

# Repositorios de Información

# Neo4j

*Ingeniería Informática del Software*

Escuela de Ingeniería Informática de Oviedo

Darío Álvarez Gutiérrez

# Neo4j

- Base de datos de grafo
  - Transacciones ACID
  - Lenguaje manejo datos
- Servidor / inmersa (*embedded*)
- Escalable / Alta disponibilidad (*v. Enterprise*)
  - Maestro / Esclavo
  - Escrituras en esclavo
- Panel web administración / manejo / visualización
- *Open Source*

# Recursos

- Neo4j.org
  - Descargas
  - Introducción, tutoriales
  - Documentación
    - <http://neo4j.com/docs>
- Consola interactiva de prueba
  - <http://console.neo4j.org>
- Sandbox – servidor Neo4j en la nube (7 días)
  - <http://neo4j.com/sandbox>

# Recursos

- Libros (algunos gratis)
  - <http://neo4j.com/books/>
- GraphAcademy
  - <http://neo4j.com/graphacademy/>
- Curso online “Getting Started with Neo4j”
  - <http://neo4j.com/graphacademy/online-course/>
- StackOverflow
  - <http://stackoverflow.com/tags/neo4j/info>

# Conceptos fundamentales

- Modelo de datos: grafo de propiedades
  - <http://neo4j.com/developer/graph-database/>
- Comparativa otros modelos datos
  - <http://neo4j.com/developer/graph-db-vs-nosql/>
  - [A Glance into the Developer's World of Data: Graphs, RDBMS and NoSQL \(Sabhya Kaushal\)](#)

# Ámbitos de aplicación

- *"Graph All the Things"*
  - <http://www.slideshare.net/maxdemarzi/data-20>

# Instalación Neo4j

- Descargar, descomprimir
- Ojo con variables de entorno
  - PATH al bin del java
  - JAVA\_HOME
- Arrancar servidor en puerto 7474 DESDE *SHELL*
  - > neo4j console (versión 3)
- Entorno web: <http://localhost:7474>
- Parar servidor
  - Ctrl-C

# Versiones

- Producto en evolución constante
  - Mejora y CAMBIOS funcionalidad
  - *CAMBIO SINTAXIS LENGUAJES*
  - Desaparición características (*deprecated*)
  - Cambio formato almacenamiento
- $1.9.x \Rightarrow 2.x.x \Rightarrow 3.x.x$ 
  - <http://www.slideshare.net/neo4j/new-in-2>
  - <http://neo4j.com/whats-new-in-neo4j-2-3/>
  - <https://neo4j.com/blog/neo4j-3-0-massive-scale-developer-productivity/>

# Interacción con SGBD

- API lenguaje (Java) (recorrido grafos)
- API REST
- Consola
  - Shell
  - REST
- Lenguajes específicos de manejo de datos
  - *Cypher*
    - Consultas declarativas
  - [*Gremlin* (versiones anteriores, instalable)]
    - Lenguaje estándar recorrido grafos, procedimental

# Panel webadmin ["clásico"] (pre v3)

- <http://localhost:7474/webadmin/>
- Visualización / edición datos
- Consola
  - Shell
  - [Gremlin (instalable)]
  - REST
- Gestión (escasa) índices “antiguos”
- Visor configuración / estadísticas

# Manejo datos visor ["clásico"] (pre v3)

- Data Browser
- +Node, +Relationship
- Modo texto / modo gráfico

# Creación grafo visor ["clásico"] (pre v3)

- Crear
- Visualizar

# Manejo datos shell

- Power tool (webadmin “clásico” *pre v3*) / neo4j-shell.bat (act. en neo4j.conf)
- help, man
- cd -> navegar por el grafo por id (cambia elemento actual)
- mknod, mkrel -> crear nodo, relación
- ls -> muestra elemento actual
- set, rm -> establece, elimina una propiedad
- rmnode, rmrel -> eliminar nodo, relación

# Creación grafo shell

- Crear
- Visualizar

# Visor nuevo

- <http://127.0.0.1:7474/browser>
- Editor consultas
- Visualización grafos
- Minitutoriales

