Neo4j

- Base de datos orientada a grafos
  Nodos y aristas
  → Cada nodo es una ocurrencia de datos
  → Entre nodos hay relaciones unidireccionales

Utilizado por:
→ Ebay
→ Airbnb
→ IBM

Ambiente

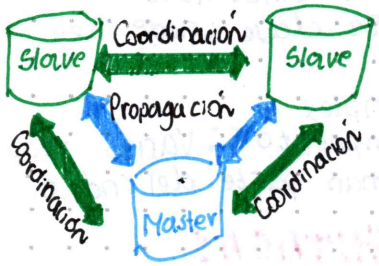
- Neo4j Desktop
- Neo4j Sandbox
- Neo4j Aura

ARQUITECTURA

- Está pensada en dos tipos de disponibilidad:
- Neo4j : Disponibilidad y Consistencia

High Available Cluster

- Arquitectura master-slave



→ Lecturas se pueden hacer sobre cualquier instancia (disponibilidad)

→ Instancias secundarias se comunican con la principal para obtener actualización de los datos (propagación)

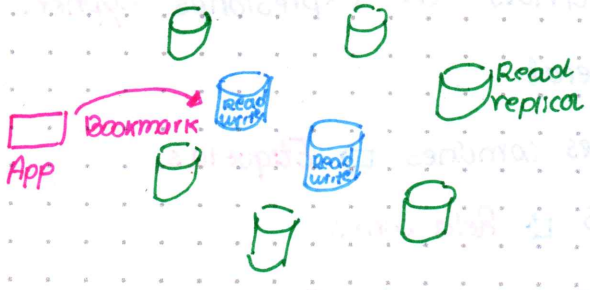
→ Escrituras pueden hacerse directamente sobre la instancia principal o secundaria.

Casual cluster

- Consistencia casual
- Existen dos tipos de componentes:

→ Core Servers: Nodos sobre los cuales se produce lectura y escritura. Su principal cometido es salvaguardar los datos.

→ Read replicas: Su cometido es el de escalar la carga de trabajo de las operaciones de lectura.

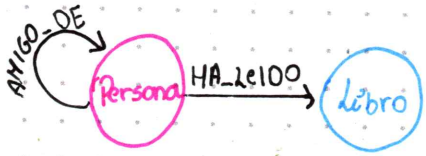


- Al ejecutar una transacción, el cliente solicita una marca (bookmark) que luego podrá "presentar" como parámetro en subsecuentes transacciones.
- Utilizando ese bookmark, el cluster puede garantizar que el cliente sólo efectuará lecturas sobre servidores que hayan procesado la transacción del cliente que generó ese bookmark.

MODELO DE DATOS

- Nodos: Representan entidades con un concepto de identidad único.
- Relaciones: Representan conexiones o interacciones entre los diferentes nodos.
- Tanto los nodos como las relaciones pueden tener propiedades.
- Relaciones son unidireccionales.
- Los nodos pueden tener etiquetas que se utilizan para agruparlos.

- Ejemplo esquema:



- El modelo de datos de grafos a menudo se llaman whiteboard friendly y nos acercan al entendimiento del problema en conjunto con el cliente.

Criterios de diseño

- Consiste en crear una estructura gráfica

1. Identificar las preguntas del dominio
2. Identificar las entidades y relaciones
3. que aparecen en estas preguntas. Traducirlas en expresiones Cypher.

- En general:

Nombres comunes ⇨ Etiquetas

Verbos ⇨ Relaciones

Nombre propio ⇨ Instancia
Características ⇨ Propiedades

CONSULTAS

- Cypher es el lenguaje desarrollado por Neo4j para la creación de bases de datos así como para realizar operaciones.

Tipos de datos:
- Básicos
- De estructura: → Node
  → Relationship
  → Path (ruta en el grafo)

- CREATE: Crear nuevos nodos

```
CREATE (LaroNodo: persona {nombre:'Laro', anio_nac:1978})
```
Operación · Variable temporal · Etiqueta · Propiedades

Para crear relaciones:

```
CREATE (ValvanuzNodo) -[:AMIGO_DE {anio_desde:2003}]-> (LaroNodo),
       (LaroNodo) -[:AMIGO_DE {anio_desde:2003}]-> (ValvanuzNodo)
```

- MATCH: Permite hacer consultas con condiciones sobre los datos de los nodos y sus relaciones.

```
MATCH (p: persona) RETURN p
```
Variable · Etiqueta

```
MATCH (p:persona) WHERE p.nombre='Laro' RETURN p.anio_nac
```

Operadores AND, OR y XOR en WHERE

Otros operadores: IN
STARTS/ENDS WITH
CONTAINS

ORDER BY
LIMIT
DISTINCT

Operaciones de agregación: Count(*)
max, sum, avg, etc.

- SET: Puede utilizarse para añadir nuevas propiedades o actualizar.
- REMOVE: Para eliminar propiedades o etiquetas.
- DELETE: Para eliminar nodos.
- Constraints: De unicidad
  De existencia
  De clave (node key)
- Índices: Simples
  Compuestos: Varias propiedades forman parte del índice.

VENTAJAS

- Modelo muy natural
- Uso de algoritmos basados en grafos para consultas referidas a estructura.
- Mapeo simple a lenguajes orientados a objetos.
- Altamente disponible y tolerante a partición. (*)

DESVENTAJAS

- Modelo de datos no está estandarizado
- Dificultad de cambio de gestor.
- Cypher no es lenguaje estandarizado.
- Falta de herramientas ETL
- Replicación de grafos completos.