

MBA⁺

**Data Science &
Artificial Intelligence**



**MBA⁺****Data Architecture,
Integration and Ingestion**

Prof.: Ivan Gancev

Email: profivan.gancev@fiap.com.br



Data Architecture, Integration and Ingestion

(O que vamos explorar?)

Aula 1 – 12/abr (qua)

- Pilares de arquitetura: persistência, integração e consumo
- Estratégias de arquitetura
- Tipos de tratamentos e arquiteturas

Aula 2 – 19/abr (qua)

- Exemplos de Bancos, diferenças e usos:
 - Bancos Relacionais
 - Bancos Colunares

Aula 3 – 26/abr (qua)

- Exemplos de Bancos, diferenças e usos:
 - Bancos de documentos
 - Bancos chave-valor
 - Bancos de Grafos

Aula 4 – 03/mai (qua)

- Ingestão de dados, tratamentos e manipulações
- Pipeline de dados, governança e qualidade
- Integração de dados
 - Cargas batch, ETL, vantagens e desvantagens

Aula 5 – 10/mai (qua)

- Eventos, APIs, NRT e casos de uso
- Arquiteturas para analytics
- Boas práticas, recomendações e cuidados

Bancos de documentos

Características:

- ✓ Emparelha cada chave com uma estrutura de dados complexa conhecida como um documento. Os documentos podem conter muitos pares chave-valor diferentes, ou pares de matriz de chaves, ou mesmo documentos aninhados.
- ✓ Os documentos são armazenados no formato JSON
- ✓ Foco no armazenamento do dado e não no relacionamento

Usos:

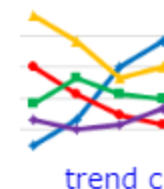
- ✓ Aplicações de distribuição de conteúdo
- ✓ Aplicações que trabalham com JSON
- ✓ Estruturas variáveis
- ✓ Perfis de usuário

DB-Engines Ranking of Document Stores

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

This is a partial list of the [complete ranking](#) showing only document stores.

Read more about the [method](#) of calculating the scores.



☐ include secondary database models

57 systems in ranking, March 2023

Rank			DBMS	Database Model	Score		
Mar 2023	Feb 2023	Mar 2022			Mar 2023	Feb 2023	Mar 2022
1.	1.	1.	MongoDB +	Document, Multi-model ⓘ	458.78	+6.02	-26.88
2.	2.	2.	Amazon DynamoDB +	Multi-model ⓘ	80.77	+1.08	-1.03
3.	3.		Databricks	Multi-model ⓘ	60.86	+0.52	
4.	4.	↓ 3.	Microsoft Azure Cosmos DB +	Multi-model ⓘ	36.10	-0.40	-4.79
5.	5.	↓ 4.	Couchbase +	Document, Multi-model ⓘ	23.36	-1.50	-6.09
6.	6.	↓ 5.	Firebase Realtime Database	Document	18.78	+0.29	-0.80
7.	7.	↓ 6.	CouchDB	Document, Multi-model ⓘ	14.46	+0.01	-3.02
8.	8.	↑ 9.	Google Cloud Firestore	Document	11.36	-0.15	+2.21
9.	9.	↓ 7.	MarkLogic	Multi-model ⓘ	8.86	+0.02	-1.04
10.	10.	↓ 8.	Realm	Document	8.53	+0.28	-1.30
11.	11.	↑ 13.	Google Cloud Datastore	Document	6.62	-0.27	+1.14
12.	12.	↓ 10.	Aerospike +	Multi-model ⓘ	6.54	-0.02	+0.31

<https://db-engines.com/en/ranking/document+store>

JSON

JSON (JavaScript Object Notation) é um modelo para armazenamento e transmissão de informações no formato texto. Apesar de muito simples, tem sido bastante utilizado por aplicações Web devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, tornando mais rápido o parsing dessas informações.

Não tem limitação de tamanho, armazena conjuntos de chave/valor e é suportado pela maioria das linguagens de programação modernas

Ref.: <https://docs.fileformat.com/web/json/>

```
{
  "name":"Jack",
  "age":30,
  "contactNumbers":[
    {
      "type":"Home",
      "number":"123 123-123"
    },
    {
      "type":"Office",
      "number":"321 321-321"
    }
  ],
  "spouse":null,
  "favoriteSports":[
    "Football",
    "Cricket"
  ]
}
```

Particionamento de dados

MongoDB

O que é?

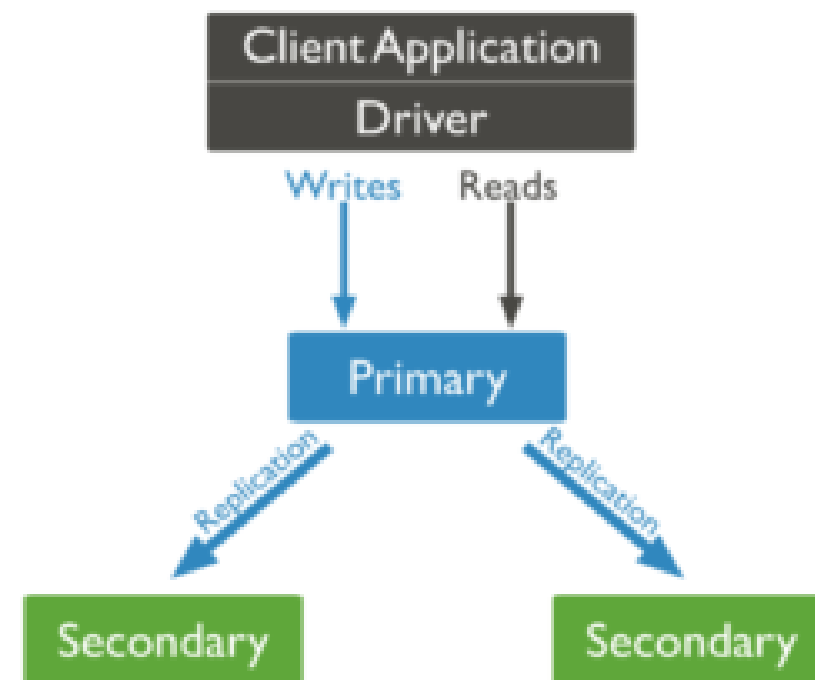
O particionamento de dados (sharding) serve para escalar consultas de dados. As gravações nas coleções de dados são feitas nas partições primárias e replicadas para as partições secundárias.

Benefícios

Essa abordagem favorece aplicações que tenham mais operações de consultas do que gravação. Uma vez que o dado sempre é gravado através da partição primária, mas pode ser consultado através de todas as partições que possuam o dado.

Limitações

Este modelo deixa explícita sua limitação na gravação dos dados. Portanto, as leituras podem ser escaláveis e flexíveis, já as operações de gravação são limitadas



Em caso de falha em um nó primário, um dos secundários é promovido a primário

Particionamento de dados

Couchbase

O que é?

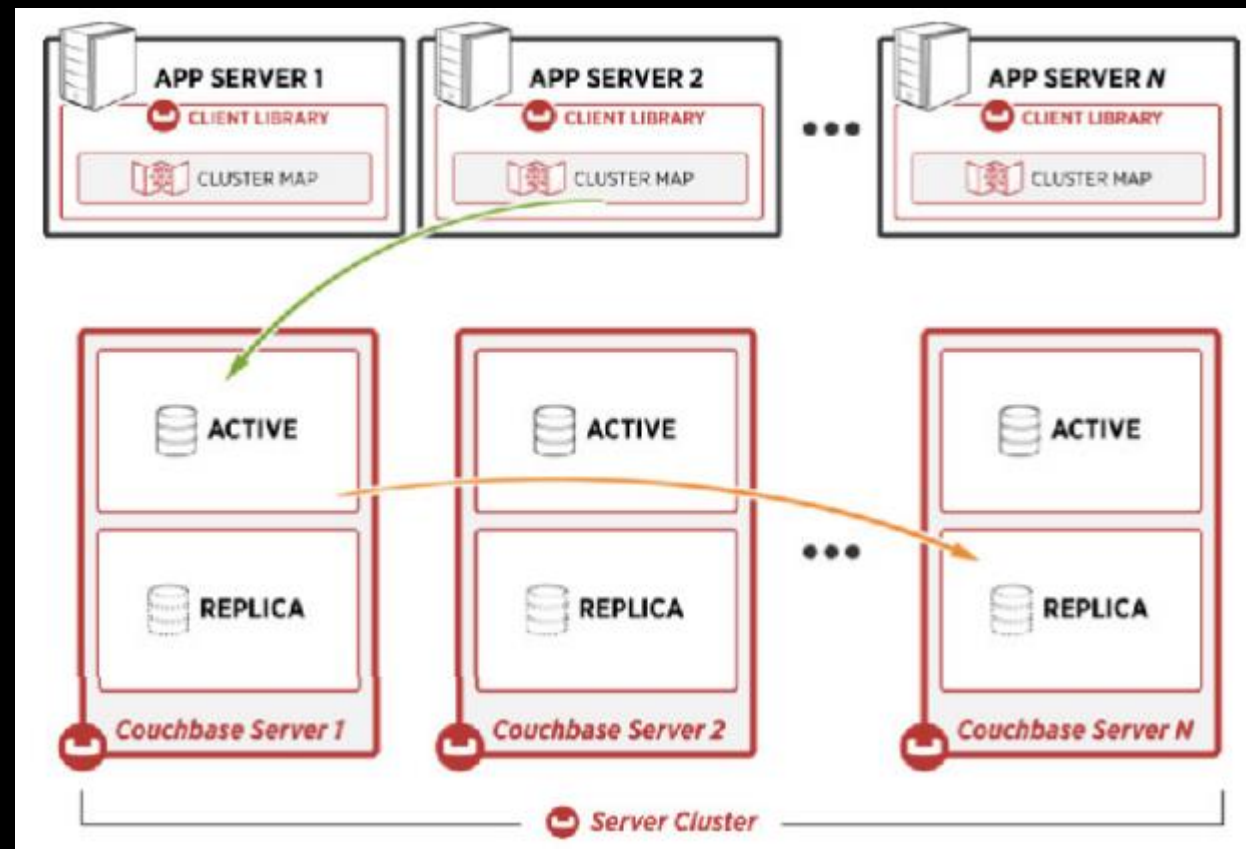
O particionamento de dados (sharding) serve para escalar consultas de dados. As gravações nas coleções de dados são feitas em qualquer nó que tenha a coleção e depois replicada para os demais.

