Транзакционная работа с топиками. Архитектура и сравнение решений в Apache Kafka и YDB

Алексей Николаевский
Яндекс, руководитель сервиса

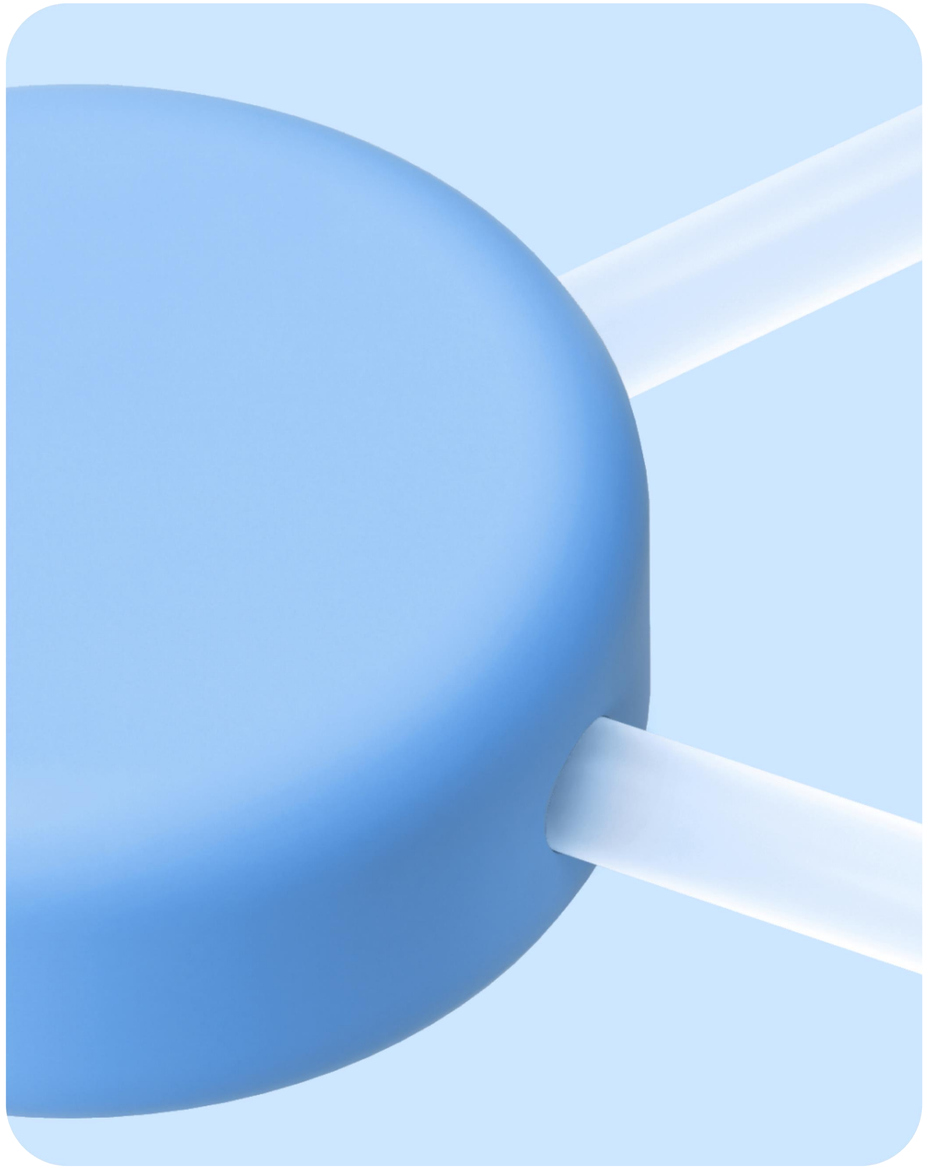
HL 2024 HighLoad++

Содержание

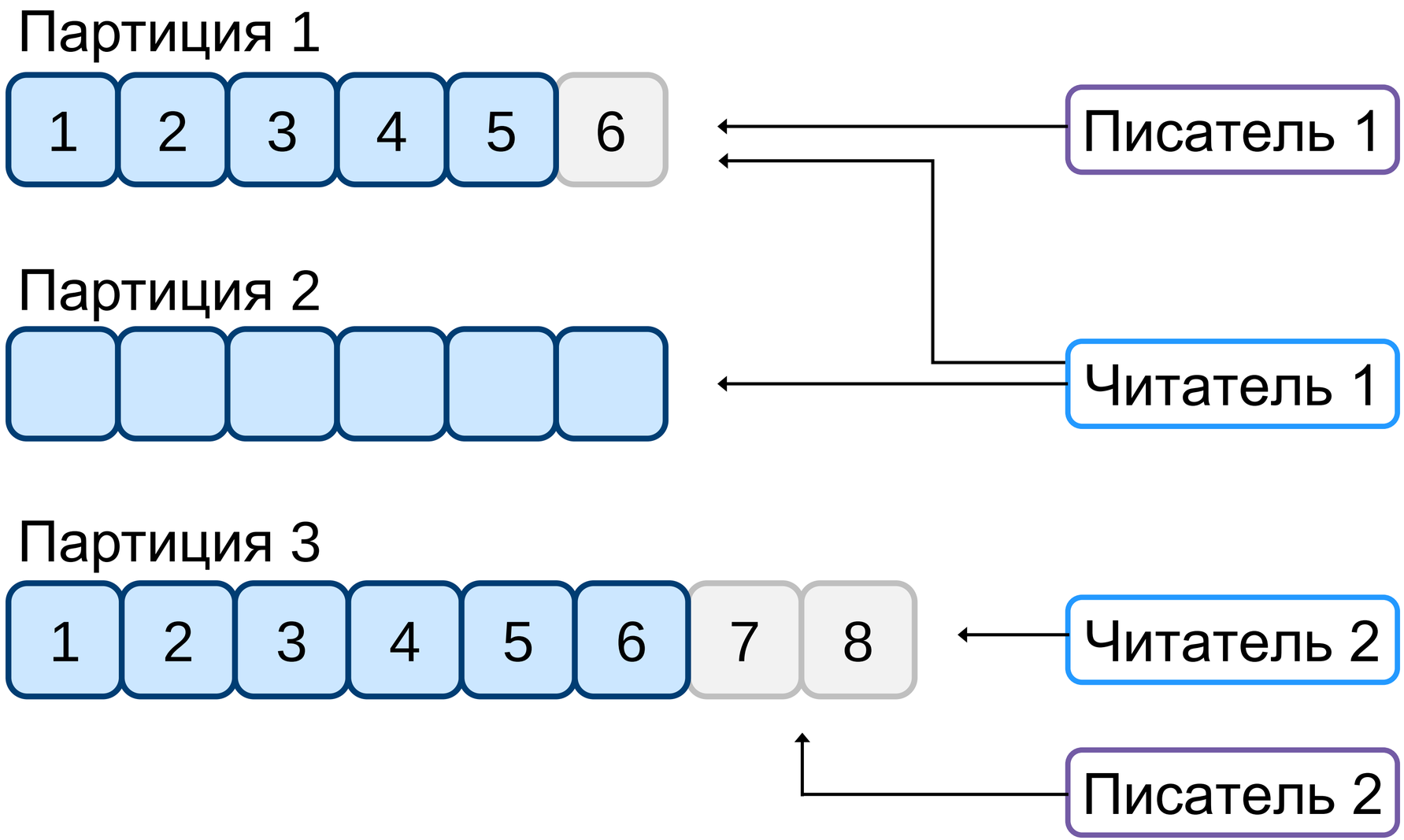
HL 2024 HighLoad++

1

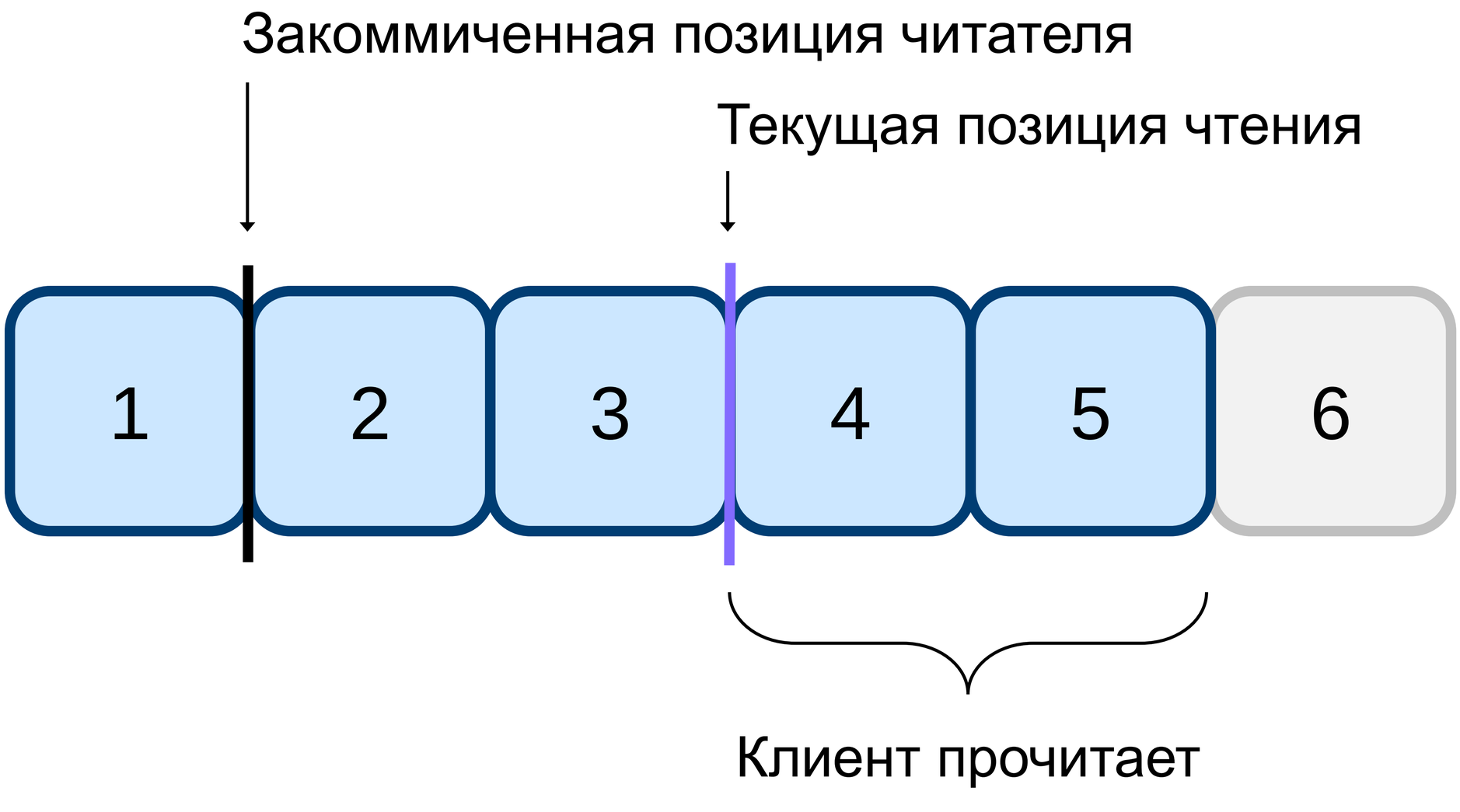
Модель данных топиков



Топик


Партиция 1
1
2
3
4
5
6
Партиция 2
Партиция 3
1
2
3
4
5
6
7
8
Писатель 1
Читатель 1
Читатель 2
Писатель 2

Топик. Позиции читателей



Топик. Позиции читателей

Транзакции

В рамках транзакции можно:

- Записать
- Прочитать
- Сдвинуть закоммиченую позицию чтения

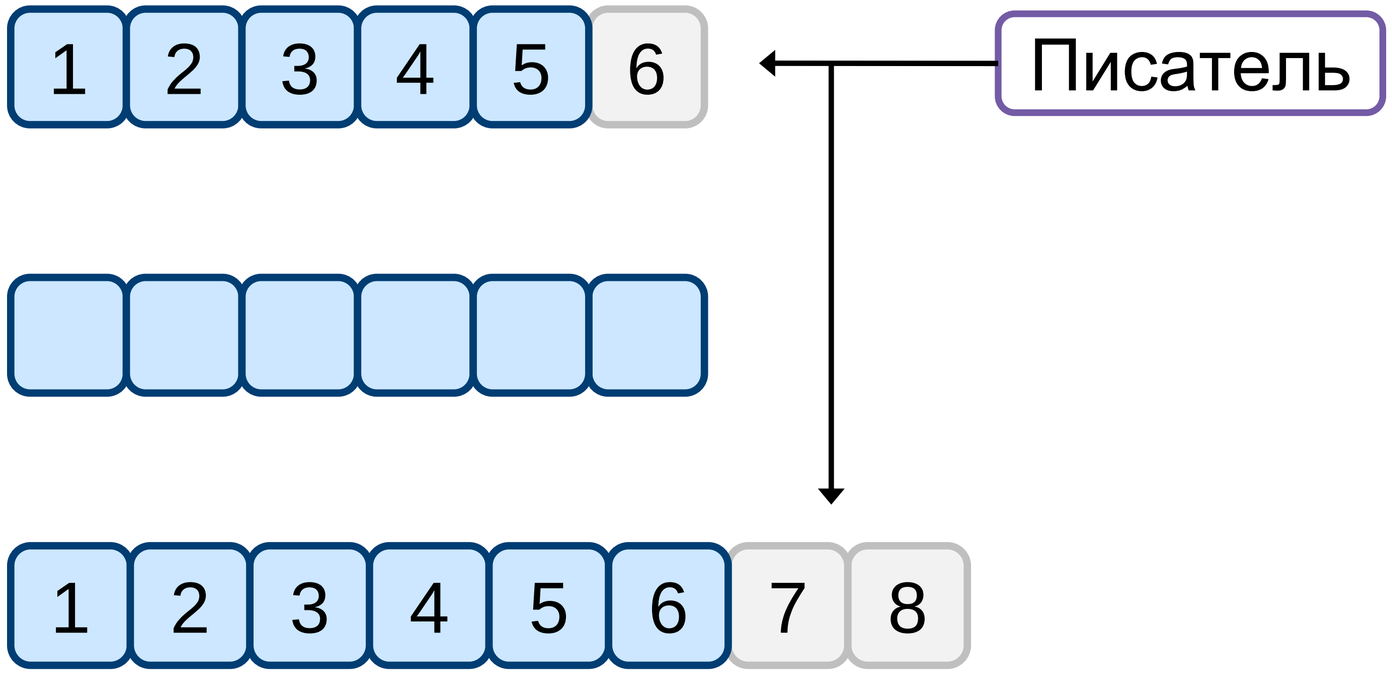
Atomicity
Consistency
Isolation
Durability

Как проверить выполнение ACID?
Только при чтении!

