(google images)

Game of Thrones TV Guide

Compiled by Chris Ognibene

CMPT 308L – 111

Professor Alan Labouseur

Design Project: 24 April 2014

Table of Contents

Executive Summary

Game of Thrones, the popular series by George R.R Martin, and a giant television franchise, has fans and viewers searching for easy-to-reach places to review their basic knowledge of Game of Thrones. The books and shows are extremely complex, with a lot of characters, locations, civilizations, creatures, relationships, deaths, marriages, among a whole assortment of information of which to remember. Wouldn't it be nice if much of the basic information were in one convenient *electronic* location, avoiding the need to flip through pages of guides and documentation?

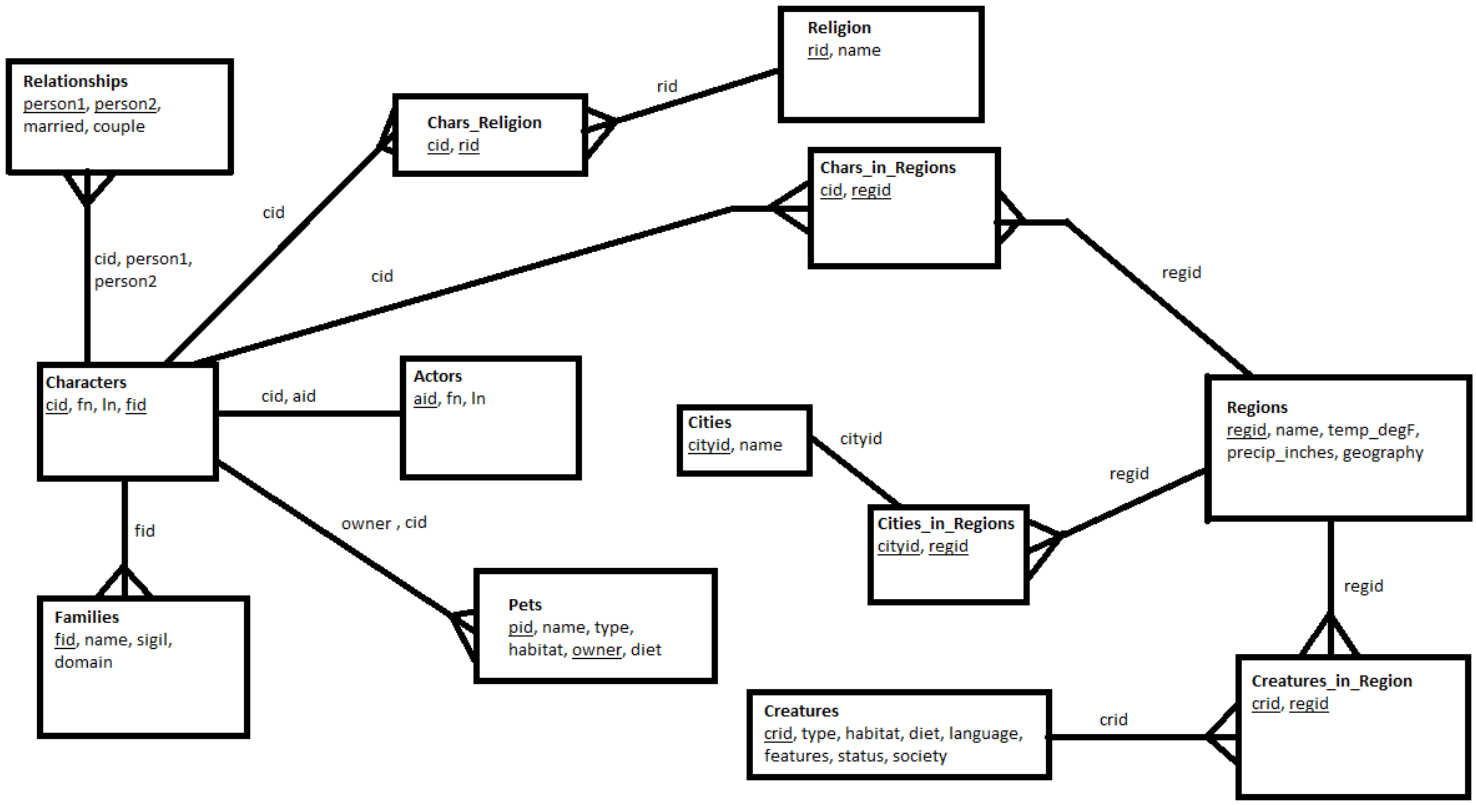
This database contains a collection of tables that promises quick access to basic character information, their associated actors and actresses, their royal families in the show, any religions they follow, any relationships in which they are involved, among other interesting facts. In this documentation, reports are produced, with a focus on generating tabular data sets of locations and cities that the characters have lived in or visited. Please note that there is a lot more information that may be extracted using creative queries, as the curious fan or database tester would desire based on his/her database experience.

