

Проектирование NoSQL - основы Redis

Рассмотрим различные паттерны проектирования Redis.

Кэширование сессий в Redis — это паттерн, который помогает быстро и эффективно управлять данными сессии пользователей. Один из примеров реализации этого паттерна ниже.

Шаг 1: Создание сессии при аутентификации

Когда пользователь успешно проходит процесс аутентификации, приложение создает уникальный идентификатор сессии (например, с помощью UUID).

Шаг 2: Сохранение данных сессии в Redis

Используя Redis, приложение сохраняет данные сессии, связывая их с уникальным идентификатором. Ключом в Redis будет идентификатор сессии, а значением — сериализованные данные сессии.

```
SET session:<session_id> <serialized_session_data>
```

Шаг 3: Установка TTL для сессии

Для каждого ключа сессии можно установить TTL (Time To Live), чтобы сессия автоматически истекла после определенного времени неактивности.

```
EXPIRE session:<session_id> 3600
```

Эта команда установит сессию на 1 час (3600 секунд).

Шаг 4: Доступ к данным сессии

При каждом запросе пользователя приложение извлекает данные сессии из Redis, используя идентификатор сессии, который обычно хранится в cookie или токене аутентификации.

```
GET session:<session_id>
```

Шаг 5: Обновление TTL при активности

Каждый раз, когда пользователь взаимодействует с приложением, TTL сессии обновляется, чтобы предотвратить ее истечение во время активной сессии.

```
EXPIRE session:<session_id> 3600
```

Шаг 6: Удаление сессии при выходе пользователя

При выходе пользователя из системы приложение должно удалять данные сессии из Redis.

```
DEL session:<session_id>
```

Использование Redis для кэширования сессий улучшает производительность, так как доступ к данным осуществляется очень быстро благодаря хранению в оперативной памяти. А автоматическое управление истечением срока действия сессий повышает безопасность и уменьшает нагрузку на сервер.

Практика

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

1. Создание сессии при аутентификации:

```
SET session:12345 "serialized_session_data_here"
```

В данном примере, "serialized_session_data_here" - это сериализованные данные сессии, а 12345 - уникальный идентификатор сессии.

2. Установка TTL для сессии (например, 1 час = 3600 секунд):

```
EXPIRE session:12345 3600
```

3. Доступ к данным сессии (по уникальному идентификатору сессии):

```
GET session:12345
```

4. Обновление TTL при активности:

```
EXPIRE session:12345 3600
```

5. Удаление сессии при выходе пользователя (по уникальному идентификатору сессии):

```
DEL session:12345
```