**Системная и программная инженерия**

Лабораторная работа №3 «Знакомство с Redis»

**Введение**

В предыдущей лабораторной работе мы познакомились с базовыми понятиями, командами и структурами данных в Redis. В данной лабораторной работе мы продолжим работу с Redis.

**Практическая часть**

В лабораторной работе №2 для удобства запускался docker-контейнер с образом Redis, в этой лабораторной работе также во всех примерах будет использоваться docker.

**List (списки)**

Списки Redis реализованы через связанные списки. Это означает, что даже если у вас внутри списка миллионы элементов, операция добавления нового элемента в начало или в конец списка выполняется за константное время. Скорость добавления нового элемента с помощью команды LPUSH в начало списка из десяти элементов такая же, как добавление элемента в начало списка из 10 миллионов элементов.

В чем недостаток? Доступ к элементу по индексу выполняется очень быстро в списках, реализованных с помощью массива (индексированный доступ с постоянным временем), и не так быстро в списках, реализованных посредством связанных списков (где операция требует объема работы, пропорционального индексу элемента, к которому осуществляется доступ).

Списки Redis реализованы со связанными списками, потому что для системы баз данных крайне важно иметь возможность очень быстро добавлять элементы в очень длинный список.

Команда LPUSH <key> <value> добавляет новый элемент в список слева (в начало), а команда RPUSH <key> <value> добавляет новый элемент в список справа (в конец):

Изображение выглядит как текст

Автоматически созданное описание

Рис. 1. Выполнение команд «*rpush*» и «*lpush*»

Команда LRANGE извлекает диапазоны элементов из списков:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 2. Выполнение команды «*lrange*»

Оба индекса могут быть отрицательными, указывая Redis, что нужно начинать отсчет с конца: таким образом, -1 — это последний элемент, -2 — предпоследний элемент списка и так далее.

LPOP <key> удаляет и возвращает элемент из начала списка; RPOP <key> делает то же самое, но из конца списка:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 3. Выполнение команд «*lpop*» и «*rpop*»

**Ограниченные списки**

Во многих случаях мы просто хотим использовать списки для хранения последних элементов, какими бы они ни были: обновления социальных сетей, журналы или что-то еще.

Redis позволяет нам использовать списки как ограниченную коллекцию, запоминая только последние N элементов и отбрасывая все самые старые элементы с помощью команды LTRIM.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 4. Выполнение команд «*ltrim*»

**Распространенные варианты использования списков**

Списки полезны для ряда задач, два очень типичных варианта использования:

* Запоминать о последних обновлениях, размещенных пользователями в социальной сети.
* Для использования шаблона потребитель-производитель, при котором производитель помещает элементы в список, а потребитель потребляет эти элементы и выполняет действия.

**Sets (Наборы)**

Sets Redis — это неупорядоченные неповторяющиеся наборы строк (элементов). Команда SADD добавляет новые элементы в набор. Также можно выполнять ряд других операций над множествами, например проверять, существует ли данный элемент, выполнять пересечение, объединение или различие между несколькими множествами и т. д.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 5. Выполнение команды «*sadd*»

В Redis есть команды для проверки членства. Например, проверка существования элемента:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 6. Выполнение команды «*sismember*»

«3» входит в набор, а «322» — нет.

Команда SINTER, которая выполняет пересечение между различными множествами:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 7. Выполнение команды «*sinter*»

С помощью команды SUNION, которая выполняет объединение между различными множествами:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 8. Выполнение команды «*sunion*»

Команда SPOP удаляет случайный элемент, возвращая его клиенту:

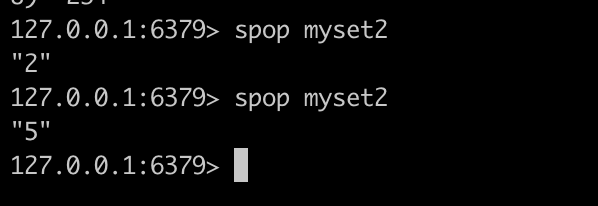


Рис. 9. Выполнение команды «*spop*»

**Некоторые примеры использования наборов Redis:**

* Отслеживание уникальных элементов (например, отслеживание всех уникальных IP-адресов).
* Представления отношений (например, набор всех пользователей с заданной ролью).

**Сортированные наборы**

Сортированные наборы — это тип данных, похожий на смесь набора и хэша. Как и наборы, отсортированные наборы состоят из уникальных неповторяющихся строковых элементов, поэтому в некотором смысле отсортированный набор также является набором.

Однако, хотя элементы внутри наборов не упорядочены, каждый элемент в отсортированном наборе связан со значением с плавающей запятой, называемым оценкой.

Они упорядочиваются по следующему правилу:

Если B и A — два элемента с разной оценкой, то A > B, если A.score > B.score.

Если B и A имеют одинаковую оценку, то A > B, если строка A лексикографически больше, чем строка B. Строки B и A не могут быть равными, поскольку отсортированные наборы содержат только уникальные элементы.

ZADD добавляет новый элемент и связанную с ним оценку в отсортированный набор. Если член уже существует, оценка обновляется:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 10. Выполнение команды «*zadd*»

ZRANGE возвращает элементы отсортированного набора, отсортированные в заданном диапазоне:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 11. Выполнение команды «*zrange*»

Также можно вернуть баллы, используя аргумент WITHSCORES:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 12. Выполнение команды «*zrange*» вместе с «*withscores*»

Возьмем всех лиц, родившихся до 1950 года включительно. Для этого используем команду ZRANGEBYSCORE:

Изображение выглядит как текст

Автоматически созданное описание

Рис. 13. Выполнение команды «*zrangebyscore*»

**Некоторые варианты использования отсортированных наборов:**

* Таблицы лидеров. Например, вы можете использовать отсортированные наборы, чтобы легко поддерживать упорядоченные списки самых высоких результатов в крупной онлайн-игре.
* Ограничители скорости. В частности, вы можете использовать отсортированный набор для создания ограничителя скорости, чтобы предотвратить чрезмерные запросы API.

