Universidade Federal da Paraíba – Campus I
Centro de Informática
Departamento de Informática

Big Data: conceitos e aplicações

Laboratório: Modelo de Dados Chave-valor

SETUP DO AMBIENTE

1) Instalar Redis para ter acesso ao redis-cli

https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-redis-on-ubuntu-20-04-quickstart-pt

https://redis.io/docs/getting-started/installation/install-redis-on-linux/

```
curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/redis-archive-keyring.gpg

echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg]
https://packages.redis.io/deb $(lsb_release -cs) main" | sudo tee
/etc/apt/sources.list.d/redis.list

sudo apt-get update

sudo apt-get install redis
```

2) Criar cluster no Redis.com, database e fazer conexão

A. Criar uma instância no Redis Cloud (https://redis.io/try-free/).
B. Obter as credenciais de conexão.
C. Testar a conexão:
   redis-cli -h <HOST> -p <PORT> -a <PASSWORD> ping
   Se retornar PONG, a conexão está funcionando corretamente.
D. Instalar as bibliotecas no Python:
   pip install redis json

https://redis.com/

https://docs.redis.com/latest/rc/rc-quickstart/

https://docs.redis.com/latest/rs/references/cli-utilities/redis-cli/

Ver comandos em: https://redis.io/docs/data-types/tutorial/

EXERCÍCIOS

1. Chaves

1. Crie a chave "simples" com o valor "um valor", e então cheque para ver se o valor foi inserido corretamente.
2. Crie 3 usuários fictícios com as chaves formatadas como "usuario:identificador", com o identificador começando em 1, depois 2, etc...
3. Liste todas as chaves já criadas.
4. Liste somente as chaves de usuários.
5. Crie mais dois usuários, com o fim do identificador sendo os números 11 e 21.
6. Liste todas as chaves de usuário com identificadores terminando em 1.
7. Crie um usuario com o identificador terminando em 1 e cujo número esteja entre 20 e 39.
8. Liste todas as chaves de usuário com identificador terminando em 1 e que esteja entre 20 e 30.
9. Mude a chave usuario:1 para usuario:01.
10. Liste todas as chaves terminadas em 1.
11. Remova o usuário com identificador 3.
12. Configure o usuario 2 para que sua chave expire em 10 segundos.

2. String

1. Crie uma chave chamada "animal" com o valor "gato".
2. Mude o valor da chave "animal" de "gato" para "peixe", retornando o valor original.
3. Acrescente "-espada" ao valor da chave "animal".
4. Retorne a substring "espada".
5. Retorne a substring "peixe".
6. Defina uma nova string chamada "count" com o valor 0.
7. Incremente "count" em 1.
8. Incremente "count" em 10.
9. Retorne o comprimento da string "count".

3. Hashes

1. Primeiramente, cheque se existe a chave "cachorros" na hash "animais".
2. Crie a hash "animais" e adicione "cachorros" com o valor 25 associado a ele.
3. Adicione "gatos" com o valor de 37 associado.
4. Adicione "peixes" com o valor de 28 associado.
5. Retorne todo o conteúdo (chaves e valores) da hash "animais".
6. Retorne o valor somente das chaves "cachorros" e "gatos".
7. Em um único comando, crie uma hash chamada "arvores" e adicione "palmeira" com o valor 81, "pinheiro" com o valor 23, e "baobá" com o valor 1.
8. Mostre somente os valores da hash "arvores".
9. Mostre somente as chaves da hash "arvores".
10. Corrija o valor de "pinheiro" para 12 e verifique.

4. Listas

1. Adicione o valor "Verificar correio" ao final da lista "tarefas", mas usando o comando que o adiciona somente se a lista já existir.
2. Repita o passo anterior, mas use a versão do comando que não verifica a existência.
3. Anexe o valor "Abrir correio" ao final da lista.
4. Adicione "Iniciar sistema" ao início da lista.
5. Obter o tamanho da lista.
6. Retorne o conteúdo de toda a lista.
7. Faça com que a primeira entrada na lista seja "Abrir o correio".
8. Usando um único comando, examine as primeiras 2 entradas na lista.
9. Remova a primeira entrada na lista.
10. Examine o valor da segunda entrada na lista.
11. Retorne toda a lista.

