EII
Escuela de Ingeniería Informática de Oviedo

C. de Valdes Salas, 11
Tel. 98 510 27 96
eii@uniovi.es
33007 Oviedo

- CYPHER Y NEO4J -

Guillermo Facundo Colunga, Pablo Menéndez Suárez y Jorge Vila Suárez.

PROFESORADO:
Darío Álvarez Gutiérrez - darioa@uniovi.es

Resumen. En este documento se presentan la propuesta, realización y conclusiones de la primera práctica de la asignatura de Repositorios de la Información 2017 de la Universidad de Oviedo. Usando el Sistema de Gestión de Bases de Datos en grafo neo4j se explorará cómo crear una base de datos sobre el dominio de aplicación escogido. Y, la realización de consultas sobre la instancia creada. El documento se acompaña del script en cypher (lenguaje de consultas en neo4j) para la creación de la instancia y los scripts, también en cypher, con las consultas que se realizarán sobre la misma.

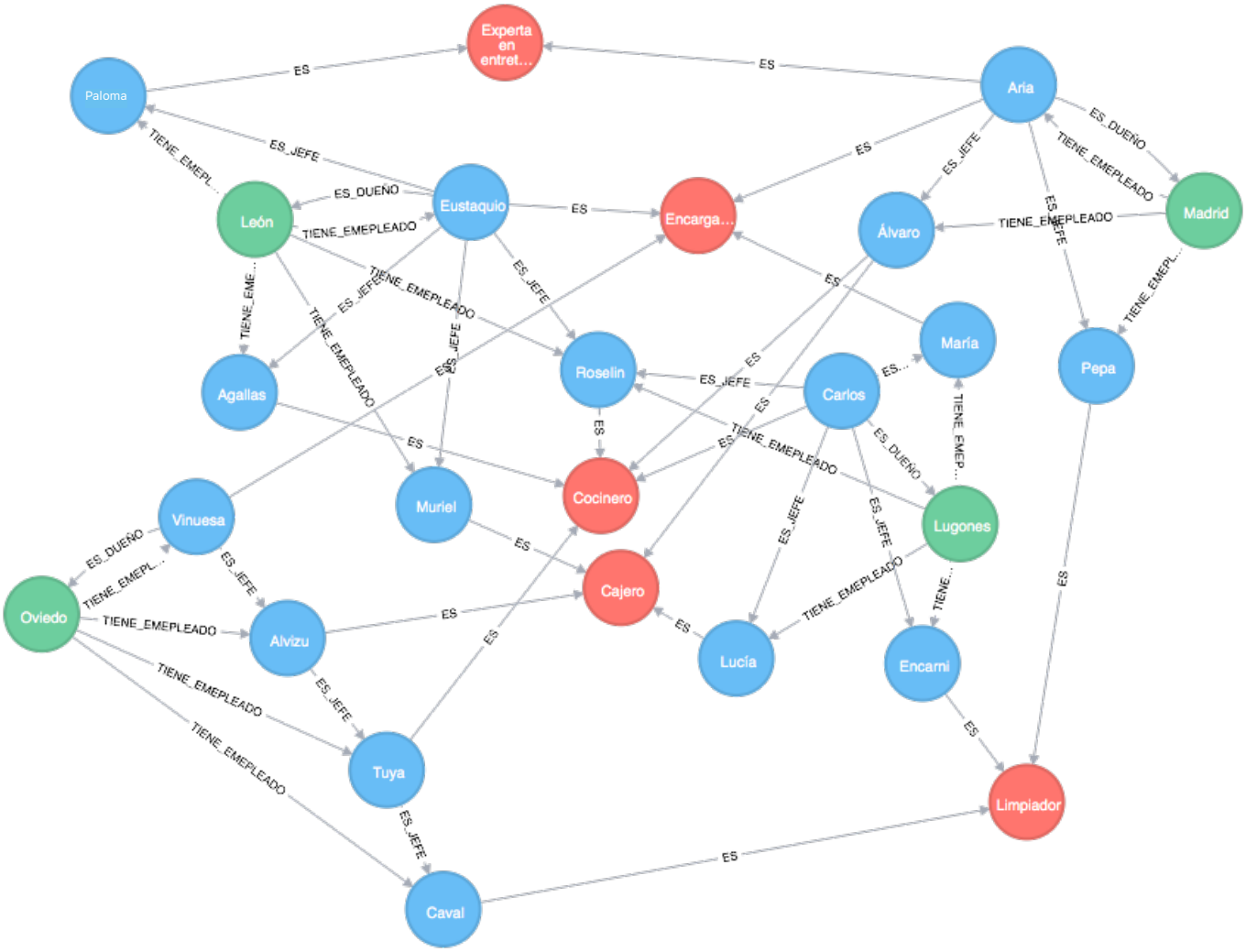
Palabras clave: neo4j, cypher, base de datos, grafo, noSQL, SGBD.