Benefícios

Essa abordagem favorece aplicações que tenham mais operações de gravação, uma vez que mais de um nó do cluster pode responder ativamente a uma solicitação de gravação.

Limitações

Este modelo enfraquece a consistência de dados, uma vez que a replicação de uma informação do nó que fez a gravação até todos os demais nós que devem receber a réplica está sujeita a latência de rede e sucesso da operação de gravação.



45697056

Vantagens dos bancos de documentos

Armazenamento: diferentes de bancos relacionais, um campo que não é informado para um registro não ocupa armazenamento.

Flexibilidade de esquema: os documentos json armazenados podem conter qualquer conteúdo dentro de uma mesma coleção de dados

Agrupamento de dados: como não é relacional, os dados de uma mesma entidade são guardados juntos (ex: O documento do cliente pode conter todos os dados do cliente)

Chaves estrangeiras: Como não há relacionamento, não há necessidade de armazenar as chaves estrangeiras em várias tabelas

Quando usar um banco de documento

Casos de uso: aqui estão alguns casos de uso populares dos bancos de dados de document-based:

- ✓ Catálogo de produtos de comércio eletrônico;
- ✓ Blogs e gerenciamento de conteúdo;
- ✓ Gerenciamento de configurações;
- ✓ Manutenção de dados baseados em localização - dados geoespaciais;
- ✓ Sites de redes móveis e sociais;
- ✓ Objetivos fracamente acoplados - o design pode mudar ao longo do tempo;

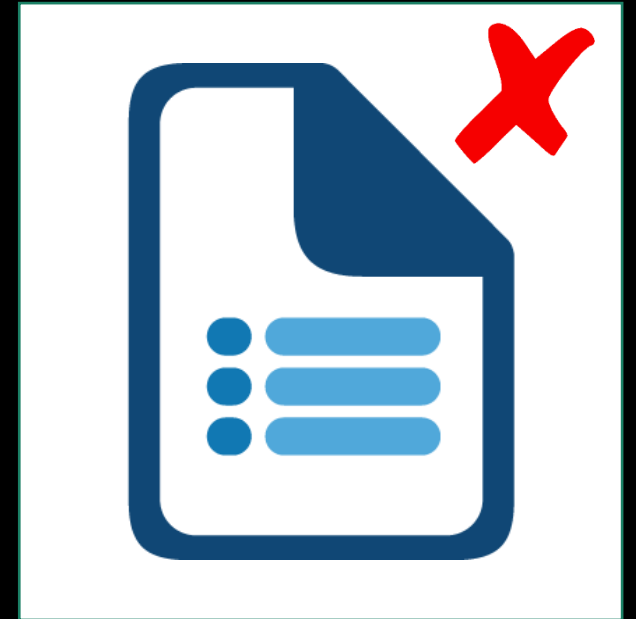


45697056

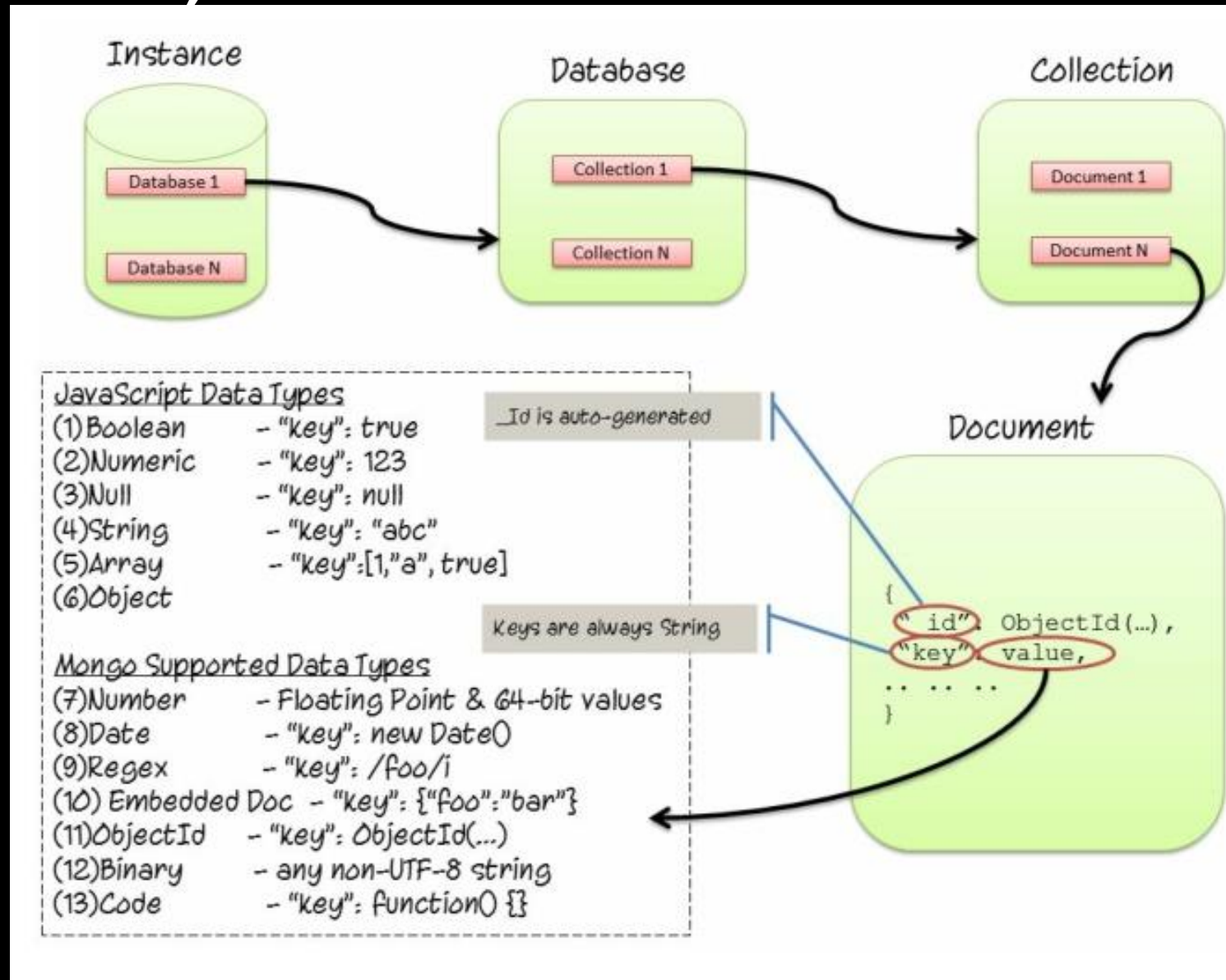
Quando **não** usar um banco de documento

No entanto, os bancos de dados document-based não são a escolha ideal para cada caso de uso quando:

- ❖ Sistemas altamente transacionais ou em que o modelo de dados é projetado antecipadamente;
- ❖ Sistemas fortemente acoplados;
- ❖ Não são a escolha certa se você precisar executar consultas de pesquisa complexas ou se o seu aplicativo exigir transações complexas com várias operações.



