

Curso NoSQL Neo4J



Intro a Bases de Grafos – Modelado – Consultas



Índice

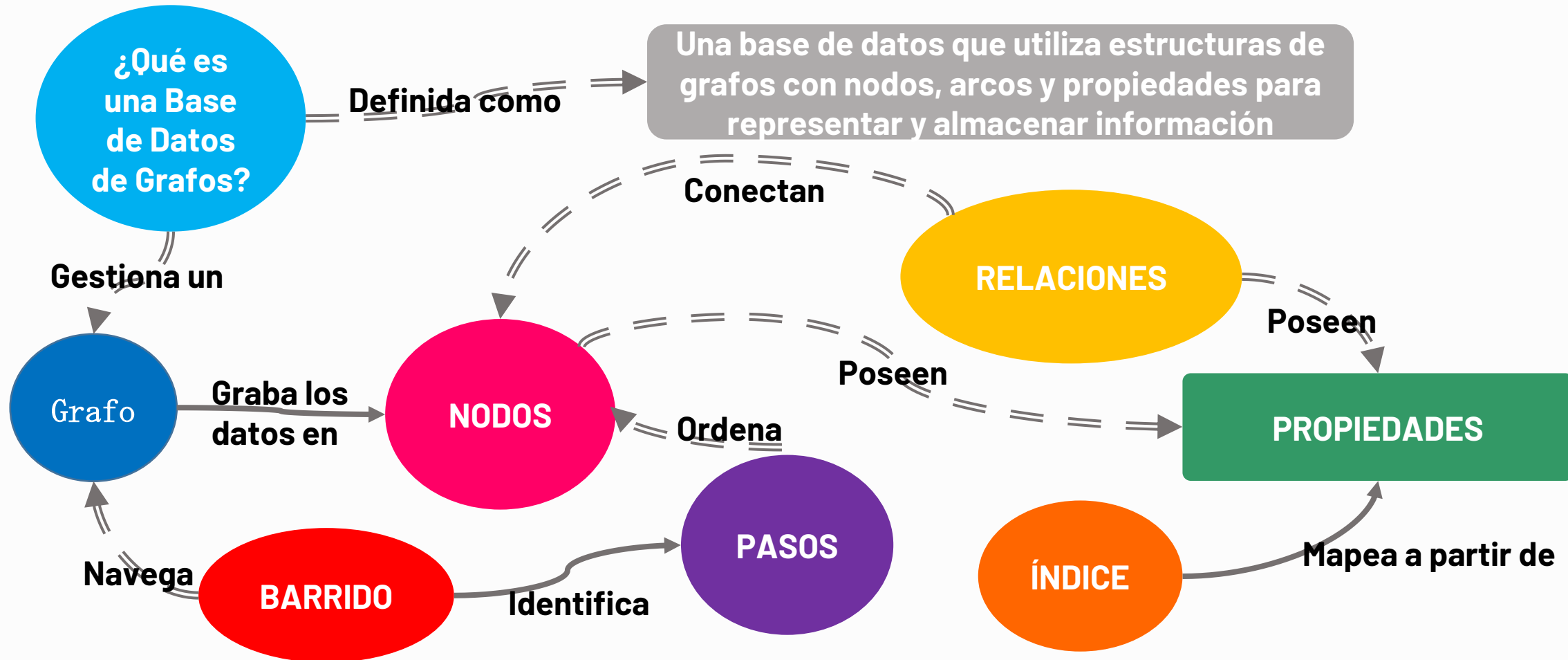
- ★ Introducción a Neo4J
- ★ Ejemplos y Categorías de BD basadas en grafos
- ★ Historia de Neo4J
- ★ Forma de consulta de datos
- ★ En qué casos usarlas y en qué casos no
- ★ Algunos casos de éxito
- ★ Caso Práctico: Red social profesional
- ★ Lenguaje Cypher
- ★ Nodos, relaciones y sus tipos de datos
- ★ Crear, consultar, modificar y eliminar nodos/relaciones
- ★ Modelando en Bases de datos Basadas en Grafos
- ★ Presentación de Neo4J Browser y Cypher Shell



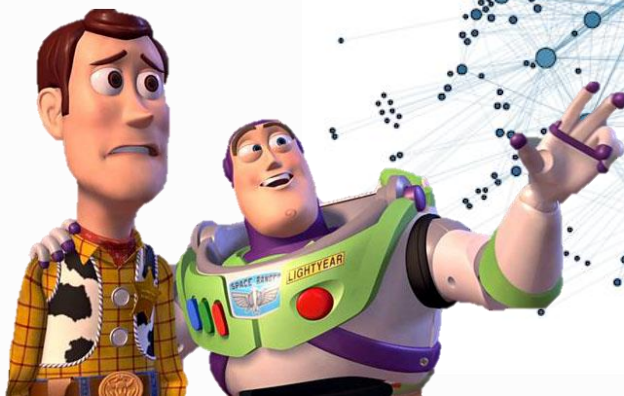
Introducción a Neo4J



Introducción



Todo es un grafo



DBlandIT



Ejemplo gráfico

Los componentes de un grafo Neo4j incluyen:

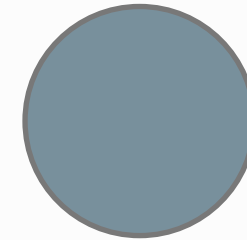
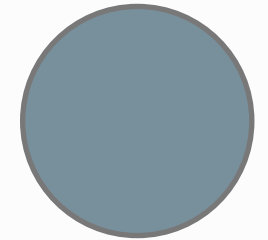
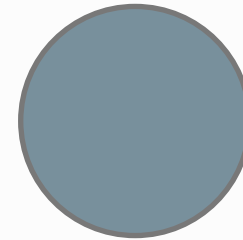
Nodos	0
Etiquetas/Labels	0
Propiedades	0
Relaciones	0
Tipo	0
Propiedades	0



Ejemplo gráfico

Los componentes de un grafo Neo4j incluyen:

Nodos	3
Etiquetas/Labels	0
Propiedades	0
Relaciones	0
Tipo	0
Propiedades	0



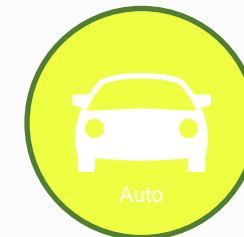
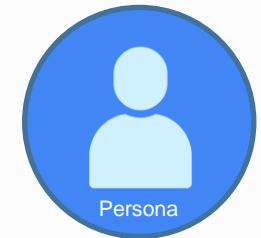
DBlandIT



Ejemplo gráfico

Los componentes de un grafo Neo4j incluyen:

Nodos	3
Etiquetas/Labels	2
Propiedades	0
Relaciones	0
Tipo	0
Propiedades	0



Ejemplo gráfico

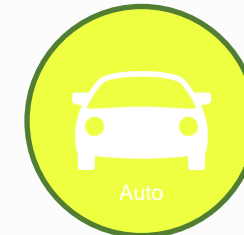
Los componentes de un grafo Neo4j incluyen:

Nodos	3
Etiquetas/Labels	2
Propiedades	7
Relaciones	0
Tipo	0
Propiedades	0

Nombre: "Juan Carlos"
Nacimiento: "15/06/1980"
Instagram: "JuanCarlos"



Nombre: "Mariela"
Nacimiento: "25/08/1982"



Marca: "Volvo"
Modelo: "v70"



DBlandIT



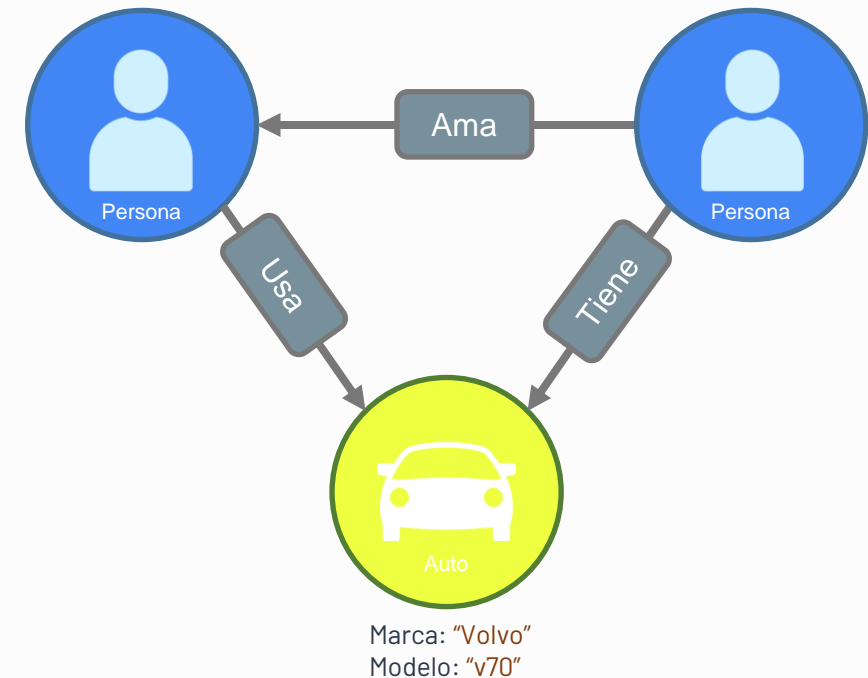
Ejemplo gráfico

Los componentes de un grafo Neo4j incluyen:

Nodos	3
Etiquetas/Labels	2
Propiedades	7
Relaciones	3
Tipo	3
Propiedades	0

Nombre: "Juan Carlos"
Nacimiento: "15/06/1980"
Instagram: "JuanCarlos"

Nombre: "Mariela"
Nacimiento: "25/08/1982"



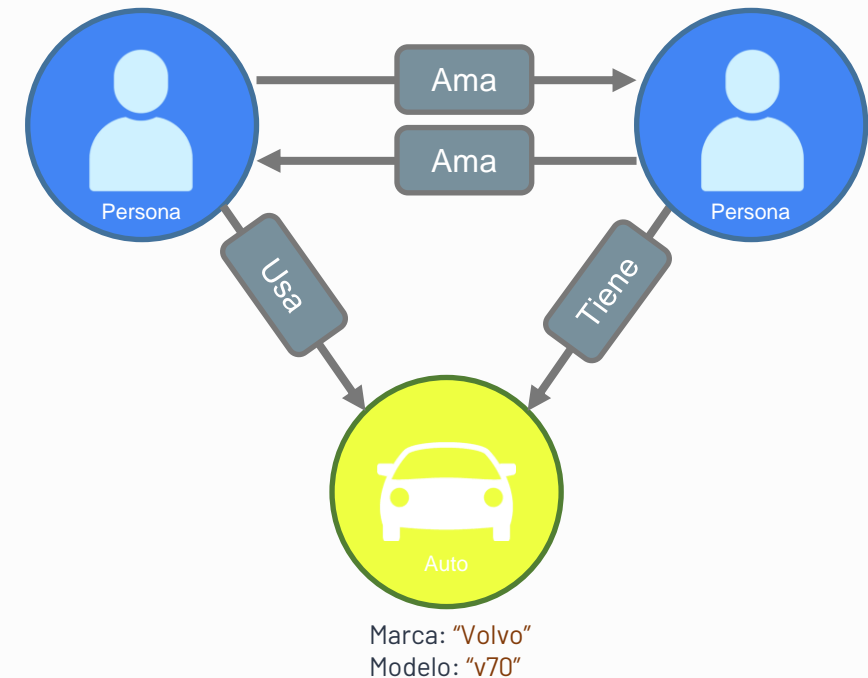
Ejemplo gráfico

Los componentes de un grafo Neo4j incluyen:

Nodos	3
Etiquetas/Labels	2
Propiedades	7
Relaciones	4
Tipo	3
Propiedades	0

Nombre: "Juan Carlos"
Nacimiento: "15/06/1980"
Instagram: "JuanCarlos"

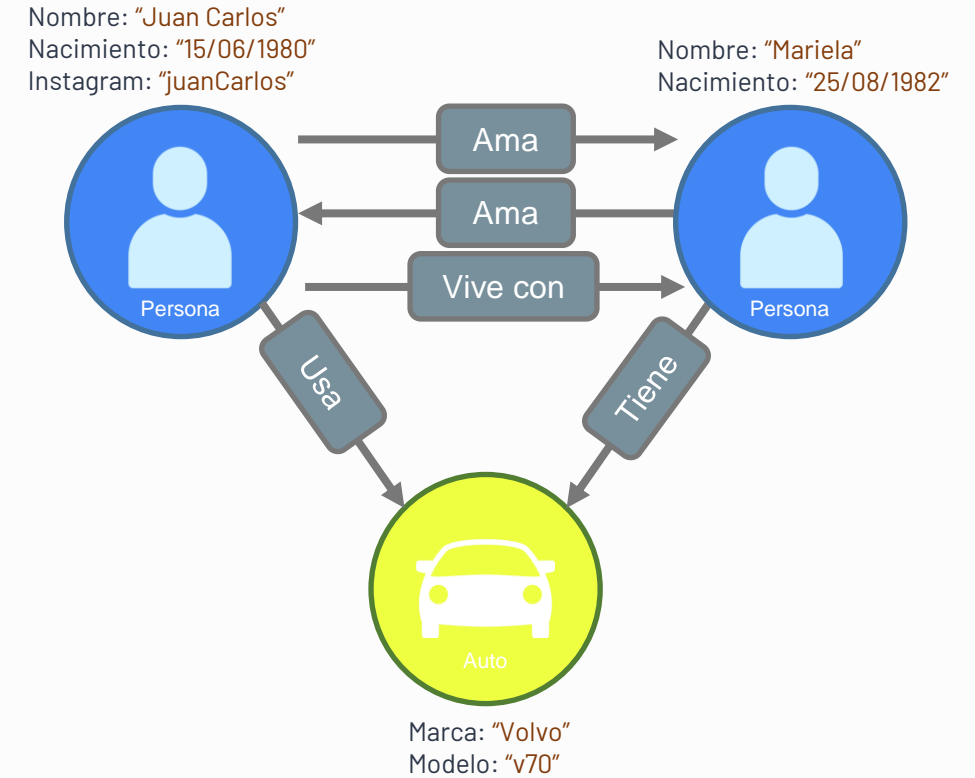
Nombre: "Mariela"
Nacimiento: "25/08/1982"



Ejemplo gráfico

Los componentes de un grafo Neo4j incluyen:

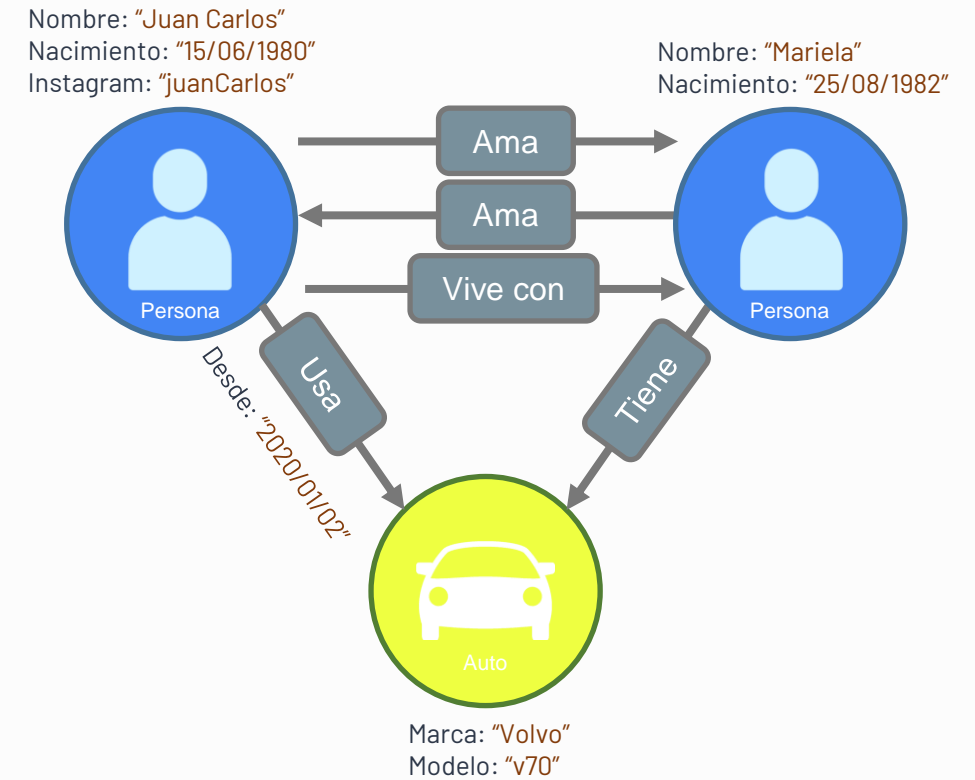
Nodos	3
Etiquetas/Labels	2
Propiedades	7
Relaciones	5
Tipo	4
Propiedades	0



