how i met your mother

Created by: Kelly Maud

December 2nd 2013

Table of contents

Executive Summary

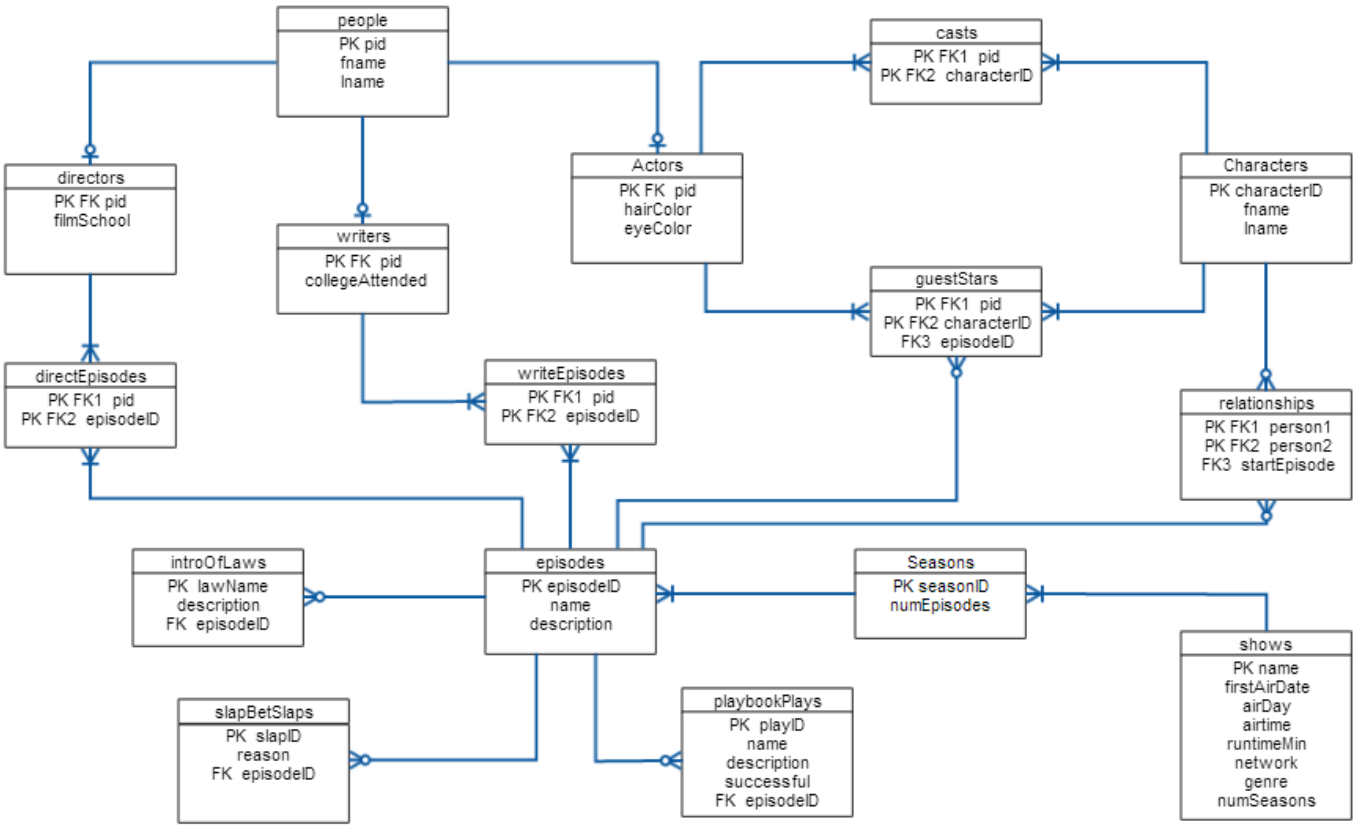
With 14 award winning prizes and 58 nominations, How I Met Your Mother is easily the best show on television. This sitcom details desperate bachelor Ted's epic search for his soul mate, told through flashbacks as an adult Ted recounts to his kids how he met their mom. As Ted bounces from one red herring to another, his best friends help keep him grounded. The nine amazing seasons have been nothing short of perfection. With millions of fans watching every Monday, a database of The Gang's adventures was necessary.

This design focuses on the minimal requirements to depict the full capability of each table. This is the first version and many future enhancements will be made. An overview of the database is presented below, followed by an explanation of each table and how they were each created.