# Lenguaje de datos Cypher

- Declarativo
- Encaje de patrones (caminos) en el grafo a partir de unos nodos iniciales (estilo SparQL parcialmente)
- Estilo SQL parcialmente
- Estilo funcional parcialmente (colecciones)
- Variables ligadas, libres
- Expresiones regulares
- *CAMBIOS versión 2.0, EVOLUCIONANDO (3.0)*

# Recursos Cypher

- Cypher tutorial
  - <http://neo4j.com/developer/cypher-query-language/>
  - Demo console
    - PISTA: Nueva línea con Shift+Enter
  - Curso introducción
  - Hoja referencia (*Cypher RefCard*), *DZone refcard* (v 2.0)
  - [Referencia del lenguaje](#)
  - [Ejemplos de uso de Cypher](#) (Sabhya Kausal)

# Cypher y versiones antiguas

- Documentación -> verificar versión (1/2/3.x)
- Diferencias principales versión 2.0 – 1.0
  - Etiquetas
  - Nodos entre paréntesis (obligatorio)
    - *MATCH (n) RETURN n*
  - START opcional
  - CREATE solo estilo nodo
    - *CREATE (pepe {nombre: "Pepe"})*
  - Propiedades no existentes devuelven NULL  
(Desaparece ? y ! en propiedades – v1.x)
  - OPTIONAL MATCH para relaciones opcionales

# Cypher estructura principal

- MATCH <patrón camino>, <patrón camino>, ...

*MATCH (n) RETURN n*

*MATCH p= (n)-->(i)<-[r:COMPRA | PIDE]-(m),  
q= (n)-[s]-(m)*

- WHERE <condición sobre variables>

*WHERE n.edad>18 AND m.edad<65  
AND NOT((n)-[:ODIA]->(i))*

- RETURN <expresión sobre variables>

*RETURN n.edad, i, m, length(nodes(p)), r.fecha,  
type(s)*

# Notación nodos y caminos

- <http://bit.ly/cypher-slide>

# Consultas sencillas: índices y etiquetas

- ...
- Índices “tradicionales” (v1.x)
  - Puntos de inicio para consulta (START)
- Etiquetas (*labels*) (v2.0)

# Etiquetas (*labels*)

- Etiquetas agrupan nodos en conjuntos
  - antonio:Persona
- Modelado, clasificación, identificación
  - NO ES-UN
  - NO atributo “tipo”
- START innecesario (v1.x)
  - Sustituye nodo referencia (etiquetar nodos especiales)
- Indexación de atributos basada en etiquetas
  - CREATE INDEX ON :Persona(nombre)

# Carga de un grafo

- Crear mediante script Cypher
  - Grafo ejemplo visor “Movies”
- Cargar el almacenamiento
  - Descargar grafo ejemplo  
“cineasts 39 movies 446 actors.zip” (v1.0)  
“cineasts 39 movies 446 actors.zip graphdb 2.0.zip”  
del campus virtual (v2.0)
    - Simplemente reemplazar directorio  
data/graph.db (v2) / data/databases/graph.db (v3)

# Actualización y visualización de un grafo

- Actualizar almacenamiento si necesario
  - allow\_store\_upgrade=true en neo4j.properties (v2) / dbms.allow\_format\_migration=true neo4j.conf (v3)
- Explorar modelo de datos
- CAMBIAR VISUALIZACIÓN

# Alternativa a los índices

- Si tengo un índice por Actor o automático (v1.x)

*START n = node:Actor(name = "Zoe Saldana")*

*START n =*

*node:node\_auto\_index(name = "Zoe Saldana")*

(Ojo, saca todos los nodos con nombre, no solo actores necesariamente)

- Si tengo información en el dominio

*[START n = node (\*)]*

*MATCH (n)*

*WHERE n.\_\_type\_\_ =*

*"org.cineasts.domain.Actor"*

*o n.\_\_type\_\_ =~ "\*Actor"*

# Sustitución de los índices: Etiquetas

- Si tengo etiquetado por Actor

```
MATCH (n:Actor)
```

```
WHERE n.name = "Zoe Saldana"
```

```
MATCH (n:Actor {name:"Zoe Saldana"})
```

- Añadir etiquetas (actualizar) SET

```
MATCH (n)
```

```
WHERE n._type_ =  
"org.neo4j.cineasts.domain.Actor"
```

```
SET n:Actor
```

# Cypher estructura completa

- [START]
- MATCH
  - WHERE
- OPTIONAL MATCH
  - WHERE
- RETURN
  - ORDER BY, SKIP, LIMIT
- WITH

# Consultas

• ...

