Estadísticas de la NBA sobre bases de datos de grafos

Santiago del Pino, Bruno Bradach
Instituto de Computación
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay

***Resumen*—El siguiente documento presenta el resultado de modelar las estadísticas históricas de jugadores de la NBA en una base de datos de grafos, con el fin de responder preguntas y confirmar resultados conocidos así como comparar y analizar la ejecución de diferentes algoritmos de centralidad.**

I. Introducción

La motivación de este trabajo fue evaluar si era factible crear una base de datos de grafos que contenga las estadísticas históricas de jugadores de la NBA y realizar consultas sobre ella para responder algunas preguntas que nos planteamos. Los datos para esta base fueron obtenidos de Kaggle, en donde logramos encontrar las estadísticas desde el 1950 al 2015 de todos los jugadores y equipos, junto a nombres de jugadores, equipos en los que jugaron, universidad a la que asistieron y múltiples estadísticas de juego.

II. Limpieza de datos

Previa a la construcción y modelado del grafo fue necesario realizar una limpieza y reorganización de los datos, ya que contenían columnas innecesarias para nuestro fin, además de algunos valores incorrectos o genéricos. En el set de datos de jugadores decidimos trabajar solamente con el nombre del jugador y la universidad a la que asistió debido a que era de particular interés para una de las consultas que planteamos. En cuanto al set de temporadas, seleccionamos el año, jugadores, equipo en el que jugó y el winshare. El winshare es una medida que distribuye la cantidad de partidos ganados por un equipo en una temporada entre sus jugadores, basándose en las estadísticas de los jugadores, o en otras palabras, qué tanto aportaron para ganar. Esta estadística nos permite identificar los mejores jugadores y con esto los mejores equipos.

Se decidió descartar todos los datos previos al año 1975 ya que las estadísticas disponibles en la NBA aumentaron significativamente ese año, y como consecuencia la fórmula del winshare se vió notoriamente modificada.

El código fuente utilizado para la limpieza y carga de los datos se puede encontrar en el siguiente repositorio:
https://gitlab.fing.edu.uy/bruno.bradach/proyecto-final-bdnr

III. Modelado de la base de datos de Grafos

Para el modelado de la base, decidimos tener nodos para cada jugador, la universidad a la que asistió y un nodo para cada equipo identificados mediante el par equipo-año. Las relaciones necesarias para la resolución de nuestras preguntas fueron ATTENDED, si el jugador asistió a una universidad, y PLAYED_IN, relacionando a cada jugador con el equipo en el que jugó. Esta última relación tiene además la propiedad WinShare explicada anteriormente. La Figura 1 representa el estado del grafo al final de la carga de los datos utilizando MERGE de forma de poder modificar nodos existentes.

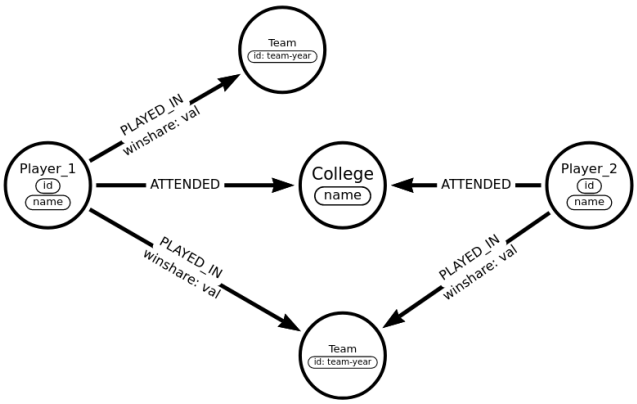

Figura 1: Modelo de la base

IV. Relación entre jugadores y universidades

En esta sección se presenta el análisis realizado sobre la relación entre jugadores y universidades. El objetivo de este análisis es responder a las siguientes preguntas:

- ¿Qué universidades brindaron más jugadores?
- ¿Qué universidades brindaron mejores jugadores?