Транзакции

- Atomicity — все действия применяются атомарно
- Consistency — система консистентна (?)
- Isolation – нельзя показывать изменения еще незакоммиченной транзакции
- Durability – система не должна потерять данные

Транзакционная запись



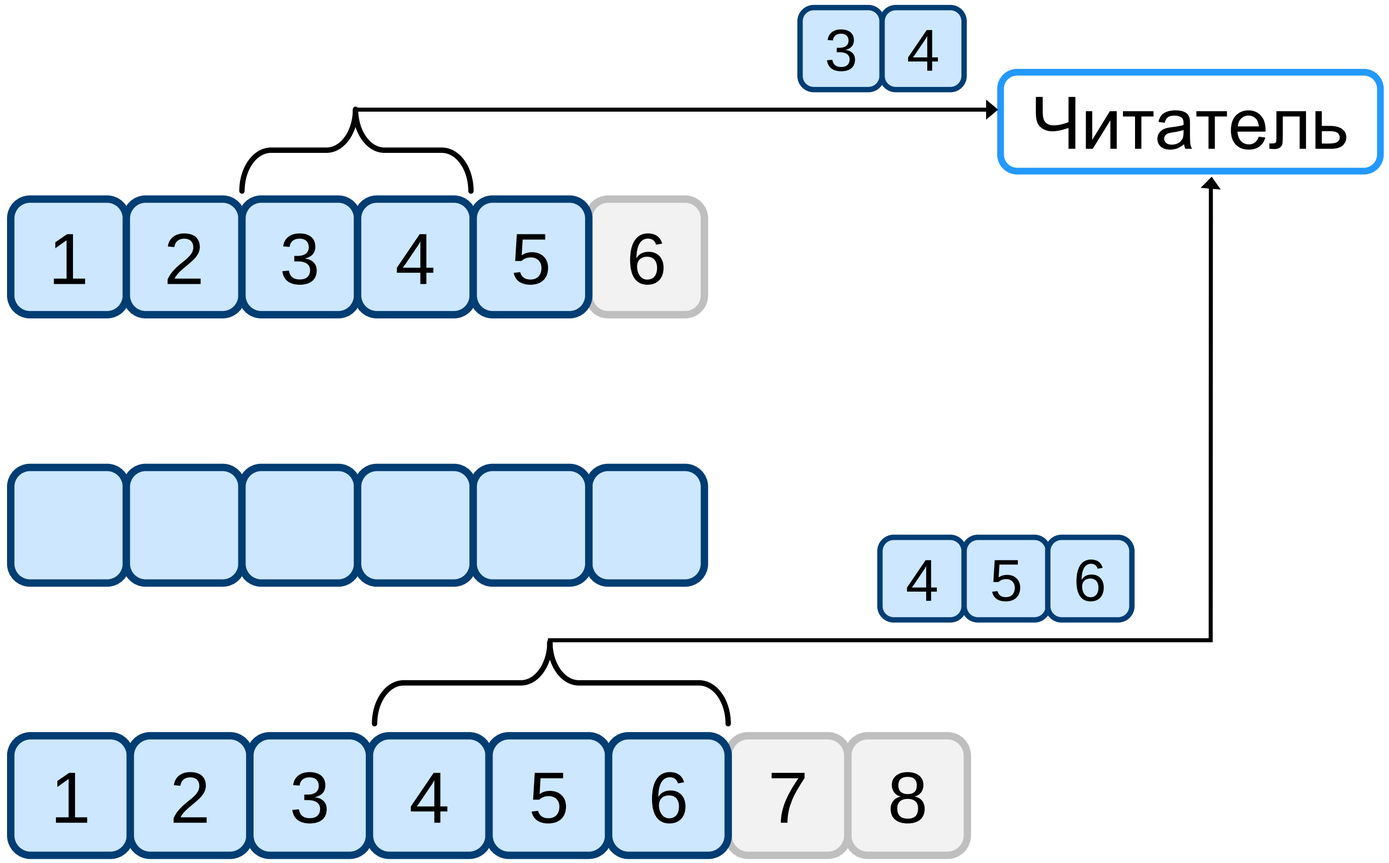
A записывается все или ничего

C неприменимо

I пока транзакцию не закоммитили, данные недоступны читателю

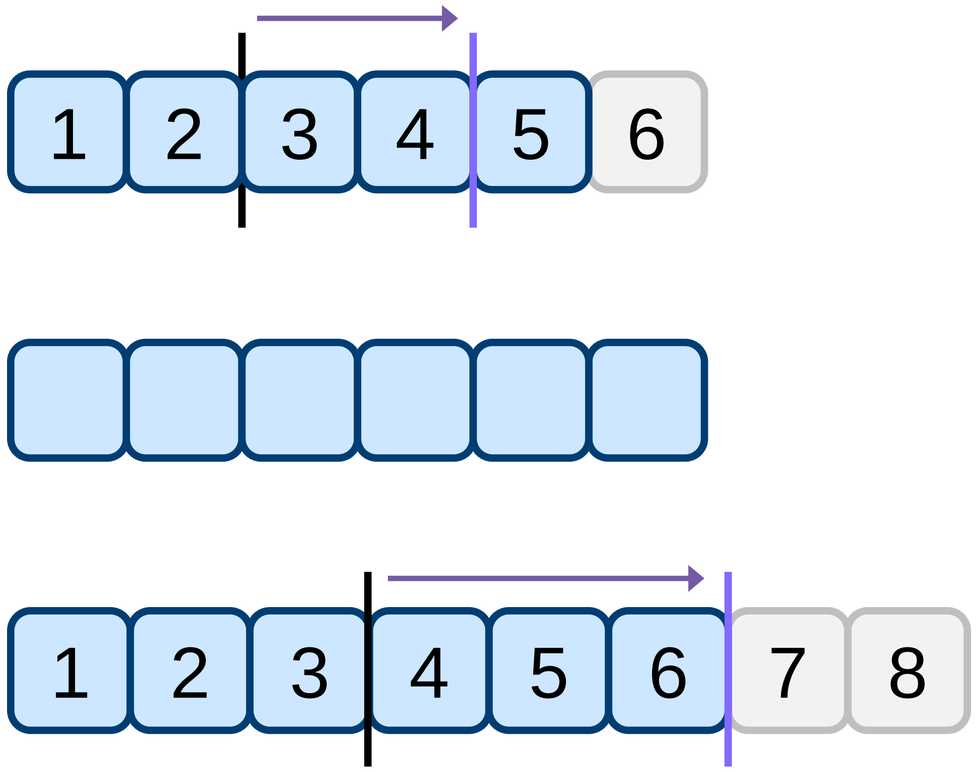
D данные не потеряются

Транзакционное чтение



- A неприменимо
- C неприменимо
- I неприменимо
- D неприменимо

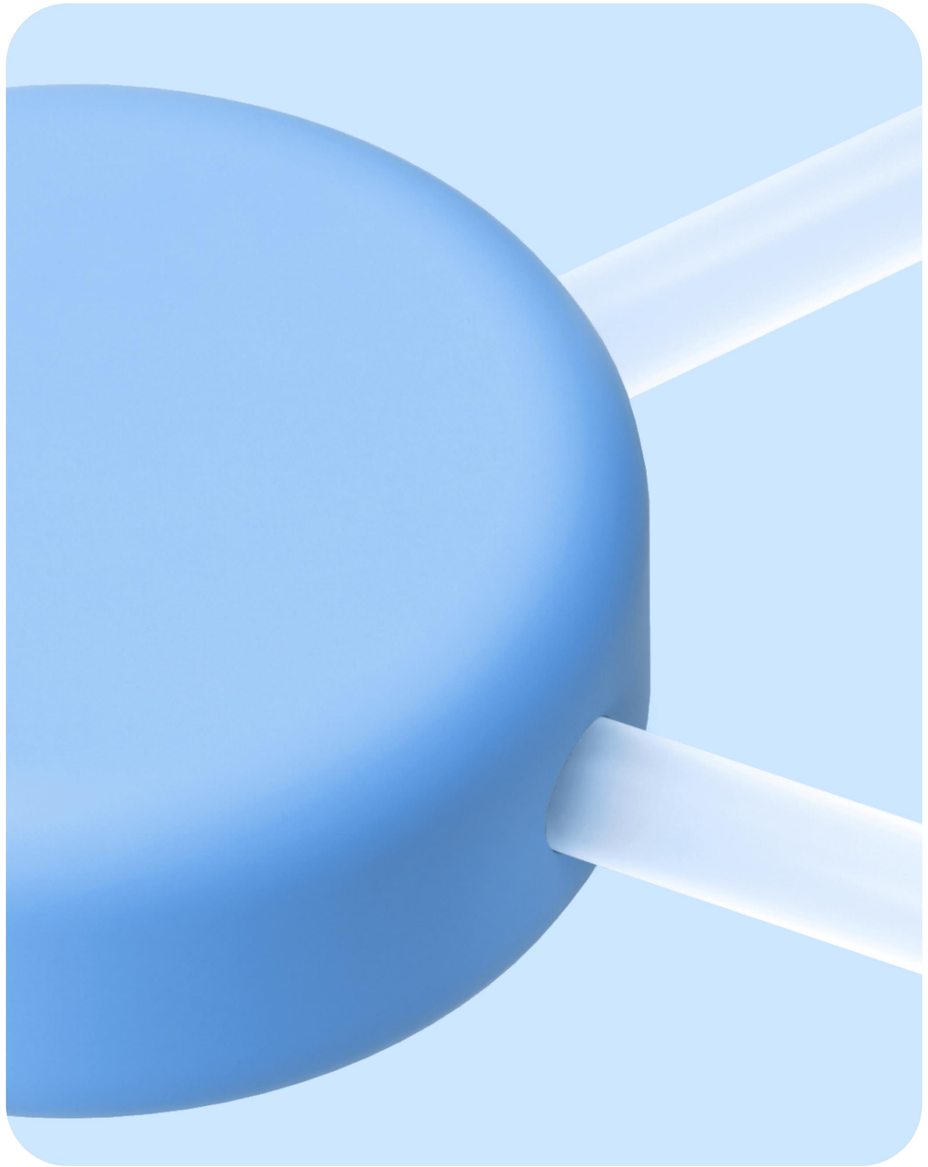
Транзакционный сдвиг позиций читателей



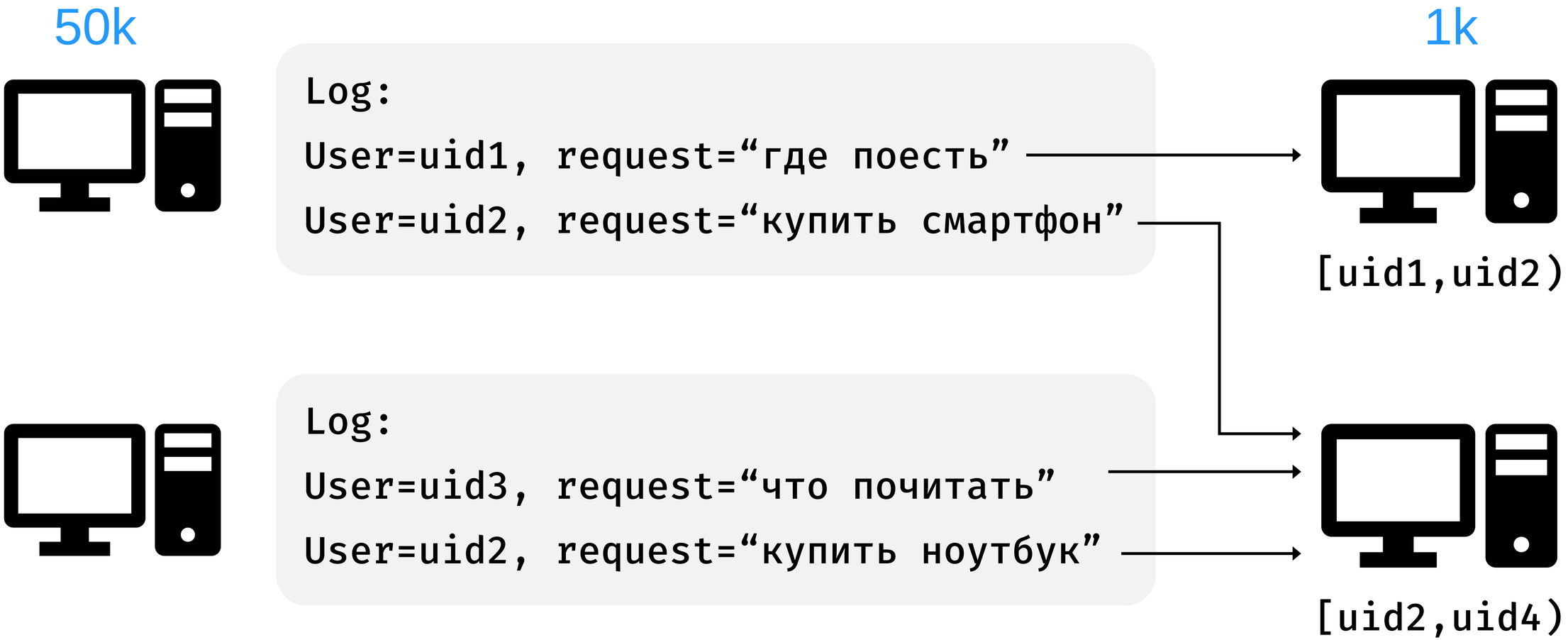
- **A** все сдвиги и все записи или ничего
- **C** ?
- **I** пока транзакцию не закоммитили, новые позиции недоступны
- **D** позиция не потеряется

