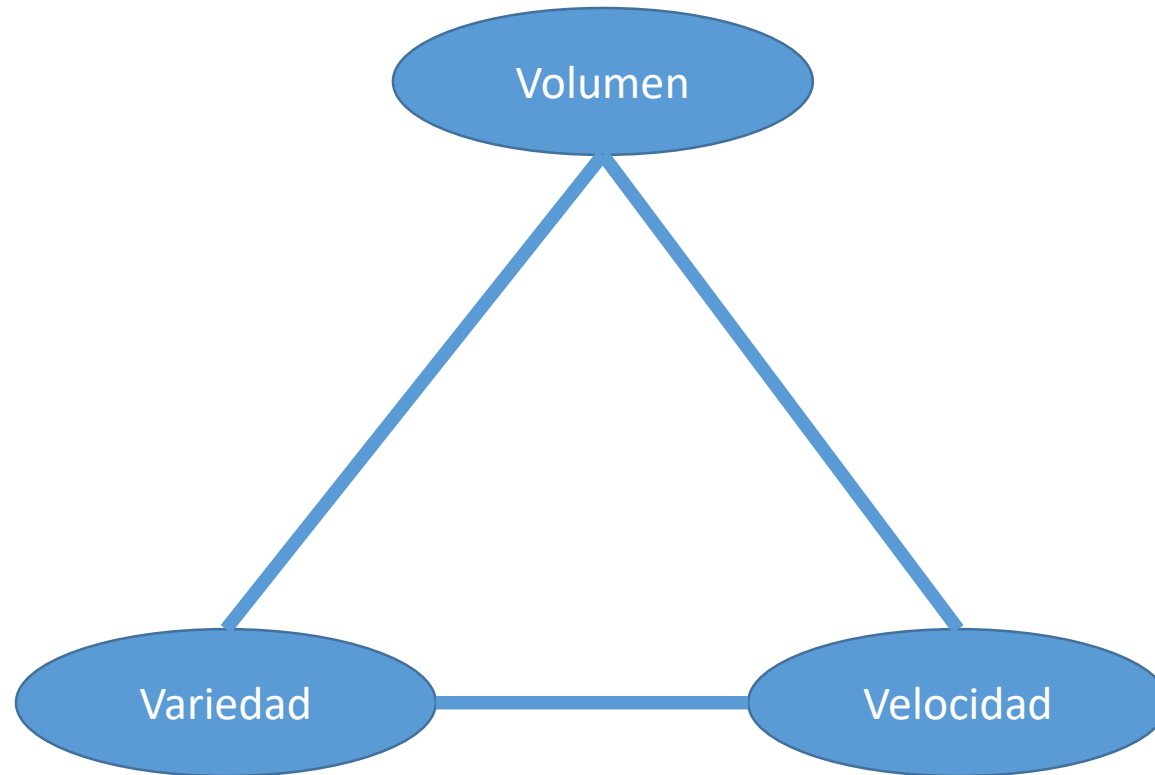


Diplomatura en Big Data

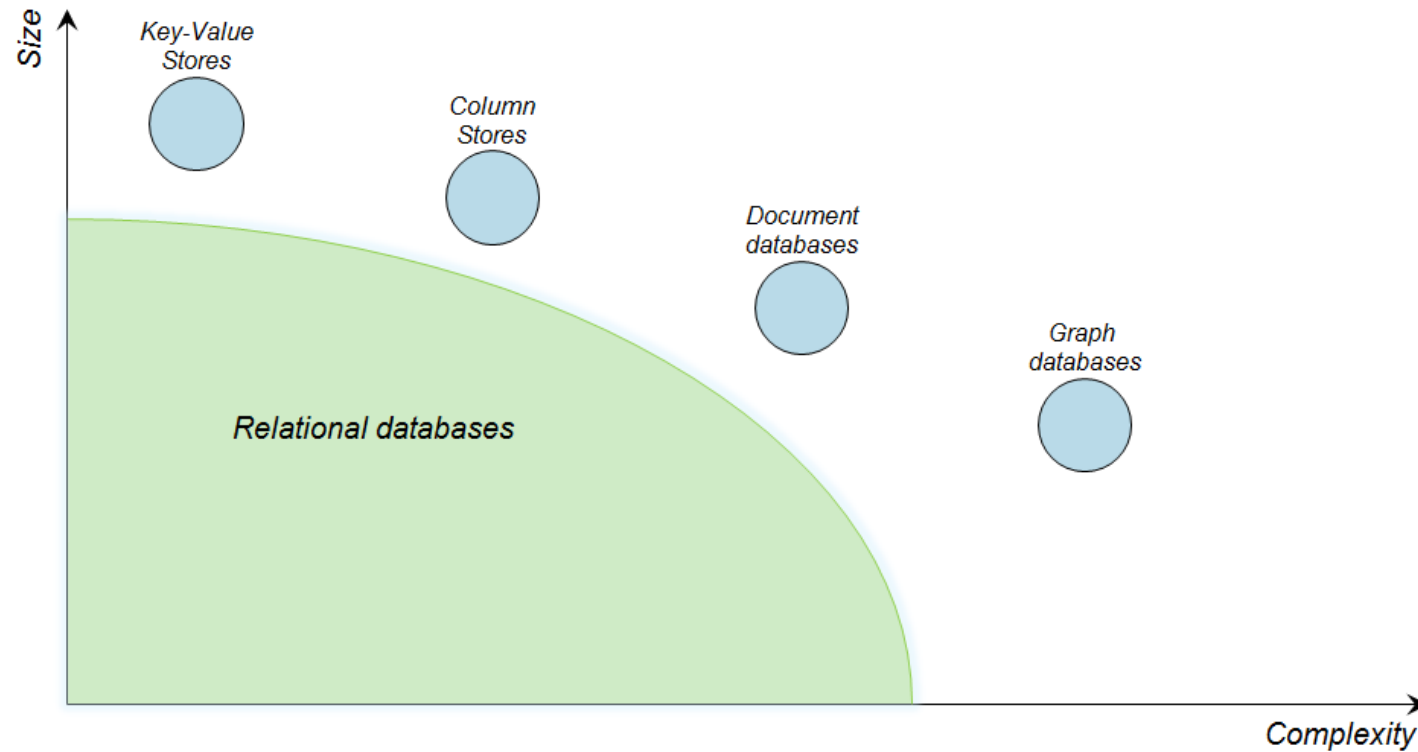
Bases de Datos de Grafos

Alejandro Vaisman
avaisman@itba.edu.ar

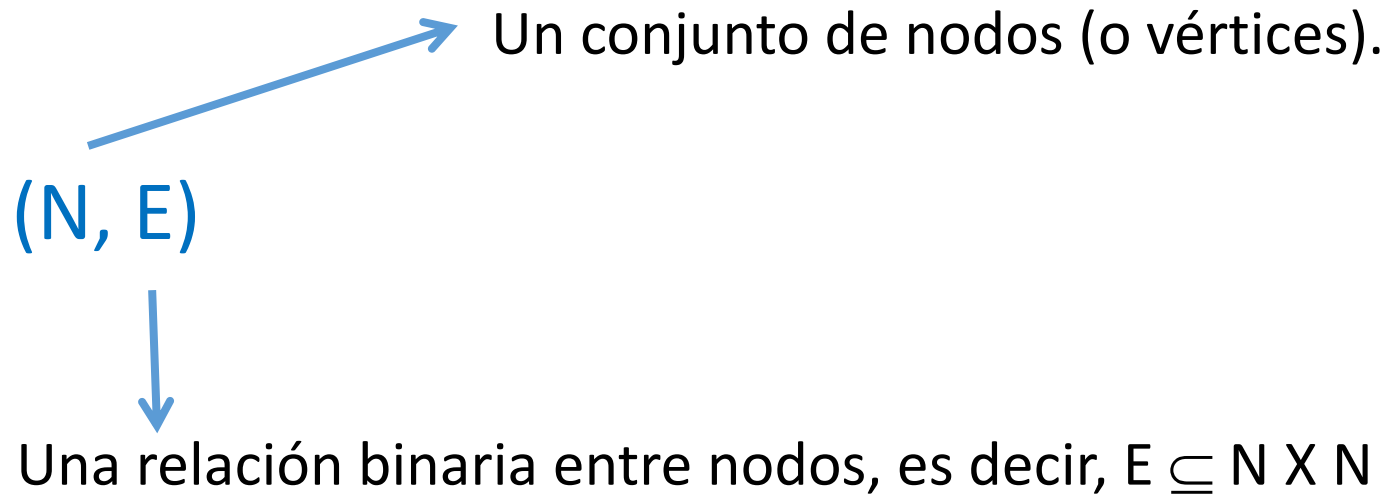
Las 3 V's



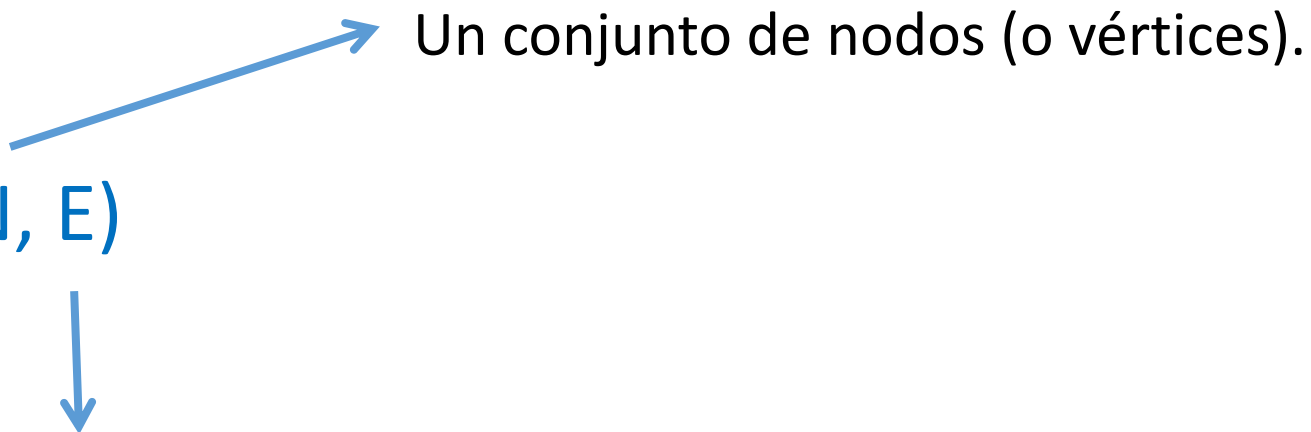
Bases de datos NoSQL



¿Qué es un Grafo?



¿Qué es un Grafo?

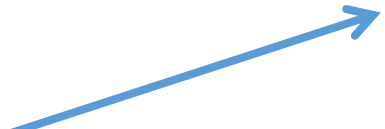
(N, E) 

Un conjunto de nodos (o vértices).

Una relación binaria entre nodos, es decir, $E \subseteq N \times N$

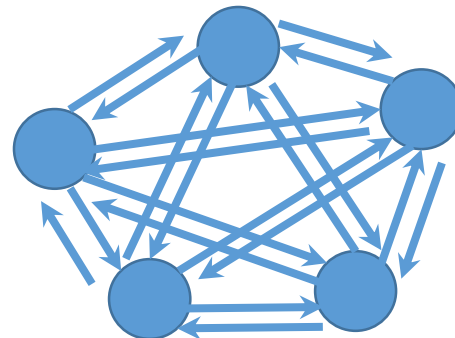
Cada uno de los elementos de esta relación se denomina eje (o arco dirigido).

¿Qué es un Grafo?

(N, E)  Un conjunto de nodos (o vértices).

 Una relación binaria entre nodos, es decir, $E \subseteq N \times N$

Ejemplo con 5 nodos:



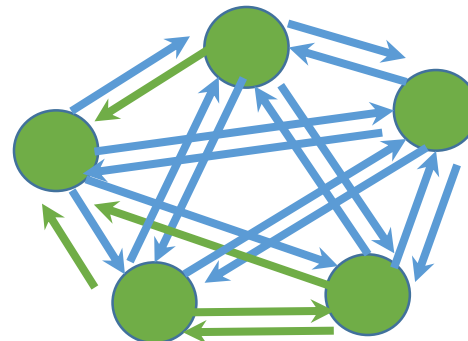
¿Qué es un Grafo?

(N, E) → Un conjunto de nodos (o vértices).

↓

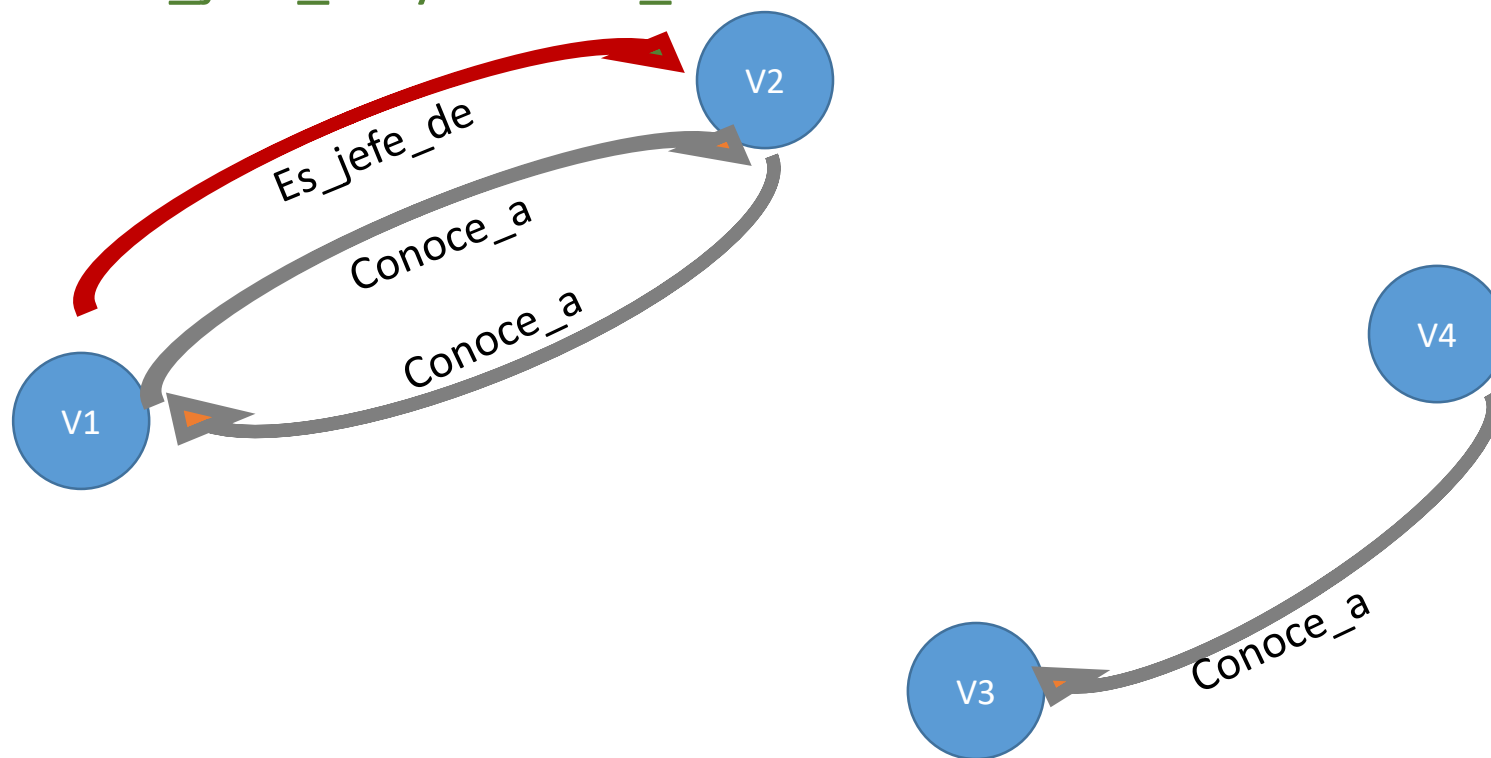
Una relación binaria entre nodos, es decir, $E \subseteq N \times N$

Ejemplo con 5 nodos:



Grafos

Ejemplo: Este grafo modela relaciones entre personas de una organización. Los nodos representan las personas y los ejes representan 2 posibles relaciones por medio de los siguientes 2 rótulos: `es_jefe_de` y `conoce_a`



