DotA 2

-Defense of the Ancients 2-

Database

Kevin Skocypec
December 1, 2014
CMPT 308L-113
Professor Alan Labouseur

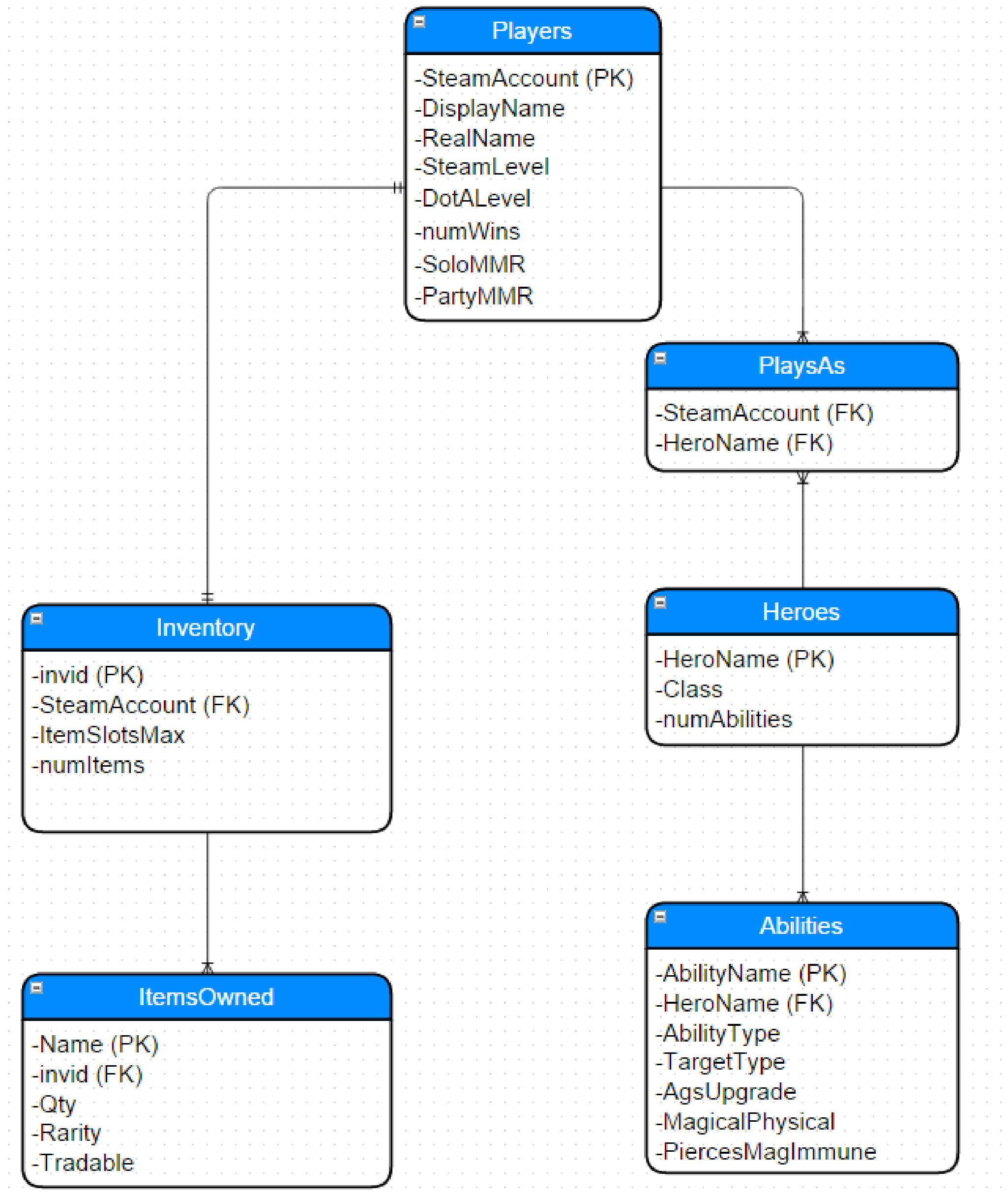
Table of Contents

Executive Summary

This paper will provide an outline of the “Remember the DotA” database. DotA is a massive multiplayer online battle arena (MOBA) with lots of data that people often need to reference. The database's purpose will be to serve as a tool containing DotA's large hero, inventory, and skill pool's relationship to each player. An E-R (Entity-Relationship) Diagram is illustrated to show functional dependencies within the database. Within this paper will be details on each element within the diagram. There will be SQL code and examples for the database to display functionality, tested using PostgreSQL 9.3 . I will also include the view of it and security required. Finally, I will explain what the known problems with the database are, as well as possible future enhancements.

