

BBDD NOSQL

Práctica 3 - Neo4j

José Manuel Bustos Muñoz

1. Describir (en un párrafo) la base de datos basada en grafos que se va a crear y la utilidad que tendrá.

Por ejemplo, base de datos con información sobre lectores y libros. Se almacenará información sobre los libros que ha leído cada lector con valoraciones de los mismos. Cada lector podrá saber qué libros han leído otros lectores que tienen los mismos gustos que ellos.

La base de datos que se va a crear contendrá la información de distintos partidos de la liga de baloncesto NBA. Se almacenará información sobre los partidos disputados, los equipos participantes, los jugadores involucrados y las canastas anotadas y falladas en cada partido. Al tener toda esta información se podrán analizar todos los partidos para ver lo que ha ocurrido. Esta información puede ser muy útil para un equipo a la hora por ejemplo de preparar otro partido contra el mismo rival, o ver que jugadores están teniendo mejores estadísticas.

En esta práctica se crean algunas relaciones y atributos para cada tipo de nodo, pero se puede ampliar con bastante más información dando lugar a poder realizar bastantes más análisis de los datos relacionados con los partidos disputados.

2. Dar un nombre que describa la aplicación que utilizará la base de datos propuesta en el apartado 1.

Por ejemplo, red social para amantes de la lectura.

Análisis estadístico de los partidos de la NBA.

3. Describir razonadamente el diseño de la base de datos.

El diseño de la base de datos sería el siguiente:

- *Tipos de nodos:*

1. **Partido:** nodo que representa los distintos partidos de los que se almacenen datos. Tendrá los atributos id identificativo, nombre del partido y tipo de partido para saber si es un partido de liga regular o de playoffs, y así poder hacer análisis separados entre la temporada regular y la posttemporada.
2. **Equipo:** nodo que representa los equipos que participen en los partidos almacenados. Tendrá los atributos id identificativo, nombre del equipo, y conferencia en la que disputa la liga el equipo para poder hacer análisis por cada conferencia si así se desea.
3. **Jugador:** nodo que representa a los jugadores pertenecientes a los equipos que disputan los partidos guardados. Tendrá los atributos id identificativo y nombre del jugador.
4. **Canasta:** nodo que representa las jugadas ocurridas en el partido. Tendrá los atributos id identificativo, valor de la canasta (0 para las acciones falladas por el jugador que la ejecuta), y minuto del partido en el que ocurre la acción.