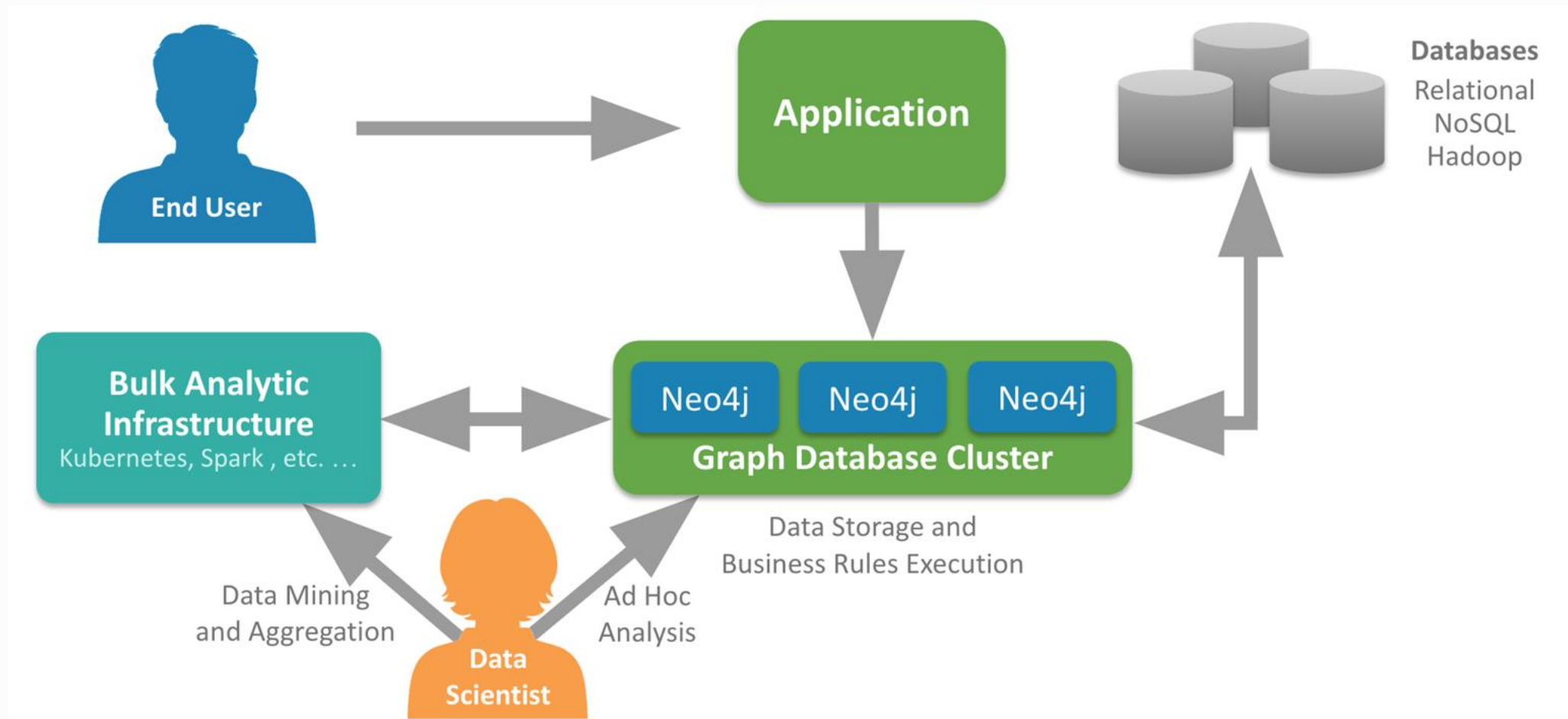
Ejemplo gráfico

Los componentes de un grafo Neo4j incluyen:

Nodos	3
Etiquetas/Labels	2
Propiedades	7
Relaciones	5
Tipo	4
Propiedades	1

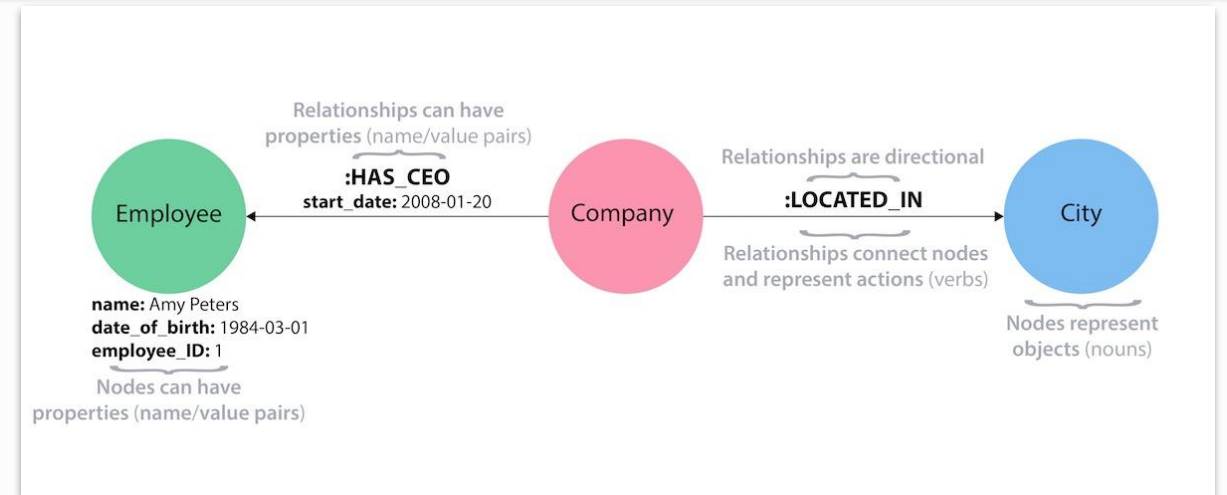
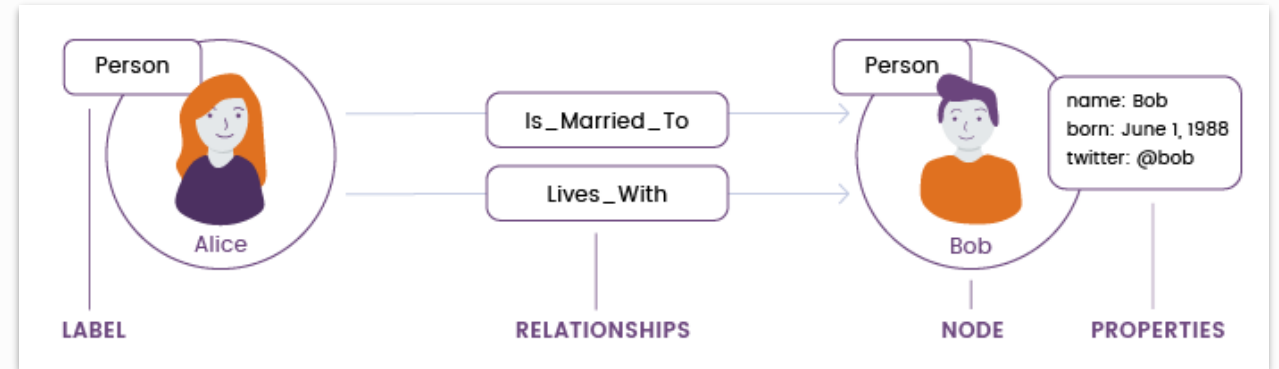


¿Qué es una base de datos de grafos?



Introducción

- Las bases de datos de grafos nos permiten almacenar **entidades y relaciones sobre esas entidades**.
- Las entidades las denominamos **Nodos**, las cuales poseen propiedades.
- Las relaciones denominadas **Arcos** pueden también tener propiedades.



Introducción

- Las bases de datos de grafos nos permiten almacenar entidades y relaciones sobre esas entidades.
- Las entidades las denominamos Nodos, las cuales poseen propiedades.
 - *Podemos hacer una analogía entre un Nodo y un objeto instanciado en una aplicación*
- Las relaciones denominadas Arcos pueden también tener propiedades.



DBlandIT

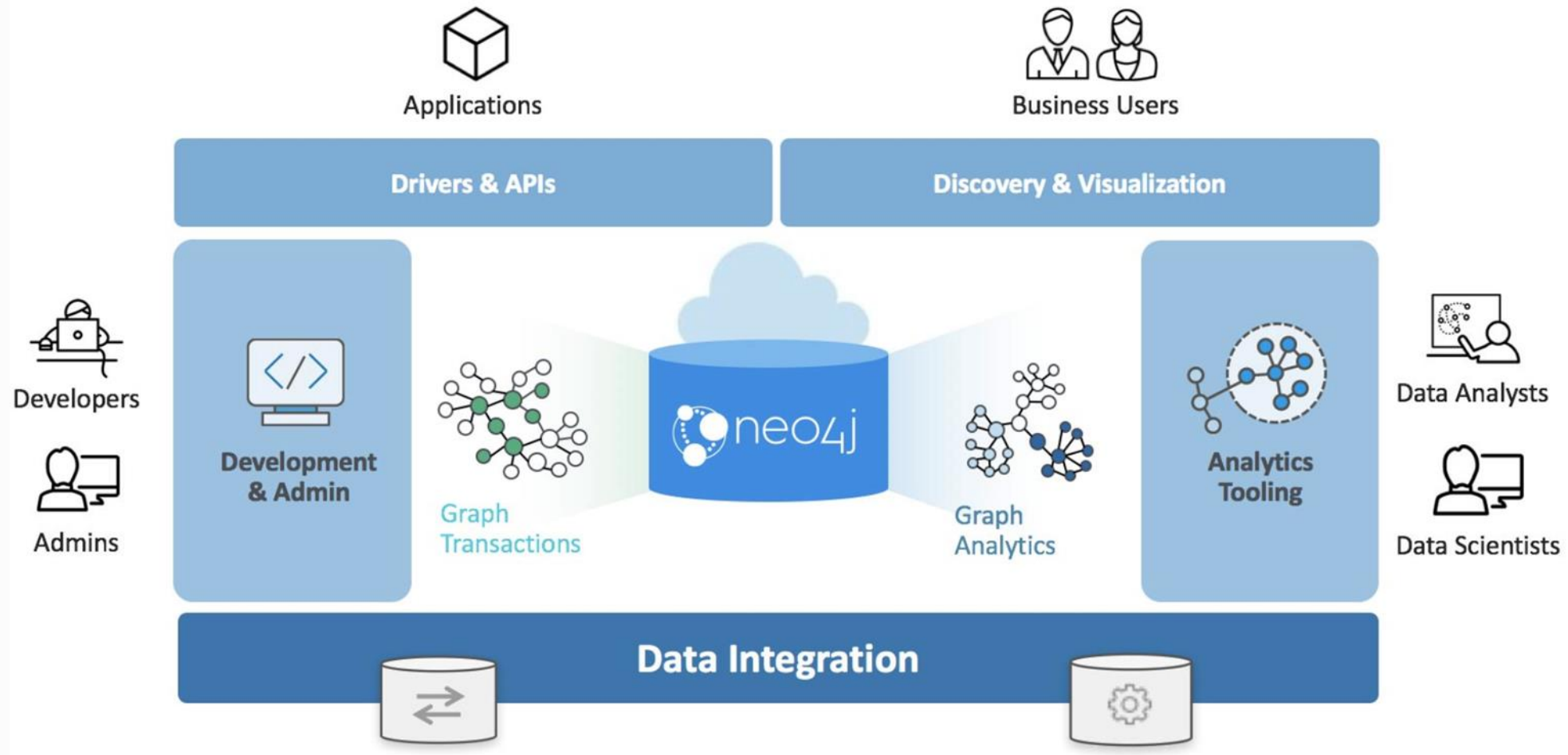


Introducción

- Las bases de datos de grafos nos permiten almacenar entidades y relaciones sobre esas entidades.
- Las entidades las denominamos Nodos, las cuales poseen propiedades.
- Las relaciones denominadas Arcos pueden también tener propiedades.
 - *Los arcos son direccionados en su creación : $()-[]\rightarrow()$*



Usos Neo4j



Introducción

- Los nodos están organizados por **relaciones**, las cuales nos permiten encontrar **patrones** entre estas, que con otro tipo de BD relacional, sería difícil de detectar.



El Consortio Internacional de Periodistas de Investigación (ICIJ - International Consortium of Investigative Journalists) utilizó Neo4j, para descubrir una de las **fugas financieras** más grande del mundo, exponiendo la **actividad de los paraísos fiscales** de muchos miembros de la élite global, y las **operaciones internas de entidades offshore**

<https://neo4j.com/news/neo4j-powers-panama-papers-investigation/>

<https://neo4j.com/blog/analyzing-panama-papers-neo4j/>

- La organización del Grafo permite que los datos sean almacenados una vez y luego **interpretados de distintas maneras** basándonos en sus relaciones.



DBlandIT



Ejemplos y Categorías de BD basadas en grafos



Ejemplos de BD basadas en grafos

El ranking **DB-Engines** clasifica los sistemas de gestión de bases de datos **según su popularidad**.
El ranking se actualiza mensualmente.

•Neo4J	53,51
•OrientDB	4,30
•JanusGraph	2,56
•Dgraph	1,78
•Giraph	1,55
•InfiniteGraph	0,53

Rank			DBMS	Database Model	Score		
Mar 2023	Feb 2023	Mar 2022			Mar 2023	Feb 2023	Mar 2022
1.	1.	1.	Neo4j +	Graph	53.51	-1.92	-6.16
2.	2.	2.	Microsoft Azure Cosmos DB +	Multi-model T	36.10	-0.40	-4.79
3.	3.	4.	Virtuoso +	Multi-model T	6.39	+0.29	+0.82
4.	4.	3.	ArangoDB +	Multi-model T	5.04	-0.26	-0.57
5.	5.	5.	OrientDB	Multi-model T	4.30	-0.24	-0.63
6.	7.	7.	Amazon Neptune	Multi-model T	2.60	-0.13	-0.09
7.	6.	8.	JanusGraph	Graph	2.56	-0.24	+0.09
8.	8.	6.	GraphDB +	Multi-model T	2.32	-0.12	-0.52
9.	9.	9.	TigerGraph	Graph	1.96	-0.21	-0.22
10.	12.	15.	NebulaGraph +	Graph	1.83	+0.04	+0.70
11.	13.	20.	Memgraph +	Graph	1.79	+0.10	+1.41
12.	11.	11.	Dgraph	Graph	1.78	-0.08	+0.09
13.	10.	12.	Fauna	Multi-model T	1.77	-0.12	+0.42
14.	14.	10.	Stardog +	Multi-model T	1.77	+0.14	-0.13
15.	15.	13.	Giraph	Graph	1.55	-0.04	+0.23
16.	16.	18.	TypeDB +	Multi-model T	1.23	-0.16	+0.53
17.	17.	14.	AllegroGraph +	Multi-model T	1.17	-0.16	-0.06

Fuente: www.db-engines.com
Marzo 2023

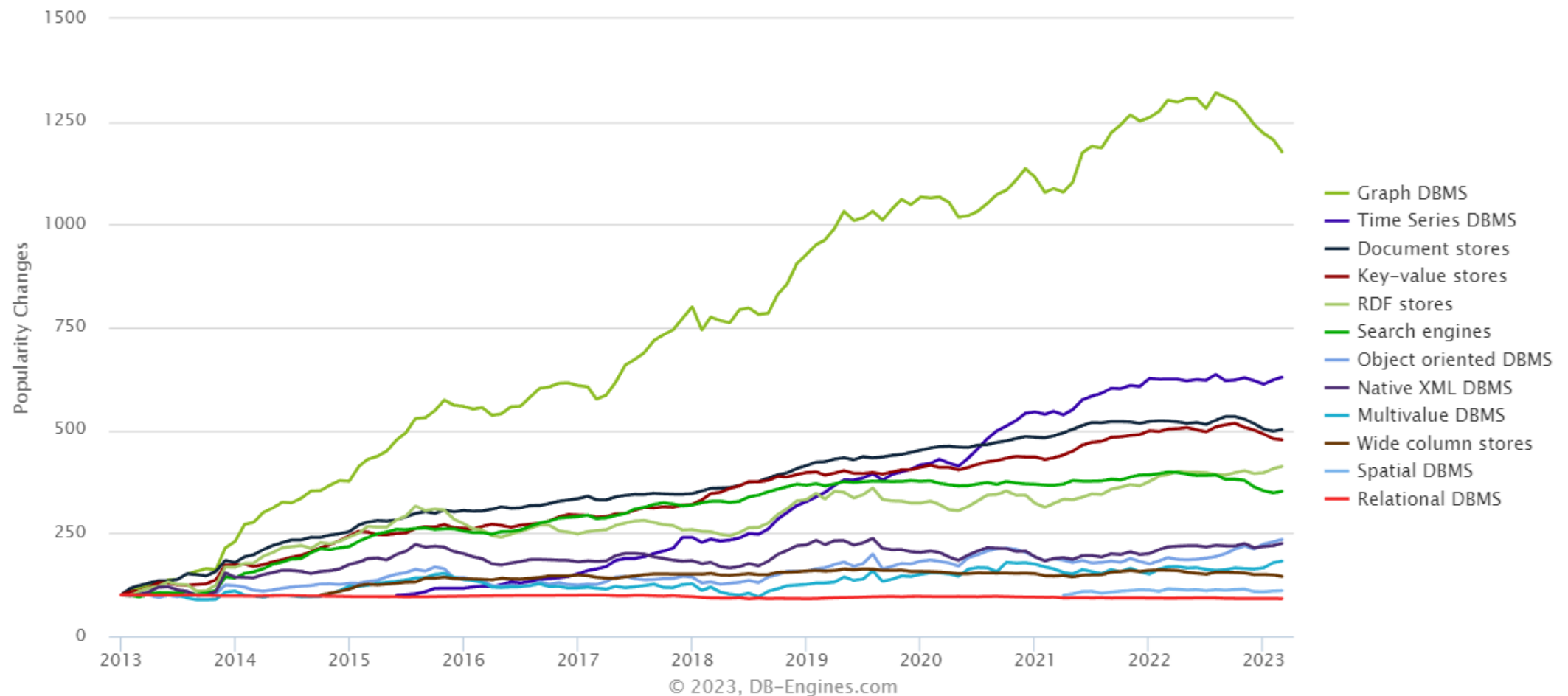
Fuente: <https://db-engines.com/en/ranking/graph+dbms>
Marzo 2023