2

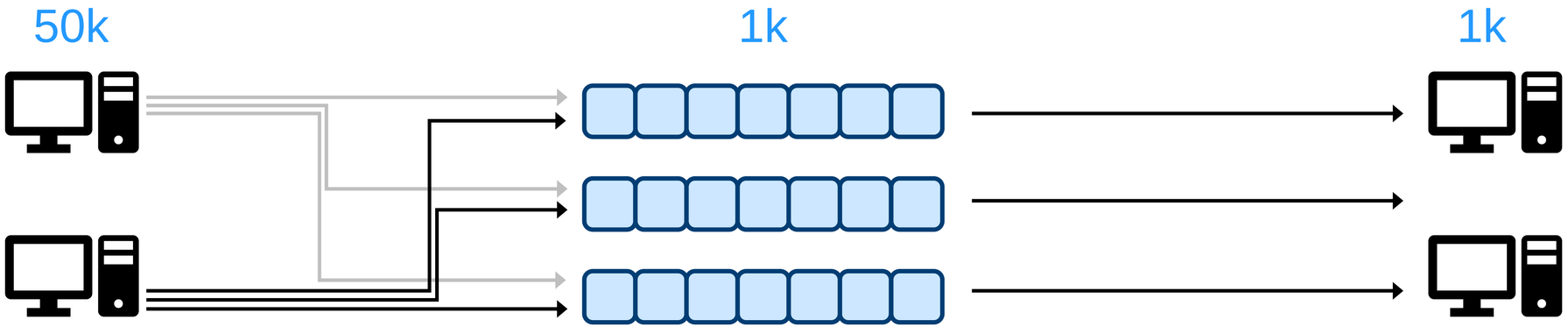
Модельная задача



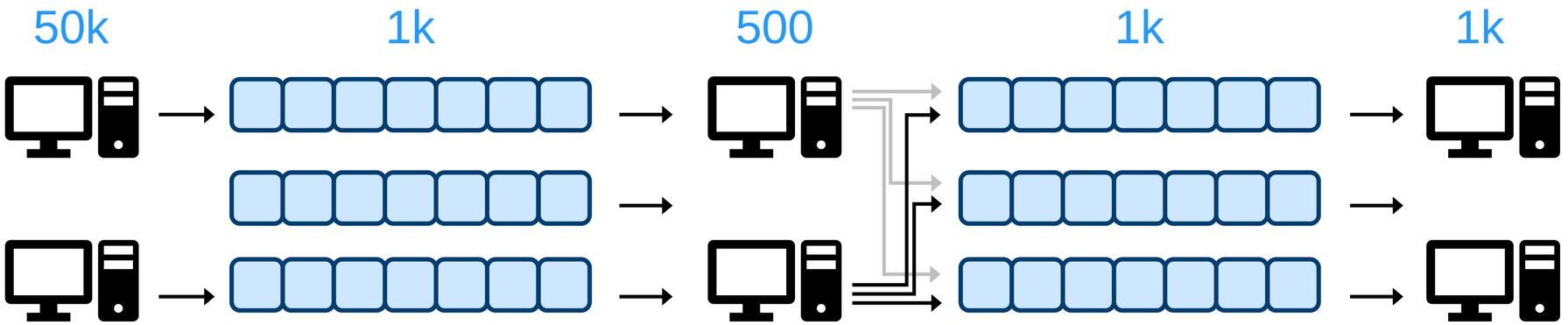
Задача решардирования. Поисковые логи



Задача решардирования. Зачем?

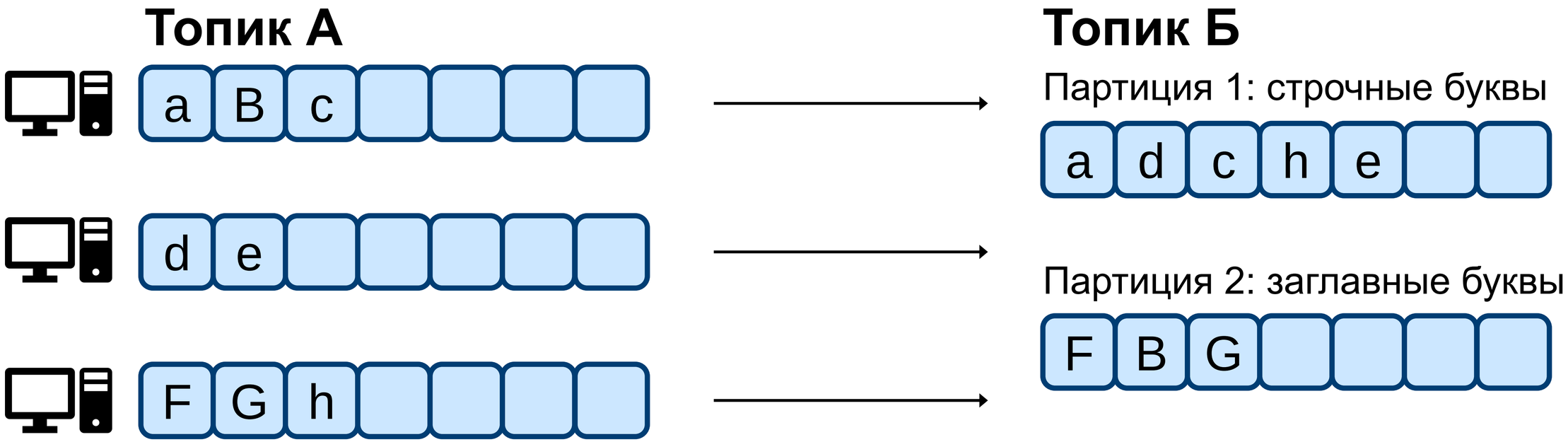


- 50 001 000 коннектов
- маленькие записи



- 552 000 коннектов
- большие записи

Задача решардирования. Формулировка



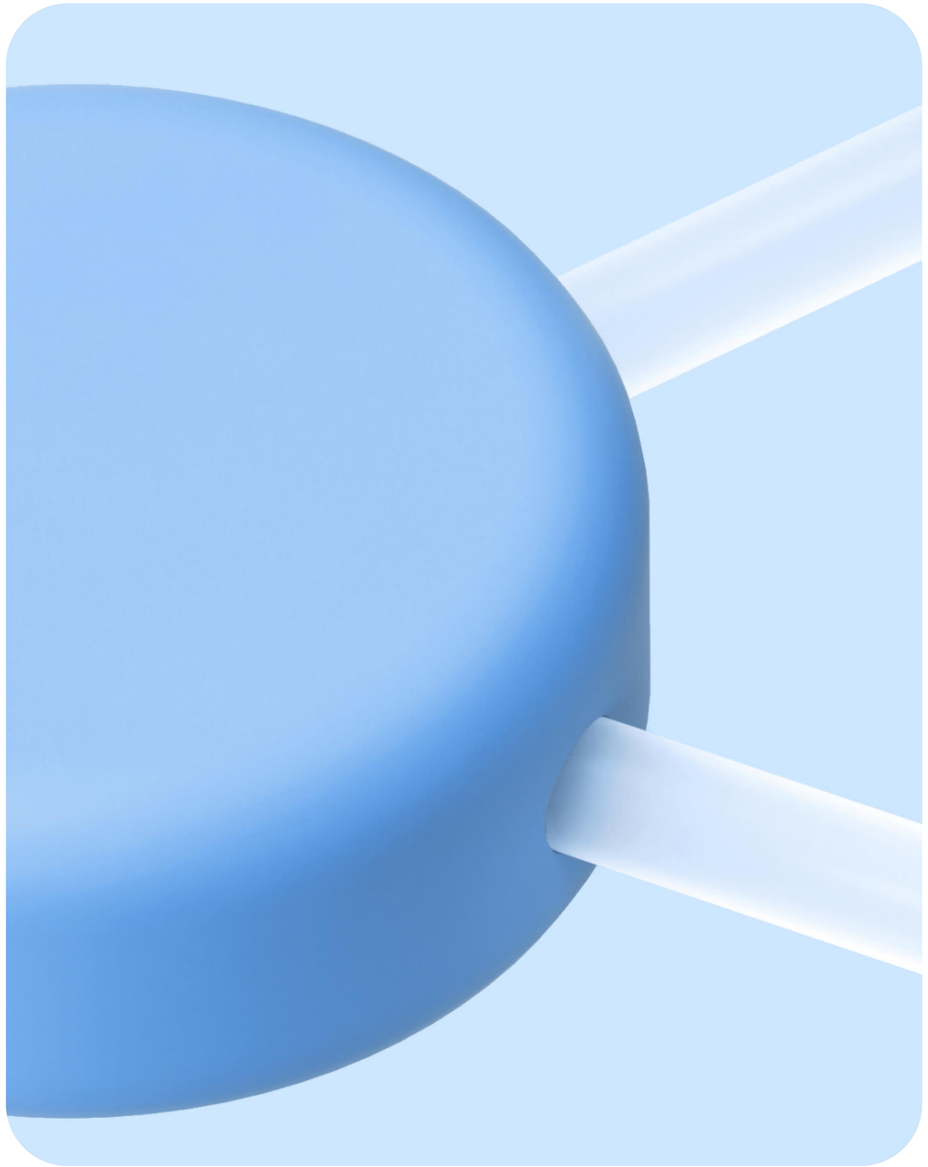
- Локальность данных
- Порядок из исходной партиции
- **Exactly-once**

Задача решардирования. С транзакциями

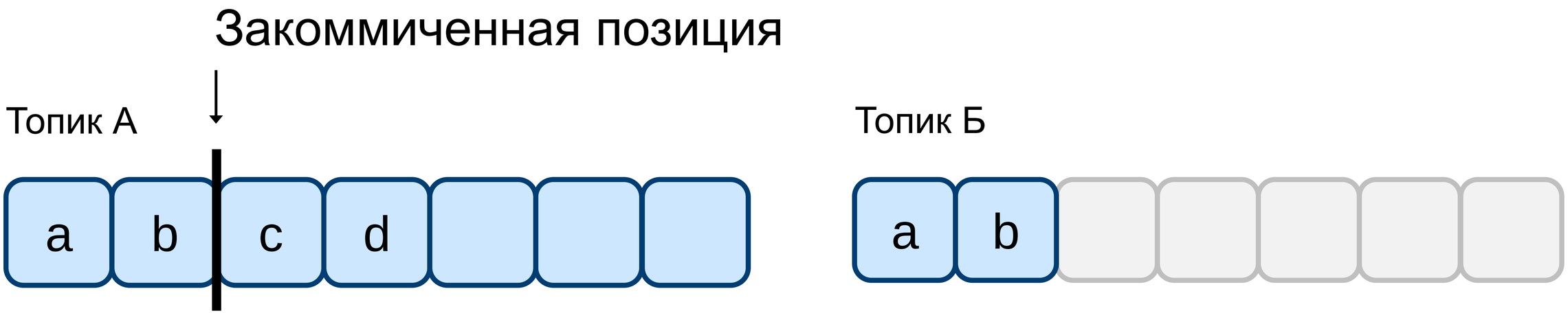
```
consumer=CreateConsumer(topicA)
producer=CreateProducer(topicB)
while(true) {
    tx = BeginTx()
    data = consumer.Read(tx)
    processed_data = reshard(data) //partition+data
    for (p: processed_data) {
        producer.Write(p.partition, p.data)
    }
    result = CommitTx(tx)
    if (!result.IsSuccess()) exit
}
```