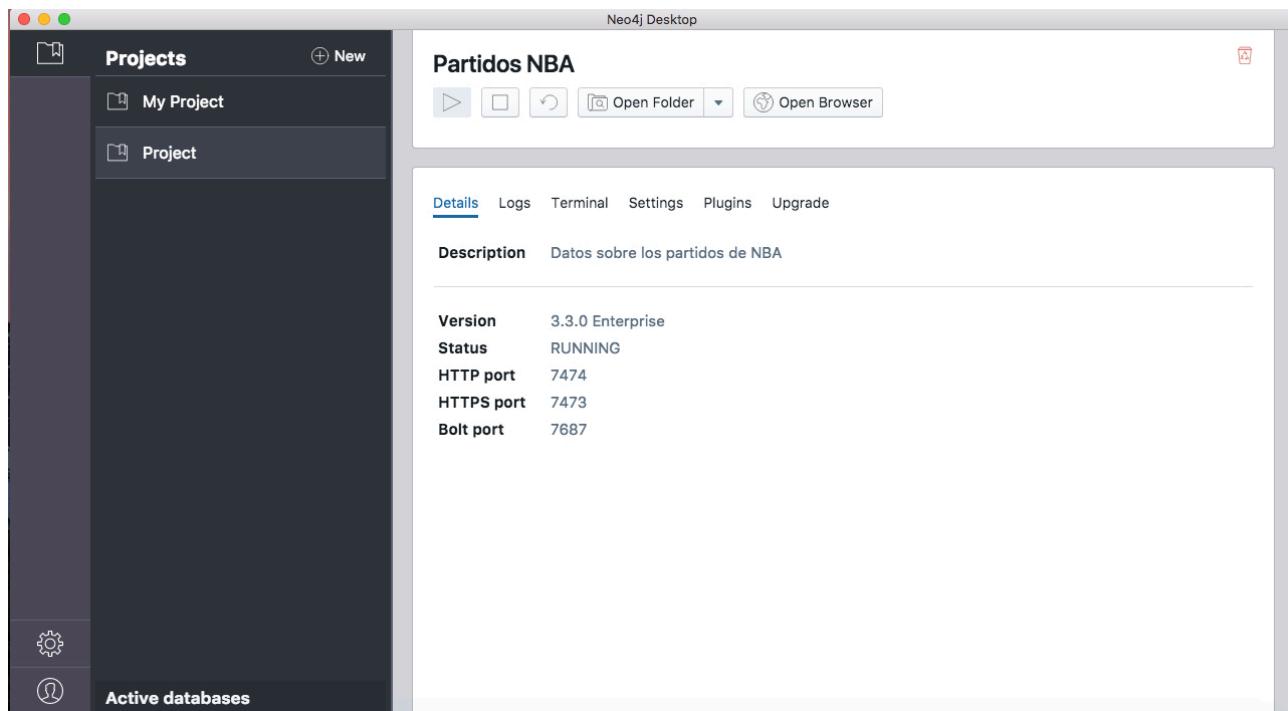
- *Relaciones:*

1. **Jugador - asiste_a - Jugador:** relación entre jugadores para analizar las asistencias que se dan en los partidos, y quién las realiza y poder contabilizar el impacto de cada jugador en un partido y en su equipo. Como en un mismo partido puede haber más de una asistencia de un jugador al mismo jugador, habrá que poner propiedades en esta relación y poder distinguir las distintas asistencias.
2. **Jugador - juega_en - Equipo:** relación entre jugadores y equipos, para saber en qué equipo juega cada jugador.
3. **Jugador - anota - Canasta:** relación entre jugadores y canastas, para saber qué jugador ejecuta cada acción del partido.
4. **Canasta - pertenece_a - Equipo:** relación entre canastas y equipos, para saber cada canasta anotada o fallada a qué equipo corresponde, y por ejemplo poder saber los puntos anotados por cada equipo.
5. **Equipo - participa_en - Partido:** relación entre equipos y partidos para tener los equipos participantes en cada partido almacenado.
6. **Canasta - ocurrio_en - Partido:** relación entre canastas y partidos para saber cada acción en qué partido ocurrió.

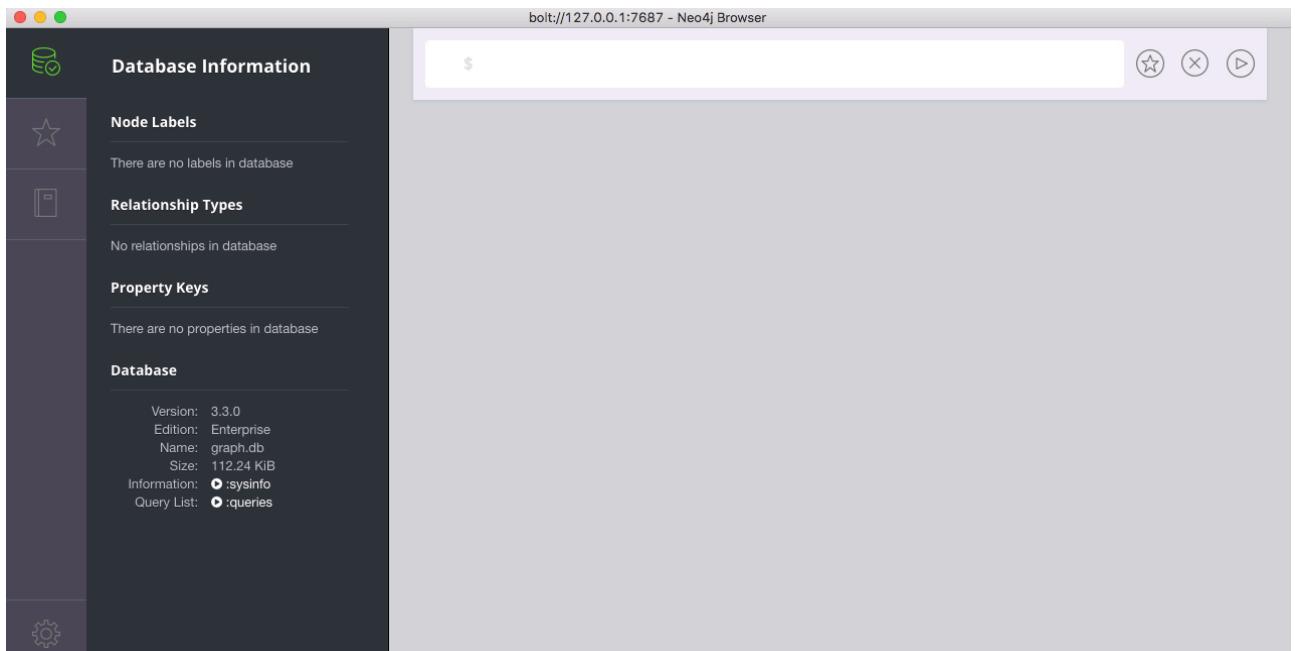
Cómo se dijo anteriormente, en cada tipo de nodo se podrían representar bastantes más atributos, así como crear más nodos y relaciones entre ellos, pero haría demasiado compleja la base de datos para la práctica.

4. Crear la base de datos en Neo4j y realizar la carga de datos describiendo el proceso seguido.

Vamos a trabajar con Neo4j, la aplicación instalada en el ordenador local. Con Neo4j Desktop podemos gestionar las bases de datos. Tenemos un proyecto, y dentro las bases de datos.



En la que corresponda hay un enlace para abrir el navegador Neo4j y así ya poder cargar datos y trabajar con la base de datos.



Tenemos los datos a cargar repartidos en distintos ficheros .csv, correspondientes a cada tipo de nodo, y a las distintas relaciones que hemos definido. La práctica está realizada en el sistema operativo de MAC, y para importar los ficheros .csv desde Neo4j se deben llevar los archivos a la carpeta “import” de la aplicación.

