# Noticias

CREATE (MIGUEL:USUARIO{nombre:"Miguel",nombreUsuario:"Miguel\_u",cuenta\_twitter:"miguelete93"}),(LARO:USUARIO{nombre:"Laro",nombreUsuario:"Laro\_u",cuenta\_twitter:"larocantabro85",telefono:"1234567890"}),(NOTICIA1LARO:NOTICIA{titulo:"Noticia1Laro",cuerpo:"lorem ipsum...",dia:22,mes:5,anio:2017}),(NOTICIA2LARO:NOTICIA{titulo:"Noticia2Laro",cuerpo:"lorem impsum...2",dia:2,mes:3,anio:2015}),(NOTICIAMIGUEL:NOTICIA{titulo:"NoticiaMiguel",cuerpo:"lorem impsum...3",dia:15,mes:3,anio:2017}),(LARO)-[:Redacta]->(NOTICIA1LARO),(LARO)-[:Redacta]->(NOTICIA2LARO),(MIGUEL)-[:Redacta]->(NOTICIAMIGUEL),(LARO)-[:Puntua{nota:3}]->(NOTICIAMIGUEL),(MIGUEL)-[:Puntua{nota:1}]->(NOTICIA1LARO),(MIGUEL)-[:Puntua{nota:5}]->(NOTICIA2LARO)

**1)Devolver todos los usuarios**

match (n:USUARIO) return n

**2)Devolver las noticias cuyo título termine en o**

match (n:NOTICIA) where n.titulo ENDS WITH "o" return n.titulo

**3)Devolver las noticias publicadas entre los años 2015 y 2017**

match (n:NOTICIA) where n.anio>=2015 and n.anio<=2017 return n

**4)Devolver los datos de las relaciones puntúa con puntuación 5**

MATCH (USUARIO)-[r:Puntua{nota:5}]->(NOTICIA) return USUARIO,NOTICIA

**5)Devolver los datos de todas las puntuaciones que ha hecho Miguel**

MATCH (MIGUEL{nombre:"Miguel"})-[r:Puntua]->(NOTICIA) return r.nota,MIGUEL.nombre,NOTICIA.titulo

**6)Devolver las cuentas de twitter de todos los usuarios que hayan valorado noticias de 2017**

match (LARO{nombre:"Laro"})-[:Redacta]->(NOTICIA) where NOTICIA.anio=2017 return NOTICIA.titulo,NOTICIA.cuerpo,NOTICIA.dia,NOTICIA.mes,NOTICIA.anio

**7)Devolver el número de contenidos publicados por Laro**

match (LARO{nombre:"Laro"})-[:Redacta]->(NOTICIA) return count(NOTICIA) as numero\_noticias

**8)Devolver los datos de las relaciones redacta de las noticias publicadas en 2017 cuyo autor se llame “Laro”.**

match (miguel{nombre:”Miguel”})-[:VIEJA:puntuo]->(not2{titulo:”Notificacion})

create la nueva relacion

delete VIEJA

# Personas

**1.- Crear los nodos**

create (maria: persona{nombre:"María", edad: 27}), (juan: persona{nombre:"Juan", edad:35}),

(javier: persona{nombre:"Javier", edad:40}),

(pedro: persona{nombre:"Pedro", edad:20}),

(ana: persona{nombre:"Ana", edad:18}),

(jorge: persona{nombre:"Jorge", edad:23}),

(antonio: persona{nombre:"Antonio", edad:60})

**2.- Crear las relaciones**

create (maria: persona{nombre:"María", edad: 27}), (juan: persona{nombre:"Juan", edad:35}),

(javier: persona{nombre:"Javier", edad:40}),

(pedro: persona{nombre:"Pedro", edad:20}),

(ana: persona{nombre:"Ana", edad:18}),

(jorge: persona{nombre:"Jorge", edad:23}),

(antonio: persona{nombre:"Antonio", edad:60}),

(javier)-[:conoce]->(juan),

(javier)-[:conoce]->(antonio),

(javier)-[:conoce]->(maria),

(antonio)-[:conoce]->(jorge),

(maria)-[:conoce]->(ana),

(maria)-[:conoce]->(pedro)

