



Бесплатная электронная книга

УЧУСЬ redis

Free unaffiliated eBook created from
Stack Overflow contributors.

#redis

.....	1
1: redis	2
.....	2
.....	2
Examples.....	2
.....	2
Redis.....	3
Redis "Hello World".....	4
Redis Docker.....	5
Redis Windows Node.js.....	5
2: Geo	7
.....	7
.....	7
Examples.....	7
GEOADD.....	7
GEODIST.....	7
3: Lua Scripting	8
.....	8
Examples.....	8
.....	8
4: Pub / Sub	10
.....	10
.....	10
.....	10
Examples.....	10
redis.....	10
5: Redis Keys	12
.....	12
.....	12
.....	12
Examples.....	13

.....	13
.....	13
.....	14
TTL	14
.....	15
Redis Keyspace.....	15
6: Redis List	17
.....	17
.....	17
.....	17
Examples.....	17
.....	17
.....	18
.....	18
7: Redis Set	19
.....	19
.....	19
.....	19
Examples.....	19
.....	19
.....	19
.....	20
8: Redis Java Jedis.....	22
.....	22
.....	22
Examples.....	22
Jedis.....	22
Redis.....	23
Get / Set.....	24
.....	24
9: redis Python.....	25
.....	25

.....	25
Examples.....	25
.....	25
.....	25
Redis.....	26
.....	26
.....	26
10:	28
.....	28
Examples.....	28
Redis	28
?.....	28
.....	28
.....	29
.....	29
.....	29
11:	30
.....	30
.....	30
.....	30
Examples.....	30
.....	30
.....	31
12: Redis String.....	33
.....	33
.....	33
Examples.....	33
.....	33
.....	34
13:	35
Examples.....	35

Redis	35
Redis	35
, Redis	35
Redis Cli	35
Redis	35
Redis Server Windows	36
14: Redis	38
.....	38
Examples	38
Redis	38
.....	38
.....	39

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [redis](#)

It is an unofficial and free redis ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official redis.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с redis

замечания

В этом разделе представлен обзор того, что такое Redis, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в Redis и ссылки на соответствующие темы. Поскольку документация для Redis является новой, вам может потребоваться создать начальные версии этих связанных тем.

Версии

Версия	Дата выхода
3.2.3	2016-08-02
3.2.2	2016-07-28

Examples

обзор

Redis - это удаленная база данных в оперативной памяти, которая предлагает высокую производительность, репликацию и уникальную модель данных для создания платформы для решения проблем. Redis - это открытый источник (лицензия BSD), структура данных в памяти, используемая как база данных, кеш и брокер сообщений. Он классифицируется как хранилище значений ключей NoSQL. Он поддерживает структуры данных, такие как строки, хэши, списки, наборы, отсортированные наборы с запросами диапазона, растровые изображения, гиперлогологи и геопространственные индексы с радиус-запросами.

Поддерживая пять различных типов структур данных,

1. STRING (Работайте со всей строкой, частями, целыми числами и поплавками)
2. СПИСОК (Push или pop предметов с обоих концов)
3. SET (добавление, выборка, удаление, проверка, пересечение, объединение, разность и т. Д.)
4. HASH (сохранение, удаление, удаление в хэше)
5. ZSET (тот же, что и установленный, но упорядоченным образом)
6. GEO (добавление, обновление, удаление широты и долготы, попадание в заданный radius)

Redis имеет встроенную репликацию, Lua-скриптинг, выключение LRU, транзакции и

различные уровни сохранения на диске (sync / async).

До версии 3 Redis работает в режиме «ведущий-ведомый» и требует Redis-Sentinel для обеспечения высокой доступности. Только мастер принимает записи и синхронизирует данные с его подчиненными устройствами путем форкирования.

Начиная с версии 3, Redis работает и рекомендует режим с несколькими мастерами, в котором встроены функции переключения на резервный ресурс, оверклование / парирование, перепродажа. Redis-Sentinel не требуется от версии-3. Для того, чтобы кластер redis работал, требуется минимум 3 основных узла / процессов.

Дополнительные функции - репликация, настойчивость и наложение на стороне клиента. Redis учитывает множество проблем, которые могут быть естественным образом сопоставлены с тем, что предлагает Redis, что позволяет решить ваши проблемы без необходимости выполнять концептуальную работу, требуемую другими базами данных.

Интерфейс командной строки Redis

`redis-cli` - это программа интерфейса командной строки Redis, которая позволяет отправлять команды Redis и читать ответы, отправленные сервером, непосредственно с терминала. Ниже приведено базовое использование командной строки:

Доступ к redis:

```
$ redis-cli
127.0.0.1:6379>
```

Доступ к redis при аутентификации:

```
$ redis-cli -a myPassword
127.0.0.1:6379>
```

Выберите базу данных и укажите размер базы данных (номер базы данных по умолчанию равен 0):

```
127.0.0.1:6379> dbsize
(integer) 2
127.0.0.1:6379> select 1
OK
127.0.0.1:6379[1]> dbsize
(integer) 20
```

Получите информацию и статистику о сервере:

```
127.0.0.1:6379> info
redis_version:2.4.10
redis_git_sha1:00000000
redis_git_dirty:0
```



```
arch_bits:64
multiplexing_api:epoll
gcc_version:4.4.6
process_id:947
uptime_in_seconds:873394
uptime_in_days:10
lru_clock:118108
used_cpu_sys:19.55
used_cpu_user:397.46
used_cpu_sys_children:0.00
used_cpu_user_children:0.00
connected_clients:1
connected_slaves:0
client_longest_output_list:0
client_biggest_input_buf:0
blocked_clients:0
used_memory:14295792
used_memory_human:13.63M
used_memory_rss:19853312
used_memory_peak:14295760
used_memory_peak_human:13.63M
mem_fragmentation_ratio:1.39
mem_allocator:jemalloc-2.2.5
loading:0
aof_enabled:0
changes_since_last_save:0
bgsave_in_progress:0
last_save_time:1468314087
bgrewriteaof_in_progress:0
total_connections_received:2
total_commands_processed:2
expired_keys:0
evicted_keys:0
keyspace_hits:0
keyspace_misses:0
pubsub_channels:0
pubsub_patterns:0
latest_fork_usec:0
vm_enabled:0
role:master
db0:keys=2,expires=0
db1:keys=20,expires=0
```

Выход из redis-cli:

```
127.0.0.1:6379> exit
```

Redis "Hello World"

Сначала вам нужно установить и запустить сервер Redis, проверьте приведенную ниже ссылку, которая поможет вам установить redis на ваш сервер или локальный компьютер.

