



Онлайн-образование

Проверить, идет ли запись!



Меня хорошо видно && слышно?

Ставьте +, если все хорошо

Напишите в чат, если есть проблемы

Redis, часть 1



Дмитрий Кириллов

Технический директор

1С-Старт

@esteps_kirillov

Правила вебинара



Активно участвуем



Задаем вопрос в чат или голосом



Off-topic обсуждаем в Slack #канал-группы или #general



Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара

01. Особенности Redis



02. Работа с данными



03. Транзакции



Цели вебинара | После занятия вы сможете

1

Определить, подходит ли Redis
для решения конкретной задачи

2

Выбрать оптимальный тип данных

3

Использовать Redis с учётом
специфических особенностей

Установка и инструменты

Docker

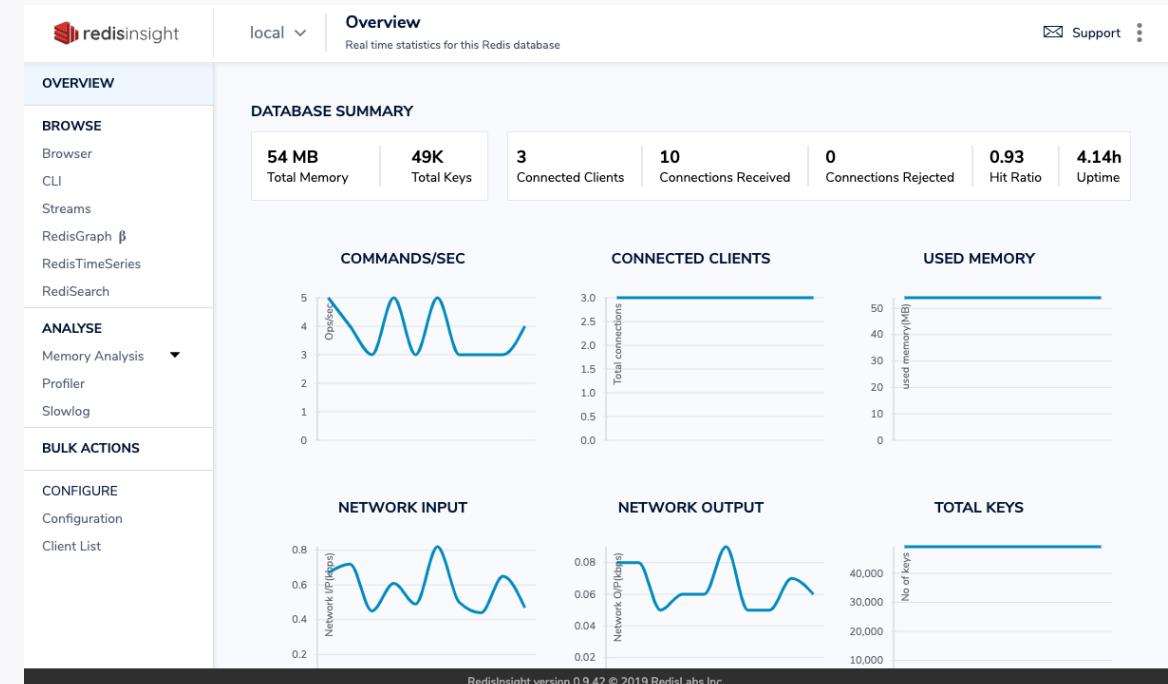
```
docker pull redis
docker run --name otus-redis -d redis
docker exec -it otus-redis redis-cli
```

Установка и инструменты

RedisInsight

<https://redis.com/redis-enterprise/redis-insight/>

- официальный бесплатный GUI для работы с Redis
- отлично подходит для образовательных целей
- работает в браузере



01. Особенности Redis

Что такое Redis

- Redis – это не только база данных
- Redis – это **сервер структур данных**,
т.е. что-то вроде "внешнего ОЗУ" для приложения
- его основная "фишка" – это возможность
выполнять типовые операции, которые обычно
реализуют на клиенте

Особенности Redis

1. Все данные хранятся в оперативной памяти

Плюсы:

- максимальная производительность

Минусы:

- при отказе сервера теряются все данные*
- объём ОЗУ на сервере намного меньше, чем объём дисков

* про персистентность мы поговорим во второй части

2. Однопоточность

Плюсы:

- отсутствие взаимных блокировок данных
- атомарность транзакций

Минусы:

- не используются преимущества многоядерных систем

3. Хранилище «ключ-значение»

Плюсы:

- практически все операции выполняются за $O(1)$

Минусы:

- в версии «из коробки» нет поиска, индексов, внешних ключей...
- нет возможности строить сложные запросы

4. Разнообразные структуры данных

- строки
- хэш-таблицы
- списки
- множества
- упорядоченные множества
- и т.д.