Database, Collections e Documents



Exercício:

Explorando um banco Mongodb



<https://www.mongodb.com/>

Exercício MongoDB



O Docker já está instalado e uma imagem chamada **mongo** já está disponível. Siga os passos iniciais abaixo:

Databases:

1. Criar um Container a partir da imagem
2. Verificar o diretório **/data/db**
3. Verificar os arquivos **/etc/mongod.config.orig** e **/ect/mongo.conf (*)**
4. Abra outra janela para verificar os logs do mongo e deixe-a aberta
5. Iniciar o serviço mongod
6. Acessar o CLI do MongoDB
7. Criar um database chamado **dbAula (*)**
8. Listar todos os databases da instância
9. Verificar em qual database está conectado

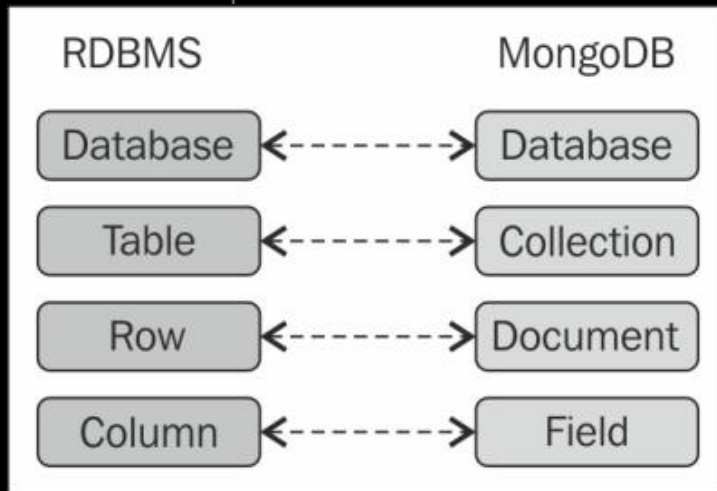
Script disponibilizado com os comandos

Arquivo de configurações do mongo. Veja a sessão de replication. Os comandos `rs.initiate()` e `rs.status` iniciam e verificam a replicação, respectivamente

A criação de um database se dá a partir do momento que algum documento é inserido

Continuação exercício MongoDB...

Collections



```
db.createCollection(name, options)
```



Parâmetro	Tipo	Descrição
Name	String	Nome da collection a ser criada
Options	Document	(Opcional) Especifica as opções de memória e índices

Campo	Tipo	Descrição
capped	Boolean	(Opcional) Se “true”, ativa uma coleção limitada. A coleção limitada é uma coleção de tamanho fixo que sobrescreve automaticamente suas entradas mais antigas quando atinge seu tamanho máximo. Se você especificar “true”, também precisará especificar o parâmetro de tamanho.
autoIndexId	Boolean	(Opcional) Se “true”, cria automaticamente o índice no campo _id.s O valor padrão é falso.
size	number	(Opcional) Especifica um tamanho máximo em bytes para uma coleção limitada. Se capped for “true”, você precisará especificar este campo também.
max	number	(Opcional) Especifica o número máximo de documentos permitidos na coleção limitada.

Continuação exercício MongoDB...



Collections

10. Criar uma collection chamada *collectionAuto*
11. Listar as collections do database
12. Inserir um documento na *collectionAuto* (*)
13. Consultar todos os registros da *collectionAuto*
14. Criar uma collection chamada *collectionCustom* (*)
15. Inserir um documento na *collectionCustom*
16. Inserir o mesmo documento duas vezes
17. Inserir 5 registros na *collectionCustom* (*)
18. Localizar 1 documento na *collectionCustom*
19. Importar registros usando o mongoimport
20. Consulte todos os registros de *collectionAuto*

O MongoDB é *case sensitive*, fique atento para letras maiúsculas e minúsculas. Um validador de json é bem-vindo

Máximo de 5 documentos

Os documentos inseridos anteriormente foram excluídos

Continuação exercício MongoDB...



Collections: Índices

- ✓ Índices aceleram as consultas, mas atrasam as gravações. Como em bancos colunares, devem ser usados com cuidado.
- ✓ A cada operação de manipulação nos dados, o banco precisa atualizar todos os índices da collections

21. Criar um índice na *collectionAuto* (*)
22. Criar um índice de texto na *collectionAuto* (*)
23. Criar um índice composto na *collectionAuto* (*)
24. Consultar todos os índices da *collectionAuto*
25. Usar o explain na consulta (*)
26. Fazer uma consulta com hint na *collectionAuto* (*)
27. Apagar um dos índices da *collectionAuto*
28. Apagar todos os índices da *collectionAuto*

O índice de campo simples indexa um campo customizado para pesquisas usando este campo. O 1 indica direção ascendente

Cria um índice para pesquisas de texto

Combina mais de um campo para indexar os valores consultados. Neste caso a ordenação fará diferença

O SGBD decide qual o melhor índice a usar. O explain exhibe qual será o(s) índice(s) usados(s)

O comando hint força o uso do índice desejado

Continuação exercício MongoDB...

Filtros

```
db.colName.find()  
db.colName.findOne()
```

A diferença é que o findOne retorna apenas o primeiro documento encontrado



Operação	Syntaxe	Exemplo	RDBMS Equivalente
Equality	{<key>:<value>}	db.mycol.find({"by":"tutorials point"}).pretty()	where by = 'tutorials point'
Less Than	{<key>:{\$lt:<value>}}	db.mycol.find({"likes":{\$lt:50}}).pretty()	where likes < 50
Less Than Equals	{<key>:{\$lte:<value>}}	db.mycol.find({"likes":{\$lte:50}}).pretty()	where likes <= 50
Greater Than	{<key>:{\$gt:<value>}}	db.mycol.find({"likes":{\$gt:50}}).pretty()	where likes > 50
Greater Than Equals	{<key>:{\$gte:<value>}}	db.mycol.find({"likes":{\$gte:50}}).pretty()	where likes >= 50
Not Equals	{<key>:{\$ne:<value>}}	db.mycol.find({"likes":{\$ne:50}}).pretty()	where likes != 50

Continuação exercício MongoDB...



Collections: Filtros

- 29. Fazer pesquisa com um valor exato
- 30. Fazer pesquisa com o operador LT
- 31. Fazer pesquisa com o operador GT
- 32. Fazer pesquisa com operador NE
- 33. Fazer pesquisa com operador AND
- 34. Fazer pesquisa com operador OR
- 35. Fazer pesquisa filtrando exibição de campos (*)
- 36. Fazer pesquisa com limite de documentos
- 37. Fazer pesquisa desconsiderando documentos
- 38. Fazer pesquisa com ordenação

O "1" e "0" servem para exibir ou ocultar, respectivamente

45697056

Continuação exercício MongoDB...

Collections: Aggregate



- 39. Fazer contagem de documentos
- 40. Fazer soma de documentos
- 41. Fazer um agrupamento

Expression	Description	Example
\$sum	Soma-se o valor definido a partir de todos os documentos da coleção.	db.mycol.aggregate([{\$group : {_id : "\$by_user", num_tutorial : {\$sum : "\$likes"}}}])
\$avg	Calcula a média de todos os valores dados a partir de todos os documentos da coleção.	db.mycol.aggregate([{\$group : {_id : "\$by_user", num_tutorial : {\$avg : "\$likes"}}}])
\$min	Obtém o mínimo dos valores correspondentes de todos os documentos da coleção.	db.mycol.aggregate([{\$group : {_id : "\$by_user", num_tutorial : {\$min : "\$likes"}}}])
\$max	Obtém o máximo dos valores correspondentes de todos os documentos da coleção.	db.mycol.aggregate([{\$group : {_id : "\$by_user", num_tutorial : {\$max : "\$likes"}}}])
\$push	Insere o valor de uma matriz no documento resultante.	db.mycol.aggregate([{\$group : {_id : "\$by_user", url : {\$push : "\$url"}}}])
\$addToSet	Insere o valor de uma matriz no documento resultante, mas não cria duplicatas.	db.mycol.aggregate([{\$group : {_id : "\$by_user", url : {\$addToSet : "\$url"}}}])
\$first	Obtém o primeiro documento dos documentos de origem de acordo com o agrupamento. Normalmente, isso só faz sentido em conjunto com alguns anteriormente aplicada "\$sort"-stage.	db.mycol.aggregate([{\$group : {_id : "\$by_user", first_url : {\$first : "\$url"}}}])
\$last	Obtém o último documento dos documentos de origem de acordo com o agrupamento. Normalmente, isso só faz sentido em conjunto com alguns anteriormente aplicada "\$sort"-stage.	db.mycol.aggregate([{\$group : {_id : "\$by_user", last_url : {\$last : "\$url"}}}])

Continuação exercício MongoDB...



Collections: Update/Delete

```
>db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
>db.COLLECTION_NAME.remove(DELETION_CRITERIA)
```

- 42. Atualizar um documento na collection *calcados*
- 43. Atualizar mais de um documento na collection *calcados* (*)
- 44. Remover documentos usando filtro na collection *calcados*
- 45. Remover todos os registros da collection *calcados*

Se não informar o parâmetro **multi: true**, o mongodb irá atualizar somente o primeiro documento

Desafios

1. Atualizar um documento que não existe
2. Exportar dados de uma collection com o mongoexport
3. Fazer agregações diferentes com diferentes consolidações
4. Verificar o diretório */data/db*
5. Excluir o database *dbAula*

`db.dropDatabase()`

Atenção com o comando, pois não especifica o database. Considera o database em uso para exclusão.

Bancos de dados chave-valor

Definição de banco chave-valor

+

Bancos chave-valor são bancos que armazenam valores indexados a uma chave de pesquisa única e somente através desta chave os dados podem ser recuperados. Portanto, não é possível fazer pesquisa através do conteúdo dos dados.

Os valores podem conter textos em formato JSON, listas e diferentes outros formatos.