It is the hope that this database and guide will serve as a handy resource as season four of the epic HBO series gets under way.

Disclaimer: Before reading any further, be warned that this database might contain spoilers of characters, locations, relationships, deaths, among other details. Query at your own risk!

Return to Table of Contents

Entity-Relationship Diagram



Return to Table of Contents

Tables Definitions:

Characters Table

Displays the ID number, first name, last name and family ID number of each character. Links table Characters with Families (soon to come) through foreign key, fid.

Create Statement

```
create table if not exists characters(

  cid varchar(5) not null unique primary key
, fn varchar(30) not null unique
, ln varchar(30)
, fid varchar(5) not null references families(fid)
);
```

Functional Dependencies

cid, fid → fn, ln

Sample Data

	cid character varying(5)	fn character varying(30)	ln character varying(30)	fid character varying(5)
1	c27	Petyr 'Littlefinge	Baelish	f14
2	c12	Stannis	Baratheon	f4
3	c4	Robert	Baratheon	f4
4	c11	Joffrey	Baratheon	f4
5	c19	Roose	Bolton	f9
6	c20	Gregor	Clegane	f10
7	c17	Walder	Frey	f7
8	c13	Theon	Greyjoy	f5
9	c15	Tywin	Lannister	f2
10	c2	Tyrion	Lannister	f2
11	c10	Jaime	Lannister	f2
12	c3	Cersei	Lannister	f2

Return to Table of Contents

Actors Table

Displays the ID number, first name, and last name of each actor/actress corresponding to their respective characters. Links tables Actors and Characters through foreign key, aid.

Create Statement

```
create table if not exists actors(

  aid varchar(5) not null unique references characters(cid)
, fn varchar(30) not null
, ln varchar(30) not null
);
```

Functional Dependencies

aid → fn, ln

Sample Data

	aid character varying(5)	fn character varying(30)	ln character varying(30)
1	c4	Mark	Addy
2	c13	Alfie	Allen
3	c1	Sean	Bean
4	c17	David	Bradley
5	c28	Oona	Chaplin
6	c5	Emilia	Clarke
7	c10	Nikolaj	Coster-Waldau
8	c15	Charles	Dance
9	c12	Stephen	Dillane
10	c2	Peter	Dinklage
11	c18	Natalie	Dormer
12	c9	Michelle	Fairley
13	c27	Aiden	Gillen

Return to Table of Contents

Families Table

Displays the ID number, name, sigil (or mascot), and governing domain of each royal family.