The How I Met Your Mother database will be implemented as a set of normalized entities that are somehow related to each other. On first implementation the database will include people, directors, writers, actors, shows, seasons, episodes, characters, relationships and many other tables illustrating traditions unique to the show.

This design was targeted for and tested on PostgreSQL 9.3rc1.

Entity Relationship Diagram



Shows Table

Functional Dependencies

Name ➡ firstAirDate, airDay, airtime, runtimeMin, network, genre, numSeasons

Table Create Statements

```
CREATE TABLE shows (
name                    TEXT,
firstAirDate            DATE NOT NULL,
airDay                  TEXT,
airtime                 TEXT,
runtimeMin              INT,
network                 TEXT NOT NULL,
genre                   TEXT NOT NULL,
numSeasons              INT NOT NULL,
PRIMARY KEY (name)
);
```

Sample Data

name text	firstairdate date	airday text	airtime text	runtimemin integer	network text	genre text	numseasons integer
how i met your mother	2005-09-19	Monday	8:00 PM	30	CBS	Comedy	9

Seasons table

Functional Dependencies

seasonID ➡ numEpisodes

Table Create Statements

```
CREATE TABLE seasons (
seasonID          TEXT,
numEpisodes       INT,
PRIMARY KEY (seasonID)
);
```

Sample Data

seasonid text	numepisodes integer
s01	22
s02	22
s03	20
s04	24
s05	24
s06	24
s07	24
s08	24
s09	24

Episodes table

Functional Dependencies

episodeID ➡ name, description

Table Create Statements

```
CREATE TABLE episodes (
episodeID       TEXT,
name            TEXT NOT NULL,
description     TEXT,
PRIMARY KEY (episodeID)
);
```

Sample Data

episodeid text	name text	description text
s01e01	Pilot	Marshall proposes to Lily and Ted meets Robin
s01e08	The Duel	Ted and Marshall fight over the apartment
s01e10	The Pineapple Incident	Ted blacks out and tries to find out what happened
s01e15	Game Night	Marshalls game night leads to revelations
s01e22	Come On	Lily breaks up with Marshall, Ted and Robin get together
s02e09	Slap Bet	The gang finds out one of Robins secrets
s02e12	First Time in New York	Robins sister visits
s02e16	Stuff	Ted and Robin fight over items from previous relationships
s03e05	How I Met Everyone Else	Ted explains how he met the gang
s03e09	Slapsgiving	Marshall and Lily host Thanksgiving
s03e11	The Platnum Rule	The gang tries to stop Ted from taking out his doctor
s04e06	Happily Ever After	Ted gets left at the alter
s05e09	Slapsgiving 2: Revenge of the Slap	Marshall and Lily host Thanksgiving again
s06e11	The Mermaid Theory	Ted begins to question his relationship with Zoey
s07e05	Field Trip	Ted takes his class on a field trip
s07e09	Disaster Averted	The gang reminisces about hurricane Irene
s08e12	The Final Page (part 2)	Barney proposes to Robin
s08e18	Weekend at Barneys	Robin finds out Barney never destroyed the Playbook
s08e21	Romeward Bound	Lily and Marshall decide to move to Rome
s09e10	Bed time stories	Marshall tries to get Marvin to fall asleep

PlaybookPlays Table

Background: The Playbook is a book that Barney wrote. It is filled with plays, tricks, and pickup lines that he performs at the bar to hopefully pickup girls.

Functional Dependencies

playID ➡ name, description, successful, episodeID

Table Create Statements

```
CREATE TABLE playbookPlays (
playID          INT,
name            TEXT NOT NULL,
description     TEXT NOT NULL,
successful      BOOLEAN,
episodeID       TEXT references episodes(episodeID),
PRIMARY KEY (playID)
);
```

Sample Data

playid integer	name text	description text	successful boolean	episodeid text
1	The Robin	long list of how Barney gets Robin to marry him	t	s08e12
2	The Dont drink that	Barney "saves" a girl from drinking something poisonous	t	s03e05
3	The Naked Man	You wait in the girls apartment naked and hope she sleeps with you	t	s01e22
4	The Weekend at Barneys	Just like Weekend At Bernies	f	s08e18
5	The Special Delivery	Dress up like a UPS man delivering a package	f	s08e18
6	The Kidney	Pretend that you gave your friend a kidney so you look like a hero	f	s08e18
7	The "Ive Got A Pet Loch Ness Monster"	Pretend to have a pet Loch Ness Monster	f	s08e18

SlapBetSlaps Table

Background: The SlapBet originated in Season 2 episode 9. Originally when The Gang learned about Robin's fear of malls, Barney and Marshall bet on why Robin wouldn't go to malls. Barney bet that Robin was in an adult film and Marshall bet that Robin was married in a mall. Neither of them were right but Barney prematurely slapped Marshall upon hearing the news that Robin was never married. Marshall wins on a technicality that just because he was wrong doesn't mean that Barney was right. Marshall was given 5 slaps that may occur at any given time. In a later episode Marshall obtains 3 more slaps for letting Barney break a previous agreement.

