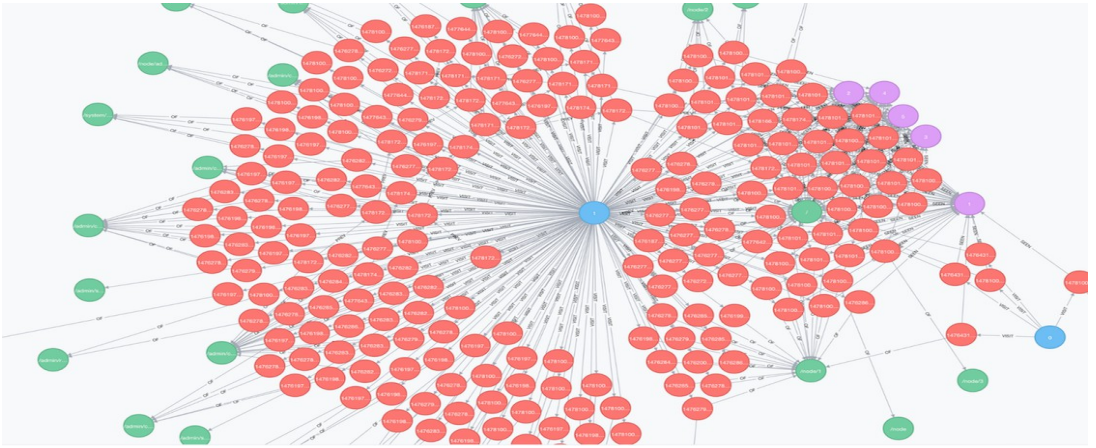
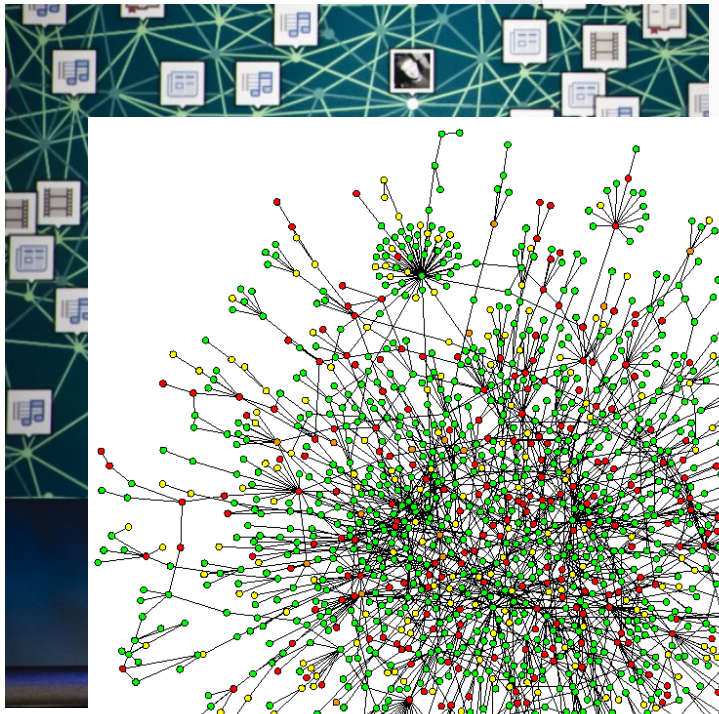
Bases de datos de grafos

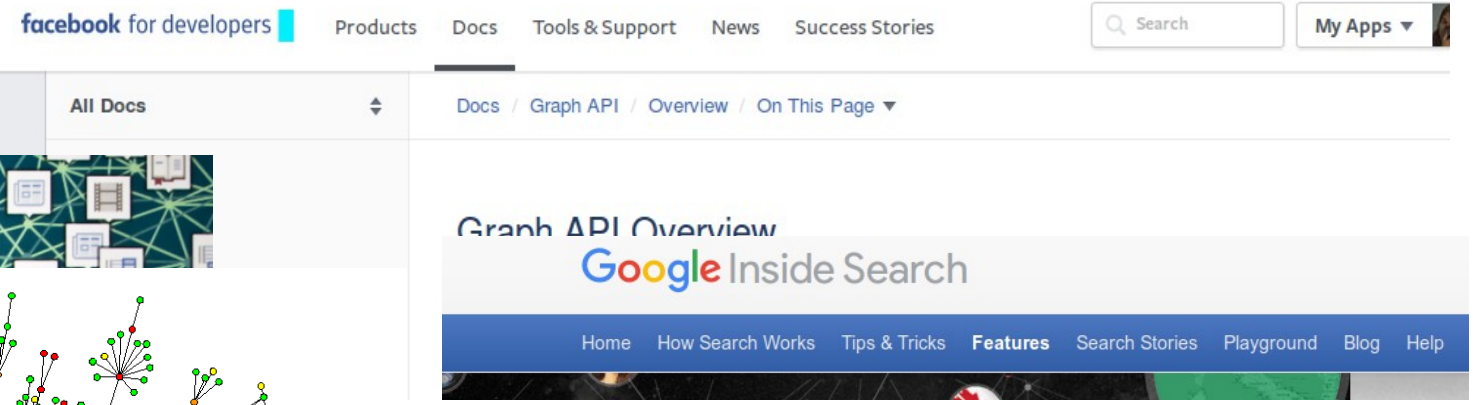


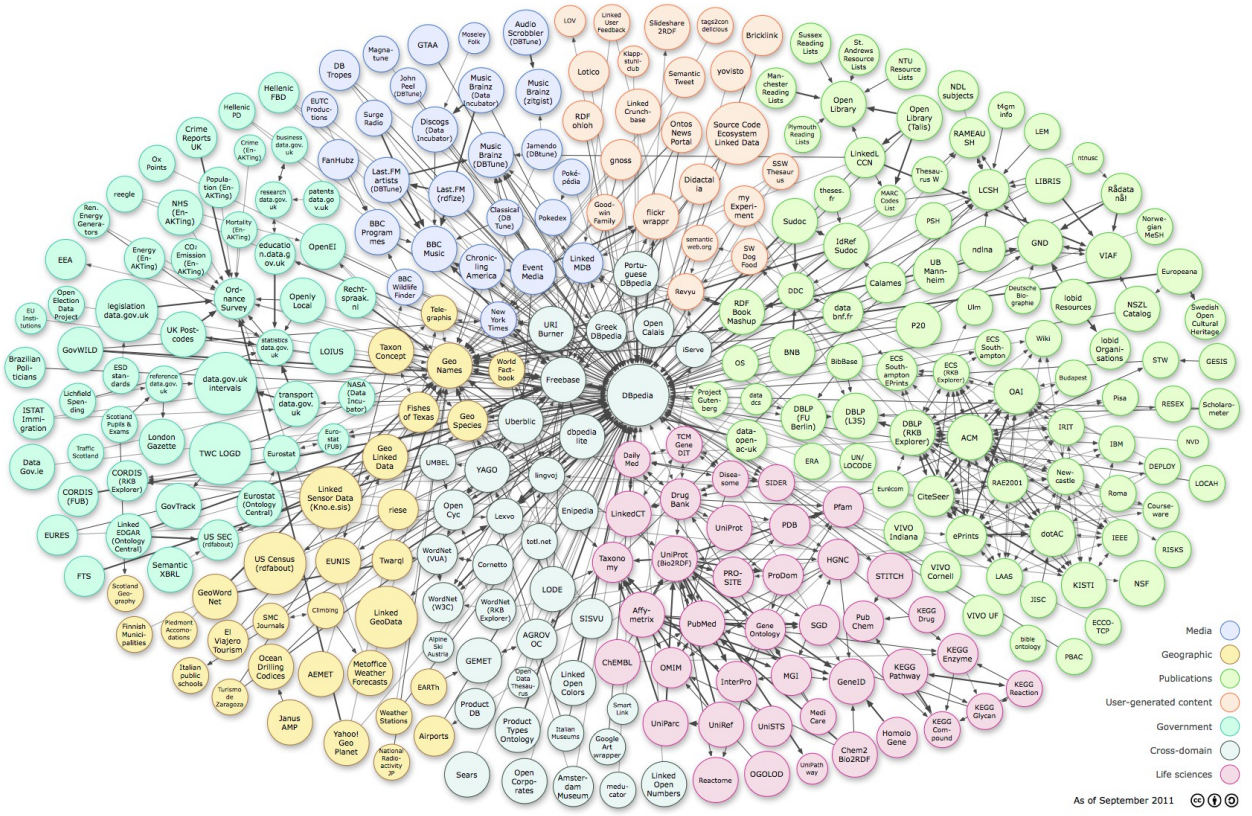
CPAP, FING, UdelaR – 2020

Modelado y Procesamiento de Grandes Volumenes de Datos

¡Hay casos en que interesa más **modelar** las **relaciones** entre cosas que las cosas!







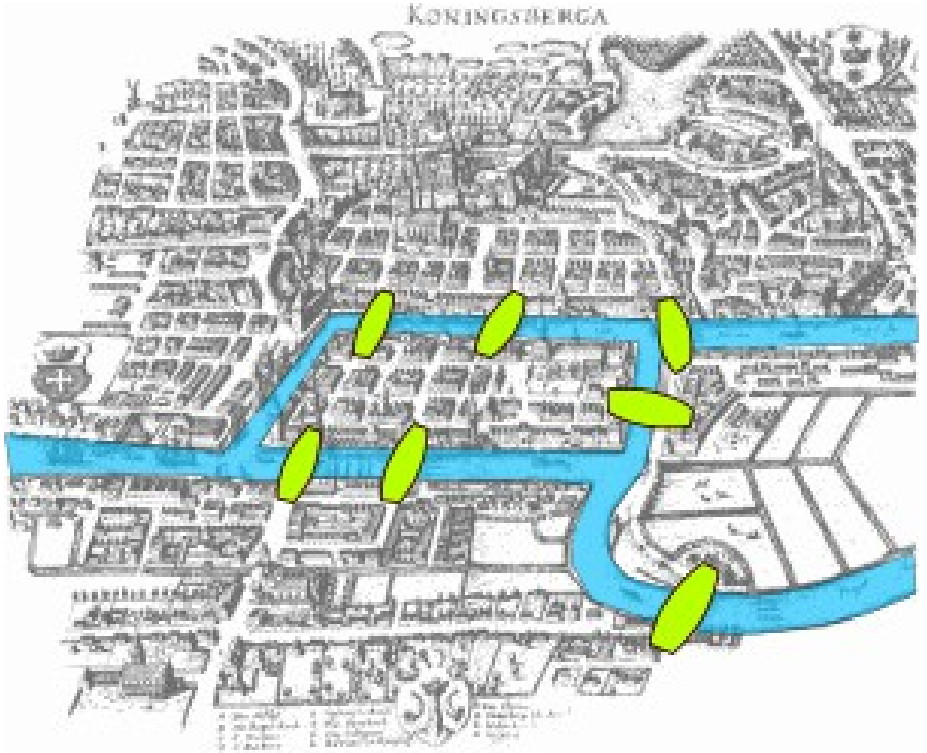
Agenda

- Bases de datos de grafos
  - Modelos de datos
- Grafos y la web semántica
  - Estrategias de almacenamiento
  - Lenguajes de consulta
- Otras bases de datos de grafos
  - Estrategias de almacenamiento
  - Lenguajes de consulta
  - Modelos de procesamiento distribuído

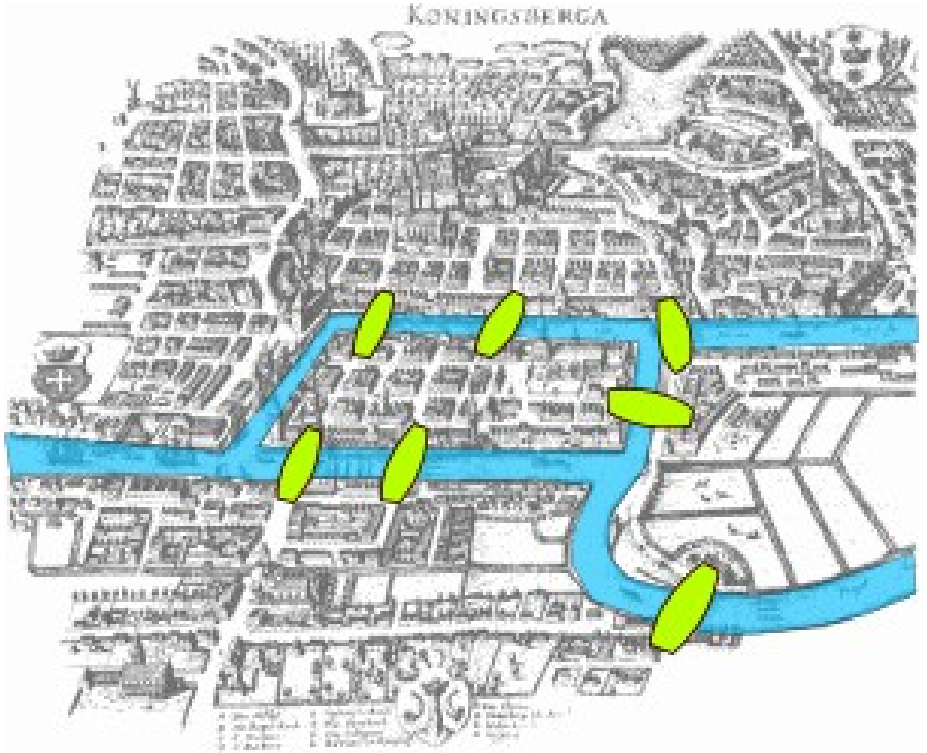
Agenda

- Bases de datos de grafos
  - Modelos de datos
- Grafos y la web semántica
  - Estrategias de almacenamiento
  - Lenguajes de consulta
- Otras bases de datos de grafos
  - Estrategias de almacenamiento
  - Lenguajes de consulta
  - Modelos de procesamiento distribuído

Problema de los puentes de Königsberg (I)



Problema de los puentes de Königsberg (I)



Problema de los puentes de Königsberg (II)

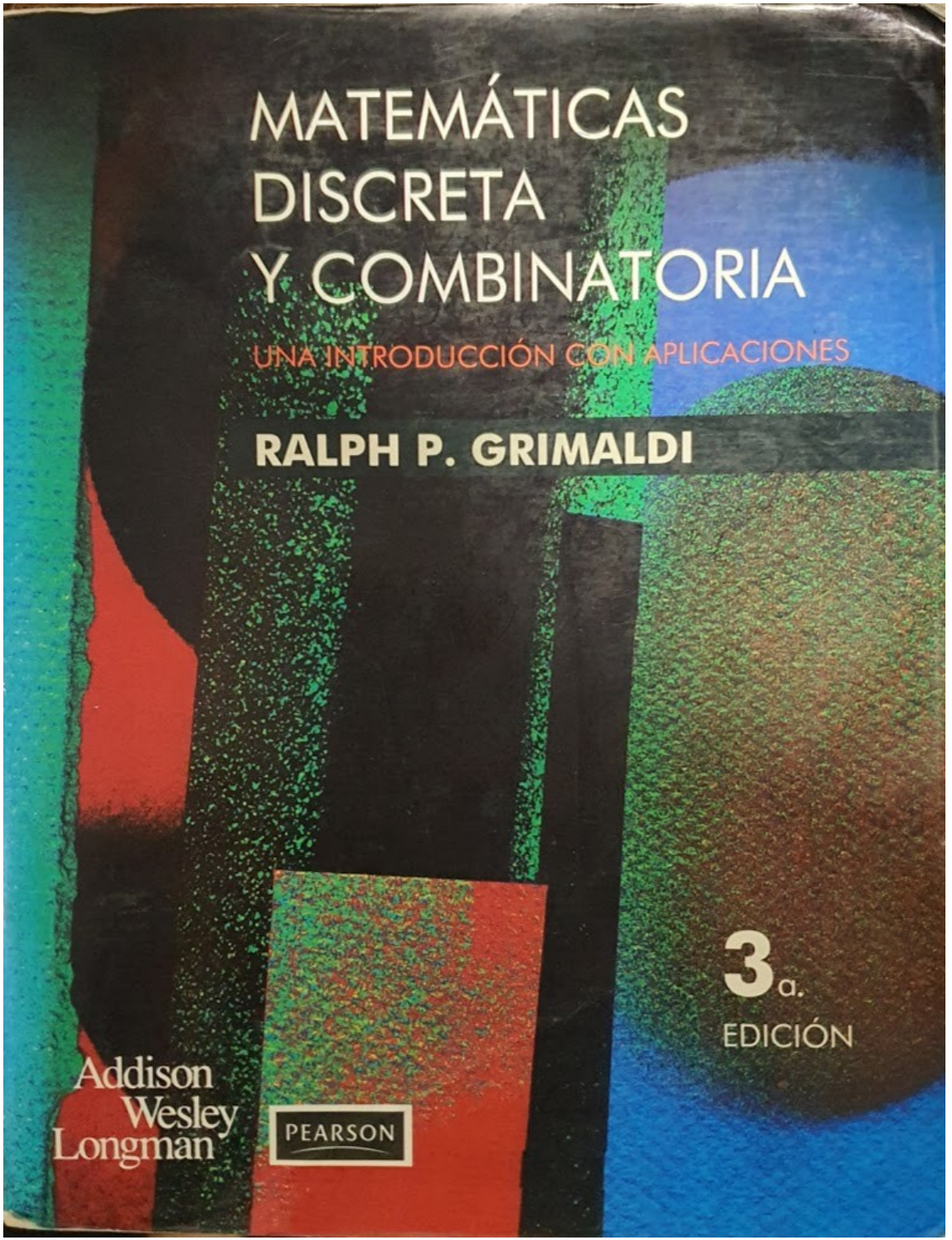
Problema de los puentes de Königsberg (III)

¿Qué es un grafo?

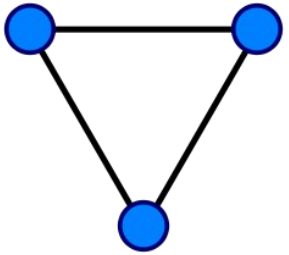
Un grafo G consiste en
un conjunto de nodos o vértices V,
y un conjunto de aristas E.
Las aristas conectan nodos entre si.

