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

By: Federica Bruno

Table of Contents

Executive Summary

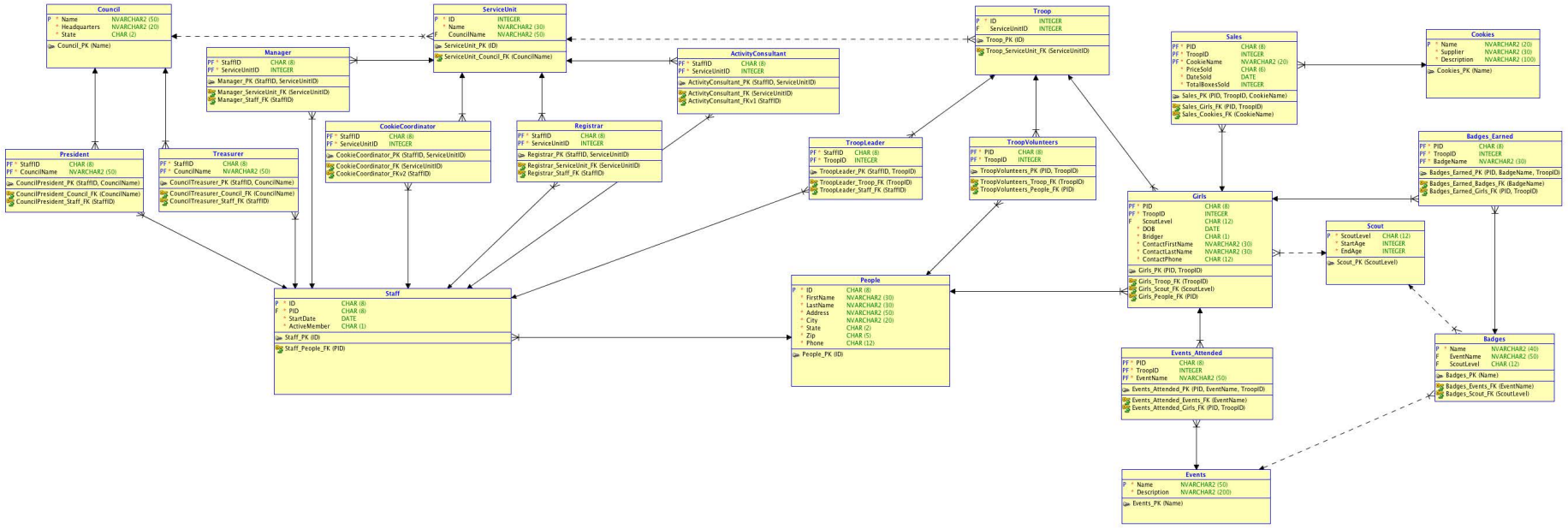
With 2.3 million growing members, roughly 890,000 volunteers, and over 200 million boxes of cookies sold each year, there is little room for mistakes, especially with the fierce and unmerciful. GSUSA (Girl Scouts of USA) has contacted me to propose a database that would help catalog scout members, staff, and even distribution of badges to help 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 was created, along with mock data for testing purposes. To ensure that the database will be accessible to authorized members only, ideal user rolls will be suggested towards the end. Sample output results and explanations have also been provided for reports and views.

Like any complex project, there is always room for improvements. Future enhancements are suggested at the end. Implementation notes and known problems are also included. The ultimate goal of this design is to have a full functioning database that can hold a large volume of accurate and consistent data.

This design was targeted for and tested on MySQL Oracle.

Entity Relationship Diagram



Tables

The People table contains all of the common attributes that all members involved in GSUSA share - such as first name, last name, home address, and phone number. The People table is a supertype to the following tables: Staff, President, Treasurer, Manager, ActivityConsultant, CookieCoordinator, Registrar, TroopLeader, TroopVolunteers, and finally, Girls.

Create Statement

```
CREATE TABLE people (
    id          CHAR(8) NOT NULL,
    firstname   NVARCHAR2(30) NOT NULL,
    lastname    NVARCHAR2(30) NOT NULL,
    address     NVARCHAR2(50) NOT NULL,
    city        NVARCHAR2(20) NOT NULL,
    state       CHAR(2) NOT NULL,
    zip         CHAR(5) NOT NULL,
    phone       CHAR(12) NOT NULL
);
```

Constraints

```
ALTER TABLE people
     ADD CONSTRAINT people_pk PRIMARY KEY
( id );
```