IV-A. Universidades que brindaron más jugadores

Se implementó una consulta en Cypher que responde a esta pregunta. La consulta cuenta la cantidad de relaciones ATTENDED para cada College, y retorna la lista de colleges ordenada por dicha cantidad. En el cuadro I se presenta una tabla con el top 10 de universidades que más jugadores brindaron a la NBA.

Listado 1 Obtener las universidades que brindaron más jugadores

```
MATCH (p: Player) - [a :ATTENDED] -> (c: College)
WITH c, count(a) as playersCount
RETURN c.name, playersCount
ORDER BY playersCount DESC
```

Cuadro I: Universidades que brindaron más jugadores

Universidad	Jugadores
University of California, Los Angeles	82
University of Kentucky	73
University of North Carolina	70
Duke University	62
University of Kansas	60
University of Arizona	50
Indiana University	45
University of Louisville	45
Syracuse University	43
University of Michigan	41

IV-B. Universidades que brindaron mejores jugadores

Para responder a esta pregunta se probaron diferentes criterios para definir el concepto de universidades que brindaron mejores jugadores.

Universidades con mayor promedio de winshare

En este caso, primero se calcula el winshare promedio de un jugador en su carrera, y luego se calcula el winshare promedio a nivel de universidad (promedio entre todos los jugadores que asistieron). En el cuadro II se presenta el top 10 de universidades con este criterio.

Listado 2 Universidades con mayor promedio de winshare

```
MATCH (t: Team) <- [pi: PLAYED_IN] - (p: Player)
     - [a :ATTENDED] -> (c: College)
CALL {
   WITH t
   MATCH (t) <- [pli: PLAYED_IN] - (pp: Player)
   RETURN pp.name, avg(pli.winshare) as playerAvgWinshare
}
WITH c, avg(playerAvgWinshare) as collegeAvgWinshare
RETURN c.name, collegeAvgWinshare
ORDER BY collegeAvgWinshare DESC
```

Cuadro II: Universidades con mayor promedio de winshare

Universidad	Promedio
University of Illinois at Chicago	4.266
State University of New York at Potsdam	3.912
Grand Canyon University	3.832
Loyola College in Maryland	3.675
University of Minnesota Duluth	3.455
University of the District of Columbia	3.408
College of the Holy Cross	3.283
University of San Diego	3.091
Longwood University	3.031
Masters College	3.026

El ranking de universidades obtenido es muy diferente a las universidades que en la realidad se consideran las que mejores jugadores aportaron.

Universidades con mayor acumulación de winshare

En este caso, se acumulan todos los win shares de todos los jugadores que asistieron a una universidad. En el cuadro III se presenta el top 10 de universidades con este criterio.

Listado 3 Universidades con mayor acumulación de winshare

```
MATCH (t: Team) <- [pi: PLAYED_IN] - (p: Player)
     - [a :ATTENDED] -> (c: College)
CALL {
   WITH t
   MATCH (t) <- [pli: PLAYED_IN] - (pp: Player)
   RETURN pp.name, sum(pli.winshare) as playerSumWinshare
}
WITH c, sum(playerSumWinshare) as collegeSumWinshare
RETURN c.name, collegeSumWinshare
ORDER BY collegeSumWinshare DESC
```

Cuadro III: Universidades con mayor acumulación de winshare

Universidad	Suma
University of North Carolina	17301
University of California, Los Angeles	15489
University of Kentucky	14423
Duke University	14089
University of Kansas	12299
University of Arizona	11667
University of Michigan	9620
University of Connecticut	9360
Georgia Institute of Technology	9303
University of Maryland	8547

En este caso, el ranking de mejores universidades se parece más a lo que en realidad se consideran las universidades que mejores jugadores aportaron. Por otro lado, también coincide en su mayoría con el Top 10 de universidades que más jugadores aportaron, por lo que este criterio se considera que está sesgado por el hecho de que hay más jugadores que aportan a la acumulación, aunque no necesariamente se hayan destacado.

Universidades que más figuran en el ranking de mejores jugadores

En este caso, primero se obtiene un ranking de los mejores jugadores de la historia. Luego, se obtiene un ranking de las universidades que más figuran en el top 1 % de dicho ranking de jugadores. En el cuadro IV se presenta el top 10 de universidades con este criterio.

