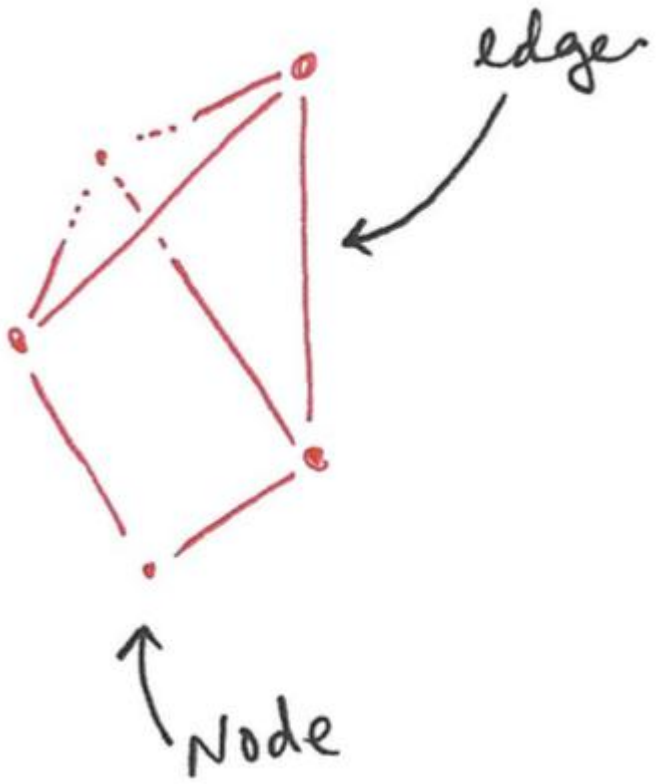
Consultas en bases de datos de grafos



Bases de Datos No Relacionales
Instituto de Computación, FING, UdelaR

¿qué es una consulta
sobre un grafo?

¿qué tipo de operaciones se hacen
sobre grafos?

Operaciones sobre grafos (i)

- **Adyacencia**: obtener los nodos adyacentes a cierto nodo.
- **Alcance**: Ej: determinar si existe un camino entre 2 nodos, y si existe cual es.
- ***Pattern matching*:** obtener sub grafos isomórficos a cierto patrón dado.
- **Agregación:** derivar de un grafo valores escalares agregados.

SPARQL para RDF
G-CORE y Cypher (Neo4j) para PGM
son lenguajes de consultas que soportan estos operadores

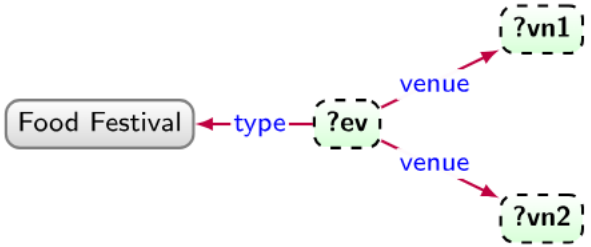
Patrones básicos sobre grafos

Son la base de todos los lenguajes de consulta sobre grafos.

Están compuestos de variables y constantes.

La evaluación del patrón consiste en mapear las variables en constantes en el grafo de datos, de forma que la imagen del patrón bajo el mapeo (sustituyendo las variables por las constantes asignadas) esté contenida dentro del grafo de datos.