Dominio de Aplicación

Para esta práctica elegimos un dominio que fuera representativo del mundo real. Así mismo como se trata de una base de datos en grafo también intentamos aproximarnos lo más posible a lo que en la vida real podían ser los nodos y sus relaciones. Es por esto que finalmente escogimos un modelo de aplicación basado en las famosas franquicias Kentucky Fried Chicken (KFC). Dentro de las franquicias intentamos definir las relaciones entre trabajadores, dueños, puestos y franquicias en si. De esa forma podemos relacionar a una persona con una franquicia, el tipo de relación que tiene y el resto de relaciones con otras personas de la misma o distinta franquicia.

De esta forma definimos las que las siguientes entidades irán representadas a través de nodos: Persona, Franquicia y Puesto. Y como relaciones encontraremos: ES, ES_DUEÑO, ES_JEFE y TIENE_EMPLEADO.

Representación de la Instancia



En la instancia anterior encontramos un grafo formado por 25 nodos y 51 relaciones. Entre los nodos nos encontramos los siguientes tipos:

Persona. Representan una persona del mundo real, en nuestro dominio pueden tener nombre y apellido.

Puesto. Representa el puesto que puede desempeñar un empleado dentro de una franquicia.

Franquicia. Es la representación del restaurante en si.

Y entre las relaciones:

ES. Se aplica a los nodos de tipo Persona y tiene como fin un nodo de tipo Puesto, en dirección Persona -> Puesto. Representando así el puesto de desempeña una persona.

ES_DUEÑO. Se aplica a nodos de tipo Persona y tiene como fin un nodo de tipo Franquicia, en dirección Persona -> Franquicia. Representando que una persona es dueña de una franquicia.

ES_JEFE. Se aplica entre nodos de tipo Persona {a, b} y (a)->(b) representa que (a) es jefe de (b).

TIENE_EMPLEADO. Se aplica a nodos de tipo Franquicia y tiene como destino nodos de tipo Persona, representando así la relación entre una franquicia y un empleado.

