A Database Design Proposal for

All Angels Academy

By: Megan O'Halloran

Table of Contents

Executive Summary

Objective

Approximately 7.6 million animals are left at a shelter, each year. Of those 7.6 million, 3.4 million are euthanized to deal with the capacity. All Angels Academy is a company that refuses to euthanize its animals. One of the causes animals are not taken in from shelters is because families do not feel comfortable adopting from a shelter. The way we plan to fix this problem is by raising awareness to families by providing both a summer camp for children and a kennel for animals.

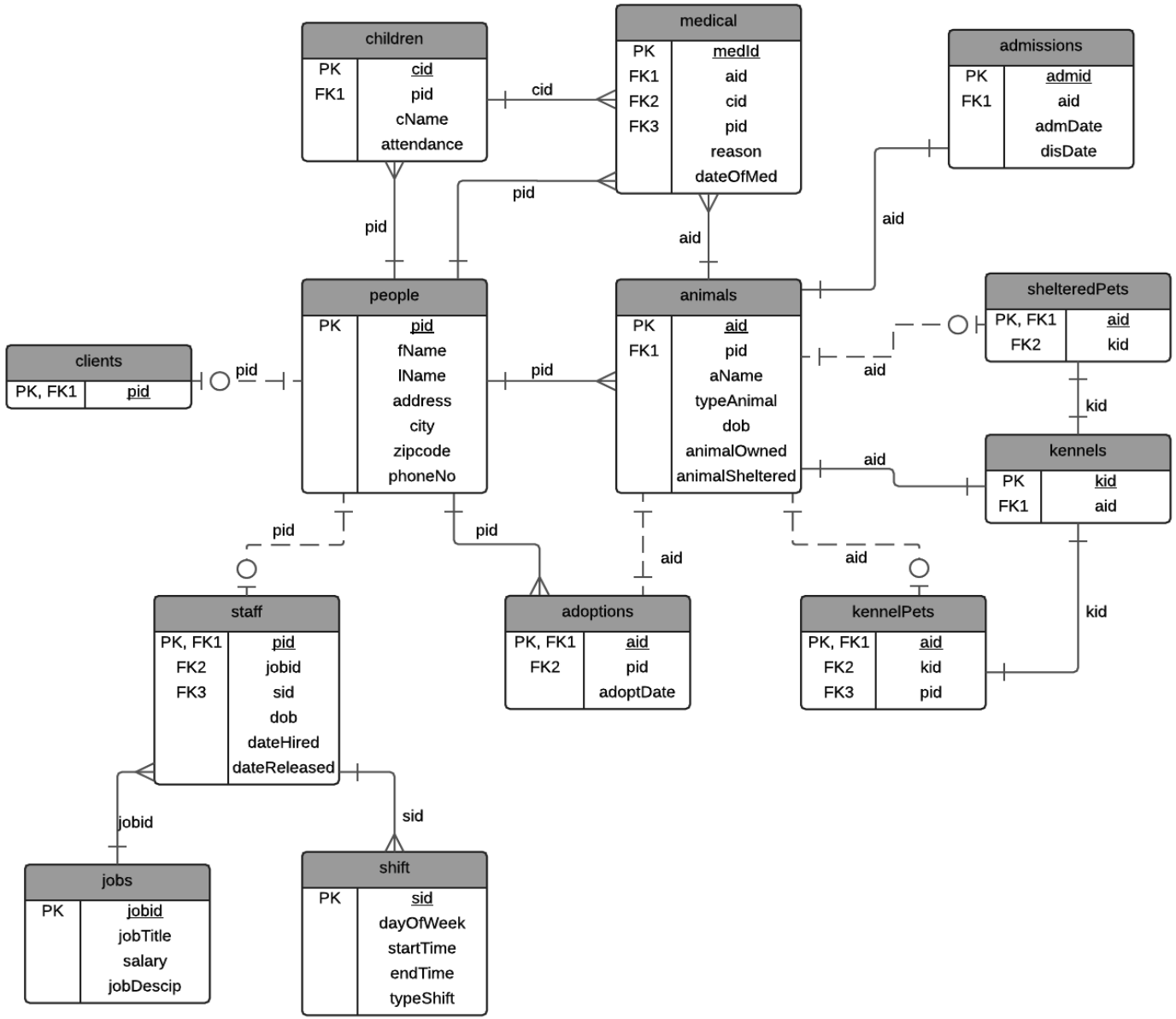
All Angels Academy needs a database that will keep track of their animals, sheltered and owned, as well as the children that are attending the camp. In addition, they need to keep track of their staff and clients, along with the jobs and shifts assigned. They would also like to keep records of all medical treatments for every animal or child that receives care.

Reports of salary, clients, and sheltered animals will be created so that the user can easily view the information. Procedures to facilitate adoptions, discharges, admissions, and animals present will also be implemented into the database system. Finally, views of the weekly schedule, and children and animals present will be provided.

