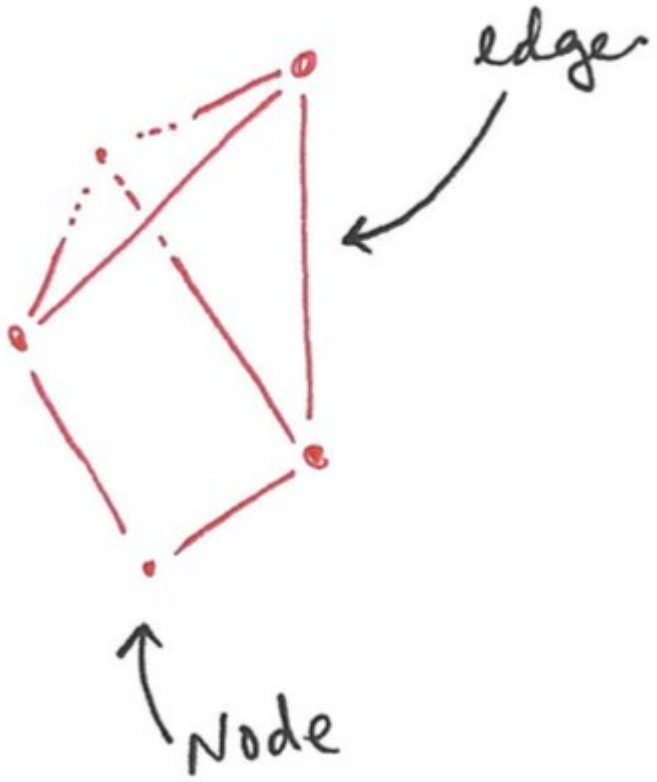
Consultas en bases de datos de grafos



Bases de Datos No Relacionales

Instituto de Computación, FING, UdelaR – 2022

CC-BY Lorena Etcheverry lorenae@fing.edu.uy

¿qué es una consulta sobre un grafo?

¿qué tipo de operaciones se hacen sobre grafos?

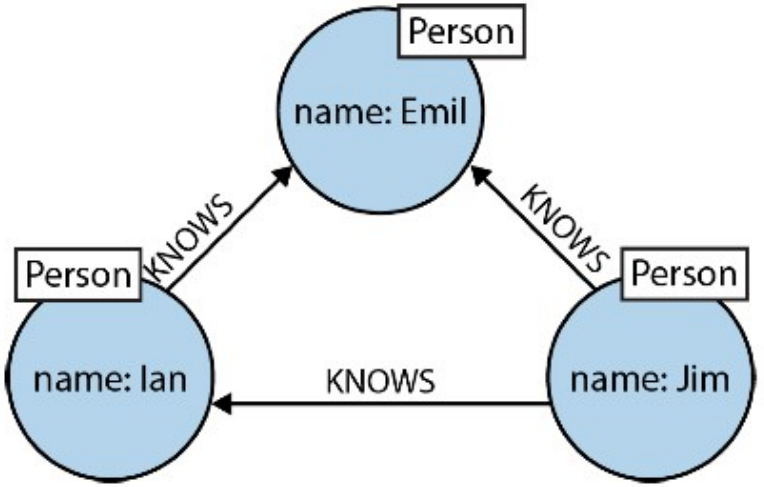
Operaciones sobre grafos (i)

- **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

Gremlin (Apache Tinkerpop) y Cypher (Neo4j) para PGM son lenguajes de consultas que soportan estos operadores

