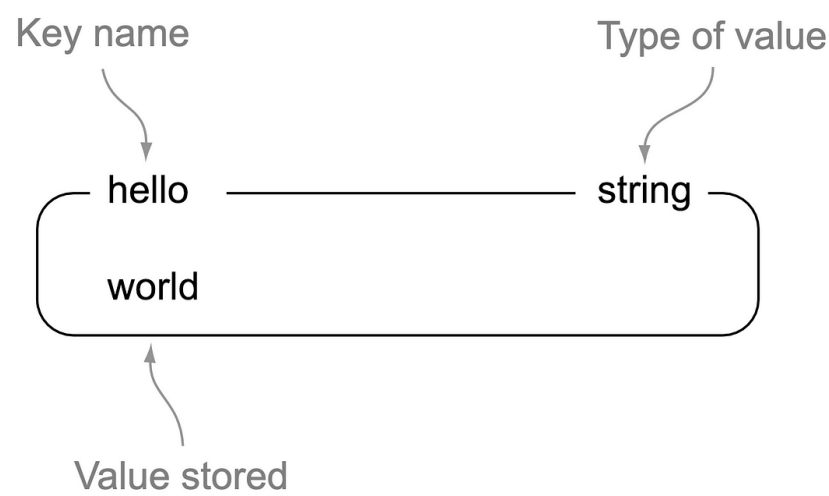


Проектирование NoSQL - основы Redis

Типы данных в Redis и время их жизни.

Redis может быть описан как система управления базами данных, работающая в оперативной памяти и организованная как хранилище данных типа "ключ-значение". Это означает, что в основе лежит большой словарь, где каждый уникальный ключ связан с определенным значением, и эти значения могут быть различных типов данных, не ограничиваясь простыми строками.



Redis не имеет такой же структуры, как SQL базы данных с таблицами и строками. В Redis данные хранятся в виде пар ключ-значение, и нет жестких ограничений на структуру данных. **Вот основные сущности в Redis:**

1. Ключи: Ключи представляют имена, по которым вы можете получать и сохранять данные в Redis. Ключи бывают строковыми и уникальными в пределах одной базы данных Redis. Примеры ключей: "user:123", "product:101:price".
2. Значения: Значения представляют данные, хранящиеся под ключами. Значения могут быть разного типа, такие как строки, числа, списки, хеш-таблицы и другие. Примеры значений: "Alice", 42, [1, 2, 3], {"name": "John", "age": 30}.

3. Базы данных: Redis может содержать несколько баз данных, и каждая база данных имеет свой уникальный набор ключей и значений. Вы можете выбирать базу данных при подключении к Redis. Базы данных идентифицируются целыми числами, обычно с 0 до 15. Например, "SELECT 0" выбирает базу данных с номером 0.
4. Пространства ключей (Key spaces): Пространства ключей - это организация ключей по категориям или типам данных. Вы можете использовать префиксы в именах ключей, чтобы группировать их внутри пространства ключей. Например, все ключи, связанные с пользователями, можно представить в пространстве ключей "user".

Примеры:

- "user:123:name" - ключ, представляющий имя пользователя с ID 123.
- "product:101:price" - ключ, представляющий цену продукта с ID 101.

Пространства ключей ("Key Spaces") - это соглашение о структуре и именовании ваших ключей, которое облегчает организацию данных и их поиск в Redis. Это не является встроенной функцией Redis, но это общепринятая практика для удобства управления данными в Redis.

Про нейминг ключей

1. Определите пространство ключей: Решите, какие категории данных вы будете хранить, и определите для каждой категории свое "пространство ключей". Например, если вы храните информацию о пользователях, вы можете создать пространство ключей "user" для всех связанных ключей.
2. Используйте понятные имена: Называйте ключи так, чтобы они были понятными для вас и других разработчиков. Например, "user:123" легче понять, чем "u123".
3. Избегайте слишком длинных ключей: Длинные ключи могут занимать больше места в памяти Redis. Старайтесь найти баланс между читаемостью и длиной ключей.
4. Добавьте контекст: Включайте в ключи контекст, который облегчит понимание их назначения. Например, "user:123:email" или "product:101:name".