3

Почему транзакции нужны в решении модельной задачи

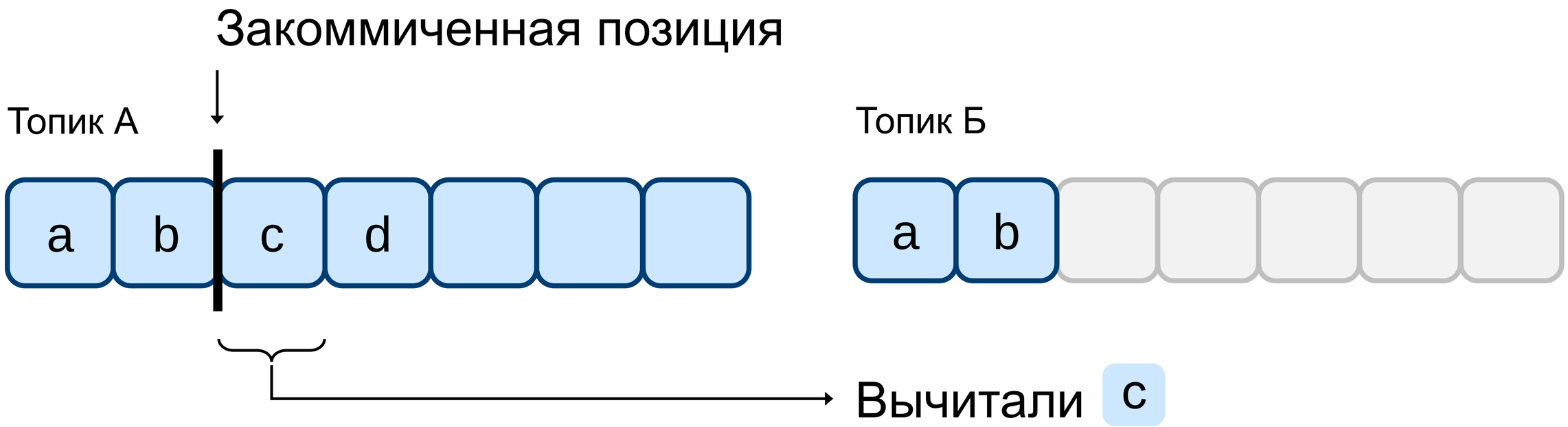


Задача решардирования. Exactly-once



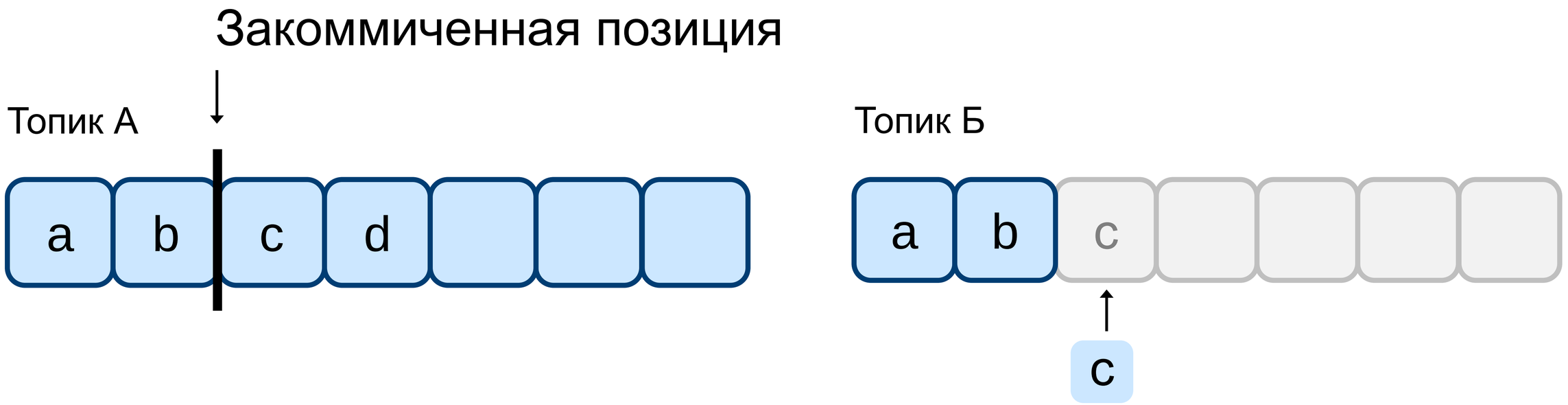
Задача решардирования. Exactly-once

1



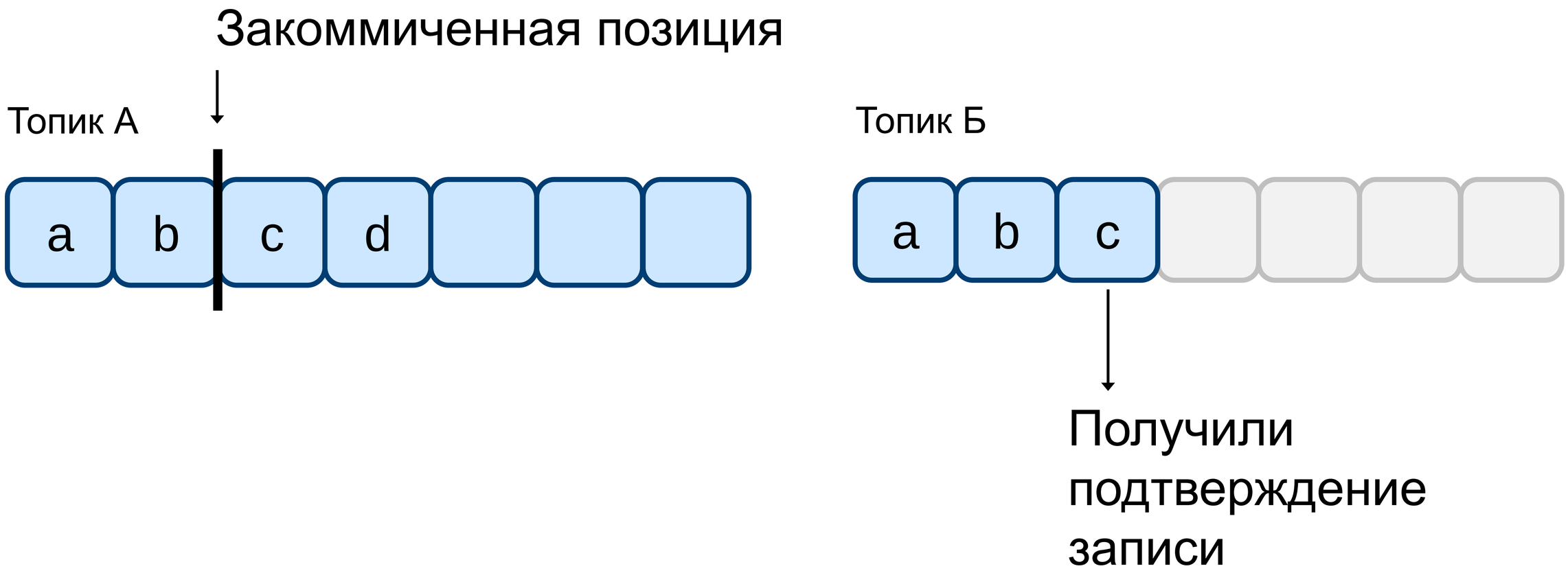
Задача решардирования. Exactly-once

2



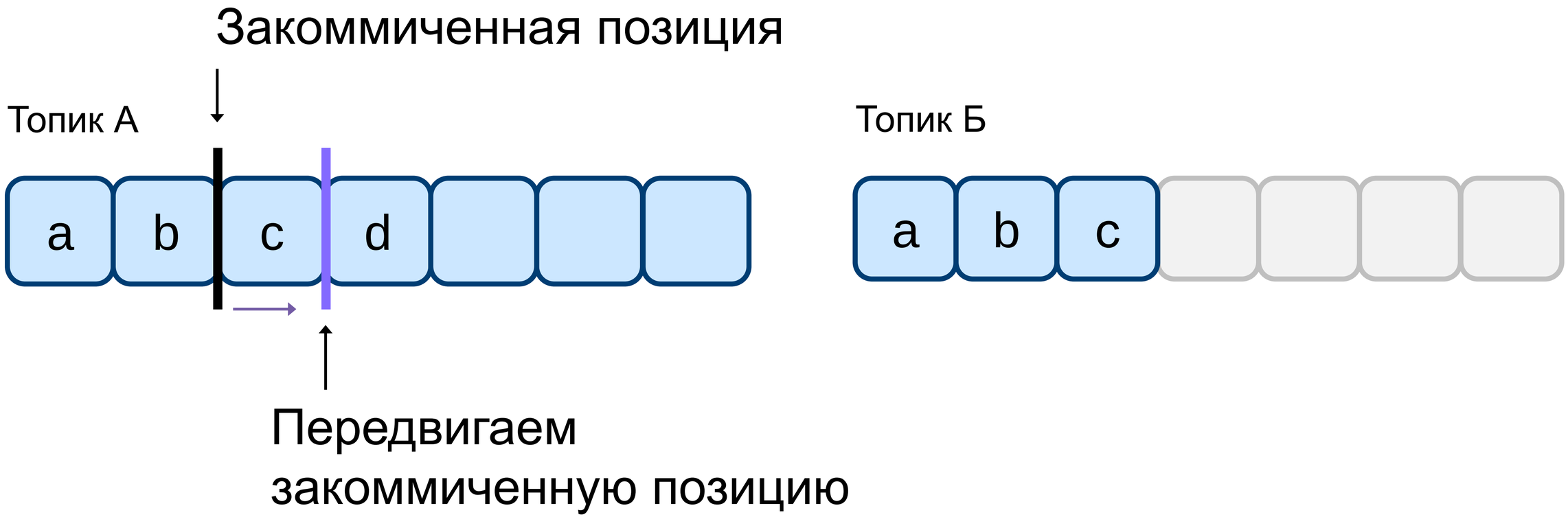
Задача решардирования. Exactly-once

3



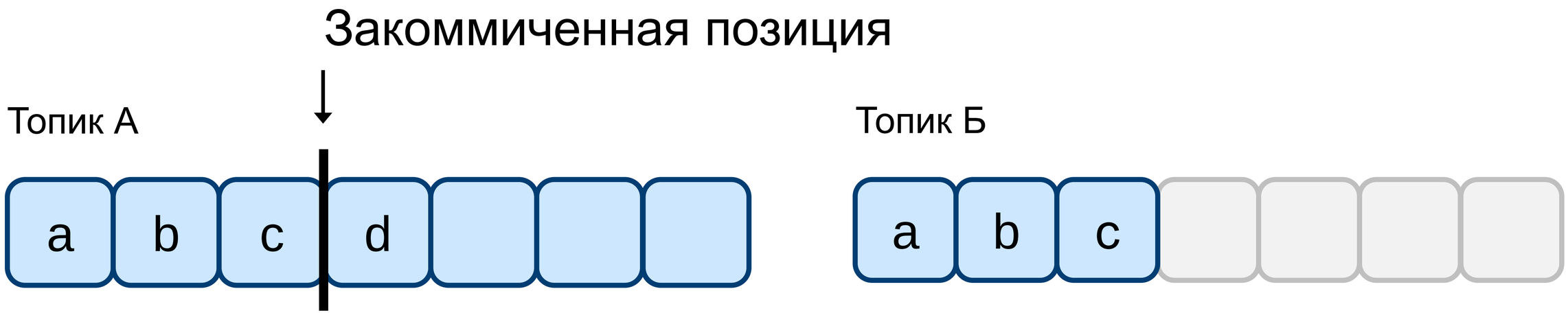
Задача решардирования. Exactly-once

4



Задача решардирования. Exactly-once

5

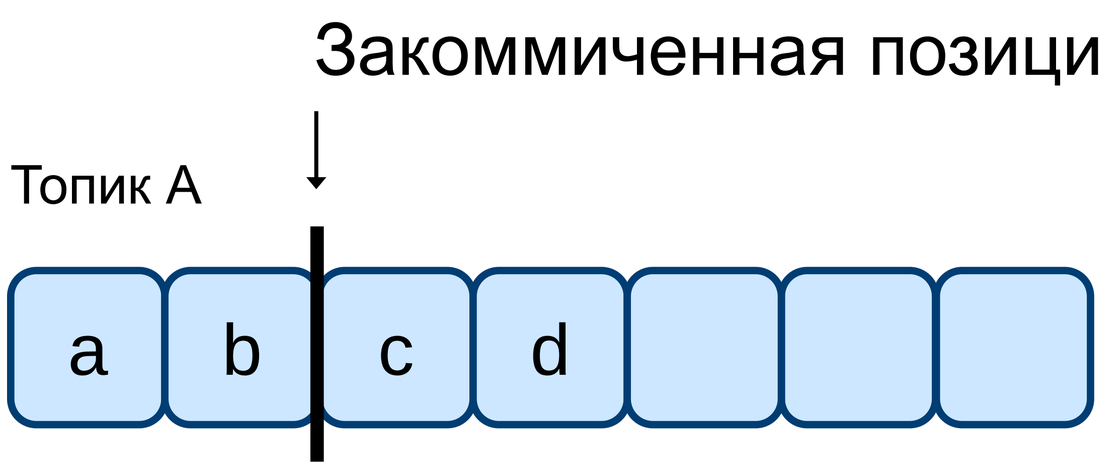


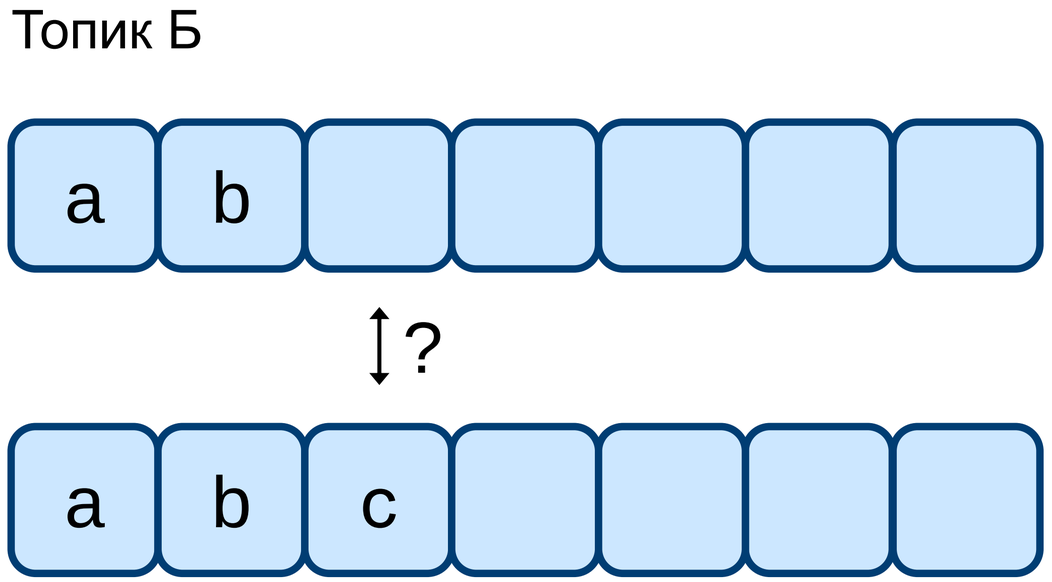
c обработан.

Начинаем новый цикл

Задача решардирования. Exactly-once

?





Был рестарт. Состояние исходного топика – c еще не обработан. А в выходном?

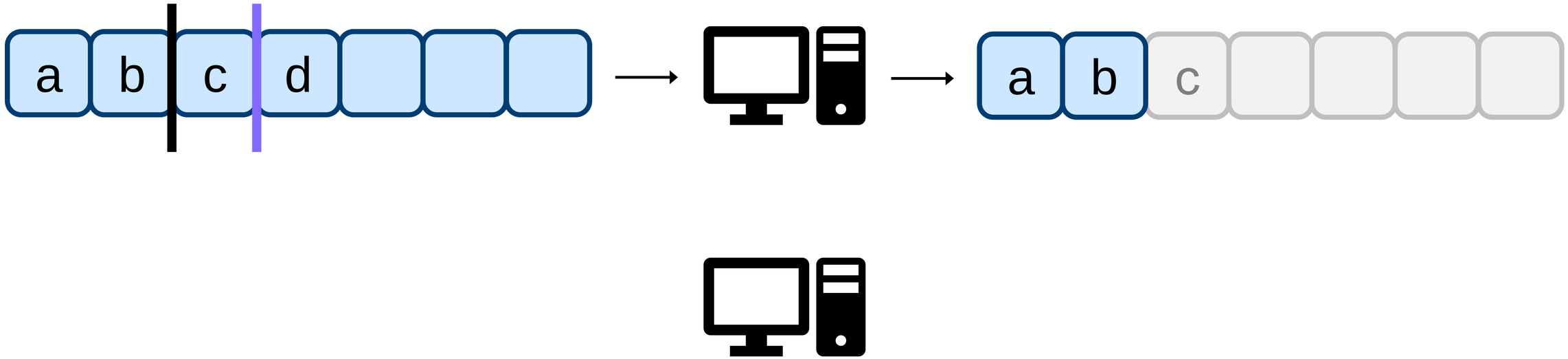
✔ Решается за счет Atomicity

Транзакции. Consistency – что за зверь?