Functional Dependencies

slapID ➡ reason, episodeID

Table Create Statements

```
CREATE TABLE slapBetSlaps (
slapID      INT,
reason      TEXT,
episodeID   TEXT references episodes(episodeID),
PRIMARY KEY (slapID)
);
```

Sample Data

slapid integer	reason text	episodeid text
1	Barney slaps Marshall because Robin is not married	s02e09
2	Marchall slaps Barney because he was prematurely slapped earlier	s02e09
3	Marshall slaps Barney to stop his boring play	s02e16
4	Marshall slaps Barney because the slapbet count down expired	s03e09
5	Marshall slaps Barney because it is slapsgiving	s05e09
6	Marshall is awarded 3 slaps because Barney wanted to take off the Ducky Tie and he slaps him	s07e09
7	Marshall slaps Barney twice in a row	s07e09

IntroOfLaws Table

Background: Barney has many theories and rules that apply to women, dating, and life in general. Many laws are reoccurring throughout the show but the table merely states the episode in which it was first introduced.

Functional Dependencies

lawName ➡ description, episodeID

Table Create Statements

```
CREATE TABLE introOfLaws (
lawName       TEXT,
description   TEXT,
episodeID     TEXT references episodes(episodeID),
PRIMARY KEY (LawName)
);
```

Sample Data

lawname text	description text	episodeid text
The Lemon Law	you can decide what you think of a date in the first 5 minutes	s01e08
The Mermaid Theory	A man will eventually want to sleep wiht any woman over time	s06e11
Hot Crazy Scale	a woman is allowed to be crazy only if she is equally as hot	s03e05
The Ewok Line	calculates birth year of a person based on weather or not they like ewoks	s07e05
The Freeway Theory	You can only exit a relationship after 6 hours, 4 days, 3 weeks, 7 months, 18 years or death	s02e12
International Date Line	Is it a date or is it not a date?	s09e10

People Table

Functional Dependencies

pid ➡ fname, lname

Table Create Statements

```
CREATE TABLE people (
pid      TEXT,
fname    TEXT NOT NULL,
lname    TEXT NOT NULL,
PRIMARY KEY (pid)
);
```

Sample Data

pid text	fname text	lname text
p01	Josh	Radnor
p02	Jason	Segal
p03	Cobie	Smulders
p04	Neil Patrick	Harris
p05	Alyson	Hannigan
p06	Bob	Saget
p07	Pamela	Fryman
p08	Carter	Bays
p09	Craig	Thomas
p10	Craig	Gerard
p11	Matthew	Zinman
p12	Becki	Newton
p13	Jennifer	Morrison
p14	Ashley	Williams
p15	Lucy	Hale
p16	Ellen	Williams
p17	Britney	Spears
p18	Mandy	Moore
p19	Danica	McKeller
p20	Benjamin	Koldyke
p21	Kal	Penn
p22	Michael	Trucco
p23	Enrique	Iglesias
p24	Hayes	MacArthur
p25	Nazanin	Boniadi
p26	Rob	Greenberg
p27	Chris	Harris
p28	Sarah	Chalke

Directors Table

Functional Dependencies

pid ➡ filmSchool

Table Create Statements

```
CREATE TABLE directors (
pid           TEXT references people(pid),
filmSchool    TEXT,
PRIMARY KEY (pid)
);
```

Sample Data

pid text	filmschool text
p07	New York Film Academy
p26	New York Film Academy

DirectEpisodes Table

Functional Dependencies

(pid, episodeID) ➡

Table Create Statements

```
CREATE TABLE directEpisodes (
pid         TEXT references people(pid),
episodeID   TEXT references episodes(episodeID),
PRIMARY KEY (pid, episodeID)
);
```

Sample Data

pid text	episodeid text
p26	s07e09
p07	s01e01
p07	s01e08
p07	s01e10
p07	s01e15
p07	s01e22
p07	s02e09
p07	s02e12
p07	s02e16
p07	s03e05
p07	s03e09
p07	s03e11
p07	s04e06
p07	s05e09
p07	s06e11
p07	s07e05
p07	s08e12
p07	s08e18
p07	s08e21
p07	s09e10

