APRESENTAÇÃO
DE APOIO

BANCOS DE DADOS NoSQL

Vinícius Kroth – Aula 03

PUCRS online uol edtech

Professores

VINÍCIUS KROTH
Professor Convidado

Desenvolvedor de aplicações SOA nas áreas de contabilidade, financeira e de comércio exterior por mais de 5 anos. Referência em assuntos como projeto e desenvolvimento de SOA e Microservices, Java (EE, Spring framework MVC/WebFlux, jUnit e Gradle), SQL e NoSQL Db's (PostgresSQL, MySQL, MongoDB, Redis e Elasticsearch), AWS Cloud computing, Stress/Chaos testing (Gatlin, Jmeter), ferramentas de Desenvolvimento (Jenkins, Docker e Terraform), Telemetria e Observabilidade (Kibana, Datadog).

EDUARDO HENRIQUE PEREIRA DE ARRUDA
Professor PUCRS

Fundador e CEO da Doc.Space Documentos Digitais. Professor da Escola Politécnica da Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS), onde atua desde 1994 em cursos de graduação, pós-graduação e extensão nas áreas de Ciência da Computação, Engenharia de Software e Sistemas de Informação. Possui graduação (1992) e mestrado (1995) em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (UFRGS) e formações complementares em gestão de TI e Segurança da Informação. Cursa Doutorado no Programa de Pós-Graduação em Ciência da Computação (PPGCC) da PUC-RS. Dedica parte de seu tempo a atividades de incentivo ao empreendedorismo inovador e investe em empresas que adotem modelos de negócio inovadores e escaláveis. Apoia projetos de empreendedorismo social, tendo sido coordenador do projeto Adoções, parceria entre o Poder Judiciário e o Ministério Público do RS com a PUC-RS que, por meio de aplicativo, realiza a aproximação entre candidatos a adoção e crianças e adolescentes em processo de adoção tardia.

Ementa da disciplina

Introdução aos conceitos e características de Big Data como: volume, velocidade, variedade, validade, volatilidade e valência. Introdução aos conceitos de cluster, domínios, agregados, distribuição, tolerância a falhas e sharding. Estudo do Teorema CAP: consistência (Consistency), disponibilidade (Availability), tolerância de partição (Partition). Introdução a Bancos de dados sem esquema prévio, a Banco de dados baseado em documentos, a Banco de dados chave-valor, a Banco de dados colunar e a Banco de dados baseado em grafos.

Redis

Porquê usar Redis?

- ***Throughput*** de leitura/escrita excelentes;
- Esquema de dados **flexível**;
- **Linguagem** simples;
- **Integração** com múltiplos clientes;
- Facilmente **escalável;**
- Suporta adição de **módulos** com **funcionalidades estendidas**;

Principais conceitos

- Todo valor salvo é atrelado à uma chave única;
- A arquitetura do Redis é ***single-threaded;***
- Os valores podem ter **formatos/estruturas** diferentes;
- Os dados são todos mantidos em **memória** (RAM);
- Cada valor tem um tamanho máximo de **512MB**;
- Apesar de não oferecer **paralelismo**, tem suporte à **concorrência**;