02. Работа с данными

Особенности хранения данных

1. Архитектура Redis

Особенности:

- вся база Redis — это один большой словарь "ключ-значение"*
- команды для работы с данными очень простые, в основном — установка / получение значения

Ключ	Значение						
site:name	Магазин игрушек						
user:1	<table><tr><td>uid</td><td>1</td></tr><tr><td>name</td><td>Елена</td></tr><tr><td>city</td><td>Москва</td></tr></table>	uid	1	name	Елена	city	Москва
uid	1						
name	Елена						
city	Москва						
user:2	<table><tr><td>uid</td><td>2</td></tr><tr><td>name</td><td>Иван</td></tr><tr><td>city</td><td>Воронеж</td></tr></table>	uid	2	name	Иван	city	Воронеж
uid	2						
name	Иван						
city	Воронеж						
orders:total	<table><tr><td>14800</td><td>user:2</td></tr><tr><td>12000</td><td>user:1</td></tr></table>	14800	user:2	12000	user:1		
14800	user:2						
12000	user:1						

* аналог — ассоциативный массив / словарь / Мар...

Особенности хранения данных

1. Архитектура Redis

Как следствие:

- в Redis **нет** внешних ключей, JOIN, подзапросов...
- обычный сценарий — сделать несколько запросов вместо одного

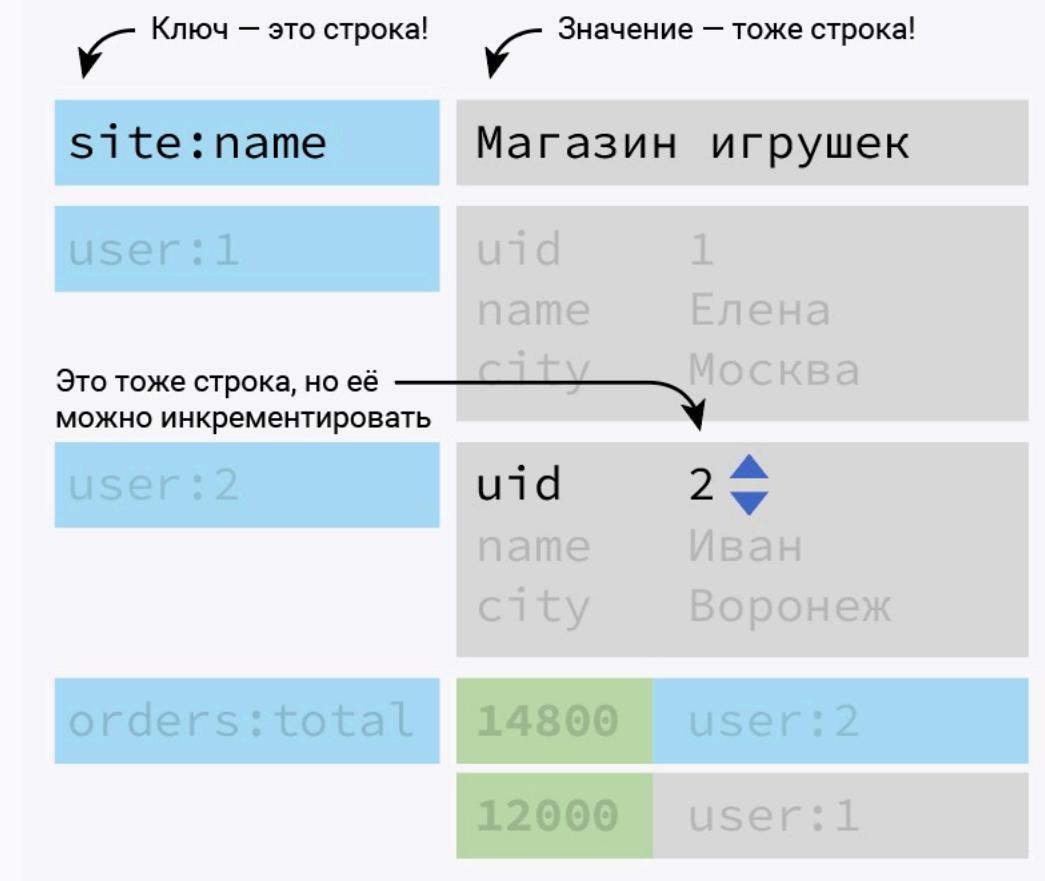


Особенности хранения данных

2. Типы данных в Redis

Особенности:

- все значения хранятся как строки*
- если строка содержит число, становятся доступны дополнительные команды (инкрементация и т.д.)



