Número Roger Federer y estadísticas sobre Tenis

Bruno Szilagyi *Facultad de Ingeniería, Universidad de la República*
Montevideo, Uruguay
BDNR 2023 - Grupo 16 bruno.szilagyi.ibarra@gmail.com
Ignacio Dorgans *Facultad de Ingeniería, Universidad de la República*
Montevideo, Uruguay
BDNR 2023 - Grupo 16 ignaciodorgans@gmail.com

***Resumen*—En este informe estaremos mostrando el resultado de modelar los datos relacionados a los torneos de Tenis desde 1997 a 2020 en una base de datos de grafos con el fin de conocer la distancia Roger Federer[1] de los jugadores y recolectar estadísticas del juego.**

I. Introducción

La motivación del trabajo es experimentar con un problema real sobre una base de datos de grafos y reconocer sus virtudes para poder realizar cálculos sobre caminos y centralidades. En este trabajo se realiza un modelo sobre datos recolectados de torneos de Tenis, sus encuentros entre jugadores y metadatos asociados a cada jugador.
El objetivo será calcular estadísticas sobre estos torneos y también el número Roger Federer.

Figura 1: Perfil de Roger Federer en ATP.

II. Solución planteada

En esta sección describimos las fuentes utilizadas para realizar el trabajo así como también la arquitectura necesaria para poder implementar los procesos de carga y consulta.

II-A. Fuentes de información

Para obtener información fiable de los resultados de los partidos, torneos y jugadores utilizamos el repositorio de **JeffSackmann**[2] sobre datos de Tenis ATP[3].

Hacemos uso de los siguientes archivos:

- `atp_ranking_current.csv` contienen todos los rankings de jugadores actual.
  Columnas: `ranking_date, ranking, player_id, ranking_points`
- `atp_matches_*.csv` contiene todos los juegos de los torneos.
  Para los matches se provee un diccionario[4] de datos ya que tiene un gran número de columnas.
- `atp_players.csv` contiene toda la metadata de los juegadores.
  Columnas: `player_id, first_name, last_name, hand, birth_date, country_code, height`

Respecto a la **limpieza de los datos**, realizamos las siguientes acciones:

- Realizamos una partición horizontal de los datos, considerando solo desde 1997 hasta 2020.
- Hicimos una partición vertical de los datos de jugadores y los partidos, ya que muchas columnas no eran de interés para nuestro análisis.

[1] https://www.atptour.com/es/players/roger-federer/f324/overview
[2] https://github.com/JeffSackmann/tennis_atp
[3] El Circuito de la ATP es una serie de torneos oficiales de tenis masculino que organiza la Asociación de Tenistas Profesionales. WMA es el femenino.
[4] https://github.com/JeffSackmann/tennis_atp/blob/master/matches_data_dictionary.txt

II-B. *Componentes de la solución*

A continuación describiremos los principales componentes de la arquitectura solución para poder realizar la carga y consulta de datos.

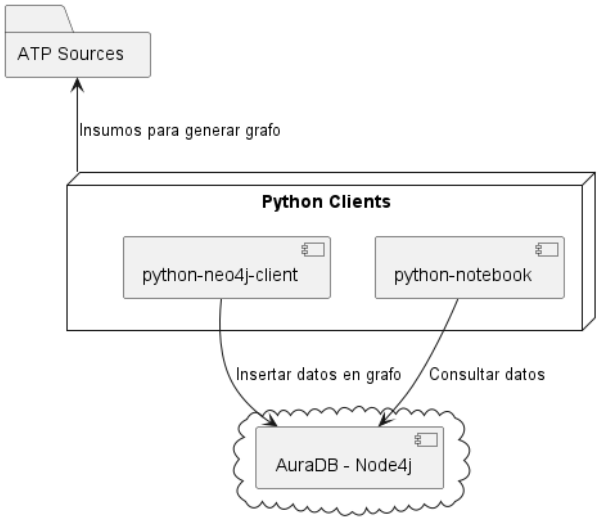

Figura 2: Diagrama de componentes.

Se distinguen los componentes de **ATP Sources**, **Python Clients** en donde tenemos una aplicación python-neo4j-client y una python notebook. Por último la instancia de AuraDB en donde se persisten los registros del grafo. El código de la aplicación y notebook se encuentra en el repositorio[5] Gitlab de la Facultad.