Listado 4 Universidades con mayor presencia en top players

```
WITH 1 as percentage
MATCH (:Player) - [:PLAYED_IN] -> (:Team)
WITH toInteger(floor(count(*) * percentage / 100)) as lim
CALL apoc.cypher.run('
   MATCH (p: Player) - [pl: PLAYED_IN] -> (t: Team)
   WITH p, avg(pl.winshare) as playerAvgWinshare
   OPTIONAL MATCH (p) - [a: ATTENDED] -> (c: College)
   RETURN p.name as player,
        playerAvgWinshare,
        c.name as college
   ORDER BY playerAvgWinshare
   DESC LIMIT \$limit', {limit : lim}) yield value
WITH value.player as Player, value.college as College,
    value.playerAvgWinshare as PlayerAvgWinshare
RETURN College,
     count(College) as PlayersCount,
     collect(Player) as Players
ORDER BY PlayersCount DESC
```

Cuadro IV: Universidades con mayor presencia en top players

Universidad	Cantidad
University of North Carolina	7
University of California, Los Angeles	7
Duke University	6
Georgetown University	5
University of Texas at Austin	4
University of Kentucky	4
Louisiana Tech University	3
Michigan State University	3
Arizona State University	3
University of Houston	3

Si bien hay dos universidades con el máximo de 7 jugadores dentro del top 1 %, se puede ver que no hay una universidad que se despegue del resto. También hay que tener en cuenta que el dataset solo considera universidades, pero hubo un período de tiempo en el que los jugadores podían pasar directamente de high school a la NBA. Esos jugadores no son tenidos en cuenta para este ranking.

V. Comunidades de jugadores destacados

En esta sección se presenta el trabajo realizado en cuanto a la detección de comunidades de jugadores. En este contexto, se busca responder a la siguiente pregunta: ¿cuáles son los grupos de jugadores que más se destacaron en la historia de la NBA?

Para lograr el objetivo se implementó una consulta Cypher que obtiene los tríos (grupos de 3) de jugadores que más se destacaron jugando juntos. Se utiliza el winshare como métrica de desempeño, y la consulta retorna una lista de los tríos que tuvieron mayor winshare combinado (sumado) en una temporada, junto con el equipo y año en que lo lograron. En el cuadro V se presenta el top 10 de tríos más destacados de la historia.

Listado 5 Universidades con mayor presencia en top players

```
MATCH (t: Team)
CALL apoc.cypher.run('
   MATCH (p: Player) - [pi: PLAYED_IN] -> (t :Team)
   WHERE id(t) = id(\$team)
   RETURN p as player, pi as playedIn
   ORDER BY pi.winshare DESC
   LIMIT 3
', {team: t}) YIELD value
WITH t, collect(value.player.name) as players,
     sum(value.playedIn.winshare) as totalWinshare
RETURN t.id as team, players
ORDER BY totalWinshare DESC
```

Cuadro V: Tríos más destacados de la historia

Equipo	Jugadores
CHI-1992	Michael Jordan, Horace Grant, Scottie Pippen
CHI-1996	Michael Jordan, Scottie Pippen, Toni Kukoc
CHI-1991	Michael Jordan, Scottie Pippen, Horace Grant
UTA-1997	Karl Malone, John Stockton, Jeff Hornacek
OKC-2013	Kevin Durant, Russell Westbrook, Serge Ibaka
BOS-1987	Larry Bird, Kevin McHale, Robert Parish
CHI-1997	Michael Jordan, Scottie Pippen, Steve Kerr
MIA-2011	LeBron James, Dwyane Wade, Chris Bosh
UTA-1996	Karl Malone, John Stockton, Jeff Hornacek
CLE-2009	LeBron James, Mo Williams, Anderson Varejao

Los resultados obtenidos son satisfactorios al compararlos con la realidad. Se identifican tríos pertenecientes a equipos como los Chicago Bulls, popularmente considerados el mejor equipo de la historia; Utah Jazz de los años 90, finalista en ambas ocasiones; Boston Celtics en los 80, popularmente conocidos como el Big Three; y más recientemente Miami Heat, conocido popularmente como el nuevo Big Three de la época.