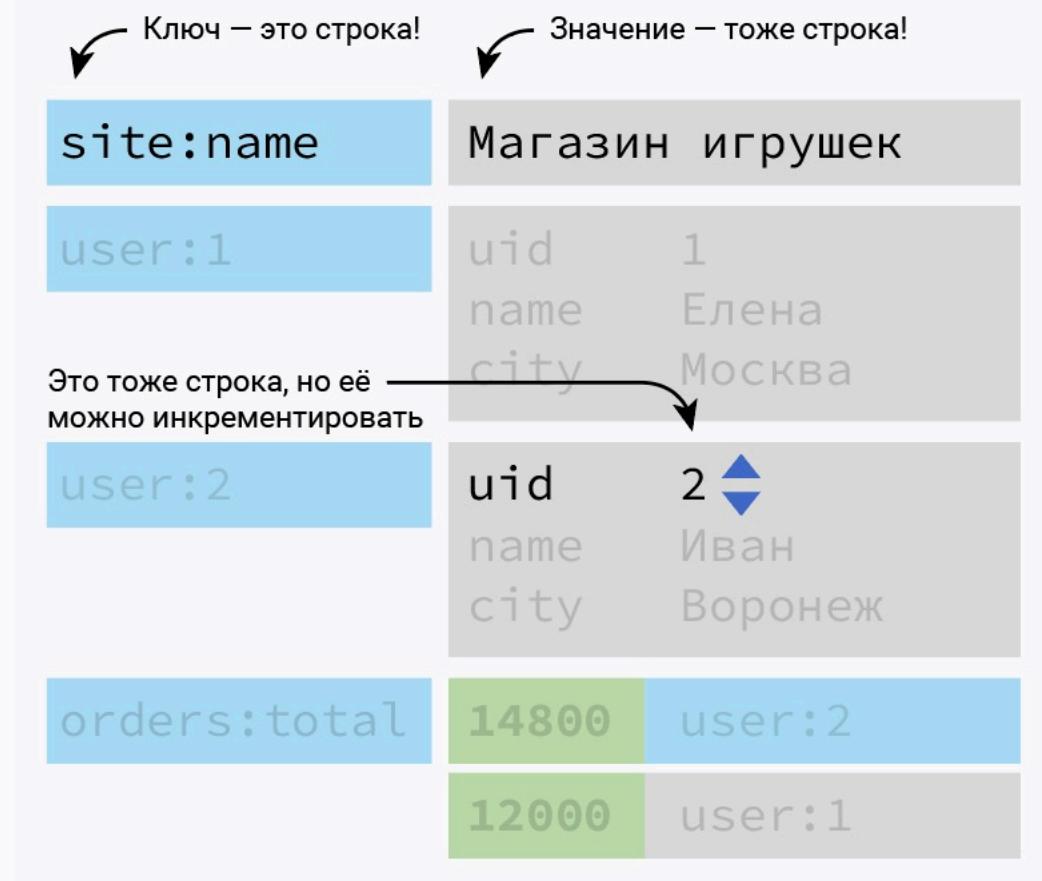
* есть нюансы реализации "под капотом"

Особенности хранения данных

2. Типы данных в Redis

Как следствие:

- значения других типов
(булевы, составные и т.д.)
нужно приводить к строкам
на клиенте



Особенности хранения данных

3. Время жизни данных в Redis

Особенности:

- для любого ключа можно указать время его жизни (TTL)
- по истечении этого времени и ключ, и значение будут автоматически удалены

site:name	Магазин игрушек	
user:1		
	uid	1
	name	Елена
	city	Москва
user:2	⌚ 15	Этот ключ (и значение) проживёт 15 секунд
	uid	2
	name	Иван
	city	Воронеж
orders:total	14800	user:2
	12000	user:1

Особенности хранения данных

3. Время жизни данных в Redis

Как следствие:

- с разработчиков снимается часть головной боли

site:name	Магазин игрушек	
user:1	uid	1
	name	Елена
	city	Москва
user:2	uid 2	
	name	Иван
	city	Воронеж
orders:total	14800	user:2
	12000	user:1

Проверка знаний

Вопросы на засыпку

1. К какому типу данных приводятся все значения в Redis?
2. У вас в Redis хранится режим работы сервиса (“Онлайн” / “Техобслуживание”). Предложите наиболее компактное представление этих данных.
3. Для каких сценариев может потребоваться время жизни ключей (TTL)?