Código de creación del grafo

```
// Creando nodos de tipo Persona.
CREATE (Paloma:Persona {nombre:'Paloma', apellidos: 'Mas Pellicer'})
CREATE (Eustaquio:Persona {nombre:'Eustaquio'})
CREATE (Muriel:Persona {nombre:'Muriel'})
CREATE (Agallas:Persona {nombre:'Agallas'})
CREATE (Roselin:Persona {nombre:'Roselin'})
CREATE (Carlos:Persona {nombre:'Carlos'})
CREATE (María:Persona {nombre:'María'})
CREATE (Pepa:Persona {nombre:'Pepa'})
CREATE (Aria:Persona {nombre:'Aria'})
CREATE (Álvaro:Persona {nombre:'Álvaro'})
CREATE (Encarni:Persona {nombre:'Encarni'})
CREATE (Lucía:Persona {nombre:'Lucía'})
CREATE (Vinuesa:Persona {nombre:'Vinuesa'})
CREATE (Alvizu:Persona {nombre:'Alvizu'})
CREATE (Tuya:Persona {nombre:'Tuya'})
CREATE (Caval:Persona {nombre:'Caval'})

// Creando nodos de tipo Franquicia.
CREATE (Leon:Franquicia {ciudad:'León', nombre :'Franquicia de León'})
CREATE (Madrid:Franquicia {ciudad:'Madrid', nombre :'Franquicia de Madrid'})
CREATE (Lugones:Franquicia {ciudad:'Lugones', nombre :'Franquicia de Lugones'})
CREATE (CAU:Franquicia {ciudad:'Oviedo', nombre :'Franquicia del CAU'})

// Creando nodos de tipo Puesto
CREATE (Encargado:Puesto {puesto:'Encargado'})
CREATE (Limpiador:Puesto {puesto:'Limpiador'})
CREATE (Cocinero:Puesto {puesto:'Cocinero'})
CREATE (Cajero:Puesto {puesto:'Cajero'})
CREATE (ExpertaEnEntretenimiento:Puesto {puesto:'Experta en entretenimiento'})

// Creando relaciones entre nodos.
CREATE
  (Paloma)-[:ES]->(ExpertaEnEntretenimiento),
  (Aria)-[:ES]->(ExpertaEnEntretenimiento),
  (Aria)-[:ES]->(Encargado),
  (Eustaquio)-[:ES]->(Encargado),
  (María)-[:ES]->(Encargado),
  (Vinuesa)-[:ES]->(Encargado),
  (Alvizu)-[:ES]->(Cajero),
  (Lucía)-[:ES]->(Cajero),
  (Muriel)-[:ES]->(Cajero),
  (Álvaro)-[:ES]->(Cajero),
  (Agallas)-[:ES]->(Cocinero),
  (Roselin)-[:ES]->(Cocinero),
  (Carlos)-[:ES]->(Cocinero),
  (Álvaro)-[:ES]->(Cocinero),
  (Tuya)-[:ES]->(Cocinero),
  (Encarni)-[:ES]->(Limpiador),
  (Caval)-[:ES]->(Limpiador),
```

```
(Pepa)-[:ES]->(Limpiador),
(Leon)-[:ES_TRABAJADOR]->(Muriel),
(Leon)-[:ES_TRABAJADOR]->(Roselin),
(Leon)-[:ES_TRABAJADOR]->(Agallas),
(Leon)-[:ES_TRABAJADOR]->(Eustaquio),
(Leon)-[:ES_TRABAJADOR]->(Paloma),
(CAU)-[:ES_TRABAJADOR]->(Vinuesa),
(CAU)-[:ES_TRABAJADOR]->(Alvizu),
(CAU)-[:ES_TRABAJADOR]->(Tuya),
(CAU)-[:ES_TRABAJADOR]->(Caval),
(Madrid)-[:ES_TRABAJADOR]->(Aria),
(Madrid)-[:ES_TRABAJADOR]->(Pepa),
(Madrid)-[:ES_TRABAJADOR]->(Álvaro),
(Lugones)-[:ES_TRABAJADOR]->(Encarni),
(Lugones)-[:ES_TRABAJADOR]->(María),
(Lugones)-[:ES_TRABAJADOR]->(Roselin),
(Lugones)-[:ES_TRABAJADOR]->(Lucía),
(Eustaquio)-[:ES_DUEÑO]->(Leon),
(Aria)-[:ES_DUEÑO]->(Madrid),
(Carlos)-[:ES_DUEÑO]->(Lugones),
(Vinuesa)-[:ES_DUEÑO]->(CAU),
(Eustaquio)-[:ES_JEFE]->(Paloma),
(Eustaquio)-[:ES_JEFE]->(Muriel),
(Eustaquio)-[:ES_JEFE]->(Roselin),
(Eustaquio)-[:ES_JEFE]->(Agallas),
(Aria)-[:ES_JEFE]->(Pepa),
(Aria)-[:ES_JEFE]->(Álvaro),
(Carlos)-[:ES_JEFE]->(María),
(Carlos)-[:ES_JEFE]->(Encarni),
(Carlos)-[:ES_JEFE]->(Lucía),
(Carlos)-[:ES_JEFE]->(Roselin),
(Vinuesa)-[:ES_JEFE]->(Alvizu),
(Alvizu)-[:ES_JEFE]->(Tuya),
(Tuya)-[:ES_JEFE]->(Caval)
```

