

Laboratorium 9

Model danych w bazie danych Redis

Cele

- Zapoznanie się z bazą danych Redis.
- Przygotowanie środowiska pracy.
- Opracowanie modelu danych typu klucz-wartość.
- Zapisanie danych w bazie Redis.
- Opracowanie metody odczytu danych z bazy danych.

Dane

- Dane pogodowe pochodzące z IMGW
- Dane astronomiczne
- Dane administracyjne

Program

- Napisz program, który zapisuje dane w bazie Redis z wykorzystaniem modelu klucz-wartość.
- Zmodyfikuj program liczący statystyki tak, aby wykorzystywał bazę Redis jako źródło danych.

Środowisko pracy

Instalacja Redis

Linux	https://redis.io/docs/latest/operate/oss_and_stack/install/install-redis/install-redis-on-linux/ <code>apt-get install redis-server</code> <code>yum install redis-server</code>
Mac	https://redis.io/docs/latest/operate/oss_and_stack/install/install-redis/install-redis-on-mac-os/ <code>brew install redis-server</code>
Windows	https://redis.io/docs/latest/operate/oss_and_stack/install/install-redis/install-redis-on-windows/ <code>apt-get install redis-server (Windows WSL)</code>

Aplikacja kliencka:

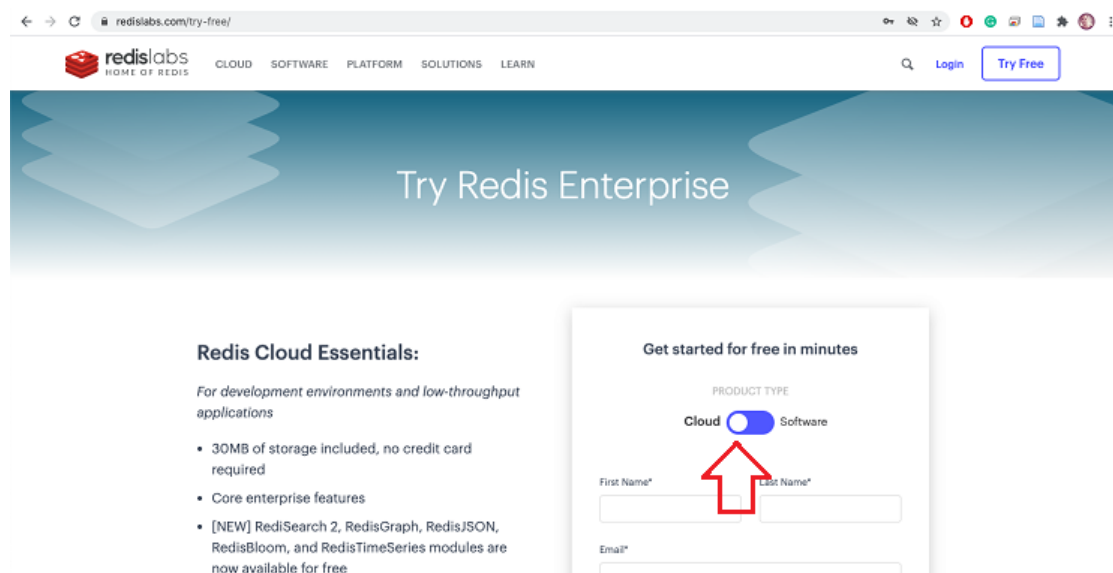
Redisinsight - <https://redis.com/redis-enterprise/redis-insight/>

Test połączenia z lokalną bazą danych

```
C:\test\redis-cli  
127.0.0.1:6379> ping  
PONG  
127.0.0.1:6379>
```

Dostęp do bazy w chmurze

Założenie konta (<https://redis.com/try-free/>):



redislabs.com/try-free/

redislabs HOME OF REDIS CLOUD SOFTWARE PLATFORM SOLUTIONS LEARN

Try Redis Enterprise

Redis Cloud Essentials:

For development environments and low-throughput applications

- 30MB of storage included, no credit card required
- Core enterprise features
- [NEW] RedisSearch 2, RedisGraph, RedisJSON, RedisBloom, and RedisTimeSeries modules are now available for free