Задача от бизнеса

Задача: кэширование ответов API

- у нас есть JSON API для списка пользователей
- мы хотим локально кэшировать ответы на GET-запросы
- записи в кэше должны храниться не более 10 минут

Пример:

```
GET /user/1  
  
{ "name": "Dmitry", "age": 41 }
```

Задача от бизнеса

Размышления о реализации

Ключ	cache:user:{uid}
Значение	JSON
Тип данных	Строка

Структуры данных: Строки

Строки

- хранятся в "глобальных" ключах
- время доступа к значению: $O(1)$
- размер значения – до 512 Mb
- значения бинарно-безопасны

site:name

Магазин игрушек

site:email

info@toy-shop.com

site:users

14

cache:front 90

<html>...</html>

Структуры данных: Строки

Основные команды

```
SET user:1:name "Dmitry"          # Установить значение  
GET user:1:name                  # Получить значение  
  
SET upload:counter 10           # Установить значение (число)  
INCR upload:counter            # Инкрементировать значение числа  
  
SET cache:api "json" EX 10     # Установить значение с TTL 10 сек  
TTL cache:api                  # Проверить TTL ключа  
GET cache:api                  # Если ключа нет, GET вернёт (nil)  
  
DEL upload:counter             # Удалить ключ  
  
KEYS user*                      # Вывести список ключей по шаблону  
KEYS *
```

Структуры данных: Строки

Сценарии использования

- кэширование
- хранение сессий
- атомарные счётчики

Структуры данных: Строки

Пара слов о ключах в Redis

Что это такое:

- аналог ключа в словаре или ассоциативном массиве
- название ключа может быть **любым**, Redis никак его не анализирует

site:name

Магазин игрушек

site:email

info@toy-shop.com

site:users

14

cache:front 90

<html>...</html>

Структуры данных: Строки

Пара слов о ключах в Redis

Как их называть:

- осмысленное название
- идём от общего к частному
- user:12:token

И ещё раз:

- двоеточия в имени ключа
нужны только для человека

site:name

Магазин игрушек

site:email

info@toy-shop.com

site:users

14

cache:front 90

<html>...</html>

Проверка знаний

Что вернёт каждая из этих команд?

1. SET user:1:name "Dmitry"

OK

2. GET user:1:name

"Dmitry"

3. GET user:1

?

4. SET user:1 "{name: Ivan}"

?

5. GET user:1:name

?

Задача от бизнеса

Задача: хранение информации о пользователе

- у нас есть список пользователей
- у каждого пользователя есть 20+ полей
- нам нужно хранить эту информацию в Redis

Вопрос на засыпку:

- почему для этого плохо подходят обычные ключи?

Структуры данных: Строки

Основные проблемы строк

- для больших сущностей
нужна уйма ключей
- обработку приходится
переносить на клиент

user:1:uid	1
user:1:name	Елена
user:1:email	elena@mail.ru
user:1:city	Москва
user:1:age	26
user:1:gender	Ж
user:1:is_active	1
user:2:uid	2
user:2:name	Иван

Структуры данных: Хэш-таблицы

Хэш-таблицы

- аналог **вложенных** ассоциативных массивов
- время доступа к значению внутри таблицы: $O(1)$
- отсутствует вложенность (только 1 уровень)
- **нет схемы данных**

user:1	uid	1
	name	Елена
	email	elena@mail.ru
	city	Москва
user:2	uid	2
	name	Иван
	email	ivan@mail.ru
	city	Воронеж
	age	45

Схемы данных нет: структуры могут отличаться

Задача от бизнеса

Размышления о реализации

Ключ

user:{uid}

Значение

Набор полей

Тип данных

Хэш-таблица

Структуры данных: Хэш-таблицы

Основные команды

HSET	<u>user:1</u> name "Dmitry"	# Установить значение
HSET	<u>user:1</u> age 40	
HGET	<u>user:1</u> name	# Получить значение поля
HSET	<u>user:1</u> name "Ivan"	# Изменить значение
HINCRBY	<u>user:1</u> age 1	# Инкрементировать значение
HGETALL	<u>user:1</u>	# Получить пары «поле-значение»
HKEYS	<u>user:1</u>	# Получить все поля
HVALS	<u>user:1</u>	# Получить все значения
HDEL	<u>user:1</u> age	# Удалить одно поле
DEL	<u>user:1</u>	# Удалить всю хэш-таблицу

Структуры данных: Хэш-таблицы

Сценарии использования

- хранение составных объектов
- вторичный ключ

Важно помнить:

- индексов, поиска и сортировки для хэш-таблиц **нет**
- типовой сценарий сохранения новых объектов:

```
INCR users:sequence          # Вернулось 15
HSET user:15 name ...        # Дальше сохраняем по этому ключу
```

Проверка знаний

Что вернёт каждая из этих команд?

1. HSET user:1 name "Dmitry" age 41 2

2. HSET user:2 name "Ivan" age 20 2

3. HGET user:1 name ?

4. GET user:1 ?

5. HSET user:3 name "Olga" vozrast 35 ?

Структуры данных: Хэш-таблицы

Это важно!

- у каждого типа данных – **свой набор команд**
- мы не можем выполнять команды одного типа при работе с другим типом

Задача от бизнеса

Задача: список последних публикаций пользователя

- у нас есть пользователи
- у каждого пользователя есть свой микроблог
- нам нужно оперативно получать N последних записей

Структуры данных: Хэш-таблицы

Проблема: хранение списков

- строки и хэш-таблицы плохо подходят для хранения последовательных списков
- мы можем быстро получить доступ к элементу по ключу
- но у нас будут сложности при переборе, удалении, добавлении...

