Лабораторна робота № 8 Бази даних та інформаційні системи

Тема: Розширені можливості Redis

Мета: Закріпити знання про роботу з Redis та ознайомитися з розширеним функціоналом — транзакціями, скриптами на Lua, публікацією/підпискою (Pub/Sub) та потоками Redis Streams.

Виконала: студентка групи МІТ-31, Панченко Владислава

Транзакції у Redis. Ознайомлення з командами MULTI, EXEC, DISCARD, WATCH. Використання WATCH для імітації конкурентного доступу.

```
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
```

Створення простої транзакції, що додає кілька ключів.

```
127.0.0.1:6379> MULTI
OK
127.0.0.1:6379(TX)> SET course "Data bases & Info systems"
QUEUED
127.0.0.1:6379(TX)> SET labname "Redis"
QUEUED
127.0.0.1:6379(TX)> SET student "Vladyslava Panchenko"
QUEUED
127.0.0.1:6379(TX)> EXEC
1) OK
2) OK
3) OK
```

Lua-скрипти в Redis. Напишіть простий Lua-скрипт, який перевіряє наявність ключа і створює його, якщо не існує.

```
127.0.0.1:6379> EVAL "if redis.call('exists', KEYS[1]) == 0 then retur
n redis.call('set', KEYS[1], ARGV[1]) else return 'exists' end" 1 test
key "value"
"exists"
```

Механізм Pub/Sub. В одному терміналі запустіть підписку на канал news, в іншому терміналі виконайте публікацію повідомлення.

```
127.0.0.1:6379> PUBLISH news "Redis is awesome!"
(integer) 1
```

```
127.0.0.1:6379> SUBSCRIBE news
1) "subscribe"
2) "news"
3) (integer) 1
1) "message"
2) "news"
3) "Redis is awesome!"
```

Redis Streams (потоки даних). Додайте події до потоку. Прочитайте останні події. Створіть споживача потоку та зчитайте нові повідомлення.

```
127.0.0.1:6379> XADD mystream * sensor-id 1234 temperature 19.8
"1746017911384-0"
127.0.0.1:6379> XRANGE mystream - +
1) 1) "1746017911384-0"
   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) "1746017911384-0"
         2) 1) "sensor-id"
            2) "1234"
            3) "temperature"
            4) "19.8"
```

Напишіть скрипт Python, який читає дані зі Stream.

```python
import redis

r = redis.Redis()

#Додавання події до потоку
r.xadd('mystream', {'sensor-id': '1234', 'temperature': '19.8'})

#Читання подій
messages = r.xrange('mystream')
print(messages)
```

Результати виконань програми:

```
[(b'1746017911384-0', {b'sensor-id': b'1234', b'temperature': b'19.8'}), (b'1746018262885-0', {b'sensor-id'
: b'1234', b'temperature': b'19.8'}), (b'1746018267567-0', {b'sensor-id': b'1234', b'temperature': b'19.8'}
), (b'1746018339328-0', {b'sensor-id': b'1234', b'temperature': b'19.8'})]
```

Визначте, які з нових механізмів Redis можуть бути корисними у великих розподілених системах.

Redis Streams — дозволяє організувати обробку потоків даних у реальному часі.

Pub/Sub — ідеально підходить для систем реального часу, таких як чати або сповіщення.

Транзакції — дозволяють атомарно виконувати кілька операцій.

Lua-скрипти — допомагають виконувати складні логічні операції на сервері.

Які плюси і мінуси Redis як брокера повідомлень?

Плюси:

- Висока швидкість.
- Простота реалізації.
- Підтримка механізму Pub/Sub для реального часу.

Мінуси:

- Відсутність гарантій доставки повідомлень (наприклад, може бути втрачено повідомлення, якщо підписник не активно слухає канал).
- Може бути складно масштабувати при великій кількості підписників або повідомлень.

Запитання для самоперевірки

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

Транзакції в Redis — це послідовність команд, які виконуються атомарно після MULTI → EXEC. WATCH використовується для "спостереження" за ключами перед початком транзакції. Якщо хоча б один ключ зміниться іншим клієнтом до EXEC, транзакція скасовується — це запобігає стану "гонки" (race condition).

2. Для чого використовують Lua-скрипти в Redis?

Lua-скрипти дозволяють виконувати складну логіку на сервері Redis в одному атомарному блоці.

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

Pub/Sub (Publish/Subscribe) — це модель, де відправник (publisher) публікує повідомлення в канал, а підписники (subscribers) отримують їх у реальному часі.

Обмеження:

- Немає гарантії доставки (якщо підписник офлайн — він пропустить повідомлення).
- Повідомлення не зберігаються.
- Не масштабуються автоматично — обмежена підтримка для високого навантаження.

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

Redis Streams — це структура даних для черги повідомлень із збереженням історії та підтримкою споживачів (consumer groups).

Зчитування:

XRANGE mystream - + — прочитати всі події

XREAD STREAMS mystream 0 — зчитати нові події

XREAD STREAMS mystream $ — тільки події після поточних (реальне очікування нових)

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

Коли потрібна висока швидкість і простота, повідомлення обробляються одразу або дуже швидко, дані не обов’язково мають зберігатись після перезавантаження.

Додаток: файл lab9.py до завдання з програмою в Python