Database Deisgn Proposal for **Girl Scouts of USA**

brought to you by: Federica Bruno

Table of Contents

Executive Summary

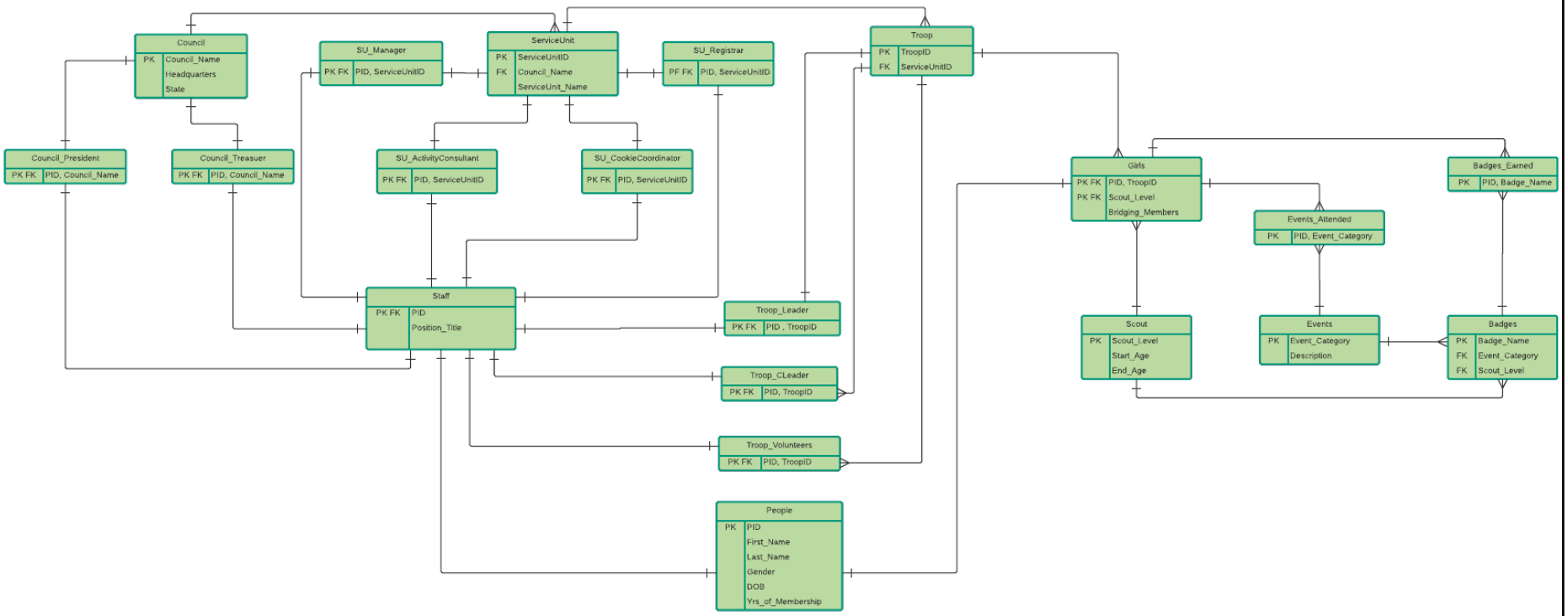
With 2.3 million growing members, over 890,000 volunteers, and over 200 million boxes of cookies sold each year, there is little room for mistakes, especially with the fierce and unmerciful. GSUSA has contacted me to propose a database that would help catalog members, staff, and even distribution of badges to maintain this tight-knit committee running smoothly and efficiently.

To demonstrate the overview of this project, an Entity Relationship Diagram will be first presented, followed by how each individual table will be created. To ensure that the database will be accessible to authorized members only, ideal user rolls will be suggested. Sample output results and explanations have also been provided for views, stored procedures, and triggers.

Like any complex project, there is always room for improvements, and therefore future features and enhancements are suggested at the end. The ultimate goal of this design is to have a full functioning database that can hold a large scale of data and ensure that data is accurate and consistent.

This design was targeted for and tested on PostgreSQL.