Overview

The purpose of this document is to outline a database system to record information such as staff, clients, children, animals, kennels, admissions, adoptions, jobs, shift, and medical. The tables of this database will be presented, followed by how each of the tables will be created, along with sample data. Views, procedures, triggers, reports, and security will also be presented throughout this proposal. Details of the implementation are provided at the end of the proposal, as well as known problems and future enhancements.

Entity-Relationship Diagram



ALL ANGELS ACADEMY

People Table

The people table contains names, address, and phone numbers of all the people involved with the company including clients and staff.

```
CREATE TABLE people (
        pid char(4) NOT NULL,
        fName text NOT NULL,
        lName text NOT NULL,
        address text NOT NULL,
        city text NOT NULL,
        zipcode integer NOT NULL,
        phoneNo text NOT NULL,
        primary key(pid)
);
```

Functional Dependencies:

(pid) → fName, lName, address, city, zipcode, phoneNo

Sample Data:

	pid character(4)	fname text	lname text	address text	city text	zipcode integer	phoneno text
1	p001	Megan	Ohalloran	20 Give Me An A Dr	Poughkeepsie	12603	(845)112-2334
2	p002	Daniel	Pafumi	15 Harley Davison Ln	Hopewell Junction	12602	223-3445
3	p003	Chrissy	Turner	2 Many Dogs Circle	Poughquag	12603	556-6778
4	p004	Liz	Jones	13 Elizabeth Ln	Wappengers Falls	12601	(845)889-1010
5	p005	Joyce	Smith	50 Felines Ct	Poughkeepsie	12603	101-0111
6	p006	Rob	Bevilacqua	23 Fish Dr	Beacon	12602	111-1122
7	p007	Alan	Labouseur	007 Sean Connery Ln	Poughquag	12603	(845)266-3007

ALL ANGELS ACADEMY

Jobs Table

The jobs table contains a list of all the jobs, along with the salary and a description.

```
CREATE TABLE jobs (
        jobid char(4) NOT NULL,
        jobTitle text NOT NULL,
        salary money NOT NULL,
        jobDescrip text NOT NULL,
        primary key(jobid)
);
```

Functional Dependencies:

(jobid) →jobTitle, salary, jobDescrip

Sample Data:

	jobid character(4)	jobtitle text	salary integer	jobdescrip text
1	j001	Vet	81460	1 position, 1st shift, no wkend, on call
2	j002	Nurse	62500	2 positions, 1st and 2nd shift, no wkend, on call
3	j003	Receptionist	25200	1 position, 1st shift, no wkend
4	j004	Child Care	40000	2 positions, 1st shift, no wkend
5	j005	Animal Care	40000	6 positions, 3 shifts, wkend
6	j006	Office	35700	1 position, 1st shift, no wkend
7	j007	Maintenance	40000	1 position, 1st shift, no wkend, on call
8	j008	Camp Counselor	18080	3 positions, 1st shift, no wkend
9	j009	Volunteer	0	Unlimited positions, all shifts, wkend

Shift Table

The shift table contains six types of shifts for different days of the week, along with a start and end time.

```
CREATE TABLE shift (
        sid char(4) NOT NULL,
        dayOfWeek text NOT NULL,
        startTime time NOT NULL,
        endTime time NOT NULL,
        typeShift text NOT NULL,
        primary key(sid)
);
```

Functional Dependencies:

(sid) →dayOfWeek, startTime, endTime, typeShift

Sample Data:

	sid character(4)	dayofweek text	starttime time without time zone	endtime time without time zone	typeshift text
1	s001	Monday - Friday	08:30:00	16:30:00	Week First Shift
2	s002	Monday - Friday	16:30:00	00:30:00	Week Second Shift
3	s003	Monday - Friday	00:30:00	08:30:00	Week Third Shift
4	s004	Saturday - Sunday	08:30:00	16:30:00	Weekend First Shift
5	s005	Saturday - Sunday	16:30:00	00:30:00	Weekend Second Shift
6	s006	Saturday - Sunday	00:30:00	08:30:00	Weekend Third Shift

Staff Table

The staff table contains a list of the staff along with their job, shift, birthday, date they were hired, and date if they were released.

```
CREATE TABLE staff (
        pid char(4) NOT NULL references people(pid),
        jobid text NOT NULL references jobs(jobid),
        dob date NOT NULL,
        dateHired date NOT NULL,
        dateReleased date,
        sid char(4) NOT NULL references shift(sid),
        primary key(pid),
        foreign key(sid) references shift(sid),
        foreign key(jobid) references jobs(jobid)
);
```

Functional Dependencies:

(pid) → jobid, sid, dob, dateHired, dateReleased

Sample Data:

	pid character(4)	jobid text	dob date	datehired date	datereleased date	sid character(4)
1	p001	j001	1985-07-09	2013-01-03		s001
2	p003	j003	1995-08-01	2015-02-06		s001
3	p005	j004	1986-10-21	2014-04-20		s001
4	p006	j007	1986-06-14	2012-08-15		s001
5	p010	j008	1993-02-26	2013-09-01		s001
6	p012	j002	1982-03-13	2013-02-22		s001
7	p013	j002	1981-09-09	2013-02-20		s002
8	p015	j004	1986-10-29	2014-03-18		s001