5. Sets

1. Adicione "maçãs", "laranjas" e "bananas" a um conjunto chamado "cesta:1".
2. Liste os membros da "cesta:1".
3. Adicione "abacaxis", "bananas" e "laranjas" a um conjunto "cesta:2".
4. Verifique os membros do "cesta:2".
5. Obtenha a interseção dos 2 conjuntos.
6. Mova "abacaxis" de "cesta:2" para "cesta:1", e verifique listando os membros.
7. Armazene a união dos 2 conjuntos em um novo conjunto chamado "allbaskets".
8. Remova "laranjas" de "cesta:1".

6. Sorted Sets

1. Adicione os seguintes membros e scores ao conjunto ordenado "equipe:1": Joao 69, Leonardo

67, Sergio 70, Sandra 68, Ana 70 e Maria 73.
2. Retorne o número de membros no conjunto.
3. Obtenha uma contagem do número de membros com scores entre 70 e 75.
4. Obtenha os membros com scores entre 65 e 70.
5. Retorne a classificação da Sandra no time.
6. Retorne, por ordem decrescente, os membros com scores entre 65 e 69.

7. TTL

1. Primeiro, usando a CLI adicione a chave "quote: 221" com um valor de "94.23" para Redis.
2. Recupere o TTL atual da chave usando o comando "TTL". Note que, uma vez que um TTL ainda não foi configurado na chave, Redis retorna -1.
3. Agora, defina a chave para expirar após 30 segundos. Antes de passarem 30 segundos, use o comando TTL novamente para obter o tempo restante antes da chave expirar.
4. Continue a pesquisar o tempo restante até que a chave expire,

8. Modelagem Básica

Criar um modelo de dados no Redis que relacione artigos e tags

UTILIZE: **HASH** para artigos e **SET** para tags

Para artigos temos:

- **article** - contém **name, description, filename, posting date**.

Mais tarde, podemos decidir adicionar mais campos, mas como o Redis não tem esquema, isso não será problema.

- Cada **article** pode ter zero, uma ou mais **tag**.
- Cada **tag** pode ter zero, um ou mais **article**.

O modelo deve ser equivalente ao modelo SQL abaixo:

```
CREATE TABLE articles(
        id      int not null primary key,
        name    varchar(100),
        `desc`  varchar(100),
        `file`  varchar(3),
        data    date
);

CREATE TABLE articles_tags(
        id      int not null,
        tag     varchar(100) not null,
        primary key(
                id,
                tag
        )
);

CREATE INDEX tag ON articles_tags(tag);
```

9. Adicione dados aos modelos de modo a relacioná-los

Como o Redis é chave/valor, precisaremos decidir que tipo de pesquisas precisaremos para:

10. Listar todos os artigos.
11. Listar um único artigo, além de listar suas tags.
12. Para determinada(s) tag(s), liste todos os artigos nela.

9. Modelagem Básica

Padrão 1: Embedded Pattern (1:1)
Objetivo: Modelar um produto e seus detalhes de forma embutida.

Exemplo de Adição de Dados:

```
import redis
import json

redis_client = redis.StrictRedis(host='<HOST>', port=<PORT>, password='<PASSWORD>',
decode_responses=True)

product = {
        "name": "Smartphone X",
        "price": 3999,
        "details": {
        "manufacturer": "TechCorp",
        "storage": "128GB",
        "color": "Black"
```

```
        }
}
```

```
redis_client.set("product:1", json.dumps(product))
```

Questões:

1. Como recuperar todas as informações do produto?
2. Como acessar apenas o nome e o preço do produto?

Padrão 2: Partial Embed Pattern (1:N Parcial)

Objetivo: Criar um produto com uma lista de avaliações recentes embutida.