VI. Detectar loops en transferencias de jugadores

Las transferencias de un jugador se definen como el derecho que posee un jugador de buscar libremente oportunidades para ampliar su potencial deportivo y de ganancia. Son conocidos los casos de contratos multimillonarios en las transferencias en el football y NBA a diferentes equipos pero son pocos los casos que se conocen oficialmente donde un jugador parte de un club en cierto año y luego vuelve al mismo años después. En football bajo la regularización FIFA y en temporada de traspases en el 2021, se dieron 5 casos en los cuales un jugador volvió a un equipo anterior. Partiendo de esta base, la idea que planteamos es identificar si creando una base en forma de grafo y utilizando los datos de la NBA, podemos encontrar la cantidad de transferencias loops y quienes fueron los jugadores, además de investigar si podemos realizar la misma consulta recorriendo relaciones PLAYED_IN u otras. Para este análisis se plantearon dos formas de realizar la consulta, por un lado utilizar el grafo anteriormente descrito, utilizando solamente los datos disponibles y por otro lado evaluar la posibilidad de crear nuevas relaciones y nodos de modo de poder acceder al resultado de forma más directa.

VI-A. Problemas identificados

En la investigación realizada para esta consulta, identificamos algunos problemas que surgen tanto del modelo creado, como de la información disponible en el set de datos. Uno de los inconvenientes al trabajar con transferencias de jugadores es que el set de datos contiene para cada jugador en qué equipo jugó, incluso si jugo en el mismo equipo en años consecutivos, eso implica que al buscar los loops en las transferencias fue necesario prestar especial atención en filtrar los duplicados tanto de equipos como de jugadores. Por otro lado, Neo4j no respeta el orden de creación. A modo de ejemplo, si un jugador estuvo en dos o más equipos en el mismo año, es posible que la consulta devuelva los equipos en el ordenes incorrectos, generando loops en transferencias donde no los hay.

VI-B. Grafo sin cambios

Utilizando el grafo planteado originalmente, es posible encontrar los loops en las transferencias utilizando la consulta planteada en el Listado 6, donde hacemos uso de procedimientos APOC (Awesome Procedures on Cypher) *dropDuplicateNeighbors* y *containsDuplicates*. El uso de *dropDuplicateNeighbors* se justifica debido a uno de los problemas identificados anteriormente, esta función elimina duplicados consecutivos y por lo tanto elimina las ”transferencias”que son dentro del mismo equipo. Luego se aplica *containsDuplicates* que determina que si la lista aún contiene duplicados es debido a que efectivamente hay un loop en las transferencias de ese jugador. Esta consulta retorna 367 jugadores que desde 1975 al 2015 se fueron de un equipo y años después volvieron al mismo. En el Cuadro VI se pueden apreciar los primeros 10 resultados y los equipos en los que jugaron.

Listado 6 Loops en transferencias en grafo original

```
MATCH (p: Player) - [pl: PLAYED_IN] -> (t: Team)
WITH p, t
ORDER BY t.year, pl.teamInSameYear
WITH p, collect(t.team) as playerCareer
WITH p, apoc.coll.dropDuplicateNeighbors(playerCareer)
as playerCareer
WHERE apoc.coll.containsDuplicates(playerCareer)
RETURN p.name, playerCareer
ORDER BY p.name
```

Cuadro VI: Jugadores con Loops en transferencias

Jugador	**Equipos**
A.C. Green	[LAL, PHO, DAL, LAL, MIA]
A.J. Price	[IND, WAS, MIN, IND, CLE, PHO]
Aaron Brooks	[HOU, PHO, SAC, HOU, DEN, CHI, IND]
Acie Law	[ATL, GSW, CHA, CHI, MEM, GSW]
Adrian Griffin	[BOS, DAL, HOU, CHI, DAL, CHI, SEA]
Al Harrington	[IND, ATL, IND, GSW, NYK, DEN, ORL, WAS]
Al Skinner	[NYN, NJN, DET, NJN, PHI]
Allen Iverson	[PHI, DEN, DET, MEM, PHI]
Alonzo Gee	[WAS, SAS, WAS, CLE, DEN, POR, NOP, DEN]