Cypher, algunas ideas básicas

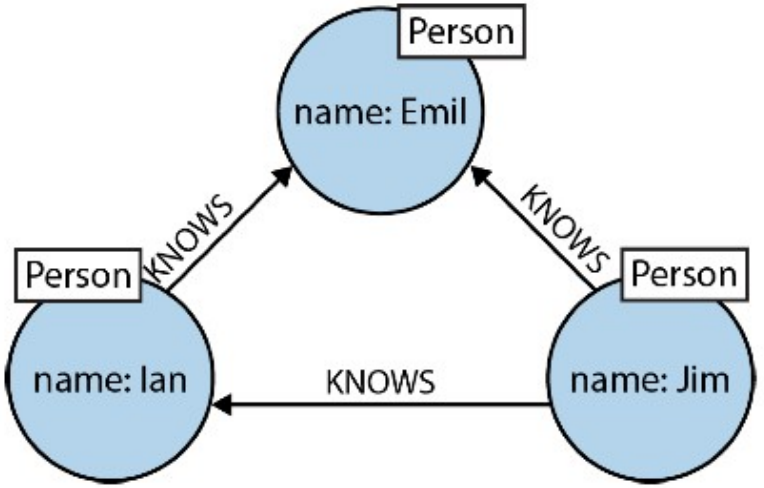


Cypher como Data Definition Language

```
(emil:Person {name:'Emil'})
    <-[:KNOWS]-(jim:Person {name:'Jim'})
    -[:KNOWS]->(ian:Person {name:'Ian'})
    -[:KNOWS]->(emil)
```

variables

Cypher como *data query language*




```
MATCH (a:Person {name:'Jim'})-[:KNOWS]->(b)-[:KNOWS]->(c),
      (a)-[:KNOWS]->(c)
RETURN b, c
```

```
MATCH (a:Person)-[:KNOWS]->(b)-[:KNOWS]->(c),
      (a)-[:KNOWS]->(c)
WHERE a.name = 'Jim'
RETURN b, c
```

Referencia del lenguaje Cypher

Más acerca de Cypher

Puedo especificar caminos de largo arbitrario, sobre ciertas propiedades, etc

```
MATCH (p:Person {name:'Al Pacino'})-[*1..4]-(p1:Person)
RETURN DISTINCT p1
```

¡Cuidado!