Consultas realizadas sobre la instancia

C.1	Dificultad: Elemental

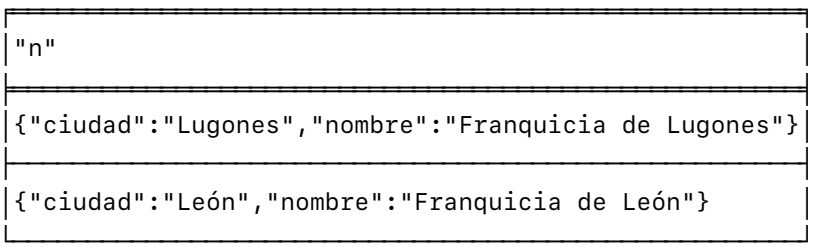
Objetivo Consulta:

Saber en qué franquicia trabaja ‘x’ persona, para este ejemplo escogeremos ‘Roselin’ que trabaja en las franquicias de Lugones y de León.

Código Consulta:

```
MATCH (n:Franquicia) -[:TIENE_EMEPLEADO]-> (:Persona {nombre:'Roselin'})
RETURN n
```

Resultado:



```
"n"
{"ciudad":"Lugones","nombre":"Franquicia de Lugones"}
{"ciudad":"León","nombre":"Franquicia de León"}
```

C.2	Dificultad: Elemental

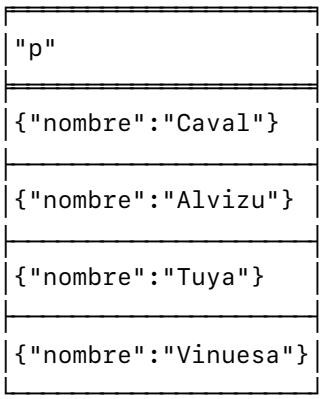
Objetivo Consulta:

En esta consulta se espera obtener los trabajadores de la franquicia de Oviedo.

Código Consulta:

```
MATCH (f:Franquicia {ciudad:'Oviedo'})-[:TIENE_EMEPLEADO]->(p:Persona)
RETURN p
```

Resultado:



```
"p"
{"nombre":"Caval"}
{"nombre":"Alvizu"}
{"nombre":"Tuya"}
{"nombre":"Vinuesa"}
```

C.3	Dificultad: Intermedia

Objetivo Consulta:

En esta consulta se esperan obtener aquellos trabajadores que siendo encargados de una franquicia no son jefe de ningún otro empleado ni propietarios de la franquicia. Una situación que pudiendo ser poco frecuente se puede dar en una franquicia mal coordinada y que puede estar inflando el gasto en sueldos para trabajadores que no desempeñan la función que deberían pues según el contrato de la franquicia “todo encargado deberá ser jefe de al menos otro empleado”.

Código Consulta:

```
MATCH ((p:Persona)-[:ES]->(pu:Puesto) {puesto: "Encargado"})
WHERE NOT ((p)-[:ES_JEFE]->(:Persona))
AND NOT ((p)-[:ES_DUEÑO]->(:Franquicia))
RETURN p
```

Resultado:

```
+--------------------+
| "p"                |
+--------------------+
| {"nombre":"María"} |
+--------------------+
```

C.4	Dificultad: Intermedia

Objetivo Consulta:

Obtener las personas que sean jefes, que no sean dueños de una franquicia y que su nombre empiece por A.

Código Consulta:

```
MATCH (p:Persona) -[:ES_JEFE]-> (:Persona)
WHERE p.nombre STARTS WITH 'A'
AND NOT (p)-[:ES_DUEÑO]->(:Franquicia)
RETURN p
```

Resultado:

```
╒═══════════════════╕
│"p"                │
╞═══════════════════╡
│{"nombre":"Alvizu"}│
└───────────────────┘
```

C.5	Dificultad: Avanzada

Objetivo Consulta:

El objetivo de esta consulta es conseguir el camino mínimo entre dos franquicias. En este caso por ejemplo se desea enviar un paquete a través de las relaciones entre empleados para no pagar a una empresa de transportes desde León hasta Lugones. A través de la siguiente query se sabrá a que empleado dar el paquete y el camino de empleados a seguir hasta la franquicia de Lugones, si es que hay algún empleado con alguna relación con la franquicia de Lugones.

