Apache Cassandra: 8 лет в продакшене

Владимир Дегтерёв

degterevvy@pochtabank.ru

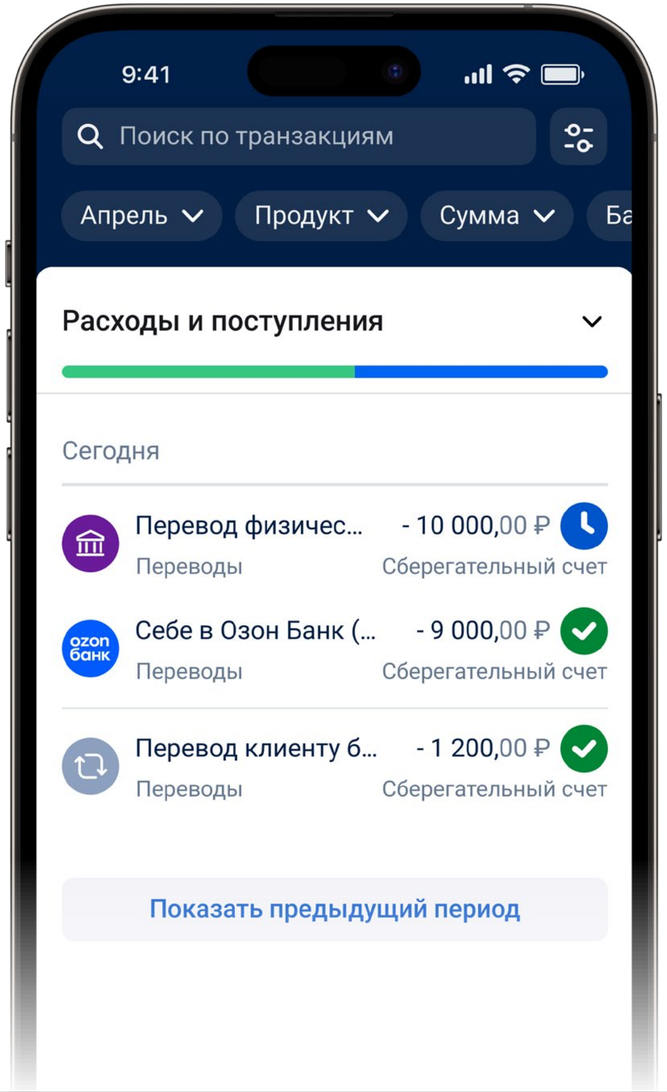
Telegram: @VladimirDegterev

О чём будем говорить

- Почему выбрали Cassandra
- Как мы писали сервис ленты операций
- Как боролись с падением скорости чтения

Постановка задачи

Разработать интернет и мобильный банк



Постановка задачи

Разработать интернет и мобильный банк

Требования к БД:

Постановка задачи

Разработать интернет и мобильный банк

Требования к БД:

- Отказоустойчивость

Постановка задачи

Разработать интернет и мобильный банк

Требования к БД:

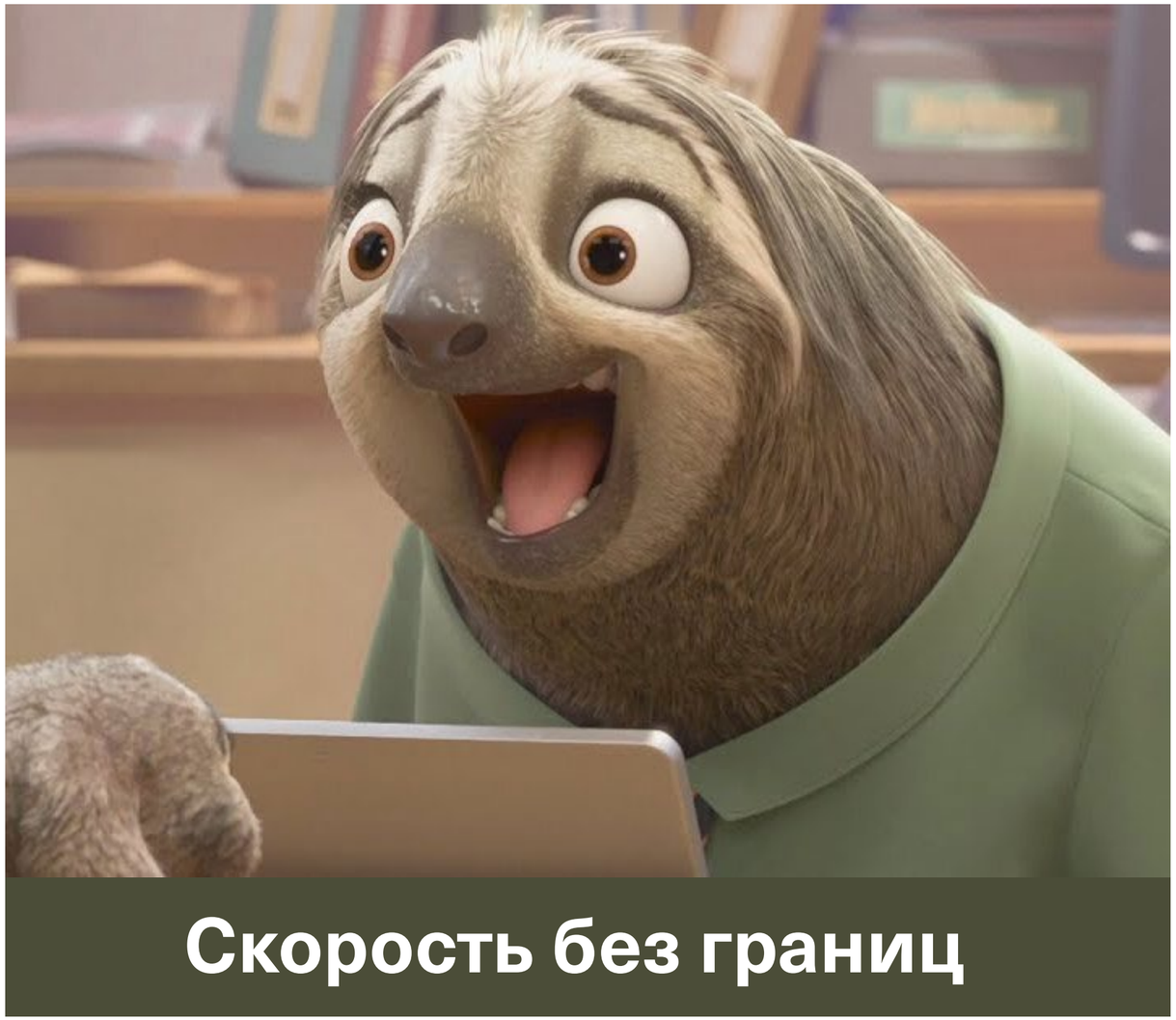
- Отказоустойчивость
- Масштабируемость

Постановка задачи

Разработать интернет и мобильный банк

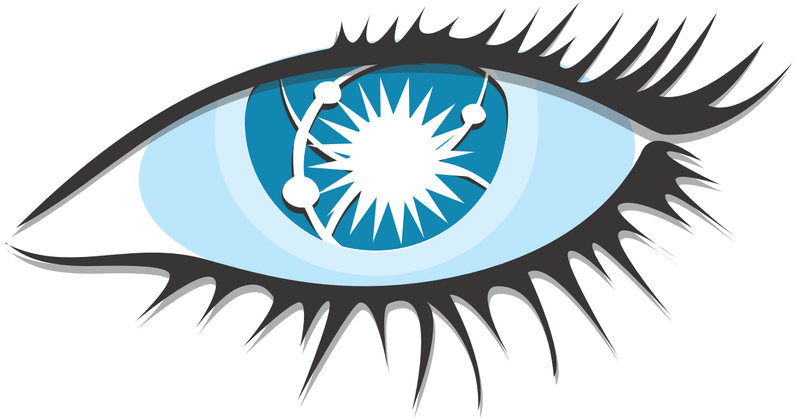
Требования к БД:

- Отказоустойчивость
- Масштабируемость
- Производительность



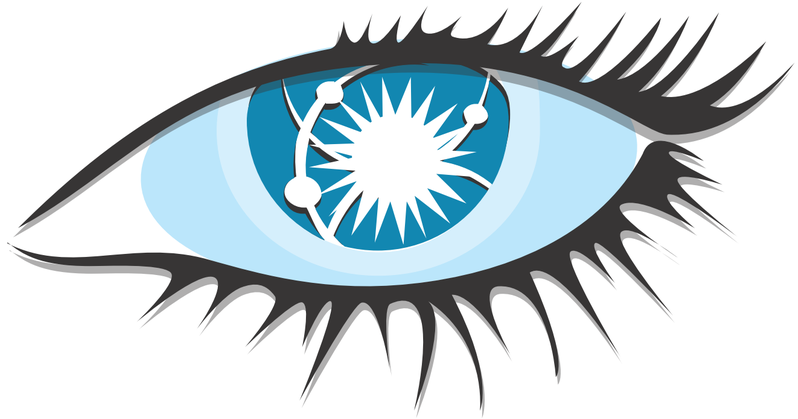
Почему именно Cassandra?

- **Настраиваемая согласованность**



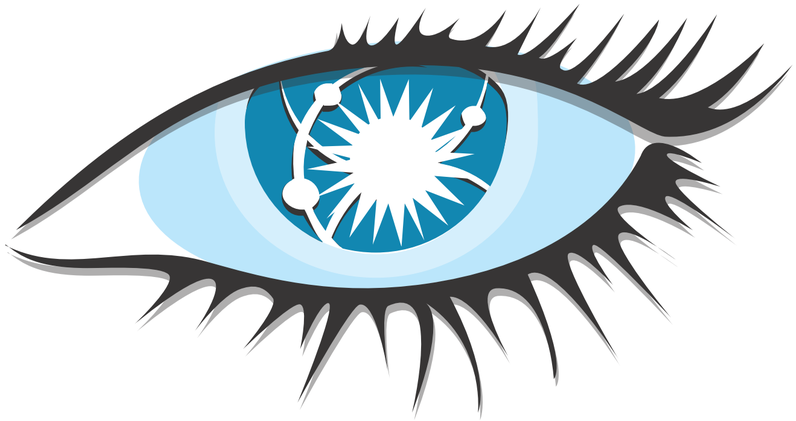
Почему именно Cassandra?

- **Настраиваемая согласованность**
- **Модель чтения/записи**



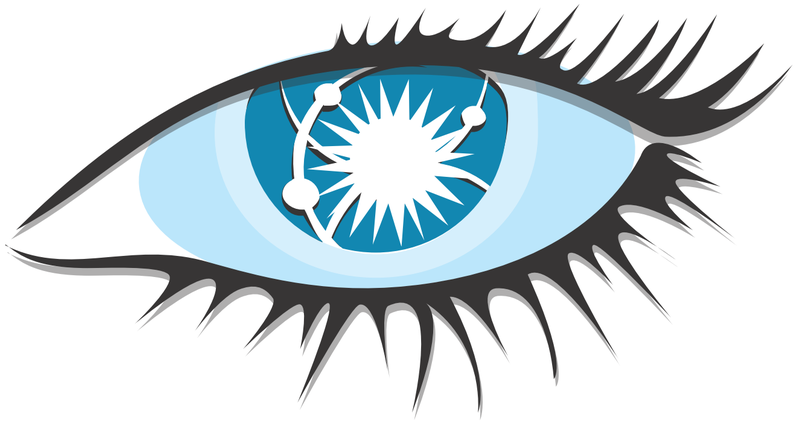
Почему именно Cassandra?

- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model



Почему именно Cassandra?

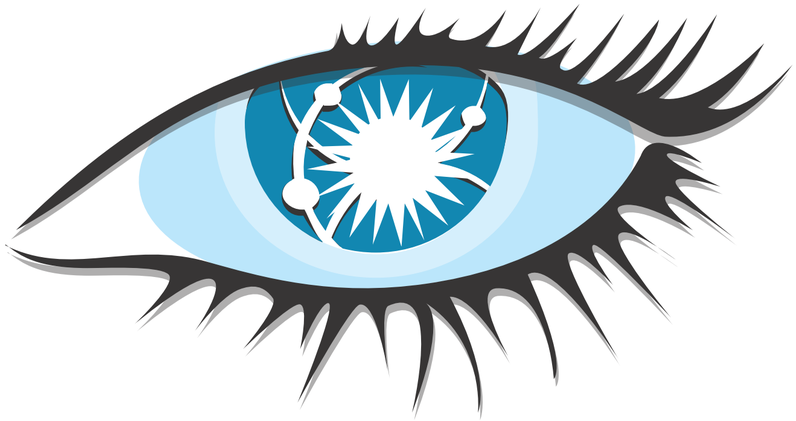
- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model



Partition key: 5	Clustering key: a	x:12	y:56	z:5	Clustering key: b	j:22	k:1

Почему именно Cassandra?

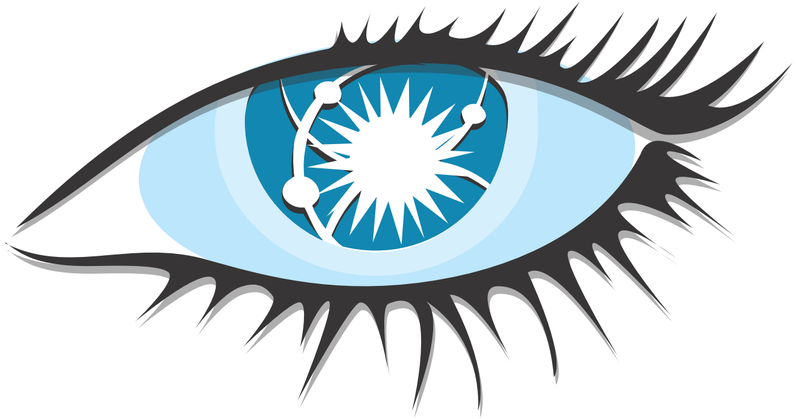
- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model



Partition key: 5	Clustering key: a	x:12	y:56	z:5	Clustering key: b	j:22	k:1

Почему именно Cassandra?

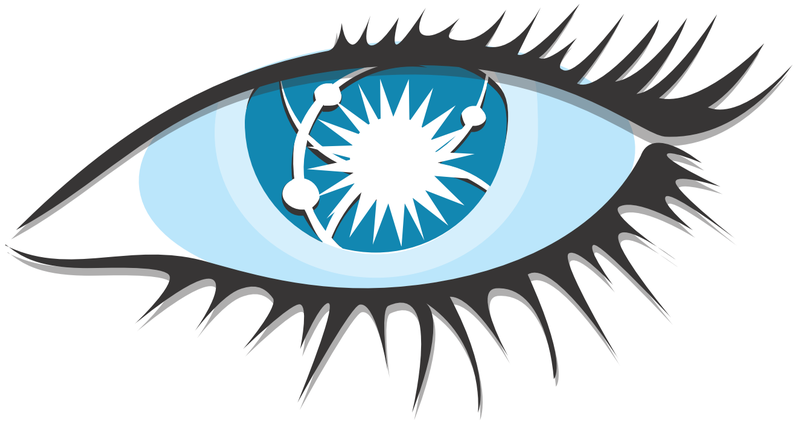
- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model



Partition key: 5	Clustering key: a	x:12	y:56	z:5	Clustering key: b	j:22	k:1

Почему именно Cassandra?

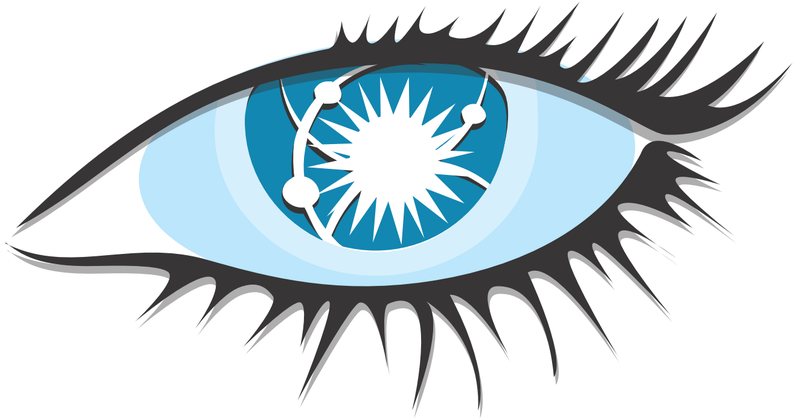
- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model



Partition key: 5	Clustering key: a	x:12	y:56	z:5	Clustering key: b	j:22	k:1

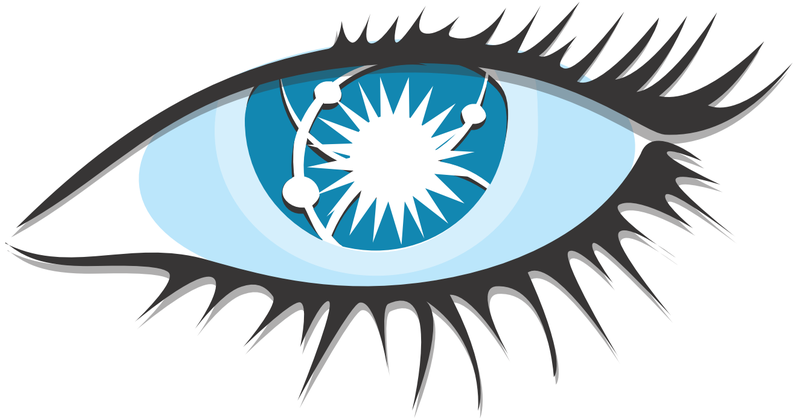
Почему именно Cassandra?

- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model



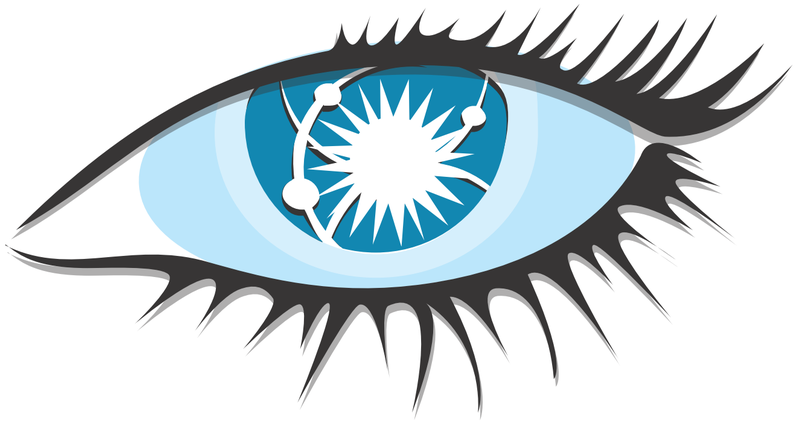
Почему именно Cassandra?

- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model
- Open source



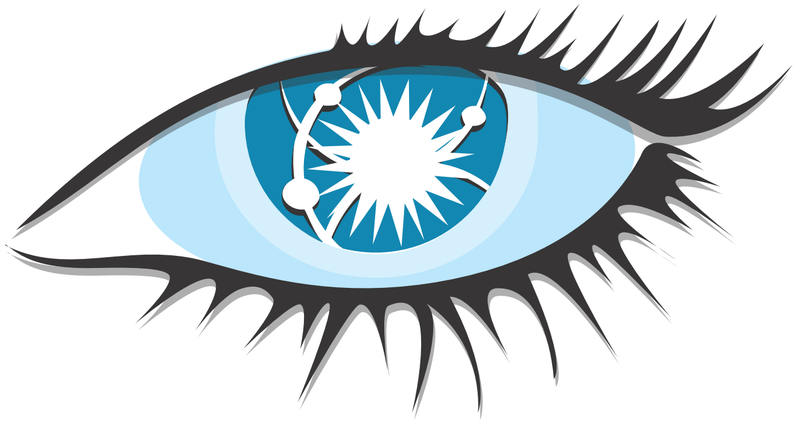
Почему именно Cassandra?

- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model
- Open source
- Java



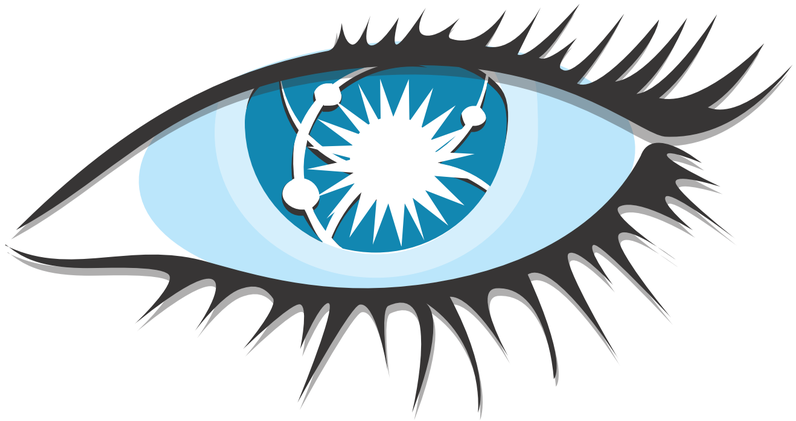
Почему именно Cassandra?

- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model
- Open source
- Java
- Зрелость проекта, сообщество



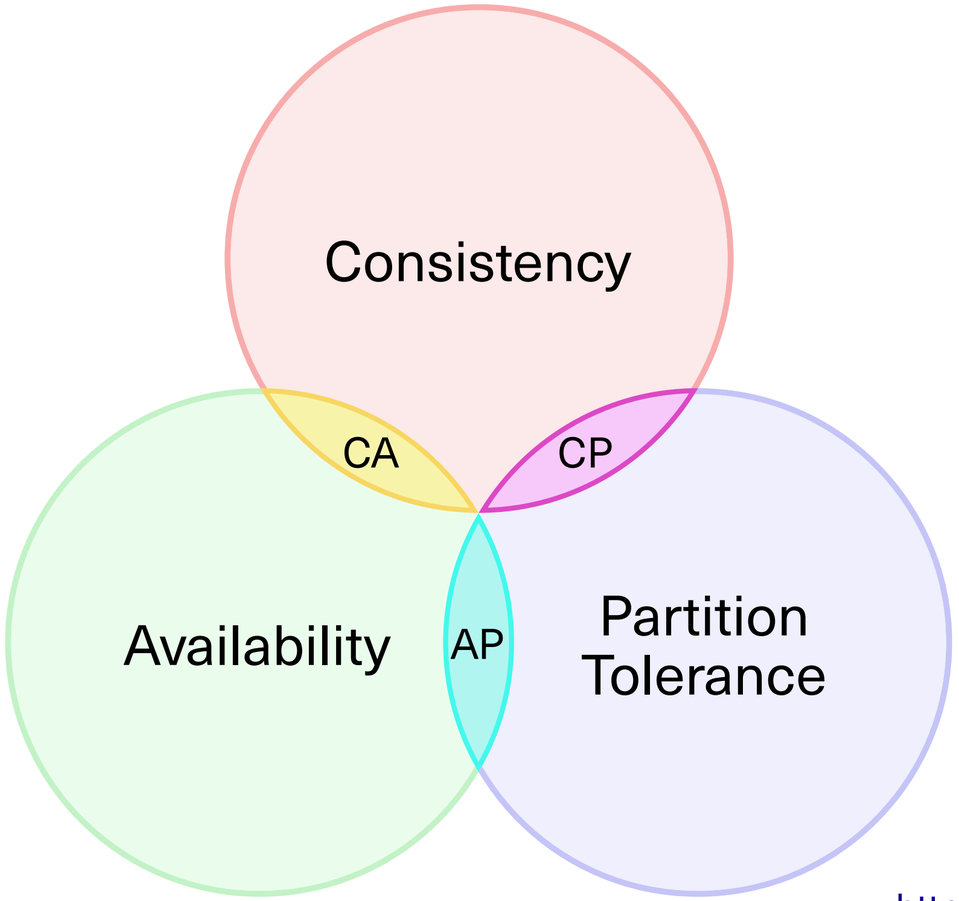
Почему именно Cassandra?

- Настраиваемая согласованность
- Модель чтения/записи
- Partitioned wide column storage model
- Open source
- Java
- Зрелость проекта, сообщество
- ScyllaDB уже вышла, но побоялись



Настраиваемая согласованность

CAP теорема



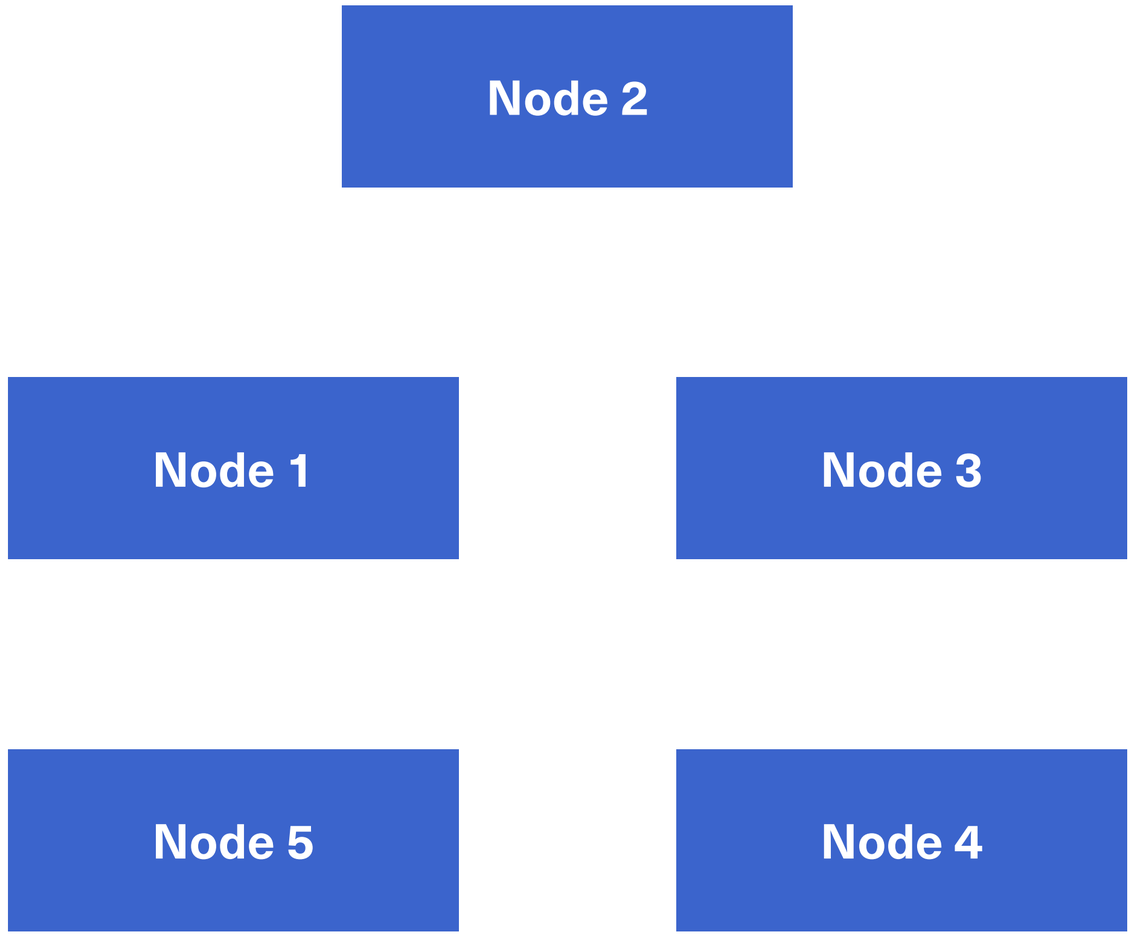
https://en.wikipedia.org/wiki/CAP_theorem

https://en.wikipedia.org/wiki/PACELC_theorem

Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

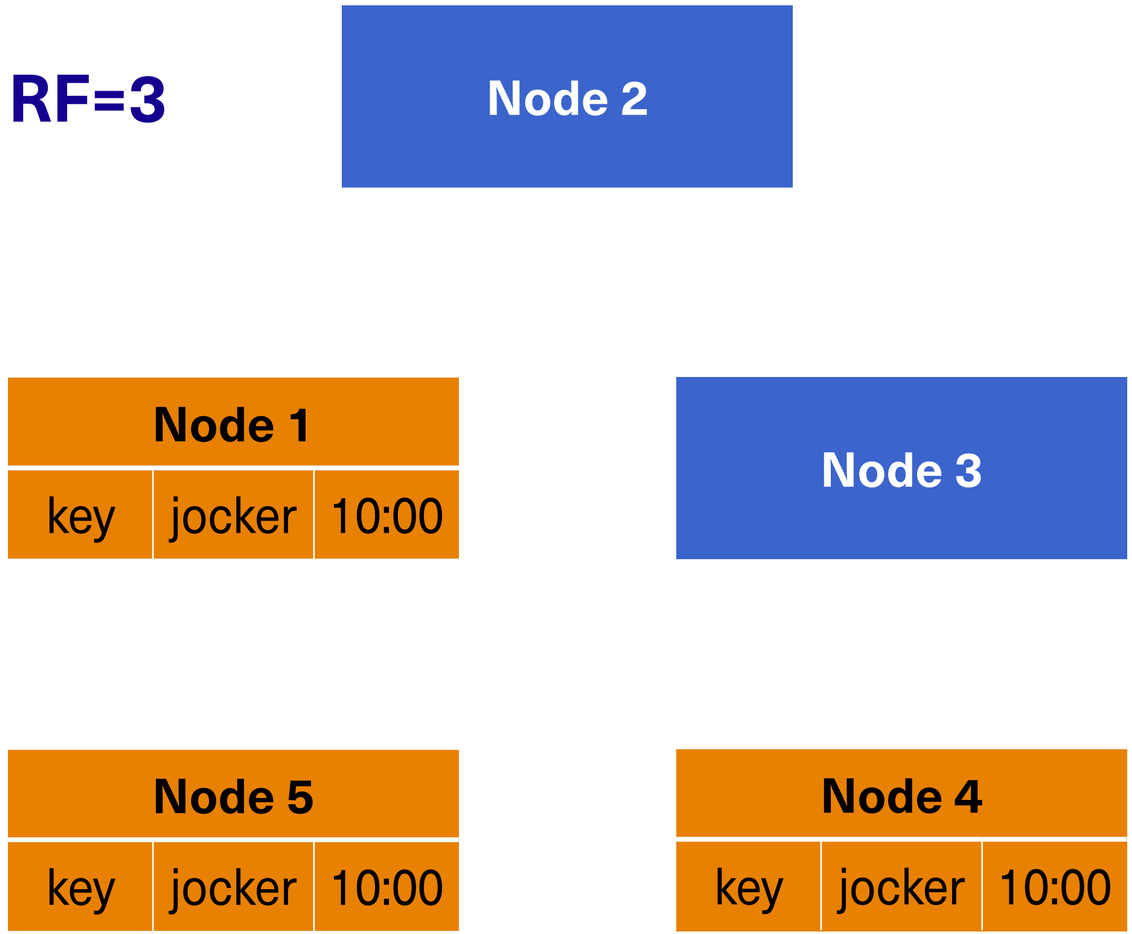
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

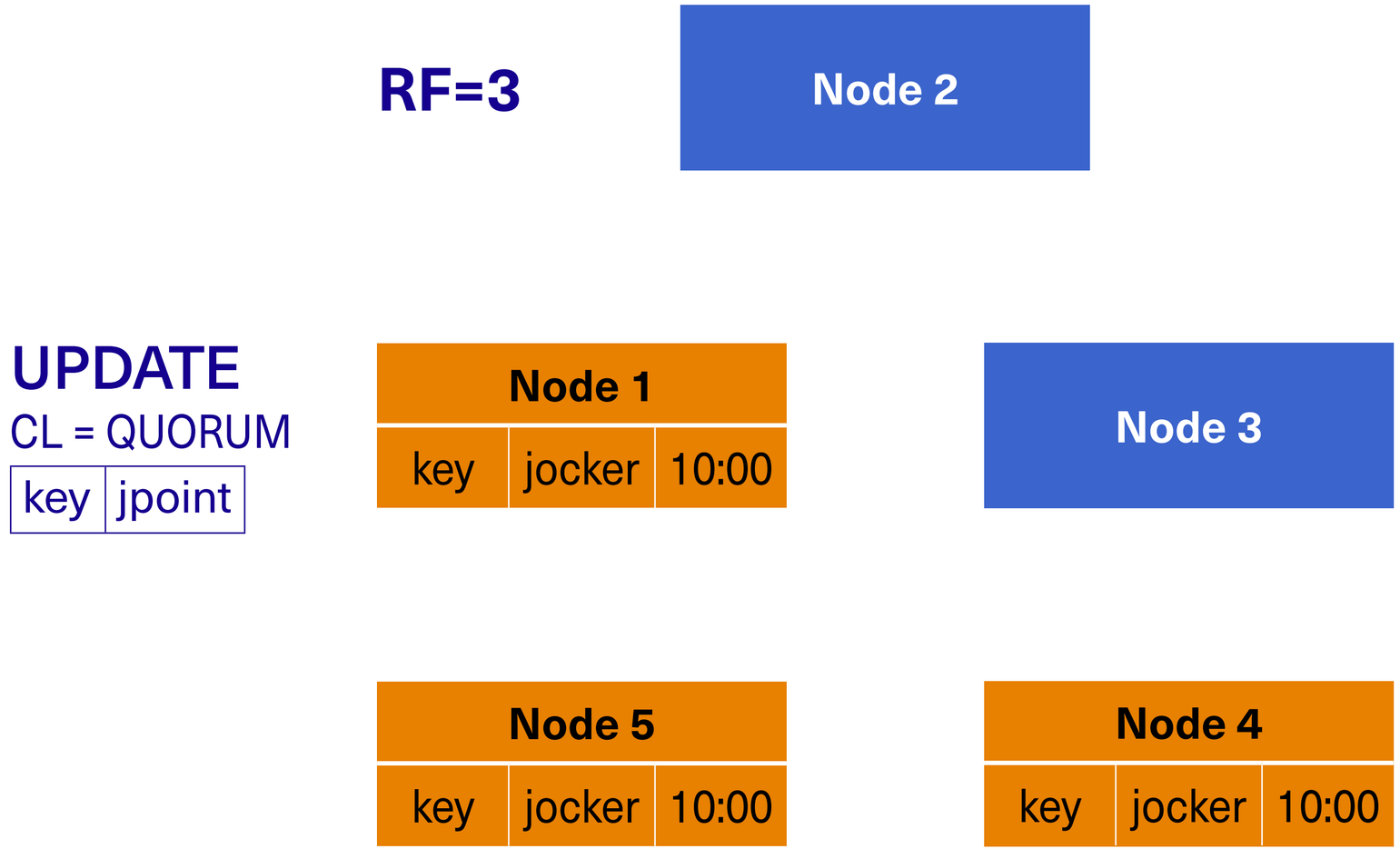
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

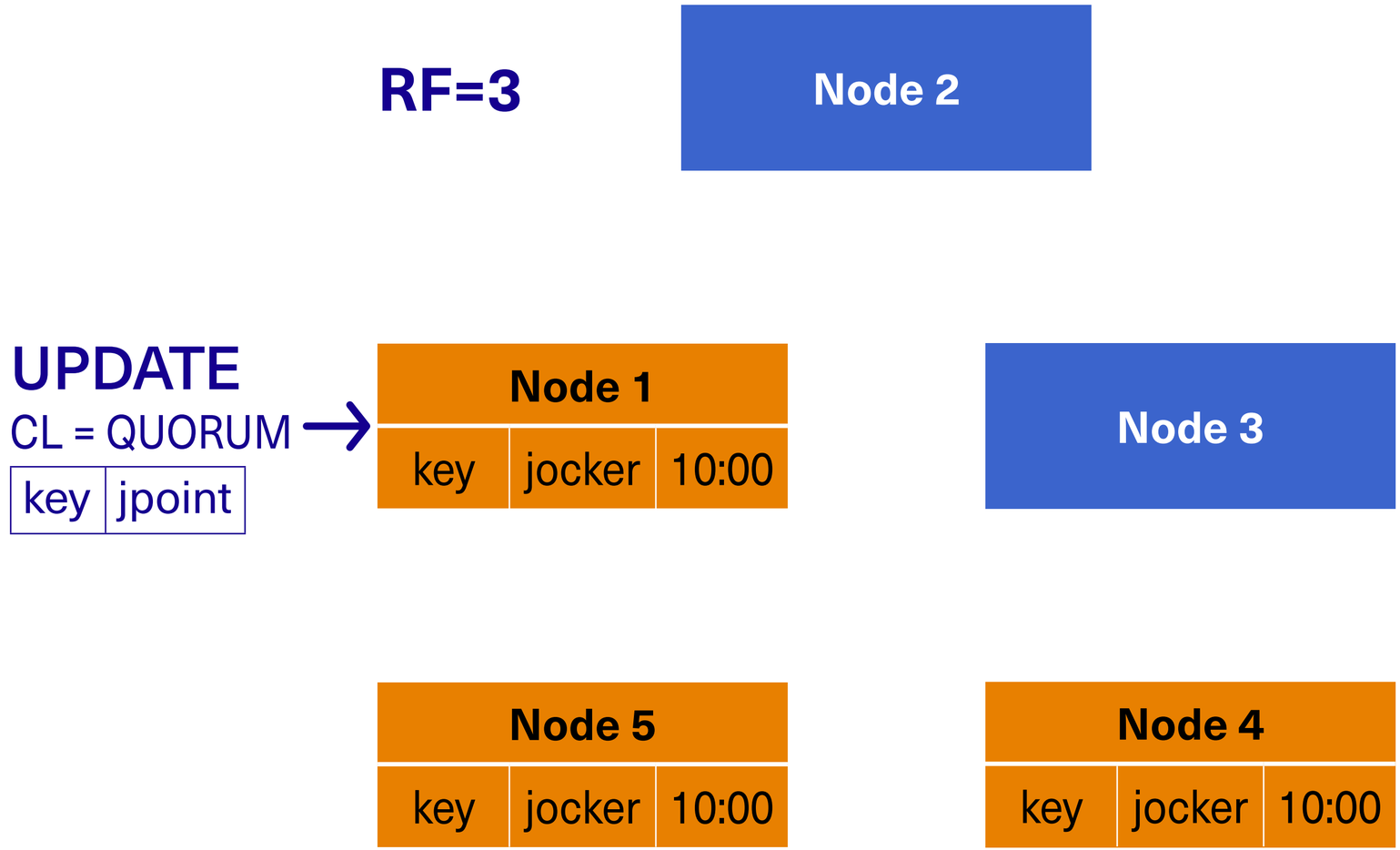
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

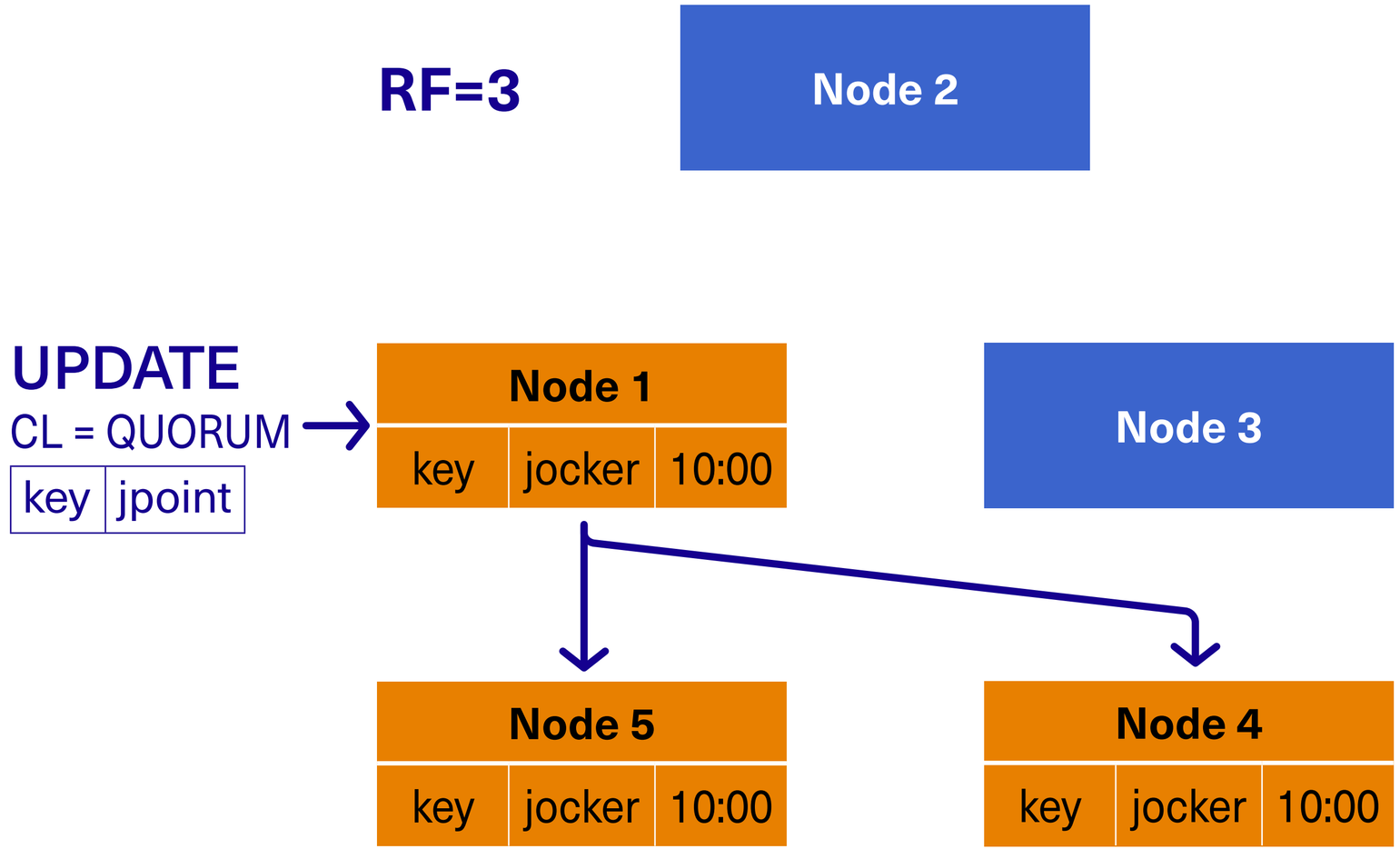
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

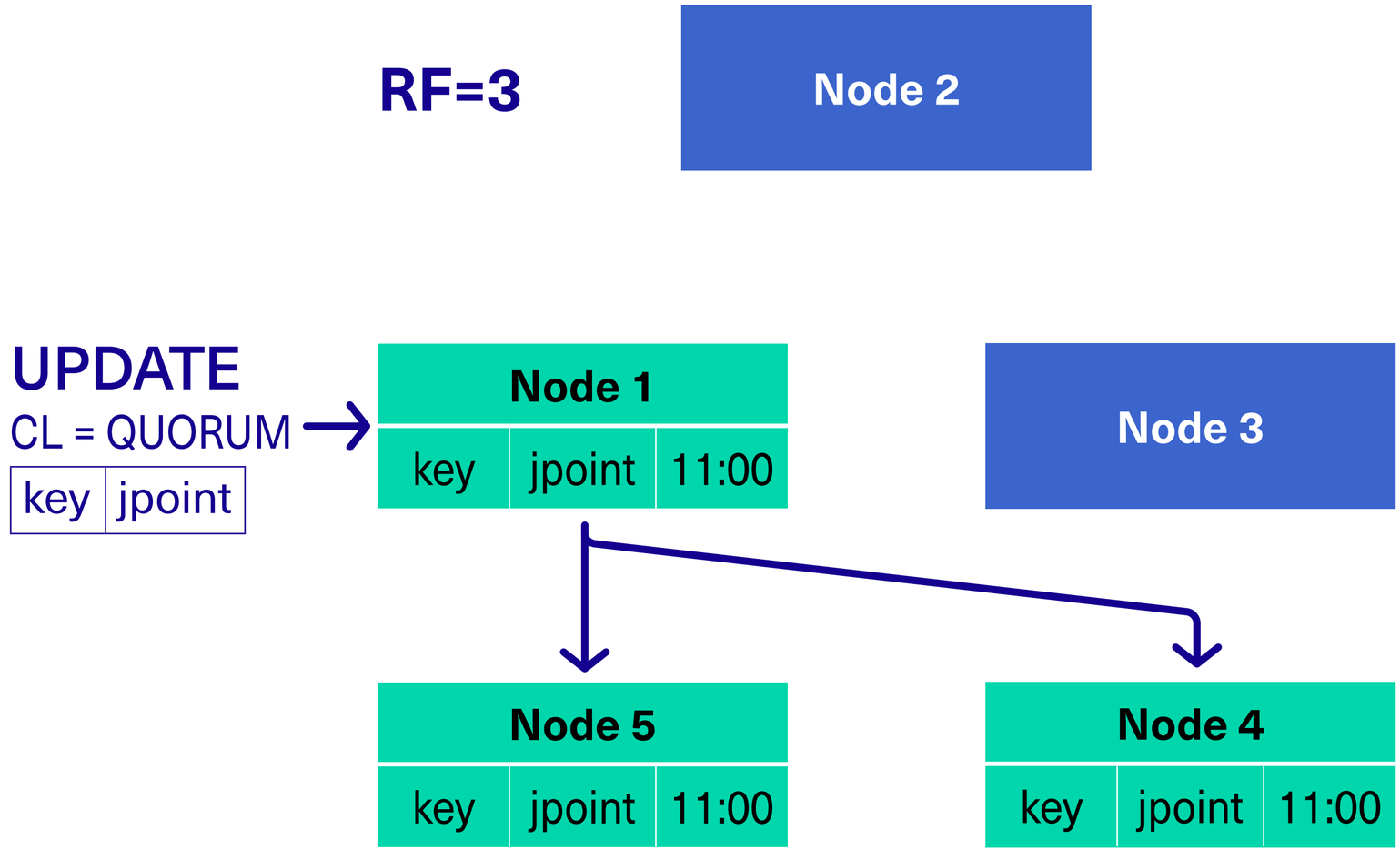
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

