Блокировки Что это? и Как работает?

Блокировки и состояние гонки являются центральными концепциями в контексте многопользовательских систем и совместного использования ресурсов. Понимание этих понятий особенно важно при работе с общими данными, с которыми могут взаимодействовать два или более пользователя.

Зачем это нужно

Возьмем для примера платформу Metabot. Её пользователи обычно действуют автономно и никак не влияют на действия друг друга. Тем не менее, случаются ситуации, когда одновременное взаимодействие нескольких пользователей с одними и теми же данными становится неизбежным.

Чтобы рассмотреть этот процесс подробнее возьмем простой пример - сервис записи на прием к специалисту. В этом сервисе, два пользователя могут одновременно попытаться зарегистрироваться на один и тот же временной слот. В базу данных поступят две заявки на одно и то же время, что приведет к проблемам, особенно если другие временные слоты уже заняты.

Здесь на помощь приходят блокировки.

Как это работает

Важно понимать, что блокировка сама по себе не блокирует доступ к ресурсу. Вместо этого, она служит своего рода условием или соглашением, общей переменной, которую могут проверять различные пользователи или процессы. Таким образом, когда пользователь (или, в нашем контексте, бот) хочет получить доступ к ресурсу, он проверяет состояние блокировки (равно ли значение переменной **true**).

Если переменная указывает, что ресурс свободен, пользователь "захватывает" блокировку, изменяя ее значение на **false** для всех остальных пользователей. Это действие символически сообщает системе, что ресурс теперь занят, и другие пользователи или процессы не могут получить к нему доступ до тех пор, пока текущий пользователь не "освободит" блокировку, вернув ей значение **true**.

Ниже можно посмотреть как данный процесс выглядит в форме кода:

```
let lockName = 'doctorsRecord' // Имя для вашей блокировки | string
let lockPrefix = '' // Добавляется к lockName, помогает в категоризации блокировок | string
let ttlSec = 30 // Через какое количество секунд блокировка снимется самостоятельно | int
let maxWaitSec = 5 // Сколько секунд бот будет пытаться получить блокировку (Максимум 300) | int
let isOpenRecord = bot.waitForBotLock(lockName, lockPrefix, ttlSec, maxWaitSec) // Метод захвата блокировки

if (isOpenRecord) {
  // Выполянется, если получилось захватить блокировку и isOpenRecord = true
} else{
  // Выполянется, если НЕ получилось захватить блокировку и isOpenRecord = false
}
```

В скрипте задаются значения переменным, необходимым для захвата блокировки. Затем вызывается метод, возвращающий значение блокировки. И в завершении происходит проверка: если блокировка свободна (её значение равно **true**), то выполняются одни действия, если блокировка занята (её значение равно **false**), то выполняются другие действия.

> Важно понимать, что операция по захвату и проверке осуществляется одновременно. Если функция вернула **true**, значит у текущего пользователя получилось захватить блокировку, а у других пользователей - нет.

Время жизни блокировки

Бывают ситуации, когда пользователь забывает или просто уходит, оставляя блокировку в "захваченном" состоянии, что закрывает доступ других пользователей к ресурсу. Для решения этой проблемы используется "время жизни" блокировки. Это означает, что блокировка автоматически "освобождается" после определенного периода времени (например, 30 секунд), даже если пользователь, изначально захвативший блокировку, не освободил её самостоятельно. Это обеспечивает более справедливый и эффективный доступ к ограниченным ресурсам.

Что это дает

Важно понимать, что само по себе использование блокировок не является гарантией исключения состояний гонки. Такие ситуации могут по-прежнему возникать, если не соблюдается порядок и последовательность операций. Блокировки просто предоставляют новый уровень контроля над использованием ресурсов, и являются семафором для сигнализации другим пользователям или процессам о статусе ресурса.

Блокировки, при правильном использовании, позволяют системам эффективно обрабатывать совместную работу с ресурсами и поддерживать согласованность данных, предотвращая конфликты и ошибки. Они являются весомым инструментом в руках разработчиков программного обеспечения для управления совместным доступом к ресурсам в многопользовательских и многопоточных системах.

Версия #5
Ирина Петрова создал 24 July 2023 08:24:13
Artem Garashko обновил 11 November 2025 07:22:33