Установка и настройка

Теперь откройте командную строку и запустите команду `redis-cli` :

Чтобы сохранить первый набор> SET 'keyname', затем 'value'

```
127.0.0.1:6379> SET hkey "Hello World!"
```

Нажмите Enter, чтобы увидеть

```
OK
```

Затем введите:

```
GET hkey
```

Тебе следует увидеть:

```
"Hello World!"
```

Пример вывода экрана:

```
127.0.0.1:6379> SET hkey "Hello World!"
OK
127.0.0.1:6379> GET hkey
"Hello World!"
127.0.0.1:6379>
```

Установите Redis с помощью Docker

Просто начать использовать Redis с помощью докеров:

```
docker pull redis
docker run -p 6379:6379 --rm --name redis redis
```

Теперь у вас работает экземпляр на порту 6397

Внимание: все данные будут удалены, когда Redis будет остановлен.

Чтобы подключить redis-cli, запустите еще один докер:

```
docker run -it --link redis:redis --rm redis redis-cli -h redis -p 6379
```

Теперь вы можете поиграть с дойкером redis.

Установка Redis в Windows с примером Node.js

У Redis есть порт Windows, предоставляемый Microsoft Open Technologies. Вы можете использовать установщик msi, расположенный по [адресу](https://github.com/MSOpenTech/redis/releases) :

<https://github.com/MSOpenTech/redis/releases>

После завершения установки вы можете увидеть, что «Redis» - это служба Windows (и ее статус должен быть «Started»)

Чтобы написать пример «Hello world», который использует Redis в Node.js (также в окнах), вы можете использовать следующий модуль npm: <https://www.npmjs.com/package/redis>

образец кода:

```
var redis = require('redis'),
    client = redis.createClient();

client.set('mykey', 'Hello World');
client.get('mykey', function(err, res) {
  console.log(res);
});
```

Прочитайте Начало работы с redis онлайн: <https://riptutorial.com/ru/redis/topic/1724/начало-работы-с-redis>

глава 2: Geo

Вступление

Redis предоставляет тип данных GEO для работы с геопространственными индексированными данными.

Синтаксис

- GEOADD ключ долготы широты член [долгота широта член ...]
- GEODIST ключ участник1 участник2 [блок]

Examples

GEOADD

Команда GEOADD позволяет пользователю добавлять геопространственную информацию (название элемента, долготу, широту) к определенному ключу.

Команда GEOADD может использоваться для добавления одного элемента к ключу:

```
GEOADD meetup_cities -122.43 37.77 "San Francisco"
```

или несколько элементов к ключу:

```
GEOADD meetup_cities -122.43 37.77 "San Francisco" -104.99 39.74 "Denver"
```

GEODIST

Команда GEODIST позволяет пользователю определять расстояние между двумя членами в пределах геопространственного индекса при указании единиц.

Чтобы найти расстояние между двумя городами встреч:

```
GEODIST meetup_cities "San Francisco" "Denver" mi
```

Прочитайте Geo онлайн: <https://riptutorial.com/ru/redis/topic/9091/geo>

глава 3: Lua Scripting

Вступление

Redis предоставляет несколько механизмов расширения функциональности базы данных. Одним из механизмов является использование серверных LUA-скриптов, которые могут быть выполнены для управления данными. Сценарии Lua могут быть полезны для выполнения дорогостоящих операций или для реализации атомных операций, требующих логики.

Examples

Команды для скриптов

Redis предоставляет семь различных операций для работы со скриптами:

- Эвальные операции (EVAL, EVALSHA)
- Операции SCRIPT (DEBUG, EXISTS, FLUSH, KILL, LOAD)

Команда EVAL оценивает сценарий, предоставляемый как строковый аргумент серверу. Скрипты могут получить доступ к указанным клавишам Redis, названным в качестве аргументов команды, и дополнительным строковым параметрам, которые пользователь хочет передать сценарию.

Например, команда:

```
EVAL "return {KEYS[1],KEYS[2],ARGV[1],ARGV[2]}" 2 key1 key2 first second
```

вызывает выполнение пользовательского сценария Lua, который просто возвращает предоставленные значения. Вызов включает в себя 2 клавиши Redis (key1 и key2) и два параметра.

Другой способ выполнения сценария Lua - сначала загрузить его в базу данных, а затем выполнить его с помощью SHA-хэша скрипта .:

```
> script load "return {KEYS[1],KEYS[2],ARGV[1],ARGV[2]}"  
"a42059b356c875f0717db19a51f6aaca9ae659ea"  
> evalsha "a42059b356c875f0717db19a51f6aaca9ae659ea" 2 key1 key2 foo bar  
1) "key1"  
2) "key2"  
3) "foo"  
4) "bar"
```

Команда загрузки сценария загружает скрипт и сохраняет его в базе данных. Шага-

подпись сценария возвращается, поэтому на нее можно ссылаться будущие вызовы.
Функция EVALSHA принимает sha и выполняет соответствующий скрипт из базы данных.

Прочитайте Lua Scripting онлайн: <https://riptutorial.com/ru/redis/topic/9112/lua-scripting>

глава 4: Pub / Sub

Вступление

Redis обеспечивает реализацию шаблона обмена публикацией / подпиской (Pub / Sub). Вместо того, чтобы отправлять сообщения конкретным приемникам, издатели отправляют сообщения заинтересованным получателям через какой-то механизм косвенности. Ресиверы задают интерес к конкретным сообщениям. В Redis эту функцию можно получить с помощью команд PUBLISH и SUBSCRIBE на каналах.

Синтаксис

- SUBSCRIBE канал [канал ...]
- UNSUBSCRIBE [канал [канал ...]]
- Сообщение канала PUBLISH
- PSUBSCRIBE pattern [pattern ...]
- PUNSUBSCRIBE [узор [узор ...]]

замечания

Чтобы обрабатывать pub / sub в redis, необходимо иметь **одного клиента для подписки и другого клиента для публикации**. Оба не могут обрабатываться одним клиентом. Хотя все остальные команды могут обрабатываться одним и тем же клиентом.

