

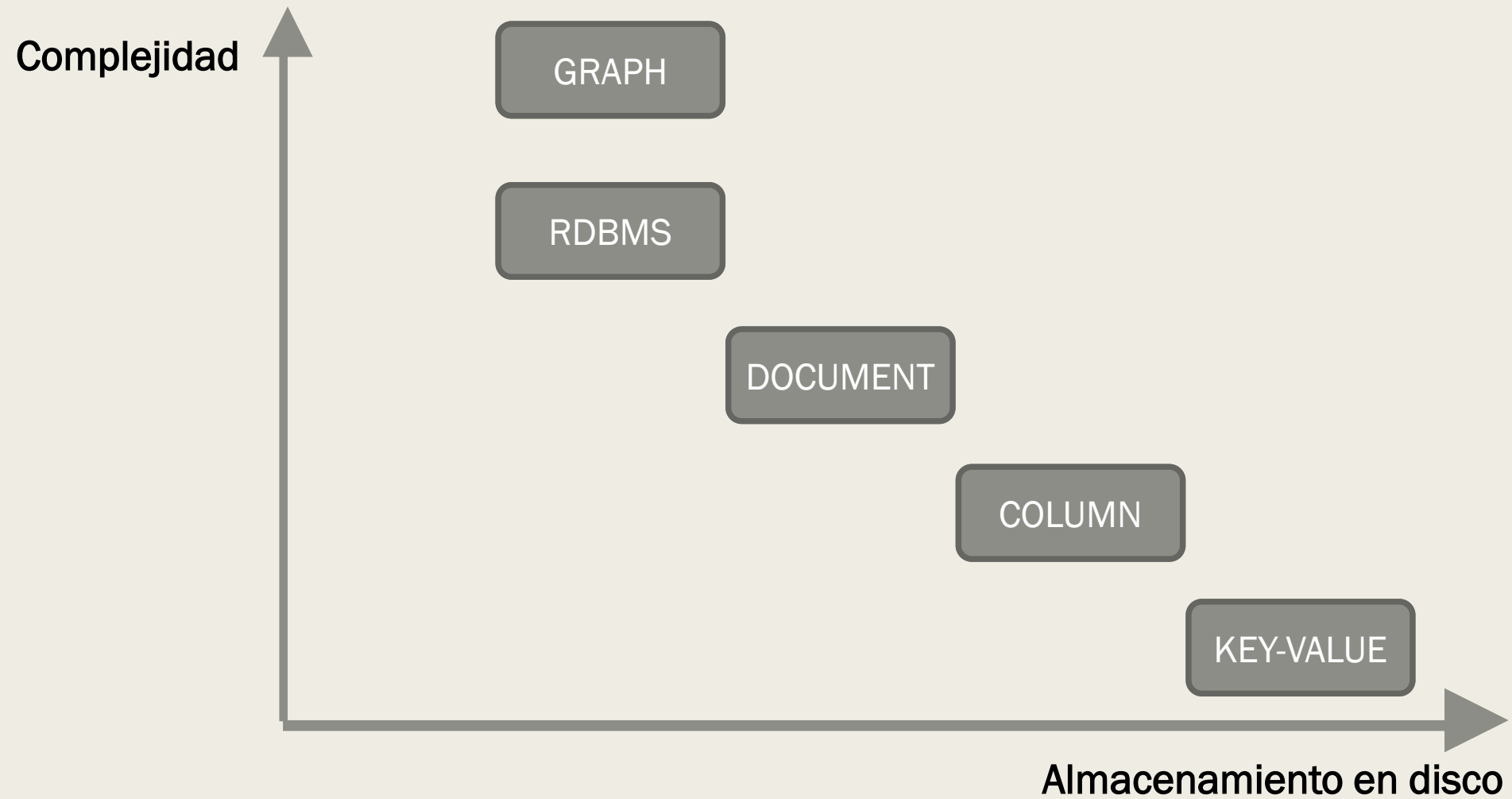


BASES DE DATOS ORIENTADAS A GRAFOS

Neo4J



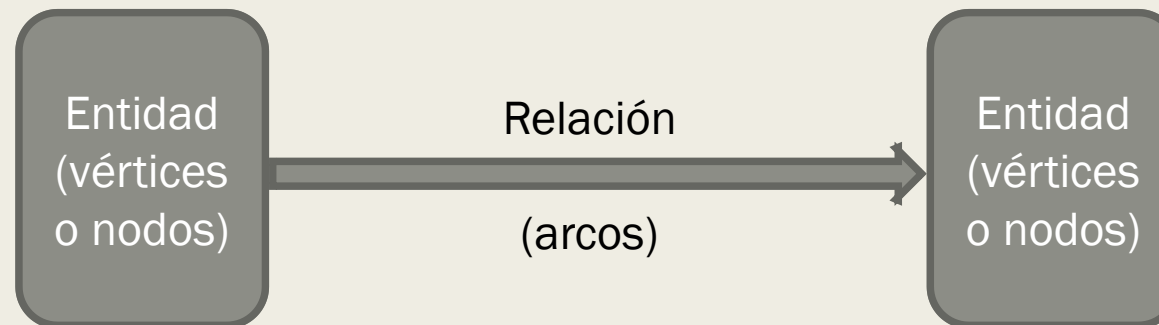
Bases de datos



Bases de datos orientadas a grafos

■ ¿Qué es un grafo?

- *Representación abstracta de un conjunto de objetos (entidad, vértice o nodo)*
- *Los nodos están relacionados a través de enlaces (relación, arco o edges)*



Bases de datos orientadas a grafos

- Los nodos y las relaciones pueden tener atributos o propiedades (pares de valores key-value)
 - *Los atributos permiten identificar de manera única a una relación o a un nodo*



Bases de datos orientadas a grafos

■ Nodos y relaciones en la vida real

El mundo está repleto de información altamente relacionada y desordenada

- *La información del mundo real y sus estructuras están continuamente cambiando*
- *Las relaciones son tan importantes como las entidades que relaciona*
- *Las relaciones permiten modelar interacciones complejas*
- *Las relaciones son parte de los datos*

Bases de datos orientadas a grafos

Sparkier, faster, more: Graph databases, and Neo4j, are moving on

New players, new features, platforms, and ticking boxes. Let's talk graph with Neo4j.



By George Anadiotis for Big on Data | October 24, 2017 -- 13:41 GMT (06:41 PDT) | Topic: Big Data Analytics

¿A Quién le importan las bases de datos orientadas a grafos?

Bing Gets Its Own Knowledge Graph Via Britannica Partnership

Bing's search results are getting a bit more informational thanks to a [new partnership](#) with Encyclopedia Britannica.



Happy Dashboarding

Using Facebook's Graph API Explorer to retrieve Insights data

Facebook Developers

Klipfolio

Apr 5, 2016

Google search now provides access to basic healthcare information in India

by Jon Russell

Google's latest move in India may help increase access to healthcare information among millions of people. The company added healthcare information to its **knowledge graph** in the U.S. last year, to help people easily find quick information in response to symptoms or concerns via its search engine, and now the feature is headed east to India. [Read More](#)