Children Table

The children table contains a list of all the children that have attended the camp, including their name, gender, birthday, attendance and a reference to the parent information.

```
CREATE TABLE children (
        cid char(4) NOT NULL,
        cName text NOT NULL,
        cGender char(1) NOT NULL CHECK(cGender = 'M'
                                       OR cGender = 'F'),
        dob date NOT NULL,
        pid char(4) references people(pid),
        attendance text,
        primary key(cid),
        foreign key(pid) references people(pid)
);
```

Functional Dependencies:

(cid) → cName, cGender, dob, pid, attendance

Sample Data:

	cid character(4)	cname text	dob date	pid character(4)	attendance text
1	c001	Sarah	2005-06-01	p002	attending
2	c002	Taylor	2010-08-14	p002	attending
3	c003	Peyton	2012-03-24	p008	
4	c004	Joey	2013-07-15	p009	attending
5	c005	James	2006-07-01	p007	
6	c006	Bond	2010-03-02	p007	attending

Animals Table

The animals table contains a list of all the animals that used the facility, including their name, type, birthday, if they are owned or sheltered, and a reference to the owner's information, if owned.

```
CREATE TABLE animals (
        aid char(4) NOT NULL,
        aName text NOT NULL,
        typeAnimal text NOT NULL,
        dob date NOT NULL,
        pid char(4) references people(pid),
        animalOwned boolean NOT NULL,
        animalSheltered boolean NOT NULL,
        primary key(aid),
        foreign key(pid) references people(pid)
);
```

Functional Dependencies:

(aid) →aName, typeAnimal, dob, pid, animalOwned, animalSheltered

Sample Data:

	aid character(4)	aname text	typeanimal text	dob date	pid character(4)	animalowned boolean	animalsheltered boolean
1	a001	Oreo	dog	2000-03-17	p008	t	f
2	a002	Tater	dog	2006-04-15	p004	t	f
3	a003	Megladon	cat	2010-08-23	p009	t	f
4	a004	Georgey	rabbit	2014-09-20	p004	t	f
5	a005	Charles	cat	2012-10-11	p001	t	f
6	a006	Nana	dog	2013-12-01	p011	t	f

Clients Table

The clients table contains a list of all the clients that have used the facility.

```
CREATE TABLE clients (
        pid char(4) NOT NULL references people(pid),
        primary key(pid)
);
```

Functional Dependencies:

(pid) → N/A

Sample Data:

	pid character(4)
1	p002
2	p008
3	p009
4	p007
5	p004
6	p001
7	p011
8	p014

Admissions Table

The admissions table contains a list of all the animals that have been admitted into facility, including a reference to the animal information, admission date, and a discharge date.

```
CREATE TABLE admissions (
        admid SERIAL,
        aid char(4) NOT NULL references animals(aid),
        admDate date NOT NULL,
        disDate date,
        primary key(admid),
        foreign key(aid) references animals(aid)
);
```

Functional Dependencies:

(admid) →aid, admDate, disDate

Sample Data:

	admid integer	aid character(4)	admdate date	disdate date
1	1	a003	2013-02-15	2013-02-27
2	2	a002	2016-03-04	
3	3	a004	2016-03-04	
4	4	a001	2014-10-05	2014-10-25
5	5	a005	2015-12-20	2015-12-28
6	6	a006	2016-01-14	2016-03-01
7	7	a007	2016-02-28	
8	8	a008	2016-02-13	
9	9	a009	2016-03-13	
10	10	a010	2016-03-03	

Adoptions Table

The adoptions table contains a list of all the sheltered animals that have been adopted, along with a reference to the owner's information and the date of adoption.

```
CREATE TABLE adoptions (
        aid char(4) NOT NULL references animals(aid),
        pid char(4) NOT NULL references people(pid),
        adoptDate date NOT NULL DEFAULT current_date,
        primary key(aid),
        foreign key(pid) references people(pid)
);
```

Functional Dependencies:

(aid) →pid, adoptDate

Sample Data:

	aid character(4)	pid character(4)	adoptdate date
1	a006	p011	2016-03-01

Kennels Table

The kennels table contains a list of all the kennels, along with a reference to the animal's information.

```
CREATE TABLE kennels(
        kid char(4),
        aid char(4),
        primary key(kid),
        foreign key(aid) references animals(aid)
);
```

Functional Dependencies:

(kid) → aid

Sample Data:

	kid character(4)	aid character(4)
1	k001	a002
2	k002	a004
3	k003	a007
4	k004	a009
5	k005	a010
6	k006	a008
7	k007	
8	k008	
9	k009	

ShelteredPets Table

The shelteredPets table contains a list of all the animals that are currently sheltered at the facility, without an owner, along with a reference to the animal's information and its kennel.

```
CREATE TABLE shelteredPets (
        aid char(4) NOT NULL references animals(aid),
        kid char(4) NOT NULL references kennels(kid),
        primary key(aid),
        foreign key(kid) references kennels(kid)
);
```

