ЛАБОРАТОРНА РОБОТА №10
Створення БД типу ключ-значення на базі Redis.
Застосування Jedis в Java додатках

Мета: знайомство з розподіленим сховищем Redis, дослідження та застосування Jedis для роботи з Redis в Java додатках.

Рекомендовані джерела:

- Introduction to Redis
- Introduction to Redis: Installation, CLI Commands, and Data Types
- Intro to Jedis – the Java Redis Client Library
- Введення в Jedis – клієнтську бібліотеку Java Redis
- Redis: Redis installation on Windows 10
- Встановлення та налаштування Redis для різних ОС

ПРАКТИЧНЕ ЗАВДАННЯ

1) За допомогою redis-cli в командному рядку продемонструвати вміння застосовувати базові операції взаємодії із розподіленим сховищем Redis.

Завдання 1.1

Створити ключ, який повинен містити в собі ваше ім'я, іменування університету, групу та номер в групі. Після чого необхідно вивести значення ключа students:name.

Приклад створення ключа:

set students:yourname "{name: YOUR_NAME, university: CHNU, group: IPZ-243, id: number_in_group]}"

Завдання 1.2

- Знайти довжину значення зв'язаного із ключем students:yourname.
- Вивести підрядок починаючи з порядкового номеру студента у списку підгрупи та закінчуючи 40-м із рядка зв'язаного із ключем students:yourname.
- Додати до цього ж значення рядок «extra !data!» та вивести кінцеве значення.

Завдання 1.3

- Створити хеш таблицю students:hash_yourname з полями name: your_name, university: chnu, group: ipz-243, id: number_in_group
- Добавити поле age з відповідним значенням.
- Вивести всі значення полів.
- Вивести значення поля id.
- Видалити поле university.

Завдання 1.4

- Створити список який описує будь-яку послідовність дій від 5 дій. Ключ – ваше прізвище.

Приклад – це дії користувача на сайті.

lpush user:123 open:page1 open:page2 order:ticket order:ticket2 buy:all

- Вивести всі значення у вашому списку (lrange).
- Скоротити список до 3 (останніх, тобто самих актуальних) елементів (ltrim) однією командою.

Завдання 1.5

- Створити множину предметів першого і другого семестрів (від 5 предметів).

Приклад:

sadd subject:1 subjectX ...
sadd subject:2 subjectY ...

- Взяти будь-який предмет із списку і перевірити, чи є він у множині першого і другого семестру.
- Перевірити чи є предмети, які спільні для першого і другого семестру та зберегти їх під іншим ключем (sinterstore).

Завдання 1.6

- Створити впорядкований список предметів за перший семестр з їх оцінками (від 5 елементів).
- Дізнатись кількість предметів, в яких оцінка більше 70.
- Вивести порядковий рейтинг одного із предметів відносно інших (zrevrank).

Завдання 1.7

- Створити ключ, який буде видалений за певний час в секундах.
- Створити ключ, який буде видалений в день вашого народження в 2025 році.
- Дізнатись скільки часу ваші ключі будуть існувати.

2) Для оптимізованої моделі предметної області (з ЛР№2) розробити БД типу ключ-значення в Redis, в якій кожна колекція відповідатиме таблиці.

3) Розробити Java-додаток та використовуючи Jedis реалізувати базові CRUD операції (create, read, update, delete) для кожної колекції.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Redis – це сховище структури даних із відкритим вихідним кодом (ліцензія BSD), яке використовується як база даних, кеш-пам’ять, брокер повідомлень і механізм потокового передавання.

Redis надає такі структури даних, як рядки, хеші, списки, набори, відсортовані набори із запитами діапазонів, растрові зображення, гіперлоглоги, геопросторові індекси та потоки.

Рядки

Рядки найпростіша структура даних у Redis. Коли ви думаєте про пару ключ-значення, то ви думаєте про рядки. Маючи унікальні імена (ключі), значення рядків можуть буди будь-якими.