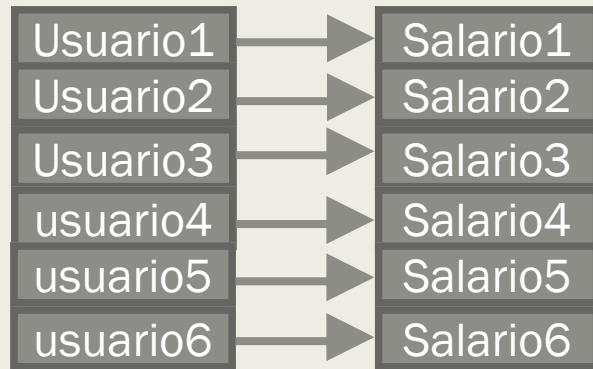
Bases de datos orientadas a grafos

- Ejemplo de un grafo



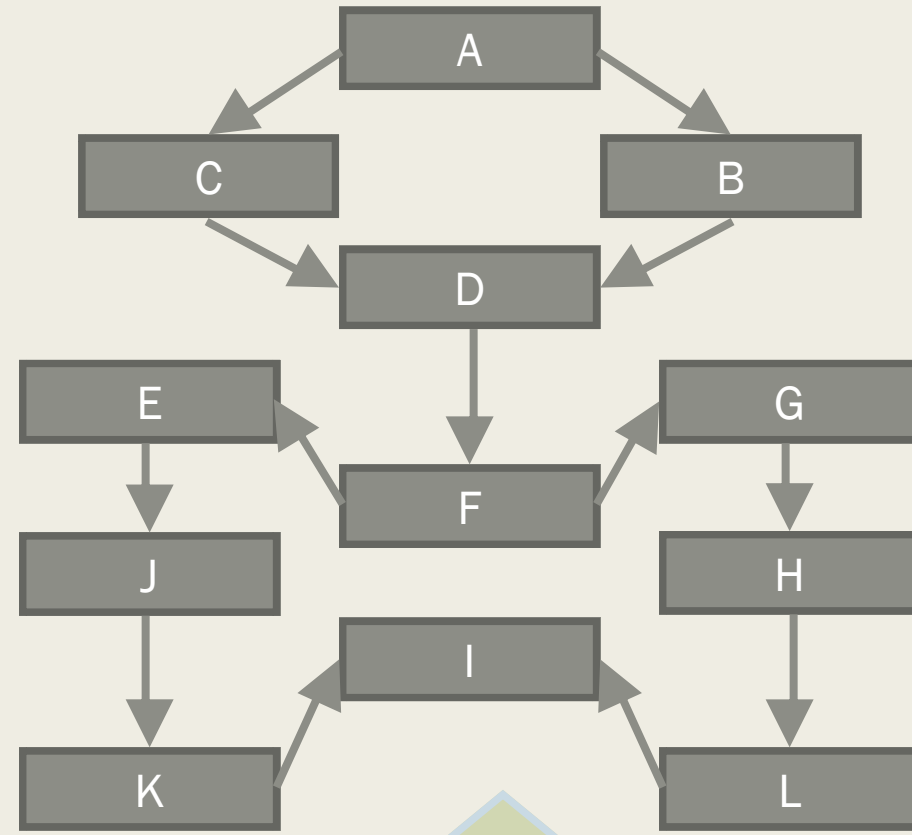
Bases de datos orientadas a grafos

Bases de datos por columnas



Optimizadas para hacer agregaciones

Bases de datos orientadas a grafos



Optimizadas para manipular relaciones

Bases de datos orientadas a grafos

■ Ventajas

- *Muy rápidas para manipular información conectada o relacionada*
- *Define transacciones (como en bases de datos relacionales)*
- *Consultas muy simples*

■ Desventajas

- *Dificultad para hacer Sharding*
 - Sharding significa particionar la base de datos en pequeños conjuntos de datos que son más eficientes y fáciles de administrar
- *Requiere de un cambio conceptual importante (a diferencia de las bases de datos relacionales)*

Ejemplos de bases de datos orientadas a grafos

Algunos ejemplos de bases de datos orientadas a grafos

- Neo4j
 - InfiniteGraph
 - InfoGrid
 - OrientDB
 - BigData
 - DEX
 - HyperGraphDB
 - OQGraph
 - ArangoDB
- TigerGraph
 - Sql Server GRAPH support
 - Microsoft CosmosDB

Neo4J

Neo4J

- Neo4J es un sistema administrador de bases de datos orientado a grafos
 - *Soporta*
 - 32 billones de nodos
 - 32 billones de relaciones
 - 63 billones de propiedades
 - *Soporte para alta disponibilidad*
 - *Soporta consultas SPARK (procesamiento de datos a gran escala)*
 - *Integra un ETL para migrar datos de bases de datos relacionales*