Functional Dependencies:

(aid) →kid

Sample Data:

	aid character(4)	kid character(4)
1	a007	k003
2	a009	k004
3	a010	k005

KennelPets Table

The kennelPets table contains a list of all the animals that are owned, along with a reference to the owner's information, the pet's information, and kennel the animal is in.

```
CREATE TABLE kennelPets (
        aid char(4) NOT NULL references animals(aid),
        kid char(4) NOT NULL references kennels(kid),
        pid char(4) NOT NULL references people(pid),
        primary key(aid),
        foreign key(kid) references kennels(kid),
        foreign key(pid) references people(pid)
);
```

Functional Dependencies:

(aid) →kid, pid

Sample Data:

	aid character(4)	kid character(4)	pid character(4)
1	a002	k001	p004
2	a004	k002	p004
3	a008	k006	p014

Medical Table

The medical table contains a list of all the medical treatments administered to either the animal or the child, the reason for visit, and date of care, along with a reference to the parent or owner.

```
CREATE TABLE medical (
        medid SERIAL,
        aid char(4) references animals(aid),
        cid char(4) references children(cid),
        reason text NOT NULL,
        dateOfMed date NOT NULL DEFAULT current_date,
        pid char(4) references people(pid),
        primary key(medid),
        foreign key(aid) references animals(aid),
        foreign key(cid) references children(cid),
        foreign key(pid) references people(pid)
);
```

Functional Dependencies:

(medid) →aid, cid, reason, dateOfMed, pid

Sample Data:

	medid character(4)	aid character(4)	cid character(4)	reason text	dateofmed date	pid character(4)
1	m001	a003		sick	2013-02-18	p009
2	m002	a006		check-up	2016-01-14	
3	m003	a007		check-up	2016-02-27	
4	m004		c003	sick	2012-03-28	p008
5	m005		c006	cut	2010-03-10	p007

View: WeeklySchedule

The weeklySchedule view will help staff know who is working on which shift.

```
CREATE VIEW weeklySchedule AS
        SELECT dayofweek, starttime, endtime, jobtitle, fname, lname, phoneno
        FROM shift, jobs, people, staff
        WHERE shift.sid = staff.sid AND jobs.jobid = staff.jobid AND people.pid = staff.pid
        ORDER BY shift.sid, jobs.jobid;
```

Sample Data:

	dayofweek text	starttime time without time zone	endtime time without time zone	jobtitle text	fname text	lname text	phoneno text
1	Monday - Friday	08:30:00	16:30:00	Vet	Megan	Ohalloran	(845)112-2334
2	Monday - Friday	08:30:00	16:30:00	Nurse	Sarah	Grunbok	(845)161-16177
3	Monday - Friday	08:30:00	16:30:00	Receptionist	Chrissy	Turner	556-6778
4	Monday - Friday	08:30:00	16:30:00	Child Care	Joyce	Smith	101-0111
5	Monday - Friday	08:30:00	16:30:00	Child Care	Liz	Schlusser	(845)191-9202
6	Monday - Friday	08:30:00	16:30:00	Animal Care	Ramona	Rodriguez	(845)202-0212
7	Monday - Friday	08:30:00	16:30:00	Office	Elizabeth	Sager	(845)242-42522
8	Monday - Friday	08:30:00	16:30:00	Maintenance	Rob	Bevilacqua	111-1122
9	Monday - Friday	08:30:00	16:30:00	Camp Counselor	Kat	Smelly	141-4155
10	Monday - Friday	08:30:00	16:30:00	Camp Counselor	Neil	Rosenfeild	(845)252-2526
11	Monday - Friday	08:30:00	16:30:00	Camp Counselor	Ryan	Cripps	262-6272
12	Monday - Friday	08:30:00	16:30:00	Volunteer	James	Dean	(845)272-7282
13	Monday - Friday	16:30:00	00:30:00	Nurse	Jackie	Morris	171-7188
14	Monday - Friday	16:30:00	00:30:00	Animal Care	Jess	Saygeh	(845)212-1222
15	Monday - Friday	16:30:00	00:30:00	Volunteer	Johnny	Depp	282-8292
16	Monday - Friday	00:30:00	08:30:00	Animal Care	Dillan	Smith	222-2232
17	Saturday - Sunday	08:30:00	16:30:00	Animal Care	Jake	Water	(845)232-32422
18	Saturday - Sunday	08:30:00	16:30:00	Volunteer	Chris	McClain	(845)303-0313

View: ChildrenPresent

The childrenPresent view displays all of the children that are currently attending the camp.

```
CREATE VIEW childrenPresent AS
        SELECT cname AS childsName, fname AS parentsFirtName, lname as parentsLastName, phoneno
        FROM children
        INNER JOIN people
                ON children.pid = people.pid
        WHERE attendance = 'attending'
        ORDER BY lname ASC;
```