Create Statement

```
create table if not exists families(

  fid varchar(5) not null unique primary key
, name varchar(50) not null unique
, sigil varchar(50)
, domain varchar(50)
);
```

Functional Dependencies

fid → name, sigil, domain

Sample Data

	fid characte	name character varying(50)	sigil character varying(50)	domain character varying(50)
1	f1	Stark	Direwolf	North (Exiled)
2	f2	Lannister	Lion	Westerlands
3	f3	Targaryen	Three-Headed Dragon	Essos
4	f4	Baratheon	Stag	Stormlands; Crownlands
5	f5	Greyjoy	Golden Kraken	Iron Islands
6	f6	Night's Watch	Black	The Wall
7	f7	Frey	Two Towers and Bridge	Riverlands
8	f8	Tyrell	Golden Rose/Green Field	Reach
9	f9	Bolton	Red, upside down flayed man	North
10	f10	Clegane	Three Black Dogs/Yellow Backgı	Westerlands
11	f11	Tully	Silver Trout	Riverlands (Exiled)
12	f12	Martell	Red Sun/Golden Spear	Dorne
13	f13	Arryn	Moon-and-Falcon	Vale of Arryn
14	f14	Baelish	Head of the Titan of Braavos	Riverlands; Vale of Arryn
15	f15	Free Folk (Wildlings)	<NULL>	North of the Wall

Return to Table of Contents

Cities Table

Displays the ID number and name of each city.

Create Statement

```
create table if not exists cities(

  cityid varchar(10) not null primary key
, name varchar(30) not null
);
```

Functional Dependencies

cityid → name

Sample Data

	cityid character varying(10)	name character varying(30)
1	city1	Winterfell
2	city2	Westerlands
3	city3	The Stormlands
4	city4	The Realm
5	city5	Dragonstone
6	city6	Iron Islands
7	city7	The Wall
8	city8	Casterly Rock
9	city9	The Crossing
10	city10	The Reach
11	city11	Dreadfort
12	city12	Asshai
13	city13	Dothraki Sea
14	city14	Lorath
15	city15	North of the Wall
16	city16	King's Landing
17	city17	Volantis

Return to Table of Contents

Regions Table

Displays the ID number, name, temperature (in °F), precipitation (in inches), and geography of all the locations. As a future enhancement, the details of the final three attributes may be added as desired.

Create Statement

```
create table if not exists regions(

  regid varchar(10) not null unique primary key
, name varchar(30) not null
, temp_degF varchar(10)
, precip_inches varchar(10)
, geography varchar(30)
);
```

Functional Dependencies

regid → name, temp_degF, precip_inches, geography

Sample Data

	regid character varying(10)	name character varying(30)	temp_degf character varying(10)	precip_inches character varying(10)	geography character varying(30)
1	reg1	Westeros	<NULL>	<NULL>	<NULL>
2	reg2	Riverlands	<NULL>	<NULL>	<NULL>
3	reg3	The North	<NULL>	<NULL>	<NULL>
4	reg4	Essos	<NULL>	<NULL>	<NULL>
5	reg5	Dothraki	<NULL>	<NULL>	<NULL>
6	reg6	Craster's Keep	<NULL>	<NULL>	<NULL>

Return to Table of Contents

Cities_in_Regions Table

Displays the city ID number and the region ID number. Links the tables Cities and Regions together.

Create Statement

```
create table if not exists cities_in_regions(

  cityid varchar(10) not null references cities(cityid)
, regid varchar(10) not null references regions(regid)
);
```

Functional Dependencies

cityid, regid →

Sample Data

	cityid character varying(10)	regid character varying(10)
1	city1	reg1
2	city2	reg1
3	city3	reg1
4	city4	reg1
5	city5	reg1
6	city6	reg1
7	city7	reg1
8	city8	reg1
9	city9	reg2
10	city10	reg1
11	city11	reg3
12	city12	reg4
13	city13	reg5
14	city14	reg1
15	city15	reg1
16	city15	reg6
17	city16	reg1
18	city17	reg1

Return to Table of Contents

Chars_in_Regions

Displays the character ID number and region ID number. Links tables Characters and Regions together.