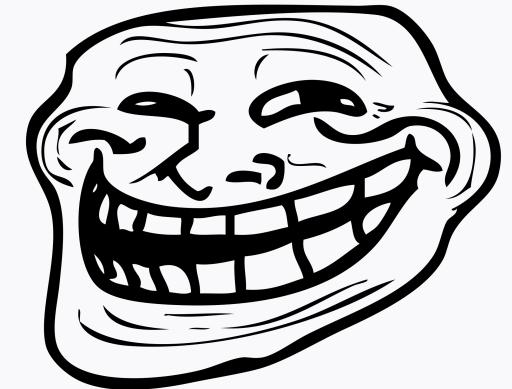
user:1:posts	
0	post:12
1	post:33
2	post:148
3	post:255
4	post:301
5	post:303

А что делать с ключами, если нам нужно удалить второй элемент?

А как узнать ключ, по которому лежит последний пост?

Мнение «иксперда»

*Ваш преподаватель врёт. Вставьте
несколько записей в хэш-таблицу Redis,
и они там станут строго по порядку.
Перебирай – не хочу)*



Структуры данных: Списки

Вопросы на засыпку:

Что такое связный список и чем он отличается от массива?

Массив (Array):

0	1	2	3
Иван	Сергей	Ирина	Степан

Связный список (Linked List):



Структуры данных: Списки

Списки

- под капотом связные списки
- время доступа
в начало/конец: $O(1)$
- время произвольного
доступа (в т.ч. поиск): $O(N)$
- могут содержать
неуникальные значения



Задача от бизнеса

Размышления о реализации

Ключ	user:{uid}:posts
Значение	Список публикаций по мере добавления
Тип данных	Список

Структуры данных: Списки

Основные команды

RPUSH	<u>user:1:posts</u>	post:10	# Добавить элемент в конец
RPUSH	<u>user:1:posts</u>	post:11	
RPUSH	<u>user:1:posts</u>	post:12	
LPUSH	<u>user:1:posts</u>	post:9	# Добавить элемент в начало
LTRIM	<u>user:1:posts</u>	0 2	# Обрезать список до 3 эл-тов
LLEN	<u>user:1:posts</u>		# Получить длину списка
LRANGE	<u>user:1:posts</u>	0 10	# Получить срез списка
LINDEX	<u>user:1:posts</u>	0	# Получить элемент по индексу
RPOP	<u>user:1:posts</u>		# Удалить элемент с конца
LPOP	<u>user:1:posts</u>		# Удалить элемент с начала

Структуры данных: Списки

Сценарии использования

- хранение событий в хронологическом порядке
- логирование
- event sourcing

Важно помнить:

- список со временем будет раздуваться
- частая практика — обрезать его длину после вставки

```
LPUSH user:1:posts post:22
```

```
LTRIM user:1:posts 0 9
```

Проверка знаний

Что вернёт каждая из этих команд?

- | | | |
|----|-----------------------------------|---|
| 1. | RPUSH user:1:posts post:1 | 1 |
| 2. | RPUSH user:1:posts post:2 | 2 |
| 3. | RPUSH user:1:posts post:2 | ? |
| 4. | RPUSH user:all:posts user:1:posts | ? |
| 5. | RPOP user:all:posts | ? |
| 6. | RPOP user:all:posts | ? |

Задача от бизнеса

Задача: список пользователей онлайн

- у нас есть пользователи
- они периодически заходят в систему
- нам нужно знать, сколько человек сейчас онлайн,
и кто эти люди

Структуры данных: Множества

Вопросы на засыпку:

— что из этого НЕ является множеством?

a)

1	2	3	4	5
---	---	---	---	---

b)

1	3	2	4	5
---	---	---	---	---

c)

1	2	3	4	4
---	---	---	---	---

d)

1	2	3	4	Z
---	---	---	---	---

Структуры данных: Множества

Множества

- содержат **только** уникальные значения
- **не сортированы**
- время добавления значения: $O(1)$
- время проверки существования значения: $O(1)$

user:1:cart

Сосиски

Шпроты

Килька

Агдам

Задача от бизнеса

Размышления о реализации

Ключ	users:online
Значение	ID пользователя
Тип данных	Множество

Структуры данных: Множества

Основные команды

SADD	<u>users:online</u>	user:1	# Добавить элемент
SADD	<u>users:online</u>	user:2	
SADD	<u>users:online</u>	user:12	
SADD	<u>users:online</u>	user:12	# Дубликат: мн-во не изменится
SCARD	<u>users:online</u>		# Получить мощность множества
SMEMBERS	<u>users:online</u>		# Получить элементы множества
SREM	<u>users:online</u>	user:2	# Удалить элемент из множества