II-C. *Componente python-neo4j-client*

Esta aplicación tiene como objetivo la carga de los datos, procesarlos realizando una limpieza y transformación de ellos y luego cargarlos a una instancia de AuraDB.

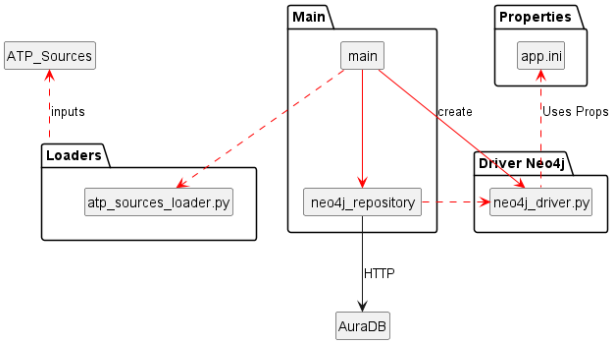

Figura 3: Diseño de la aplicación.

Encontramos proceso de carga en auraDB como un desafío; primero por performance en la carga, ya que estamos manejando decenas de miles de nodos y luego porque la cantidad de registros a escribir no podía superar el umbral algunos megas.

[5]https://gitlab.fing.edu.uy/bdnr-2023/python-neo4j-client

Para mejorar los tiempos de carga, utilizamos la estrategia de levantar el CSV en python transformando cada fila a un mapa de propiedades y luego que tenemos la lista de mapas, se puede abrir una sesión con el driver de neo4j e insertar todos los nodos en la misma transacción. Eso nos evita tener que abrir una sesión por cada nodo a insertar y mejora mucho la performance.

Listado 1 Snippet código cypher para carga entidad

```
UNWIND $players_map as map
CREATE (p:Player)
SET p = map
RETURN p
```

Con esta estrategia pudimos insertar 60 mil jugadores en 15-20 segundos.

Por otro lado, para mantenernos dentro de los umbrales tuvimos que generar los nodos año a año a nivel aplicativo para poder para poder reducir la cantidad de datos a enviar en cada sesion. Con este proceso, pudimos procesar desde 1997 hasta 2020, 101 mil nodos y 160 mil relaciones.

II-D. *Diseño del grafo*

En esta sección mostramos el diseño del grafo en cuanto a sus nodos y relaciones.

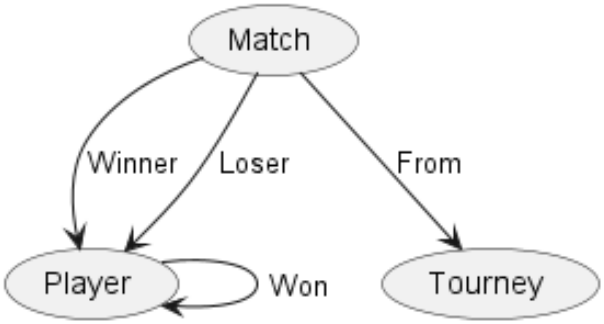

Figura 4: Diseño del grafo.

Podemos distinguir las entidades más importantes, el jugador, los torneos y los encuentros dentro de los torneos. Allí tendremos la información relacionada a como fue el resultado de cada encuentro, las características del torneo como por ejemplo el tipo de cancha y las características de cada jugador, por ejemplo si es diesto y su altura.

III. Experimentación

Para esta sección mediante el python notebook desarrollado[6] vamos ejecutando las consultas y visualizando su resultado.

III-A. Número Roger Federer: Pablo Cuevas

Esta primer consulta busca evidenciar la potencialidad de realizar recorridos en los grafos y lograr tener búsquedas de caminos con origen y destino. En este caso queremos calcular el Número Roger Federer desde el Tenista Uruguayo Pablo Cuevas.
El Número Roger Federer se calcula como el largo del camino de partidos ganados entre jugadores hasta llegar a ganarle a Roger Federer y comenzando en el jugador buscado, o sea, es la cantidad de ”ganados por transitiva”.

En este caso queremos un camino partidos ganados entre jugadores desde Pablo Cuevas a Roger Federer. Lo realizamos con el siguiente código:

Listado 2 Nro Roger Federer

```
MATCH path = shortestPath((start:Player
{player_id: '104655'})-[:WON*1..10]->
(end:Player {player_id: '103819'}))
RETURN nodes(path)
```

Figura 5: Resultado Pablo Cuevas.