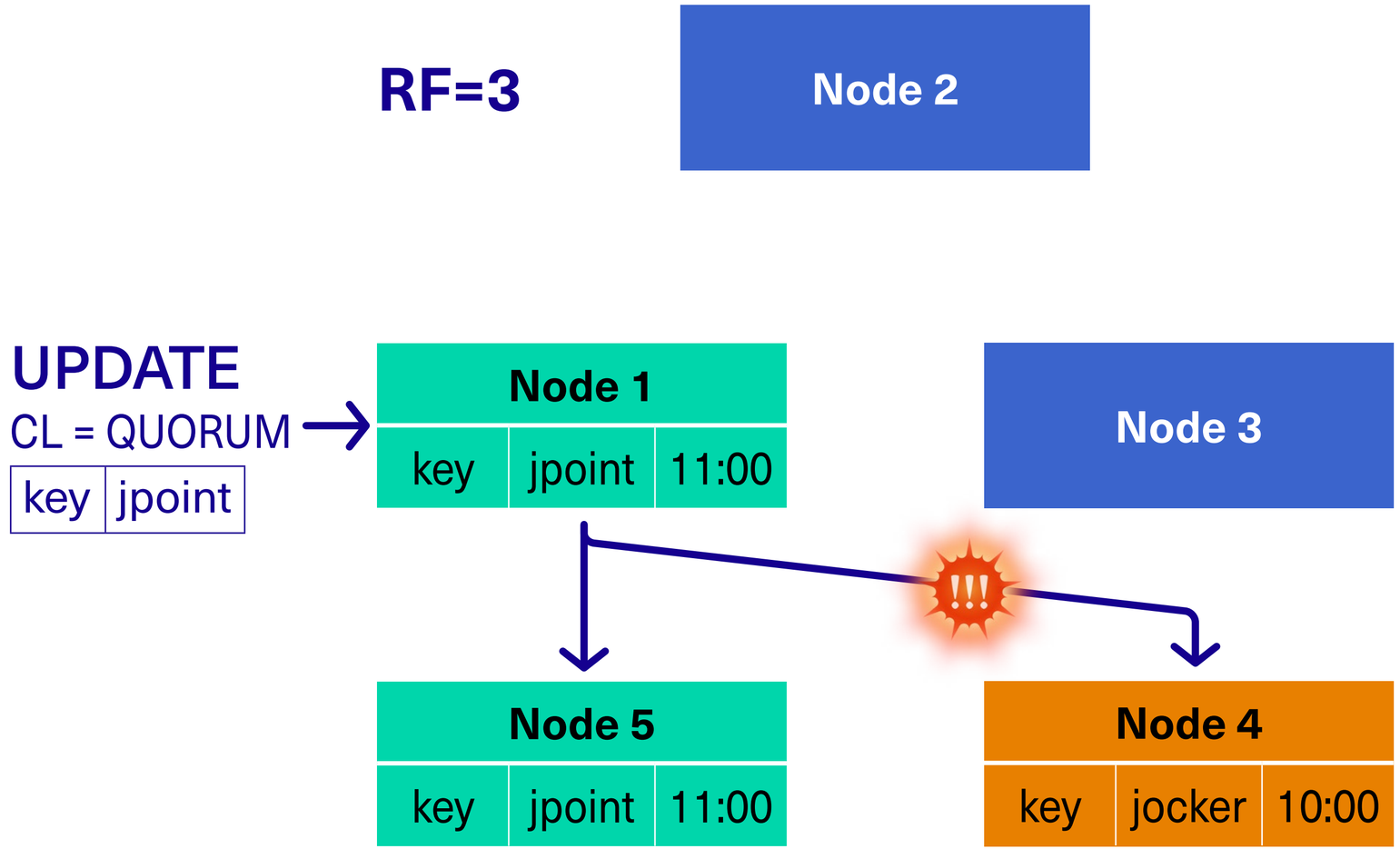
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

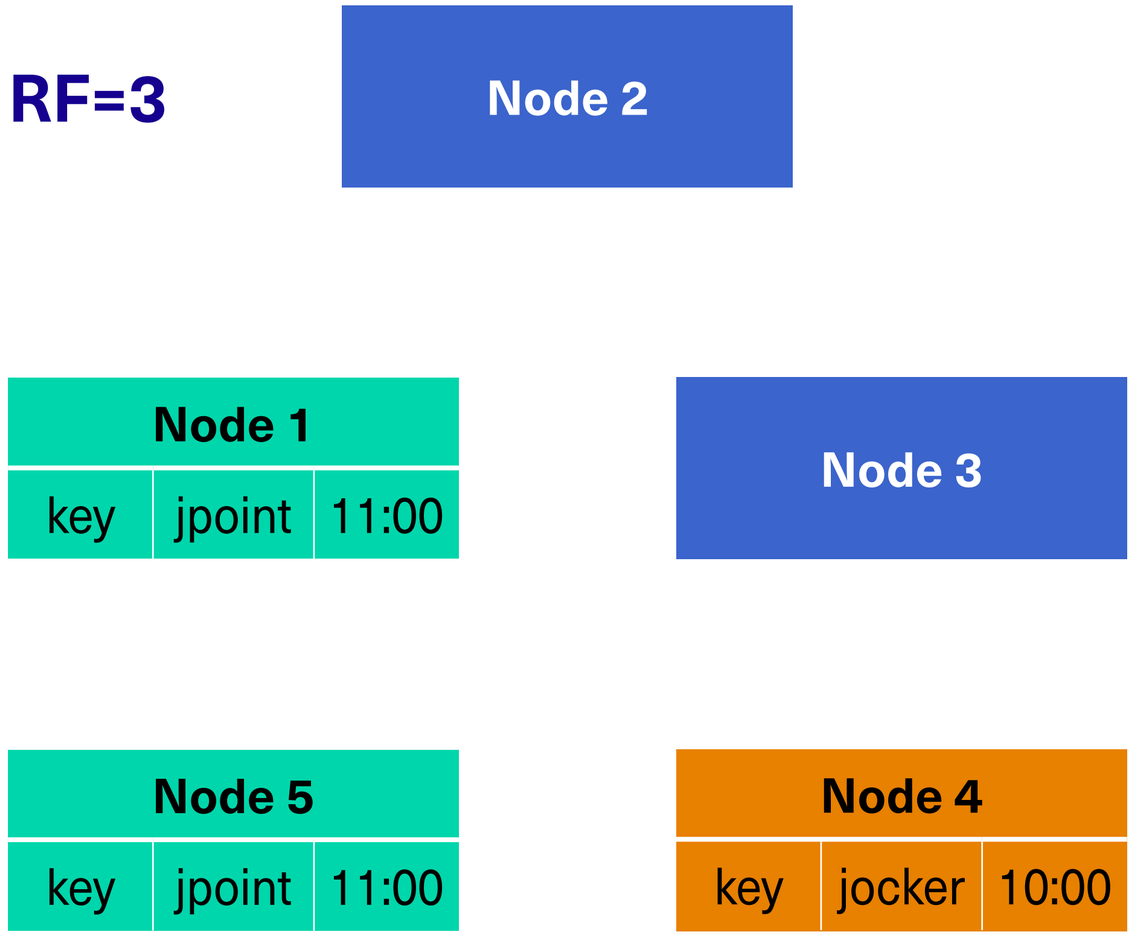
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

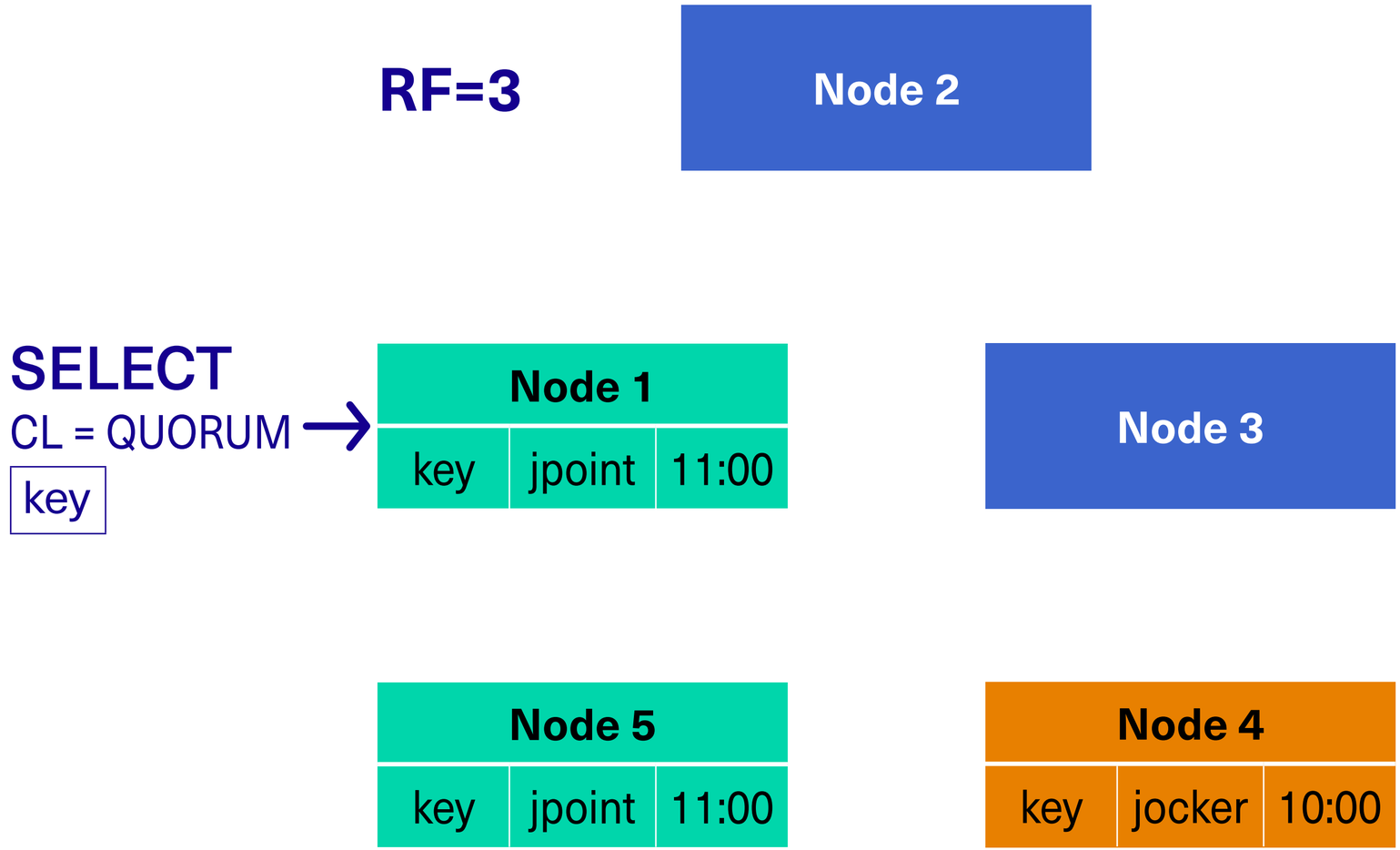
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

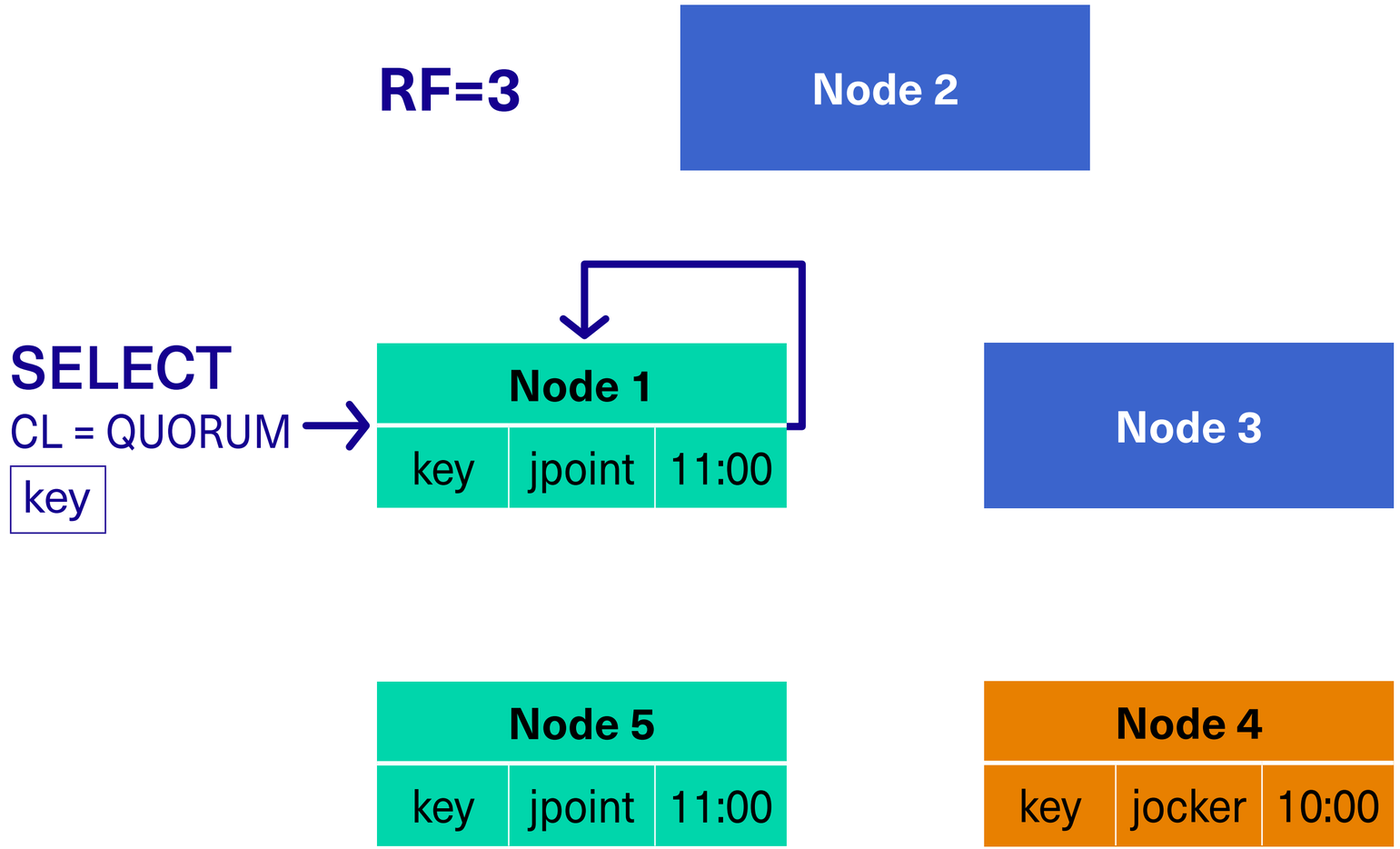
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

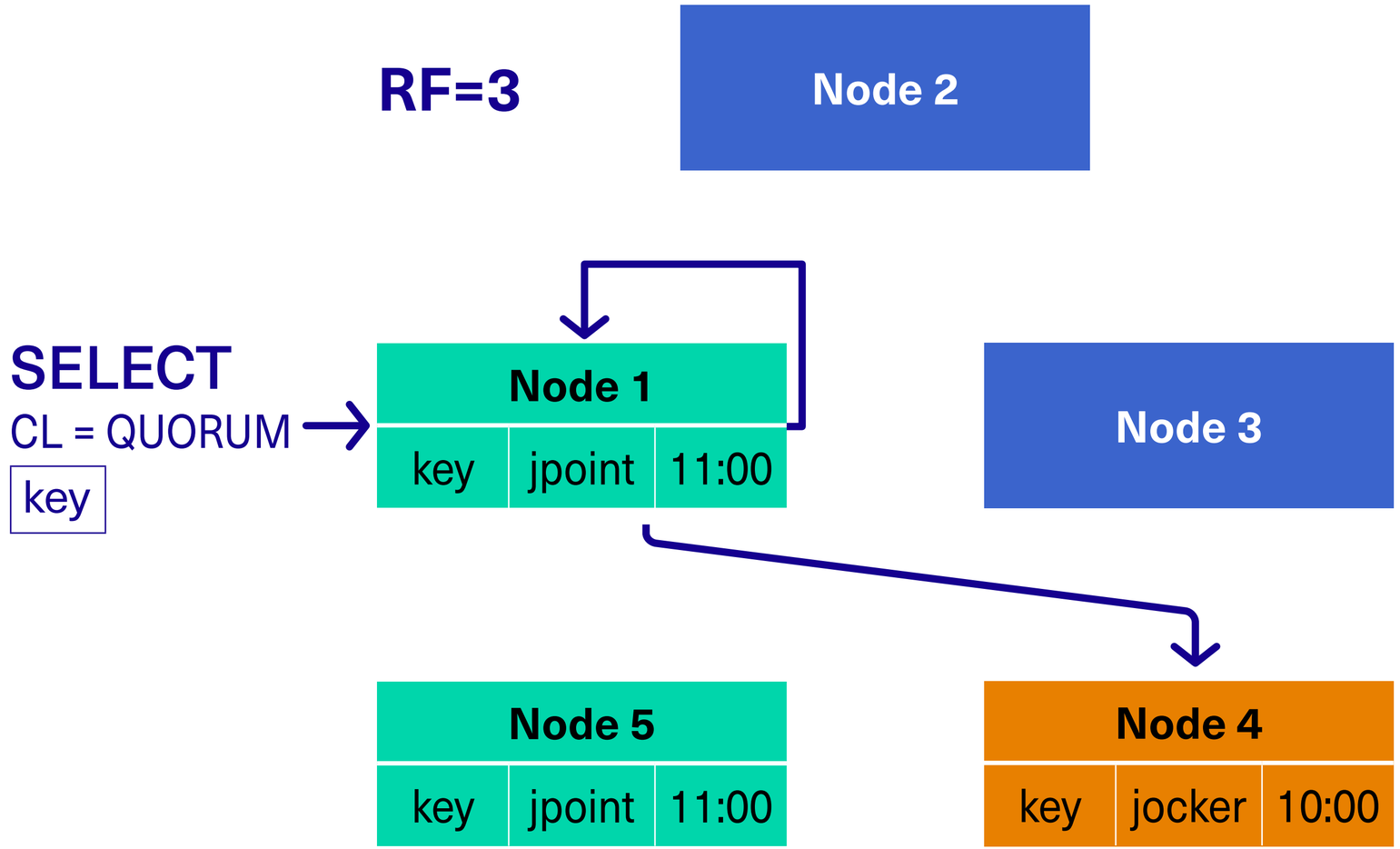
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

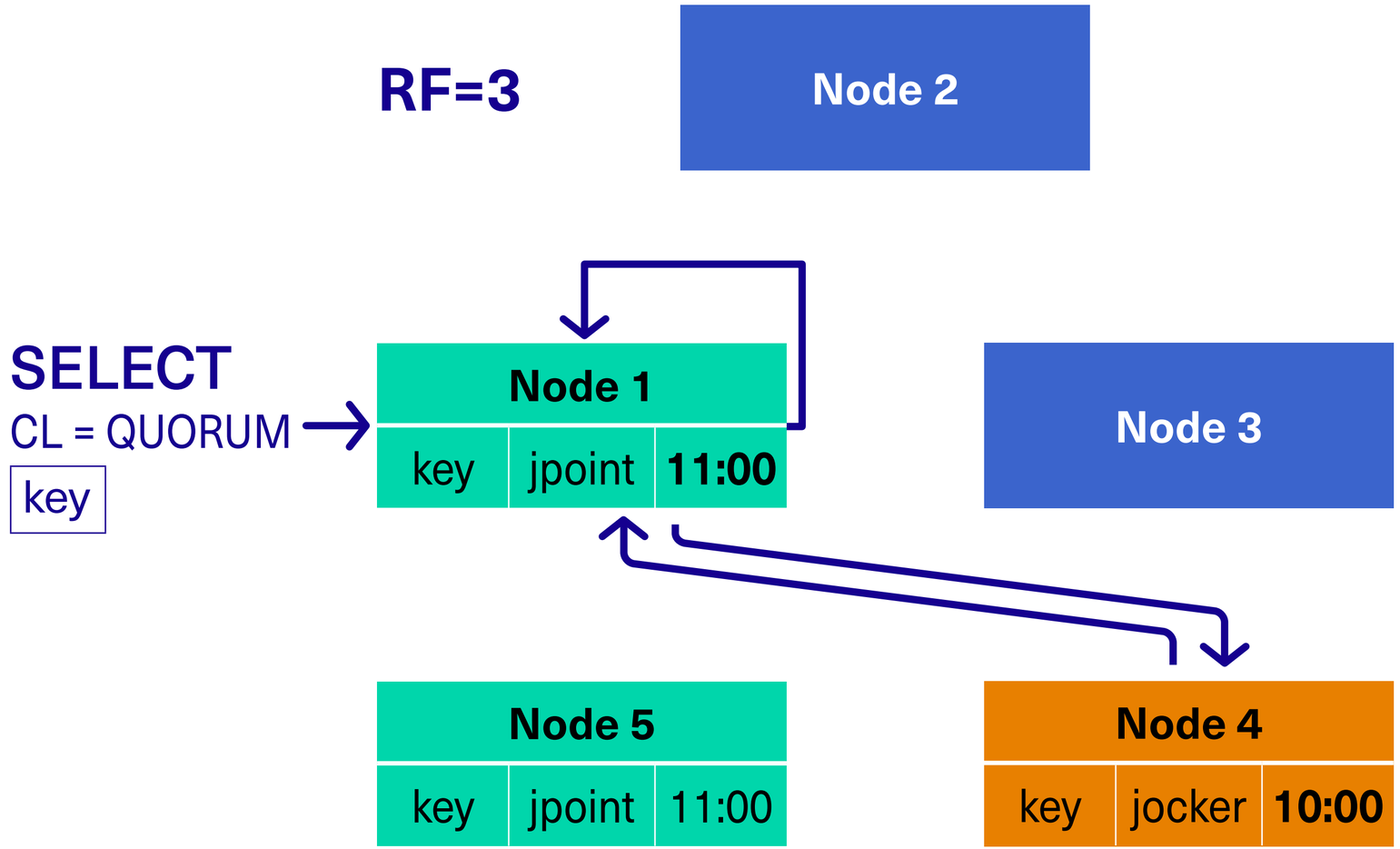
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

