REDIS
you probably doesn’t know

by Michał Wawrzyniak

What is Redis?

Redis is an open source (BSD licensed), in-memory data structure store used as a database, cache, message broker, and streaming engine.

Redis provides data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs, geospatial indexes, and streams.

Redis has built-in replication, Lua scripting, LRU eviction, transactions, and different levels of on-disk persistence.

Redis basics

Redis basics

Get Redis

```
docker run --name my-redis -p 6379:6379 -d redis
```

```
sudo apt-get install redis
```

Redis basics

Get Redis

```
docker run --name my-redis -p 6379:6379 -d redis
```

```
sudo apt-get install redis
```

Connecting to Redis

Redis basics

Get Redis

```
docker run --name my-redis -p 6379:6379 -d redis
```

```
sudo apt-get install redis
```

Connecting to Redis

```
root@localhost:/# redis-cli ping
PONG
```

Redis basics

Get Redis

```
docker run --name my-redis -p 6379:6379 -d redis
```

```
sudo apt-get install redis
```

Connecting to Redis

```
root@localhost:/# redis-cli ping
PONG
```

```
root@localhost:/# redis-cli
```

Redis basics

Get Redis

```
docker run --name my-redis -p 6379:6379 -d redis
```

```
sudo apt-get install redis
```

Connecting to Redis

```
root@localhost:/# redis-cli ping
PONG
```

```
root@localhost:/# redis-cli
```

```
> SET pystok_edition 63
OK
```

Redis basics

Get Redis

```
docker run --name my-redis -p 6379:6379 -d redis
```

```
sudo apt-get install redis
```

Connecting to Redis

```
root@localhost:/# redis-cli ping
PONG
```

```
root@localhost:/# redis-cli
```

```
> SET pystok_edition 63
OK
```

```
> GET pystok_edition
"63"
```

Redis CLI (1)

Redis CLI (1)

```
> INCR pystok_edition
(integer) 64
```

Redis CLI (1)

```
> INCR pystok_edition
(integer) 64

> GET pystok_edition
"64"
```

Redis CLI (1)

```
> INCR pystok_edition
(integer) 64

> GET pystok_edition
"64"

> SET pystok:63:date "20-12-2023"
OK
```

Redis CLI (1)

```
> INCR pystok_edition
(integer) 64

> GET pystok_edition
"64"

> SET pystok:63:date "20-12-2023"
OK

> KEYS pystok*
1) "pystok:63:date"
2) "pystok_edition"
```

Redis CLI (1)

```
> INCR pystok_edition
(integer) 64

> GET pystok_edition
"64"

> SET pystok:63:date "20-12-2023"
OK

> KEYS pystok*
1) "pystok:63:date"
2) "pystok_edition"

> INCR pystok:63:date
(error) ERR value is not an integer or out of range
```

Redis CLI (2)

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1
```

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1

> TYPE pystok_edition
string
```

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1

> TYPE pystok_edition
string

> DEL pystok_edition
(integer) 1
```

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1

> TYPE pystok_edition
string

> DEL pystok_edition
(integer) 1

> EXISTS pystok_edition
(integer) 0
```

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1

> TYPE pystok_edition
string

> DEL pystok_edition
(integer) 1

> EXISTS pystok_edition
(integer) 0
```

```
> select 1
OK
```

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1

> TYPE pystok_edition
string

> DEL pystok_edition
(integer) 1

> EXISTS pystok_edition
(integer) 0
```

```
> select 1
OK

[1]> select 20
(error) ERR DB index is out of range
```

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1

> TYPE pystok_edition
string

> DEL pystok_edition
(integer) 1

> EXISTS pystok_edition
(integer) 0
```

```
> select 1
OK

[1]> select 20
(error) ERR DB index is out of range

[1]> KEYS *
(empty list or set)
```

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1

> TYPE pystok_edition
string

> DEL pystok_edition
(integer) 1

> EXISTS pystok_edition
(integer) 0
```

```
> select 1
OK

[1]> select 20
(error) ERR DB index is out of range

[1]> KEYS *
(empty list or set)

> FLUSHDB
OK
```