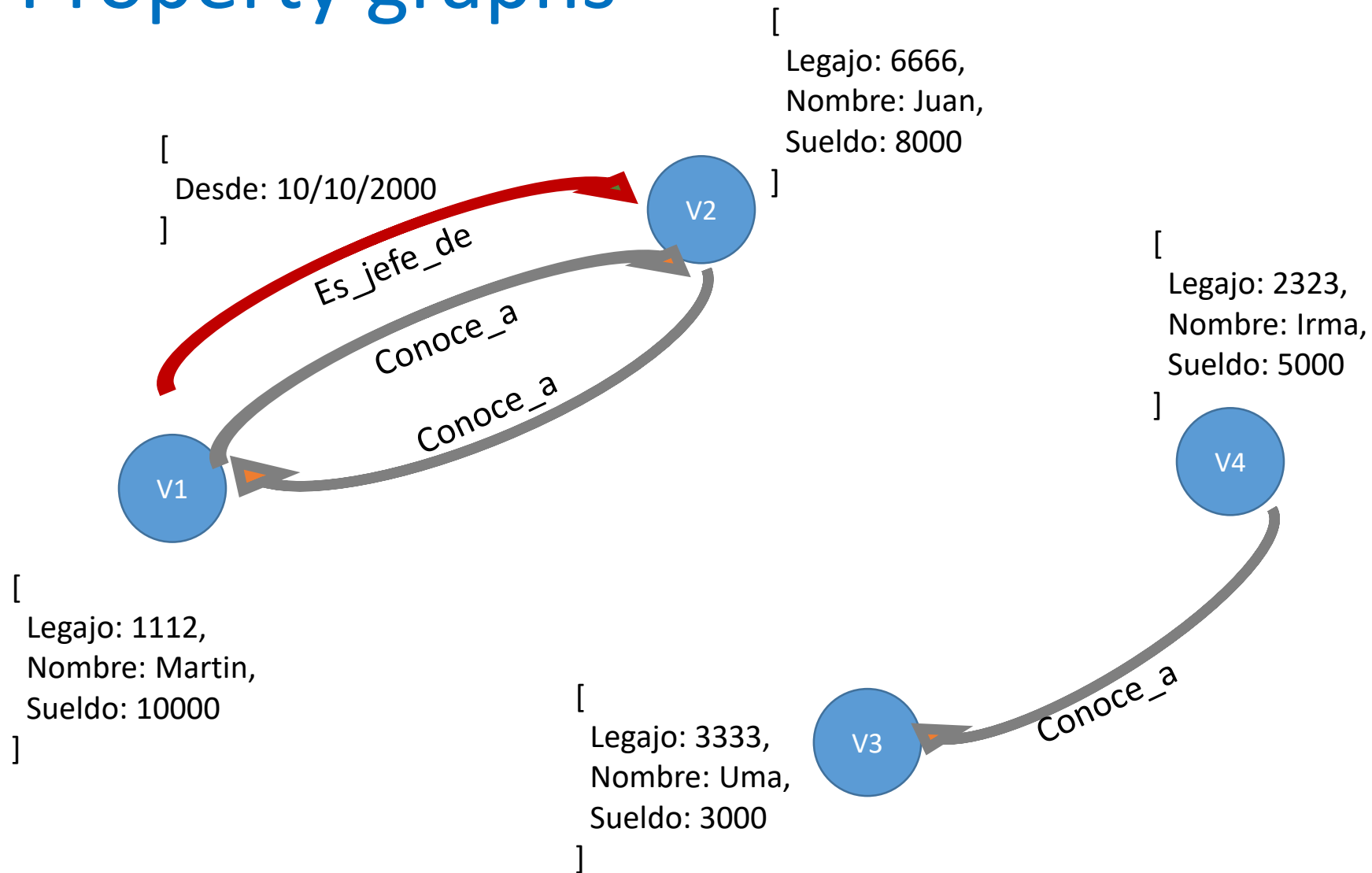
Grafos

Tanto a los nodos como a los ejes se los puede caracterizar por información adicional expresada por «propiedades».

Cada propiedad es un par dado por un «nombre» (o key) y el «valor asociado» a la misma.

Grafos de este tipo se denominan “property graphs”

Property graphs



Base de datos de grafos

1) **A nivel repositorio:** Se almacenan grafos.

«Variedad» de Big Data

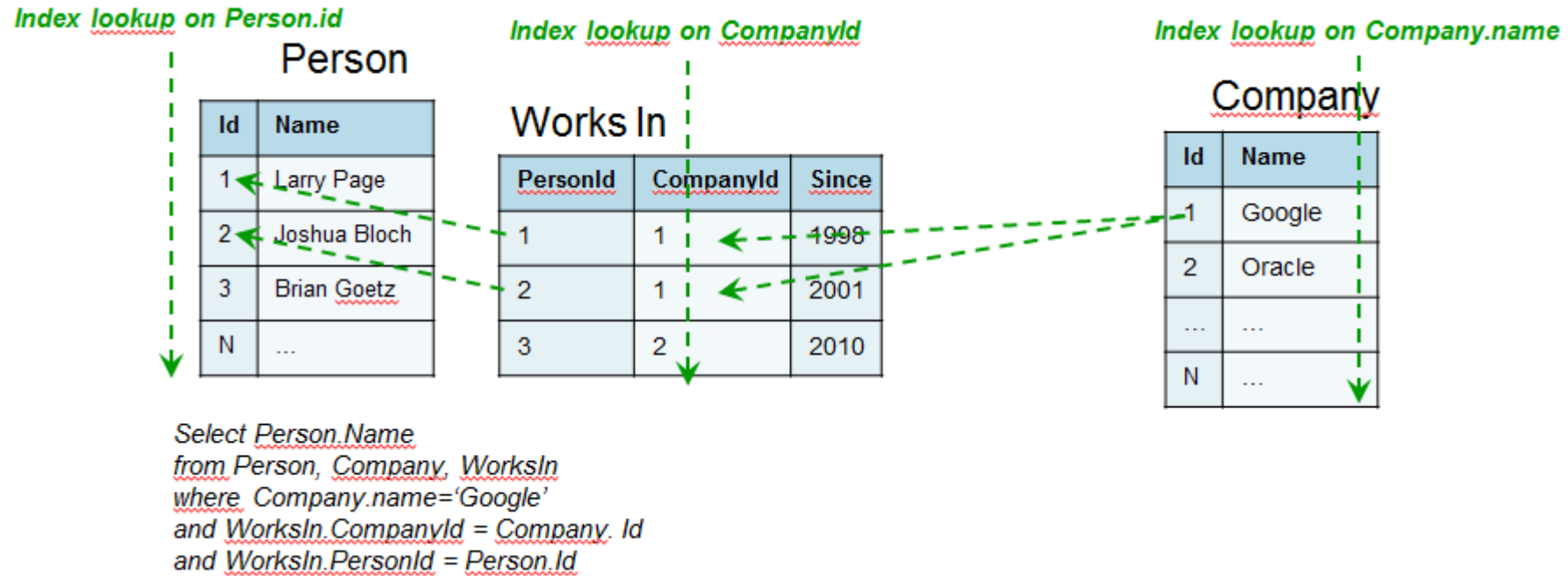
2) **A nivel transaccional:** Se define algún tipo de propiedades ACID (**A**tomicity-**C**onsistency-**I**solation-**D**urability) para las transacciones.

3) **A nivel lenguaje:** Se dispone de algún lenguaje de consulta de alto nivel orientados a grafos, lo cuales **no está estandarizados hasta el momento** (a diferencia de SQL). En general, también poseen API REST y se pueden acceder programáticamente desde algún lenguaje de programación. Manejan estrategias de optimización.

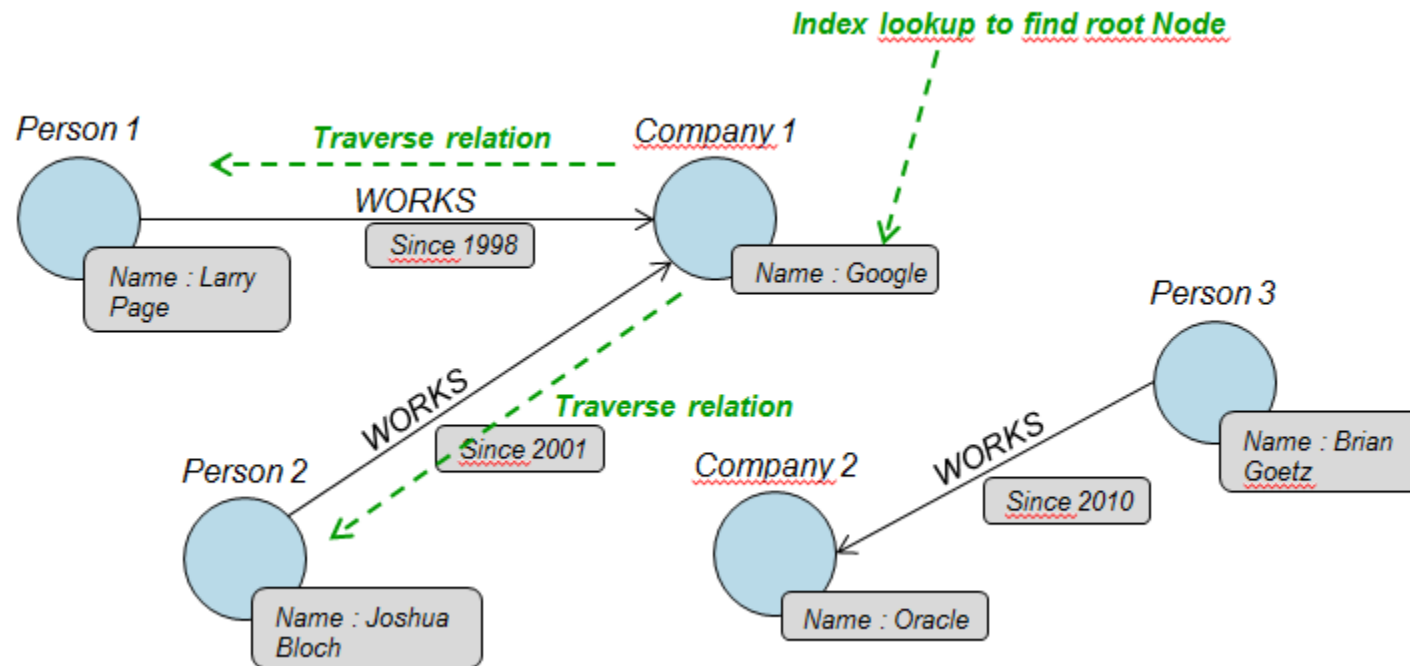
4) **A nivel arquitectura:** funcionan en clusters de nodos de computadoras y deben escalar horizontalmente. Permiten manejar tolerancia a fallas.

«Volumen» de Big Data

Consulta típica en SQL



La misma consulta sobre grafos



A mayor profundidad de navegación, mayor diferencia de performance con las BDR

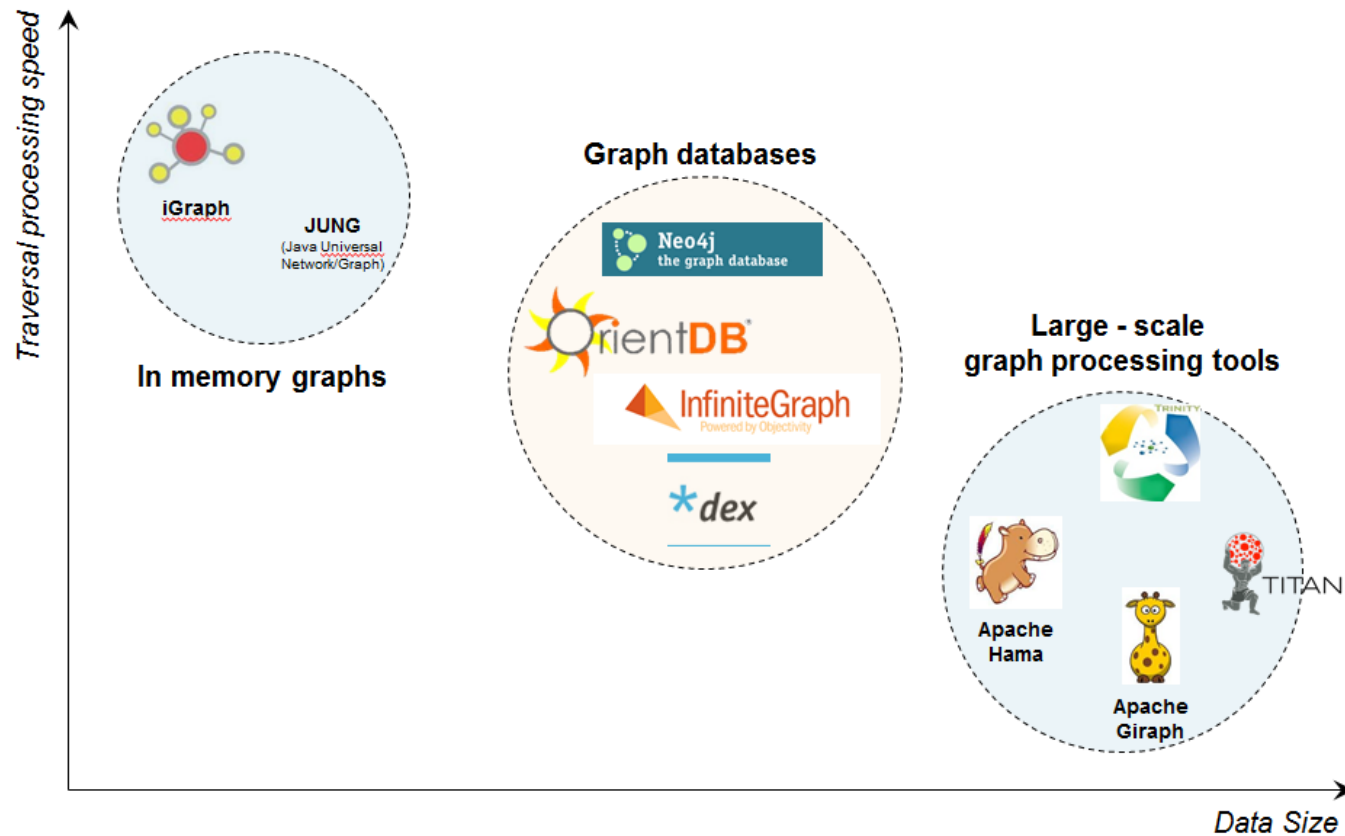
Graph storage vs graph processing

- Graph databases <> large-scale graph processing frameworks (e.g., Pregel de Google)
- Representan datos de la misma forma pero
 - graph processing tools se centran en la explotación
 - graph databases se enfocan en almacenamiento y transacciones
- Graph databases escalan verticalmente
- Pregel ejecuta procesamiento distribuido en commodity servers
- Entonces tenemos dos familias de productos distintos:
 - Una BDG va a ser adecuada para navegar el grafo, o computar shortest paths
 - Un graph processing framework sirve para resolver problemas de clustering, p. ej.

