UNIVERSIDAD TECNICA FEDERICO SANTA MARIA
DEPARTAMENTO DE INFORMÁTICA

INF-325 Bases de Datos Avanzadas: Laboratorio 3 - Neo4J

Sofía Carrasco
sofia.carrascoc@sansano.usm.cl

Martín Menares
martin.menares@sansano.usm.cl

Jorge Salas
jorge.salasl@sansano.usm.cl

Claudio Vergara
claudio.vergaramo@sansano.usm.cl

Javiera Villarroel
javiera.villarroel@sansano.usm.cl

Viernes, 17 de Diciembre de 2021

Resumen

En este documento se trabaja con el motor de bases de datos basados en grafos Neo4j para analizar y extraer información de una base de datos histórica que recopila datos de todos los juegos olímpicos que se han realizado desde la edición Atenas 1896 hasta Río 2016. Para trabajar la base de datos se utilizó la GUI que tiene Neo4j, denominada Neo4j Desktop que es ideal para poder visualizar y trabajar de forma más sencilla cada una de las tareas requeridas como la construcción misma de la base de datos así como también las consultas en si. Debido a que, como generalmente ocurre, los dataset vienen con datos erróneos, con mal formato o desestandarizado, se lleva a cabo una limpieza y transformación de datos para poder obtener resultados más consistentes. Además, debido a la gran cantidad de datos que posee el dataset original, el equipo se ve en la necesidad de muestrear dicho dataset por medio de la plataforma Google Collaboratory. Dado que las consultas son originales del equipo, los resultados y los aprendizajes son bastante significativos. Como que un deportista a sus 64 años de edad fue el deportista más longevo en obtener una medalla de oro, que los deportistas con menor peso que han pasado por los juegos olímpicos poseen solo 30 kilogramos, entre otros. Finalmente, como equipo, se aprendió que este tipo de bases de datos tienen un gran potencial y es el que están aprovechando diferentes industrias del mundo actualmente, como lo pueden ser eBay, Walmart [1], Cisco, Accenture, entre muchos otros [4].

1 Introducción

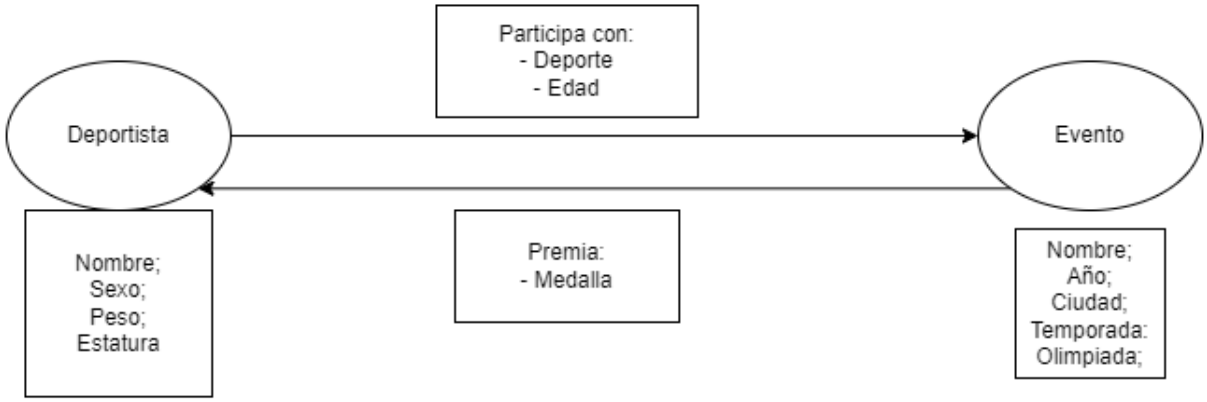
Los juegos olímpicos son reconocidos como la mayor fiesta mundial del deporte multidisciplinario, en el cual han participado diversos deportistas provenientes de un variado abanico de países. Según Kristine Toohey [1], es tal la envergadura de este evento, que trasciende el mero ámbito deportivo. Es un evento que llama la atención del mundo audiovisual, es un potente apoyo al turismo de la ciudad en la que se lleva a cabo, es una fuente de inspiración para los jóvenes. Por lo tanto, los juegos olímpicos, aparte de relacionarse netamente con el mundo deportivo, deja huella en el mundo cultural, político y económico mundial. Al desglosar este enorme evento deportivo uno puede percatarse de que estos juegos olímpicos se llevan a cabo en dos diversas temporadas o estaciones del año (son realizados con un intervalo de dos años entre sí): los Juegos Olímpicos de Verano y los Juegos Olímpicos de Invierno. Dentro de este contexto, se nos entrega un dataset provisto por la plataforma Kaggle, el cual contiene datos biográficos básicos de los atletas participantes y de los resultados que estos han adquirido en términos de medallas desde la primera edición de los Juegos Olímpicos realizada en Atenas el año 1896 hasta la última edición, que se ha podido realizar en territorio brasileño, en Río de Janeiro, el año 2016. Esto es debido a que por causa de la pandemia del Coronavirus, los Juegos Olímpicos de Tokyo del 2020 tuvieron que ser pospuestos para el año en curso, por lo que no se cuenta con dicha información más actualizada.

