



UNIVERSIDAD POLITECNICA SALESIANA

SEDE CUENCA

CARRERA: INGENIERIA DE SISTEMAS

Nombre: *Bryam Gabriel Mora Lituma*

Materia: *Sistemas Expertos*

Fecha: *31/01/2021*

Diseño y Desarrollo de un Algoritmo Knn en Neo4j.

► Fila B - 1: Este es un conjunto de datos de empleados en una empresa y el resultado es estudiar sobre la deserción de los empleados, para ello se debe descargar los datos del siguiente link:

<http://smalldatabrains.com/wpcontent/uploads/2018/03/data.csv>

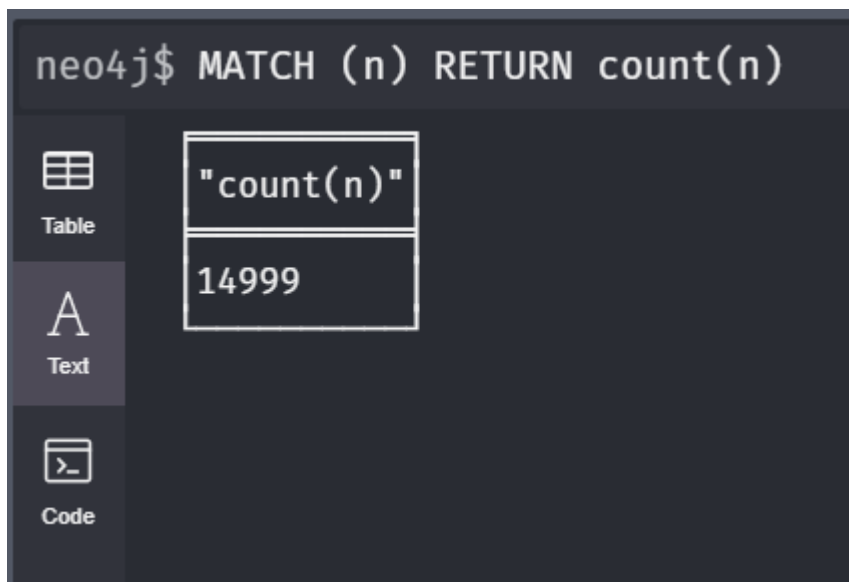
Creación de nodos desde Python.

In []: ▶

```
1 import csv
2 import pandas as pd
3 from neo4j import StructuredNode, StringProperty, RelationshipTo, RelationshipFrom, config, IntegerPr
4 config.DATABASE_URL = 'bolt://neo4j:sexpertos@localhost:7687'
5
6 df = pd.read_csv(r"data.csv", sep=';')
7 lista = [list(row) for row in df.values]
8
9 class Datos(StructuredNode):
10     empleado = StringProperty(index=True)
11     nivel_de_satisfacción = FloatProperty(index=True)
12     ultima_evaluacion = FloatProperty(index=True)
13     proyecto_de_número = FloatProperty(index=True)
14     promedio_de_horas_mensuales = FloatProperty(index=True)
15     tiempo_dedicado_a_la_compañía = FloatProperty(index=True)
16     accidente_laboral = FloatProperty(index=True)
17     promoción_últimos_5_años = FloatProperty(index=True)
18     izquierda = FloatProperty(index=True)
19
20 f=1
21 s="Empleado"
22
23 for x in lista:
24     f=f+1
25     t=s+str(f)
26     datos=Datos(
27         empleado = t,
28         nivel_de_satisfacción = (x[0]),
29         ultima_evaluacion = (x[1]),
30         proyecto_de_número = (x[2]),
31         promedio_de_horas_mensuales = (x[3]),
32         tiempo_dedicado_a_la_compañía = (x[4]),
33         accidente_laboral = (x[5]),
34         promoción_últimos_5_años = (x[6]),
35         izquierda = (x[7]),
36     ).save()
```

Match de los nodos ingresados por Python.