Examples

Опубликовать и подписаться с redis

Redis публикует / подписывается для отправки сообщений. Это осуществляется путем подписки на канал и публикацию на канал. Да, подписчики подписываются на один или несколько каналов. Издателю не нужно знать, кто все подписчики. Вместо этого издатель будет публиковать на конкретном канале. Все подписчики, подписавшиеся на этот канал, получают сообщение. Эта развязка издателей и подписчиков может обеспечить большую масштабируемость и более динамичную топологию сети.

Пример: Пользователь подписывается на 2 канала, скажем, foo & boo

```
SUBSCRIBE foo boo
```

В консоли redis-client1:

```
127.0.0.1:6379> SUBSCRIBE foo boo
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "foo"
3) (integer) 1
1) "subscribe"
2) "boo"
3) (integer) 2
```

Он начнет слушать сообщение. На публикацию будут получены данные для соответствующего канала.

Например: когда вы хотите отправить сообщение всем подписчикам, подключенным к boo, необходимо опубликовать их на этом канале.

```
PUBLISH boo "Hello Boo"
```

В консоли redis-client1:

```
1) "message"
2) "boo" //channel name
3) "Hello Boo" //Actual data
```

Чтобы отказаться от подписки на канал в любой момент, используйте

```
UNSUBSCRIBE // to unsubscribe from all channels
UNSUBSCRIBE foo // to unsubscribe from specific channel
```

Также можно подписаться на основе шаблона. Когда имя канала не обязательно / хочет подписаться на основе шаблона, используйте **PSUBSCRIBE** .

Аналогично отписке, основанной на шаблоне, используйте **PUNSUBSCRIBE**

Прочитайте Pub / Sub онлайн: <https://riptutorial.com/ru/redis/topic/5071/pub---sub>

глава 5: Redis Keys

Вступление

Ключевое слово Redis можно рассматривать как хеш-таблицу или ключи сопоставления словаря структурам данных в базе данных.

Redis предоставляет широкий спектр команд, которые работают с ключами для управления ключевым словом, включая возможность удаления ключей, проверки ключевых метаданных, поиска ключей и изменения определенных свойств ключей.

Синтаксис

- Шаблон KEYS
- Ключ PERSIST
- Секунды клавиши EXPIRE
- Временная метка ключа EXPIREAT
- Ключ TTL
- PEXPIRE ключ миллисекунды
- PEXPIREAT key milliseconds-timestamp
- Кнопка PTTL
- UNLINK [ключ ...]
- Клавиша DEL [ключ ...]
- Курсор SCAN [шаблон MATCH] [COUNT count]

замечания

Для действительных символов в клавишах Redis [руководство объясняет это полностью](#) :

Клавиши Redis бинарно безопасны, это означает, что вы можете использовать любую двоичную последовательность в виде ключа, начиная с строки, например «foo», до содержимого файла JPEG. Пустая строка также является допустимым ключом.

Несколько других правил о ключах:

Очень длинные ключи - это не очень хорошая идея, например, ключ с 1024 байтами - плохая идея не только по памяти, но и потому, что поиск ключа в наборе данных может потребовать нескольких дорогостоящих ключевых сравнений. Даже когда задача состоит в том, чтобы соответствовать наличию большого значения, прибегнуть к хэшированию (например, с SHA1), это лучшая идея, особенно с точки зрения памяти и пропускной способности.

Очень короткие ключи часто не очень хорошая идея. Существует мало смысла писать «u1000flw» в качестве ключа, если вы можете вместо этого написать «user: 1000: followers». Последний является более читаемым, а добавленное пространство является незначительным по сравнению с пространством, используемым самим объектом ключа и объектом value. В то время как короткие клавиши, очевидно, потребляют немного меньше памяти, ваша задача - найти правильный баланс.

Постарайтесь придерживаться схемы. Например, «object-type: id» - хорошая идея, как в «user: 1000». Точки или тире часто используются для многословных полей, как в комментарии «1234: reply.to» или «comment: 1234: reply-to».

Максимально допустимый размер ключа - 512 МБ.

Будьте осторожны с использованием команды KEYS против производственной системы, это может вызвать серьезные проблемы с производительностью. Если вам нужно выполнить поиск в ключевом пространстве, команды [SCAN](#) являются лучшей альтернативой.

Examples

Действующие ключи

Клавиши Redis являются двоично-безопасными, поэтому буквально все может использоваться как ключ. Единственные ограничения в том, что они должны быть меньше 512 МБ.

Примеры действительных ключей:

```
7
++++
`~!@#$%^&*()-_+=+
user:10134
search/9947372/?query=this%20is%20a%28test%29%20query
<div id="div64">

Any other string less than 512MB in size.
The raw binary content of an image or other binary file.
An entire multi-line text document.
An entire SQL query.
Any integer, hexadecimal, octal, or binary value.
Anything else you can think of less than 512MB in size.
```

Недействительные ключи Redis:

```
Anything larger than 512MB.
```

Ключевые схемы именования

Для ясности и ремонтпригодности часто рекомендуется разработать систему или схему для обозначения ваших ключей Redis. Вот несколько примеров общих и поддерживаемых систем для обозначения ваших ключей:

```
user:10134
user:10134:favorites
user:10134:friends
user:10134:friends-of-friends

user:10134
user:10134/favorites
user:10134/friends
user:10134/friends.of.friends

user/10134
user/10134/favorites
user/10134/friends
user/10134/friends of friends
```

Обратите внимание, что, хотя это разрешено, большие клавиши используют больше памяти и приводят к более медленному времени поиска, поэтому использование ключа 500 МБ может быть отличной идеей для производительности. Лучшей идеей может быть использование SHA-1, SHA-256 или MD5 хэша большого двоичного объекта в качестве ключа:

```
image/9517bb726d33efdc503a43582e6ea2eea309482b
image52e9df0577fca2ce022d4e8c86b1eccb070d37bef09dec36df2fabbfa7711f5c
```

Перечисление всех ключей

Вы можете перечислить все ключи в базе данных Redis, выполнив следующие команды из `redis-cli`:

```
KEYS *
```

Параметр для `KEYS` - это шаблонное соответствие шаблону. Примеры поддерживаемых шаблонов включают:

```
h?llo matches hello, hallo and hxllo
h*llo matches hllo and heeeello
h[ae]llo matches hello and hallo, but not hillo
h[^e]llo matches hallo, hbllo, ... but not hello
h[a-b]llo matches hallo and hbllo
```

Использование команды `KEYS *` может отрицательно сказаться на производительности, поэтому ее не рекомендуется использовать для производственных экземпляров. Используйте операцию `SCAN` для поиска ключей в производственном коде.