Graph storage vs graph processing

- Problema de GBD: es difícil particionar un grafo, sobre todo en tiempo real.
- No obstante, se suele decir que para volúmenes razonables, típicos de las organizaciones, funcionan bien
- Cada problema requiere un estudio de las soluciones disponibles

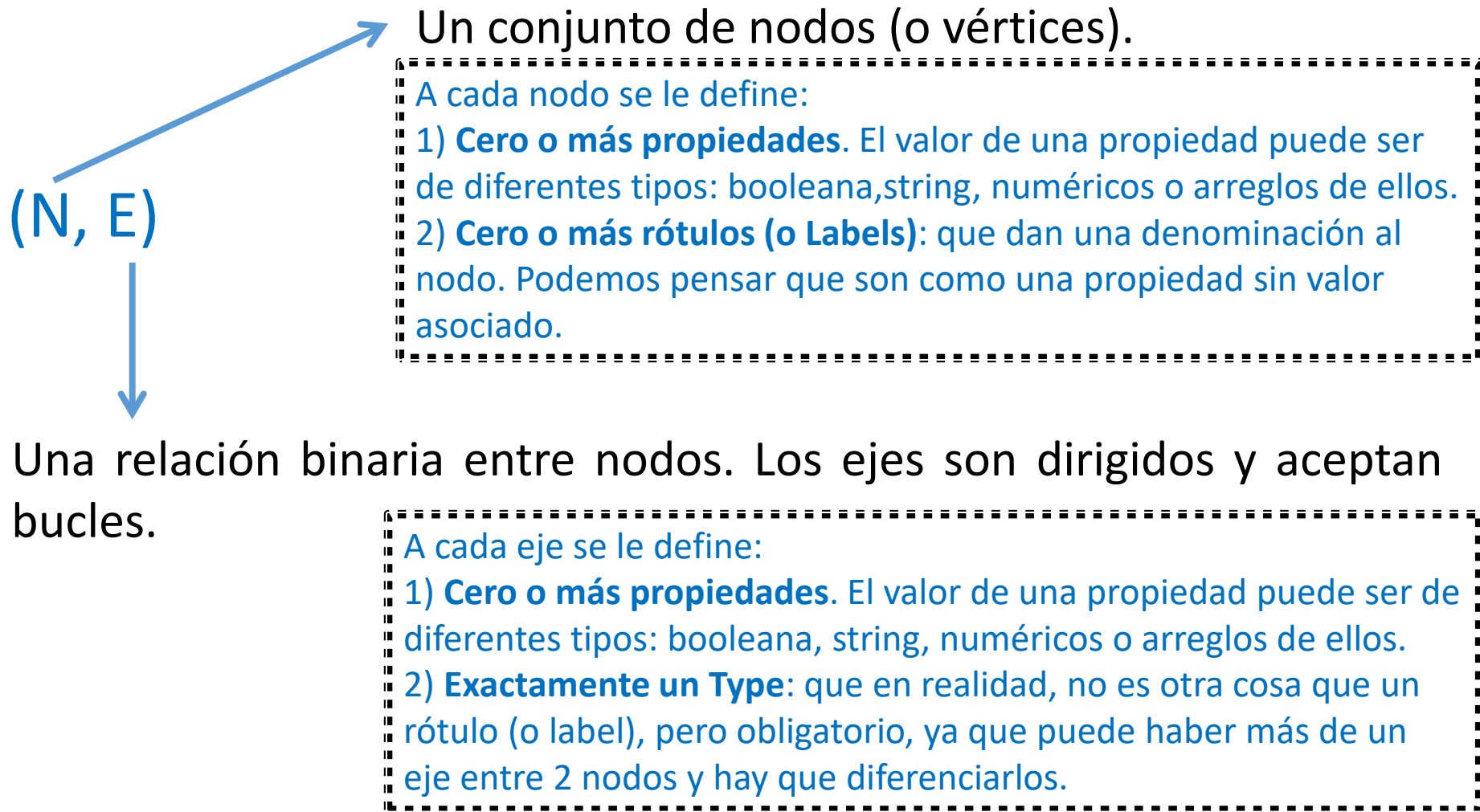
Comparación



Base de datos de grafos: Neo4j www.neo4j.com

- Es Open Source.
- Tiene versiones para Linux, Win, Mac. Está implementada en Java.
- Su lenguaje de consulta de alto nivel se llama Cypher.
- Tiene clientes como: Lufthansa, LinkedIn, InfoJobs (para la red de búsqueda de trabajo), gameSys (para redes sociales), eBay (para sus rutas), FiftyThree (para recomendaciones), Accenture, National Geographic, CISCO, HP, Telenor, etc.
- Se puede hacer OLAP sobre una BD de Grafos?

Grafo en Neo4j



Cypher

Para crear los grafos: nodos y
ejes con sus propiedades.

Para modificar o borrar esa
información.

Para consultar los grafos: expresando
condición informacional y/o
información topológica

Cypher

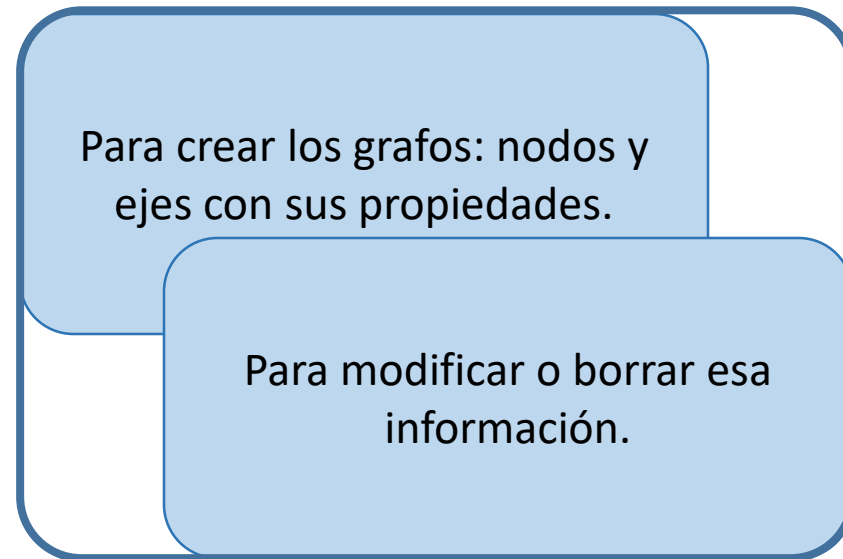
Para crear los grafos: nodos y
ejes con sus propiedades.

Para modificar o borrar esa
información.

Muy diferente al Modelo Relacional (y SQL) donde:

- 1) Primero se crea la estructura (o sea la tabla) que almacenará sus (homogéneas) componentes (las tuplas).
- 2) Las FKs se definen a nivel estructural.
- 3) Después se insertan, actualizan o y borran tuplas.

Cypher



Directamente se crean «nodos sueltos» o «ejes» (si es que antes se crearon los nodos que vinculan) o «camino entre nodos y ejes». Las propiedades, rótulos y Types son la estructura informacional.

La topología que «podría pensarse como» el rol de la clave foránea en el modelo relacional, directamente se define entre «2 nodos en particular», o sea, a través de los datos (al definir un eje dirigido).

Después puede cambiarse cualquiera de las 2 estructuras: informacional o topológica.

DependeDirectamenteDe	
Jefe	Empleado
A	C
A	D
C	E
C	M
E	T

DependeDe	
Jefe	Empleado
A	C
A	D
C	E
C	M
E	T
A	E
A	M
C	T
A	T

Iteración 1

Iteración 2

Iteración 3

De esta relación que representa “DependeDirectamenteDe” quiero calcular la clausura, o sea quien “DependeDe” en forma directa o indirecta. SQL tiene previsto este tipo de consulta recursivas.

```
WITH recursive DependeDe(Jefe, Empleado) AS
    (SELECT Jefe, Empleado
     FROM DependeDirectamenteDe
    UNION ALL
     SELECT calculado.jefe, origen.empleado
     FROM DependeDe as calculado, DependeDirectamenteDe as origen
     WHERE calculado.empleado = origen.jefe )
SELECT * FROM DependeDe
```

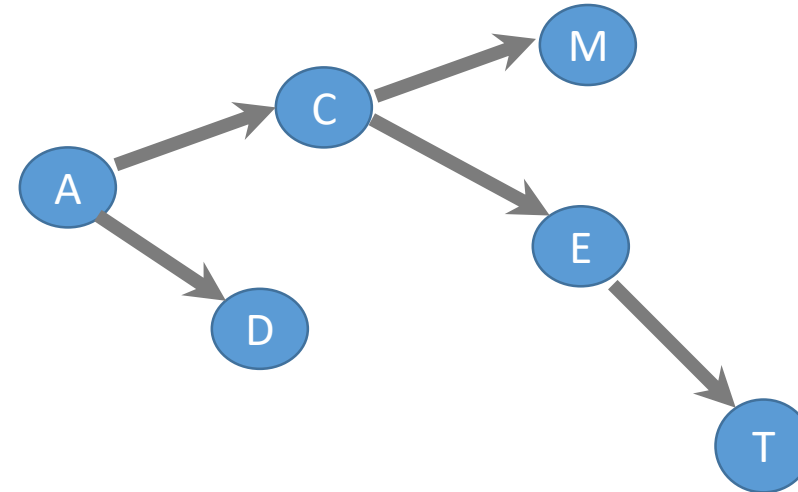
Este tipo de consultas del **Modelo Relacional** son de las más **COSTOSAS**, porque implica realizar **MÚLTIPLES JUNTAS (Joins)**.

Los Joins se expresan a nivel esquema (no a nivel instancia). Hay que realizar un producto cartesiano y descartar las tuplas que no satisfacen el predicado del **WHERE**.



Cómo sería la representación en el **Modelo De Grafos**?

DependeDirectamenteDe	
Jefe	Empleado
A	C
A	D
C	E
C	M
E	T



Los caminos en el grafo se expresan a nivel instancia (ya que no existe el esquema). No hay más que ver si desde un nodo hay eje saliente y usarlo!

Cypher - nodos

`(v :Rotulo1:Rotulo2...:RotuloN { Prop1: Value1, Prop2: Value2, ... Propk: Valuek })`

Se coloca la lista de K Propiedades (opcionales) que corresponden al nodo.
Cada propiedad está conformada por su nombre y valor, separados por el símbolo ":"
A su vez la lista de propiedades se coloca entre llaves y se separan con el símbolo ","

Se coloca la lista de los N rótulos (opcionales) que corresponden al nodo. Cada uno se prefija con el símbolo :

A un nodo se lo coloca entre los símbolos "()". Dentro debe obligatoriamente colocarse el nombre de una variable cualquiera que cumple el rol de identificarlo dentro de dicha expresión, con alcance es local, si se realiza return.

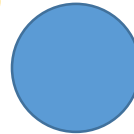
Cypher - nodos

Tenemos una nueva BD Neo4j.

Crear un nodo sin propiedades y sin rótulos:

```
$ CREATE (v)  
RETURN v;
```

<id>: 0



El ID lo asigna y usa internamente el DBMS con un número diferente en cada nodo/eje, pero no conviene usarlo desde las aplicaciones porque si borramos nodo/eje, podría reutilizarlo. No es un SERIAL.

Crear otro nodo sin propiedades y sin rótulos:

```
$ CREATE ();
```

Como no usé la cláusula «return», lo agrega al grafo pero no me lo devuelve.
Para ver lo que tengo hasta ahora, debería luego consultar el grafo y obtendría:

<id>: 0



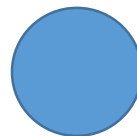
<id>: 1



Cypher - nodos

Crear otro nodo con 2 rótulos:

```
$ CREATE (v :Alumno:ITBA)  
RETURN v;
```



Alumno ITBA <id>: 2

Crear otro nodo con 1 rótulo y 3 propiedades:

```
$ CREATE (n :Alumno { Nombre: 'Juan Polo',  
                        FechaNac: '12/04/2000',  
                        Mails: ['jmpolo@itba.edu.ar', 'juan@yahoo.com'] })  
  
RETURN n;
```

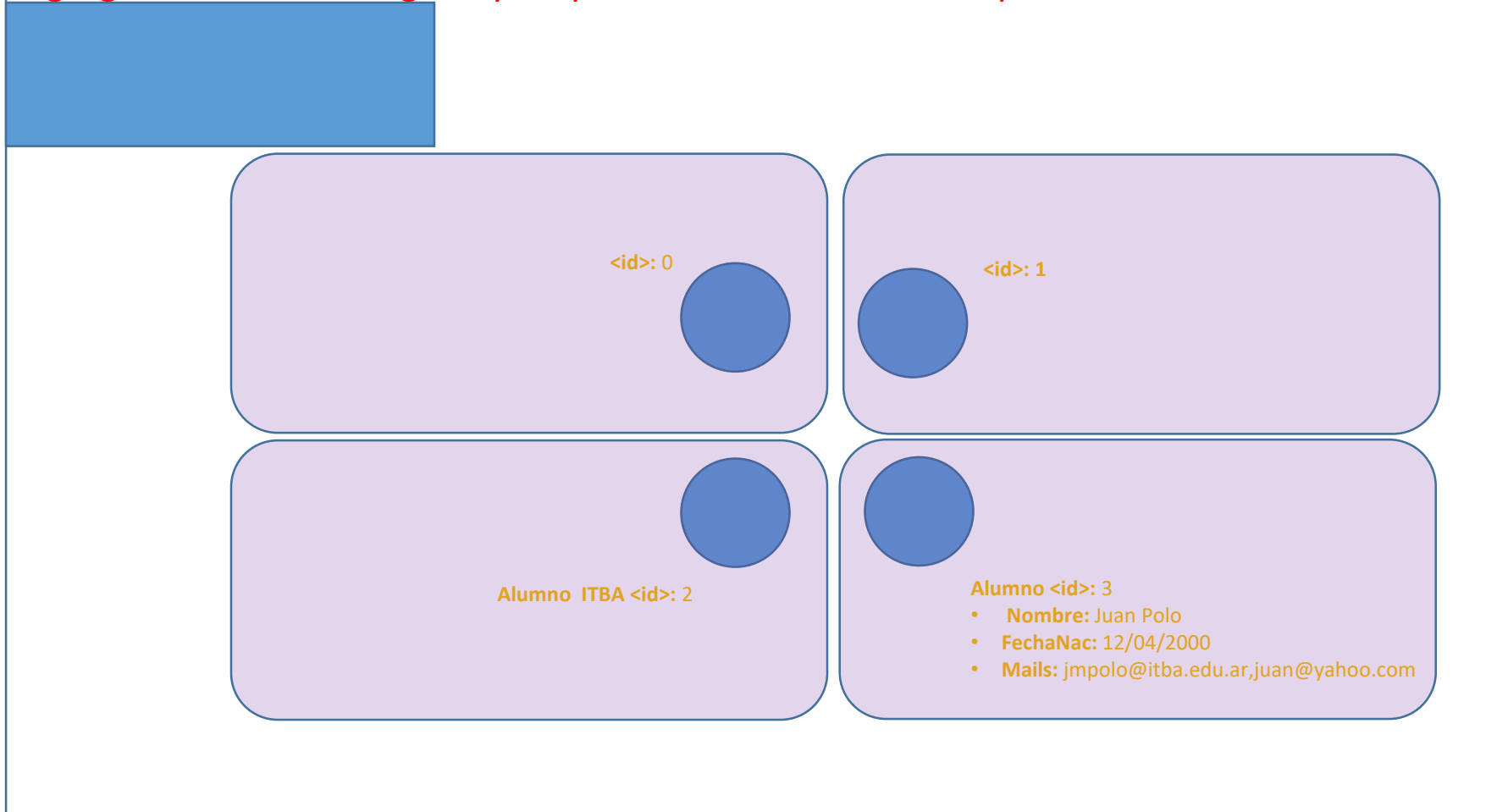


Alumno <id>: 3

- **Nombre:** Juan Polo
- **FechaNac:** 12/04/2000
- **Mails:** jmpolo@itba.edu.ar,juan@yahoo.com

Cypher - nodos

Agregar los rótulos “Inglés” y “Español” a todos los nodos previamente creados.



Cypher - nodos

Agregar los rótulos “Inglés” y “Español” a todos los nodos previamente creados.

```
$ MATCH (n)
```

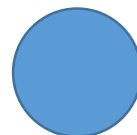
```
SET n :Inglés:Español
```

```
RETURN n;
```

Inglés Español <id>: 0



Inglés Español <id>: 1



Alumno ITBA Inglés Español <id>: 2



Alumno Inglés Español <id>: 3

- **Nombre:** Juan Polo
- **FechaNac:** 12/04/2000
- **Mails:** jmpolo@itba.edu.ar,juan@yahoo.com



Cypher - nodos

Eliminar los rótulos Inglés y Español solo al nodo con rótulo ITBA

\$

Inglés Español <id>: 0



Inglés Español <id>: 1



Alumno ITBA Inglés Español <id>: 2



Alumno Inglés Español <id>: 3

- **Nombre:** Juan Polo
- **FechaNac:** 12/04/2000
- **Mails:** jmpolo@itba.edu.ar,juan@yahoo.com



Cypher - nodos

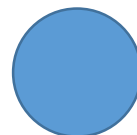
Eliminar los rótulos Inglés y Español solo al nodo con rótulo ITBA

```
$ MATCH (n :ITBA)  
  REMOVE n :Inglés:Español
```

Inglés Español <id>: 0



Inglés Español <id>: 1



Alumno ITBA <id>: 2



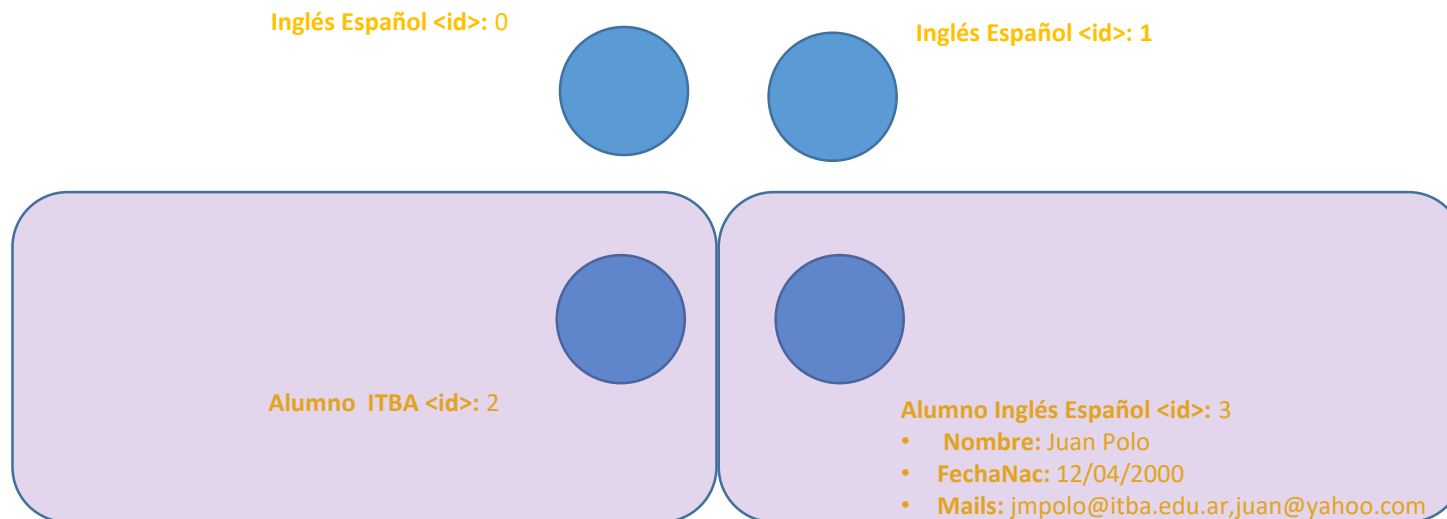
Alumno Inglés Español <id>: 3

- **Nombre:** Juan Polo
- **FechaNac:** 12/04/2000
- **Mails:** jmpolo@itba.edu.ar,juan@yahoo.com



Cypher - nodos

Borrar la propiedad FechaNac, Nombre y Edad de los nodos con rótulo Alumno.
Las propiedades se refieren con la expresión `nodo.nombrePropiedad`



Cypher - nodos

Borrar la propiedad FechaNac, Nombre y Edad de los nodos con rótulo Alumno.

