Lore Database

Bryan Rockwood

December 2, 2013

Table of Contents

Table of Contents (cont.)

Executive Summary

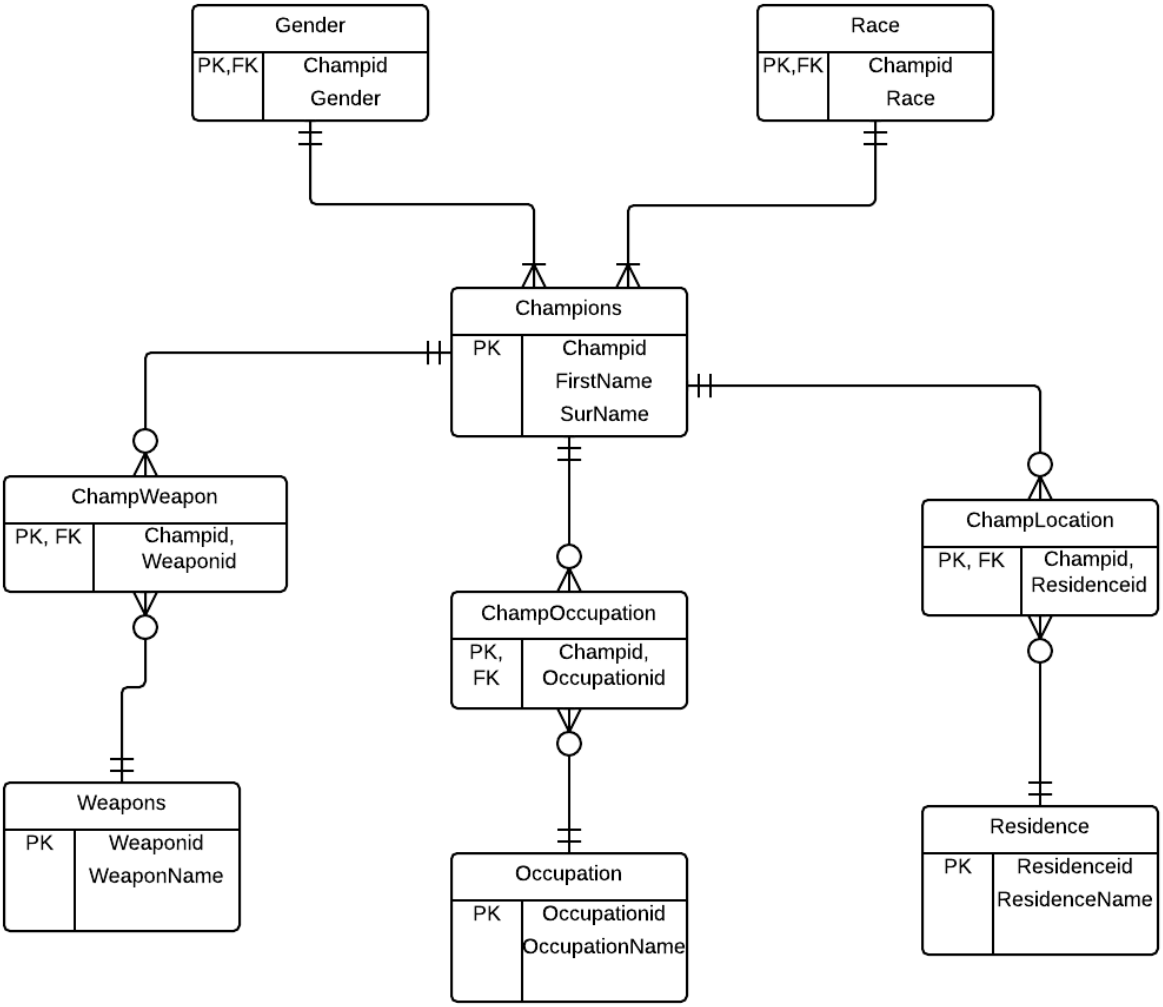
As of 2013, League of Legends is currently the most played videogame in the world. With 32 million active players, a game of this size has to have the content to keep players entertained. As the community grows, the developers at Riot Games work hard to create new characters and new play styles. But as new characters enter the canon, new elements are brought into the game to enhance the user experience.

One of the more overlooked and vital elements of League of Legends is character lore. As each character is introduced, they bring with them a background story; a life before entering the game. Where the majority of players do not care for each characters life story, those who do enjoy learning about their favorite character, need to read each character's lore to fully understand the game's world.