Get started for free in minutes

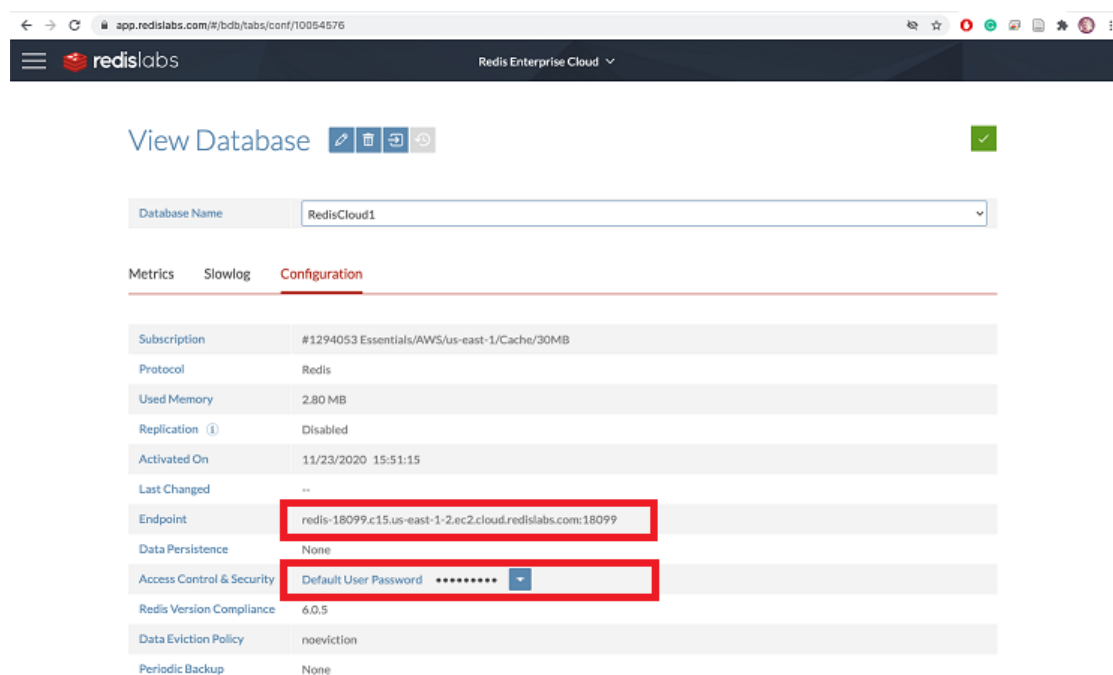
PRODUCT TYPE

Cloud ☒ Software

First Name* Last Name*

Email*

Konfiguracja bazy danych



app.redislabs.com/#bdb/tabs/conf/10054576

redislabs Redis Enterprise Cloud

View Database

Database Name: RedisCloud1

Metrics Slowlog **Configuration**

Subscription	#1294053 Essentials/AWS/us-east-1/Cache/30MB
Protocol	Redis
Used Memory	2.80 MB
Replication	Disabled
Activated On	11/23/2020 15:51:15
Last Changed	...
Endpoint	redis-18099.c15.us-east-1-2.ec2.cloud.redislabs.com:18099
Data Persistence	None
Access Control & Security	Default User Password
Redis Version Compliance	6.0.5
Data Eviction Policy	noeviction
Periodic Backup	None

Oznaczone pola definiują parametry dostępu do bazy.

Podstawowe polecenia

set - zapisanie pary klucz-wartość

get - pobranie wartości dla podanego klucza

del - usunięcie podanych kluczy

dbsize - liczba kluczy w bazie

keys * - lista kluczy w bazie

Zapis i odczyt danych

```
127.0.0.1:6379> set imie "Anna"
"OK"
127.0.0.1:6379> set nazwisko "Kowalska"
"OK"
127.0.0.1:6379> get imie
"Anna"
127.0.0.1:6379> keys *
  1) "imie"
  2) "nazwisko"
127.0.0.1:6379> del imie nazwisko
(integer) 2
```

Nawiązanie połączenia z lokalną bazą Redis z poziomu aplikacji w języku Python

```
import redis
pool = redis.ConnectionPool(host='127.0.0.1', port=6379, db=0)
db = redis.Redis(connection_pool=pool)
```

Nawiązanie połączenia ze zdalną bazą Redis z poziomu aplikacji w języku Python

```
import redis
pool = redis.ConnectionPool(host='redis-18099.c15.us-east-1-
2.ec2.cloud.redislabs.com', port=18099, password='xxxxxxxxxxxxxxxx')
db = redis.Redis(connection_pool=pool)
```

Przepisanie modelu BDOT10k do bazy Redis

```
objectkey = 'PL.PZGIK.BDOT10k.BUCMA.04.1549'  
objectval = 'BUCM01, Ort, Trn, Eks, MA, 2015-02-10, 2015-02-10, 2012-11-2...'  
db.set(objectkey, objectval)
```

Wyświetlenie informacji o liczbie obiektów w bazie

```
print(db.dbsize())
```

Porównanie czasu wyszukania obiektu w pliku tekstowym z czasem wyszukania go przez bazę Redis

```
db.get('PL.PZGIK.BDOT10k.BUCMA.04.1549')
```

