GLEN ROCK FOOTBALL LEAGUE

LARGE SCALE DATABASE PROPOSAL PLAN

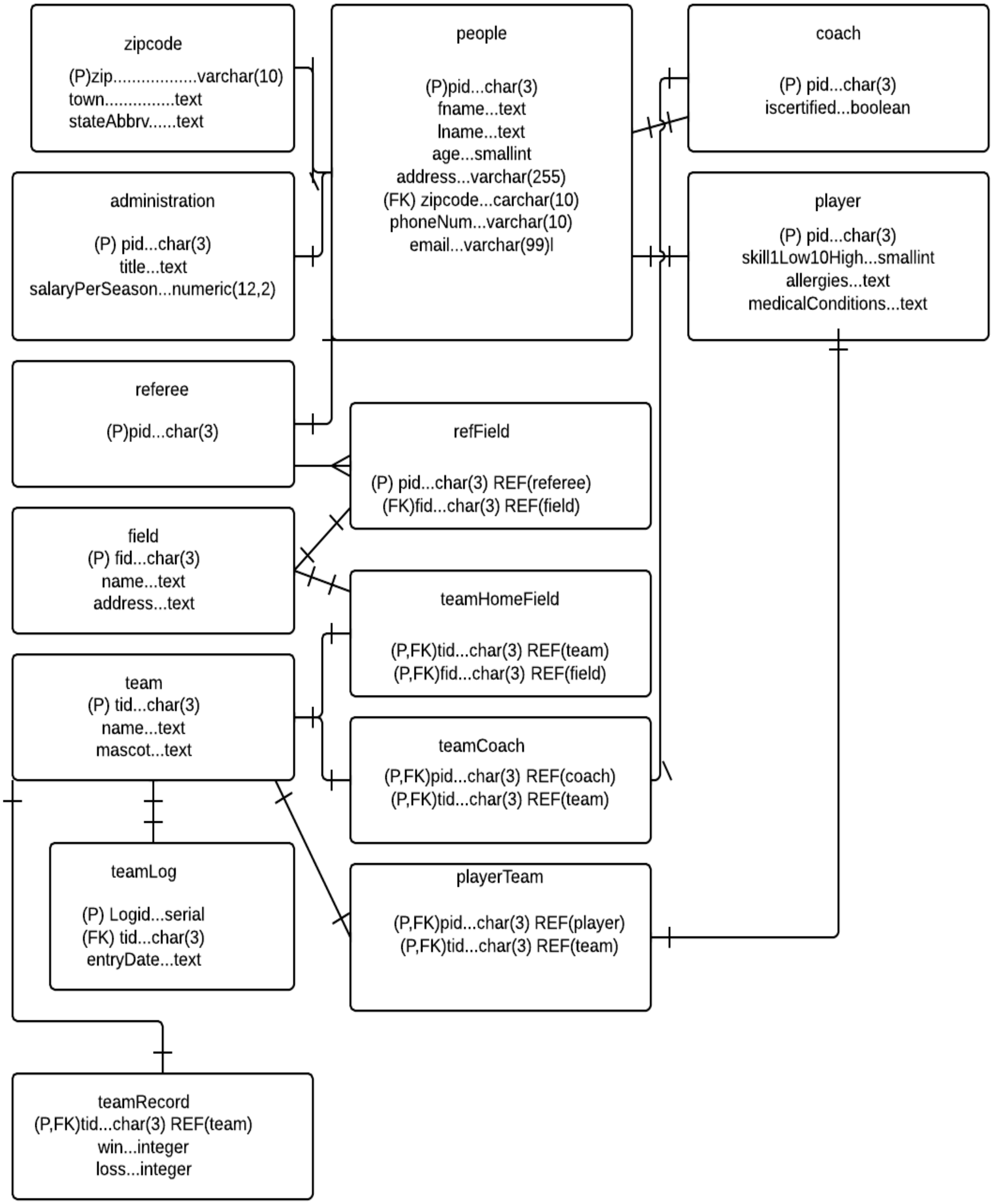
GLEN ROCK JUNIOR FOOTBALL LEAGUE DATABASE OUTLINE.

[1 DEMAREST ST]
[GLEN ROCK, NJ 07452]

Table of Contents

Entity Relationship Diagram

SHOWS THE RELATIONS BETWEEN TABLES



PROPOSAL AND DATABASE PLAN

GLEN ROCK'S BEST OPPORTUNITY FOR LAYING OUT DATA ASSOCIATED WITH JUNIOR FOOTBALL

EXECUTIVE SUMMARY

Overview

This proposal highlights the design of a database for Glen Rock Junior Football. Data may include, but it not limited to: Players, Coaches, Events, Games, Referees, etc…

Note: All Data in this report is faux and for presentation purposes only. All data is not meant to be taken as real or literal, and should be known it is only to show a model.

Objectives

- To show and implement a secure and data driven database without any duplicate data to help layout the foundation of junior football.

- To save time in the future for administrators and apply useful queries to make information out of data so everyone can receive the proper benefits that will be available from this implementation.

- To Create useful Tables that will help the functionality of the league and communication with information about, but not limited to: player statistics, coaching roles, referee game scheduling, opponent town information, field availability, etc…

Table: zipcode

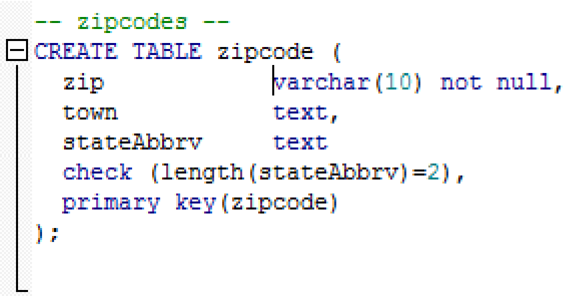
Purpose

To create a table with the leagues opponents and zip codes to relay information based on location to help scheduling.