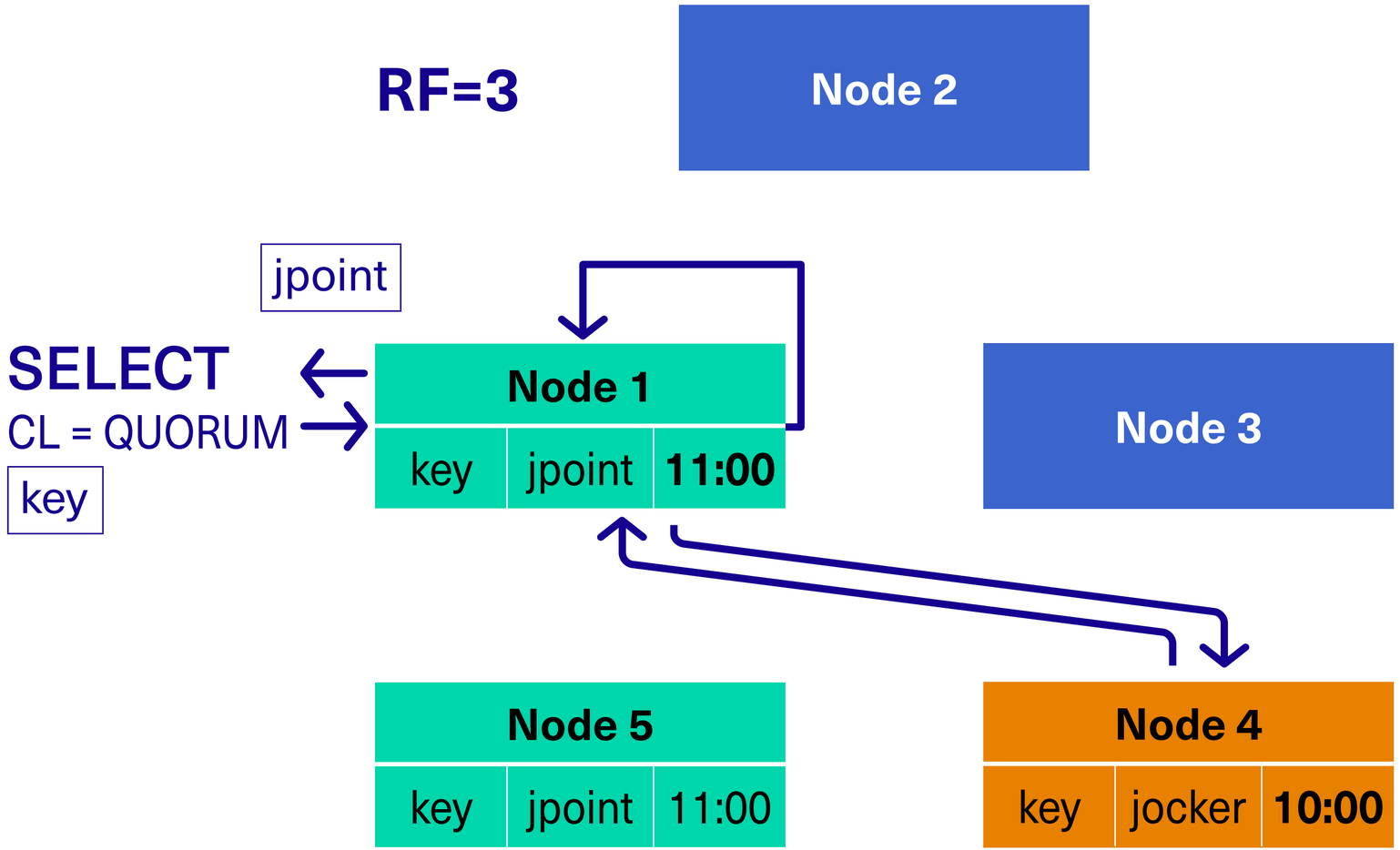
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

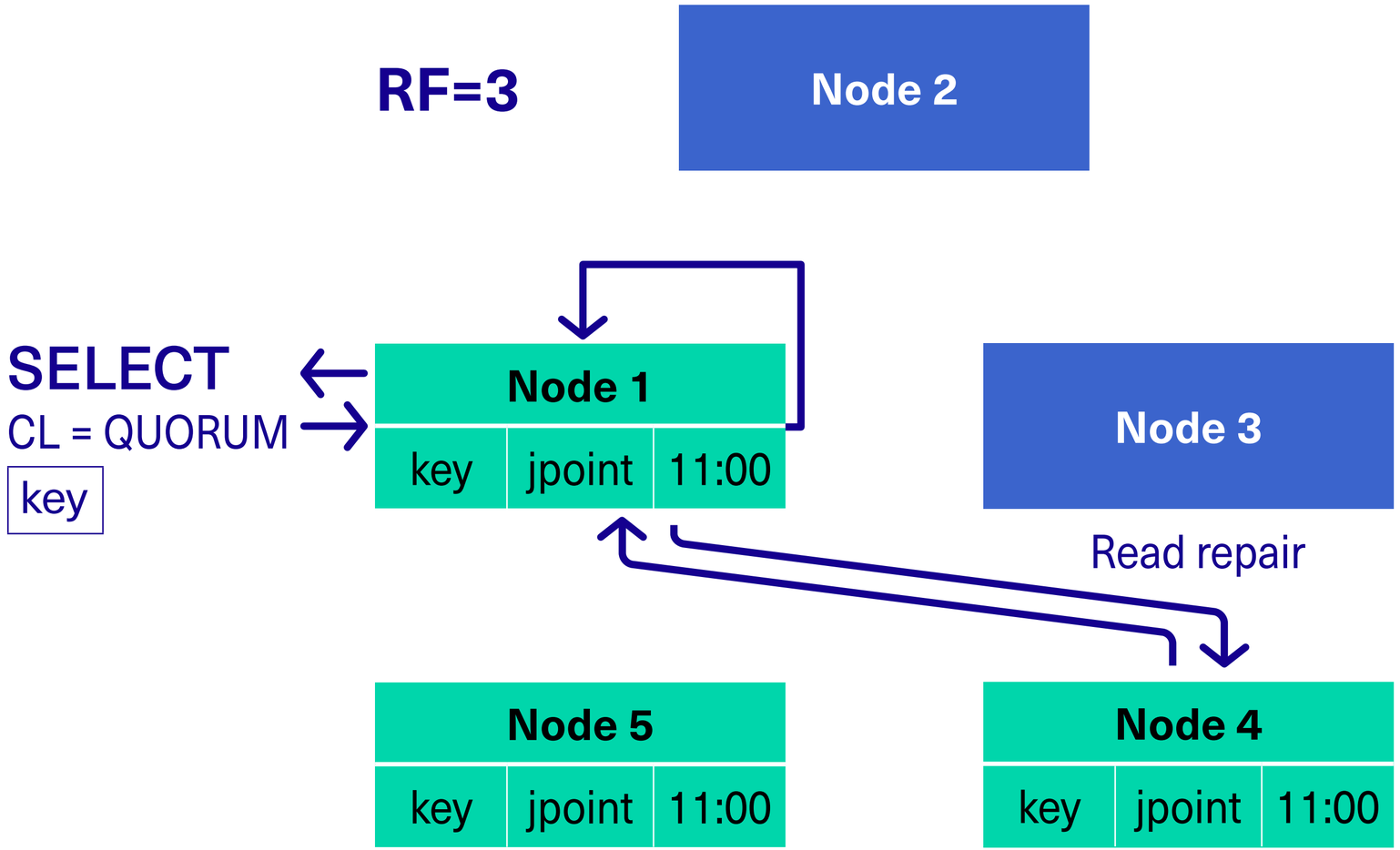
Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»



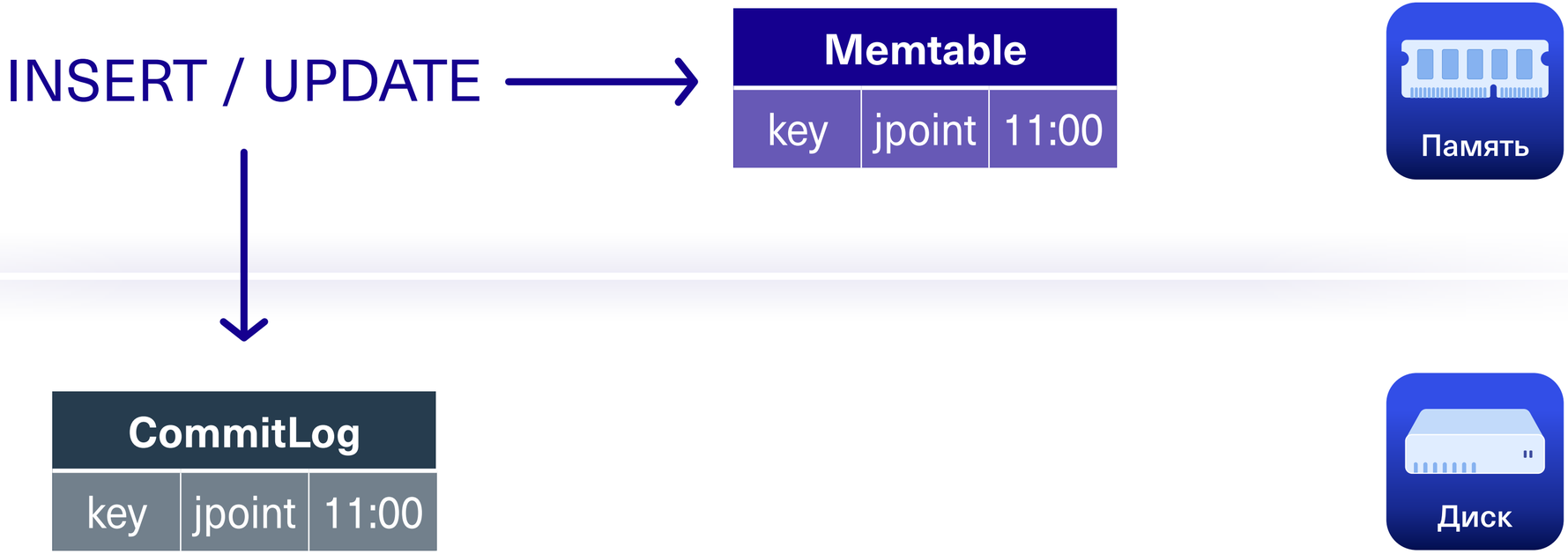
Настраиваемая согласованность

Replication factor (RF) — сколько узлов хранят копии одной партиции

Consistency level (CL) — сколько из этих узлов должны подтвердить операцию, чтобы вернуть клиенту «успех»

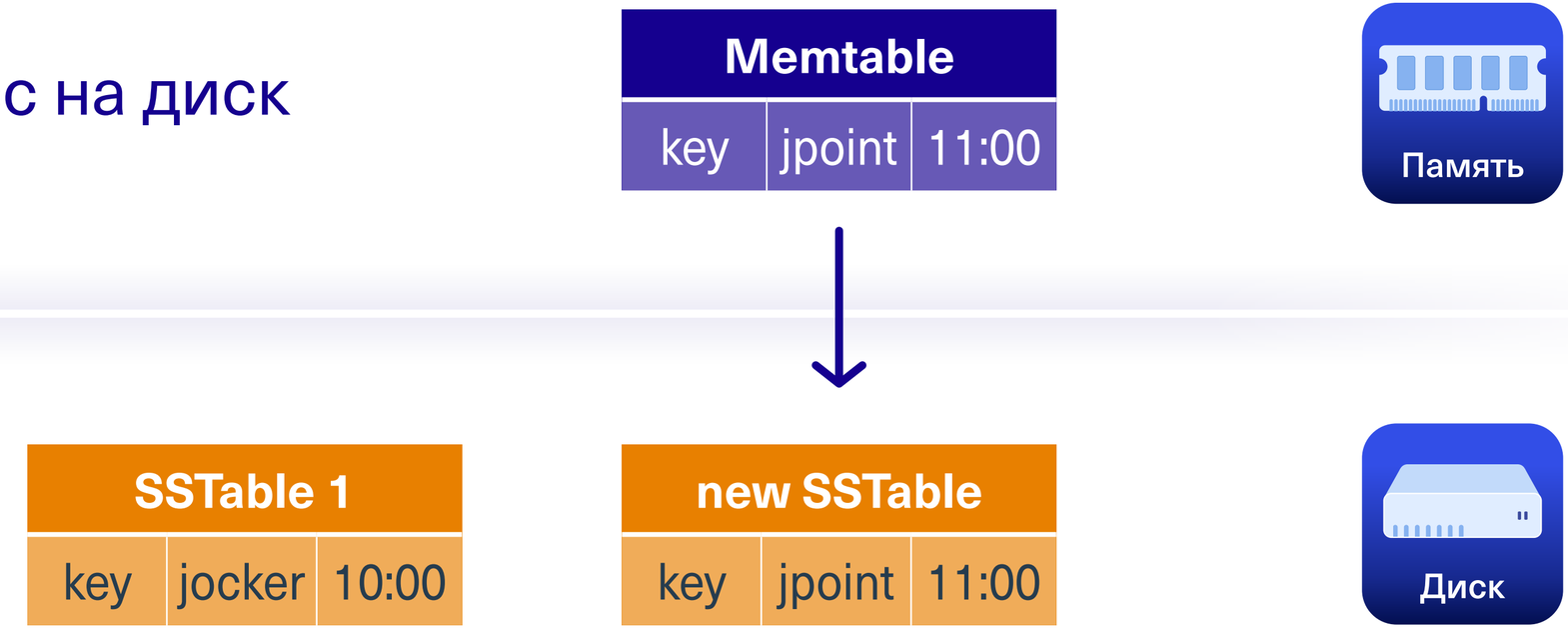


Запись

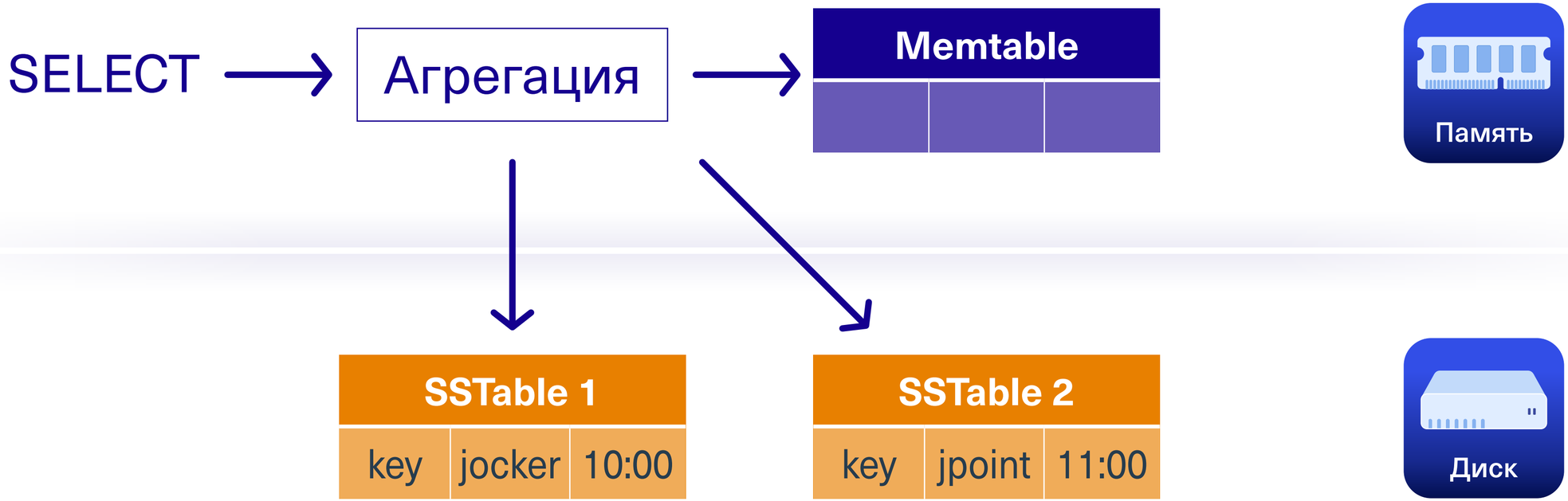

INSERT / UPDATE
Memtable
key
jpoint
11:00
Память
CommitLog
key
jpoint
11:00
Диск

Запись

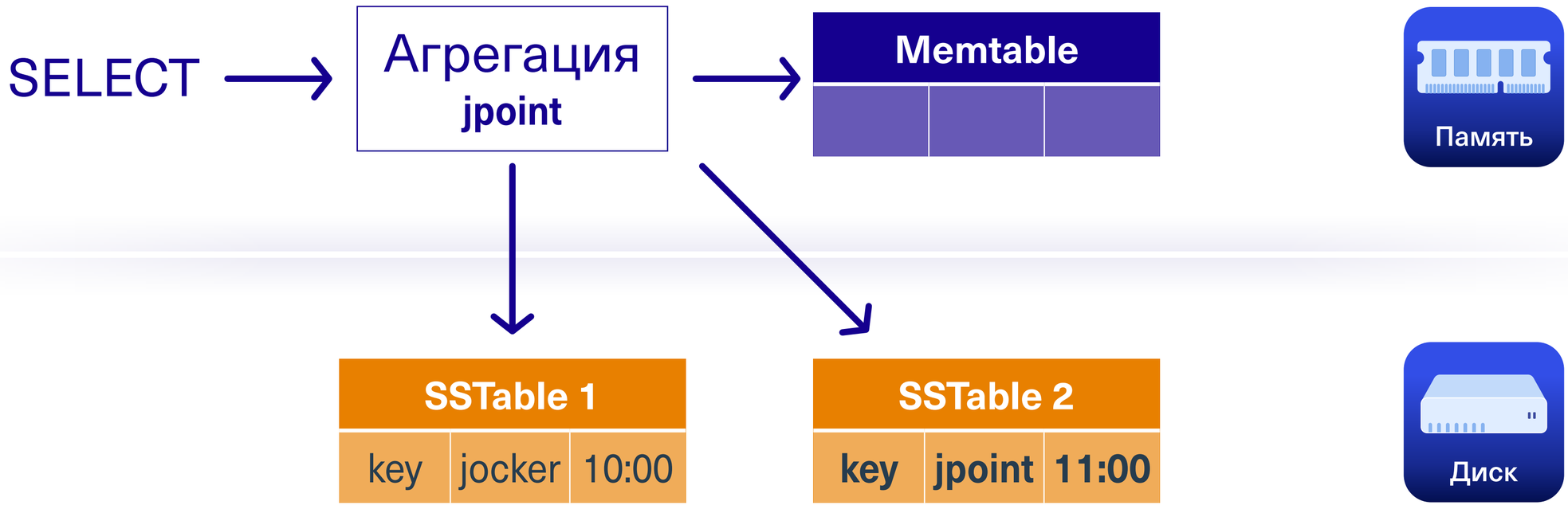
Сброс на диск



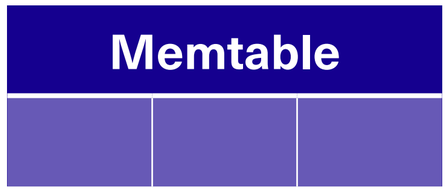
Чтение


SELECT
Агрегация
Memtable
Память
SSTable 1
key
jocker
10:00
SSTable 2
key
jpoint
11:00
Диск

Чтение

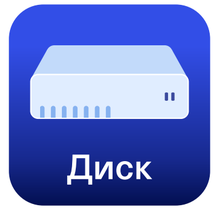

SELECT
Агрегация
jpoint
Memtable
Память
SSTable 1
key
jocker
10:00
SSTable 2
key
jpoint
11:00
Диск