VI-C. Grafo modificado

Con el fin de poder evaluar si existía alguna forma de obtener el mismo resultado recorriendo solamente relaciones, planteamos la idea de generar nodos para cada transferencia del jugador, además de relaciones OF; para indicar a que jugador pertenece la transferencia, FROM; para indicar desde qué equipo es la transferencia y TO; para indicar el equipo destino.

Luego de haber creado cada una de las transferencias para los jugadores, el nuevo esquema de la base se puede observar en la figura 2.

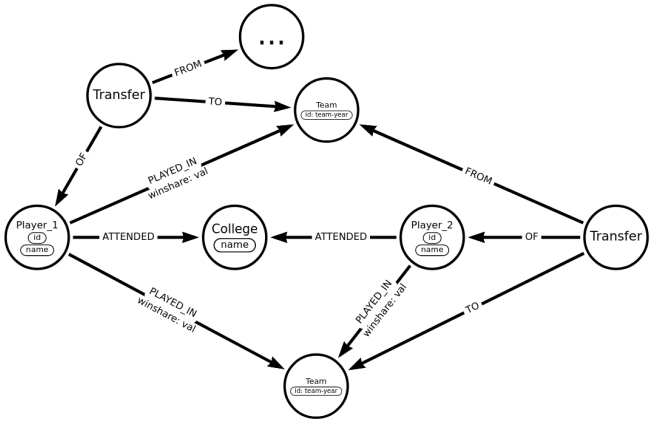

Figura 2: Modelo de la base utilizando transferencias

Utilizando los nuevos nodos y relaciones podemos plantear la misma consulta para detectar los loops utilizando el código en el Listado 7. Primero se obtienen los caminos Transfer $\Rightarrow$ Player $\Leftarrow$ Trasfer donde la primer transferencia pertenece al ingreso del jugador al equipo y la segunda es el retorno del jugador. Para que la consulta sea válida, es necesario agregar además la condición de que el equipo de la primer transferencia sea el mismo que en la segunda y que la diferencia entre los años sea mayor a 0 de forma de eliminar loops sobre el mismo equipo y el mismo año.

Esta consulta retorna los mismos 367 jugadores que la consulta del Listado 6 y los primeros 10 resultados se puede observar en el cuadro VII.

Listado 7 Loops en transferencias usando transfers nodes

```
MATCH (t_one:Team) <- [:FROM] -
(first_transfer:Transfer) - [:OF] -> (p:Player) <-
[:OF] - (second_transfer:Transfer) - [:TO] -> (t_two:Team)
WITH p, t_one, t_two, first_transfer, second_transfer
ORDER BY first_transfer.id, second_transfer.id
WHERE t_one.team = t_two.team AND t_two.year - t_one.year > 0
WITH collect(DISTINCT t_one.id) as transfered,
collect(DISTINCT t_two.id) as returned, p as p
RETURN DISTINCT p.name, transfered, returned
ORDER BY p.name
```

Cuadro VII: Jugadores con Loops en trasferencias usando transfers nodes

Jugador	Transferido	Retorno
A.C. Gree	LAL-1993	LAL-2000
A.J. Price	IND-2012	IND-2015
Aaron Brooks	HOU-2011	HOU-2013
Acie Law	GSW-2010	GSW-2011
Adrian Griffin	DAL-2003, CHI-2005	DAL-2006, CHI-2007
Al Harrington	IND-2004	IND-2007
Al Skinner	NJN-1978	NJN-1979
Allen Iverson	PHI-2007	PHI-2010
Alonzo Gee	WAS-2010, DEN-2015	WAS-2011, DEN-2017
Alonzo Mourning	MIA-2002	MIA-2005

VII. Análisis de centralidad

La centralidad en un grafo puede ser entendida como una medida o un valor que posee un vértice dentro de un grafo. Este al ser evaluado en una escala, determina su relevancia dentro del grafo y permite comparar o contrastar dicho vértice con otros [1]. Lo que planteamos en este documento es aplicar algunos algoritmos de centralidad sobre el grafo de la NBA y evaluar si obtenemos resultados relevantes. En las siguientes secciones se describirán los resultados obtenidos.