**3.- Muestra todos los amigos de Javier.**

**match (javier{nombre:"Javier"})-[:conoce]->(persona)**

**return persona**

**4.- Muestra los conocidos indirectos.**

**match (javier{nombre:"Javier"})-[:conoce\*2]->(persona)**

**return persona**

**5.- Muestra los conocidos directos e indirectos.**

**match (javier{nombre:"Javier"})-[:conoce\*1..2]->(persona)**

**return persona**

**6.- Muestra los conocidos a partir del nivel 1.**

**match (javier{nombre:"Javier"})-[:conoce\*]->(persona)**

**return persona**

**7.- Muestra la media de edad de todas las personas.**

**MATCH (PERSONA)RETURN AVG(PERSONA.edad)**

**8.- Muestra una lista con todas las edades**

**match (PERSONA) return COLLECT (PERSONA.edad)**

**9.- Utilizando el operador especial label, muestra la etiqueta, el nodo raíz y contar la cantidad de personas que conoce Javier. No sabemos el nombre de la relación ni a donde apunta.**

match (javier{nombre:"Javier"})-[r]->()

return labels(javier),count(\*)

**10.- Utilizando el operador especial type, cuenta las relaciones agrupadas por le tipo de las mismas.**

match (javier{nombre:"Javier"})-[r]->()

return type(r),count(\*)

**11.- Crea una relación nueva (con un nombre diferente) entre Javier y otro nodo.**

match (javier{nombre:"Javier"}),(maria{nombre:"María"})

create (javier)-[:Hermanos]->(maria)

**12.- Repite ahora el ejercicio 8.**

**Calcula la edad mínima, la máxima y la suma de las edades.**

match (p:persona)

return min(p.edad) as Edad,max(p.edad),sum(p.edad)

# Evaluable

create (pedro: persona{nombre:"Pedro", edad: 45}), (marta: persona{nombre:"Marta", edad:27, ciudad:"Cáceres"}),

(pepe: persona{nombre:"Pepe", edad:25}),

(eva: persona{nombre:"Eva", edad:33, ciudad:"Badajoz"}),

(elisa: persona{nombre:"Elisa", edad:34, ciudad:"Badajoz"}),

(sergio: persona{nombre:"Sergio", edad:26}),

(celia: persona{nombre:"Celia", edad:20}),

(pedro)-[:Es\_amigo{desde:"31/12/2021"}]->(marta),

(pedro)-[:Es\_amigo]->(eva),

(pedro)-[:Es\_amigo]->(pepe),

(pepe)-[:Es\_amigo]->(sergio),

(eva)-[:Es\_amigo]->(marta),

(marta)-[:Es\_amigo]->(eva),

(marta)-[:Es\_amigo]->(elisa),

(sergio)-[:Es\_amigo]->(celia)

1 y 2.-

create (carmen:persona{nombre:"Carmen", edad: 62, ciudad:"Badajoz"}),

(carlos:persona{nombre:"Carlos", edad: 63, ciudad: "Badajoz"}),

(jose:persona{nombre:"José", edad: 42, ciudad: "Badajoz"}),

(jesus:persona{nombre:"Jesús", edad: 35, ciudad: "Badajoz"}),

(jose)-[:Es\_hermano]->(jesus),

(jesus)-[:Es\_hermano]->(jose),

(carmen)-[:Es\_madre]->(jesus),

(carmen)-[:Es\_madre]->(jose),

(carlos)-[:Es\_padre]->(jesus),

(carlos)-[:Es\_padre]->(jose)

3.-

merge (pedro{nombre:"Pedro"}) set pedro.ciudad="Cáceres"

merge (pepe{nombre:"Pepe"}) set pepe.ciudad="Cáceres"

merge (sergio{nombre:"Sergio"}) set sergio.ciudad="Cáceres"

merge (celia{nombre:"Celia"}) set celia.ciudad="Badajoz"

4.-

match (persona{ciudad:"Badajoz"}) return persona

match (persona{ciudad:"Cáceres"}) return persona

5.-

match (n) detach delete n

create (pedro: persona{nombre:"Pedro", edad: 45}), (marta: persona{nombre:"Marta", edad:27, ciudad:"Cáceres", provincia:"Extremadura"}),

