CS496 Assignment 3 part 1
Eric Rouse

The Data

The data I'll be modeling is the NCAAF bowl games and a group of players attempting to guess the team that will beat the spread of each game.

This is based on a game I played with some friends this year. We entered a contest where we all tried to pick the most spread winners (where spread is determined by Las Vegas 1 week before games started) for all the games (each worth 1 point). We also tried to pick the national champion (worth 3 points for picking the winner or -1 if the team you select isn't playing). There is also an option high-risk/high-reward section where you pick the spread winner of three games for a higher point payout (but if that team loses, you lose that many points). The person with the most points after all the games are played wins.

For each bowl we'll need to keep track of the favored team, the underdog, the predicted spread, the actual score of each team, the total points scored in the game and the resultant actual point spread. Since the games are not all played at once, we'll also keep track of whether the game was played or not and, we'll also keep track of the winning team name as well as the losing teams name.

For each player we'll need to keep track of their pick for the team to beat the predicted spread in each bowl game.

Since the championship is a bit different, and is handled differently than the other games it should get its own database. The championship is almost the same as a bowl, but spread doesn't matter, only the teams that are in it and the point results.

See the table below for all properties in a tabulated format:

TYPE	**Bowl**	**Player**	**Championship Game**
PROPERTIES	Bowl Game Name	First Name	Bowl Game Name
	Favorite Team Name	Last Name	Favorite Team Name
	Underdog Team Name	List of Spread Winners	Underdog Team Name
	Predicted Point Spread	List of High Stakes Bets	Favorite Team Actual Score
	Favorite Team Actual Score	Predicted Championship Team	Underdog Team Actual Score
	Underdog Team Actual Score	Championship Game Score	Total Score
	Total Score	Number of Points	Winning Team Name
	Winning Team Name		Losing Team Name
	Losing Team Name		Has Game Been Played?
	Actual Point Spread		
	Has Game Been Played?		

Data Modeling Plan

I plan to use redis as my non-relational database, for three reasons.

1. Redis is a key-value cache that supports all the datatypes I need (hashes, sets, strings and lists).
2. Redis is a CP database, it allows for consistency (all clients always have the same view of the data) and partition tolerance (system works well despite physical network positions). But, it has several atomic operations to make it more available (each client can always read and write). For instance the INCR operator would be super useful for increasing a players score (from redis documentation):
   There is something special about INCR. Why do we provide such an operation if we can do it ourselves with a bit of code? After all it is as simple as:
   x = GET count
   x = x + 1
   SET count x
   The problem is that doing the increment in this way will only work as long as there is a single client using the key. See what happens if two clients are accessing this key at the same time:

   Client A reads count as 10.
   Client B reads count as 10.
   Client A increments 10 and sets count to 11.
   Client B increments 10 and sets count to 11.
   We wanted the value to be 12, but instead it is 11! This is because incrementing the value in this way is not an atomic operation. Calling the INCR command in Redis will prevent this from happening, because it is an atomic operation. Redis provides many of these atomic operations on different types of data.
3. It is free and easy to install on Heroku.

Redis stores data as key value pairs, and has several different data types it can store. One of the coolest is the hash. The hash allows you to store a bunch of data like an object. They are maps between string fields and string values. This is perfect for my setup; I need a hash for each bowl game, a hash for each player and a hash for the championship game.

A sample bowl setup would look something like:

```
>HMSET bowl:33 name “Orange” favorite “Mississippi St” underdog “Ga Tech” spread “24” fav_score “33” und_score “55” winner “Ga Tech” loser “Mississippi St” act_spread “-9” game_played  “true”
```

And a retrieval would look like:

```
> HGETALL bowl:33
>1) "name" 2) "“Orange”" 3) "favorite" 4) "“Mississippi" 5) "St”" 6) "underdog" 7) "“Ga" 8) "Tech”" 9) "spread" 10) "“24”" 11) "fav_score" 12) "“33”" 13) "und_score" 14) "“55”" 15) "winner" 16) "“Ga" 17) "Tech”" 18)
```

```
"loser" 19) "“Mississippi" 20) "St”" 21) "act_spread" 22) "“-9”" 23)
"game_played" 24) "“true”"
```

Thus, all data in the set could be stored and retrieved with ease. As a bonus, the name:## nomenclature will make it easy to programmatically cycle through all the bowls. Or, alternatively I could create an ordered set of all the bowls, adding bowls to the set as they are created. The same could be done for the players.

Additional Database

An additional database I looked at was Hadoop. The primary reason hadoop isn't the right database for this project is that is design for large and complex datasets that don't fit well into tables. According to Cloudera CEO Mike Olson:

> *The Hadoop platform was designed to solve problems where you have a lot of data — perhaps a mixture of complex and structured data — and it doesn't fit nicely into tables. It's for situations where you want to run analytics that are deep and computationally extensive, like clustering and targeting. That's exactly what Google was doing when it was indexing the web and examining user behavior to improve performance algorithms.*
>
> *Hadoop applies to a bunch of markets. In finance, if you want to do accurate portfolio evaluation and risk analysis, you can build sophisticated models that are hard to jam into a database engine. But Hadoop can handle it. In online retail, if you want to deliver better search answers to your customers so they're more likely to buy the thing you show them, that sort of problem is well addressed by the platform Google built. Those are just a few examples.*
>
> -Mike Olson, Interviewed by O'Reilly Media.

That is more than enough reason right there. My datasets are not complicated and the calculations are simple. The calculations will be run once per day, by the administrator, not thousands of times per second by any user.

I also looked at CouchDB, and it is pretty nice. It is an AP database and is "eventually consistent". Since it is document-oriented (vs. key-value) it would be rather easy to set up "documents" that are objects, each bowl and player could be modeled as an object. Hence, CouchDB makes a great deal of sense for my data set.

I chose redis over couchDB because of support on Heroku my chosen platform, redis is a free and fully supported add-on. CouchDB is not, the documentation isn't as good. And, since I don't need to be changing user data very often, the CP model is probably more useful than the AP model, making redis a more obvious choice.