E-R Diagram

This diagram shows relationships among each table within the database. Further information will be provided on the following pages.



Heroes Table

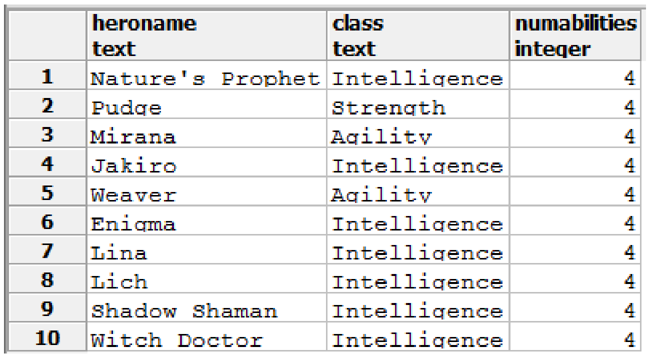
There are many heroes in DotA (as of November 30, 109 have been released) and any player can play these heroes each game. In this table is information about each hero, including its name, class, and the number of abilities he or she knows.

```
CREATE TABLE Heroes (
  HeroName                  text not null,
  Class                     text not null,
  numAbilities              int,
  primary key(HeroName)
);
```

Functional Dependencies:

HeroName → Class, numAbilities

Sample Data:



	heroname text	class text	numabilities integer
1	Nature's Prophet	Intelligence	4
2	Pudge	Strength	4
3	Mirana	Agility	4
4	Jakiro	Intelligence	4
5	Weaver	Agility	4
6	Enigma	Intelligence	4
7	Lina	Intelligence	4
8	Lich	Intelligence	4
9	Shadow Shaman	Intelligence	4
10	Witch Doctor	Intelligence	4

Abilities Table

Each Hero in DotA2 has atleast 4 abilities. Although most have 4, some have more.

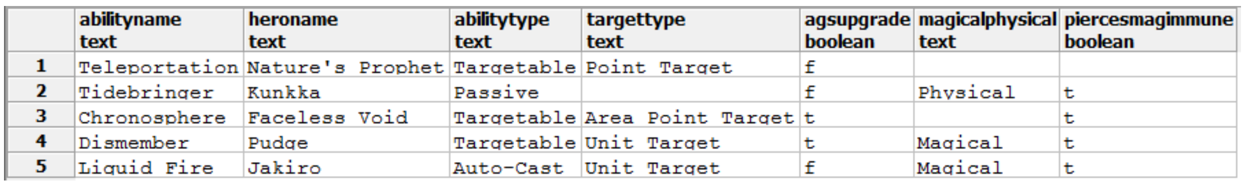
This table is used to store what they abilities each Hero has, and what their properties are.

```
CREATE TABLE Abilities (
  AbilityName               text not null,
  HeroName                  text not null references Heroes (HeroName),
  AbilityType               text not null,
  TargetType                text,
  AgsUpgrade                boolean not null,
  MagicalPhysical           text,
  PiercesMagImmune          boolean,
  primary key(AbilityName)
);
```

Functional Dependencies:

AbilityName → AbilityType, TargetType, AgsUpgrade, MagicalPhysical, PiercesMagImmune

Sample Data:



	abilityname text	heroname text	abilitytype text	targettype text	agsupgrade boolean	magicalphysical text	piercesmagimmune boolean
1	Teleportation	Nature's Prophet	Targetable	Point Target	f		
2	Tidebringer	Kunkka	Passive		f	Physical	t
3	Chronosphere	Faceless Void	Targetable	Area Point Target	t		t
4	Dismember	Pudge	Targetable	Unit Target	t	Magical	t
5	Liquid Fire	Jakiro	Auto-Cast	Unit Target	f	Magical	t

Players Table

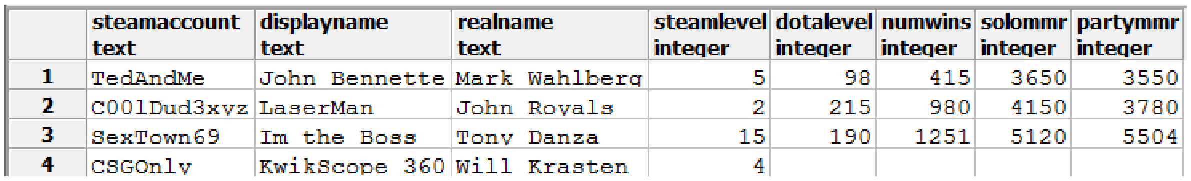
In the Players Table, I have included data for the player's Steam Account Name, Display Name, and Real Name, as well as other information related to their account such as level. This table is extremely important, since it is directly related to the user.

```
CREATE TABLE Players (
  SteamAccount          text not null,
  DisplayName           text not null,
  RealName              text,
  SteamLevel            int not null,
  DotALevel             int,
  numWins               int,
  SoloMMR               int,
  PartyMMR              int,
  primary key(SteamAccount)
);
```