Sample Data:

	childsname text	parentsfirtname text	parentslastname text	phoneno text
1	Joey	Corey	Doe	131-3144
2	Bond	Alan	Labouseur	(845)266-3007
3	Taylor	Daniel	Pafumi	223-3445
4	Sarah	Daniel	Pafumi	223-3445

View: OwnedPetsPresent

The ownedPetsPresent view displays all of the owned pets that are currently kenneled at the facility.

```
CREATE VIEW ownedPetsPresent AS
        SELECT typeanimal, aname AS petName, kid AS kennelNo, fname AS ownerFirstName, lname AS ownerLastName, phoneno
        FROM animals
        INNER JOIN people
                ON people.pid = animals.pid
        INNER JOIN kennels
                ON animals.aid = kennels.aid
                WHERE animals.aid IN(SELECT aid
                                FROM admissions
                                WHERE disdate IS NULL)
        ORDER BY typeanimal ASC;
```

Sample Data:

	typeanimal text	petname text	kennelno character(4)	ownerfirstname text	ownerlastname text	phoneno text
1	cat	Rocky	k007	Megan	Ohalloran	(845)112-2334
2	dog	Tater	k001	Liz	Jones	(845)889-1010
3	rabbit	Georgey	k002	Liz	Jones	(845)889-1010

View: OrphanedPetsPresent

The orphanedPetsPresent view displays all of the unowned pets that are currently sheltered in the facility.

```
CREATE VIEW orphanedPetsPresent AS
        SELECT typeanimal, aname AS petName, kid AS kennelNo
        FROM animals, kennels, admissions
        WHERE animals.aid = kennels.aid
          AND animals.aid = admissions.aid
          AND animals.pid IS NULL
          AND disdate IS NULL
         ORDER BY typeanimal ASC;
```

Sample Data:

	typeanimal text	petname text	kennelno character(4)
1	cat	Mozzel	k003
2	cat	Tabatha	k004
3	dog	Scruffy	k005
4	rabbit	Nibbles	k008

Report: Salaries

This report returns the monthly and yearly salaries for all staff members.

```
SELECT jobtitle, fname AS employee_first_name, lname AS employee_last_name,
       (salary/12) AS monthly_salary, salary AS yearly_salary
FROM jobs, people, staff
WHERE jobs.jobid = staff.jobid
  AND staff.pid = people.pid
GROUP BY jobtitle, people.fname, people.lname, jobs.salary
ORDER BY salary DESC;
```

Sample Data:

	jobtitle text	employee_first_name text	employee_last_name text	monthly_salary money	yearly_salary money
1	Vet	Megan	Ohalloran	$6,788.33	$81,460.00
2	Nurse	Jackie	Morris	$5,208.33	$62,500.00
3	Nurse	Sarah	Grunbok	$5,208.33	$62,500.00
4	Animal Care	Blake	Shelton	$3,333.33	$40,000.00
5	Animal Care	Dillan	Smith	$3,333.33	$40,000.00
6	Animal Care	Jake	Water	$3,333.33	$40,000.00
7	Animal Care	Jess	Saygeh	$3,333.33	$40,000.00
8	Animal Care	Maria	McCue	$3,333.33	$40,000.00
9	Animal Care	Ramona	Rodriguez	$3,333.33	$40,000.00
10	Child Care	Joyce	Smith	$3,333.33	$40,000.00

Report: Clients with Animals

This report returns the percent of clients that currently have an animal in the kennel.

```
SELECT TRUNC(
        CAST(
                (SELECT COUNT(aid)
                  FROM animals
                 WHERE pid != '') AS DECIMAL(5,2))
                /
                (SELECT COUNT(pid)
                   FROM clients)
        * 100) AS Percent_of_clients_with_animals;
```

Sample Data:

	percent_of_clients_with_animals numeric
1	80

Report: Animals in Shelter Over a Month

This report returns the number of animals that have been in the shelter for over a month.

```
SELECT aname AS animal_name, kid AS kennel, age(current_date, admDate) AS length_of_stay
FROM admissions
INNER JOIN animals
        ON animals.aid = admissions.aid
INNER JOIN kennels
        ON kennels.aid = admissions.aid
WHERE date_part('month', age(current_date, admDate)) >= 1
        AND disDate IS NULL;
```

Sample Data:

	animal_name text	kennel character(4)	length_of_stay interval
1	Tater	k001	1 mon 12 days
2	Georgey	k002	1 mon 12 days
3	Mozzel	k003	1 mon 17 days
4	Tabatha	k004	1 mon 3 days
5	Scruffy	k005	1 mon 13 days

Stored Procedure: Adopted_Pet()

In adopted_Pet(), when an animal is adopted it is given a discharge date, given an owner, removed from a kennel, and removed from sheltered pets. This is triggered by the adopted_Pet trigger.

```
CREATE OR REPLACE FUNCTION adopted_pet() RETURNS trigger AS
$BODY$
BEGIN
        UPDATE admissions
        SET disdate = adoptdate
        FROM adoptions
        WHERE adoptions.aid = admissions.aid;

        UPDATE animals
        SET
           pid = adoptions.pid,
           animalOwned = true,
           animalSheltered = false
        FROM adoptions
        WHERE animals.aid = adoptions.aid;

        UPDATE kennels
        SET aid = NULL
        FROM adoptions
        WHERE kennels.aid = adoptions.aid;

        DELETE FROM shelteredPets using adoptions
        WHERE shelteredPets.aid = adoptions.aid;

        RETURN new;
END;
$BODY$
LANGUAGE plpgsql;
```

Sample data for this procedure will be displayed in its paired trigger, which is in the following section.