Exemplo de Adição de Dados:

```
product_with_reviews = {
        "name": "Laptop Ultra",
        "price": 7999,
        "recent_reviews": [
        {"user": "Alice", "rating": 5, "comment": "Excelente desempenho!"},
        {"user": "Bob", "rating": 4, "comment": "Ótimo, mas bateria poderia ser melhor."}
        ]
}

redis_client.set("product:2", json.dumps(product_with_reviews))
```

Questões:

1. Como acessar apenas as avaliações recentes?
2. Como adicionar uma nova avaliação e manter no máximo 3 registros?

Padrão 3: Aggregate Pattern

Objetivo: Criar um controle de avaliações agregadas para otimizar a recuperação.

Exemplo de Adição de Dados:

```
redis_client.hset("product:1:ratings", mapping={"total_reviews": 2, "sum_ratings": 9})
```

Questões:

1. Como calcular a média das avaliações?
2. Como atualizar os valores ao adicionar uma nova avaliação?

Padrão 4: Polymorphic Pattern

Objetivo: Criar uma coleção de produtos de diferentes categorias.

Exemplo de Adição de Dados:

```
redis_client.set("product:3", json.dumps({"type": "game_console", "name": "GameBox", "storage": "1TB", "hdmi_ports": 2}))
redis_client.set("product:4", json.dumps({"type": "earbuds", "name": "SoundBuds", "battery_life": "10h", "connection_type": "Bluetooth"}))
```

Questões:

1. Como listar todos os produtos?
2. Como filtrar apenas os produtos do tipo "game_console"?

Padrão 5: Bucket Pattern

Objetivo: Criar um histórico de medições de temperatura agrupado em buckets.

Exemplo de Adição de Dados:

```
from datetime import datetime

redis_client.zadd("temperature:sensor:1", {str(datetime.now()): 22.5})
redis_client.zadd("temperature:sensor:1", {str(datetime.now()): 23.1})
```

Questões:

1. Como listar as últimas 5 medições?
2. Como calcular a temperatura média do dia?

Padrão 6: Tree and Graph Pattern

Objetivo: Criar um organograma de empresa modelado como grafo.

Exemplo de Adição de Dados:

```
redis_client.sadd("employee:1:reports_to", "employee:2", "employee:3")
redis_client.sadd("employee:2:reports_to", "employee:4")
```

Questões:

1. Como listar todos os subordinados do funcionário 1?
2. Como encontrar o gerente direto do funcionário 4?

Padrão 7: Revision Pattern

Objetivo: Criar um sistema de versões para documentos.

Exemplo de Adição de Dados:

```
post = {
        "title": "Guia Redis",
        "body": "Conteúdo inicial...",
        "revisions": [
        {"title": "Guia Redis", "body": "Conteúdo atualizado 1"},
        {"title": "Guia Redis", "body": "Conteúdo atualizado 2"}
        ]
}

redis_client.set("post:1", json.dumps(post))
```

Questões:

1. Como listar todas as revisões do documento?
2. Como adicionar uma nova revisão mantendo no máximo 5 versões?

Padrão 8: Schema Version Pattern

Objetivo: Criar um esquema que possa ser atualizado sem quebrar versões anteriores.

Exemplo de Adição de Dados:

```
user_v1 = {"schema": "1", "name": "Carlos", "email": "carlos@email.com"}
redis_client.set("user:1", json.dumps(user_v1))
```

Questões:

1. Como modificar o esquema para permitir múltiplos emails sem quebrar o modelo antigo?
2. Como verificar qual versão do esquema um usuário está usando?

Conclusão

Este laboratório abordou os 8 padrões de modelagem de dados no Redis, focando na população dos valores e deixando as consultas como desafios.

Próximos passos:
- Explorar RedisJSON e RedisGraph para consultas avançadas.
- Testar desempenho com conjuntos grandes de dados.
- Implementar consultas otimizadas utilizando índices e pipelines.