No devuelve un grafo,

devuelve una tabla

Aún más

- Puedo usar funciones que implementan algoritmos sobre los grafos

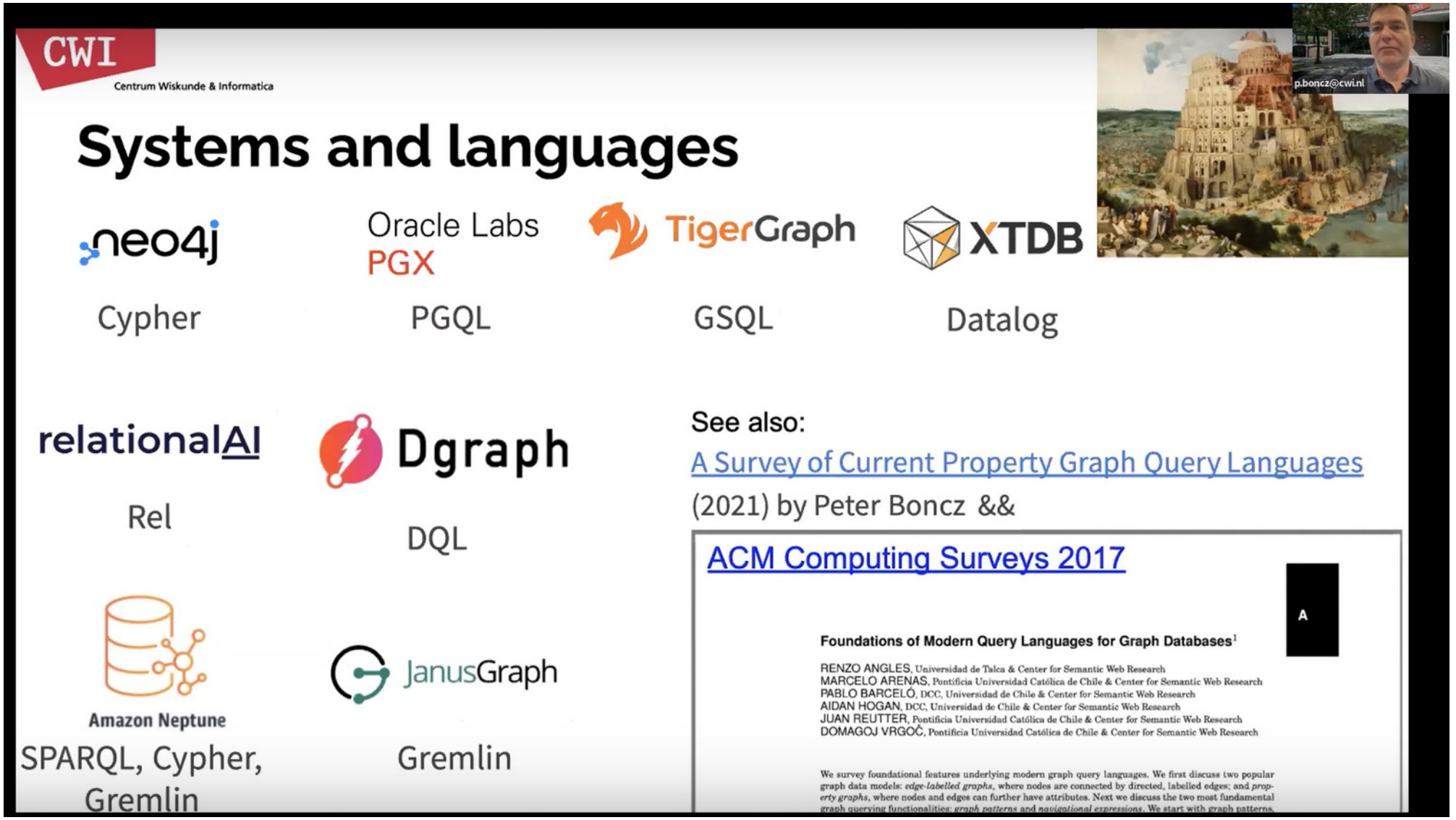
  Ej: el camino más corto

  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

Otros modelos de procesamiento de grafos

Las bases de datos de grafos resuelven consultas en forma eficiente, pero pueden no ser **eficientes** para procesar iterativamente grafos grandes.

Algoritmos usados para análisis de grafos como P*ageRank*, conteo de triángulos, búsqueda de componentes conexas, etc. requieren iterar sobre el grafo.

Surgen entornos de procesamiento de grafos, inspirados en Google Pregel[1]

Procesamiento distribuído.
Bajo intercambio entre procesos.
Tolerancia a fallas.
Paradigma “Think like a vertex”.

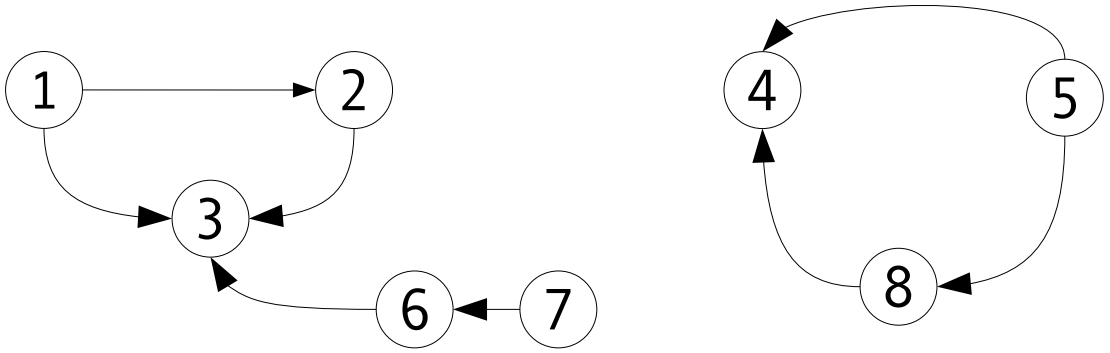
Apache Giraph es la implementación opensource de Pregel.

[1] *Pregel: a system for large-scale graph processing.*
G. Malewicz et al, SIGMOD 2010

Ejemplo: hallar componentes débilmente conexas[1]

Componente conexa: existe un camino entre cualquier par de nodos.

Componente débilmente conexa (WCC): se ignora la dirección de las aristas.



$V_{C1}=\{1,2,3,6,7\}$ $V_{C2}=\{4,5,8\}$

1 *Management and Analysis of Big Graph Data: Current Systems and Open Challenges,* M.Junghanns et al. Hanbook of Big Data Technologies, Springer 2017