Neo4J

- Está optimizado para:
 - *Datos altamente conectados (relacionados)*
 - *Encontrar caminos entre nodos*
 - *Generar recomendaciones*
- Utiliza un lenguaje de consulta basado en patrones
 - *CYPHER*

Neo4J Nodos

■ Nodos en CYPHER

- *Se utilizan paréntesis para representar nodos ()*
- *Algunos ejemplos de nodos*

()

(matrix) matrix es un identificado aleatorio para el nodo

(:Movie) Movie es una etiqueta que permite agrupar nodos

(matrix:Movie) Una combinación de los anteriores

(matrix:Movie {title: "The Matrix"}) El nodo matrix tiene asociado el atributo “title”

(matrix:Movie {title: "The Matrix", released: 1999}) Más de un atributo asociado al nodo

Neo4J - Nodos

- Creación de un nodo vacío

create(emp:Employee) //donde emp es el nombre del nodo y Employee una etiqueta (tipo de nodo)



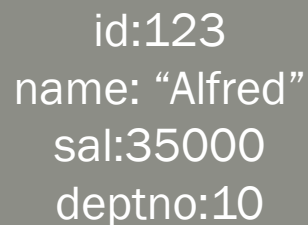
Employee

A gray rounded rectangle representing a node in a graph database. The word "Employee" is centered inside the rectangle in white text.

- Creación de nodos con Atributos

- **create(emp:Employee {id:123,name:"Alfred", sal:35000, deptno:10})** //Los atributos son pares de valores key:value que están delimitados por { }

Employee



id:123
name: "Alfred"
sal:35000
deptno:10

A gray rounded rectangle representing a node in a graph database. The word "Employee" is centered above the rectangle in bold black text. Inside the rectangle, the attributes are listed in white text: "id:123", "name: 'Alfred'", "sal:35000", and "deptno:10".

Neo4J - Nodos

- Búsqueda de nodos (continuación)

match(dept:Dept) return dept.deptno, dept.dname //Match no puede ser una instrucción individual, siempre va con un *return, delete, remove, etc.*

- Ejemplos:

- ° match(tom {name:"Brad Pitt"}) return tom //tom es un identificador arbitrario

- ° match(movies:Movie) return movies.title LIMIT 2

- ° match(personas:Person) where personas.born >= 1960 AND
personas.born <= 1969 return personas

Neo4J Relaciones

- Las relaciones permiten asociar a diferentes nodos
- Las relaciones se representan por: -->
- Para definir atributos para las relaciones se utilizan
 - [] corchetes para indicar que se trata de una relación
 - Pares de valores dentro de { }
- Ejemplos
 - (actor) – [a:ActuoEn] -> (película)
 - (persona1) – [c:Conoce] -> (persona2)

Donde: a y c son nombres arbitrarios para referirnos a una relación

Neo4J - Relaciones

■ Creación de relaciones entre nodos

- `create(p1:Profile1)-[r1:LIKES]->(p2:Profile2)` // Las relaciones pueden ser unidireccionales (->) o bidireccionales (<->). Las relaciones se especifican dentro de los corchetes [nombreRelacion:EtiquetaRelacion]

■ Asignar relaciones a nodos existentes

- `create(e:Customer {id:"1001",name:"Abc",dob:"01/10/1982"})`
- `create(e:Customer {id:"1002",name:"Ariel",dob:"01/14/1980"})`
- `create`
`(cc:CreditCard{id:"5001",number:"1234567890",cvv:"221",expiredate:"20/17"})`
- `match (cust:Customer), (cc:CreditCard) where cust.id="1001" and cc.id="5001"`
`create(cust)-[r:DO_SHOPING_WITH{shopdate:"12/12/2014",price:55000}]->(cc)`
`return r`

