

NoSQL

Prof. Igor Avila Pereira
igor.pereira@riogrande.ifrs.edu.br

Divisão de Computação
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Câmpus Rio Grande

Agenda

- 1 NoSQL
- 2 Redis
- 3 Jedis
- 4 Gson

Agenda

1 NoSQL

- Introdução
- BD's Relacionais vs BD's NoSQL
- NoSQL: Tipos

2 Redis

3 Jedis

4 Gson

Introdução

- O termo NoSQL surgiu em 1998, mas foi em 2006, quando foi citado pelo Google, que o termo popularizou-se.
- Era uma época onde os bancos relacionais não mais suportavam a massa de dados da internet.
- Só a internet hoje armazena alguns terabytes de dados.

Introdução

O que são bancos de dados NoSQL?

- Os bancos de dados NoSQL são, basicamente, bancos de dados que não são relacionais (SQL).
- O nome NoSQL já indica **Not Only SQL**
- Os BD'S NoSQL não precisam, necessariamente, ser parecidos entre si.

Introdução

Como funciona um banco de dados NoSQL (não relacional)?

- Os bancos de dados NoSQL usam uma variedade de modelos de dados para acessar e gerenciar os dados.
- Esses tipos de banco de dados são otimizados, especificamente, para aplicativos que exigem modelos de grande volume de dados, baixa latência e flexibilidade.
- Esses requisitos são atendidos mediante o relaxamento de algumas restrições de consistência de dados dos outros bancos.

BD's Relacionais vs BD's NoSQL

Ex: Considere um banco de dados simples de livros:

- Em um B.D relacional, um registro de livro é normalmente disfarçado (ou normalizado) e armazenado em **tabelas** separadas, e os **relacionamentos** são definidos por restrições de **chave primária** e **chave estrangeira**.
 - Neste exemplo, a tabela Livros têm **colunas** para ISBN, Título do livro e Número da edição, a **tabela Autores** têm **colunas** para AuthorID e Nome do autor e, finalmente, a **tabela Author-ISBN** tem colunas para AuthorID e ISBN.
 - O **modelo relacional** é projetado para permitir que o banco de dados **imponha a integridade referencial entre as tabelas** no banco de dados, normalizadas para reduzir a redundância e geralmente otimizadas para armazenamento.

BD's Relacionais vs BD's NoSQL

- Em um banco de dados NoSQL, um registro de livro é normalmente armazenado como um documento JSON.
- Para cada livro, o item, o ISBN, o Título do livro, o Número de edição, o Nome do autor e o AuthorID são armazenados como atributos em um único documento.
- Neste modelo, os dados são otimizados para desenvolvimento intuitivo e escalabilidade horizontal.

BD's Relacionais vs BD's NoSQL

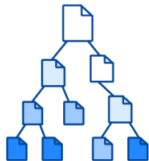
SQL	NoSQL
Armazenamento de Dados Estruturados por Tabela	Armazenamento de Dados estruturados e não-estruturados por colunas, grafos, chave-valor e documentos.
Esquema estático	Esquema dinâmico
Necessidade de predefinição de um esquema de tabela antes da adição de qualquer dado	Altamente flexível (fácil adição de colunas e campos de dados não estruturados)
O desempenho não é alto em todas as consultas. Não suporta pesquisas e cruzamentos muito complexos.	Alto desempenho em consultas

BD's Relacionais vs BD's NoSQL

- BD's NoSQL não vieram para substituir os BD's relacionais.
- Diferente do banco de dados relacional, em que o foco principal é voltado à integridade dos dados
 - os modelos existentes em NoSQL tendem a sacrificar uma ou mais propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), para assim oferecer maior desempenho e escalabilidade às soluções que lidam com grande volume de dados

NoSQL: Tipos

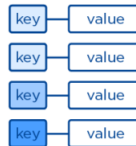
Document



Graph



Key-Value



Wide-column



NoSQL: Tipos

Chave/valor	 RocksDB   redis
Documento	 mongoDB  Couchbase
Coluna	  cassandra  amazon DynamoDB  APACHE HBASE 
Grafo	 JanusGraph  neo4j

NoSQL: Tipos

- **Modelo Colunas:**

- No modelo colunas, o banco de dados faz armazenamento em linhas particulares de tabela.
- Esse esquema é o perfeito oposto dos bancos relacionais, que armazenam conjuntos de dados em uma única linha.
- Exemplos clássicos do modelo de colunas são os bancos **Hbase** e **Cassandra**;

NoSQL: Tipos

- **Modelo Grafos:**

- Armazena dados na forma de grafo.
- Isto é, aqui os dados são dispostos no formato de arcos conectados por arestas.
- Podemos definir como um conjunto de linhas conectadas por vértices também.
- O modelo de grafos é vantajoso frente à pesquisas complexas, pois a latência e a performance promete ser menor do que no modelo chave-valor, por exemplo.
- Um exemplo prático disso é o banco Neo4j.