Las bases de datos de grafos implementan diferentes variantes[1].

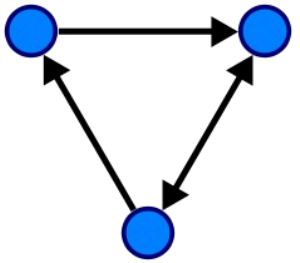
[1] *Survey of Graph Database Models*, Angles and Gutierrez, ACM Computing Surveys, 2008


MATEMÁTICAS
DISCRETA
Y COMBINATORIA
UNA INTRODUCCIÓN CON APLICACIONES
RALPH P. GRIMALDI
3a.
EDICIÓN
Addison
Wesley
Longman
PEARSON

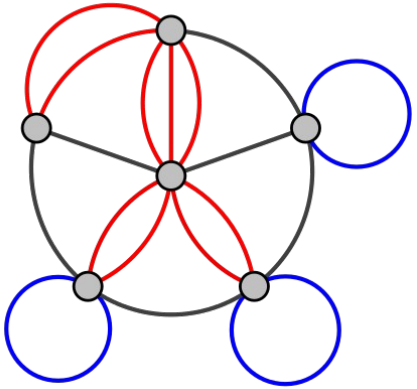
Grafos según su estructura



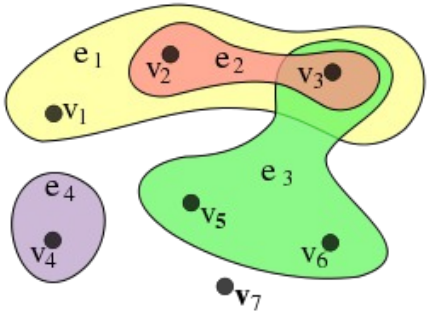
Grafo no dirigido



Grafo dirigido

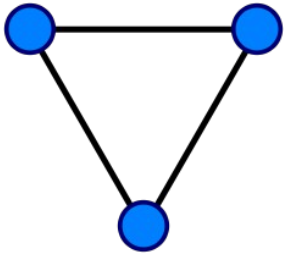


Multigrafos y pseudografos

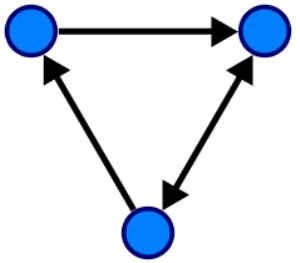


Hipergrafos

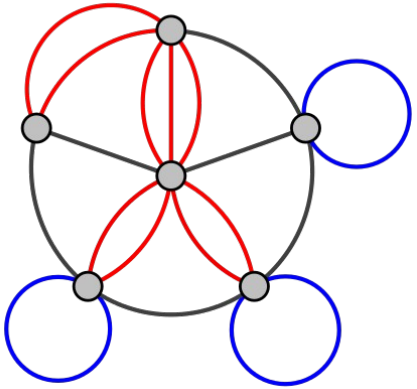
Grafos según su estructura



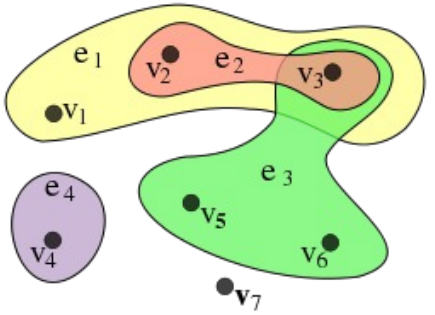
Grafo no dirigido



Grafo dirigido

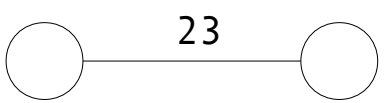


Multigrafos y pseudografos



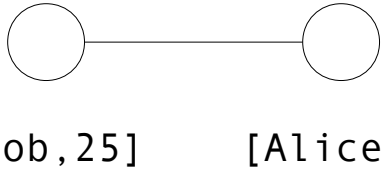
Hipergrafos

Grafos según los datos asociados



Etiquetas en las aristas

Etiquetas en los nodos



Atributos en los nodos

¿qué tipos de grafos se representan en las bases de datos de grafos?

¿qué tipos de grafos se representan en las bases de datos de grafos?

Existen muchos modelos

(ver *Survey of Graph Database Models*, Angles and Gutierrez, ACM Computing Surveys, 2008)

Los sistemas más populares actualmente implementan

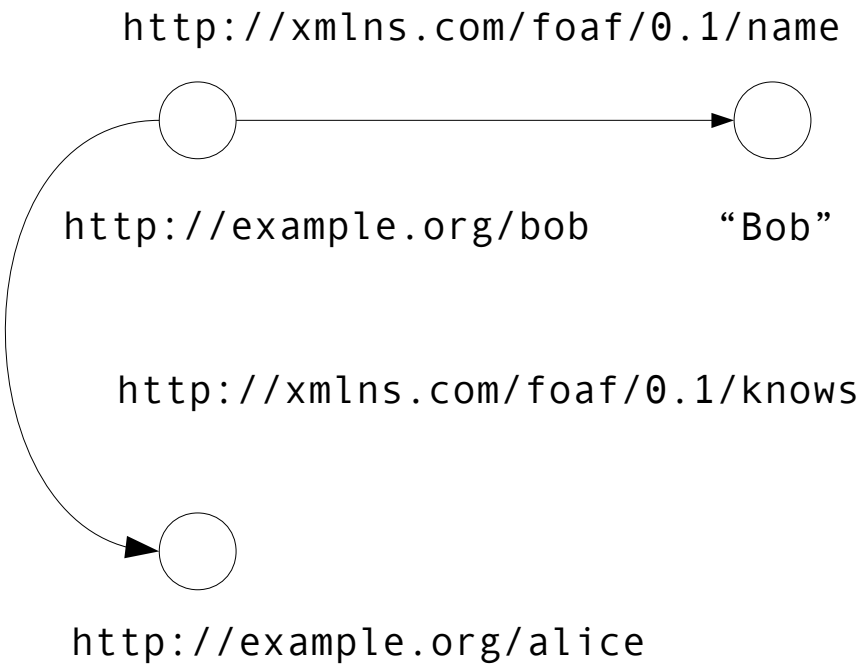
RDF o property graphs

Resource Description Framework (RDF)

Multigrafos dirigidos.

Etiquetas en nodos y aristas.

Las etiquetas pueden ser: IRIs, literales, o nodos blancos.

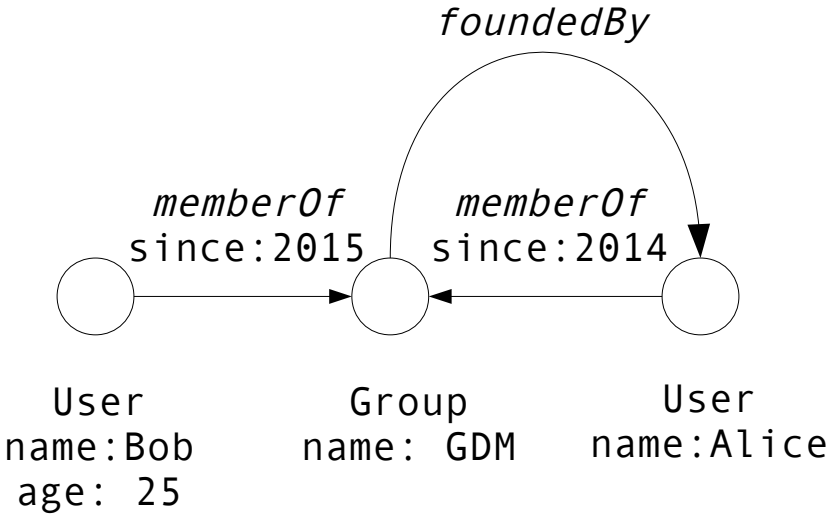


Virtuoso y Stardog son ejemplos de sistemas de bases de datos que soportan RDF.

Property Graph Model (PGM)

Multigrafos dirigidos.

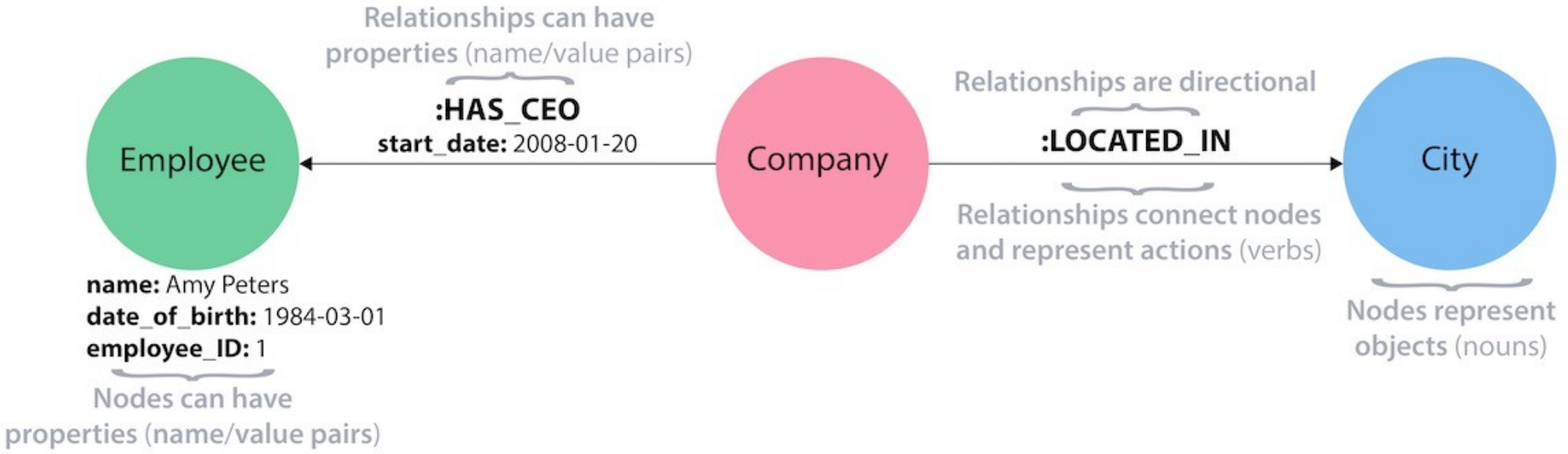
Parejas (clave,valor) llamadas *propiedades* asociadas a nodos y aristas.



Además puedo etiquetar nodos y aristas.

Neo4j y Titan son ejemplos de sistemas de bases de datos que soportan PGM.

Property Graph Model (PGM)



Grafos y la web semántica

La web clásica es una red de

documentos,

interpretable por **humanos**,

donde las relaciones entre documentos no tienen un

significado.

- SNAP for C++
- SNAP for Python
- SNAP Datasets
- BIOSNAP Datasets
- What's new
- People
- Papers
- Projects
- Citing SNAP
- Links
- About
- Contact us

Open positions

Open research positions in **SNAP** group are available at **undergraduate, graduate** and **postdoctoral** levels.

Google web graph

Dataset information

Nodes represent web pages and directed edges represent hyperlinks between them. The data was released in 2002 by Google as a part of Google Programming Contest.

Dataset statistics	
Nodes	875713
Edges	5105039
Nodes in largest WCC	855802 (0.977)
Edges in largest WCC	5066842 (0.993)
Nodes in largest SCC	434818 (0.497)
Edges in largest SCC	3419124 (0.670)
Average clustering coefficient	0.5143
Number of triangles	13391903
Fraction of closed triangles	0.01911
Diameter (longest shortest path)	21
90-percentile effective diameter	8.1

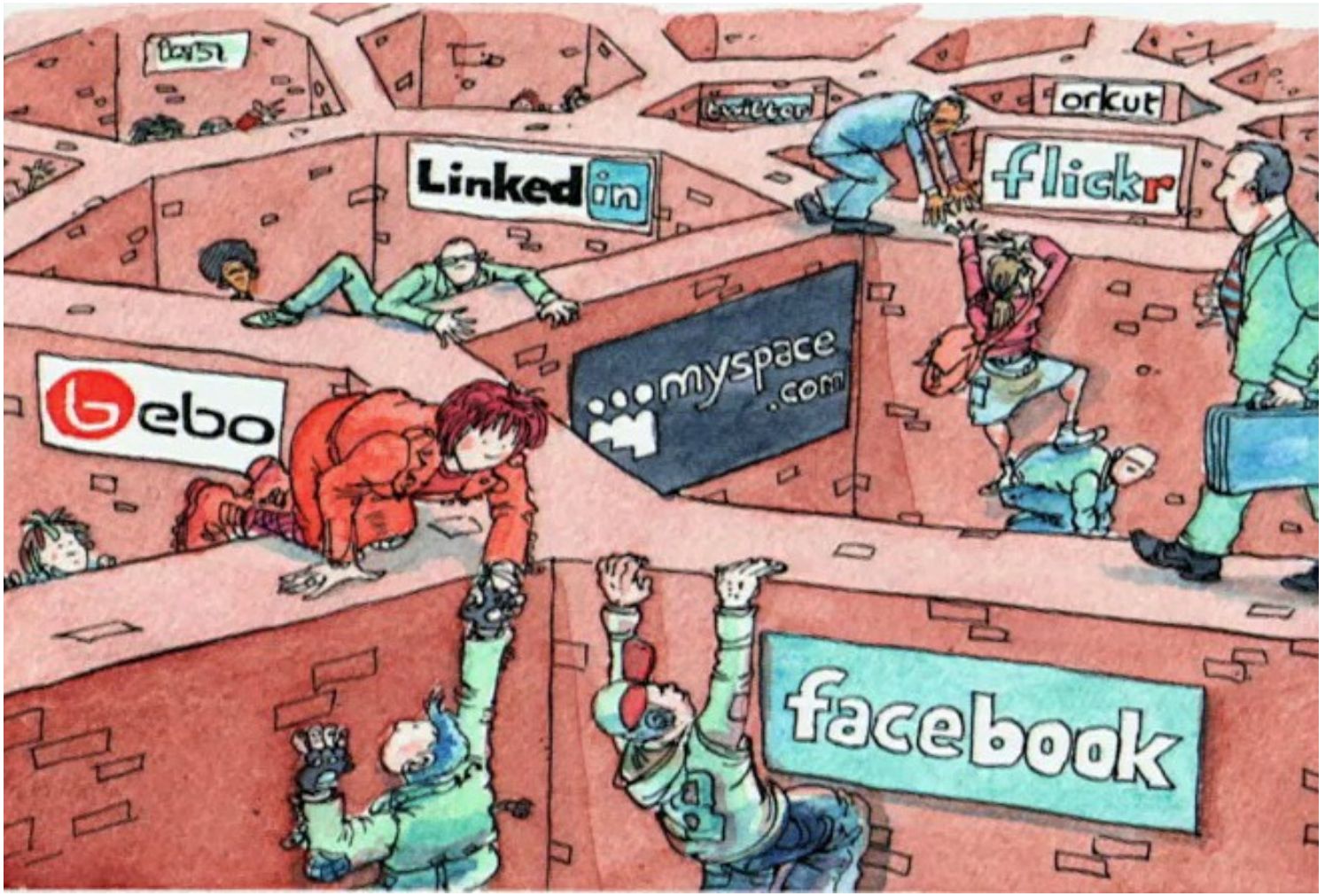
Source (citation)

- J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics 6(1) 29--123, 2009.
- Google programming contest, 2002

Files

File	Description
web-Google.txt.gz	Webgraph from the Google programming contest, 2002

Además, la mayoría de los datos en la web están a i s l a d o s



La web de datos es una red de afirmaciones, interpretable por humanos y máquinas, donde las afirmaciones se relacionan y tienen un significado.

RDF: un modelo de datos basado en triplas que permite representar relaciones.

