

Neo4J

José R.R. Viqueira

- **Introducción**
- **Conceptos de BD de Grafos**
- **Lenguaje Cypher**



■ Ediciones

- ▷ Community Edition
 - Despliegue en una sola instancia
 - Funcionalidad completa: ACID, Cypher, APIs de programación.
 - Aprendizaje, pequeños proyectos personales, pequeñas aplicaciones.
- ▷ Enterprise Edition
 - Mejor rendimiento y escalabilidad (Clustering, seguridad, etc.)
- ▷ Características principales de cada edición en
 - <https://neo4j.com/docs/operations-manual/current/introduction/>

■ Versiones

- ▷ Mayor.Menor.patch
- ▷ Versiones mayores pueden ser incompatibles con anteriores
- ▷ Versiones menores son compatibles hacia atrás
- ▷ Los patch corrigen errores



- Interacción con lenguajes de programación
 - ▷ Drivers oficiales para: Java, JavaScript, Python, .Net, Go
- Otras herramientas
 - ▷ Shell de Cypher: Realizar consultas Cypher on-line.
 - ▷ Browser: Interaccionar con la BD, crear consultas Cypher, visualización básica.
 - ▷ Desktop: IDE de desarrollo y Entorno de gestión para la BD.
 - ▷ Bloom: Permite explorar y visualizar el grafo.
- Conceptos relacionados con la gestión de las BDs
 - ▷ DBMS: Sistema Gestor e Bases de datos. Puede gestionar varias BDs. Puede gestionar un servidor standalone o un cluster.
 - ▷ Instancia: Proceso java en el que se ejecuta código del servidor
 - ▷ Dominio de transacción: Colección de grafos que se pueden modificar en el contexto de una transacción
 - ▷ Contexto de ejecución: Entorno en el que se ejecuta una petición (consulta transacción o función o procedimiento interno)
 - ▷ Base de datos: Uno o más grafos. Define el dominio de una transacción o el contexto de una petición. Instalación por defecto tiene 2 BDs (system y neo4j)
 - ▷ Grafo: Modelo de datos dentro de una base de datos. Normalmente hay solo uno en cada BD

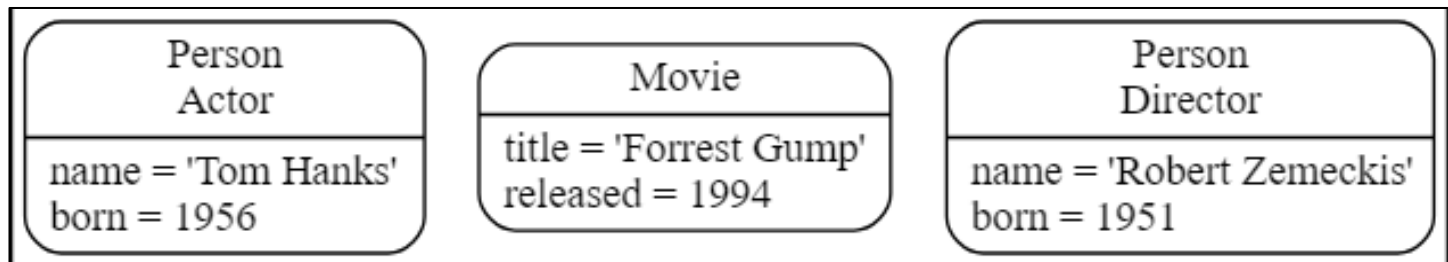
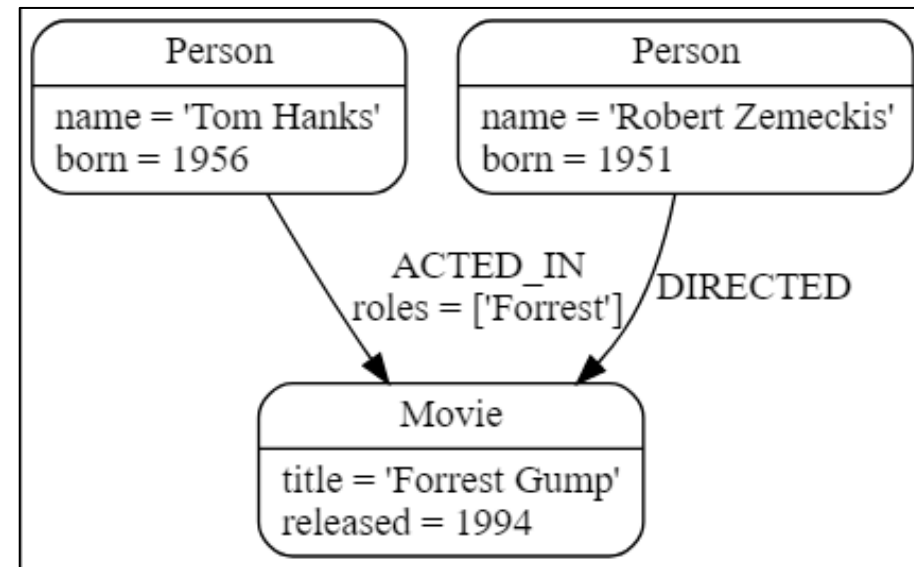
■ Ejemplo de Grafo