Writers Table

Functional Dependencies

pid ➡ collegeAttended

Table Create Statements

```
CREATE TABLE writers (
pid                 TEXT references people(pid),
collegeAttended     TEXT,
PRIMARY KEY (pid)
);
```

Sample Data

pid text	collegeattended text
p08	Wesleyan University
p09	Wesleyan University
p10	
p11	

WriteEpisodes Table

Functional Dependencies

(pid, episodeID) ➡

Table Create Statements

```
CREATE TABLE writeEpisodes (
pid          TEXT references people(pid),
episodeID    TEXT references episodes(episodeID),
PRIMARY KEY (pid, episodeID)
);
```

Sample Data

pid text	episodeid text
p08	s01e01
p09	s01e01
p11	s01e08
p08	s01e10
p09	s01e10
p27	s01e15
p08	s01e22
p09	s01e22
p11	s02e09
p11	s02e12
p27	s02e16
p11	s03e05
p08	s03e09
p09	s03e09
p08	s03e11
p09	s03e11
p27	s04e06
p11	s05e09
p11	s06e11
p27	s07e05
p08	s07e09
p09	s07e09
p08	s08e12
p09	s08e12
p27	s08e18
p08	s08e21
p08	s09e10
p09	s09e10

Actors Table

Functional Dependencies

pid ➡ hairColor, eyeColor

Table Create Statements

```
CREATE TABLE actors (
pid         TEXT references people(pid),
hairColor   TEXT,
eyeColor    TEXT,
PRIMARY KEY (pid)
);
```

Sample Data

pid text	haircolor text	eyecolor text
p01	Brown	Brown
p02	Brown	Brown
p03	Brown	Blue
p04	Blonde	Blue
p05	Red	Hazel
p06	Brown	Brown
p12	Blonde	Green
p13	Blonde	Blue
p14	Brown	Blue
p15	Brown	Brown
p16	Brown	Brown
p17	Blonde	Brown
p18	Brown	Hazel
p19	Brown	Brown
p20	Brown	Brown
p21	Brown	Brown
p22	Brown	Blue
p23	Brown	Brown
p24	Brown	Brown
p25	Brown	Brown
p28	Blonde	Blue

Characters Table

Functional Dependencies

characterID ➡ fname, lname

Table Create Statements

```
CREATE TABLE characters (
characterID      TEXT,
fname            TEXT NOT NULL,
lname            TEXT,
PRIMARY KEY (characterID)
);
```

Sample Data

characterid text	fname text	lname text
c001	Ted	Mosby
c002	Barney	Stinson
c003	Marshall	Eriksen
c004	Lily	Aldrin
c005	Robin	Scherbatsky
c006	Narrator	
c007	Victoria	
c008	Stella	Zinman
c009	Nora	
c010	Kevin	Venkataraghavan
c011	Quinn	Garvey
c012	Zoey	Pierson
c013	Don	Frank
c014	Katie	Scherbatsky
c015	Patrice	
c016	Trudy	
c017	Amy	
c018	Abby	
c019	Gael	
c020	Nick	Podarutti
c021	Curt	Irons

Casts Table

Functional Dependencies

(pid, characterID) ➡

Table Create Statements

```
CREATE TABLE casts (
pid           TEXT references people(pid),
characterID   TEXT references characters(characterID),
PRIMARY KEY (pid, characterID)
);
```

Sample Data

pid text	characterid text
p01	c001
p02	c003
p04	c002
p05	c004
p03	c005
p06	c006
p14	c007
p28	c008
p25	c009
p21	c010
p12	c011
p13	c012
p20	c013
p22	c020

GuestStars Table

Functional Dependencies

(pid, characterID) ➡ episodeID

Table Create Statements

```
CREATE TABLE guestStars (
pid           TEXT references people(pid),
characterID   TEXT references characters(characterID),
episodeID     TEXT references episodes(episodeID),
PRIMARY KEY (pid, characterID)
);
```

Sample Data

pid text	characterid text	episodeid text
p15	c014	s02e12
p16	c015	s08e12
p19	c016	s01e10
p18	c017	s03e05
p17	c018	s01e15
p23	c019	s03e05
p24	c021	s03e11