# Modificación datos

- CREATE
  - Uso de mapas {propiedad: “valor”, ...}
  - Creación múltiple de nodos
    - Útil para el grafo de la práctica. Véase creación del grafo “Movies” en el visor
- CREATE UNIQUE => MERGE (“create or replace”)
- DELETE nodos, relaciones
- REMOVE propiedades, etiquetas
- SET propiedades, etiquetas

# Esquema – Restricciones Integridad

- CREATE / DROP CONSTRAINT (v. 3.0)
- Inicialmente solo IS UNIQUE (usa etiquetas)
  - *CREATE CONSTRAINT ON (persona:Persona) ASSERT persona.DNI IS UNIQUE*
  - *DROP CONSTRAINT ON (persona:Persona) ASSERT persona.DNI IS UNIQUE*
- Propiedad obligatoria en nodos o relaciones
  - *CREATE CONSTRAINT ON (persona:Persona) ASSERT exists(persona.DNI)*
  - *CREATE CONSTRAINT ON ()-[odia:ODIA]-() ASSERT exists(odia.motivo)*

# Esquema – Restricciones Integridad

- Clave de nodos
  - *CREATE CONSTRAINT ON (c:Clase)  
ASSERT (c.edificio, c.sala) IS NODE KEY*
  - Permite claves compuestas
  - Existencia de propiedades es obligatoria

# Más ejemplos de Cypher

- Ejemplos Cypher, diseño, implementación, comparativa SQL (v1.x)
  - <http://bit.ly/cypher-slide>
- Modelado de dominios con ejemplos de consultas en Cypher
  - <http://neo4j.com/docs/stable/data-modeling-examples.html>
- Comparativa consultas SQL - Cypher
  - <https://neo4j.com/docs/2.3.7/cypherdoc-cypher-vs-sql.html>

# Información adicional

- Bases de datos de ejemplo (data/graph.db)
  - <http://neo4j.com/developer/example-data/>
- Ejemplos de ámbitos de aplicación de bases de datos de grafo
  - <http://www.slideshare.net/peterneubauer/spotifylovesgraphstraversesneo4j>



# Cypher estructura principal (v1.x)

- START <nodos partida>, <nodos partida>,...

*START n = node(1),  
m = node:IndiceCliente(ciudad = "Oviedo"),  
r = relationship(45, 46, 47)*

- MATCH <patrón camino>, <patrón camino>,...

*MATCH p= n-->i<- [r:COMPRA | PIDE]-m, q= n-[s]-m*

- WHERE <condición sobre variables>

*WHERE n.edad>18 AND m.edad<65 AND NOT(n-[:ODIA]->i)*

- RETURN <expresión sobre variables>

*RETURN n.edad, i, m, length(p), r.fecha, type(s)*

# Índices “tradicionales” (v1.x)

- Índices sobre nodos / relaciones
- Asocian una clave-valor ARBITRARIA con un objeto
- Varias claves en el mismo índice
  - Ej: Películas por año, por título, y por nacionalidad
- ADICIÓN MANUAL
  - Orden “index” en el shell
- Índices automáticos globales
  - node\_auto\_index, relationship\_auto\_index

# Reconstruir node\_auto\_index (v1.x)

- Para utilizar la indexación automática (node\_auto\_index, relationship\_auto\_index)
  - Exportar la base de datos a fichero
    - Gremlin: `g.saveGraphML("BD.xml")`
    - Renombrar las propiedades "id" por algo distinto ("\_id")
  - Parar el servidor, activar auto-indexación e indicar atributos a indexar
    - config/neo4j.properties
  - Arrancar el servidor y borrar la base de datos
    - Gremlin: `g.clear()`
  - Cargar la base de datos del fichero (reindexa)
    - Gremlin: `g.loadGraphML('file:BD.xml')`