Cargamos uno a uno los ficheros:

- Carga de nodo jugadores: Importamos el fichero jugadores.csv → `LOAD CSV WITH HEADERS FROM "file:///jugadores.csv" AS csvLine CREATE (j:Jugador { id:toInt(csvLine.id), nombre: csvLine.nombre, posicion: csvLine.posicion})`



Se añaden los 20 jugadores.

- Carga de nodo equipos: Importamos el fichero equipos.csv → **LOAD CSV WITH HEADERS FROM "file:///equipos.csv" AS csvLine CREATE (e:Equipo { id:toInt(csvLine.id), nombre:csvLine.nombre})**

The screenshot shows the Neo4j Browser interface. At the top, a query window displays the command: **\$ LOAD CSV WITH HEADERS FROM "file:///equipos.csv" AS csvLine CREATE (e:Equipo { id:toInt(csvLine.id), nombre:csvLine.nombre})**. Below the query window, a results table shows the output of the query: **\$ MATCH (n:Equipo) RETURN n LIMIT 25**. The results pane displays four nodes, each represented by a red circle containing the team name: **Houston Rockets**, **Boston Celtics**, **Cleveland**, and **Golden State Warrio...**. A tooltip indicates there are four nodes in total. On the left side, a sidebar provides navigation options: Graph (selected), Table, Text, and Code.

Se cargan los 4 equipos.

- Carga de nodo partidos: Importamos el fichero partidos.csv → **LOAD CSV WITH HEADERS FROM "file:///partidos.csv" AS csvLine CREATE (p:Partido { id:toInt(csvLine.id), nombre:csvLine.nombre, tipo:csvLine.tipo})**

The screenshot shows the Neo4j Browser interface. At the top, a query window displays the command: **\$ LOAD CSV WITH HEADERS FROM "file:///partidos.csv" AS csvLine CREATE (p:Partido { id:toInt(csvLine.id), nombre:csvLine.nombre, tipo:csvLine.tipo})**. Below the query window, a results table shows the output of the query: **\$ LOAD CSV WITH HEADERS FROM "file:///partidos.csv" AS csvLine CREATE (p:Part...**. The results pane displays a message: **Added 2 labels, created 2 nodes, set 6 properties, completed after 111 ms.** A tooltip indicates two nodes were created. On the left side, a sidebar provides navigation options: Table (selected), Graph, Text, and Code.

- Carga de nodo canastas: Importamos el fichero canastas.csv → LOAD CSV WITH HEADERS FROM "file:///canastas.csv" AS csvLine CREATE (c:Canasta { id:toInt(csvLine.id), valor:toInt(csvLine.valor), minuto:toInt(csvLine.minuto)})

Se añaden las canastas de los 2 partidos. En total 40 canastas.

The screenshot shows the Neo4j Database Information interface. On the left, under 'Node Labels', 'Canasta' is selected. Under 'Relationship Types', there are none listed. Under 'Property Keys', 'id', 'minuto', 'nombre', 'posicion', 'tipo', and 'valor' are listed. In the main area, a query window contains the command:

```
$ LOAD CSV WITH HEADERS FROM "file:///canastas.csv" AS csvLine CREATE (c:Canasta { id:toInt(csvLine.id), valor:csvLine.valor, minuto:csvLine.minuto})
```

Below the query, a message says: "Added 40 labels, created 40 nodes, set 120 properties, completed after 126 ms."

