

Популярная реализация транзакции - **двухфазный коммит**.

Протокол Two-Phase Commit (2PC) - это метод управления распределенными транзакциями, используемый для обеспечения согласованности данных в нескольких системах. Он помогает координировать транзакцию между различными участниками, гарантируя, что либо все участники зафиксируют транзакцию, либо никто из них.

Проще говоря, протокол 2PC состоит из двух основных этапов:

- 1. Подготовительная фаза: на этом этапе координатор спрашивает всех участников, готовы ли они совершить транзакцию. Каждый участник подготавливает свою часть транзакции, блокирует необходимые ресурсы и сообщает, готов ли он к фиксации или нет.
- 2. Фаза фиксации: если все участники сообщают, что они готовы к фиксации, координатор дает им команду зафиксировать транзакцию. После этого все участники делают свои изменения постоянными и освобождают все заблокированные ресурсы. Если кто-то из участников не может зафиксировать транзакцию, координатор дает команду всем участникам прервать транзакцию, и они откатывают все изменения, сделанные на этапе подготовки.

Протокол Two-Phase Commit гарантирует, что транзакция будет либо полностью зафиксирована всеми участниками, либо полностью откатена, что предотвращает несогласованность данных и частичные обновления.

Ниже высокоуровневая иллюстрация того, как вы можете разработать API для поддержки протокола 2PC. Предположим, у нас есть служба координатора и два менеджера ресурсов (RM1 и RM2), которые управляют различными частями транзакции. Мы собираемся перевести средства между двумя счетами, управляемыми этими менеджерами ресурсов.

- **Подготовительная фаза.** Клиент посылает POST-запрос координатору, чтобы инициировать транзакцию:

POST /coordinator/transactions HTTP/1.1

Content-Type: application/json

```
{  
  
  "действие": "transfer",  
  
  "fromAccount": "123",  
  
  "toAccount": "456",  
}
```

```
"сумма": 100
}
```

- Координатор назначает ID транзакции и отправляет запросы PREPARE в RM1 и RM2:

POST /rm1/prepare HTTP/1.1

Content-Type: application/json

```
{
  "transactionId": "abc123",
  "account": "123",
  "amount": -100
}
```

POST /rm2/prepare HTTP/1.1

Content-Type: application/json

```
{
  "transactionId": "abc123",
  "account": "456",
  "amount": 100
}
```

- Оба менеджера ресурсов готовят транзакцию и отвечают своей готовностью к фиксации или прерыванию:

HTTP/1.1 200 OK

Content-Type: application/json

```
{  
  "transactionId": "abc123",  
  "vote": "COMMIT"  
}
```

- **Фаза фиксации.** Если все участники согласны на фиксацию, координатор посылает сообщение COMMIT на RM1 и RM2:

POST /rm1/commit HTTP/1.1

Content-Type: application/json

```
{  
  "transactionId": "abc123"  
}
```

POST /rm2/commit HTTP/1.1

Content-Type: application/json

```
{  
  "transactionId": "abc123"  
}
```

- Оба менеджера ресурсов фиксируют свою часть транзакции и отвечают статусом успеха:

HTTP/1.1 200 OK

Content-Type: application/json

```
{  
  "transactionId": "abc123",
```

```
"status": "COMMITTED"
```

```
}
```

- Если любой участник решает прервать процесс, координатор посылает сообщение ABORT всем участникам:

```
POST /rm1/abort HTTP/1.1
```

```
Content-Type: application/json
```

```
{
```

```
  "transactionId": "abc123"
```

```
}
```

```
POST /rm2/abort HTTP/1.1
```

```
Content-Type: application/json
```

```
{
```

```
  "transactionId": "abc123"
```

```
}
```

Обратите внимание, что данный пример является иллюстрацией высокого уровня и может не охватывать все аспекты готовой к производству реализации протокола 2PC через HTTP.

Используя транзакции, мы можем обеспечить атомарное выполнение нескольких операций, что означает, что все они будут либо успешно завершены, либо откачены, если какая-либо из них завершится неудачей. Это помогает поддерживать согласованность данных и предотвращать их потерю или повреждение.

В целом, обработка параллелизма и согласованности очень важна для создания надежных и прочных API, которые могут обрабатывать одновременный доступ и изменение данных несколькими пользователями или клиентами. Применяя такие методы, как оптимистическая блокировка, пессимистическая блокировка, транзакции или условные запросы, вы можете гарантировать, что ваш API поддерживает согласованность и обрабатывает одновременный доступ предсказуемым и эффективным образом.