Sua principal característica é a excepcional performance para consulta e manipulação dos dados e é considerado o mais simples dos NoSQL

DB-Engines Ranking of Key-value Stores

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

This is a partial list of the [complete ranking](#) showing only key-value stores.

Read more about the [method](#) of calculating the scores.



☐ include secondary database models

68 systems in ranking, March 2023

Rank			DBMS	Database Model	Score		
Mar 2023	Feb 2023	Mar 2022			Mar 2023	Feb 2023	Mar 2022
1.	1.	1.	Redis +	Key-value, Multi-model i	172.45	-1.39	-4.31
2.	2.	2.	Amazon DynamoDB +	Multi-model i	80.77	+1.08	-1.03
3.	3.	3.	Microsoft Azure Cosmos DB +	Multi-model i	36.10	-0.40	-4.79
4.	4.	4.	Memcached	Key-value	22.61	-0.56	-3.06
5.	5.	6. ↑	Hazelcast	Key-value, Multi-model i	8.63	-0.48	-1.43
6.	6.	5. ↓	etcd	Key-value	8.60	+0.09	-3.23
7.	7.	10. ↑	Aerospike +	Multi-model i	6.54	-0.02	+0.31
8.	8.	9. ↑	Ehcache	Key-value	6.03	-0.01	-0.52
9.	9.	14. ↑	Google Cloud Bigtable	Multi-model i	5.44	-0.47	+1.10
10.	10.	8. ↓	Ignite	Multi-model i	5.29	-0.21	-1.58
11.	12. ↑	7. ↓	Riak KV	Key-value	5.28	+0.14	-1.75
12.	11. ↓	11. ↓	ArangoDB +	Multi-model i	5.04	-0.26	-0.57
13.	15. ↑	15. ↑	RocksDB +	Key-value	4.57	+0.11	+0.51

Quando usar um banco chave-valor

- ✓ Quando seu aplicativo precisa lidar com muitas pequenas leituras e gravações contínuas, isso pode ser volátil. Os bancos de dados de valor-chave oferecem acesso rápido à memória.
- ✓ Ao armazenar informações básicas, como detalhes do cliente; armazenar páginas da web com o URL como chave e a página da web como valor; armazenar o conteúdo do carrinho de compras, categorias de produtos, detalhes de produtos de comércio eletrônico
- ✓ Para aplicações que não requerem atualizações frequentes ou precisam suportar consultas complexas.

Banco chave-valor: ReDis

Remote Dictionary Server

O Redis é um banco de dados NoSQL que segue o princípio do armazenamento de chave-valor.

O Redis suporta vários tipos de estruturas de dados, como strings, hashes, listas, conjuntos, conjuntos de classificação, bitmaps, hiperloglogs e índices geoespaciais com consultas radius

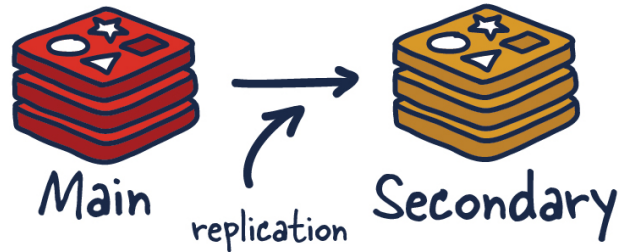


Arquiteturas do redis

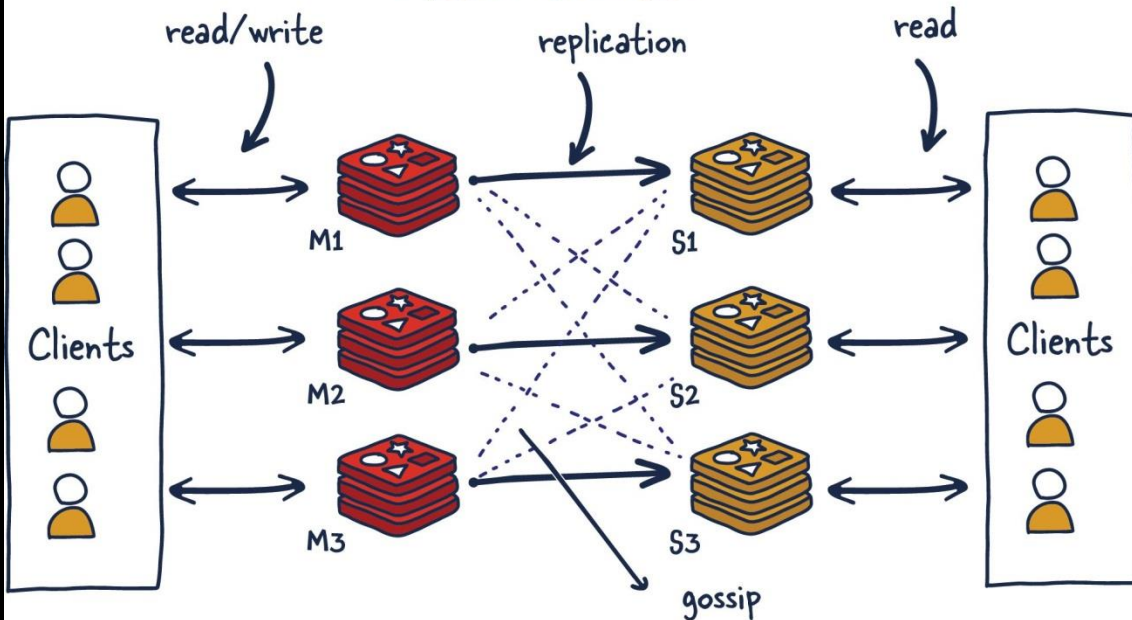
Simple Database



HA Database

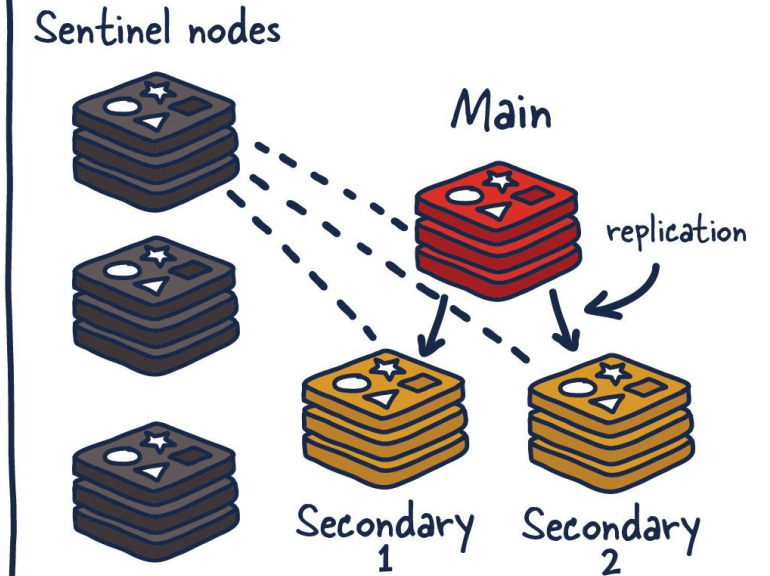


Redis Cluster



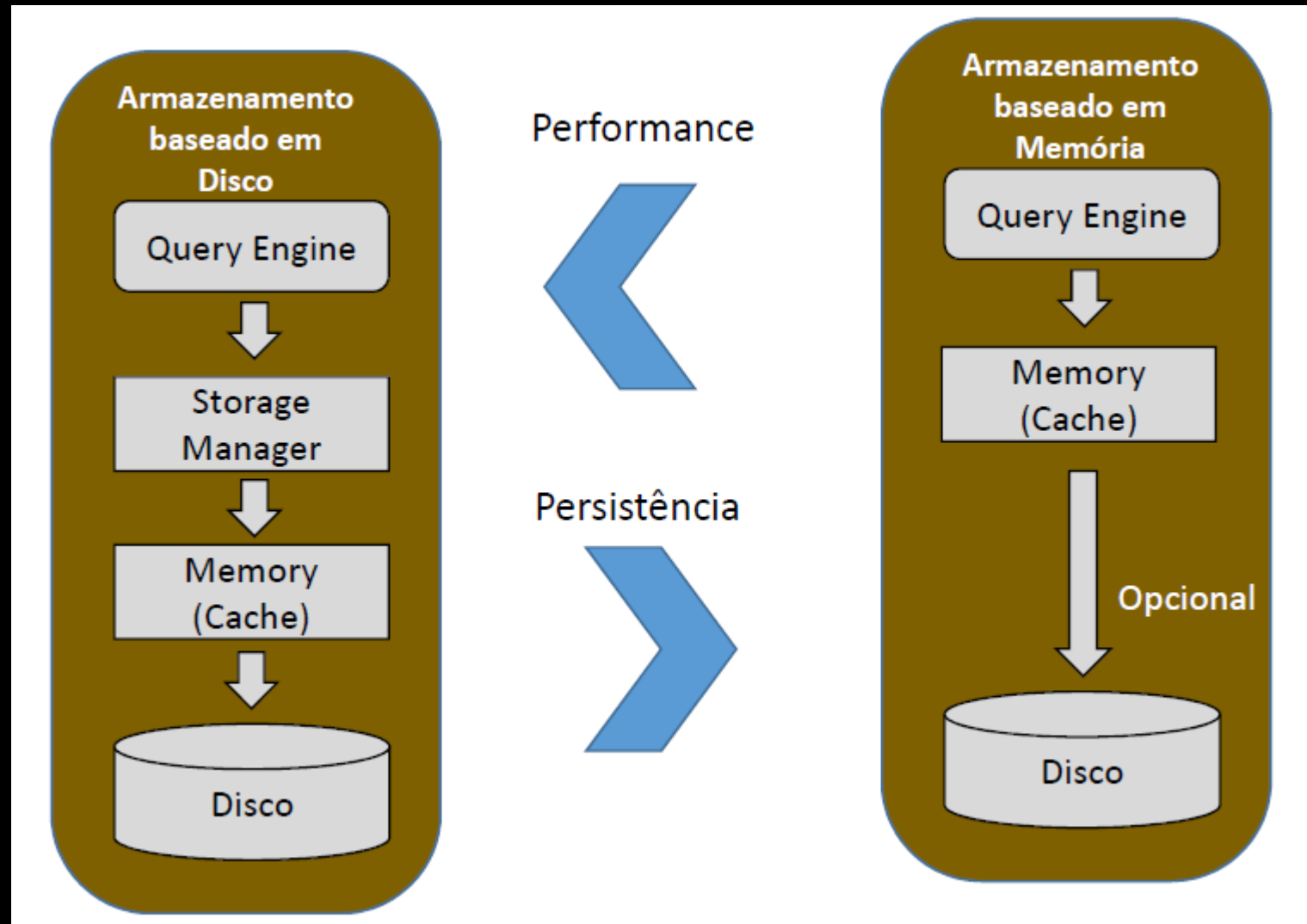
Gossip é a checagem constante entre os nós que, em caso de galha, promove um nó secundário para primário