Ми вже бачили типовий приклад використання рядків: зберігання об’єктів з відповідними ключами. Це саме те, з чим вам доведеться працювати найчастіше:

set users:leto "{name: leto, planet: dune, likes: [spice]}"

Додатково, Redis дозволяє виконувати деякі стандартні дії з рядками. Наприклад:

- *strlen <ключ>* використовується для обчислення довжини значення, пов'язаного з ключем;
- *getrange <ключ> <початок> <кінець>* повертає підрядок з рядка;
- *append <ключ> <значення>* додає введене значення до кінця існуючого рядка (або створює нове значення, якщо зазначений ключ не визначений).

Приклади команд:

strlen users:leto
(integer) 42

getrange users:leto 27 40
"likes: [spice]"

append users:leto " OVER 9000!!"
(integer) 54

Хеш (словники)

Хеш – гарний приклад того, чому Redis не просто сховище ключ-значення. Хеш багато в чому схожий на рядок. Важливе те, що вони додають окремий рівень адресації даних – поля (fields). Еквівалентом команди *set* і *get* є *hset* та *hget.*

hset users:goku powerlevel 9000
hget users:goku powerlevel

За допомогою хешів ми можемо отримувати відразу декілька полів, всі поля зі значеннями та видаляти окремі поля. Хеші дають трішки більше контролю, а ніж рядки.

hmset users:goku race saiyan age 737
hmget users:goku race powerlevel
hgetall users:goku
hkeys users:goku
hdel users:goku age

Замість того, щоб зберігати дані про користувача у вигляді одного серіалізованого значення, ми можемо використовувати хеш для більш точного представлення. Перевагою буде можливість вилучення, зміни та видалення окремих частин даних без необхідності читати і записувати все значення повністю.

Розглядаючи хеш як структуровані об'єкти, такі як дані користувача, важливо розуміти, як вони працюють. Такий підхід може бути корисний для збільшення продуктивності, проте хеш може використовуватися для організації даних і зручності запитів до них, саме для цього вони особливо корисні.

Списки (lists)

Списки дозволяють зберігати та маніпулювати масивами значень для заданого ключа. Можна добавляти значення в список, отримувати перше і останнє значення із списку і маніпулювати значення із заданими індексами. Списки зберігають порядок своїх значень і мають ефективні операції із використанням індексів.

Множини (Sets)

Множини використовуються для зберігання унікальних значень і представляють набір операцій – таких, як об'єднання. Множини не впорядковані, але представляють ефективні операції із значеннями.

Наприклад, список друзів являється класичним прикладом множин:

sadd friends:leto ghanima paul chani jessica
sadd friends:duncan paul jessica alia

Незалежно від того, скільки друзів має користувач, ми можемо ефективно (O(1)) визначити, чи є користувач userX і userY друзями чи ні.

sismember friends:leto jessica
sismember friends:leto ivan

Впорядковані множини (Sorted Sets)

Останній і самий потужний тип – це впорядковані множини. Хеши схожі на рядки, але мають поля, а впорядковані множини схожі на множини, але мають лічильники. Лічильники надають можливість впорядковувати і ранжирувати.

Наприклад, якщо ми хочемо отримати список друзів, ми можемо зробити наступне:

zadd friends:leto 100 ghanima 95 paul 95 chani 75 jessica 1 vladimir

Робота із запитами set | get (ключі та значення)

Не дивлячись на те, що Redis більше, ніж просто сховище типу ключ-значення, в його основі кожна з п'яти використовуючих структур даних має, як мінімум, ключ і значення. Дуже важливо розібратися в тому, що таке ключ і що таке значення, перед тим як рухатись далі.

Ключ – це те, чим ми маркуємо частини інформації (ідентифікатори). Ми будемо користуватися ключами часто, але поки досить знати, що ключ може виглядати так users:leto.

Під таким ключем можна очікувати інформацію про користувача під іменем leto. Двокрапка не має ніякого особливого значення, але використання подібних розділювачів є гарним тоном написання.