Como se mencionó anteriormente, en este dataset encontraremos las diversas participaciones de todos los deportistas, incluyendo el equipo y país al que representan, así como también en qué años y temporadas han participado.

Por otro lado, en esta ocasión habrá una cierta lejanía a lo que uno se encuentra generalmente acostumbrado a ver cuando trabaja con bases de datos, que son las tablas, ya que el motor de bases de datos con el que se trabaja a continuación está basado en grafos. A pesar de que es un cambio considerable en el paradigma clásico, este tipo de bases de datos son aplicables a diversos ámbitos [2] como la química, la biología, la web semántica, las redes sociales o, también los sistemas recomendadores. Dentro de la emergente gama de motores de bases de datos basadas en grafos, se trabajará con el motor Neo4j, una base de datos de grafos *open source* que está en el mercado desde el año 2003 [3]. Volviendo por unos instantes al mundo relacional, sabemos que motores como Oracle, Postgres, entre otros; son manejados por el reconocido lenguaje SQL. Para Neo4j también existe un lenguaje que está integrado dentro del motor, el cual se llama Cypher, y es el lenguaje que se emplea tanto en la construcción de la base de datos (2.2) como en el desarrollo de las consultas (2.4).

2. Desarrollo

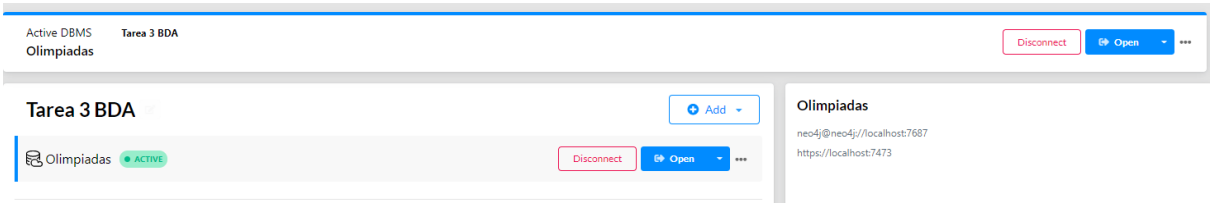
2.1 Diagrama de grafo



Como se ve en el grafo presentado, existe la relación de “Un deportista participa (con ciertas circunstancias) de un evento” y que “El evento premia al deportista”. Para la entidad Deportista se le asocian los atributos que, generalmente, permanecen constantes, tales como el nombre, el sexo, el peso y la estatura. Sin embargo, a pesar que la edad también es del deportista, esta edad va cambiando y en cada nueva ceremonia, el deportista tiene una edad diferente, por lo que es un atributo de la relación misma. De igual forma, este deportista puede ser multifacético y tanto practicar como participar en más de un deporte por año.

2.2 Construir base de datos

Se crea una instancia localmente en Docker, donde fue necesario especificar las carpetas donde se encontrarán los datos (archivos csv) para luego realizar la conexión por localhost mediante Neo4j:



Para agilizar las consultas y la carga de los datos se utiliza la creación de índices, de forma simple por los deportistas mediante su ID, e índices compuestos para los eventos, pues cada evento en particular se realiza en un año, temporada y ciudad específica.

```
1  create index on:Deportista(ID);
2  create index on:Evento(year, season, city, event)
```

Los datos de los deportistas, eventos y relaciones se separaron en archivos CSV distintos, como se presentará en la siguiente sección. A continuación, se cargan los datos de los nodos de deportistas mediante el comando LOAD CSV , utilizando los atributos nombre, sexo, altura y peso, y se hace merge mediante su ID aprovechando el índice creado anteriormente. Se transforman a float las columnas de altura y peso debido a que Neo4j trata los valores de CSV como una cadena (string). Esta conversión es posible debido a que se reemplazaron datos vacíos con 0.

```
load csv with headers from "file:///df_deportistas.csv" as row
merge(d:Deportista{ID:row.ID})
on create set
d.nombre = row.Name,
d.sexo = row.Sex,
d.altura = tofloat(row.Height),
d.peso = tofloat(row.Weight);
```

Added 135571 labels, created 135571 nodes, set 677855 properties, completed after 2319 ms.

Table Code

Luego, se cargan los datos de los eventos, realizando un merge identificando los eventos por los atributos mencionados anteriormente en el índice compuesto, y además, agregando el atributo del nombre de la olimpiada en la que ocurrió ese evento.