Stored Procedure: Animals_Present()

Animals_Present(), given an animal type, the user can display a list and their relevant information, such as name, kennel ID, and owner if applicable.

```
CREATE OR REPLACE FUNCTION animals_present(IN animaltype text)
RETURNS TABLE(kennel_id char(4), animal_name text, animal_type text, owner_fName text, owner_lName text, phoneNo text) AS
$BODY$
BEGIN
        RETURN QUERY SELECT kennels.kid, animals.aname, animals.typeanimal, people.fname, people.lname, people.phoneNo
                        FROM kennels, animals LEFT OUTER JOIN people
                                ON animals.pid = people.pid
                        WHERE animaltype = animals.typeanimal
                                AND kennels.aid = animals.aid
                        ORDER BY kennels.kid ASC;
END;
$BODY$
LANGUAGE plpgsql;
```

Sample Data:

	animals_present record
1	(k002,Georgey,rabbit,Liz,Jones,"(845)889-1010")

ALL ANGELS ACADEMY

Stored Procedure: New_Animal()

New_Animal() checks if the animal is owned or sheltered. It updates the tables accordingly, as well as the kennels, admissions, and medical tables. This is triggered by the new_Animal trigger.

```
CREATE OR REPLACE FUNCTION new_animal() RETURNS trigger AS
$BODY$
DECLARE kidAssigned char(4) DEFAULT NULL;
BEGIN
        INSERT INTO admissions(aid, admdate)
                VALUES(new.aid, current_date);

        UPDATE kennels
        SET aid = new.aid
        FROM animals
        WHERE kid in (SELECT MIN(kid)
                        FROM kennels
                        WHERE aid IS NULL);

        kidAssigned = (SELECT kid
                         FROM kennels
                         WHERE aid = new.aid);

        IF new.animalOwned = TRUE THEN
                INSERT INTO kennelPets(aid, kid, pid)
                         VALUES(new.aid, kidAssigned, new.pid);
        END IF;

        IF new.animalSheltered = TRUE THEN
                INSERT INTO shelteredPets(kid, aid)
                         VALUES(kidAssigned, new.aid);
                INSERT INTO medical(aid, reason, dateOfMed, pid)
                         VALUES(new.aid, 'check-up', current_date, new.pid);
        END IF;

        RETURN new;
END;
$BODY$
LANGUAGE plpgsql;
```

Sample data for this procedure will be displayed in its paired trigger.

Stored Procedure: Discharge_Animal()

Discharge_Animal() removes the animal from the kennels table and the kenneled animals table. This is triggered by the discharge_Animal trigger.

```
CREATE OR REPLACE FUNCTION discharge_animal() RETURNS trigger AS
$BODY$
BEGIN
        DELETE FROM kennelPets
        WHERE kennelPets.aid = new.aid;

        UPDATE kennels
        SET aid = NULL
        WHERE kennels.aid = new.aid;

        RETURN new;
END;
$BODY$
LANGUAGE plpgsql;
```

Sample data for this procedure will be displayed in its paired trigger, which is in the following section.

Trigger: Adopted_Pet

Adopted_Pet is triggered when an animal is added to the adoptions table, which then executes the stored procedure Adopted_Pet().

```
CREATE TRIGGER adopted_pet
AFTER INSERT ON adoptions
FOR EACH ROW
EXECUTE PROCEDURE adopted_pet();
```

Triggered by:

```
INSERT INTO adoptions(aid, pid, adoptdate)
        VALUES('a007', 'p029', '3-28-2016');
```

Sample Data:

Admissions Table Before:

	admid integer	aid character(4)	admdate date	disdate date
7	7	a007	2016-02-28	

Admissions Table After:

	admid integer	aid character(4)	admdate date	disdate date
1	7	a007	2016-02-28	2016-03-28

ALL ANGELS ACADEMY

Trigger: Adopted_Pet(cont)

Animals Table Before:

	aid character(4)	aname text	typeanimal text	dob date	pid character(4)	animalowned boolean	animalsheltered boolean
7	a007	Mozzel	cat	2014-01-01		f	t

Animals Table After:

	aid character(4)	aname text	typeanimal text	dob date	pid character(4)	animalowned boolean	animalsheltered boolean
1	a007	Mozzel	cat	2014-01-01	p029	t	f