Las propiedades se refieren con la expresión nodo.nombrePropiedad

```
$ MATCH (n :Alumno)
```

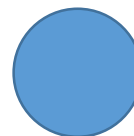
```
  REMOVE n.FechaNac, n.Nombre, n.mails, n.edad
```

```
  RETURN n
```

Inglés Español <id>: 0



Inglés Español <id>: 1



Aquellas propiedades que no están definidas NO producen error al intentar borrarlas de los nodos.
Análogamente si borrara rótulos.

Alumno ITBA <id>: 2



Alumno Inglés Español <id>: 3



- Mails: jmpolo@itba.edu.ar,juan@yahoo.com

Cypher - ejes

(n)- [e :Type { Prop₁: Value₁, Prop₂: Value₂, ... Prop_k: Value_k }] -> (v)

Se coloca la lista de K Propiedades (opcionales) que corresponden al nodo. Cada propiedad está conformada por su nombre y valor, separados por el símbolo ":" La lista de propiedades se coloca entre llaves y se separan con el símbolo ","

Se coloca un solo Type (obligatorio) y se prefija con el símbolo :

A un eje se lo coloca entre los símbolos []. Como es un eje dirigido, alrededor debe aparecer su dirección entre 2 nodos (en este caso n y v). La misma se indica entre 2 símbolos de guión y el destino lleva un < o > según corresponda. O sea si va de "n" a "v" se coloca - [] -> , pero si fuera de "v" a "n" se coloca <- [] -

Dentro debe obligatoriamente colocarse el nombre de una variable cualquiera que cumple el rol de identificarlo dentro de dicha expresión. Su alcance es local.

Cypher - ejes

Tenemos una nueva BD Neo4j. Supongamos que hemos creado previamente los nodos:

```
$ CREATE (n :Empleado { Nombre: 'Ariel Casso',  
  Sueldo: 10000,  
  Mails: ['acasso@itba.edu.ar', 'acasso@yahoo.com']  });
```

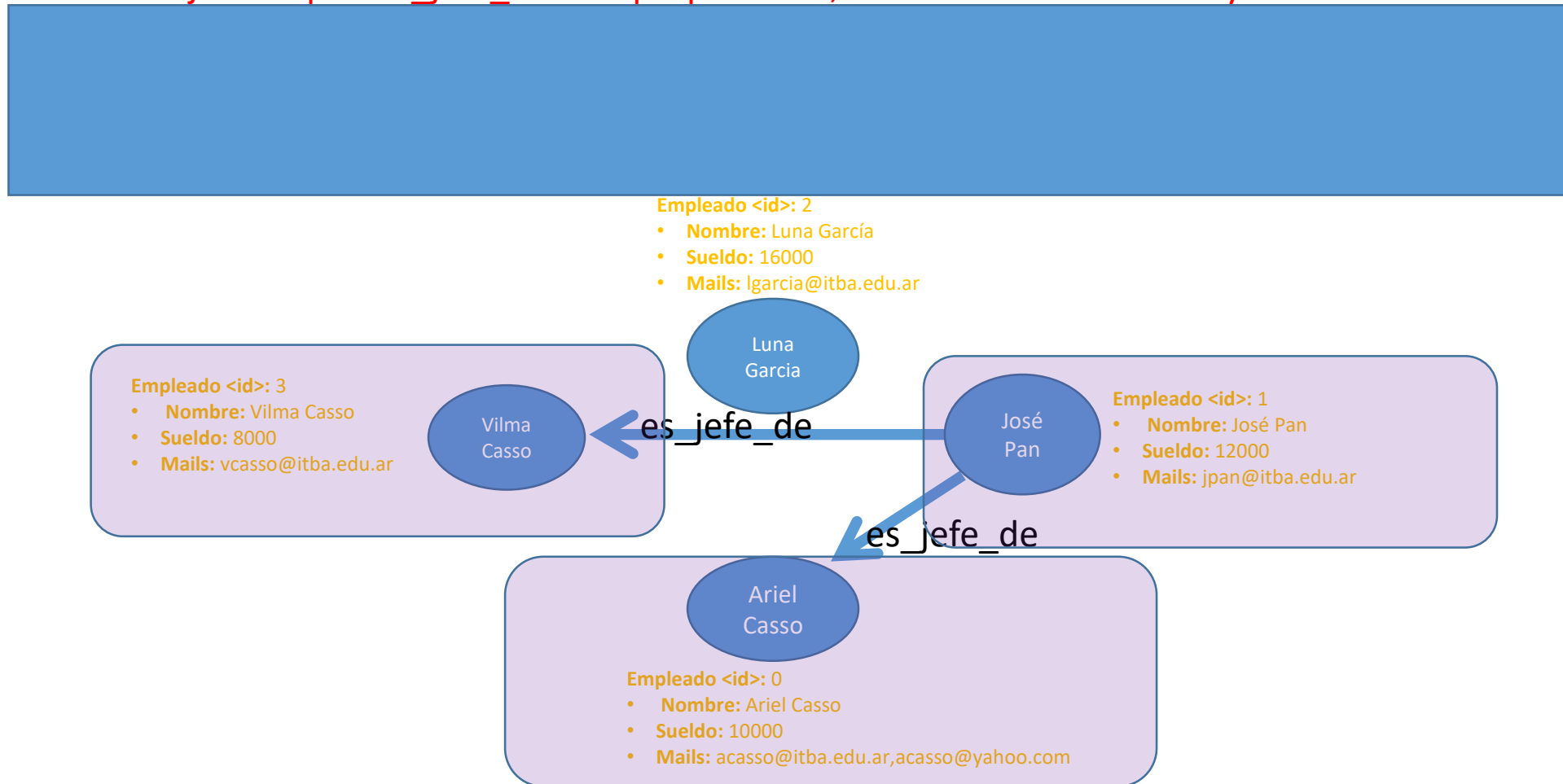
```
CREATE (n :Empleado { Nombre: 'José Pan',  
  Sueldo: 12000,  
  Mails: ['jpan@itba.edu.ar']  });
```

```
CREATE (n :Empleado { Nombre: 'Luna García',  
  Sueldo: 16000,  
  Mails: ['lgarcia@itba.edu.ar']  });
```

```
CREATE (n :Empleado { Nombre: 'Vilma Casso',  
  Sueldo: 8000,  
  Mails: ['vcasso@itba.edu.ar']  });
```

Cypher - ejes

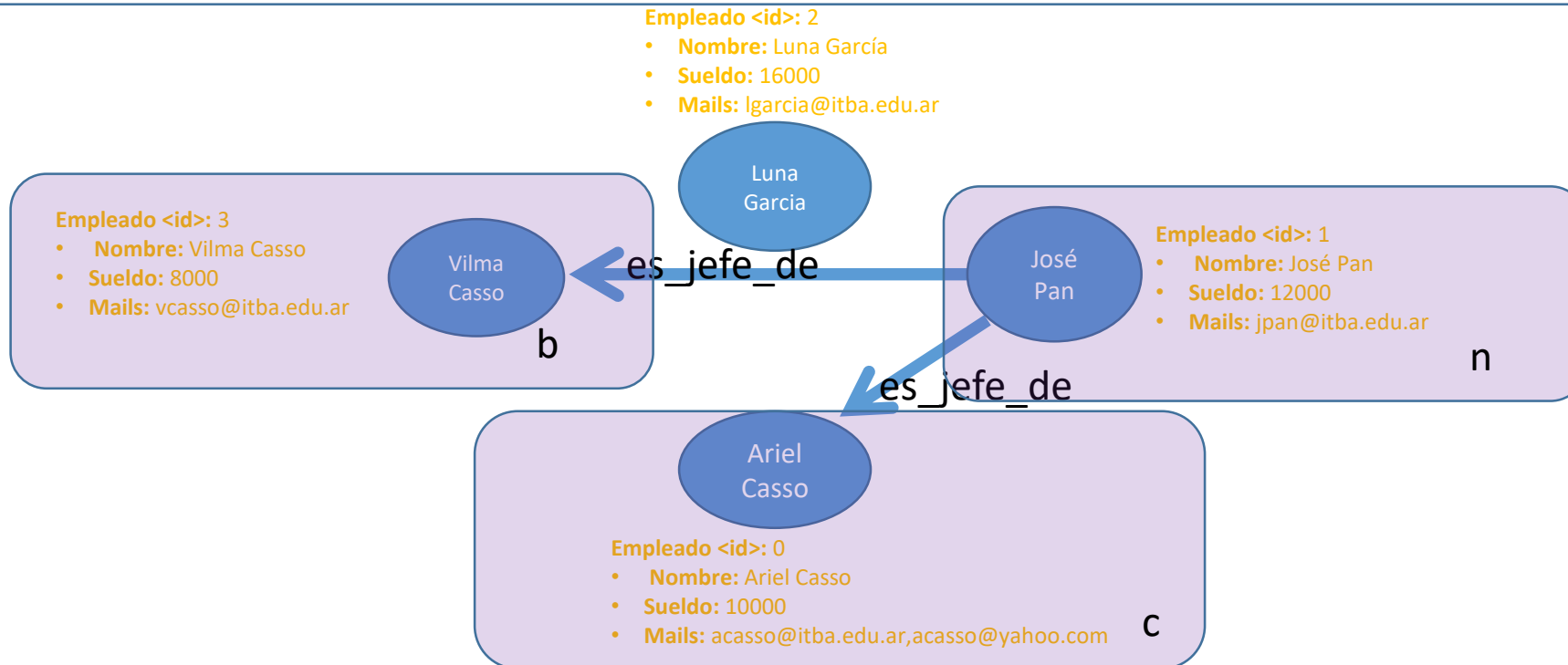
Crear un eje del tipo «es_jefe_de» sin propiedades, desde José Pan a Vilma y Ariel Casso:



Cypher - ejes

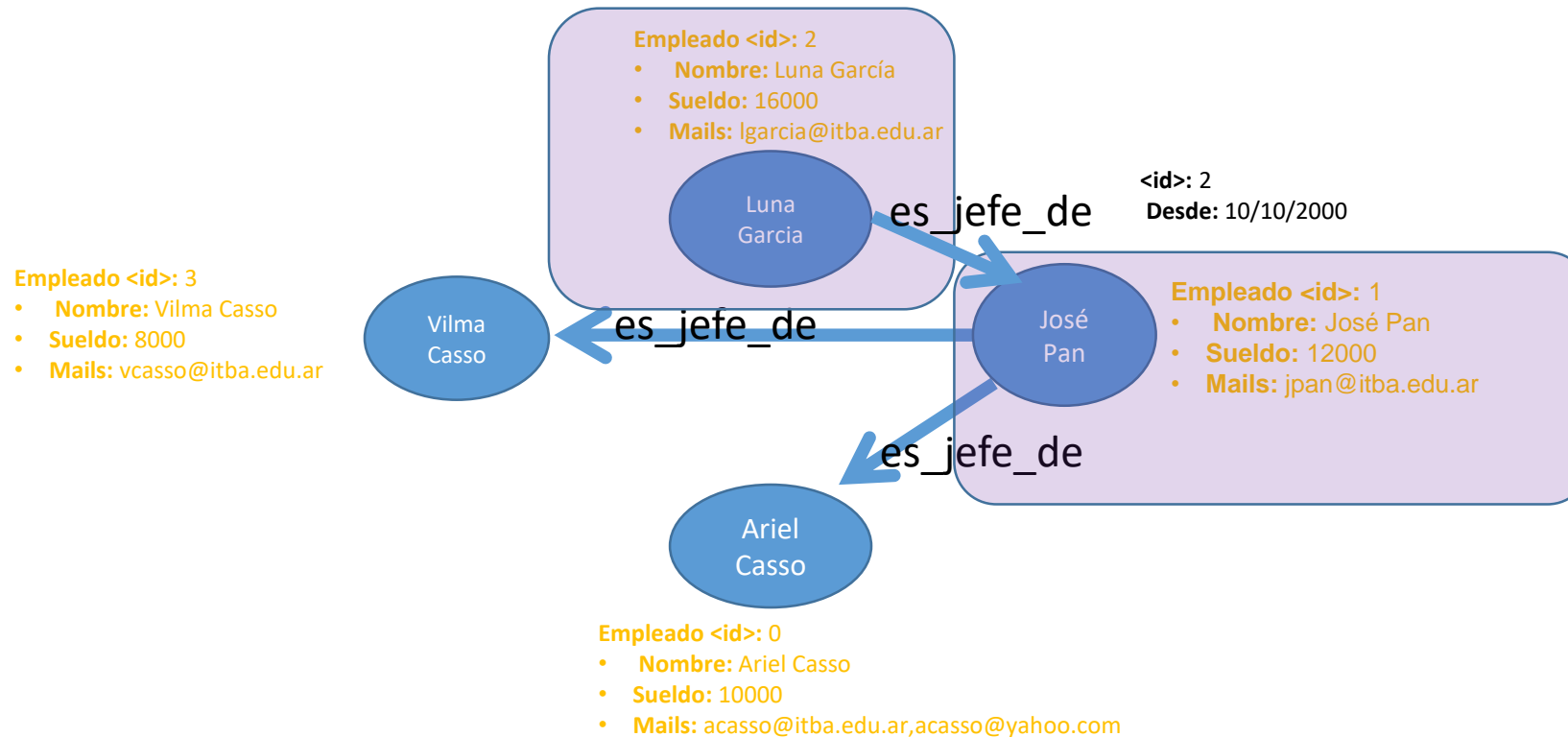
Crear un eje del tipo «es_jefe_de» sin propiedades, desde José Pan a Vilma y Ariel Casso:

```
$ MATCH ( n :Empleado {Nombre: 'José Pan'} ), ( b :Empleado {Nombre: 'Vilma Casso'} ), ( c :Empleado {Nombre: 'Ariel Casso'} )
CREATE (b) <- [r1 :es_jefe_de] - (n) - [r2 :es_jefe_de] -> (c)
RETURN r1, r2
```



