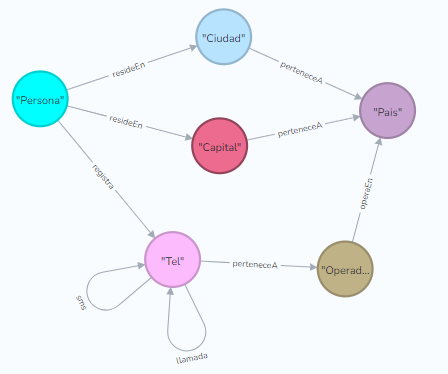
## *Trabajo Práctico Nro 1: Neo4*

Dado el grafo telco.db que representa información de Personas, comunicaciones telefónicas (sms/llamada), Operadores e información espacial (Paises y Ciudades)



Donde

**Persona** “residenEn” **Ciudad o Capital**

Ambas (ciudad/capital) “perteneceA” **Pais**.

Persona “registra” Tel.

Los **Tel** se comunican entre sí por “sms” o “llamada”

Cada Tel “perteneceA” **Operador**

Un Operador “operaEn” Pais.

Resolver las siguientes consultas.

***Ejercicio 0***

Antes de comenzar, reparar el grafo porque la propiedad Idiomas de un Pais está en formato string, pero debería estar en formato Arreglo De Strings, ya que debe contener una colección con los idiomas que se habla oficialmente en un Pais. Para ello, ejecutar la siguiente sentencia:

**MATCH (n :Pais)**

**SET n.Idiomas= split( replace(replace( replace(n.Idiomas, '[', ''), ']', ''), '"', '') , ',' )**

Ejercicio 1

Escribir una consulta para conocer los Paises donde se habla más de 2 idiomas y uno de ellos es 'Italiano'

Rta

***Ejercicio 2***

Calcular los pares de personas que residen en la misma ciudad.

Rta

***Ejercicio 3***

¿Por qué esta variante devuelve 0 matching? ¿Cuál es el error?

MATCH (p1: Persona) -[e :resideEn]-> (c :Ciudad),

      (p2 :Persona) -[ e :resideEn] -> (c :Ciudad)

RETURN  p1, p2, c

Rta

***Ejercicio 4***

Explicar por qué esta consulta es errónea para resolver lo anterior. De hecho, se obtiene Romina en el resultado que reside en Berlin

MATCH path= (p :Persona)-[\*]->(n :Pais)

WHERE n.Nombre='Espana'

RETURN path

Rta:

***Ejercicio 5***

Escribir una consulta Cypher para calcular por cada sexo del nodo Persona, cuántos hay.

Rta

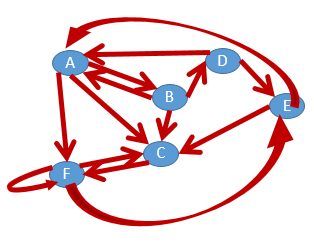
***Ejercicio 6***

Escribir una consulta Cypher para calcular por cada sexo del nodo Persona, cuántos hay. Mostrar en el resultado sólo si la cantidad obtenida es menor que 8

Rta

***Ejercicio 7***

Vamos a crear un grafo adicional temporario. Es decir, nodos rotulados de tal manera que después sean fáciles borrarlos.



Para ello, ejecutar la siguiente sentencia

create (a :URL {link: "A"}), (b :URL {link: "B"}),

(c :URL {link: "C"}), (d :URL {link: "D"}),

(e :URL {link: "E"}), (f :URL {link: "F"}),

(a)-[:follows]->(b), (a)-[:follows]->(c), (a)-[:follows]->(f),

(b)-[:follows]->(a), (b)-[:follows]->(c), (b)-[:follows]->(d),

(c)-[:follows]->(f),

(d)-[:follows]->(a), (d)-[:follows]->(e),

(e)-[:follows]->(a), (e)-[:follows]->(c),

(f)-[:follows]->(c), (f)-[:follows]->(e), (f)-[:follows]->(f)

RETURN a, b, c, d, e, f

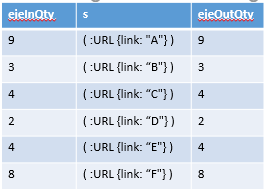
Se quiere saber para cada Nodo cuantos lo siguen, a cuantos sigue él. O sea, calcular la cantidad de ejes entrantes y salientes a cada nodo. **Explicar** por qué al ejecutar la consulta Cypher sobre este grafo no se obtiene lo deseado.