Kennels Table Before:

	kid character(4)	aid character(4)
3	k003	a007

Kennels Table After:

	kid character(4)	aid character(4)
1	k003	

Trigger: New_Animal

When a new animal is added to the animals table, new_Animal is triggered and executes the stored procedure new_Animal().

```
CREATE TRIGGER new_animal
AFTER INSERT ON animals
FOR EACH ROW
EXECUTE PROCEDURE new_animal();
```

Triggered by:

```
INSERT INTO animals(aid, typeAnimal, dob, aName, animalOwned, animalSheltered)
        VALUES('a013', 'rabbit', '3-9-2015', 'Fluffinutter', false, true);
```

Sample Data:

Admissions Table Before:

	admid integer	aid character(4)	admdate date	disdate date
1	1	a003	2013-02-15	2013-02-27
2	2	a002	2016-03-04	
7	7	a007	2016-02-28	2016-03-28
8	8	a008	2016-02-13	2016-04-08
9	9	a009	2016-03-13	
10	10	a010	2016-03-03	
11	11	a011	2016-04-02	
12	12	a012	2016-04-02	

Admissions Table After:

	admid integer	aid character(4)	admdate date	disdate date
1	1	a003	2013-02-15	2013-02-27
2	2	a002	2016-03-04	
7	7	a007	2016-02-28	2016-03-28
8	8	a008	2016-02-13	2016-04-08
9	9	a009	2016-03-13	
10	10	a010	2016-03-03	
11	11	a011	2016-04-02	
12	12	a012	2016-04-02	
13	13	a013	2016-04-16	

Trigger: New_Animal (Cont.)

Kennels Table Before:

	kid character(4)	aid character(4)
1	k001	a002
2	k002	a004
3	k003	
4	k004	a009
5	k005	a010
6	k006	
7	k007	a011
8	k008	a012
9	k009	

Kennels Table Before:

	kid character(4)	aid character(4)
1	k001	a002
2	k002	a004
3	k003	a013
4	k004	a009
5	k005	a010
6	k006	
7	k007	a011
8	k008	a012
9	k009	

ShelteredPets Table Before:

	aid character(4)	kid character(4)
1	a009	k004
2	a010	k005
3	a012	k008

ShelteredPets Table After:

	aid character(4)	kid character(4)
1	a009	k004
2	a010	k005
3	a012	k008
4	a013	k003

Medical Table Before:

	medid integer	aid character(4)	cid character(4)	reason text	dateofmed date	pid char
7	7	a010		check-up	2016-03-03	
8	8	a012		check-up	2016-04-02	

Medical Table After:

	medid integer	aid character(4)	cid character(4)	reason text	dateofmed date	pid char
7	7	a010		check-up	2016-03-03	
8	8	a012		check-up	2016-04-02	
9	9	a013		check-up	2016-04-16	

Trigger: Discharge_Animal

Discharge_Animal is triggered when a discharge date is put in the admissions table, it executes the discharge_animal() procedure.

```
CREATE TRIGGER discharge_animal
AFTER UPDATE ON admissions
FOR EACH ROW
EXECUTE PROCEDURE discharge_animal();
```

Triggered by:

```
UPDATE admissions SET disdate = NULL WHERE aid = 'a008';
```

Sample Data:

KennelPets Table Before:

	aid character(4)	kid character(4)	pid character(4)
1	a002	k001	p004
2	a004	k002	p004
3	a008	k006	p014

KennelPets Table After:

	aid character(4)	kid character(4)	pid character(4)
1	a002	k001	p004
2	a004	k002	p004

Trigger: Discharge_Animal(Cont)

Kennels Table Before:

	kid character(4)	aid character(4)
1	k001	a002
2	k002	a004
3	k003	a007
4	k004	a009
5	k005	a010
6	k006	a008

Kennels Table After:

	kid character(4)	aid character(4)
1	k001	a002
2	k002	a004
3	k003	a013
4	k004	a009
5	k005	a010
6	k006	

Security: Admin

Security was given to the admin, also known as the owner, which is able to make any change to any tables.

```
REVOKE ALL PRIVILEGES ON admissions FROM DB_admin;
REVOKE ALL PRIVILEGES ON animals FROM DB_admin;
REVOKE ALL PRIVILEGES ON children FROM DB_admin;
REVOKE ALL PRIVILEGES ON kennels FROM DB_admin;
REVOKE ALL PRIVILEGES ON people FROM DB_admin;
REVOKE ALL PRIVILEGES ON adoptions FROM DB_admin;
REVOKE ALL PRIVILEGES ON clients FROM DB_admin;
REVOKE ALL PRIVILEGES ON kennelPets FROM DB_admin;
REVOKE ALL PRIVILEGES ON shelteredPets FROM DB_admin;
REVOKE ALL PRIVILEGES ON jobs FROM DB_admin;
REVOKE ALL PRIVILEGES ON medical FROM DB_admin;
REVOKE ALL PRIVILEGES ON shift FROM DB_admin;
REVOKE ALL PRIVILEGES ON staff FROM DB_admin;

CREATE ROLE DB_admin;
GRANT ALL ON ALL TABLES
IN SCHEMA PUBLIC
TO DB_admin;
```