Cypher - ejes

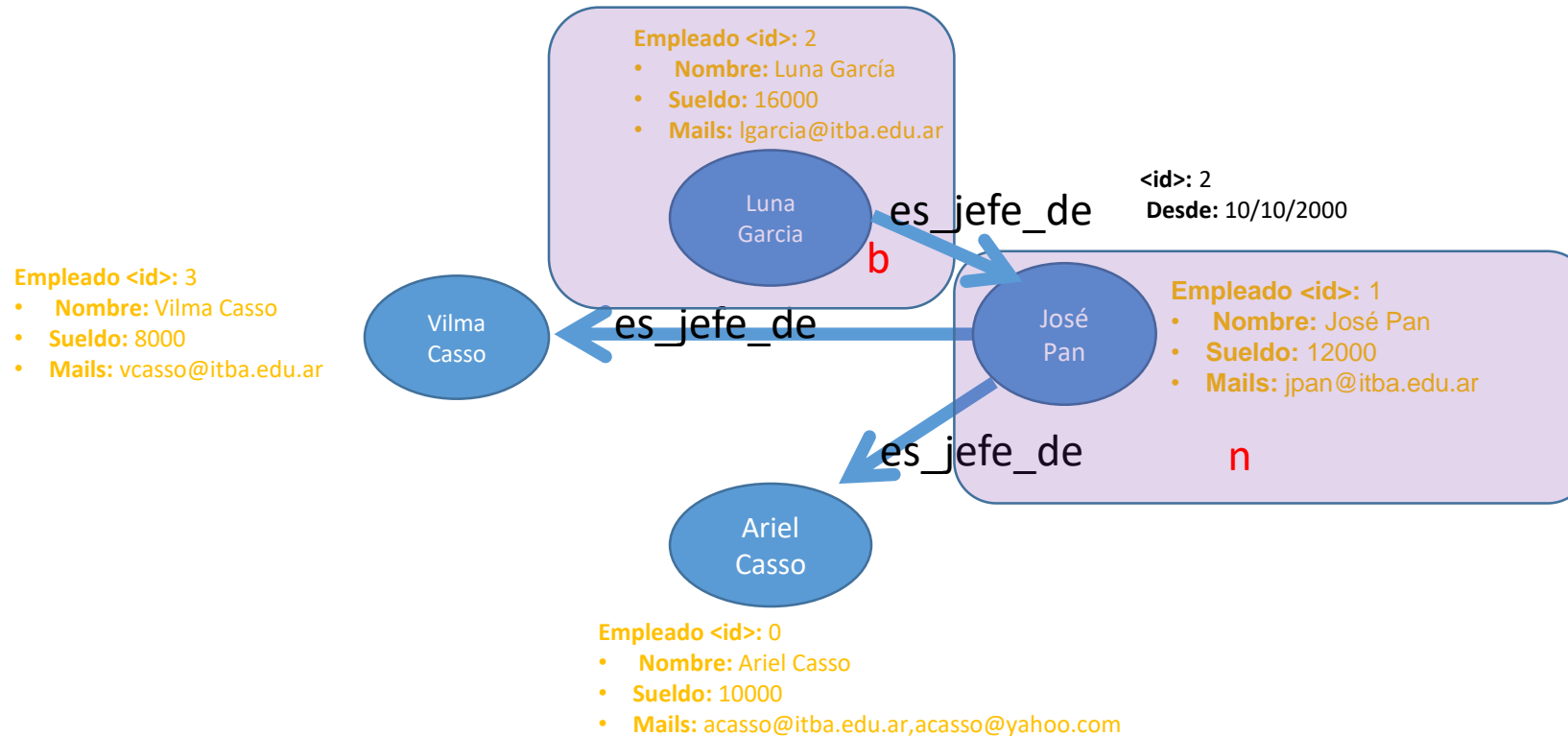
Crear otro eje del tipo «es_jefe_de» con propiedad “desde”, de L. García a José Pan



Cypher - ejes

Crear otro eje del tipo «es_jefe_de» con propiedad “desde”, de L. García a José Pan

```
$ MATCH ( n :Empleado {Nombre: 'José Pan'} ),( b :Empleado {Nombre: 'Luna García'})
  CREATE (n) <- [r :es_jefe_de {Desde: '10/10/2000'} ] - (b)
  RETURN n, r, b
```



Cypher – consultas

Lenguaje de consultas de alto nivel
basado en pattern matching

Para consultar los grafos: expresando
condición informacional y/o
información topológica

Cypher – consultas

MATCH

OPTIONAL MATCH

WHERE

RETURN

ORDER BY

LIMIT

SKIP

«Match» expresa un patrón que el DBMS buscará entre los nodos.

OPTIONAL MATCH funciona «como un outer join en SQL», es decir en donde no matchea coloca nulls.

La clausula WHERE indica más detalle pero «es parte del MATCH o del OPTIONAL MATCH». No se puede asumir que hay un orden en las evaluaciones de clausula WHERE respecto de las otras. Eso lo decide el DBMS.

LIMIT devuelve un subtotal de lo obtenido. SKIP saltea los primeros valores. Pero atención, salvo que se haya usado ORDER BY no se puede asumir cuáles valores se descartan.

De la evaluación anterior se obtienen «subgrafos» y se puede devolver cualquier parte del patrón que tuvo matching. De expresar «RETURN DISTINCT» elimina filas repetidas en el resultado

Cypher – consultas

Ya vimos patrones para nodos () y ejes -[]-> o <[]-. Además:

- 1) Si no interesa referir a un nodo en la expresión, se puede colocar () sin variables.
- 2) Si no interesa referir al eje, omitirlo. Ej: (a) --> (b) indica un eje entre a y b
- 3) Si no interesa el «sentido» del eje, se puede usar directamente un -- (sin el < o >)
- 4) Cada eje tiene un único Type asociado, pero si se quiere expresar un patrón donde pueda matchear más de uno, usar el PIPE. Ej: [:es_jefe_de | :alumno]
- 5) Si se quiere expresar un camino de cualquier longitud entre nodos, se usa [*]. Si se lo quiere limitar a longitud 3, se le agrega el número [*3]
- 6) Si se lo quiere limitar inferiormente o superiormente también se lo puede hacer. Ej: [*2..4] que significa rango entre 2 y 4. Si se lo quiere limitar solo superiormente o solo inferiormente se omite después del doble punto el valor. Ej: [*2 ..] significa camino de longitud 2 o más.

Cypher – funciones SQL-like

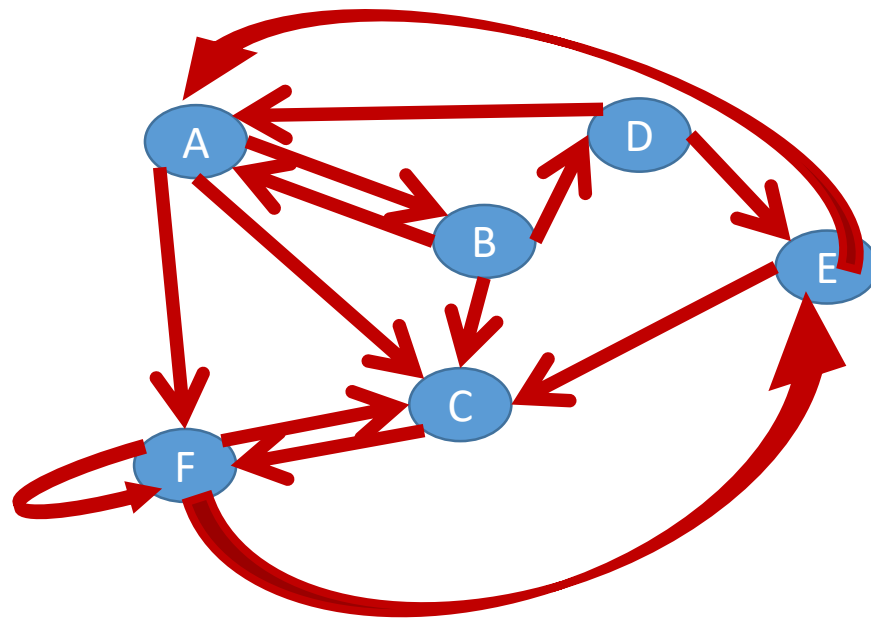
- 1) UNION y UNION ALL para unir resultados.
- 2) Usar IS NULL para comparar con NULL (=null da siempre FALSE).
- 3) Ofrece los 2 tipos de CASE para convertir expresiones.
- 4) Se pueden realizar agregaciones (similar al GROUP BY de SQL), pero se mezcla en una sola cláusula: lo que no está afectado de función de agregación es lo que armaría los grupos (como un GROUP BY implícito). Permite usar DISTINCT dentro de las mismas.
- 5) Si en vez de usar las agregaciones en la cláusula RETURN, se la usa en la **cláusula WITH** se puede usar un **WHERE** luego, y funciona similar al HAVING.
- 6) Función COALESCE, análoga a SQL.
- 7) Para chequear si algo pertenece a una colección se utiliza IN.
- 8) Función TIMESTAMP() que devuelve la cantidad de ms transcurridos desde 1 Jan de 1970, pero en una misma expresión se evalúa sólo una vez.

Cypher – Ejemplo

La consulta:

```
$ MATCH (p)-[]->(s)-[]->(x)  
RETURN Count(p), s, Count(x)
```

Devuelve lo siguiente. Por qué???



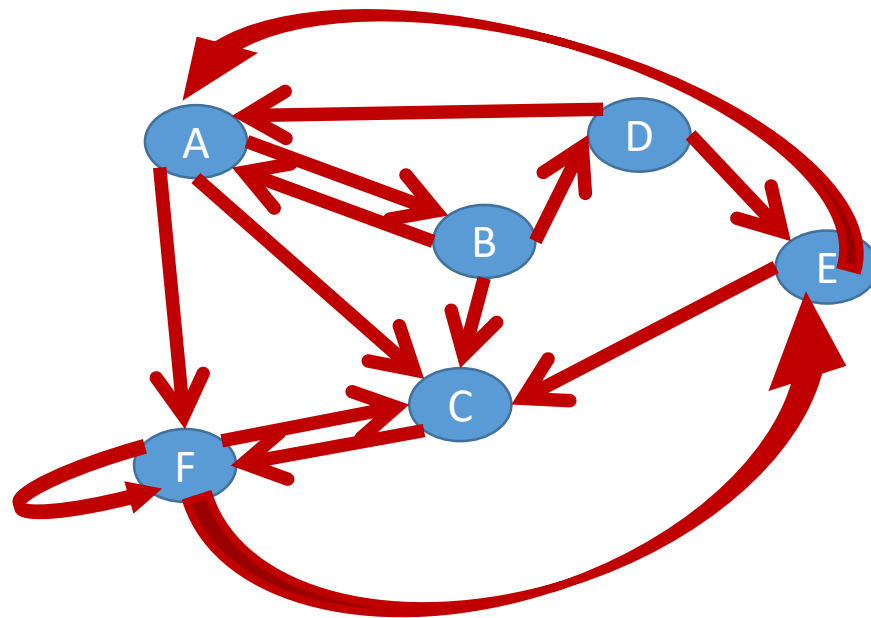
Cypher – Ejemplo

La consulta:

```
$ MATCH (p)-[]->(s)-[]->(x)
  RETURN Count(p), s, Count(x)
```

Devuelve lo siguiente. Por qué???

«Count(p)»	«s»	«Count(x)»
9	A	9
3	B	3
4	C	4
2	D	2
4	E	4
8	F	8



Cypher – Ejemplo

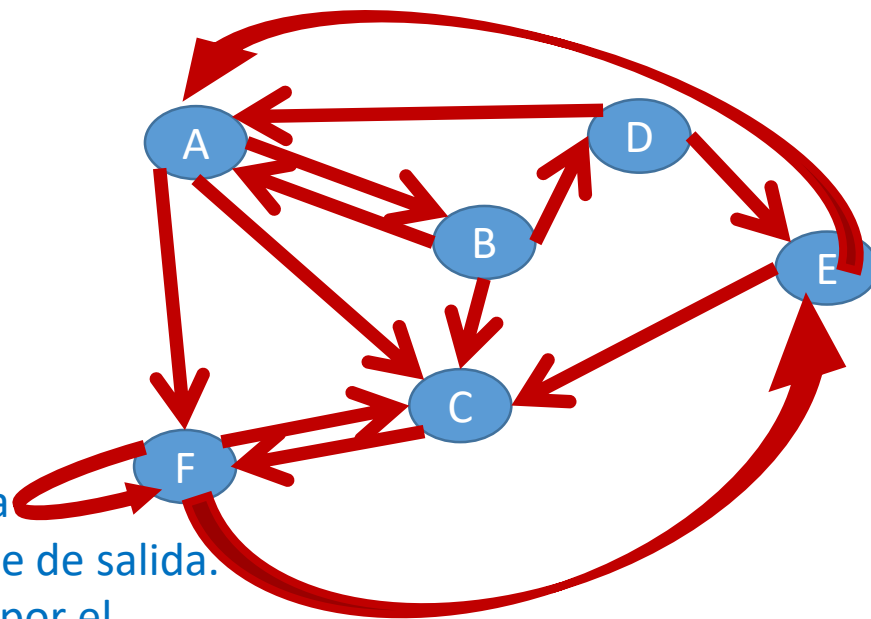
```
$ MATCH (p)-[]->(s)-[]->(x)
RETURN Count(p), s, Count(x)
```

La primera cláusula calcula caminos donde haya un nodo (s) que tenga un eje de entrada y un eje de salida.
Ej: para el nodo «c» esos caminos están dados por el siguiente patrón de matching:

- (a) -- (c) → (f)
- (f) -- (c) → (f)
- (b) -- (c) → (f)
- (e) -- (c) → (f)

La segunda cláusula agrupa esos 4 caminos para dicho nodo, y devuelve cuántas primeros nodos hubo y cuántos segundos nodo hubo. Por eso se obtiene.

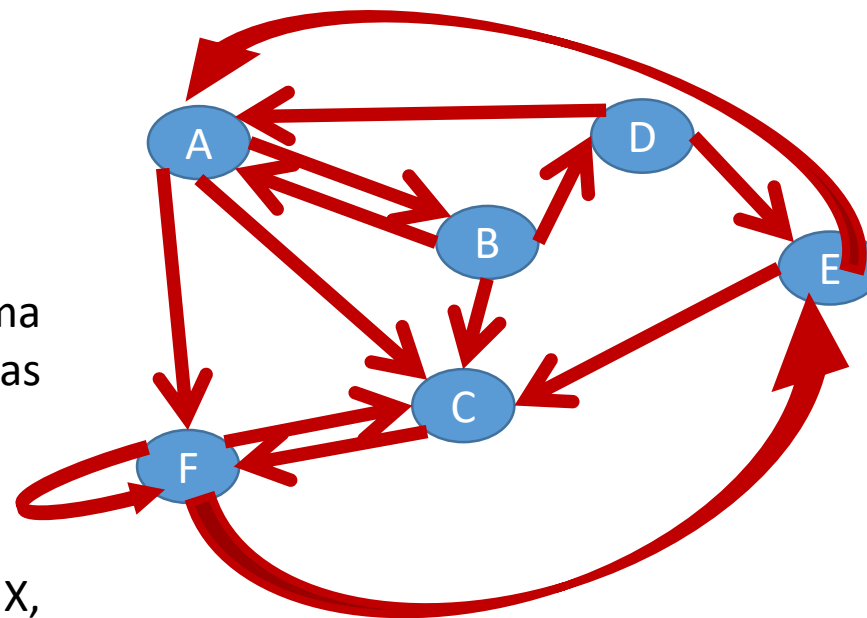
4	c	4
---	---	---



Cypher – Ejemplo

El puntaje que una página X recibe es la suma de todos los votos que le dan cada una de las páginas que la referencian.

Si una página Z referencia a una página X, entonces, Z le da a X un voto normalizado que es la inversa de la cantidad de páginas en total que Z referencia. Sin embargo, para evitar considerar votos de páginas que se auto-referencian o se referencian entre sí, si Z referencia a una página X y X la referencia a Z, entonces Z le da 0 votos a X.