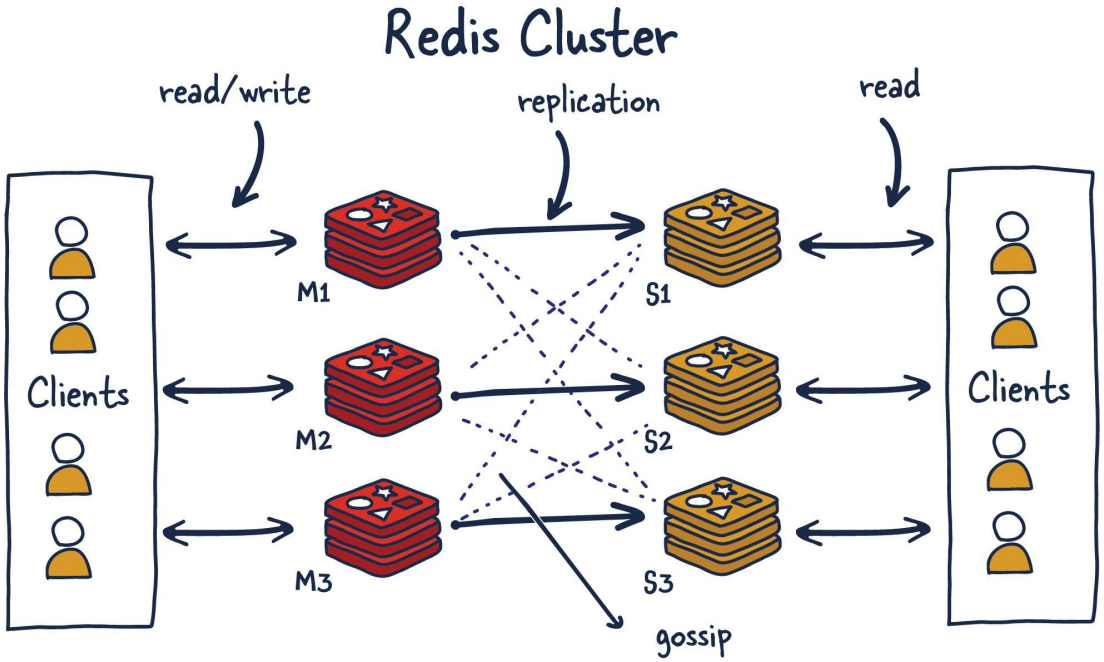
Modelos de persistência

- ***RDB*** (***Redis Database***): Realiza ***snapshots*** do banco de dados, de tempos em tempos, salvando o estado atual. Comum em casos onde o Redis é um **banco secundário**;

- ***AOF*** (***Append Only File***): Cria um arquivo de **log,** que salva todos os comandos realizados no Redis, que são repetidos quando o banco é reinicializado; Comum em casos onde o Redis é o **banco principal/fonte da verdade**;

- **Sem persistência:** Desabilitar persistência por completo, comum em casos onde o Redis é utilizado para ***caching*****;**

Arquitetura Redis: Como escalar?



Redis Search

- Módulo que dá ao Redis a funcionalidade de indexar, e realizar **buscas** por **elementos**;
- Suporte à **paginação**;
- Suporte à funções de **agregação**;

Casos de uso

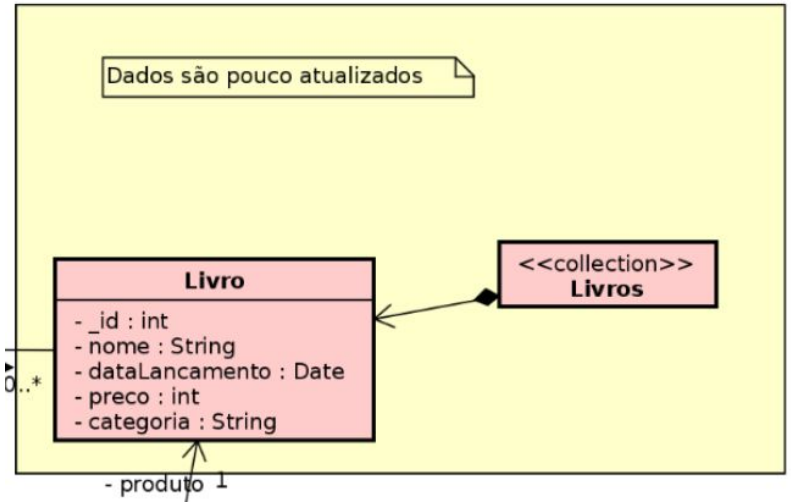
- Cache de informações **frequentemente** acessadas;
- Armazenamento de informação de **sessão** de usuário;
- **Chats**, sistemas de mensagem;
- **Placares** de jogos;
- Análise de dados em **tempo real** (**Redis Search**);




Tradeoffs

- ***Single*-*threaded***, podendo gerar gargalo;
- **Custo** de armazenar os dados em memória;
- Natureza **efêmera**, podendo gerar ***downtime***;
- Armazenamento de cada valor tem um **limite físico;**
- Funcionalidades **completas** dos módulos extras, só se encontram em versão paga;
- Sem a utilização dos **módulos**, só é eficiente acessar os dados pela chave;

Exemplo prático



Exemplo prático

- Utilizar o Redis para cachear os dados de livros;
- Utilizar o Redis Search para criar um índice de livros;
- Realizar operações de *FTS*;

Neo4J

Porquê usar Neo4J?