Структуры данных: Множества

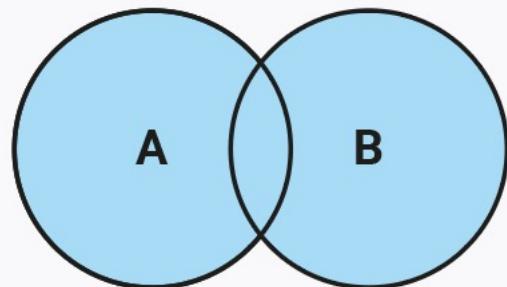
Сценарии использования

- аналитика
- хранение связей между сущностями
- таксономия (тэги)

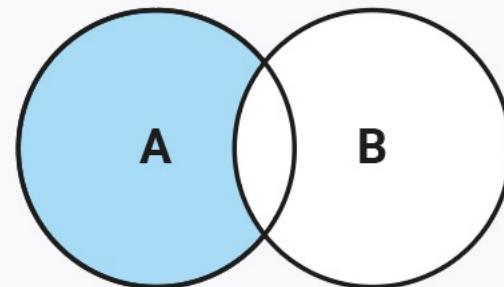
Структуры данных: Множества

Над множествами можно выполнять операции

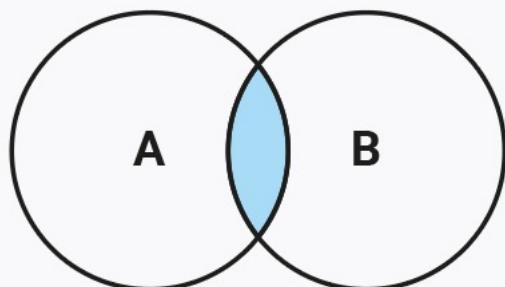
Объединение
(union)



Разность
(diff)



Пересечение
(intersect)



Результат любой операции — новое множество

Структуры данных: Множества

Операции над множествами

```
SDIFF users:all      users:online      # Разность  
SINTER users:online  users:premium   # Пересечение  
SUNION users:premium users:vip       # Объединение
```

Результат любой операции – новое **множество**

Проверка знаний

Вопросы на засыпку

1. Что будет, если добавить в множество значения 1 и "1"?
2. В SQL тоже есть UNION. Попробуйте вспомнить два основных правила для него.

Задача от бизнеса

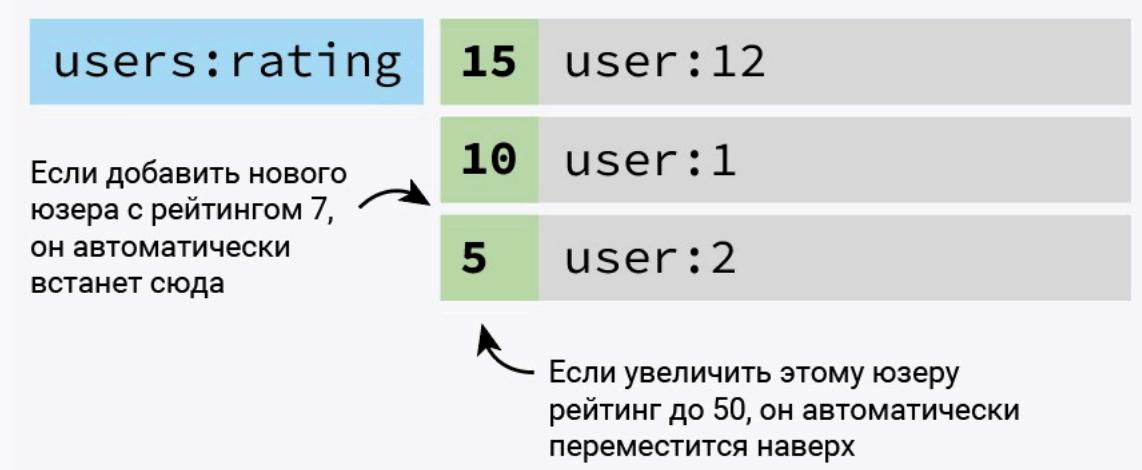
Задача: рейтинг пользователей

- у нас есть пользователи
- у каждого пользователя есть рейтинг (число)
- нам нужно оперативно получать N самых популярных пользователей

Структуры данных: Упорядоченные множества

Упорядоченные множества

- содержат **только** уникальные значения
- для каждого значения указывается счёт (score)
- значения автоматически сортируются по счёту
- время добавления и поиска – $O(\log n)$



Задача от бизнеса

Размышления о реализации

Ключ	users:rating
Значение	Счёт: рейтинг, значение: ID пользователя
Тип данных	Упорядоченное множество