To help the fan base that drives their game, a lore-directed database will improve the efficiency of a user's ability to research League of Legends characters. Focusing on a character's important information – Residence, Weapon, Race, and Gender – a player can find a champion that can relate to them. With an interactive database, a player can learn about their favorite character quickly, or find a new character by their choosing a weapon and residence.

Although this database is not perfect, there are known improvements that can be made to further the effectiveness of this database.

Entity-Relationship Diagram



Create Table Statements

Champions Table

Here is the table that will show all of our current characters. The table needs to be flexible in order to accommodate for any further additions to the League of Legends roster.

Create Statement

```
CREATE TABLE IF NOT EXISTS Champions (
Champid    serial NOT NULL,
Name       varchar (20) NOT NULL,
SurName    varchar (20) NOT NULL,
Primary key (Champid)
 );
```

Functional Dependencies

Champid → Name, SurName

Sample Data

Champid	Name	SurName
1	Aatrox	The Darkin Blade
2	Arhi	The Nine-Tailed Fox
3	Akali	The Fist of Shadow
4	Alistar	The Minotaur
5	Amumu	The Sad Mummy
6	Anivia	The Cryophoenix
7	Annie	The Dark Child
8	Ashe	The Frost Archer
9	Blitzcrank	The Great Steam Golem
10	Brand	The Burniing Vengeance
11	Caitlyn	The Sheriff of Piltover
12	Cassiopeia	The Serpent's Embrace
13	Cho_Gath	The Terror of the Void
14	Corki	The Daring Bombardier
15	Darius	The Hand of Noxus
16	Diana	The Scorn of the Moon
17	Dr.Mundo	The Madman of Zaun
18	Draven	The Glorious Executioner
19	Elise	The Spide Queen

Weapons Table

The weapons table is used to illustrate the variety of different weapons featured in the game. A user can get a feel for the creativity used in the game and he or she can choose a character based on a certain weapon.

Create Statement

```
CREATE TABLE IF NOT EXISTS Weapons (
Weaponid        varchar (10)NOT NULL,
WeaponName      varchar (20) NOT NULL,
Primary key (Weaponid)
);
```

Functional Dependencies

Weaponid → WeaponName

Sample Data

Weaponid	WeaponName
w001	Darkin Blade
w002	Spriit Orb
w003	Kamas
w004	Fists
w005	Bandages
w006	Ice
w007	Fire
w008	Ice Enchanted Bow
w009	Rifle
w010	Poison
w011	Mouth
w012	ROFL Copter
w013	Gigantic Axe
w014	Crescent Blade
w015	Cleaver
w016	Throwing Axes
w017	Spiders

Occupation Table

Although each character in the game is an excellent fighter, they do also have lives outside of the League. This table helps display jobs and positions characters possess outside of the standard scope of the game.

Create Statement

```
CREATE TABLE IF NOT EXISTS Occupation (
Occupationid            varchar (10) NOT NULL,
OccupationName          varchar (20) NOT NULL,
Primary key (Occupationid)
);
```

Functional Dependencies

Occupationid → OccupationName

Sample Data

Occupationid	OccupationName
OC001	Soldier for Hire
OC002	Student
OC003	Ninja
OC004	Philantropist
OC005	Sheriff
OC006	Elemental Guardian
OC007	Daughter
OC008	Leader
OC009	Unemployed
OC010	Pirate
OC011	Pilot
OC012	Assassin
OC013	Executioner

Residence Table

In the League of Legends lore, there are many different cities and provinces that encompass the world. Each area is unique, and each character is shaped by their surroundings. As a player, a user can see where a character is from, and make a decision based on the characters residence.

Create Statement

```
CREATE TABLE IF NOT EXISTS Residence (

Residenceid serial NOT NULL,

ResidenceName varchar (20) NOT NULL,

Primary key (Residenceid)

);
```

Functional Dependencies

Residenceid → ResidenceName

Sample Data

Residenceid	ResidenceName
r001	Nomad
r002	Ionia
r003	Bandle City
r004	Freljiord
r005	VooDoo Lands
r006	Zaun
r007	Demacia
r008	Piltover
r009	Shurima
r010	Void
r011	Noxus
r012	Shadow Isles
r013	Bilgewater

Race Table

In a game filled with characters from various backgrounds, it is important to know what your character is as a being. League of Legends is home of several different types of champions from a wide array of backgrounds, and this table helps display the various races that exist in this universe.

