RedisTimeSeries quick start

Prerequisites

For this quick start tutorial, you need:

- A Redis database with the RedisTimeSeries module enabled. You can use either:
  - A Redis Cloud database
  - A Redis Enterprise Software database
- redis-cli command-line tool
- redis-py client library v4.0.0 or greater

RedisTimeSeries with redis-cli

The redis-cli command-line tool is included with the Redis installation. You can use it to connect to your Redis database and test RedisTimeSeries commands.

The following examples show you how to create a time series that represents weather measurements.

Connect to a database

```bash
$ redis-cli -h <endpoint> -p <port> -a <password>
127.0.0.1:12543>
```

Create a time series

Use TS.CREATE to create a Redis key with a time series value.

Create a time series with the key name temperature:1:20 to represent the temperatures at sensor 1 and area 20.

```
127.0.0.1:12543> TS.CREATE temperature:1:20 LABELS sensor_id 1 area_id 20 data_type "temperature" "OK"
```

The LABELS parameter allows you to label time series keys with metadata that can be filtered.

Create a time series with the key name humidity:1:20 to represent the humidity at sensor 1 and area 20.

```
127.0.0.1:12543> TS.CREATE humidity:1:20 LABELS sensor_id 1 area_id 20 data_type "humidity" "OK"
```

Create temperature and humidity time series keys for other temperature and humidity sensors in a different area.
Add data to a time series

Use `TS.ADD` to add new data to a time series.

Add a few temperature measurements from area 20 to `temperature:1:20`.

```
127.0.0.1:12543> TS.ADD temperature:1:20 1652824475581 23.9
127.0.0.1:12543> TS.ADD temperature:1:20 1652828015127 22.8
127.0.0.1:12543> TS.ADD temperature:1:20 1652831657886 21.7
```

The second argument in `TS.ADD` is the UNIX timestamp for the sample in milliseconds. You can also specify `*` for the current time on the Redis server.

Add a few humidity measurements from area 20 to `humidity:1:20`.

```
127.0.0.1:12543> TS.ADD humidity:1:20 1652824475581 29
127.0.0.1:12543> TS.ADD humidity:1:20 1652828015127 27
127.0.0.1:12543> TS.ADD humidity:1:20 1652831657886 28
```

You can also update multiple time series at once with `TS.MADD`.

Use `TS.MADD` to add multiple measurements from area 24 to `temperature:2:24` and `humidity:2:24`.

```
127.0.0.1:12543> TS.MADD temperature:2:24 1652824475581 18.3 humidity:2:24 1652824475581 51
127.0.0.1:12543> TS.MADD temperature:2:24 1652828015127 17.2 humidity:2:24 1652828015127 54
```

Get data from a time series

Use `TS.GET` to get the latest entry in the time series.
Use `TS.GET` to get a specific entry in the time series:

```
127.0.0.1:12543> TS.GET temperature:1:20
1) "1652831657886"
2) "21.7"
```

Use `TS.RANGE` to get a range of timestamps and values in the time series, ordered from earliest to latest:

```
127.0.0.1:12543> TS.RANGE temperature:2:24
1) 1) "1652824475581"
    2) "18.3"
2) 1) "1652828015127"
    2) "17.2"
```

`TS.REVRANGE` orders the values from latest to earliest.

You can specify `0` and `+` for the starting and ending timestamps to get all of the entries in the time series:

```
127.0.0.1:12543> TS.REVRANGE temperature:2:24 0 +
1) 1) "1652831657886"
    2) "16.7"
2) 1) "1652828015127"
    2) "17.2"
3) 1) "1652824475581"
    2) "18.3"
```

Use `TS.MGET` to get the last entry in all time series keys with an `area_id` of 20:

```
127.0.0.1:12543> TS.MGET WITHLABELS FILTER area_id=20
1) 1) "humidity:1:20"
    2) 1) "sensor_id"
        2) "1"
    2) "area_id"
        2) "20"
    3) "data_type"
        2) "humidity"
    3) 1) "1652831657886"
        2) "28"
2) 1) "temperature:1:20"
    2) 1) "sensor_id"
        2) "1"
    2) "area_id"
        2) "20"
    3) "data_type"
        2) "temperature"
    3) 1) "1652831657886"
        2) "21.7"
```

Use `TS.MRANGE` to get a range of entries in all time series keys with a `sensor_id` of 2 from earliest to latest.
Then, use `TS.MREVRANGE` to get a range of entries in all time series keys that are humidity keys from latest to earliest.