DBlandIT



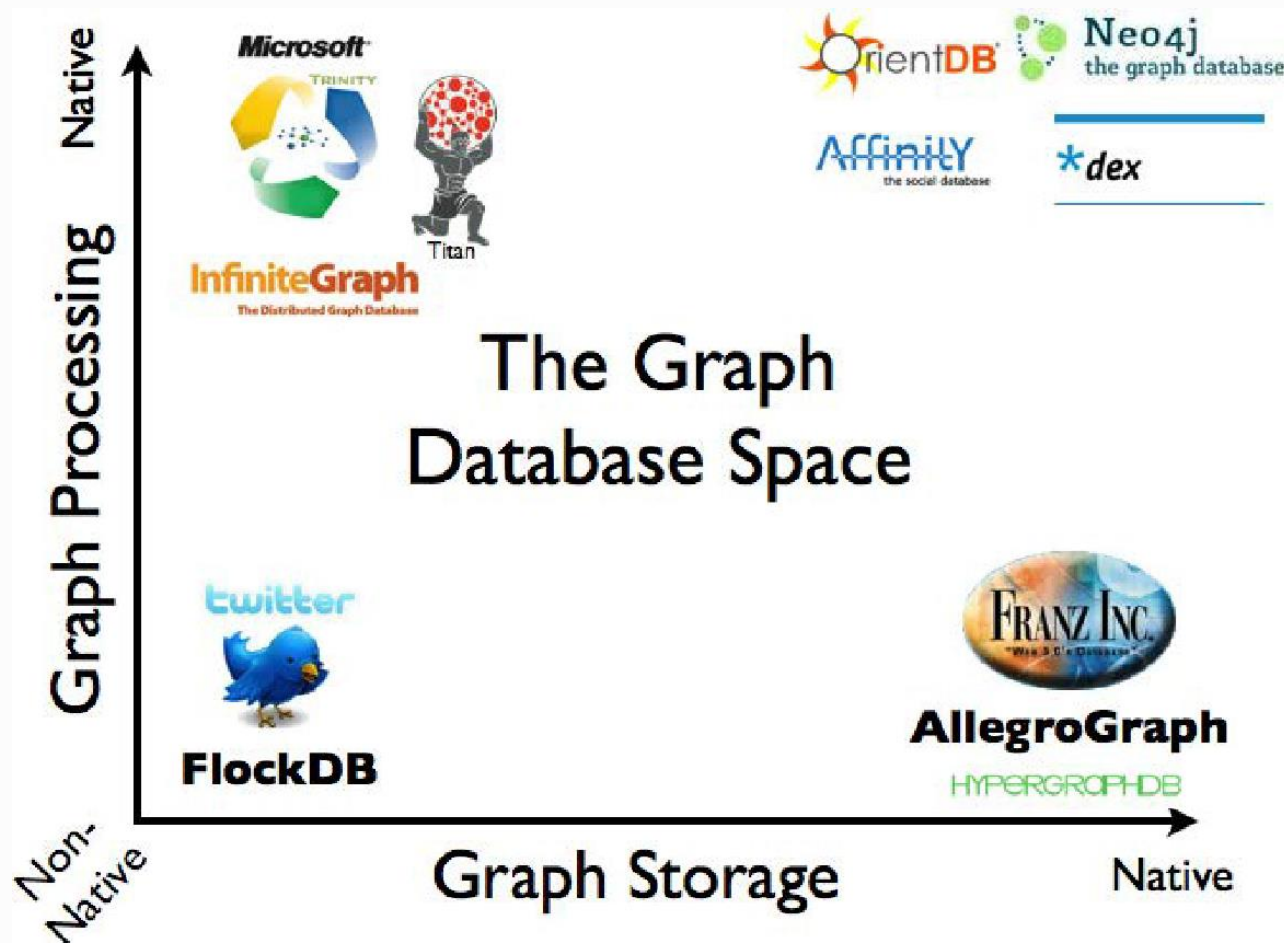
Categorías de BD basadas en grafos



DBlandIT



Categorías de BD basadas en grafos



Dos ejes:

- Almacenamiento interno del grafo
- Procesamiento del grafo



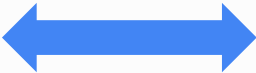

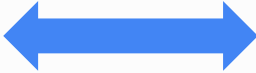

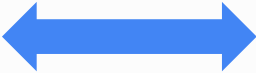

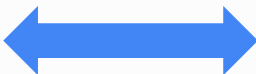

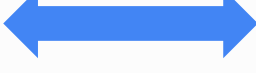

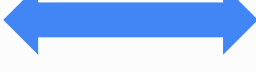
DBlandIT



Historia de Neo4J



Historia de Neo4J

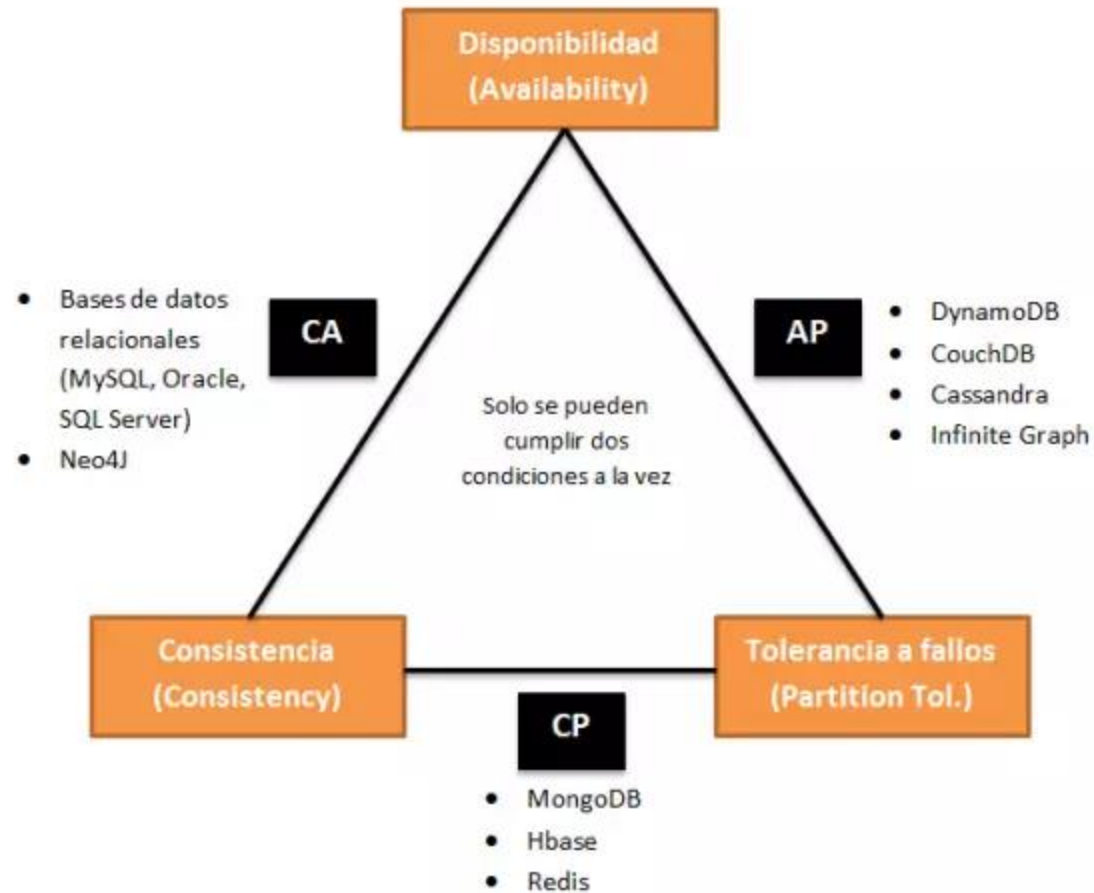
2000		Los fundadores de Neo encontraban problemas de performance usando RDBMS y empezaron a construir el primer prototipo de Neo4j.
2002		Se desarrolló la primera versión de Neo4j.
2003		Primer despliegue a producción 7x24
2007		Se formó detrás de Neo4j una compañía con sede en Suecia.
2009		Continuó desarrollándose con la inversión de los fondos Sunstone y Conor.
2010		Se lanzó la versión 1.0.
2011		Movió su sede a Silicon Valley.
2013		Se lanzó la versión 2.0.
2016		Se lanzó la versión 3.0.
2019		Se lanzó la versión 4.0.
2023		Última versión 5.5.0 (16/02/2023)



DBlandIT



Teorema CAP ¿En dónde se ubica Neo4J?



Forma de consulta de datos



Forma de consulta de datos

- Las Bases de datos de Grafos soportan acceso a sus datos a través de lenguajes de consulta tales como **Gremlin** (lenguaje específico para recorrer grafos).
- Gremlin puede ser utilizado en todas las bases orientadas a Grafos que implementen "Blueprints property graph" (*).
- Neo4J también cuenta con el lenguaje de consulta **Cypher** para recorrer grafos.
 - *Cypher usa las keywords **MATCH** para búsqueda de patrones en relaciones, **WHERE** para filtrar propiedades en un nodo o relación, y **RETURN** para especificar qué retorna una consulta.*
 - *Cypher también provee métodos para operar los datos tales como **ORDER, AGGREGATE, SKIP, LIMIT**, entre otros.*

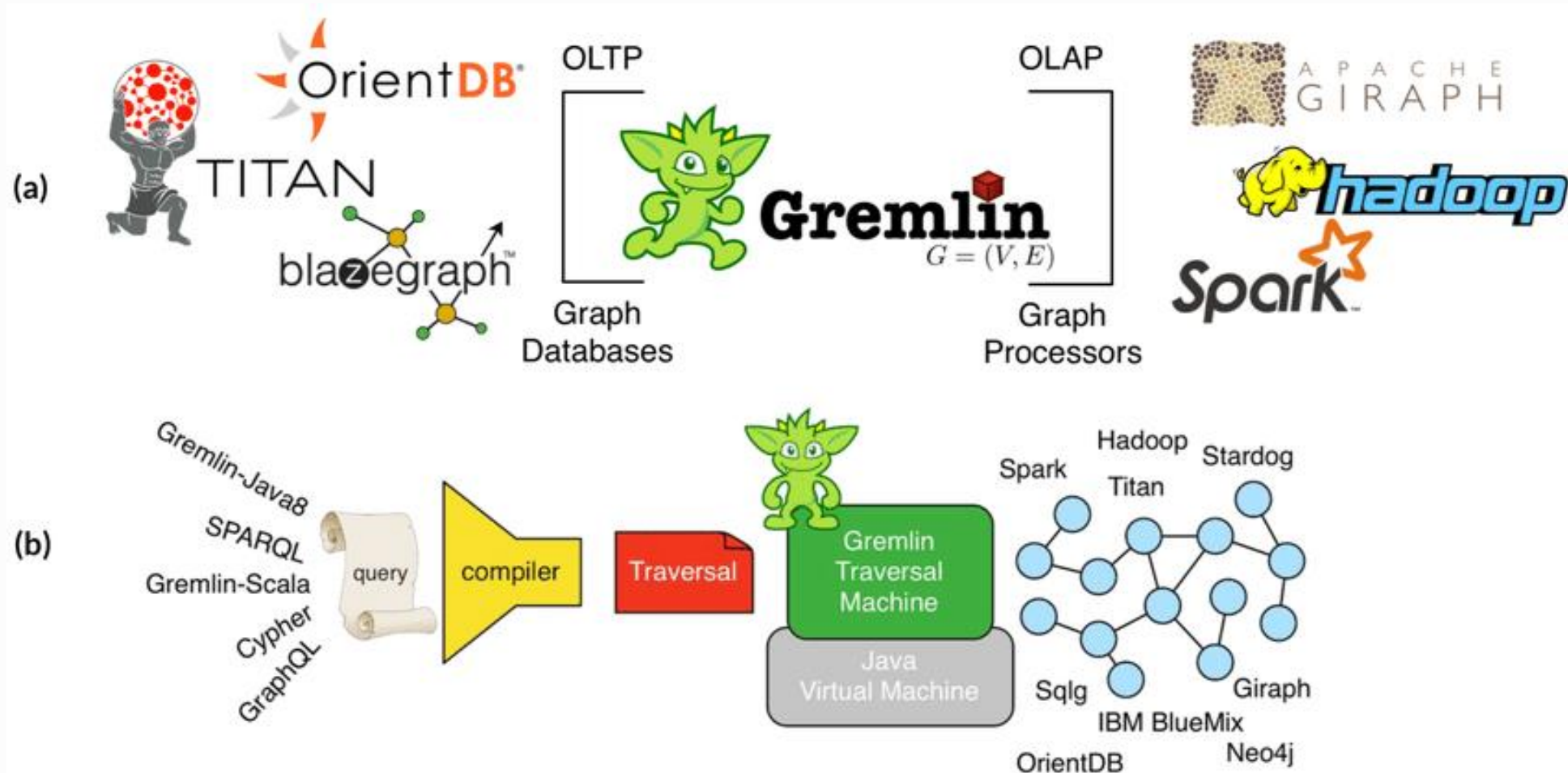
(*) Blueprints es análogo a JDBC para las bases de datos orientadas a grafos.



DBlandIT



Forma de consulta de datos



Forma de consulta de datos

- Aparte de estos lenguajes de consulta, Neo4J nos permite **consultar las propiedades** de los nodos, **recorrer los grafos, o navegar por las relaciones** de nodos utilizando "bindings" con diferentes lenguajes de programación (Java/.NET/JavaScript/Ruby/Python/PHP/R/Perl...).
- Índices. Se pueden crear índices sobre propiedades de un nodo.
- Se pueden aplicar filtros direccionales (entrantes, salientes o ambos).



DBlandIT



**En qué casos usarlas y
en qué casos no**



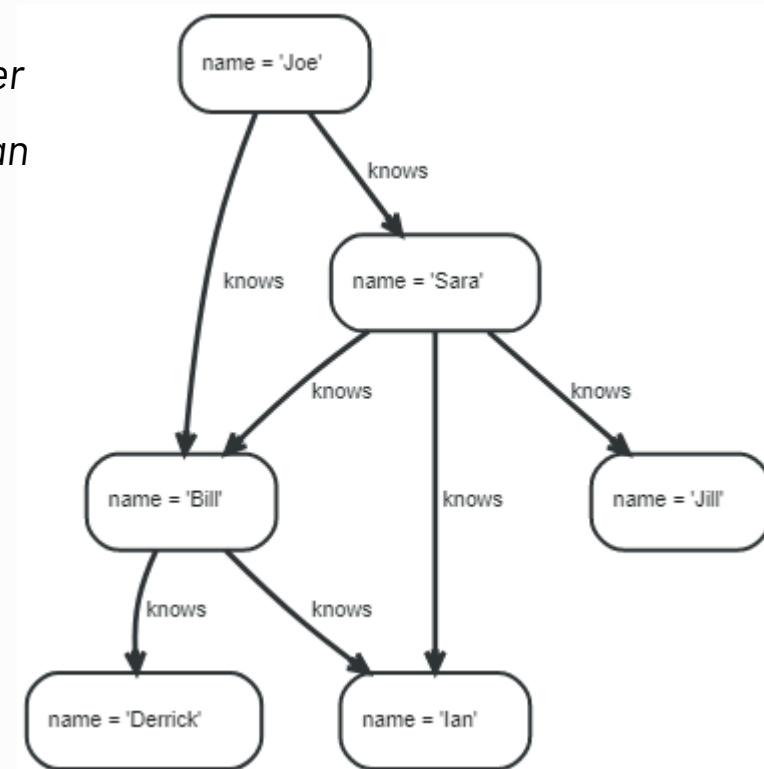
¿En qué casos usarlas?

- Datos interconectados:
 - **Redes sociales** (likes, amigos, seguidores, etc.) ó laborales, por ej. representar a los empleados, sus conocimientos, y la relación de trabajo con otros empleados en diferentes proyectos
 - **Relaciones entre entidades de dominio de diferentes dominios** (por ejemplo, sociales, espaciales, comercio, redes y operaciones de IT, identidad y gestión de accesos) en una sola base de datos, puede hacer estas relaciones más valiosas, proporcionando la capacidad de recorrer a través de dominios.
- Servicios de Ruteo, Despacho
 - Cada ubicación en nuestra red de despacho es un nodo, las relaciones pueden contener la distancia entre las ubicaciones como propiedad.



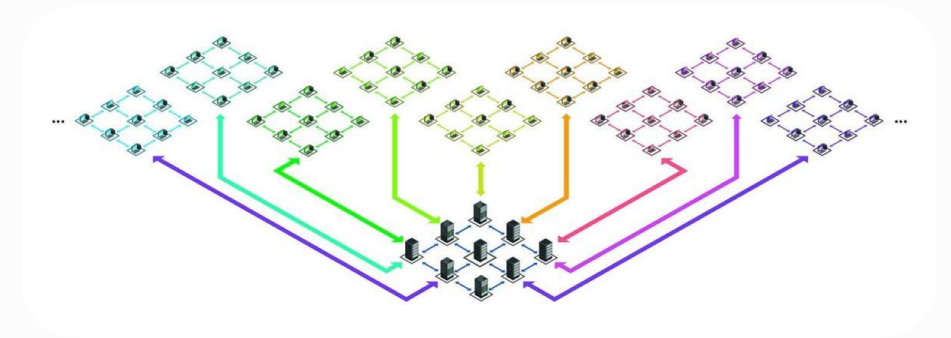
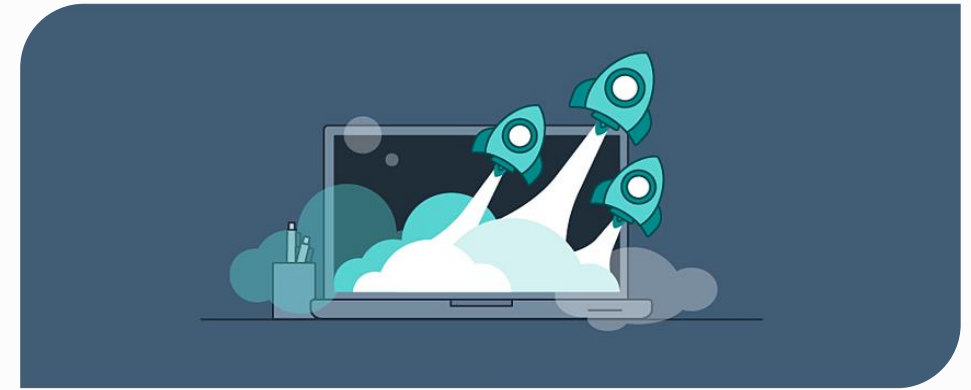
¿En qué casos usarlas?

- Motores de Recomendaciones:
 - *Como los nodos y las relaciones se crean en el sistema, que pueden ser utilizados para hacer recomendaciones como "sus amigos también han comprado este producto" o "al facturar este artículo, estos otros artículos suelen ser facturados" en tiempo real.*
- Sistemas con búsquedas recursivas con n niveles.
- Búsqueda de patrones en las relaciones para detectar el fraude en las transacciones en tiempo real.
- Gestión de datos maestros: líneas de organización y producción que naturalmente se modelan como grafos.



¿En qué casos NO usarlas?