Functional Dependencies

ID --> FirstName, LastName, Address, City, State, Zip, Phone

Sample Data

ID	FIRSTNAME	LASTNAME	ADDRESS	CITY	ST	ZIP	PHONE
P6676208	Monica	MacClery	2880 Frederick Street	Sacramento	CA	95814	209-512-6793
P0298082	Cathleen	Whitaker	2492 Providence Lane	La Puente	CA	91744	209-298-2742
P5654565	Corinne	Peterson	2701 Delaware Avenue	San Francisco	CA	94143	916-492-6861
P4769670	Sunny	Walsh	4257 Ella Street	Palo Alto	CA	94306	916-951-5375
P0669387	Ariel	Parks	2714 Dennison Street	Stockton	CA	9520	704-797-6365
P1209264	Faith	Hunt	1615 Park Avenue	Sacramento	CA	95814	980-330-4650
P2086128	Lila	Morris	1885 Alexander Avenue	San Rafael	CA	91405	925-627-7352
P3275765	Frances	May	3878 Koontz Lane	Van Nuys	CA	91405	818-786-2842
P6130842	Larissa	Cooke	49 Jail Drive	Los Angeles	CA	90024	310-206-6266
P7509911	Bailey	Ackerman	3926 Southside Lane	Irvine	CA	92614	323-712-4884
P0153951	Christina	Cullen	3569 Station Street	San Jose	CA	95113	510-931-2241

Tables

The Staff table contains all members involved in GSUSA, excluding the girl scouts. This table is a subtype of the People table and is a supertype to the following tables: President, Treasurer, Manager, ActivityConsultant, CookieCoordinator, Registrar, and TroopLeader.

Create Statement

```
CREATE TABLE staff (
    id              CHAR(8) NOT NULL,
    pid             CHAR(8) NOT NULL,
    startdate       DATE NOT NULL,
    activemember    CHAR(1) NOT NULL
);
```

Constraints

```
ALTER TABLE staff
     ADD CONSTRAINT staff_pk PRIMARY KEY
( id );

ALTER TABLE staff
    ADD CONSTRAINT staff_people_fk FOREIGN
KEY ( pid )
        REFERENCES people ( id );
```

Functional Dependencies

```
ID --> PID, StartDate, ActiveMember
```

Sample Data

ID	PID	STARTDATE	A
S186040	P6676208	24-DEC-90	T
S4670040	P0298082	26-SEP-91	T
S5043855	P5654565	01-JUN-93	T
S6217218	P4769670	27-JUN-94	T
S1924501	P0669387	14-AUG-96	T
S2877307	P1209264	24-DEC-96	T
S7263925	P2086128	19-JAN-98	F
S9782908	P3275765	26-FEB-98	T
S2876519	P6130842	13-AUG-98	T
S3011783	P7509911	21-JUN-99	F
S5932608	P0153951	06-DEC-01	T

Tables

GSUA is subdivided into Councils. A state can have more than one Council, and they are distinguished by their county name. For the sake of simplicity, this particular database focuses only on the Councils of California. The table sets a default value to the state attribute as California.

Create Statement

```
CREATE TABLE council (
      name            NVARCHAR2(30) NOT NULL,
      headquarters    NVARCHAR2(20) NOT NULL,
      state           CHAR(2) DEFAULT 'CA' NOT NULL
);
```

Constraints

```
ALTER TABLE council
     ADD CONSTRAINT council_pk PRIMARY
KEY ( name );
```

Functional Dependencies

```
Name --> Headquarters, State
```

Sample Data

NAME	HEADQUARTERS	ST
Girl Scouts of Central California South	Fresno	CA
Girl Scouts of Californias Central Coast	Camarillo	CA
Girl Scouts of Northern California	San Jose	CA
Girl Scouts of Greater Los Angeles	Los Angeles	CA
Girl Scout Council of Orange County	Irvine	CA
Girl Scouts San Diego	San Diego	CA

Tables

Every Council has one President, identified by its StaffID and Council Name. This table is a subtype of both the Staff table and the People table. ***Note****: There is a much larger executive board within a Council; however, this database focuses on the main board members: Council President and Council Treasurer.*

Create Statement

```
CREATE TABLE president (
     staffid   CHAR(8) NOT NULL,
     councilname   NVARCHAR2(50) NOT NULL
);
```

Constraints

```
ALTER TABLE president
     ADD CONSTRAINT
councilpresident_pk PRIMARY KEY
( staffid, councilname );

ALTER TABLE president
    ADD CONSTRAINT
councilpresident_council_fk FOREIGN
KEY ( councilname )
      REFERENCES council ( name );

ALTER TABLE president
    ADD CONSTRAINT
councilpresident_staff_fk FOREIGN KEY
( staffid )
         REFERENCES staff ( id );
```