MATCH (p :URL)-[]->(s)-[]->(x :URL)

RETURN Count(p) AS ejeInQty, s, Count(x) AS ejeOutQty

ORDER BY s.link

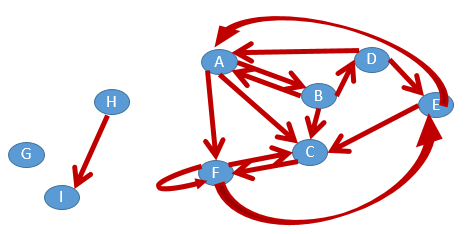
Y se obtiene



Rta

***Ejercicio 8***

a) Proponer una consulta Cypher para calcular lo anterior. Como debe resultar correcta para cualquier grafo (inclusive uno que tenga nodos aislados y nodos con solo ejes de entrada o solo ejes de salida), vamos a agregar 3 nodos más al grafo anterior para que quede así



Ejecutar la sentencia que agregue esa info, es decir

create (g :URL {link: "G"}), (h :URL {link: "H"}), (i :URL {link: "I"}),

(h)-[:follows]->(i)

Chequear la consulta arreglada sobre este grafo. Debe obtenerse 0 para el nodo aislado, etc.

Rta

b) Borrar el grafo temporario de la siguiente manera

**MATCH (:URL)-[r :follows]->(:URL) DELETE r**

Y luego

**MATCH (n:URL) DELETE n**

***Ejercicio 9***

Proponer una consulta Cypher para calcular la longitud promedio de las “llamadas” emitidas entre cada par de teléfonos que se comunicaron en el tiempo. En la respuesta deberá aparecer: Nro de teléfono del emisor, Nro de teléfono del receptor y el promedio de dichas duración. Ordenar el resultado por ambos números de teléfono.

El resultado esperado es de 7 tuplas:

|  |  |  |
| --- | --- | --- |
| NroEmisor | NroReceptor | promedioDuración |
| (103)1111-1111 | (104)1111-1111 | **7.0** |
| (104)1111-1111 | (103)1111-1111 | **2.5** |
| (125)1111-1111 | (126)1111-1112 | **20.0** |
| (125)1111-1111 | (126)1111-1113 | **17.0** |
| (126)1111-1113 | (127)1111-1113 | **3.0** |
| (158)1111-1111 | (160)1111-1113 | **6.5** |
| (158)1111-1112 | (160)1111-1113 | **1.0** |

Tener en cuenta que para sacar el promedio debe aplicarse sobre tipo de datos numérico. Si el valor de la duración de la llamada es de tipo string, transformarla previamente con la siguiente sentencia para que la BD quede reparada:

MATCH ()-[e :llamada]->()

SET e.Duracion = toInteger(e.Duracion)

Además, arreglar lo mismo para la longitud de los SMS

MATCH ()-[e :sms]->()

SET e.Longitud = toInteger(e.Longitud)

Rta

***Ejercicio 10***

Parecido al anterior, pero sumarizar la duración de la llamada en sí, sin importar quien la inició.

Es decir, en el resultado anterior estos 2 teléfonos se llaman entre sí:

|  |  |  |
| --- | --- | --- |
| (103)1111-1111 | (104)1111-1111 | **7.0** |
| (104)1111-1111 | (103)1111-1111 | **2.5** |

y no se sumarizan entre ellos porque se diferenció quien inició la llamada.