- Sistemas que requieren de actualizaciones masivas sobre todas las entidades o un conjunto de entidades para un atributo o conjunto de atributos específicos.
- Sistemas que requieren una alta distribución de datos debido a su gran tamaño.



Algunos casos de éxitos



Algunos casos de éxito de Neo4j



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



DBlandIT

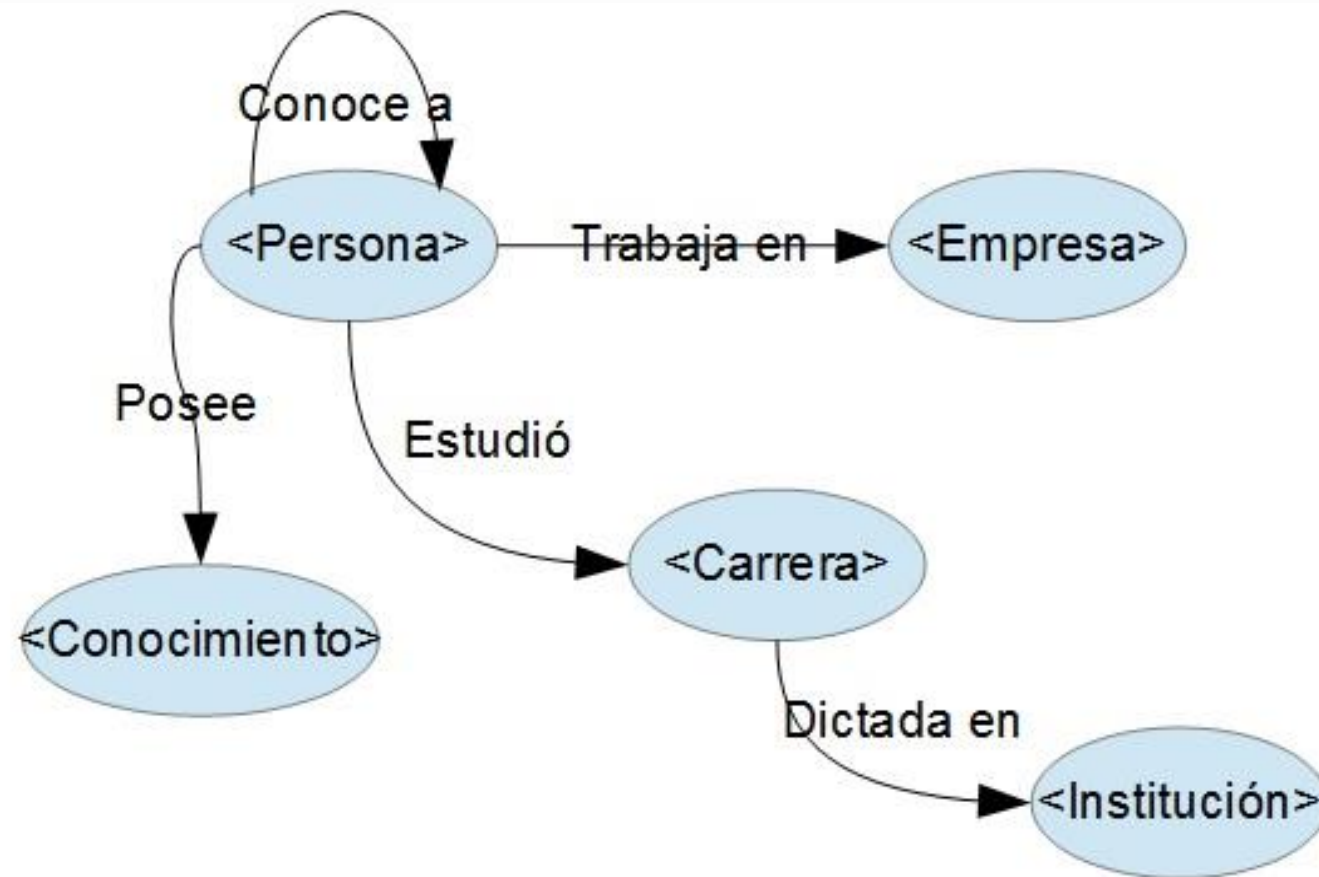


Caso Práctico: Red social profesional



Caso Práctico: Red social profesional

Dominio del negocio

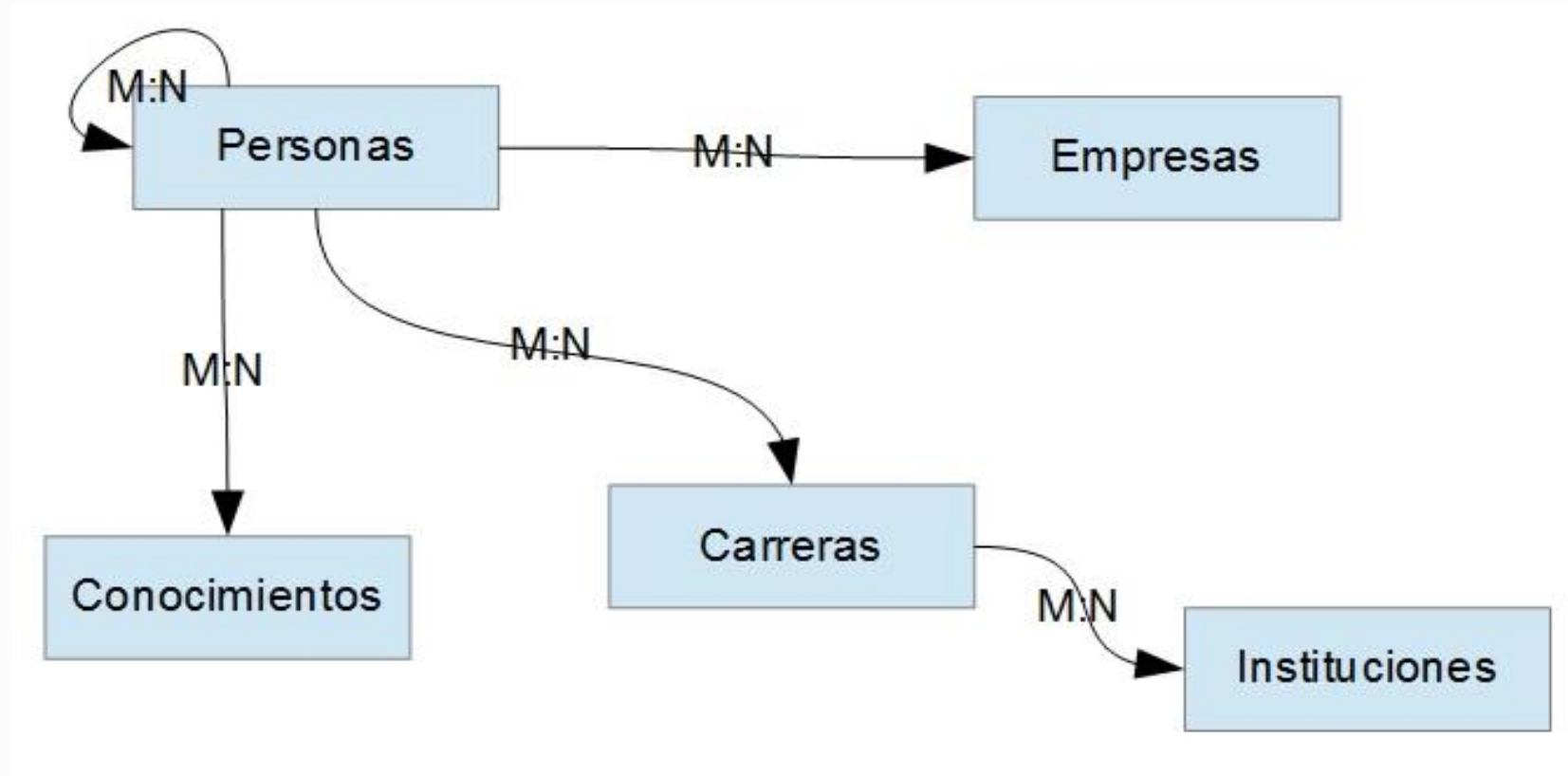


DBlandIT



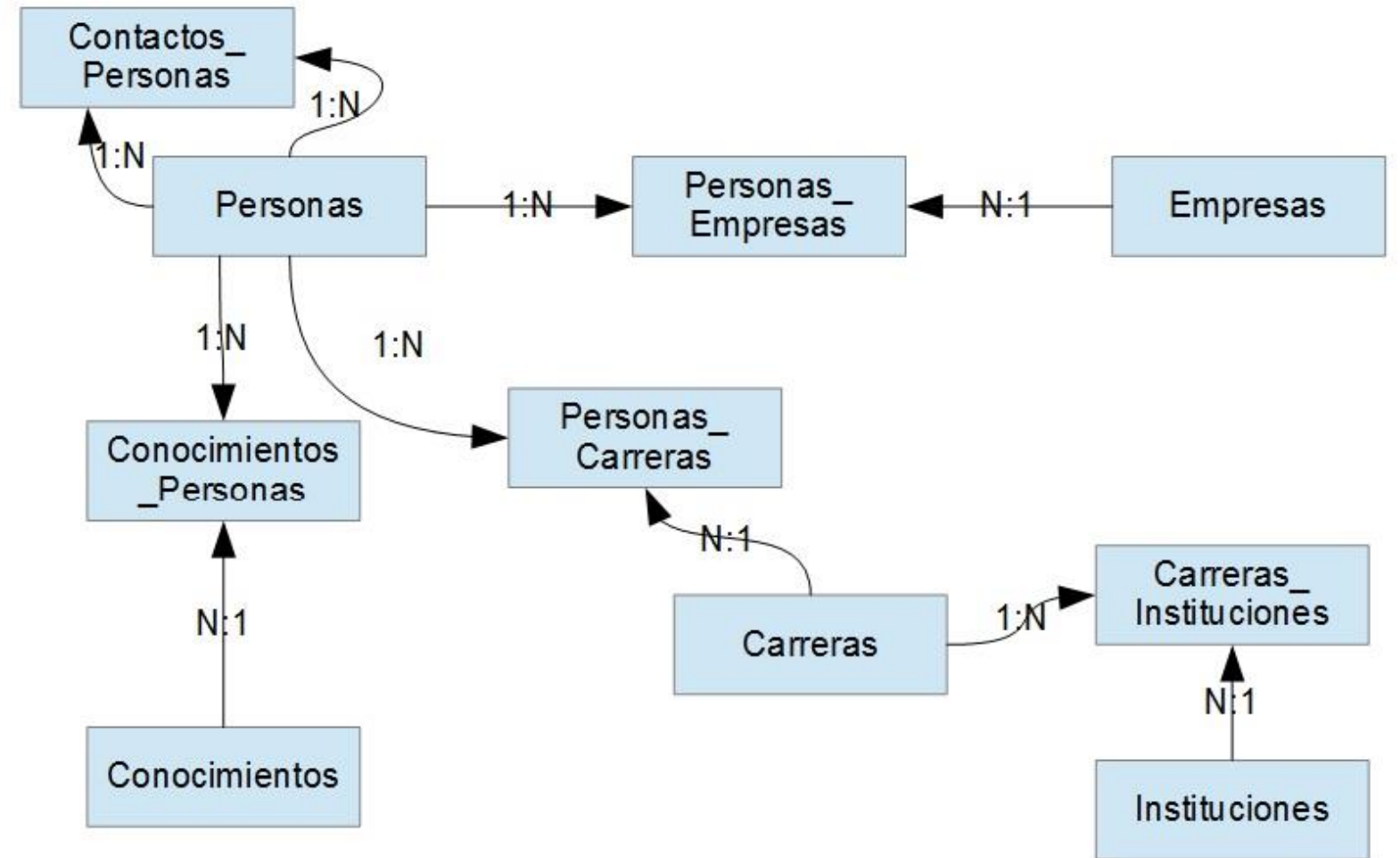
Caso Práctico: Red social profesional

Diseño Lógico Relacional



Caso Práctico: Red social profesional

Diseño Físico Relacional



Caso Práctico: Red social profesional

Algunas consultas posibles

- Lista de personas que **trabajan o trabajaron** en empresas en las que una persona determinada trabajó, pero que no son sus contactos, para sugerirle nuevos contactos.
- Lista de personas que **estudiaron o estudian** en la misma institución que una persona determinada, pero que no son sus contactos, para sugerirle nuevos contactos.
- Ranking de los primeros 2 conocimientos que poseen más personas egresadas de la carrera "Ing en Sistemas de Información".
- Ranking de las 3 personas más populares: las que son conocidas por más personas en la red.

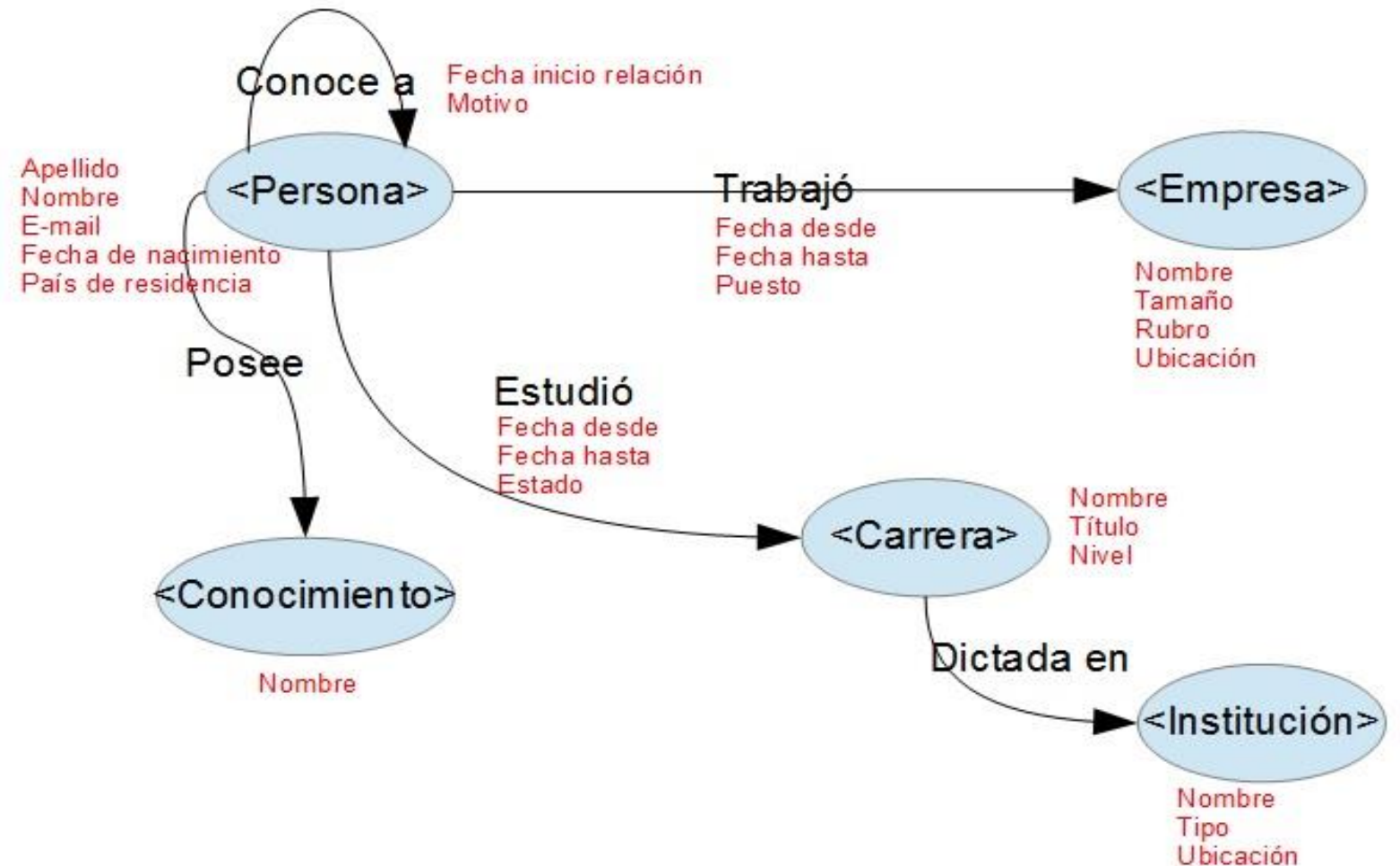


DBlandIT



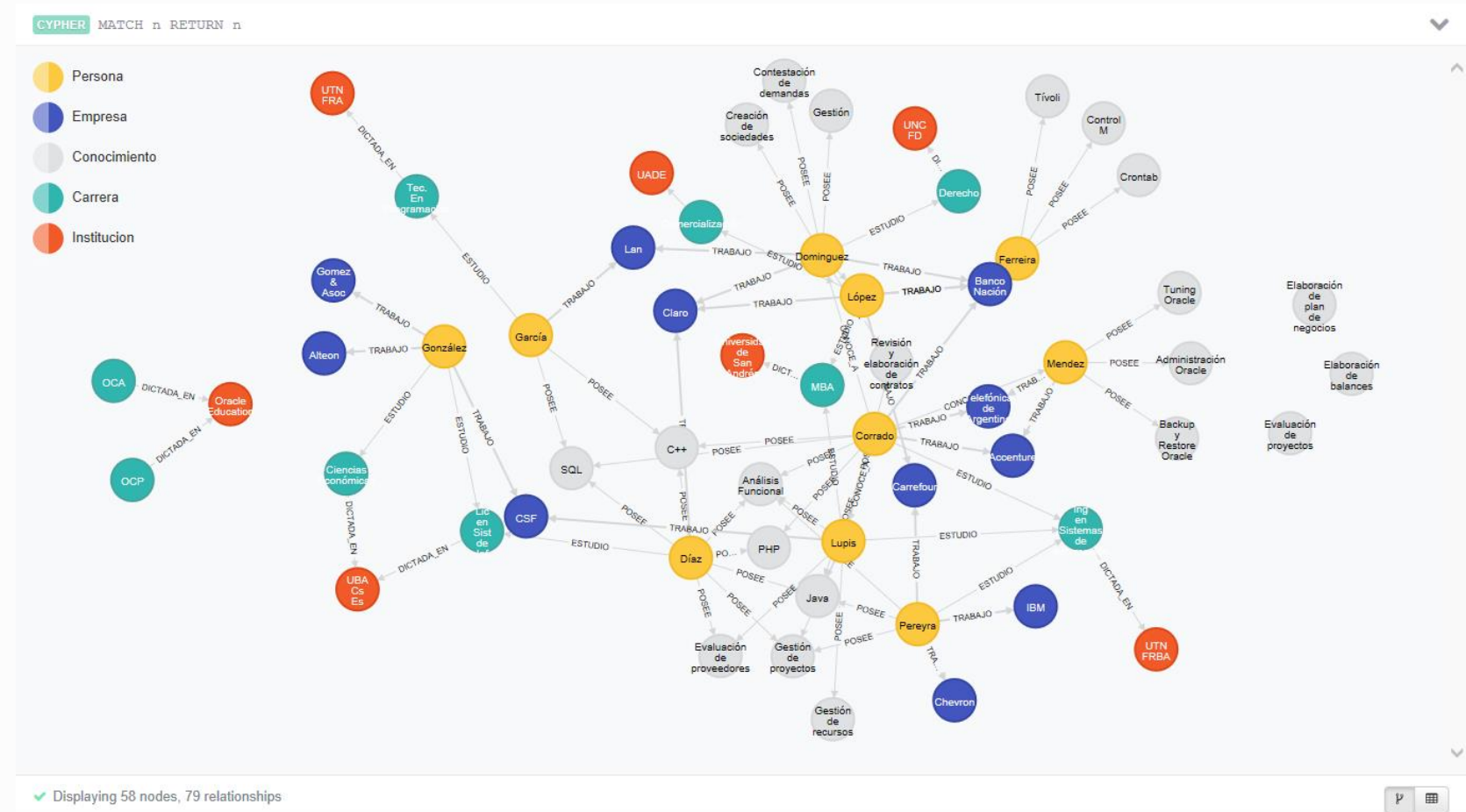
Caso Práctico: Red social profesional

Grafo de Propiedades



Caso Práctico: Red social profesional

Grafo con datos



Lenguaje Cypher



Lenguaje Cypher

- Lenguaje declarativo: enfoque en el dominio y no en la BD (qué recuperar y no cómo).
- Cláusulas para lectura y para escritura, que se encadenan entre sí.
- Actualización del grafo dentro de una transacción.



