📖Capítulo 6 – Neo4j

Dia 3 – Distributed High Availability

No terceiro dia com o Neo4j, o foco é compreender como o banco de grafos se comporta em ambientes **distribuídos**, garantindo **alta disponibilidade (HA)** e **escalabilidade**.

🔦Replica Sets vs. Clustering no Neo4j

Diferente de bancos como MongoDB ou CouchDB, o Neo4j não se baseia em replicação simples. Ele utiliza um **cluster** com múltiplas instâncias coordenadas.

- **Primary (Leader)** → recebe todas as escritas.
- **Followers** → replicam dados do líder e podem atender consultas de leitura.
- **Read Replicas** → nós adicionais usados apenas para escalar leituras.

📌Esse modelo é chamado de **Causal Clustering**, garantindo consistência transacional e tolerância a falhas.

🩲Causal Clustering

- Baseado no protocolo **Raft**, que coordena quem será o líder.
- Escritas → sempre vão para o líder.
- Leituras → podem ser servidas por followers ou replicas.
- Em caso de falha do líder, ocorre uma **eleição automática** para escolher um novo líder.

Exemplo de configuração mínima: - 1 Leader. - 2 Followers. - (Opcional) várias Read Replicas.

🔔Esse arranjo garante tanto **disponibilidade** quanto **distribuição de carga**.

🗄️Alta Disponibilidade

- **Failover automático** → se o líder cair, outro nó assume.
- **Sincronização contínua** → dados replicados em tempo real entre nós.
- **Leituras distribuídas** → escalam horizontalmente sem sobrecarregar o líder.

⚖️Trade-offs (CAP Theorem)

O Neo4j privilegia: - **Consistência** (C). - **Disponibilidade** (A).

📌A partição de rede (P) pode afetar temporariamente escritas, mas o cluster mantém integridade dos dados.

🛠️Ferramentas de Operação

- **Neo4j Browser** → interface web para queries e monitoramento.
- **Neo4j Bloom** → visualização gráfica de redes complexas.
- **Neo4j Ops Manager** → gerenciar clusters e monitorar métricas.

🔗Resumo do Dia 3

- O Neo4j garante alta disponibilidade via **Causal Clustering**.
- Líder recebe escritas; seguidores e réplicas ampliam leituras.
- Failover automático com protocolo Raft.
- Perfeito para sistemas de missão crítica que exigem **consistência forte em grafos**.

🕑Wrap-Up do Capítulo 6 – Neo4j

- **Dia 1** → CRUD e Cypher (fundamentos de grafos).
- **Dia 2** → REST API, índices e algoritmos de grafos.
- **Dia 3** → Escalabilidade e alta disponibilidade com clusters.

📌Conclusão: o Neo4j é ideal quando **relacionamentos são centrais** para o domínio do problema (ex.: redes sociais, rotas, recomendação, detecção de fraude).