Compaction

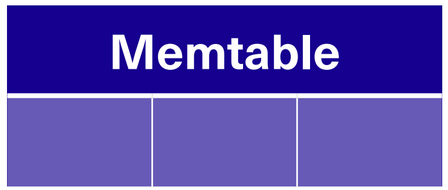

Memtable

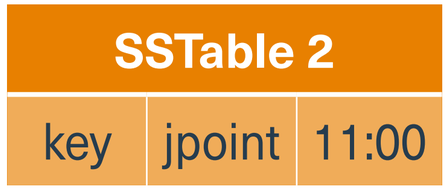

Память

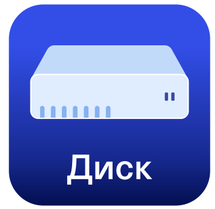

SSTable 1
key
jocker
10:00
SSTable 2
key
jpoint
11:00
new SSTable


Диск

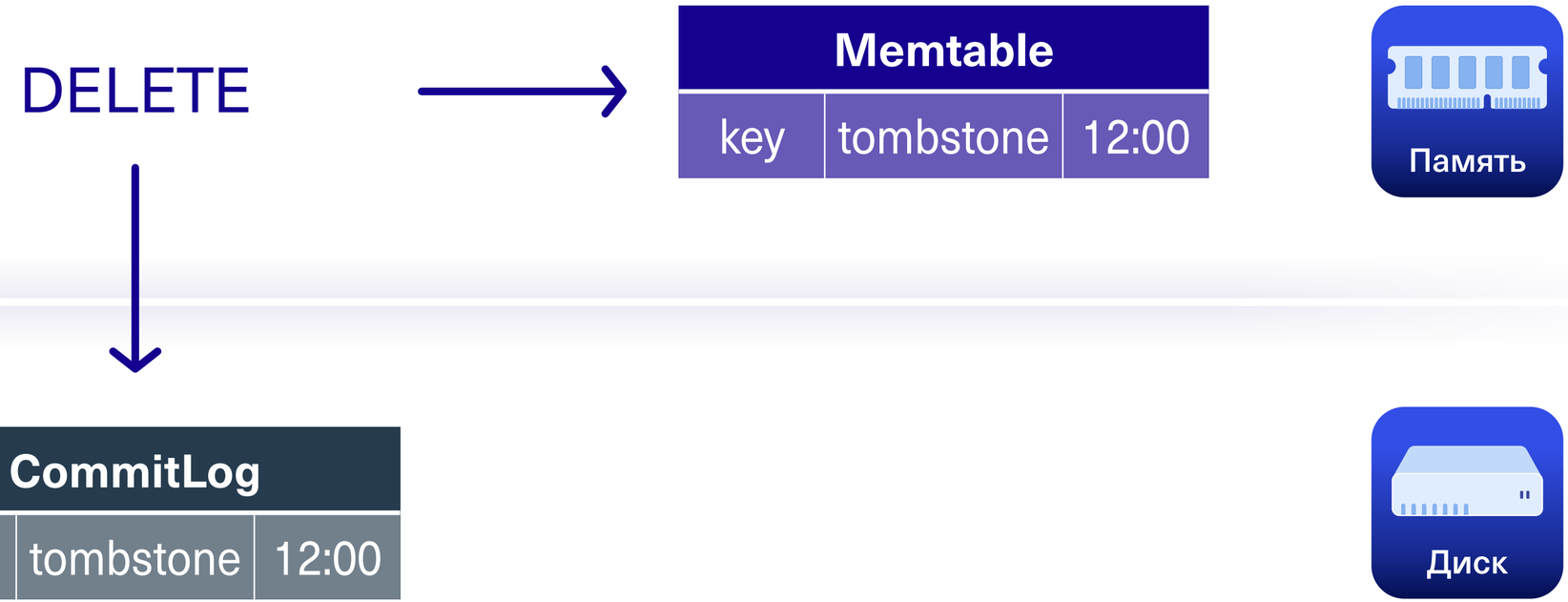
Compaction





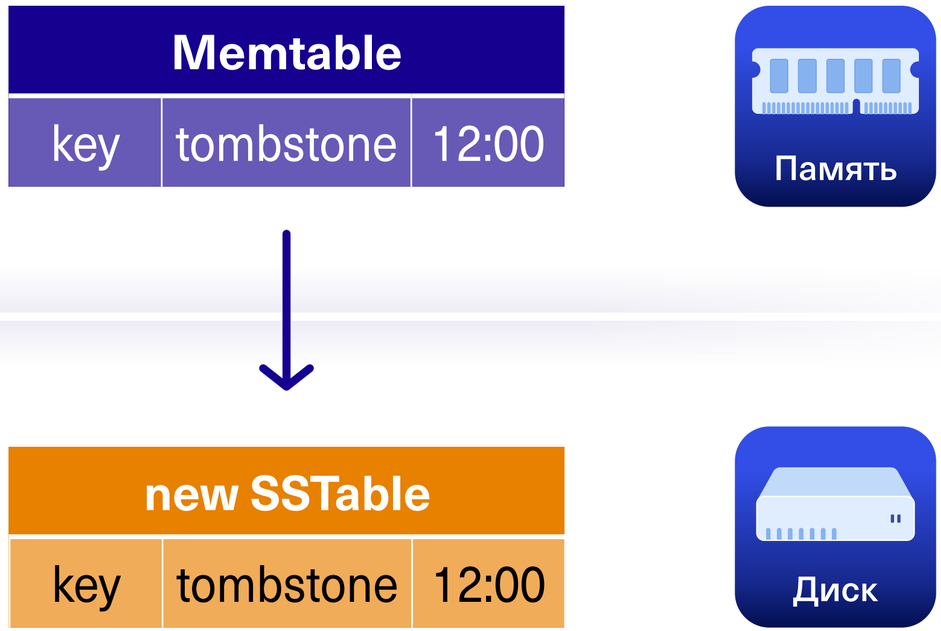


Удаление

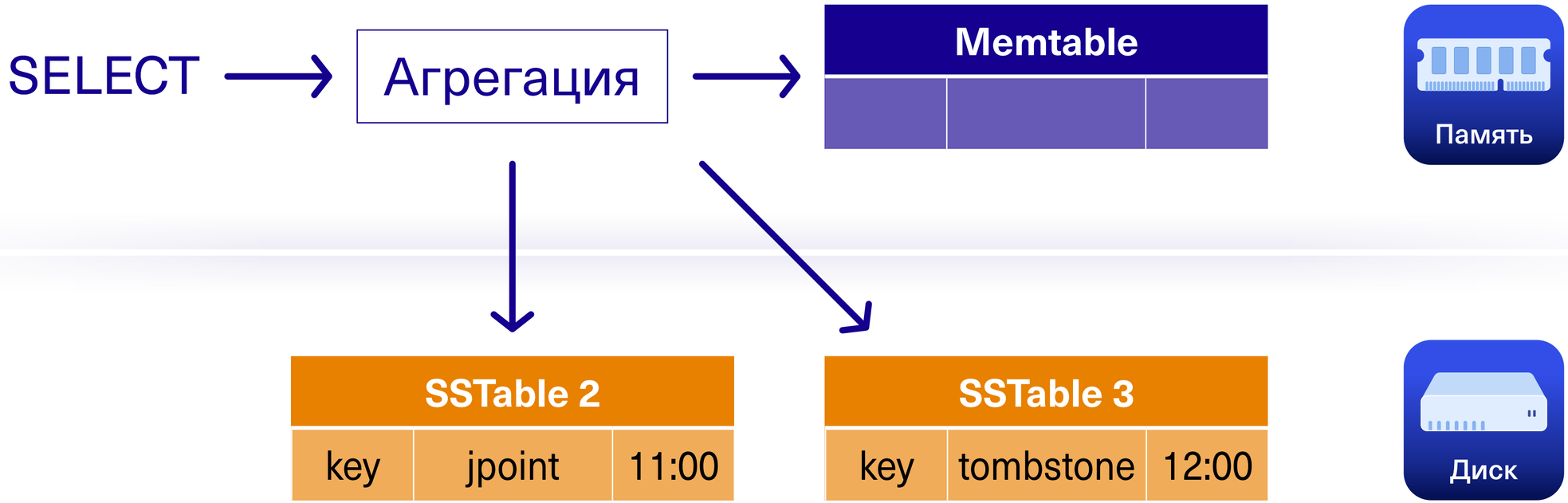

DELETE
Memtable
key
tombstone
12:00
Память
CommitLog
key
tombstone
12:00
Диск

Удаление

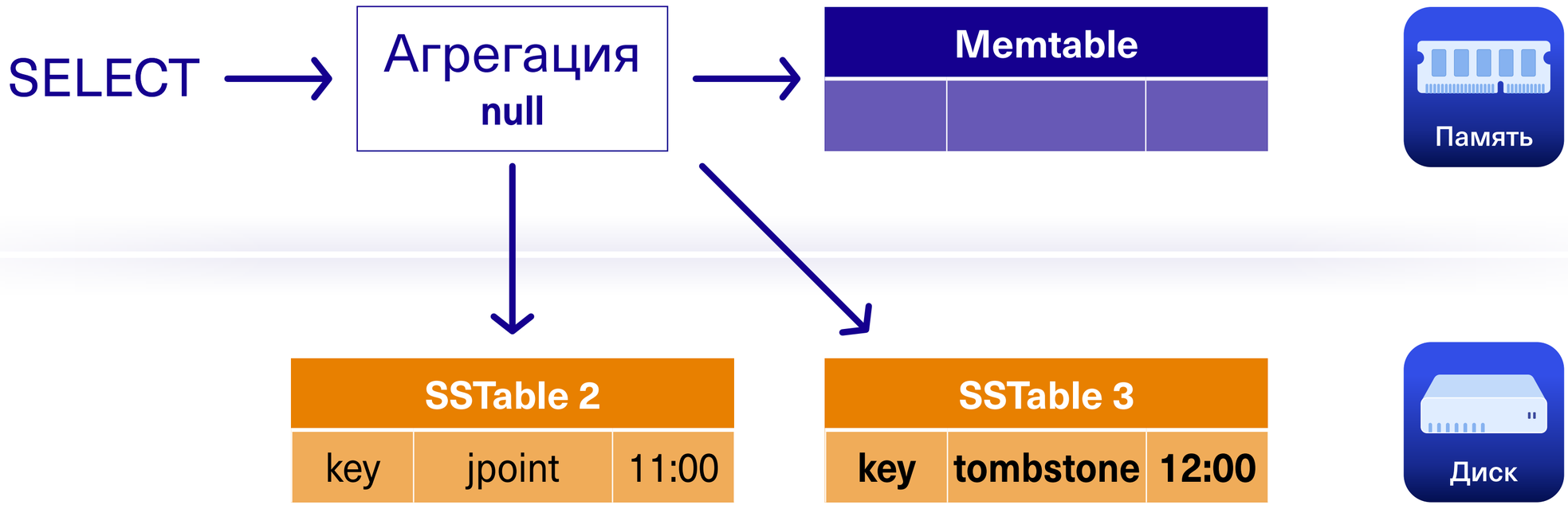
Сброс на диск



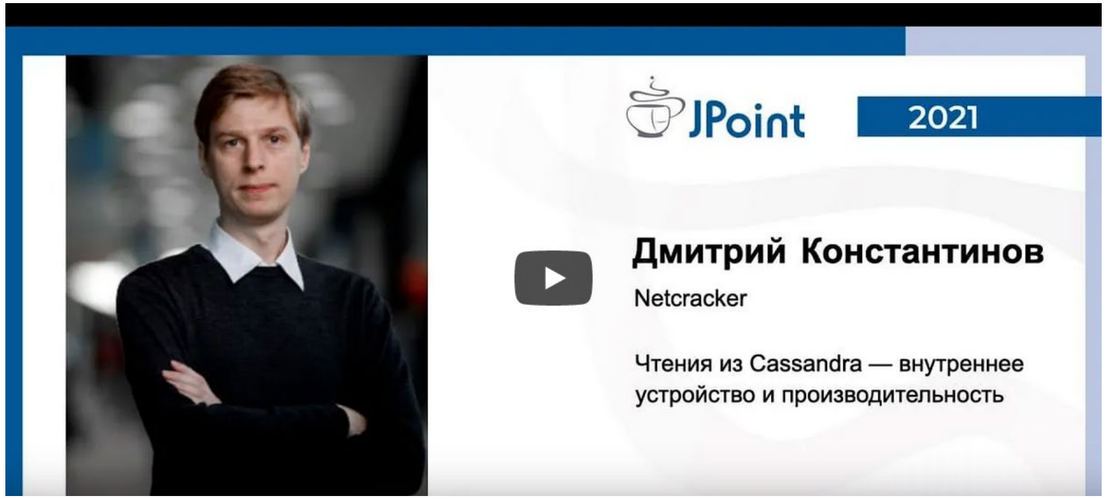
Чтение удалённого значения



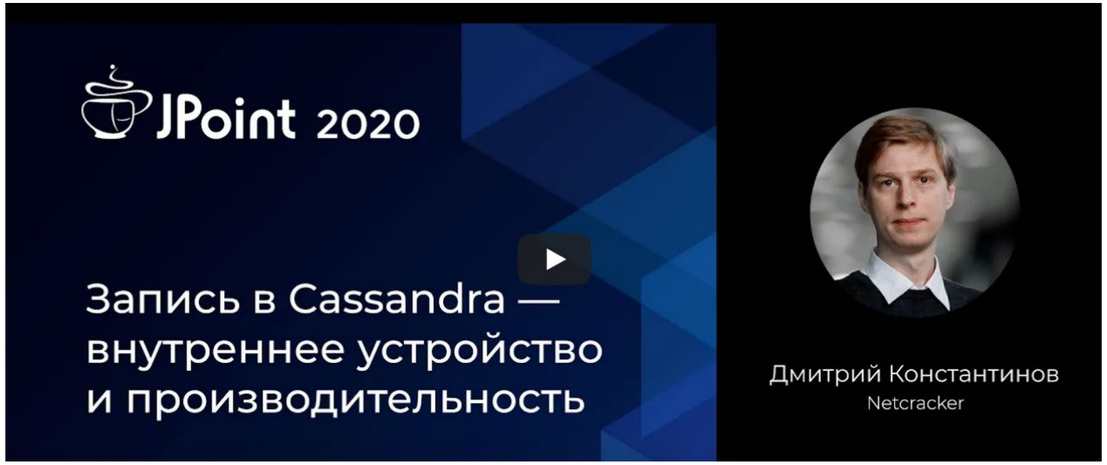
Чтение удалённого значения



Чтение и запись подробнее

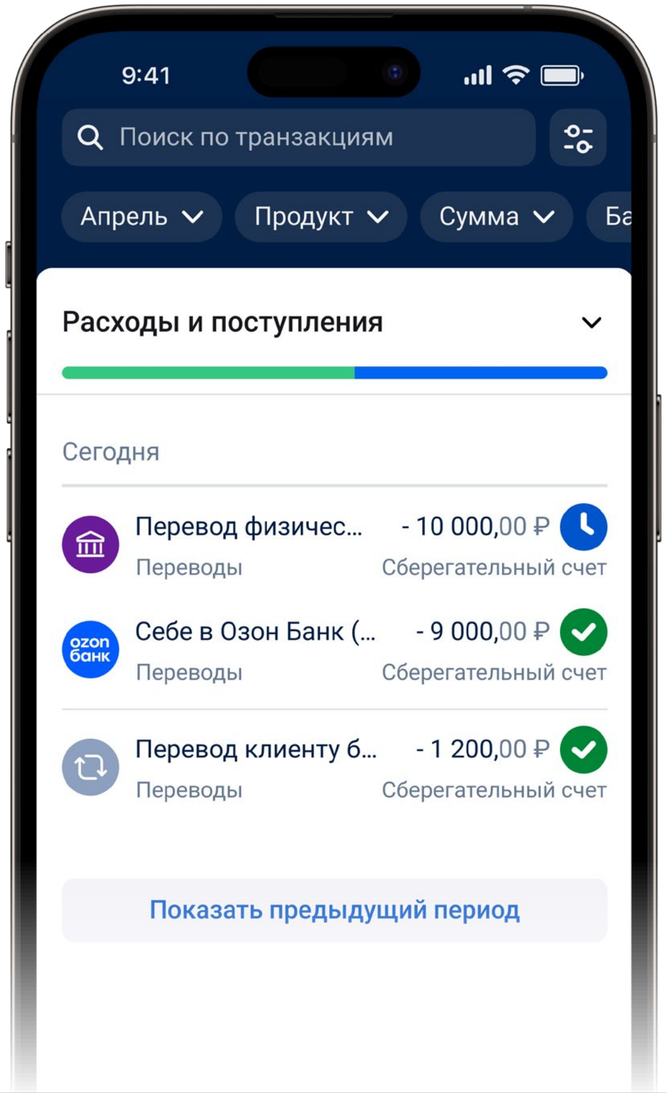


[«Чтение из Cassandra — внутреннее устройство и производительность» Jpoint 2021]

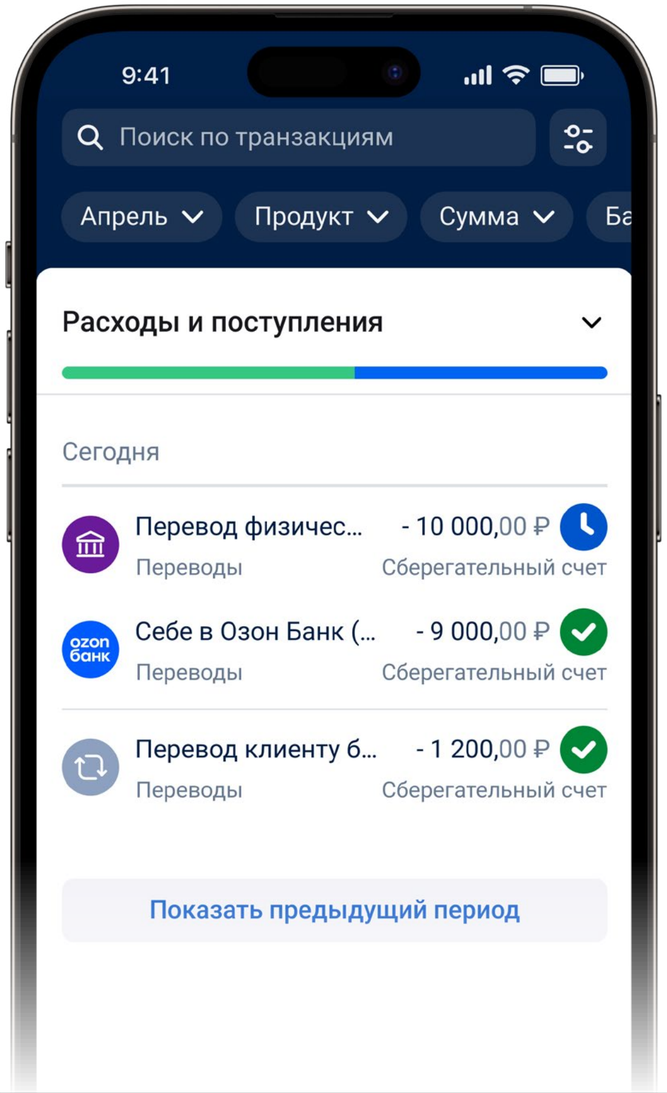


[«Запись в Cassandra — внутреннее устройство и производительность» Jpoint 2020]

Пишем ленту операций

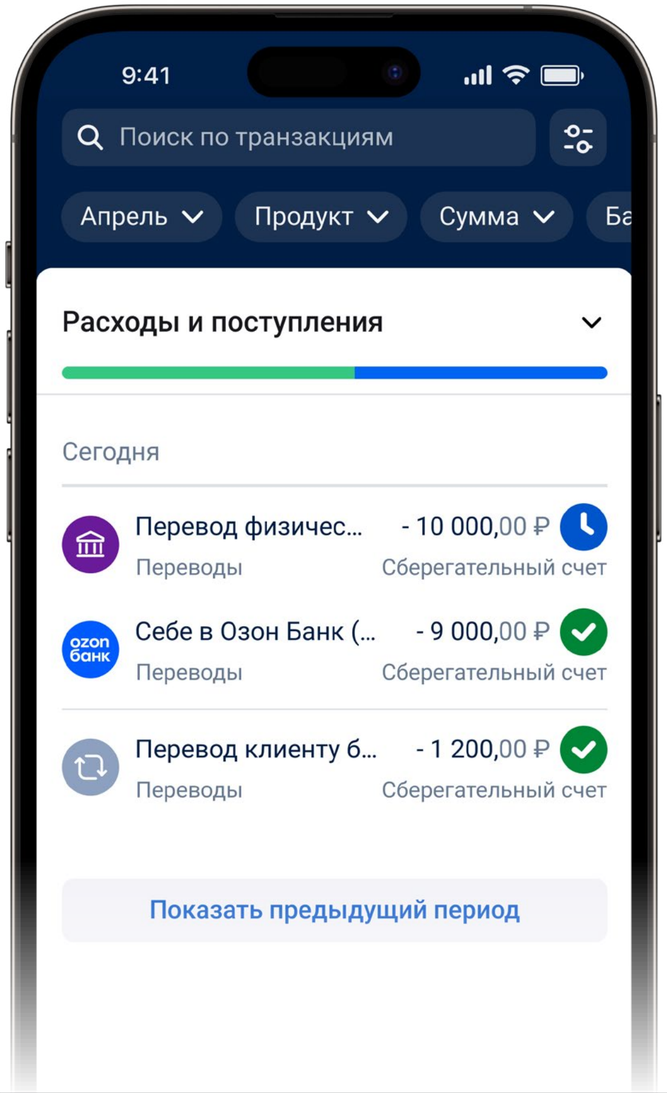


Пишем ленту операций

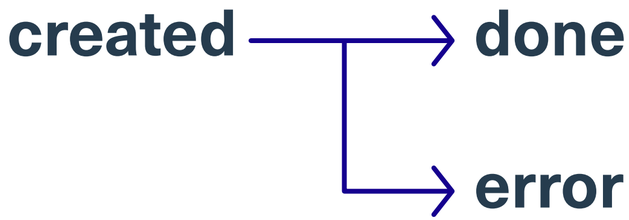


- Много записей, удалений нет, поиск ID клиента и дате

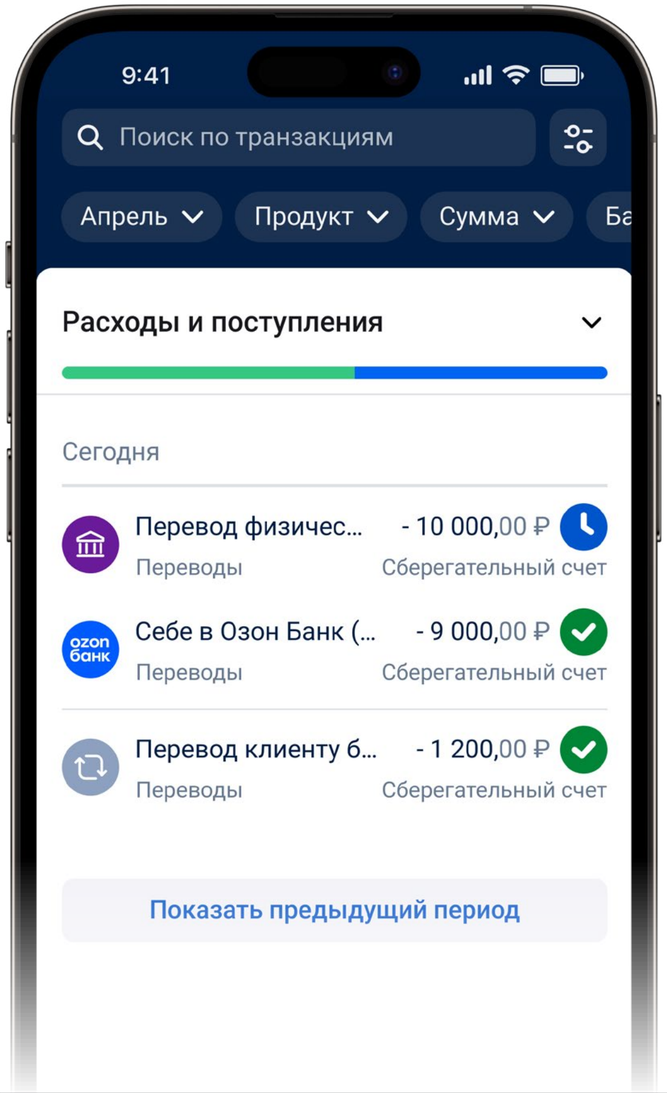
Пишем ленту операций



- Много записей, удалений нет, поиск ID клиента и дате
- Статус операции может меняться