Entity Relationship Diagram



Tables

The People table contains all of the common attributes that all members involved in girl scouts share. The table has 11 subtypes: Staff, Council_President, Council_Treasurer, SU_Manager, SU_ActivityConsultant, SU_CookieCoordinator, SU_Registrar, Troop_Leader, Troop_CoLeader, Troop_Volunteers, and Girls.

Create Statement

```
CREATE TABLE People(
    PID char(8) NOT NULL,
    First_Name text NOT NULL,
    Last_Name text NOT NULL,
    Gender char(1) NOT NULL,
    Yrs_of_Membership int,
    DOB date NOT NULL,
   CONSTRAINT check_gender CHECK
(gender = 'M' or gender = 'F'),
   primary key(PID)
);
```

Functional Dependencies

PID→ First_Name, Last_Name, Gender, DOB, Yrs_of_Membership

Sample Data

pid character	first_name text	last_name text	gender character	yrs_of_membership integer	dob date
P125	Casey	Arnold	F	1	2010-10-15
P126	Zoe	Murphy	F	9	2000-04-09
P127	Penelope	Potter	F	3	2006-10-07
P128	Alan	Labouseur	F	7	2005-03-10
P129	Brandie	Vincent	F	9	2000-04-08
P130	Melissa	Tuft	F	12	1973-03-26
P145	Adele	Rakes	F	4	1967-04-18
P146	Angela	Fairfield	F	11	1973-07-05
P147	Mitch	Bourne	M	14	1972-02-14
P148	Richard	Darwin	M	8	1970-06-19
P149	Victoria	Hawkins	F	10	1967-05-11
P150	Toby	Beckham	M	9	1973-12-14
P151	Ken	Nielson	M	2	1977-01-24
P152	Lauren	Clark	F	7	1969-05-16
P153	Jolie	Noel	F	5	1972-04-23
P154	Jake	Robin	M	9	1974-02-04
P155	Steven	Emerson	M	3	1976-11-23

Tables

GSUA is subdivided into state Councils. Every state has more than one council, distinguished by their county. Each council is identified by its Council Name. This database focuses on the Councils of California. This table uses a default value by setting all states as California.

Create Statement

```
CREATE TABLE Council(
    Council_Name text NOT NULL,
    State text DEFAULT California,
    Headquarters text NOT NULL,
   primary key (Council_Name)
);
```

Sample Data

council_name text	state text	headquarters text
Girl Scouts of Central California South	California	Fresno
Girl Scouts of Californias Central Coast	California	Camarillo
Girl Scouts of Northern California	California	San Jose
Girl Scouts of Greater Los Angeles	California	Los Angeles
Girl Scout Council of Orange County	California	Irvine
Girl Scouts San Diego	California	San Diego

Functional Dependencies

Council_Name→ State, Headquarters

Tables

Every Council has one President, identified by its PID and Council_Name. This table is a subtype of both the People and Staff table.

Note: There is a much larger executive board within a Council; however, this database focuses on the 2 main boards members: President and Treasurer

Create Statement

```
CREATE TABLE Council_President(
   PID char(8) NOT NULL references
   People(PID),
   Council_Name text NOT NULL
   references Council(Council_Name),
  primary key(PID)
);
```

Sample Data

pid character	council_name text
P149	Girl Scouts of Northern California
P152	Girl Scout Council of Orange County
P147	Girl Scouts of Greater Los Angeles
P166	Girl Scouts San Diego

No Functional Dependencies

PID, Council_Name→

Tables

Every Council has one Treasurer, identified by its PID and Council_Name. This table is also a subtype of both the People and Staff table

Create Statement

```
CREATE TABLE Council_Treasurer(
    PID char(8) NOT NULL references
    People(PID),
    Council_Name text NOT NULL
    references Council(Council_Name),
  primary key(PID)
);
```

Sample Data

pid character	council_name text
P143	Girl Scouts of Northern California
P161	Girl Scout Council of Orange County
P148	Girl Scouts of Greater Los Angeles
P150	Girl Scouts San Diego

No Functional Dependencies

PID, Council_Name→

Tables

Councils are further subdivided into Service Units, and its name is usually its respectable county. Each Service Unit is identified by a unique 3 digit number.