Create Statement

```
CREATE TABLE IF NOT EXISTS Race (
Champid      serial NOT NULL REFERENCES Champions (Champid),
Race   varchar (20) NOT NULL,
Primary key (Champid)
);
```

Functional Dependencies

Champid → Race

Sample Data

Champid	Race
1	Human
2	Half-Human
3	Yordle
4	Golem
5	Elemental Creature
6	Undead
7	Half-Human
8	Angel
9	Cyborg
10	Extra-Planar Being

Gender Table

The gender table is needed to show that there is a fair balance between men and women fighters inside the League of Legends lore. This table is also helpful for the player who may not notice which characters are what gender.

Create Statement

```
CREATE TABLE IF NOT EXISTS Gender (
Champid      serial NOT NULL REFERENCES Champions (Champid),
Gender       varchar (8) NOT NULL,
Primary key (Champid)
);
```

Functional Dependencies

Champid → Gender

Sample Data

Champid	Gender
1	Male
2	Female

ChampWeapon

This table is created to avoid the many to many relationship that would be created by connecting the champions table and the weapons table.

Create Table Statement

CREATE TABLE ChampWeapon (

Champid serial NOT NULL REFERENCES Champions (Champid),

Weaponid varchar (10) NOT NULL REFERENCES Weapons (Weaponid),

primary key (Champid,Weaponid)

);

Functional Dependencies

None. These are not unique keys.

Sample Data

Champid	Weaponid
1	w001
2	w002
3	w003
4	w004
5	w005
6	w006
7	w007
8	w008
9	w009
10	w010
11	w011
12	w012
13	w013
14	w014
15	w015
16	w016
17	w017

ChampResidence

This table is created to avoid the many to many relationship that would be created by connecting the champions table and the residence table.

Create Table Statement

```
CREATE TABLE ChampResidence(
 Champid     serial NOT NULL REFERENCES Champions (Champid),
 Residenceid varchar (10) NOT NULL REFERENCES Residence (Residenceid),
 primary key (Champid,Residenceid)
 );
```

Functional Dependencies

None. These are not unique keys.

Sample Data

Champid	Residenceid
1	r001
2	r002
3	r003
4	r004
5	r005
6	r006
7	r007
8	r008
9	r009
10	r010
11	r011
12	r012
13	r013
14	r014

ChampOccupation

This table is created to avoid the many to many relationship that would be created by connecting the champions table and the occupation table.

Create Table Statement

CREATE TABLE ChampResidence(

Champid serial NOT NULL REFERENCES Champions (Champid),

Occupationid varchar (10) NOT NULL REFERENCES Occupation (Occupationid),

primary key (Champid,Occupationid)

);

Functional Dependencies

None. These are not unique keys.

Sample Data

Champid	Occupationid
1	OC001
2	OC002
3	OC003
4	OC004
5	OC005
6	OC006
7	OC007
8	OC008
9	OC009
10	OC010
11	OC011
12	OC012
13	OC013

Views

Here we see a view that showcases how to find a character based on their residence. The residence, Piltover for example, is a great starting point to determine what character to play because of the personality of each area. We will try to find a female human as our first example.

```
Create View PiltoverChampions

As

Select C.Name, C.SurName, Re. ResidenceName, Ra.Race,
G.GenderType

From Champions C, Residence Re, Race Ra, Gender G

Where Re.ResidenceName = "Piltover"

And    Ra.Race = "Human"

And   G.Gender = "Female"

OrderBy

C.Name ASC ;
```

Reports and Queries

Here we have some potential query examples that would occur in this database. First we have someone looking for a character that was once an executioner.

```
Select O.OccupationName, C.Champid, C.Name, C.SurName
From Occupation O, Champions C
Where O.OccupationName = "Executioner"
And C.Champid = O.Occupationid;
```

Another query can help a user find all of the female half-humans in the database.

```
Select g.gender, r.Race, c.champid, c.name, c.surname
From Gender g, Race r, Champions c
Where Gender = "Female"
And     Race = " Half-Human"
```

Stored Procedures

The stored procedure is used to make sure that each champion has a weapon associated with their character. This will avoid any nulls that may occur in the table.

```
CREATE FUNCTION WeaponAssigned (w.WeaponName text)
    RETURNS Weapons AS $$
BEGIN
    IF Weaponid is NULL or Weaponid <1
    THEN
    return "unarmed" ;
End IF;
Select w.weaponid
From Weapons w, champion c
Where w.weaponid = c.champid
End
    Language plpgsql;
```