Redis CLI (2)

```
> EXISTS pystok_edition
(integer) 1

> TYPE pystok_edition
string

> DEL pystok_edition
(integer) 1

> EXISTS pystok_edition
(integer) 0
```

```
> select 1
OK

[1]> select 20
(error) ERR DB index is out of range

[1]> KEYS *
(empty list or set)

> FLUSHDB
OK

> FLUSHALL
OK
(0.81s)
```

redis-py

redis-py

Install redis-py

```
pip install redis
```

redis-py

Install redis-py

```
pip install redis
```

Connect to Redis

```
>>> import redis
>>> r = redis.Redis(host="localhost", port=6379, db=0)
>>> r.set("foo", "bar", 120)
True
>>> r.ttl("foo")
114
>>> r.get("foo")
b'bar'
>>> r.ttl("foo")
104
```

Lists

Lists

```
> LPUSH sentence "therefore"
(integer) 1
> LPUSH sentence "think"
(integer) 2
> RPUSH sentence "I"
(integer) 3
> LPUSH sentence "I"
(integer) 4
> RPUSH sentence "am"
(integer) 5
```

Lists

```
> LPUSH sentence "therefore"
(integer) 1
> LPUSH sentence "think"
(integer) 2
> RPUSH sentence "I"
(integer) 3
> LPUSH sentence "I"
(integer) 4
> RPUSH sentence "am"
(integer) 5
```

```
> LLEN sentence
(integer) 5
```

Lists

```
> LPUSH sentence "therefore"
(integer) 1
> LPUSH sentence "think"
(integer) 2
> RPUSH sentence "I"
(integer) 3
> LPUSH sentence "I"
(integer) 4
> RPUSH sentence "am"
(integer) 5
```

```
> LLEN sentence
(integer) 5

> LRANGE sentence 0 -1
1) "I"
2) "think"
3) "therefore"
4) "I"
5) "am"
```

List - Queue FIFO

List - Queue FIFO

```
> LPUSH user:1:tasks task1 task2 task3
(integer) 4
> LRANGE user:1:tasks 0 -1
1) "task4"
2) "task3"
3) "task2"
4) "task1"
```

List - Queue FIFO

```
> LPUSH user:1:tasks task1 task2 task3
(integer) 4
> LRANGE user:1:tasks 0 -1
1) "task4"
2) "task3"
3) "task2"
4) "task1"

> RPOP user:1:tasks 2
1) "task1"
2) "task2"
> LPUSH user:1:tasks task5
(integer) 4
```

List - Queue FIFO

```
> LPUSH user:1:tasks task1 task2 task3
(integer) 4
> LRANGE user:1:tasks 0 -1
1) "task4"
2) "task3"
3) "task2"
4) "task1"

> RPOP user:1:tasks 2
1) "task1"
2) "task2"
> LPUSH user:1:tasks task5
(integer) 4
```

```
> LRANGE user:1:tasks 0 -1
1) "task5"
2) "task4"
3) "task3"
```

Hashes

Hashes

```
> HSET car:1 brand VW model Passat mileage 425673 year 2013 price 40000
(integer) 5
```

Hashes

```
> HSET car:1 brand VW model Passat mileage 425673 year 2013 price 40000
(integer) 5

> HGET car:1 brand
"VW"
```

Hashes

```
> HSET car:1 brand VW model Passat mileage 425673 year 2013 price 40000
(integer) 5

> HGET car:1 brand
"VW"

> HMGET car:1 brand model
1) "VW"
2) "Passat"
```

Hashes

```
> HSET car:1 brand VW model Passat mileage 425673 year 2013 price 40000
(integer) 5

> HGET car:1 brand
"VW"

> HMGET car:1 brand model
1) "VW"
2) "Passat"

> HSET car:1 color red
(integer) 1
```

Hashes

```
> HSET car:1 brand VW model Passat mileage 425673 year 2013 price 40000
(integer) 5

> HGET car:1 brand
"VW"

> HMGET car:1 brand model
1) "VW"
2) "Passat"

> HSET car:1 color red
(integer) 1
```

```
> HKEYS car:1
1) "brand"
2) "model"
3) "mileage"
4) "year"
5) "price"
6) "color"
```

Hashes

Hashes

```
> HINCRBY car:1 mileage -230000
(integer) 195673
```

Hashes

```
> HINCRBY car:1 mileage -230000
(integer) 195673

> HINCRBY car:1 price 10000
(integer) 50000
```

Hashes

```
> HINCRBY car:1 mileage -230000
(integer) 195673

> HINCRBY car:1 price 10000
(integer) 50000
```

```
> HGETALL car:1
 1) "brand"
 2) "VW"
 3) "model"
 4) "Passsat"
 5) "mileage"
 6) "195673"
 7) "year"
 8) "2013"
 9) "price"
10) "50000"
11) "color"
12) "red"
```