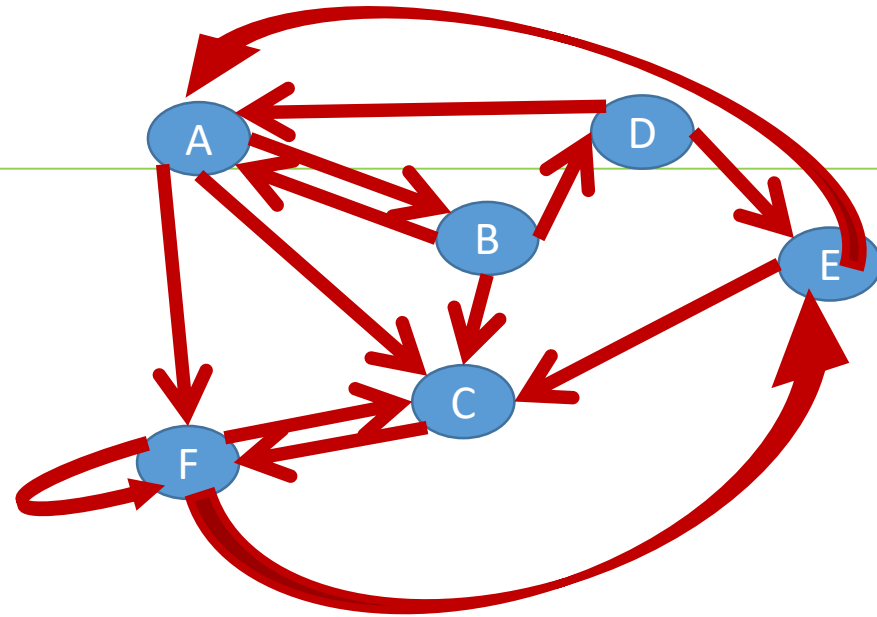


Calcular el page rank para cada página

Cypher – Ejemplo

Posible solución:

```
$ MATCH (p) --> (r)
  WITH p, 1.0 / count(r) as voto
MATCH (p) --> (x)
WHERE NOT ( (x) --> (p) )
RETURN SUM(voto) AS Rank, x
ORDER BY x.URL
```



«p»	«voto»
A	0.333
B	0.333
C	1
D	0.5
E	0.5
F	0.333

El primer MATCH WITH calcula de cada nodos la inversa de la cantidad de ejes salientes y se los pasa (pipe) a la siguiente cláusula.

Cypher – Ejemplo

Posible solución:

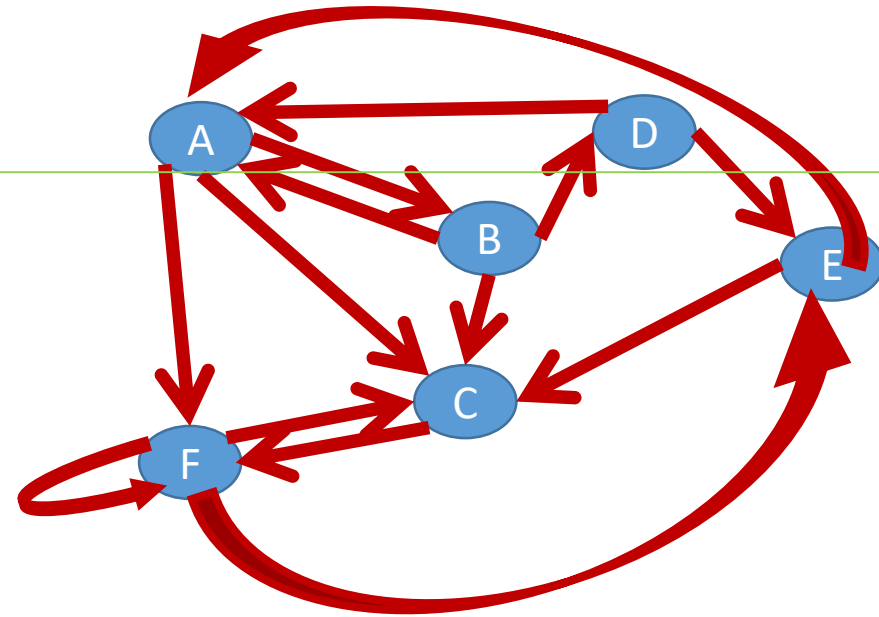
```
$ MATCH (p) --> (r)
  WITH p, 1.0 / count(r) as voto
```

```
MATCH (p) --> (x)
WHERE NOT ( (x) --> (p) )
```

```
RETURN SUM(voto) AS Rank, x
ORDER BY x.URL
```

«p»	«voto»
A	0.333
B	0.333
C	1
D	0.5
E	0.5
F	0.333

«p»	«x»
A	C
A	F
B	C
B	D
D	A
D	E
E	A
E	C
F	E



Ahora, para cada uno de esos 6 “p” nodos busca caminos de longitud 1 donde no haya reciprocidad

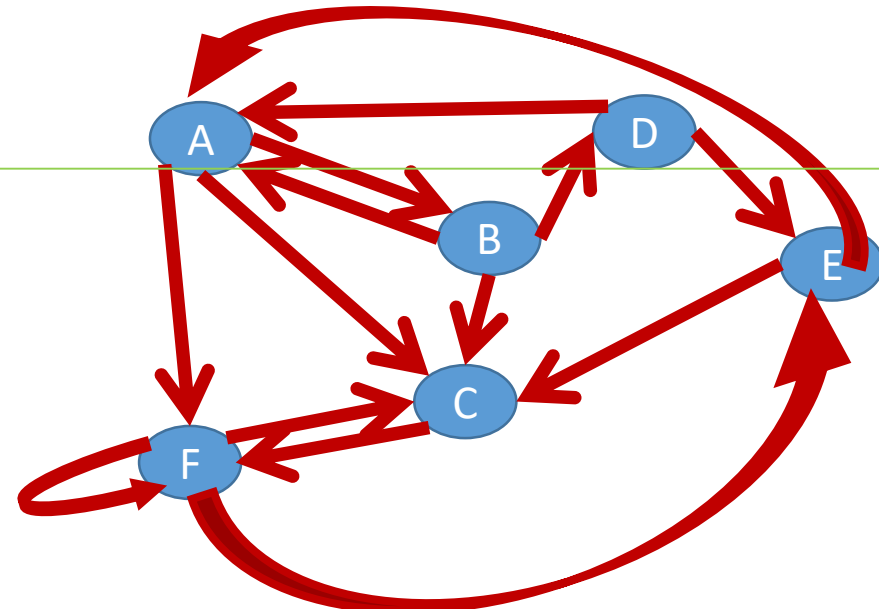
Cypher – Ejemplo

Posible solución:

```

$ MATCH (p) --> (r)
  WITH p, 1.0 / count(r) as voto
  MATCH (p) --> (x)
  WHERE NOT ( (x) --> (p) )
RETURN SUM(voto) AS Rank, x
ORDER BY x.URL

```



«p»	«voto»
A	0.333
B	0.333
C	1
D	0.5
E	0.5
F	0.333

«p»	«x»
A	C
A	F
B	C
B	D
D	A
D	E
E	A
E	C
F	E

«p» agrupados	«x»	«Rank»	«x»
D, E	A	$\frac{1}{2} + \frac{1}{2}$	A
A, B, E	C	$\frac{1}{3} + \frac{1}{3} + \frac{1}{2}$	C
B	D	$\frac{1}{3}$	D
D, F	E	$\frac{1}{2} + \frac{1}{3}$	E
A	F	$\frac{1}{3}$	F

Finalmente agrupa por la segunda componente y ordena

Y OLAP?

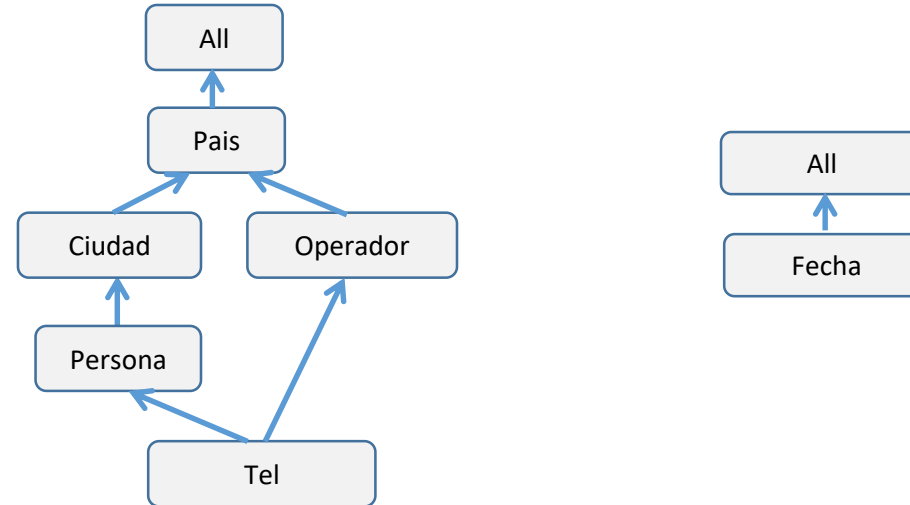
- Mucho trabajo en sumarización de grafos
- Pero poco en definir OLAP
- El modelo de grafos puede ser útil para tratar problemas que en OLAP tradicional son difíciles
 - Por ej., cuando el número de dimensiones intervinientes en una fact table no es fijo
 - Caso: llamadas grupales
- Vamos a dar un ejemplo de cómo un caso tradicional podría verse como un grafo e implementarse en Neo4j

Ejemplo

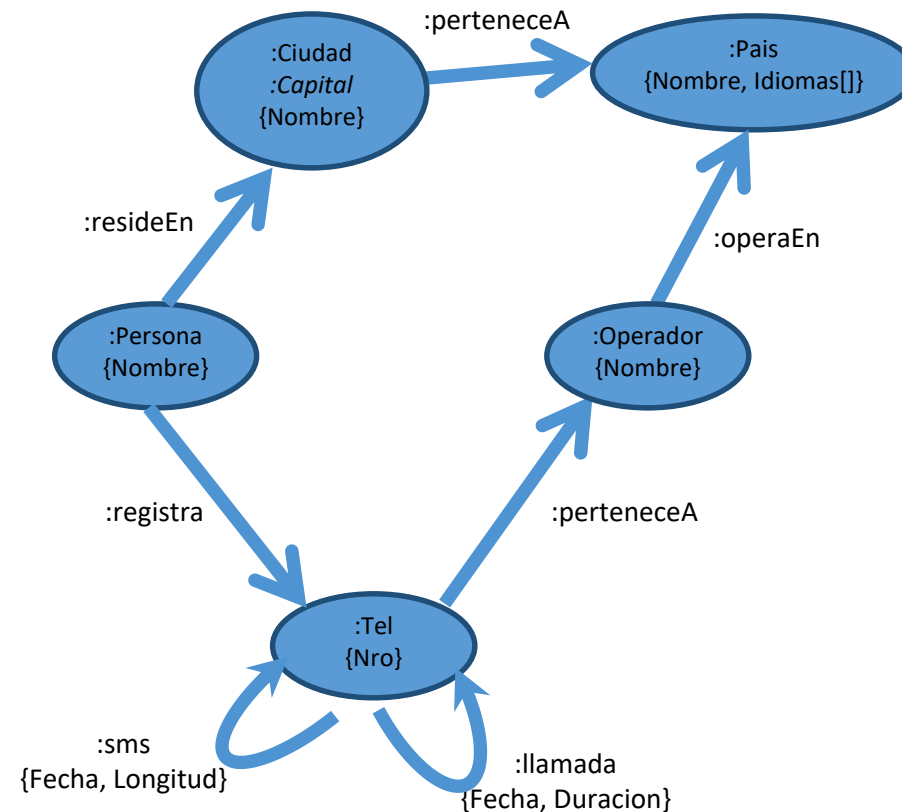
- Se dispone de información sobre las comunicaciones establecidas por telefonía en para analizar tráfico y mejorar la infraestructura. Más precisamente se conoce:
- **Geografía**: Países y ciudades. De los primeros se conoce los idiomas oficiales que se hablan. Algunas ciudades son también capitales.
- **Operadores** telefónicos por país.
- **Números de teléfonos** que pertenecen a cada operador.
- **Personas** que han registrado esos teléfonos y sus ciudades de residencia. Las personas pueden tener varios teléfonos, pero residen en un único lugar.
- **Comunicación** establecida entre los teléfonos, diferenciando entre sms o llamada. Para la primera se registra la fecha y longitud del texto emitido. Para la segunda la fecha y duración de la llamada.

Modelo conceptual

- 3 dimensiones: Emisor, Receptor, Tiempo
- 2 métricas: longitud (SMS), duración (llamada)



Modelo lógico en Neo4j (“esquema”)



Operaciones: SLICE

- **Slice** sobre la dimensión Tiempo (no tenerla en cuenta) y un **Slice** en la métrica longitud, sumalizando las métricas restantes con la función de agregación **avg**. Esta consulta OLAP calcula el promedio de longitud de llamada emitidas correspondiente a cada par de teléfono emisor a un receptor.

Operaciones: SLICE

- **Slice** sobre la dimensión Tiempo (no tenerla en cuenta) y un **Slice** en la métrica longitud, sumalizando las métricas restantes con la función de agregación **avg**. Esta consulta OLAP calcula el promedio de duración de llamada emitidas correspondiente a cada par de teléfono emisor a un receptor.

MATCH (n :Tel) -[r :llamada]-> (m: Tel)

RETURN n as TelEmisor, m as TelReceptor, AVG(r.Duracion)

AS PromedioDuracion

Operaciones: SLICE

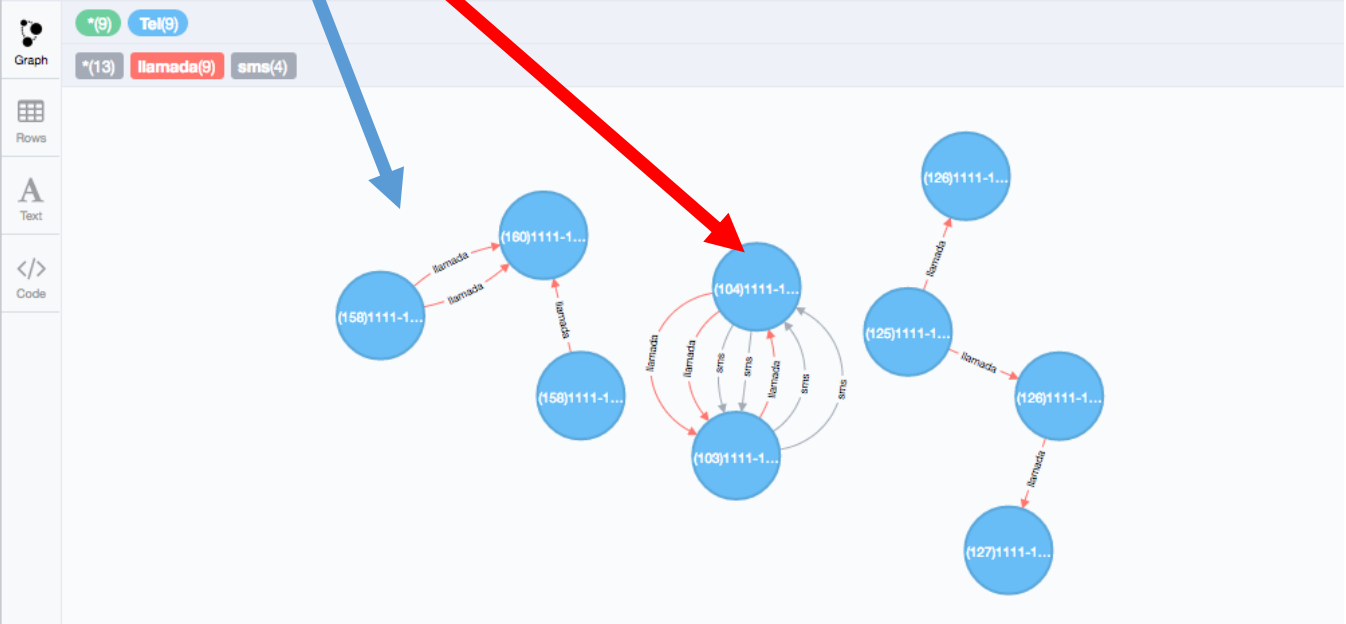
MATCH (n :Tel) -[r :llamada]-> (m: Tel)

RETURN n as TelEmisor, m as TelReceptor, AVG(r.Duracion)
AS PromedioDuracion

TelEmisor	TelReceptor	PromedioDuracion
(158)1111-1111	(160)1111-1113	6.5 (hubo 2 llamadas, una duró 12 y otra 1)
(104)1111-1111	(103)1111-1111	2.5 (hubo 2 llamadas, una duró 2 y otra 3)
(103)1111-1111	(104)1111-1111	7 (hubo una llamada que duró 7)
(125)1111-1111	(126)1111-1113	17 (hubo una llamada que duró 17)
(126)1111-1113	(127)1111-1113	3 (hubo una llamada que duró 3)
(158)1111-1112	(160)1111-1113	1 (hubo una llamada que duró 1)
(125)1111-1111	(126)1111-1112	20 (hubo una llamada que duró 20)

Operaciones: SLICE

TelEmisor	TelReceptor	PromedioDuracion
(158)1111-1111	(160)1111-1113	6.5 (hubo 2 llamadas, una duró 12 y otra 1)
(104)1111-1111	(103)1111-1111	2.5 (hubo 2 llamadas, una duró 2 y otra 3)
(103)1111-1111	(104)1111-1111	7 (hubo una llamada que duró 7)
(125)1111-1111	(126)	
(126)1111-1113	(127)	
(158)1111-1112	(160)	
(125)1111-1111	(126)	