DBlandIT



Nodos, relaciones y sus tipos de datos



Nodos

- Unidad fundamental de almacenamiento de información.

`(identificador)`

`(identificador: etiqueta)`

`(identificador: etiqueta { propiedades })`

Ej.:

`(nodo1: Persona { apellido:"Corrado", nombre:"Gustavo" })`



DBlandIT



Relaciones

- Son claves para identificar la información relacionada en el grafo.
- Tienen dirección y tipo.

`(nodo1) --> (nodo2)`

`(nodo1) - [identificador] -> (nodo2)`

`(nodo1) - [identificador:TIPO] -> (nodo2)`

`(nodo1) - [identificador:TIPO { propiedades }] -> (nodo2)`

`(nodo1) -- (nodo2)`

Ej.: `(Gonzalez)-[:ESTUDIO { estado: "Completo" }]->(CsEs)`



Tipos de datos

Property types:

- Numérico, como un tipo abstracto, que tiene los subtipos Integer y Float
- String
- Boolean
- El tipo espacial Point
- Tipos temporales: Date, Time, LocalTime, DateTime, LocalDateTime and Duration



Tipos de datos

Structural types

- Nodos :
 - Id
 - Label(s)
 - Map(of properties)
- Relaciones :
 - Id
 - Type
 - Map(of properties)
 - Id of the start and end nodes
- Caminos :
 - An alternating sequence of nodes and relationships



**Crear, consultar,
modificar y eliminar
nodos/relaciones**



Creación de un Nodo

Creación de un Nodo

Crea un nodo con las propiedades dadas.

```
CREATE (n:Etiqueta {Atributo: {valor}})
```

Ejemplos

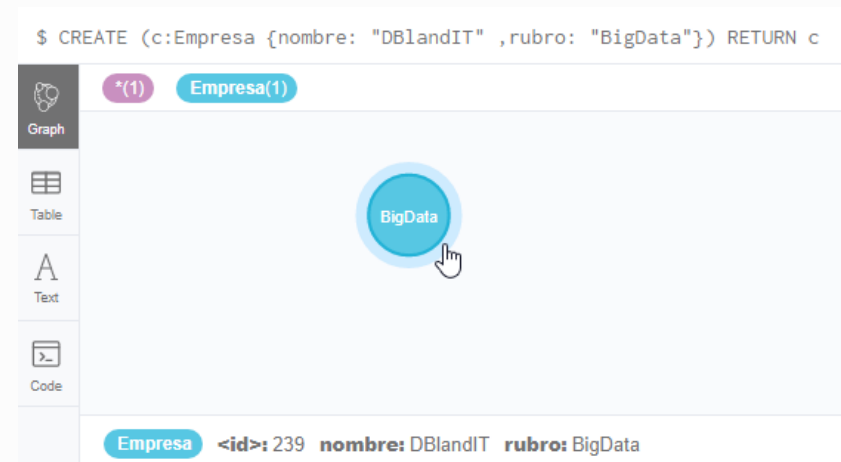
Crear un nodo para la persona Analía Martinelli.

```
CREATE (n:Persona {nombre: "Analía",  
apellido: "Martinelli"})  
RETURN n
```



Crear un nodo para la empresa DBlandIT.

```
CREATE (c:Empresa {nombre: "DBlandIT",  
rubro: "BigData"})  
RETURN c
```



Consultar Nodos

Consultar un Nodo

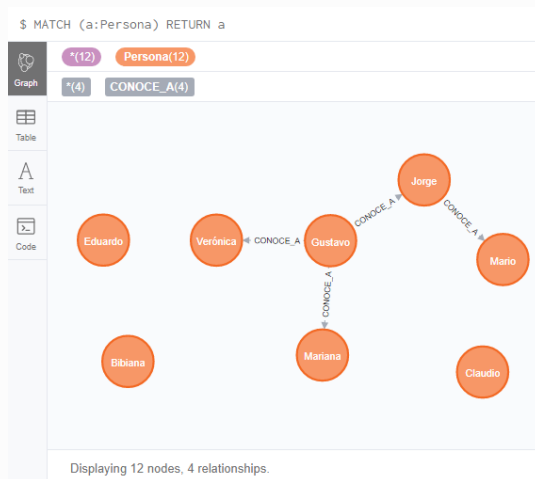
Consultar un nodo con las propiedades dadas.

```
MATCH (n:Etiqueta { propiedades } )  
WHERE condición
```

Ejemplos

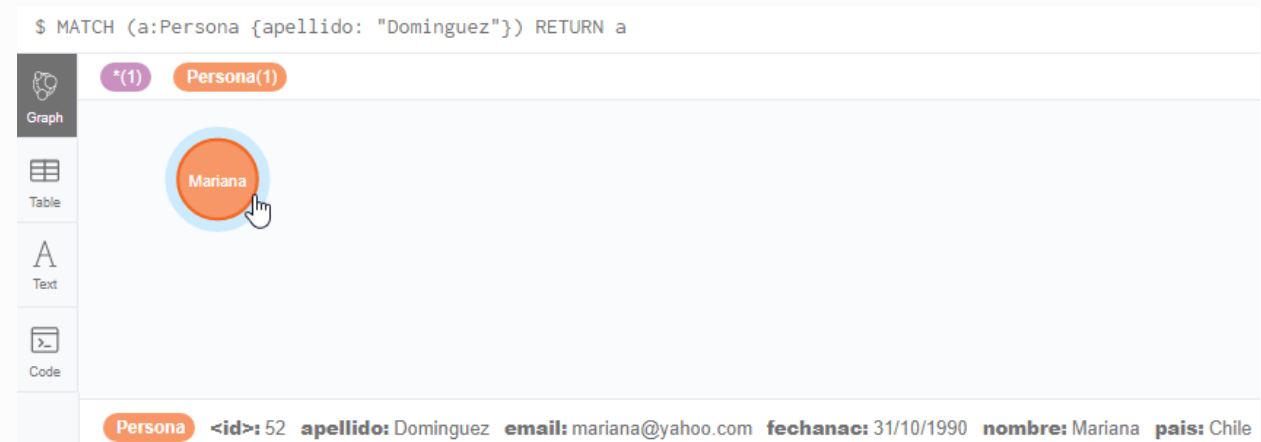
Obtener los nodos de todas las personas de la red.

```
MATCH (a:Persona) RETURN a
```



Obtener el nodo correspondiente a la persona de apellido Domínguez.

```
MATCH (a:Persona {apellido: "Dominguez"})  
RETURN a
```



Consultar Nodos

Otros Ejemplos

Obtener los nodos de todas las personas de la red informando su apellido y nombre.

```
MATCH (a:Persona)  
RETURN a.apellido, a.nombre
```

Contar la cantidad de Personas de la Base de Datos.

```
MATCH (a:Persona)  
RETURN count(*)
```

Obtener un listado de personas ordenado alfabéticamente por apellido, mostrando las primeras cinco.

```
MATCH (a:Persona)  
RETURN a.apellido, a.nombre  
ORDER BY a.apellido  
LIMIT 5
```



Creación de una Relación

Creación de una Relación

Crea una relación con el tipo y dirección dados, y le enlaza un identificador.

```
CREATE (n) - [r:TIPO] -> (m)
```

Crea una relación con el tipo, dirección y propiedades dados.

```
CREATE (n) - [:TIPO {propiedad: {valor}}] -> (m)
```

Ejemplos

Para crear una relación entre dos nodos, primero obtenemos los dos nodos. Una vez que están cargados en variables, creamos una relación entre ellos.

Crear para Analía Martinelli una relación Estudio con la carrera "Lic en Sist de Inf", con estado "En curso".

```
MATCH (n:Persona { nombre: "Analía", apellido: "Martinelli" }),  
      (c:Carrera { nombre: "Ingeniería en Sistemas de Información"})  
CREATE (n) - [r:ESTUDIO { estado: "En curso" }] -> (c)  
RETURN n, r, c
```



Creación de una Relación

Otros Ejemplos

Analía Martinelli conoce a Verónica Mendez. Crear la relación asegurando que se cree sólo una sola vez, o sea que si ya existe no crea la relación.

```
MATCH (a:Persona {apellido: "Martinelli"}),  
      (b:Persona {apellido: "Mendez"})  
CREATE UNIQUE (a)-[r:CONOCE_A]->(b)  
RETURN r
```

Crear la relación de trabajo entre Analía Martinelli y DBlandIT en el puesto de Data Scientist

```
MATCH (a:Persona {apellido: "Martinelli"}),  
      (b:Empresa {nombre: "DBlandIT"})  
CREATE (a)-[r:TRABAJO {puesto: "Data Scientist"}]->(b)  
RETURN a, r, b
```



Consultar Relaciones y Nodos

Consultar Relaciones y Nodos

Consultar una relación.

```
MATCH (n:Etiqueta { propiedades } )-[r:Relacion {propiedades}]->(m)
WHERE condición
```

Ejemplos

Obtener la lista de empresas en las que trabajó Domínguez.

```
MATCH (a:Persona {apellido:"Dominguez"})-[r:TRABAJO]->(b:Empresa)
RETURN b
```



DBlandIT



Eliminar Relaciones y/o Nodos

Eliminar Relaciones y Nodos

Consultar una relación.

```
MATCH (n:Etiqueta { propiedades } )-[r:Relacion {propiedades}]-()  
DELETE n,r
```

Para eliminar un nodo, el mismo no puede tener relaciones existentes. Por lo que habría que eliminar ambos.

Ejemplos

Eliminar las relaciones de TRABAJO para la empresa DBlandIT.

```
MATCH ()-[r:TRABAJO]->(b:Empresa {nombre:"DBlandIT"})  
DELETE r
```



DBlandIT



Eliminar Relaciones y/o Nodos

Ejemplos

Eliminar las relaciones de TRABAJO para la empresa DBlandIT y el nodo de dicha empresa.

```
MATCH () - [r:TRABAJO] -> (b:Empresa {nombre:"DBlandIT"})  
DELETE r,b
```

Eliminar la persona con apellido Martinelli y todas sus relaciones entrantes y salientes.

```
MATCH (p:Persona {apellido:"Martinelli"}) - [r] - ()  
DELETE p,r
```



Eliminar Relaciones y/o Nodos

Modificar Relaciones y Nodos

Consultar una relación.

```
MATCH (n:Etiqueta { propiedades } )-[r:Relacion {propiedades}]-()  
SET n:nuevaEtiqueta,n.propiedad:valor  
RETURN n
```

Para eliminar un nodo, el mismo no puede tener relaciones existentes. Por lo que habría que eliminar ambos.

Ejemplos

Agregarle a Analía Martinelli la etiqueta "Empleado" y el país Argentina.

```
MATCH (a:Persona {apellido: "Martinelli"})  
SET a:Empleado, a.pais="Argentina"  
RETURN a
```



DBlandIT



Eliminar Relaciones y/o Nodos

Otros Ejemplos

Eliminar de la persona con apellido Martinelli la propiedad país.

```
MATCH (a:Persona {apellido: "Martinelli"})  
REMOVE a.pais  
RETURN a
```

Modificar la relación TRABAJO de la Persona Martinelli a la empresa DBlandIT, agregándole la fecha de inicio.

```
MATCH (a:Persona {apellido: "Martinelli"})-[r:TRABAJO]-(e:Empresa  
{nombre:"DBlandIT"})  
SET r += {fechaInicio:"2015-04-01",sueldo:55000}  
RETURN r
```



DBlandIT



Consultas con Agregados

Agregados

Sumar los sueldos de todos los TRABAJOS de todas las personas.

```
match (a) - [r:TRABAJO] -> () return sum(r.sueldo)
```

Obtener la cantidad de empleados, el promedio de sueldos pagados, el sueldo maximo y el sueldo minimo pagado.

```
MATCH (a) - [r:TRABAJO] -> ()  
RETURN AVG(r.sueldo), MAX(r.sueldo), MIN(r.sueldo)
```



Modelando en Bases de datos Basadas en Grafos



Modelado de Datos

¿Por qué modelamos?

- Respaldar parte o la totalidad de una aplicación.
- Responder a los casos de uso claves de la aplicación.
- Brindar el mejor rendimiento de consulta para los casos de uso claves de la aplicación.



DBlandIT



Modelado de Datos

Proceso de Modelado de datos a nivel Macro:

1. Comprender el dominio y definir casos de uso específicos para la aplicación.
2. Desarrollar el modelo de datos del grafo inicial incluyendo el modelado de los nodos y las relaciones entre los nodos.
3. Probar los casos de uso contra el modelo de datos inicial, determinando si las etiquetas, los tipos de relación y las propiedades se tienen en cuenta en los casos de uso de la aplicación.
4. Crear el grafo con datos de prueba usando Cypher.
5. Probar los casos de uso incluyendo pruebas de performance.
6. Refactorizar o mejorar el modelo de datos, ya sea por un cambio en los casos de uso principales o por motivos de performance.
7. Implementar la refactorización del grafo y volver a testear.



Modelado de Datos

¿Cómo comprender el dominio de nuestra aplicación?

1. Identificar a las partes interesadas (stakeholders)
2. Juntos con los stakeholders y los desarrolladores debemos:
 - Escribir una descripción detallada de lo que esperamos que realice la aplicación y su alcance.
 - Identificar todos los usuarios que lo van a utilizar (Tanto personas como sistemas)
3. Acordar los casos de uso de la aplicación.
4. Rankear los casos de uso por nivel de importancia.



Modelado de Datos

Data Model - Modelo de Datos



- Describe las relaciones de las etiquetas y las propiedades del grafo
- No tiene datos específicos que se crearán en el grafo



Modelado de Datos

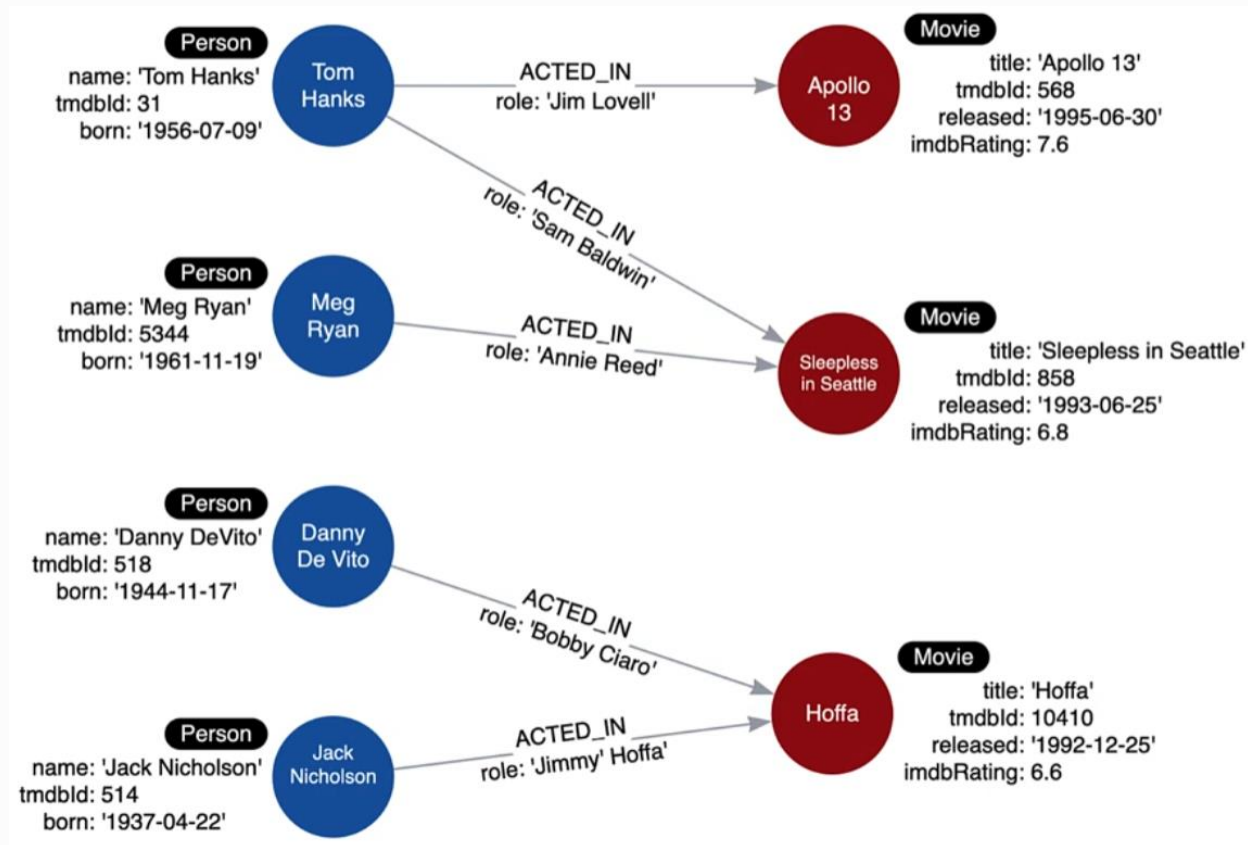
Convenciones:

- ❑ La **etiqueta** comienza con una letra mayúscula y puede seguir con camelCase. Es decir, en conjunto queda como PascalCase
 - ❑ Ejemplos: Persona, Compañía, RepoDeGit
- ❑ La **relación** está en mayúsculas separadas con guión bajo
 - ❑ Ejemplos: CASADO_CON, SIGUE_A
- ❑ La **propiedad** para un nodo o relación comienza con una letra minúscula y puede seguir con camelCase
 - ❑ Ejemplos: deptold,telefonoPrincipal,telefonoSecundario



Modelado de Datos

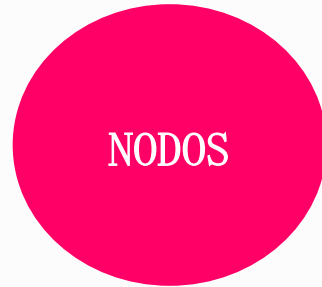
Instance Model - Modelo instanciado



Modelado de Datos

- Las unidades fundamentales que forman un grafo son los nodos y las relaciones.

