

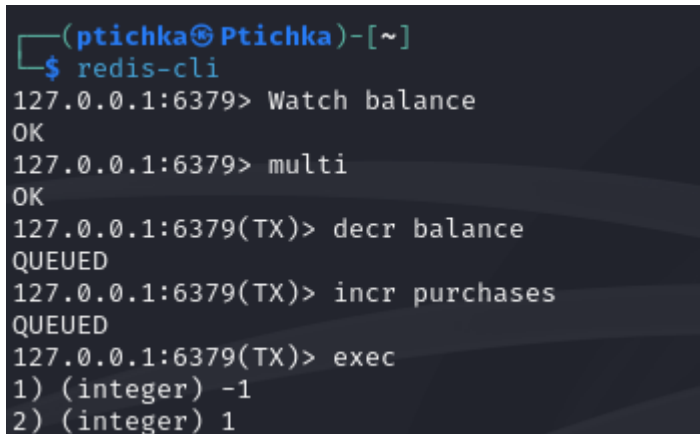
Звіт по роботі з розширеними можливостями Redis

Вступ

Метою роботи було ознайомлення з механізмами Redis для роботи з транзакціями, Lua-скриптами, підпискою та публікацією повідомлень (Pub/Sub), а також потоками Redis Streams. Практична частина передбачала виконання команд у redis-cli та написання додаткового скрипта на Python.

Практична частина

1. Транзакції з MULTI, EXEC, WATCH



```
(ptichka@Ptichka)-[~]  
$ redis-cli  
127.0.0.1:6379> Watch balance  
OK  
127.0.0.1:6379> multi  
OK  
127.0.0.1:6379(TX)> decr balance  
QUEUED  
127.0.0.1:6379(TX)> incr purchases  
QUEUED  
127.0.0.1:6379(TX)> exec  
1) (integer) -1  
2) (integer) 1
```

```

127.0.0.1:6379> set balance 100
OK
127.0.0.1:6379> set purchases 5
OK
127.0.0.1:6379> WATCH balance
OK
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379(TX)> DECR balance
QUEUED
127.0.0.1:6379(TX)> INCR purchases
QUEUED
127.0.0.1:6379(TX)> EXEC
1) (integer) 99
2) (integer) 6
127.0.0.1:6379> SET balance 90
OK

```

Створення кількох ключів у транзакції:

```

127.0.0.1:6379> WATCH key1 key2
OK
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379(TX)> SET key1 "value1"
QUEUED
127.0.0.1:6379(TX)> SET key2 "value2"
QUEUED
127.0.0.1:6379(TX)> EXEC
1) OK
2) OK
127.0.0.1:6379> SET key1 "new value"

```

Lua-скрипт із перевіркою наявності ключа:

```

127.0.0.1:6379> EVAL "if redis.call('exists', KEYS[1]) = 0 then return redis.call('set', KEYS[1], ARGV[1]) else re
turn 'exists' end" 1 testkey "value"
OK
127.0.0.1:6379> GET testkey
"value"
127.0.0.1:6379> █