TTL и срок действия ключа

Значения срока действия ключа могут управляться пользователем вне команд обновления. Redis позволяет пользователю определить текущее время жизни (TTL) ключа с помощью команды TTL:

```
TTL key
```

Эта команда вернет TTL ключа в секундах или вернет специальные значения -1 или -2. А -1 указывает, что ключ является постоянным (не истекает), а -2 указывает, что ключ не существует.

Ключ с истечением срока действия может быть выполнен с использованием команды PERSIST:

```
PERSIST KEY
```

и постоянный ключ может быть истечен с использованием команды EXPIRE:

```
EXPIRE KEY seconds
```

Expire также может использоваться для изменения TTL существующего ключа. Кроме того, вы можете использовать команду EXPIREAT с отметкой времени UNIX, чтобы установить время истечения срока действия.

Существуют миллисекундные версии команд TTL, EXPIRE и EXPIREAT с префиксом P.

Удаление ключей

Redis предоставляет две функции для удаления ключей из базы данных: del и unlink.

Функция del удаляет из базы данных один или несколько ключей. Команда del заставляет Redis немедленно восстановить память для удаленной клавиши в текущем потоке выполнения. Время выполнения для del пропорционально количеству отдельных элементов, удаленных из всех ключей.

Функция unlink действует как команда del, она удаляет один или несколько ключей из базы данных. Однако, в отличие от команды del, любая память, используемая этими ключами, восстанавливается асинхронно в другом потоке.

Сканирование Redis Keyspace

Redis предоставляет команду SCAN для итерации по ключам в базе данных, соответствующей определенному шаблону. Redis поддерживает сопоставление шаблонов стиля glob в команде SCAN.

Команда SCAN предоставляет итератор на основе курсора в ключевом пространстве Redis.

Итерационная последовательность вызовов для SCAN начинается с того, что пользователь совершает вызов с аргументом указателя, установленным в 0. Результатом этого вызова является пакет элементов и обновленный курсор, который передается на следующий вызов SCAN. Эта итерация продолжается до тех пор, пока Redis не вернет курсор 0.

Следующая функция Python демонстрирует основное использование SCAN:

```
def scan_keys(r, pattern):
    "Returns a list of all the keys matching a given pattern"

    result = []
    cur, keys = r.scan(cursor=0, match=pattern, count=2)
    result.extend(keys)
    while cur != 0:
        cur, keys = r.scan(cursor=cur, match=pattern, count=2)
        result.extend(keys)

    return result
```

Команда SCAN является рекомендуемым способом поиска ключей в базе данных и рекомендуется по команде `KEYS *`.

Прочитайте Redis Keys онлайн: <https://riptutorial.com/ru/redis/topic/3916/redis-keys>

глава 6: Redis List Тип данных

Вступление

Тип данных List в Redis - это упорядоченный набор элементов, на который ссылается ключ Redis. Redis позволяет вам получать доступ и изменять список по индексам или push / pop. В Redis два конца списка называются левыми и правыми. Левый соответствует первому элементу или голове списка, а правый соответствует последнему элементу или хвосту списка.

Синтаксис

- Значение ключа LPUSH [значение ...]
- Значение ключа RPUSH [значение ...]
- Кнопка LPOP
- Кнопка RPOP
- Клавиша LLEN

замечания

Более подробную информацию о типе списка List и всех командах, которые можно использовать в сочетании с ними, можно найти в официальной документации Redis по адресу [Redis.io](https://redis.io).

Examples

Добавление элементов в список

Redis позволяет добавлять элементы вправо или влево от списка.

Если бы я работал со списком, my_list и я хотели бы добавить 3 к списку, я мог бы сделать это с помощью команды Redis LPUSH:

```
LPUSH my_list 3
```

Если бы я хотел добавить 3 в my_list, я бы вместо этого использовал команду RPUSH:

```
RPUSH my_list 3
```

Команда LPUSH и RPUSH автоматически создадут для вас новый список, если поставляемый ключ не существует. Две альтернативные команды LPUSHX и RPUSHX

могут использоваться только для работы с ключом списка, если он уже существует.

Получение элементов из списка

Redis предоставляет команды LPOP и RPOP в качестве аналога команд LPUSH и RPUSH для извлечения элементов данных.

Если я работал со списком `my_list`, в котором уже было несколько элементов данных, я могу получить первый элемент в списке, используя команду LPOP:

```
LPOP my_list
```

Результат этой команды вернет значение первого элемента из списка и удалит его из `my_list`. Например, если бы у меня был список `[1, 3, 2, 4]`, и я применил к нему LPOP, после этого у меня был бы список `[3, 2, 4]` в памяти.

Аналогично, я могу удалить из конца списка с помощью RPOP:

```
RPOP my_list
```

вернет значение для последнего элемента из списка, а затем удалит его из `my_list`. Используя наш пример, `[1, 2, 3, 4]` после вызова RPOP в этом списке, список в памяти будет `[1, 2, 3]`.

Размер списка

Размер списка Redis можно определить с помощью команды LLEN. Если у меня есть список из четырех элементов, хранящихся в ключе `my_list`, я могу получить размер, используя:

```
LLEN my_list
```

который вернется 4.

Если пользователь указывает ключ, который не существует для LLEN, он возвращает ноль, но если используется ключ, который указывает на элемент другого типа данных, будет возвращена ошибка.

Прочитайте [Redis List Тип данных онлайн](https://riptutorial.com/ru/redis/topic/9107/redis-list-тип-данных): <https://riptutorial.com/ru/redis/topic/9107/redis-list-тип-данных>

глава 7: Redis Set Тип данных

Вступление

Redis поддерживает набор типов данных, аналогичных математическим наборам для моделирования данных в базе данных. Наборы представляют собой составной тип данных, состоящий из группы уникальных неупорядоченных членов. Устанавливает поддержку добавления и удаления членов, операций с размерами, а также комбинированных операций, которые принимают два набора и генерируют третий набор. Наборы в Redis аналогичны наборам на большинстве языков программирования.