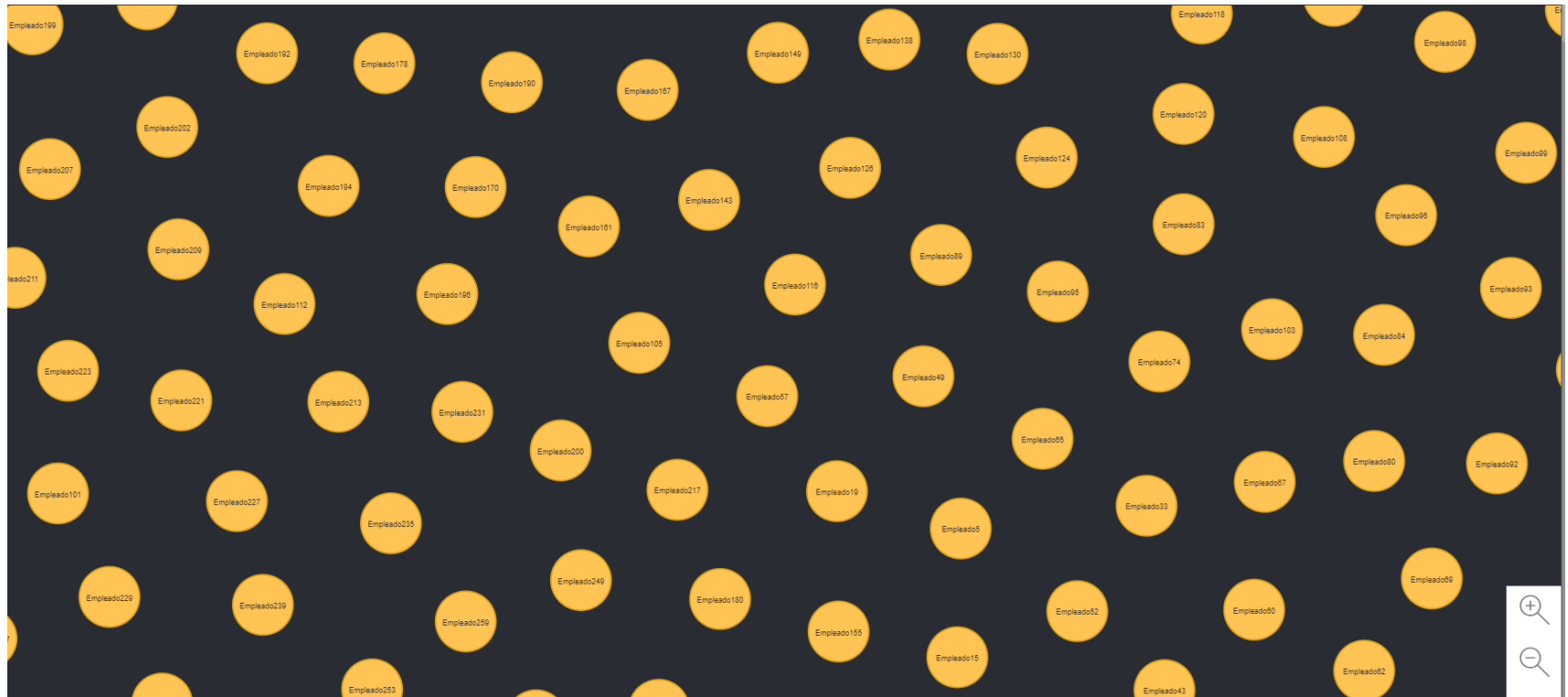
MATCH (n) RETURN count(n)



The screenshot shows a Jupyter Notebook interface with a dark theme. At the top, a code cell contains the query `neo4j$ MATCH (n) RETURN count(n)`. Below the code cell, there is a sidebar with three icons: a table icon labeled "Table", a text icon labeled "Text", and a code icon labeled "Code". The "Table" icon is selected. To the right of the sidebar, a table displays the result of the query. The table has two rows: the first row contains the string `"count(n)"`, and the second row contains the integer `14999`.