Create Statement

```
CREATE TABLE ServiceUnit(
   ServiceUnitID int NOT NULL,
   Council_Name text NOT NULL
   references Council(Council_Name),
   ServiceUnit_Name text NOT NULL,
  primary key(ServiceUnitID)
);
```

Functional Dependencies

ServiceUnitID→ Council__Name,
ServiceUnit_Name, County

Sample Data

serviceunitid character varying	council_name text	serviceunit_name text
378	Girl Scouts of Central California South	Annadale
128	Girl Scouts San Diego	Maple Lakes
129	Girl Scouts San Diego	Bloomington
467	Girl Scout Council of Orange County	Northern Woods
468	Girl Scout Council of Orange County	Monticello
469	Girl Scout Council of Orange County	Stone Arch
135	Girl Scouts of Northern California	Shingle Creek
136	Girl Scouts of Northern California	White Water
679	Girl Scouts of Californias Central Coast	Robinsondale
752	Girl Scouts of Greater Los Angeles	Cambridge

Tables

Every Service Unit has one Service Unit Manager, identified by its PID and ServiceUnitID. This table is a subtype of both the People and Staff table.

Note: There are many job positions in Service Units; however this database focuses on are the 4 main staff members: Manager, Activity Consultant, Cookie Coordinator, and Registrar.

Create Statement

```
CREATE TABLE SU_Manager(
    PID char(8) NOT NULL references
    People(PID),
    ServiceUnitID int NOT NULL
    references
    ServiceUnit(ServiceUnitID),
  primary key(PID)
);
```

Sample Data

pid character	serviceunitid character varying
P159	136
P145	467
P158	752
P156	129

No Functional Dependencies

PID, ServiceUnitID →

Tables

Every Service Unit has one Activity Consultant, identified by its PID and ServiceUnitID. This table is a subtype of both the People and Staff table

Create Statement

```
CREATE TABLE SU_ActivityConsultant(
    PID char(8) NOT NULL references
    People(PID),
    ServiceUnitID int NOT NULL
    references
    ServiceUnit(ServiceUnitID),
  primary key(PID)
);
```

Sample Data

pid character	serviceunitid character varying
P155	136
P144	467
P146	752
P157	129

No Functional Dependencies

PID, ServiceUnitID →

Tables

Every Service Unit has one Cookie Coordinator, identified by its PID and ServiceUnitID. This table is a subtype of both the People and Staff table

Create Statement

```
CREATE TABLE SU_CookieCoordinator(
    PID char(8) NOT NULL references
    People(PID),
    ServiceUnitID int NOT NULL
    references
    ServiceUnit(ServiceUnitID),
  primary key(PID)
);
```

Sample Data

pid character	serviceunitid character varying
P160	136
P162	467
P154	752
P153	129

No Functional Dependencies

PID, ServiceUnitID →

Tables

Every Council has one Registrar, identified by its PID and ServiceUnitID. This table is a subtype of both the People and Staff table .

Create Statement

```
CREATE TABLE SU_Registrar(
    PID char(8) NOT NULL references
    People(PID),
    ServiceUnitID int NOT NULL
    references
    ServiceUnit(ServiceUnitID),
  primary key(PID)
);
```

Sample Data

pid character	serviceunitid character varying
P151	136
P161	467
P160	752
P162	129

No Functional Dependencies

PID, ServiceUnitID →

Tables

Finally, Service Units are even further subdivided into local Troops, which consist of the girl scout members themselves. Troops are identified by a unique 4 digit number.

Create Statement

```
CREATE TABLE Troop(
   TroopID varchar(4) NOT NULL,
   ServiceUnitID int NOT NULL
   references
   ServiceUnit(ServiceUnitID),
  primary key(TroopID)
);
```

Functional Dependencies

TroopID→ ServiceUnitID

Sample Data

troopid character varying	serviceunitid character varying
3457	752
1369	752
2689	135
1213	135
2679	135
2346	679
1248	136
3579	469
1602	468
9527	467
4125	129
8176	128
7836	128
6281	378

Tables

Every Troop has one Leader, identified by its PID and ServiceUnitID. This table is a subtype of both the People and Staff table