Operaciones: SLICE

- **Slice** sobre la dimensión Tiempo (no tenerla en cuenta) y un **Slice** en la métrica longitud, sumalizando las métricas restantes con la función de agregación **avg**. *Pero, a diferencia del anterior sumarizar la duración de las llamadas sin importar quien las inició.*

Operaciones: SLICE

- **Slice** sobre la dimensión Tiempo (no tenerla en cuenta) y un **Slice** en la métrica longitud, sumando las métricas restantes con la función de agregación **avg**. *Pero, a diferencia del anterior sumarizar la duración de las llamadas sin importar quien las inició.*

MATCH (n :Tel) -[r :llamada]- (m: Tel)

WHERE n.Nro < m.Nro

RETURN n as Tel1, m as Tel2, AVG(r.Duracion) As PromedioDuracion;

Operaciones: SLICE

MATCH (n :Tel) -[r :llamada]- (m: Tel)

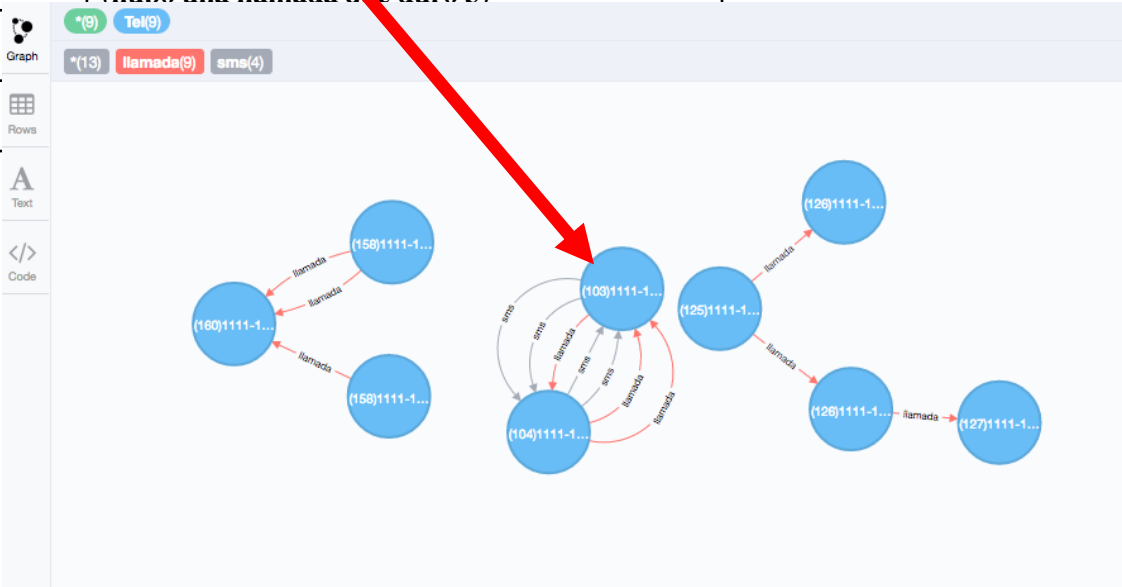
WHERE **n.Nro** < **m.Nro**

RETURN n as Tel1, m as Tel2, AVG(r.Duracion) As PromedioDuracion;

Tel1	Tel2	PromedioDuracion
(158)1111-1111	(160)1111-1113	6.5 (hubo 2 llamadas, una duró 12 y otra 1)
(103)1111-1111	(104)1111-1111	4 (se sumaron las 3 llamadas en total, sin importar quien la inició: 2, 3 y 7)
(125)1111-1111	(126)1111-1113	17 (hubo una llamada que duró 17)
(126)1111-1113	(127)1111-1113	3 (hubo una llamada que duró 3)
(158)1111-1112	(160)1111-1113	1 (hubo una llamada que duró 1)
(125)1111-1111	(126)1111-1112	20 (hubo una llamada que duró 20)

Operaciones: SLICE

Tel1	Tel2	PromedioDuracion
(158)1111-1111	(160)1111-1113	6.5 (hubo 2 llamadas, una duró 12 y otra 1)
(103)1111-1111	(104)1111-1111	4 (se sumaron las 3 llamadas en total, sin importar quien la inició: 2, 3 y 7)
(125)1111-1111	(126)1111-1113	17 (hubo una llamada que duró 17)
(126)1111-1113	(127)1111-1113	3 (hubo una llamada que duró 3)
(158)1111-1112	(160)1111-1113	
(125)1111-1111	(126)1111-1112	



Operaciones: Rollup

- Ahora, queremos hacer la agregación anterior pero haciendo previamente **RollUp** a **Persona** tanto para la dimensión Emisor como para Receptor. Es decir, los teléfonos que corresponden a la mismas personas deben **sumarizarse**.

Operaciones: Rollup

- Ahora, queremos hacer la agregación anterior pero haciendo previamente **RollUp** a **Persona** tanto para la dimensión Emisor como para Receptor. Es decir, los teléfonos que corresponden a la mismas personas deben **sumarizarse**.

```
MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -[r :llamada]- (m: Tel)<-[r2: registra]-  
(y :Persona)
```

```
WHERE x.Nombre < y.Nombre
```

```
RETURN x as Persona1, y as Persona2 , AVG(r.Duracion) As PromedioDuracion;
```

Operaciones: Rollup

MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -[r :llamada]- (m: Tel)<-[r2: registra]-(y :Persona)

WHERE x.Nombre < y.Nombre

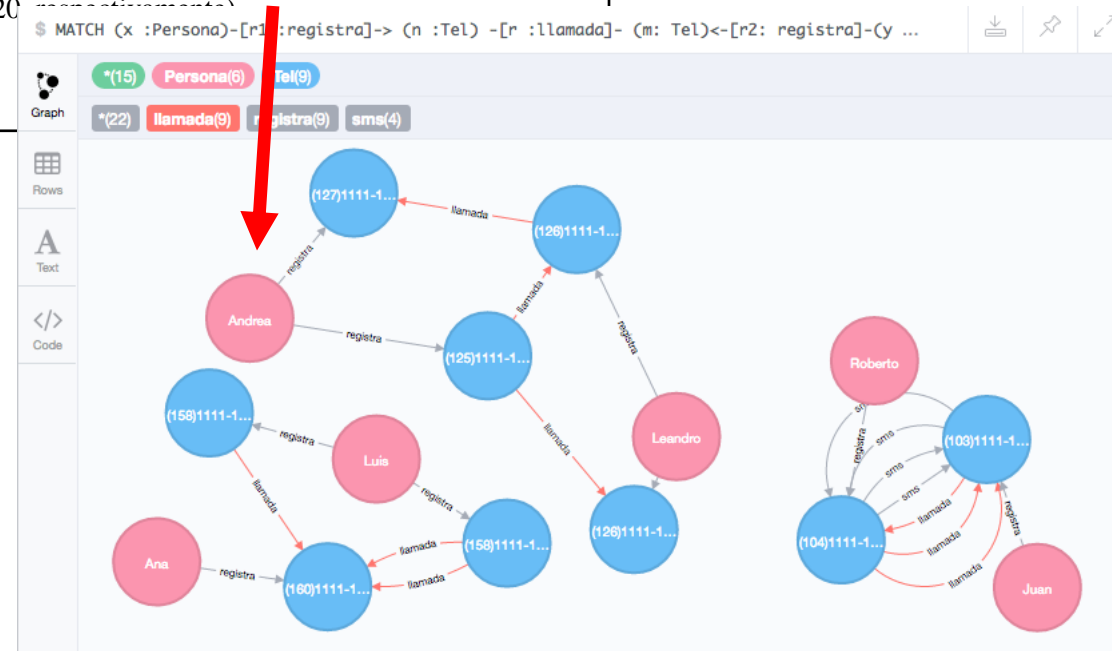
RETURN x as Persona1, y as Persona2 , AVG(r.Duracion) As PromedioDuracion;

Persona1	Persona2	Promedio Duración
Nombre: Ana (Liverpool) ID: 315 Sexo: F	Nombre: Luis (Londres) ID: 313 Sexo: M	4.666667 (hubo 3 llamadas, con duración 12, 1 y 1 respectivamente)
Nombre: Juan (Amberes) ID: 300 Sexo: M	Nombre: Roberto (Amberes) ID: 301 Sexo: M	4 (hubo 3 llamadas, con duración 2, 3 y 7, respectivamente)
Nombre: Andrea (Roma) ID: 307 Sexo: M	Nombre: Leandro (Roma) ID: 308 Sexo: M	13.333333 (hubo 3 llamadas con duración 17, 3 y 20, respectivamente)

Operaciones: Rollup

Persona1	Persona2	Promedio Duración
Nombre: Ana (Liverpool) ID: 315 Sexo: F	Nombre: Luis (Londres) ID: 313 Sexo: M	4.666667 (hubo 3 llamadas, con duración 12, 1 y 1 respectivamente)
Nombre: Juan (Amberes) ID: 300 Sexo: M	Nombre: Roberto (Amberes) ID: 301 Sexo: M	4 (hubo 3 llamadas, con duración 2, 3 y 7, respectivamente)
Nombre: Andrea (Roma) ID: 307 Sexo: M	Nombre: Leandro (Roma) ID: 308 Sexo: M	13.333333 (hubo 3 llamadas con duración 17, 3 y 20 respectivamente)

Nota: para mostrar el grafo, se agregó "r" al output



Operaciones: Dice

- Realizar la misma consulta anterior, pero con un **Dice** para filtrar pares de usuarios del mismo sexo, o sea F-F o M-M.

Operaciones: Dice

- Realizar la misma consulta anterior, pero con un **Dice** para filtrar pares de usuarios del mismo sexo, o sea F-F o M-M.

```
MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -[r :llamada]- (m: Tel)<-[r2: registra]-(y :Persona)
```

```
WHERE x.Nombre < y.Nombre AND x.Sexo = y.Sexo
```

```
RETURN x as Persona1, y as Persona2 , AVG(r.Duracion) As PromedioDuracion;
```

Operaciones: Dice

MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -[r :llamada]- (m: Tel)<-[r2: registra]-(y :Persona)

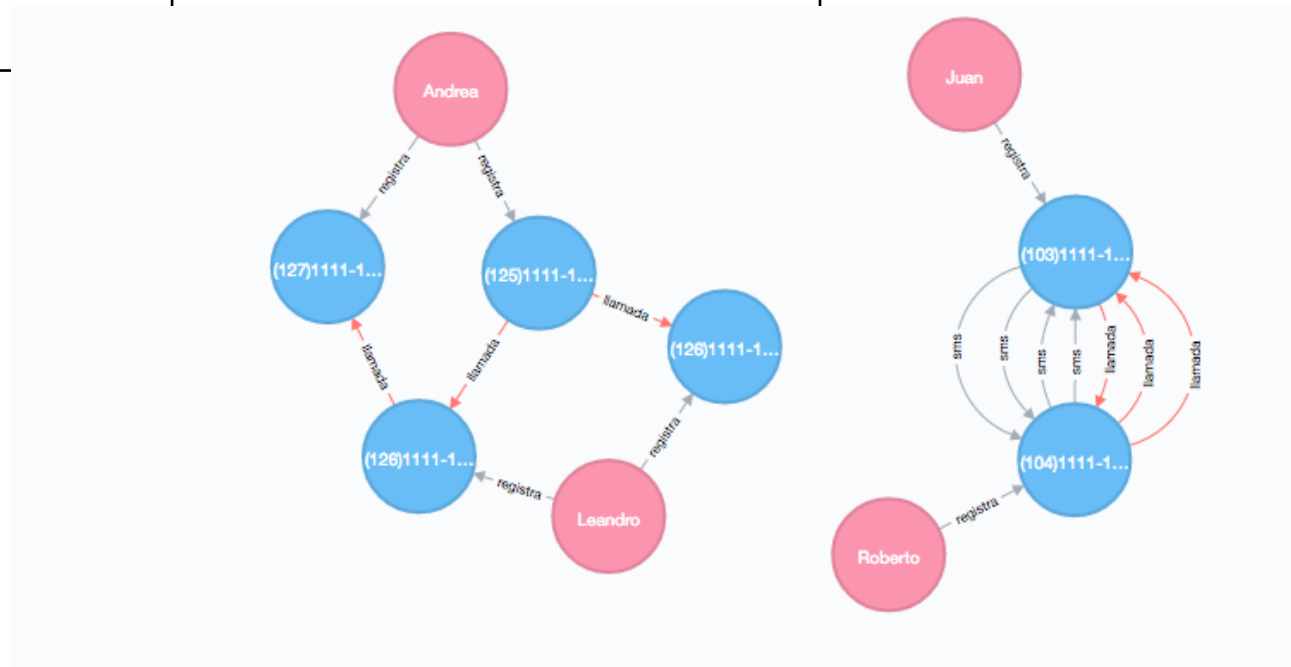
WHERE x.Nombre < y.Nombre AND x.Sexo = y.Sexo

RETURN x as Persona1, y as Persona2 , AVG(r.Duracion) As PromedioDuracion;

Persona1	Persona2	Promedio Duración
Nombre: Juan (Amberes) ID: 300 Sexo: M	Nombre: Roberto (Amberes) ID: 301 Sexo: M	4 (hubo 3 llamadas, con duración 2, 3 y 7, respectivamente)
Nombre: Andrea (Roma) ID: 307 Sexo: M	Nombre: Leandro (Roma) ID: 308 Sexo: M	13.333333 (hubo 3 llamadas con duración 17, 3 y 20, respectivamente)

Operaciones: Dice

Persona1	Persona2	Promedio Duración
Nombre: Juan (Amberes) ID: 300 Sexo: M	Nombre: Roberto (Amberes) ID: 301 Sexo: M	4 (hubo 3 llamadas, con duración 2, 3 y 7, respectivamente)
Nombre: Andrea (Roma) ID: 307 Sexo: M	Nombre: Leandro (Roma) ID: 308 Sexo: M	13.333333 (hubo 3 llamadas con duración 17, 3 y 20, respectivamente)



Operaciones – agregando métricas

- Similar al ejemplo de rollup, agregando al nivel Persona en cada dimensión sin importar el sexo, pero realizando un **SLICE** sobre la **métrica** correspondiente a llamadas, o sea **agregando longitud del mensaje de texto**. Excluir aquellas personas que se envían mensajes a sí mismas.

Operaciones – agregando métricas

- Similar al ejemplo de rollup, agregando al nivel Persona en cada dimensión sin importar el sexo, pero realizando un **SLICE** sobre la **métrica** correspondiente a llamadas, o sea **agregando longitud del mensaje de texto**. Excluir aquellas personas que se envían mensajes a sí mismas.

```
MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -[r :sms]- (m: Tel)<-[r2: registra]-(y :Persona)
```

```
WHERE x.Nombre < y.Nombre
```

```
RETURN x as Persona1, y as Persona2 , AVG(r.Longitud) As PromedioLongitud
```

```
ORDER BY x.Nombre
```

Operaciones – agregando métricas

MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -
[r :sms]- (m: Tel)<-[r2: registra]-(y :Persona)

WHERE x.Nombre < y.Nombre

RETURN x as Persona1, y as Persona2 ,
AVG(r.Longitud) As PromedioLongitud

ORDER BY x.Nombre

Persona1	Persona2	PromedioLongitud
ID: 311 Nombre: Andrea Sexo: F	ID: 304 Nombre: Romina Sexo: F	120
ID: 303 Nombre: Jimena Sexo: F	ID: 300 Nombre: Juan Sexo: M	2
ID: 300 Nombre: Juan Sexo: M	ID: 304 Nombre: Romina Sexo: F	12.5
ID: 300 Nombre: Juan Sexo: M	ID: 301 Nombre: Roberto Sexo: M	85
ID: 305 Nombre: Juana Sexo: F	ID: 306 Nombre: Luis Sexo: M	20
ID: 304 Nombre: Romina Sexo: F	ID: 314 Nombre: Silvio Sexo: M	7

Operaciones – más ejemplos (coalesce)

- Mostrar las 2 métricas, con un **Rollup** a Persona, sin tener en cuenta quien inició la llamada o envío del mensaje. Realizar sólo SLICE temporal.
 - Si hay un par de Personas que sólo poseen sms o sólo poseen llamadas deben aparecer en el resultado, con valor en la columna donde no se registra con 0. Si hay pares de personas que no tuvieron ningún movimiento de comunicación no deben aparecer en el resultado. Esta vez considerar que si la misma persona se manda sms o se llama por teléfono, deberá aparecer.

