Proyecto Final

Bases de Datos no Relacionales

Grupo 8

Martín Ghazarian 3894679-9

Gianni Testa Abelar 4891674-8

Índice

1. Introducción

El siguiente informe documenta el proyecto final de la materia Bases de Datos no relacionales, dictado en 2022. El objetivo del proyecto es armar una base de datos de gráfos con datos de Jugadores, Equipos y Partidos de la NBA, relacionando cada jugador y equipo con los partidos que jugó junto a las estadísticas de cada partido. A su vez, otro de los objetivos del proyecto es poder encontrar parejas de jugadores que hayan jugado el mismo partido. La motivación de este trabajo vino de averiguar con que estrellas de la NBA compartió un partido Esteban Batista, único jugador de basketball uruguayo en jugar en la NBA, inspirándonos tanto en el proyecto“El número del Loco” del año pasado y el proyecto “The Oracle of Kevin Bacon”

2. Desarrollo del Proyecto

2.1. Obtención de los datos

Los datos fueron obtenidos de la página **Kaggle.com**, los cuales fueron capturados mediante *scrapping* desde la página oficial de la NBA. Los datos estan disponibilizados en archivos csv, los cuales tienen la información de los equipos y jugadores, además de las estadísticas de los partidos para cada equipo y para cada jugador, desde el año 2004 hasta el año 2021.

2.2. Diseño del modelo

En cuanto al diseño de la estructura del grafo para la base de datos sabiamos como relacionar a los equipos y los jugadores con los partidos y a su vez, cada partido con su respectiva temporada.Pero lo complicado era cómo relacionar el jugador con el equipo para el que jugó ese partido, ya que un jugador puede pertenecer a mas de un equipo durante toda la temporada. En este sentido, optamos por agregar el identificador del equipo como un campo en la relación del jugador con el partido.

El resultado generó el siguiente modelo:

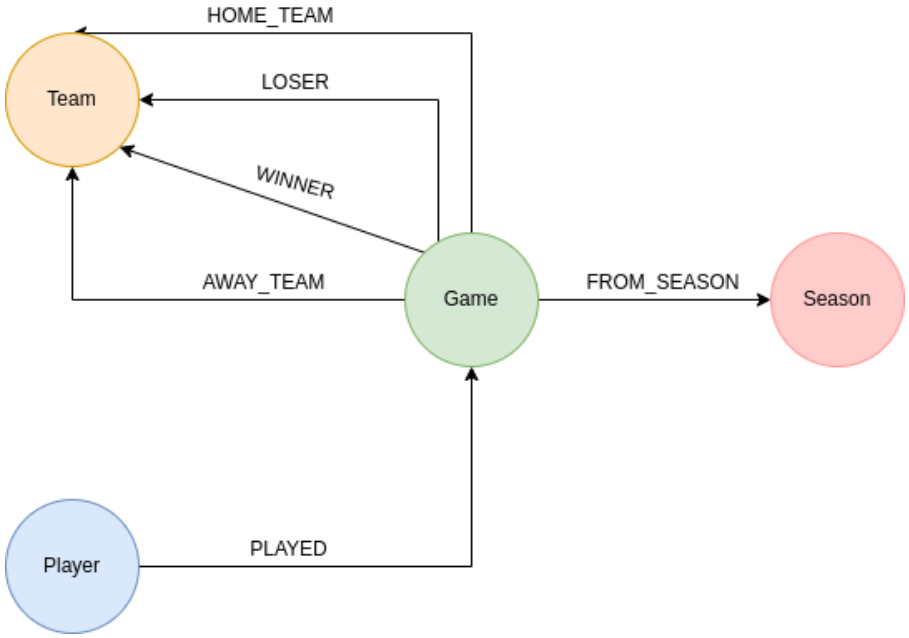

Figura 2.1: Estructura del grafo

Donde tenemos las siguientes entidades (nodos):

- **Team**
  - Representa a los Equipos, por ejemplo en el caso de Brooklyn Nets:

```
{
    "<id>" : 31, //identificador del nodo dentro de la base de datos
    "abbreviation" : "BKN",
    "arena" : "Barclays Center",
    "city" : "Brooklyn",
    "id" : 1610612751, //identificador dentro de los datos de carga
    "nickname" : "Nets"
}
```

- **Player**
  - Representa a los Jugadores, conteniendo el campo con su nombre:

```
{
    "<id>" : 26033, //identificador del nodo dentro de la base de datos
    "id" : 201939, //identificador dentro de los datos de carga
    "name" : "Stephen Curry"
}
```