Синтаксис

- SADD ключевой элемент [член ...]
- Ключевой элемент SISMEMBER
- Клавиша SCARD
- SADD ключевой элемент [член ...]

замечания

Полную документацию по набору данных типа Redis можно найти в [Redis.io](https://redis.io).

Examples

Размер набора

Размер набора может быть определен с помощью команды SCARD. SCARD вернет мощность набора или количество элементов в наборе. Например, если у меня есть Redis, установите `my_set`, который хранится в базе данных, похожей на (Apple, Orange, Banana), я мог бы получить размер, используя следующий код:

```
SCARD my_set
```

В случае моего набора примеров это вернет 3. Если пользователь выполнит команду SCARD для ключа, которого не существует, Redis вернет 0.

Добавление элементов в набор

Основная команда Redis для добавления элемента в набор - SADD. Он принимает ключ и один или несколько членов и добавляет их в набор, хранящийся на данном ключе.

Например, скажем, что я хотел создать набор с элементами apple, pear и banana. Я мог бы выполнить одно из следующих действий:

```
SADD fruit apple
SADD fruit pear
SADD fruit banana
```

или же

```
SADD fruit apple pear banana
```

После выполнения либо у меня будет набор фруктов с тремя предметами.

Попытка добавить элемент, который уже находится в наборе, не будет иметь никакого эффекта. После того, как я установил свой набор фруктов, используя код выше, если я снова попытаюсь добавить яблоко:

```
SADD fruit apple
```

Redis попытается добавить яблоко в набор фруктов, но поскольку он уже установлен, ничего не изменится.

Результатом команды SADD всегда является количество элементов, добавленных Redis в набор. Поэтому попытка повторного добавления яблока вернет результат 0.

Элементы-члены в Redis чувствительны к регистру, поэтому Apple и Apple рассматриваются как два отдельных элемента.

Тестирование членства

Redis предоставляет команду SISMEMBER, чтобы проверить, является ли определенный элемент уже членом набора. Используя команду SISMEMBER, я могу проверить и посмотреть, является ли яблоко уже членом моего набора фруктов.

Если я создам свой набор фруктов из предыдущего примера, я могу проверить и посмотреть, содержит ли он яблоко, используя следующий тест:

```
SISMEMBER fruit apple
```

SISMEMBER вернет 1, поскольку элемент уже существует.

Если я попытаюсь увидеть, является ли собака членом моего набора фруктов:

```
SISMEMBER fruit dog
```

Redis вернет 0, поскольку собака не находится в наборе фруктов.

Если пользователь пытается использовать команду SISMEMBER с ключом, который не существует, Redis вернет 0, указывающий на отсутствие членства, но если вы используете SISMEMBER с ключом, который уже содержит не заданный тип данных, Redis вернет ошибку.

Прочитайте Redis Set Тип данных онлайн: <https://riptutorial.com/ru/redis/topic/9109/redis-set-тип-данных>

глава 8: Как подключиться к Redis в Java с помощью Jedis

Вступление

Существует более десяти различных клиентских библиотек для использования с Redis в Java. Одним из самых популярных клиентов является [Jedis](#).

замечания

Дальнейшая информация:

- [Клиенты Java Redis](#)
- [Репозиторий Jedis Github](#)
- [Документация / Wiki](#)

Examples

Получение Jedis

Библиотека Jedis обычно добавляется в проект Java, используя систему управления зависимостями, встроенную в среду сборки проекта. Двумя популярными системами построения Java являются Maven и Gradle.

Использование Gradle

Чтобы добавить библиотеку Jedis в проект Gradle, вам потребуется настроить репозиторий и добавить зависимость. Следующий фрагмент показывает, как добавить версию 2.9.0 библиотеки Jedis в проект Gradle.

```
repositories {  
    mavenCentral()  
}  
  
dependencies {  
    compile 'redis.clients:jedis:2.9.0'  
}
```

Использование Maven

Чтобы добавить Jedis в проект Maven, вам нужно добавить зависимость в список зависимостей и предоставить координаты библиотеки. Следующий фрагмент будет добавлен в ваш файл pom.xml:

```
<dependencies>
  <dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>2.9.0</version>
  </dependency>
</dependencies>
```

Подключение к Redis

Использование пула

Большинство кода захотят подключиться к Redis, используя пул общих объектов подключения. Подключение к Redis с использованием пула включает в себя два разных блока кода. Во время инициализации вашему приложению необходимо создать пул соединений:

```
JedisPoolConfig poolCfg = new JedisPoolConfig();
poolCfg.setMaxTotal(3);

pool = new JedisPool(poolCfg, hostname, port, 500, password, false);
```

`JedisPoolConfig` предоставляет опции для настройки пула.

Поскольку ваше приложение обрабатывает рабочую нагрузку, вам нужно будет получить соединение из общего пула, используя следующий код:

```
try (Jedis jedis = pool.getResource()) {

    ...

}
```

Лучшей практикой является получение объекта соединения `Jedis` из пула в блоке `try-with-resources`.

Без бассейнов

В некоторых случаях, таких как простое приложение или тест интеграции, вы можете не иметь дело с общими пулами и вместо этого напрямую создавать объект соединения `Jedis`. Это можно сделать с помощью следующего кода:

```
try (Jedis jedis = new Jedis(hostname, port)) {
    jedis.connect();
    jedis.auth(password);
    jedis.select(db);

    . . .
}
```

Опять же, лучшей практикой является создание объекта клиента `Jedis` в блоке `try-with-`

resources.

Выполнение базовых команд Get / Set

Как только вы установили соединение с Redis, вы можете получить и установить значения с помощью объекта соединения `Jedis` :

Получить

```
String value = jedis.get(myKey);
```

Задавать

```
jedis.put(myKey, "some value");
```

Выполнение команд

Чтобы выполнить команду Redis с помощью `Jedis`, вы вызываете вызовы методов против объекта `Jedis` , созданного вами из пула. `Jedis` выдает команды Redis как вызовы методов, например:

```
- String get(String key)
- Long geoadd(String key, double longitude, double latitude, String member)
- List<String> hmget(String key, String... fields)
- Long hsetnx(String key, String field, String value)
```

Если вы хотите установить значение `String` в Redis, вы будете использовать блок кода, подобный:

```
try (Jedis jedis = pool.getResource()) {

    String myKey = "users:20";
    String myValue = "active";

    jedis.set(myKey, myValue);
}
```

Прочитайте [Как подключиться к Redis в Java с помощью Jedis онлайн](https://riptutorial.com/ru/redis/topic/9712/как-подключиться-к-redis-в-java-с-помощью-jedis):

<https://riptutorial.com/ru/redis/topic/9712/как-подключиться-к-redis-в-java-с-помощью-jedis>

глава 9: Подключение к redis с использованием Python

Вступление

Для подключения к Redis в Python требуется использование клиентской библиотеки. Для Python существует много разных клиентских библиотек, но **redis-py** - один из самых популярных клиентов.