Redis sentinel



Sentinel é um serviço distribuído que avalia a resposta de cada nó e elege um nó como principal, direcionando novas conexões para o mesmo.

Armazenamento de dado



Tipos de dados Redis

Strings: são o tipo de dados Redis mais básico, representando uma sequência de bytes

Lists: listas de strings classificadas por ordem de inserção

Sets: coleções não ordenadas de strings únicas

Hashes: tipos de registro modelados como coleções de pares campo-valor

Sorted sets: coleções de strings únicas que mantêm a ordem pela pontuação associada a cada string

Streams: uma estrutura de dados que age como um log apenas de inclusão (*append only*)

Geospatial indexes: úteis para encontrar locais dentro de um determinado raio geográfico

Bitmaps: permite executar operações bit a bit em strings

Bitfields: codificam com eficiência vários contadores em um valor de string. Bitfields fornecem operações atômicas de obtenção, definição e incremento e oferecem suporte a diferentes políticas de estouro

HyperLogLog: estruturas de dados fornecem estimativas probabilísticas da cardinalidade de grandes conjuntos

Chaves Redis

As chaves no ReDis são **Binary-Safe Strings**;

Significa que você pode usar qualquer sequência binária como uma chave, de uma string como "foo" ao conteúdo de um arquivo JPEG.

Regras:

- ✓ Chaves muito longas não são uma boa ideia;
- ✓ Chaves muito curtas não costumam ser uma boa ideia;
- ✓ Tente ficar com um esquema. Por exemplo, "object-type";
- ✓ O tamanho máximo permitido da chave é de 512 MB.

45697056

Strings no Redis

Strings;

O tipo String é o tipo mais simples de valor que você pode associar a uma chave.

Como as chaves Redis são strings, quando usamos o tipo string como um valor também, estamos mapeando uma string para outra string. O tipo de dados da string é útil para vários casos de uso, como o armazenamento em cache de fragmentos ou páginas HTML.

Exercício:

Explorando um banco redis



Exercício Redis



O Docker já está instalado e uma imagem chamada **redis** já está disponível. Siga os passos iniciais abaixo:

Script disponibilizado com os comandos

Comandos básicos:

1. Criar um container a partir da imagem baixada
2. Iniciar o server do redis
3. Executar comando para ver configurações
4. Criar chave com comando SET (*)
5. Recuperar chave com comando GET
6. Criar chave com expiração
7. Criar chave usando NX e XX
8. Alterar chave com SETRANGE (*)

SET key value [expiration EX seconds | PX milliseconds] [NX | XX]

EX - define o tempo de expiração especificado, em segundos.

PX - define o tempo de expiração especificado, em milissegundos.

NX - define a chave somente se ela ainda não existir.

XX - somente configure a chave, se já existir.

SETRANGE key offset value

offset – posição numérica de onde será iniciada a alteração de string.

value – string de substituição

Continuação exercício Redis...



Mais alguns comandos...

9. Alterar chave com comando APPEND (*)
10. Criar chave bitmap com setbit (*)
11. Consultar chave bitmap com getbit
12. Incrementar numérico com INCR (*)
13. Incrementar numérico com INCRBY (*)
14. Forçar a expiração de uma chave (*)

APPEND key value

SET key offset value

offset – posição numérica de onde o flag será ligado ou desligado.

value – valor do bit (0 | 1)

INCR key

INCRBY key value

expire key seconds

Continuação exercício Redis...

Mais comandos para chaves

Comando	Descrição
DEL key	Este comando exclui a chave, se ela existir.
DUMP key	Este comando retorna uma versão serializada do valor armazenado na chave especificada.
EXISTS key	Este comando verifica se a chave existe ou não.
EXPIRE key seconds	Define a expiração da chave após o tempo especificado.
EXPIREAT key timestamp	Define a expiração da chave após o tempo especificado. Aqui, a hora está no formato Unix timestamp.
PEXPIRE key milliseconds	Define a expiração da chave em milissegundos.
PEXPIREAT key milliseconds-timestamp	Define a expiração da chave no timestamp Unix especificado como milissegundos.
KEYS pattern	Localiza todas as chaves correspondentes ao padrão especificado.
MOVE key db	Move uma chave para outro banco de dados.
PERSIST key	Remove a expiração da chave.
PTTL key	Obtém o tempo restante na expiração das chaves em milissegundos.
TTL key	Obtém o tempo restante na expiração das chaves.
RANDOMKEY	Retorna uma chave aleatória do Redis.
RENAME key newkey	Altera o nome da chave.
RENAMENX key newkey	Renomeia a chave, se uma nova chave não existir.
TYPE key	Retorna o tipo de dados do valor armazenado na chave.

Continuação exercício Redis...



Hashes...

Redis Hashes são mapas entre os campos de string e os valores de string.

No Redis, cada hash pode armazenar até mais de 4 bilhões de pares de valor de campo.

15. Definir um conjunto de campos (*)

HMSET key field1 value1 [field2 value2]

16. Consultar um conjunto de campos (*)

HMGET key field1 [field2]

17. Consultar todos os campos do Hash (*)

HGETALL key

Continuação exercício Redis...



Mais comandos para Hashes

Comando	Descrição
HDEL key field2 [field2]	Exclui um ou mais campos de hash.
HEXISTS key field	Determina se um campo hash existe ou não.
HGET key field	Obtém o valor de um campo hash armazenado na chave especificada.
HGETALL key	Obtém todos os campos e valores armazenados em um hash na chave especificada
HINCRBY key field increment	Incrementa o valor inteiro de um campo hash pelo número fornecido
HINCRBYFLOAT key field increment	Incrementa o valor flutuante de um campo hash pelo valor especificado
HKEYS key	Obtém todos os campos em um hash
HLEN key	Obtém o número de campos em um hash
HMGET key field1 [field2]	Obtém os valores de todos os campos de hash fornecidos
HMSET key field1 value1 [field2 value2]	Define vários campos de hash para vários valores
HSET key field value	Define o valor de string de um campo hash
HSETNX key field value	Define o valor de um campo de hash, somente se o campo não existir
HVALS key	Obtém todos os valores em um hash
HSCAN key cursor [MATCH pattern] [COUNT count]	Itera incrementalmente campos de hash e valores associados

Continuação exercício Redis...



Lists... +

Lists são simplesmente listas de strings, classificadas por ordem de inserção. Você pode adicionar elementos nas listas do Redis no início ou no final da lista. Armazena mais de 4 bilhões de elementos por lista

18. Criar lista com mais de um item (*)

LPUSH key value1 [value2]

19. Adicionar itens a lista

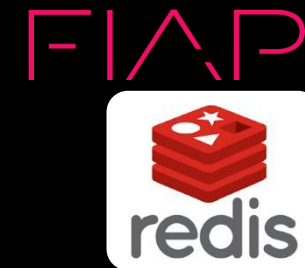
RPOP key

20. Remover um item do final (*)

LLEN key

21. Verificar o tamanho da lista (*)

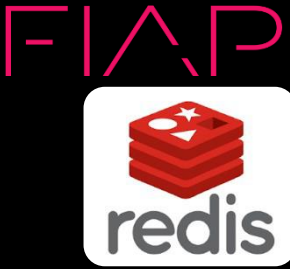
Continuação exercício Redis...