5. Избегайте пробелов и специальных символов: Redis не ограничивает вас в выборе символов, но для удобства использования лучше избегать пробелов и специальных символов в ключах.
6. Будьте осторожны с регистром: Redis чувствителен к регистру символов, поэтому ключи "user:123" и "User:123" будут считаться разными.

Базовые операции в Redis предельно просты и в большинстве случаев сводятся к установке значения по ключу и извлечению значения по ключу. Это делает Redis легким в освоении и быстрым в использовании для различных задач, от кэширования до управления сессиями.

Особенность Redis заключается в его гибкости в отношении типов данных, которые могут быть использованы в качестве значений для ключей:

1. Строки: Это основной тип данных, который может хранить текст, числа или двоичные данные. Строки в Redis уникальны тем, что если они содержат числовые данные, можно использовать дополнительные команды для их инкрементации или декрементации, что делает Redis мощным инструментом для счетчиков и других числовых операций.
2. Списки: Списки представляют собой последовательности строк, упорядоченные в порядке вставки, что делает их подходящими для реализации очередей или стеков.
3. Множества: Множества – это наборы строк, где каждый элемент уникален. Важной особенностью множеств в Redis является возможность выполнять операции объединения, пересечения и разности, что полезно для работы с наборами данных.
4. Хеши: Хеши в Redis аналогичны словарям, где каждый ключ хранит внутри себя маппинг, связывающий строки (поля) со строковыми значениями, что делает их идеальными для представления объектов.
5. Упорядоченные множества: Это множества с дополнительной характеристикой - каждому элементу присваивается значение сортировки, что позволяет упорядочивать элементы внутри множества.

Keys	Values
page:index.html	→ <html><head>[...] ← String
login_count	→ 7464
users_logged_in_today	→ { 1, 2, 3, 4, 5 } ← Set
latest_post_ids	→ [201, 204, 209,...] ← List
user:123:session	→ time => 10927353 username => joe ← Hash
users_and_scores	→ joe ~ 1.3483 bert ~ 93.4 fred ~ 283.22 chris ~ 23774.17 ← Sorted (scored) Set

Под капотом Redis использует различные оптимизации для эффективного хранения этих типов данных, например, компактное хранение маленьких множеств или списков, что позволяет экономить память и увеличивать производительность.

В Redis, время жизни данных (TTL, Time To Live) позволяет задать ограниченный жизненный цикл для ключей. TTL определяет временной интервал, по истечении которого ключ и его значение будут автоматически удалены из базы данных. Это нужно для кэширования и других операций, где данные должны оставаться актуальными только в течение определенного времени.

Когда вы устанавливаете TTL для ключа, вы задаете время в секундах. Например, если вы хотите, чтобы данные исчезли через час, вы зададите TTL равным 3600 секундам. Используя команду `EXPIRE key seconds`, вы задаете Redis задачу автоматически удалить ключ после указанного количества секунд.

Однако, важно понимать, как различные команды влияют на TTL ключа:

1. При полной перезаписи значения: Если вы используете команду, которая полностью перезаписывает значение ключа (например, `SET`), TTL ключа будет сброшен. Это означает, что если ключ имел установленный TTL, после выполнения команды `SET` время жизни ключа будет неопределенным, и ключ не будет удален автоматически.

2. При модификации значения: В случае если команда модифицирует значение, не перезаписывая его полностью (например, `INCR` для увеличения числового значения или `LPUSH` для добавления элемента в список), TTL ключа остается неизменным. Таким образом, ключ все еще будет удален по прошествии первоначально установленного TTL.

Redis также предоставляет команды для управления TTL:

- `TTL key`: Возвращает оставшееся время жизни ключа в секундах.
- `PTTL key`: Возвращает оставшееся время жизни ключа в миллисекундах.
- `PERSIST key`: Удаляет TTL с ключа, делая его постоянным.

Такая возможность автоматического контроля времени жизни данных делает Redis ценным для систем, которые должны активно управлять памятью и гарантировать, что устаревшие данные не занимают ресурсы памяти. Это помогает предотвратить необходимость ручного управления циклом жизни данных, что может быть сложно в системах с большим объемом данных или высокими требованиями к производительности.