**Команды для работы с геоданными**

Геопространственные индексы Redis позволяют хранить координаты и искать их. Эта структура данных полезна для поиска ближайших точек в пределах заданного радиуса.

Структура индекса Geo — это просто отсортированный набор.

Чтобы добавить новый список (или новый элемент в существующий список) в хранилище Redis используется команда **GEOADD**

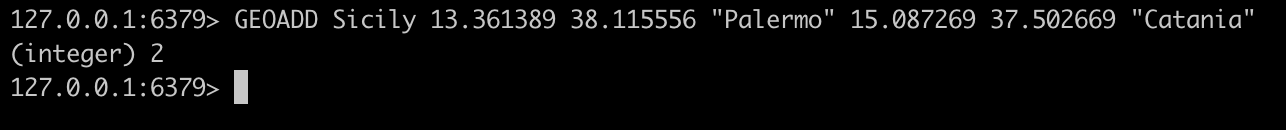


Рис. 14. Выполнение команды «*GEOADD*»

**Команда GEODIST** возвращает расстояние между двумя элементами в геопространственном индексе.

Если один или оба члена отсутствуют, команда возвращает NULL.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 15. Выполнение команды «*GEDIST*»

Единица измерения должна быть одной из следующих, по умолчанию это метры:

* **m** - метры.
* **km** – километры.
* **mi -** мили.
* **ft** - футы.

**Команда GEOSEARCH** возвращает всех членов отсортированного набора, заполненного геопространственной информацией которые находятся в пределах указанной области.

Центральная точка запроса предоставляется одним из следующих обязательных параметров:

* FROMMEMBER: использует позицию существующего <member> в отсортированном наборе.
* FROMLONLAT: Используйте <longitude> и <latitude> .

Форма запроса предоставляется одним из следующих обязательных параметров:

* BYRADIUS: аналогично GEORADIUS, поиск внутри круглой области по заданному <radius>.
* BYBOX: Поиск внутри прямоугольника, определяемого <height>и <width>.

Команда дополнительно возвращает дополнительную информацию, используя следующие параметры:

* WITHDIST: Возвращает расстояние возвращенных элементов от указанной центральной точки. Расстояние возвращается в тех же единицах, которые указаны для аргументов радиуса или высоты и ширины.
* WITHCOORD: Возвращает долготу и широту совпадающих элементов.
* WITHHASH: Возвращает необработанный отсортированный набор очков, закодированный с помощью геохэша, в виде 52-битного целого числа без знака

По умолчанию совпадающие элементы возвращаются несортированными. Чтобы отсортировать их, используйте один из следующих двух вариантов:

* ASC: сортировка возвращенных элементов от ближайшего к дальнему относительно центральной точки.
* DESC: Сортировка возвращенных элементов от самого дальнего к ближайшему относительно центральной точки.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 16. Выполнение команды «*GEOSEARCH*»

**Redis persistence**

Redis persistence - относится к записи данных в долговременное хранилище. Redis предоставляет ряд вариантов сохранения. К ним относятся:

* RDB (Redis Data Base): сохранение с помощью RDB выполняет моментальные снимки вашего набора данных на определенный момент времени через определенные промежутки времени.
* AOF (Append Only File): сохранение с помощью AOF регистрирует каждую операцию записи, полученную сервером. Затем эти операции можно воспроизвести снова при запуске сервера, реконструируя исходный набор данных. Команды регистрируются в том же формате, что и сам протокол Redis.
* Нет сохранения: вы можете полностью отключить сохранение. Это иногда используется при кэшировании.
* RDB + AOF : вы также можете комбинировать AOF и RDB.

Для резервного копирования текущего набора данных в Redis используется команда SAVE. Команда создаст снимок, содержащий все данные в кластере Redis в двоичном формате файла dump.rdb.

Изображение выглядит как текст

Автоматически созданное описание

Рис. 17. Выполнение команды «*SAVE*»

Для восстановления ваших наборов данных переместите файл резервной копии dump.rdp в корень сервера Redis.

**Практическое задание**

1. Продемонстрировать основные операции со списками (LPUSH, RPUSH, LPOP,RPOP, LLEN, LRANGE, LTRIM). Данные для демонстрации работы команд можно взять производные.
2. Продемонстрировать основные операции с наборами (SADD, SINTER, SUNION, SPOP). Данные для демонстрации работы команд можно взять производные.
3. Продемонстрировать основные операции с сортированными наборами (ZADD, ZRAGNE, ZRANK, ZRANGEBYSCORE). Данные для демонстрации работы команд можно взять производные.
4. Необходимо взять координаты 3-х произвольных точек в городе Москва и вывести на экран расстояние:
5. От первой точки до второй
6. От первой точки до третьей
7. От второй точки до третьей

**Теоретические вопросы**

1. Что такое списки (Lists) в Redis? Перечислите основные команды для работы с наборами.
2. Какова временная сложность добавления элемента в начало и конец списка?
3. Какие есть недостатки в работе со списками?
4. Приведите практический пример применения списков.
5. Что такое наборы (sets) в Redis? Перечислите основные команды для работы с наборами.
6. В чем отличие наборов от сортированных наборов в Redis?
7. Приведите практический пример применения наборов и сортированных наборов.
8. Расскажите об геопространственных индексах Redis, перечислите основные команды, приведите практические примеры работы с геопространственными индексами Redis.