Пишем ленту операций

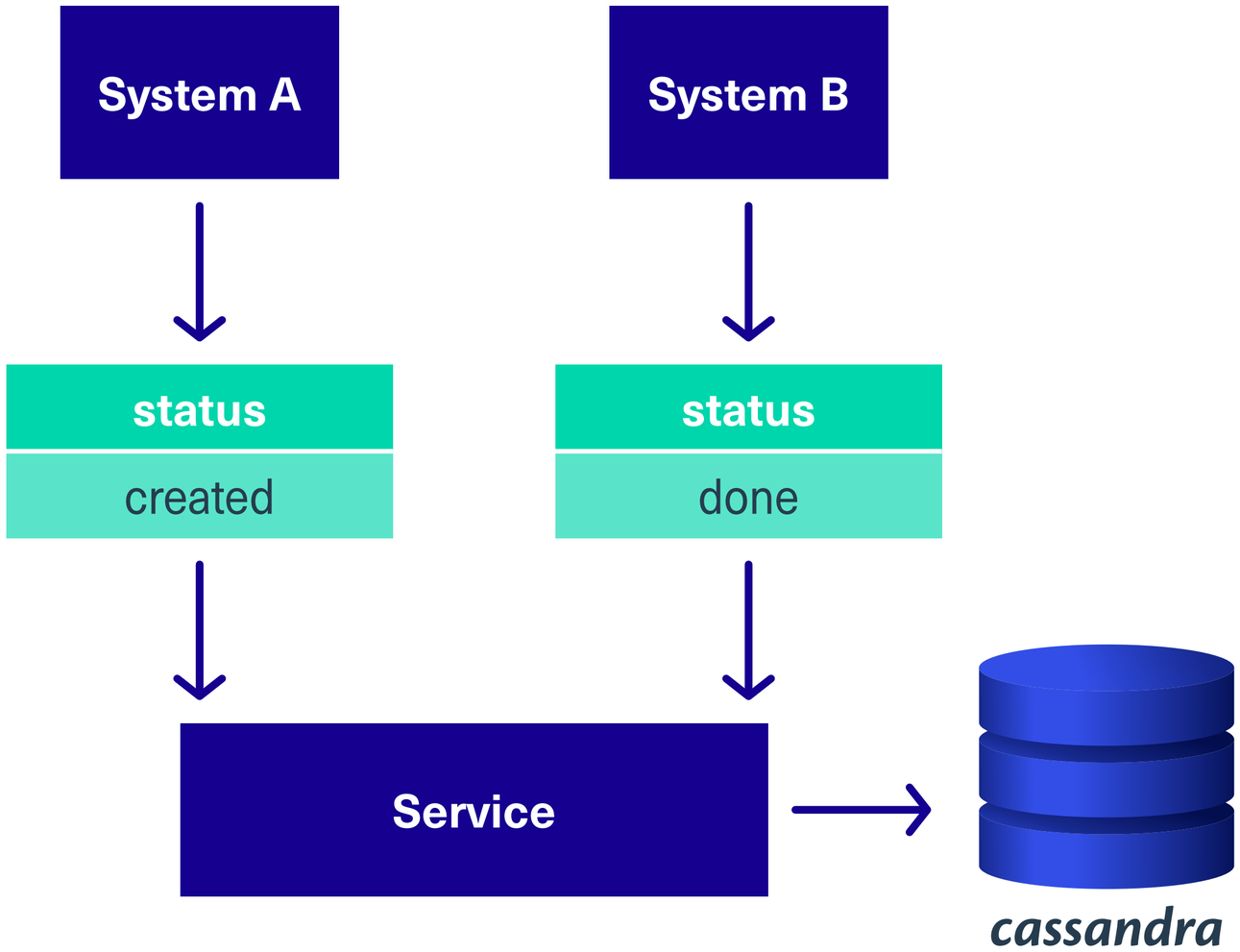


- Много записей, удалений нет, поиск ID клиента и дате
- Статус операции может меняться

[«С SQL на Cassandra», Jocker 2023]

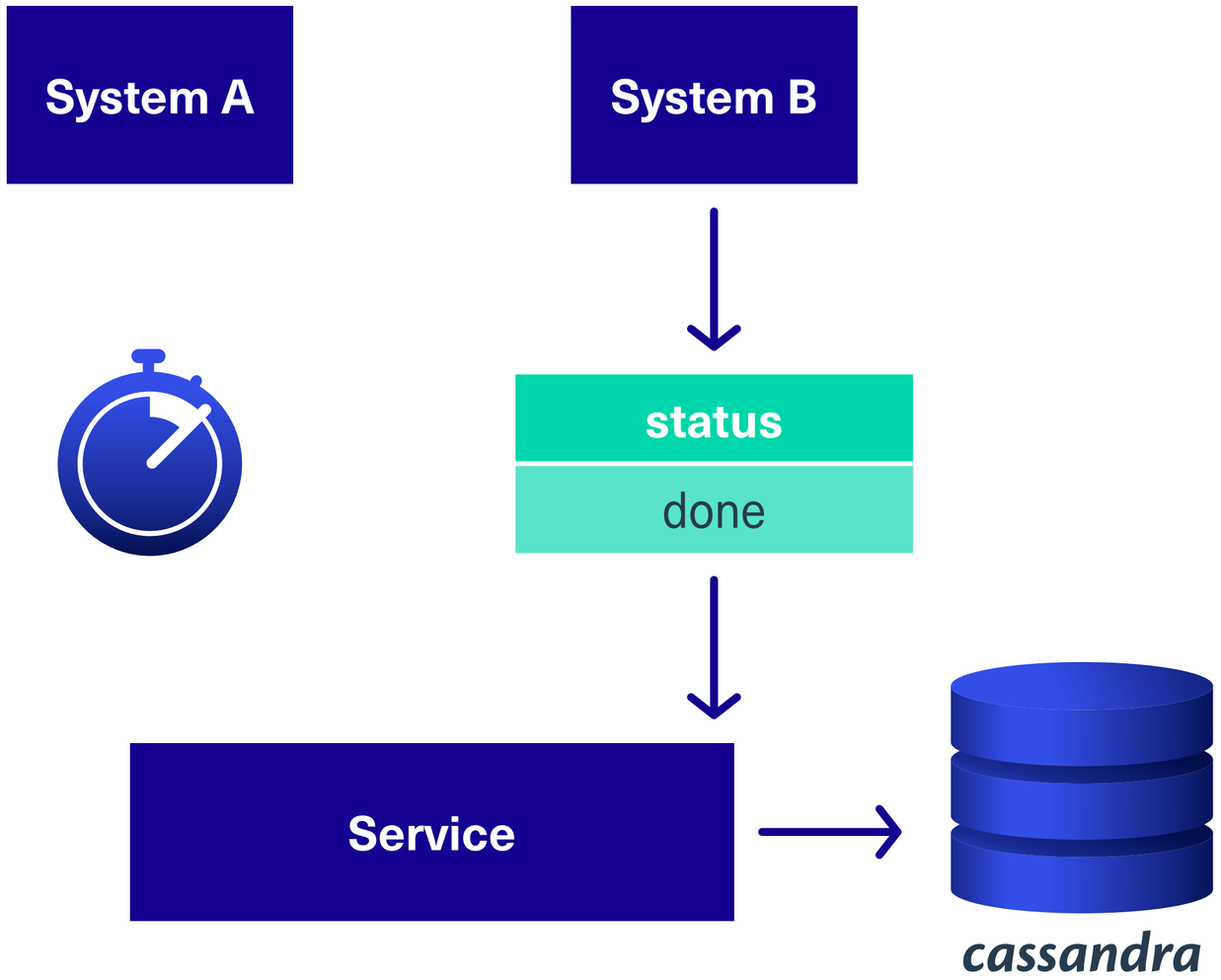
Таблицы необходимо проектировать от запросов!

Пишем ленту операций



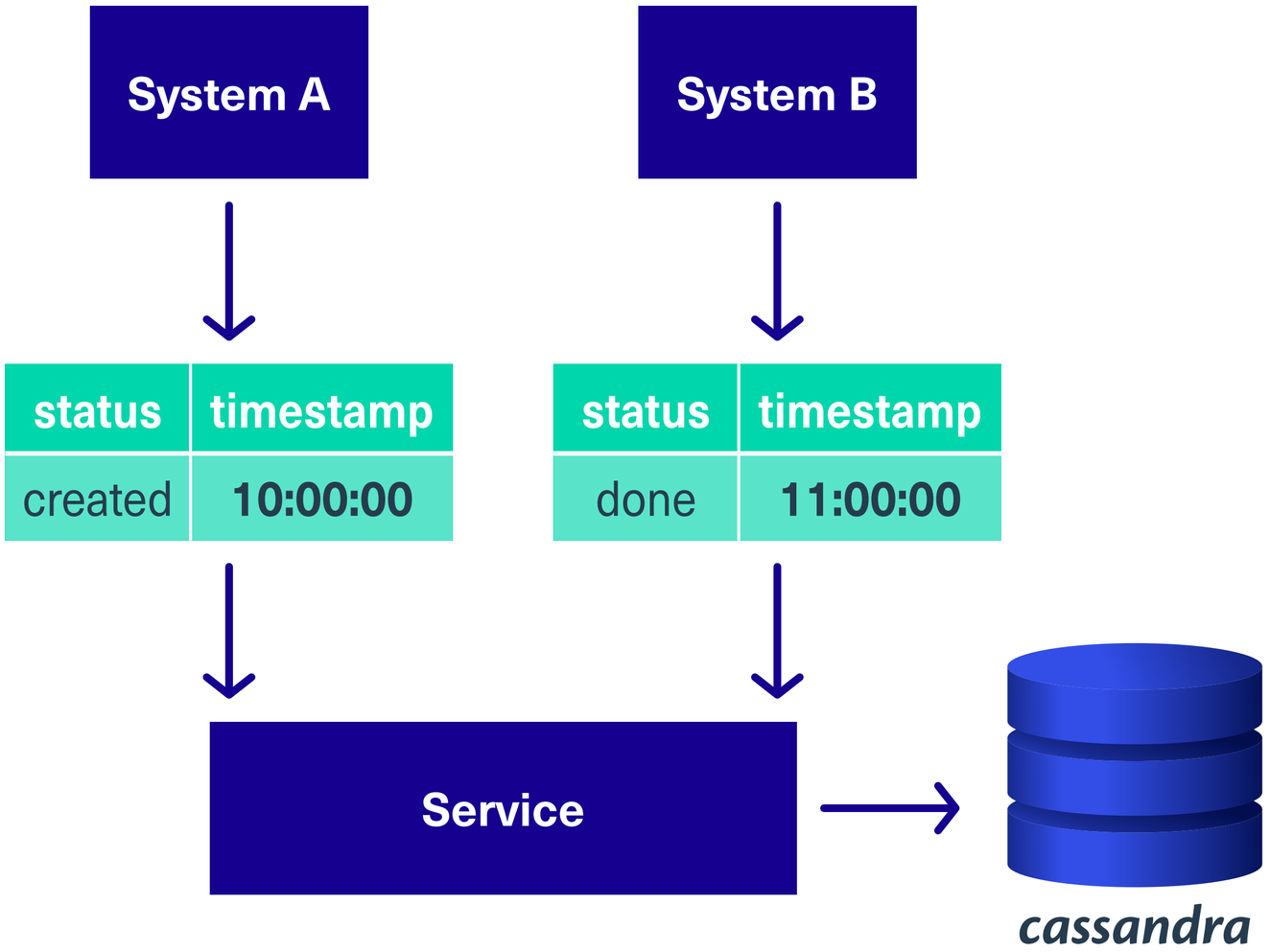
- Много записей, удалений нет, поиск ID клиента и дате
- Статус операции может меняться
- Порядок событий обновления статуса не гарантирован

Пишем ленту операций



- Много записей, удалений нет, поиск ID клиента и дате
- Статус операции может меняться
- Порядок событий обновления статуса не гарантирован

Пишем ленту операций



- Много записей, удалений нет, поиск ID клиента и дате
- Статус операции может меняться
- Порядок событий обновления статуса не гарантирован
- Есть временная метка по которой можно понять порядок

Пишем ленту операций

1. Lightweight transactions (LWT) <, <=, >, >=, !=, IN, EXISTS, NOT EXISTS

```
UPDATE conf.timeline
SET status = ‘done’
WHERE client_id = 1
AND year = 2024
AND month = 4
AND date = ‘2024-04-25’
AND op_id = 333
IF time < ‘11:00:00’
```

status	timestamp
done	11:00:00

Пишем ленту операций

1. Lightweight transactions (LWT) <, <=, >, >=, !=, IN, EXISTS, NOT EXISTS
   но это дорого т. к. 4 запроса вместо одного (и один из них чтение)

```
UPDATE conf.timeline
SET status = ‘done’
WHERE client_id = 1
AND year = 2024
AND month = 4
AND date = ‘2024-04-25’
AND op_id = 333
IF time < ‘11:00:00’
```

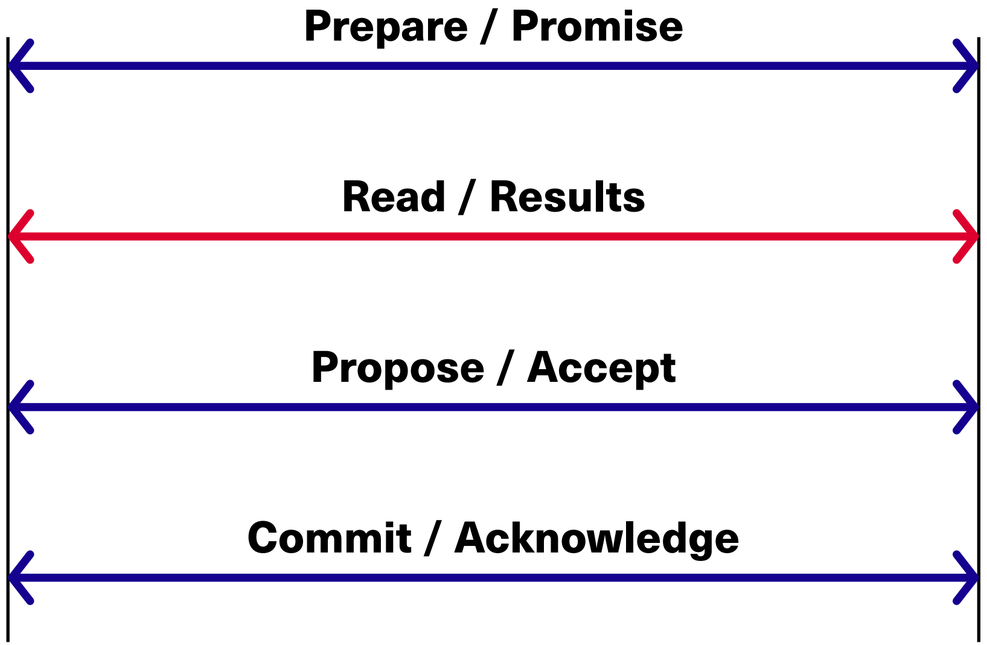
status	timestamp
done	11:00:00

Пишем ленту операций

1. Lightweight transactions (LWT) <, <=, >, >=, !=, IN, EXISTS, NOT EXISTS
   но это дорого т. к. 4 запроса вместо одного (и один из них чтение)

Compare and set (CAS) на базе PAXOS

[Light-weight transactions and Paxos algorithm]



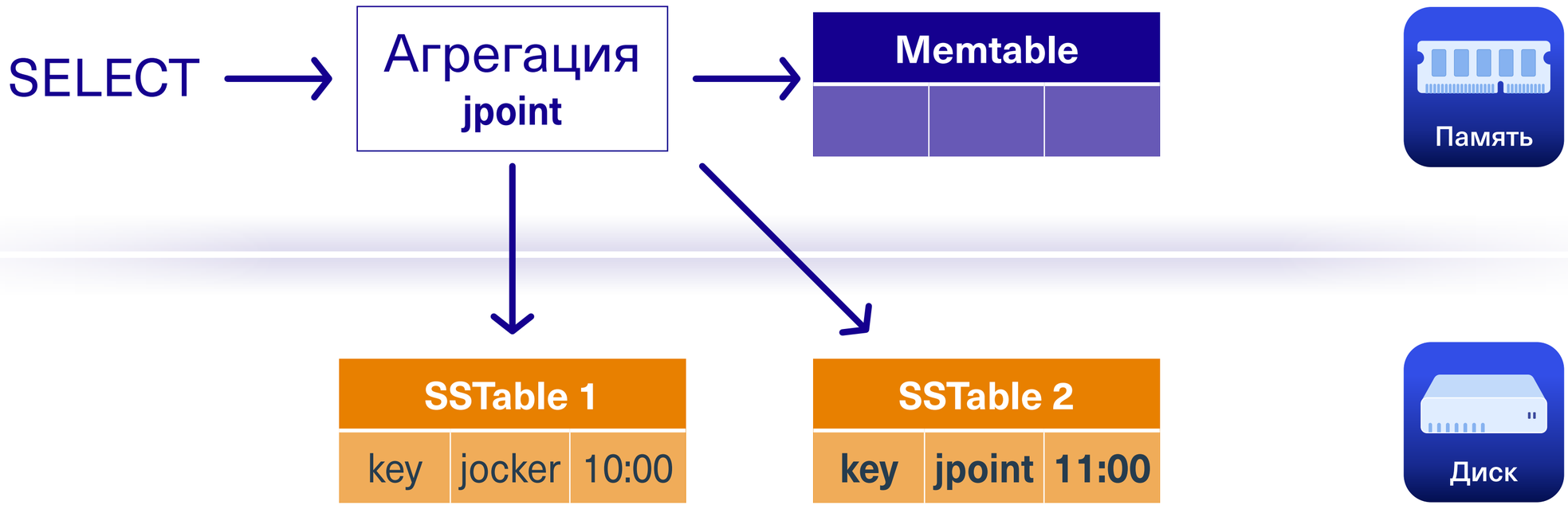
Пишем ленту операций

1. Lightweight transactions (LWT) <, <=, >, >=, !=, IN, EXISTS, NOT EXISTS
   но это дорого т. к. 4 запроса вместо одного (и один из них чтение)
2. USING TIMESTAMP
   это дёшево т.к. не добавляет новой логики

```
UPDATE conf.timeline
USING TIMESTAMP 1714028400000
SET status = ‘done’
WHERE client_id = 1
and year = 2024
AND month = 4
and date = ‘2024-04-25’
and op_id = 333
```

status	timestamp
done	11:00:00

Чтение


SELECT
Агрегация
jpoint
Memtable
Память
SSTable 1
key
jocker
10:00
SSTable 2
key
jpoint
11:00
Диск

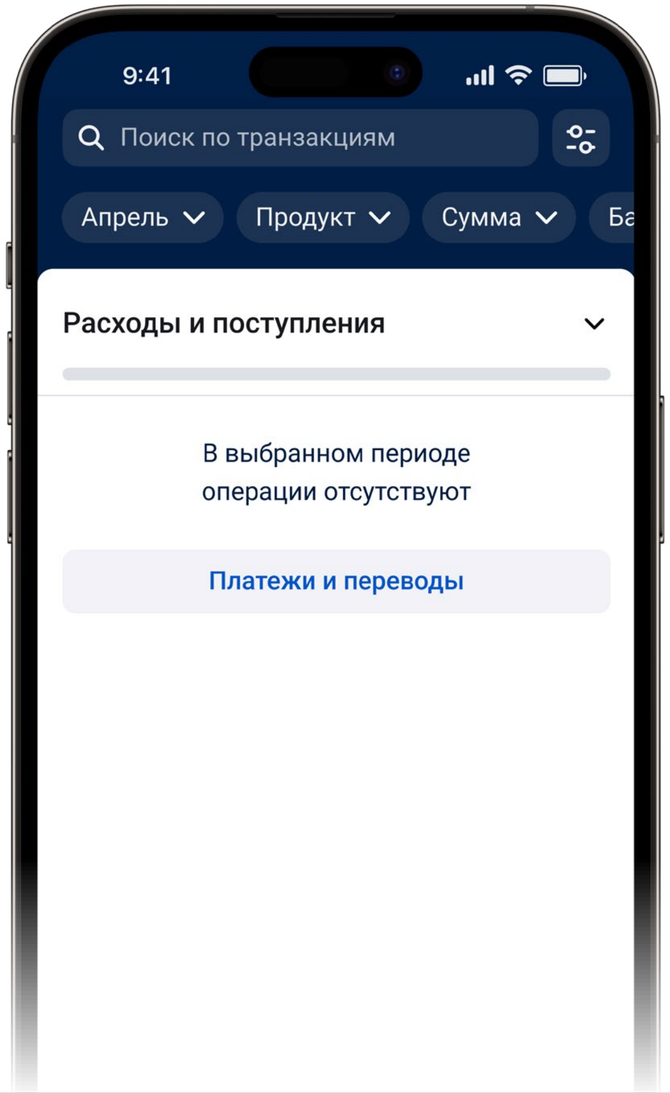
Пишем ленту операций

1. Lightweight transactions (LWT) <, <=, >, >=, !=, IN, EXISTS, NOT EXISTS
   но это дорого т. к. 4 запроса вместо одного (и один из них чтение)
2. USING TIMESTAMP
   это дёшево т.к. не добавляет новой логики