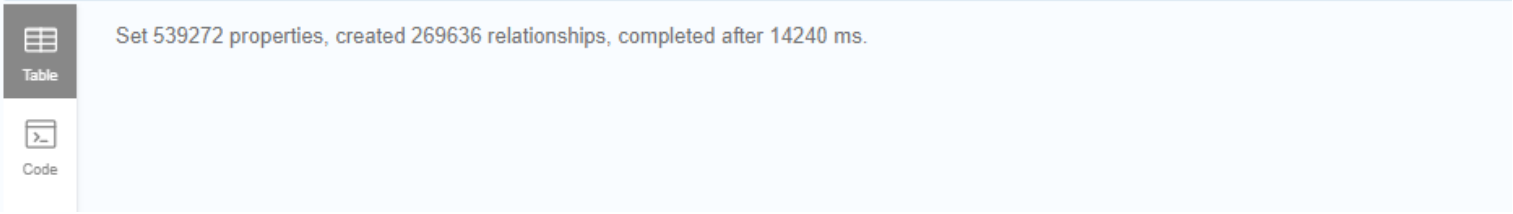
```
load csv with headers from "file:///df_eventos.csv" as row
merge(e:Evento{year: row.Year, season: row.Season, city: row.City, event: row.Event})
on create set
e.Olimpiada = row.Games;
```

Added 6192 labels, created 6192 nodes, set 24768 properties, completed after 3723 ms.

Table Code

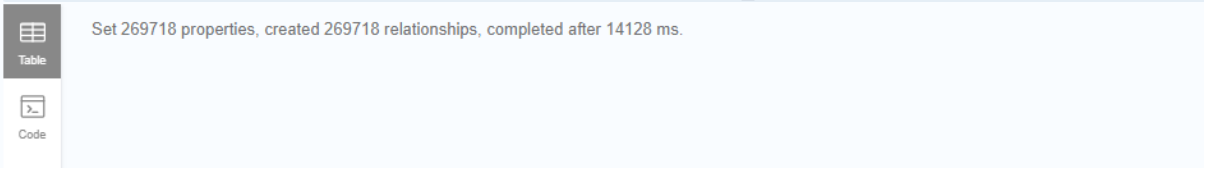
Se cargan los datos para las relaciones, creando la relación “deportista *participa_en*{deporte, edad} evento”. Se utiliza match para encontrar los nodos deportista y evento ya creados para unirlos mediante la relación.

```
load csv with headers from "file:///df_relacion.csv" as row
match(d:Deportista{ID:row.ID})
match(e:Evento{year: row.Year, season: row.Season, city: row.City, event: row.Event})
merge (d) -[r:participa_en{deporte: row.Sport, edad: row.Age}]→ (e);
```



También, se cargan los datos creando la relación “evento *premia_con*{medalla} a deportista”, puesto que existen distintos eventos en las olimpiadas, donde los deportistas pueden recibir distintas medallas en cada evento.

```
load csv with headers from "file:///df_relacion.csv" as row
match(d:Deportista{ID:row.ID})
match(e:Evento{year: row.Year, season: row.Season, city: row.City, event: row.Event})
merge (e) -[r:premia_con{medalla: row.Medal}]→ (d);
```



2.3 Procesar y depurar los datos

Primeramente, se limpiaron las comillas de cada una de las celdas en excel. Además los valores NA fueron reemplazados por 0 para evitar problemas al momento de cargar los datos.

ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
1	A Dijiang	M	24	180	80	China	CHN	1992 Sumı	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	0
2	A Lamusi	M	23	170	60	China	CHN	2012 Sumı	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	0
3	Gunnar Ni	M	24	0	0	Denmark	DEN	1920 Sumı	1920	Summer	Antwerpe	Football	Football Men's Football	0
4	Edgar Lind	M	34	0	0	Denmark/	DEN	1900 Sumı	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
5	Christine .	F	21	185	82	Netherlan	NED	1988 Wint	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	0
5	Christine .	F	21	185	82	Netherlan	NED	1988 Wint	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 1,000 metres	0
5	Christine .	F	25	185	82	Netherlan	NED	1992 Wint	1992	Winter	Albertvill	Speed Skating	Speed Skating Women's 500 metres	0
5	Christine .	F	25	185	82	Netherlan	NED	1992 Wint	1992	Winter	Albertvill	Speed Skating	Speed Skating Women's 1,000 metres	0
5	Christine .	F	27	185	82	Netherlan	NED	1994 Wint	1994	Winter	Lillehamn	Speed Skating	Speed Skating Women's 500 metres	0
5	Christine .	F	27	185	82	Netherlan	NED	1994 Wint	1994	Winter	Lillehamn	Speed Skating	Speed Skating Women's 1,000 metres	0
6	Per Knut A	M	31	188	75	United Sta	USA	1992 Wint	1992	Winter	Albertvill	Cross Country Skiing	Cross Country Skiing Men's 10 kilometres	0
6	Per Knut A	M	31	188	75	United Sta	USA	1992 Wint	1992	Winter	Albertvill	Cross Country Skiing	Cross Country Skiing Men's 50 kilometres	0
6	Per Knut A	M	31	188	75	United Sta	USA	1992 Wint	1992	Winter	Albertvill	Cross Country Skiing	Cross Country Skiing Men's 10/15 kilometres Pu	0
6	Per Knut A	M	31	188	75	United Sta	USA	1992 Wint	1992	Winter	Albertvill	Cross Country Skiing	Cross Country Skiing Men's 4 x 10 kilometres Re	0
6	Per Knut A	M	33	188	75	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Country Skiing	Cross Country Skiing Men's 10 kilometres	0
6	Per Knut A	M	33	188	75	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Country Skiing	Cross Country Skiing Men's 30 kilometres	0
6	Per Knut A	M	33	188	75	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Country Skiing	Cross Country Skiing Men's 10/15 kilometres Pu	0
6	Per Knut A	M	33	188	75	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Country Skiing	Cross Country Skiing Men's 4 x 10 kilometres Re	0
7	John Aalb	M	31	183	72	United Sta	USA	1992 Wint	1992	Winter	Albertvill	Cross Country Skiing	Cross Country Skiing Men's 10 kilometres	0
7	John Aalb	M	31	183	72	United Sta	USA	1992 Wint	1992	Winter	Albertvill	Cross Country Skiing	Cross Country Skiing Men's 50 kilometres	0
7	John Aalb	M	31	183	72	United Sta	USA	1992 Wint	1992	Winter	Albertvill	Cross Country Skiing	Cross Country Skiing Men's 10/15 kilometres Pu	0
7	John Aalb	M	31	183	72	United Sta	USA	1992 Wint	1992	Winter	Albertvill	Cross Country Skiing	Cross Country Skiing Men's 4 x 10 kilometres Re	0
7	John Aalb	M	33	183	72	United Sta	USA	1994 Wint	1994	Winter	Lillehamn	Cross Country Skiing	Cross Country Skiing Men's 10 kilometres	0

En el procesamiento y depuración de datos, se toma la decisión de dividir el archivo en 3 partes, debido a que el archivo es demasiado grande para ser tratado directamente con Neo4j. A continuación, se muestra el flujo de trabajo descrito implementado en la herramienta Google Collaboratory utilizando Python y la librería pandas. En este procesamiento de los datos, se crean los datasets para deportista y eventos, eliminando los datos duplicados, de forma que se tenga cada deportista y cada evento una vez en sus CSV correspondientes.

```
[5] df_deportistas=df[{'ID', 'Name', 'Sex', 'Weight', 'Height'}]
```

```
[6] df_deportistas.head()
```

	Sex	Height	Name	ID	Weight
0	M	180.0	A Dijiang	1	80.0
1	M	170.0	A Lamusi	2	60.0
2	M	0.0	Gunnar Nielsen Aaby	3	0.0
3	M	0.0	Edgar Lindenau Aabye	4	0.0
4	F	185.0	Christine Jacoba Aaftink	5	82.0

```
[8] df_deportistas = df_deportistas.drop_duplicates()
```

```
[11] df_deportistas
```

```
[10] df_deportistas.dtypes
```

```
[12] df_eventos = df[{'Year', 'Season', 'City', 'Event'}]
```

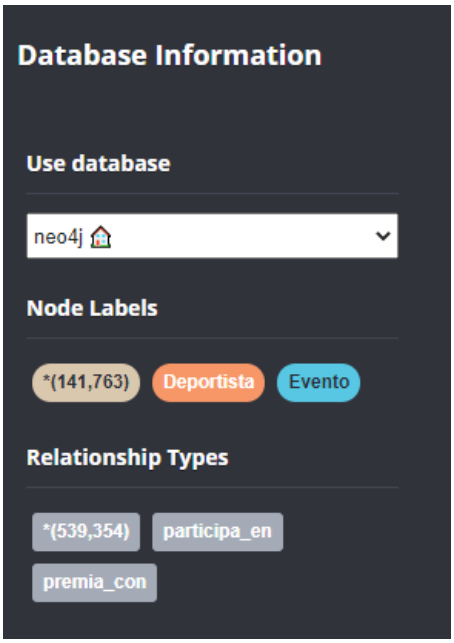
```
[13] df_eventos = df_eventos.drop_duplicates()
```

```
[14] df_eventos
```

```
[15] df_relacion_dep_ev = df[{'ID', 'Age','Sport', 'Medal','Event', 'Season', 'City', 'Year', 'Event', 'Games'}]
```