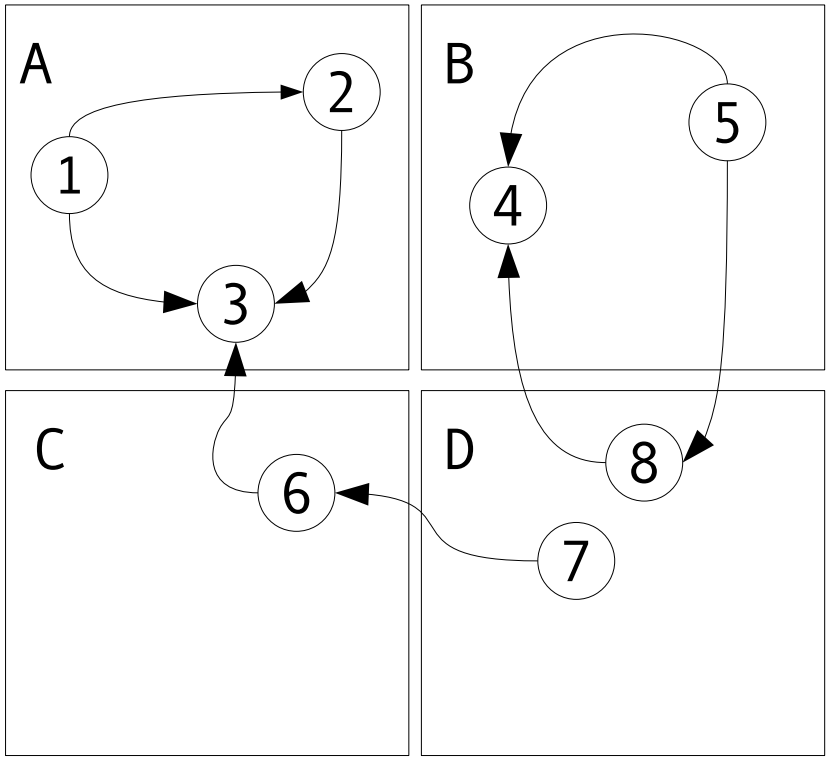
Procesamiento distribuído (i)

Master node (mn): coordina
Worker nodes (wn): realizan el cómputo.
El grafo se particiona entre los *wn*.

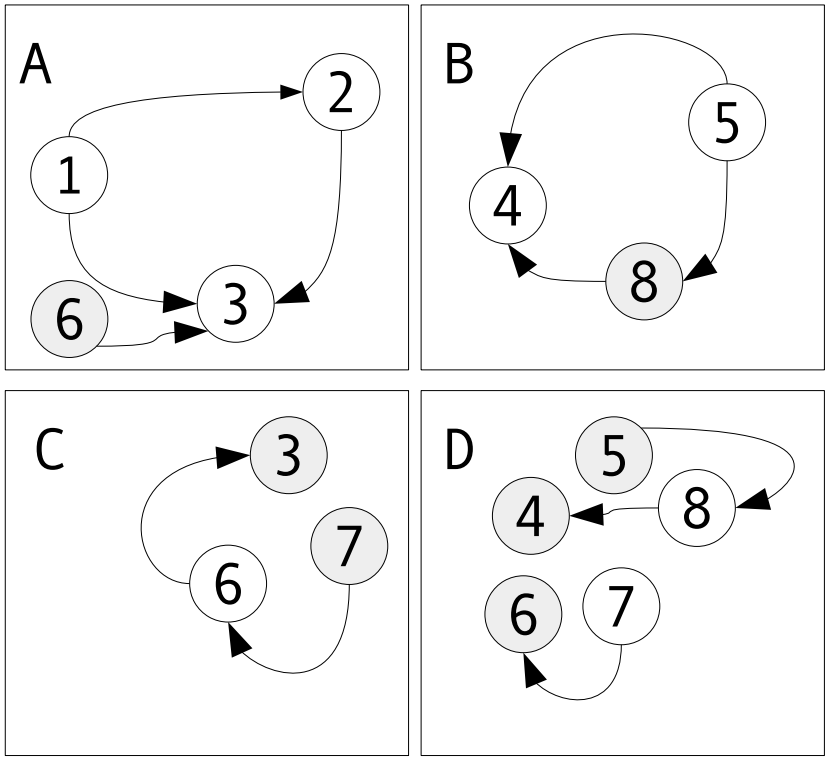
Cada *wn* conoce un conjunto de nodos y de cada nodo:

- el valor asociado al nodo
- las aristas salientes con sus valores
- los ids de nodos en el otro extremo de las aristas entrantes

Procesamiento distribuído (ii)



Modelo
"think like a vertex"



Modelo
"think like a graph"

Modelo de cómputo

Se basa en *vertex compute function (vcf)* que consiste en:

1) leer los mensajes entrantes

2) actualizar el valor del nodo

3) enviar mensajes a los nodos adyacentes

La invocación a la vcf se organiza en ***supersteps***.