- Atomicity – есть
- Consistency – ?
- Isolation – есть
- Durability – есть

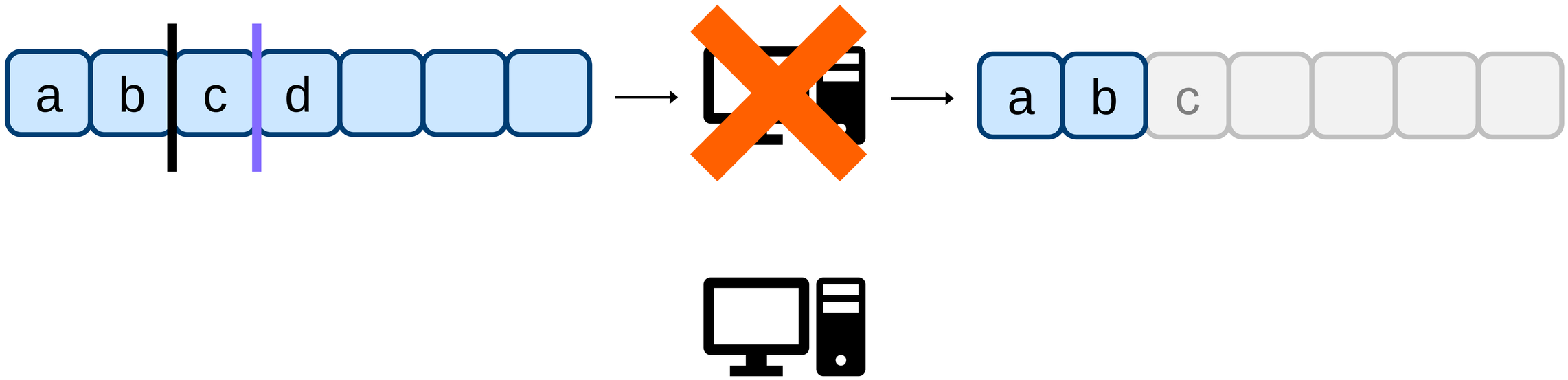
Конфликты между транзакциями

1



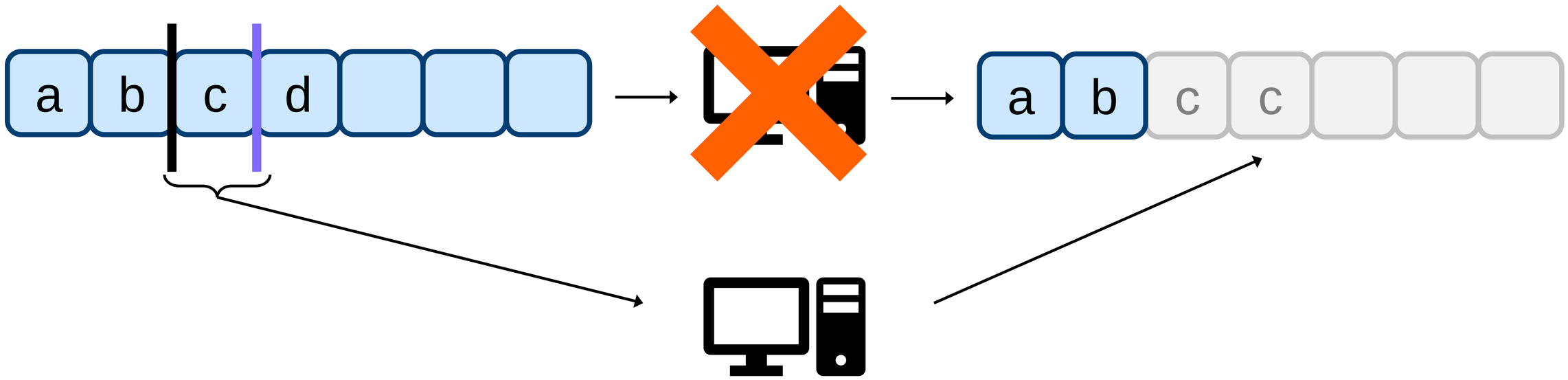
Конфликты между транзакциями

2



Конфликты между транзакциями

?



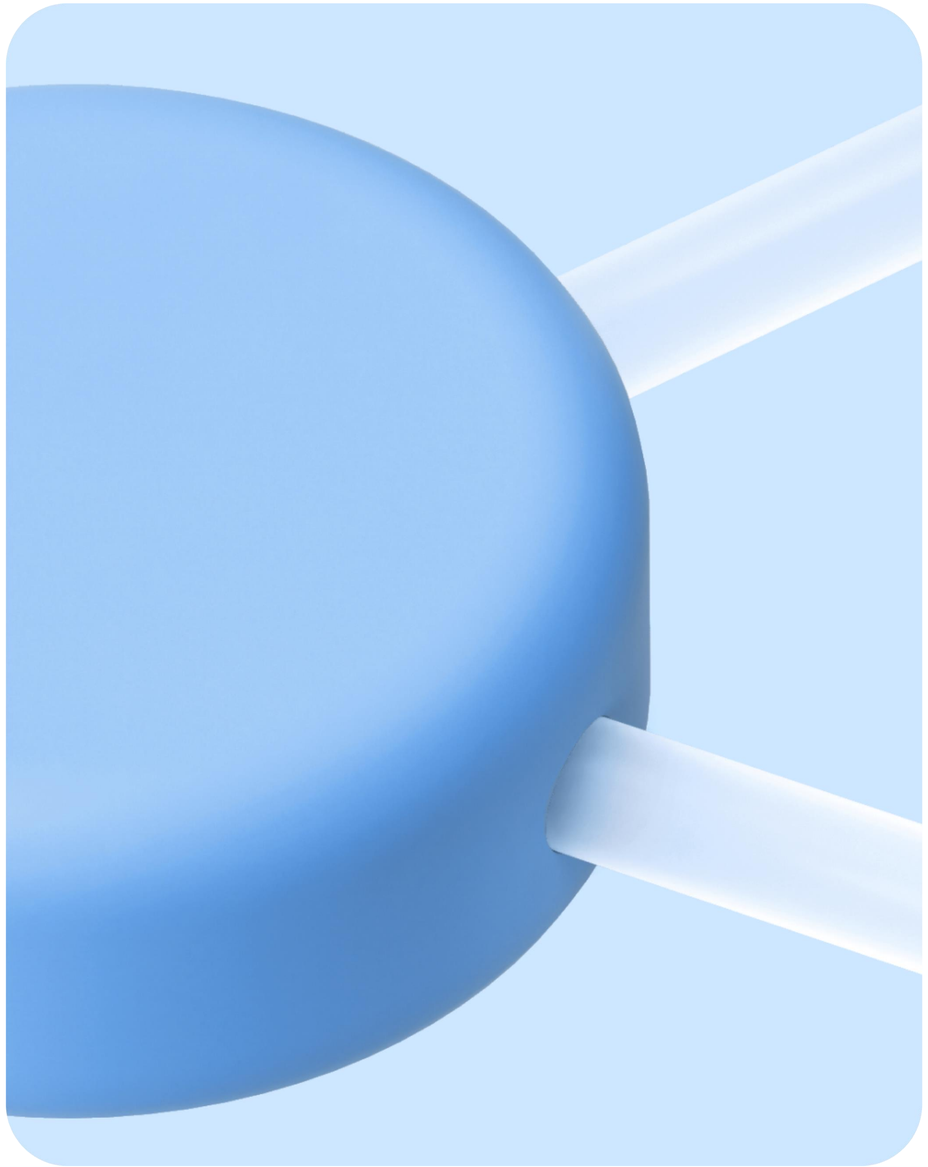
Обе транзакции коммитят.

c будет записан 2 раза!

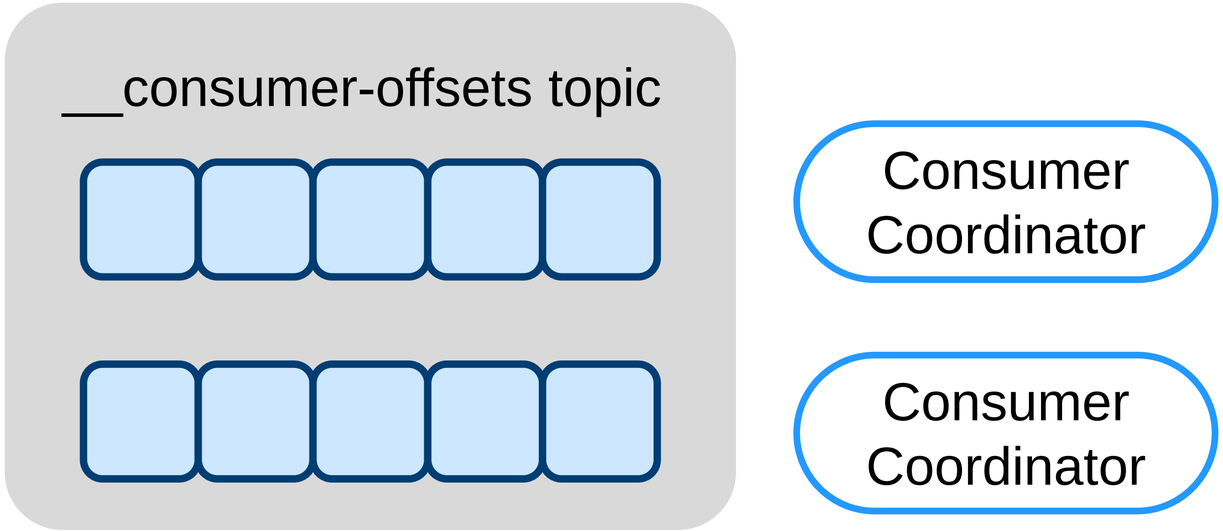
Решается
за счет (C)onsistency

4

Транзакции в Apache Kafka



Архитектура транзакций Kafka. Consumer coordinator

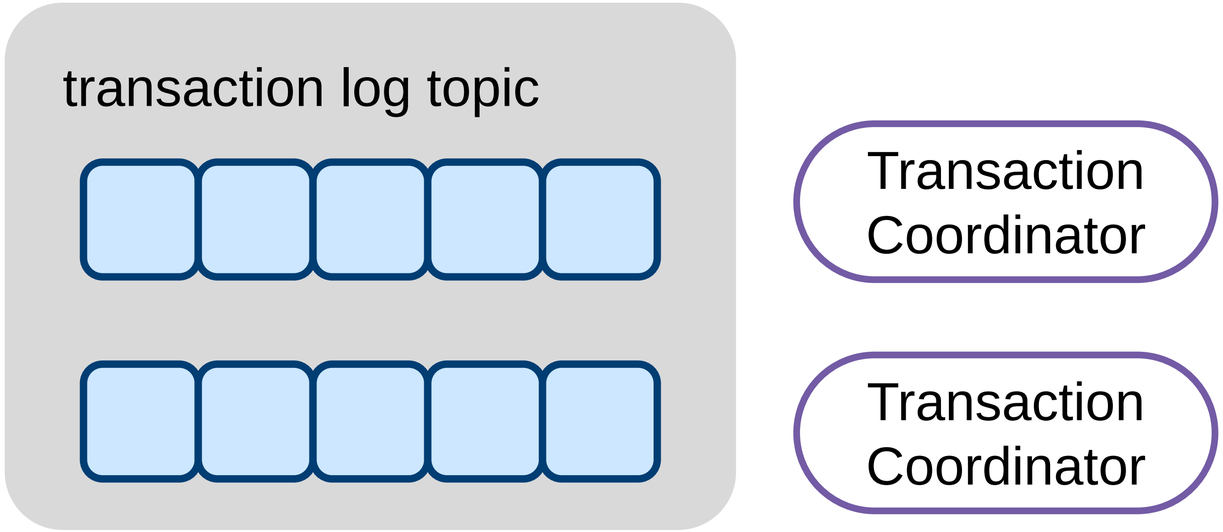


- Compacted topic. Ключ – Consumer
- В топике хранятся состояния читателей
- Поверх партиции есть процесс, отвечающий за читателей в этой партиции

Состояние читателя:

- список закоммиченных позиций по партициям
- **список транзакционных изменений, еще незакоммиченных**

Архитектура транзакций Kafka. Transaction coordinator



- Compacted topic. Ключ – TxnId
- В топике хранятся состояния транзакций
- Поверх партиции есть процесс, отвечающий за транзакции внутри неё

Состояние транзакции:

- epoch – инкрементальное поколение транзакции
- список партиций в транзакции
- список читателей в транзакции
- статус транзакции: в процессе, коммитится, закоммичена, отменяется, отменена

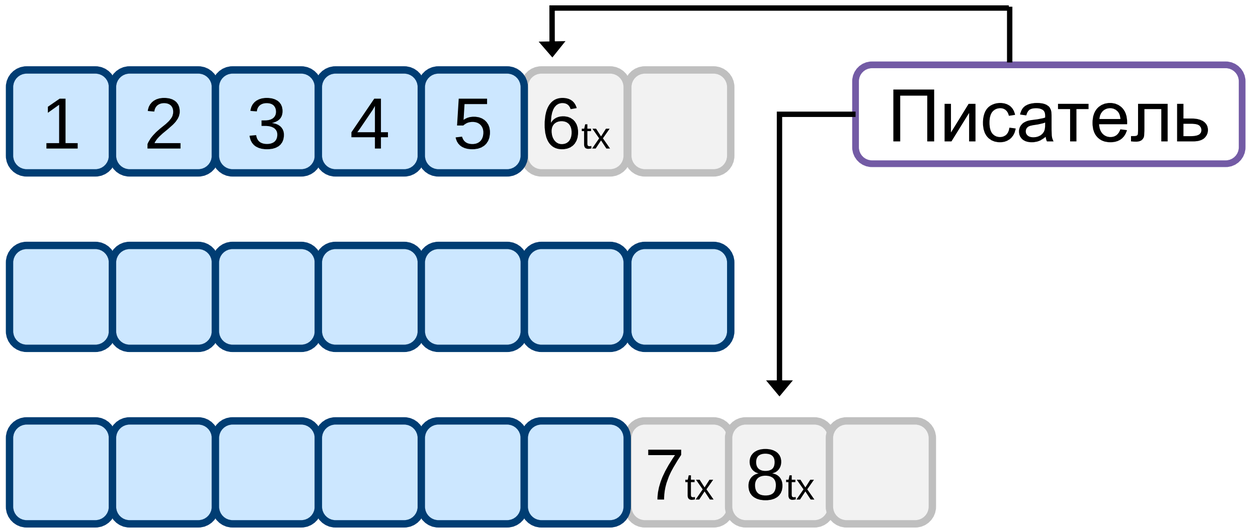
Архитектура транзакций Kafka. Coordinators