Patrón



Grafo

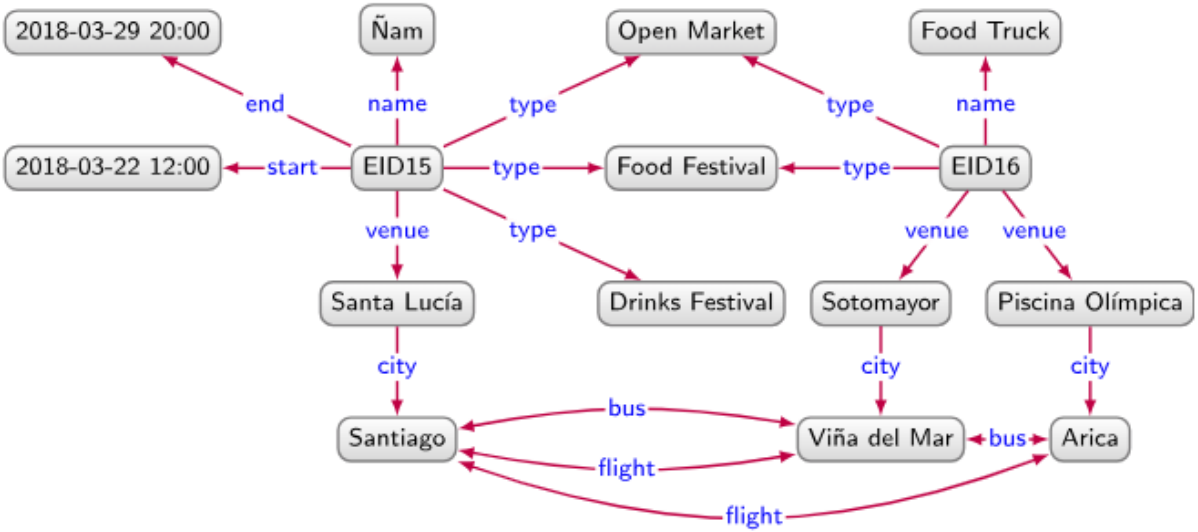


Figure 2.1: Directed edge-labelled graph describing events and their venues

Mapeos bajo semántica de homorfismo (más de una variable mapea a un término)

?ev	?vn1	?vn2
EID16	Piscina Olímpica	Sotomayor
EID16	Sotomayor	Piscina Olímpica
EID16	Piscina Olímpica	Piscina Olímpica
EID16	Sotomayor	Sotomayor
EID15	Santa Lucía	Santa Lucía

SPARQL

Mapeos bajo semántica de isomorfismo (cada variable se mapea a un término)

?ev	?vn1	?vn2
EID16	Piscina Olímpica	Sotomayor
EID16	Sotomayor	Piscina Olímpica

Cypher

Patrones complejos sobre grafos

Pueden verse como la combinación mediante operadores del algebra relacional de los resultados de la aplicación de patrones simples.

operadores unarios

- proyección (π) para obtener un subconjunto de columnas,
- selección (σ) para obtener un subconjunto de filas que cumplan una **condición** determinada
- cambio de nombre de columnas (ρ).

operadores binarios

- unión (∪), diferencia (-) e intersección (∩)
- join (⋈) para ampliar las filas de una tabla con filas de la otra tabla que satisfagan una condición de unión.
- antijoin (▷, también conocida como resta) para obtener filas de la primera tabla para las que no hay filas joineables en la segunda tabla
- join a la izquierda (⟕, también conocida como opcional) para realizar un join pero manteniendo filas de la primera tabla sin una fila compatible en la segunda tabla, etc.

Grafo

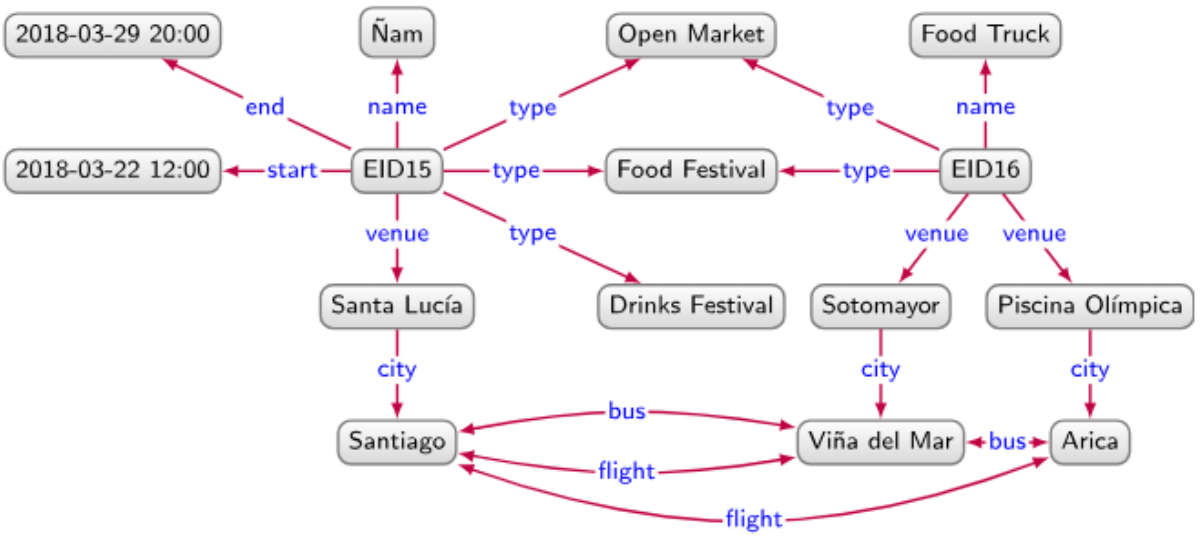
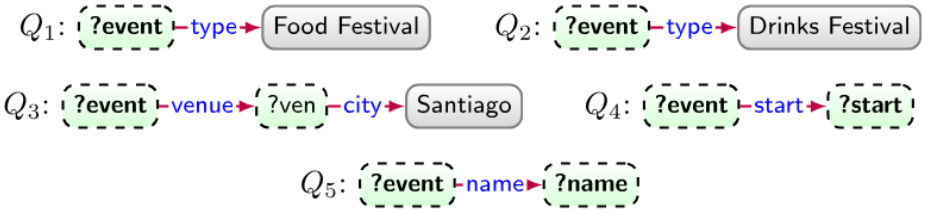