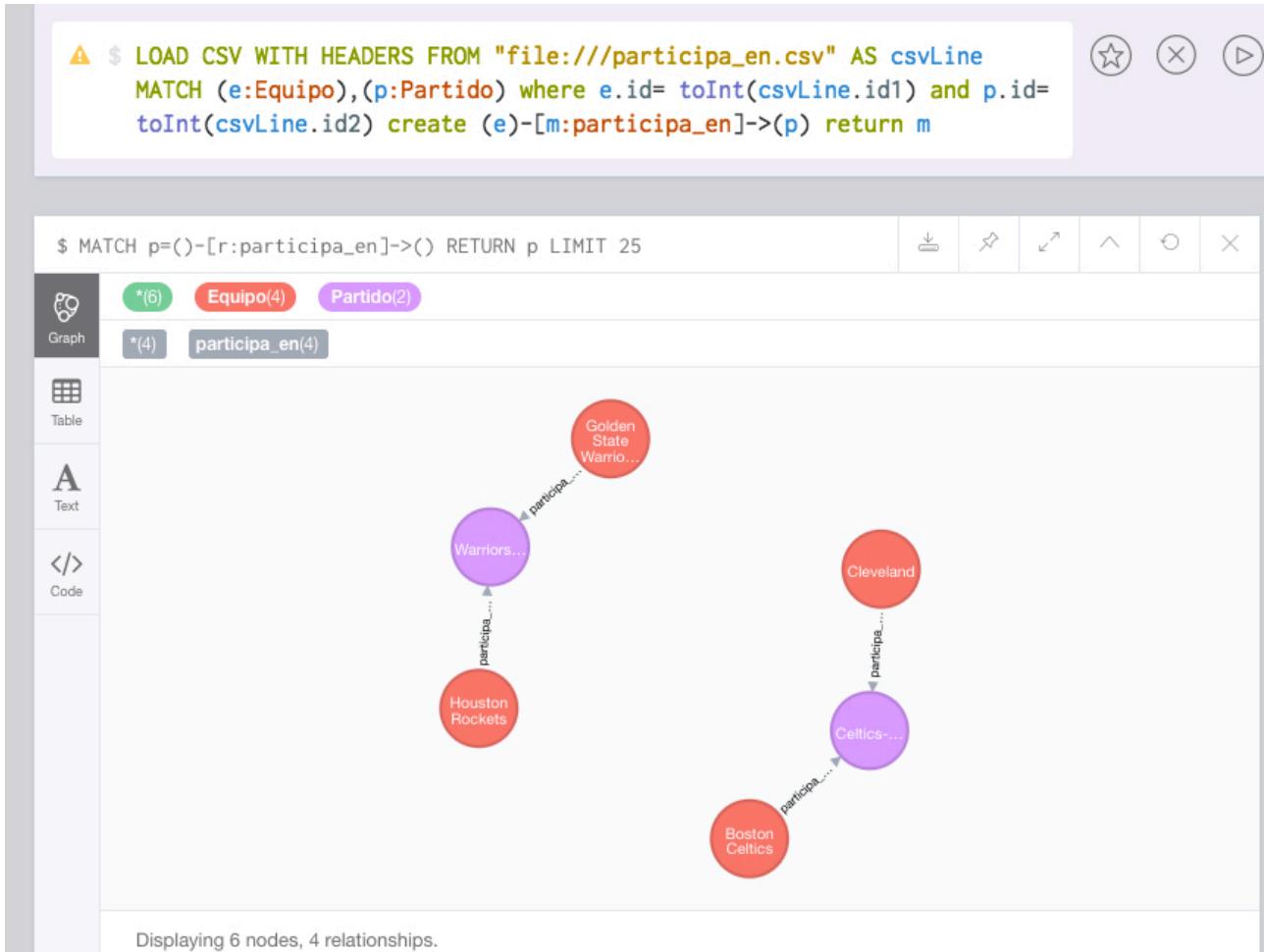
- Carga de relación juega_en: Importamos el fichero juega_en.csv → LOAD CSV WITH HEADERS FROM "file:///juega_en.csv" AS csvLine MATCH (j:Jugador),(e:Equipo) where j.id=toInt(csvLine.id1) and e.id=toInt(csvLine.id2) create (j)-[m:juega_en]->(e) return m

The screenshot shows the Neo4j Graph browser. At the top, a query window contains the command:

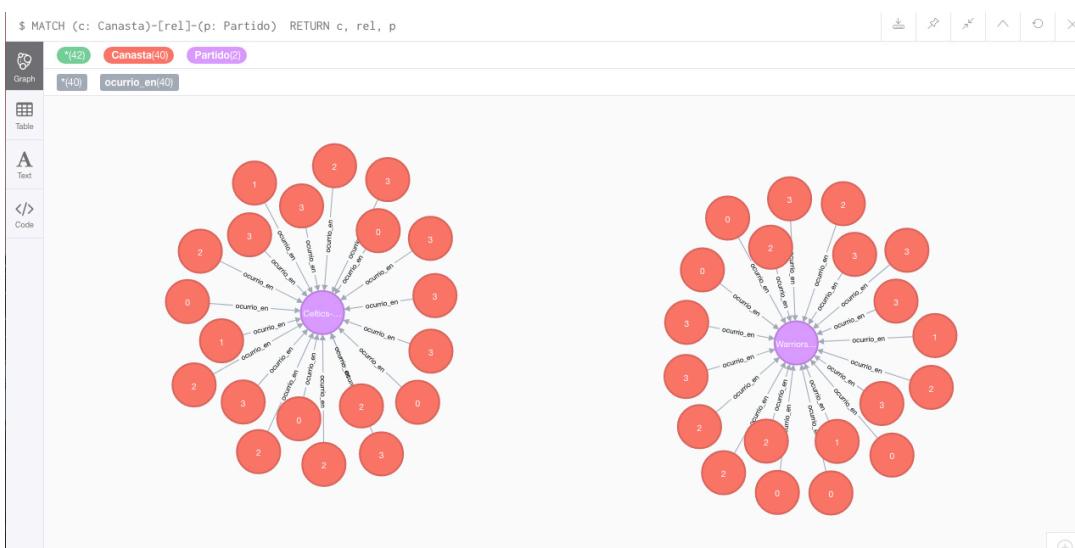
```
$ LOAD CSV WITH HEADERS FROM "file:///juega_en.csv" AS csvLine MATCH (j:Jugador),(e:Equipo) where j.id=toInt(csvLine.id1) and e.id=toInt(csvLine.id2) create (j)-[m:juega_en]->(e) return m
```

Below the query, a results window shows the query: \$ MATCH p=()-[r:juega_en]->() RETURN p LIMIT 25. The results pane displays a graph with 24 nodes and 20 relationships. Nodes are categorized: Jugador(20) in green, Equipo(4) in red, and relaciones(20) in grey. The graph shows clusters of blue nodes (players) connected to red nodes (teams) via grey arrows labeled 'juega_en'. One cluster of players is connected to the 'Houston Rockets' team, and another cluster is connected to the 'Cleveland' team.