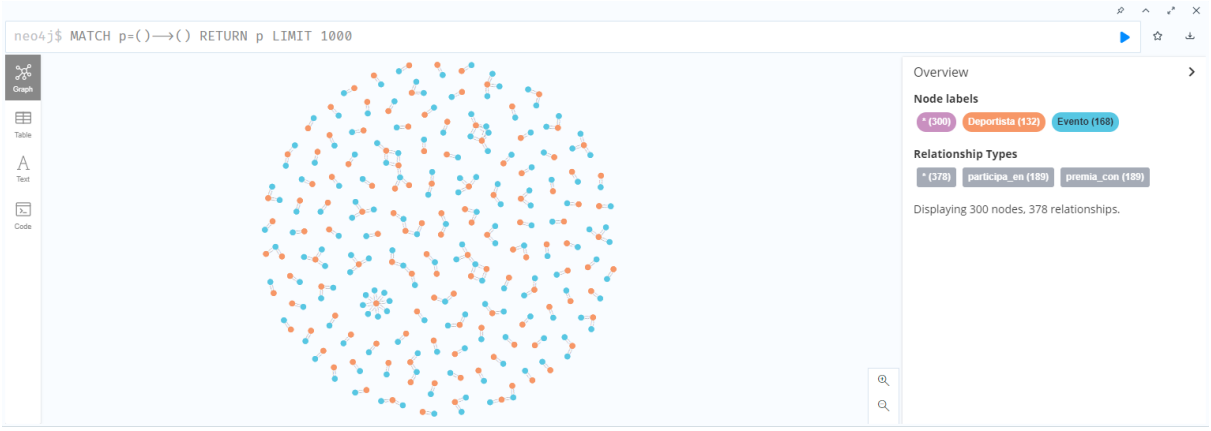
```
[16] df_relacion_dep_ev = df_relacion_dep_ev.drop_duplicates()
```

En la base de datos, se tendrán dos nodos principales, los que serán “Deportista” y “Evento”. Esto indica que el modelo está hecho en base al diagrama propuesto de la base, notando como se crean las relaciones *“participa_en”* y *“premia_con”*, siendo en total 539354 relaciones y 141763 nodos.



2.4 Desarrollo de consultas

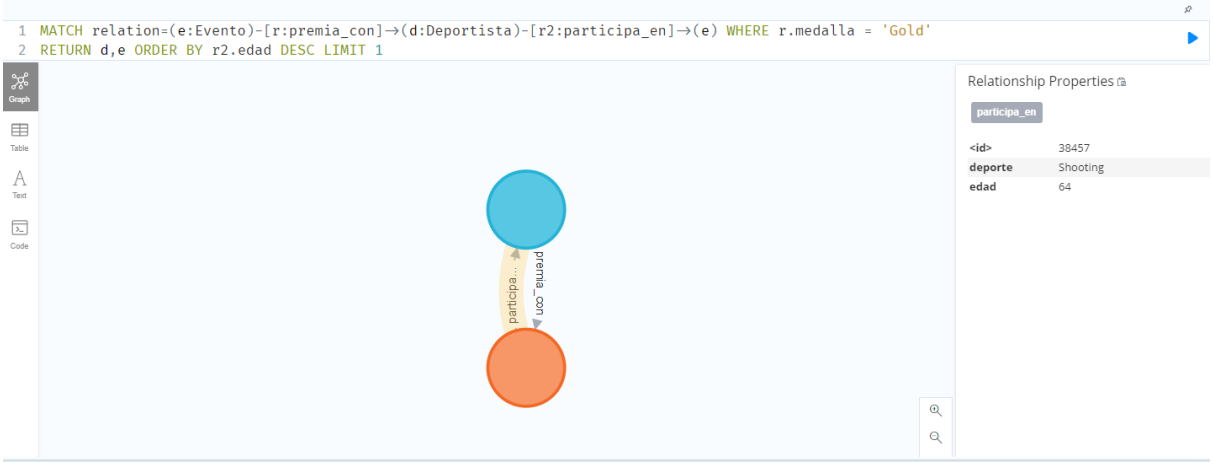
Grafo generado en Neo4J Desktop, luego de cargar los datos, donde la visualización del grafo se limita a 1000 debido a la gran cantidad de datos.



Consultas:

- El deportista de mayor edad en llevarse una medalla de oro:

```
MATCH relation=(e:Evento)-[r:premia_con]→(d:Deportista)-[r2:participa_en]→(e) WHERE r.medalla = 'Gold'
RETURN d,e ORDER BY r2.edad DESC LIMIT 1
```



En esta consulta, se obtiene que fue el deportista que participó con el deporte “Shooting” y con edad de 64 años, obteniendo una medalla de oro. Obteniendo más información del nodo, el nombre del deportista es Oscar Gomer Swahn.