En cada superstep cada *wn:*

1) llama a la *vcf* para cada nodo activo

2) marca como inactivo un nodo si se llamó a voteToHalt()

3) recoge los mensajes de salida

Cuando todos los wn terminan se mandan los mensajes.

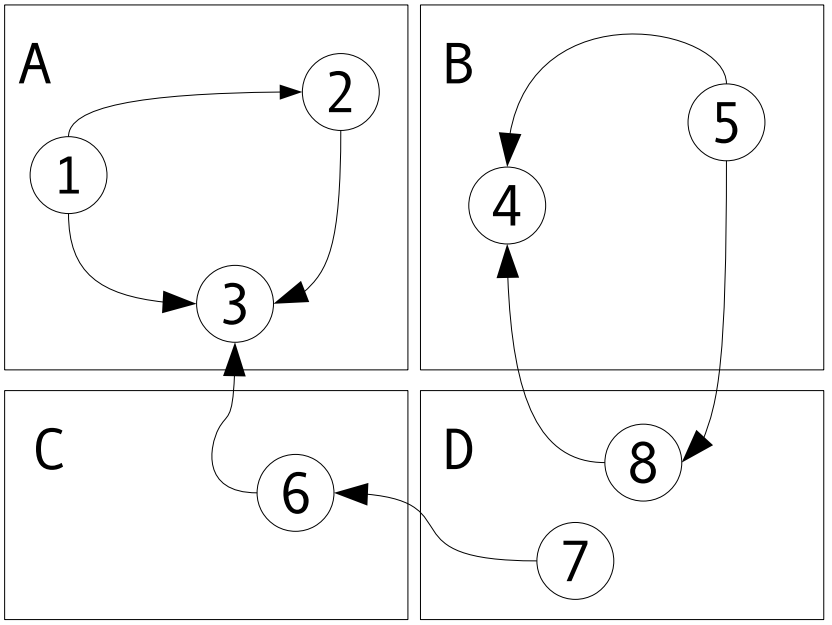
Los nodos que reciben mensajes pasan a activos.

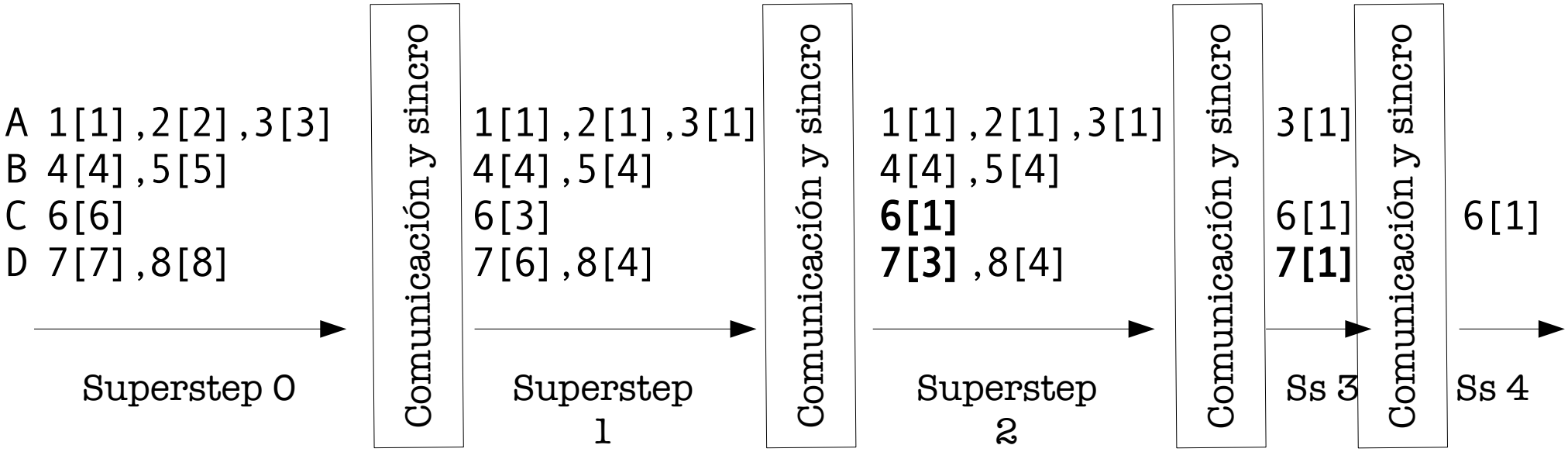
Modelo de cómputo (ii)