```

Pub/Sub

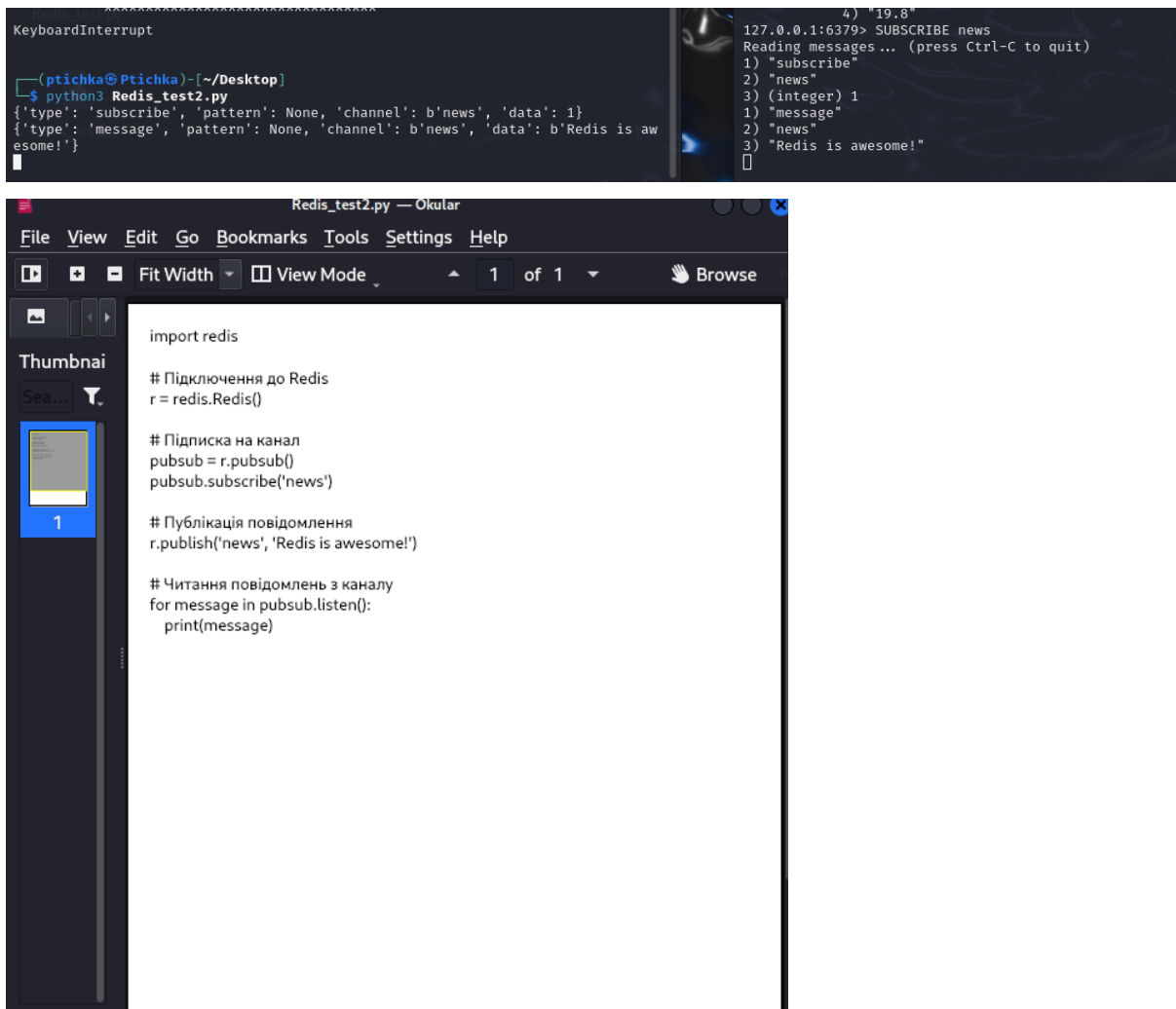
```
ptichka@Ptichka:~$ redis-cli
127.0.0.1:6379> SUBSCRIBE news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "Redis is awesome!"

ptichka@Ptichka:~$ redis-cli
127.0.0.1:6379(TX)> EXEC
1) OK
2) OK
127.0.0.1:6379> SET key1 "new_value"
OK
127.0.0.1:6379>
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379(TX)> SET key1 "new_value"
QUEUED
127.0.0.1:6379(TX)> EXEC
1) OK
127.0.0.1:6379> set balance 100
OK
127.0.0.1:6379> set purchases 5
OK
127.0.0.1:6379> WATCH balance
OK
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379(TX)> DECR balance
QUEUED
127.0.0.1:6379(TX)> INCR purchases
QUEUED
127.0.0.1:6379(TX)> EXEC
1) (integer) 99
2) (integer) 6
127.0.0.1:6379> SET balance 90
OK
127.0.0.1:6379> EVAL "if redis.call('exists', KEYS[1]) = 0 then
Invalid argument(s)
127.0.0.1:6379> EVAL "if redis.call('exists', KEYS[1]) = 0 then return redis.call('set', KEYS[1], ARGV[1]) else re
turn 'exists' end" 1 testkey "value"
OK
127.0.0.1:6379> GET testkey
"value"
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379>
127.0.0.1:6379> SUBSCRIBE news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
^C(15.21s)
127.0.0.1:6379> PUBLISH news "Redis is awesome!"
(integer) 0
127.0.0.1:6379> PUBLISH news "Redis is awesome!"
(integer) 1
127.0.0.1:6379>
```