Figure 2.1: Directed edge-labelled graph describing events and their venues

Patrón



$$Q(G) = \begin{array}{|l|l|l|} \hline \textbf{?event} & \textbf{?start} & \textbf{?name} \\ \hline \texttt{EID16} & & \texttt{Food Truck} \\ \hline \end{array}$$

$$Q := ((((Q_1 \cup Q_2) \triangleright Q_3) \bowtie Q_4) \bowtie Q_5),$$

Consultas en Cypher y SPARQL

Consultas en Cypher y SPARQL

Cypher y SPARQL como DQL (Data Query Language)

Cypher	SPARQL
MATCH (patrones)	SELECT (variables)
RETURN (variables)	WHERE (patrones)

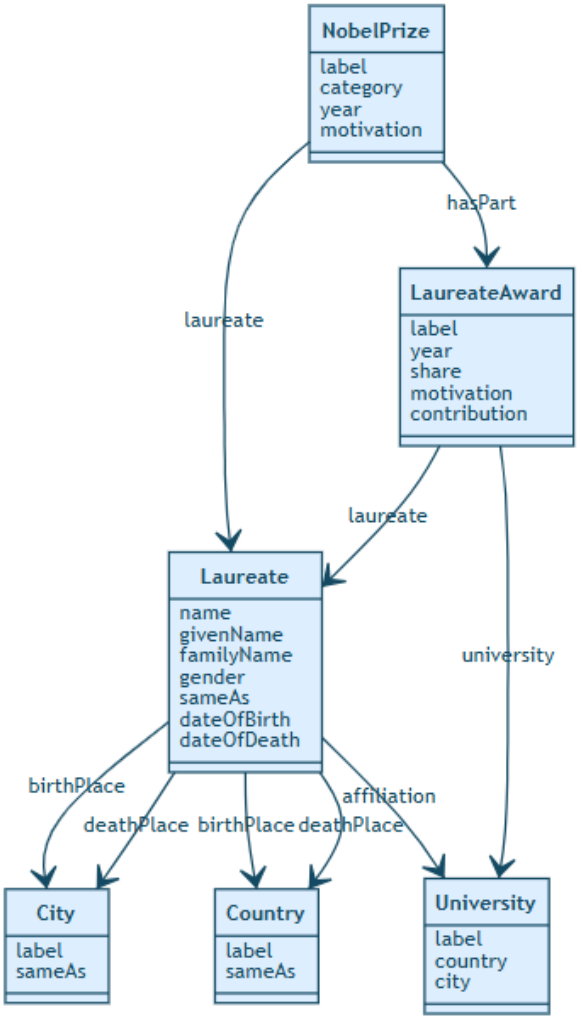
Ejemplos en SPARQL: premios Nobel

Tenemos que tener en cuenta el esquema para poder armar nuestras consultas.