Functional Dependencies

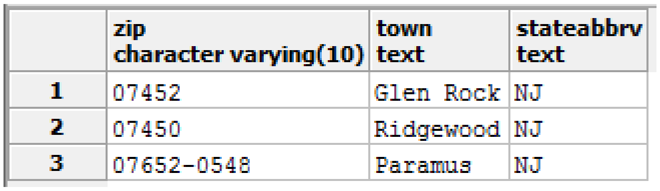
zip → town , stateAbbrv

CREATE



```
-- zipcodes --
CREATE TABLE zipcode (
  zip            varchar(10) not null,
  town           text,
  stateAbbrv     text
  check (length(stateAbbrv)=2),
  primary key(zipcode)
);
```

SAMPLE



	zip character varying(10)	town text	stateabbrv text
1	07452	Glen Rock	NJ
2	07450	Ridgewood	NJ
3	07652-0548	Paramus	NJ

Table: people

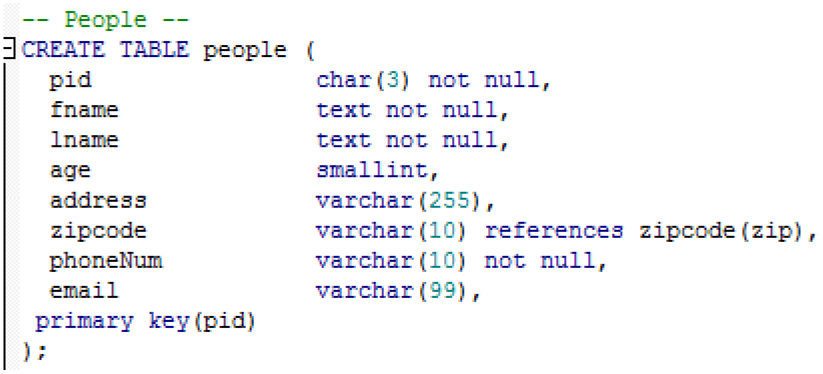
Purpose

To create a table with the People associated with the league. This data will help communication between the league officials, players, refs, parents, etc…

Functional Dependencies

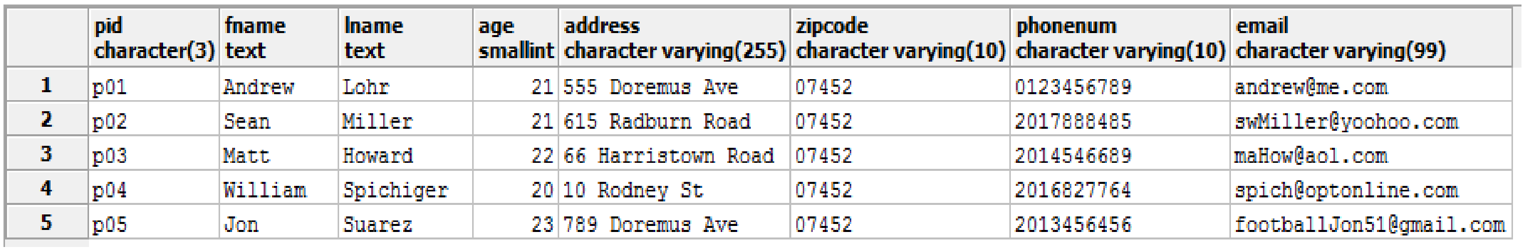
pid → fname, lname, age, address, zipcode, phoneNum, email

CREATE



```
-- People --
CREATE TABLE people (
  pid                  char(3) not null,
  fname                text not null,
  lname                text not null,
  age                  smallint,
  address              varchar(255),
  zipcode              varchar(10) references zipcode(zip),
  phoneNum             varchar(10) not null,
  email                varchar(99),
 primary key(pid)
);
```

SAMPLE



	pid character(3)	fname text	lname text	age smallint	address character varying(255)	zipcode character varying(10)	phonenum character varying(10)	email character varying(99)
1	p01	Andrew	Lohr	21	555 Doremus Ave	07452	0123456789	andrew@me.com
2	p02	Sean	Miller	21	615 Radburn Road	07452	2017888485	swMiller@yoohoo.com
3	p03	Matt	Howard	22	66 Harristown Road	07452	2014546689	maHow@aol.com
4	p04	William	Spichiger	20	10 Rodney St	07452	2016827764	spich@optonline.com
5	p05	Jon	Suarez	23	789 Doremus Ave	07452	2013456456	footballJon51@gmail.com

Table: coach

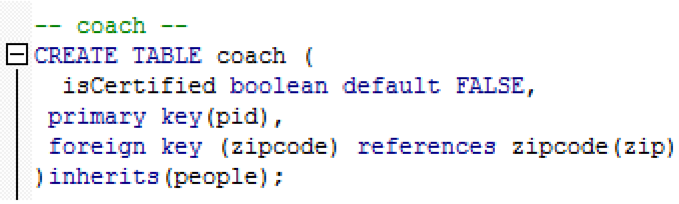
Purpose

To create table inherited from people with additional entries associated with being a coach in the league.

Functional Dependencies

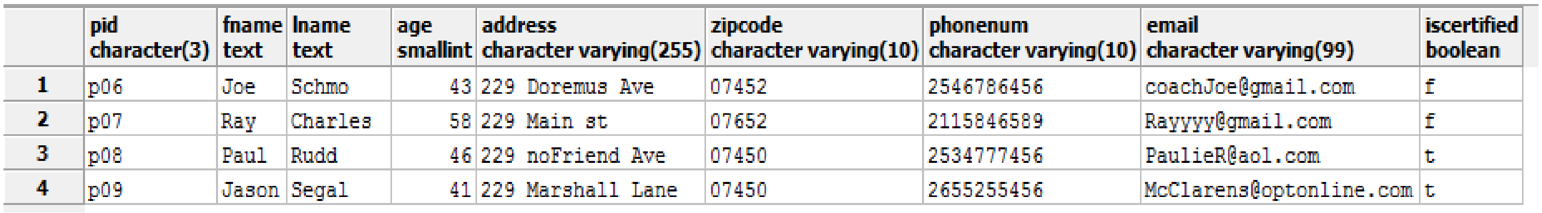
pid → fname, lname, age, address, zipcode, phoneNum, email, isCertified

CREATE



```
-- coach --
CREATE TABLE coach (
  isCertified boolean default FALSE,
 primary key(pid),
 foreign key (zipcode) references zipcode(zip)
)inherits(people);
```