Functional Dependencies

```
StaffID, CouncilName --> none
```

Sample Data

SID	COUNCILNAME
S186040	Girl Scouts of Northern California
S4670040	Girl Scout Council of Orange County
S5043855	Girl Scouts of Greater Los Angeles
S6217218	Girl Scouts San Diego

Tables

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

Create Statement

```
CREATE TABLE treasurer (
     staffid       CHAR(8) NOT NULL,
     councilname   NVARCHAR2(30) NOT NULL
);
```

Functional Dependencies

```
StaffID, CouncilName --> none
```

Constraints

```
ALTER TABLE treasurer
     ADD CONSTRAINT
counciltreasurer_pk PRIMARY KEY
( staffid, councilname );

ALTER TABLE treasurer
    ADD CONSTRAINT
counciltreasurer_council_fk FOREIGN
KEY ( councilname )
     REFERENCES council ( name );

ALTER TABLE treasurer
    ADD CONSTRAINT
counciltreasurer_staff_fk FOREIGN KEY
( staffid ) REFERENCES staff ( id );
```

Sample Data

SID	COUNCILNAME
S1924501	Girl Scouts of Northern California
S2877307	Girl Scout Council of Orange County
S7263925	Girl Scouts of Greater Los Angeles
S9782908	Girl Scouts San Diego

Tables

Councils are further subdivided into Service Units. A Service Unit is identified by a unique 3 digit number and are named after their city of residency.

Create Statement

```
CREATE TABLE serviceunit (
    id              INTEGER(3) NOT NULL,
    name            NVARCHAR2(30) NOT NULL,
    councilname     NVARCHAR2(50)
);
```

Constraints

```
ALTER TABLE serviceunit
     ADD CONSTRAINT serviceunit_pk PRIMARY
KEY ( id );

ALTER TABLE serviceunit
     ADD CONSTRAINT serviceunit_council_fk
FOREIGN KEY ( councilname )
        REFERENCES council ( name )
            ON DELETE CASCADE;
```

Sample Data

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

Functional Dependencies

```
StaffID --> Name, CouncilName
```

Tables

Every Service Unit has one Manager, identified by its StaffID and ServiceUnitID. This table is a subtype of both the Staff table and the People table. ***Note****: There are many job positions in a Service Unit; however this database focuses on are the 4 main staff members: Manager, Activity Consultant, Cookie Coordinator, and Registrar.*

Create Statement

```
CREATE TABLE manager (
    staffid         CHAR(8) NOT NULL,
    serviceunitid   INTEGER(3) NOT NULL
);
```

Constraints

```
ALTER TABLE manager
     ADD CONSTRAINT manager_pk PRIMARY KEY
( staffid, serviceunitid );

ALTER TABLE manager
    ADD CONSTRAINT manager_serviceunit_fk
FOREIGN KEY ( serviceunitid )
        REFERENCES serviceunit ( id );

ALTER TABLE manager
    ADD CONSTRAINT manager_staff_fk
FOREIGN KEY ( staffid )
        REFERENCES staff ( id );
```

Functional Dependencies

```
StaffID, ServiceUnitID --> none
```

Sample Data

STAFFID	SERVICEUNITID
S2876519	136
S3011783	467
S5932608	752
S7729519	129

Tables

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

Create Statement

```
CREATE TABLE activityconsultant (
    staffid         CHAR(8) NOT NULL,
    serviceunitid   INTEGER(3) NOT NULL
);
```

Constraints

```
ALTER TABLE activityconsultant
     ADD CONSTRAINT activityconsultant_pk
PRIMARY KEY ( staffid, serviceunitid );

ALTER TABLE activityconsultant
    ADD CONSTRAINT activityconsultant_fk
FOREIGN KEY ( serviceunitid )
        REFERENCES serviceunit ( id );

ALTER TABLE activityconsultant
    ADD CONSTRAINT activityconsultant_fkv1
FOREIGN KEY ( staffid )
        REFERENCES staff ( id );
```

Functional Dependencies

```
SraffID, ServiceUnitID--> none
```

Sample Data

STAFFID	SERVICEUNITID
S1626201	752
S4557787	467
S6668631	129
S8973388	136

Tables

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

Create Statement

```
CREATE TABLE cookiecoordinator (
    staffid         CHAR(8) NOT NULL,
    serviceunitid   INTEGER(3) NOT NULL
);
```