Node Properties

Deportista

<id>	211766
ID	117046
altura	0.0
nombre	Oscar Gomer Swahn
peso	0.0
sexo	M

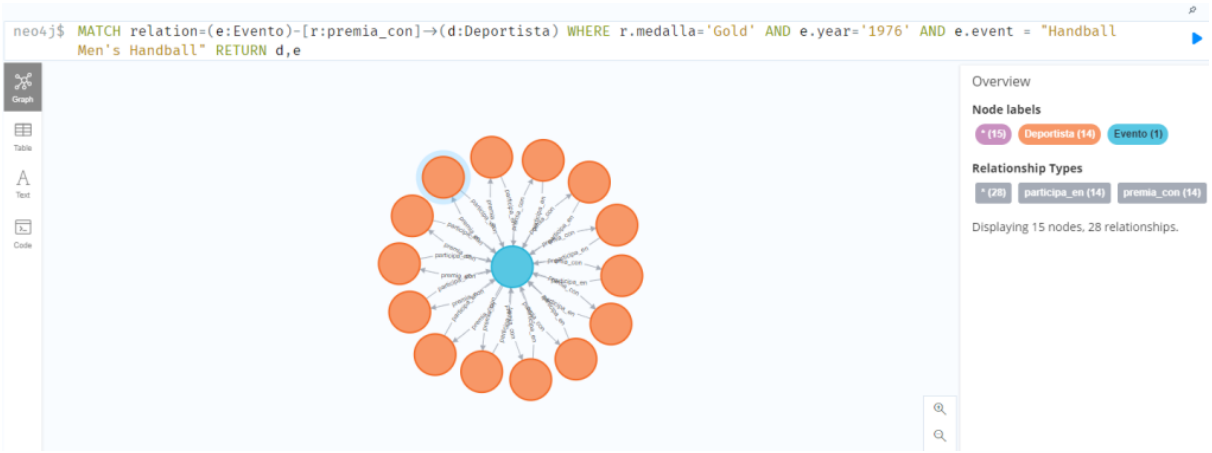
- Cuántas personas en total participaron en natación en las olimpiadas del 2004:

```
neo4j$ MATCH relation=(d:Deportista)-[r:participa_en]→(e:Evento) WHERE e.year ='2004' AND r.deporte='Swimming' RETURN count(distinct
d)
```

count(distinct d)
937

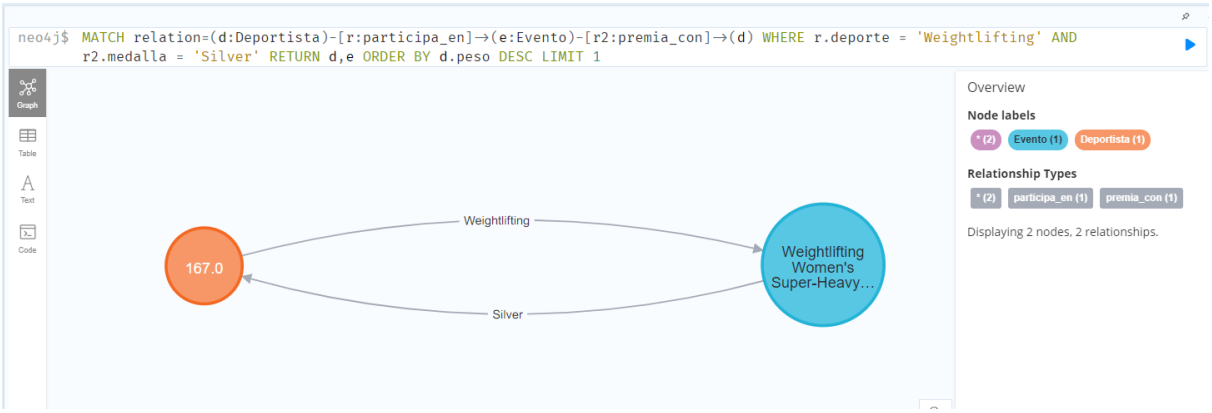
Para obtener la cantidad de deportistas que participaron en natación en las olimpiadas del 2004, se utilizó la función de agregación *count*, y debido a que un deportista puede participar en varios eventos de natación en una olimpiada, se utiliza *distinct* para contarlos solo una vez. De esta forma, se obtiene que participaron 937 deportistas.

- Los deportistas que ganaron oro en el evento “Handball Men's Handball” de 1976:

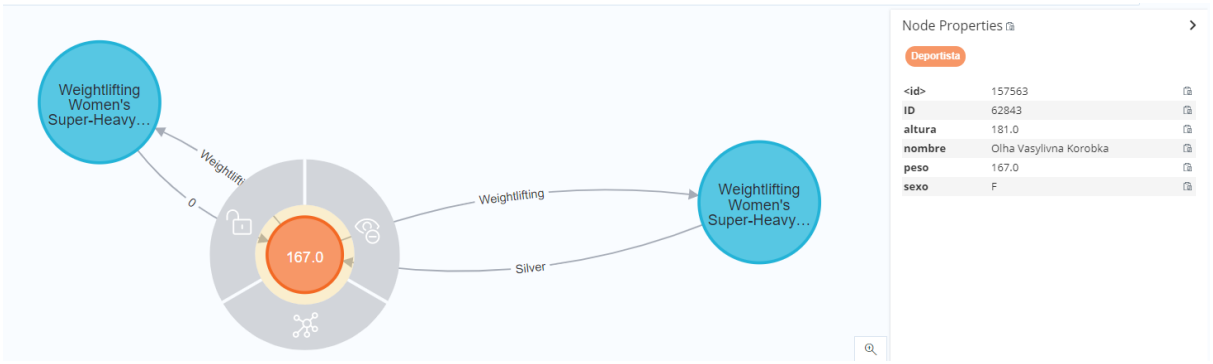


A pesar de que en nuestro modelo no se guardó a qué equipo pertenece cada deportista, se puede realizar esta consulta debido a que Handball es un deporte en equipo, aquellos deportistas que reciban oro serán parte del equipo ganador en un evento específico. Se puede notar que se obtienen 14 deportistas e investigando sobre el Handball, se averigua que en cada equipo participan 7 deportistas, y pueden contar con otros 7 jugadores.