Functional Dependencies:

SteamAccount → DisplayName, RealName, SteamLevel, DotALevel, numWins, SoloMMR, PartyMMR

Sample Data:



	steamaccount text	displayname text	realname text	steamlevel integer	dotalevel integer	numwins integer	solommr integer	partymmr integer
1	TedAndMe	John Bennette	Mark Wahlberg	5	98	415	3650	3550
2	C00lDud3xyz	LaserMan	John Royals	2	215	980	4150	3780
3	SexTown69	Im the Boss	Tony Danza	15	190	1251	5120	5504
4	CSGOnly	KwikScope 360	Will Krasten	4				

Inventory Table

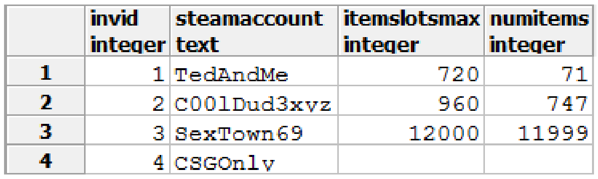
In the Inventory Table, we store information about each player's inventory, including the invid (which each player is given uniquely), the max number of item slots (as you may purchase more), and how many item slots are actually in use via items.

```
CREATE TABLE Inventory (
  invid                    serial,
  SteamAccount             text not null references Players (SteamAccount),
  ItemSlotsMax             int,
  numItems                 int,
  primary key(invid)
);
```

Functional Dependencies:

invid → ItemSlotsMax, numItems

Sample Data:



	invid integer	steamaccount text	itemslotsmax integer	numitems integer
1	1	TedAndMe	720	71
2	2	C00lDud3xyz	960	747
3	3	SexTown69	12000	11999
4	4	CSGOnly		

ItemsOwned Table

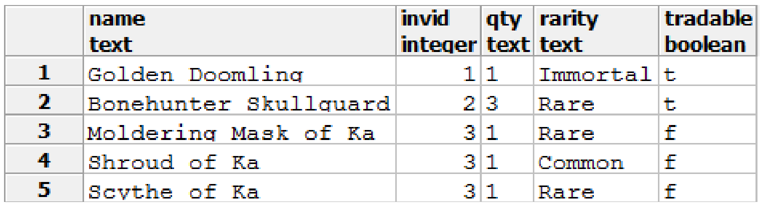
The Inventory table stores data about the user's inventory, and the ItemsOwned table stores information about the items within that inventory. It contains its name, different properties, quantity of the given item, and references the invid from inventory as a foreign key as to know whose inventory the item belongs to.

```
CREATE TABLE ItemsOwned (
  Name                    text not null,
  invid                   int references Inventory (invid),
  Qty                     text not null,
  Rarity                  text not null,
  Tradable                boolean not null,
  primary key(Name)
);
```

Functional Dependencies:

Name → Qty, Rarity, Tradable

Sample Data:



	name text	invid integer	qty text	rarity text	tradable boolean
1	Golden Doomling	1	1	Immortal	t
2	Bonehunter Skullguard	2	3	Rare	t
3	Moldering Mask of Ka	3	1	Rare	f
4	Shroud of Ka	3	1	Common	f
5	Scythe of Ka	3	1	Rare	f

PlaysAs Table

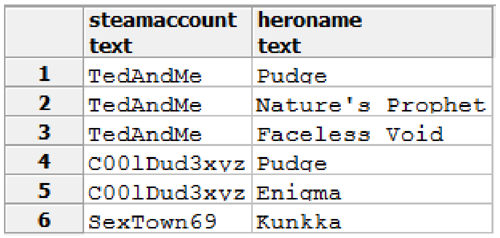
The PlaysAs table acts as a way to relate Players to Heroes. Each Player can play every Hero more than once, so many players play many heroes. I created the PlaysAs table to act as the separator, so that every player has the ability to play every hero, while avoiding the compromising Many-to-Many relationships.

```
CREATE TABLE PlaysAs (
  SteamAccount                    text not null references Players (SteamAccount),
  HeroName                    text not null references Heroes (HeroName)
);
```

Functional Dependencies:

There exist no functional dependencies for PlaysAs.