Functional Dependencies

```
SraffID, ServiceUnitID--> none
```

Constraints

```
ALTER TABLE cookiecoordinator
     ADD CONSTRAINT cookiecoordinator_pk
PRIMARY KEY ( staffid, serviceunitid );

ALTER TABLE cookiecoordinator
    ADD CONSTRAINT cookiecoordinator_fk
FOREIGN KEY ( serviceunitid )
        REFERENCES serviceunit ( id );

ALTER TABLE cookiecoordinator
    ADD CONSTRAINT cookiecoordinator_fkv2
FOREIGN KEY ( staffid )
        REFERENCES staff ( id );
```

Sample Data

STAFFID	SERVICEUNITID
S2043005	467
S4087969	752
S5118220	129
S8858981	136

Tables

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

Create Statement

```
CREATE TABLE registrar (
    staffid         CHAR(8) NOT NULL,
    serviceunitid   INTEGER(3) NOT NULL
);
```

Constraints

```
ALTER TABLE registrar
     ADD CONSTRAINT registrar_pk PRIMARY
KEY ( staffid, serviceunitid );

ALTER TABLE registrar
    ADD CONSTRAINT registrar_serviceunit_fk
FOREIGN KEY ( serviceunitid )
        REFERENCES serviceunit ( id );

ALTER TABLE registrar
    ADD CONSTRAINT registrar_staff_fk
FOREIGN KEY ( staffid )
        REFERENCES staff ( id );
```

Functional Dependencies

```
SraffID, ServiceUnitID--> none
```

Sample Data

STAFFID	SERVICEUNITID
S1203792	129
S4933210	136
S5461746	467
S6210458	752

Tables

Finally, Service Units are even further subdivided into local Troops, which consist of Troop Leaders, Troop Volunteers, and the Girl Scouts themselves. Troops are identified by a unique 4 digit number.

Create Statement

```
CREATE TABLE troop (
    id               INTEGER NOT NULL,
    serviceunitid    INTEGER
);
```

Functional Dependencies

```
ID --> ServiceUnitID
```

Constraints

```
ALTER TABLE troop
     ADD CONSTRAINT troop_pk PRIMARY KEY
( id );

ALTER TABLE troop
    ADD CONSTRAINT troop_serviceunit_fk
FOREIGN KEY ( serviceunit )
        REFERENCES serviceunit ( id );
```

Sample Data

ID	SERVICEUNITID
3457	752
1369	752
2689	135
1213	135
2679	135
2346	679
1248	136
3579	469
1602	468
9527	467
4125	129

Tables

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

Create Statement

```
CREATE TABLE troopleader (
    staffid   CHAR(8) NOT NULL,
    troopid   INTEGER NOT NULL
);
```

Functional Dependencies

```
StaffID, TroopID --> none
```

Constraints

```
ALTER TABLE troopleader
  ADD CONSTRAINT troopleader_pk PRIMARY
              KEY ( staffid, troopid );
ALTER TABLE troopleader
    ADD CONSTRAINT
troopleader_staff_fk FOREIGN KEY
( staffid )
        REFERENCES staff ( id );
ALTER TABLE troopleader
    ADD CONSTRAINT
troopleader_troop_fk FOREIGN KEY
( troopid )
        REFERENCES troop ( id );
```

Sample Data

STAFFID	TROOPID
S3853486	1213
S4547053	1602
S6723911	1369
S7130743	1369

Tables

A Troop can have many Volunteers and they are identified by their PID (since Volunteers are not categorized under Staff) and their TroopID. This table is a subtype of the People table.

Create Statement

```
REATE TABLE troopvolunteers (
    pid       CHAR(8) NOT NULL,
    troopid   INTEGER NOT NULL
);
```

Constraints

```
ALTER TABLE troopvolunteers
     ADD CONSTRAINT
troopvolunteers_pk PRIMARY KEY ( pid,
troopid );
ALTER TABLE troopvolunteers
    ADD CONSTRAINT
troopvolunteers_people_fk FOREIGN KEY
( pid )
       REFERENCES people ( id );
ALTER TABLE troopvolunteers
    ADD CONSTRAINT
troopvolunteers_troop_fk FOREIGN KEY
( troopid )
        REFERENCES troop ( id );
```

Functional Dependencies

```
PID, TroopID --> none
```

Sample Data

PID	TROOPID
P3590179	4125
P5476217	8176
P6203489	4125
P6531732	8176
P6533998	8176
P6846684	4125

Tables

The Scout table contains the 5 scout categories 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 (
    scoutlevel CHAR(12) NOT NULL,
    startage   INTEGER NOT NULL,
    endage     INTEGER NOT NULL
);
```

Constraints