Mais comandos para Listas

Comando	Descrição
BLPOP key1 [key2] timeout	Remove e obtém o primeiro elemento em uma lista ou bloqueia até que um esteja disponível
BRPOP key1 [key2] timeout	Remove e obtém o último elemento em uma lista ou bloqueia até que um esteja disponível
BRPOPLPUSH source destination timeout	Extraí um valor de uma lista, empurra-o para outra lista e o retorna; ou bloqueia até que um esteja disponível
LINDEX key index	Obtém um elemento de uma lista por seu índice
LINSERT key BEFORE AFTER pivot value	Insere um elemento antes ou depois de outro elemento em uma lista
LLEN key	Obtém o comprimento de uma lista
LPOP key	Remove e obtém o primeiro elemento em uma lista
LPUSH key value1 [value2]	Adiciona um ou vários valores a uma lista
LPUSHX key value	Anexa um valor a uma lista, somente se a lista existir
LRANGE key start stop	Obtém um intervalo de elementos de uma lista
LREM key count value	Remove elementos de uma lista
LSET key index value	Define o valor de um elemento em uma lista por seu índice
LTRIM key start stop	Corta uma lista para o intervalo especificado
RPOP key	Remove e obtém o último elemento de uma lista
RPOPLPUSH source destination	Remove o último elemento de uma lista, acrescenta-o a outra lista e devolve-o
R PUSH key value1 [value2]	Anexa um ou vários valores a uma lista
RPUSHX key value	Anexa um valor a uma lista, somente se a lista existir

Continuação exercício Redis...



Sets... +

coleções não ordenadas de strings únicas

22. Criar coleção com mais de um item (*)

23. Consultar coleção (*)

24. Adicionar itens a coleção

25. Verificar se um valor está na coleção (*)

26. Remover itens da coleção (*)

SADD key member1 [member2]

SMEMBERS key

SISMEMBER key member

SREM key member1 [member2]

45697056

Continuação exercício Redis...

Mais comandos para Sets

Comando	Descrição
SADD key member1 [member2]	Adiciona um ou mais membros a um conjunto
SCARD key	Obtém o número de membros em um conjunto
SDIFF key1 [key2]	Subtrai vários conjuntos
SDIFFSTORE destination key1 [key2]	Subtrai vários conjuntos e armazena o conjunto resultante em uma chave
SINTER key1 [key2]	Intersecciona vários conjuntos
SINTERSTORE destination key1 [key2]	Intersecciona vários conjuntos e armazena o conjunto resultante em uma chave
SISMEMBER key member	Determina se um determinado valor é membro de um conjunto
SMEMBERS key	Obtém todos os membros em um conjunto
SMOVE source destination member	Move um membro de um conjunto para outro
SPOP key	Remove e retorna um membro aleatório de um conjunto
SRANDMEMBER key [count]	Obtém um ou vários membros aleatórios de um conjunto
SREM key member1 [member2]	Remove um ou mais membros de um conjunto
SUNION key1 [key2]	Adiciona vários conjuntos
SUNIONSTORE destination key1 [key2]	Adiciona vários conjuntos e armazena o conjunto resultante em uma chave
SSCAN key cursor [MATCH pattern] [COUNT count]	Itera incrementalmente os elementos do conjunto

Bancos de dados de grafos

Funcionamento de bancos de grafos

Um banco de dados gráfico armazena nós e relacionamentos em vez de tabelas ou documentos.

Os dados são armazenados da mesma forma que você esboça ideias em um quadro branco.

Não há restrição a um modelo pré-definido, permitindo uma forma muito flexível de os pensar e utilizar.

DB-Engines Ranking of Graph DBMS

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

This is a partial list of the [complete ranking](#) showing only graph DBMS.

Read more about the [method](#) of calculating the scores.

☐ include secondary database models

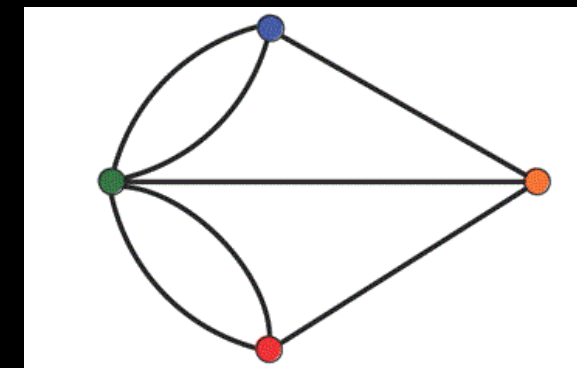
39 systems in ranking, March 2023

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	36.10	-0.40	-4.79
3.	3.	4.	Virtuoso	Multi-model	6.39	+0.29	+0.82
4.	4.	3.	ArangoDB	Multi-model	5.04	-0.26	-0.57
5.	5.	5.	OrientDB	Multi-model	4.30	-0.24	-0.63
6.	7.	7.	Amazon Neptune	Multi-model	2.60	-0.13	-0.09
7.	6.	8.	JanusGraph	Graph	2.56	-0.24	+0.09
8.	8.	6.	GraphDB	Multi-model	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

Teoria dos grafos

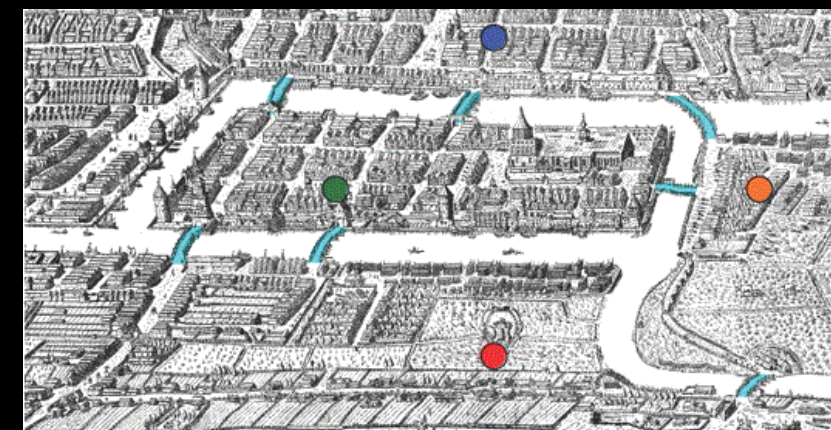
1793

Leonard Euler, através do mito das pontes de Königsberg, comprovou através de grafos que não era possível resolver essa façanha;



A ideia de Grafo surgiu independente das diversas áreas de conhecimento, no entanto é considerada como uma área da matemática aplicada. A mais antiga menção sobre o assunto ocorreu no trabalho de **Euler** (pronuncia-se Óiler), no ano de 1736 para modelar e explicar um problema chamado “**Pontes de Königsberg**”. A teoria dos grafos surgiu pela primeira vez em 1736 por Leonhard Euler para resolver um problema chamado de as 7 pontes de Königsberg (atual Kaliningrado). Seis delas interligavam duas ilhas às margens do Rio Pregel e uma que fazia a ligação entre as duas ilhas. O problema consistia na seguinte questão: **como seria possível fazer um passeio a pé pela cidade de forma a se passar uma única vez por cada uma das sete pontes e retornar ao ponto de partida?**

Euler focou apenas no problema removendo todos os detalhes geométricos do mesmo (distância, tamanho das pontes, formas etc) e apresentou uma solução que além de provar que não era possível passar apenas uma vez por cada ponte, como a forma de solução poderia ser aplicada a outros problemas semelhantes. Assim surgiu a teoria dos grafos.