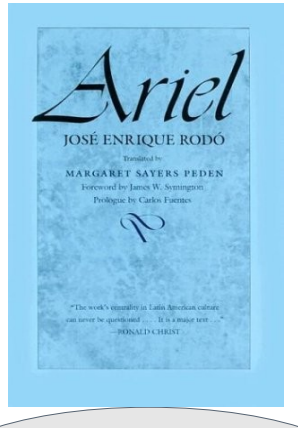
SPARQL: el lenguaje de consultas sobre RDF.

http://www.w3.org/2001/sw/wiki/RDF

http://www.w3.org/2001/sw/wiki/SPARQL

afirmación o tripla

<sujeto, predicado, objeto>

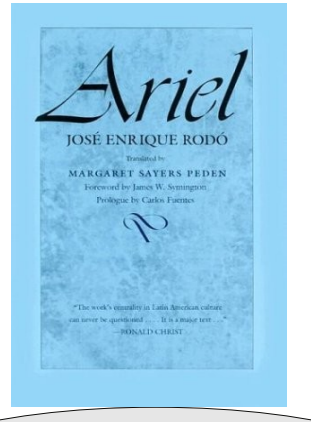


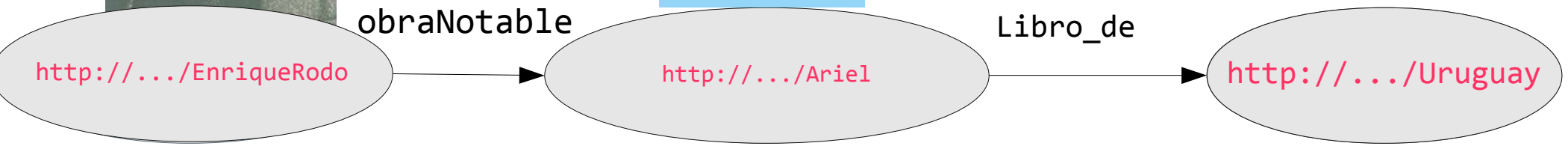
obraNotable

José E. Rodó → Ariel

afirmación o tripla

<sujeto, predicado, objeto>

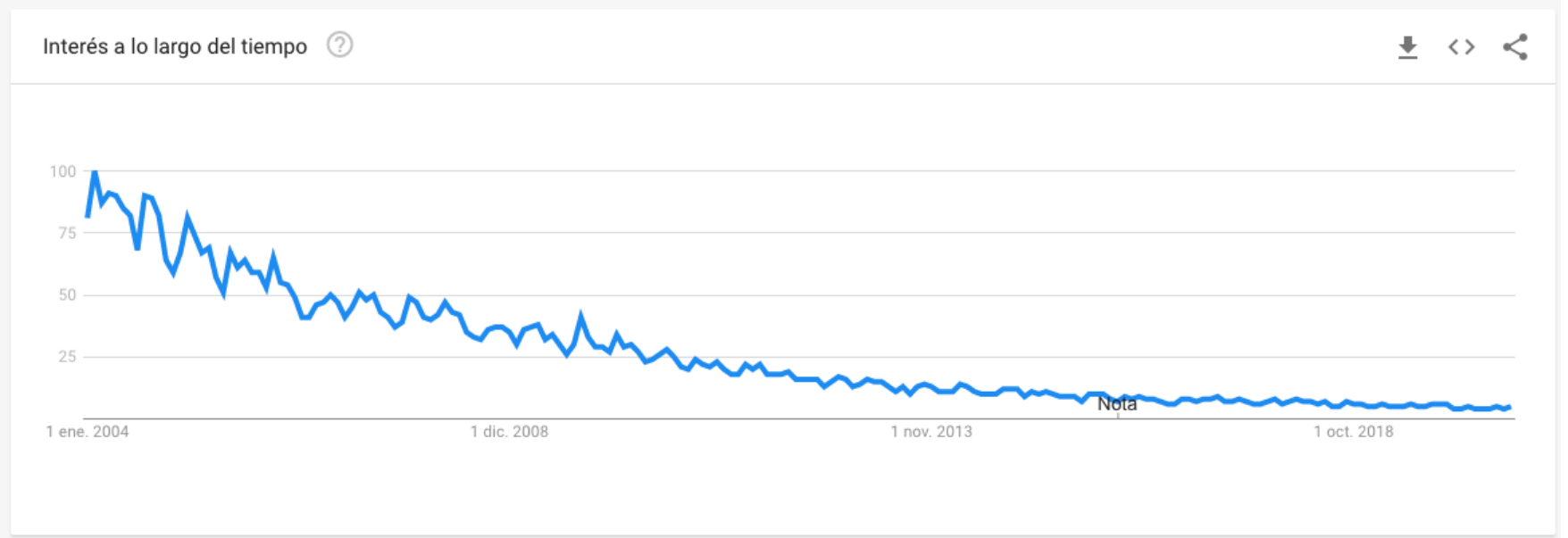




En RDF todo es una tripla.

Los recursos y las propiedades se identifican por URIs

Se habla menos de semantic web...



jose enrique rodo

Todo | Maps | Imágenes | Vídeos | Noticias | Más | Preferencias | Herramientas

Cerca de 608.000 resultados (0,61 segundos)

es.wikipedia.org › wiki › José_Enrique_Rodó

José Enrique Rodó - Wikipedia, la enciclopedia libre

José Enrique Camilo **Rodó** Piñeyro (Montevideo, Uruguay, 15 de julio de 1871 - Palermo, Italia, 1 de mayo de 1917) fue un escritor y político uruguayo. Creador ...

Nombre de nacimiento: José Enrique Camil… **Género**: Ensayo
Partido político: Partido Colorado **Nacionalidad**: Uruguayo

Biografía · Obras

Visitaste esta página 2 veces. Última visita: 16/08/20

es.wikipedia.org › wiki › José_Enrique_Rodó_(ciudad)

José Enrique Rodó (ciudad) - Wikipedia, la enciclopedia libre

José Enrique Rodó es una localidad uruguaya del departamento de Soriano y es sede del municipio homónimo. Índice. 1 Geografía; 2 Historia; 3 Servicios ...

Prefijo telefónico: +598 4538 XXXX **Código postal**: 75002

Historia · Población

www.academiadeletras.gub.uy › jose-enrique-rodo

José Enrique Rodó - Academia de Letras

José Enrique Rodó. **José Enrique Rodó**. Nació en Montevideo el 15 de julio de 1871 y falleció en Palermo (Sicilia) el 1° de mayo de 1917 ...

blogs.ceibal.edu.uy › formacion › jose-enrique-rodo

José Enrique Rodó | Plan Ceibal – Formación

José Enrique Rodó. Escritor, filósofo, profesor, crítico y político, nació el 15 de julio de 1871 en Montevideo. Falleció ...

28 abr. 2015 - Subido por Magdalena Lallo

www.cervantesvirtual.com › obra-visor › html

José Enrique Rodó : (1871-1917) | Biblioteca Virtual Miguel ...

Nace **José Enrique Rodó**, el 15 de julio en Montevideo (Uruguay), en el seno de una familia acomodada. Su padre era un comerciante catalán y su madre ...

internet.com.uy › biografias › biorodo › biorodo

José Enrique Rodó

Escritor

José Enrique Camilo Rodó Piñeyro fue un escritor y político uruguayo. Creador del arielismo, corriente ideológica basada en un aprecio de la tradición grecolatina, sus obras expresaron el malestar finisecular hispanoamericano con un estilo refinado y poético, típico del modernismo. Wikipedia

Fecha de nacimiento: 15 de julio de 1871, Montevideo

Fallecimiento: 1 de mayo de 1917, Palermo, Italia

Obras notables: Ariel

Movimiento: Generación del ´900

Seudónimo: Maestro de la juventud

Libros

Ver 20 más

Ariel
1900

Motivos de Proteo
1909

Ariel: El Camino De Paros
2012

La vida nueva
1897

Ariel ; Motivos de Proteo

quien es el autor de ariel

Todo | Imágenes | Vídeos | Noticias | Maps | Más | Preferencias | Herramientas

Cerca de 15.600.000 resultados (0,52 segundos)

Ariel / Autor

José Enrique Rodó

Ariel es un ensayo publicado por el escritor uruguayo **José Enrique Rodó** en 1900, considerado como una de las obras de mayor influencia en el campo de la cultura y la política latinoamericanas, a pesar de ser esta una modesta copia del Caliban (1878) de Ernest Renan y los postulados modernistas de Rubén Darío.

es.wikipedia.org › wiki › Ariel_(ensayo)

Ariel (ensayo) - Wikipedia, la enciclopedia libre

Otras personas también buscan

Ver 10 más


Delmira Agustini


Francisco Espínola


Emilio Frugoni


José Martí


Leopoldo Lugones


Manuel Gutiérrez Nájera


José Santos Chocano

Comentarios

es.wikipedia.org › wiki › Ariel_(ensayo)

Ariel (ensayo) - Wikipedia, la enciclopedia libre

Ariel es un ensayo publicado por el **escritor** uruguayo José Enrique Rodó en 1900, considerado como una de las obras de mayor influencia en el campo de la ...

Estilo literario · Contenido filosófico · Recepción de la obra

Linked Data

es un conjunto de
buenas prácticas para
publicar y *relacionar*
datos en la web,
usando
tecnologías de la
web semántica

Principios de Linked Data

1) Usar **URIs** para nombrar cosas

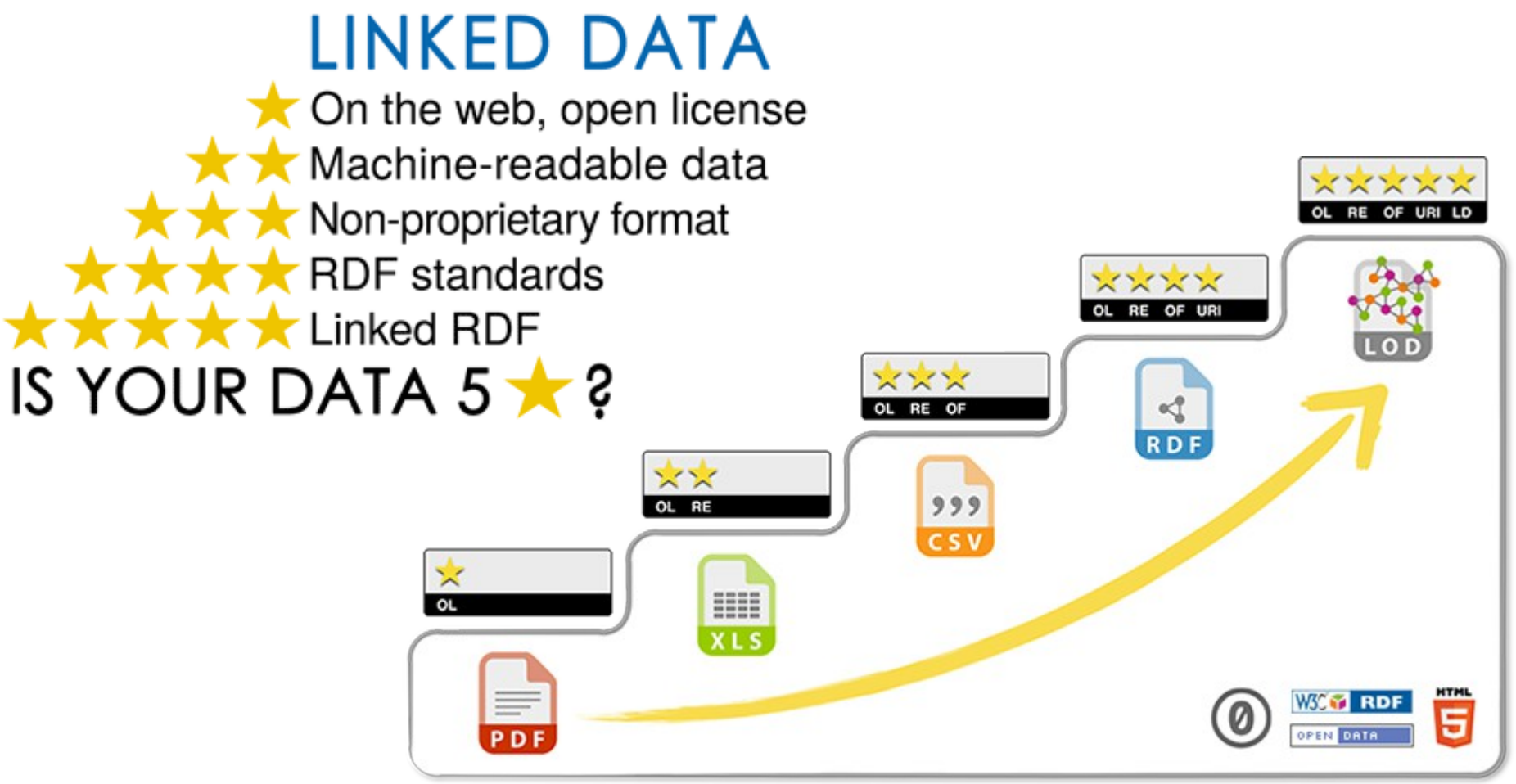
2) Usar **URIs HTTP** que sean consultables por humanos y máquinas

3) Proveer información **útil** acerca de cada URI en RDF

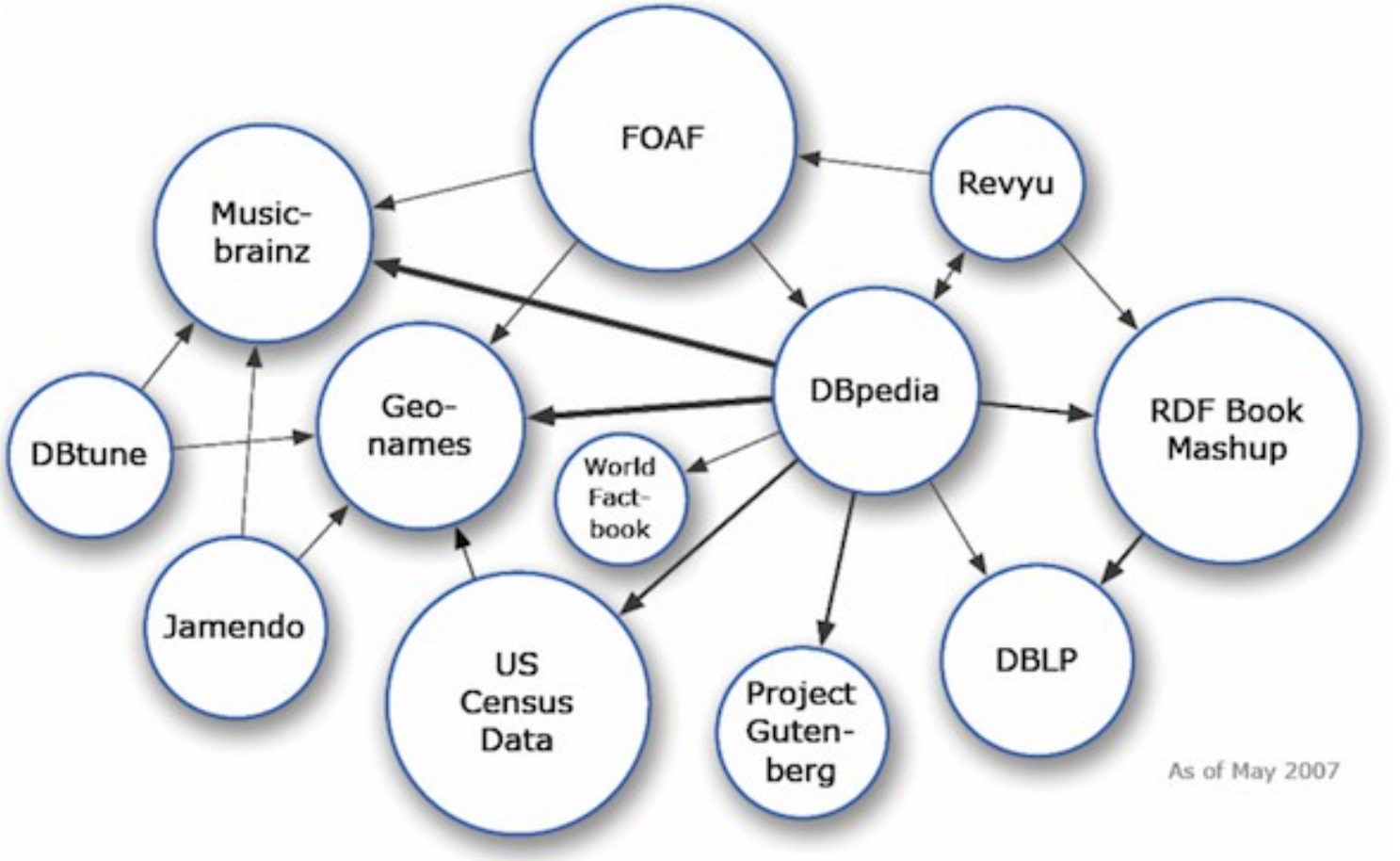
4) Crear **links** entre URIs

Tim Berners-Lee – Linked Data (2006)
http://www.w3.org/DesignIssues/LinkedData
#TED Talk Tim Berners-Lee on the next Web (2009)
http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html