Operaciones – más ejemplos (coalesce)

- Mostrar las 2 métricas, con un **Rollup** a Persona, sin tener en cuenta quien inició la llamada o envío del mensaje. Realizar sólo SLICE temporal.
 - Si hay un par de Personas que sólo poseen sms o sólo poseen llamadas deben aparecer en el resultado, con valor en la columna donde no se registra con 0. Si hay pares de personas que no tuvieron ningún movimiento de comunicación no deben aparecer en el resultado. Esta vez considerar que si la misma persona se manda sms o se llama por teléfono, deberá aparecer.

MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -[r]- (m: Tel)<-[r2: registra]-(y :Persona)

WHERE x.Nombre <= y.Nombre

RETURN x as Persona1, y as Persona2 , **COALESCE**(AVG(r.Duracion), 0) As PromedioDuracion , **COALESCE**(AVG(r.Longitud), 0) As PromedioLongitud

ORDER BY x.Nombre

Operaciones – más ejemplos (coalsece)

```
MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -[r ]- (m: Tel)<-[r2: registra]-(y :Persona)
```

```
WHERE x.Nombre <= y.Nombre
```

```
RETURN x as Persona1, y as Persona2 , COALESCE(AVG(r.Duracion), 0) As PromedioDuracion ,  
COALESCE(AVG(r.Longitud), 0) As PromedioLongitud
```

```
ORDER BY x.Nombre
```

SOLUCIÓN ALTERNATIVA

```
MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -[r :sms |:llamada]- (m: Tel)<-[r2: registra]-(y :Persona)
```

```
WHERE x.Nombre <= y.Nombre
```

```
RETURN x as Persona1, y as Persona2 ,
```

```
CASE WHEN AVG(r.Duracion)IS NULL THEN 0 ELSE AVG(r.Duracion) END As PromedioDuracion,
```

```
CASE WHEN AVG(r.Longitud)IS NULL THEN 0 ELSE AVG(r.Longitud) END As PromedioLongitud
```

```
ORDER BY x.Nombre
```

Operaciones – más ejemplos (coalesce)

MATCH (x :Persona)-[r1 :registra]-> (n :Tel) -
[r]- (m: Tel)<-[r2: registra]-(y :Persona)

WHERE x.Nombre <= y.Nombre

RETURN x as Persona1, y as Persona2 ,

COALESCE(AVG(r.Duracion), 0) As

PromedioDuracion ,

COALESCE(AVG(r.Longitud), 0) As

PromedioLongitud

ORDER BY x.Nombre

Persona1	Persona2	PromedioDura cion	PromedioLongitud
Nombre: Ana ID: 315 Sexo: F	Nombre: Luis ID: 313 Sexo: M	4.666667	0
Nombre: Andrea ID: 307 Sexo: M	Nombre: Leandro ID: 308 Sexo: M	13.333333	0
Nombre: Andrea ID: 311 Sexo: F	Nombre: Romina ID: 304 Sexo: F	0	120
Nombre: Jimena ID: 303 Sexo: F	Nombre: Juan ID: 300 Sexo: M	0	2
Nombre: Juan ID: 300 Sexo: M	Nombre: Romina ID: 304 Sexo: F	0	12.5
Nombre: Juan ID: 300 Sexo: M	Nombre: Roberto ID: 301 Sexo: M	4	85
Nombre: Juana ID: 305 Sexo: F	Nombre: Juana ID: 305 Sexo: F	0	220
Nombre: Juana ID: 305 Sexo: F	Nombre: Luis ID: 306 Sexo: M	0	20
Nombre: Romina ID: 304 Sexo: F	Nombre: Silvio ID: 314 Sexo: M	0	7

Operaciones – DICE

- Igual al anterior, pero realizar un DICE para mostrar sólo pares de personas si es que las mismas residen en DISTINTOS países.

Operaciones – DICE

- Igual al anterior, pero realizar un DICE para mostrar sólo pares de personas si es que las mismas residen en DISTINTOS países.

```
MATCH (p1 :Pais) <-[*2..2]-(x:Persona)-[r1 :registra]-> (n1 :Tel) -[r :sms|:llamada]- (n2: Tel)<-[r2:registra]-(y :Persona)-[*2..2]->(p2 :Pais)
```

```
WHERE x.Nombre <= y.Nombre AND p1.Nombre <> p2.Nombre
```

```
RETURN x as Persona1, y as Persona2,
```

```
CASE WHEN AVG(r.Duracion) IS NULL THEN 0 ELSE AVG(r.Duracion) END As PromedioDuracion,
```

```
CASE WHEN AVG(r.Longitud)IS NULL THEN 0 ELSE AVG(r.Longitud) END As PromedioLongitud
```

```
ORDER BY x.Nombre
```

Operaciones – DICE

MATCH (p1 :Pais) <-[*2..2]-(x:Persona)-[r1 :registra]-> (n1 :Tel) -[r :sms|:llamada]- (n2: Tel)<-[r2:registra]-(y :Persona)-[*2..2]->(p2 :Pais)

WHERE x.Nombre <= y.Nombre AND p1.Nombre <> p2.Nombre

RETURN x as Persona1, y as Persona2,

CASE WHEN AVG(r.Duracion) IS NULL THEN 0 ELSE AVG(r.Duracion) END As PromedioDuracion,

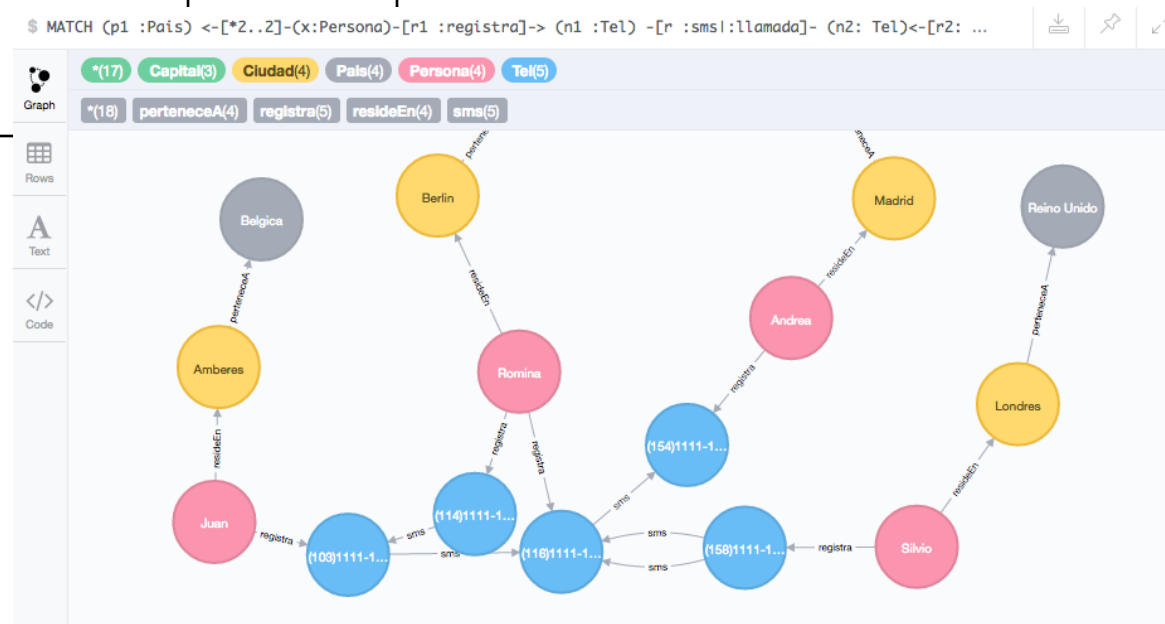
CASE WHEN AVG(r.Longitud)IS NULL THEN 0 ELSE AVG(r.Longitud) END As PromedioLongitud

ORDER BY x.Nombre

Persona1	Persona2	PromedioDuracion	PromedioLongitud
Nombre: Andrea ID: 311 Sexo: F	Nombre: Romina ID: 304 Sexo: F	0	120
Nombre: Juan ID: 300 Sexo: M	Nombre: Romina ID: 304 Sexo: F	0	12.5
Nombre: Romina ID: 304 Sexo: F	Nombre: Silvio ID: 314 Sexo: M	0	7

Operaciones – DICE

Persona1	Persona2	PromedioDuracion	PromedioLongitud
Nombre: Andrea ID: 311 Sexo: F	Nombre: Romina ID: 304 Sexo: F	0	120
Nombre: Juan ID: 300 Sexo: M	Nombre: Romina ID: 304 Sexo: F	0	12.5
Nombre: Romina ID: 304 Sexo: F	Nombre: Silvio ID: 314 Sexo: M	0	7



Operaciones – DICE

- Igual al anterior, pero realizar un DICE para mostrar sólo pares de personas si es que las mismas residen en el mismo país pero en distintas ciudades.

Operaciones – DICE

- Igual al anterior, pero realizar un DICE para mostrar sólo pares de personas si es que las mismas residen en el mismo país pero en distintas ciudades.

```
MATCH (e1 :Pais)<-[:perteneceA]-(c1 :Ciudad)<-[:resideEn]-(x :Persona)-[r1 :registra]-> (n1 :Tel)
```

```
MATCH (e2 :Pais)<-[:perteneceA]-(c2 :Ciudad)<-[:resideEn]-(y :Persona)-[r2 :registra]-> (n2 :Tel)
```

```
MATCH (n1 :Tel) -[r ]- (n2: Tel)
```

```
WHERE x.Nombre <= y.Nombre AND e1.Nombre = e2.Nombre AND c1.Nombre<>c2.Nombre
```

```
RETURN x as Persona1, y as Persona2, coalesce(AVG(r.Duracion), 0) as PromedioDuracion,  
coalesce(AVG(r.Longitud), 0) As PromedioLongitud
```

```
ORDER BY x.Nombre
```

Operaciones – DICE

MATCH (e1 :Pais)<-[:perteneceA]-(c1 :Ciudad)<-[:resideEn]-(x :Persona)-[r1 :registra]-> (n1 :Tel)

MATCH (e2 :Pais)<-[:perteneceA]-(c2 :Ciudad)<-[:resideEn]-(y :Persona)-[r2 :registra]-> (n2 :Tel)

MATCH (n1 :Tel) -[r]- (n2: Tel)

WHERE x.Nombre <= y.Nombre AND e1.Nombre = e2.Nombre AND c1.Nombre<>c2.Nombre

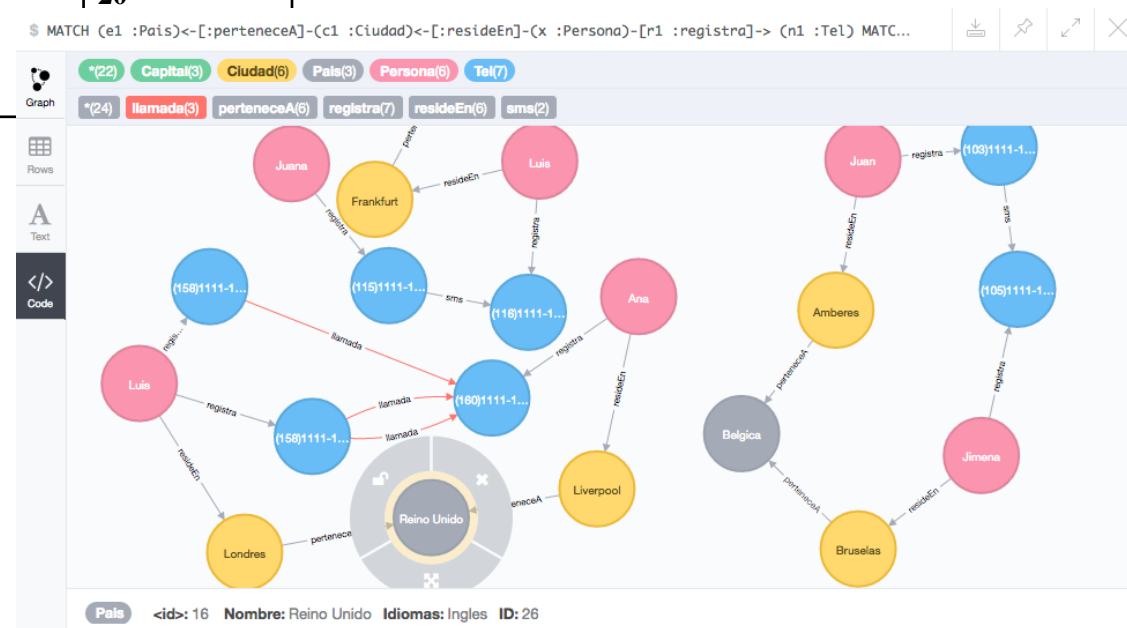
RETURN x as Persona1, y as Persona2, coalesce(AVG(r.Duracion), 0) as PromedioDuracion,
coalesce(AVG(r.Longitud), 0) As PromedioLongitud

ORDER BY x.Nombre

Persona1	Persona2	PromedioDuracion	PromedioLongitud
Nombre: Ana ID: 315 Sexo: F	Nombre: Luis ID: 313 Sexo: M	4.666666666666667	0
ID: 303 Nombre: Jimena Sexo: F	ID: 300 Nombre: Juan Sexo: M	0	2
ID: 305 Nombre: Juana Sexo: F	ID: 306 Nombre: Luis Sexo: M	0	20

Operaciones – DICE

Persona1	Persona2	PromedioDuracion	PromedioLongitud
Nombre: Ana ID: 315 Sexo: F	Nombre: Luis ID: 313 Sexo: M	4.66666666666667	0
ID: 303 Nombre: Jimena Sexo: F	ID: 300 Nombre: Juan Sexo: M	0	2
ID: 305 Nombre: Juana Sexo: F	ID: 306 Nombre: Luis Sexo: M	0	20



Operaciones – DICE sobre métricas

- Como la anterior, pero realizando un DICE sobre métricas, mostrando aquellas con un valor mayor a 4

Operaciones – DICE sobre métricas

- Como la anterior, pero realizando un DICE sobre métricas, mostrando aquellas con un valor mayor a 4

```
MATCH (e1 :Pais)<-[:perteneceA]-(c1 :Ciudad)<-[:resideEn]-(x :Persona)-[r1 :registra]-> (n1 :Tel)
```

```
MATCH (e2 :Pais)<-[:perteneceA]-(c2 :Ciudad)<-[:resideEn]-(y :Persona)-[r2 :registra]-> (n2 :Tel)
```

```
MATCH (n1 :Tel) -[r ]- (n2: Tel)
```

```
WHERE x.Nombre <= y.Nombre AND e1.Nombre = e2.Nombre AND c1.Nombre<>c2.Nombre
```

```
WITH x as Persona1, y as Persona2, coalesce(AVG(r.Duracion), 0) as PromedioDuracion,  
coalesce(AVG(r.Longitud), 0) As PromedioLongitud
```

```
WHERE PromedioDuracion >= 4 OR PromedioLongitud >= 4
```

```
RETURN Persona1, Persona2, PromedioDuracion, PromedioLongitud
```

```
ORDER BY Persona1.Nombre
```

Operaciones – DICE sobre métricas

MATCH (e1 :Pais)<-[:perteneceA]-(c1 :Ciudad)<-[:resideEn]-(x :Persona)-[r1 :registra]-> (n1 :Tel)

MATCH (e2 :Pais)<-[:perteneceA]-(c2 :Ciudad)<-[:resideEn]-(y :Persona)-[r2 :registra]-> (n2 :Tel)

MATCH (n1 :Tel) -[r]- (n2: Tel)

WHERE x.Nombre <= y.Nombre AND e1.Nombre = e2.Nombre AND c1.Nombre<>c2.Nombre

WITH x as Persona1, y as Persona2, coalesce(AVG(r.Duracion), 0) as PromedioDuracion, coalesce(AVG(r.Longitud), 0)
As PromedioLongitud

WHERE PromedioDuracion >= 4 OR PromedioLongitud >= 4

RETURN Persona1, Persona2, PromedioDuracion, PromedioLongitud

ORDER BY Persona1.Nombre

Persona1	Persona2	PromedioDuracion	PromedioLongitud
Nombre: Ana ID: 315 Sexo: F	Nombre: Luis ID: 313 Sexo: M	4.666666666666667	0
ID: 305 Nombre: Juana Sexo: F	ID: 306 Nombre: Luis Sexo: M	0	20

Ejercicio

- Implementar una jerarquía típica para la parte temporal. Ej: Niveles Quarter, Year.
- ¿Cómo se pueden representar las instancias de tiempo en cada nivel?
- Proponer una consulta que permita calcular un RollUp temporal al nivel Year y resolverla.