```
ALTER TABLE scout
     ADD CONSTRAINT scout_pk PRIMARY KEY
( scoutlevel );
```

Functional Dependencies

```
ScoutLevel --> StartAge, EndAge
```

Sample Data

SCOUTLEVEL	ST	EN
Daisy	5	6
Brownie	7	8
Junior	9	10
Cadette	11	13
Senior	14	15
Ambassador	16	18

Tables

The Girls table contains all of the Girl Scout members. The girls are identified by their PID and TroopID. A girl scout sticks with the same troop throughout her girl scout experience. A bridging member/'bridger' is simply a girl scout who still has advancing scout levels to complete. A false bridging member status would indicate that a girl scout wishes to no longer be active. The scout level Ambassador is set to false, because it is the last level a girl scout can achieve.

CREATE TABLE girls (

```
    pid                 CHAR(8) NOT NULL,
    troopid             INTEGER NOT NULL,
    scoutlevel          CHAR(12),
    dob                 DATE NOT NULL,
    Bridger             CHAR(1) NOT NULL,
    contactfirstname    NVARCHAR2(30) NOT NULL,
    contactlastname     NVARCHAR2(30) NOT NULL,
    contactphone        CHAR(12) NOT NULL
);
```

Functional Dependencies

```
PID, TroopID --> ScoutLevel, DOB, Bridger,
ContactFirstName, ContactLastName, ContactPhone
```

Constraints

```
ALTER TABLE girls
      ADD CONSTRAINT girls_pk PRIMARY KEY
( pid, troopid );
ALTER TABLE girls
    ADD CONSTRAINT girls_people_fk FOREIGN KEY
( pid )
           REFERENCES people ( id );
ALTER TABLE girls
    ADD CONSTRAINT girls_scout_fk FOREIGN KEY
( scoutlevel )
       REFERENCES scout ( scoutlevel );
ALTER TABLE girls
    ADD CONSTRAINT girls_troop_fk FOREIGN KEY
( troopid )
         REFERENCES troop ( id );
```

Sample Output

PID	TROOPID	SCOUTLEVEL	DOB	B	CONTACTFIRSTNAME	CONTACTLASTNAME	CONTACTPHONE
9838394	1602	Cadette	09-AUG-06	T	Ann	Reed	718-351-0931
P4649750	4125	Brownie	18-DEC-10	T	Jessica	Griffin	628-704-2570
P6435221	1213	Ambassador	22-APR-01	F	Peter	Murphy	347-600-2659
P8088710	1602	Cadette	01-FEB-05	T	Cheryl	Torres	257-597-1877
P9178792	1213	Daisy	01-SEP-13	T	Richard	Johnson	740-737-6901
P0120929	1369	Daisy	12-FEB-12	T	John	Adams	564-525-1982
P1092096	2346	Senior	20-MAR-04	T	Joe	Howard	432-634-9135
P4812421	2689	Junior	03-MAR-09	T	Joan	Green	524-995-4760
P5877450	1248	Ambassador	01-NOV-02	F	Chris	Baker	236-499-7072
P8808075	6281	Senior	18-DEC-03	F	Terry	Flores	922-483-6672

Tables

The girls must attend events to earn their badges. Selling cookies is a shared activity throughout all scout levels, but all scout levels host special events for the girls to learn important life lessons and get a chance to earn a badge unique to their scout level. There are hundreds of events the girls can choose from to attend too, but for the sake of simplicity, this database focuses on 3 events: Skill Learning, Cookie Business, and Financial Literacy.

Create Statement

```
CREATE TABLE events (
    name          NVARCHAR2(50) NOT NULL,
    description   NVARCHAR2(200) NOT NULL
);
```

Functional Dependencies

```
Name --> Description
```

Constraints

```
ALTER TABLE events
     ADD CONSTRAINT events_pk PRIMARY
KEY ( name );
```

Sample Output

NAME	DESCRIPTION
Skill Learning	Girls will learn skills from different areas of life such as: food, athletics, art, and the environment, b
Financial Literacy	Girls will learn the basics of credit, how to be frugal with their savings, and planning for their future.
Cookie Business	Girls will learn when and where to sell cookies, develop money management skills and people skills by inte

Tables

The Events_Attended table is a many-to-many relationship between the events and the girls who attend the events. Therefore, this table has no functional dependencies and uses PID, TroopID (which together identify the Girls), and EventName as its composite primary key.

Create Statement