VII-A. Algoritmo Page Rank

El algoritmo Page Rank mide la importancia de cada nodo en el grafo basado en la cantidad de relaciones entrantes y la importancia de los nodos de origen [2]. Este algoritmo tiene un uso específico y es para encontrar aquellos nodos que están conectados con nodos de diferentes tipos, y por lo tanto son capaces de difundir información de forma fluida. En este caso tenemos 3 nodos diferentes por lo tanto no esperamos obtener ningún resultado relevante. El código que permite ejecutar este algoritmo se encuentra en el Listado 8 y en el cuadro VIII y cuadro IX se pueden ver los resultados obtenidos.

Listado 8 Aplicacion PageRank para requipos

```
CALL gds.pageRank.stream('full_graph', {
  maxIterations: 20,
  dampingFactor: 0.85,
  relationshipWeightProperty: 'winshare'
})
YIELD nodeId, score
WHERE labels(gds.util.asNode(nodeId)) = ["Team"]
RETURN gds.util.asNode(nodeId).id AS team_year, score
ORDER BY score DESC
```

Para los equipos no logramos encontrar ninguna relación importante entre la estadísticas y el puntaje obtenido según el algoritmo. El equipo de Philadelphia 76ers, tuvo en 2017 21 jugadores en esa temporada y un average winshare de 1.304 mientras que Toronto 2017, que obtuvo un puntaje de 1.10, tuvo 17 jugadores en 2017 y un average de 3.18, sin embargo Philadelphia-2017 tiene mejor puntaje que Toronto-2017.

Para los jugadores tampoco logramos encontrar una relación entre el puntaje obtenido y las relaciones y/o el winshare de cada uno. A modo de ejemplo, Karl Malone jugó durante 19 años para dos equipos, con un average de 12.34, mientras que LeBron James jugo durante 14 años para dos equipos con un average winshare de 14.68. A pesar de no encontrar relación, los mejores jugadores rankeados según PageRank son todos actuales o futuros miembros del salón de la fama de la NBA.

Si bien el algoritmo tiene disponible algunos parámetros, cambiarlos provoca que el puntaje cambie pero el orden de equipos o jugadores es siempre el mismo.

Cuadro VIII: Resultados PageRank para equipos

Equipo	Año	Puntaje
PHI	2017	1.317131
CHI	1975	1.184102
NOJ	1975	1.146065
SEA	1976	1.117746
TOR	2017	1.108093
BRK	2017	0.992870
DEN	2017	0.969025
OKC	2017	0.968081
POR	1975	0.956462
UTA	2015	0.949836

Cuadro IX: Resultados PageRank para jugadores

Jugador	Puntaje
Karl Malone	0.721707
Kevin Garnett	0.693609
Kobe Bryant	0.664237
Reggie Miller	0.663618
John Stockton	0.659981
Dirk Nowitzki	0.653047
Charles Barkley	0.652071
Michael Jordan	0.648823
LeBron James	0.639577
Moses Malone	0.632866

VII-B. Algoritmo Degree Centrality

El algoritmo de Degree Centrality puede ser utilizado para encontrar los nodos más populares en un grafo. Mide la cantidad de relaciones entrantes y/o salientes de un nodo tomando en cuenta además el peso de la arista. El codigo que permite ejecutar este algoritmo para jugadores se encuentra en el Listado 9 y en el cuadro X y cuadro XI se pueden ver los resultados obtenidos.