La especificación completa del dataset está disponible en: https://data.nobelprize.org/specification/

Vamos a escribir consultas SPARQL y ejecutarlas directamente sobre este endpoint:

https://data.nobelprize.org/sparql



Ejemplos en SPARQL: selección

1) Premios recibidos por Marie Curie

Prefijos utilizados

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX nobel: <http://data.nobelprize.org/terms/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
```

Versión 1

```
SELECT ?person ?prizeName
WHERE {
        ?person rdf:type nobel:Laureate .
        ?person foaf:familyName "Curie“.
        ?person foaf:givenName "Marie“.
        ?person nobel:nobelPrize ?prize.
        ?prize rdfs:label ?prizeName.
}
```

Versión 2

```
SELECT ?person ?prizeName
WHERE {
        ?person rdf:type nobel:Laureate  ;
                foaf:familyName "Curie“ ;
                foaf:givenName "Marie“ ;
                nobel:nobelPrize ?prize .
        ?prize rdfs:label ?prizeName.
}
```

Todos los patrones en amarillo aplican al mismo sujeto

Ejemplos en SPARQL: selección

2) Personas y premios otorgados en 1911

Versión 1

```
SELECT DISTINCT ?name ?prizeName
WHERE {
        ?person rdf:type nobel:Laureate;
                rdfs:label ?name;
                nobel:nobelPrize ?prize.
        ?prize rdfs:label ?prizeName;
                nobel:year 1911.
}
```

Versión 2

```
SELECT DISTINCT ?name ?prizeName
WHERE {
        ?person rdf:type nobel:Laureate;
                rdfs:label ?name;
                nobel:nobelPrize ?prize.
        ?prize rdfs:label ?prizeName;
                nobel:year ?year.
FILTER (?year = 1911)
}
```

Fijo la constante o impongo una condición de FILTER sobre el resultado.
Son análogas, la primera puede ser más eficiente.

Ejemplos en SPARQL: selección

3) Mujeres que recibieron premios en física

```
SELECT DISTINCT ?name ?prizeName
WHERE {
  ?person rdf:type nobel:Laureate;
            rdfs:label ?name;
            foaf:gender ?gender;
            nobel:nobelPrize ?prize.
  ?prize rdfs:label ?prizeName;
            nobel:category ?category.
FILTER (lang(?prizeName) = 'en' && ?gender = 'female' && ?category = nobel:Physics )
}
```

Puedo componer condiciones de filtrado usando operadores booleanos AND (&&) OR (||)

Ejemplos en SPARQL: agregación

4) Cantidad de premios recibidos por cada universidad, ordenados en forma descendente

```
SELECT DISTINCT ?univName (COUNT(distinct ?la) AS ?count)
WHERE {
        ?person rdf:type nobel:Laureate;
                rdfs:label ?name;
                nobel:nobelPrize ?prize;
                nobel:laureateAward ?la.
  OPTIONAL{
    ?la nobel:university ?univ.
    ?univ rdfs:label ?univName
  }
  FILTER (lang(?univName) = 'en')
} GROUP BY ?univName
ORDER BY DESC(?count)
```

Devolvemos el tamaño de cada grupo (en cantidad de premiados)

Algunas entradas no tienen datos sobre la universidad

Agrupamos por universidad

Ejemplos en SPARQL: caminos

4) Cantidad de premios recibidos por cada universidad, ordenados en forma descendente