Sets

Sets

```
> SADD manufacturer:bikes Honda Suzuki Yamaha Kawasaki
(integer) 4
> SADD manufacturer:trucks Volvo Isuzu Man Ford
(integer) 4
> SADD manufacturer:cars Suzuki Volvo Mazda BMW Ford Opel Honda
(integer) 7
```

Sets

```
> SADD manufacturer:bikes Honda Suzuki Yamaha Kawasaki
(integer) 4
> SADD manufacturer:trucks Volvo Isuzu Man Ford
(integer) 4
> SADD manufacturer:cars Suzuki Volvo Mazda BMW Ford Opel Honda
(integer) 7

> SISMEMBER manufacturer:bikes Honda
(integer) 1
```

Sets

```
> SADD manufacturer:bikes Honda Suzuki Yamaha Kawasaki
(integer) 4
> SADD manufacturer:trucks Volvo Isuzu Man Ford
(integer) 4
> SADD manufacturer:cars Suzuki Volvo Mazda BMW Ford Opel Honda
(integer) 7

> SISMEMBER manufacturer:bikes Honda
(integer) 1

> SISMEMBER manufacturer:bikes Toyota
(integer) 0
```

Sets

Sets

```
> SINTER manufacturer:trucks manufacturer:cars
1) "Ford"
2) "Volvo"
```

Sets

```
> SINTER manufacturer:trucks manufacturer:cars
1) "Ford"
2) "Volvo"

> SDIFFSTORE manufacturer:bikes_not_cars manufacturer:bikes manufacturer:cars
(integer) 2
```

Sets

```
> SINTER manufacturer:trucks manufacturer:cars
1) "Ford"
2) "Volvo"

> SDIFFSTORE manufacturer:bikes_not_cars manufacturer:bikes manufacturer:cars
(integer) 2

> SMEMBERS manufacturer:bikes_not_cars
1) "Yamaha"
2) "Kawasaki"
```

Sorted Sets

Sorted Sets

```
> ZADD trust_ranking 20 Donek 30 Mati 5 Jaro 35 Kosiniak 0 Zero 32 Kamysz
```

Sorted Sets

```
> ZADD trust_ranking 20 Donek 30 Mati 5 Jaro 35 Kosiniak 0 Zero 32 Kamysz

> ZREVRANGEBYSCORE trust_ranking +inf -inf WITHSCORES
```

Sorted Sets

```
> ZADD trust_ranking 20 Donek 30 Mati 5 Jaro 35 Kosiniak 0 Zero 32 Kamysz

> ZREVRANGEBYSCORE trust_ranking +inf -inf WITHSCORES

1) "Kosiniak"
2) "35"
3) "Kamysz"
4) "32"
5) "Mati"
6) "30"
7) "Donek"
8) "20"
9) "Jaro"
10) "5"
11) "Zero"
12) "0"
```

Sorted Sets

```
> ZADD trust_ranking 20 Donek 30 Mati 5 Jaro 35 Kosiniak 0 Zero 32 Kamysz

> ZREVRANGEBYSCORE trust_ranking +inf -inf WITHSCORES

1) "Kosiniak"
2) "35"
3) "Kamysz"
4) "32"
5) "Mati"
6) "30"
7) "Donek"
8) "20"
9) "Jaro"
10) "5"
11) "Zero"
12) "0"
```

```
> ZREMRANGEBYRANK trust_ranking 0 -3
```

Sorted Sets

```
> ZADD trust_ranking 20 Donek 30 Mati 5 Jaro 35 Kosiniak 0 Zero 32 Kamysz

> ZREVRANGEBYSCORE trust_ranking +inf -inf WITHSCORES

1) "Kosiniak"
2) "35"
3) "Kamysz"
4) "32"
5) "Mati"
6) "30"
7) "Donek"
8) "20"
9) "Jaro"
10) "5"
11) "Zero"
12) "0"
```

```
> ZREMRANGEBYRANK trust_ranking 0 -3

> ZREVRANGEBYSCORE trust_ranking
> +inf -inf WITHSCORES
1) "Kosiniak"
2) "35"
3) "Kamysz"
4) "32"
5) "Mati"
6) "30"
```