Create Statement

```
create table if not exists chars_in_regions(

  cid varchar(5) not null references characters(cid)
, regid varchar(10) not null references regions(regid)
);
```

Functional Dependencies

cid, regid →

Sample Data

	cid character varying(5)	regid character varying(10)
1	c1	reg1
2	c6	reg1
3	c7	reg1
4	c8	reg1
5	c9	reg1
6	c2	reg1
7	c3	reg1
8	c10	reg1
9	c4	reg1
10	c11	reg1
11	c12	reg1
12	c5	reg1
13	c13	reg1
14	c14	reg1
15	c15	reg1
16	c16	reg1
17	c17	reg2
18	c18	reg1
19	c19	reg3
20	c20	reg1
21	c21	reg4
22	c22	reg5
23	c23	reg1
24	c24	reg1
25	c25	reg1
26	c26	reg6
27	c27	reg1
28	c28	reg1
29	c29	reg6

Return to Table of Contents

Pets Table

Displays the ID number, name, type, habitat, owner, and diet of pets that characters own. Pets is linked to table Characters by foreign key "owner".

Create Statement

```
create table if not exists pets(

  pid varchar(10) not null unique primary key
, name varchar(50)
, type varchar(50)
, habitat varchar(50)
, owner varchar(5) not null references characters(cid)
, diet varchar(50)
);
```

Functional Dependencies

pid, owner → name, type, habitat, diet

Sample Data

	pid character	name character varyin	type character varying(5	habitat character varying(50)	owner characte	diet character va
1	p1	Drogon	dragon	Volcanic Mountains	c5	Carnivore
2	p2	Viserion	dragon	Volcanic Mountains	c5	Carnivore
3	p3	Rhaegal	dragon	Volcanic Mountains	c5	Carnivore
4	p4	Grey Wind	Direwolf	Forests,Mountains	c6	Carnivore
5	p5	Lady	Direwolf	Forests,Mountains	c8	Carnivore
6	p6	Nymeria	Direwolf	Forests,Mountains	c7	Carnivore
7	p7	Summer	Direwolf	Forests,Mountains	c9	Carnivore
8	p8	Ghost	Direwolf	Forests,Mountains	c15	Carnivore
9	p9	The Silver	Dothrak Horse	Plains	c5	Herbivore

Return to Table of Contents

Creatures Table

Displays the ID number, type, habitat, diet, language, features, status (whether extinct, living, etc.), and societal living unit (clan-based, pack-based, etc.) of creatures. Additional details may be filled in at the administrator's or editor's discretion (see "Security" section).

Create Statement

```
create table if not exists creatures(

  crid varchar(10) not null unique primary key
, type varchar(50) not null
, habitat varchar(50)
, diet varchar(50)
, language varchar(50)
, features varchar(50)
, status varchar(50)
, society varchar(50)
);
```

Functional Dependencies

crid → type, habitat, diet, language, features, status, society

Sample Data

	crid character	type character varying(50)	habitat character varying(50)	diet character va	language character v	features character varying(50)	status character	society character va
1	cr1	Manticore	Jade Sea-Essos	Carnivore	<NULL>	<NULL>	<NULL>	<NULL>
2	cr2	Raven	Forests,Plains,Wes	Omnivore	<NULL>	<NULL>	Active	<NULL>
3	cr3	Wight	Beyond the Wall,We	<NULL>	<NULL>	Glowing Blue Eyes	Active	<NULL>
4	cr4	White Walkers	Lands of Always Wi	<NULL>	<NULL>	Glowing blue eyes,pale and gaunt	Active	<NULL>
5	cr5	Giant	Beyond the Wall,We	<NULL>	<NULL>	Twice human height	Active	Clan-based
6	cr6	Children of the Forest	Westeros	<NULL>	<NULL>	human child height,4-fingered hands	Active	Clan-based

Return to Table of Contents

Creatures_in_Region Table

Displays the creature ID number and region ID number. Links tables Creatures and Regions together through foreign keys, crid and regid.