```
SELECT DISTINCT ?univName (COUNT(distinct ?la) AS ?count)
WHERE {
        ?person rdf:type nobel:Laureate;
                rdfs:label ?name;
                nobel:nobelPrize ?prize;
                nobel:laureateAward ?la.
  OPTIONAL{
    ?la nobel:university ?univ.
    ?univ rdfs:label ?univName
  }
  FILTER (lang(?univName) = 'en')
} GROUP BY ?univName
ORDER BY DESC(?count)
```

Devolvemos el tamaño de cada grupo (en cantidad de premiados)

Algunas entradas no tienen datos sobre la universidad

Agrupamos por universidad

Aún más

- Puedo usar funciones que implementan algoritmos sobre los grafos
- Ej: el camino más corto
- *Número de Bacon sobre el grafo de películas*

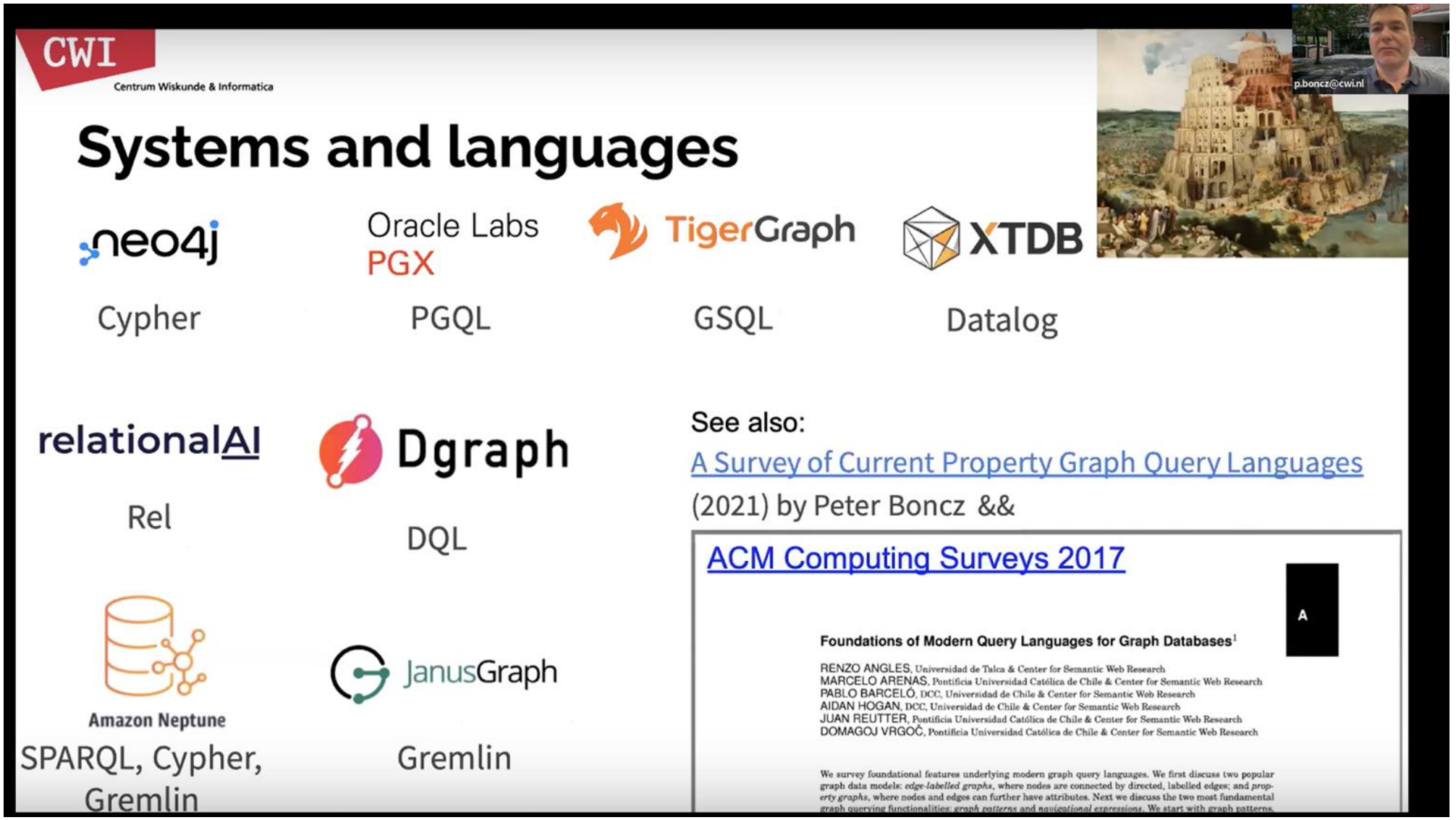
```
MATCH p=shortestPath((kevin:Person)-[r:ACTED_IN*]-(actor))

WHERE kevin.name='Kevin Bacon' AND actor.name='Al Pacino'