■ Nodos

- ▷ Representan entidades
- ▷ Grafo más simple: 1 nodo

■ Etiquetas

- ▷ Agrupan nodos en conjuntos
 - Person, Director, Movie, etc.
- ▷ Se podrá operar en los nodos de una determinada etiqueta
- ▷ Se pueden incluir en tiempo de ejecución, con lo que podrían ser temporales
 - Suspended
- ▷ Un nodo puede no tener etiqueta y puede tener varias

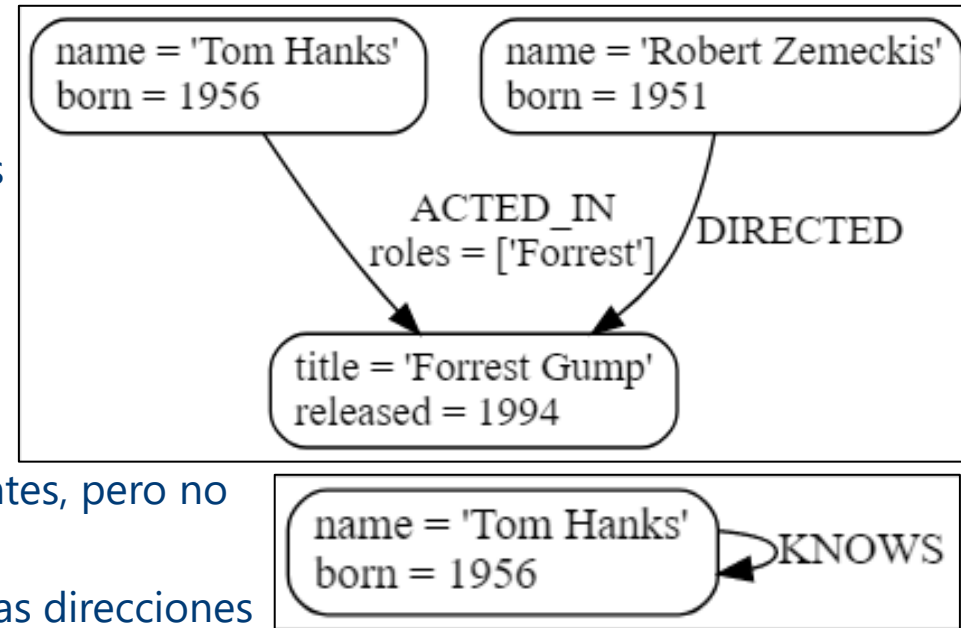


■ Relaciones

- ▷ Conecta dos nodos
- ▷ Permiten organizar los datos en estructuras complejas

■ Tipos de relaciones

- ▷ Solo un tipo
 - **ACTED_IN, DIRECTED, etc.**
- ▷ Relaciones entrantes y salientes, pero no ambas al mismo tiempo.
- ▷ No es necesario añadir ambas direcciones al grafo, ya que vamos a poder navegar hacia adelante y hacia atrás
- ▷ Las relaciones pueden entrar y salir del mismo nodo (reflexivas).