Sorted Sets - Sliding Window Rate Limiter

Sorted Sets - Sliding Window Rate Limiter

```
# create cache key for user
key = f"limiter:{username}:{window_size_ms}:{max_hits}"
# get current timestamp
dt = datetime.datetime.now()
timestamp = int(round(dt.timestamp() * 1000))
# prepare data
data = {f"{timestamp}-{random.random()}": timestamp}
# add key to sorted set
redis.zadd(key, data)
# remove items not in given time selection
redis.zremrangebyscore(key, 0, timestamp - window_size_ms)
hits = redis.zcard(key)

if hits > max_hits:
    raise RateLimitExceededException()
```

GEO geospatial

GEO geospatial

```
redis.geoadd("cities", [53.1276998, 23.0736664, "Bialystok"])
redis.geoadd("cities", [53.4043988, 22.7846956, "Monki"])
redis.geoadd("cities", [53.1692581,22.0376219, "Lomza"])
redis.geoadd("cities", [52.7320113,23.4982175, "Hajnowka"])
redis.geoadd("cities", [53.4056239,23.4870356, "Sokolka"])
```

GEO geospatial

```
redis.geoadd("cities", [53.1276998, 23.0736664, "Bialystok"])
redis.geoadd("cities", [53.4043988, 22.7846956, "Monki"])
redis.geoadd("cities", [53.1692581,22.0376219, "Lomza"])
redis.geoadd("cities", [52.7320113,23.4982175, "Hajnowka"])
redis.geoadd("cities", [53.4056239,23.4870356, "Sokolka"])

redis.geosearch(
    "cities",
    longitude=53.1276998,
    latitude=23.0736664,
    radius=50,
    unit="km",
)
```

GEO geospatial

```
redis.geoadd("cities", [53.1276998, 23.0736664, "Bialystok"])
redis.geoadd("cities", [53.4043988, 22.7846956, "Monki"])
redis.geoadd("cities", [53.1692581,22.0376219, "Lomza"])
redis.geoadd("cities", [52.7320113,23.4982175, "Hajnowka"])
redis.geoadd("cities", [53.4056239,23.4870356, "Sokolka"])

redis.geosearch(
    "cities",
    longitude=53.1276998,
    latitude=23.0736664,
    radius=50,
    unit="km",
)

[b'Monki', b'Bialystok']
```

GEO geospatial

```
redis.geodist("cities", "Bialystok", "Monki", "km")
42.8538
```

GEO geospatial

```
redis.geodist("cities", "Bialystok", "Monki", "km")
42.8538

redis.georadiusbymember(
    "cities",
    "Bialystok",
    100,
    "km",
    withdist=True,
    withcoord=True,
    sort="ASC"
)
```

GEO geospatial

```
redis.geodist("cities", "Bialystok", "Monki", "km")
42.8538

redis.georadiusbymember(
    "cities",
    "Bialystok",
    100,
    "km",
    withdist=True,
    withcoord=True,
    sort="ASC"
)

[[b'Bialystok', 0.0, (53.12770217657089, 23.073666834948263)],
 [b'Monki', 42.8538, (53.40439885854721, 22.784695949177824)],
 [b'Sokolka', 54.0391, (53.405621945858, 23.487036765776743)],
 [b'Hajnowka', 62.1619, (52.73201197385788, 23.498217420810583)]]
```

Pipelines and transactions

Pipelines and transactions

```
redis = redis.Redis("localhost", 6379)
# Use the pipeline() method to create a pipeline instance
pipe = redis.pipeline(transaction=True)
# The following SET commands are buffered
pipe.set("pystok_edition", 63)
pipe.set("pystok:63:date", "20-12-2023")
pipe.incr("pystok_edition")
# the EXECUTE call sends all buffered commands to the server, returning
# a list of responses, one for each command.
pipe.execute()
```

```
[True, True, 64]
```

Pipeline performance

Pipeline performance

```
incr_value = 100000
redis.set("incr_key", "0")
```

Pipeline performance

```
incr_value = 100000
redis.set("incr_key", "0")

for _ in range(incr_value):
    redis.incr("incr_key")
value = redis.get("incr_key")
```

Pipeline performance

```
incr_value = 100000
redis.set("incr_key", "0")

for _ in range(incr_value):
    redis.incr("incr_key")
value = redis.get("incr_key")
```

```
pipe = redis.pipeline()
for _ in range(incr_value):
    pipe.incr("incr_key")
pipe.get("incr_key")
value = pipe.execute()[-1]
```

Pipeline performance

```
incr_value = 100000
redis.set("incr_key", "0")

for _ in range(incr_value):
    redis.incr("incr_key")
value = redis.get("incr_key")
```