Nodos

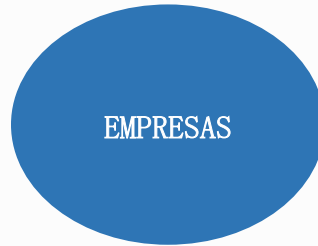


- Es la primera entidad que se debe identificar en su dominio.
- Son usados para representar entidades, y tipos de valores complejos para nuestro modelo.
- Pueden contener propiedades (pares clave valor)
- Los nodos pueden tener propiedades distintas.



Modelado de Datos

Etiquetas



- Luego de detectar nodos se deberá decidir qué Etiqueta asignó a esos nodos.
- Una etiqueta es un nombre que es usado para agrupar nodos en conjuntos. Todos los nodos etiquetados con la misma etiqueta pertenecen al mismo conjunto.
- Un nodo podría ser etiquetado por cualquier cantidad de etiquetas, inclusive por ninguno.



Modelado de Datos

Relaciones



- Identifican interacciones entre nodos. Los nodos están conectados por relaciones.
- Cada relación puede tener un nombre y una dirección.
- Le dan la estructura al grafo y proveen un contexto semántico a los nodos.
- Pueden contener propiedades, representan una cualidad, peso o metadata de las relaciones.
- Cada Relación tiene que tener obligatoriamente un Nodo Desde y un Nodo Hasta.



Modelado de Datos

En resumen

Nodos: Representan las entidades de nuestro modelo.

Etiquetas: Agrupan Nodos por Roles.

Relaciones: Conectan entidades, le dan estructura al dominio.

Propiedades: Son atributos de las entidades, cualidades de las relaciones y/o metadata.



DBlandIT



Modelado de Datos

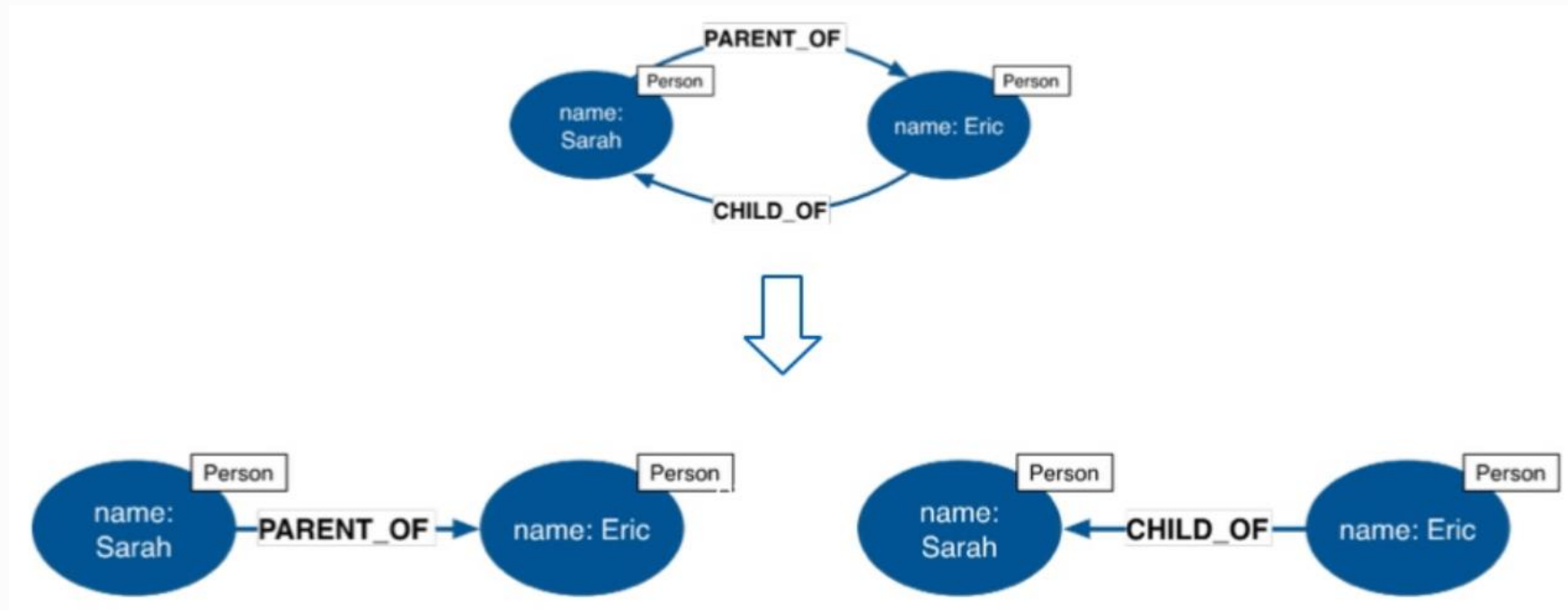
Modelado Incremental:

1. Identificamos entidades
2. Definimos nodos con sus correspondientes etiquetas.
3. Conectamos los nodos con relaciones con un nombre.
4. Asignamos propiedades (atributos) a los nodos.
5. Asignamos propiedades (cualificadores, valores, metadata) a las relaciones.
6. Continuamos identificando nuevas entidades y relaciones.



Modelado de Datos

Relaciones Simétricas



Se resuelve poniendo sólo una de las dos relaciones.
En las consultas se infiere la relación simétrica.



Modelado de Datos

Relaciones Simétricas



Find child:

```
MATCH (parent{name:'Sarah'})  
  -[:PARENT_OF]->(child)  
RETURN child
```

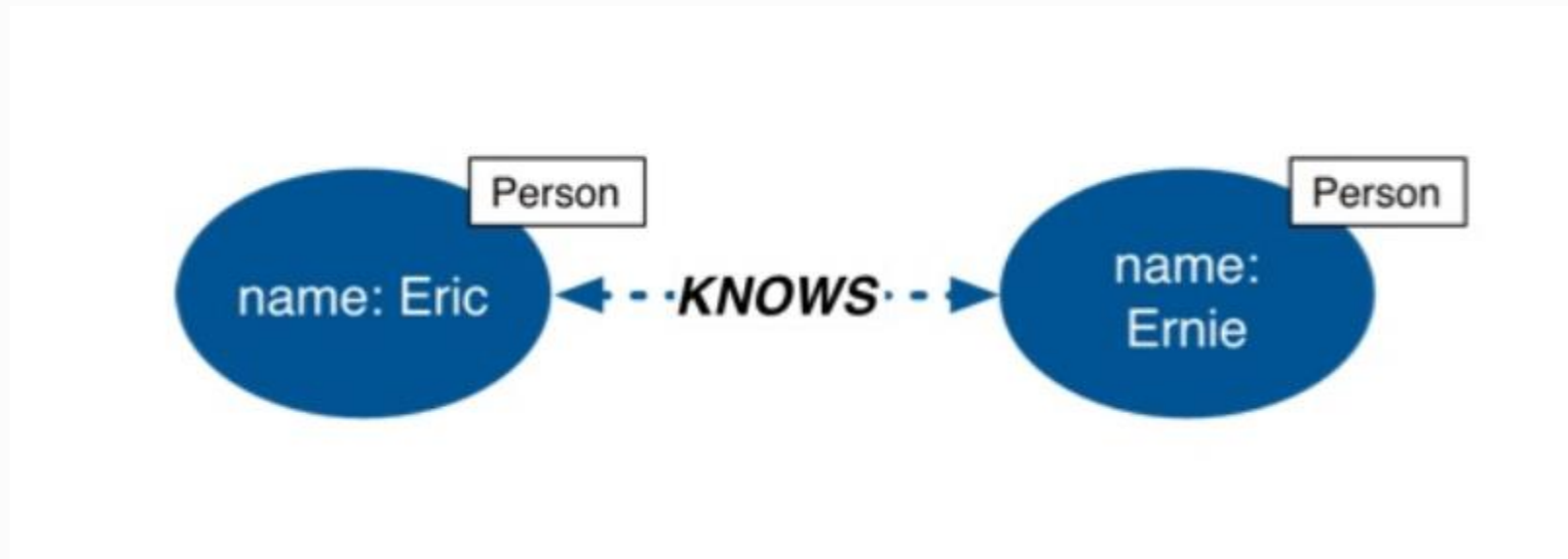
Find parent:

```
MATCH (parent)-[:PARENT_OF]->  
  (child{name:'Eric'})  
RETURN parent
```



Modelado de Datos

Relaciones Bi-direccionales



Se resuelve usando **relaciones simples e ignorando la dirección** en las consultas.

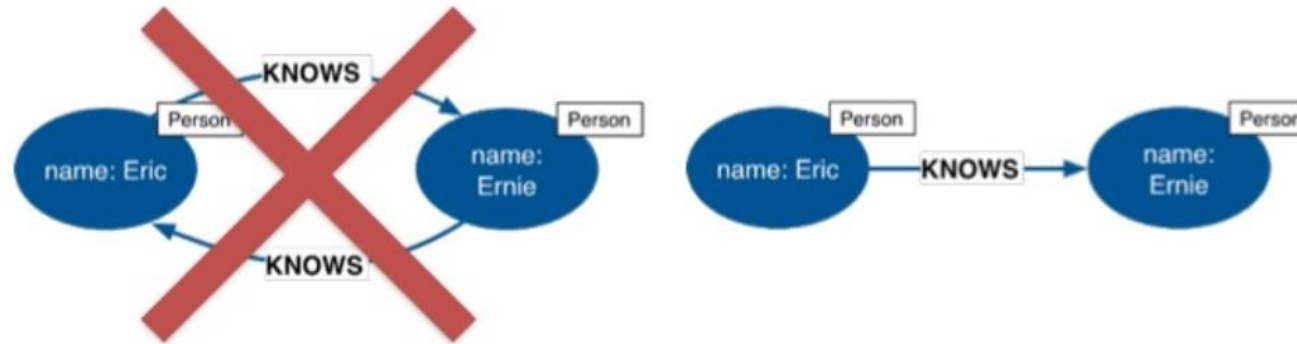


DBlandIT



Modelado de Datos

Relaciones Bi-direccionales

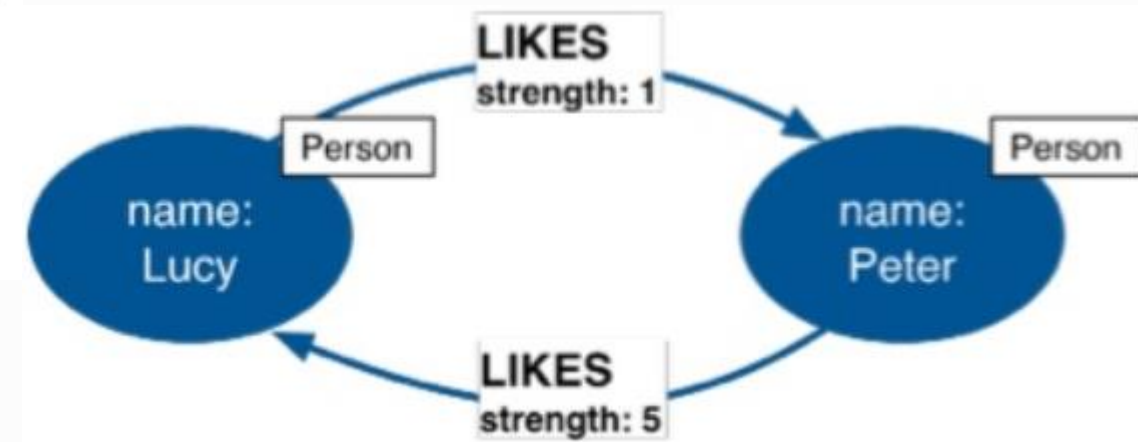
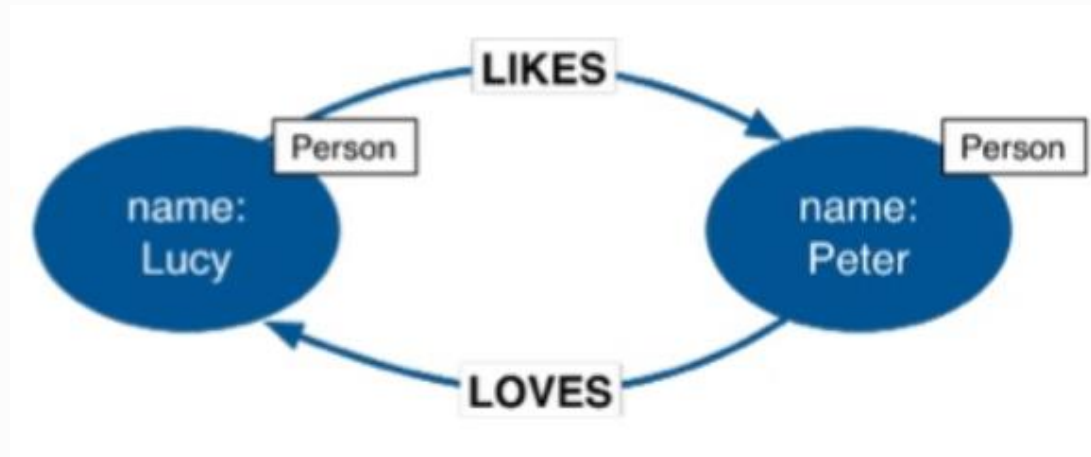


```
MATCH (p1{name:'Eric'})  
      -[:KNOWS]-(p2)  
RETURN p2
```



Modelado de Datos

Relaciones Bi-direccionales Calificadas



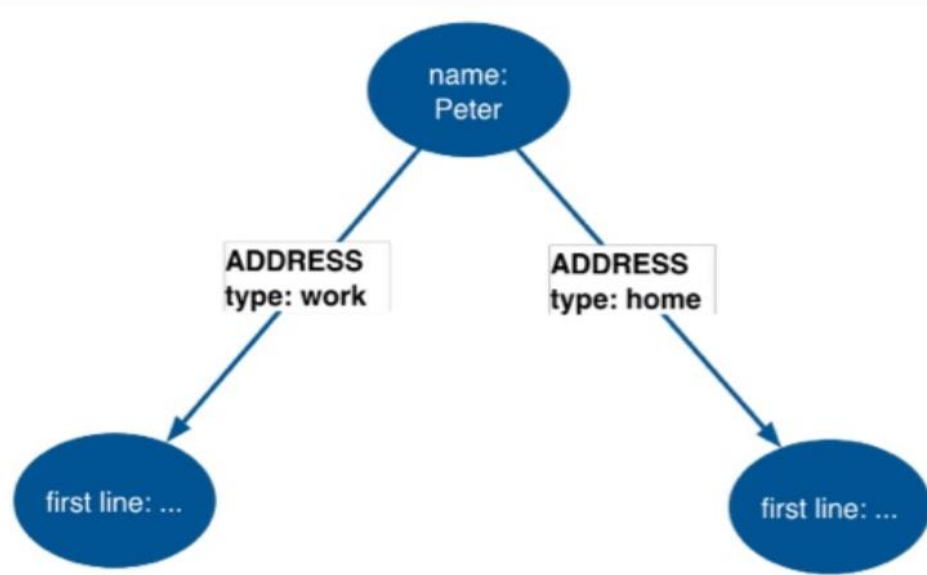
Se podría resolver con una misma bidireccional con el mismo nombre, pero con un **atributo que cualifique lo intenso de la relación**.