RETURN p, length(p)
```

- The Neo4j Graph Data Science Library

Pero aún no hay un standard



EDBT 2022, Keynote: Peter Boncz - The (sorry) State of Graph Database Systems
https://www.youtube.com/watch?v=aDoorU4X6Jk&t=423s

Recursos sobre RDF y SPARQL

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

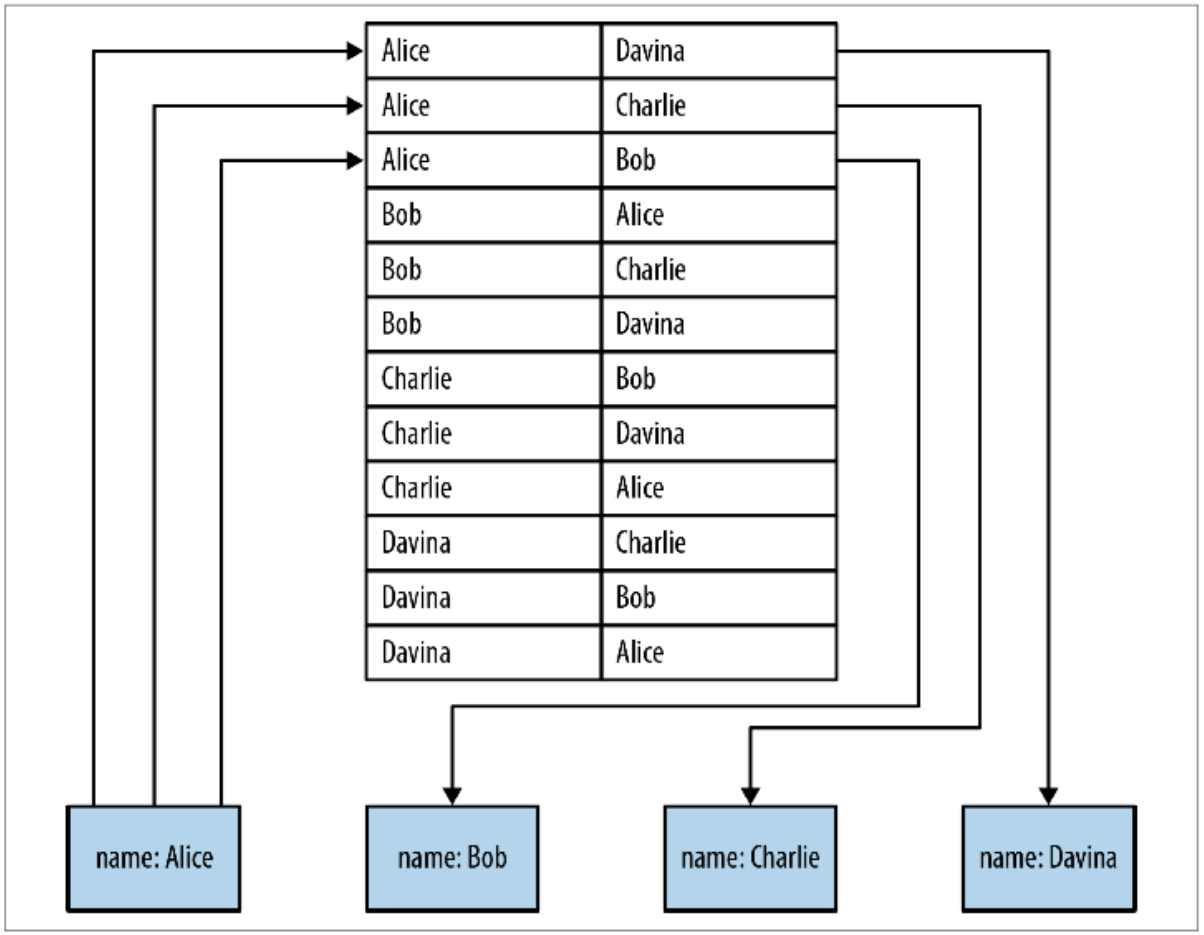
- Curso *Linked Data Engineering* del Hasso-Plattner Institut https://open.hpi.de/courses/semanticweb2016

Bases de datos de grafos: modelo físico

Product	Link	Database engine
WhiteDB	http://whitedb.org	Tuple store
GraphDB	https://www.ontotext.com/products/graphdb	Tuple store
OrientDB	https://www.orientdb.org	Document store
ArangoDB	https://www.arangodb.com	Document store
Azure Cosmos DB	https://azure.microsoft.com/es-es/services/cosmos-db	Document store
FaunaDB	https://fauna.com	Document store
RedisGraph	https://oss.redislabs.com/redisgraph	Key-value store
Dgraph	https://dgraph.io	Key-value store