- Carga de relación participa_en: Importamos el fichero participa_en.csv → LOAD CSV WITH HEADERS FROM "file:///participa_en.csv" AS csvLine MATCH (e:Equipo), (p:Partido) where e.id=toInt(csvLine.id1) and p.id=toInt(csvLine.id2) create (e)-[m:participa_en]->(p) return m



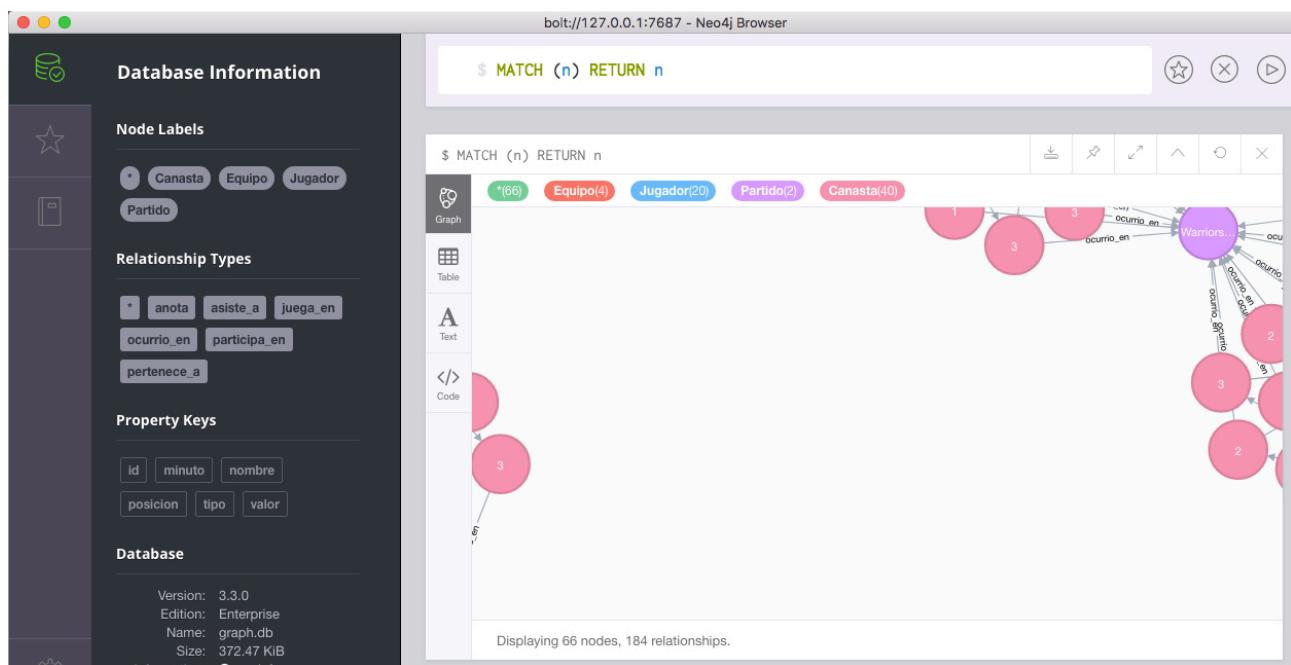
- Carga de relación ocurrio_en: Importamos el fichero ocurrio_en.csv → LOAD CSV WITH HEADERS FROM "file:///ocurrio_en.csv" AS csvLine MATCH (c:Canasta),(p:Partido) where c.id=toInt(csvLine.id1) and p.id=toInt(csvLine.id2) create (c)-[m:ocurrio_en]->(p) return m