- **Game**
  - Representa a los Partidos, tiene los campos que lo identifican en la base de datos y en el documento:

```
{
    "<id>" : 67, //identificador del nodo dentro de la base de datos
    "id" : 22100994, //identificador dentro de los datos de carga
    "arena" : "Barclays Center",
}
```

- **Season**
  - Representa a cada temporada y se compone de un único campo year que equivale al año de la temporada.

```
{
    "<id>" : 26033, //identificador del nodo dentro de la base de datos
    "year" : 2020-2021 //Los aos que se jug\'o la temporada
}
```

Y las siguientes relaciones entre entidades:

- **FROM_SEASON**
  - Vincula un partido con una temporada. No tiene atributos.
- **HOME_TEAM**
  - Vincula un partido con el equipo que fue local junto con las estadísticas del mismo.

```
{
    "<id>" : 81, //identificador del nodo dentro de la base de datos
    "AST" : 25.0 //asistencias
    "FG_PCT" : 0.512 //porcentaje de tiros de campo
    "FT_PCT" : 0.765 //porcentajes de tiros libres
    "PTS" : 108.0 //puntos
    "REB" : 32.0 //rebotes
}
```

- **AWAY_TEAM**
  - Vincula un partido con el equipo que fue visitante junto con las estadísticas del mismo.

```
{
    "<id>" : 81, //identificador del nodo dentro de la base de datos
    "AST" : 25.0 //asistencias
    "FG_PCT" : 0.512 //porcentaje de tiros de campo
    "FT_PCT" : 0.765 //porcentaje de tiros libres
    "PTS" : 108.0 //puntos
    "REB" : 32.0 //rebotes
}
```

- **WINNER**
  - Vincula un partido con el equipo que ganó el partido. No tiene atributos.
- **LOSER**
  - Vincula un partido con el equipo que perdió el partido. No tiene atributos.

- **PLAYED**
  - Representa a cada temporada y se compone de un unico campo year que equivale al año de la temporada.

```
{
    "<id>" : 129596 //identificador del nodo dentro de la base de datos
    "AST" : 2.0 //asistencias
    "BLK" : 0.0 //bloqueos
    "DREB" : 2.0 //rebotes defensivos
    "FG3A" : 3.0 //triples realizados
    "FG3M" : 1.0 //triples convertidos
    "FGA" : 9.0 //tiros de campo realizados
    "FGM" : 4.0 //tiros de campos encestados
    "FTA" : 2.0 //tiros libres realizados
    "FTM" : 2.0 //tiros libres encestados
    "OREB" : 2.0 //rebotes ofensivos
    "PF" :  1.0 //faltas personales
    "PTS" : 11.0 //puntos
    "REB" : 4.0 //rebotes
    "SECS" : 1784 //segundos en cancha
    "STL" : 1.0 //robos de bal]'on
    "TO" :  3.0 //p\'erdidas
    "team" : 1610612750 //identificador del equipo extraido de los datos de
    carga
}
```

2.3. Carga de datos

En cuanto a la carga de datos en un primer lugar se optó por utilizar el comando de Cypher ***LOAD CSV***, el cual es nativo de Neo4j y bastante sencillo de utilizar, con la limitante de no manejar archivos muy grandes. Para la carga de los nodos de los Equipos, como para la de los partidos, no tuvimos problema, pero en cuanto quisimos agregar los jugadores y su relación con los partidos la carga tomaba mucho tiempo y colapsaba por falta de memoria. Dado este inconveniente optamos por la carga de estos nodos y relaciones por medio de python, recorriendo el archivo *game_detail.csv* y mediante una conexión con la base de datos de Neo4j ir agregando uno por uno.
Las librerias utilizadas fueron:

- **pandas**: Es una librería opensource de python para manipulación y análisis de datos. En particular utilizamos la herramienta ***read_csv*** para poder iterar las filas de los archivos csv y la función ***isnull*** para determinar valores nulos y ponerles valores por defecto *'hardcoded'*
- **neo4j**: Es la librería para poder conectarse y manipular bases de datos de grafos en Neo4j con python.