Create Statement

```
create table if not exists creatures_in_region(

  crid varchar(10) not null references creatures(crid)
, regid varchar(10) not null references regions(regid)
);
```

Functional Dependencies

crid, regid →

Sample Data

	crid character varying(10)	regid character varying(10)
1	cr1	reg4
2	cr2	reg1
3	cr3	reg3
4	cr4	reg3
5	cr5	reg3
6	cr6	reg2

Return to Table of Contents

Relationships Table

Displays the ID number of the first person, ID number of the second person, whether the two individuals are married, or together in a relationship. Links tables Characters and Relationships through foreign keys, person1 and person2. Additional information may be inserted

Create Statement

```
create table if not exists relationships(

  person1 varchar(5) not null references characters(cid)
, person2 varchar(5) not null references characters(cid)
, married varchar(30)
, couple varchar(30)
);
```

Functional Dependencies

person1, person2 → married, couple

Sample Data

	person1 character varying(5)	person2 character varying(5)	married character varying(30)	couple character varying(30)
1	c1	c9	yes	<NULL>
2	c6	c28	yes	<NULL>
3	c5	c22	yes	<NULL>
4	c14	c25	<NULL>	yes
5	c26	c29	<NULL>	yes
6	c2	c8	yes	<NULL>

Return to Table of Contents

Religion Table

Displays the ID number and name of each religion.

Create Statement

```
create table if not exists religion(

  rid varchar(5) not null unique primary key
, name varchar(50)
);
```

Functional Dependencies

rid → name

Sample Data

	rid character varying(5)	name character varying(50)
1	r1	Night's Watch Vows
2	r2	Seven Pointed Star
3	r3	Drowned Man
4	r4	Fiery Heart of the Lord Light
5	r5	Titan of Braavos
6	r6	The Thirteen (Council)
7	r7	Good Masters

Return to Table of Contents

Chars_Religion Table

Displays the character ID number and religion ID number. Links the tables Characters and Religion together to indicate religions that certain characters practice.

Create Statement

```
create table if not exists chars_religion(

  cid varchar(5) not null references characters(cid)
, rid varchar(5) not null references religion(rid)
);
```

Functional Dependencies

cid, rid →

Sample Data

	cid character varying(5)	rid character varying(5)
1	c14	r1
2	c26	r1
3	c21	r4

Return to Table of Contents

View Definitions:

Characters_in_families View

Displays the first name, last name, and family name of every character.

Create Statement

```
create or replace view characters_in_families
as
select c.fn, c.ln, f.name
from characters c,
     families f
where c.fid = f.fid;
```

Sample Output

	fn character varying(30)	ln character varying(30)	name character varying(50)
1	Eddard	Stark	Stark
2	Tyrion	Lannister	Lannister
3	Cersei	Lannister	Lannister
4	Robert	Baratheon	Baratheon
5	Daenerys	Targaryen	Targaryen
6	Robb	Stark	Stark
7	Arya	Stark	Stark
8	Sansa	Stark	Stark
9	Catelyn	Stark	Stark
10	Jaime	Lannister	Lannister
11	Joffrey	Baratheon	Baratheon
12	Stannis	Baratheon	Baratheon
13	Theon	Greyjoy	Greyjoy
14	Jon	Snow	Night's Watch
15	Tywin	Lannister	Lannister
16	Hodor	<NULL>	Stark
17	Walder	Frey	Frey
18	Margaery	Tyrell	Tyrell
19	Roose	Bolton	Bolton
20	Gregor	Clegane	Clegane
21	Melisandre	<NULL>	Baratheon
22	Drogo	<NULL>	Targaryen
23	Shae	<NULL>	Lannister
24	Osha	<NULL>	Stark
25	Ygritte	<NULL>	Free Folk (Wildlin
26	Samwell	Tarly	Free Folk (Wildlin
27	Petyr 'Littlefinge	Baelish	Baelish
28	Talisa	Stark	Stark
29	Gilly	<NULL>	Free Folk (Wildlin

Return to Table of Contents

Owned_pets View

Displays the first name, last name, pet name, and pet type of characters.