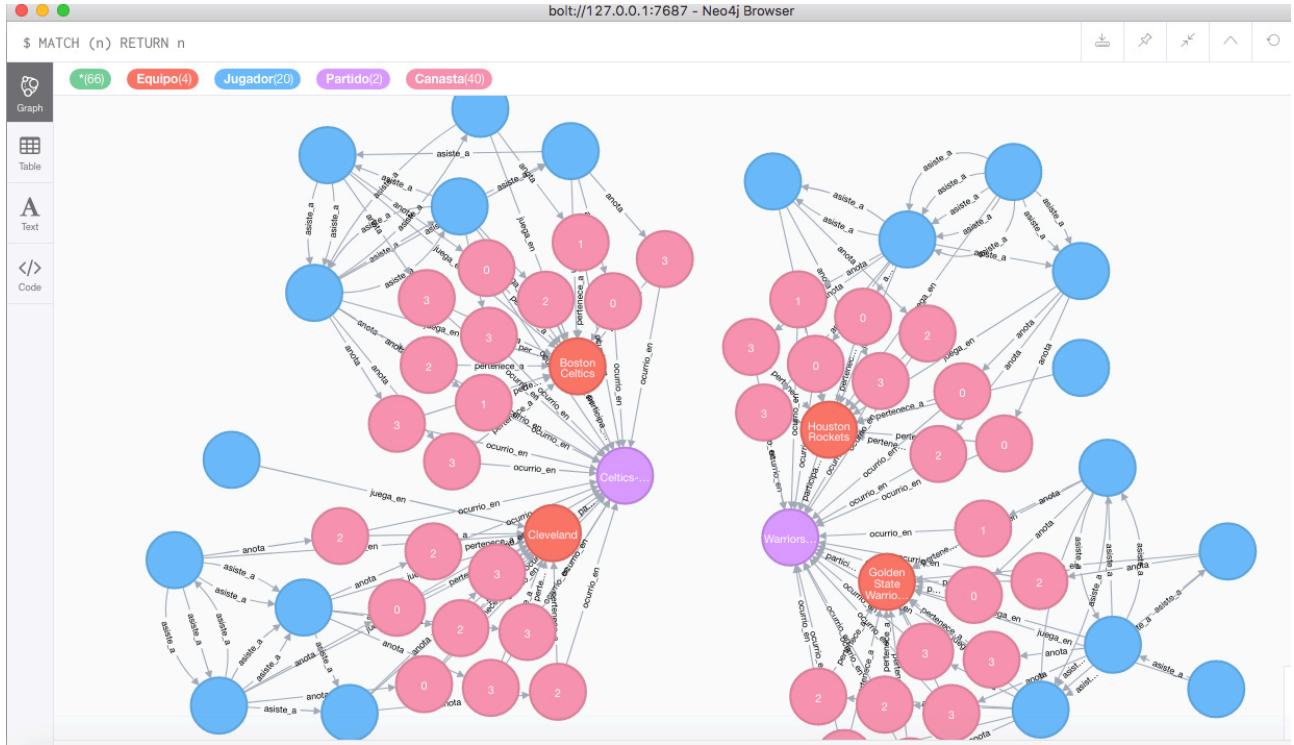
- Carga de relación pertenece_a: Importamos el fichero pertenece_a.csv → LOAD CSV WITH HEADERS FROM "file:///pertenece_a.csv" AS csvLine MATCH (c:Canasta), (e:Equipo) where c.id=toInt(csvLine.id1) and e.id=toInt(csvLine.id2) create (c)-[m:pertenece_a]->(e) return m
- Carga de relación anota: Importamos el fichero anota.csv → LOAD CSV WITH HEADERS FROM "file:///anota.csv" AS csvLine MATCH (j:Jugador),(c:Canasta) where j.id=toInt(csvLine.id1) and c.id=toInt(csvLine.id2) create (j)-[m:anota]->(c) return m
- Carga de relación asiste: Importamos el fichero asiste.csv → LOAD CSV WITH HEADERS FROM "file:///asiste_a.csv" AS csvLine MATCH (j1:Jugador),(j2:Jugador) where j1.id=toInt(csvLine.id1) and j2.id=toInt(csvLine.id2) create (j1)-[m:asiste_a{valor:toInt(csvLine.valor)}]->(j2) return m

Al terminar de cargar los archivos ya tendríamos todos los datos introducidos, tanto los distintos nodos como las relaciones.

Se puede visualizar el grafo entero:



Aquí podríamos ver los dos partidos que son los nodos de color morado, con los 2 equipos participantes en cada uno de los partidos, y alrededor de cada equipo las canastas que le pertenecen y sus jugadores.



5. Proponer y resolver ocho consultas en Neo4j en las que aparezcan, al menos, las siguientes cláusulas CQL: MATCH, WHERE, RETURN, ORDER BY, REMOVE, SET, DISTINCT, COUNT, LIMIT, SKIP, DELETE.

Consulta 1: Consultar los jugadores que juegan en el equipo “Boston Celtics”.

`MATCH (j:Jugador)-[juega_en]->(e:Equipo) WHERE e.nombre="Boston Celtics" RETURN j.nombre`

The screenshot shows the Neo4j browser interface with the following details:

- Code Tab:** Contains the Cypher query:

```
$ MATCH (j:Jugador)-[juega_en]->(e:Equipo) WHERE e.nombre="Boston Celtics" RETURN j.nombre AS Jugadores_Boston_Celtics
```
- Table Tab:** Contains the results of the query:

Jugadores_Boston_Celtics
"Horford"
"Tatum"
"Hayward"
"Brown"
"Irving"
- Text Tab:** Shows the message: "Started streaming 5 records after 1 ms and completed after 1 ms."

Consulta 2: Modificar el nombre del partido “Celtics-Cavs”.

`MATCH (p:Partido) WHERE p.id = 1 SET p.nombre = "Boston-Cleveland" RETURN p`

The screenshot shows the Neo4j browser interface with the following details:

- Query Bar:** Displays the Cypher query: `$ MATCH (p:Partido) WHERE p.id = 1 SET p.nombre = "Boston-Cleveland" RETURN p`.
- Result Table:** Shows a single row with the following data:

p
{ "nombre": "Boston-Cleveland", "tipo": "Regular", "id": 1 }
- Left Sidebar:** Contains tabs for Graph, Table (selected), Text, and Code.
- Status Bar:** At the bottom, it says "Set 1 property, started streaming 1 records after 15 ms and completed after 16 ms."