Структуры данных: Упорядоченные множества

Основные команды

ZADD	<u>users:rating</u> 10 user:1	# Добавить элемент
ZADD	<u>users:rating</u> 5 user:2	
ZADD	<u>users:rating</u> 15 user:12	
ZADD	<u>users:rating</u> 15 user:12	# Дубликат: мн-во не изменится
ZCARD	<u>users:rating</u>	# Получить мощность множества
ZCOUNT	<u>users:rating</u> 5 10	# Кол-во эл-тов в диапазоне
ZRANGE	<u>users:rating</u> 5 10 BYSCORE	# Получить эл-ты в диапазоне
ZRANK	<u>users:rating</u> user:12	# Получить место эл-та в мн-ве
ZREM	<u>users:rating</u> user:12	# Удалить элемент из множества

Структуры данных: Упорядоченные множества

Сценарии использования

- рейтинги
- аналог индекса в РСУБД (значение – ключ сущности)

Важно помнить:

- для вывода в порядке убывания используются zrev*
- если обновить рейтинг, он перезапишет старый

Структуры данных

Что и когда использовать?

Записи	Порядок не важен	Порядок важен
Неуникальные	—	Список
Уникальные	Множество	Уп. множество

Структуры данных: Битовые строки

Вопросы на засыпку:

Каким будет результат выполнения этих операций?

0	1	0	1
1	0	0	1
<hr/>			

AND

Структуры данных: Битовые строки

Вопросы на засыпку:

Каким будет результат выполнения этих операций?

0	1	0	1
1	0	0	1
<hr/>			
0	0	0	1

AND

Структуры данных: Битовые строки

Вопросы на засыпку:

Каким будет результат выполнения этих операций?

0	1	0	1
1	0	0	1
<hr/>			
0	0	0	1

AND

0	1	0	1
1	0	0	1
<hr/>			
1	0	0	1

OR

Структуры данных: Битовые строки

Вопросы на засыпку:

Каким будет результат выполнения этих операций?

<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td colspan="4"><hr/></td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr></table>	0	1	0	1	1	0	0	1	<hr/>				0	0	0	1	AND	<table border="1"><tr><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td colspan="4"><hr/></td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr></table>	0	1	0	1	1	0	0	1	<hr/>				1	1	0	1	OR
0	1	0	1																																
1	0	0	1																																
<hr/>																																			
0	0	0	1																																
0	1	0	1																																
1	0	0	1																																
<hr/>																																			
1	1	0	1																																

Структуры данных: Битовые строки

Битовые строки

- это обычные строки
- к отдельным битам можно обращаться по индексу
- есть все основные битовые операции
- большинство команд выполняется за $O(N)$

views:2022	0	0	0	0	0	0	0	0	0
views:2021	0	0	1	0	0	0	1	0	0
views:2020	1	1	1	0	1	0	1	1	0

0 1 2 3 4 5 6 7

Нумерация битов в Redis идёт слева направо

Структуры данных: Битовые строки

Основные команды

SETBIT	<u>views:today</u>	1	1	# 00 1 0
SETBIT	<u>views:today</u>	3	1	# 1 010
GETBIT	<u>views:today</u>	2		# Получение бита по индексу
BITCOUNT	<u>views:today</u>			# Кол-во установленных битов
SETBIT	<u>views:yesterday</u>	0	1	# 000 1
SETBIT	<u>views:yesterday</u>	1	1	# 00 1 1
SETBIT	<u>views:yesterday</u>	2	1	# 0 111
BITOP OR	<u>views:total</u>			# Сюда будет записан результат
	<u>views:today</u>			# 1010
	<u>views:yesterday</u>			# 0111
				# -----
BITCOUNT	<u>views:total</u>			# 1111 (4 установленных бита)

Структуры данных: Битовые строки

Сценарии использования

- аналитика
- фильтрация с подсчётом

Важно помнить:

- при установке "далёкого" бита размер строки резко увеличивается
- для разреженных данных лучше подходят обычные множества

Проверка знаний

С помощью чего можно реализовать...

Хранение сессий

Общие знакомые Стаса и Жени

10 **лучших** постов Стаса

10 **последних** постов Стаса

...с датой, названием и тэгами

Строки

Множества

Упорядоченные множества

Списки

Хэш-таблицы (доп. запросы)

Варианты ответа: Строки, Хэш-таблицы, Списки, Множества, Упорядоченные множества

Проверка знаний

Вопросы для обсуждения:

Redis:

- заточен под быстрый произвольный доступ
- выполняет атомарные операции над данными
- позволяет перенести часть вычислений с клиента

Для каких сценариев вы бы использовали Redis – и в связке с какими БД?