(pepe: persona{nombre:"Pepe", edad:25}),

(eva: persona{nombre:"Eva", edad:33, ciudad:"Badajoz", provincia:"Extremadura"}),

(elisa: persona{nombre:"Elisa", edad:34, ciudad:"Badajoz", provincia:"Extremadura"}),

(sergio: persona{nombre:"Sergio", edad:26}),

(celia: persona{nombre:"Celia", edad:20}),

(pedro)-[:Es\_amigo{desde:"31/12/2021"}]->(marta),

(pedro)-[:Es\_amigo]->(eva),

(pedro)-[:Es\_amigo]->(pepe),

(pepe)-[:Es\_amigo]->(sergio),

(eva)-[:Es\_amigo]->(marta),

(marta)-[:Es\_amigo]->(eva),

(marta)-[:Es\_amigo]->(elisa),

(sergio)-[:Es\_amigo]->(celia)

create (carmen:persona{nombre:"Carmen", edad: 62, ciudad:"Badajoz", provincia:"Extremadura"}),

(carlos:persona{nombre:"Carlos", edad: 63, ciudad: "Badajoz", provincia:"Extremadura"}),

(jose:persona{nombre:"José", edad: 42, ciudad: "Badajoz", provincia:"Extremadura"}),

(jesus:persona{nombre:"Jesús", edad: 35, ciudad: "Badajoz", provincia:"Extremadura"}),

(jose)-[:Es\_hermano]->(jesus),

(jesus)-[:Es\_hermano]->(jose),

(carmen)-[:Es\_madre]->(jesus),

(carmen)-[:Es\_madre]->(jose),

(carlos)-[:Es\_padre]->(jesus),

(carlos)-[:Es\_padre]->(jose)

6.-

match (eva{nombre:"Eva"})-[:Es\_amigo\*1..2]-(persona) return persona

7.-

match (pedro{nombre:"Pedro"})-[:Es\_amigo\*1..2]-(persona) return persona.nombre, persona.edad as nombre, persona

8)

match (pedro{nombre:”Pedro”}) -[r:Es\_amigo]->(marta{nombre:”Marta”})

return r.desde

9)

MATCH (madre)-[r:Es\_Madre]->(hijo)

return madre.nombre

10)

match (padre)-[:Es\_Padre]->(hijo)<-[:Es\_Madre]-(madre)

return hijo

# Cursos

1

)NODOS

CREATE (JAVA:CURSO {nombre:'Programación Java estándar', duracion:120, precio:80})

CREATE (ANGULAR:CURSO {nombre:'Angular', duracion:30, precio:110})

CREATE (SPRING:CURSO {nombre:'Spring', duracion:80, precio:200})

CREATE(PEPE:ALUMNO {nombre:'Pepe', edad:20})

CREATE(ANA:ALUMNO {nombre:'Ana', edad:40})

CREATE(ELENA:ALUMNO {nombre:'Elena', edad:34})

CREATE(MARIO:ALUMNO {nombre:'Mario', edad:19})

2)RELACION

MATCH (PEPE:ALUMNO {nombre:'Pepe', edad:20}) MATCH (JAVA:CURSO {nombre:'Programación Java estándar', duracion:120, precio:80}) CREATE (PEPE) -[:CURSA{HORARIO:[‘Mañana’]}]->(JAVA)

MATCH (PEPE:ALUMNO {nombre:'Pepe', edad:20}) MATCH (ANGULAR:CURSO {nombre:'Angular', duracion:30, precio:110}) CREATE (PEPE)-[:CURSA{HORARIO:[‘Tarde’]}]->(ANGULAR )

/////////

CREATE (PEPE)-[:Horario\_Mañana]->(JAVA)

CREATE (PEPE)-[:Horario\_Tarde]->(ANGULAR)

CREATE (ELENA)-[:Horario\_Tarde]->(JAVA)

CREATE (ANA)-[:Horario\_Mañana]->(ANGULAR)

CREATE (MARIO)-[:Horario\_Manaña]->(SPRING)

1. VICTOR

##Creacion de alumnos##

CREATE (Pepe : Alumno {name: "Pepe", age: 20})