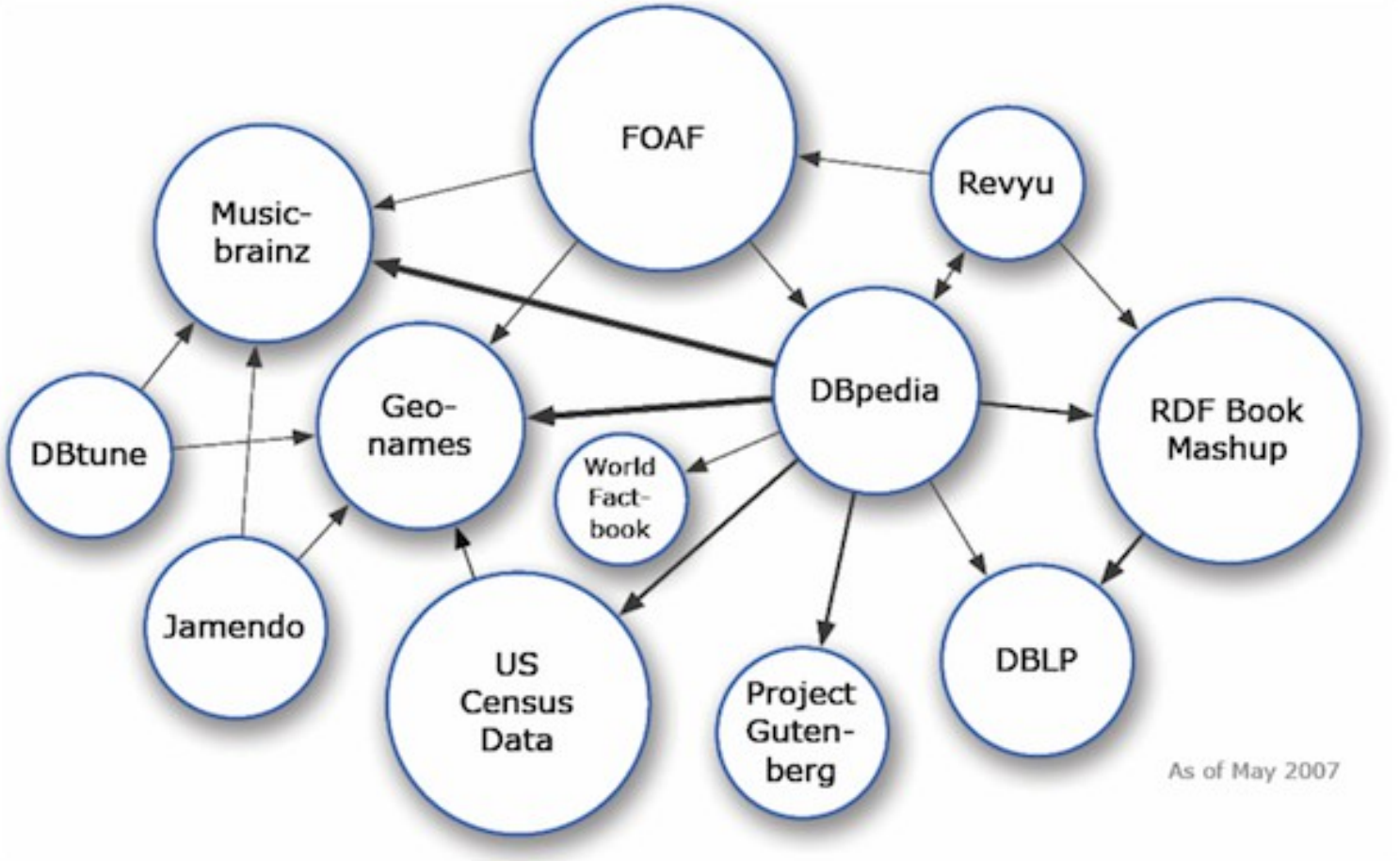
Linked Data + Open Data
= LOD



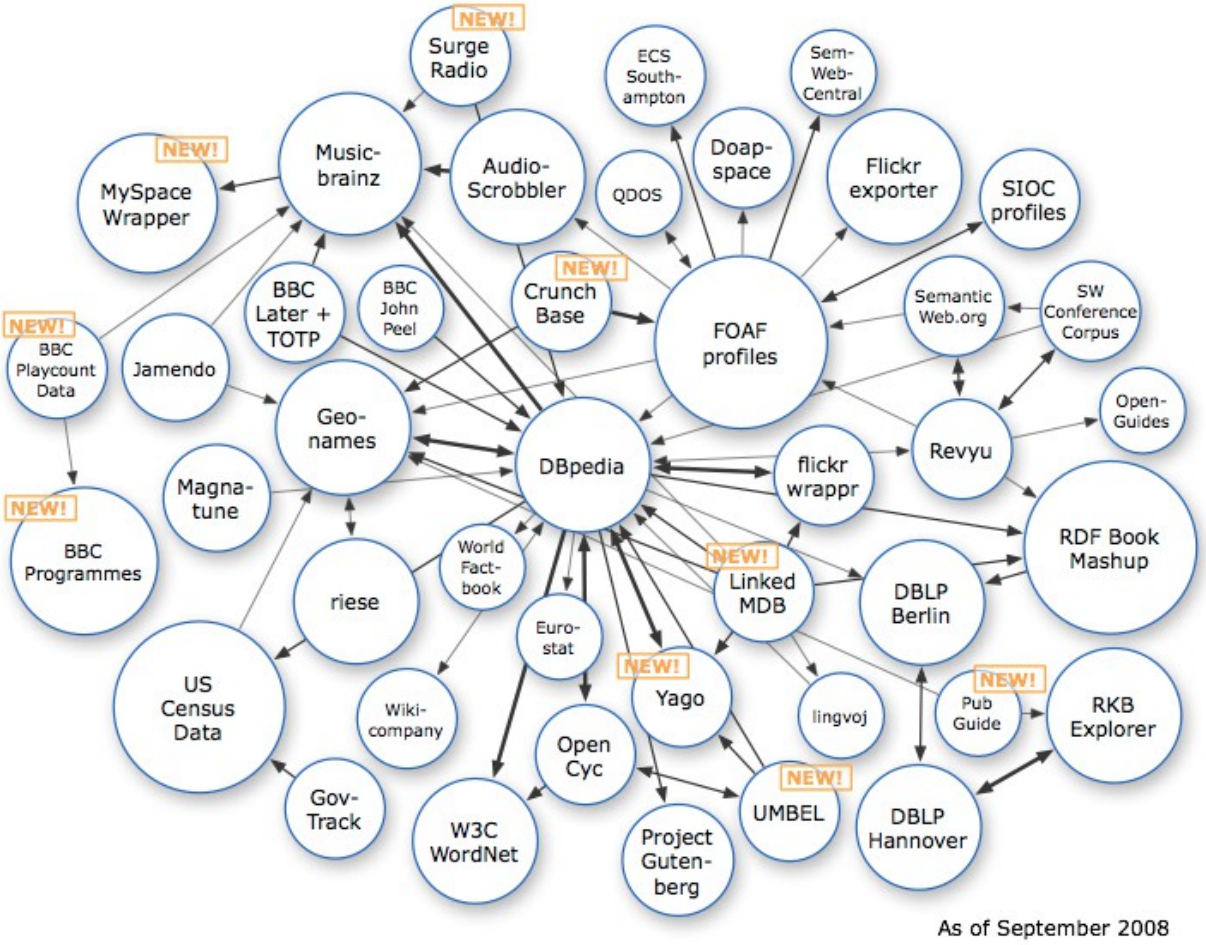
http://5stardata.info/en/



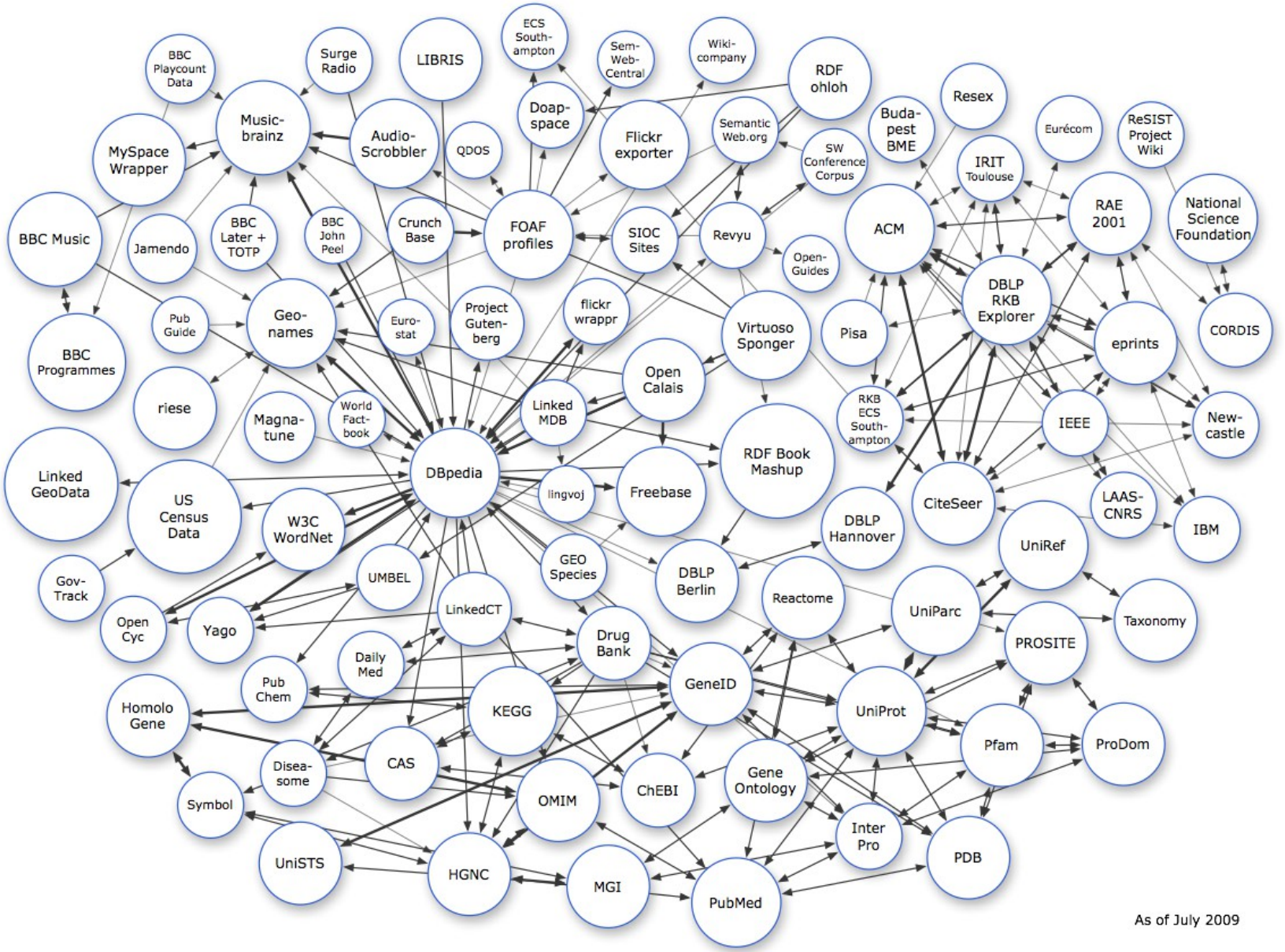
http://lod-cloud.net/ 2007



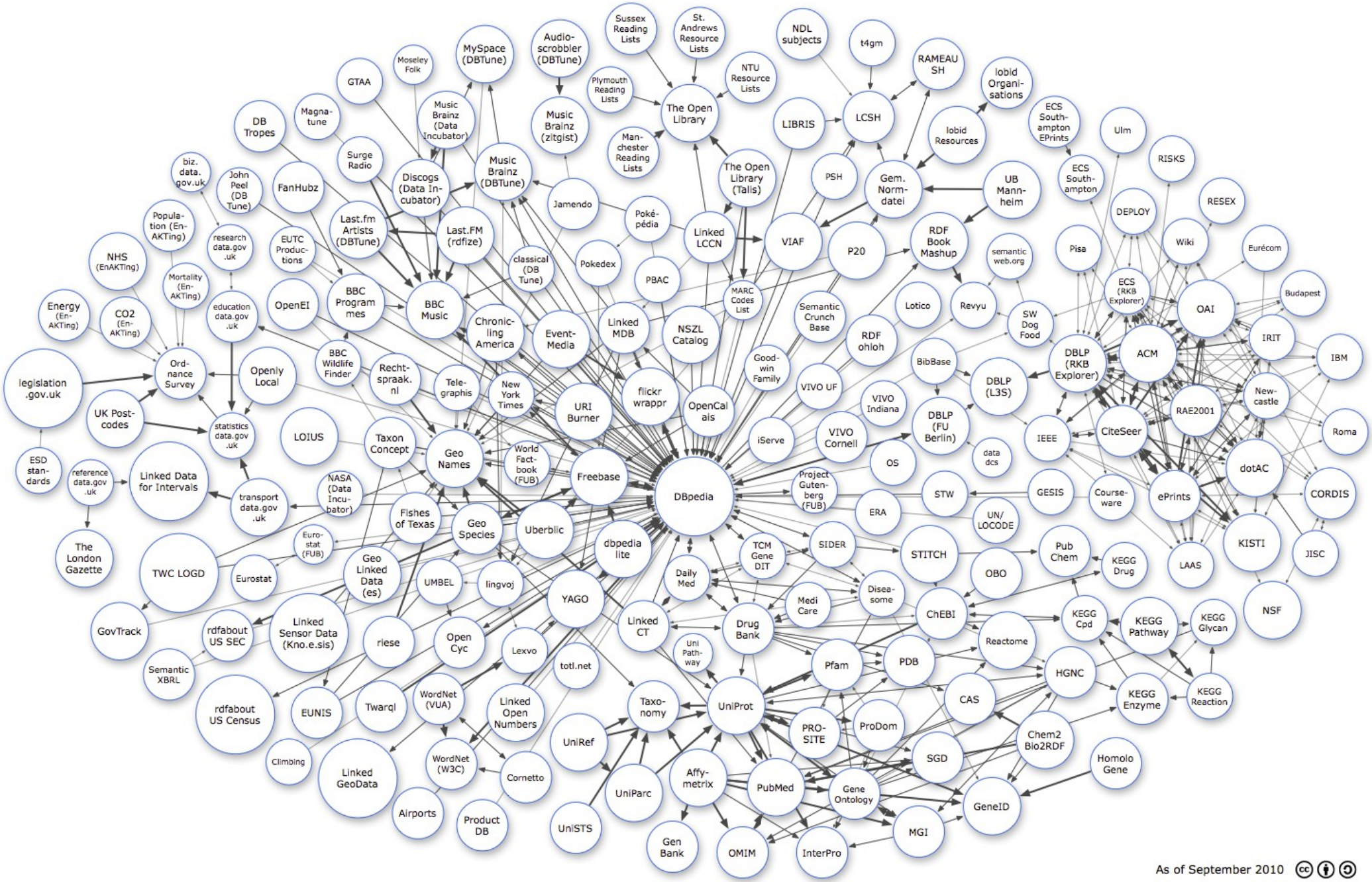
http://lod-cloud.net/ 2007



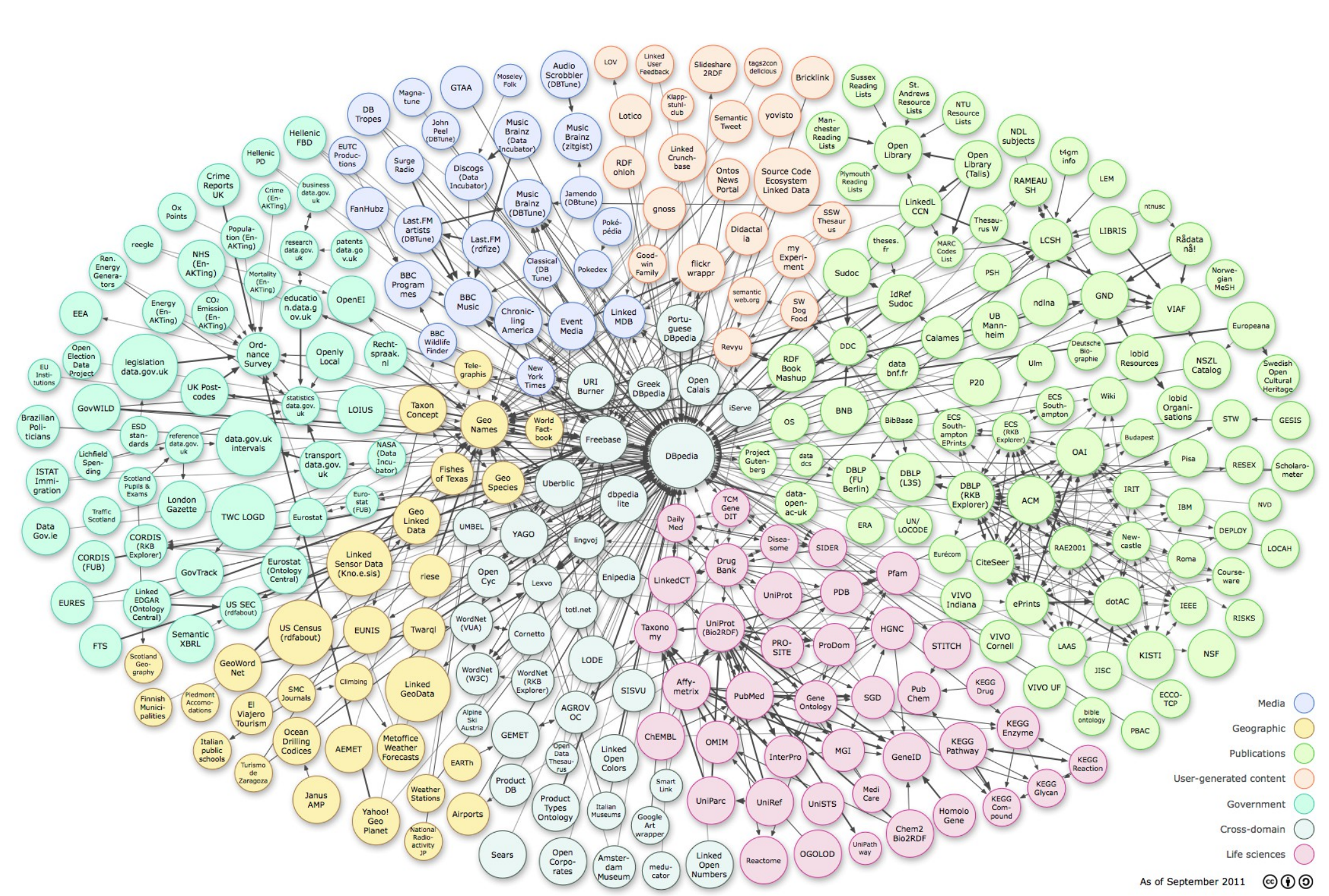
http://lod-cloud.net/ 2008



http://lod-cloud.net/ 2009



http://lod-cloud.net/ 2010



http://lod-cloud.net/ 2011

Almacenamiento de RDF

- Los datos RDF se almacenan en *triplestores*.
- Algunos se implementan sobre bases de datos relacionales : ej OpenLink Virtuoso
- Otros almacenan el grafo en forma “nativa”