Listado 9 Aplicacion Degree Centrality para jugadores

```
CALL gds.degree.stream(
   'full_graph',
   { relationshipWeightProperty: 'winshare' }
)
YIELD nodeId, score
WHERE labels(gds.util.asNode(nodeId)) = ["Player"]
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC LIMIT 10
```

En el caso de los equipos, se nota la influencia del winshare en los resultados ya que 7 de los 10 equipos listados en el

[1] https://www.grapheverywhere.com/centralidad/

[2] https://neo4j.com/docs/graph-data-science/current/algorithms/page-rank/

cuadro X fueron campeones o vicecampeones. Por otro lado, el puntaje en los jugadores es en realidad la suma de todos los winshare que el jugador tuvo en cada equipo que estuvo, es decir que el algoritmo devuelve el mejor jugador tomando en cuenta el winshare. A modo de ejemplo, se puede obtener el mismo resultado para Karl Malone ejecutando la consulta en el Listado 10.

Listado 10 Consulta para obtener suma de WinShare de Karl Malone

```
MATCH (t: Team) <- [pi: PLAYED_IN] -
(p: Player {name:"Karl Malone"})
RETURN p.name, sum(pi.winshare) as playerSumWinshare
```

Cuadro X: Resultados Degree Centrality para equipos

Equipo	Año	Puntaje
CHI	1996	75.5
CHI	1997	71.3
BOS	2008	70.2
SAS	2016	69.1
CHI	1992	68.8
CLE	2009	68.2
OKC	2013	67.7
GSW	2016	67.0
GSW	2017	66.8
GSW	2015	66.6

Cuadro XI: Resultados Degree Centrality para jugadores

Jugador	Puntaje
Karl Malone	234.6
Michael Jordan	213.9
John Stockton	207.6
Tim Duncan	206.3
LeBron James	205.6
Dirk Nowitzki	201.3
Kevin Garnett	191.5
David Robinson	178.8
Charles Barkley	177.1
Reggie Miller	174.5

VII-C. Algoritmo Betweenness Centrality

El algoritmo Betweeness Centrality es una forma de detectar la influencia de un nodo sobre el flujo de información de un grafo. Es generalmente utilizado para encontrar nodos que sirven como puente entre una sección del grafo y otra. El algoritmo calcula todos los caminos más cortos y el puntaje se basa en la cantidad de caminos cortos que pasan por ese nodo. Con este algoritmo esperábamos encontrar los nodos que más interconectados estén, es decir quienes tengan mayor cantidad de equipos, transferencias, etc. Una desventaja de este algoritmo es que requiere un flujo de información muy grande por lo tanto solo logramos aplicarlo a los jugadores. El código para ejecutar este algoritmo se encuentra en Listado 11 y el resultado se puede observar en el cuadro XII

Listado 11 Aplicacion Betweenness Centrality para jugadores

```
CALL gds.betweenness.stream('full_graph')
YIELD nodeId, score
WHERE labels(gds.util.asNode(nodeId)) = ["Player"]
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC LIMIT 10
```

Si comparamos los grafos de Joe Smith, quien obtuvo el primer lugar, y LeBron James, quien obtuvo un puntaje de 24 y esta fuera del top 10, se puede observar que la cantidad de relaciones y nodos en el grafo de Joe Smith es significativamente mayor que en el de LeBron James, confirmando nuestra suposición de encontrar los nodos más interconectados. El detalle de los grafos se observa en la figura 3 y figura 4 para Joe Smith y LeBron James respectivamente.

Cuadro XII: Resultados Betweenness Centrality para jugadores

Jugador	Puntaje
Joe Smith	280.0
John Lucas	252.0
Kevin Ollie	238.0
Tony Massenburg	224.0
Juwan Howard	210.0
Chucky Brown	208.0
Kevin Willis	207.0
Jim Jackson	198.0
Tyrone Corbin	190.0
Theo Ratliff	190.0