CREATE (Elena : Alumno {name: "Elena", age: 34})

CREATE (Ana : Alumno {name: "Ana", age: 40})

CREATE (Mario : Alumno {name: "Mario", age: 19})

##Creacion de Cursos##

CREATE (Java: Curso {name:"Java",duracion: 120,precio:80})

CREATE (Angular:Curso {name:"Angular",duracion: 30,precio:110})

CREATE (Spring:Curso {name:"Spring",duracion: 80,precio:200})

##Creacion de Relaciones##

create(Pepe)-[b:Mañana]->(Java)

create(Elena)-[o:Tarde]->(Java)

create(Pepe)-[h:Tarde]->(Angular)

create(Ana)-[k:Mañana]->(Angular)

create(Mario)-[l:Mañana]->(Spring)

3.

1. **Mostrar el nombre de los estudiante que hacen cursos:**

match (Alumno) -[:realiza]->(curso)

return Alumno.nombre

1. **Buscar el curso de spring**

match (curso{nombre:"Spring"})

return curso

1. **Mostrar los cursos que realiza pepe**

match (alumno{nombre:”Pepe”}) - [:realiza] -> (curso)

return curso

1. **Mostrar el nombre de los alumnos que realizan el curso de angular**

match (Alumno) -[:realiza]->(curso{nombre:”Angular”})

return Alumno.nombre

1. **¿Y si queremos saber la cantidad?**

match (Alumno) -[:realiza]->(curso{nombre:”Angular”})

return count (Alumno) as Numero\_Alumnos

1. **Modificar la edad de Ana de 40 a 25 años. Mostrar la persona que tiene la edad =25**

merge (ana{nombre:”Ana”}) set ana.edad=25

return ana

match (alumno{edad:25})

return alumno

1. **Damos de baja a Mario.**

match (mario{nombre:”Mario”})

detach delete mario

**8. Borrar la relacion entre ana y el curso Angular**

match(alumno{nombre:”Ana”}) - [r:realiza] -> (curso{nombre:”angular”})

delete r

**9. Eliminamos el árbol**

match (n)

detach delete n

Teoría

Vamos a crear un nodo en el que indicamos que Silvia es profesora tiene 30 años y es mujer. Lo haremos de la siguiente forma:

CREATE ( “etiqueta del nodo (SILVIA) : Tipo de Nodo (TEACHER) + información del nodo en formato JSON (NAME, AGE, SEX))

**CREATE (Silvia : teacher {name: "Silvia Pérez", age: 30, sex: "M"})**

Podemos ver que se ha creado ejecutando:

**match (n) return n.**

Creamos ahora un nodo España de tipo country

**CREATE (ES:country {name:"España"})**

Llegados a este punto tenemos dos nodos diferentes que podemos mostrar de nuevo con:

**match (n) return n.**

Vamos a crear una relación nacionalidad entre los dos nodos, de forma que indicamos que la nacionalidad de Silvia es España.

**create(Silvia)-[r:nacionalidad]->(ES)**

**return Silvia,ES**

Vemos que ahora no se muestra todo. Es porque estamos ejecutándolo en un terminal diferente. Para verlo todo o bien lo ejecutamos todo seguido o bien hacemos delante un match.

Para borrar todo ejecutamos

**match (n) detach delete n**

A continuación vamos a crearlo todo en el mismo terminal.

**CREATE (Silvia : teacher {name: "Silvia Pérez", age: 30, sex: "M"})**

**CREATE (ES:country {name:"España"})**

**create(Silvia)-[r:nacionalidad]->(ES)**

**return Silvia,ES**

Vamos a crear ahora un nodo idioma Español

**create (idioma:languaje {lang: "Español"})**

Podemos crear una relación mediante la cláusula CREATE. Especificaremos la relación dentro de los corchetes "[ ]" dependiendo de la dirección de la relación que se coloca entre el guión " - " y la flecha " ->".

Para crear la relación entre el nodo silvia y el nodo español en primer lugar debemos mostrarlos para que no nos suceda lo mismo que nos pasó anteriormente. Ejecutamos:

**match (p:teacher),(i:languaje) where p.name="Silvia Pérez" and i.lang="Español"**

**create (p)-[r:habla]->(i)**

**return p,i**

Mostramos todos.