Podstawowe polecenia dotyczące obsługi wartości tekstowych

Podstawowe polecenia operujące na tekstach

set - zapisanie pary klucz-wartość

get - pobranie wartości dla podanego klucza

del - usunięcie podanych kluczy

Przykład w języku Python

```
import redis
import json

# Connect to the Redis database
r = redis.Redis(host='localhost', port=6379, db=0)

# Create a dictionary with dummy values
string1 = {
    'name': 'John Doe',
    'age': 30,
    'city': 'New York'
}

# Convert the dictionary to a JSON string
string1_json = json.dumps(string1)

# Save the dictionary to Redis
r.set('string1', string1_json)

# Read the dictionary back from Redis
string2_json = r.get('string1')

# Convert the JSON string back to a dictionary
string2 = json.loads(string2_json)

print(string2)
```

Podstawowe polecenia dotyczące obsługi list, tablic mieszających, zbiorów i zbiorów sortowanych.

Typy danych stosowane w Redis: <https://redis.io/topics/data-types-intro>

Wykaz wszystkich poleceń bazy Redis znajduje się na stronie: <https://redis.io/commands>

Pełna dokumentacja bazy Redis: <https://redis.io/documentation>

Biblioteka dostępu do bazy Redis z poziomu języka Python: <https://github.com/andymccurdy/redis-py>

Listy

Podstawowe polecenia operujące na listach

LPUSH - zapisanie wartości na początku listy

LPOP - pobranie wartości z początku listy

RPUSH - zapisanie wartości na końcu listy

RPOP - pobranie wartości z końca listy

LLEN – pobranie liczby elementów listy

LRANGE – pobranie kluczy w podanym zakresie

LTRIM – odrzucenie elementów listy spoza podanego zakresu

Zapis i odczyt danych typu lista

```
127.0.0.1:6379> LPUSH mojalista pierwszy
127.0.0.1:6379> LPUSH mojalista drugi
127.0.0.1:6379> RPUSH mojalista trzeci
127.0.0.1:6379> LRANGE mojalista 0 2
1) „drugi”
2) „pierwszy”
3) „trzeci”
127.0.0.1:6379> LPOP mojalista
„drugi”
```

```
key = 'mojedane'

db.rpush(key, 20)
db.rpush(key, 175)

db.lrange(key, 0, -1)
```

Tablice mieszające (hash tables)

Podstawowe polecenia operujące na tablicach mieszających

HSET – zapisanie pojedynczej pary klucz-wartość stanowiącej element tablicy mieszającej zapisanej w podanym kluczu

HGET – pobranie wartości dla podanego klucza element tablicy mieszającej zapisanej w podanym kluczu

HMSET - zapisanie wielu par klucz-wartość stanowiących element tablicy mieszającej zapisanej w podanym kluczu

HMGET - pobranie wartości dla podanych kluczy wielu elementów tablicy mieszającej zapisanej w podanym kluczu

HGETALL - pobranie całej tablicy mieszającej zapisanej w podanym kluczu

Zapis i odczyt danych typu tablica mieszająca

Przykład zapisu danych w postaci hash mapy:

klucz (identyfikator)	wartość	
klucz	klucz1	wartość1
	klucz2	wartość2
	klucz3	wartość3

```
127.0.0.1:6379> HSET klucz klucz1 wartość1
127.0.0.1:6379> HGET klucz klucz1
127.0.0.1:6379> HMSET klucz klucz2 wartość2 klucz3 wartość3
127.0.0.1:6379> HMGET klucz klucz1 klucz2
127.0.0.1:6379> HGETALL klucz
```


Przykład danych wejściowych BDOT10k:

```
{
  'klasa': 'OT_BUCM_A ',
  'gmlid': 'PL.PZGIK.BDOT10k.BUCMA.04.636',
  'idIIP': {
    'lokalnyId': '2CE79A9C-1876-6A1D-E053-CC2BA8C00F96',
    'przestrzenNazw': 'PL.PZGIK.994.BDOT10k',
    'wersjaId': '2015-02-10T00:00:00'
  },
  'czyObiektBD00': 'false',
  'x_kod': 'BUCM01',
  'x_skrKarto': {
    'nil': true,
    'nilReason': 'inapplicable'
  },
  'x_katDoklGeom': 'dokladny',
  'x_zrodloDanychG': 'ortofotomapa',
  'x_zrodloDanychA': 'pomiarTerenowy',
  'x_katIstnienia': 'eksploatowany',
  'x_rodzajReprGeom': 'maksymalnyZasieg',
  'x_uzytkownik': 'Uzytkownik04.xml',
  'x_aktualnoscG': '2015-02-10',
  'x_aktualnoscA': '2015-02-10',
  ...
}
```