После установки клиентской библиотеки вы можете получить доступ к Redis в своем приложении, импортировав соответствующий модуль, установив соединение и выполнив команду.

замечания

Чтобы подключиться к redis с помощью python, вам нужно установить **клиент** . Вы можете установить с помощью pip, используя:

```
pip install redis
```

это установит **redis-py**

Возможно, вы захотите установить **hiredis-py**, который делегирует синтаксический анализ сообщений протокола клиенту C hiredis. Это может обеспечить значительное улучшение производительности во многих ситуациях. Вы можете установить hiredis с помощью pip, выполнив:

```
pip install hiredis
```

Examples

Добавить элемент в список

```
import redis

r = redis.StrictRedis(host='localhost', port=6379, db=0)

r.lpush('myqueue', 'myelement')
```

Добавление полей в хэш

В Redis (HSET и HMSET) есть две основные функции: добавление полей в хэш-ключ. Обе

функции доступны в `redis-py`.

Использование HSET:

```
import redis

r = redis.StrictRedis(host='myserver', port=6379, db=0)
r.hset('my_key', 'field0', 'value0')
```

Использование HMSET:

```
import redis

r = redis.StrictRedis(host='myserver', port=6379, db=0)
r.hmset('my_key', {'field0': 'value0', 'field1': 'value1', 'field2': 'value2'})
```

Настройка подключения к Redis

Клиент `redis-py` предоставляет два класса `StrictRedis` и `Redis` для установления базового соединения с базой данных Redis. Класс `Redis` предоставляется для обратной совместимости, а новые проекты должны использовать класс `StrictRedis`.

Одним из рекомендуемых способов установления соединения является определение параметров соединения в словаре и передача словаря в конструктор `StrictRedis` с использованием синтаксиса `**`.

```
conn_params = {
    "host": "myredis.somedomain.com",
    "port": 6379,
    "password": "sekret",
    "db": 0
}

r = redis.StrictRedis(**config)
```

Создание транзакции

Вы можете установить транзакцию, вызвав метод `pipeline` на `StrictRedis`. Команды Redis, выполненные против транзакции, выполняются в одном блоке.

```
# defaults to transaction=True
tx = r.pipeline()
tx.hincrbyfloat(debit_account_key, 'balance', -amount)
tx.hincrbyfloat(credit_account_key, 'balance', amount)
tx.execute()
```

Выполнение команд напрямую

Redis-py предоставляет метод `execute_command` для непосредственного вызова операций

Redis. Эта функция может использоваться для доступа к любым модулям, которые не могут иметь поддерживаемый интерфейс в клиенте redis-ру. Например, вы можете использовать команду `execute_command` для отображения всех модулей, загруженных на сервер Redis:

```
r.execute_command('MODULE', 'LIST')
```

Прочитайте Подключение к redis с использованием Python онлайн:

<https://riptutorial.com/ru/redis/topic/9103/подключение-к-redis-с-использованием-python>

глава 10: Резервное копирование

Вступление

Резервное копирование удаленного экземпляра Redis может быть достигнуто с помощью репликации. Это полезно, если вы хотите сделать снимок набора данных перед обновлением, удалением или изменением базы данных Redis.

Examples

Резервное копирование удаленного экземпляра Redis в локальный экземпляр

На машине, где вы хотите сделать резервную копию, перейдите в CLIS Redis:

```
redis-cli
```

Пароль?

Если ваш мастер Redis DB (тот, который вы хотите реплицировать), имеет пароль:

```
config set masterauth <password>
```

Начало репликации

Для начала репликации выполните следующее:

```
SLAVEOF <host> <port>
```

Для проверки выполнения репликации выполняется:

```
INFO replication
```

И вы должны увидеть вывод следующим образом:

```
# Replication
role:slave
master_host:some-host.compute-1.amazonaws.com
master_port:6519
master_link_status:up
master_last_io_seconds_ago:3
master_sync_in_progress:0
slave_repl_offset:35492914
```

```
slave_priority:100
slave_read_only:1
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
```

Обратите внимание на `master_link_status` должна быть `up` .

Проверка синхронизации

Когда синхронизация завершена, `INFO replication` должна показывать:

```
master_sync_in_progress:0
```

Чтобы проверить, что набор данных был синхронизирован, вы можете сравнить размер базы данных:

```
DBSIZE
```

Сохранение дампа данных на диск

Чтобы сохранить БД на диск асинхронно:

```
BGSAVE
CONFIG GET dir
```

Затем вы должны найти файл `dump.rdb` в каталоге, указанном командой `config`.

Закрытие репликации

Вы можете остановить репликацию с помощью:

```
SLAVEOF NO ONE
```

Ссылка: [руководство по тиражированию Redis](#)

Прочитайте Резервное копирование онлайн: <https://riptutorial.com/ru/redis/topic/9369/резервное-копирование>

глава 11: Сортированные наборы

Вступление

Тип данных Sorted Set в Redis является упорядоченной версией типа данных Set. Сортированный набор Redis состоит из коллекции уникальных членов. Каждый член в сортированном наборе может рассматриваться как пара, состоящая из члена и оценки. Счет используется для упорядочивания членов в наборе в порядке возрастания.

Синтаксис

- ZADD ключ [NX | XX] [CH] [INCR] оценка участник [оценка участник ...]
- Ключ ZCARD
- Ключ ZCOUNT мин. Макс.
- Ключ ZLEXCOUNT мин. Макс.

замечания

Официальную документацию для Sorted Sets можно найти на сайте [Redis.io](https://redis.io).

Сортированные наборы иногда называются zsets. Если вы используете команду TYPE на сортированном наборе ключей, значение zset будет возвращено.