Para los curiosos

- Tutorial SPARQL by Example

  http://www.cambridgesemantics.com/semantic-university/sparql-by-example

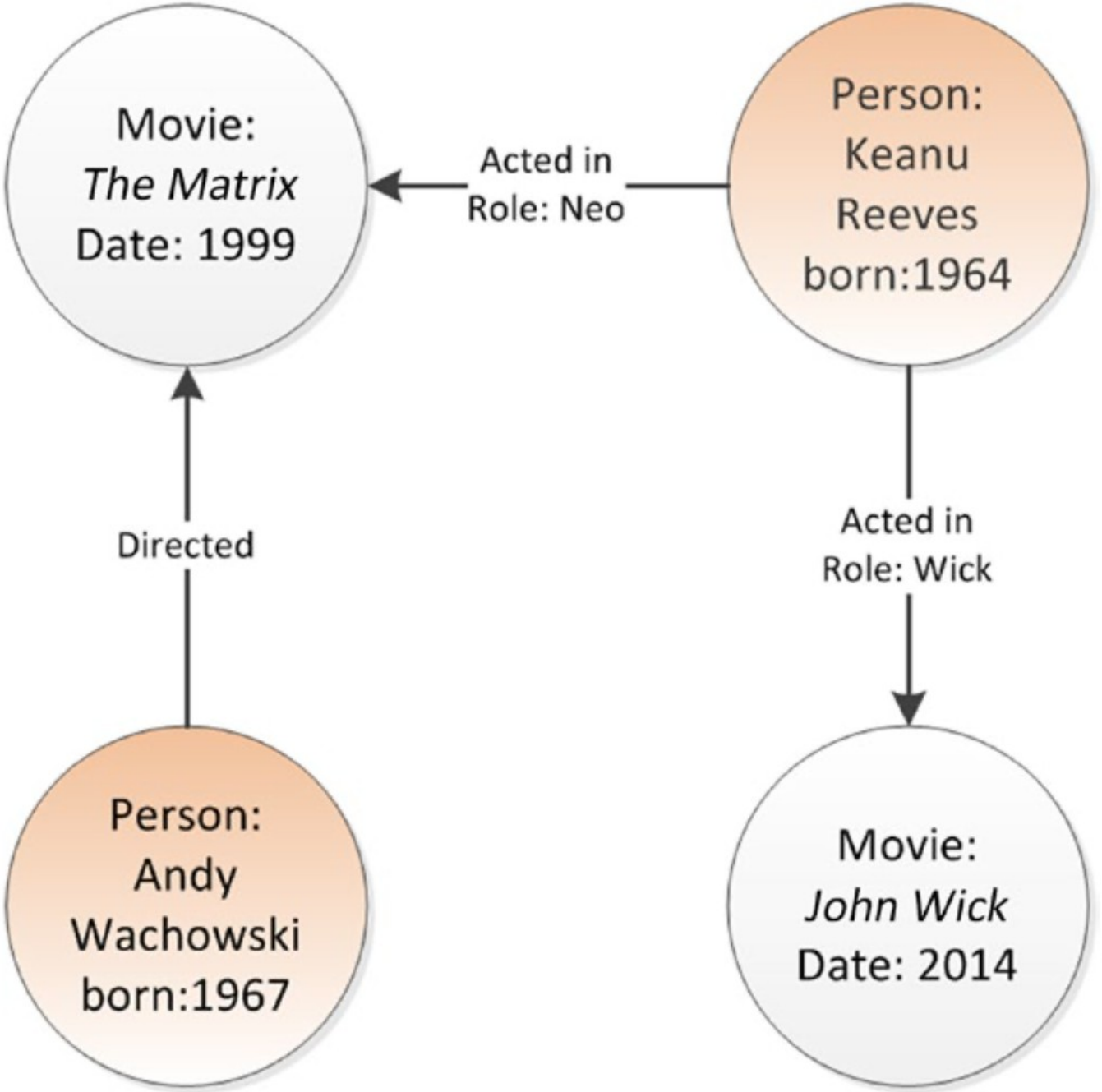
- Curso Linked Data Engineering del Hasso-Plattner Institut

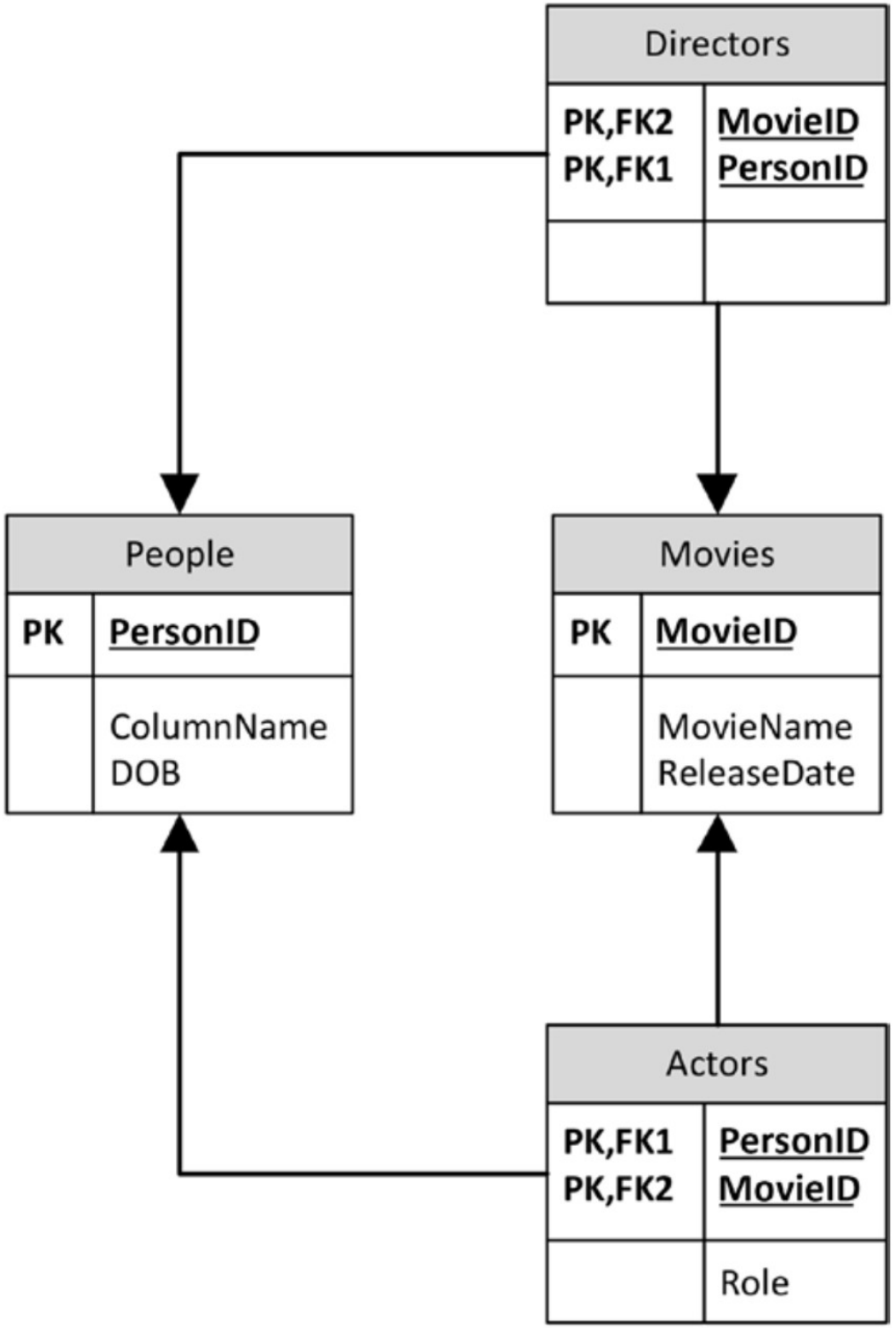
  https://open.hpi.de/courses/semanticweb2016

¿Qué es Neo4j?

- Una base de datos de grafos + índices Lucene
- Open source (Java)
- Representa *property graphs* en forma **nativa**
- Soporta ACID
- Cantidad "acotada" de elementos:
  - 32k millones de nodos
  - 32k millones de relaciones
  - 64k millones de propiedades
- Cypher como lenguaje de consulta +APIs

Volvamos al modelado

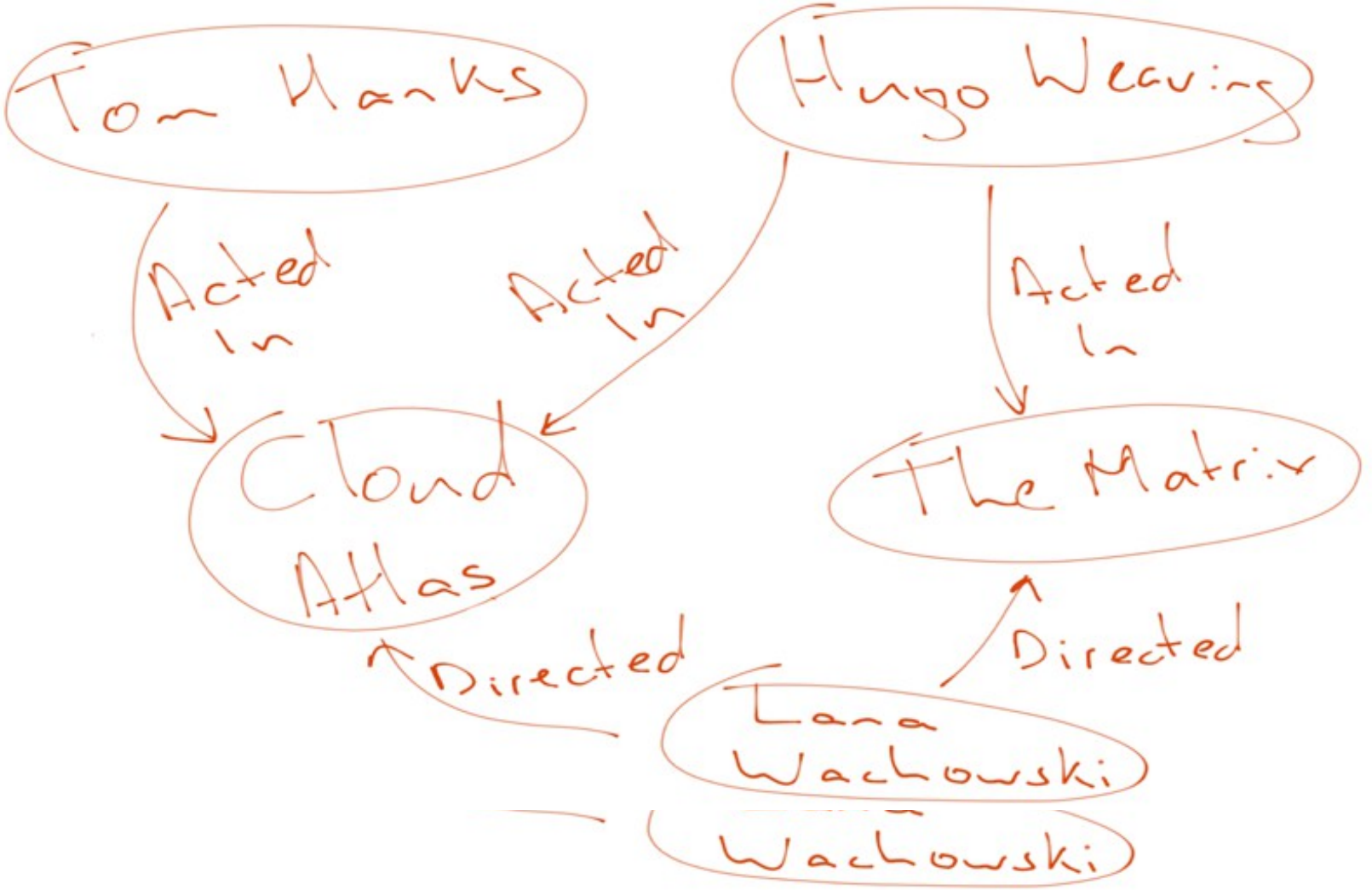

Movie:
The Matrix
Date: 1999
Acted in
Role: Neo
Person:
Keanu
Reeves
born:1964
Directed
Acted in
Role: Wick
Person:
Andy
Wachowski
born:1967
Movie:
John Wick
Date: 2014


Directors
PK,FK2
PK,FK1
MovieID
PersonID
People
PK
PersonID
ColumnName
DOB
Movies
PK
MovieID
MovieName
ReleaseDate
Actors
PK,FK1
PK,FK2
PersonID
MovieID
Role

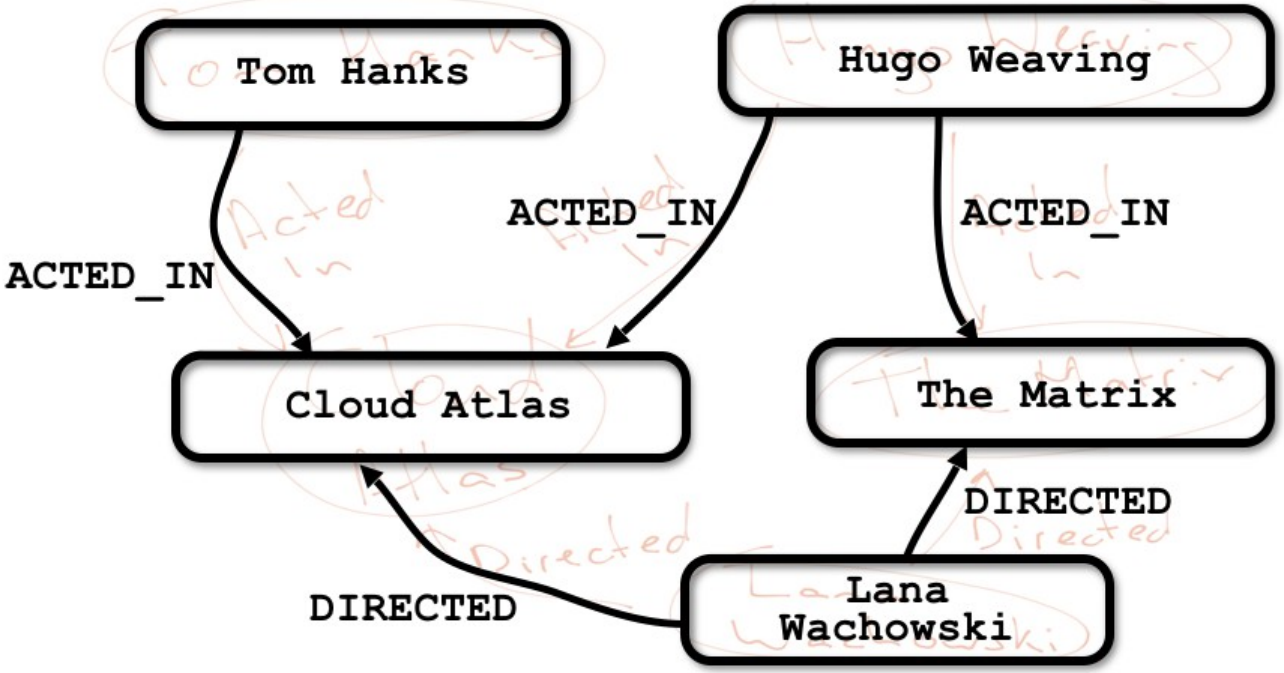
¿Qué busca esta consulta?

```
1  SELECT p2.personname, m1.movieName
2    FROM people p1
3         JOIN actors a1 ON (p1.personid = a1.personid)
4         JOIN movies m1 ON (a1.movieid = m1.movieid)
5         JOIN actors a2 ON (a2.movieid = m1.movieid)
6         JOIN people p2 ON (p2.personid = a2.personid)
7   WHERE p1.personname = 'Keanu Reeves';
8
```

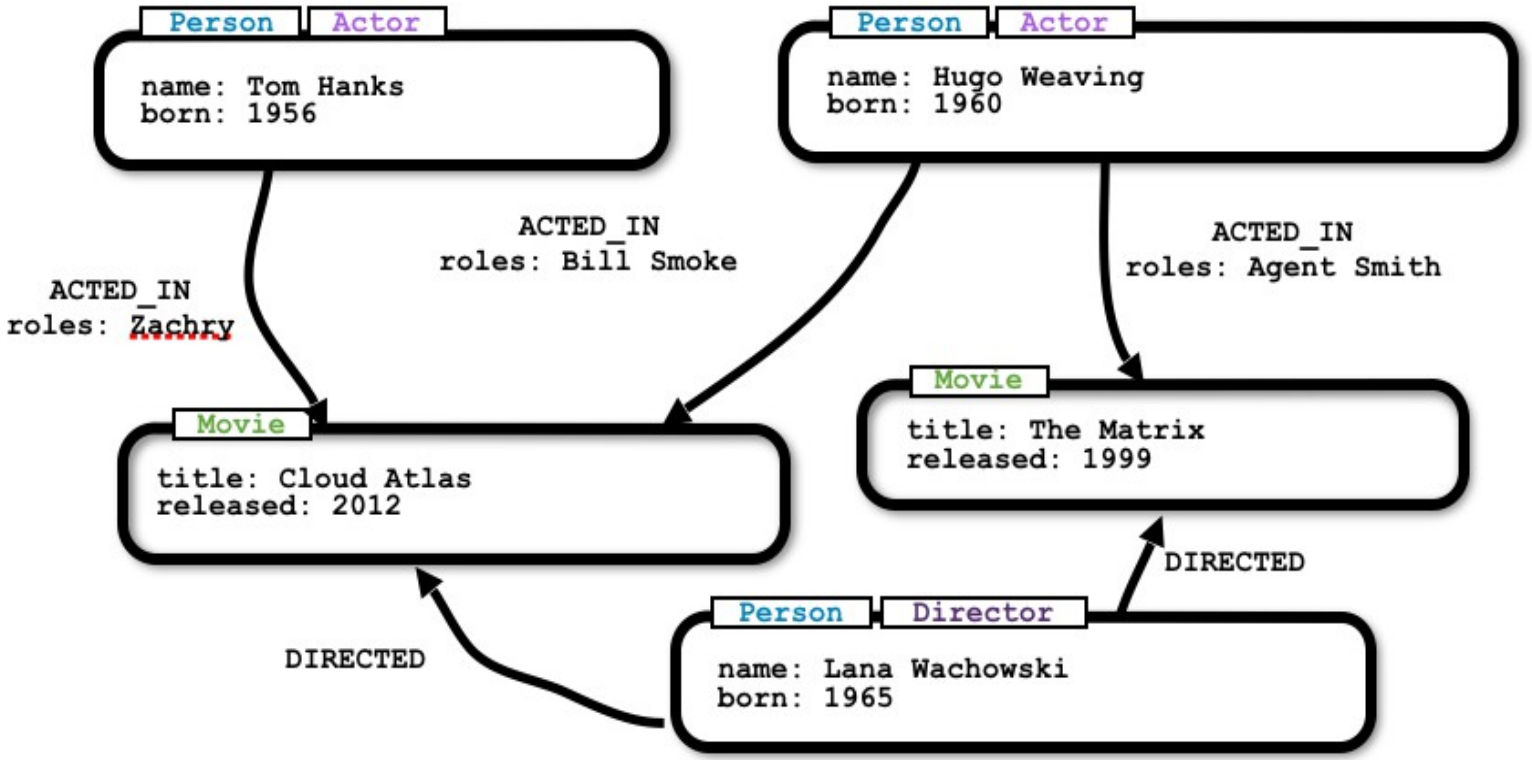
Modelo de Grafos es compatible con pizarrones



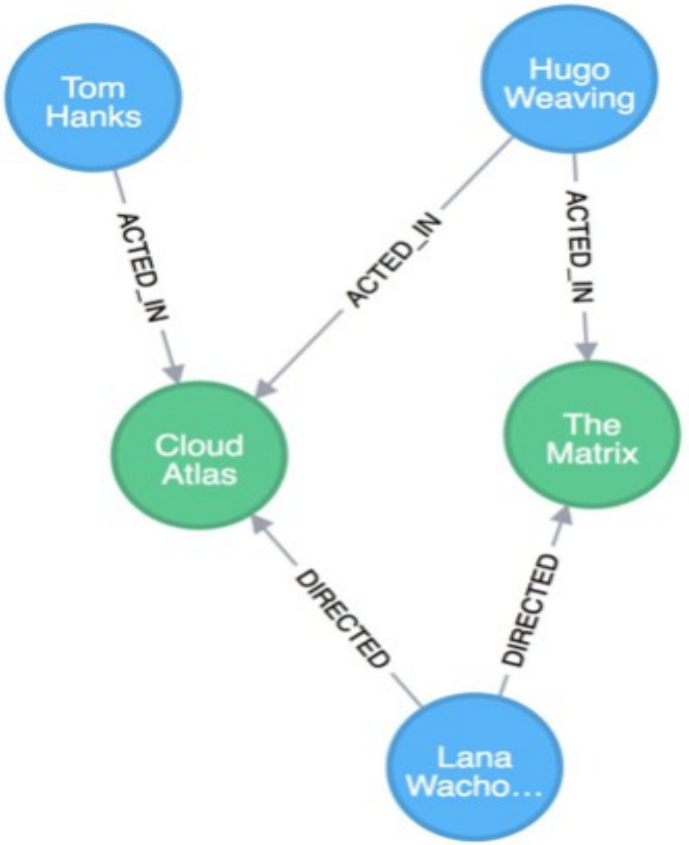
Modelo de Grafos es compatible con pizarrones (II)



Modelo de Grafos es compatible con pizarrones (III)



Modelo de Grafos es compatible con pizarrones (IV)



¿Cuáles fueron los pasos que seguimos?