Relationships Table

Functional Dependencies

(person1, person2) ➡ startEpisode

Table Create Statements

```
CREATE TABLE relationships (
person1      TEXT references characters(characterID),
person2      TEXT references characters(characterID),
startEpisode TEXT references episodes(episodeID),
PRIMARY KEY (person1, person2)
);
```

Sample Data

person1 text	person2 text	startepisode text
c001	c005	s01e22
c001	c007	s01e15
c001	c008	s03e11
c001	c016	s01e10
c001	c012	s06e11
c001	c017	s03e05
c002	c005	s08e12
c002	c009	s06e11
c002	c011	s04e06
c002	c018	s01e15
c003	c004	s01e01
c005	c010	s06e11
c005	c019	s03e05
c005	c013	s04e06
c005	c021	s03e11

Views

Direct_guest_stars View

Purpose: This view will show who directed the episode that had a specific quest star appearance.

Create Statements:

```
CREATE VIEW direct_guest_stars (DirectorFirstName, DirectorLastName, GuestFirstName, GuestLastName) AS
SELECT p1.fname,
       p1.lname,
       p2.fname,
       p2.lname
FROM people p1,
     directors d,
     directEpisodes de,
     episodes e,
     guestStars g,
     actors a,
     people p2
WHERE p1.pid = d.pid
  AND d.pid = de.pid
  AND de.episodeID = e.episodeID
  AND e.episodeID = g.episodeID
  AND g.pid = a.pid
  AND a.pid = p2.pid;
```

Sample Query:

```
SELECT *
FROM direct_guest_stars
WHERE GuestFirstName = 'Britney'
  AND GuestLastName = 'Spears';
```

Query Output:

directorfirstname text	directorlastname text	guestfirstname text	guestlastname text
Pamela	Fryman	Britney	Spears

Writer_of_slaps View

Purpose: this view shows who wrote any episode where a slap from the slap bet occurred as well as the reason for the slap and the episodeID.

Create Statements:

```
CREATE VIEW writer_of_slaps AS
SELECT p.fname AS "First Name",
       p.lname AS "Last Name",
       s.reason,
       e.episodeID
FROM people p,
     writers w,
     writeEpisodes we,
     episodes e,
     slapBetSlaps s
WHERE p.pid = w.pid
  AND w.pid = we.pid
  AND we.episodeID = e.episodeID
  AND e.episodeID = s.episodeID
ORDER BY e.episodeID;
```

Sample Query:

```
SELECT *
FROM writer_of_slaps;
```

Query Output:

First Name text	Last Name text	reason text	episodeid text
Matthew	Zinman	Barney slaps Marshall because Robin is not married	s02e09
Matthew	Zinman	Marchall slaps Barney because he was prematurely slapped earlier	s02e09
Carter	Bays	Marshall slaps Barney because the slapbet count down expired	s03e09
Craig	Thomas	Marshall slaps Barney because the slapbet count down expired	s03e09
Matthew	Zinman	Marshall slaps Barney because it is slapsgiving	s05e09
Carter	Bays	Marshall is awarded 3 slaps because Barney wanted to take off the Ducky Tie	s07e09
Craig	Thomas	Marshall is awarded 3 slaps because Barney wanted to take off the Ducky Tie	s07e09
Carter	Bays	Marshall slaps Barney twice in a row	s07e09
Craig	Thomas	Marshall slaps Barney twice in a row	s07e09

Characters_in_relationships View

Purpose: This view gives all of the first names of characters that were once dating

Create Statements:

```
CREATE VIEW characters_in_relationships (Character1FirstName, Character2FirstName) AS
SELECT c1.fname,
       c2.fname
FROM characters c1,
     characters c2,
     relationships r
WHERE r.person1 = c1.characterID
  AND r.person2 = c2.characterID;
```

Sample Query:

```
SELECT *
FROM characters_in_relationships;
```

Query Output:

character1firstname text	character2firstname text
Ted	Robin
Ted	Victoria
Ted	Stella
Ted	Trudy
Ted	Zoey
Ted	Amy
Barney	Robin
Barney	Nora
Barney	Quinn
Barney	Abby
Marshall	Lily
Robin	Kevin
Robin	Gael
Robin	Don
Robin	Curt

Sample Query for just one character:

```
SELECT *
FROM characters_in_relationships
WHERE Character1FirstName = 'Robin'
   OR Character2FirstName = 'Robin';
```

Query Output:

character1firstname text	character2firstname text
Ted	Robin
Barney	Robin
Robin	Kevin
Robin	Gael
Robin	Don
Robin	Curt

Writer_Director_Episode

Purpose: this procedure will output the director and writer for a specific episode