Delete data from a time series

Use `TS.DEL` to delete all timestamps in a specified range from a time series.

RedisTimeSeries with Python

If you want to use RedisTimeSeries within an application, you can use one of the client libraries.

The following example uses the Redis Python client library `redis-py`, which supports RedisTimeSeries commands as of v4.0.0.
This Python code adds multiple time series keys to Redis, adds data to the time series, retrieves data from the time series, and then deletes the time series keys.

```python
import redis
import json

# To make the responses easier to read

# Data from two sensors
sensor1 = {
    "sensorId": 1,
    "areaId": 20,
    "data": [
        { "timestamp": 1652824475581, "temperature": 23.9, "humidity": 29 },
        { "timestamp": 1652828015127, "temperature": 22.8, "humidity": 27 },
        { "timestamp": 1652831657886, "temperature": 21.7, "humidity": 28 },
        { "timestamp": 1652835287321, "temperature": 19.4, "humidity": 35 },
        { "timestamp": 1652838756149, "temperature": 17.8, "humidity": 40 }
    ]
}
sensor2 = {
    "sensorId": 2,
    "areaId": 24,
    "data": [
        { "timestamp": 1652824475581, "temperature": 18.3, "humidity": 51 },
        { "timestamp": 1652828015127, "temperature": 17.2, "humidity": 54 },
        { "timestamp": 1652831657886, "temperature": 16.7, "humidity": 56 },
        { "timestamp": 1652835287321, "temperature": 15.6, "humidity": 60 },
        { "timestamp": 1652838756149, "temperature": 14.4, "humidity": 65 }
    ]
}
sensors = [ sensor1, sensor2 ]

# Connect to a Redis database
r = redis.Redis(host="<endpoint>", port="<port>", password="<password>"

# Create a list to store keys
key_list = []
# Create a list of keys and values for ts.madd
humidity_madd_list = []
# For each sensor:
for sensor in sensors:
    # Create time series keys
    key_end = "{}:{}".format(sensor["sensorId"], sensor["areaId"])
    temperature_key = "temperature-py:" + key_end
    humidity_key = "humidity-py:" + key_end
    key_list.extend((temperature_key, humidity_key))

    print("Create {} and {}".format(temperature_key, humidity_key))

    labels = { "sensorId" : sensor["sensorId"], "areaId" : sensor["areaId"] }
    labels["dataType"] = "temperature"
    r.ts().create(temperature_key, labels=labels)

    labels["dataType"] = "humidity"
    r.ts().create(humidity_key, labels=labels)
```
# Add temperature data to time series keys with ts.add
print("Add data to {}".format(temperature_key))
# For each data entry:
for entry in sensor["data"]:
    r.ts().add(temperature_key, entry["timestamp"], entry["temperature"])
# Create a list of humidity data for ts.madd
humidity_madd_list.append( (humidity_key, entry["timestamp"], entry["humidity"]))

# Add humidity data to time series keys with ts.madd
print("Add data with ts.madd:
{}").format(humidity_madd_list))

r.ts().madd(humidity_madd_list)
input("\nAdded all data. Press enter to continue..."

print("Get the last value from temperature-py:1:20...")
get_reply = r.ts().get("temperature-py:1:20")
print(json.dumps(get_reply, indent=4) + "\n")

print("Get all of the values from humidity-py:2:24...")
range_reply = r.ts().range("humidity-py:2:24", 0, "+")
print(json.dumps(range_reply, indent=4) + "\n")

print("Get last values from area 20...")
mget_filters = ["areaId=20"]
mget_reply = r.ts().mget(mget_filters, with_labels=True)
print(json.dumps(mget_reply, indent=4) + "\n")

print("Get all of the values from sensor 2 from earliest to latest...")
mrange_filters = ["sensorId=2"]
mrange_reply = r.ts().mrange(0, "+", mrange_filters, with_labels=True)
print(json.dumps(mrange_reply, indent=4) + "\n")

