

Redis

План семинара

Блок 1: Установка

- Установка Redis
- Подключение
- Краткий экскурс
- Форматы ключей

Блок 2: CLI CRUD команды

- SET - вставка ключей
- GET - получение
- RENAME - переименование
- EXISTS KEYS SCAN
- TTL

Блок 3: Атомарные операции

- MULTI
- EXEC
- DISCARD
- WATCH

Блок 4: Очереди

- Pub
- Sub
- Очередь на практике

Блок 1: Установка и подключение

Шаг 1

Поднимаем Redis в Docker контейнере и подключаемся

```
# Поднимаем Redis
docker run --name redis -p 6379:6379 -d redis

# Проверяем, что Redis поднят
docker ps
```

```
# Подключаемся
docker exec -it redis bash

# Подключаемся к CLI
redis-cli
```

Шаг 2 Форматы ключей

Есть общепринятый формат хранения ключей в Redis в человекочитаемом виде

`<namespace>:<object_id>:<field_name>`

Т.е. если необходимо хранить пользователей, то ключи будут следующего формата

`users:123:name`

Получается ключ, под которым хранится имя пользователя с ID 123

С форматом ключей необходимо определиться заранее и не делать их слишком длинными, так как это может замедлить поиск

Блок 2: CLI CRUD команды

Добавление/получение пользователя

```
SET user:1:name "Bob"
GET user:1:name
> "Bob"
```

В качестве хранения Redis использует binary-safe-string, это значит, что даже при установке значения бинарными байтами, оно останется как есть

```
SET user:1:image "\x00\x01ABC"
GET user:1:image
> "\x00\x01ABC"
```

Читаем старое значение и устанавливаем новое

```
GETSET user:1:name "Anton"
> "Bob"
```

Меняем название ключа

```
RENAME user:1:image user:1:icon
```

Проверка на наличие ключа

```
EXISTS user:1:image
> 0
EXISTS user:1:icon
```

```
> 1
```

Работаем с *

```
KEYS user:1:*  
> user:1:icon  
> user:1:name
```

Удаление ключа

```
DEL user:1:icon  
KEYS user:1:*  
> user:1:name
```

Сохранение множества при помощи хэшей

```
HSET user:1 name "Bob" icon "\x00\x01ABC"  
HGET user:1  
> name, Bob, icon, \x00\x01ABC  
HGET user:1 name  
> Bob
```

Блок 3: Атомарные операции

План:

- Начинаем транзакцию
- Добавляем ключ age
- Увеличиваем его на один
- Заканчиваем транзакцию

```
MULTI  
> OK  
SET users:1:age "22"  
> QUEUED  
INCR users:1:age  
> QUEUED  
EXEC  
> OK  
> (integer) 23
```

План:

- Начинаем транзакцию
- Добавляем ключ age
- Увеличиваем его на один
- Отменяем транзакцию
- Проверим, есть ли значение по ключу

```
MULTI  
> OK
```

```
SET users:2:age "22"
> QUEUED
INCR users:2:age
> QUEUED
DISCARD
> OK
GET users:2:age
> (nil) ← Такого значения нет, так как мы отменили транзакцию
```

Ещё немного про атомарные операции

Открываем ещё один терминал и повторяем команды

```
# Подключаемся
docker exec -it redis bash

# Подключаемся к CLI
redis-cli
```

Немного вернёмся к атомарным операциям

Создаём ещё один ключ

```
SET users:2:age "22"
```

Терминал 1:

Начинаем “следить” за ключом и стартуем транзакцию

```
WATCH users:2:age
> OK
MULTI
> OK
```

Терминал 2:

Изменяем значение созданного ключа

```
INCR users:2:age
> (integer) 1
```

Терминал 1:

Тоже изменяем значение и завершаем транзакцию

```
INCR users:2:age
> QUEUED
EXEC
> (nil) ← Из-за того, что в процессе выполнения транзакции значение было
изменено другим клиентом, Redis отменил транзакцию и вернул nil
```

Блок 4: Pub Sub

PUBLISH и SUBSCRIBE

Откроем ещё один терминал (повторить действия из предыдущего этапа)

План:

- 1 терминал подписывается на **tasks**
- 2 терминал подписывается на **offers**
- 3 терминал отправляет сообщения в эти топики

Терминал 1

Подписываемся на **tasks** командой

```
SUBSCRIBE tasks
```

Терминал 2

Подписываемся на **offers** командой

```
SUBSCRIBE offers
```

Терминал 3

Отправляем сообщения в топики

```
PUBLISH offers "Offer5219"  
PUBLISH offers "Offer2349"  
PUBLISH offers "Offer9331"  
PUBLISH offers "Offer0001"  
PUBLISH tasks "Task5219"  
PUBLISH tasks "Task2349"  
PUBLISH tasks "Task9331"  
PUBLISH tasks "Task0001"
```

Проверяем результаты

- В терминале 1 должны были появиться только сообщения с содержанием ("Task5219" "Task2349" "Task9331" "Task0001")

- В терминале 2 должны были появиться только сообщения с содержанием ("Offer5219" "Offer2349" "Offer9331" "Offer0001")

ДЗ 8

Необходимо интегрировать Redis в приложение, реализованное в рамках домашнего задания номер 6. У вас должен быть следующий REST метод

- Получение всех товаров вместе с названиями их категорий (намек на join-запрос)

Необходимо закешировать ответ от этого метода, используя Redis, а также продумать допустимый TTL. Пример с интеграцией Redis находится в [репозитории курса](#)