Código Consulta:

```
MATCH (leon:Franquicia { ciudad: 'León' }),(lugones:Franquicia {ciudad: 'Lugones'}),
p = shortestPath((leon)-[*..15]-(lugones))
RETURN p
```

Resultado:

```
╒═══════════════════════════════════════════════════════════════════════════════╕
│"p"                                                                            │
╞═══════════════════════════════════════════════════════════════════════════════╡
│[{"ciudad":"León","nombre":"Franquicia de León"},{},{"nombre":"Roselin         │
│"},{"nombre":"Roselin"},{},{"ciudad":"Lugones","nombre":"Franquicia de         │
│ Lugones"}]                                                                    │
└───────────────────────────────────────────────────────────────────────────────┘
```

C.6	Dificultad: Avanzada

Objetivo Consulta:

Para todos las personas que desarrollan el puesto de encargado obtener el camino mínimo mínimo a una experta en entretenimiento. Esto es importante para las franquicias ya que los padres avisan con muy poco tiempo a los encargados de que desean realizar un cumpleaños en alguna franquicia. De esta forma los encargados saben a quien avisar de forma directa para contactar con alguna de las expertas en entretenimiento de la red de KFCs.

Código Consulta:

```
MATCH (persona:Persona) -[:ES]->(:Puesto {puesto:'Encargado'}) ,
(puesto:Puesto { puesto: 'Experta en entretenimiento' }),
p = shortestPath((persona)-[*..15]-(puesto))
RETURN p
```

Resultado:

```
╒══════════════════════════════════════════════════════════════════════════════╕
│"p"                                                                           │
╞══════════════════════════════════════════════════════════════════════════════╡
│[{"nombre":"Aria"},{},{"puesto":"Experta en entretenimiento"}]                │
├──────────────────────────────────────────────────────────────────────────────┤
│[{"nombre":"Eustaquio"},{},{"apellidos":"Mas Pellicer","nombre":"Palom│
│a"},{"apellidos":"Mas Pellicer","nombre":"Paloma"},{},{"puesto":"Exper│
│ta en entretenimiento"}]                                                      │
├──────────────────────────────────────────────────────────────────────────────┤
│[{"nombre":"María"},{},{"puesto":"Encargado"},{"puesto":"Encargado"},{│
│},{"nombre":"Aria"},{"nombre":"Aria"},{},{"puesto":"Experta en entrete│
│nimiento"}]                                                                   │
├──────────────────────────────────────────────────────────────────────────────┤
│[{"nombre":"Vinuesa"},{},{"puesto":"Encargado"},{"puesto":"Encargado"}│
│,{},{"nombre":"Aria"},{"nombre":"Aria"},{},{"puesto":"Experta en entre│
│tenimiento"}]                                                                 │
└──────────────────────────────────────────────────────────────────────────────┘
```

Conclusiones

Primero. Las bases de datos en modelo de grafo, pese a estar muy orientadas a casos particulares, son realmente potentes para operaciones como caminos mínimos o cierres transitivos. Aquí es precisamente donde se diferencian con las bases de datos relacionales.

Segundo. Cypher es un lenguaje de consultas que se asemeja en gran medida a otros lenguajes de consultas como SQL en su sintaxis, lo que permite reducir la curva de aprendizaje y por lo tanto facilita el acceso a neo4j.

Tercero. Las bases de datos noSQL aportan gran flexibilidad sobre los datos que se guardan, en este caso usamos un modelo de grafo. En nuestro ejemplo esto se puede ver en el nodo de tipo Persona (Paloma) y cualquier otro del mismo tipo. Mientras que paloma tiene nombre y apellidos, el resto tiene sólo nombre.