Hallar componentes débilmente conexas en Apache Giraph

```
void compute(Vertex v) {
  if (getSuperstep() == 0)
    v.setValue(v.getVertexID())
    sendMessageToAllEdges(v.getVertexValue())
  else
    minValue = min(v.getMessages())
    if (minValue < v.getVertexValue())
      v.setVertexValue(minValue)
      sendMessageToAllEdges(v.getVertexValue())
    v.voteToHalt();
}
```

Modelo de cómputo (iii)





Algunos desafíos

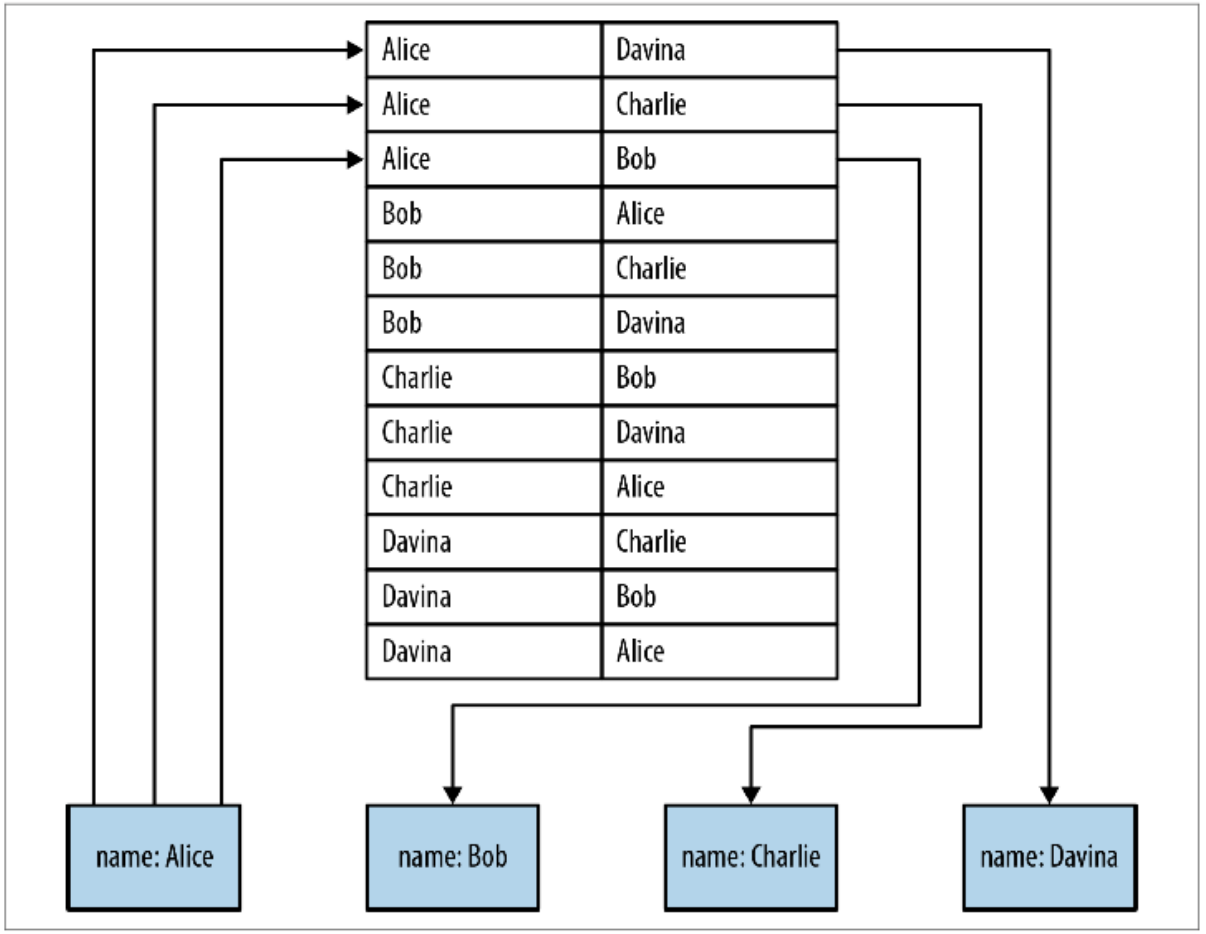
- Particionamiento de grafos
  - Es un problema NP-hard
  - Soluciones aproximadas y estáticas
  - ¿qué pasa cuando la cantidad de nodos cambia por partición? Balance de carga
- Manejo y análisis de grafos dinámicos
- Visualización
- Construcción de benchmarks