SAMPLE



	pid character(3)	fname text	lname text	age smallint	address character varying(255)	zipcode character varying(10)	phonenum character varying(10)	email character varying(99)	iscertified boolean
1	p06	Joe	Schmo	43	229 Doremus Ave	07452	2546786456	coachJoe@gmail.com	f
2	p07	Ray	Charles	58	229 Main st	07652	2115846589	Rayyyy@gmail.com	f
3	p08	Paul	Rudd	46	229 noFriend Ave	07450	2534777456	PaulieR@aol.com	t
4	p09	Jason	Segal	41	229 Marshall Lane	07450	2655255456	McClarens@optonline.com	t

Table: player

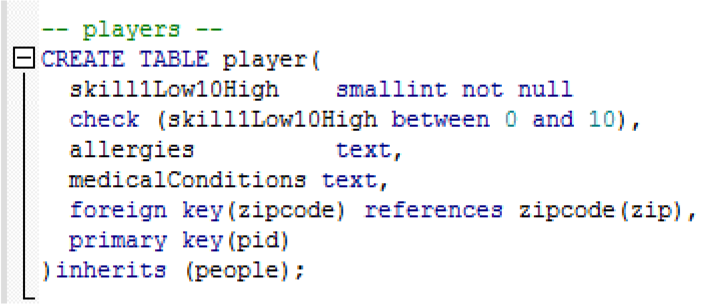
Purpose

To create table inherited from people with additional entries associated with being a player in the league. Will help with distinguishing players and creating information sheets for coaches so they know any important information for the kids on his team.

Functional Dependencies

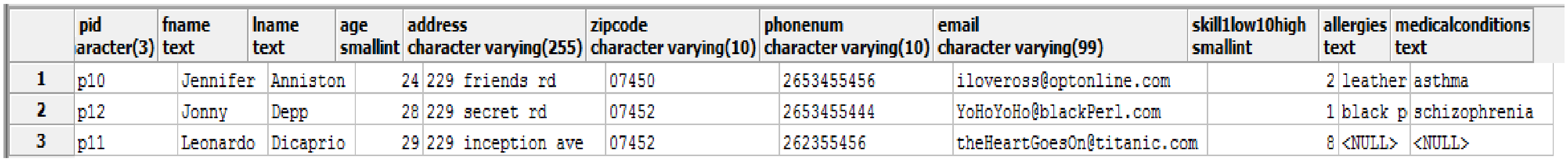
pid → fname, lname, age, address, zipcode, phoneNum, email, skill1low10high, allergies, medical conditions

CREATE



```
-- players --
CREATE TABLE player(
  skill1Low10High    smallint not null
  check (skill1Low10High between 0 and 10),
  allergies          text,
  medicalConditions text,
  foreign key(zipcode) references zipcode(zip),
  primary key(pid)
)inherits (people);
```

SAMPLE



	pid aracter(3)	fname text	lname text	age smallint	address character varying(255)	zipcode character varying(10)	phonenum character varying(10)	email character varying(99)	skill1low10high smallint	allergies text	medicalconditions text
1	p10	Jennifer	Anniston	24	229 friends rd	07450	2653455456	iloveross@optonline.com	2	leather	asthma
2	p12	Jonny	Depp	28	229 secret rd	07452	2653455444	YoHoYoHo@blackPerl.com	1	black p	schizophrenia
3	p11	Leonardo	Dicaprio	29	229 inception ave	07452	262355456	theHeartGoesOn@titanic.com	8	<NULL>	<NULL>

Table: administration

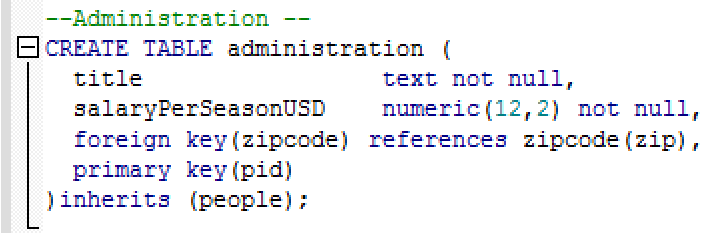
Purpose

To create table inherited from people with additional entries associated with being an administrator in the league. Will help with distinguishing administration and provide information such as salary and title.

Functional Dependencies

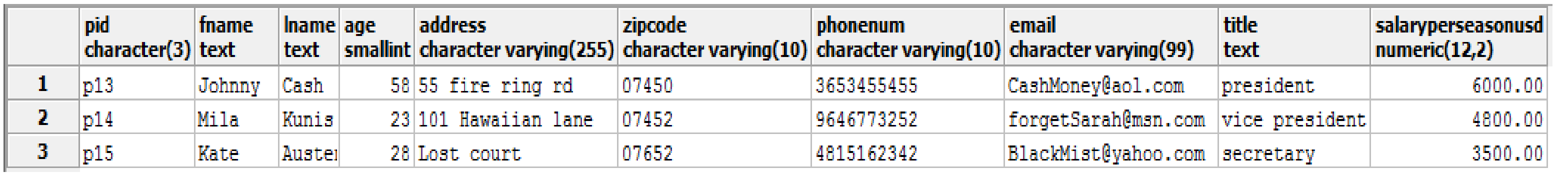
pid → fname, lname, age, address, zipcode, phoneNum, email, title, salaryPerSeasonUSD

CREATE



```
--Administration --
CREATE TABLE administration (
  title                    text not null,
  salaryPerSeasonUSD       numeric(12,2) not null,
  foreign key(zipcode) references zipcode(zip),
  primary key(pid)
)inherits (people);
```

SAMPLE



	pid character(3)	fname text	lname text	age smallint	address character varying(255)	zipcode character varying(10)	phonenum character varying(10)	email character varying(99)	title text	salaryperseasonusd numeric(12,2)
1	p13	Johnny	Cash	58	55 fire ring rd	07450	3653455455	CashMoney@aol.com	president	6000.00
2	p14	Mila	Kunis	23	101 Hawaiian lane	07452	9646773252	forgetSarah@msn.com	vice president	4800.00
3	p15	Kate	Auste	28	Lost court	07652	4815162342	BlackMist@yahoo.com	secretary	3500.00

Table: referee

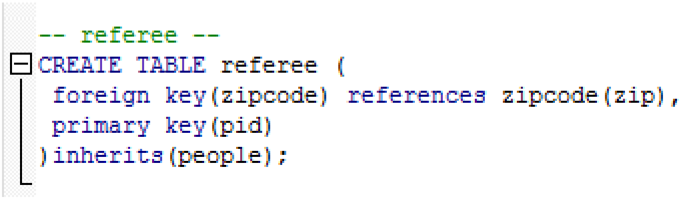
Purpose

To create table inherited from people that holds registered referee's that are available to the league.