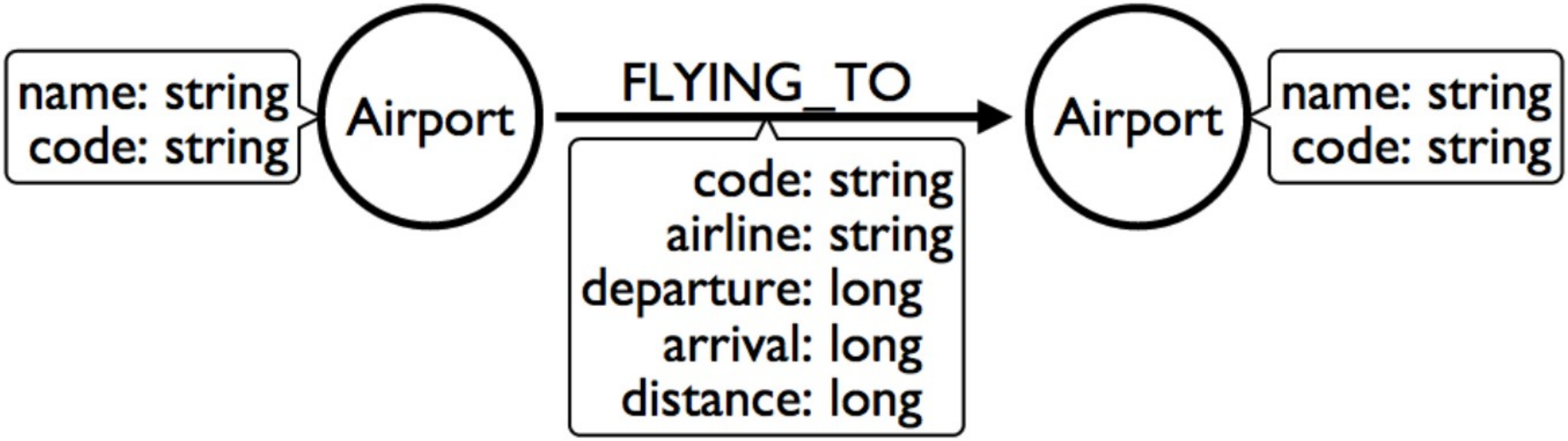
- Identificamos entidades y las representamos como nodos (sustantivos)
- A los nodos les asignamos etiquetas, de forma de agruparlos o categorizalos (actor, director)
- Identificamos las relaciones (acciones, verbos)
- Incorporamos propiedades

Modelando sin esquema

- Neo4j es *schema-free*
- Un modelo de datos puede ser mejor o peor
- La mejor opción suele estar dada por las consultas que necesitamos ejecutar sobre nuestros datos
- Veamos un ejemplo...

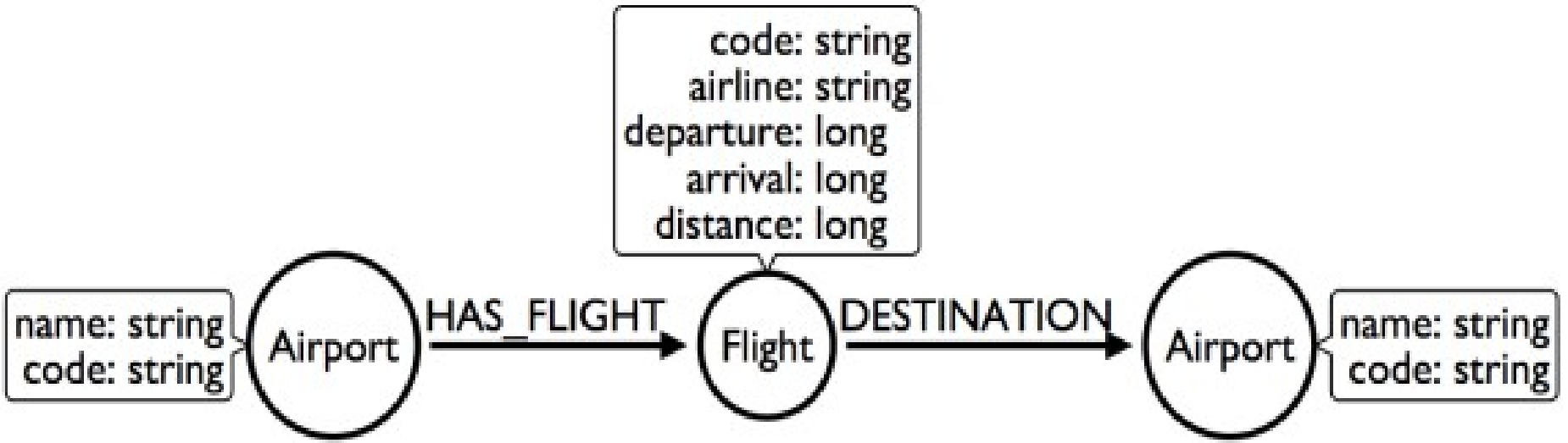
Ejemplo de modelado (I)

- Una aerolínea tiene un vuelo particular, en un día determinado hacia y desde una ubicación específica



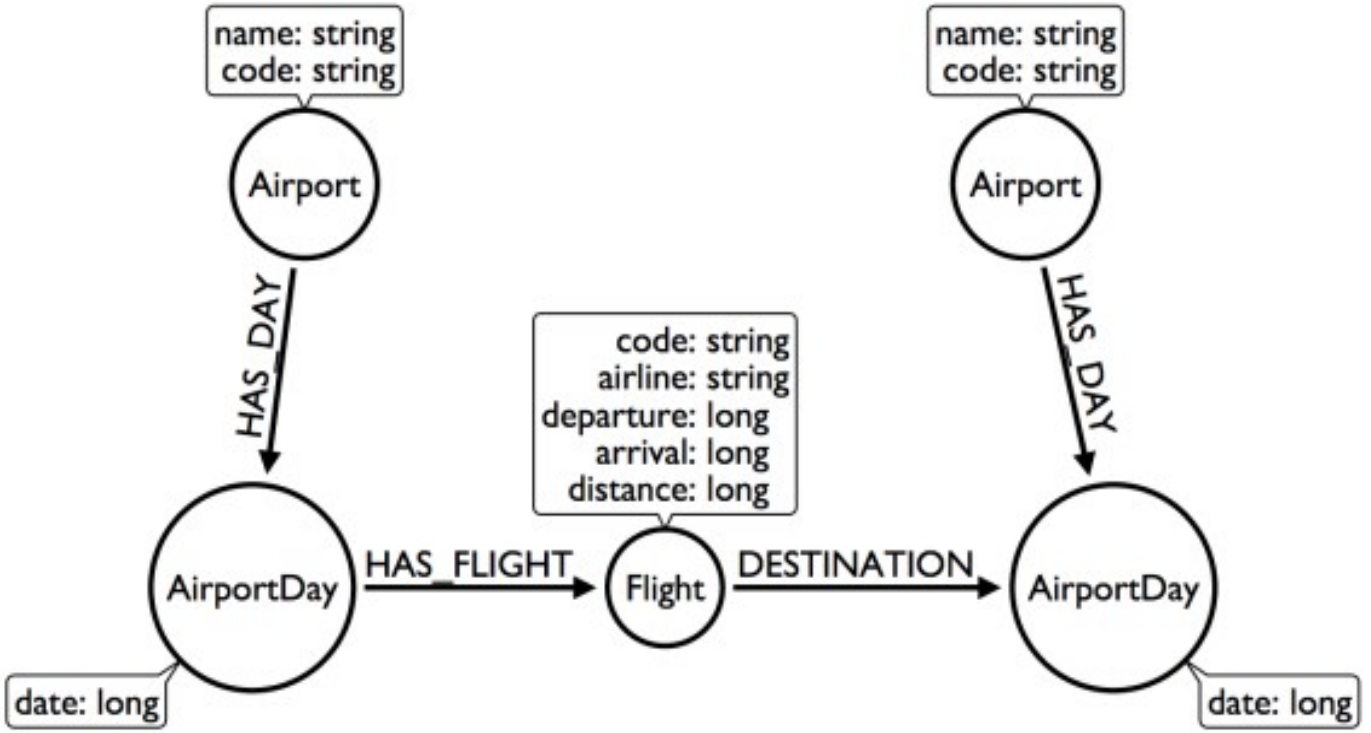
Ejemplo de modelado (II)

- Esta parece una mejor idea...



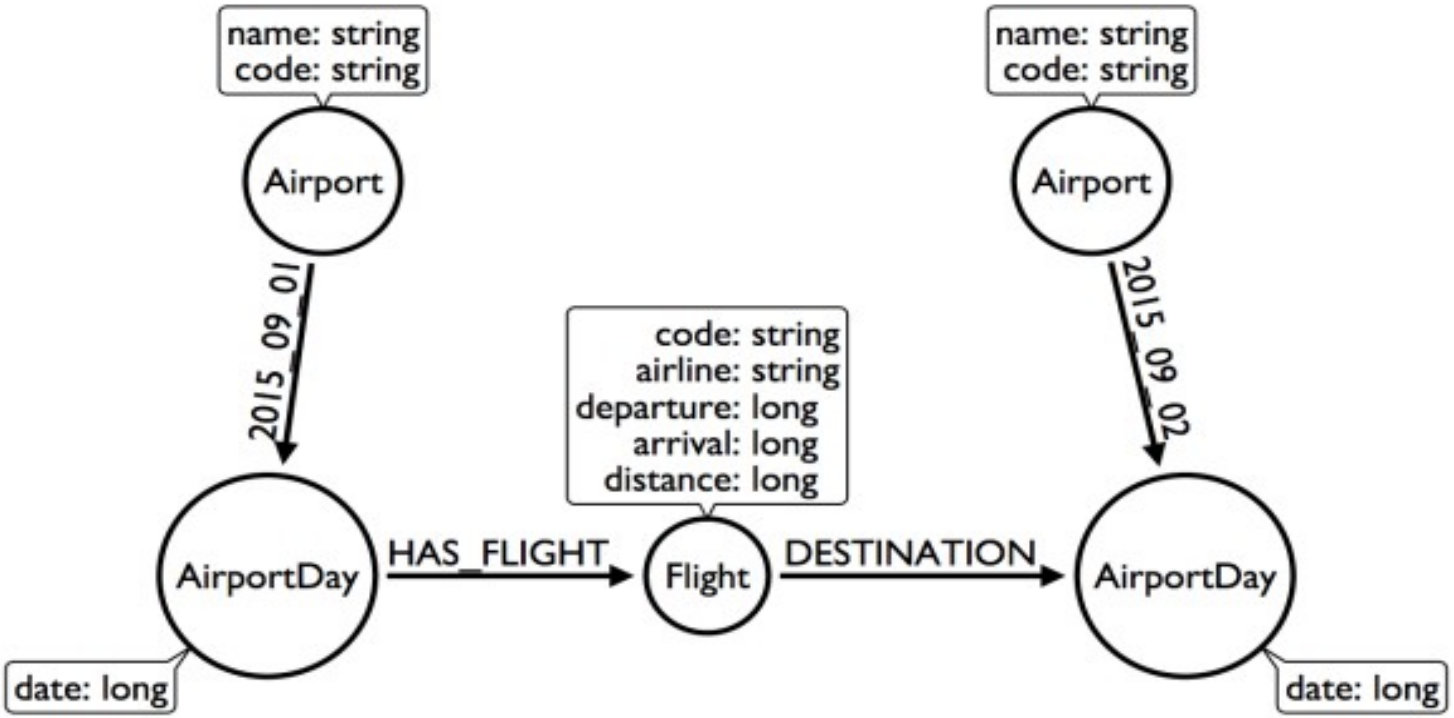
Ejemplo de modelado (III)

- Esta parece una mejor idea...



Ejemplo de modelado (IV)

- Esta parece una mejor idea...

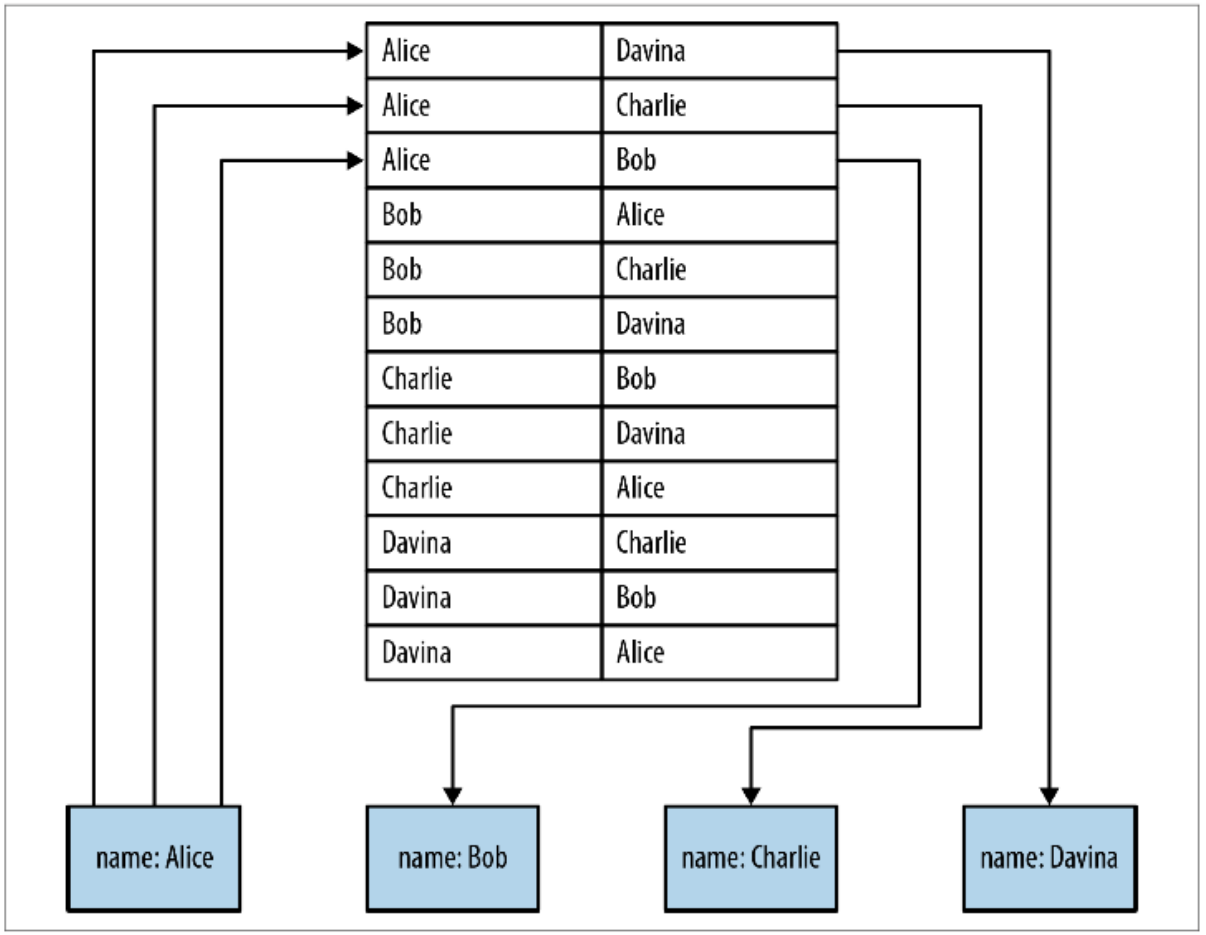


Para discutir:

Modelado relacional vs modelado en grafos

Estrategias de almacenamiento

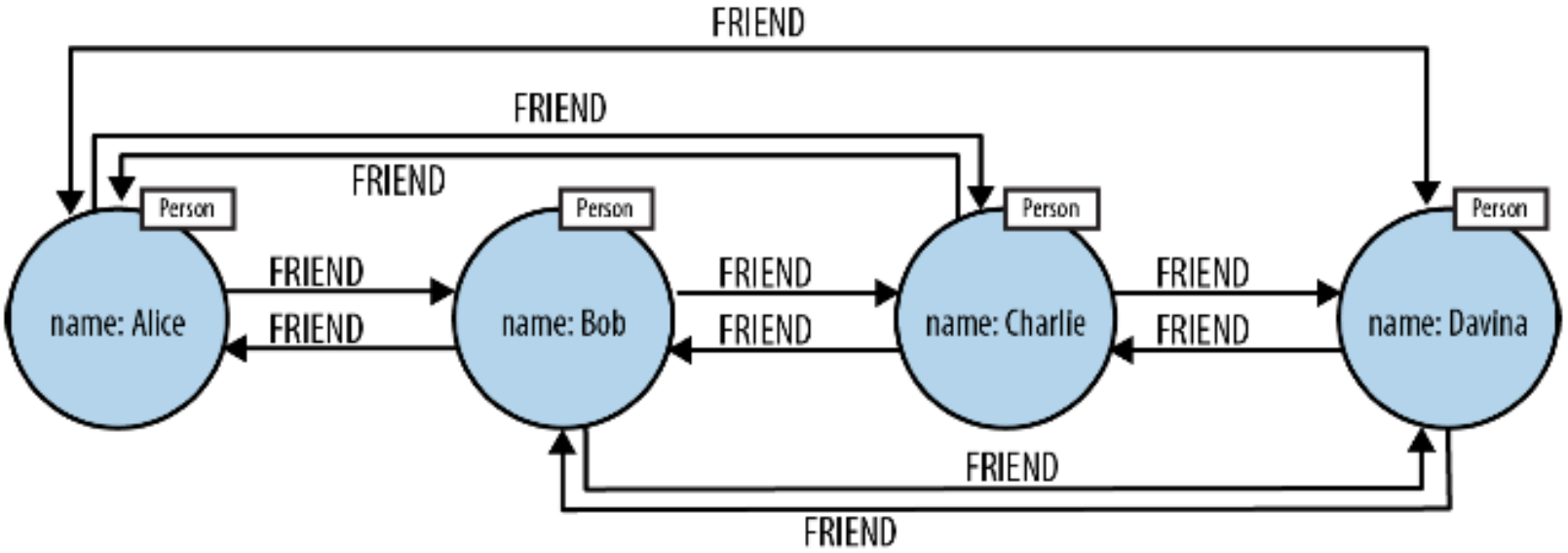
¿cómo almacenar grafos? (i)



Usando índices

- Búsqueda en índice O(log n) (depende de la implementación).
- Atravesar un camino de largo m tiene un costo O(m log n)
- Índices en una sola dirección

¿cómo almacenar grafos? (ii)



Index-free adjacency: acceder al adyacente en O(1)

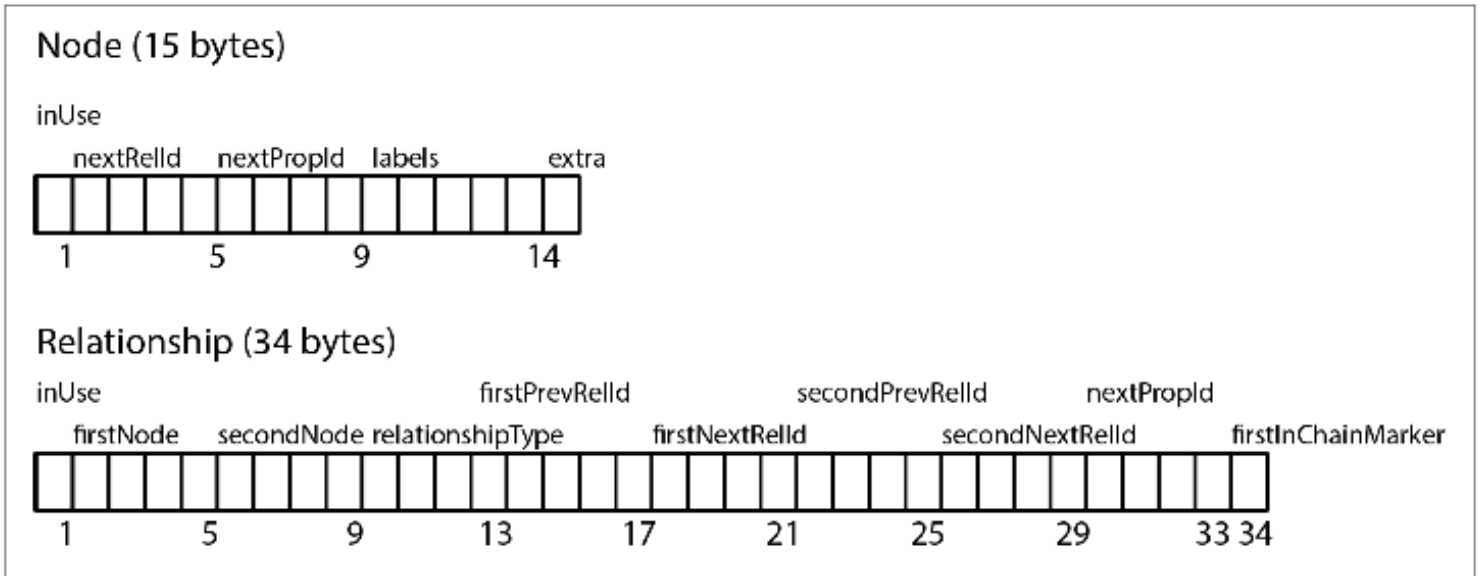
¿cómo se implementa?

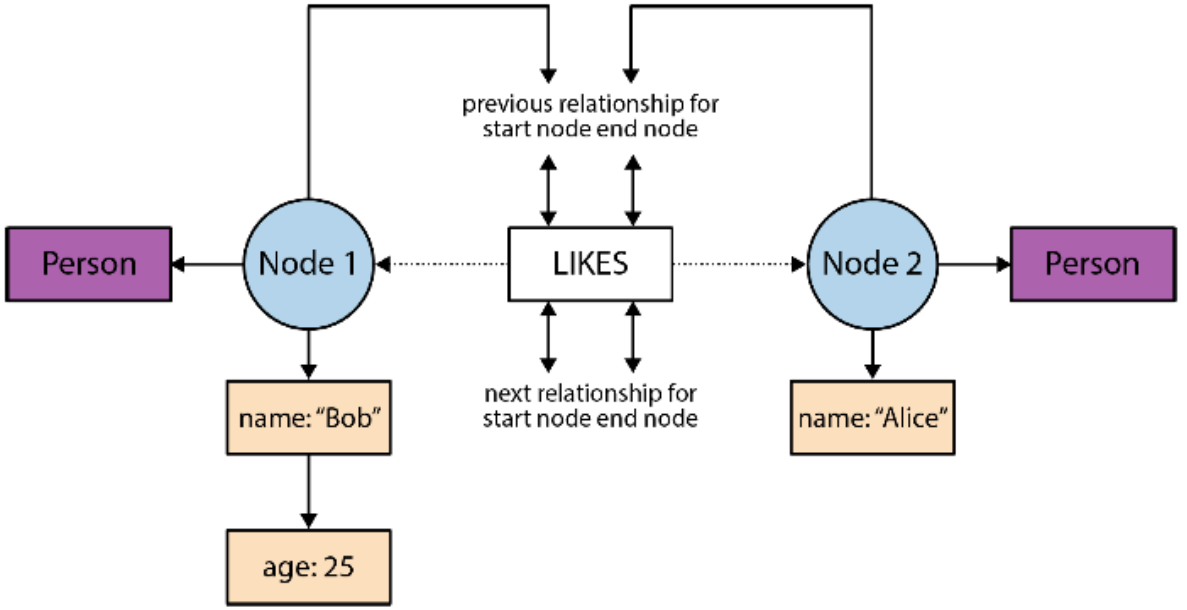
Almacenamiento en Neo4j

- Datos almacenados en diferentes *store files*, un por cada parte del grafo (nodos, relaciones, propiedades y etiquetas)
- Registros de largo fijo:
  - Permiten computar el offset fácilmente
- Punteros entre *store files*.
- Las listas de relaciones son doblemente enlazadas.

¡Multiestructuras!

Ejemplo: nodos y relaciones





Operaciones sobre grafos

Operaciones básicas

Dado un nodo G:

- AddNode (G, x): agrega el nodo x al grafo G
- DeleteNodo (G, x): borra el nodo x del grafo G
- Adjacent(G,x,y): chequea si existe una arista de x a y
- AdjacentEdges(G,x,y): etiquetas de las aristas de x a y
- Add(G,x,y,l): agrega una arista entre x e y con etiqueta l
- Delete (G,x,y,l): elimina una arista entre x e y con etiqueta l

Operaciones básicas

Dado un nodo G:

- Reach (G,x,y): chequea si existe un camino de x a y
- Path (G,x,y): un camino de x a y (el más corto)
- 2-hop(G,x): conjunto de nodos y, tal que existe un camino de largo 2 entre y e y, o de y a x
- n-hop(G,x): conjunto de nodos y, tal que existe un camino de largo n entre y e y, o de y a x

Cypher

- Lenguaje desarrollado por Neo4j, tanto para la creación de bases de datos cono para su manipulación (DML, DDL)
  - Lenguaje declarativo, inspirado en SQL
  -
- Los tipos de datos se dividen en dos tipos
  - Básicos:
    - Booleanos, integer, Float, String, List (lista ordenada), Map (clave, valor)
  - De estructura:
    - Node: almacena un nodo del grafo con sus propiedades
    - Relationship: almacena una relación del grafo con sus propieades
    - Path: almacena una ruta del grafo

Cypher: Pattern-Matching

- Pattern-matching
  - Es el “corazón” de Cypher
  - Usando patterns, especificamos la forma de los datos que requerimos, luego Cypher resuelve como obtiene los mismos.

Cypher: Pattern-Matching

- Pattern-matching
  - Pattern para un nodo: (a)
    - Este pattern describe un nodo, y nombra el nodo usando la variable a
  - Patterns para nodos relacionados: (a) --> (b)
    - Describe dos nodos y una relación simple entre ellos. Los nodos se llaman a y b y la relación es dirigida, desde a hacia b
- Patterns para etiquetas: (a:Persona)-->(b)
  - Se especifica la etiqueta que debe tener el nodo a

Cypher: Pattern-Matching

- Pattern-matching
  - Propiedades: (a {name:'Seba'})
    - Los patterns asociados a propiedades se escriben entre llaves. En este caso, todos los nodos cuya propiedad name sea 'Seba'

  - Largo variable: (a)-[*2]->(b)
    - Describe un grafo de tres nodos y dos relaciones, en un solo camino (de largo 2). Es equivalente a (a)-->()-->(b)

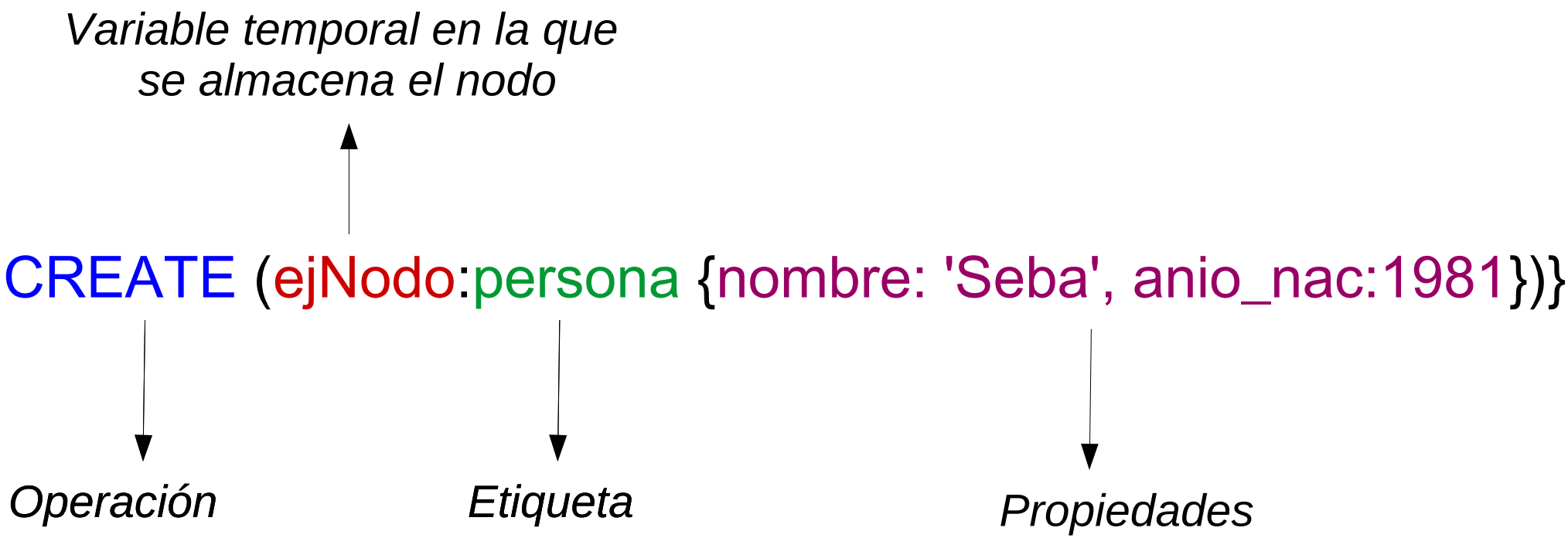
Cypher: Pattern-Matching

- Pattern-matching
  - Largo variable: (a)-[*3..5]->(b)

    Largo mínimo de 3 y máximo de 5. Describe un grafo de 4 nodos y 3 relaciones, 5 nodos y 4 relaciones o 6 nodos y 5 relaciones, todos conectados por un solo camino.

Cypher: creación de un nodo

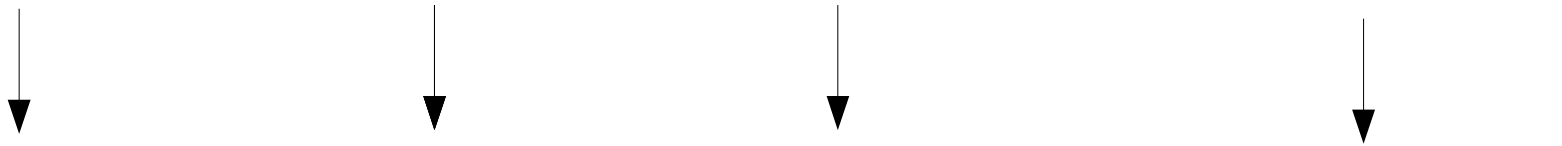
- La operación CREATE permite crear nuevos nodos
- Ejemplo de creación de un nodo con etiqueta y propiedades



Cypher: crear relaciones

- La operación CREATE también permite crear relaciones entre nodos. Las diferentes operaciones CREATE deben de ejecutarse juntas para poder acceder a las variables temporales.

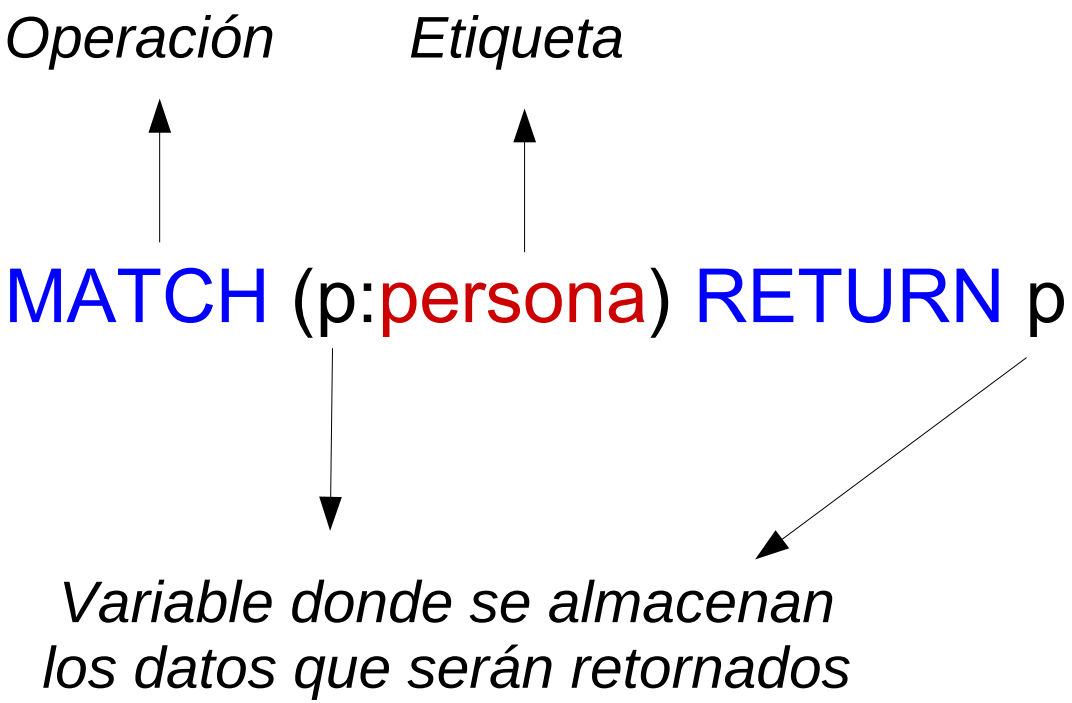
```
CREATE (SebaNodo:persona {nombre: 'Seba', anio_nac:1981})}

CREATE (CursoNodo:curso {nombre: 'MPGVD'})}

CREATE (SebaNodo)-[:DICTA {anio_desde:[2020]}]->(CursoNodo),
       CursoNodo)-[:DICTADO_POR {anio_desde:[2020]}]->(SebaNodo))
```



Nodo origen *Relación* *Propiedades de la relación* *Nodo destino*

Cypher: consultas con el operador MATCH

- El operador MATCH permite hacer consultas con condiciones sobre los datos de los nodos y sus relaciones
- La siguiente consulta retorna todos los nodos etiquetados como “persona”



Cypher: consultas con el operador MATCH

- ¡Cuidado! En las consultas debemos prestar atención a la aplicación de las etiquetas

```
MATCH (p) RETURN p
```

Cypher: consultas con el operador MATCH

- Es posible proyectar las propiedades que se quieren incluir en el resultado

```
MATCH (p:persona) RETURN p.nombre, p.anio_nac
```