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, baab026. https://doi.org/10.1093/database/baab026

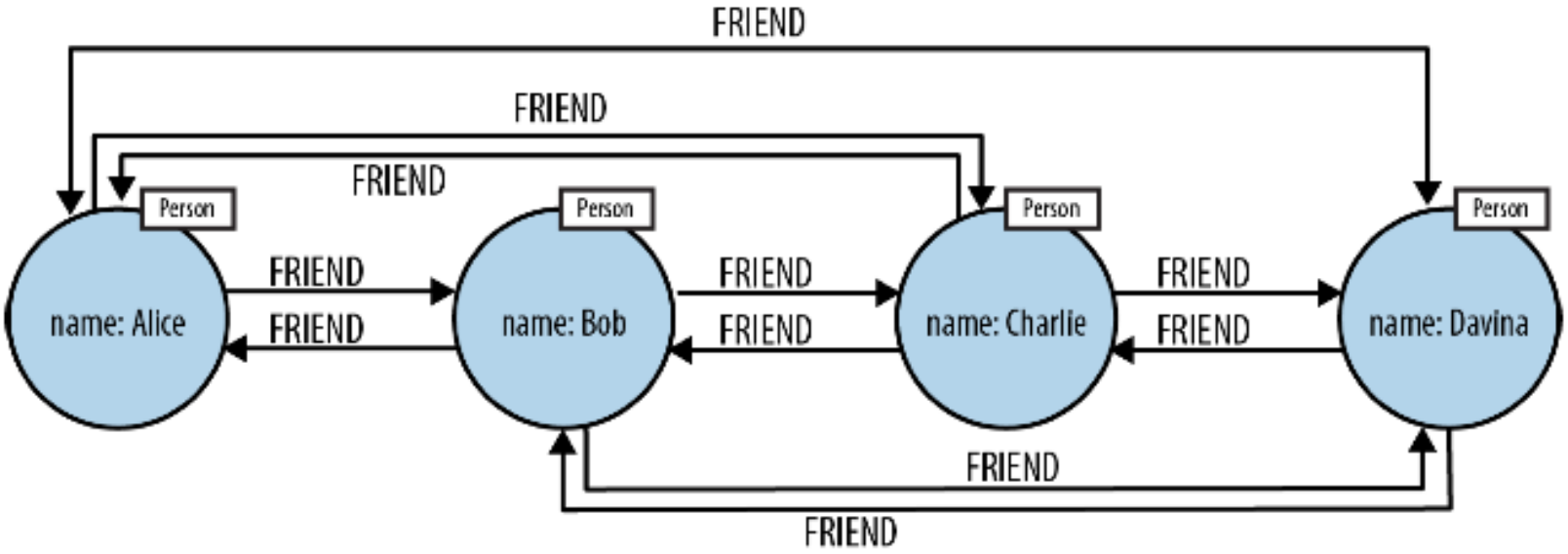
Sobre un RDBMS



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)