Functional Dependencies

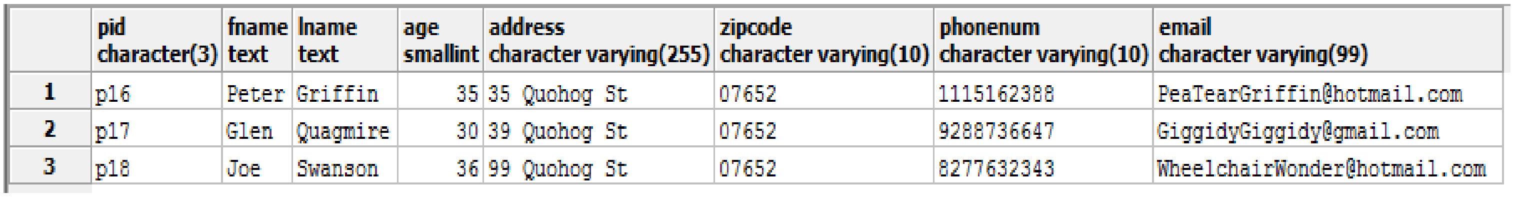
pid → fname, lname, age, address, zipcode, phoneNum, email

CREATE



```
-- referee --
CREATE TABLE referee (
 foreign key(zipcode) references zipcode(zip),
 primary key(pid)
)inherits(people);
```

SAMPLE



	pid	fname	lname	age	address	zipcode	phonenum	email
	character(3)	text	text	smallint	character varying(255)	character varying(10)	character varying(10)	character varying(99)
1	p16	Peter	Griffin	35	35 Quohog St	07652	1115162388	PeaTearGriffin@hotmail.com
2	p17	Glen	Quagmire	30	39 Quohog St	07652	9288736647	GiggidyGiggidy@gmail.com
3	p18	Joe	Swanson	36	99 Quohog St	07652	8277632343	WheelchairWonder@hotmail.com

Table: field

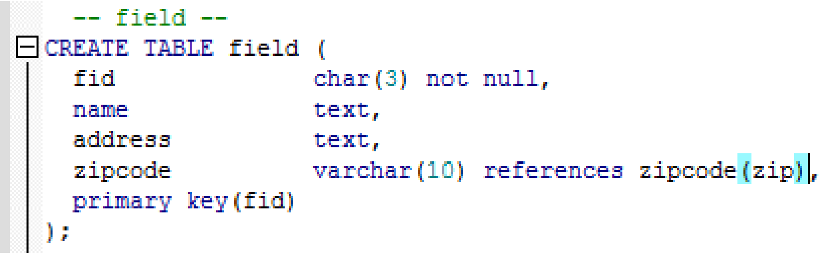
Purpose

To create table that includes all possible fields to play at.

Functional Dependencies

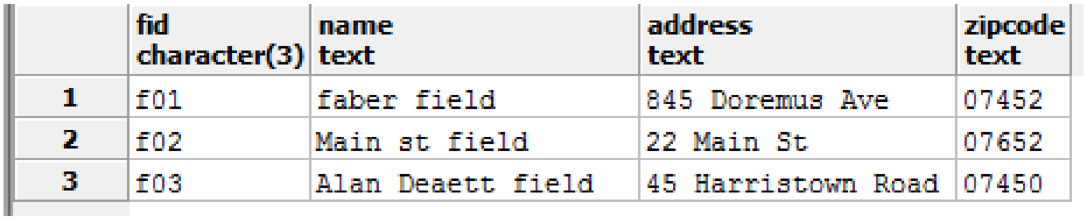
fid → name, address, zipcode

CREATE



```
  -- field --
CREATE TABLE field (
  fid                 char(3) not null,
  name                text,
  address             text,
  zipcode             varchar(10) references zipcode(zip),
  primary key(fid)
);
```

SAMPLE



	fid character(3)	name text	address text	zipcode text
1	f01	faber field	845 Doremus Ave	07452
2	f02	Main st field	22 Main St	07652
3	f03	Alan Deaett field	45 Harristown Road	07450

Table: team

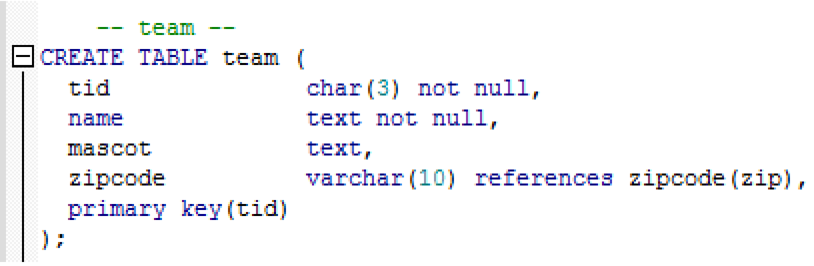
Purpose

To create table that shows all team names in the league

Functional Dependencies

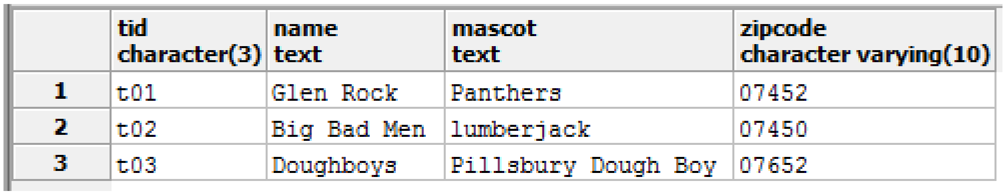
tid → name, mascot, zipcode

CREATE



```
    -- team --
CREATE TABLE team (
  tid                char(3) not null,
  name               text not null,
  mascot             text,
  zipcode            varchar(10) references zipcode(zip),
  primary key(tid)
);
```

SAMPLE



	tid character(3)	name text	mascot text	zipcode character varying(10)
1	t01	Glen Rock	Panthers	07452
2	t02	Big Bad Men	lumberjack	07450
3	t03	Doughboys	Pillsbury Dough Boy	07652