Redis Streams:

```
127.0.0.1:6379> XADD mystream * sensor-id 1234 temperature 19.8
"1746557491059-0"
127.0.0.1:6379> XRANGE mystream - +
1) 1) "1746557491059-0"
   2) 1) "sensor-id"
      2) "1234"
      3) "temperature"
      4) "19.8"
127.0.0.1:6379> XREAD COUNT 2 STREAMS mystream 0
1) 1) "mystream"
   2) 1) 1) "1746557491059-0"
      2) 1) "sensor-id"
         2) "1234"
         3) "temperature"
         4) "19.8"
127.0.0.1:6379>
```

Додаткове завдання:



The image shows two windows. The top window is a terminal with a dark background. It shows a 'KeyboardInterrupt' message, followed by a prompt to run 'python3 Redis_test2.py'. The output shows a successful subscription to the 'news' channel and the receipt of a message: 'Redis is awesome!'. The bottom window is a code editor titled 'Redis_test2.py — Okular'. It displays the Python code for the test script, which imports Redis, connects to a local instance, subscribes to the 'news' channel, publishes a message, and listens for it.

```
KeyboardInterrupt

(ptichka@Ptichka)[:-/~/Desktop]
$ python3 Redis_test2.py
{'type': 'subscribe', 'pattern': None, 'channel': b'news', 'data': 1}
{'type': 'message', 'pattern': None, 'channel': b'news', 'data': b'Redis is awesome!'}

127.0.0.1:6379> SUBSCRIBE news
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "Redis is awesome!"
[]
```

```
import redis

# Підключення до Redis
r = redis.Redis()

# Підписка на канал
pubsub = r.pubsub()
pubsub.subscribe('news')

# Публікація повідомлення
r.publish('news', 'Redis is awesome!')

# Читання повідомлень з каналу
for message in pubsub.listen():
    print(message)
```

Теоретичні висновки

Механізми Redis, як-от транзакції, скрипти, потоки й Pub/Sub, дозволяють будувати системи, що потребують високої швидкодії, реактивності та простої синхронізації. Особливо це корисно у реальному часі — сповіщення, сенсорні дані, черги подій.

Відповіді на запитання для самоперевірки

1. Як працюють транзакції в Redis і для чого потрібна команда WATCH?

Redis транзакції групують команди, які виконуються послідовно після

EXEC. Команда WATCH "спостерігає" за ключами — якщо вони зміняться до EXEC, транзакція не виконується (імітуючи механізм конкурентності).

2. Для чого використовують Lua-скрипти в Redis?

Lua-скрипти забезпечують атомарність і дають змогу виконувати складну логіку на стороні Redis без затримки клієнт-сервер. Наприклад: умовні перевірки, цикли, комбінації команд.

3. У чому суть моделі Pub/Sub і які її обмеження?

Модель Pub/Sub дозволяє одному клієнту публікувати повідомлення, а іншим — підписуватись на канали. Обмеження: немає збереження повідомлень, якщо підписника немає онлайн, то повідомлення втрачається.

4. Що таке Redis Streams? Як зчитуються події зі стріму?

Redis Streams — це лог структурованих подій із ключами-значеннями. Події додаються через XADD, а зчитуються через XRANGE (історія) або XREAD (нові події).

5. У яких випадках Redis може бути альтернативою черзі повідомлень (наприклад, RabbitMQ)?

Redis підходить для простих або надшвидких черг — наприклад, для логів, лічильників, повідомлень у реальному часі. Але не має складної маршрутизації, гарантій доставки, черг із підтвердженням (як у RabbitMQ).

Загальний висновок

Під час роботи було вивчено ключові можливості Redis: транзакції, Lua-скрипти, Pub/Sub та Streams. Ці інструменти роблять Redis не лише сховищем даних, а й потужним засобом для обробки подій, повідомлень і атомарних операцій у реальному часі. Redis може бути легкою та швидкою альтернативою брокерам повідомлень.