■ Propiedades

- ▷ Pares (clave-valor) que describen nodos y relaciones
 - **name, born, roles, title, etc.**
- ▷ Tipos de dato del valor: número, string, boolean, etc.
 - Tipos compuestos: Listas, mapeos.
- ▷ <https://neo4j.com/docs/cypher-manual/4.2/syntax/values/>



■ Recorridos y caminos

- ▷ Recorridos: Forma de realizar consultas en una BD de grafos
- ▷ Recorren el grafo visitando nodos a través de relaciones, siguiendo ciertas reglas
 - Películas en las que actuó Tom Hanks

■ Esquema

- ▷ Compuesto por índices y restricciones. No son necesarios
- ▷ Índices: Mejoran el rendimiento
- ▷ Restricciones: Aseguran que los datos siguen ciertas reglas

■ Reglas de nombrado y recomendaciones

- ▷ Lenguaje sensitivo al uso de mayúsculas
- ▷ Etiquetas Nodos: Usar Camel case (empieza con mayúscula)
 - VehicleOwner
- ▷ Tipos de relaciones: Mayúsculas y guión bajo para unir componentes.
 - OWS_VEHICLE
- ▷ Propiedades: Usar Lower camel case (empieza con minúscula)
 - firstName



■ Patrones

esto devuelve tuplas (no es un sistema cerrado, ya que consulta grafos y devuelve tuplas)

- ▷ Encajan con estructuras complejas de grafos
- ▷ Patrón simple: Par de nodos unidos por una relación
 - _ a Person LIVES-IN a City
- ▷ Se pueden formar patrones más complejos
 - _ (:Person) -[:LIVES_IN]-> (:City) -[:PART_OF]-> (:Country)
- ▷ Sintaxis para nodos
 - _ () Nodo anónimo
 - _ (matrix) Variable que itera por nodos
 - _ (:Movie) Nodo que tenga una etiqueta Movie
 - _ (matrix:Movie) matrix es una variable que itera por los nodos con etiqueta Movie
 - _ (matrix:Movie {title: "The Matrix"})
 - Ahora matrix itera por los nodos de tipo Movie con el valor "The Matrix" en el título.
 - _ (matrix:Movie {title:"The Matrix", released:1997})



■ Patrones

▷ Sintaxis para relaciones

- _ -- Relación sin dirección concreta
- _ --> Relación saliente
- _ <-- Relación entrante
- _ -[role]-> role es una variable que itera sobre relaciones salientes
- _ -[:ACTED_IN]-> relaciones salientes de tipo ACTED_IN
- _ -[role:ACTED_IN]-> role itera sobre relaciones salientes de tipo ACTED_IN
- _ -[role:ACTED_IN {roles: ["Neo"]}]->
 - También se pueden añadir patrones sobre propiedades

▷ Sintaxis de los patrones

- _ Combinan nodos con relaciones. Ejemplo
 - (keanu:Person:Actor {name: "Keanu Reeves"})
 - [role:ACTED_IN {roles: ["Neo"]}]->
 - (matrix:Movie {title: "The Matrix"})

▷ Variables de patrón

- _ Se pueden asignar variables a los patrones
 - acted_in = (:Person)-[:ACTED_IN]->(:Movie)
- _ Se puede acceder a las distintas partes de los caminos recuperados. nodes(path), relationships(path), length(path)



■ Cláusulas

- ▷ Las sentencias en Cypher tienen varias cláusulas
- ▷ Tareas que se realizan con la cláusulas

- Crear patrones en el grafo
- Encajar patrones en el grafo
- Filtrar resultados
- Proyectar partes del resultado
- Ordenar y paginar resultados
- Componer sentencias parciales

se usa para las bases de datos en las que hay que hacer joins entre muchas tablas para bases de datos centralizadas. No vamos a tener un cluster, ya que en ellos cuando queremos acelerar, particionamos. Un grafo no se particiona.

▷ Ejemplos

- ```
MATCH (p:Person { name:"Tom Hanks" })
CREATE (m:Movie { title:"Cloud Atlas",released:2012 })
CREATE (p)-[r:ACTED_IN { roles: ['Zachry']}]-(m)
RETURN p,r,m
```
- ```
MATCH (p:Person)-[:ACTED_IN]->(m)
WHERE NOT (p)-[:DIRECTED]->()
RETURN p,m
```
- ```
MATCH (actor:Person)-[:ACTED_IN]->(movie:Movie)<-[:DIRECTED]-(director:Person)
RETURN actor, director, count(*) AS collaborations
```

el group by se hace por todo lo que no se mete en agregacion

# Neo4J

<https://neo4j.com/docs/>

José R.R. Viqueira