Table: teamRecord

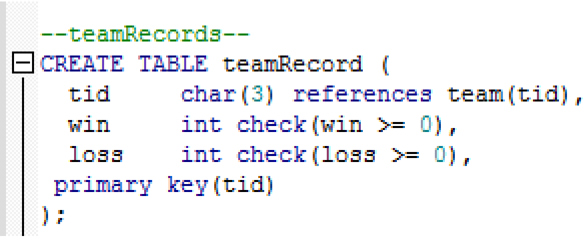
Purpose

To create table that shows all team wins / losses in the league

Functional Dependencies

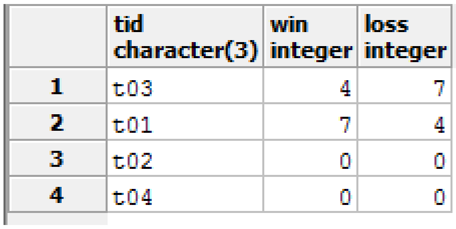
tid → win, loss

CREATE



```
--teamRecords--
CREATE TABLE teamRecord (
  tid      char(3) references team(tid),
  win      int check(win >= 0),
  loss     int check(loss >= 0),
 primary key(tid)
);
```

SAMPLE



	tid character(3)	win integer	loss integer
1	t03	4	7
2	t01	7	4
3	t02	0	0
4	t04	0	0

Table: teamLog

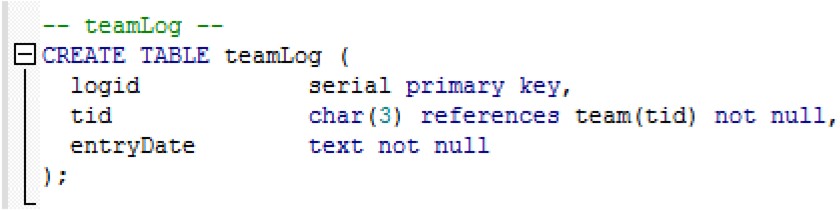
Purpose

To create table that audits whenever a new team is added to the league. This table is the result of a trigger when a new team is added to the table, team

Functional Dependencies

logid → tid, entryDate

CREATE



```
-- teamLog --
CREATE TABLE teamLog (
  logid             serial primary key,
  tid               char(3) references team(tid) not null,
  entryDate         text not null
);
```

NOTE: For Sample please refer to page 20.

Table: refField

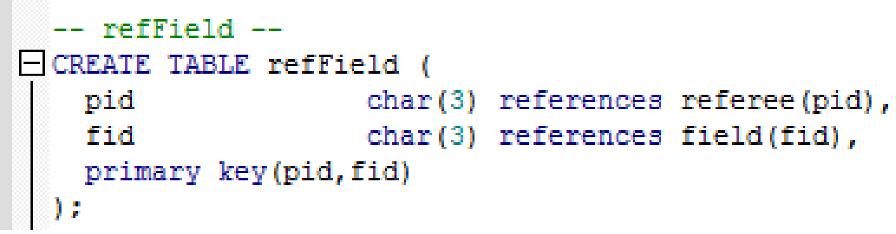
Purpose

To create table that shows which refs are responsible for which fields.

Functional Dependencies

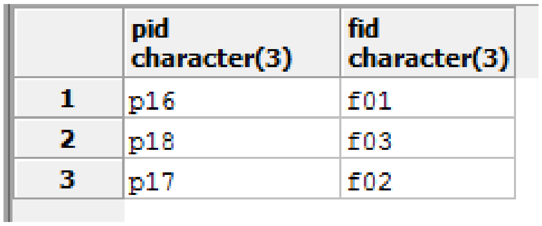
pid,rid →

CREATE



```
-- refField --
CREATE TABLE refField (
  pid                char(3) references referee(pid),
  fid                char(3) references field(fid),
  primary key(pid,fid)
);
```

SAMPLE



	pid character(3)	fid character(3)
1	p16	f01
2	p18	f03
3	p17	f02

Table: teamHomeField

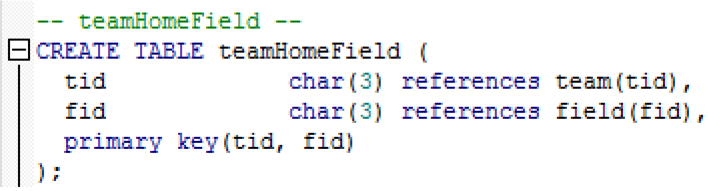
Purpose

To create table that shows which fields are the home field for which teams.

Functional Dependencies

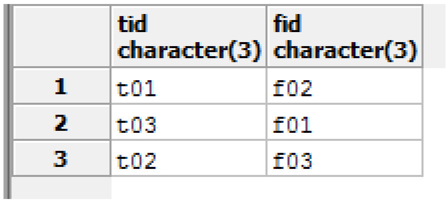
tid,fid →

CREATE



```
-- teamHomeField --
CREATE TABLE teamHomeField (
  tid               char(3) references team(tid),
  fid               char(3) references field(fid),
  primary key(tid, fid)
);
```

SAMPLE



	tid character(3)	fid character(3)
1	t01	f02
2	t03	f01
3	t02	f03

Table: teamCoach

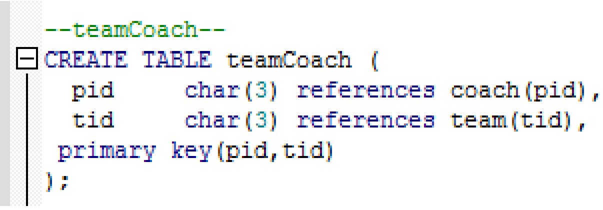
Purpose

To create table that shows which coaches coach which teams.