El resultado esperado es de 6 tuplas (no repetir los pares A B y B A). Elegir que aparezca en el resultado el par NroEmisor < NroReceptor:

|  |  |  |
| --- | --- | --- |
| Nro1 | Nro2 | promedioDuración |
| (103)1111-1111 | (104)1111-1111 | **4.0**  **(hubo 3 llamadas entre ellos. Una del 103 al 104 que duró 7; y otras en sentido contrario que duraron 2 y 3** |
| (125)1111-1111 | (126)1111-1112 | **20.0**  **(hubo una llamada que duró 20)** |
| (125)1111-1111 | (126)1111-1113 | **17.0**  **(hubo una llamada que duró 17)** |
| (126)1111-1113 | (127)1111-1113 | **3.0**  **(hubo una llamada que duró 3)** |
| (158)1111-1111 | (160)1111-1113 | **6.5**  **(hubo una llamada que duró 12 y otra que duró 1)** |
| (158)1111-1112 | (160)1111-1113 | **1.0**  **(hubo una llamada que duró 1)** |

Rta

***Ejercicio 11***

Ahora se quiere hacer la agregación anterior (la que no diferencia quién inicia la llamada), pero haciendo rollUp a Persona. Es decir, si tenemos

Luis (de Londres) es el dueño de (158)1111-1111 y (158)1111-1112

Andrea (Roma) es el dueño de (125)1111-1111 y (127)1111-1113

Leandro (Roma) es el dueño de (126)1111-1112 y (126)1111-1113

Contabilizar el promedio de la duración de la llamada a nivel pares de Personas (y las personas pueden tener varios teléfonos)

En nuestro caso tenemos que (marcamos en negrita los que hicieron o recibieron llamadas, o sea, van a aparecer en el resultado)

**Ana (ID 315, de Liverpool) es la dueña de (160)1111-1113 y (161)1111-1111**

**Andrea (ID 307, de Roma) es el dueño de (125)1111-1111 y (127)1111-1113**

Andrea (ID 311, de Madrid ) es dueña de (151)1111-1112 y (154)1111-1112

Jimena (ID 303, de Bruselas) es dueña de (105)1111-1113

**Juan (ID 300, de Amberes) es dueño de (104)1111-1112 y (103)1111-1111**

Juan (ID 312, de Ginebra) es dueño de (155)1111-1111, (156)1111-1112 y (157)1111-1113

Juana (ID 305, de Berlin ) es dueña de (114)1111-1113 y (115)1111-1111

**Leandro (ID 308, de Roma) es el dueño de (126)1111-1112 y (126)1111-1113**

Leticia (ID 309, de Monaco) es el dueña de (128)1111-1112

Luis (ID 306, de Frankfurt) es el dueño de (116)1111-1111 y (116)1111-1112

**Luis (ID 313, de Londres) es el dueño de (158)1111-1111, (158)1111-1112 y (159)1111-1113**

Marta (ID 302, de Bruselas) es dueña de (103)1111-1112

**Roberto (ID 301, de Amberes) es dueño de (104)1111-1111**

Romina ( ID 304, de Berlin) es dueña de (114)1111-1111 y (116)1111-1113

Silvio (ID 310, de Madrid) es dueño de (151)1111-1113 y (151)1111-1111

Silvio (ID 314, de Londres) es dueño de (159)1111-1112 y (158)1111-1113

Esta vez, desplegar todas las propiedades de los nodos Persona en el resultado, pero ordenarlo por el nombre (que no aparezca Luis-Ana y Ana-Luis sino sólo uno de ellos. Por si los nombre se repitieran considerar que lo que seguro no se repite es sus IDs. No hubo llamadas entre diferentes teléfonos de la misma Persona). El resultado esperado es:

|  |  |  |
| --- | --- | --- |
| Persona1 | Persona2 | promedioDuración |
| ID: 307  Nombre: Andrea  Sexo: M | ID: 308  Nombre: Leandro  Sexo: M | **13.333333333333334**  **(Se realizó el promedio entre las llamadas que duraron 17, 20 y 3)** |
| ID: 300  Nombre: Juan  Sexo: M | ID: 301  Nombre: Roberto  Sexo: M | **4.0**  **(Se realizó el promedio entre las llamadas que duraron 2, 7 y 3)** |
| ID: 313  Nombre: Luis  Sexo: M | ID: 315  Nombre: Ana  Sexo: F | **4.666666666666667**  **(Se realizó el promedio entre las llamadas que duraron 12, 1 y 1)** |

Rta

***Ejercicio 12***

Técnicamente hablando un teléfono no puede llamar al mismo. Sin embargo, sí es posible que un teléfono envíe un sms al mismo teléfono.

12.1) Proponer una consulta que muestre si hay alguna Persona que se envió mensajes de texto a sí misma .

Con el grafo deberíamos obtener a Juana.