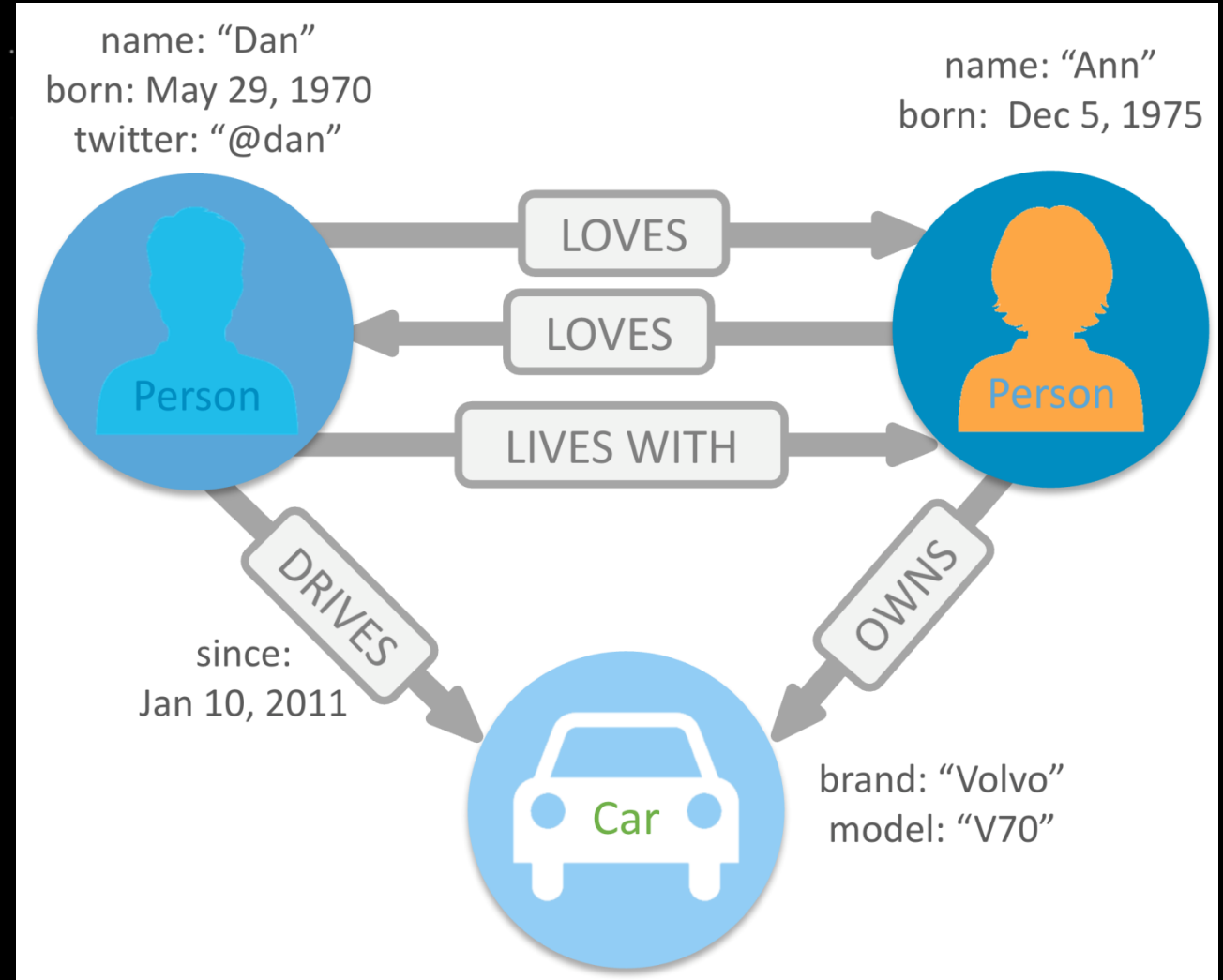


Banco de grafos: Neo4j

+

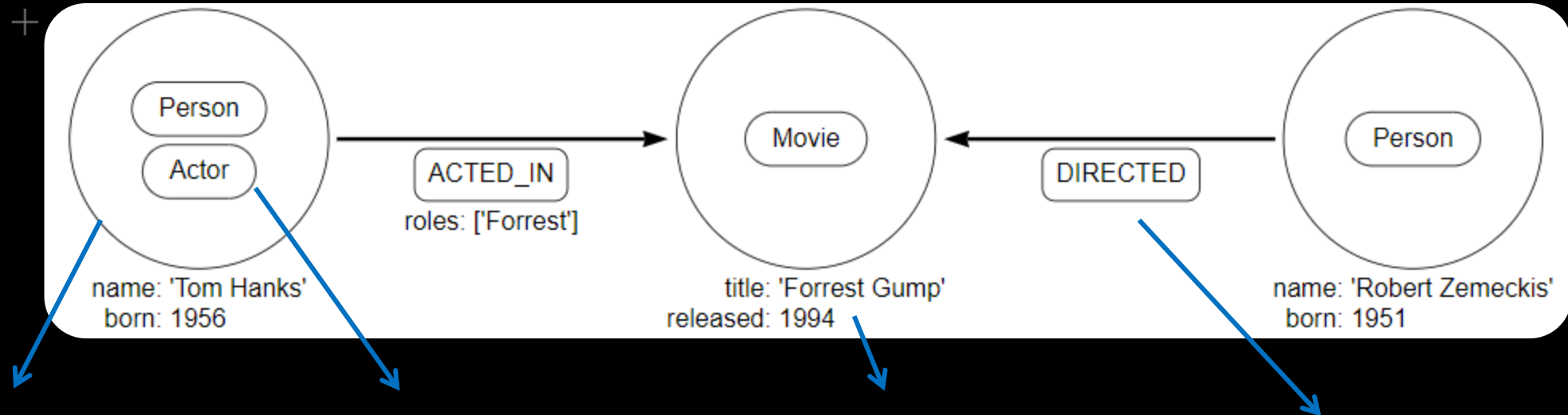
O **Neo4j** é um banco de dados Open Source baseado no conceito NoSQL (Banco de Dados que não utiliza os conceitos estruturados).

As informações não são armazenadas em tabelas, mas sim na forma de Grafos e suas estruturas são representadas de forma que o conhecimento é representado pelos conceitos matemáticos da Teoria de Grafos.



<https://neo4j.com/developer/graph-database/>

Neo4j



NÓ

- ✓ São os objetos de um domínio
- ✓ Podem ter zero ou mais **rótulos** (labels)
- ✓ Podem ter zero ou mais **propriedades**
- ✓ Podem ter zero ou mais **relacionamentos**

RÓTULO (Label)

- ✓ Definem domínios através do agrupamentos de nós.
- ✓ Podem ser adicionados e removidos livremente

PROPRIEDADES

- ✓ Pares de chave-valor usados para armazenar dados sobre os nós
- ✓ Podem ser numéricos strings ou booleanos

RELACIONAMENTO

- ✓ São conexões do tipo origem-destino entre os nós
- ✓ Podem ligar um nó a ele próprio
- ✓ Podem possuir propriedades

Características do Neo4j

- ✓ **Dados não estruturados;**
- ✓ **Usa o armazenamento gráfico nativo com o GPE nativo (Graph Processing Engine);**
- ✓ **JSON e XLS support;**
Suporta a exportação de dados de consulta para o formato JSON e XLS;
- ✓ **Ele fornece a API REST para ser acessada por qualquer linguagem de programação, como Java, Spring, Scala etc.**
- ✓ **Schema-less e Type-less;**
- ✓ **Suporte Full a ACID;**
A integridade dos dados por meio de transações compatíveis com ACID.

Características do Neo4j

- ✓ **Free-for-all**
Não adiciona constraints;
- ✓ **Suporta 34.4 BI de nodes e 34.4BI de relationships;**
- ✓ **Não faz sharding de subgrafos, limitando o tamanho dos grafos;**
- ✓ **Opções de Cluster HA com o uso da versão Enterprise;**
Gerenciamento de memória fora do heap, Causal Clustering, que otimiza tanto o acesso somente leitura quanto o acesso de leitura / gravação, alta disponibilidade (HA), recuperação de desastre e suporte a vários data centers.
- ✓ **Cypher - Linguagem (Query) para Neo4j;**
É uma linguagem de consulta legível por humanos projetada especificamente para manipular dados gráficos conectados.

Aplicações de bancos de grafos

- ✓ Sistemas de recomendações;
- ✓ Usado para Business Intelligence;
- ✓ Sistema Geoespaciais;
- ✓ O mundo é baseado em relacionamentos;
- ✓ Os dados e suas estruturas são dinâmicos

Exercício:

Explorando um banco neo4j



Ref.: <https://neo4j.com/docs/cypher-manual/current/introduction/>

Exercício Neo4j



O Docker já está instalado e uma imagem chamada *neo4j* já está disponível. Siga os passos iniciais abaixo:

Script disponibilizado com os comandos

Comandos básicos:

1. Criar um container a partir da imagem publicando a porta 7474
2. Listar os comandos do neo4j e neo4j-admin
3. Iniciar o serviço do neo4j
4. Verifique os diretórios configurados
5. Verificar o status do servidor neo4j
6. Acessar o console em localhost:7474 (*)
7. Definir uma nova senha "fiap5dts" (*)

Usuário: neo4j Senha: neo4j

A tela inicial será aberta

Continuação exercício Neo4j...



Interface gráfica

Editor

The screenshot displays the Neo4j Browser interface. On the left, a sidebar contains 'Database Information', 'Node Labels' (Football_Team, Hobby, Person, University), 'Relationship Types' (A_FAN_OF, FRIENDS, GRADUATED_FROM, LIKES, WORKS_FOR), and 'Property Keys' (Friends_since, current_occupation, hobby_type, label, name, name_of_Team, started_since, university_name). The main area shows a graph with nodes like 'University', 'Kurie Adolabi', 'Elizabeth Adewale', 'Patricia Oke', 'Football', 'Cycling', 'Mechest', and 'Jackson Oke'. Relationships include 'GRADUATED_FROM', 'FRIENDS', 'LIKES', 'A_FAN_OF', and 'WORKS_FOR'. A Cypher query is entered in the editor: `$ MATCH (d)-[b]->(n)-[r]->(v) RETURN d,b,n,r,v`. The results show counts for various relationship types. A tooltip for 'Patricia Oke' shows properties: `Person <id>: 44 names Patricia Oke`.