Functional Dependencies

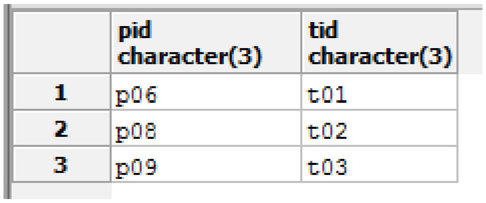
pid,tid →

CREATE



```
--teamCoach--
CREATE TABLE teamCoach (
  pid      char(3) references coach(pid),
  tid      char(3) references team(tid),
 primary key(pid,tid)
);
```

SAMPLE



	pid character(3)	tid character(3)
1	p06	t01
2	p08	t02
3	p09	t03

Table: playerTeam

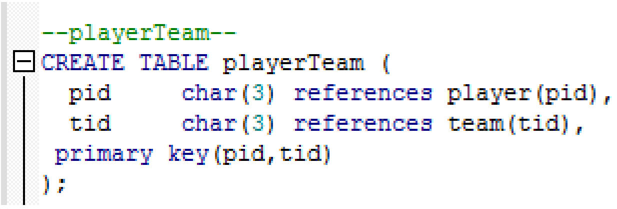
Purpose

To create table that shows which players belong to which teams.

Functional Dependencies

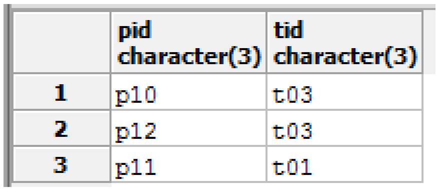
pid,tid →

CREATE



```
--playerTeam--
CREATE TABLE playerTeam (
  pid      char(3) references player(pid),
  tid      char(3) references team(tid),
 primary key(pid,tid)
);
```

SAMPLE

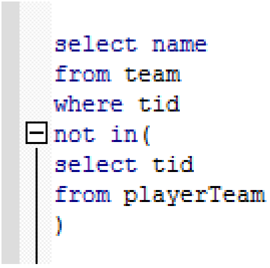


	pid character(3)	tid character(3)
1	p10	t03
2	p12	t03
3	p11	t01

Sample Queries.

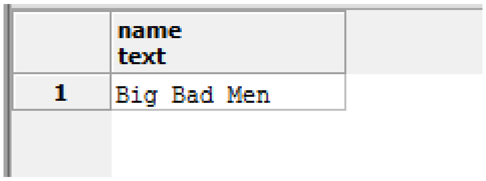
1. List all teams who have no players assigned to them.

QUERY



```
select name
from team
where tid
not in(
select tid
from playerTeam
)
```

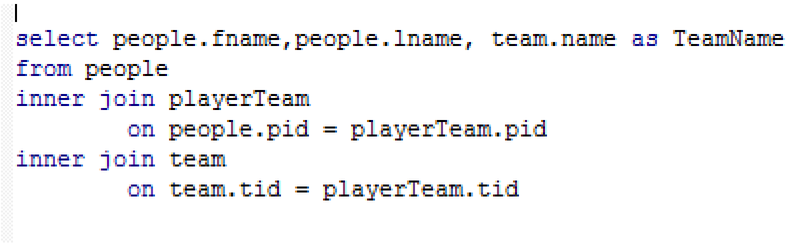
OUTPUT



	name text
1	Big Bad Men

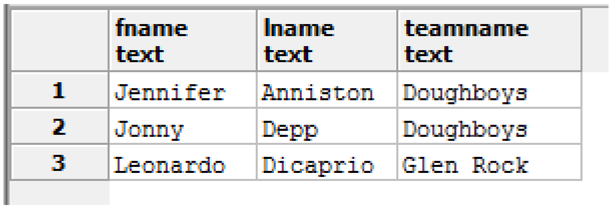
2. List All players and which team they belong to.

QUERY



```
select people.fname,people.lname, team.name as TeamName
from people
inner join playerTeam
        on people.pid = playerTeam.pid
inner join team
        on team.tid = playerTeam.tid
```

OUTPUT

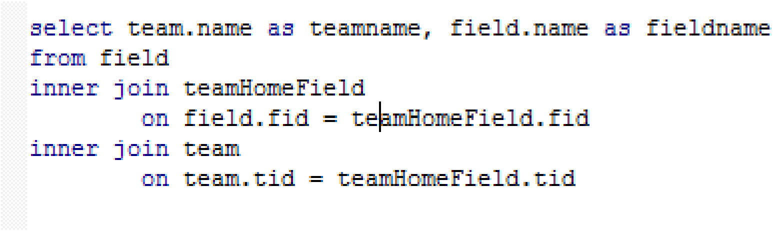


	fname text	lname text	teamname text
1	Jennifer	Anniston	Doughboys
2	Jonny	Depp	Doughboys
3	Leonardo	Dicaprio	Glen Rock

Sample Queries (cont.)

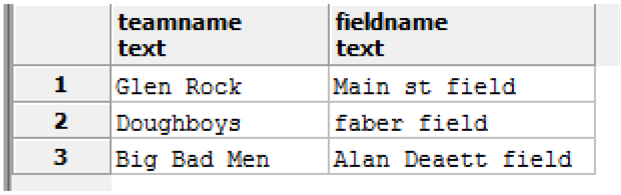
3. List the home fields of each team.

QUERY



```
select team.name as teamname, field.name as fieldname
from field
inner join teamHomeField
        on field.fid = teamHomeField.fid
inner join team
        on team.tid = teamHomeField.tid
```

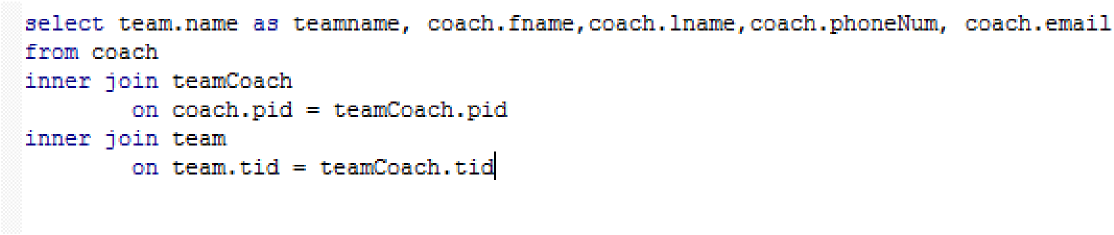
OUTPUT



	teamname text	fieldname text
1	Glen Rock	Main st field
2	Doughboys	faber field
3	Big Bad Men	Alan Deaett field

4. List the coaches of each team and their contact information

QUERY



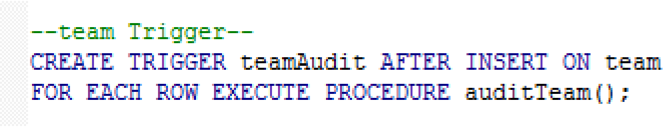
```
select team.name as teamname, coach.fname,coach.lname,coach.phoneNum, coach.email
from coach
inner join teamCoach
        on coach.pid = teamCoach.pid
inner join team
        on team.tid = teamCoach.tid
```