Create Statements:

```
CREATE FUNCTION Writer_Director_Episode (episode_id TEXT)
RETURNS TABLE (DirectorFirst TEXT, DirectorLast TEXT, WriterFirst TEXT, WriterLast TEXT, Episode TEXT) AS $$
BEGIN
        RETURN QUERY SELECT p1.fname,
                            p1.lname,
                            p2.fname,
                            p2.lname,
                            e.episodeID
                     FROM people p1,
                          people p2,
                          directors d,
                          writers w,
                          directEpisodes de,
                          writeEpisodes we,
                          episodes e
                     WHERE p1.pid = d.pid
                       AND d.pid = de.pid
                       AND de.episodeID = e.episodeID
                       AND e.episodeID = we.episodeID
                       AND we.pid = w.pid
                       AND w.pid = p2.pid
                       AND e.episodeID = episode_id;
END
$$ LANGUAGE PLPGSQL;
```

Sample Query:

```
SELECT Writer_Director_Episode ('s01e01');
```

Query Output:

writer_director_episode record
(Pamela,Fryman,Carter,Bays,s01e01)
(Pamela,Fryman,Craig,Thomas,s01e01)

Reports and their queries

Every season must have total of 25 entries whether it is an episode or a special. This report calculates how many specials are needed in each season based on how many episodes are in each season.

Create Statements:

```
CREATE FUNCTION check_specials(season_id TEXT)
RETURNS NUMERIC AS $$
        DECLARE specials INTEGER;
        BEGIN
                specials = 25 - (SELECT numEpisodes
                                 FROM seasons
                                 WHERE seasonID = season_id);
                RETURN specials;
        END;
$$ LANGUAGE plpgsql;
```

Query:

```
SELECT seasonID,
       numEpisodes,
       check_specials (seasonID) AS num_specials_needed
FROM seasons;
```

Output:

seasonid text	numepisodes integer	num_specials_needed numeric
s01	22	3
s02	22	3
s03	20	5
s04	24	1
s05	24	1
s06	24	1
s07	24	1
s08	24	1
s09	24	1

Trigger Functions

Check_duplicate_people()

Purpose: To eliminate duplicate entries in the people table.

Create Statements:

```
CREATE FUNCTION check_duplicate_people()
RETURNS TRIGGER AS $check_duplicate_people$
        BEGIN
                IF EXISTS (SELECT *
                           FROM people
                           WHERE fname = NEW.fname
                             AND lname = NEW.lname)
                THEN
                    RAISE EXCEPTION 'Cannot duplicate entries in People';
                END IF;
                RETURN NEW;
        END;
$check_duplicate_people$ LANGUAGE plpgsql;
```

Triggers

Check_duplicate_people

```
CREATE TRIGGER check_duplicate_people
BEFORE INSERT ON people
FOR EACH ROW EXECUTE PROCEDURE check_duplicate_people();
```

Security

Two user-types were identified

- Admins: The admin performs general maintenance on the database such as inserting new episodes, writers, directors, guest stars etc. or updating information such as changed last names etc. These require full read and write capabilities to all views and base tables.

```
CREATE ROLE admins;

GRANT SELECT, INSERT, UPDATE
ON ALL TABLES IN SCHEMA PUBLIC
TO admins;
```

- Public User: the public user is restricted to select operations on the views and a few of the base tables. They should have minimal knowledge of the base tables.

```
CREATE ROLE public_user;

GRANT SELECT
ON characters_in_relationships,
   writer_of_slaps,
   direct_guest_stars,
   shows,
   seasons,
   episodes,
   slapBetSlaps,
   playbookPlays,
   introOfLaws
TO public_user;
```

Known Problems

- This current design only allows for one director and one writer per output in the writer_director_episodes view.
- Minimal information is provided for each episode.
- You are unable to query the database for all the episodes in a specific season.
- There is no end date for relationships.
- A relationship may be duplicated by adding "Robin and Ted" and "Ted and Robin"

Future Enhancements

- Add all the episodes… the episodes chosen were strictly to demonstrate the other tables such as guestStars and slapBetSlaps.
- Along with the episodes, almost all the other tables are not filled and the additional information is a necessary addition.
- Add a table for doppelgangers and another table for who is living in The Apartment for each episode.
- Update the descriptions of episodes and slaps to be longer and more informative
- Query the database for important times in the main characters lives.
- Update the relationships table to have an optional end date
- Add more stored procedures and views so the public_user can view those instead of some base tables.