A continuación dejamos todos los script de carga mediante el comando ***LOAD CSV***:

- ```
  LOAD CSV WITH HEADERS FROM 'file:///games.csv' AS row
  MERGE (s: Season {year: row.SEASON + "-"+toString(toInteger(row.SEASON)+1)})
  ```

Listing 1: Creación de los nodos de las temporadas

```
LOAD CSV WITH HEADERS FROM 'file:///teams.csv' AS row
MERGE (t: Team {id: row.TEAM_ID, nickname: row.NICKNAME, abbreviation: row.ABBREVIATION,
city: row.CITY, arena: row.ARENA })
```

Listing 2: Creación de los nodos de los equipos

- ```
  LOAD CSV WITH HEADERS FROM 'file:///games.csv' AS row
  MERGE (g: Game {id: row.GAME_ID, date: date(row.GAME_DATE_EST)})
  ```

Listing 3: Creación de los nodos de los partidos

```
MATCH (g:Game)-[:HOME_TEAM]->(t:Team)
SET g.arena = t.arena
```

Listing 4: Agregamos la información del estadio donde se jugó el partido (luego de cargar las relaciones "HOME_TEAM")

- ```
  LOAD CSV WITH HEADERS FROM 'file:///games.csv' AS row
  MERGE (g:Game {id: row.GAME_ID})
  MERGE (s:Season {year: row.SEASON + "-"+toString(toInteger(row.SEASON)+1)})
  MERGE (g)-[:FROM_SEASON]->(s)
  ```

Listing 5: Creación de la relación entre los partidos y las temporadas

Tal como comentamos previamente, disponibilizamos dos archivos python para la carga de los datos de Jugadores, partidos jugados por cada jugador junto a sus estadísticas y equipos locales y visitantes de cada partido y sus estadísticas. Los archivos son:

- ***create_played.py :*** Crea los nodos con la información de los jugadores como también la relación (junto a sus estadísticas) a cada partido jugado.
- ***create_home_away.py :*** Crea las relaciones (junto con las estadísticas) entre cada partido y los equipos local (HOME_TEAM) y visitantes (AWAY_TEAM).
  ```

2.4. Experimentación

2.4.1. Consultas sobre las distancias entre Esteban Batista y jugadores famosos de la NBA

La motivación principal del proyecto fue encontrar jugadores famosos de la NBA con los que Batista haya compartido la cancha, utilizando la función del ***shortest path*** así obtener si jugaron en el mismo partido o si jugó con alguien que jugó en otro partido contra esa estrella. Creamos algunas Queries con los siguientes jugadores y este fue el resultado:

- ```
  MATCH sp=shortestPath((p:Player)-[:PLAYED*]-(rival))
  WHERE p.name='Esteban Batista' AND rival.name='LeBron James'
  RETURN sp, length(sp)
  ```

Listing 6: LeBron James

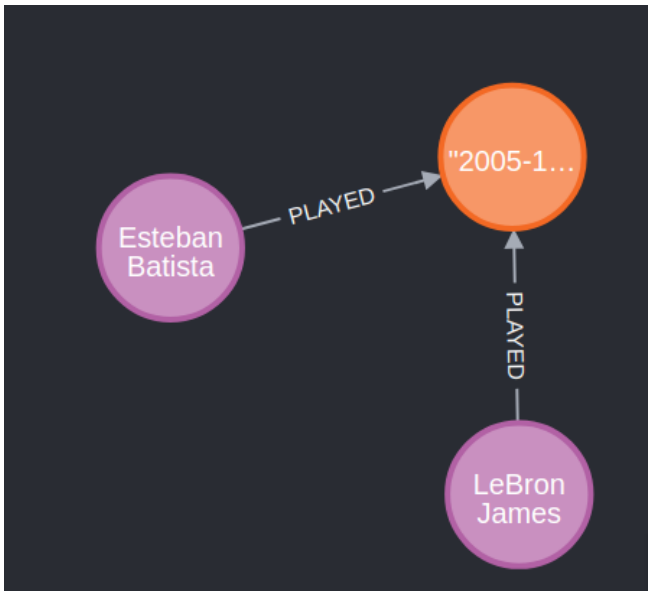

Figura 2.2: Jugaron el mismo partido el día 13/12/2005
  ```

```
MATCH sp=shortestPath((p:Player)-[:PLAYED*]-(rival))
WHERE p.name='Esteban Batista' AND rival.name='Stephen Curry'
RETURN sp, length(sp)
```