Create Statement

```
CREATE TABLE Troop_Leader(
    PID char(8) NOT NULL references
    People(PID),
    TroopID int NOT NULL references
    Troop(TroopID),
  primary key(PID)
);
```

No Functional Dependencies

TroopID, ServiceUnitID→

Sample Data

pid character	troopid character varying
P137	1213
P131	1602
P135	1369

Tables

Every Troop has at least one CoLeader, so a Troop can have many CoLeaders. CoLeaders are identified by their PID and ServiceUnitID. This table is a subtype of both the People and Staff table

Create Statement

```
CREATE TABLE Troop_CoLeader(
    PID char(8) NOT NULL references
    People(PID),
    TroopID int NOT NULL references
    Troop(TroopID),
  primary key(PID)
);
```

Sample Data

pid character	troopid character varying
P130	1602
P139	1602
P132	1213
P136	1213
P140	1369

No Functional Dependencies

TroopID, ServiceUnitID→

Tables

A Troop can have many Volunteers as well and they too are identified by their PID and TroopID. This table is a subtype of both the People and Staff table

Create Statement

```
CREATE TABLE Troop_Volunteers(
    PID char(8) NOT NULL references
    People(PID),
    TroopID int NOT NULL references
    Troop(TroopID),
  primary key(PID)
);
```

Sample Data

pid character	troopid character...
P145	4125
P141	4125
P133	4125
P134	8176
P142	8176

No Functional Dependencies

TroopID, ServiceUnitID→

Tables

The Staff table contains all members involved in girl scouts excluding the individual girl themselves. The table is a subtype of the People table and is a superset of: Council_President, Council_Treasurer, SU_Manager, SU_ActivityConsultant, SU_CookieCoordinator, SU_Registrar, Troop_Leader, Troop_CoLeader, & Troop_Volunteers,

Create Statement

```
CREATE TABLE Staff(
    PID char(8) NOT NULL references
    People(PID),
    Position_Title text NOT NULL,
  primary key (PID)
);
```

Functional Dependencies

PID→Position_Title

Sample Data

pid character	position_title text
P150	Council Treasurer
P151	Service Unit Registrar
P152	Council President
P153	Service Unit Cookie Coordinator
P154	Service Unit Cookie Coordinator
P155	Service Unit Activity Consultant
P156	Service Unit Manager
P157	Service Unit Activity Consultant
P158	Service Unit Manager
P159	Service Unit Manager
P160	Service Unit Registrar
P161	Service Unit Registrar
P162	Service Unit Registrar
P163	Service Unit Cookie Coordinator
P164	Council Treasurer
P165	Service Unit Cookie Coordinator

Tables

This table consists the 5 scout levels of Girl Scouts. They are: Daisy (Ages 5-6), Brownie (Ages 7-8), Junior (Ages 9-10), Cadette (Ages 11-13), Senior (Ages 14-15),and Ambassador (Ages 16-18). Girl Scouts move or "bridge" to the next level usually at the end of the school year when they reach their age to advance.

Create Statement

```
CREATE TABLE Scout(
   Scout_Level text NOT NULL,
   Start_Age int NOT NULL,
   End_Age int NOT NULL,
  primary key(Scout_Level)
);
```

Sample Data

scout_level text	start_age smallint	end_age smallint
Daisy	5	6
Brownie	7	8
Junior	9	10
Cadette	11	13
Senior	14	15
Ambassador	16	18

Functional Dependencies

Scout_Level→ Start_Age, End_Age

Tables

The Girls table contains all of the actual Girl Scout members, including Cadette Alan. (sorry Professor, I couldn't help myself.) The girls are associated by their PID, Scout_Level, and TroopID. A girl scout sticks with the same troop throughout their girl scout experience. A 'bridging member' is simply a girl scout who still has advancing scout levels to complete. Ambassador is set to false because it is the last girl scout level a girl can achieve.