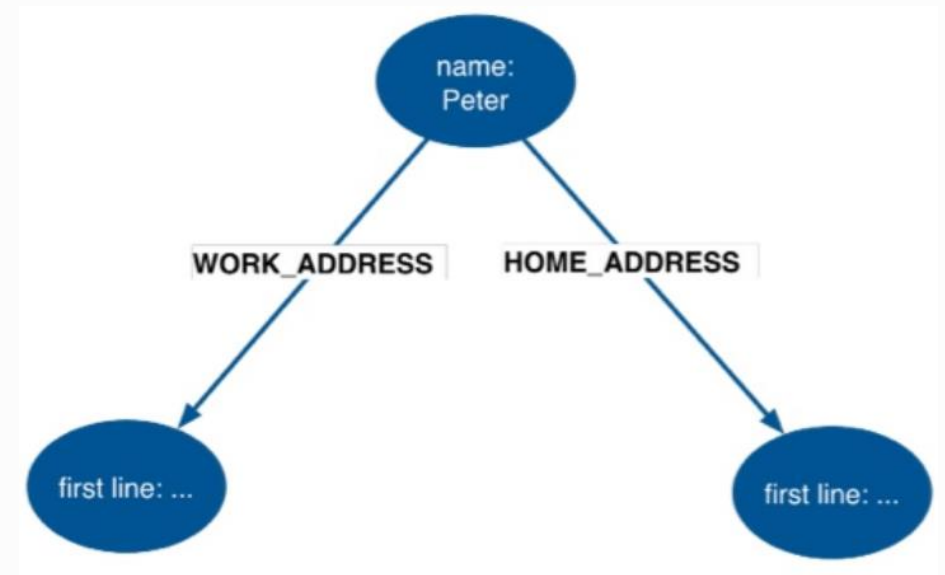
Modelado de Datos

Relaciones Generales vs. Específicas

Relaciones Generales



Relaciones Específicas



Modelado de Datos

Nodos Intermedios



Conectan a 2 o más Nodos en un único contexto

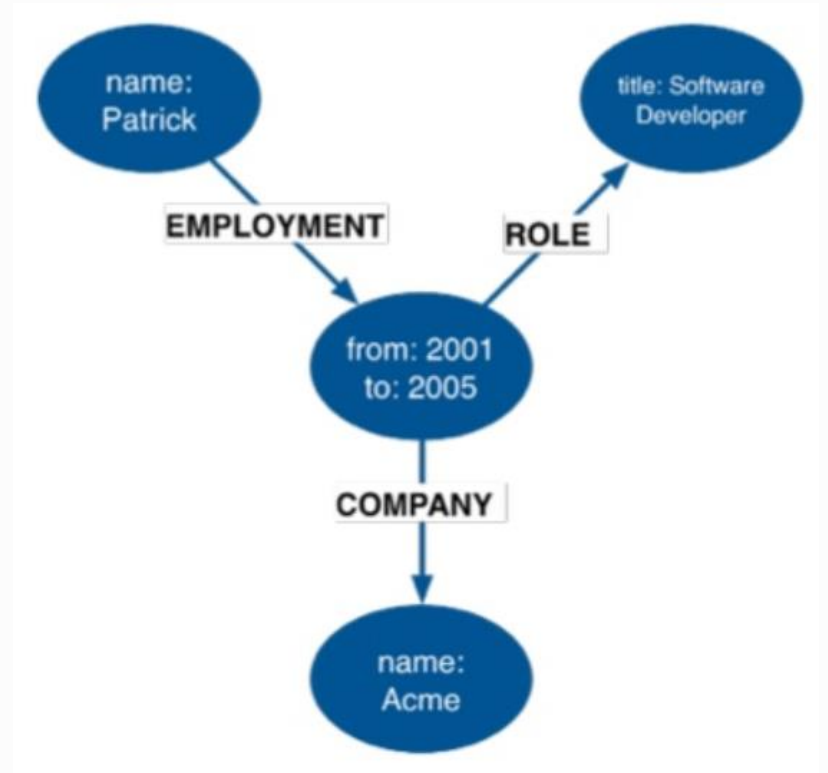
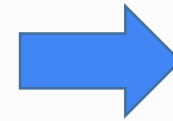
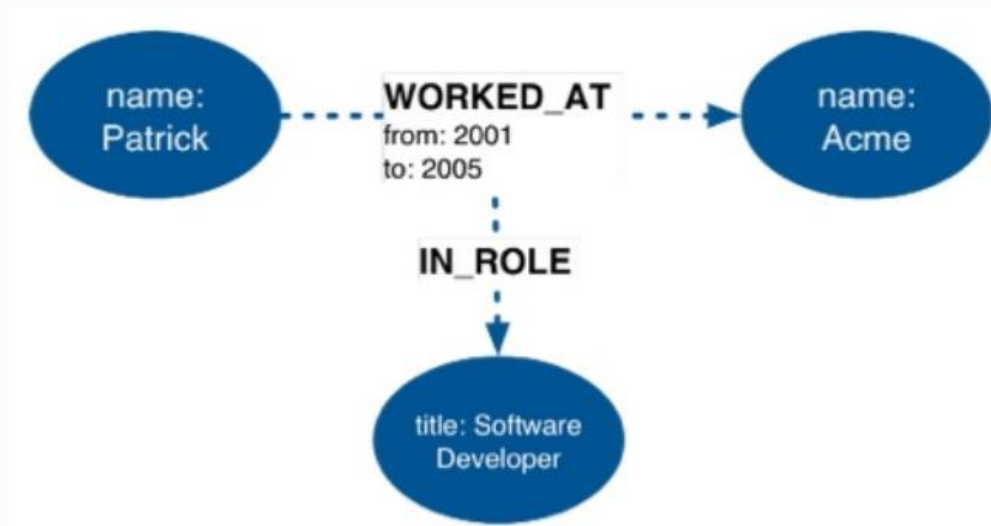


DBlandIT



Modelado de Datos

Nodos Intermedios

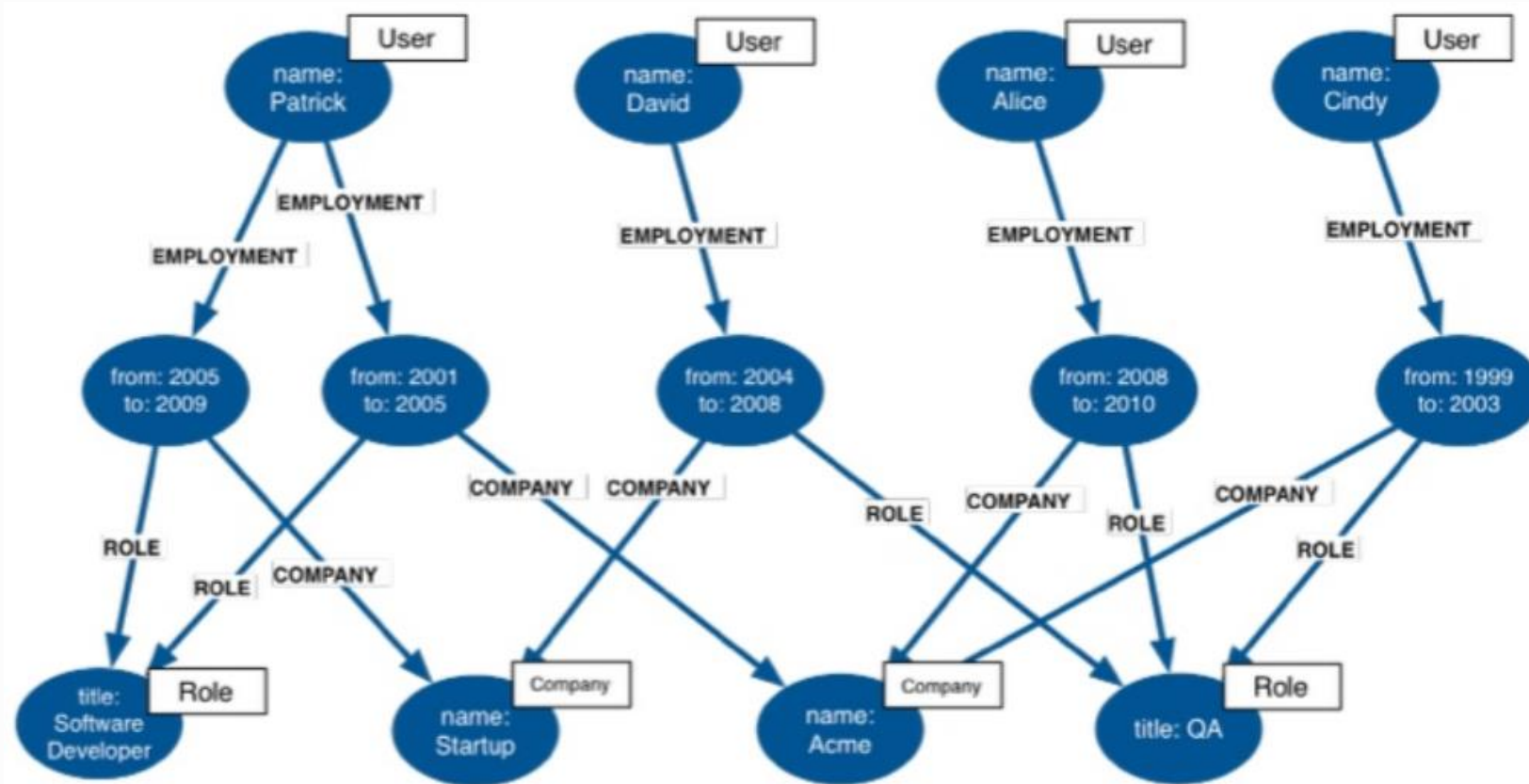


- Conectan a 2 o más Nodos en un único contexto.
- Contexto Rico
- Múltiples Dimensiones



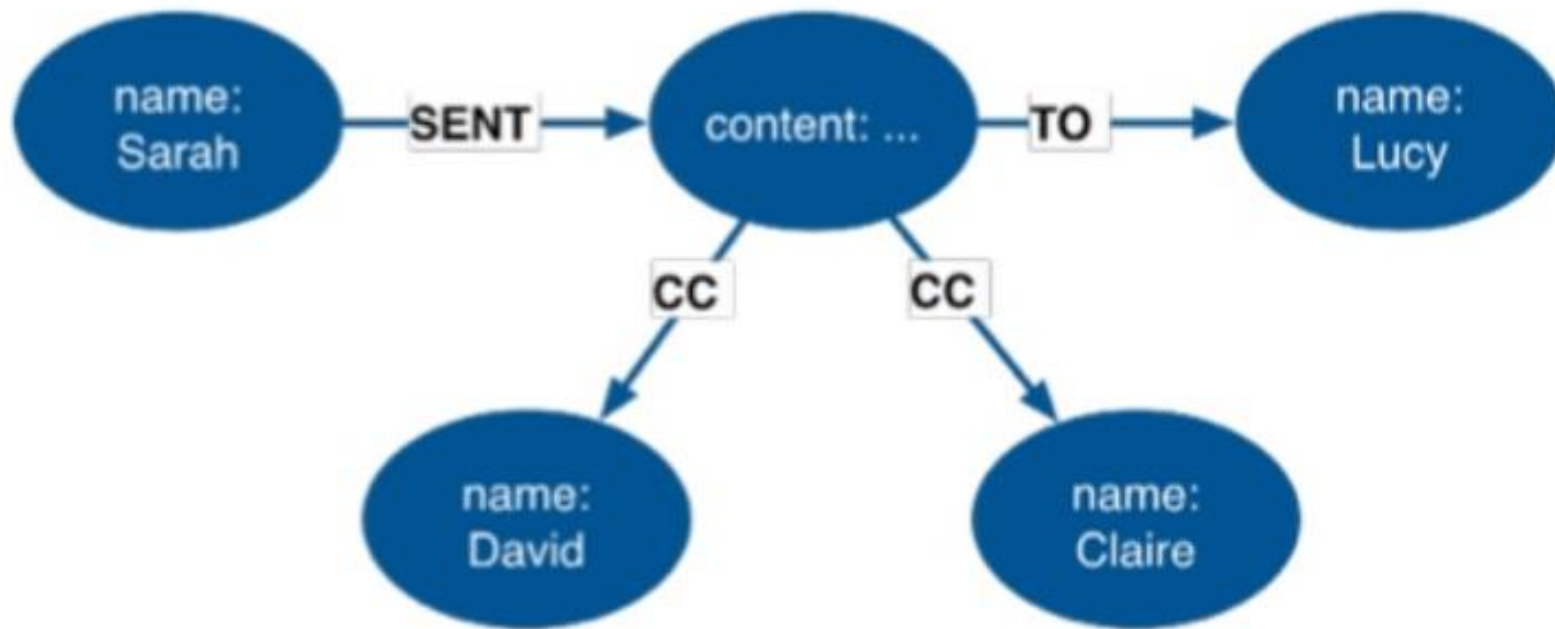
Modelado de Datos

Nodos Intermedios entre contextos



Modelado de Datos

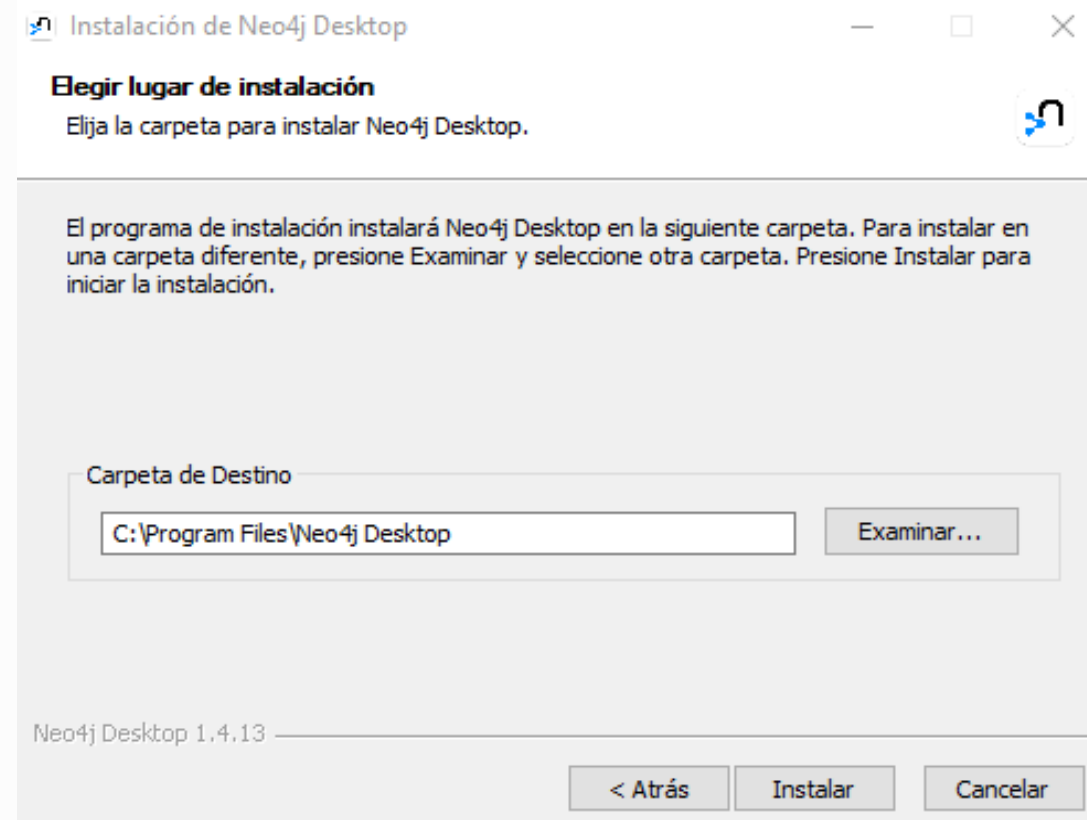
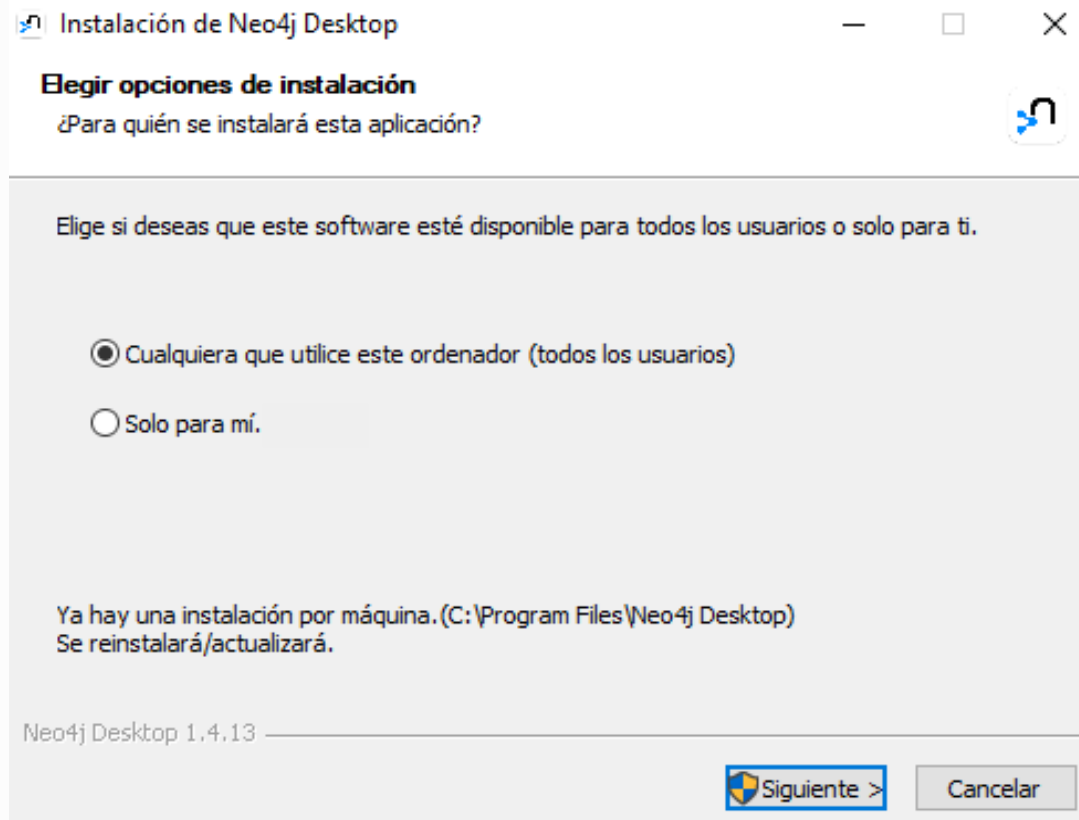
Relaciones Múltiples