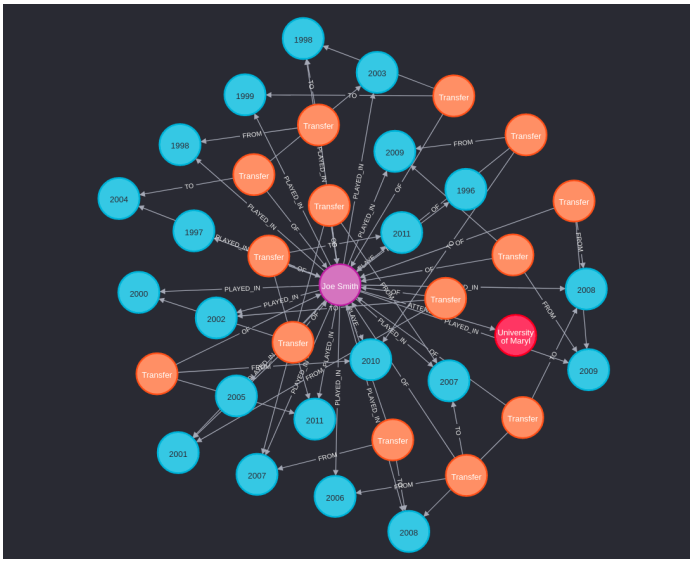

Figura 3: Grafo de Joe Smith

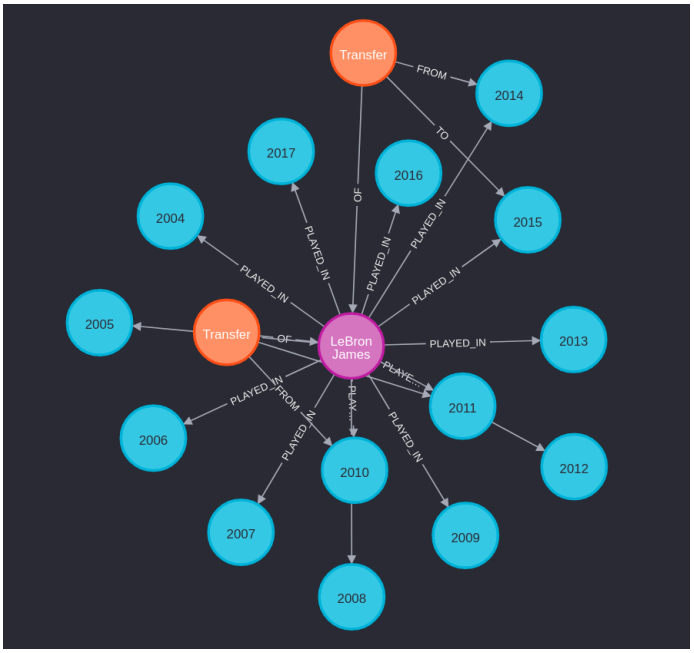

Figura 4: Grafo de LeBron James

VIII. Conclusiones y trabajo futuro

El foco de este trabajo estuvo en modelar una realidad relacionada a la NBA y sus jugadores utilizando grafos, almacenar la información en una base de datos de grafos como Neo4j y extraer información relevante de dicha base. Como objetivo general, se lo considera un objetivo cumplido satisfactoriamente.

A su vez el trabajo tenía objetivos específicos como responder preguntas puntuales sobre la actividad de los jugadores en la NBA. En ese sentido, se pudieron determinar las universidades que más jugadores brindaron a la NBA, y cuáles brindaron mejores jugadores, en base a un criterio definido. Además, se pudieron encontrar los grupos de jugadores que mejor se desempeñaron jugando juntos, arrojando resultados razonables en comparación con los que popularmente se identifican como los mejores compañeros de equipo.

Por otro lado, se logró responder al problema de obtener los jugadores que cambiaron de equipo y en el futuro retornaron. Se exploraron dos enfoques diferentes para abordar el problema, uno con el modelo original, y otro adicionando nodos y relaciones que simplifiquen las consultas. Se considera como trabajo futuro relevante realizar pruebas de performance para comparar ambos enfoques.

Por último, se exploraron los algoritmos de centralidad provistos por Neo4j y se pudo ver que algunos algoritmos no arrojan información relevante en nuestro modelo, pero otros sí lo hacen. En cualquier caso, los resultados obtenidos se consideran satisfactorios para los objetivos de exploración que se habían planteado originalmente.