11.286101s

```
pipe = redis.pipeline()
for _ in range(incr_value):
    pipe.incr("incr_key")
pipe.get("incr_key")
value = pipe.execute()[-1]
```

0.993571s

Lua script

Lua script

```
-- Lua script to compare set if lower
-- key: the name a Redis string storing a number
-- new: a number which, if smaller, will replace the existing number
local key = KEYS[1]
local new = ARGV[1]
local current = redis.call('GET', key)
if (current == false) or
   (tonumber(new) < tonumber(current)) then
  redis.call('SET', key, new)
  return 1
else
  return 0
end
```

Lua script

Lua script

```
update_if_lower = redis.register_script(SCRIPT)
```

Lua script

```
update_if_lower = redis.register_script(SCRIPT)

redis.set("my_key", 10)
update_if_lower(keys=["my_key"], args=[str(15)])
redis.get("my_key")
b'10'
```

Lua script

```
update_if_lower = redis.register_script(SCRIPT)

redis.set("my_key", 10)
update_if_lower(keys=["my_key"], args=[str(15)])
redis.get("my_key")
b'10'

redis.set("my_key", 10)
update_if_lower(keys=["my_key"], args=[str(5)])
redis.get("my_key")
b'5'
```

Redis Stack

Redis Stack extends the core features of Redis OSS and provides a complete developer experience for debugging and more.

In addition to all of the features of Redis OSS, Redis Stack supports:

- Probabilistic data structures
- Queryable JSON documents
- Querying across hashes and JSON documents
- Time series data support (ingestion & querying), including full-text search

Redis JSON & search

Redis JSON

- Full support for the JSON standard
- A JSONPath syntax for selecting/updating elements inside documents
- Incremental indexing on JSON and hash documents
- Full-text search
- Aggregations
- Geospatial queries

Redis OM

```python
from datetime import date
from typing import Optional, List

from redis_om import (
    EmbeddedJsonModel,
    JsonModel,
    Field,
    Migrator,
)
```

```python
class Prelection(EmbeddedJsonModel):
    title: str
    description: Optional[str]
    author: str = Field(index=True)
    duration: int

class Event(JsonModel):
    edition: int = Field(index=True)
    date: date = Field(index=True)
    members: int = Field(index=True)
    prelections: List[Prelection]
    description = Field(
      index=True,
      full_text_search=True,
      default="")
```

Redis OM

```
Migrator().run()

event = Event(
    edition=63,
    date=datetime.date(2023, 12, 20),
    members=120, description="Pystok event", prelections=[
        Prelection(title="Django", author="John", duration=20),
        Prelection(title="Flask", author="Andrey", duration=25)
])
event.save()
Event(pk='01HHYM9TRWPA3R0Q3HEWCY6NBQ', edition=63, ...

assert Event.get(event.pk) == event
```