III-B. Estadísticas Roger Federer Win-Rate

En la siguiente consulta recolectamos estadísticas sobre el Jugador Roger Federer. Conoceremos la cantidad de partidos ganados, perdidos y el Win-Rate.

Listado 3 Estadísticas Federer

```
MATCH (player:Player {player_id: '103819'})
OPTIONAL MATCH (player)-[:WINNER]->(won:Match)
WITH player, count(won) AS matches_won
OPTIONAL MATCH (player)-[:LOSER]->(lost:Match)
WITH player, matches_won, count(lost) AS matches_lost
RETURN player.name_last as Last_Name, matches_won as Wins,
matches_lost as Loses,
toFloat(matches_won)/(toFloat(matches_lost)
+toFloat(matches_won))
AS win_rate
```

[6]https://gitlab.fing.edu.uy/bdnr-2023/python-neo4j-client/-/blob/main/sources/notebooks/BDNR2023_tareafinal.ipynb

Cuadro I: Estadíticas Roger Federer

Wins	***Loses***	***Win Rate***
1251	274	0.82

En el cuadro visto, se pudo observar que Roger Federer presenta un Win-Rate de 0.82, en sus más de 1500 juegos. Esto es, gana 8 de cada 10 juegos.

En GrandSlam Wimbledon (match-level G y tipo cancha pasto)

Cuadro II: Estadíticas Roger Federer - Grand Slam - Wimbledon

Wins	***Loses***	***Win Rate***
102	13	0.88

Podemos ver que al acotar la búsqueda a solo partidos del torneo Grand Slam el Win-Rate incrementa a casi 9 partidos de cada 10. Roger Federer tiene 20 torneos Grand Slam ganados y fue finalista en otros 11 de ellos.

III-C. Top Jugadores en Wimbledon y Roland Garros

En este análisis continuamos indagando sobre los Torneos ’G’ (Grand Slam) y ahora teniendo en cuenta el tipo de cancha podemos distinguir si es Wimbledon (Grass) o Roland Garros (Clay).

Listado 4 Top 3 Players

```
 MATCH (player:Player)
OPTIONAL MATCH (player)-[:WINNER]->
(won:Match {level: 'G', surface: 'Clay'}) // Grass
WITH player, count(won) AS matches_won
OPTIONAL MATCH (player)-[:LOSER]->
(lost:Match {level: 'G', surface: 'Clay'}) //Grass
WITH player, matches_won, count(lost) AS matches_lost
RETURN (player.name_first + ' ' + player.name_last) as name,
matches_won, matches_lost,
  CASE matches_lost
    WHEN 0 THEN 0
    ELSE toFloat(matches_won)/toFloat(matches_lost)
  END AS win_loss_ratio
ORDER BY win_loss_ratio DESC
LIMIT 3
```

Ejecutando la consulta anterior, obtenemos:

Cuadro III: Top 3 jugadores - Roland Garros

Jugador	***Wins***	***Loses***	***Win Rate***
Nadal	93	3	0.96
Kuerten	36	7	0.83
Djokovic	69	14	0.83

Cuadro IV: Top 3 jugadores - Wimbledon

Jugador	***Wins***	***Loses***	***Win Rate***
Federer	102	13	0.88
Djokovic	72	10	0.87
Murray	57	10	0.85

III-D. Ranking Killer Players

Esta estadística busca conocer el TOP de jugadores que haya ganado más partidos contra jugadores en una mayor posición del ranking.

Listado 5 Ranking Killer Players Top

```
MATCH (f:Player) - [w:WINNER]-> (m:Match) <-[l:LOSER]
- (p:Player)
WHERE toInteger(w.rank) > toInteger(l.rank)
RETURN (f.name_first + ' ' + f.name_last) as name,
count(m) as won_best_ranked
ORDER BY won_best_ranked DESC
LIMIT 5
```

Cuadro V: Top 5 jugadores ranking killers

Jugador	***Wins***
Feliciano Lopez	146
Mikhail Youzhny	138
Tommy Hass	135
Ivo Karlovic	130
Julien Benneteu	128

Según el experimiento, el jugador que más ha ganado contra jugadores de mayor ranking es Feliciano Lopez, con 146 victorias.