Przykład zapisu danych BDOT10k w postaci hash mapy:

klucz (identyfikator)	wartość	
'PL.PZGIK.BDOT10k.BUCMA.04.636'	'lokalnyId'	'2CE79A9C-1876-6A1D-E053-CC2BA8C00F96'
	'przestrzenNazw'	'PL.PZGIK.994.BDOT10k'
	'wersjaId'	'2015-02-10T00:00:00'
	'czyObiektBD00'	'false'


```
# mojedane[wiek] = 20

key = 'mojedane'

db.hset(key, 'wiek', 20)

data = {"wiek":20, "wzrost":175, "waga":48}
db.hmset(key, data)

db.hgetall(key)
```

Przykład w języku Python

```
import redis
import json

# Connect to the Redis database
r = redis.Redis(host='localhost', port=6379, db=0)

# Create a dictionary with dummy values
dict1 = {
    'name': 'John Doe',
    'age': 30,
    'city': 'New York'
}

# Save the dictionary to Redis using hset
for key, value in dict1.items():
    r.hset('dict1', key, value)

# Read the dictionary back from Redis
dict2 = r.hgetall('dict1')

# Convert the retrieved values from bytes to strings
dict2 = {key: value for key, value in dict2.items()}

print(dict2)
```

Zbiory (sets)

Podstawowe polecenia operujące na zbiorach

SADD – dodanie element do zbioru zapisanego w podanym kluczu

SISMEMBER – test obecności podanej wartości w zbiorze zapisanym w podanym kluczu

SMEMBERS – pobranie wszystkich elementów zbioru zapisanego w podanym kluczu

SUNION – pobranie sumy wartości podanych zbiorów

SINTER - pobranie części wspólnej podanych zbiorów

SDIFF – pobranie różnicy podanych zbiorów

Zapis i odczyt danych typu zbiór

```
127.0.0.1:6379> SADD klucz element
127.0.0.1:6379> SISMEMBER klucz element
127.0.0.1:6379> SMEMBERS klucz
127.0.0.1:6379> SUNION klucz1 klucz2
127.0.0.1:6379> SINTER klucz1 klucz2
127.0.0.1:6379> SDIFF klucz1 klucz2
```

Zbiory sortowane (sorted sets)

Podstawowe polecenia operujące na zbiorach sortowanych

ZADD – dodanie wartości do zbioru wraz z podaniem wagi

ZRANGE – pobranie podzbioru zbioru sortowanego z podanego zakresu według wag

ZRANK – pobranie wagi dla podanej wartości w kluczu

Zapis i odczyt danych typu zbiór sortowany

```
127.0.0.1:6379> ZADD myzset 1 "one"
(integer) 1
127.0.0.1:6379> ZADD myzset 2 "two"
(integer) 1
127.0.0.1:6379> ZADD myzset 3 "three"
(integer) 1
127.0.0.1:6379> ZRANK myzset "three"
(integer) 2
127.0.0.1:6379> ZRANK myzset "four"
(nil)
```

HyperLogLog

Podstawowe polecenia operujące na strukturach HyperLogLog

PFADD – dodanie wartości do podanego klucza

PFCOUNT – pobranie przybliżonej liczby elementów dla podanego klucza

Zapis i odczyt danych typu HyperLogLog

```
127.0.0.1:6379> PFADD hll foo bar zap
(integer) 1
127.0.0.1:6379> PFADD hll zap zap zap
(integer) 0
127.0.0.1:6379> PFADD hll foo bar
(integer) 0
127.0.0.1:6379> PFCOUNT hll
(integer) 3
```

GeoSet

Podstawowe polecenia operujące na zbiorach GeoSet

GEOADD - dodanie obiektu opisanego współrzędnymi

GEOPOS - zwrócenie informacji o lokalizacji obiektu

GEOHASH - zwrócenie geohash obiektu

GEORADIUS - zwrócenie listy obiektów w zadanej odległości od podanego punktu

GEORADIUSBYMEMBER - zwrócenie listy obiektów w zadanej odległości od danego obiektu

GEODIST – zwrócenie wyliczonej odległości pomiędzy dwoma punktami

Zapis i odczyt danych typu GeoSet

```
GEOADD points 21.010792 52.220425 PW 126.990003 37.577783 Changdeokgung

GEODIST points PW Changdeokgung
"7742920.4626"

GEOHASH points PW Changdeokgung
1) "u3qcn47w4b0"
2) "wydmc8z63w0"
```

Odnosiniki

Dokumentacja bazy Redis	https://redis.io/documentation
Podręcznik "Redis in Action"	https://redislabs.com/redis-in-action/
Typy danych stosowane w Redis	https://redis.io/topics/data-types-intro
Wykaz wszystkich poleceń bazy Redis	https://redis.io/commands
Biblioteka dostępu do bazy Redis z poziomu języka Python	https://github.com/andymccurdy/redis-py