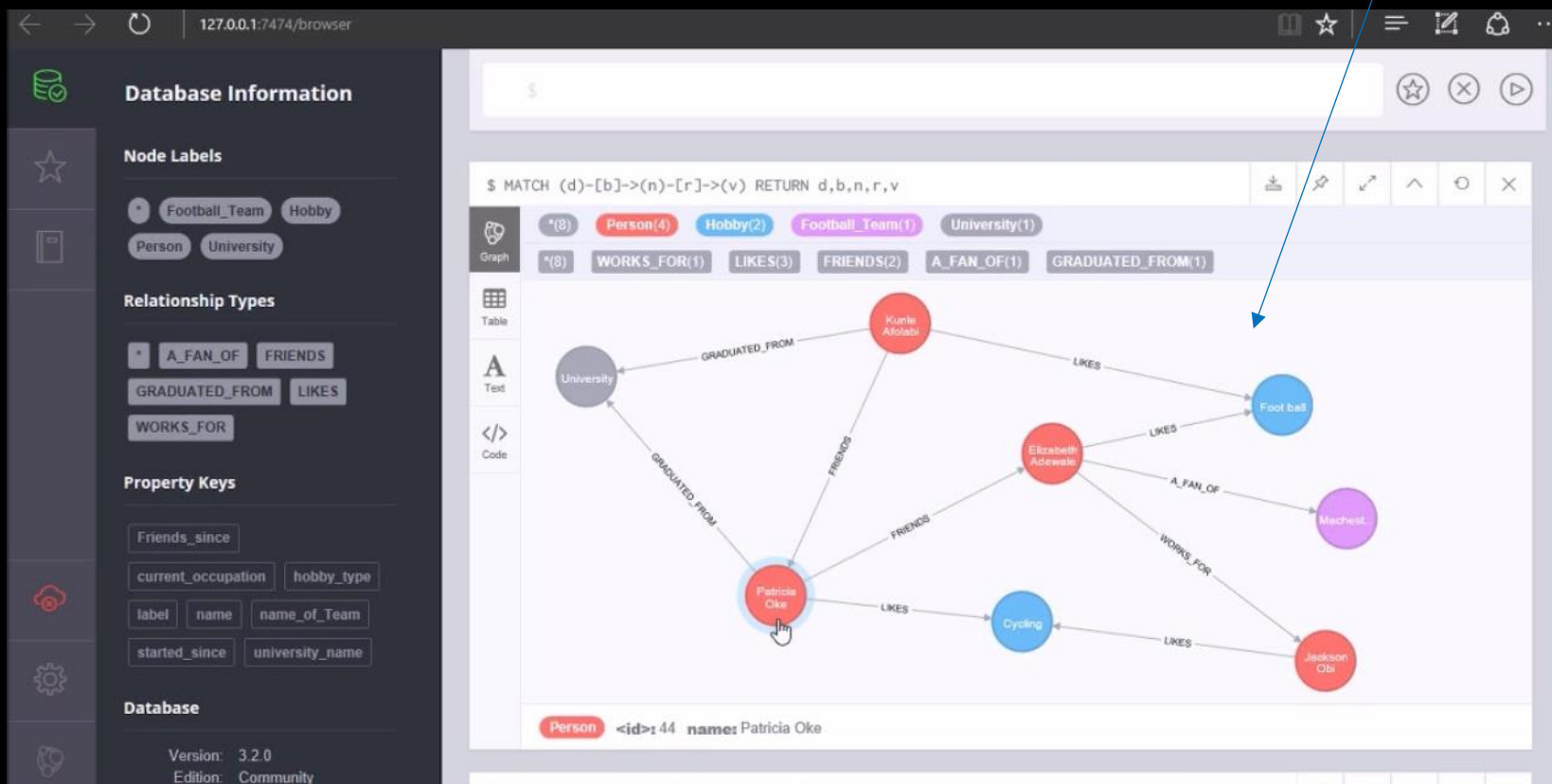
O editor é a interface principal para entrar e executar comandos. Insira as consultas do Cypher para trabalhar com dados do gráfico. Use comandos do lado do cliente como: ajuda para outras operações.

- ✓ Edição de linha única para breves consultas ou comandos
- ✓ Mudar para edição multi-linha com <shift-enter>
- ✓ Execute uma consulta com <ctrl-enter>
- ✓ O histórico é mantido para recuperar facilmente os comandos anteriores

Continuação exercício Neo4j...



Interface gráfica



Um quadro de resultado é criado para cada execução de comando, adicionado à parte superior do fluxo para criar uma coleção rolável em ordem cronológica inversa.

- ✓ Quadros especiais como visualização de dados
- ✓ Expandir um quadro para tela cheia
- ✓ Remover um quadro específico do fluxo
- ✓ Limpe o fluxo com o comando **:clear**

Continuação exercício Neo4j...



Interface gráfica

The screenshot shows the Neo4j Browser interface. On the left is a sidebar with 'Database Information', 'Node Labels' (Football_Team, Hobby, Person, University), 'Relationship Types' (A_FAN_OF, FRIENDS, GRADUATED_FROM, LIKES, WORKS_FOR), 'Property Keys' (Friends_since, current_occupation, hobby_type, label, name, name_of_Team, started_since, university_name), and 'Database' (Version: 3.2.0, Edition: Community). The main area displays a Cypher query: `$ MATCH (u:Usuário), (m:ModuloSistema) return *`. Below the query, a table shows the query details: Server version (Neo4j/3.4.0), Server address (localhost:7687), Query (MATCH (u:Usuário), (m:ModuloSistema) return *), Summary ({"statement": {"text": "MATCH (u:Usuário), (m:ModuloSistema)"}}, and Response (["keys": [...]. A blue arrow points to the 'Code' tab in the left sidebar, which is labeled 'Code'.


A guia de código exibe tudo enviado e recebido do servidor Neo4j, incluindo:


- ✓ Solicitar URI, método HTTP e cabeçalhos
- ✓ Código e cabeçalhos de resposta HTTP de resposta
- ✓ Pedido bruto e conteúdo da resposta no formato JSON


Continuação exercício Neo4j...


Interface gráfica

```
$ MATCH (u:Usuário), (m:ModuloSistema) return *
```


Graph


Table


Text


Code

"m"	"u"
{"Nome": "Compras"}	{"Nome": "Usuário 1", "Id": 1}
{"Nome": "Financeiro"}	{"Nome": "Usuário 1", "Id": 1}
{"Nome": "Compras"}	{"Nome": "Usuário 2", "Id": 2}
{"Nome": "Financeiro"}	{"Nome": "Usuário 2", "Id": 2}

```
$ MATCH (u:Usuário), (m:ModuloSistema) return *
```


Graph


Table


Text


Code

```
{
  "Nome": "Compras"
}
```

```
{
  "Nome": "Usuário 1",
  "Id": 1
}
```

```
{
  "Nome": "Financeiro"
}
```

```
{
  "Nome": "Usuário 1",
  "Id": 1
}
```

```
{
  "Nome": "Compras"
}
```

```
{
  "Nome": "Usuário 2",
  "Id": 2
}
```

Started streaming 4 records after 3 ms and completed after 8 ms.

Continuação exercício Neo4j...



Interface gráfica

The screenshot shows the Neo4j Browser interface. On the left is a sidebar with sections: Database Information, Node Labels (Football_Team, Hobby, Person, University), Relationship Types (A_FAN_OF, FRIENDS, GRADUATED_FROM, LIKES, WORKS_FOR), Property Keys (Friends_since, current_occupation, hobby_type, label, name, name_of_Team, started_since, university_name), and Database (Version: 3.2.0, Edition: Community). The main area displays a graph visualization with nodes like University, Karim Adewale, Elizabeth Adewale, Patricia Oke, Cycling, Foot ball, Mchael, and Jackson Obi, connected by relationships like GRADUATED_FROM, FRIENDS, LIKES, and WORKS_FOR. A query bar at the top shows a Cypher query: `$ MATCH (d)-[b]->(n)-[r]->(v) RETURN d,b,n,r,v`. A sidebar label 'Sidebar' with an arrow points to the left sidebar.

A barra lateral se expande para revelar diferentes painéis funcionais para consultas e informações comuns.

- ✓ Metadados do banco de dados e informações básicas
- ✓ Scripts salvos organizados em pastas
- ✓ Links de informações para documentos e referência
- ✓ Créditos e informações de licenciamento

Continuação exercício Neo4j...



Comandos administrativos...

8. Listar databases
9. Listar usuários
10. Criar um usuário
11. Contar os nós do database

Continuação exercício Neo4j...



Criando nós, rótulos, propriedades e relacionamentos...

Cole as 2 linhas e use
ctrl+enter para executar

12. Criar 2 nós para pilotos de F1(*)
13. Criar 1 nó para equipe de F1
14. Consultar o resultado
15. Criar 2 nós para pilotos de F1
16. Criar 1 nó para equipe de F1
17. Consultar o resultado
18. Criar os relacionamentos entre os pilotos de 1 equipe
19. Consultar o resultado
20. Criar os relacionamentos entre os pilotos de 1 equipe
21. Consultar os resultado

45697056

Desafios

1. Conectar com o usuário fiap criado
2. Contar os nós do database neo4j
3. Listar o usuário corrente conectado
4. Conectar com o usuário neo4j
5. Apagar o usuário fiap


MBA⁺

Copyright © **2023** Profs. Ivan Gancev e Leandro Mendes

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, dos Professores Ivan Gancev e Leandro Mendes

profivan.gancev@fiap.com.br

profleandro.mendes@fiap.com.br