- Suporta transações **ACID**;
- Base de dados **flexível**;
- Provêm protocolo de comunicação **REST**;
- Muito performático em situações de **conexões**/**relacionamentos** entre os dados;
- Consultas muito **eficientes**, quando comparadas com outros bancos;

Principais conceitos

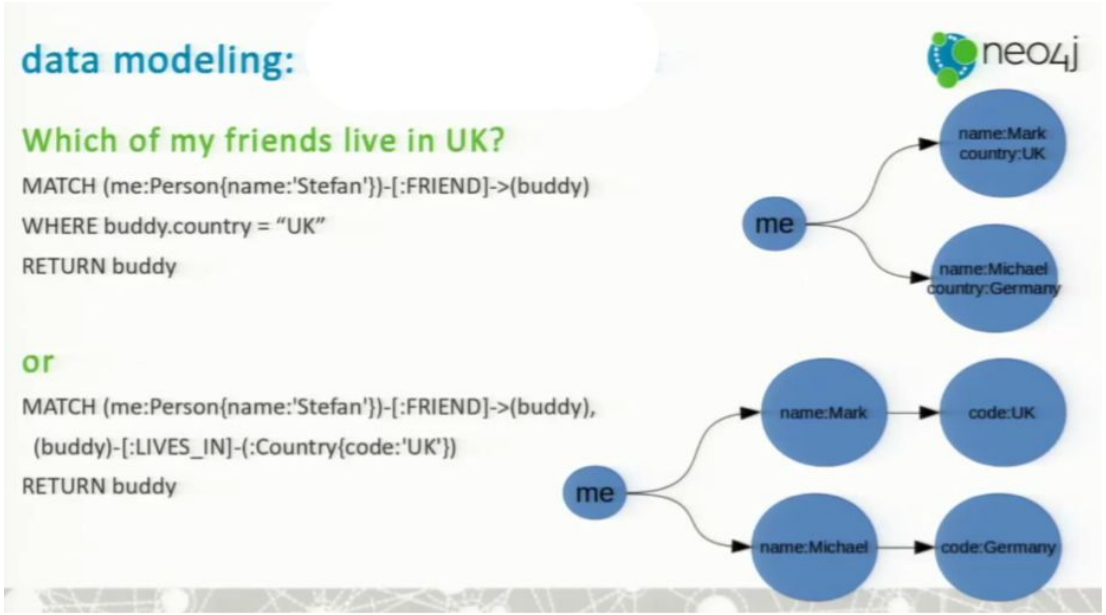
- **Nós:** A unidade básica, o registro no banco;
- ***Labels:*** Agrupamento/conjunto de nós;
- **Relacionamentos:** Conexão/ligação entre os nós;
- ***Aspects:*** Informações de cada nodo;
- Baseado em **algoritmos** tradicionais (ex: *Dijkstra)*
- Linguagem de busca ***Cypher*** baseada em SQL;
- Feito em **Java;**

Benchmarks oficiais

- Cenário: Buscar todas as relações de **segundo**, **terceiro**, **quarto** e **quinto** nível de uma pessoa em redes sociais:

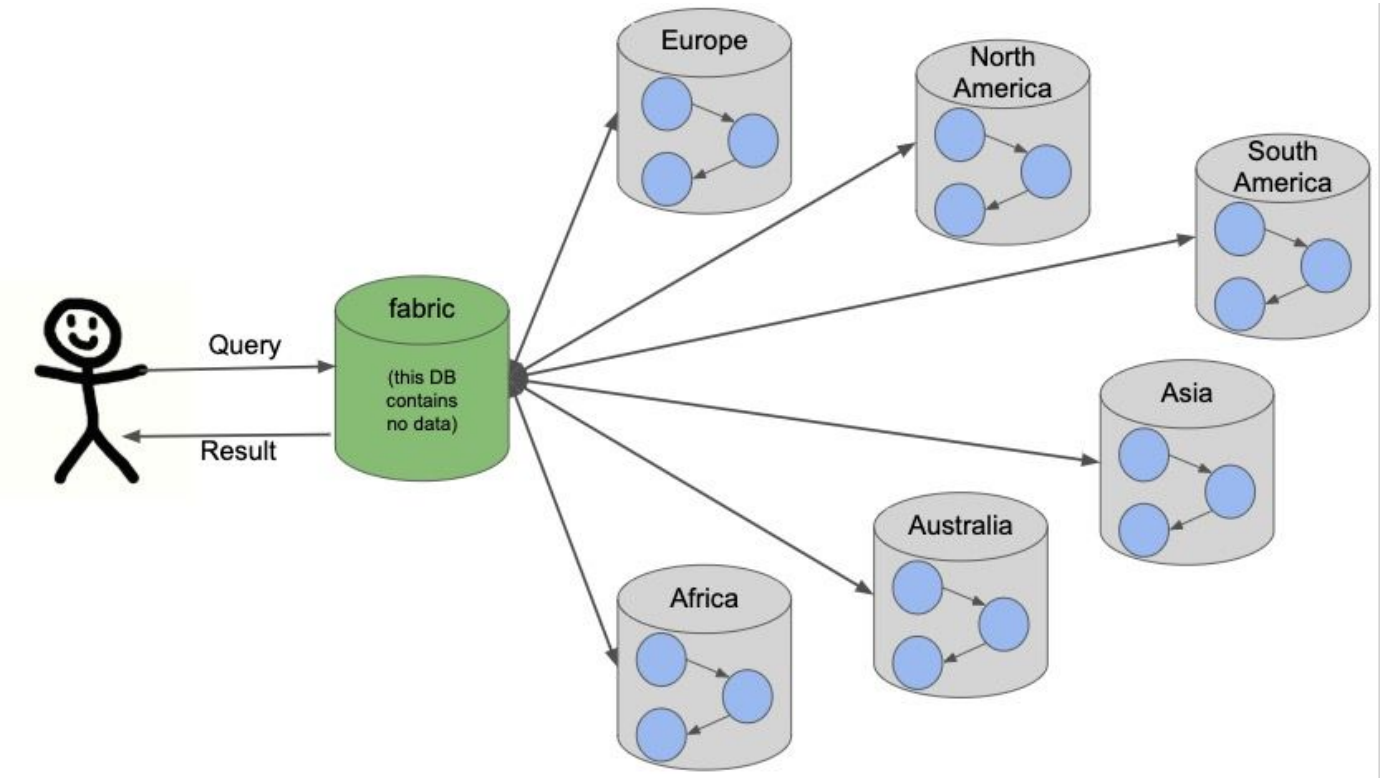
Depth	Execution Time – MySQL	Execution Time –Neo4j
2	0.016	0.010
3	30.267	0.168
4	1,543.505	1.359
5	Not Finished in 1 Hour	2.132

