

Проектирование NoSQL - основы Redis

Рассмотрим различные паттерны проектирования Redis.

Также среди паттернов использования Redis выделяются:

- 1. Очередь событий: Управление последовательностью действий или задач в фоновом режиме.
- 2. Блокировка с Redlock: Распределенная блокировка для синхронизации доступа к общим ресурсам.
- 3. Pub/Sub: Система публикации и подписки для обмена сообщениями и уведомлений в реальном времени.
- 4. Распределённые события: Синхронизация событий и состояний между множеством компонентов в распределенной системе.

Давайте рассмотрим простой пример на основе системы управления заказами в интернет-магазине.

Очередь событий

Сценарий: Когда пользователь оформляет заказ, система должна выполнить ряд последовательных действий, таких как обновление складского запаса, обработка платежа и уведомление службы доставки.

Реализация:

- При оформлении заказа создается событие заказа и помещается в очередь Redis с использованием команд `LPUSH` (добавление в начало списка) или `RPUSH` (добавление в конец списка).
- Фоновый процесс, например, `worker`, постоянно прослушивает очередь с использованием команды `BRPOP` (блокирующее извлечение элемента из конца списка), обрабатывает заказы и выполняет необходимые действия.

Блокировка с Redlock

Сценарий: Предотвратить одновременное выполнение операции обновления складских запасов несколькими процессами.

Реализация:

- Когда worker начинает обработку заказа, он пытается получить блокировку с помощью алгоритма Redlock, который представляет собой распределенную блокировку для предотвращения одновременного доступа к ресурсу.
- Если блокировка получена, worker обновляет складские запасы. Если блокировка не получена, worker откладывает обработку до тех пор, пока блокировка не станет доступной.

Pub/Sub

Сценарий: Уведомление всех заинтересованных сторон о статусе заказа, таких как служба доставки, бухгалтерия и клиент.

Реализация:

- Система публикует сообщение о статусе заказа с использованием команды `PUBLISH` в канале Redis, который представляет собой статус заказа.
- Службы доставки, бухгалтерии и интерфейс клиента подписываются на этот канал с помощью команды `SUBSCRIBE` и получают уведомления в реальном времени о любых изменениях статуса заказа.

Распределённые события

Сценарий: Синхронизация статуса заказа между несколькими микросервисами или распределенными системами, которые могут находиться в разных географических регионах.

Реализация с Redis:

- Каждый микросервис использует Redis для публикации событий, связанных с заказом, например, завершение обработки платежа или изменение статуса доставки.
- Другие микросервисы, отвечающие за разные аспекты обработки заказа, подписываются на эти события и реагируют на них соответствующим образом, например, обновляют пользовательский интерфейс или выполняют логистическое планирование.