Project 5: Neo4j

Justin Hink

Dataset Explanation

For this project, I'm going to use the same source data that I did for project 2. It is a csv file containing all seasons for Major League Baseball hitters. For this exercise, I'm going to explore the relationship between players and the teams they played for. Graph databases aren't the most adept at examining the actual statistical measures for the hitters (OBP, wRC+, WAR etc) so those metrics will not be loaded into Neo4j.

Load Process

Please see the accompanying file: proj5.dataload.r.

This file contains code that will import the data from the source csv file (also included in the GitHub repository) into R. The code then leverages the RNeo4j (https://github.com/nicolewhite/Rneo4j) package to transform the data into a series of nodes and relationships and to push it to the database server. Running this one file should be all you need to do to replicate my results.

We will have 2 Node types:

- Player
- Team

And 1 relationship

- Playedfor

Where a player playedfor a given team (past tense as the data does not reflect any player movements in the offseason thus far).

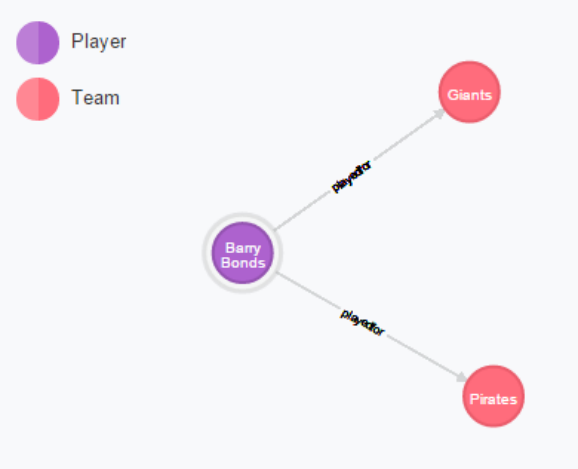
Note: You may want to cut down the amount of data you push to the database. Pushing the whole dataset, (over 2500 nodes and many more relationships) took a good deal of time on my relatively powerful workstation.

See the comments in the code file that should explain the process step by step.

Write One Cypher Query

Lets see who Barry Bonds played for:

```
match(p:Player {name:"Barry Bonds"} ) -[r:playedfor]->(d) return d
```



Looks correct. Fangraphs confirms.

Season	Team	G	PA	HR	R	RBI	SB	BB%	K%	ISO	BABIP	AVG	OBP	SLG	wOBA	wRC+	BsR	Off	Def	WAR
1986	Pirates	113	484	16	72	48	36	13.4 %	21.1 %	.194	.256	.223	.330	.416	.333	108	4.4	8.9	8.0	3.3
1987	Pirates	150	611	25	99	59	32	8.8 %	14.4 %	.230	.270	.261	.329	.492	.351	114	1.9	12.6	20.0	5.3
1988	Pirates	144	614	24	97	58	17	11.7 %	13.4 %	.208	.295	.283	.368	.491	.376	146	-1.3	29.5	2.2	5.4
1989	Pirates	159	679	19	96	58	32	13.7 %	13.7 %	.178	.265	.248	.351	.426	.341	121	2.3	17.5	30.1	7.1
1990	Pirates	151	621	33	104	114	52	15.0 %	13.4 %	.264	.301	.301	.406	.565	.420	165	5.0	52.3	21.5	9.9
1991	Pirates	153	634	25	95	116	43	16.9 %	11.5 %	.222	.292	.292	.410	.514	.396	155	3.4	43.0	11.4	7.8
1992	Pirates	140	612	34	109	103	39	20.8 %	11.3 %	.313	.300	.311	.456	.624	.459	198	4.6	70.0	-0.4	9.6
1993	Giants	159	674	46	129	123	29	18.7 %	11.7 %	.341	.321	.336	.458	.677	.467	193	1.0	78.4	1.9	10.5
1994	Giants	112	474	37	89	81	29	15.6 %	9.1 %	.335	.271	.312	.426	.647	.442	173	1.8	46.3	0.1	6.0
1995	Giants	144	635	33	109	104	31	18.9 %	13.1 %	.283	.294	.294	.431	.577	.425	163	1.7	52.2	5.5	7.7
1996	Giants	158	675	42	122	129	40	22.4 %	11.3 %	.308	.289	.308	.461	.615	.446	179	4.6	72.1	3.5	9.2
1997	Giants	159	690	40	123	101	37	21.0 %	12.6 %	.293	.280	.291	.446	.585	.430	165	4.0	61.2	6.9	8.9
1998	Giants	156	697	37	120	122	28	18.7 %	13.2 %	.306	.303	.303	.438	.609	.434	170	0.4	62.0	3.1	8.5
1999	Giants	102	434	34	91	83	15	16.8 %	14.3 %	.355	.225	.262	.389	.617	.416	148	2.1	30.3	-8.5	3.3
2000	Giants	143	607	49	129	106	11	19.3 %	12.7 %	.381	.271	.306	.440	.688	.456	174	0.9	62.3	-0.9	7.6
2001	Giants	153	664	73	129	137	13	26.7 %	14.0 %	.536	.266	.328	.515	.863	.537	235	1.3	118.0	-12.0	12.5
2002	Giants	143	612	46	117	110	9	32.4 %	7.7 %	.429	.330	.370	.582	.799	.544	244	-4.1	106.0	-2.0	12.4
2003	Giants	130	550	45	111	90	7	26.9 %	10.5 %	.408	.304	.341	.529	.749	.503	212	-0.7	78.7	5.6	10.1
2004	Giants	147	617	45	129	101	6	37.6 %	6.6 %	.450	.310	.362	.609	.812	.537	233	-2.7	103.3	-4.4	11.6
2005	Giants	14	52	5	8	10	0	17.3 %	11.5 %	.381	.219	.286	.404	.667	.426	162	0.0	4.0	0.6	0.6
2006	Giants	130	493	26	74	77	3	23.3 %	10.3 %	.275	.251	.270	.454	.545	.409	146	-2.4	27.4	-10.2	3.3
2007	Giants	126	477	28	75	66	5	27.7 %	11.3 %	.288	.254	.276	.480	.565	.428	157	-2.2	33.4	-14.5	3.4
Total	- - -	2986	12606	762	2227	1996	514	20.3 %	12.2 %	.309	.285	.298	.444	.607	.435	173	26.0	1169.4	67.6	164.0

Scenario Specific Advantages of a Graph DB

- Fast, efficient queries to examine relationships between teams and players.
    - Pointer based, no joins
- Very easy to traverse the graph to examine which players have played for common teams
- Easily scalable as teams and number of players grows
    - Expanding this to the minor leagues would be very easy for example and our query performance should not degrade in the foreseeable future

Disadvantages

- Expanding the application to include statitistics is possible but, as mentioned, the queries as the number of properties per node that you need to aggregate over degrade quickly with the number of nodes in the DB. For example, I would not want to calculate year specific linear weights based off of this data structure.