Ahora vamos a crear un nodo diferente. De tipo curso

**create (java:course{name:"Programación en Java", duration:20})**

Vamos a crear una relación que nos indique que Silvia es está matriculada del curso de programación en Java que acabamos de crear.

**match (t:teacher),(c:course) where t.name="Silvia Pérez" and c.name="Programación en Java"**

**Create (t)-[r:matricula{precio:2000}]->(c)**

**return t,c**

Vamos a buscar a la persona Silvia Pérez

**match (n{name:"Silvia Pérez"}) return n**

Vamos a mostrar los cursos en lso q ue está matriculada Silvia

**match (n{name:"Silvia Pérez"})-[:matricula]->(c)**

**return c**

Repetimos la búsqueda anterior, pero queremos que sólo nos muestre la duración del curso.

**match (n{name:"Silvia Pérez"})-[:matricula]->(c)**

**return c.duration.**

Vamos a crear más nodos para poder realizar consultas más elaboradas (ANALIZANDO EL CÓDIGO)

**create (tele:companies{name:"telefonica",years:20})**

**create (tele1:companies{name:"yoigo",years:3})**

**create (tele2:companies{name:"movistar",years:2})**

Y ahora creamos las relaciones entre ellos:

**match(t:teacher),(c:companies) where t.name="Silvia Perez" and c.name="telefonica"**

**create(t)-[r:trabaja{salario:1300}]→(c)**

**match(t:teacher),(c:companies) where t.name="Silvia Perez" and c.name="movistar"**

**create(t)-[r:trabaja{salario:1200}]->(c)**

**match(t:teacher),(c:companies) where t.name="Silvia Perez" and c.name="yoigo"**

**create(t)-[r:trabaja{salario:1500}]->(c)**

Lista de nodos

—----------------------------------------------------------------------------------------------------

NODOS

1. CURSO: Java, curso: ‘Programación Java estándar', duración: 120, precio: 80
2. CURSO: angular, curso: ‘Angular', duración: 30, precio: 110
3. CURSO: spring, curso:'Spring', duración: 80, precio: 200
4. ALUMNO: Pepe, nombre: ‘Pepe', edad: 20
5. ALUMNO: Ana, nombre: ‘Ana', edad: 40
6. ALUMNO: Elena, nombre: ‘Elena', edad: 34
7. ALUMNO: Mario, nombre: ‘Mario', edad: 19

Lista de relaciones

RELACIONES

1. Pepe realiza en horario de mañana el curso de Java
2. Pepe realiza en horario de tarde el curso de angular
3. Elena realiza en horario de tarde el curso de Java
4. Ana realiza en horario de mañana el curso de angular
5. Mario realiza en horario de mañana el curso de spring

Lista de búsquedas

BUSQUEDAS

1. Mostrar el nombre de los estudiante que hacen cursos:
2. Buscar el curso de spring
3. Mostrar los cursos que realiza pepe
4. Mostrar el nombre de los alumnos que realizan el curso de angular
5. ¿Y si queremos saber la cantidad?
6. Modificar la edad de Ana de 40 a 25 años. Mostrar la persona que tiene la edad =25
7. Damos de baja a Mario.
8. Eliminamos el árbol

# SOLUCIONES

CREATE (java: curso {nombre:"Programación Java estándar", duracion: 120, precio:80})

CREATE (angular: curso {nombre:"Angular", duracion: 30, precio:120})

CREATE (spring: curso {nombre:"Spring", duracion: 80, precio:200})

CREATE (pepe: alumno {nombre:"Pepe", edad:20})

CREATE (ana: alumno {nombre:"Ana", edad:40})

CREATE (elena: alumno {nombre:"Elena", edad:34})

CREATE (mario: alumno {nombre:"Mario", edad:19})

match (pepe:alumno), (java:curso) where pepe.nombre="Pepe" and java.nombre= "Programación Java estándar"

CREATE (pepe)-[r:realiza{turno:"Mañana"}]->(java)

return pepe, java

match (pepe:alumno), (angular:curso) where pepe.nombre="Pepe" and angular.nombre= "Angular"