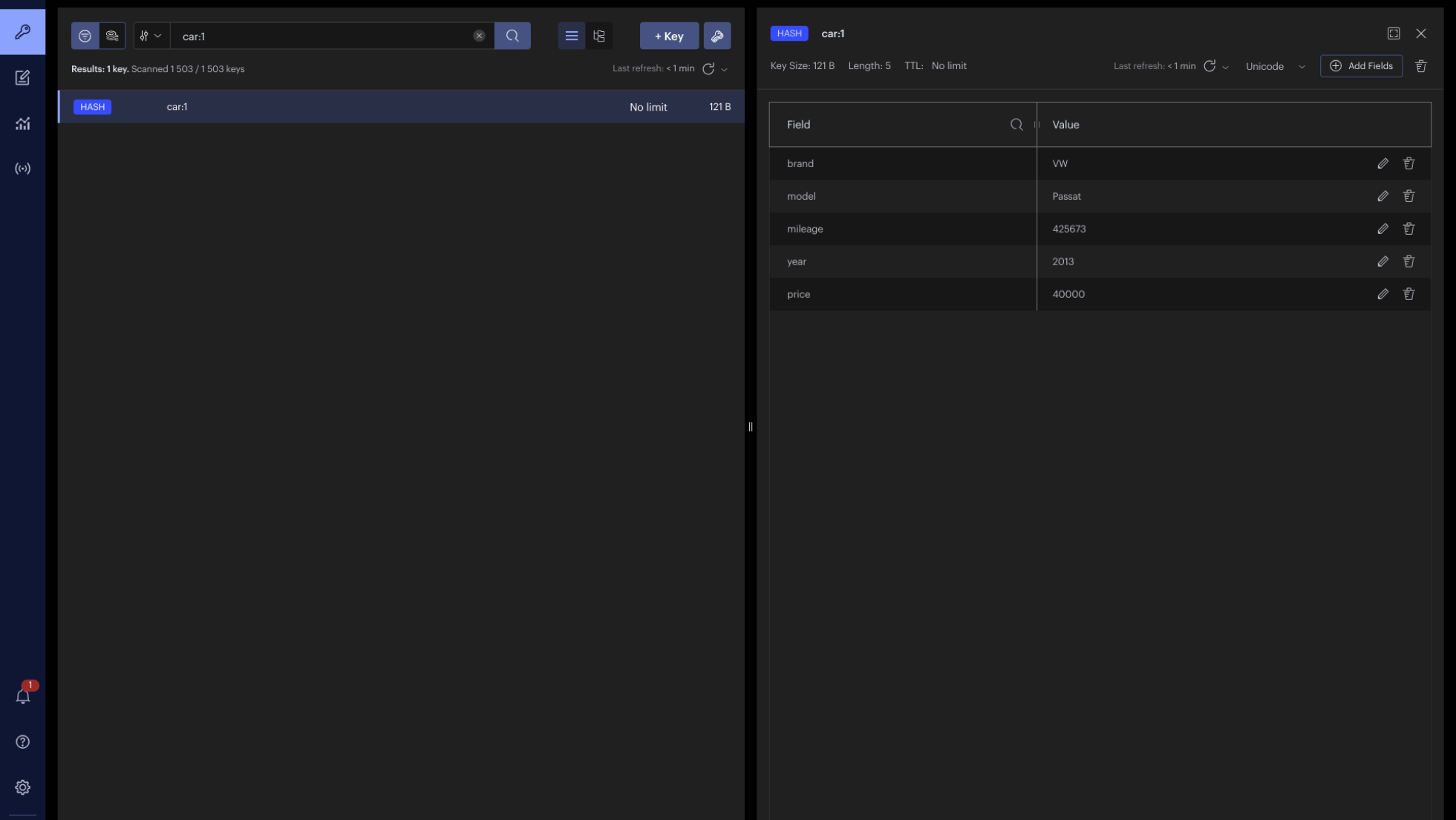
Redis OM

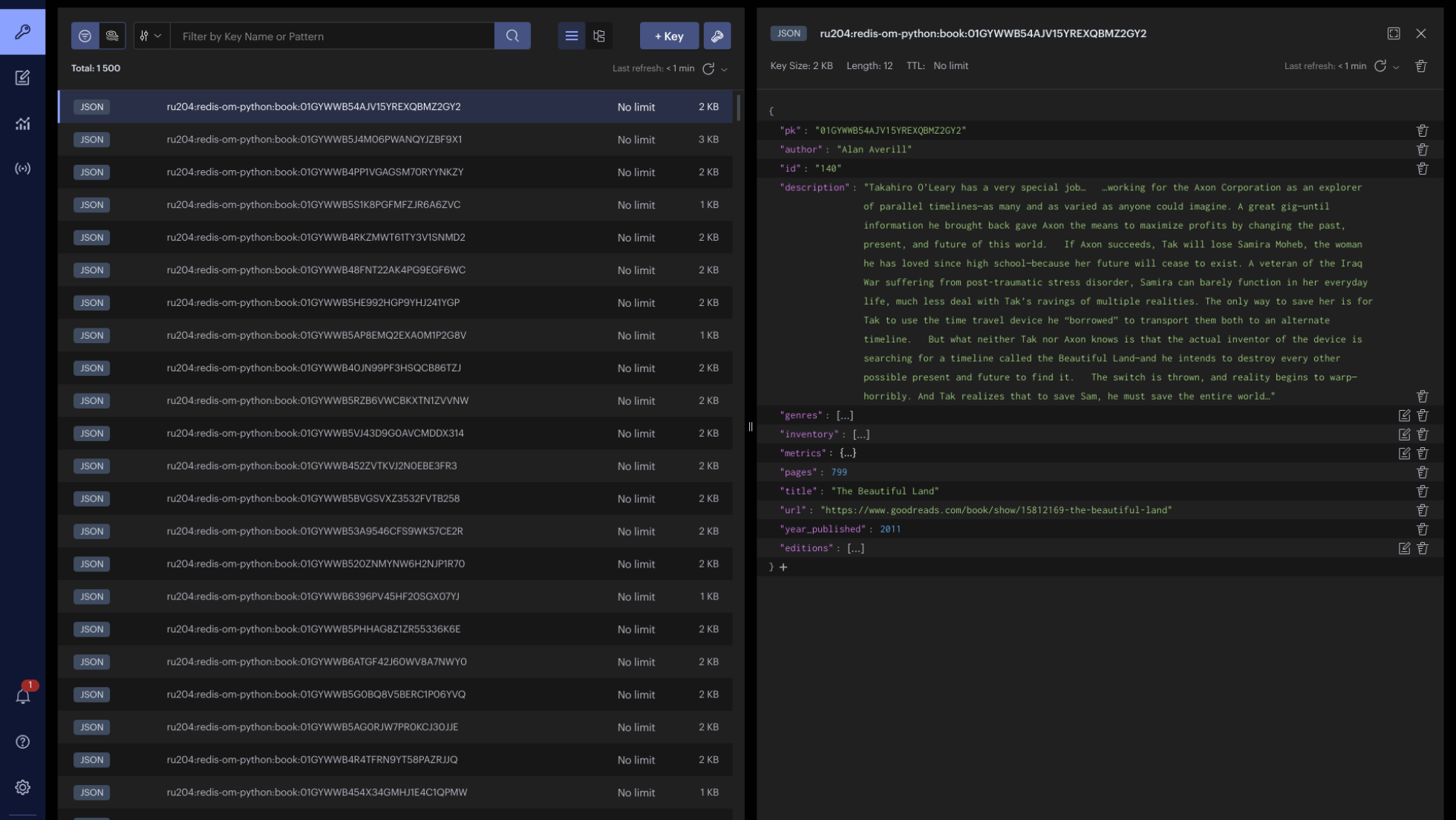
```
Event.find(Event.edition == 63).all()

Event.find(Event.prelections.author == "John").all()

Event.find(Event.description % "pystok").all()

Event.find(
  (Event.members > 80) & (Event.members < 200)
).all()
```

