

Проектирование NoSQL - основы Redis

Рассмотрим различные паттерны проектирования Redis.

Кэширование данных с помощью Redis — это процесс хранения результатов запросов, вычислений или любых других операций, которые требуют значительных ресурсов для повторного выполнения. Целью кэширования является сокращение времени доступа к данным и уменьшение нагрузки на первичные хранилища данных. Ниже пример реализации.

Шаг 1: Определение данных для кэширования

Выберите данные, которые часто запрашиваются и/или требуют значительных вычислительных ресурсов для генерации. Это могут быть результаты сложных SQL-запросов, вычисленные агрегации, HTML страницы и т.д.

Шаг 2: Генерация уникального ключа кэша

Для каждого уникального запроса или набора данных создайте уникальный ключ, который будет использоваться для сохранения и извлечения данных из кэша. Ключ может быть составным, включающим идентификаторы, параметры запроса или хеш.

Шаг 3: Проверка наличия данных в кэше

Перед выполнением операции (например, запроса к базе данных) проверьте, есть ли уже кэшированный результат в Redis.

```
GET cache:<unique_key>
```

Шаг 4.1: Использование кэшированных данных (если доступны)

Если данные уже кэшированы, извлеките их из Redis и используйте, тем самым избегая затратного процесса их повторного создания.

Шаг 4.2: Вычисление и кэширование данных (если их нет в кэше)

Если кэшированные данные отсутствуют, выполните необходимую операцию для получения данных, затем сохраните результат в Redis с уникальным ключом.

```
SET cache:<unique_key> <data>
```

Шаг 5: Установка TTL для кэшированных данных

Установите время жизни кэшированных данных, чтобы они автоматически удалялись после истечения определенного времени. Это предотвратит устаревание данных и их хранение, когда они больше не нужны.

```
EXPIRE cache:<unique_key> <ttd>
```

Шаг 6: Инвалидация кэша при изменении данных

Когда основные данные изменяются, соответствующий кэш должен быть инвалидирован (удален), чтобы гарантировать, что пользователи получают актуальные данные.

```
DEL cache:<unique_key>
```

Паттерн кэширования данных в Redis значительно повышает производительность приложений, снижая время отклика и уменьшая нагрузку на базы данных и другие системы. В высоконагруженных системах, где миллисекунды имеют значение, и при работе с большими объемами данных стандартный паттерн - перед базой данных ставить кэш Redis.

Практика

На ресурсе <https://try.redis.io/> вы можете ввести команды ниже, чтобы "вживую потрогать" Redis. Введите команды по очереди ниже.

Шаг 1: Определение данных для кэширования

Мы будем кэшировать результат сложения чисел 5 и 3.

Шаг 2: Генерация уникального ключа кэша

Уникальный ключ `sum_result`, который будет использоваться для сохранения и извлечения данных из кэша.

Шаг 3: Проверка наличия данных в кэше

Давайте сначала проверим, есть ли уже кэшированный результат в Redis. Для этого выполним команду GET для ключа `sum_result`. Если результат уже есть, мы будем его использовать.

```
GET sum_result
```

Шаг 4.1: Использование кэшированных данных

Команда GET не вернула результат, это значит, что данных нет. Мы не можем использовать этот результат.

Шаг 4.2: Вычисление и кэширование данных

Если результат не был найден в кэше, выполним операцию сложения двух чисел (5 и 3) и сохраним результат в Redis с ключом `sum_result`. Для этого выполним команду SET.

```
SET sum_result 8
```

Теперь результат сложения (8) сохранен в кэше.

Шаг 5: Установка TTL для кэшированных данных

Устанавливать TTL не будем в данном примере, чтобы данные оставались в кэше бесконечно. В реальных приложениях, вы бы установили TTL, чтобы данные автоматически удалялись после определенного времени.

Мы просто в данном случае инвалидируем значение.

Шаг 6: Инвалидация кэша при изменении данных

Для инвалидации кэша (удаления данных из кэша), выполните команду DEL для ключа `sum_result`, если базовые данные изменились (представим что они изменились).

```
DEL sum_result
```

Теперь "кэш" для этого результата удален. И команда GET `sum_result` снова ничего не вернёт.