Create Statement

```
CREATE TABLE Girls(
    PID char(8) NOT NULL references
    People(PID),
    TroopID int NOT NULL references
    Troop(TroopID),
    Scout_Level text NOT NULL
    references Scout(Scout_Level),
    Bridging_Member boolean,
primary key(PID)
);
```

Functional Dependencies

PID, TroopID → Scout_Level, Bridging_Member

Sample Data

pid character	troopid integer	scout_level text	bridging_member boolean
P115	1369	Daisy	true
P116	2346	Senior	true
P117	2689	Junior	true
P118	1248	Ambassador	false
P119	6281	Cadette	true
P120	2679	Senior	true
P121	4125	Brownie	true
P122	9527	Junior	true
P123	3579	Daisy	true
P124	9527	Junior	true
P125	7836	Brownie	true
P126	8176	Ambassador	false
P127	6281	Cadette	true
P128	6281	Cadette	true
P129	8176	Ambassador	false

Tables

Members of all troops in all scout levels must attend events to earn their badges. All scout levels share common themed events such as selling cookies, but every scout level has a different agenda at events thus having different badges to earn. There are hundreds of events, but the 3 events this database focuses on are: Skill Learning, Cookie Business, and Financial Literacy.

Create Statement

```
CREATE TABLE Events(
   Event_Category text NOT NULL,
   Description text,
  primary key(Event_Category)
);
```

Functional Dependencies

Event _Category→ Description

Data Sample

event_category text	description text
Skill Learning	Girls will learn skills from different areas of life such as: food, athletics, art, ...
Financial Literacy	Girls will learn the basics of credit, how to be frugal with their savings, and ...
Cookie Business	Girls will learn how to set personal and team goals, make important decision...

Tables

The Events_Attended table is a many to many relationship between the events themselves and the girls who attend the events, therefore having no functional dependencies and uses a composite key as its primary key.

Create Statement

```
CREATE TABLE Events_Attended(
   PID char(8) NOT NULL references
   People(PID),
   Event_Category text NOT NULL
   references
   Events(Event_Category),
  primary key(PID, Event_Category)
);
```

No Functional Dependencies

PID, Event_Category →

Sample Data

pid character	event_category text
P100	Skill Learning
P111	Financial Literacy
P114	Skill Learning
P115	Financial Literacy
P117	Cookie Business
P119	Skill Learning
P120	Financial Literacy
P122	Cookie Business
P125	Cookie Business
P126	Skill Learning

Tables

Badges are earned through attending events. Badge names are unique to each scout level, meaning a Daisy cannot earn Brownie badges, because each scout has a different agenda at an event. You do not need to earn all or a certain number of badges in a particular scout to advance to the next scout level.

Create Statement

```
CREATE TABLE Badges(
   Badge_Name text NOT NULL,
   Event_Category text NOT NULL
   references
   Events(Event_Category),
   Scout_Level text NOT NULL
   references Scout(Scout_Level),
  primary key(Badge_Name)
);
```

Functional Dependencies

Badge_Name→ Event_Category,
Scout_Level

Sample Data

badge_name text	event_category text	scout_level text
Baby Sitter	Skill Learning	Cadette
Animal Helper	Skill Learning	Cadette
Website Designer	Skill Learning	Senior
Room Makeover	Skill Learning	Senior
Traveler	Skill Learning	Senior
Womens Health	Skill Learning	Ambassador
Car Care	Skill Learning	Ambassador
Business Etiquette	Skill Learning	Ambassador
Money Counts	Financial Literacy	Daisy
Making Choices	Financial Literacy	Daisy
Money Manager	Financial Literacy	Brownie
Philanthropist	Financial Literacy	Brownie
Business Owner	Financial Literacy	Junior
Savvy Shopper	Financial Literacy	Junior
Budgeting	Financial Literacy	Cadette
Comparison Shopping	Financial Literacy	Cadette
Count It Up	Cookie Business	Daisy
Talk It Up	Cookie Business	Daisy
Meet My Customers	Cookie Business	Brownie
Give Back	Cookie Business	Brownie
CookieCEO	Cookie Business	Junior
Customer Insights	Cookie Business	Junior
Business Plan	Cookie Business	Cadette

Tables

The Badges_Earned table is a many to many relationship between the badges themselves and the girls who earn them, therefore having no functional dependencies and uses a composite key as its primary key.