Modelando dados, boas práticas



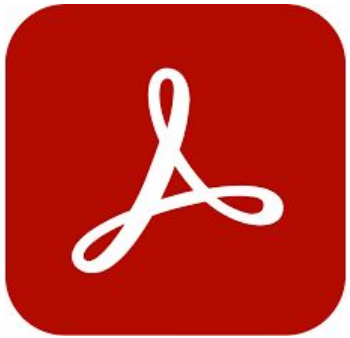
- Extrair as informações (aspectos), para **fora** do **nodo** quando forem ser usadas em **consultas**;
- Quando **necessário**, modelar relações de "volta" (ex: joão mora em paris, paris tem como morador joão);

Arquitetura Neo4J: Como escalar?



Casos de uso

- Redes **sociais**;
- Detecção de **fraude;**
- **Recomendação de conteúdo;**
- Controle e gestão de **identidade;**
- Análise de **riscos**;
- **Mapas** e **coordenação de rotas;**



VOLVO

J.P.Morgan

ebay

Tradeoffs

- Casos de uso **específicos**, não é de propósito geral;
- Falta de suporte para operações de **agregação;**
- Comunidade relativamente **pequena**;
- Não é **apropriado** para escritas massivas (problemas de memória);

Exemplo prático

- Introdução à linguagem *CYPHER;*
- Visualização de grafos;
- Operações em datasets de exemplos (https://neo4j.com/developer/example-data/);

Meus 20 centavos

- **Padrões** de acesso;
- **Formato** dos dados;
- **Consistência** dos dados;
- Frequência de **acesso**;
- Mais **escrita** ou mais **leitura**?
- Durabilidade dos dados;
- Bancos cada vez mais **evoluídos** e **polivalentes** (ex: -> redis search);
- Relacional deve resolver a maioria dos **problemas**;
- Persistência **poliglota** (problemas **complexos** -> soluções **complexas**);

Leituras recomendadas:

- **NoSQL Essencial**: https://www.amazon.com.br/NOSQL-Essencial-Pramod-J-Sadalage/dp/8575223380
- **Designing Data-Intensive Application**: https://www.amazon.com.br/Designing-Data-Intensive-Applications-Martin-Kleppmann/dp/1449373321
- **Philosophy of Software Design:** https://www.amazon.com.br/Philosophy-Software-Design-John-Ousterhout/dp/1732102201

Documentação Oficial:

- **MongoDB**: https://www.mongodb.com/docs/
- **Cassandra**: https://cassandra.apache.org/doc/latest/
- **Redis:** https://redis.io/docs/
- **Neo4J:** https://neo4j.com/docs/

Muito obrigado!

PUCRS online uol edtech