Consulta 3: Consultar las canastas ocurridas en el partido Boston-Cleveland, y ordenarlas de forma descendente por el minuto del partido en el que ocurrieron.

```
MATCH (c:Canasta)-[ocurrio_en]->(p:Partido) WHERE p.nombre="Boston-Cleveland"
RETURN c.valor AS Valor_Canasta,c.minuto AS Minuto_Canasta ORDER BY c.minuto DESC
```

The screenshot shows a Neo4j browser window. On the left, there is a sidebar with three buttons: 'Table' (selected), 'Text', and 'Code'. The main area displays a query result table. The table has two columns: 'Valor_Canasta' and 'Minuto_Canasta'. The data is sorted by 'Minuto_Canasta' in descending order. The table contains 15 rows of data.

	Valor_Canasta	Minuto_Canasta
	1	48
	2	45
	2	41
	2	35
	3	32
	3	31
	3	29
	0	28
	2	28
	3	25
	3	24
	3	21
	0	19
	0	17
	2	12
	2	11

Consulta 4: Consultar las canastas anotadas por el jugador Irving, viendo su valor y el minuto en el que la anotó. No se tendrán en cuenta las canastas con valor = 0, que son fallos.

```
MATCH (j:Jugador)-[anota]->(c:Canasta) WHERE j.nombre = "Irving" AND c.valor <> 0  
RETURN c.valor AS Valor_Canasta, c.minuto AS Minuto_Canasta ORDER BY  
Minuto_Canasta ASC
```

```
$ MATCH (j:Jugador)-[anota]->(c:Canasta) WHERE j.nombre = "Irving" AND c..
```

	Valor_Canasta	Minuto_Canasta
Table	2	28
A	3	29
Text	3	31
</>	1	48
Code		

Consulta 5: Consultar los nombres de los jugadores que recibieron las asistencias de Irving, sin repetir aunque haya más de una asistencia para el mismo compañero.

```
MATCH (j:Jugador)-[asiste_a]->(j2:Jugador) WHERE j.nombre = "Irving" RETURN DISTINCT  
j2.nombre AS Compañero_Asistido
```

```
$ MATCH (j:Jugador)-[asiste_a]->(j2:Jugador) WHERE j.nombre = "Irving" R..
```

	Compañero_Asistido
Table	"Brown"
A	"Tatum"
Text	"Hayward"
</>	
Code	

Consulta 6: Consultar los puntos anotados por los Boston Celtics.

```
MATCH (c:Canasta)-[pertenece_a]->(e:Equipo) WHERE e.nombre = "Boston Celtics"  
RETURN SUM(c.valor) AS Anotación_Boston_Celtics
```

```
$ MATCH (c:Canasta)-[pertenece_a]->(e:Equipo) WHERE e.nombre = "Boston  
Celtics" RETURN SUM(c.valor) AS Anotación_Boston_Celtics
```

```
$ MATCH (c:Canasta)-[pertenece_a]->(e:Equipo) WHERE e.nombre = "Boston C..
```



Table



Text



Code

Anotación_Boston_Celtics

21

Podemos utilizar la propiedad UNION para obtener el resultado del partido Boston Celtics vs Cleveland Cavaliers.

```
MATCH (c:Canasta)-[pertenece_a]->(e:Equipo) WHERE e.nombre = "Boston Celtics"
RETURN e.nombre AS Equipo, SUM(c.valor) AS Anotación UNION
MATCH (c:Canasta)-[pertenece_a]->(e:Equipo) WHERE e.nombre = "Cleveland Cavaliers" RETURN e.nombre
AS Equipo, SUM(c.valor) AS Anotación
```

```
$ MATCH (c:Canasta)-[pertenece_a]->(e:Equipo) WHERE e.nombre = "Boston
Celtics" RETURN e.nombre AS Equipo, SUM(c.valor) AS Anotación UNION
MATCH (c:Canasta)-[pertenece_a]->(e:Equipo) WHERE e.nombre =
"Cleveland Cavaliers" RETURN e.nombre AS Equipo, SUM(c.valor) AS
Anotación
```

```
$ MATCH (c:Canasta)-[pertenece_a]->(e:Equipo) WHERE e.nombre = "Boston C..."
```



Table



Text



Code

Equipo	Anotación
"Boston Celtics"	21
"Cleveland Cavaliers"	17

Consulta 7: Obtener la primera canasta anotada en el partido Boston-Cleveland.

```
MATCH (c:Canasta)-[ocurrio_en]->(p:Partido) WHERE p.nombre="Boston-Cleveland" AND c.valor <> 0 RETURN c.valor AS Valor_Canasta,c.minuto AS Minuto_Canasta ORDER BY c.minuto ASC LIMIT 1
```

```
$ MATCH (c:Canasta)-[ocurrio_en]->(p:Partido) WHERE p.nombre="Boston-Cleveland" AND c.valor <> 0 RETURN c.valor AS Valor_Canasta,c.minuto AS Minuto_Canasta ORDER BY c.minuto ASC LIMIT 1
```

```
$ MATCH (c:Canasta)-[ocurrio_en]->(p:Partido) WHERE p.nombre="Boston-Cle..."
```