Cypher: consultas con el operador MATCH

- Se dispone de una operación WHERE (dijimos que estaba inspirado en SQL)

```
MATCH (p:persona) WHERE p.nombre = 'Seba' RETURN p.nombre, p.anio_nac
```

Cypher: operador SET

- El operador SET se utiliza para añadir nuevas propiedades a un nodo
- Si la propiedad ya existe, se actualizará con el nuevo valor

```
MATCH (p:persona) WHERE p.nombre = 'Seba'
SET p.apellido='Garcia'
RETURN p
```

- También es posible eliminar propiedades

```
MATCH (p:persona) WHERE p.nombre = 'Seba'
SET p.apellido=NULL
RETURN p
```

Cypher: operador REMOVE

- El operador REMOVE puede usarse, al igual que el operador SET, para eliminar propiedades:

```
MATCH (p:persona) WHERE p.nombre = 'Seba'
REMOVE p.apellido
RETURN p
```

- También es posible eliminar etiquetas de un nodo

```
MATCH (p) WHERE p.nombre = 'Seba'
REMOVE p:persona
RETURN p
```

Cypher: operador DELETE

- El operador DELETE se utiliza para eliminar nodos

```
MATCH (p:persona) WHERE p.nombre = 'Seba' DELETE p
```

- ¡Cuidado! Para eliminar un nodo, primer deben de eliminarse sus relaciones. También se pueden eliminar con el operador DELETE.

```
MATCH (p1:persona)-[r:DICTA]
->(c:curso) WHERE p1.nombre='Seba' DELETE r
```

Cypher: operador DELETE

- En una sola operación pueden eliminarse nodos y aristas
- El operador DETACH elimina el nodo y todas sus relaciones de origen y destino

```
MATCH (p:persona) WHERE p.nombre = 'Seba' DETACH DELETE p
```

Cypher: constraints

- Cypher permite definir *constraints*
  - De unicidad: los valores de una propiedad o de un conjunto de ellas no pueden repetirse en nodos que compartan una etiqueta

```
CREATE CONTRAINT ON (p:persona) ASSERT p.ci IS UNIQUE
```

  - De existencia: obliga a que la propiedad exista en todos los nodos con la misma etiqueta

```
CREATE CONTRAINT ON (p:persona) ASSERT EXISTS (p.nombre)
```

Cypher: constraints

- Cypher permite definir *constraints*
  - De clave (*Node key*): obliga a que una propiedad o conjunto de estas existan y sean únicas. Concepto similar al de PK en el modelo relacional (Sólo disponible en versión Enterprise)

```
CREATE CONTRAINT ON (p:persona) ASSERT p.ci IS NODE KEY
```

Cypher: funciones de agregación

- Cypher dispone de funciones de agregación

```
MATCH (n:Person) RETURN avg(n.edad)
```

Cualquier valor null es excluído del cálculo

avg(null) retorna null

Cypher: funciones de agregación

- avg
- collect
- count
- max
- min
- percentileCont (interpolación lineal)
- percentileDisc (método de redondeo)
- stDev
- sum

Cypher: operaciones más complejas

- Buscar un camino más corto entre dos nodos, con un cantidad máxima de 15 relaciones.

```
MATCH (martin:Person { name: 'Martin Sheen' }),(oliver:Person { name: 'Oliver Stone' }),
p = shortestPath((martin)-[*..15]-(oliver))
RETURN p
```

GQL Standard

- Graph Query Language Standard
- ISO/IEC 39075 Information Technology, Database Languages, GQL
- Propuesta aceptada en 9/2019
- Standard propuesto para 8/2021
- Basado en standard SQL
- Fuerte participación de Neo4j y Oracle
  - Gramática de Cypher liberada como open source en 2015

GQL Standard

- Graph Query Language Standard
- ISO/IEC 39075 Information Technology, Database Languages, GQL
- Propuesta aceptada en 9/2019
- Standard propuesto para 8/2021
- Basado en standard SQL
- Fuerte participación de Neo4j y Oracle
  - Gramática de Cypher liberada como open source en 2015

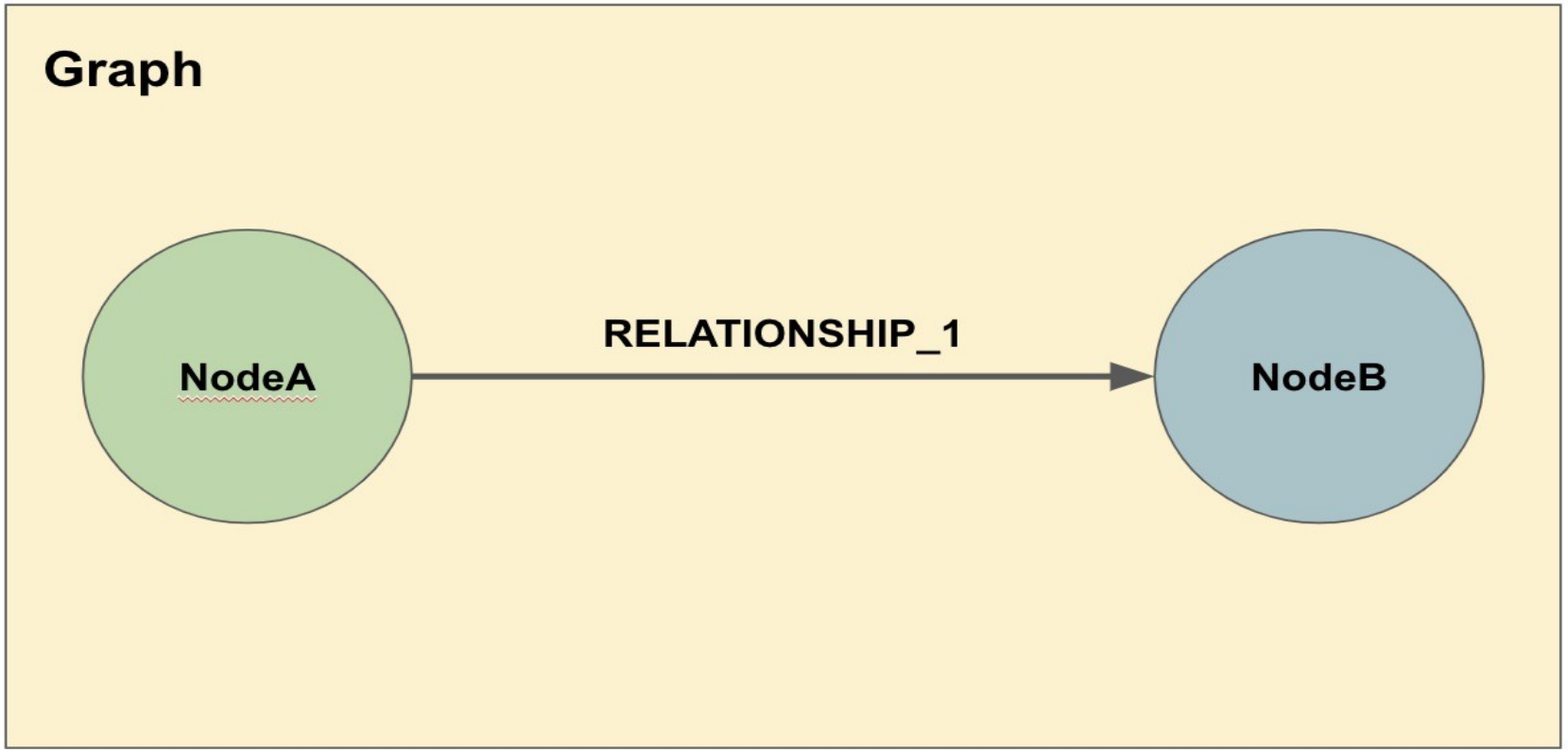
Sharding Graph Data

- Repaso:
  - Data Sharding: división horizontal de los datos, en particiones organizadas en distintos servidores
  - ¿Por qué usar sharding?
    - Razones legales o de privacidad (GDPR)
    - Minimizar latencia de consultas entre varias regiones
    - Segmentos relevantes de los grafos pueden almacenarse en servidores cercanos a los que ejecutan la consulta
    - El tamaño del grafo se hace demasiado grande (billones de nodos), por lo que debe de ser dividido en grafos más pequeños

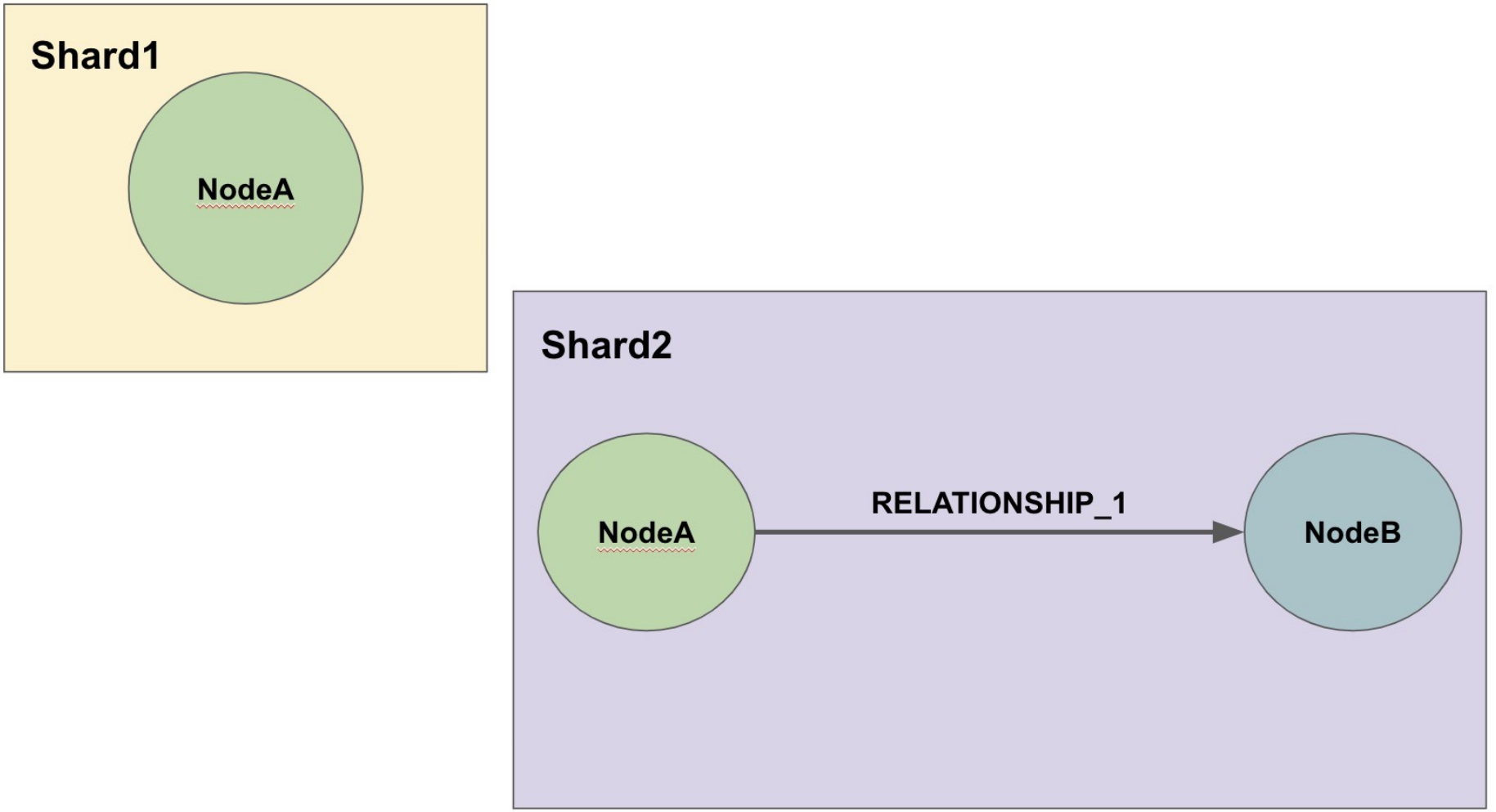
Sharding Graph Data

- La implementación del sharding depende fuertemente del nivel de conexión de los datos
- Mientras más conectados estén los datos, más complejo será hacer el sharding
- Se debe utilizar redundancia de datos para generar puentes entre shards
  - No es posible tener relaciones entre un nodos de diferentes shards
- Neo4j Fabric (disponible desde versión 4.0) es la solución de sharding que permite dividir grandes grafos en instancias más pequeñas para luego almacenarlas en bases de datos independientes

Sharding Graph Data



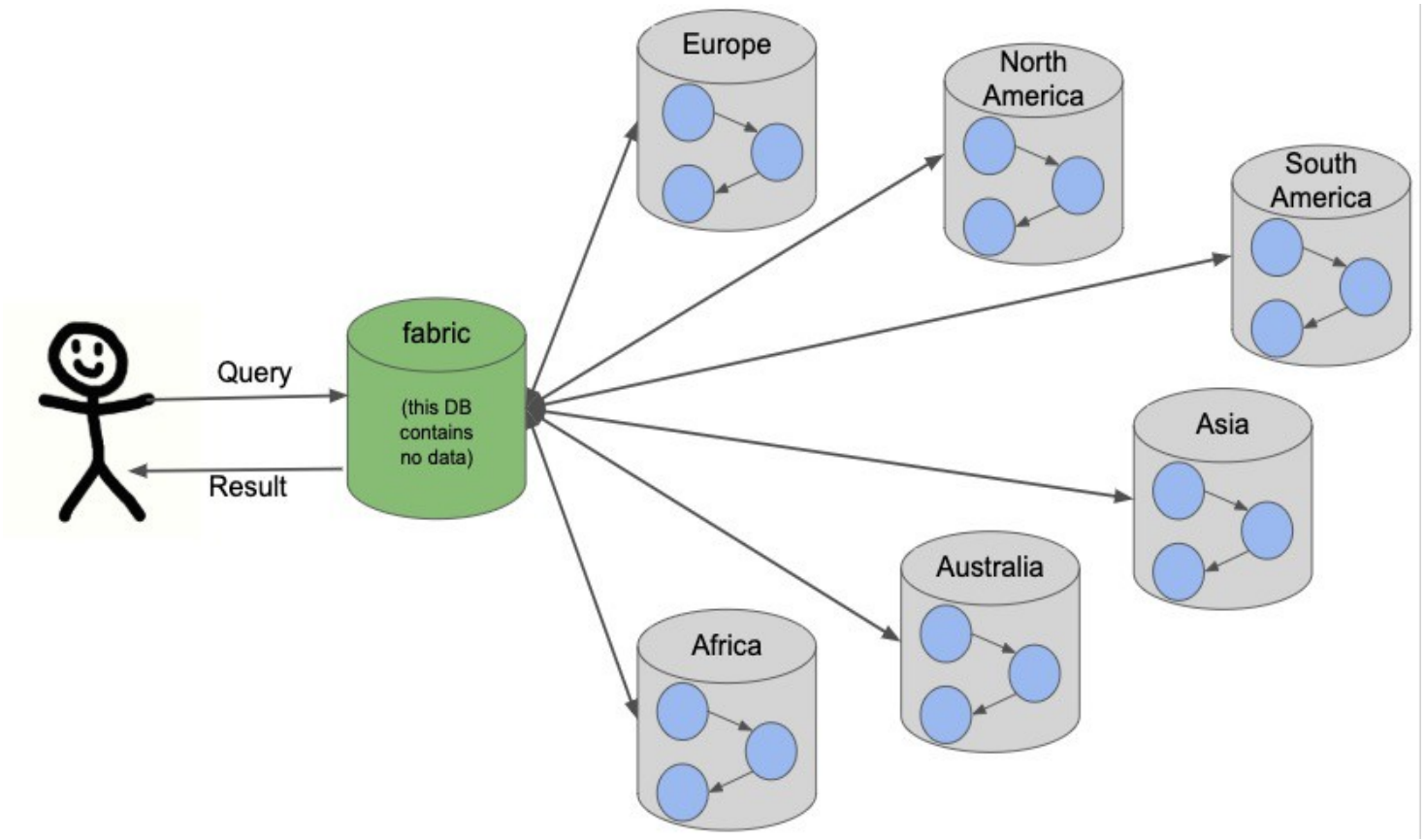
Sharding Graph Data



Sharding Graph Data

- Se decide dejar todos los nodos de NodeA en un shard y mantener las relaciones entre los NodeA y NodeB en otro shard
- Cuando son ejecutadas las consultas, Fabric buscará los nodos NodeA que interesan y tomará las relaciones del otro shard, devolviendo los datos integrados.

Sharding Graph Data



Sharding Graph Data

Consultar un shard específico

```
USE fabric.shard1
MATCH (n) RETURN count(n)
```

Sharding Graph Data

```
CALL {
  // Query January 20202 for a flight
  USE fabric.january2020
  MATCH
    (flight:Flight {id: "2013-1-1--1545"})-
[:ORIGIN]->(o:Airport),
    (flight)-[:DESTINATION]->(d:Airport)
  RETURN flight, o, d
}
// No puede accederse a un nodo de otro shard.
Se deben de obtener los valores de las
propiedades y buscarlas en el otro shard

WITH flight, o.code AS originCode, d.code AS
destinationCode

CALL {
  // Take variables from previous
  WITH originCode, destinationCode
  // Find the nodes in the airports shard
  USE fabric.airports
  MATCH (origin:Airport {code: originCode})
  MATCH (destination:Airport {code:
destinationCode})
  RETURN origin, destination
}
RETURN flight, origin, destination
```

Material adicional

- Graph Databases, Ian Robinson et al, O'Reilly 2015. http://graphdatabases.com/?ref=blog
- Documentación, cursos y videos de Neo4J https://neo4j.com/developer/
- Tom Heath and Christian Bizer (2011) Linked Data: Evolving the Web into a Global Data Space. http://linkeddatabook.com/editions/1.0/
- W3C, Best Practice Recipes for Publishing RDF Vocabularies, http://www.w3.org/TR/swbp-vocab-pub/