CREATE (pepe)-[r:realiza{turno:"Tarde"}]->(angular)

return pepe, angular

match (elena:alumno), (java:curso) where elena.nombre="Elena" and java.nombre= "Programación Java estándar"

CREATE (elena)-[r:realiza{turno:"Tarde"}]->(java)

return elena, java

match (ana:alumno), (angular:curso) where ana.nombre="Ana" and angular.nombre= "Angular"

CREATE (ana)-[r:realiza{turno:"Mañana"}]->(angular)

return ana, angular

match (mario:alumno), (spring:curso) where mario.nombre="Mario" and spring.nombre= "Spring"

CREATE (mario)-[r:realiza{turno:"Mañana"}]->(spring)

return mario, spring

--------- OTRA FORMA --------

CREATE (java: curso {nombre:"Programación Java estándar", duracion: 120, precio:80})

CREATE (angular: curso {nombre:"Angular", duracion: 30, precio:120})

CREATE (spring: curso {nombre:"Spring", duracion: 80, precio:200})

CREATE (pepe: alumno {nombre:"Pepe", edad:20})

CREATE (ana: alumno {nombre:"Ana", edad:40})

CREATE (elena: alumno {nombre:"Elena", edad:34})

CREATE (mario: alumno {nombre:"Mario", edad:19})

CREATE (pepe)-[:realiza{turno:"Mañana"}]->(java)

CREATE (pepe)-[:realiza{turno:"Tarde"}]->(angular)

CREATE (elena)-[:realiza{turno:"Tarde"}]->(java)

CREATE (ana)-[:realiza{turno:"Mañana"}]->(angular)

CREATE (mario)-[:realiza{turno:"Mañana"}]->(spring)

return pepe, ana, elena, mario, spring, angular, java

---------------------- **BÚQUEDAS** —-----------------------------

**1.- Mostrar el nombre de los estudiante que hacen cursos:**

match (alumno) -[:realiza]-> (curso)

return alumno.nombre

**2.- Buscar el curso de spring**

match (curso{nombre:"Spring"})

return curso

match (c:curso) where c.nombre="Spring"

return c

Otra posibilidad

**3.- Mostrar los cursos que realiza pepe**

match (alumno{nombre:"Pepe"}) -[:realiza]-> (curso)

return curso

**4.- Mostrar el nombre de los alumnos que realizan el curso de angular**

match (alumno) -[:realiza]-> (curso{nombre:"Angular"})

return alumno.nombre

**5.- ¿Y si queremos saber la cantidad?**

match (alumno) -[:realiza]-> (curso{nombre:"Angular"})

return count (alumno) as Numero\_alumnos

**6.- Modificar la edad de Ana de 40 a 25 años. Mostrar la persona que tiene la edad =25**

merge (ana{nombre:"Ana"}) set ana.edad=25

return ana

match (alumno{edad:25})

return alumno

**7.- Damos de baja a Mario.**

match (mario{nombre:"Mario"})

detach delete mario

Borrar la relación entre Ana y el curso Angular

match (alumno{nombre:"Ana"}) -[r:realiza]-> (curso{nombre:"Angular"})

delete r

**8.- Borrar todo el grafo**

match (n) detach delete n

# EVALUABLE

CREATE (pedro: persona{Nombre:"Pedro", Edad:45}),

(marta: persona{Nombre:"Marta", Edad:27, Ciudad:"Cáceres"}),

(pepe: persona{Nombre:"Pepe", Edad:25}),

(eva: persona{Nombre:"Eva", Edad:23, Ciudad:"Badajoz"}),

(elisa: persona{Nombre:"Elisa", Edad:34, Ciudad:"Badajoz"}),

(sergio: persona{Nombre:"Sergio", Edad:26}),

(celia: persona{Nombre:"Celia", Edad:20})

CREATE (pedro)-[:es\_amigo{Desde:"31-12-2021"}]->(marta),

(pedro)-[:es\_amigo]->(eva),

(pedro)-[:es\_amigo]->(pepe),

(pepe)-[:es\_amigo]->(sergio),

(eva)-[:es\_amigo]->(marta),

(marta)-[:es\_amigo]->(eva),

(marta)-[:es\_amigo]->(elisa),

(sergio)-[:es\_amigo]->(celia)