Create Statement

```
create or replace view owned_pets
as
select c.fn "owner first name", c.ln "owner last name", p.name
"pet name", p.type "pet type"
from characters c
   , pets p
where c.cid = p.owner;
```

Sample Output

	owner first name character varying(30)	owner last name character varying(30)	pet name character varying(50)	pet type character varying(50)
1	Daenerys	Targaryen	The Silver	Dothrak Horse
2	Daenerys	Targaryen	Rhaegal	dragon
3	Daenerys	Targaryen	Viserion	dragon
4	Daenerys	Targaryen	Drogon	dragon
5	Robb	Stark	Grey Wind	Direwolf
6	Arya	Stark	Nymeria	Direwolf
7	Sansa	Stark	Lady	Direwolf
8	Catelyn	Stark	Summer	Direwolf
9	Tywin	Lannister	Ghost	Direwolf

Return to Table of Contents

Characters_around_creatures View

Displays the character first name, character last name, and creature type of any characters who are around any creatures.

Create Statement

```
create or replace view characters_around_creatures
as
select distinct ch.fn "first name", ch.ln "last name", cr.type
"creature type", r.name "region"
from characters ch
   , chars_in_regions ch_r
   , regions r
   , creatures_in_region cr_r
   , creatures cr
where ch.cid = ch_r.cid
  and ch_r.regid = r.regid
  and r.regid = cr_r.regid
  and cr_r.crid = cr.crid
  and r.name = 'Essos';
```

Sample Output

	first name character varying(30)	last name character varying(30)	creature type character varying(50)	region character varying(30)
1	Melisandre	<NULL>	Manticore	Essos

Return to Table of Contents

Reports:

Relations Report

Displays the first person's first and last name, the second persons first and last name, and whether they are married or in a relationship.

Create Statement

```
create or replace view relations
as
(
select c1.fn as person1first, c1.ln as person1last, c2.fn as
person2first, c2.ln as person2last, married, couple
from characters c1
   , characters c2
   , relationships r
where c1.cid = r.person1
  and c2.cid = r.person2
);
```

Sample Output

	person1first character varying(30)	person1last character varying(30)	person2first character varying(30)	person2last character varying(30)	married character varying(30)	couple character varying(30)
1	Eddard	Stark	Catelyn	Stark	yes	<NULL>
2	Robb	Stark	Talisa	Stark	yes	<NULL>
3	Daenerys	Targaryen	Drogo	<NULL>	yes	<NULL>
4	Jon	Snow	Ygritte	<NULL>	<NULL>	yes
5	Samwell	Tarly	Gilly	<NULL>	<NULL>	yes
6	Tyrion	Lannister	Sansa	Stark	yes	<NULL>

Return to Table of Contents

Inhabitants_Casterly_Rock Report

Displays the character ID number, character first name, character last name, and the city name.

Specifically, the report compiles all the inhabitants of city Casterly Rock.

Create Statement

```
create or replace view inhabitants_Casterly_Rock
as
select distinct ch.cid, ch.fn, ch.ln, ci.name as "City Name"
from characters ch
   , chars_in_regions ch_reg
   , cities_in_regions ci_reg
   , regions reg
   , cities ci
where ch.cid = ch_reg.cid
  and ch_reg.regid = reg.regid
  and reg.regid = ci_reg.regid
  and ci_reg.cityid = ci.cityid
  and ci.name = 'Casterly Rock';
```

Sample Output

	cid character varying(5)	fn character varying(30)	ln character varying(30)	City Name character varying(30)
1	c14	Jon	Snow	Casterly Rock
2	c2	Tyrion	Lannister	Casterly Rock
3	c11	Joffrey	Baratheon	Casterly Rock
4	c13	Theon	Greyjoy	Casterly Rock
5	c10	Jaime	Lannister	Casterly Rock
6	c7	Arya	Stark	Casterly Rock
7	c5	Daenerys	Targaryen	Casterly Rock
8	c18	Margaery	Tyrell	Casterly Rock
9	c12	Stannis	Baratheon	Casterly Rock
10	c27	Petyr 'Littlefinge	Baelish	Casterly Rock
11	c1	Eddard	Stark	Casterly Rock
12	c8	Sansa	Stark	Casterly Rock
13	c15	Tywin	Lannister	Casterly Rock
14	c25	Ygritte	<NULL>	Casterly Rock
15	c28	Talisa	Stark	Casterly Rock
16	c24	Osha	<NULL>	Casterly Rock
17	c3	Cersei	Lannister	Casterly Rock
18	c4	Robert	Baratheon	Casterly Rock
19	c9	Catelyn	Stark	Casterly Rock
20	c6	Robb	Stark	Casterly Rock
21	c20	Gregor	Clegane	Casterly Rock
22	c23	Shae	<NULL>	Casterly Rock
23	c16	Hodor	<NULL>	Casterly Rock