```
CREATE TABLE events_attended (
    pid          CHAR(8) NOT NULL,
    troopid      INTEGER NOT NULL,
    eventname    NVARCHAR2(50) NOT NULL
);
```

Constraints

```
ALTER TABLE events_attended
     ADD CONSTRAINT events_attended_pk
PRIMARY KEY ( pid, eventname, troopid );
ALTER TABLE events_attended
    ADD CONSTRAINT
events_attended_events_fk FOREIGN KEY
( eventname )
       REFERENCES events ( name );
ALTER TABLE events_attended
    ADD CONSTRAINT
events_attended_girls_fk FOREIGN KEY
( pid, troopid )
        REFERENCES girls ( pid,
troopid );
```

Functional Dependencies

```
PID, TroopID, EventName --> none
```

Sample Data

PID	TROOPID	EVENTNAME
9838394	1602	Skill Learning
P0120929	1369	Cookie Business
P1092096	2346	Skill Learning
P4649750	4125	Skill Learning
P4812421	2689	Financial Literacy
P5877450	1248	Cookie Business
P6435221	1213	Financial Literacy
P8088710	1602	Skill Learning
P8808075	6281	Cookie Business
P9178792	1213	Financial Literacy

Tables

Badges are earned by attending events. Badge names are unique to each scout level, meaning a Daisy scout cannot earn Brownie badges, because each scout has a different agenda at an event. ***Note****: 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 (
    name          NVARCHAR2(30) NOT NULL,
    eventname     NVARCHAR2(50),
    scoutlevel    CHAR(12)
);
```

Constraints

```
ALTER TABLE badges
     ADD CONSTRAINT badges_pk PRIMARY
KEY ( name );

ALTER TABLE badges
    ADD CONSTRAINT badges_events_fk
FOREIGN KEY ( eventname )
        REFERENCES events ( name );

ALTER TABLE badges
    ADD CONSTRAINT badges_scout_fk
FOREIGN KEY ( scoutlevel )
        REFERENCES scout ( scoutlevel );
```

Functional Dependencies

```
Name --> EventName, ScoutLevel
```

Sample Data

NAME	EVENTNAME	SCOUTLEVEL
Give Back	Cookie Business	Brownie
CookieCEO	Cookie Business	Junior
Customer Insights	Cookie Business	Junior
Business Plan	Cookie Business	Cadette
Marketing	Cookie Business	Cadette
Thing Big	Cookie Business	Cadette
Customer Loyalty	Cookie Business	Senior
My Portfolio	Cookie Business	Senior
Research and Development	Cookie Business	Ambassador

Tables

The Badges_Earned table is a many-to many-relationship between the badges and the girls who earn them. Therefore, this table has no functional dependencies and uses PID, TroopID, and BadgeName as its composite primary key.

Create Statement

```
CREATE TABLE badges_earned (
    pid          CHAR(8) NOT NULL,
    troopid      INTEGER NOT NULL,
    badgename    NVARCHAR2(30) NOT NULL
);
```

Constraints

```
ALTER TABLE badges_earned
      ADD CONSTRAINT badges_earned_pk
PRIMARY KEY ( pid, badgename, troopid );

ALTER TABLE badges_earned
    ADD CONSTRAINT
badges_earned_badges_fk FOREIGN KEY
( badgename )
      REFERENCES badges ( name );

ALTER TABLE badges_earned
    ADD CONSTRAINT
badges_earned_girls_fk FOREIGN KEY
( pid, troopid )
       REFERENCES girls ( pid,
troopid );
```

Functional Dependencies

```
PID, TroopID, BadgeName --> none
```

Sample Data

PID	TROOPID	BADGENAME
9838394	1602	Public Speaker
P0120929	1369	Animal Helper
P1092096	2346	Financing My Future
P4649750	4125	Money Manager
P4812421	2689	Savvy Shopper
P5877450	1248	Meet My Customers
P6435221	1213	Respect Authority
P8088710	1602	Money Counts
P8808075	6281	Customer Loyalty
P9178792	1213	CookieCEO

Tables

It wouldn't be a GSUSA Database without their famous cookies. The Cookies are identified by their name only. ***Note****: There are 12 Girl Scout cookies, but for the sake of simplicity, the Cookies table contains just my personal favorites: Thin Mints, Samoas, Tagalongs, Trefoils, Do-si-dos, and Lemonades.*

Create Statement

```
CREATE TABLE cookies (CREATE TABLE cookies
(
    name           NVARCHAR2(20) NOT NULL,
    supplier       NVARCHAR2(30) NOT NULL,
    description    NVARCHAR2(100) NOT NULL
);
```