**1.- Añade a Carmen con 62 años de Badajoz y Carlos de 63 años también de Badajoz,**

**José de 42 años de Badajoz y Jesús de 35 de Badajoz**

CREATE (carmen:persona{Nombre:"Carmen",Edad:62, Ciudad:"Badajoz"}),

(carlos:persona{Nombre:"Carlos",Edad:63, Ciudad:"Badajoz"}),

(jose:persona{Nombre:"José",Edad:42, Ciudad:"Badajoz"}),

(jesus:persona{Nombre:"Jesús",Edad:35, Ciudad:"Badajoz"})

**2.- Indica que José y Jesús son hermanos (“Es\_Hermano”) y que Carmen es la**

**madre de ambos (“Es\_Madre”) y Carlos es el padre de ambos (“Es\_Padre”)**

match (carmen:persona{Nombre:"Carmen"}),(carlos:persona{Nombre:"Carlos"}),

(jesus:persona{Nombre:"Jesús"}),(jose:persona{Nombre:"José"})

CREATE (carmen)-[:Es\_Madre]->(jose),

(carmen)-[:Es\_Madre]->(jesus),

(jose)-[:Es\_Hermano]->(jesus),

(jesus)-[:Es\_Hermano]->(jose),

(carlos)-[:Es\_Padre]->(jose),

(carlos)-[:Es\_Padre]->(jesus)

**3.- Indica que Pedro, Pepe y Sergio son de Cáceres y Celia es de Badajoz.**

merge (pedro{Nombre:"Pedro"}) set pedro.Ciudad="Cáceres"

merge (pepe{Nombre:"Pepe"}) set pepe.Ciudad="Cáceres"

merge (sergio{Nombre:"Sergio"}) set sergio.Ciudad="Cáceres"

merge (celia{Nombre:"Celia"}) set celia.Ciudad="Badajoz"

return pedro, pepe, sergio, celia

**4.- Muestra todas las personas que vivan en Badajoz y después todas**

**las personas que vivan en Cáceres.**

match (persona{Ciudad:"Badajoz") return persona

match (persona{Ciudad:"Cáceres") return persona

**5.- Indica para todos los que sean de Cáceres y Badajoz que pertenecen**

**a la Comunidad de Extremadura (Comunidad: “Extremadura”).**

match (mangurrinos{Ciudad:"Cáceres"}), (belloteros{Ciudad:"Badajoz"})

set mangurrinos.Comunidad="Extremadura", belloteros.Comunidad="Extremadura"

**6.- Recupera los amigos directos e indirectos de Eva.**

match (eva{Nombre:"Eva"})-[:es\_amigo\*1..2]->(persona)

return persona

**7.- Recupera el nombre y la edad de los amigos de Pedro**

match (pedro{Nombre:"Pedro"})-[:es\_amigo]->(persona)

return persona.Nombre, persona.Edad

**8.- Muestra desde cuando son amigos Pedro y Marta**

match (pedro{Nombre:"Pedro"})-[r:es\_amigo]->(marta{Nombre:"Marta"})

return r.Desde

**9.- Muestra todas las personas que sean madre.**

match (madre)-[r:Es\_Madre]->(hijo)

return madre.nombre

**10.- Muestra a todas las personas que tengan asignados a alguno de sus**

**progenitores.**

match (padre)-[:Es\_Padre]->(hijo)<-[:Es\_Madre]-(madre)

return hijo

# NOTICIAS

Vamos a diseñar un blog de noticias donde los usuarios puedan publicar sus comentarios. Necesitamos los siguientes tres tipos de nodos: usuario, noticia.