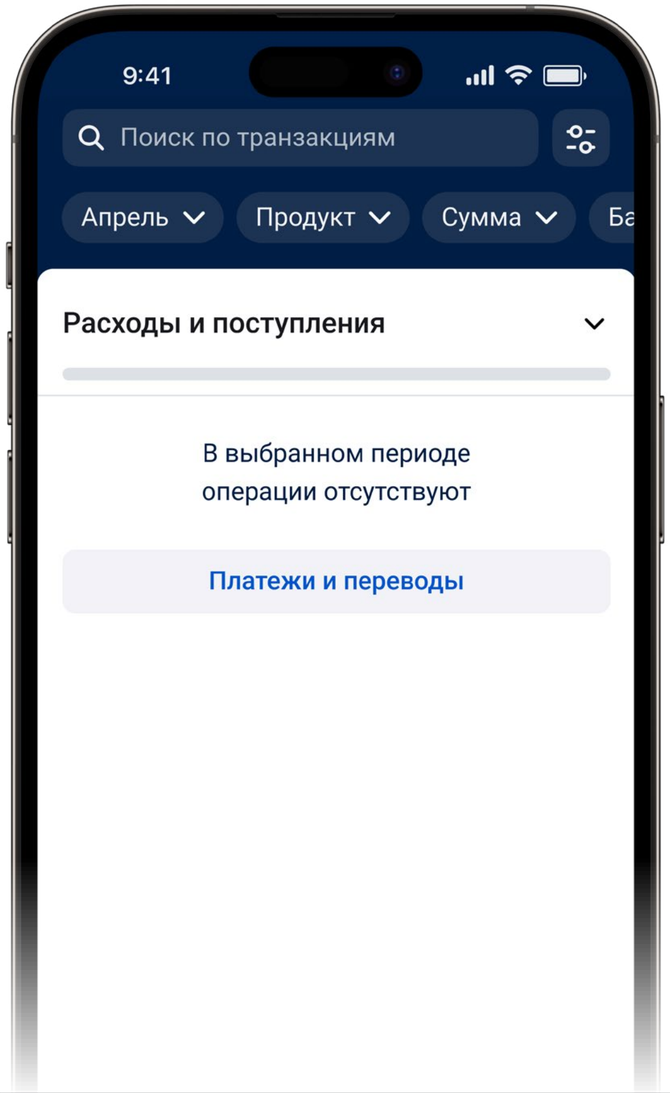
```
UPDATE conf.timeline
USING TIMESTAMP 1714028400000
SET status = ‘done’
WHERE client_id = 1
and year = 2024
AND month = 4
and date = ‘2024-04-25’
and op_id = 333
```