RedisInsights

The RedisInsight graphic user interface helps you visually browse and interact with Redis data.

- Browse, filter, and visualize Redis keys, perform CRUD operations, or delete keys in bulk.
- Display data in pretty-print JSON, hexadecimal, MessagePack, and many other formats.
- User friendly keyboard navigation.
- Use the Tree view to group data and enhance the navigation.


car:1
+ Key
Results: 1 key. Scanned 1 503 / 1 503 keys
Last refresh: < 1 min
HASH
car:1
No limit
121 B
HASH
car:1
Key Size: 121 B
Length: 5
TTL: No limit
Last refresh: < 1 min
Unicode
Add Fields
Field
Value
brand
VW
model
Passat
mileage
425673
year
2013
price
40000


Filter by Key Name or Pattern
+ Key
Total: 1 500
Last refresh: < 1 min
JSON ru204:redis-om-python:book:01GYWWB54AJV15YREXQBMZ2GY2 No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5J4MO6PWANQYJZBF9X1 No limit 3 KB
JSON ru204:redis-om-python:book:01GYWWB4PP1VGAGSM70RYYNKZY No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5S1K8PGFMFZJR6A6ZVC No limit 1 KB
JSON ru204:redis-om-python:book:01GYWWB4RKZMWT61TY3V1SNMD2 No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB48FNT22AK4PG9EGF6WC No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5HE992HGP9YHJ241YGP No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5AP8EMQ2EXA0M1P2G8V No limit 1 KB
JSON ru204:redis-om-python:book:01GYWWB40JN99PF3HSQCB86TZJ No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5RZB6VWCBKXTN1ZVVNW No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5VJ43D9G0AVCMDDX314 No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB452ZVTKVJ2N0EBE3FR3 No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5BVGSVXZ3532FVTB258 No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB53A9546CFS9WK57CE2R No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB52OZNMYNW6H2NJP1R7O No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB6396PV45HF2OSGX07YJ No limit 1 KB
JSON ru204:redis-om-python:book:01GYWWB5PHHAG8Z1ZR55336K6E No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB6ATGF42J6OWV8A7NWYO No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5G0BQ8V5BERC1PO6YVQ No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB5AGORJW7PROKCJ3OJJE No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB4R4TFRN9YT58PAZRJJQ No limit 2 KB
JSON ru204:redis-om-python:book:01GYWWB454X34GMHJ1E4C1QPMW No limit 1 KB
JSON ru204:redis-om-python:book:01GYWWB54AJV15YREXQBMZ2GY2
Key Size: 2 KB Length: 12 TTL: No limit
Last refresh: < 1 min
{
"pk" : "01GYWWB54AJV15YREXQBMZ2GY2"
"author" : "Alan Averill"
"id" : "140"
"description" : "Takahiro O'Leary has a very special job… …working for the Axon Corporation as an explorer of parallel timelines—as many and as varied as anyone could imagine. A great gig—until information he brought back gave Axon the means to maximize profits by changing the past, present, and future of this world. If Axon succeeds, Tak will lose Samira Moheb, the woman he has loved since high school—because her future will cease to exist. A veteran of the Iraq War suffering from post-traumatic stress disorder, Samira can barely function in her everyday life, much less deal with Tak's ravings of multiple realities. The only way to save her is for Tak to use the time travel device he "borrowed" to transport them both to an alternate timeline. But what neither Tak nor Axon knows is that the actual inventor of the device is searching for a timeline called the Beautiful Land—and he intends to destroy every other possible present and future to find it. The switch is thrown, and reality begins to warp—horribly. And Tak realizes that to save Sam, he must save the entire world…"
"genres" : [...]
"inventory" : [...]
"metrics" : {...}
"pages" : 799
"title" : "The Beautiful Land"
"url" : "https://www.goodreads.com/book/show/15812169-the-beautiful-land"
"year_published" : 2011
"editions" : [...]
} +