OUTPUT

TRIGGER

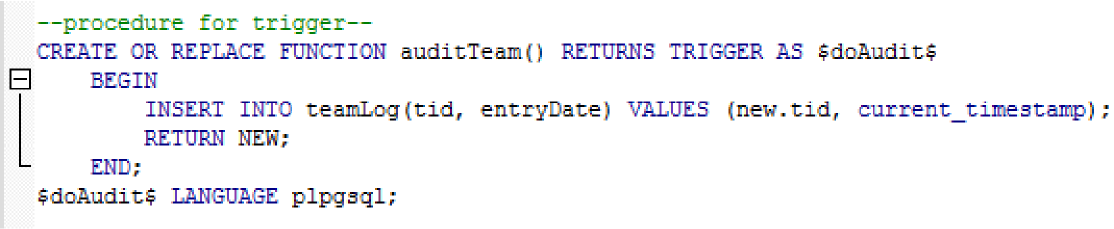
THIS TRIGGER'S RESPONSIBILITY IS TO HELP AUDIT (TIMESTAMP) WHEN NEW TEAMS ARE ENTERED INTO THE LEAGUE.

TRIGGER CREATE



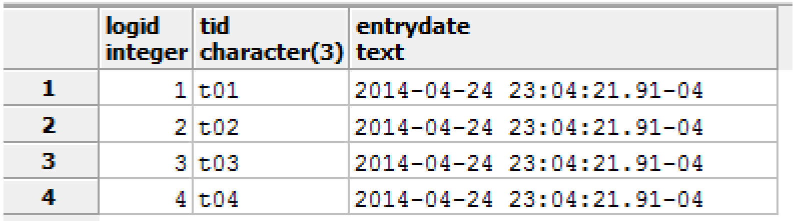
```
--team Trigger--
CREATE TRIGGER teamAudit AFTER INSERT ON team
FOR EACH ROW EXECUTE PROCEDURE auditTeam();
```

TRIGGER PROCEDURE



```
--procedure for trigger--
CREATE OR REPLACE FUNCTION auditTeam() RETURNS TRIGGER AS $doAudit$
    BEGIN
        INSERT INTO teamLog(tid, entryDate) VALUES (new.tid, current_timestamp);
        RETURN NEW;
    END;
$doAudit$ LANGUAGE plpgsql;
```

teamLog OUTPUT



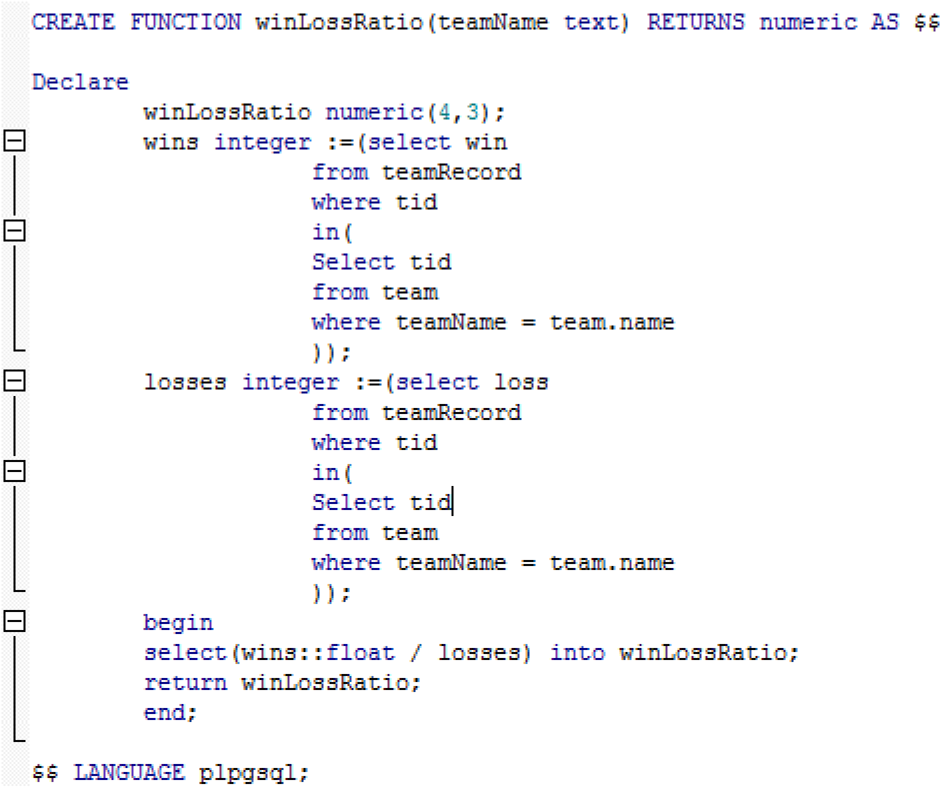
	logid	tid	entrydate
	integer	character(3)	text
1	1	t01	2014-04-24 23:04:21.91-04
2	2	t02	2014-04-24 23:04:21.91-04
3	3	t03	2014-04-24 23:04:21.91-04
4	4	t04	2014-04-24 23:04:21.91-04

PROCEDURES

FUNCTIONS HELP WITH DAY TO DAY CALCULATIONS IN THE LEAGUE DATABASE.

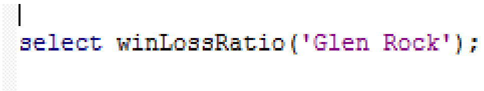
winLossRatio(text) Takes a team name as a parameter and returns that team's win / loss ratio.