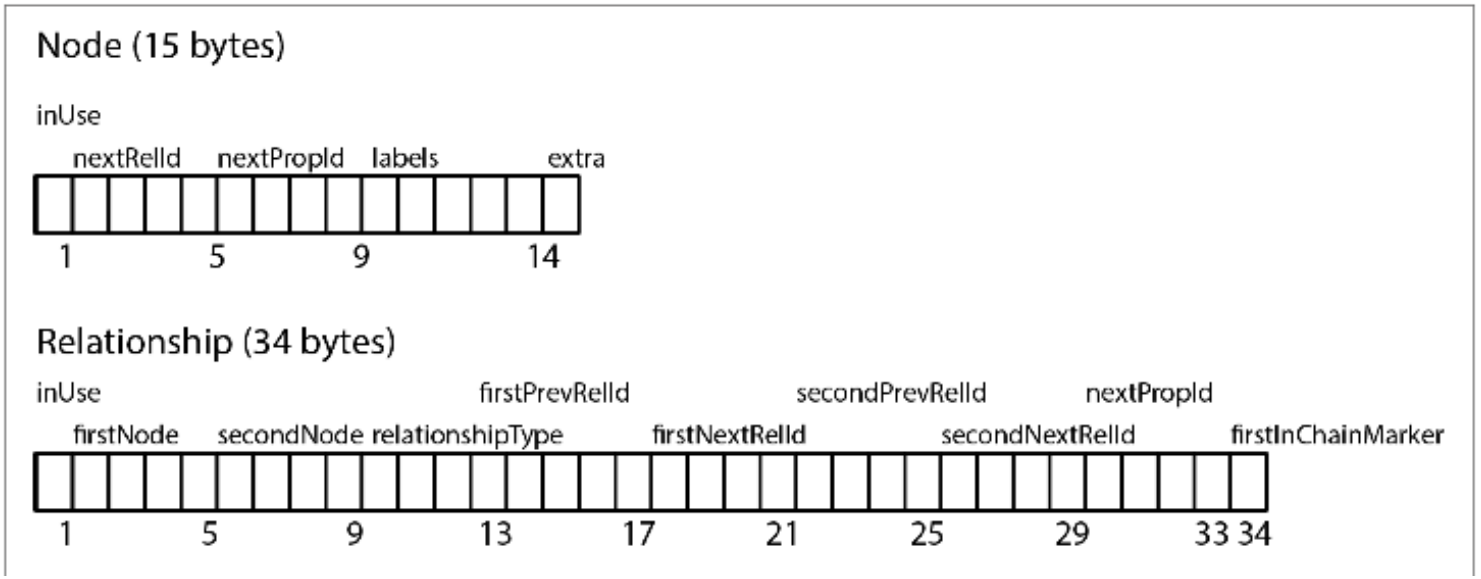
¿cómo se implementa?

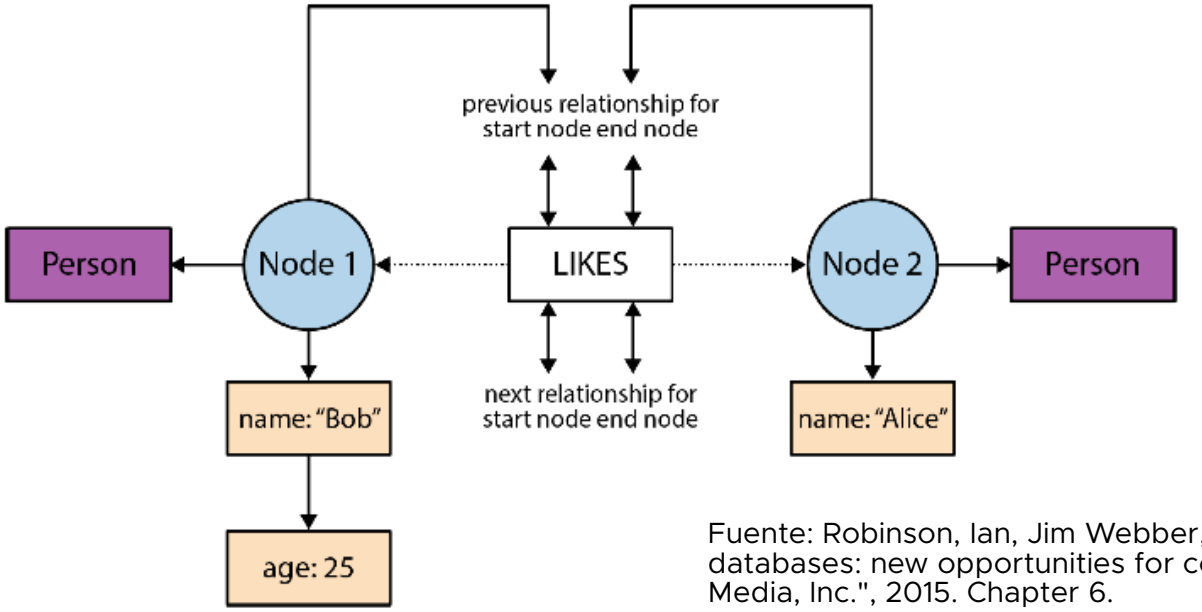
Almacenamiento en Neo4j

- 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

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