print("Get all humidity values from latest to earliest...")
mrevrange_filters = ["dataType=humidity"]
mrevrange_reply = r.ts().mrevrange(0, "+", mrevrange_filters, with_labels=True)
print(json.dumps(mrevrange_reply, indent=4) + "\n")

print("Delete range of values from temperature-py:1:20...")
r.ts().delete("temperature-py:1:20", 0, 1652831657886)
range_reply = r.ts().range("temperature-py:1:20", 0, "+")
print(json.dumps(range_reply, indent=4) + "\n")
input("Press enter to finish...")
print("Deleting keys: {}".format(key_list))
for key in key_list:
    r.delete(key)
print("Done!")

Example output

$ python3 quick_start.py
Create temperature-py:1:20 and humidity-py:1:20
Add data to temperature-py:1:20
Create temperature-py:2:24 and humidity-py:2:24
Add data to temperature-py:2:24
Add data to temperature.py:2:24

Add data with ts.madd:

```python
[(
    'humidity-py:1:20', 1652824475581, 29,
    'humidity-py:1:20', 1652831657886, 28,
    'humidity-py:1:20', 1652835287321, 35,
    'humidity-py:1:20', 165284475581, 51,
    'humidity-py:2:24', 1652828015127, 54,
    'humidity-py:2:24', 1652831657886, 56,
    'humidity-py:2:24', 1652835287321, 60,
    'humidity-py:2:24', 1652838756149, 65)
]
```

Added all data. Press enter to continue...

Get the last value from temperature.py:1:20...

```python
[1652838756149, 17.8]
```

Get all of the values from humidity.py:2:24...

```python
[
    [1652824475581, 51.0],
    [1652828015127, 54.0],
    [1652831657886, 56.0],
    [1652835287321, 60.0],
    [1652838756149, 65.0]
]
```

Get last values from area 20...

```json
[
    {
        "humidity-py:1:20": [
            {
                "sensorId": "1",
                "areaId": "20",
                "dataType": "humidity"
            },
            1652838756149,
            40.0
        ]
    },
    {
        "temperature-py:1:20": [
            {
                "sensorId": "1",
                "areaId": "20",
                "dataType": "temperature"
            }
        ]
    }
]```
Get all of the values from sensor 2 from earliest to latest...

```
{
    "humidity-py:2:24": [
        {
            "sensorId": "2",
            "areaId": "24",
            "dataType": "humidity"
        },
        [1652824475581, 51.0],
        [1652828015127, 54.0],
        [1652831657886, 56.0],
        [1652835287321, 60.0],
        [1652838756149, 65.0]
    ],
    "temperature-py:2:24": [
        {
            "sensorId": "2",
            "areaId": "24",
            "dataType": "temperature"
        },
        [1652824475581, 18.3],
        [1652828015127, 17.2],
        [1652831657886, 16.1],
        [1652835287321, 15.0],
        [1652838756149, 14.0]
    ]
}
```
Get all humidity values from latest to earliest...

```json
{
  "humidity-py:1:20": [
    {
      "sensorId": "1",
      "areaId": "20",
      "dataType": "humidity"
    },
    [
      1652838756149,
      40.0
    ],
    [
      1652835287321,
      35.0
    ],
    [
      1652831657886,
      28.0
    ],
    [
      1652828015127,
      27.0
    ],
    [
      1652824475581,
      29.0
    ]
  },
  "humidity-py:2:24": [
    {
      "sensorId": "2",
      "areaId": "24",
      "dataType": "humidity"
    },
    [
      1652838756149,
      15.6
    ],
    [
      1652835287321,
      16.7
    ],
    [
      1652831657886,
      14.4
    ],
    [
      1652828015127,
      27.0
    ],
    [
      1652824475581,
      29.0
    ]
  }
}
```
Delete range of values from temperature-py:1:20...

[ [ 1652835287321, 19.4 ], [ 1652835287321, 19.4 ] ]

Press enter to finish...
Done!

More info

- RedisTimeSeries commands
- RedisTimeSeries client libraries

Updated: May 20, 2022