Return to Table of Contents

Number_inhabitants Report

Displays the domain and the corresponding number of inhabitants.

Create Statement

```
create or replace view number_inhabitants
as
select f.domain, count(c.cid) "number of inhabitants"
from characters c
   , families f
where c.fid = f.fid
group by f.domain
order by "number of inhabitants" desc;
```

Sample Output

	domain character varying(50)	number of inhabitants bigint
1	North (Exiled)	8
2	Westerlands	6
3	Stormlands; Crownlands	4
4	North of the Wall	3
5	Essos	2
6	Riverlands	1
7	Iron Islands	1
8	Riverlands; Vale of Arryn	1
9	The Wall	1
10	North	1
11	Reach	1

Return to Table of Contents

Stored Procedures:

getCharInfo() function

This function that takes a character ID as input and returns his/her family information, *contingent* on the fact that they have an official relationship. In other words, if the character is not in the relationships table, then an empty relation is returned.

Create Statement

```
create or replace function getCharInfo(varchar,refcursor)
returns refcursor as
$$
declare
     char_id varchar := $1;
     resultset refcursor := $2;

begin
     open resultset for
          select ch1.fn as "Character FN", ch1.ln as "Character
LN", famil.name as "Family Name", famil.sigil "Family Mascot",
famil.domain "Family Domain",
               relat.married "Married?", relat.couple "Couple?",
ch2.fn "Partner's FN", ch2.ln "Partner's LN"
          from characters ch1
             , characters ch2
             , actors act
             , relationships relat
             , families famil
          where ch1.cid = act.aid
            and ch1.fid = famil.fid
            and ch1.cid = relat.person1
            and ch2.cid = relat.person2
            and ch1.cid = char_id;
     return resultset;
end;
$$
language plpgsql;
```

Sample Output

```
select getCharInfo('c1','results');
fetch all from results;
```

	Character FN character varyi	Character LN character vary	Family Name character varyi	Family Mascot character varying	Family Domain character varying(	Married? character v	Couple? character v	Partner's FN character varyi	Partner's LN character vary
1	Eddard	Stark	Stark	Direwolf	North (Exiled)	yes	<NULL>	Catelyn	Stark

Return to Table of Contents

getCharsFromCity() function

This function takes as input a city from the cities table, and returns all the characters from that particular city.

Create Statement

```
create or replace function getCharsFromCity(varchar,refcursor)
returns refcursor as
$$
declare
     city_input varchar := $1;
     resultset refcursor := $2;
begin
     open resultset for
          select distinct ch.fn, ch.ln, cit.name "City Name"
          from      cities cit
                , regions reg
                , chars_in_regions ch_reg
                , cities_in_regions ci_reg
                , characters ch
          where ch.cid = ch_reg.cid
            and ch_reg.regid = reg.regid
            and reg.regid = ci_reg.regid
            and ci_reg.cityid = cit.cityid
            and cit.name = city_input;
     return resultset;
end;
$$
language plpgsql;
```