Presentación de Neo4J Browser y Cypher Shell



Neo4J Desktop



Neo4J Desktop

Última Versión

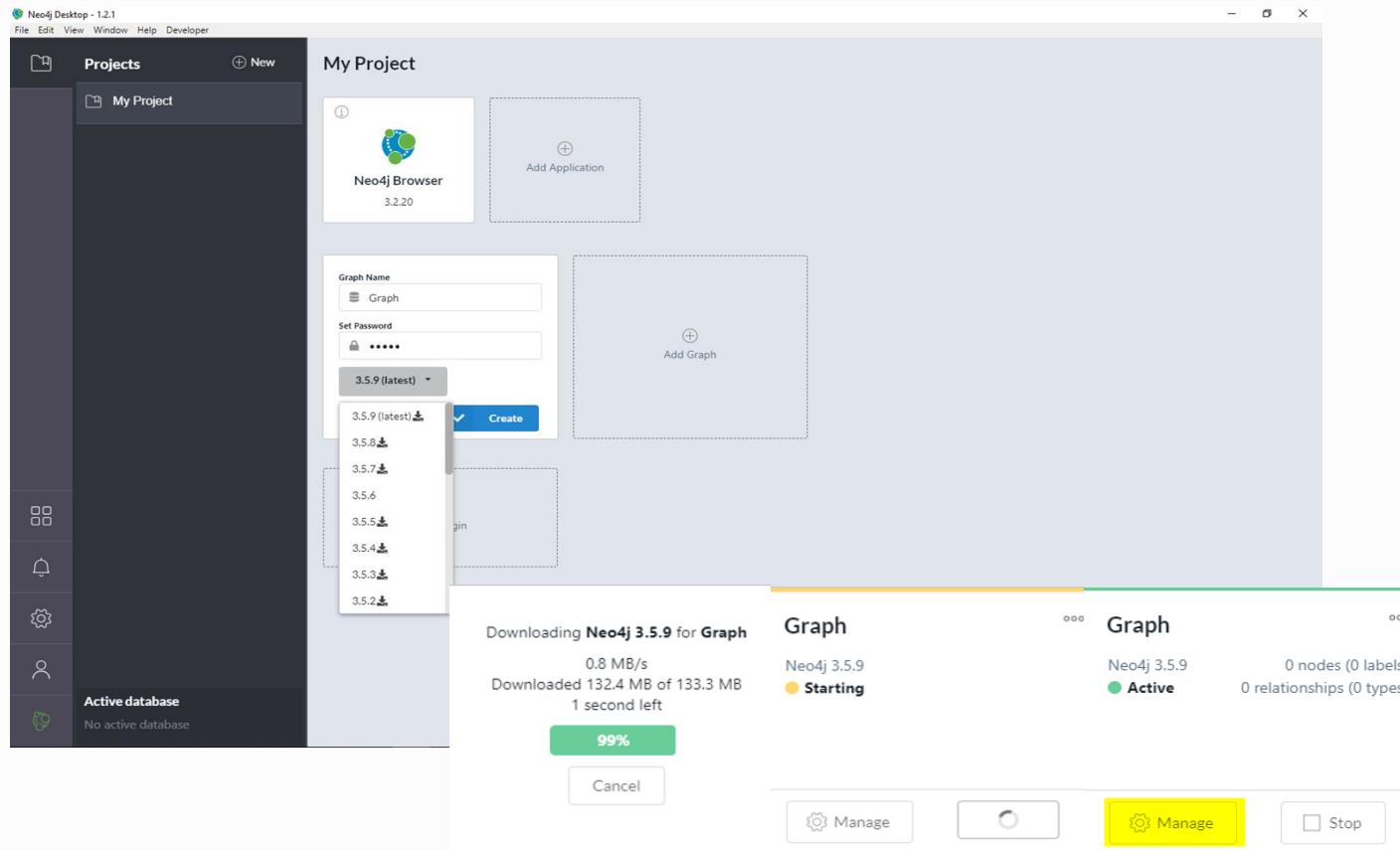
Neo4j Desktop 1.5.7

<https://neo4j.com/download-center/#desktop>

Download Neo4j Desktop

Current Releases

Enterprise Server	Community Server	Neo4j Desktop
Current Release		
Neo4j Desktop 1.5.7		
OS	Download	
Mac	Neo4j Desktop (dmg)	
Linux	Neo4j Desktop (ApplImage)	
Windows	Neo4j Desktop (exe)	



DBlandIT






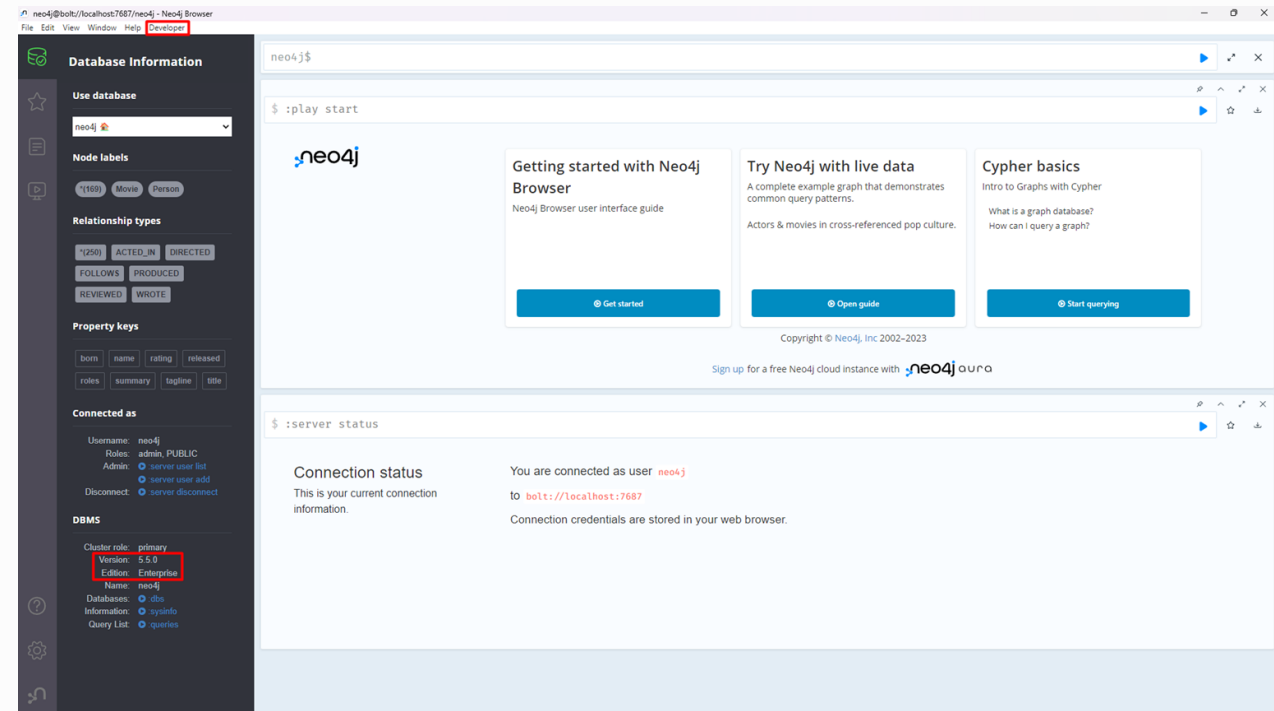
Neo4J Desktop

Graph



Details Logs Terminal Settings Plugins Upgrade Administration

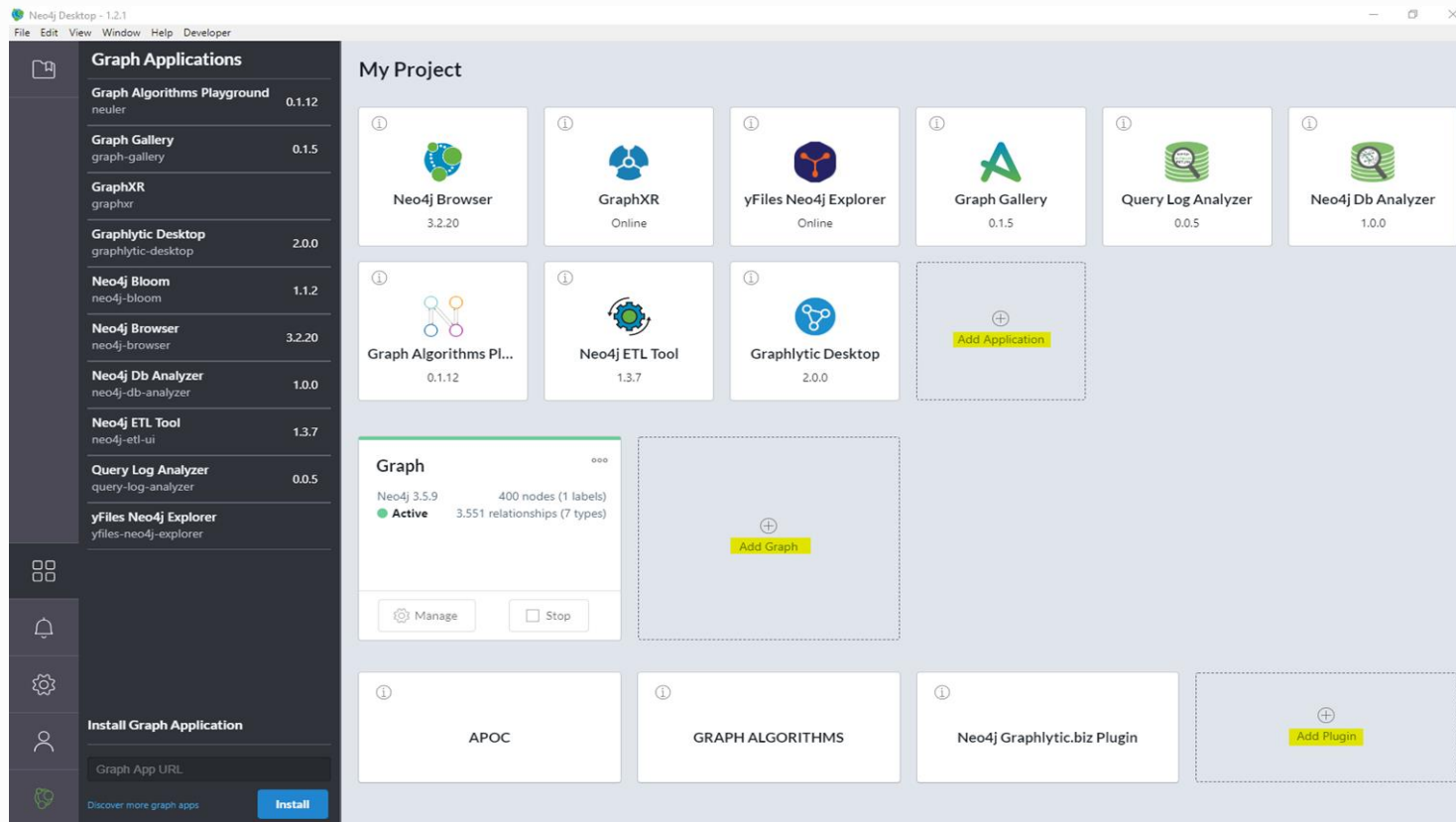
Version	3.5.9 Enterprise
Status	RUNNING
Nodes	0
Labels	0
Relationships	0
Relationship types	0
IP address	localhost
Bolt port	7687 
HTTP port	7474 
HTTPS port	7473 



DBlandIT

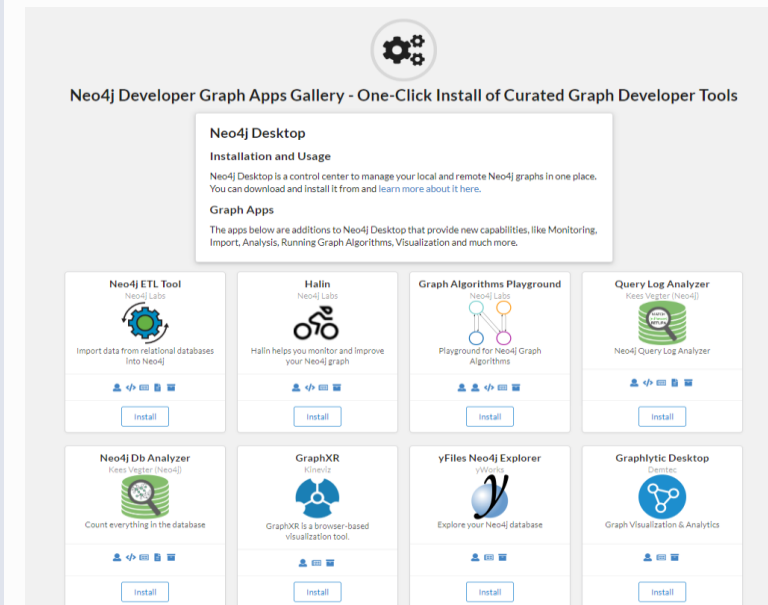


Neo4J Desktop



Aplicaciones de grafos:

Las siguientes aplicaciones son adicionales a Neo4j Desktop. Proporcionan nuevas capacidades, como Monitoreo, Importación, Análisis, Algoritmos de gráficos en ejecución, Visualización y mucho más.



<https://install.graphapp.io/>



DBlandIT



Neo4J Browser

<http://127.0.0.1:7474/browser/>



Learn about Neo4j

A graph epiphany awaits you.



What is a graph database?
How can I query a graph?
What do people do with Neo4j?

[Start Learning](#)

Jump into code

Use Cypher, the graph query language.



Code walk-throughs
RDBMS to Graph

Monitor the system

Key system health and status metrics.



Disk utilization
Cache activity
Cluster health and status

127.0.0.1:7474/browser/

Database Information

Node Labels

- Carrera
- Conocimiento
- Empresa
- Institucion
- Persona
- proceso

Relationship Types

- CONOCE_A
- DICTADA_EN
- ESTUDIO
- POSEE
- PRECEDE_A
- TRABAJO

Property Keys

- Abreviacion
- Asistencia
- Grupo
- Nota
- anio
- ano
- apellido
- apellido
- año
- calificacion
- calificacion
- cargo
- codigo
- cuatr
- cuatrimestre
- curso
- descripcion
- edad
- electiva
- email
- esElectiva
- es_electiva
- estado
- fecha
- fechamicio
- fechad
- fechah
- fechanac

127.0.0.1:7474/browser/

\$ MATCH (n) RETURN n LIMIT 25

Graph

Table

Text

Code

Displaying 25 nodes, 36 relationships.

\$:play start

neo4j

Learn about Neo4j

Jump into code

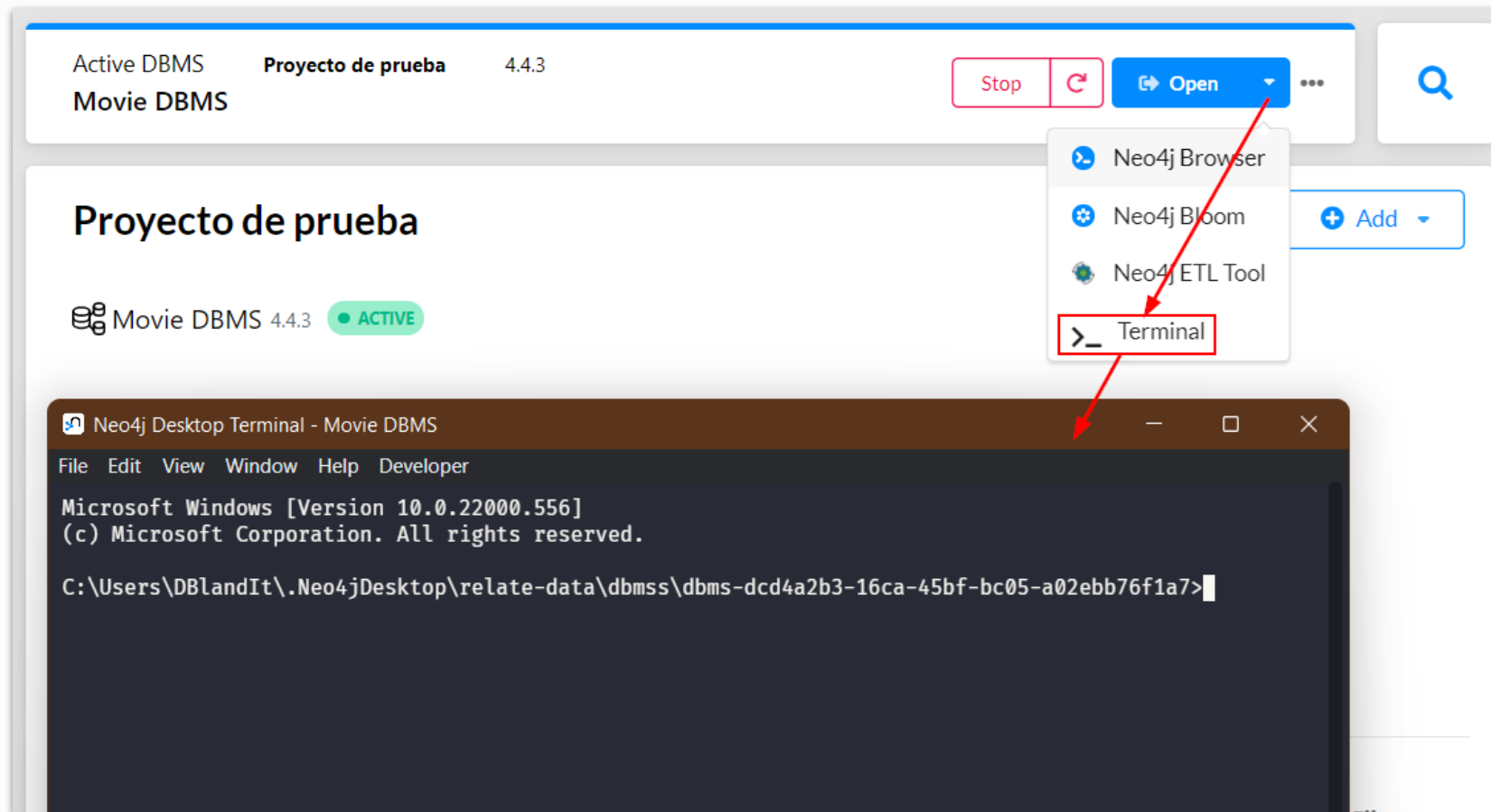
Monitor the system



DBlandIT



Cypher Shell





DBlandIT

info@dblandit.com
+54 11 3889-4009
www.dblandit.com/