Listing 7: Stephen Curry

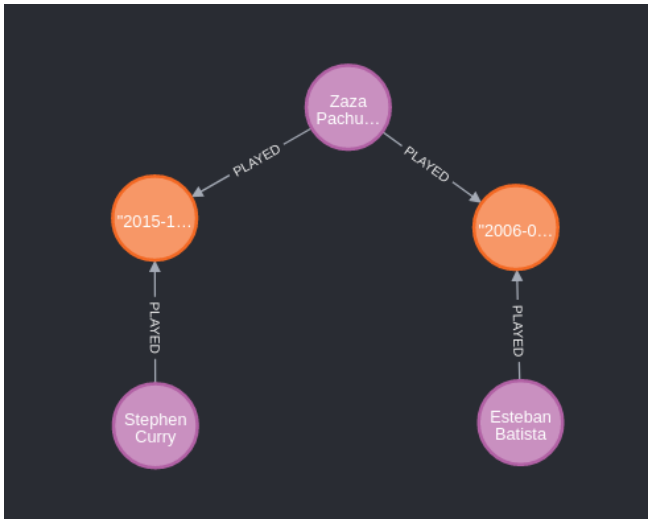

Figura 2.3: Nunca jugaron un mismo partido

2.4.2. Otras Consultas de Interés

Dado a que logramos obtener una buena cantidad de datos y una estructura del grafo con conexiones fuertes entre los jugadores, los partidos y los equipos , decidímos obtener algunas estadísticas “de color” basándonos en una de las super estrellas de la liga como lo es **LeBron James**.

```
MATCH (p:Player {name: "LeBron James"})-[pl:PLAYED]->(g:Game)
MATCH (g)-[w:WINNER]->(t:Team)
WHERE t.id = pl.team
RETURN g.arena as Arena,COUNT(w) as wins
ORDER BY wins DESC
```

Listing 8: Listado de estadios donde ganó mas partidos LeBron

Resultado:


```
[
  {
    "Arena": "Quicken Loans Arena",
    "wins": 405
  },
  {
    "Arena": "AmericanAirlines Arena",
    "wins": 178
  },
  {
    "Arena": "Staples Center",
    "wins": 109
  },
  {
    "Arena": "State Farm Arena",
    "wins": 28
  },
  ...
]
```


```
MATCH (p:Player {name: "LeBron James"})-[pl:PLAYED]->(g:Game)
MATCH (g)-[w:WINNER]->(t:Team)
WHERE t.id = pl.team
RETURN g.arena AS Arena,SUM(pl.PTS) as Total_PTS, COUNT(w) as WINS
ORDER BY Total_PTS DESC
```

Listing 9: Estadio donde metio más puntos ganando el partido en su carrera

Resultado:


```
[
  {
    "Arena": "Quicken Loans Arena",
    "Total_PTS": 11016.0,
    "WINS": 405
  },
  {
    "Arena": "AmericanAirlines Arena",
    "Total_PTS": 4691.0,
    "WINS": 178
  },
  {
    "Arena": "Staples Center",
    "Total_PTS": 2920.0,
    "WINS": 109
  },
  ...
]
```

```
MATCH (p:Player {name: "LeBron James"})-[pl:PLAYED]->(g:Game)
MATCH (g)-[l:LOSER]->(t:Team)
WHERE t.id = pl.team
RETURN g.arena as Arena,COUNT(l) as loses
ORDER BY loses DESC
```

Listing 10: Estadio donde perdió más partidos

Resultado:

```
[
  {
    "Arena": "Quicken Loans Arena",
    "loses": 142
  },
  {
    "Arena": "Staples Center",
    "loses": 72
  },
  {
    "Arena": "AmericanAirlines Arena",
    "loses": 59
  },
  {
    "Arena": "TD Garden",
    "loses": 31
  },
  ...
]

```

```
MATCH (p:Player {name: "LeBron James"})-[pl:PLAYED]->(g:Game)
MATCH (g)-[l:LOSER]->(t:Team)
MATCH (g)-[tw:WINNER]->(rival:Team)
WHERE t.id = pl.team
RETURN rival.city AS City, rival.nickname AS NickName,COUNT(l) as loses
ORDER BY loses DESC
```

Listing 11: Equipos con los que perdió más partidos

Resultado:


```
[
  {
    "City": "Boston",
    "NickName": "Celtics",
    "loses": 47
  },
  {
    "City": "Indiana",
    "NickName": "Pacers",
    "loses": 36
  },
  {
    "City": "Golden State",
    "NickName": "Warriors",
    "loses": 34
  },
  {
    "City": "Chicago",
    "NickName": "Bulls",
    "loses": 34
  },
  {
    "City": "San Antonio",
    "NickName": "Spurs",
    "loses": 34
  },
  ...
]

```


Estos son solo algunos ejemplos, tambien se podria buscar los equipos a los que un Jugador le metió mas puntos, o el estadio donde jugó mas minutos un jugador, como tambien buscar los mejores 10 partidos en porcentaje de triples, o tiros de campo, por ejemplo. Tambien diferenciar estadisticas entre los partidos que jugó como local y los de visitante, los que ganó y los que perdió, y las opciones son bastantes como para poder utilizar con fines de analisís de la NBA.