Create Statement

```
CREATE TABLE Badges_Earned(
   PID char(8) NOT NULL references
   People(PID),
   Badge_Name text NOT NULL
   references Badges(Badge_Name),
  primary key(PID, Badge_Name)
);
```

Sample Data

pid character	badge_name text
P100	Public Speaker
P111	Money Manager
P114	Respect Authority
P115	Money Counts
P117	CookieCEO
P119	Animal Helper
P120	Financing My Future
P122	Savvy Shopper
P125	Meet My Customers
P126	Customer Loyalty

No Functional Dependencies

PID, Badge_Name →

Query 1

Query all of the volunteers involved in troop 4125, part of Service Unit 129 and the council of San Diego

```
SELECT council.council_name, serviceunit.serviceunitID, troop.troopID,
   troop_volunteers.pid, people.first_name, people.last_name
   FROM council, serviceunit, troop, troop_volunteers, people
   WHERE council.council_name = serviceunit.council_name
   AND serviceunit.serviceunitID = troop.serviceunitID
   AND troop.troopID = troop_volunteers.troopIDAND people.PID =
   troop_volunteers.PID
   AND serviceunit.serviceunitID = 129;
```

	council_name text	serviceu... integer	troopid integer	pid character	first_name text	last_name text
	Girl Scouts San Diego	129	4125	P145	Adele	Rakes
	Girl Scouts San Diego	129	4125	P141	Michael	Sterling
	Girl Scouts San Diego	129	4125	P133	Kasandra	Devin

Query 2

Query all of the girl scouts who earned badges, including the event name, the associated badge name, and their scout level.

```
SELECT people.first_name, people.last_name,
   scout.scout_level, badges_earned,
   events.event_category
   FROM people, badges_earned, badges, scout, events
   WHERE people.pid = badges_earned.PID
   AND events.event_category = badges.event_category
   AND badges_earned.badge_name = badges.badge_name
   AND badges.scout_level = scout.scout_level;
```

first_name text	last_name text	scout_level text	badges_earned badges_earned	event_category text
Monica	MacClery	Cadette	("P100 ","Public Speaker")	Skill Learning
Cathleen	Whitaker	Brownie	("P111 ","Money Manager")	Financial Literacy
Ariel	Parks	Daisy	("P114 ","Respect Authority")	Skill Learning
Faith	Hunt	Daisy	("P115 ","Money Counts")	Financial Literacy
Frances	May	Junior	("P117 ",CookieCEO)	Cookie Business
Bailey	Ackerman	Cadette	("P119 ","Animal Helper")	Skill Learning
Christina	Cullen	Senior	("P120 ","Financing My Future")	Financial Literacy
Tiffany	Mathews	Junior	("P122 ","Savvy Shopper")	Financial Literacy
Casey	Arnold	Brownie	("P125 ","Meet My Customers")	Cookie Business
Zoe	Murphy	Senior	("P126 ","Customer Loyalty")	Cookie Business

Views

Badges are usually distributed at the end of the month. A Troop Leader would need to request a list of badges earned so that an inventory sheet can be sent to accumulate the badges in time for Badge Day. A view of the previous query would be a quick way for a Troop Leader to view all badges earned and by whom so that the right badges are distributed to the right girl scout.

```
DROP VIEW IF EXISTS BadgeDistribution;
CREATE VIEW BadgeDistribution as
   SELECT people.first_name, people.last_name, scout.scout_level,
   badges_earned, events.event_category
   FROM people, badges_earned, badges, scout, events
   WHERE people.pid = badges_earned.PID
   AND events.event_category = badges.event_category
   AND badges_earned.badge_name = badges.badge_name
   AND badges.scout_level = scout.scout_level;
```

```
SELECT * FROM BadgeDistribution;
```

Stored Procedures

As explained before, a girl scout's bridging value is set to true unless you are an Ambassador (since there is no next level). However, for those who no longer want to participate in girl scouts or 'bridge' to the next level, should have a false value once their scout_level becomes null. This stored procedure, sets the bridging_member to false if its scout_level is updated to null once its trigger is called.

```
CREATE OR REPLACE FUNCTION update_bridging_status()
RETURNS TRIGGER AS
$$
BEGIN
   IF NEW.PID is NULL THEN
         UPDATE Girls
        SET Bridging_Member = FALSE
        WHERE NEW.PID = Girls.PID;
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE PLPGSQL;
```