Значення – це дані, які асоційовані з ключем. Це може бути що завгодно. Іноді – це рядки, іноді числа, а іноді там зберігаються серіалізовані об'єкти (у вигляді JSON, XML або будь-яких інших форматів). В основному, Redis розглядає значення, як масив байт і не цікавиться тим, що вони собою представляють. Зверніть увагу, що різні драйвера виконують серіалізацію по різному. Тому, тут ми будемо говорити тільки про рядки (string), цілі числа (integer) та JSON.

Давайте попрактикуємось. Виконайте наступні команди:

set users:leto "{name: leto, planet: dune, likes: [spice]}"

Майже всі команди Redis виглядають подібним чином. Спочатку йде сама команда, в нашому випадку set, а потім параметри. Команда set приймає два параметри: ключ який ми зберігаємо, і значення, яке пов'язане з ним. Багато, але не всі, команди приймають аргументом ключ (це зазвичай перший параметр).

Для того, щоби отримати значення ключа необхідно вказати команду *get*.

get users:leto

Ключі Redis є бінарними, це означає, що ви можете використовувати будь-яку двійкову послідовність як ключ, від рядка на зразок «foo» до вмісту файлу JPEG. Порожній рядок також є дійсним ключем.

Ще кілька правил щодо ключів:

- Дуже довгі ключі не є гарною ідеєю. Наприклад, ключ розміром 1024 байти є поганою ідеєю не тільки з точки зору пам'яті, але й тому, що пошук ключа в наборі даних може вимагати кількох дорогих порівнянь ключів. Навіть якщо поставлене завдання полягає в тому, щоб зіставити існування великого значення, його хешування (наприклад, за допомогою SHA1) є кращою ідеєю, особливо з точки зору пам'яті та пропускної здатності.
- Дуже короткі ключі часто не є гарною ідеєю. Немає сенсу писати «u1000flw» як ключ, якщо замість цього можна написати «user:1000:followers». Останній є більш читабельним, а доданий простір незначний у порівнянні з простором, який використовується самим ключовим об'єктом і об'єктом значення. Хоча короткі ключі, очевидно, споживатимуть трохи менше пам'яті, ваше завдання знайти правильний баланс.
- Спробуйте дотримуватися схеми. Наприклад, «тип об'єкта:id» є хорошою ідеєю, як у «user:1000». Крапки або тире часто використовуються для багатослівних полів, як-от «comment:4321:reply.to» або «comment:4321:reply-to».

Адміністрування Redis

TTL

Redis дозволяє назначати ключам час існування (TTL). Ви можете використовувати абсолютні значення в часі в форматі Unix (Unix timestamp https://www.unixtimestamp.com/), або час, який залишився ключу існувати.

expire pages:about 30
expireat pages:about 1600473600

Перша команда видалить ключ через 30 секунд. Друга це зробить в 00:00, 19 вересня 2020 року. Це робить Redis ідеальною базою для кешування. Ви можете дізнатись, скільки ще буде існувати даний елемент з допомогою команди *ttl*, а також видалити термін існування з допомогою команди *persist.*

Налаштування паролю

Redis можна налаштувати так, щоб він вимагав пароль. За це відповідає спеціальна опція конфігурація *requirepass* (встановлюється через *redis.conf* або командою *config set*). Коли опція *requirepass* встановлюється в будь-яке значення (пароль), клієнт повинен виконати авторизацію *auth password* при з'єднанні із сервером. Як тільки клієнт пройде авторизацію він може виконати будь яку команду в будь якій базі. В тому числі *flushall,* яка видаляє всі ключі із всіх баз. Конфігураційний файл можна редагувати через CLI з допомогою редактора nano або vim:

sudo nano /etc/redis/redis.conf

У відкритому файлі можна буде замінити команду *flushall,* яка знаходиться в заголовку Security. Писати команду потрібно нижче зазначеного прикладу. Для збереження змін натискаємо ctrl + X та підтверджуємо зміни. Але зауваження: зміни

спрацюють тільки після перезапуску сервера. Тому, можемо скористатися командою, яка наведена нижче:

sudo /etc/init.d/redis-server restart

Контрольні запитання

1) Що таке база даних типу ключ-сховище?
2) Які юз-кейси використання бази даних ключ-сховище?
3) Наведіть приклади можливого використання Redis для вашої предметної області.