HyperGraphDB	http://www.hypergraphdb.org	Key-value store
MS Graph Engine	https://www.graphengine.io	Key-value store
Titan	https://titan.thinkaurelius.com	Wide-column store
JanusGraph	https://janusgraph.org	Wide-column store
DSE Graph	https://www.datastax.com/products/datastax-graph	Wide-column store
InfiniteGraph	https://www.objectivity.com/products/infinitegraph	Object-oriented store
ThingSpan	https://www.objectivity.com/products/thingspan	Object-oriented store
VelocityDB	https://velocitydb.com	Object-oriented store
Oracle Spatial and Graph	https://www.oracle.com/technetwork/database-options/spatialandgraph/overview/spatialandgraph-1707409.html	RDBMS
Sparksee/DEX	http://www.sparsity-technologies.com	Native graph database
TigerGraph	https://www.tigergraph	Native graph database
GraphBase	https://graphbase	Native graph database
Memgraph	https://memgraph.co	Native graph database
Neo4j	https://neo4j.com	Native graph database

Fuente: Timón-Reina, S., Rincón, M., & Martínez-Tomás, R. (2021). An overview of graph databases and their applications in the biomedical domain. Database : the journal of biological databases and curation, 2021, https://doi.org/10.1093/database/baab026

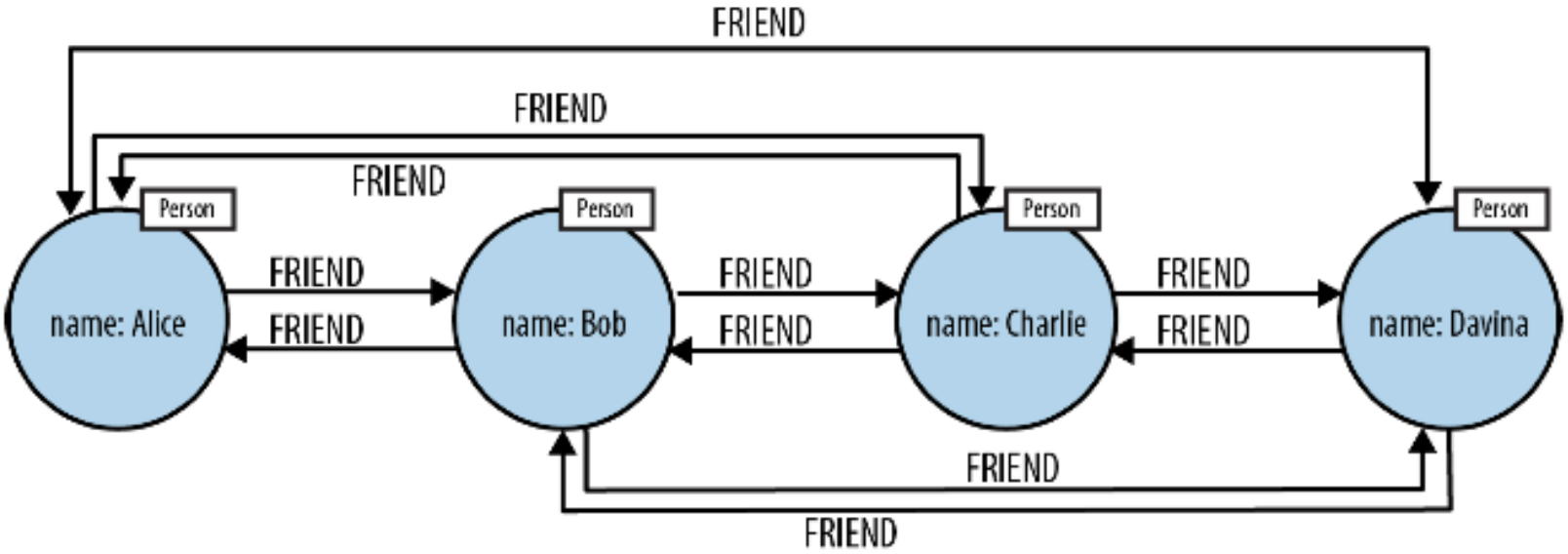
Implementados sobre un RDBMS



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

Index-free adjacency



Objetivo: acceder al adyacente en O(1)

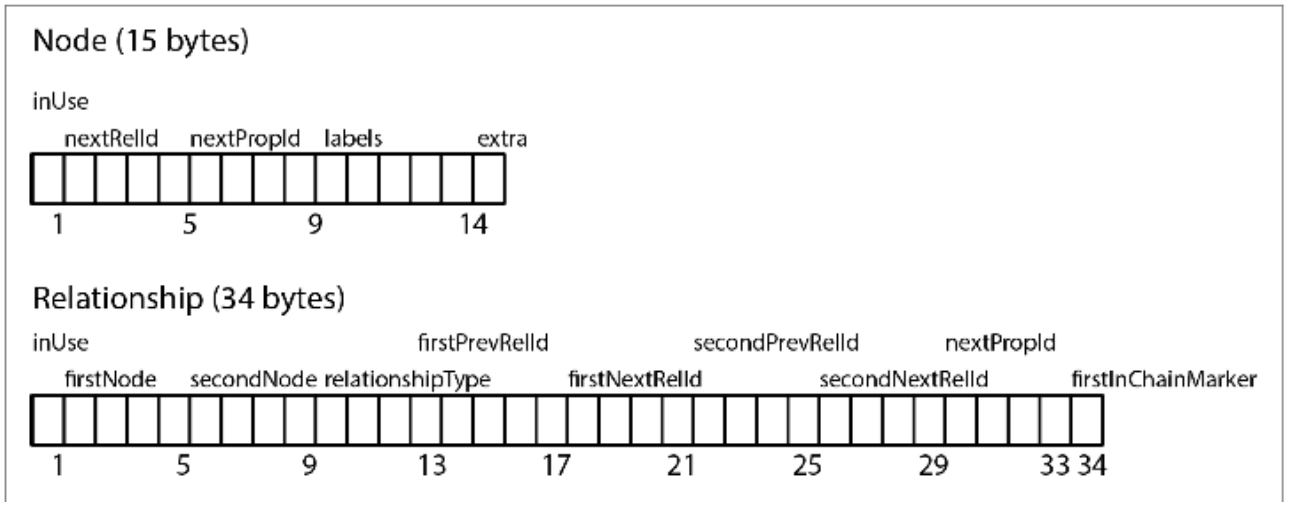
¿cómo se implementa?

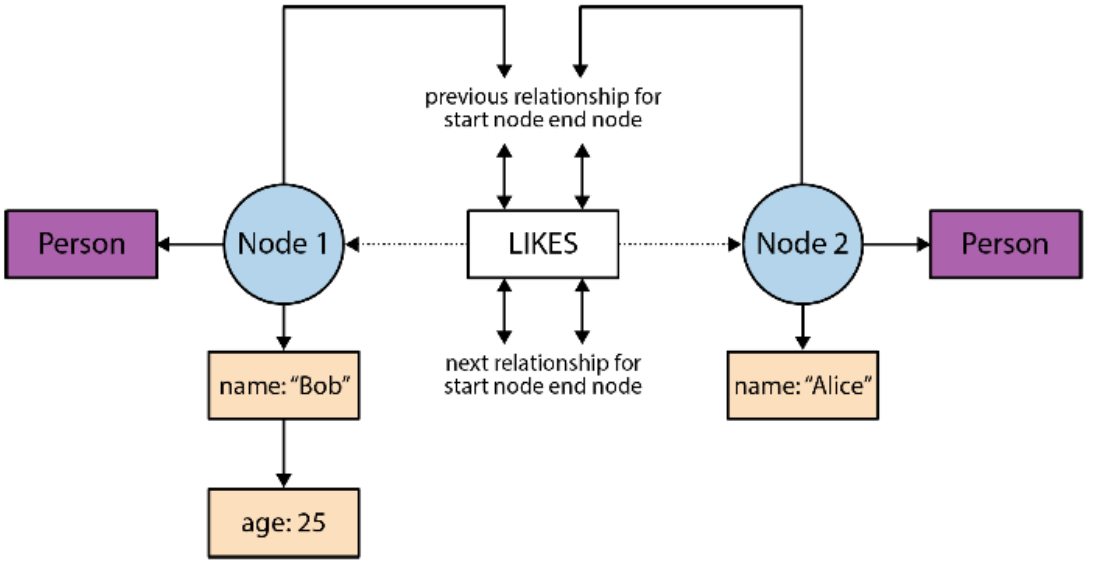
Almacenamiento en Neo4j

- 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





Fuente: Robinson, Ian, Jim Webber, and Emil Eifrem. Graph databases: new opportunities for connected data. "

O'Reilly Media, Inc.", 2015. Chapter 6.

Material adicional sobre Neo4J

- Graph Databases, Ian Robinson et al, O'Reilly 2015.
  - http://graphdatabases.com/?ref=blog
- Documentación, cursos y videos de Neo4J
  - https://neo4j.com/developer/