"count(n)"
14999

MATCH (n) RETURN n



Algoritmo de Knn para encontrar la similitud en Neo4J.

```
In [ ]: 1 CALL gds.graph.create(  
2         'Empleados',  
3         {  
4             Datos: {  
5                 label: 'Datos',  
6                 properties: 'nivel_de_satisfacción'  
7             }  
8         },  
9         '*'  
10    );
```

```
In [ ]: ▶ 1 CALL gds.beta.knn.stream('Empleados', {
2         topK: 1,
3         nodeWeightProperty: 'nivel_de_satisfacción',
4         // The following parameters are set to produce a deterministic result
5         randomSeed: 42,
6         concurrency: 1,
7         sampleRate: 1.0,
8         deltaThreshold: 0.0
9     })
10 YIELD node1, node2, similarity
11 RETURN gds.util.asNode(node1).empleado AS Empleado1, gds.util.asNode(node2).empleado AS Empleado2, similarity
12 ORDER BY similarity DESCENDING, Empleado1, Empleado2
```

Resultados obtenidos de la similitud y aplicando el algoritmo KNN.

	Empleado1	Empleado2	similarity
197	"Empleado14662"	"Empleado7533"	0.9900990099009903
198	"Empleado9482"	"Empleado3965"	0.9900990099009903
199	"Empleado10"	"Empleado12306"	1.0
200	"Empleado100"	"Empleado1188"	1.0
201	"Empleado1000"	"Empleado1381"	1.0
202	"Empleado10000"	"Empleado11849"	1.0
203	"Empleado10001"	"Empleado8076"	1.0

Started streaming 14999 records after 2 ms and completed after 8 ms, displaying first 1000 rows.

También se exporto los resultados a un archivo csv para visualizar todos los resultados

Creamos 3 nuevos nodos.

```
In [ ]: 1 CREATE (empleadon1:Datos {empleado: 'Empleado1', nivel_de_satisfacción : 0.12})
        2 CREATE (empleadon2:Datos {empleado: 'Empleado2', nivel_de_satisfacción : 0.14})
        3 CREATE (empleadon3:Datos {empleado: 'Empleado3', nivel_de_satisfacción : 0.15})
```

Match de los nuevos nodos.

MATCH (n:Datos) where n.empleado='Empleadon1' or n.empleado='Empleadon2' or n.empleado='Empleadon3' RETURN n



Creamos denuevo grap para sacar la relacion entre los 3 nuevos nodos

```
In [ ]: 1 CALL gds.graph.create(  
2         'Empleados',  
3         {  
4           Datos: {  
5             label: 'Datos',  
6             properties: 'nivel_de_satisfacción'  
7           }  
8         },  
9         '*',  
10        );
```

```

In [ ]: ► 1 CALL gds.beta.knn.stream('Empleados', {
2         topK: 1,
3         nodeWeightProperty: 'nivel_de_satisfacción',
4         // The following parameters are set to produce a deterministic result
5         randomSeed: 42,
6         concurrency: 1,
7         sampleRate: 1.0,
8         deltaThreshold: 0.0
9     })
10 YIELD node1, node2, similarity
11 RETURN gds.util.asNode(node1).empleado AS Empleado1, gds.util.asNode(node2).empleado AS Empleado2, similarity
12 ORDER BY similarity DESCENDING, Empleado1, Empleado2

```

Aplicando el algoritmo de solicitud Knn vemos la similitud de los nuevos nodos ingresados.

Empleado13906,Empleadon2,1.0

Empleado3130,Empleadon1,1.0

Empleado5337,Empleadon1,1.0

Empleadon1,Empleado5337,1.0

Empleadon2,Empleado13906,1.0

Empleadon3,Empleado11058,1.0

Generar otro entorno en donde solo ingrese el 70% de los datos y validar con el 30%. Agregar el grafico con los nodos conformados.

Creación del 70% de los nodos

In []: ▶

```
1 import csv
2 import pandas as pd
3 from neo4j import StructuredNode, StringProperty, RelationshipTo, RelationshipFrom, config, IntegerPr
4 config.DATABASE_URL = 'bolt://neo4j:ddd@localhost:7687'
5
6 df = pd.read_csv(r"data2.csv", sep=';')
7 lista = [list(row) for row in df.values]
8
9 class Datos(StructuredNode):
10     empleado = StringProperty(index=True)
11     nivel_de_satisfacción = FloatProperty(index=True)
12     ultima_evaluacion = FloatProperty(index=True)
13     proyecto_de_número = FloatProperty(index=True)
14     promedio_de_horas_mensuales = FloatProperty(index=True)
15     tiempo_dedicado_a_la_compañía = FloatProperty(index=True)
16     accidente_laboral = FloatProperty(index=True)
17     promoción_últimos_5_años = FloatProperty(index=True)
18     izquierda = FloatProperty(index=True)
19
20 f=1
21 s="Empleado"
22
23 for x in lista:
24     f=f+1
25     t=s+str(f)
26     datos=Datos(
27         empleado = t,
28         nivel_de_satisfacción = (x[0]),
29         ultima_evaluacion = (x[1]),
30         proyecto_de_número = (x[2]),
31         promedio_de_horas_mensuales = (x[3]),
32         tiempo_dedicado_a_la_compañía = (x[4]),
33         accidente_laboral = (x[5]),
34         promoción_últimos_5_años = (x[6]),
35         izquierda = (x[7]),
36     ).save()
```