Redis Stack

WORKING WITH JSON

In this tutorial, we will go through an example of a bike shop. We will show the different capabilities of Redis Stack.

Storing and managing JSON

Let's examine the query for creating a single bike:

Create a bike

You can use JSONPath to access any part of the stored JSON document:

Get specific fields

Update specific fields

Now let's load a lot more bikes so we can play around with the queries:

Load more bikes

Indexing and querying

In addition to storing and managing JSON, Redis Stack provides the capability to index and query it. After we specify what we want to be indexed, the indexing happens automatically (and synchronously) when new data is saved.

Create an index

Now that we have our data indexed, it's really easy to run full-text searches on it and filter by number, tag and even geo position.

Query by type

Search by category and price

Even more search parameters

Aggregate search

Bike count per category

1 of 5 Next

SET key value [condition] [get] [expiration]

1 SET

-r

1 Command(s) - 1 success, 0 error(s) Dec 18, 15:37:54 1.239 ms

```
> SET users:10:email test@test.com
"OK"
```

Redis University

Redis University is place where you can learn about:

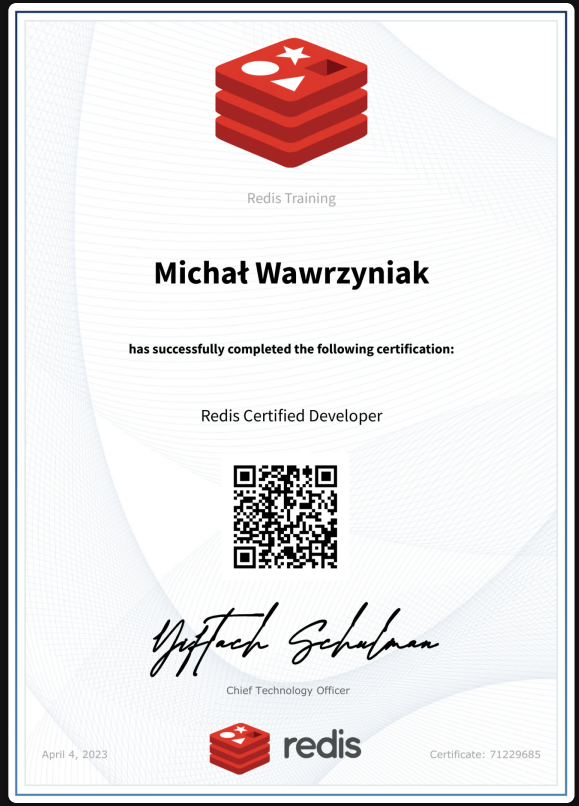
- basic Redis Data Structures
- using Redis with one of languages: Python, Java, JavaScript or .NET
- Redis Streams
- Querying, Indexing, and Full-Text Search
- Redis security
- and much more…

Each course is usually divided in 3 or 4 sections with examples, videos and quizes.

At the end of each course there is an exam. After passing exam you can receive certificate.

Redis Certified Developer

- Duration: 90 minutes
- Number of questions: 80
- Passing score: 72% (500/700)
- Languages: English
- Befeore exam you have to complete at least Redis basic + two other courses



THE END