III-E. Big3 Killer Players

Similar a la anterior, pero en este caso queríamos conocer cuales son los top 3 jugadores que han salido más victoriosos en los encuentros contra los Big 3 (Nadal, Federer, Djokovic).

Listado 6 Big3 Killer Players Top

```
MATCH (p:Player) - [w1:WON]-> (r:Player{player_id:'104745'})
MATCH (p) - [w2:WON] -> (n:Player{player_id:'104925'})
MATCH (p) - [w3:WON] -> (f:Player{player_id:'103819'})
RETURN (p.name_first + ' ' + p.name_last) as name,
w1.counter + w2.counter + w3.counter as total
ORDER BY total DESC
```

Cuadro VI: Top 3 jugadores Big3 killers

Jugador	***Wins***
Andy Murray	30
Jo-Wilfried Tsonga	18
Juan Martin del Potro	17

III-F. Jugadores Zurdos vs Diestros

Para explotar características sobre los jugadores, buscamos conocer la distribución de ganadores en partidos donde se enfrentaron jugadores diestros vs zurdos.

Listado 7 Zurdos vs Diestros

```
MATCH (:Player{hand:'L'}) - [:WINNER]-> (l:Match)
<-[:LOSER]-(:Player{hand:'R'})
MATCH (:Player{hand:'R'}) - [:WINNER]-> (r:Match)
<-[:LOSER]-(:Player{hand:'L'})
RETURN count(DISTINCT r) as cant_diestros,
count(DISTINCT l) as cant_zurdos
```

Cuadro VII: Zurdos vs Diestros

Wins Z	***Wins D***
5852	6163

Por lo tanto, con estos resultados, podemos ver que los tenistas diestros tienen una ligera ventaja en cuanto al win-rate sobre los tenistas zurdos.

III-G. CENTRALIDAD

En el contexto del análisis de grafos, la centralidad de grado es una medida que cuantifica la importancia de un nodo en función del número de conexiones que tiene con otros nodos en el grafo. El algoritmo de grado calcula esta medida para cada nodo, asignando un puntaje basado en el número de relaciones que tiene.

En este caso, utilizamos el algoritmo de grado para calcular la centralidad de grado de los jugadores en función del número de partidos ganados. La consulta contabiliza el número de partidos ganados para cada jugador y los suma para obtener el puntaje final de grado.

Dado que no pudimos utilizar la librería de Graph Data Science (GDS) en nuestra versión de Neo4j, implementamos el cálculo de grado con pesos manualmente en la consulta Cypher. Asignamos un peso específico a cada nivel de partido y luego sumamos los pesos correspondientes a los partidos ganados por cada jugador.

Listado 8 Centralidad

```
MATCH (p:Player)-[w:WINNER]->(m:Match)
WITH p, m, CASE m.level WHEN 'A' THEN 1 WHEN 'G' THEN 5
WHEN 'M' THEN 2 WHEN 'F' THEN 3 ELSE 0 END AS weight
RETURN (p.name_first + ' ' + p.name_last) AS playerName,
sum(weight) AS score
ORDER BY score DESC LIMIT 10
```

Cuadro VIII: Calculo centralidad

Jugador	***score***
Roger Federer	3159
Rafael Nadal	2472
Novak Djokovic	2433
Andy Murray	1658
David Ferrer	1504
Tomas Berdych	1402
Lleyton Hewitt	1325
Andy Roddick	1279
Stan Wawrinka	1247
Tommy Haas	1114

Los resultados obtenidos muestran los jugadores con mayor centralidad de grado en función del número de partidos ganados. Roger Federer obtiene el puntaje más alto, lo que indica que ha ganado la mayor cantidad de partidos en comparación con otros jugadores en el grafo. Rafael Nadal y Novak Djokovic también tienen puntajes altos, lo que indica su éxito en los partidos ganados.

IV. Conclusiones y trabajo futuro

Según lo visto se pueden realizar análisis muy interesantes recopilando estadísticas y realizando operaciones sobre caminos. Creemos que para culmninar este trabajo habría que realizar una interfaz Web en donde el usuario pueda explorar y realizar consultas de forma dinámica.

Por otro lado, podríamos agregar más fuentes de datos, considerando por ejemplo no solo ATP sino también WMA que son todos los torneos femeninos de tenis.