Constraints

```
ALTER TABLE cookies
     ADD CONSTRAINT cookies_pk PRIMARY
KEY ( name );
```

Functional Dependencies

```
Name --> Supplier, Description
```

Sample Data

NAME	SUPPLIER	DESCRIPTION
Thin Mints	Little Brownie Bakers	Round, mint-flavored cookies with a delicious chocolaty coating.
Samoas	Little Brownie Bakers	Caramel and toasted coconut-covered cookies.
Tagalongs	ABC Bakers	Peanut butter patty with a rich chocolaty coating.
Trefoils	ABC Bakers	Trefoil shaped shortbread cookies
Do-si-dos	Little Brownie Bakers	Peanut butter patty with crisp and crunchy oatmeal on the outside.
Lemonades	ABC Bakers	Tangy lemon-icing-topped shortbread cookies.

Tables

The Sales table is a many-to-many relationship between the cookies and the girls who sell them. This table use PID, TroopID, and CookieName as its composite primary key. ***Note****: The DateSold attribute was included because cookie prices may not always stay the same. It is important for the database to hold a record of past prices so that history is not erased if a price change should happen.*

Create Statement

```
CREATE TABLE sales (
    pid              CHAR(8) NOT NULL,
    troopid          INTEGER NOT NULL,
    cookiename       NVARCHAR2(20) NOT NULL,
    pricesold        CHAR(6) NOT NULL,
    datesold         DATE NOT NULL,
    totalboxessold   INTEGER NOT NULL
);
```

Functional Dependencies

```
PID, TroopID, CookieName -->
PriceSold, DateSold, TotalBoxesSold
```

Constraints

```
ALTER TABLE sales
    ADD CONSTRAINT sales_pk PRIMARY KEY
( pid, troopid, cookiename );

ALTER TABLE sales
    ADD CONSTRAINT sales_cookies_fk
FOREIGN KEY ( cookiename )
        REFERENCES cookies ( name );

ALTER TABLE sales
    ADD CONSTRAINT sales_girls_fk
FOREIGN KEY ( pid, troopid )
        REFERENCES girls ( pid,
troopid );
```

Sample Data

PID	TROOPID	COOKIENAME	PRICES	DATESOLD	TOTALBOXESSOLD
9838394	1602	Thin Mints	$4.00	01-JAN-18	12
P4649750	4125	Thin Mints	$4.00	08-JAN-18	10
P6435221	1213	Do-si-dos	$4.00	12-JAN-18	5
P8088710	1602	Tagalongs	$4.00	17-JUN-18	23
P8088710	1602	Samoas	$4.00	17-JUN-18	16
P8088710	1602	Thin Mints	$4.00	17-JUN-18	19
P4812421	2689	Trefoils	$4.00	11-APR-18	8
P5877450	1248	Trefoils	$4.00	24-APR-18	14
P5877450	1248	Samoas	$4.00	24-APR-18	17
P1092096	2346	Lemonades	$4.00	12-MAR-18	11

Report 1

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

```
SELECT council.name, serviceunit.ID, troop.ID, troopvolunteers.pid,
people.firstname, people.lastname
          FROM council, serviceunit, troop, troopvolunteers, people
          WHERE council.name = serviceunit.councilname
               AND serviceunit.ID = troop.serviceunitID
               AND troop.ID = troopvolunteers.troopID
               AND people.ID = troopvolunteers.PID
               AND serviceunit.ID = 129;
```

Sample Results

NAME	ID	ID	PID	FIRSTNAME	LASTNAME
Girl Scouts San Diego	129	4125	P3590179	David	Washington
Girl Scouts San Diego	129	4125	P6203489	Brooke	Clark
Girl Scouts San Diego	129	4125	P6846684	Kerrie	Herbert

Report 2

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

```
SELECT p.firstname, p.lastname, b.scoutlevel, be.badgename, b.eventname
  FROM people p, badges_earned be, scout s, events e, badges b
  WHERE p.ID = be.PID
       AND b.scoutlevel = s.scoutlevel
       AND be.badgename = b.name
       AND b.eventname = e.name;
```