- El deportista con más peso en obtener una medalla de plata en Weightlifting:

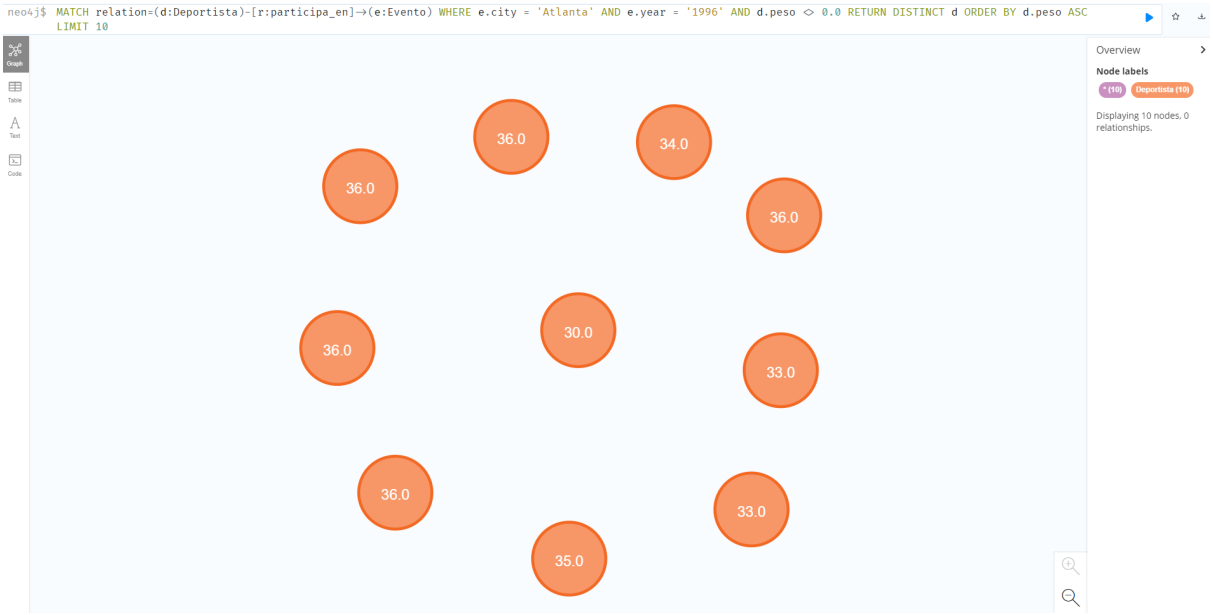


Se obtiene que el deportista con más peso en obtener una medalla de plata con el deporte Weightlifting pesa 167 kilogramos. Se puede notar que el evento corresponde a “Weightlifting Women’s Super-Heavy…”, y viendo más información sobre el nodo deportista encontrado, se descubrió que la persona retornada es mujer. Además, al seleccionar el nodo y la opción de ver sus otras relaciones, se nota lo siguiente:



Donde participó en el mismo evento en el año 2004 con 18 años, donde no ganó ninguna medalla, sin embargo, en 2008 con 22 logró mejorar y ganar la medalla de plata.

- Los 10 deportistas de menor peso que han participado en las olimpiadas de Atlanta de 1996:

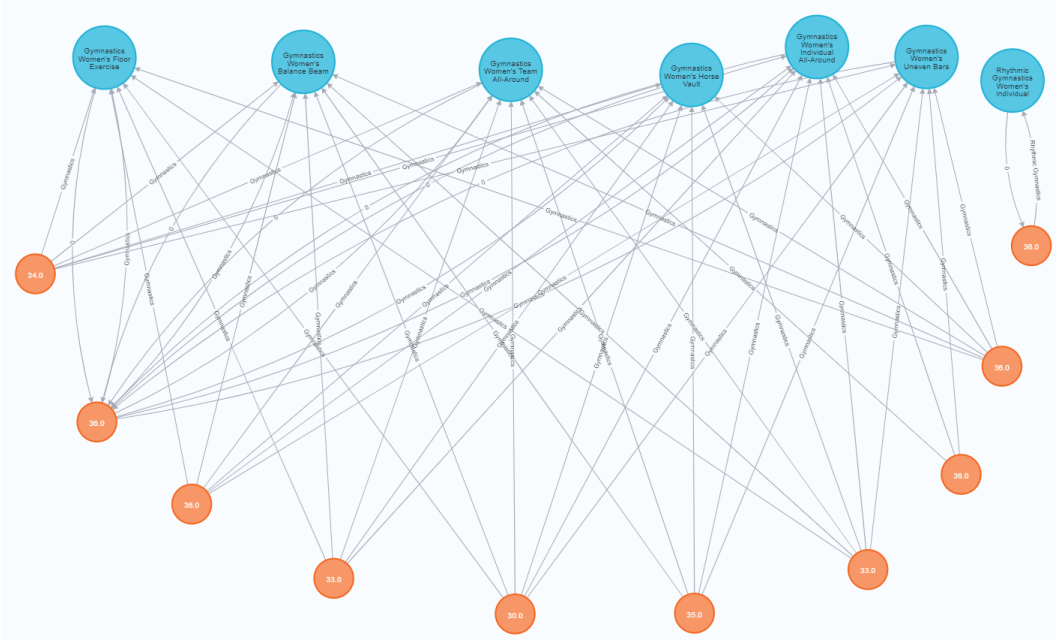


En este caso, se mostró el grafo de los deportistas encontrados con menor peso, siendo el menor peso registrado de 30 kilogramos. También se utilizó distinct debido a que si no, podría retornar a la misma deportista que participó en varios eventos de las olimpiadas de Atlanta en 1996. A continuación, se muestra en formato texto para obtener más información sobre los deportistas.

"d"
{"peso":30.0,"altura":136.0,"ID":"109593","sexo":"F","nombre":"Liubov Sheremeta"}
{"peso":33.0,"altura":138.0,"ID":"82189","sexo":"F","nombre":"Lisa Moro"}
{"peso":33.0,"altura":144.0,"ID":"72589","sexo":"F","nombre":"Oksana Vasilyevna Lyapina"}
{"peso":34.0,"altura":139.0,"ID":"80754","sexo":"F","nombre":"Dominique Helena Moceanu (-Canales)"}
{"peso":35.0,"altura":142.0,"ID":"11223","sexo":"F","nombre":"Bi Wenjing"}
{"peso":36.0,"altura":161.0,"ID":"18240","sexo":"F","nombre":"Alba Carride Costas"}
{"peso":36.0,"altura":142.0,"ID":"122229","sexo":"F","nombre":"Vasiliki Tsavdaridou"}
{"peso":36.0,"altura":144.0,"ID":"50131","sexo":"F","nombre":"Naho Hoshiyama"}
{"peso":36.0,"altura":145.0,"ID":"74677","sexo":"F","nombre":"Mao Yanling"}
{"peso":36.0,"altura":148.0,"ID":"92437","sexo":"F","nombre":"Gemma Paz Ortega"}

Al obtener las relaciones de las deportistas utilizando la opción mencionada anteriormente, se obtiene el siguiente grafo, donde se puede notar que en este evento, todas las deportistas participaron en eventos relacionados a Gimnasia.

```
MATCH relation=(d:Deportista)-[r:participa_en]→(e:Evento) WHERE e.city = 'Atlanta' AND e.year = '1996' AND d.peso <> 0.0
RETURN DISTINCT d ORDER BY d.peso ASC LIMIT 10
```



3 Conclusiones

Según lo discutido en clases, pudimos comprobar el modelo de datos que tiene el gestor Neo4j, el cual corresponde a una base de datos basada en grafos. Esto puede ser considerado una gran ventaja, debido a que es una estructura naturalmente entendible ya que cada dato es un nodo y están relacionados por aristas. Además, la interfaz utilizada permite interactuar con el grafo, con distintas opciones sobre qué atributo mostrar en la visualización de los nodos o relaciones, como también permite mostrar las relaciones que salen y/o entran de un nodo en específico, como también ocultarlo en la visualización/respuesta de la consulta. Otro factor importante a considerar cuando se trabaja con Neo4j es que este motor no tiene buen rendimiento cuando se trabaja con volúmenes considerables de datos, ya que en un principio, con los más de 270.000 registros del csv, Neo4j fue simplemente incapaz de procesar todos los datos, se perdieron bastantes horas esperando alguna respuesta o reacción del motor de bases de datos y el hardware de los computadores se vieron bastante sobreexigidos. De esa forma, podemos concluir que es necesario hacer muestras de datos representativas para poder trabajar en Neo4j, además de aprovechar el uso de índices.

4 Referencias Bibliográficas

[1] C. Kemper, Beginning Neo4j, Apress, 2015.

[2] K. Toohey & A. Veal, The Olympic Games: A social science perspective, Cabi, 2007.

[3] J. Miller, Graph Database Applications and Concepts with Neo4j, Georgia Southern University., 2013.

[4] M. Lal, Neo4j graph data modelling, Packt Publishing Ltd, 2015. .