Sample Data:



	steamaccount text	heroname text
1	TedAndMe	Pudge
2	TedAndMe	Nature's Prophet
3	TedAndMe	Faceless Void
4	C00lDud3xyz	Pudge
5	C00lDud3xyz	Enigma
6	SexTown69	Kunkka

Views

Using this view, we are able to display each hero in the database, as well as each hero's abilities. It simplifies the messiness of wanting to display a certain hero's abilities without having to search through the Abilities table. It will display every property for the Abilities as well as its relevant Hero's properties without repeating the HeroName.

```
CREATE VIEW HeroesAndAbilities AS
SELECT h.HeroName, h.Class, h.numAbilities,
a.AbilityName, a.AbilityType, a.TargetType, a.AgsUpgrade, a.MagicalPhysical,
a.PiercesMagImmune
FROM Heroes h
JOIN Abilities a
ON h.HeroName = a.HeroName;
```

Sample Output:

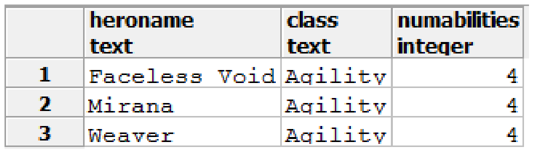
```
SELECT * FROM HeroesAndAbilities
WHERE HeroName = 'Nature''s Prophet'
ORDER BY Class ASC;
```

Reports

If you were to need all heroes in the database with the class Agility, one could use this code, which is then ordered nicely in ascending order by HeroName.

```
SELECT * FROM Heroes
WHERE Class = 'Agility'
Order BY HeroName ASC;
```

Output:



	heroname text	class text	numabilities integer
1	Faceless Void	Agility	4
2	Mirana	Agility	4
3	Weaver	Agility	4

One could also select Strength or Intelligence for the class to find the heroes that fall under those classes.

Stored Procedures

The following stored procedure will be used when you have decided you are done playing DotA and have traded all of your items to other people. It will use the ItemsOwned table.

```
CREATE or REPLACE FUNCTION tradedAllItems()
RETURNS void AS $$
begin
ALTER TABLE ItemsOwned
DROP TABLE ItemsOwned;
end;
$$ language plpgsql;

select tradeAllItems();
Fetch all from results;
```

Triggers

To reduce repetition, we will use the previous Stores Procedure for our trigger. When the Inventory is deleted, you will no longer have any items and can thus execute the stored procedure, tradedAllItems(), as you will have no more ItemsOwned.

```
CREATE TRIGGER quittingDotA
after DELETE on Inventory
for each row execute procedure tradedAllItems()
```

SECURITY

This database can be accessed by Administrators and Users.

Administrators

GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES TO administrator

Users

GRANT SELECT ON ALL Tables TO users

Gabe

GRANT ALL PRIVILEGES ON ALL TABLES TO gabe

Implementation Notes

- Many elements within this database were made on the presumption that the user plays DotA. However, the option of having *null* for most DotA specific entites is included, incase an account has been made, but DotA has never been opened on Steam.
- Valve continues to add and modify Heroes and Items, so it is necessary to understand how often change can happen within the game, and thus, the database

Known Problems

- A player can run out of space in their inventory, thus preventing them from adding more items. I am unaware how Valve handles this issue and was unable to incorporate a solution within here.
- Players can play a hero an infinite number of times, so it is impossible for he or she to play a hero when that hero has a certain set of four abilities, but then play the hero at a later date when these abilities have been modified. The easiest solution will probably to leave it as it is an just have it always display the current abilities.
- It is possible to delete your Steam account. If this happens, then everything must be dropped/deleted for that player, which is currently not a functional use.

Future Enhancements

- Items may have e-signatures by professional DotA players, so it would be nice to implement a features that discusses who signed the item and when.
- Items may also have a Gem within one of the item's sockets. This makes it hard because there are many gems that keep track of many statistics. In order to implement it into this system, we must remove Quantity for ItemsOwned (when the Item has a Gem) because it is rare two items with a Gem will be identical. The Gem could keep track of Kills, Gold Spent, etc. and the counter for these Kills, Gold Spent, etc. will always be changing.
- Items within the game that heroes may use could be implemented as well, but these items are constantly changing within the game and would be hard to update each time an item is purchased or sold, since games last for roughly 40 minutes. We could still have a database to keep track of what items have ever been purchased however.
- Finally, something that would be nice to implement is a progress statistic. This might have potential being in a new table to keep track of development when using certain items or heroes.