Sample Results

FIRSTNAME	LASTNAME	SCOUTLEVEL	BADGENAME	EVENTNAME
Becci	Hawkins	Cadette	Animal Helper	Skill Learning
Kelly	Beckham	Senior	Financing My Future	Financial Literacy
Allie	Rakes	Brownie	Money Manager	Financial Literacy
Stephanie	Nielson	Junior	Savvy Shopper	Financial Literacy
Gina	Clark	Brownie	Meet My Customers	Cookie Business
Alexa	Fairfield	Daisy	Respect Authority	Skill Learning
Nina	Noel	Senior	Customer Loyalty	Cookie Business
Jackie	Darwin	Junior	CookieCEO	Cookie Business

View 1

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.

```
CREATE VIEW BadgeDistribution as

SELECT p.firstname, p.lastname, b.scoutlevel, be.badgename, b.eventname
  FROM people p, badges_earned be, scout s, events e, badges b
     WHERE p.ID = be.PID
          AND b.scoutlevel = s.scoutlevel
          AND be.badgename = b.name
          AND b.eventname = e.name;
```

```
SELECT * FROM BadgeDistribution;
```

View 2

At the end of each cookie-selling period, Troop Leaders are anxious to see which scout from their troop put their Cookie Business skills to use. This view queries which girl scout sold the most Samoas boxes.

```
CREATE VIEW WhoSoldTheMostSamoas as

SELECT DISTINCT p.firstname, p.lastname, s.cookiename, s.totalboxessold
  FROM people p, sales s, cookies c, girls g, scout sc
     WHERE p.ID = s.PID
          AND g.scoutlevel = sc.scoutlevel
          AND s.cookiename = c.name
        AND s.totalboxessold > 10
        AND cookiename = 'Samoas';
```

```
SELECT * FROM WhoSoldTheMostSamoas;
```

Sample Results

FIRSTNAME	LASTNAME	COOKIENAME	TOTALBOXESSOLD
Gina	Clark	Samoas	17

Security

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

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

The Service Unit Registrar’s job is to handle all existing and new memberships and so they should be able to insert and update all councils, its respectable service units and its troops, as well as all participants involved.

```
CREATE ROLE REGISTRAR

REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC TO REGISTRAR

GRANT ALL ON ALL TABLES IN SCHEMA PUBLIC TO REGISTRAR

GRANT INSERT ON PEOPLE, STAFF, COUNCIL, PRESIDENT, TREASURER,
SERVICEUNIT, MANAGER, ACTIVITYCONSULTANT, COOKIECOORDINATOR, REGISTRAR,
TROOP, TROOPLEADER, TROOPVOLUNTEERS, GIRLS, STAFF, PEOPLE

GRANT UPDATE ON PEOPLE, STAFF, COUNCIL, PRESIDENT, TREASURER,
SERVICEUNIT, MANAGER, ACTIVITYCONSULTANT, COOKIECOORDINATOR, REGISTRAR,
TROOP, TROOPLEADER, TROOPVOLUNTEERS, GIRLS, STAFF, PEOPLE
```

Security

The Service Unit Activity Consultant and Cookie Coordinator's job is to handle events and so they should be able to insert and update events, insert and update the girl scouts members who attend these events, and insert and update badge names and the girl scouts who earned them.

```
CREATE ROLE ACTIVITYCONSULTANT

REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC TO 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 TROOPLEADER

REVOKE ALL ON ALL TABLES IN SCHEMA PUBLIC TO TROOPLEADER

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 and Known Problems

Because GSUSA is a large organization with many divisions and subdivisions, remember to always include the subdivision's dependency on their larger division. For example, since troops are part of service units and service units are part of councils, it is important to include foreign keys and reference their PK FK composite keys. Failing to do so when referring to functionally dependent data will cause the database to throw errors or result in inconsistent data.

A reoccurring issue I encountered was whenever I ran my SQL file with my insert statements through SQLDeveloper. It threw a 'unique constraint violated' error for the TroopLeader table on all four columns. I encountered a similar error with the Sales table. After I ran my insert statements, it threw a 'integrity constraint violated' error on just three out of the nine columns inserted. I couldn't resolve these issues, but the sample output seemed to be sufficient enough for testing purposes.

Finally, the Security portion of this project was not tested, and was strictly for personal research and suggesting ideal roles that should and would most likely be implemented in a large enterprise database such as this one.

Future Enhancements

One enhancement that I would have liked to have added was a stored procedure that would automatically advance or 'bridge' a girl scout to her next scout level. One way I could have went about this was adding a timestamp on each of the girl scout member's DOB and calling a trigger that would calculate/derive an age from that DOB, and have that value match the 'EndAge' attribute in the Scouts table. Ultimately, when a girl scout turns the age to advance or 'bridge' to the next scout level, say Cadette to Senior, the scoutlevel attribute will automatically update.