',11-APR-18,2);
insert into donates
values (1,13,'Turnips',10,'pounds',15-APR-18,1);
insert into donates
values (1,14,'Arugula',3,'pounds',18-APR-18,3);
insert into donates
values (100,1001,'Sweet Potatoes',2,'pounds',08-APR-18,3);
insert into donates
values (100,1002,'Sweet Potatoes',2,'pounds',08-APR-18,4);
insert into donates
values (100,1003,'Sweet Potatoes',2,'pounds',08-APR-18,3);
insert into donates
values (100,1004,'Bell Peppers',2,'pounds',15-APR-18,2);
insert into donates
values (10,111,'Basil',.75,'pounds',08-APR-18,4);
insert into donates
values (10,123,'Onions',3,'pounds',11-APR-18,1);
insert into donates
values (10,134,'Turnip Greens',17,'bunches',11-APR-18,1);
```


phase4cursor.sql
createTables.sql
DONATES
Columns Data Model Constraints Grants Statistics Triggers Dependencies Details Partitions Indexes SQL
Sort..
Filter:

	ID	DONATIONID	CROP	QUANTITY	UNITOFMEASUREMENT	DONATIONDATE	PANTRY_ID
1	1	11	Arugula	2	pounds	08-APR-18	1
2	1	12	Spinach	6	pounds	11-APR-18	2
3	1	13	Turnips	10	pounds	15-APR-18	1
4	1	14	Arugula	3	pounds	18-APR-18	3
5	100	1001	Sweet Potatoes	2	pounds	08-APR-18	3
6	100	1002	Sweet Potatoes	2	pounds	08-APR-18	4
7	100	1003	Sweet Potatoes	2	pounds	08-APR-18	3
8	100	1004	Bell Peppers	1	pounds	15-APR-18	2
9	10	111	Basil	12	ounces	08-APR-18	4
10	10	123	Onions	3	pounds	11-APR-18	1
11	10	134	Turnip Greens	17	bunches	11-APR-18	1

THE CODE — MANIPULATING THE TABLES

1. This query is intended to inform the analyst of how many donations included a particular crop.

```
set serveroutput on;
DECLARE
        na integer :=0;
BEGIN
        Select count(*)
                into na
        from DONATES
        where crop='Arugula';
        if (na > 0) then
                DBMS_OUTPUT.PUT_LINE('There have been a total of ' || na || ' donations with
                arugula.');
        else
                DBMS_OUTPUT.PUT_LINE('No arugula has been donated.');
        end if;
END;
```

2. This query uses a loop and 2 arrays to count the number of times each crop has been donated.

```
set serveroutput on;
DECLARE
        type cropCount is varray(8) of  integer;
        cpcnt cropCount;
        type crop is varray(8) of VARCHAR2(30);
        cp crop;
        cnt number;
BEGIN
        cpcnt := cropCount(0,0,0,0,0,0,0,0);
        cp  :=  crop('Arugula','Spinach','Turnips','Sweet Potatoes','Bell Peppers',
        'Basil', 'Onions', 'Turnips Greens');
        cnt := cp.count;

for i in 1..cnt loop
```

```
            select count(*)
            into cpcnt(i)
            from DONATES
            where crop = cp(i);
            dbms_output.put_line('The crop ' || cp(i) || ' has been donated '
            || cpcnt(i) || ' times ');
      end loop;
END;
```

3. This cursor formula selects a tuple based on the grower’s ID# and replaces their phone number.

```
Create procedure update_Phone (ID in growers.ID%type, new_Phone in
growers.phone%type) as
new_phone number;

DECLARE
   cursor c1 is
      select * from growers
      for update;
      c1_rec c1%rowtype;
BEGIN
      for c1_rec in c1 loop
      if (c1_rec.ID = 1) then
         update growers
         set phone = 4172620688
         where current of c1;
      end if;
   end loop;
END;
```

4. This trigger fires when a pantry’s contact information is updated. It creates a new table called “PanUpdates” and provides a framework for updating the contact name and email.

```
drop trigger pantry_deleteTrigger;
drop table PanUpdates;
create table PanUpdates (
newContact varChar2(50),
oldContact varChar2(50),
newEmail varChar2(50),
oldEmail varChar2(50),
delDate date
);

create or replace trigger pantry_deleteTrigger
BEFORE update on pantry for each row

BEGIN
insert into PanUpdates values (
:new.contact, :old.contact, :new.Email, :old.email, sysdate);
END;
```

5. This query piggybacks onto the previous query and updates the contact name and email of the selected tuple from the Pantry table.

```
update pantry
set contact = 'Scharmel', email = 'scharmel@yahoo.com'
where email = 'brittany@yahoo.com';

select * from pantry;
select * from PanUpdates;
```

6. This trigger prevents anyone from deleting a tuple from the table PANTRY and raises an application error.

```
create or replace trigger pantry_deleteTrigger BEFORE delete on pantry
for each row

BEGIN
raise_application_error(-20001,'Failed operation. Unauthorized deletion of tuples
in PANTRY table');
END;
```

7. This query pulls the quantity of produces (in pounds) donated to any specified pantry. It will likely be the case that it is best to use one standard unit of measurement, such as pounds, rather than including ounces and such as well. Even though this may be slightly more challenging on the data entry side of the operation, it will prove much easier to discover this kind of information.

```
set serveroutput on;
DECLARE
na integer :=0;
nb varChar2(50);

BEGIN
Select count (*) quantity
into na
from DONATES
where pantry_id = 1;
select pantry_name
into nb
from PANTRY
where pantry_id = 1;

   if (na > 0) then
         DBMS_OUTPUT.PUT_LINE('There have been a total of ' || na || ' pounds
         donated to ' || nb || '.');
   else
      DBMS_OUTPUT.PUT_LINE('There have been no donations.');
   end if;
END;
```

8. This query pulls the all-time quantity of produce donated by Sarah Facen. Another challenge would be to confine the numbers to a period of time.

```
set serveroutput on;

DECLARE
na integer :=0;
nb varChar2(30);
BEGIN
Select count (*) quantity
into na
from DONATES
where id = 100;

select name
into nb
from GROWERS
where id = 100;

   if (na > 0) then
      DBMS_OUTPUT.PUT_LINE('Grower ' || nb || ' has donated a total of ' || na || '
pounds of produce.');
   else
      DBMS_OUTPUT.PUT_LINE('' || nb || ' has made no donations.');
   end if;
END;
```