Neo4j Búsquedas - Ejemplos

```
match(emp:Empleyee) where emp.id = 123 return emp
```

```
match (tom:Person{name:"Tom Hanks"}) -[:ACTED_IN]->(pelis) return tom,  
pelis
```

```
match (movie{title:"You've Got Mail"})<-[:DIRECTED]-(director) return movie,  
director
```

```
match (tom:Person{name:"Tom Hanks"})-[:ACTED_IN]->(movie)<-[:ACTED_IN]-  
(others) return tom, movie, others
```

Neo4j Búsquedas - Ejemplos

```
match (personas:Person)-[algunaRelacion]-(:Movie {title:"Apollo 13"}) return
personas.name, Type(algunaRelacion),algunaRelacion
```

```
match(bacon:Person {name:"Kevin Bacon"})-[* 1..4]-(others) return others
```

```
match p = shortestPath( (bacon:Person{name:"Al Pacino"})-[*]-
(meg:Person{name:"Tom Cruise"}) ) return p
```

```
match (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-
(coActors), (coActors)-[:ACTED_IN]->(m2)<-[:ACTED_IN]-(cruise:Person
{name:"Tom Cruise"}) return tom, m, coActors, m2, cruise
```

Neo4J – Shortest path

- Busca el camino más corto para encontrar dos nodos que cumplan cierta condición
 - *Neo4j implementa el algoritmo The Shortest Path para encontrar de manera óptima la ruta más corta*

```
match p = shortestPath( (bacon:Person{name:"Al Pacino"})-[*]-  
  (meg:Person{name:"Tom Cruise"}) ) return p
```

- El * indica que busque todos los posibles saltos entre nodos y relaciones
 - *Se puede especificar un número máximo de saltos entre relaciones [1..4]*

Neo4j Borrar nodos

- Borrar un grupo de nodos
 - `match(e:Employee) delete e`
- Borrar una relación entre nodos
 - `match(s:Us1)-[r]-(t:Us2) delete r`
- Borrar una relación y los nodos de la relación
 - `match(s:Us1)-[r]-(t:Us2) delete r,s,t`

Neo4j Borrar nodos

- Borrar atributos de nodos específicos

```
CREATE (book:Book {id:122,title:"C++ Tutorial",pages:340,price:250})
```

```
CREATE (book:Book {id:123,title:"C p Second",pages:340,price:250})
```

```
match (b:Book{id:122}) remove b.price return b
```

```
match (b:Book) remove b.price return b
```

- Eliminar todos los nodos y todas las relaciones
 - `match (n) detach delete n`

Neo4J – Todos los nodos y relaciones

- Buscar todos los nodos de la base de datos (con sus respectivas relaciones)
 - *Hay dos formas de mostrar todos los nodos y todas las relaciones:*

a) `start n=node(*) return n`

b) `start n=node(*) match (n)-[r]-(m) return n,r,m`

Ejercicio 1

```
CREATE (js:Person { name: "Johan", from: "Sweden", learn: "surfing" }),  
(ir:Person { name: "Ian", from: "England", title: "author" }),  
(rvb:Person { name: "Rik", from: "Belgium", pet: "Orval" }),  
(ally:Person { name: "Allison", from: "California", hobby: "surfing" }),  
(ee)-[:KNOWS {since: 2001}]->(js),(ee)-[:KNOWS {rating: 5}]->(ir),  
(js)-[:KNOWS]->(ir),(js)-[:KNOWS]->(rvb),  
(ir)-[:KNOWS]->(js),(ir)-[:KNOWS]->(ally),  
(rvb)-[:KNOWS]->(ally)
```

```
match(js:Person)-[:KNOWS]-(a)-[:KNOWS]-(surfer) where js.name = "Johan" AND surfer.hobby = "surfing"  
return surfer, js, a
```

Ejercicio 2

1. Mostrar el top 5 de actores que han actuado en más películas
2. Mostrar el actor que ha actuado en más películas
3. Mostrar el actor con más edad
4. Mostrar el actor con menos edad
5. mostrar los actores que actuaron en las películas que revisó la persona que más ha revisado películas