NoSQL: Tipos

- **Modelo Documento:**

- Neste modelo, os dados são **documentos**.
- É o esquema de armazenamento do **MongoDB**, por exemplo.
- Esse modelo é altamente flexível e não carece de colunas pré montadas, como é o caso do Cassandra.
- Esse modelo é especialmente eficiente para tratar dados não estruturados, já que uma única coleção pode contar com grupos de dados (documentos) de diversos formatos diferentes.

NoSQL: Tipos

- **Modelo Chave-Valor:**

- Em chave-valor, nós temos um banco que é formado por conjuntos de chaves, que por sua vez são acompanhados de valores como tabelas *hash*.
- A estrutura chave-valor também é bem flexível e própria para armazenamento de *Big Data*.
- É interessante também frisar que esse formato é altamente disponível.
- Exemplos práticos são REDIS e **MemcacheD**.

Agenda

1 NoSQL

2 Redis

- Introdução
- Instalação
- redis-cli
- Comandos

3 Jedis

4 Gson

Introdução

- O Redis é um modelo de armazenamento de dados, *open source* lançado em 2009.
- Os dados são armazenados na forma de **chave-valor** e na memória do Redis, o que o torna rápido e flexível.
- Trata-se do Banco NoSQL mais famoso do tipo chave-valor.
- O Redis possui baixíssima latência
- O Redis é também fácil de usar e muito rápido.

Introdução

O que é um armazenamento chave-valor?

- Atribua valores às chaves para facilitar o acesso e o armazenamento desses valores, que sempre são encontrados através das suas chaves.
- Pense em mapas de *hash* e você tem a ideia (dicionários em Python).
- O Redis mantém seus pares de valores-chave na memória, tornando seu acesso rápido.

Introdução

- Se a durabilidade dos dados pode ser sacrificada (principalmente com dados não críticos, ou em situações de somente leitura),
 - ser capaz de renunciar a escritas de dados significa que esses dados somente em memória possuem um desempenho incrivelmente rápido
- Ao longo dos anos, muitas APIs foram desenvolvidas para uma variedade incrivelmente ampla de linguagens de programação, tornando o Redis uma escolha fácil para desenvolvedores.

Instalação

- **Getting-Started:**

<https://redis.io/docs/getting-started/>

- **Debian/Ubuntu:**

- **Via apt:** <https://redis.io/docs/getting-started/installation/install-redis-on-linux/>

- **Via Snap:** <https://redis.io/docs/getting-started/installation/install-redis-on-linux/#install-from-snapcraft>

- **Windows:** <https://redis.io/docs/getting-started/installation/install-redis-on-windows/>

- **Docker:**

```
docker run --name redis -p 6379:6379 -d redis:latest
docker exec -it redis redis-cli
127.0.0.1:6379> ping
PONG -- se deu tudo certo
```

redis-cli

- **Porta Padrão:** 6379
- **Lista completa de comandos:**
`https://redis.io/commands/`

Comandos

Principais Commandos:

```
-- mostra todas as chaves  
KEYS *  
  
-- deleta todas as chaves da conexão.  
FLUSHDB  
  
-- deleta tudo de todos os bancos  
FLUSHALL  
  
-- armazena  
SET <key> <value>  
  
-- retorna o valor de uma chave  
GET <key>  
  
-- deleta uma chave  
DEL <key>  
  
-- seleciona um bd (de 0 a 15)  
SELECT <index>
```

Agenda

1 NoSQL

2 Redis

3 Jedis

- Introdução
- Instalação
- Como Usar?

4 Gson

Introdução

Jedis é um *client* Java para o Redis.

- **Site Oficial:** <https://github.com/redis/jedis>

Instalação

Instalação:

- 1 jar files
- 2 maven dependency

Instalação

Instalação via Maven

```
<dependency>  
  <groupId>redis.clients</groupId>  
  <artifactId>jedis</artifactId>  
  <version>2.8.1</version>  
</dependency>
```

Como usar?

Guias:

- <https://www.baeldung.com/jedis-java-redis-client-library>
- <https://github.com/redis/jedis/wiki/Getting-started>
- <https://www.baeldung.com/redis-delete-data#running-redis>

Agenda

- 1 NoSQL
- 2 Redis
- 3 Jedis
- 4 **Gson**
 - Instalação

Gson

Biblioteca do Google para trabalhar com JSON em Java.

Site Oficial:

- <https://github.com/google/gson/blob/master/UserGuide.md>

Outros

- <https://howtodoinjava.com/gson/gson-parse-json-array/>

Instalação

Instalação via Maven

```
<dependency>  
  <groupId>com.google.code.gson</groupId>  
  <artifactId>gson</artifactId>  
  <version>2.9.0</version>  
  <scope>compile</scope>  
</dependency>
```

NoSQL

Prof. Igor Avila Pereira
igor.pereira@riogrande.ifrs.edu.br

Divisão de Computação
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS)
Câmpus Rio Grande