Rta

12.2) Proponer una consulta para averiguar aquellas Personas que se envío mensajes de texto a sí misma pero por el mismo teléfono.

Con el grafo obtenido se obtiene conjunto vacío.

Rta

***Ejercicio 13***

Por cada par de **personas diferentes** que tuvieron interacción entre ellas, ya sea por **sms o por llamada**, se desea que aparezcan en el resultado. Sin tener en cuenta quien inició la llamada o quien envió el sms, deberá aparecer una columna que diga cuántas interacciones tuvieron. Ej: Si A envió 3 mensajes de texto a B y esa es la única interacción con B, aparecerá un 3 en el casillero que las vincule. Ej: Si C envió un mensaje de texto y recibió 4 llamadas de D, aparecerá un 5 en el casillero que las vincule

Esta vez, desplegar todas las propiedades de los nodos Persona en el resultado, pero ordenarlo por el nombre (que no aparezca Luis-Ana y Ana-Luis sino sólo uno de ellos. Por si los nombres se repitieran considerar que lo que seguro no se repite es sus IDs. El resultado esperado es:

|  |  |  |
| --- | --- | --- |
| Persona1 | Persona2 | Cantidad |
| ID: 307  Nombre: Andrea  Sexo: M | ID: 308  Nombre: Leandro  Sexo: M | **3** |
| ID: 300  Nombre: Juan  Sexo: M | ID: 303  Nombre: Jimena  Sexo: F | **1** |
| ID: 300  Nombre: Juan  Sexo: M | ID: 301  Nombre: Roberto  Sexo: M | **7** |
| ID: 300  Nombre: Juan  Sexo: M | ID: 304  Nombre: Romina  Sexo: F | **2** |
| ID: 305  Nombre: Juana  Sexo: F | ID: 306  Nombre: Luis  Sexo: M | **1** |
| ID: 313  Nombre: Luis  Sexo: M | ID: 315  Nombre: Ana  Sexo: F | **3** |
| ID: 304  Nombre: Romina  Sexo: F | ID: 311  Nombre: Andrea  Sexo: F | **1** |
| ID: 304  Nombre: Romina  Sexo: F | ID: 314  Nombre: Silvio  Sexo: M | **2** |

Rta

***Ejercicio 14***

Ídem al ejercicio 13, pero solo mostrándolos en el resultado si tuvieron por lo menos 3 interacciones.

Con los datos del ejemplo debería obtenerse:

|  |  |  |
| --- | --- | --- |
| Persona1 | Persona2 | Cantidad |
| ID: 307  Nombre: Andrea  Sexo: M | ID: 308  Nombre: Leandro  Sexo: M | **3** |
| ID: 300  Nombre: Juan  Sexo: M | ID: 301  Nombre: Roberto  Sexo: M | **7** |
| ID: 313  Nombre: Luis  Sexo: M | ID: 315  Nombre: Ana  Sexo: F | **3** |

Rta

***Ejercicio 15***

Los nodos Ciudad tienen información georeferenciada (lat/lon)

Escribir una consulta para mostrar las ciudades que poseen longitud negativa

Rta

***Ejercicio 16***

Arreglar el error cambiando las propiedades con la siguiente sentencia

MATCH (c: Ciudad) SET c.lat=toFloat(c.lat), c.lon=toFloat(c.lon)

Volver a ejecutar la query y verificar que se obtiene lo esperado.

***Ejercicio 17***

Proponer una consulta que muestre cada Ciudad (todas sus props) y un boolean que indique si está al este del Meridiano de Greenwich y otra que indique si está al Oeste del mismo. Obviamente esas columnas son excluyentes.