Triggers

This trigger calls on the update_bridging_status stored procedure mentioned earlier.

```
CREATE TRIGGER update_bridging_status_trigger
   AFTER INSERT ON girls
    FOR EACH STATEMENT
    EXECUTE PROCEDURE update_bridging_status();
```

pid character	troopid integer	scout_le... text	bridging... boolean
P100	1602		false
P111	4125	Brownie	true
P112	1213	Daisy	true

Security

- The Admin's (Database Administrator (DBA)) job is to maintain the database and thus should have full access to all tables.

```
CREATE ROLE ADMIN
GRANT ALL ON ALL TABLES IN SCHEMA PUBLIC TO ADMIN
```

- It is the Service Unit Registrar's to handle all existing and new memberships and so it should be able to insert and update all councils, and its respectable service units, and its troops as well as all participants involved.

```
CREATE ROLE SU_REGISTRAR
REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC TO SU_REGISTRAR
GRANT ALL ON ALL TABLES IN SCHEMA PUBLIC TO SU_REGISTRAR
GRANT INSERT ON PEOPLE, STAFF, COUNCIL, COUNCIL_PRESIDENT,
COUNCIL_TREASURER, SERVICEUNIT, SU_MANAGER, SU_SU_ACTIVITYCONSULTANT,
SU_COOKIECOORDINATOR, SU_REGISTRAR, TROOP, TROOP_LEADER, TROOP_COLEADER,
TROOP_VOLUNTEERS, GIRLS, STAFF, PEOPLE
GRANT UPDATE ON PEOPLE, STAFF, COUNCIL, COUNCIL_PRESIDENT,
COUNCIL_TREASURER, SERVICEUNIT, SU_MANAGER, SU_SU_ACTIVITYCONSULTANT,
SU_COOKIECOORDINATOR, SU_REGISTRAR, TROOP, TROOP_LEADER, TROOP_COLEADER,
TROOP_VOLUNTEERS, GIRLS, STAFF, PEOPLE
```

Security continued

- It is the Service Unit Activity Consultant and Cookie Coordinator to handle events and so they should be able to insert and update events and the girl scouts who attended these events as well as badge names and the girl scouts who earned them.

```
CREATE ROLE SU_ACTIVITYCONSULTANT

REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC TO SU_REGISTRAR

GRANT INSERT ON SERVICEUNIT, TROOP, GIRLS, EVENTS,
EVENTS_ATTENDED, BADGES, BADGES_EARNED

GRANT UPDATE ON SERVICEUNIT, TROOP, GIRLS, EVENTS,
EVENTS_ATTENDED, BADGES, BADGES_EARNED
```

- The Troop Leader should be able to keep attendance of his/her troop members as well as their achievements.

```
CREATE ROLE TROOP_LEADER

REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC TO TROOP_LEADER

GRANT SELECT ON GIRLS, EVENTS, EVENTS_ATTENDED, BADGES, BADGES_EARNED

GRANT INSERT ON GIRLS, EVENTS, EVENTS_ATTENDED, BADGES, BADGES_EARNED

GRANT UPDATE ON GIRLS, EVENTS, EVENTS_ATTENDED, BADGES, BADGES_EARNED
```

Implementation Notes, Problems & Future Enhancements

- Implementation Notes:

Because GSUSA is a large organization with many divisions and subdivisions, be careful to always include subdivision's dependency on their larger division. In other words, because troops are part of service units and service units are part of councils, be sure to always include foreign keys or to use both a primary and foreign key as a composite key to refer to functionally dependent data otherwise queries will result in many duplicates and inconsistent data.

- Problems & Future Enhancements

I am aware that the update_bridging_status trigger calling on its stored procedures is a bit messy and not the optimal way to go about bridging girl scouts to their next level. A better way would be to store a timestamp on each of the girl's birthdate and have a trigger calculate an age, and have that value match the end_age of a scout_level; so when a girl scout reaches the age to bridge, her scout_level will automatically update.