Security: Medical Staff

The security that was given to the medical staff is the ability to make a select, insert, and update on specified tables. The medical staff is the vets and nurses.

```
REVOKE ALL PRIVILEGES ON animals FROM Medical_staff;
REVOKE ALL PRIVILEGES ON children FROM Medical_staff;
REVOKE ALL PRIVILEGES ON medical FROM Medical_staff;
REVOKE ALL PRIVILEGES ON people FROM Medical_staff;
REVOKE ALL PRIVILEGES ON kennels FROM Medical_staff;
REVOKE ALL PRIVILEGES ON admissions FROM Medical_staff;
REVOKE ALL PRIVILEGES ON kennelPets FROM Medical_staff;
REVOKE ALL PRIVILEGES ON shelteredPets FROM Medical_staff;

CREATE ROLE Medical_staff;
GRANT SELECT ON animals, children, medical,
                people, kennels, admissions,
                kennelPets, shelteredPets
TO Medical_staff;

GRANT UPDATE ON medical
TO Medical_staff;

GRANT INSERT ON medical
TO Medical_staff;
```

Security: Office Staff

The security given to the office staff is the ability to select on any table. As well as insert, update, and delete on the specified tables. The office staff is the receptionist and the office staff.

```
REVOKE ALL PRIVILEGES ON admissions FROM Office_staff;
REVOKE ALL PRIVILEGES ON animals FROM Office_staff;
REVOKE ALL PRIVILEGES ON children FROM Office_staff;
REVOKE ALL PRIVILEGES ON kennels FROM Office_staff;
REVOKE ALL PRIVILEGES ON people FROM Office_staff;
REVOKE ALL PRIVILEGES ON adoptions FROM Office_staff;
REVOKE ALL PRIVILEGES ON clients FROM Office_staff;
REVOKE ALL PRIVILEGES ON kennelPets FROM Office_staff;
REVOKE ALL PRIVILEGES ON shelteredPets FROM Office_staff;
REVOKE ALL PRIVILEGES ON jobs FROM Office_staff;
REVOKE ALL PRIVILEGES ON medical FROM Office_staff;
REVOKE ALL PRIVILEGES ON shift FROM Office_staff;
REVOKE ALL PRIVILEGES ON staff FROM Office_staff;

CREATE ROLE Office_staff;
GRANT SELECT ON ALL TABLES
IN SCHEMA PUBLIC
TO Office_staff;

GRANT UPDATE ON admissions, animals, children,
                kennels, people
TO Office_staff;

GRANT INSERT ON admissions, adoptions, animals,
                children, clients, kennelPets,
                shelteredPets, people
TO Office_staff;

GRANT DELETE ON shelteredPets, kennelPets
TO Office_staff;
```


Security: Child Care

The security given to child care is only a select on the specified tables. The child care is child care and camp counselors.

```
REVOKE ALL PRIVILEGES ON children FROM Child_care;
REVOKE ALL PRIVILEGES ON people FROM Child_care;
REVOKE ALL PRIVILEGES ON medical FROM Child_care;
REVOKE ALL PRIVILEGES ON clients FROM Child_care;

CREATE ROLE Child_care;
GRANT SELECT ON children, people,
                medical, clients
TO Child_care;
```


Security: Animal Care

The security given to animal care is only a select on the specified tables. The animal care is the animal care and maintenance crew.

```
REVOKE ALL PRIVILEGES ON admissions FROM Animal_care;
REVOKE ALL PRIVILEGES ON animals FROM Animal_care;
REVOKE ALL PRIVILEGES ON clients FROM Animal_care;
REVOKE ALL PRIVILEGES ON kennelPets FROM Animal_care;
REVOKE ALL PRIVILEGES ON shelteredPets FROM Animal_care;
REVOKE ALL PRIVILEGES ON kennels FROM Animal_care;
REVOKE ALL PRIVILEGES ON medical FROM Animal_care;
REVOKE ALL PRIVILEGES ON people FROM Animal_care;

CREATE ROLE Animal_care;
GRANT SELECT ON admissions, animals, clients,
                kennelPets, shelteredPets,
                kennels, medical, people
TO Animal_care;
```

Implementation Notes:

- Given the complexity of an animal kennel and shelter, not all possible queries were implemented in this project.
- Only the most relevant tables were implemented in this initial database design.
- It is assumed that when a person adopts a pet that they already exist in the people table.

Known Problems:

- There is no functionality to handle if an adopted animal is returned to the facility; it will acquire a new I.D. rather than its old one.

- There is no functionality to handle if a sheltered animal passes away while still under the care of the facility.
- The need for additional tables which will be addressed in future enhancements.

Future Enhancements:

- Some tables to consider having implemented in the future:
  - Suppliers Table
  - Supplies Table
  - Contributors/Donators Table
- Add additional staff positions.
- Add additional shift schedules to accommodate all possibilities.