03. Транзакции

Какую проблему мы хотим решить

- в Redis часто отправляются **наборы команд**
- при этом возникает существенный сетевой overhead
- это **заметно** влияет на производительность

Как решается эта проблема

Механизм пайплайнов

- реализуется на клиенте (**это не команда**)
- клиент в одном сообщении просто отправляет несколько команд, разделенных разрывом строки
- Redis в ответ возвращает сразу несколько ответов

Pipeline

Пример пайплайнов

Было (3 отдельных запроса):

```
SET test:1 10 EX 5
SET test:2 20 EX 5
INCR test:1
```

Стало (1 общий запрос):

```
SET test:1 10 EX 5\r\nSET test:2 20 EX 5\r\nINCR test:1\r\n
```

Pipeline

Плюсы и минусы пайплайна

Плюсы:

- тривиальная реализация на клиенте

Минусы:

- ответ, в т.ч. сообщения об ошибках, вернётся только после попытки **полного** выполнения всех команд
- Redis не гарантирует атомарность выполнения команд внутри пайплайна

Какую проблему мы хотим решить

- нам иногда нужно обработать несколько команд по принципу "всё или ничего"
- пайплайны для этого не подходят

Транзакции

Как решается эта проблема

Механизм транзакций

- во многом похож на транзакции в РСУБД
- клиент начинает транзакцию и отправляет команды, которые **становятся в очередь**
- команды начинают выполняться **только в момент фиксации транзакции**, причём они выполняются **атомарно**

Простая транзакция

```
MULTI          # Начало транзакции  
  
SET t1 "a"    # Эти команды не выполняются,  
SET t2 10     # а ставятся в очередь  
INCR t2      #  
  
EXEC         # Фиксация транзакции
```

Отмена транзакции

```
MULTI          # Начало транзакции  
  
SET t1 "a"  
SET t2 10  
INCR t2  
  
DISCARD      # Отмена транзакции
```

Обработка ошибок

Если возникла синтаксическая ошибка:

- транзакция сразу откатывается

Если возникла логическая ошибка:

- команды транзакции продолжают выполняться
- это может привести систему в неконсистентное состояние

Транзакция с синтаксической ошибкой

```
MULTI          # Начало транзакции  
  
SET t1 "a"  
SETT t1        # Ошибка  
SET t2 10  
  
EXEC          # Фиксация транзакции
```

Транзакция с логической ошибкой

```
MULTI          # Начало транзакции  
  
SET t1 "a"  
INCR t1      # Ошибка  
SET t2 10  
  
EXEC          # Фиксация транзакции
```

"Изоляция" транзакций

- все команды выполняются только в момент фиксации транзакции
- есть риск, что за это время другой клиент изменит значения нужных нам ключей
- Redis позволяет отслеживать конкретные ключи
- если их значение изменит кто-то другой — транзакция просто не выполнится

Транзакция с отслеживанием переменной

```
WATCH w1      # Начало отслеживания значения w1

SET t1 10      # Начиная с этого момента (даже до вызова
# MULTI!) и вплоть до вызова EXEC
# любое изменение w1
# приведёт к тому, что транзакция
# не выполнится (EXEC вернёт nil)
MULTI
SET t2 10
EXEC
```

Плюсы и минусы транзакций

Плюсы:

- гарантируют атомарность

Минусы:

- выполняются только в момент вызова EXEC
- могут привести к неожиданному состоянию данных

Транзакции

Вопросы для обсуждения:

Redis:

- однопоточный
- максимально простой
- предлагает атомарные транзакции

Можно ли утверждать, что Redis соответствует критериям ACID?

04. Секретный раздел

Что ещё умеет Redis

Расширенные возможности Redis

HyperLogLog

- способ оценить размер множества с высокой точностью (99,2%)
- занимает всего 12Kb **независимо** от кол-ва элементов

Географические данные

- поиск и фильтрация точек на карте
- применение: доставка, недвижимость, логистика...

Что ещё умеет Redis

Расширенные возможности Redis

Механизм подписки (Pub/Sub)

- одновременная рассылка нескольким клиентам
- применение: оповещения, чаты...

Потоки (Streams)

- полноценный брокер сообщений
- применение: разделение источников и обработки данных
- концепции: event sourcing, reactive programming...

Секретный раздел

Открытый урок

<https://youtu.be/F7G094ioQfE>

- разбираем строки
- изучаем необычное поведение INCR
- "ломаем" статистику потребления памяти



Цели вебинара | Проверка достижения целей

1 Определить, подходит ли Redis
для решения конкретной задачи

2 Выбрать оптимальный тип данных

3 Использовать Redis с учётом
специфических особенностей

Следующий вебинар

Тема: Redis, часть 2



Пятница, 23 декабря, в 20:00



Ссылка на вебинар будет в ЛК за 15 минут



Материалы к занятию
в ЛК – можно
изучать



Обязательный
материал обозначен
красной лентой



Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары



Дмитрий Кириллов

Технический директор
1С-Старт
[@esteps_kirillov](https://www.instagram.com/@esteps_kirillov)