Match del 70% de los nodos



Creación del 30% de los nodos

In []: ▶

```
1 import csv
2 import pandas as pd
3 from neo4j import StructuredNode, StringProperty, RelationshipTo, RelationshipFrom, config, IntegerPr
4 config.DATABASE_URL = 'bolt://neo4j:ddd@localhost:7687'
5
6 df = pd.read_csv(r"data3.csv", sep=';')
7 lista = [list(row) for row in df.values]
8
9 class Datos2(StructuredNode):
10     empleado = StringProperty(index=True)
11     nivel_de_satisfacción = FloatProperty(index=True)
12     ultima_evaluacion = FloatProperty(index=True)
13     proyecto_de_número = FloatProperty(index=True)
14     promedio_de_horas_mensuales = FloatProperty(index=True)
15     tiempo_dedicado_a_la_compañía = FloatProperty(index=True)
16     accidente_laboral = FloatProperty(index=True)
17     promoción_últimos_5_años = FloatProperty(index=True)
18     izquierda = FloatProperty(index=True)
19
20 f=1
21 s="Trabajador"
22
23 for x in lista:
24     f=f+1
25     t=s+str(f)
26     datos=Datos2(
27         empleado = t,
28         nivel_de_satisfacción = (x[0]),
29         ultima_evaluacion = (x[1]),
30         proyecto_de_número = (x[2]),
31         promedio_de_horas_mensuales = (x[3]),
32         tiempo_dedicado_a_la_compañía = (x[4]),
33         accidente_laboral = (x[5]),
34         promoción_últimos_5_años = (x[6]),
35         izquierda = (x[7]),
36     ).save()
```

Match del 30% de los nodos



Algoritmo knn para el 70% comparando con el 30% de los datos.

```
In [ ]: ► 1 CALL gds.graph.create(  
2         'Empleados',  
3         {  
4             Datos: {  
5                 label: 'Datos',  
6                 properties: 'nivel_de_satisfacción'  
7             },  
8             Datos2: {  
9                 label: 'Datos2',  
10                properties: 'nivel_de_satisfacción'  
11            },  
12        },  
13        '*',  
14    );
```

```
In [ ]: ► 1 CALL gds.beta.knn.stream('Empleados', {  
2     topK: 1,  
3     nodeWeightProperty: 'nivel_de_satisfacción',  
4     // The following parameters are set to produce a deterministic result  
5     randomSeed: 42,  
6     concurrency: 1,  
7     sampleRate: 1.0,  
8     deltaThreshold: 0.0  
9 })  
10 YIELD node1, node2, similarity  
11 RETURN gds.util.asNode(node1).empleado AS Empleado1, gds.util.asNode(node2).empleado AS Empleado2, similarity  
12 ORDER BY similarity , Empleado1, Empleado2
```

Resultados de la comparación

	Empleado1	Empleado2	similarity
195	"Empleado9482"	"Empleado3965"	0.9900990099009903
196	"Trabajador1208"	"Empleado2391"	0.9900990099009903
197	"Trabajador1406"	"Empleado7571"	0.9900990099009903
198	"Trabajador4164"	"Empleado7533"	0.9900990099009903
199	"Empleado10"	"Trabajador1808"	1.0
200	"Empleado100"	"Empleado1188"	1.0
201	"Empleado1000"	"Empleado1381"	1.0
Started streaming 14999 records after 2 ms and completed after 6 ms, displaying first 1000 rows.			