Con los datos del ejemplo, deberían obtener:

|  |  |  |
| --- | --- | --- |
| n | este | oeste |
| (:Ciudad:Capital {Nombre: Viena, lon: 16.363449, ID: 200, lat: 48.210033}) | false | true |
| (:Ciudad {Nombre: Amberes, lon: 4.402771, ID: 201, lat: 51.26019}) | false | true |
| (:Ciudad:Capital {Nombre: Bruselas, lon: 4.34878, ID: 202, lat: 50.85045}) | false | true |
| (:Ciudad {Nombre: Brujas, lon: 3.22424, ID: 203, lat: 51.20892}) | false | true |
| (:Ciudad:Capital {Nombre: Paris, lon: 2.349014, ID: 204, lat: 48.864716}) | false | true |
| (:Ciudad:Capital {Nombre: Berlin, lon: 13.404954, ID: 205, lat: 52.520008}) | false | true |
| (:Ciudad {Nombre: Frankfurt, lon: 8.682127, ID: 206, lat: 50.110924}) | false | true |
| (:Ciudad:Capital {Nombre: Roma, lon: 12.496366, ID: 207, lat: 41.902782}) | false | true |
| (:Ciudad {Nombre: Milan, lon: 9.18854, ID: 208, lat: 45.464664}) | false | true |
| (:Ciudad {Nombre: Napoles, lon: 14.3055, ID: 209, lat: 40.853294}) | false | true |
| (:Ciudad {Nombre: Venecia, lon: 12.32714, ID: 210, lat: 45.4387}) | false | true |
| (:Ciudad:Capital {Nombre: Monaco, lon: 7.416667, ID: 211, lat: 43.733334}) | false | true |
| (:Ciudad {Nombre: Montecarlo, lon: 7.416667, ID: 212, lat: 43.733334}) | false | true |
| (:Ciudad:Capital {Nombre: Madrid, lon: -3.70379, ID: 213, lat: 40.416775}) | true | false |
| (:Ciudad {Nombre: Bilbao, lon: -2.935013, ID: 214, lat: 43.262985}) | true | false |
| (:Ciudad {Nombre: Sevilla, lon: -5.994, ID: 215, lat: 37.392}) | true | false |
| (:Ciudad:Capital {Nombre: Berna, lon: 7.44744, ID: 216, lat: 46.94809}) | false | true |
| (:Ciudad {Nombre: Ginebra, lon: 6.143158, ID: 217, lat: 46.204391}) | false | true |
| (:Ciudad:Capital {Nombre: Londres, lon: -0.118092, ID: 218, lat: 51.509865}) | true | false |
| (:Ciudad {Nombre: Glasgow, lon: -4.251433, ID: 219, lat: 55.860916}) | true | false |
| (:Ciudad {Nombre: Edimburgo, lon: -3.188267, ID: 220, lat: 55.953251}) | true | false |
| (:Ciudad {Nombre: Liverpool, lon: -2.983333, ID: 221, lat: 53.400002}) | true | false |

Rta

***Ejercicio 18***

Proponer una consulta para calcular cuántas ciudades están al Este del Meridiano de Greenwich y cuántas al oeste. Con los datos del ejemplo, deberían obtener:

CantEste CantOeste

7 15

Rta

***Ejercicio 19***

Se quiere calcular la distancia entre Personas que se comunicaron por llamada, segun el lugar que dijeron que residían.

Para preguntar por distancia se genera un punto a partir de lon /lan y se pide distancia en KM

Ejemplo:

RETURN point.distance(

   point( { longitude: **53.1**, latitude: **12.78**, crs:  'WGS-84'}),

   point( { longitude: **56.1**, latitude: **12.78**, crs:  'WGS-84'})) / 1000

La consulta, es entonces

MATcH (p1: Persona)-[r1 :resideEn]->(c1 :Ciudad),

(p2: Persona)-[r2 :resideEn]->(c2 :Ciudad)

WHERE p1.ID < p2.ID

AND exists( (p1)-[:registra]->(:Tel)-[:llamada]-(:Tel)-[:registra]-(p2) )

RETURN p1, p2,

point.distance(

    point({longitude: c1.lon, latitude: c1.lat} ),   point( {longitude: c2.lon, latitude: c2.lat} )

    ) / 1000 as km

Se obtiene

|  |  |  |
| --- | --- | --- |
| Persona1 | Persona2 | km |
| ID: 307  Nombre: Andrea  Sexo: M | ID: 308  Nombre: Leandro  Sexo: M | **0.0** |
| ID: 300  Nombre: Juan  Sexo: M | ID: 301  Nombre: Roberto  Sexo: M | **0.0** |
| ID: 313  Nombre: Luis  Sexo: M | ID: 315  Nombre: Ana  Sexo: F | **286.4002897484332** |

Explicar el resultado

Rta