Sample Output

```
select getCharsFromCity('The Stormlands','results');
fetch all from results;
```

	fn character varying(30)	ln character varying(30)	City Name character varying(30)
1	Gregor	Clegane	The Stormlands
2	Jaime	Lannister	The Stormlands
3	Tywin	Lannister	The Stormlands
4	Shae	<NULL>	The Stormlands
5	Arya	Stark	The Stormlands
6	Hodor	<NULL>	The Stormlands
7	Robb	Stark	The Stormlands
8	Theon	Greyjoy	The Stormlands
9	Osha	<NULL>	The Stormlands
10	Ygritte	<NULL>	The Stormlands
11	Eddard	Stark	The Stormlands
12	Stannis	Baratheon	The Stormlands
13	Margaery	Tyrell	The Stormlands
14	Sansa	Stark	The Stormlands
15	Talisa	Stark	The Stormlands
16	Catelyn	Stark	The Stormlands
17	Tyrion	Lannister	The Stormlands
18	Joffrey	Baratheon	The Stormlands
19	Petyr 'Littlefinge	Baelish	The Stormlands
20	Jon	Snow	The Stormlands
21	Robert	Baratheon	The Stormlands
22	Daenerys	Targaryen	The Stormlands
23	Cersei	Lannister	The Stormlands

Return to Table of Contents

Triggers:

check_characters trigger

The below trigger definition corresponds to the `getCharInfo()` function. It is supposed to check, after insertion into the characters table that there are invalid entries, or as specified in the parameter list, that there must be a cid value.

Create Statement

```
create trigger check_characters after insert on characters
for each row execute procedure getCharInfo(varchar,refcursor);
```

Return to Table of Contents

check_cities trigger

The below trigger definition corresponds to the `getCharsFromCity()` function. It is supposed to check, after an update of the cities table, for null values in the city ID and name value. If nulls are found, then the update should not occur.

Create Statement

```
create trigger check_cities after update on cities for each row
execute procedure getCharsFromCity(varchar,refcursor);
```

Return to Table of Contents

Security:

The three levels of access are assigned and implemented as follows.

Admin Role

```
create role admin;
grant all privileges
on all tables in schema public
to admin;
```

Return to Table of Contents

Editor Role

```
drop role editor;
create role editor;
grant select, insert, update
on all tables in schema public
to editor;
```

Return to Table of Contents

Viewer Role

```
drop role viewer;
create role viewer;
grant select
on all tables in schema public
to viewer;
```

Return to Table of Contents

Implementation Notes and Known Problems:

For the most part, implementation was not difficult. Basic planning and scrapwork was done using MS Paint software to draw the E-R diagrams. Microsoft Excel was then used to plan out the tables, allowing for easy reconstruction of table according to normalization rules. Once all of my sample data and primary and foreign keys were drawn up, the tables were then coded into PostgreSQL. An observation I witnessed many times is that referential integrity assures you of so much; as I inserted test data, foreign key-primary key relationships ensured that no data was erroneous nor inconsistent.

The one struggle that I faced was creating triggers based on my stored procedures. Since we had not done an example of writing a trigger in class in *Postgres*, I was unsure of the syntax. However, I utilized the PostgreSQL documentation on the website and wrote my triggers as best as they can. In fact, the triggers compiled almost completely, generating one error message saying the function (in both triggers) could not be recognized. I could not see how I had gone wrong, after successfully running both functions again and again.

Return to Table of Contents

Future Enhancements:

The most major upgrade that may occur in the future is plentiful additions to the database to include even more information about characters, more locations, and more attributes about characters. Immediate upgrades though that I would have liked to add would be a complete infusion of all the episodes aired to date, with their air date, season number, and episode number in the series. In addition, I might also consider attaching character foreign keys to episodes so that I can query exactly which episodes contained which characters. Likewise, the locations table can also be linked to the episodes so that I would be able to query which locations were visited in each episode.

I would also include a partnerships table of some sort, to show the alliances that form throughout the show, such as the bond between Arya Stark and the Hound, who offers Arya's protection.

Ultimately, though, I hope that I have developed a clear enough database as to allow future programmers to take full charge and full comprehension of the database in order to be able to go forth seamlessly, independently, and confidently and make any improvements to my work.

Return to Table of Contents