Triggers

This trigger is currently in place to make sure each character is being assigned "unarmed" if they are not given a weapon.

```
CREATE TRIGGER WeaponAssignCheck

AFTER INSERT ON Weapons

FOR EACH ROW EXECUTE PROCEDURE WeaponAssigned();

Select Weaponid

From Weapons

Where WeaponName = "Unarmed"
```

Security

Security is very important, especially with a game of this magnitude. Any change in the database can change the lore for the worst. So in order to avoid all problems regarding security, we will create an admin that will control the information in the database.

Admin Role

```
Create User LeagueAdmin With Password 'Alpaca' ;
Revoke all on Champions From LeagueAdmin;
Revoke all on Weapons From League Admin;
Revoke all on Residence From LeagueAdmin;
Revoke all on Occupation From League Admin;
Revoke all on Gender From League Admin;
Revoke all on Race From LeagueAdmin;
Revoke all on ChampWeapon From League Admin;
Revoke all on ChampResidence From LeagueAdmin;
Revoke all on ChampOccupation From League Admin;
```

```
Grant insert, update, and remove on Champions
    to LeagueAdmin;
Grant insert, update, and remove on Weapons
    to LeagueAdmin;
Grant insert, update, and remove on Residence
    to LeagueAdmin;
Grant insert, update, and remove on Occupation
    to LeagueAdmin;
Grant insert, update, and remove on Race
    to LeagueAdmin;
Grant insert, update, and remove on Gender
    to LeagueAdmin;
Grant insert, update, and remove on ChampWeapons
    to LeagueAdmin;
Grant insert, update, and remove on ChampResidence
    to LeagueAdmin;
Grant insert, update, and remove on ChampOccupation
    to LeagueAdmin;
```

User Role

```
Create User LeagueUser With Password 'Alapacauser' ;
Revoke all on Champions From LeagueUser;
Revoke all on Weapons From LeagueUser;
Revoke all on Residence From LeagueUser;
Revoke all on Occupation From LeagueUser;
Revoke all on Race From LeagueUser;
Revoke all on Gender From LeagueUser;
Revoke all on ChampWeapon From LeagueUser;
Revoke all on ChampResidence From LeagueUser;
Revoke all on ChampOccupation From LeagueUser;
Grant Select on Champions to LeagueAdmin;
Grant Select on Weapons to LeagueAdmin;
Grant Select on Residence to LeagueAdmin;
Grant Select on Occupation to LeagueAdmin;
Grant Select on Race to LeagueUser;
Grant Select on Gender to LeagueAdmin;
Grant Select on ChampWeapons to LeagueAdmin;
Grant Select on ChampResidence to LeagueAdmin;
```

Grant Select on ChampOccupation to LeagueAdmin;

We do not want the user to be changing data, so we do not grant them access to change information.

Implementation

Implementing this database would be simple and hassle free. Adding a search function on the League of Legends website would grant ease of access to the public. Here on an international platform, even casual players could use the database to find information about the many characters in the game.

Known Problems

One of the major known problems is the inherent volatile nature of League of Legends. The game is constantly changing, and is very prone to massive shifts in lore. The developers continue to expand the League of Legends universe, and thus will continue to improve upon character's lore. As more characters are added, more need to be connected to fit within the league.

In addition to the constantly changing nature of this game, the data within the game is massive. There is a very high possibility of human error that could cause the database to not work properly.

On a technical note, currently there is no check for characters regarding rivalries and familial ties. If a user were to look up what characters were related to who, the query would come up empty, and the user would have to search through each individual character lore to figure out the relationship between characters.

During the time of writing, the use of PostGres was unavailable for testing due to computer issues. This led to the use of the command shell to run all SQL code. As testing continued, the use of the command shell became detrimental and thus caused most of the SQL queries to not run properly.

Future Improvements

In the near future, the database hopes to continue growing and expanding, but manageable at the same rate. Although a tough task, it is necessary to be able to accommodate with the ever growing League of Legends community and the ever growing League of Legends game.

I plan on creating a more in depth query that will allow a user to see all the rivals and allies associated with a searched champion. This will provide a deeper look into the background information of any given character.

A greater description of weapons and occupation is also in need of adjusting, the current values in these tables are suited only for the scope of this paper, but are in need of further elaboration.