- Transaction и Consumer Coordinator – **Replicated State Machines**
- Coordinator рано или поздно запустится вместе с лидером партиции
- Не более одного coordinator, способного писать в партицию
- Coordinator – надежный, консистентный и высокодоступный

Архитектура транзакций Kafka. Алгоритм

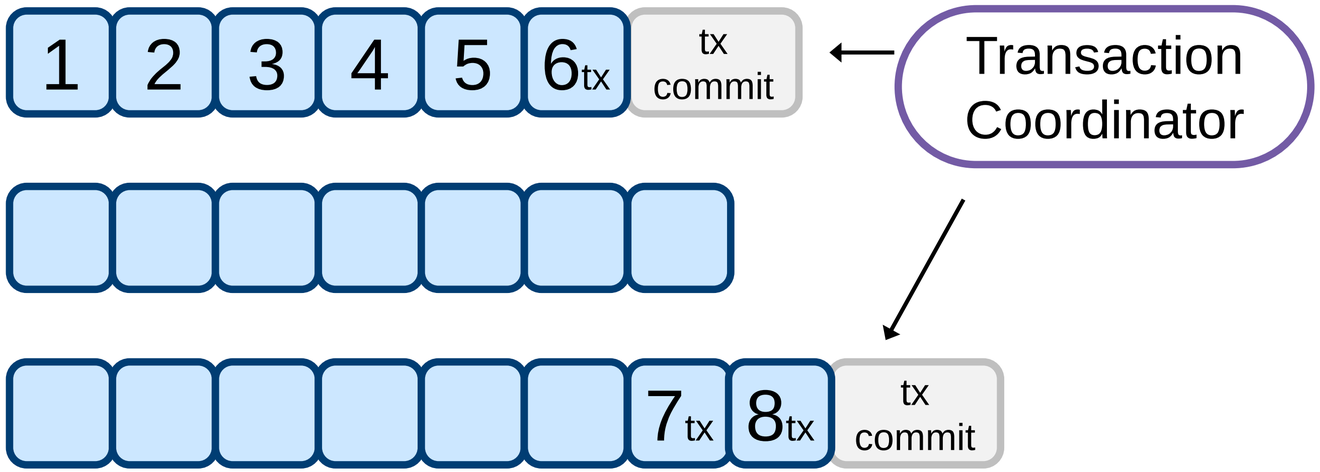
1. В transaction coordinator фиксируются участники транзакции (партиции для записи, читатели для изменений позиций)
2. Фиксируются изменения в каждом участнике(в партиции и consumer coordinator’e), но они недоступны клиенту («сбоку»)
3. Фиксируется желание закоммитить транзакцию в transaction coordinator
4. Transaction coordinator рано или поздно опубликует изменения во всех участниках

Архитектура транзакций Kafka. Запись. Фиксация изменений «сбоку»



- Данные пишутся с указанием transactionId в метаданных
- Данные доступны на чтение сразу
- Вся магия на чтении

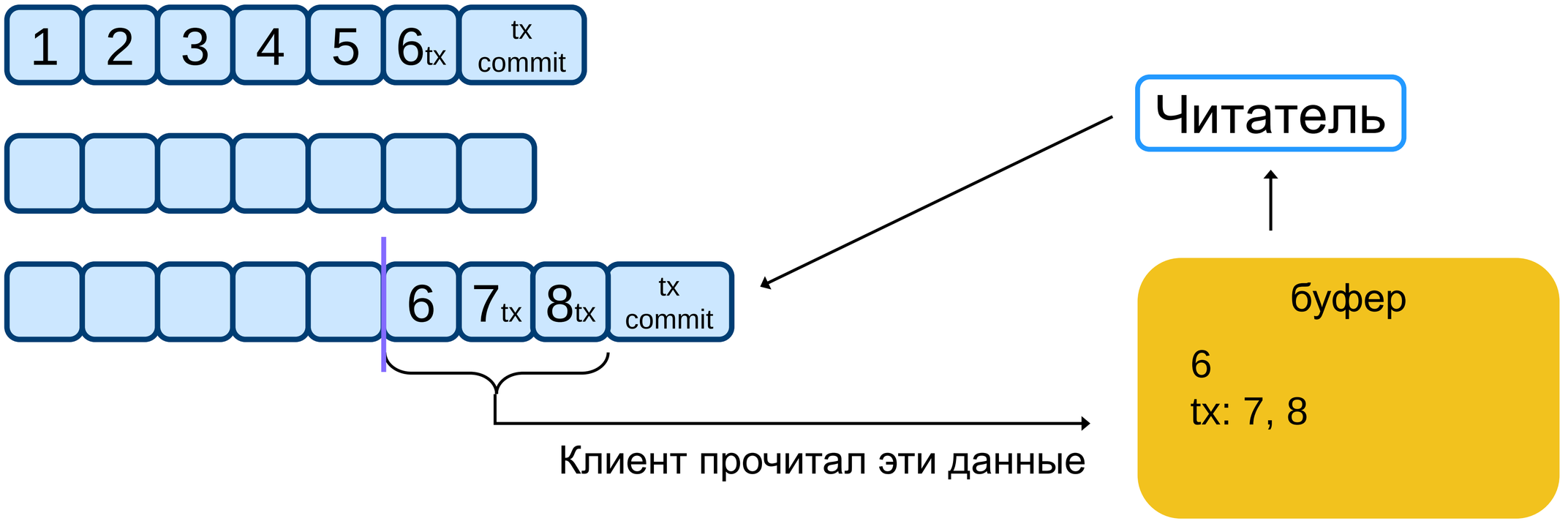
Архитектура транзакций Kafka. Запись. Публикация изменений



- Transaction coordinator пишет специальное сообщение (commit marker)
- Это сообщение доступно на чтение сразу
- Вся магия на чтении

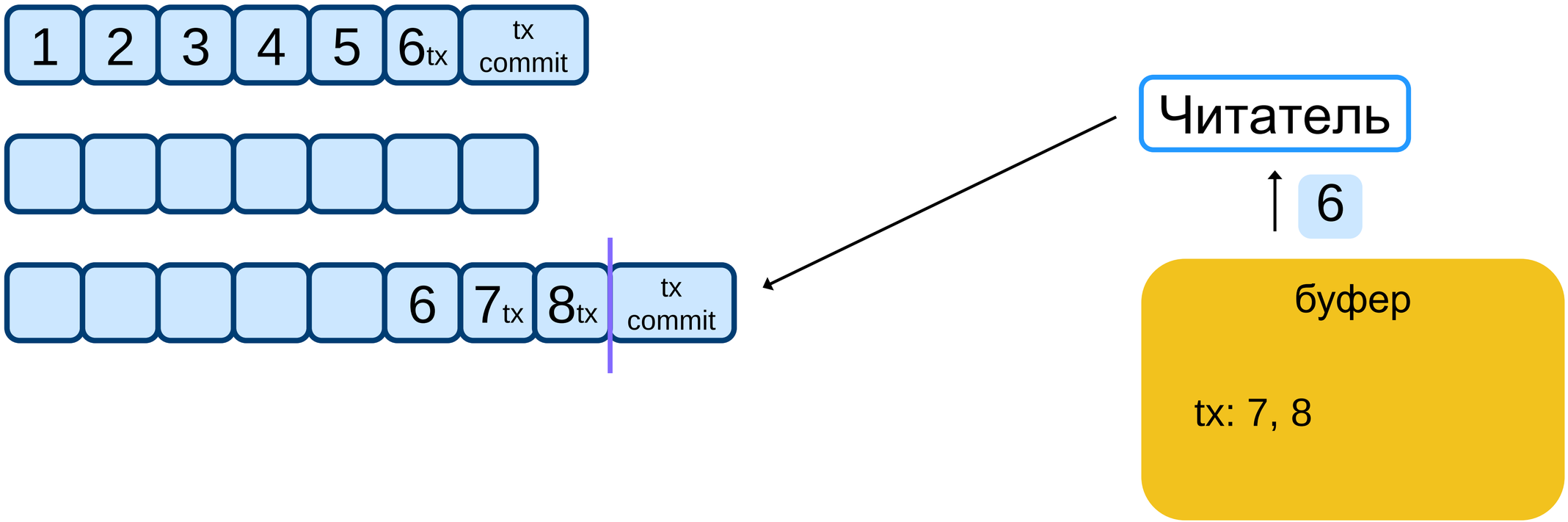
Архитектура транзакций Kafka. Чтение

1



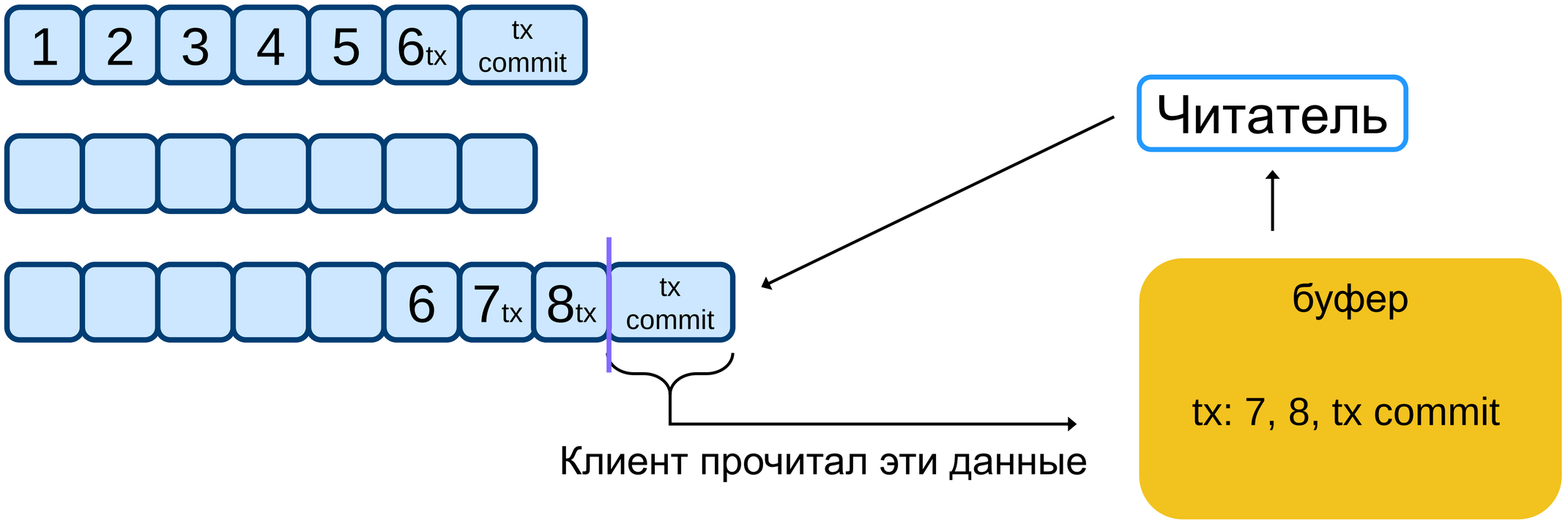
Архитектура транзакций Kafka. Чтение

2



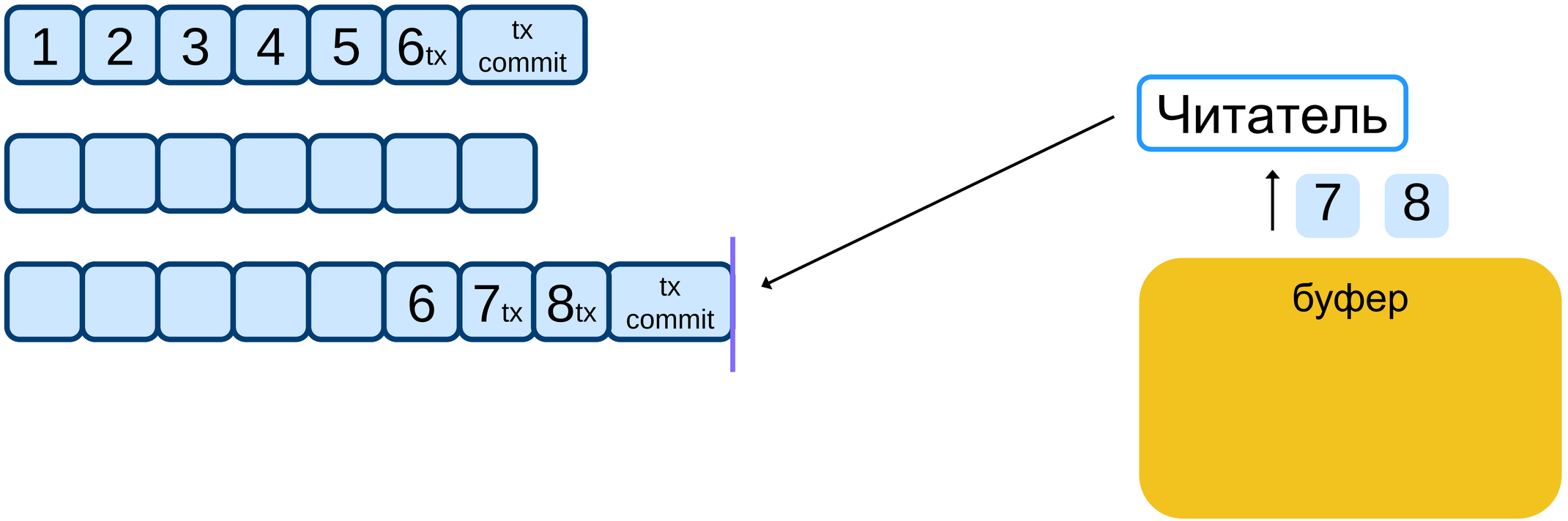
Архитектура транзакций Kafka. Чтение

3

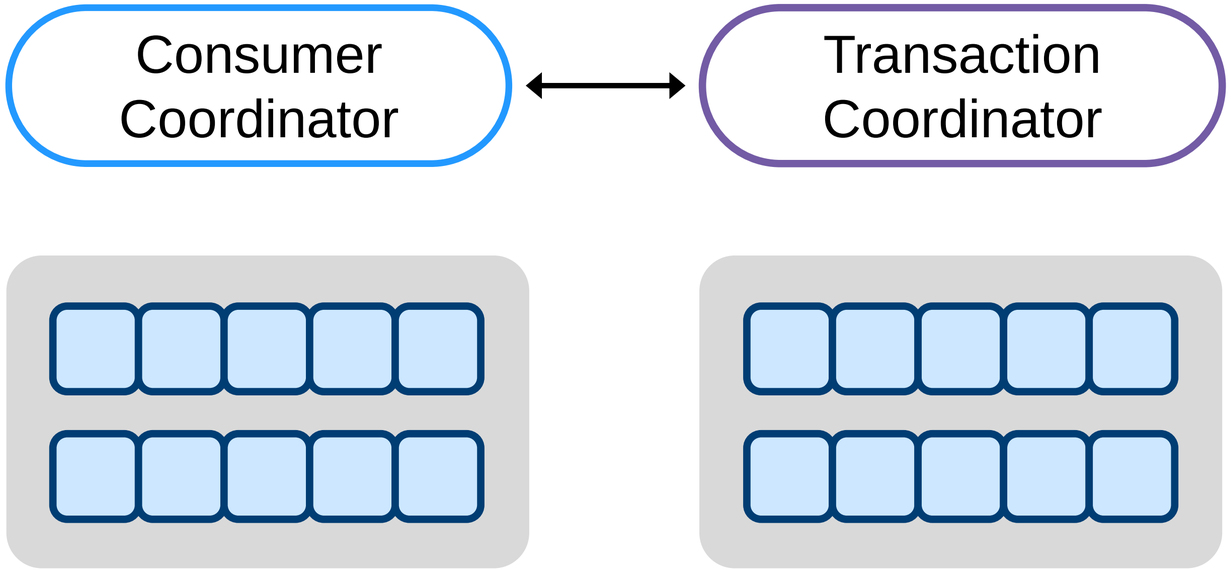


Архитектура транзакций Kafka. Чтение

4



Архитектура транзакций Kafka. Изменение позиций



- Фиксация изменений = сохранить в состоянии читателя новую позицию «сбоку»
- Эти изменения недоступны клиенту
- Публикация изменений = поменять закоммиченную позицию