FUNCTION



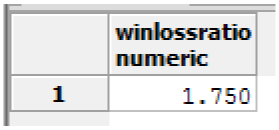
```
CREATE FUNCTION winLossRatio(teamName text) RETURNS numeric AS $$

Declare
        winLossRatio numeric(4,3);
        wins integer :=(select win
                    from teamRecord
                    where tid
                    in(
                    Select tid
                    from team
                    where teamName = team.name
                    ));
        losses integer :=(select loss
                    from teamRecord
                    where tid
                    in(
                    Select tid
                    from team
                    where teamName = team.name
                    ));
        begin
        select(wins::float / losses) into winLossRatio;
        return winLossRatio;
        end;

$$ LANGUAGE plpgsql;
```

QUERY



```
select winLossRatio('Glen Rock');
```

OUTPUT

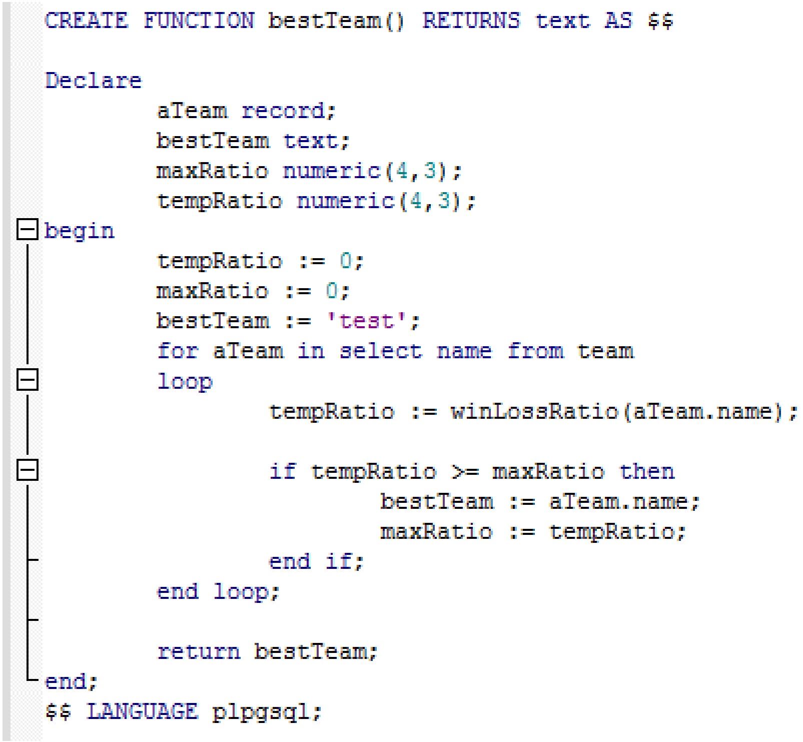


	winlossratio numeric
1	1.750

PROCEDURES (cont.)

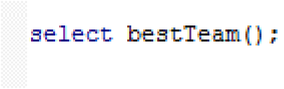
getBestTeam() Returns the team with the best Win/Loss Ratio. Uses function winLossRatio(text)

FUNCTION



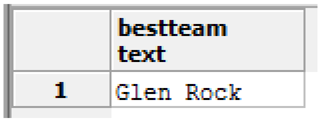
```
CREATE FUNCTION bestTeam() RETURNS text AS $$

Declare
        aTeam record;
        bestTeam text;
        maxRatio numeric(4,3);
        tempRatio numeric(4,3);
begin
        tempRatio := 0;
        maxRatio := 0;
        bestTeam := 'test';
        for aTeam in select name from team
        loop
                tempRatio := winLossRatio(aTeam.name);

                if tempRatio >= maxRatio then
                        bestTeam := aTeam.name;
                        maxRatio := tempRatio;
                end if;
        end loop;

        return bestTeam;
end;
$$ LANGUAGE plpgsql;
```

QUERY



```
select bestTeam();
```

OUTPUT

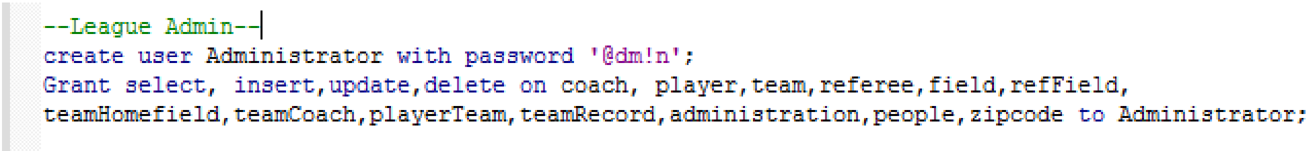


	bestteam text
1	Glen Rock

View & Security

League Admin

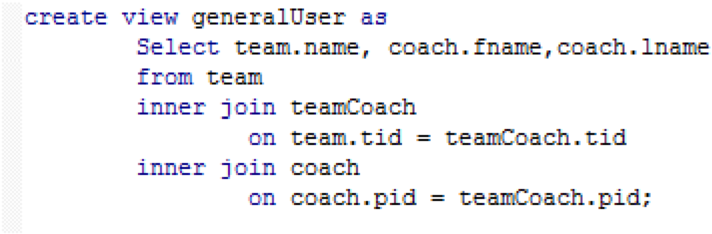
The league Administrator will have all privileges to select, insert, update, and delete all relations listed.



```
--League Admin--
create user Administrator with password '@dm!n';
Grant select, insert,update,delete on coach, player,team,referee,field,refField,
teamHomefield,teamCoach,playerTeam,teamRecord,administration,people,zipcode to Administrator;
```

General User

General users will have access to select a View which will currently show them all listed teams and their associated coach.



```
create view generalUser as
        Select team.name, coach.fname,coach.lname
        from team
        inner join teamCoach
                on team.tid = teamCoach.tid
        inner join coach
                on coach.pid = teamCoach.pid;
```

Implementation Notes

WHO IS THIS FOR AND WHAT IS NEXT?

The database implementation is designed for a small town to use for their junior league football programs. This database has the potential to be scalable, but would require much, much more work. There are check constraints on some of the database objects and more incorrect data would be detected before being entered into the database, for example checking to make sure a player's skill level is only between 0 and 10. With references between foreign keys and their respectable primary key we keep referential integrity throughout the database. A good example of that is the popular foreign key zip code in many of the tables. Overall the database needs more work before a lot can be demanded out of it.

Known Bugs / Problems & The Future!

The database is not 100% bug free, as is true with many things. We need to scale this up, up, and further up while keeping it concise. We will have to keep adding more check constraints as I believe it could easy to add incorrect data so some of the fields. The future looks bold and so does our bank account… maybe…hopefully… We would intend to make this database available to larger customers, while being able to keep the small towns happy with simplicity. After all the simpler, the better!