Valor_Canasta	Minuto_Canasta
1	4

Table

A
Text

</>
Code

Con el uso de SKIP podemos al contrario obtener todas las canastas del partido y el minuto en el que ocurrieron pero sin tener en cuenta la primera canasta.

```
MATCH (c:Canasta)-[ocurrio_en]->(p:Partido) WHERE p.nombre="Boston-Cleveland" AND c.valor <> 0 RETURN c.valor AS Valor_Canasta,c.minuto AS Minuto_Canasta ORDER BY c.minuto ASC SKIP 1
```

```
$ MATCH (c:Canasta)-[ocurrio_en]->(p:Partido) WHERE p.nombre="Boston-Cleveland" AND c.valor <> 0 RETURN c.valor AS Valor_Canasta,c.minuto AS Minuto_Canasta ORDER BY c.minuto ASC SKIP 1
```

```
$ MATCH (c:Canasta)-[ocurrio_en]->(p:Partido) WHERE p.nombre="Boston-Cle..."
```



Table



Text



Code

Valor_Canasta	Minuto_Canasta
3	6
3	10
2	11
2	12
3	21
3	24
3	25
2	28
3	29
3	31
3	32
2	35
2	41
2	45
1	48

Consulta 8: Con Remove se pueden eliminar propiedades de nodos. Se va a eliminar la propiedad “tipo” del nodo “partido” ya que no se va a diferenciar entre partidos de la liga regular y partidos de los playoffs.

MATCH (p:Partido) REMOVE p.tipo RETURN p AS Partidos

The screenshot shows a Neo4j browser window. On the left, there is a sidebar with four tabs: Graph (selected), Table, Text, and Code. The main area has a title bar with the query: '\$ MATCH (p:Partido) REMOVE p.tipo RETURN p AS Partidos'. Below the title bar, there are three icons: a download arrow, a magnifying glass, and a checkmark. The main content area is titled 'Partidos' and contains two JSON objects. The first object is:

```
{  
    "nombre": "Boston-Cleveland",  
    "id": 1  
}
```

The second object is:

```
{  
    "nombre": "Warriors-Rockets",  
    "id": 2  
}
```

La canasta anotada por el jugador Thompson fue revisada y finalmente no fue anotada por él, así que se hace uso de DELETE para eliminar dicha relación.

MATCH (j:Jugador{nombre:'Thompson'})-[rel:anota]-(c:Canasta) DELETE rel

```
$ MATCH (j:Jugador{nombre: 'Thompson'})-[rel:anota]-(c:Canasta) DELETE rel
```

```
$ MATCH (j:Jugador{nombre: 'Thompson'})-[rel:anota]-(c:Canasta) DELETE rel
```



Deleted 1 relationship, completed after 10 ms.



Code

6. Proponer y resolver dos consultas de pattern matching.

- Consultar los jugadores que han participado en el partido Boston-Cleveland.

```
MATCH (j:Jugador)-[:juega_en]-(e:Equipo)-[:participa_en]-(p:Partido) WHERE p.nombre="Boston-Cleveland" RETURN DISTINCT j.nombre AS Jugadores, e.nombre AS Equipo
```

The screenshot shows a Neo4j browser window with the following details:

- Code Tab:** Contains the Cypher query: `$ MATCH (j:Jugador)-[:juega_en]-(e:Equipo)-[:participa_en]-(p:Partido) WHERE p.nombre="Boston-Cleveland" RETURN DISTINCT j.nombre AS Jugadores, e.nombre AS Equipo`.
- Table Tab:** Active tab, showing the results of the query in a table format.
- Text Tab:** Shows the raw query text.
- Code Tab:** Shows the raw query text.
- Table:** Displays the results of the query:

Jugadores	Equipo
"Horford"	"Boston Celtics"
"Irving"	"Boston Celtics"
"Tatum"	"Boston Celtics"
"Wade"	"Cleveland Cavaliers"
"Thomas"	"Cleveland Cavaliers"
"Brown"	"Boston Celtics"
"Hayward"	"Boston Celtics"
"James"	"Cleveland Cavaliers"
"Crowder"	"Cleveland Cavaliers"
"Love"	"Cleveland Cavaliers"

- Obtener todas las canastas de los Boston Celtics, mostrando el jugador que la ha anotado, el valor de la canasta y el minuto en el que fue anotada.

```
MATCH (j:Jugador)-[:anota]-(c:Canasta)-[:pertenece_a]-(e:Equipo) WHERE e.nombre="Boston Celtics" AND c.valor <> 0 RETURN j.nombre AS Jugador, c.valor AS Puntuación, c.minuto AS minuto ORDER BY minuto ASC
```

```
$ MATCH (j:Jugador)-[:anota]-(c:Canasta)-[:pertenece_a]-(e:Equipo)
WHERE e.nombre="Boston Celtics" AND c.valor <> 0 RETURN j.nombre AS Jugador, c.valor AS Puntuación, c.minuto AS minuto ORDER BY minuto ASC
```

\$ MATCH (j:Jugador)-[:anota]-(c:Canasta)-[:pertenece_a]-(e:Equipo) WHERE...

Table
Text
Code

"Jugador"	"Puntuación"	"minuto"
"Brown"	1	4
"Hayward"	3	6
"Horford"	3	10
"Tatum"	3	21
"Irving"	2	28
"Irving"	3	29
"Irving"	3	31
"Hayward"	2	45
"Irving"	1	48