Examples

Добавление элементов в отсортированный набор

Redis предоставляет команду ZADD для добавления элементов в отсортированный набор. Основной формой команды ZADD является указание набора, добавляемого элемента и его оценки. Например, если бы я хотел построить упорядоченный набор моей любимой пищи (от наименьшего к большему), я мог бы использовать любой из:

```
zadd favs 1 apple
zadd favs 2 pizza
zadd favs 3 chocolate
zadd favs 4 beer
```

или альтернативно:

```
zadd favs 1 apple 2 pizza 3 chocolate 4 beer
```

Функция ZADD работает аналогично функции SADD с несортированным множеством.

Результатом команды ZADD является количество элементов, которые были добавлены. Поэтому, после создания моего набора, как указано выше, если я снова попытаюсь пить ZADD:

```
ZADD favs 4 beer
```

Я бы получил 0 результат, если бы решил, что лучше шоколад, чем пиво, я мог бы выполнить:

```
ZADD favs 3 beer 4 chocolate
```

чтобы обновить мои настройки, но я все равно получаю результат с возвратом 0, так как пиво и шоколад уже находятся в наборе.

Подсчет элементов в отсортированном наборе

Redis предоставляет три команды для подсчета элементов в отсортированном наборе: ZCARD, ZCOUNT, ZLEXCOUNT.

Команда ZCARD является основным тестом для мощности набора. (Это аналогично команде SCARD для наборов.). ZCARD возвращает количество элементов набора. Выполнение следующего кода для добавления элементов в набор:

```
zadd favs 1 apple
zadd favs 2 pizza
zadd favs 3 chocolate
zadd favs 4 beer
```

работает ZCard:

```
zcard favs
```

возвращает значение 4.

Команды ZCOUNT и ZLEXCOUNT позволяют подсчитать подмножество элементов в отсортированном наборе на основе диапазона значений. ZCOUNT позволяет вам подсчитывать элементы в определенном диапазоне баллов, а ZLEXCOUNT позволяет подсчитать количество элементов в определенном лексикографическом диапазоне.

Используя наш набор выше:

```
zcount favs 2 5
```

вернет 3, так как есть три предмета (пицца, шоколад, пиво), которые имеют баллы от 2 до 5 включительно.

ZLEXCOUNT предназначен для работы с наборами, где каждый элемент имеет одинаковую оценку, форсирование и упорядочивание имен элементов. Если мы создали такой набор, как:

```
zadd favs 1 apple
zadd favs 1 pizza
zadd favs 1 chocolate
zadd favs 1 beer
```

мы могли бы использовать ZLEXCOUNT для получения количества элементов в определенном лексографическом диапазоне (это делается побайтовым сравнением с использованием функции `memscr`).

```
zlexcount favs [apple (chocolate
```

вернется 2, поскольку два элемента (яблоко, пиво) попадают в диапазон яблока (включительно) и шоколада (эксклюзивный). В качестве альтернативы мы могли бы сделать оба конца включительно:

```
zlexcount favs [apple [chocolate
```

и получить результат 3.

Прочитайте [Сортированные наборы онлайн](https://riptutorial.com/ru/redis/topic/9111/сортированные-наборы): <https://riptutorial.com/ru/redis/topic/9111/сортированные-наборы>

глава 12: Тип данных Redis String

Вступление

Redis предоставляет строковый тип данных, который используется для связывания данных с определенным ключом. Строка Redis - это самый простой тип данных, доступный в Redis, и один из первых типов данных, с которыми пользователи учатся работать.

Строки часто связаны с текстовыми данными, но строки Redis больше похожи на буферы, которые могут использоваться для хранения широкого спектра различных данных. Строки Redis могут использоваться для представления целых чисел, чисел с плавающей запятой, растровых изображений, текста и двоичных данных.

Синтаксис

- Значение ключа SET [EX секунд] [PX миллисекунды] [NX | XX]
- Ключ INCR
- Приращение ключа INCRBY
- Ключевое приращение INCRBYFLOAT
- Кнопка DECR
- Уменьшение ключа DECRBY

Examples

Работа со строками как целые числа

Несколько команд позволяют работать со строками, представляющими целочисленные значения.

Пользователь может установить целочисленное значение ключа с помощью команды:

```
SET intkey 2
```

Команда set создаст ключ, если необходимо, или обновит его, если он уже существует.

Значение целочисленного ключа может быть обновлено на сервере с помощью команд INCR или INCRBY. INCR увеличит значение ключа на 1, а INCRBY увеличит значение ключа по предоставленному шагу.

```
INCR intkey  
INCRBY intkey 2
```

Если значение ключа, указанного для INCR или INCRBY, не может быть выражено как целое число, Redis вернет ошибку. Если ключ не существует, ключ будет создан, и операция будет применена к значению по умолчанию 0.

Команды DECR и DECRBY работают в обратном порядке для уменьшения значения.

Работа со строками в виде чисел с плавающей запятой

Redis позволяет использовать тип данных String для хранения чисел с плавающей запятой.

Пользователь может установить значение поплавка ключа с помощью команды:

```
SET floatkey 2.0
```

Команда set создаст ключ, если необходимо, или обновит его, если он уже существует.

Значение ключа может быть обновлено на сервере с помощью команды INCRBYFLOAT. INCRBYFLOAT увеличит значение ключа по предоставленному значению приращения.

```
INCRBYFLOAT floatkey 2.1
```

Если значение ключа, указанного для INCRBYFLOAT, не может быть выражено как плавающая точка, Redis вернет ошибку. Если ключ не существует, ключ будет создан, и операция будет применена к значению по умолчанию 0.0.

Ключи могут быть уменьшены путем передачи отрицательного приращения команде INCRBYFLOAT.