2.4.3. Comparativa de rendimiento frente a una base de datos relacional

En este análisis se creó una base de datos de Grafos en Neo4j para cargar los data sets que contienen información de la NBA (partidos, jugadores, equipos) y además se creó una base de datos relacional en MS SQL Server. Ambos motores se cargaron en un mismo hardware con las siguientes características:

SO: Windows 10
CPU: 4
Memoria: 8 RAM

Para el análisis comparativo, se eligió el algoritmo de "camino más corto" entre 2 nodos, en lenguaje de negocio lo que se quiso resolver con las consultas fue "qué tan lejos estuvo Estaban Batista de jugar con Lebron James, entendiendo como "lejos" la cantidad de saltos que tiene que dar en un modelo de grafos que relaciona los partidos que jugaron entre jugadores.
Para dicho análisis se utilizó la función shortestPath en Cypher para la base Neo4j, obteniendo el resultado en <1 seg.
Para MSSQL Server se creó una CTE (common table expression), para hacer recursiones sobre la misma tabla y el resultado se obtuvo en 4 segundos.
Concluímos que para la cantidad de datos cargados y el hardware utilizado podemos demostrar empíricamente la diferencia en la performance para obtener los datos de camino más corto.
A continuación la Query utilizada en BD Neo4j para obtener el camino más corto:

```
MATCH sp=shortestPath((p:Player)-[:PLAYED*]-(rival))
WHERE p.name='Esteban Batista' AND rival.name='LeBron James'
RETURN sp, length(sp)
```

Listing 12: Query en Cypher

Y también la query de SQL con CTE Recursivo, para obtener todos los partidos que se jugaron contra Batista, y luego en el resultado ver en cuál estuvo presente "LeBron James".

```
Declare @RowNo int =0;
WITH CTE as
   (
    SELECT 0 as N
    , batista.game_id
    , PLAYER_NAME
    FROM [DWGrafos].[dbo].[games_details] batista
    WHERE batista.PLAYER_NAME = 'Esteban Batista'

    UNION ALL

    SELECT  N+1
    , CTE.game_id
    , compl.PLAYER_NAME
    FROM [DWGrafos].[dbo].[games_details] compl
    INNER JOIN CTE ON compl.game_id = CTE.GAME_ID
    WHERE N<2
    )

SELECT *
FROM CTE
WHERE N=1
AND cte.PLAYER_NAME = 'LeBron James'
```

CTE lo usamos tomando un caso base “Esteban Batista”, y luego en el UNION ALL agregamos todos los “nodos” adyacentes a “Esteban Batista”
Luego, filtramos por cte.PLAYER_NAME = “LeBron James” para tomar el nodo de dicho jugador, de esta manera vemos el “camino más corto” entre ambos jugadores.

2.5. Disposición de los datos

Los datos estan dispuestos en un repositorio público de GitLab Fing, en el cual encontrarán:

- Archivos .csv con los datos originales que fueron utilizados en este proyecto
- Un archivo .txt con las queries en Cypher para la carga de datos, junto con los script de python previamente mencionados.
- archivo .dump para cargar directamente la base de datos en Neo4j
- Readme con la descripción del proyecto y de los datos incluídos en el repositorio.

3. Conclusiones

Todo el trabajo realizado en este proyecto es una interesante introducción a la herramienta Neo4j y a las herramientas que provee tanto Neo4j como las bases de datos de grafos en general. En un principio la idea fue simplemente ver las conexiones entre jugadores, para ver con quien habia jugado Esteban Batista, pero mediante fuimos creando el modelo y cargando la base de datos, nos dimos cuenta que tiene bastantes datos y potencial para ser utilizado como herramienta de análisis de estadísticas de juego.

Una de las mejoras sería obtener más información acerca de, por ejemplo, la cantidad de posesiones de balón que tuvo cada equipo en el partido, ya que hoy en día se manejan muchas estadísticas de avanzada en base a un numero de posesiones, por ejemplo, puntos/rebotes/asistencias cada 100 posesiones.

También se podría obtener los rangos de entrada y salida de los jugadores en el partido, para no solo tener los minutos jugados si no también saber si dos jugadores estuvieron en cancha en el mismo momento, por ejemplo, como también poder obtener la eficiencia ofensiva de un equipo cuando cierto número de jugadores está en cancha.