Los datos a insertar son:

Usuario -> {nombre: ‘Miguel’, nombreUsuario: ‘Miguel\_u’, cuenta\_twitter: ‘miguelete93’}

Usuario ->{nombre: ‘Laro’, nombreUsuario: ‘Laro\_u’, cuenta\_twitter: ‘larocantabro85’, telefono: ‘1234567890’}

Noticia -> {titulo: ‘Noticia1Laro’, cuerpo: ‘lorem ipsum…’, dia: 22, mes: 5, anio: 2017}

Noticia -> {titulo: ‘Noticia2Laro’, cuerpo: ‘lorem ipsum…2’, dia: 2, mes: 3, anio: 2015}

Noticia -> {titulo: ‘NoticiaMiguel’, cuerpo: ‘lorem ipsum…3’, dia: 15, mes: 3, anio: 2017}

Las relaciones serán Redacta y Puntúa:

Laro redacta las noticias Noticia1Laro y Noticia2Laro

Miguel redacta la noticia NoticiaMiguel

Laro puntúa con un 3 la noticia de Miguel

Miguel puntúa con 1 la noticia 1 de Laro y con un 5 la noticia 2.

Realiza las siguientes consultas:

Devolver todos los usuarios

Devolver las noticias cuyo título termine en o

Devolver las noticias publicadas entre los años 2015 y 2017

Devolver los datos de las relaciones puntúa con puntuación 4,

Devolver los datos de todas las puntuaciones que ha hecho Miguel

Devolver las cuentas de twitter de todos los usuarios que hayan valorado noticias de 2017

Devolver el número de contenidos publicados por Laro

Devolver los datos de las relaciones redacta de las noticias publicadas en 2017 cuyo autor se llame “Laro”.

# Solución

CREATE (laro:usuario{nombre: "Laro", nombreUsuario: "Laro\_u", cuenta\_twitter: "larocantabro85", telefono: "1234567890"}),

(miguel:usuario{nombre: "Miguel", nombreUsuario: "Miguel\_u", cuenta\_twitter: "miguelete93"}),

(not1l:noticia{titulo: "Noticia1Laro", cuerpo: "lorem ipsum…", dia: 22, mes: 5, anio: 2017}),

(not2l:noticia{titulo: "Noticia2Laro", cuerpo: "lorem ipsum…2", dia: 2, mes: 3, anio: 2015}),

(notmig:noticia{titulo: "NoticiaMiguel", cuerpo: "lorem ipsum…3", dia: 15, mes: 3, anio: 2017})

CREATE (laro)-[:redacta]->(not1l), (laro)-[:redacta]->(not2l),

(miguel)-[:redacta]->(notmig),

(laro)-[:puntua{Puntuación:3}]->(notmig),

(miguel)-[:puntua{Puntuación:1}]->(not1l),

(miguel)-[:puntua{Puntuación:5}]->(not2l)

**1.- Devolver todos los usuarios**

match (u:usuario) return u

**2.- Devolver las noticias cuyo título termine en o**

match (n:noticia) WHERE n.titulo ENDS WITH "o" return n.titulo

**3.- Devolver las noticias publicadas entre los años 2015 y 2017**

match (n:noticia) WHERE n.anio>=2015 AND n.anio<=2017 return n

**4.- Devolver los datos de las relaciones puntúa con puntuación 5**

match (usuario)-[:puntua{Puntuación:5}]->(noticia) return usuario,noticia

match (usuario)-[r:puntua]->(noticia) WHERE r.Puntuación= 5

return usuario,noticia

**5.- Devolver los datos de todas las puntuaciones que ha hecho Miguel**

match (miguel{nombre:"Miguel"})-[r:puntua]->(noticia)

return r.Puntuación, miguel.Nombre, noticia.titulo

**6.- Devolver las cuentas de twitter de todos los usuarios que hayan valorado**

**noticias de 2017**

match (usuario)-[:puntua]->(noticia) WHERE noticia.anio=2017 return usuario.cuenta\_twitter

match (usuario)-[:puntua]->(noticia{anio:2017}) return usuario.cuenta\_twitter

**7.- Devolver el número de contenidos publicados por Laro**

match (laro{nombre:"Laro"})-[:redacta]->(noticia)

return count (noticia) as numero\_noticias

**8.- Devolver los datos de las relaciones redacta de las noticias publicadas**

**en 2017 cuyo autor se llame “Laro”.**

match (laro{nombre:"Laro"})-[:redacta]->(noticia{anio:2017})

return noticia, laro