Архитектура транзакций Kafka. Конфликты между транзакциями

- Epoch транзакции гарантирует zombie fencing в части записи
- Двигать позицию читателя может только текущий процесс чтения с точки зрения consumer coordinator’a

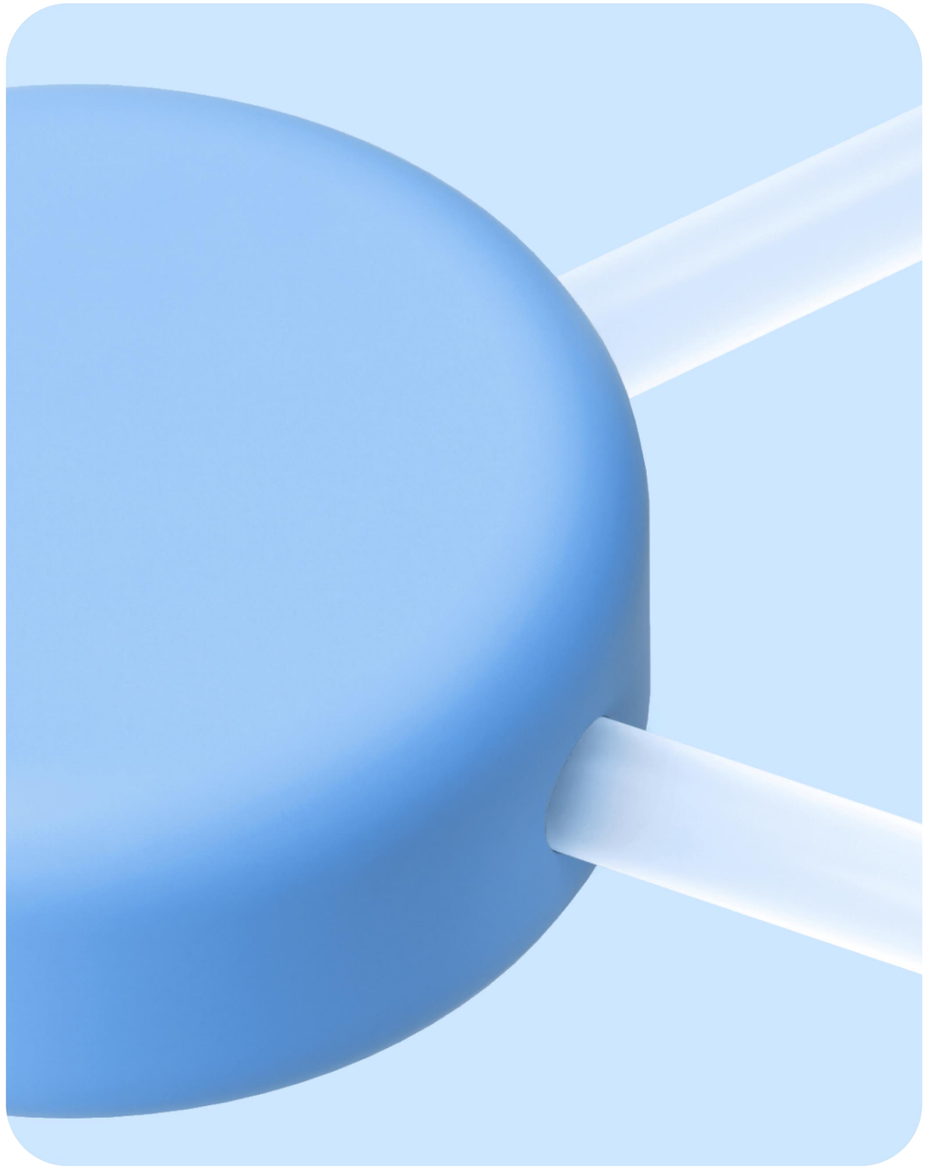
Защита работает с Kafka версии 3.7 и выше и актуальными SDK

Архитектура транзакций Kafka. Итог

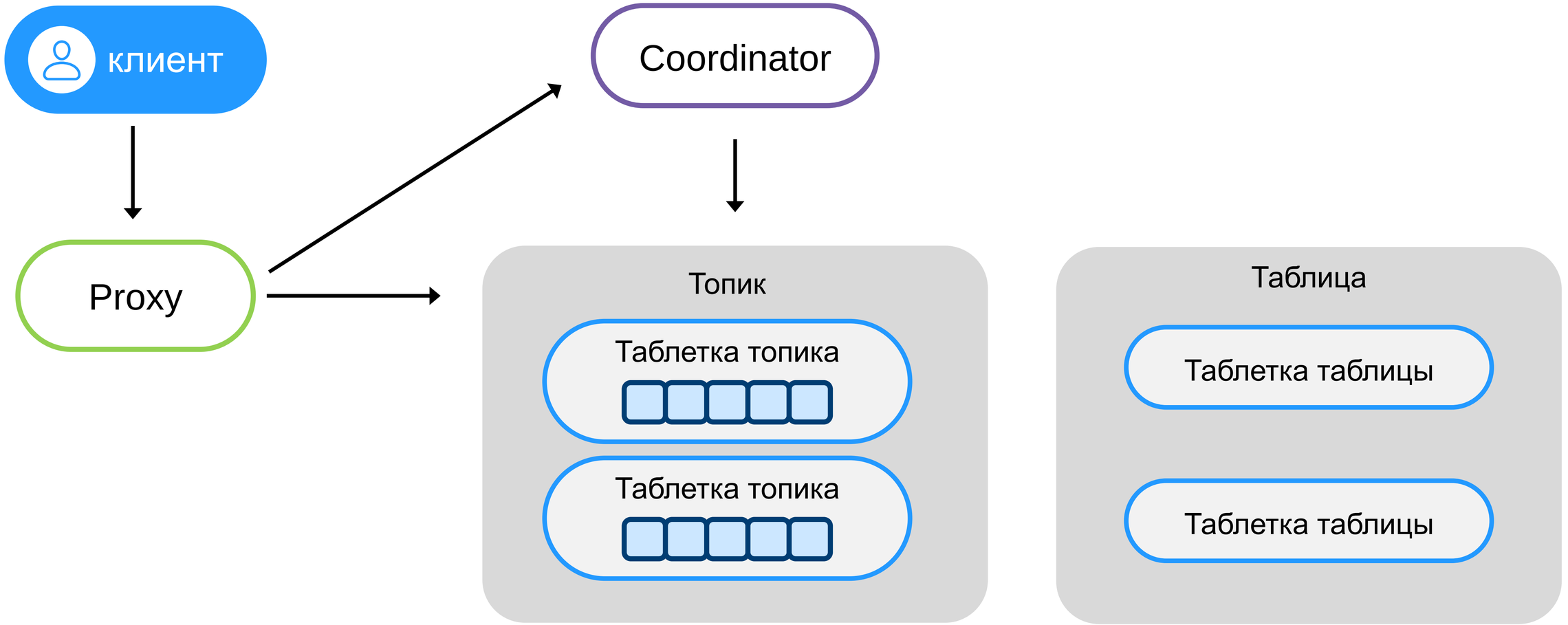
1. Кафка изначально дизайнилась без учета транзакций
2. Транзакции внесены в топики инородно, с размазыванием логики между записью, чтением и клиентом
3. Кафка позволяет делать Replicated State Machines и активно внедряет этот механизм

5

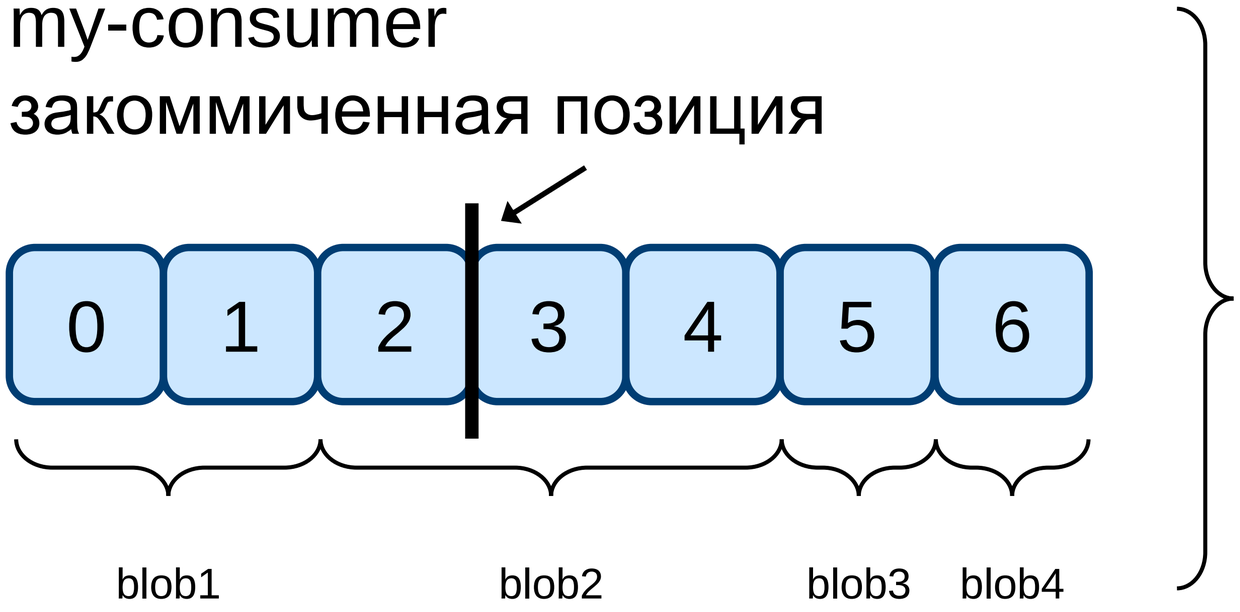
Транзакции в YDB



Архитектура транзакций YDB. Компоненты



Архитектура транзакций YDB. Таблетка топика



Key	Value
b_0	blob1_id
b_2	blob2_id
b_5	blob3_id
b_6	blob4_id
c_my-consumer	3

1. **Таблетка – Replicated State Machine**
2. Состояние таблетки – таблица
3. В таблице может быть как значение, так и ссылка на блоб данных

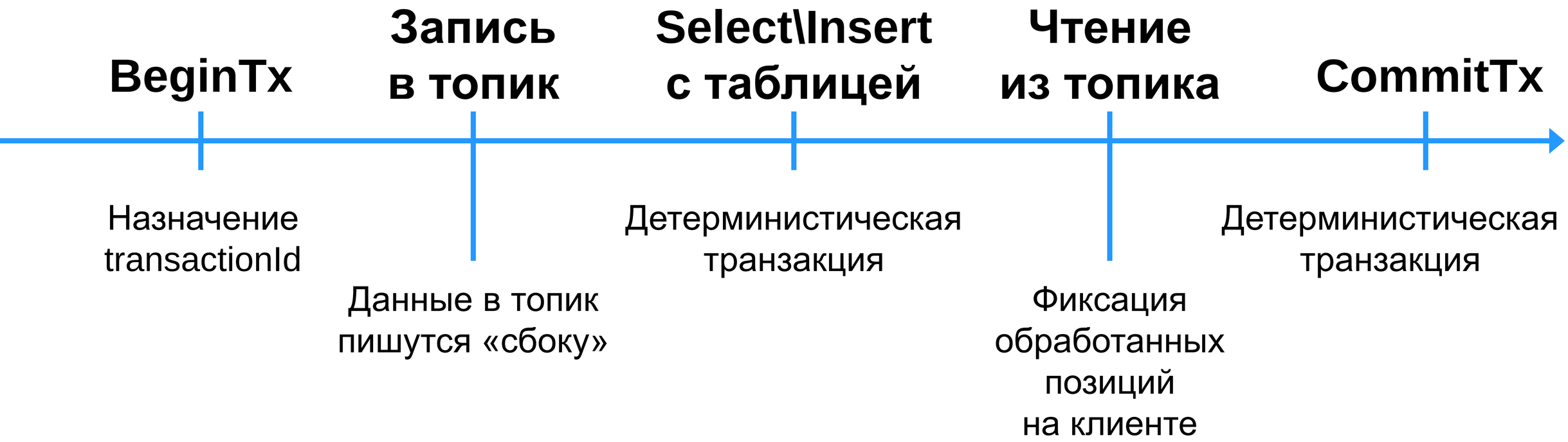
Архитектура транзакций YDB. Детерминистические транзакции