Прочитайте Тип данных Redis String онлайн: <https://riptutorial.com/ru/redis/topic/9507/тип-данных-redis-string>

глава 13: Установка и настройка

Examples

Установка Redis

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make
```

Запуск Redis

```
redis-server
```

Проверьте, работает ли Redis

```
redis-cli ping
```

Это должно вернуть PONG

Доступ к Redis Cli

Предполагая, что вы используете redis server на localhost, вы можете ввести команду

```
redis-cli
```

После появления этой команды выполните команду redis командной строки

```
127.0.0.1:6379>
```

Типы данных Redis

Ниже приведен список всех структур данных, поддерживаемых Redis:

- **Бинарно-безопасные строки**
- **Списки** : коллекции элементов строки, отсортированные в соответствии с порядком вставки.
- **Наборы** : коллекции уникальных, несортированных строковых элементов.
- **Сортированные наборы** : аналогичны **наборам**, но где каждый строковый элемент связан с числом с плавающим числом, называемым счетом.
- **Хеши** : это карты, состоящие из полей, связанных со значениями.
- **HyperLogLogs** : это вероятностная структура данных, которая используется для

оценки мощности множества.

Основываясь на официальной документации **redis.io**
















Установка и запуск Redis Server в Windows

Примечание. Проект Redis официально не поддерживает Windows.

Тем не менее, группа **Microsoft Open Tech** разрабатывает и поддерживает этот порт Windows, ориентированный на Win64. [Официальный redis.io/download](https://redis.io/download)

Вы можете загрузить различные версии или последнюю версию Redis github.com/MSOpenTech/redis/releases

1. **Загрузите** файл .msi или .zip, этот учебник позволит вам загрузить последний zip-файл [Redis-x64-3.2.100.zip](#).
2. **Извлеките zip-файл** в подготовленный каталог.

This PC > BackUp (E:) > redis				
Name	Date modified	Type	Size	
 EventLog.dll	01/07/2016 16:27	Application extens...		
 Redis on Windows Release Notes.docx	01/07/2016 16:07	Microsoft Word D...		
 Redis on Windows.docx	01/07/2016 16:07	Microsoft Word D...		
 redis.windows.conf	01/07/2016 16:07	CONF File		
 redis.windows-service.conf	01/07/2016 16:07	CONF File		
 redis-benchmark.exe	01/07/2016 16:28	Application		
 redis-benchmark.pdb	01/07/2016 16:28	PDB File	4	
 redis-check-aof.exe	01/07/2016 16:28	Application		
 redis-check-aof.pdb	01/07/2016 16:28	PDB File	3	
 redis-cli.exe	01/07/2016 16:28	Application		
 redis-cli.pdb	01/07/2016 16:28	PDB File	4	
 redis-server.exe	01/07/2016 16:28	Application	1	
 redis-server.pdb	01/07/2016 16:28	PDB File	6	
 Redis-x64-3.2.100.zip	14/04/2017 14:19	WinRAR ZIP archive	5	
 Windows Service Documentation.docx	01/07/2016 09:17	Microsoft Word D...		

3. **Запустите redis-server.exe**, вы можете напрямую запустить redis-server.exe, щелкнув или запустив его в командной строке.

```
E:\redis\redis-server.exe

[8992] 14 Apr 14:44:30.147 # Warning: no config file specified, using the default
t config. In order to specify a config file use E:\redis\redis-server.exe /path/
to/redis.conf

Redis 3.2.100 (00000000/0) 64 bit

Running in standalone mode
Port: 6379
PID: 8992

http://redis.io

[8992] 14 Apr 14:44:30.150 # Server started, Redis version 3.2.100
[8992] 14 Apr 14:44:30.150 * The server is now ready to accept connections on po
rt 6379
```

4. **Запустите redis-cli.exe** после успешного запуска redis-сервера. Вы можете получить к нему доступ и проверить команды, запустив redis-cli.exe Те

```
E:\redis\redis-cli.exe

127.0.0.1:6379>
```

Команда **PING** используется для проверки работоспособности соединения.

```
E:\redis\redis-cli.exe

127.0.0.1:6379> ping
PONG
127.0.0.1:6379> ping "hello world"
"hello world"
127.0.0.1:6379>
```

Теперь вы можете начать использовать Redis, пожалуйста, обратитесь к дополнительным командам в [официальных документах](#)

Прочитайте Установка и настройка онлайн: <https://riptutorial.com/ru/redis/topic/2898/установка-и-настройка>

глава 14: Хранилище сохранения Redis

Вступление

Redis поддерживает два основных режима сохранения: RDB и AOF. Режим упорства RDB берет моментальный снимок вашей базы данных в определенный момент времени. В режиме RDB Redis отключает процесс сохранения базы данных на диск. AOF регистрирует каждую операцию, выполненную против сервера, в журнал повтора, который можно обрабатывать при запуске, чтобы восстановить состояние базы данных.

Examples

Отключите все хранилища сохраняемости в Redis

Существует два вида режимов постоянного хранения в Redis: AOF и RDB. Чтобы временно отключить RDB, выполните следующие команды в командной строке Redis:

```
config set save ""
```

для временного отключения AOF выполните из командной строки Redis следующее:

```
config set appendonly no
```

Изменения будут сохраняться до тех пор, пока сервер не будет перезапущен, а затем сервер вернется обратно в любые режимы, настроенные в файле `redis.conf` сервера.

Команда `CONFIG REWRITE` может использоваться для изменения файла `redis.conf`, чтобы отразить любые динамические изменения конфигурации.

Получить статус сохранения настойчивости

Следующий код получит текущую конфигурацию для состояния постоянной памяти. Эти значения могут быть изменены динамически, поэтому они могут отличаться от конфигурации в `redis.conf`:

```
# get
config get appendonly
config get save
```

Прочитайте Хранилище сохранения Redis онлайн: <https://riptutorial.com/ru/redis/topic/7871/хранилище-сохранения-redis>

кредиты

S. No	Главы	Contributors
1	Начало работы с redis	Ahamed Mustafa M , Alexander V. , Aminadav , Community , Florian Hämmerle , Itamar Haber , Prashant Barve , RLaaa , Sagar Ranglani , sirin
2	Geo	Tague Griffith
3	Lua Scripting	Tague Griffith
4	Pub / Sub	jerry , Tague Griffith
5	Redis Keys	Tague Griffith , Will
6	Redis List Тип данных	Tague Griffith
7	Redis Set Тип данных	JonyD , Tague Griffith
8	Как подключиться к Redis в Java с помощью Jedis	Tague Griffith
9	Подключение к redis с использованием Python	Gianluca D'Ardia , Tague Griffith , ystark
10	Резервное копирование	odlp
11	Сортированные наборы	Tague Griffith
12	Тип данных Redis String	Tague Griffith
13	Установка и настройка	Cristiana214 , Gianluca D'Ardia , Sagar Ranglani
14	Хранилище сохранения Redis	Aminadav , Tague Griffith