- Точечные распределенные транзакции
- Упорядочены глобально по виртуальному времени
- О возможности выполнить транзакцию таблетки договариваются в момент выполнения

Архитектура транзакций YDB. Координатор

- Координатор – это таблетка
- Запоминает участников детерминистической транзакции
- Назначает виртуальное время транзакции
- Сообщает участникам очередное время и какие транзакции выполнить

Архитектура транзакций YDB. Большая транзакция



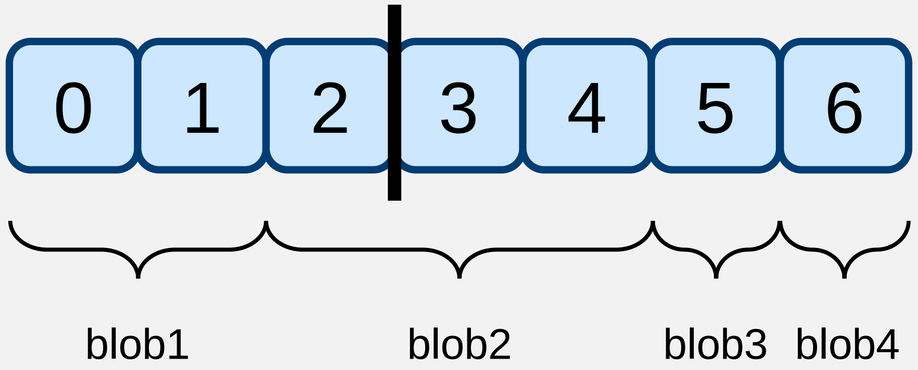
Архитектура транзакций YDB. Алгоритм

1. Фиксируются данные записи в партиции «сбоку»
2. Во всех партициях фиксируется: dTxId, обработанные позиции читателей, привязка записанных ранее данных
3. В coordinator фиксируется: желание закоммитить транзакцию dTxId, список партиций
4. Coordinator рано или поздно пришлет команду на выполнение транзакции всем партициям
5. Партиции договариваются о возможности выполнить транзакцию и публикуют изменения

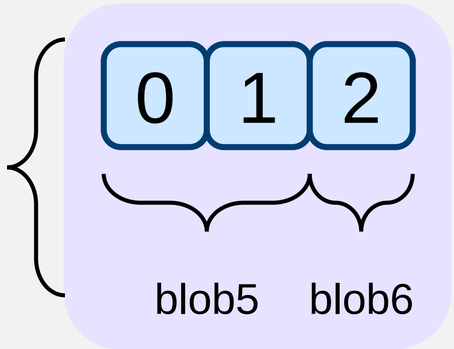
Детерминистическая транзакция

Архитектура транзакций YDB.
Запись. Фиксация изменений «сбоку»

Таблетка партиции

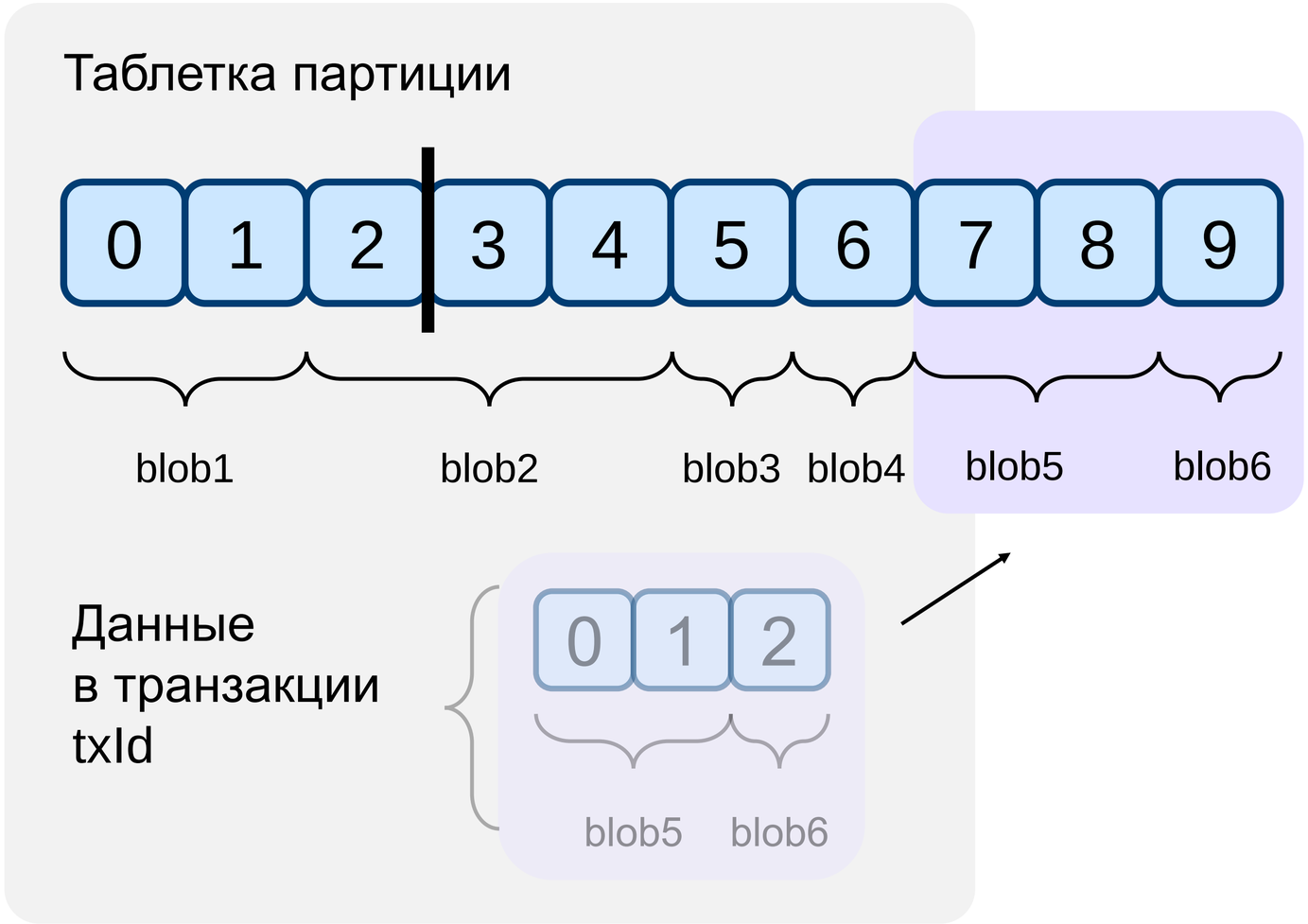


Данные
в транзакции
txId



Key	Value
b_0	blob1_id
b_2	blob2_id
b_5	blob3_id
b_6	blob4_id
c_my-consumer	3
t_txId_0	**blob5_id**
t_txId_2	**blob6_id**

Архитектура транзакций YDB.
Запись. Публикация изменений



Key	Value
b_0	blob1_id
b_2	blob2_id
b_5	blob3_id
b_6	blob4_id
c_my-consumer	3
b_7	blob5_id
b_9	blob6_id

Архитектура транзакций YDB. Чтение

Без изменений!

Архитектура транзакций YDB. Изменение позиций

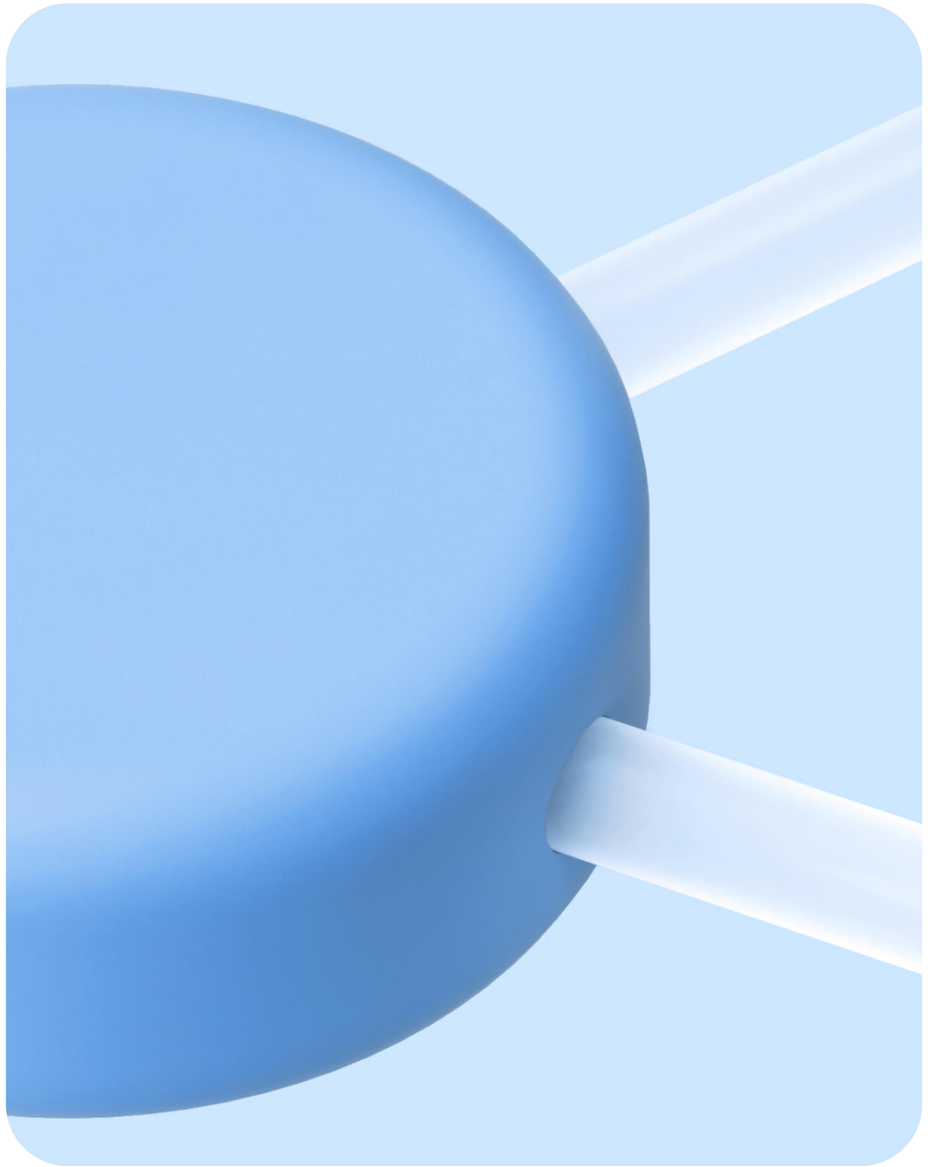
- Клиент (SDK) запоминает **все вычитанные позиции** в виде отрезков позиций
- Клиент передает на сервер желание закоммитить ровно эти позиции, а не одно число, как в Kafka
- Если позиция в партиции пересекается с переданными позициями, **транзакция будет отменена**

Архитектура транзакций YDB. Итог

1. В YDB были транзакции между таблицами, топики органично вписались в эту архитектуру
2. Транзакционная запись не влияет на чтение
3. В YDB в одной транзакции можно использовать и топики и таблицы

6

Сравнение транзакций в Kafka и YDB



Сравнение. Алгоритмы

Kafka	YDB	Сохранение на диск: Kafka	Сохранение на диск: YDB
Регистрация TxId в transaction coordinator	Регистрация транзакции в Proxy	✓	✗
Регистрация партиций/консумеров в transaction coordinator	Регистрация партиций/читателей в Proxy	✓	✗
Сохранение сообщений/позиций «сбоку» в партиции/Consumer coordinator	Сохранение сообщений «сбоку» в партиции	✓	✓
—	Proxy фиксирует во всех партициях изменения для публикации	—	✓
Фиксация требования закоммитить транзакцию в transaction coordinator	Фиксация в coordinator требования закоммитить транзакцию	✓	✓
Публикация изменений в партиции/consumer coordinator по требованию transaction coordinator	Публикация изменений в партиции по требованию coordinator (вычисление и обмен предикатами, публикация изменений)	✓	✓ x3

Сравнение. Возможности

- **Таблицы и транзакции с ними**
  У YDB есть полноценные таблицы и транзакции с ними, у Kafka – нет

- **У Kafka пессимистичные транзакции, у YDB оптимистичные**
  У Kafka новая транзакция будет ждать коммита/отмены старой, у YDB не будет, но одна из транзакций будет отменена в момент попытки коммита

- **Способ сохранения сообщений «сбоку»**
  У Kafka прост и гениален, но усложняет чтение, восстановление после сбоев

Сравнение. Производительность

Параметры теста:

- Интервал коммита – 500мс
- 100 партиций
- 100mb/s записи

Параметры железа:

8 хостов; 2 процессора CPU Xeon E5-2660V4 (56 логических ядер в сумме); 256 ГБ ОЗУ; 2 nMVE Micron 3.2 ТБ U.2

Задержка записи:

Система	p50 latency ms	p95 latency ms	p99 latency ms	p99.9 latency ms
YDB	4	79	153	275
Kafka	28	73	96	109

End-to-end задержка:

Система	p50 latency ms	p95 latency ms	p99 latency ms	p99.9 latency ms
YDB	367	575	644	690
Kafka	321	550	577	605

Сравнение. Производительность

Параметры теста:

- Интервал коммита – 500мс
- 100 партиций
- 800mb/s записи

Параметры железа:

8 хостов; 2 процессора CPU Xeon E5-2660V4 (56 логических ядер в сумме); 256 ГБ ОЗУ; 2 nMVE Micron 3.2 ТБ U.2

Система	CPU (cores)	Network in (mb/s)	Network out (mb/s)	Disk write (mb/s)
YDB с транзакциями	82	4400	3800	3800
YDB без тразнакций	56	4200	3350	3500
Kafka с транзакциями	24	2400	1600	3600
Kafka без транзакций	16	2400	1600	3600

Выводы

Шли с разных концов,
получили почти одно и то же

Kafka + PostgreSQL
+ транзакции -> YDB

Обе архитектуры можно
улучшить

Дальнейшие планы:

- сравнить работу систем в случае сбоев
- улучшать транзакции в YDB

Спасибо за внимание. Голосуйте за доклад!

Алексей Николаевский

Яндекс, руководитель сервиса

Яндекс

HL 2024 HighLoad++