status	timestamp
done	11:00:00

Обновляем с timestamp

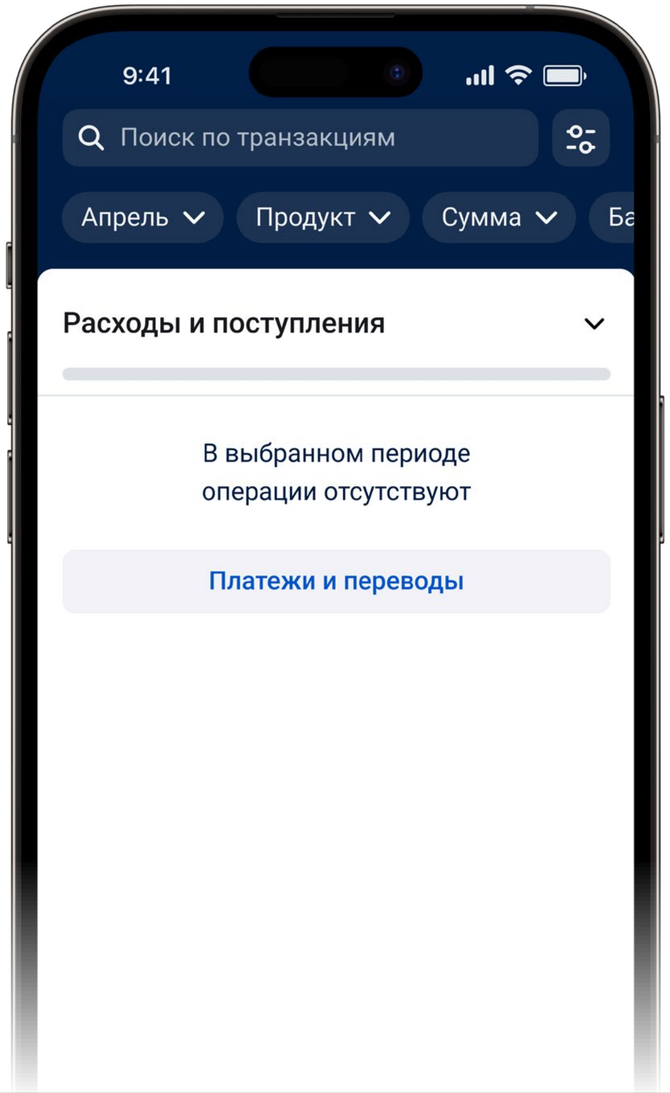


Обновляем с timestamp



SSTable 1	
created	10:00

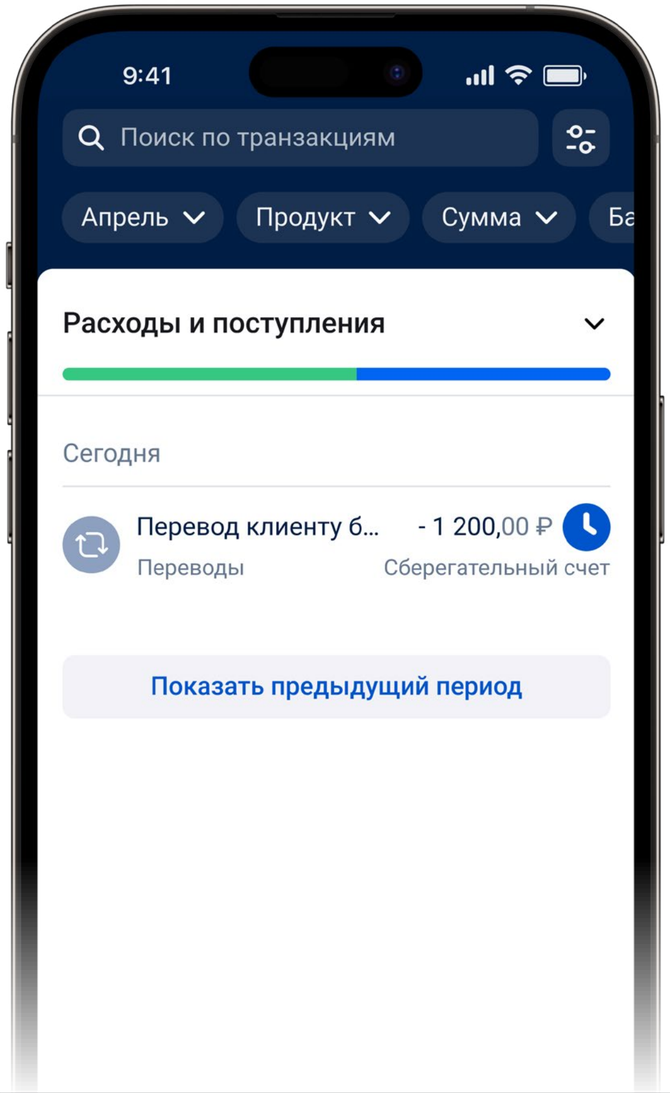
Обновляем с timestamp



Агрегация
status = **created**

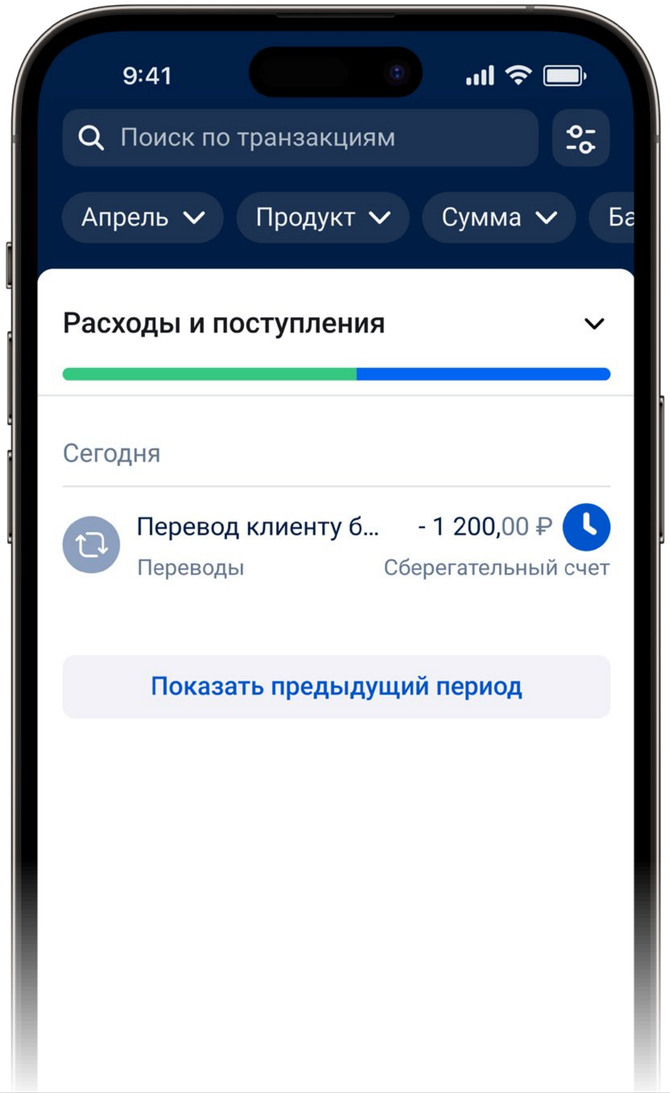
SSTable 1	
created	**10:00**

Обновляем с timestamp



SSTable 1	
created	**10:00**

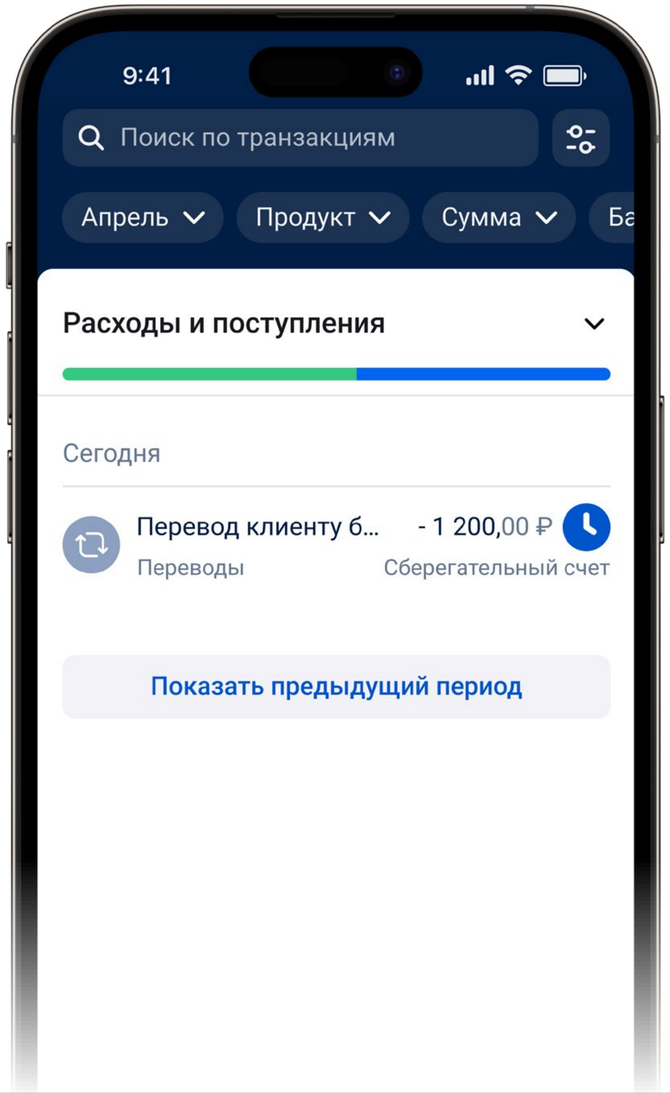
Обновляем с timestamp



SSTable 1	
created	10:00

SSTable 2	
done	10:05

Обновляем с timestamp

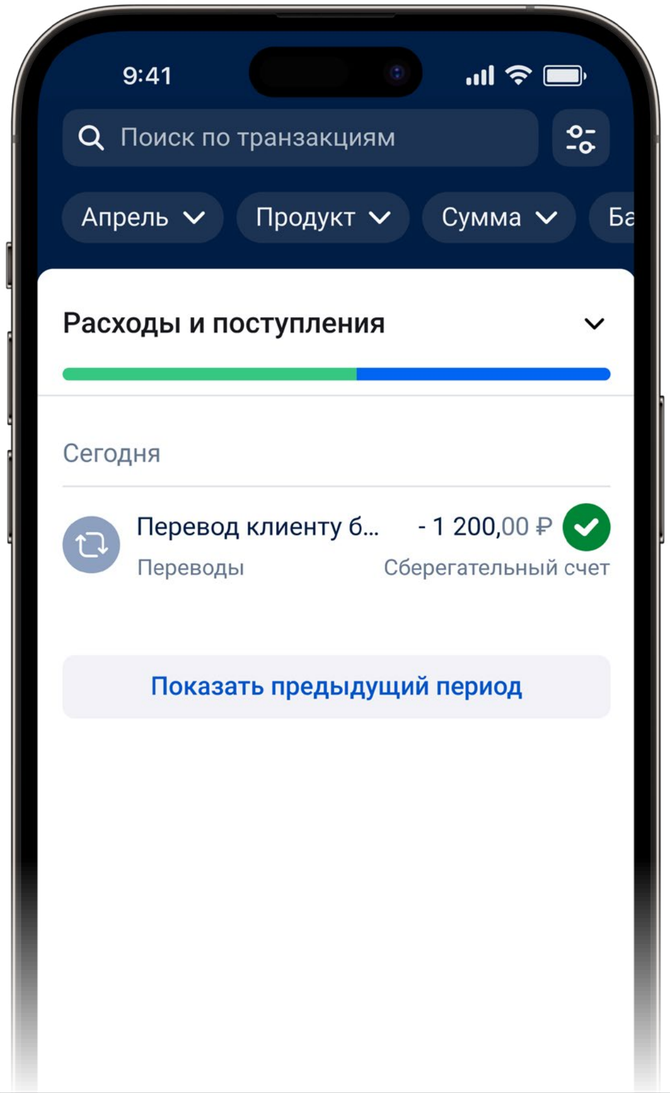


Агрегация
status = **done**

SSTable 1	
created	10:00

SSTable 2	
done	**10:05**

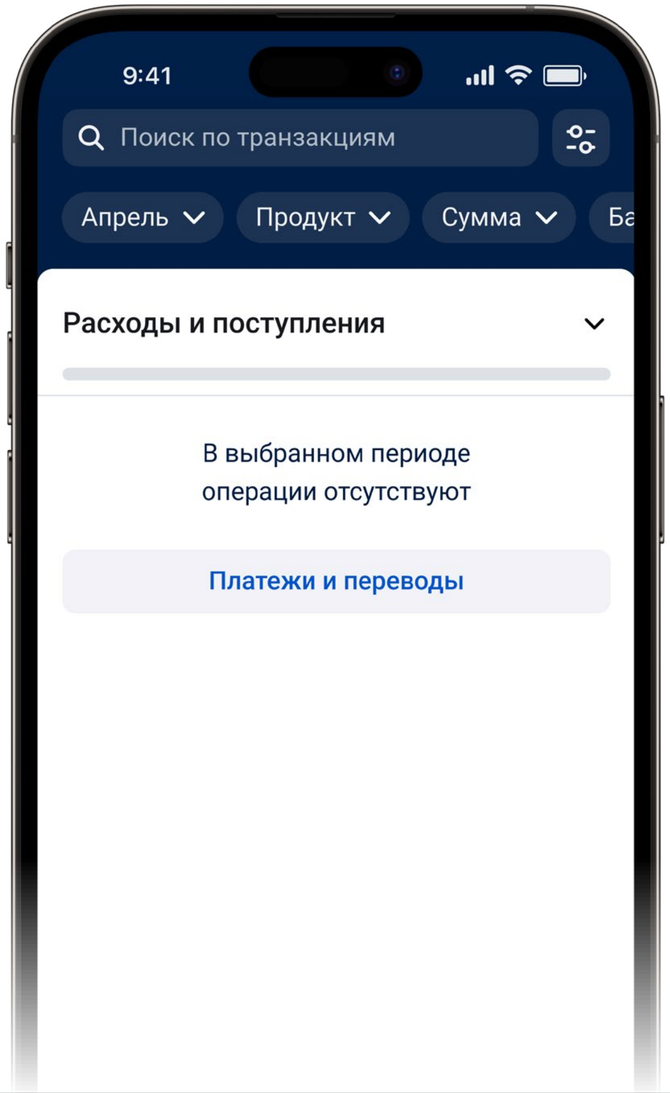
Обновляем с timestamp



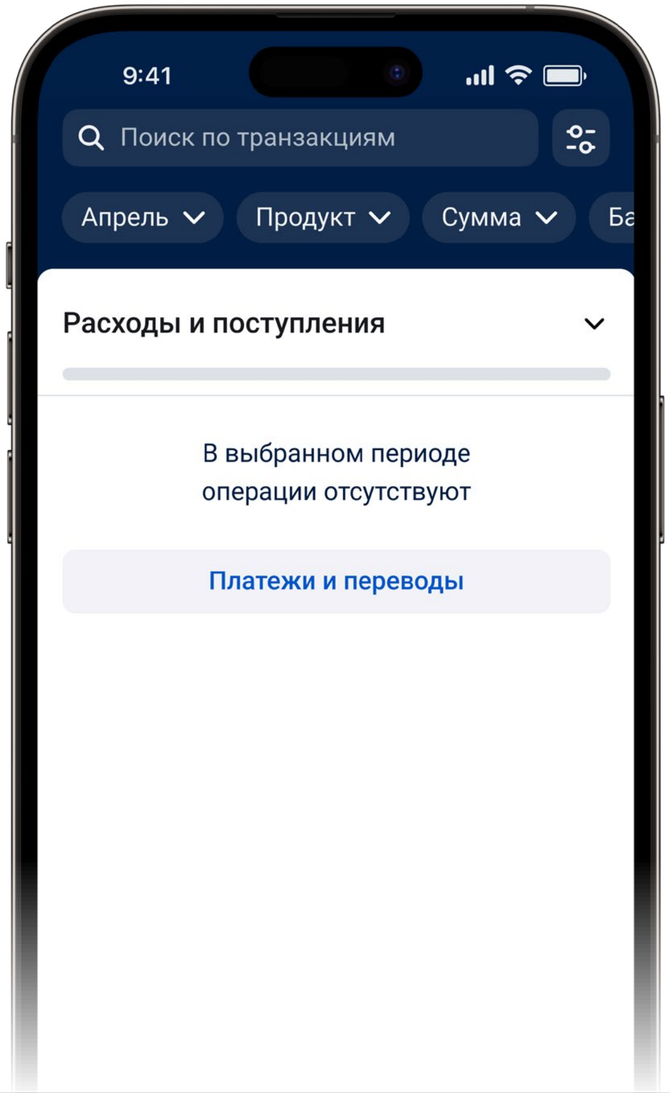
SSTable 1	
created	10:00

SSTable 2	
done	10:05

Обновляем с timestamp

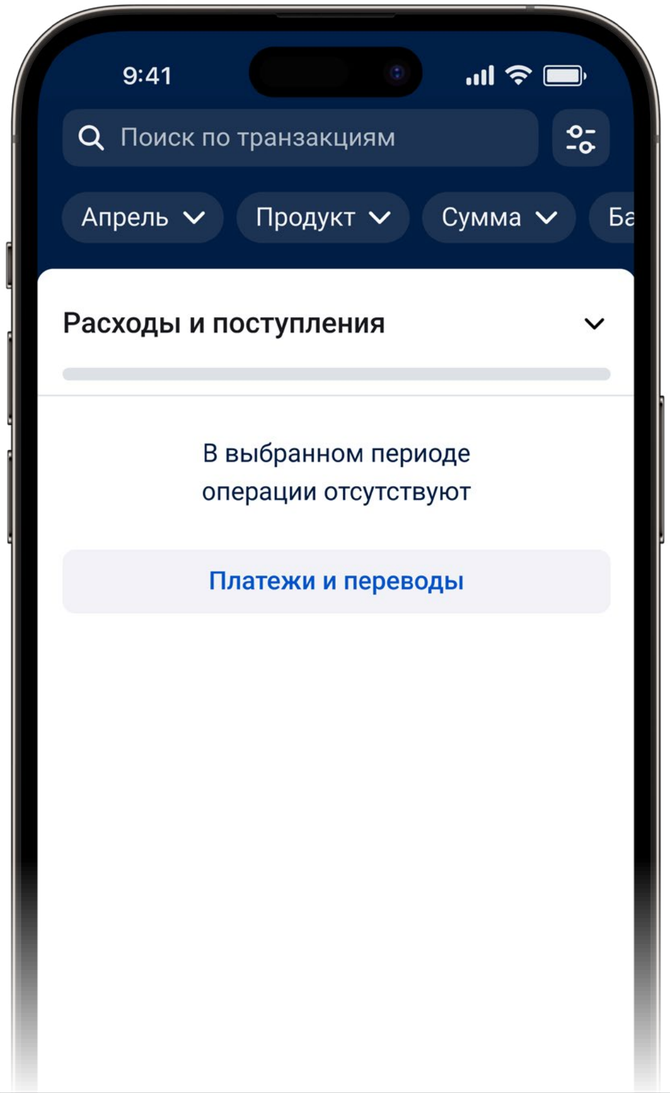


Обновляем с timestamp



SSTable 1	
done	10:05

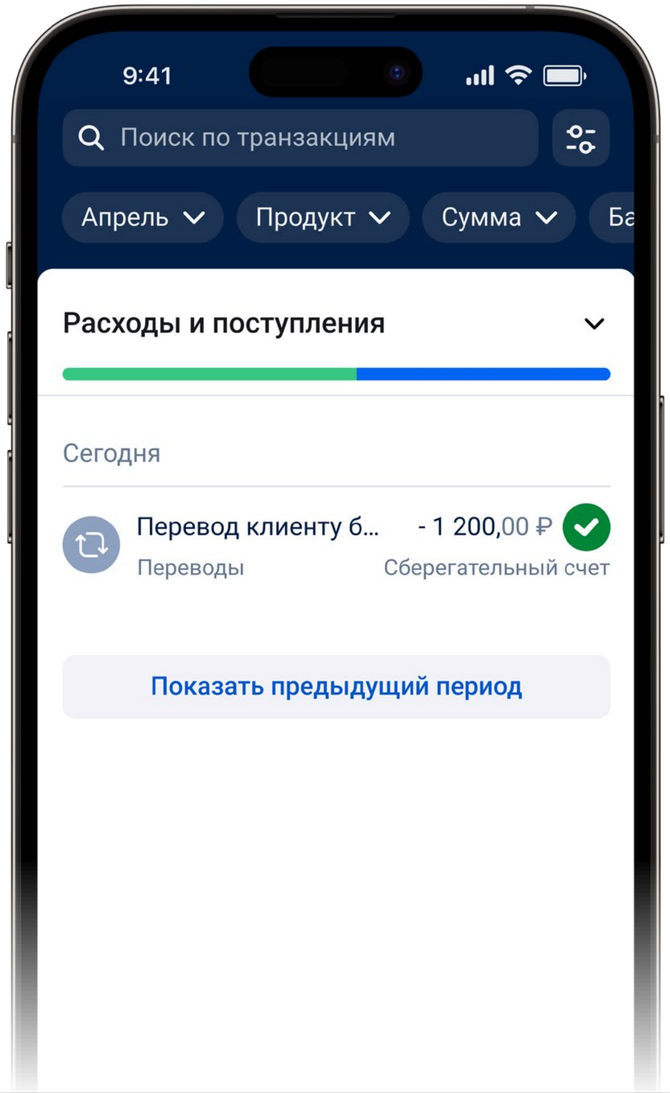
Обновляем с timestamp



Агрегация
status = **done**

SSTable 1	
done	10:05

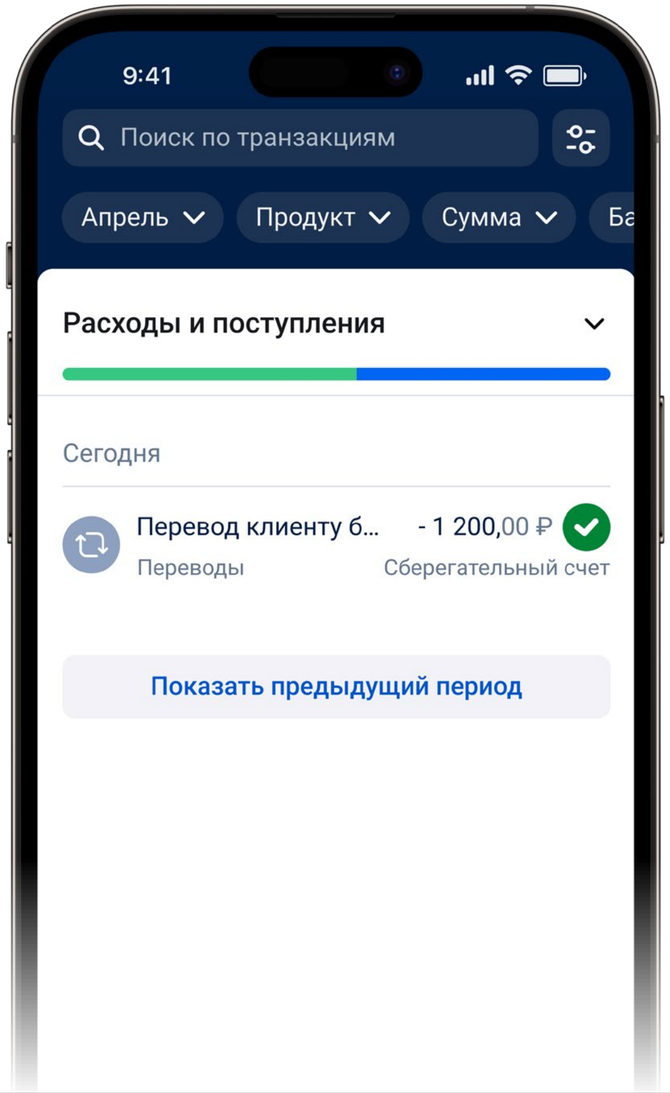
Обновляем с timestamp



Агрегация
status = **done**

SSTable 1	
done	**10:05**

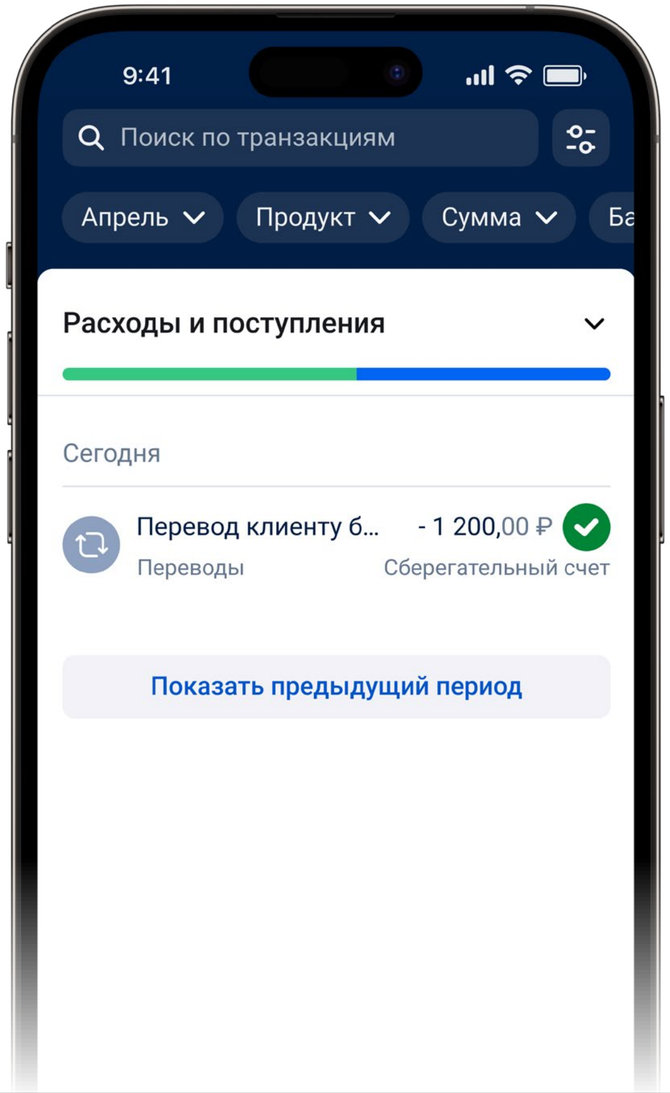
Обновляем с timestamp



SSTable 1	
done	10:05

SSTable 2	
created	10:00

Обновляем с timestamp

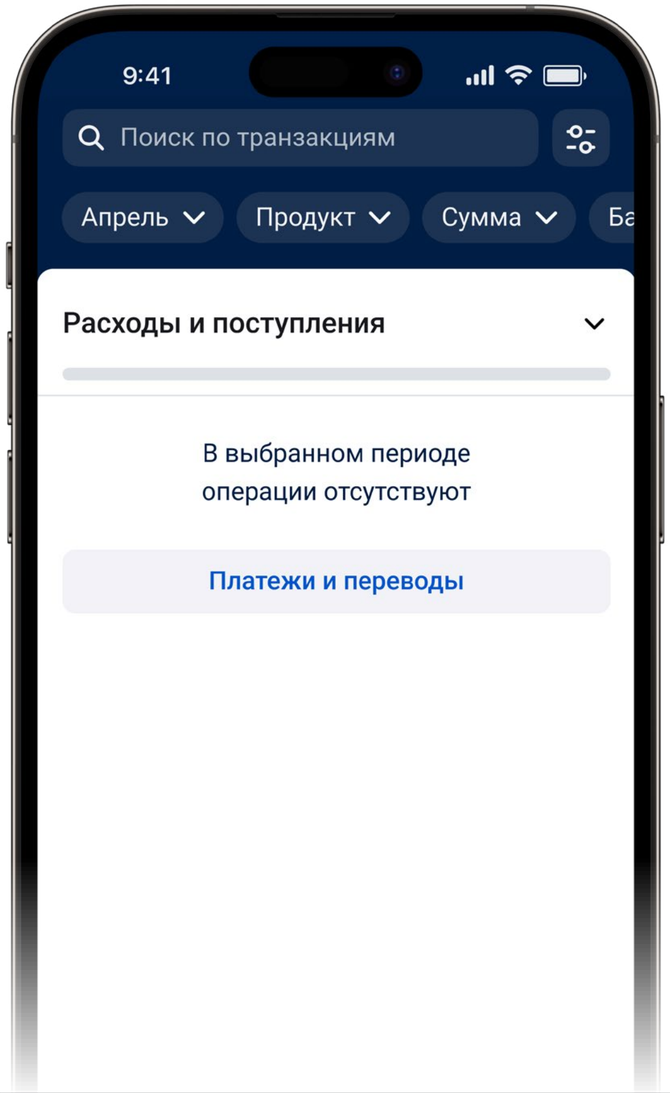


Агрегация
status = **done**

SSTable 1	
done	**10:05**

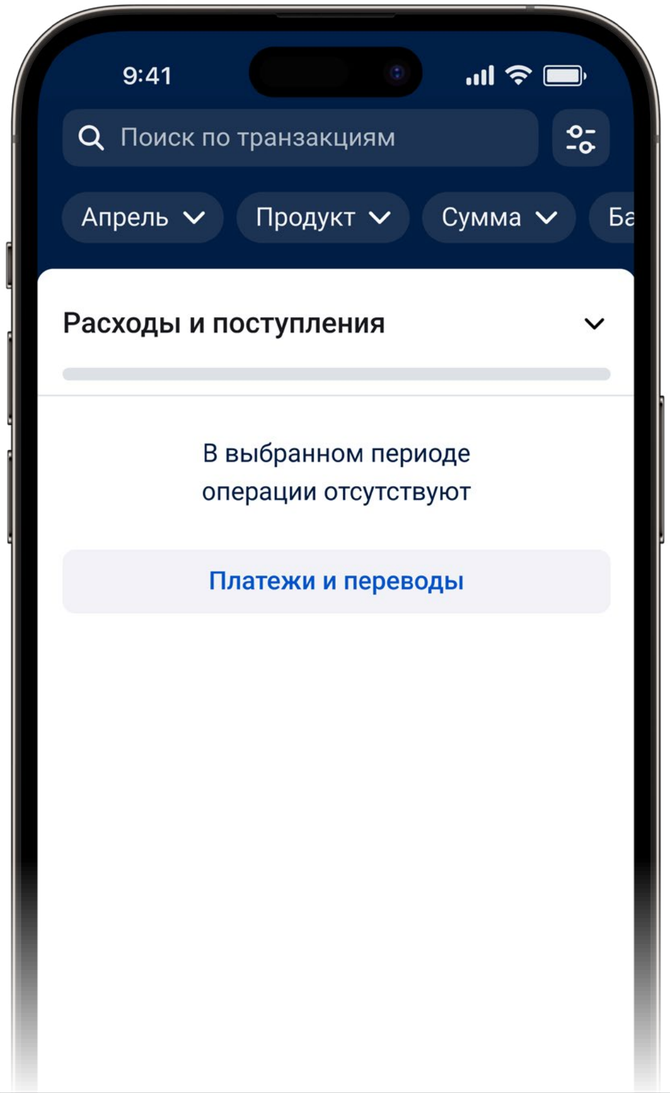
SSTable 2	
created	10:00

Удаляем ошибочную запись



SSTable 1	
error	10:00

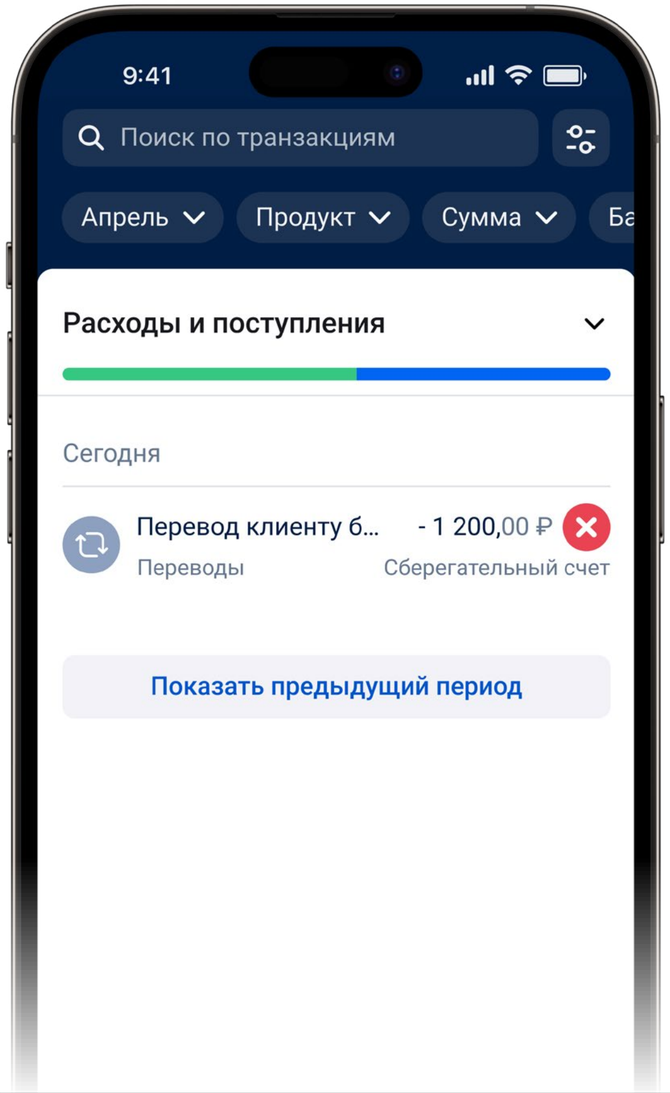
Удаляем ошибочную запись



Агрегация
status = **error**

SSTable 1	
error	**10:00**

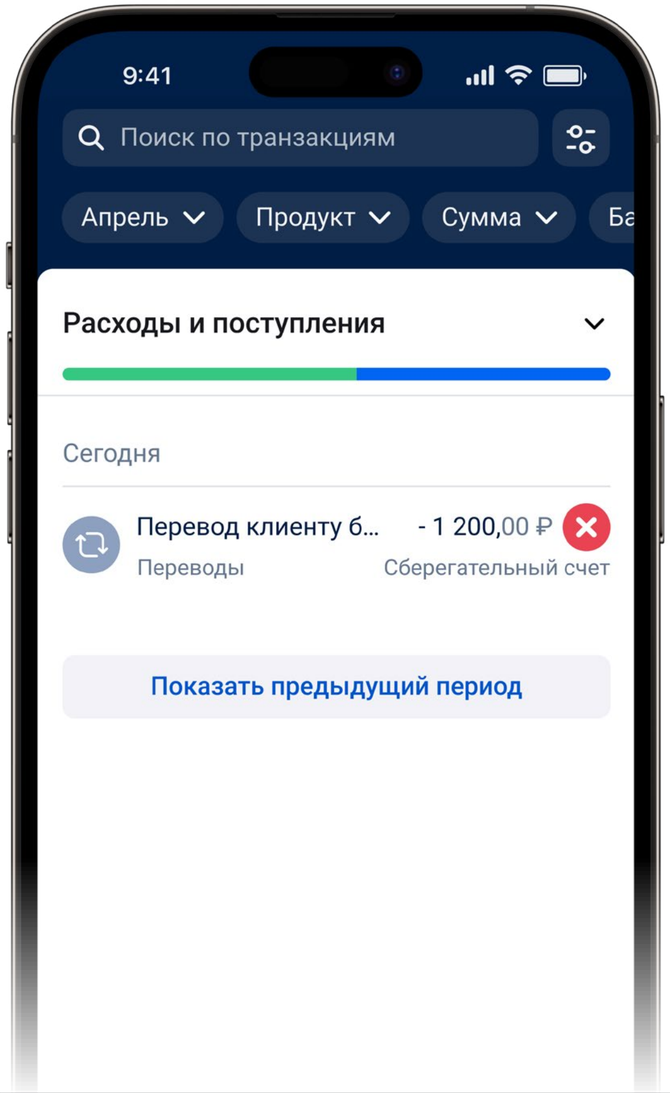
Удаляем ошибочную запись



Агрегация
status = **error**

SSTable 1	
error	**10:00**

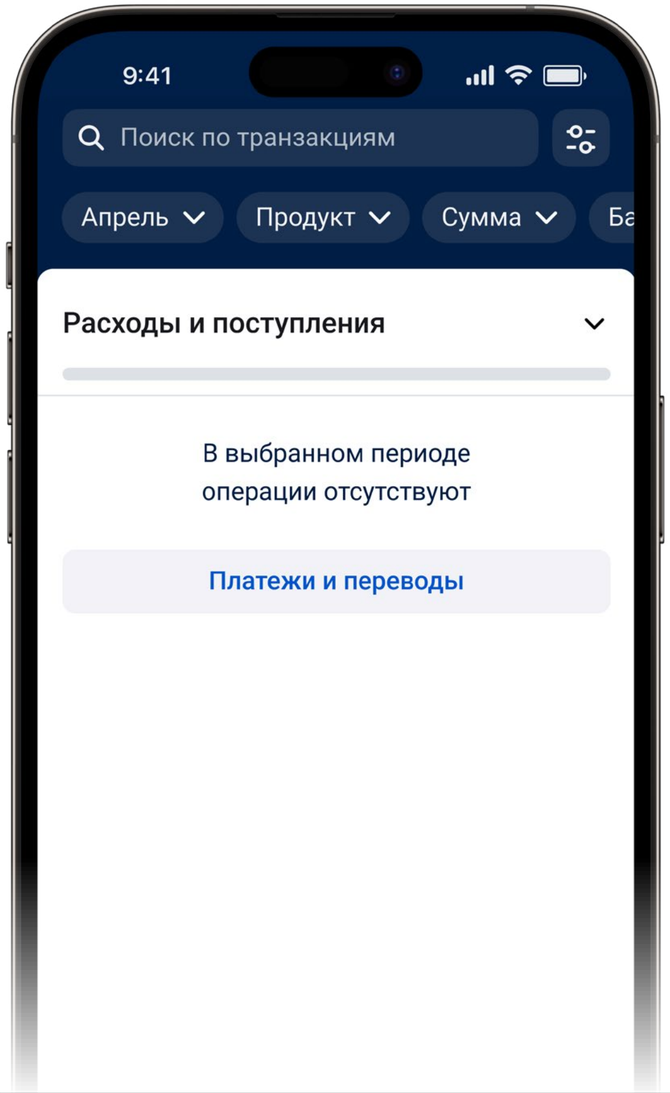
Удаляем ошибочную запись



SSTable 1	
error	10:00

SSTable 2	
tombstone	11:00

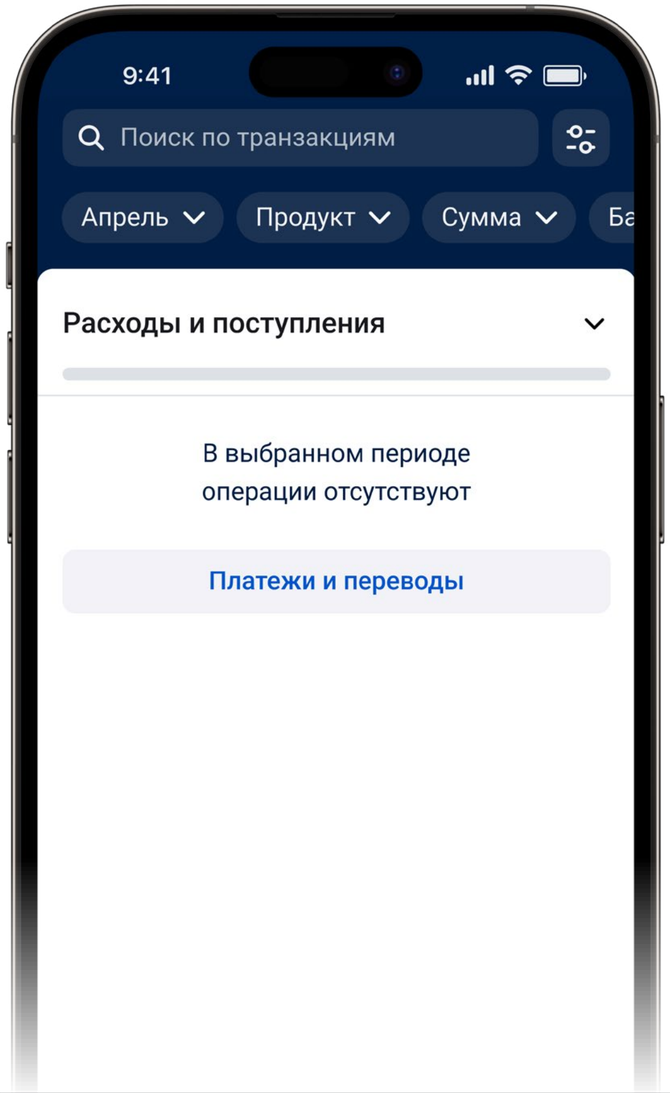
Удаляем ошибочную запись



SSTable 1	
error	10:00

SSTable 2	
tombstone	11:00

Удаляем ошибочную запись

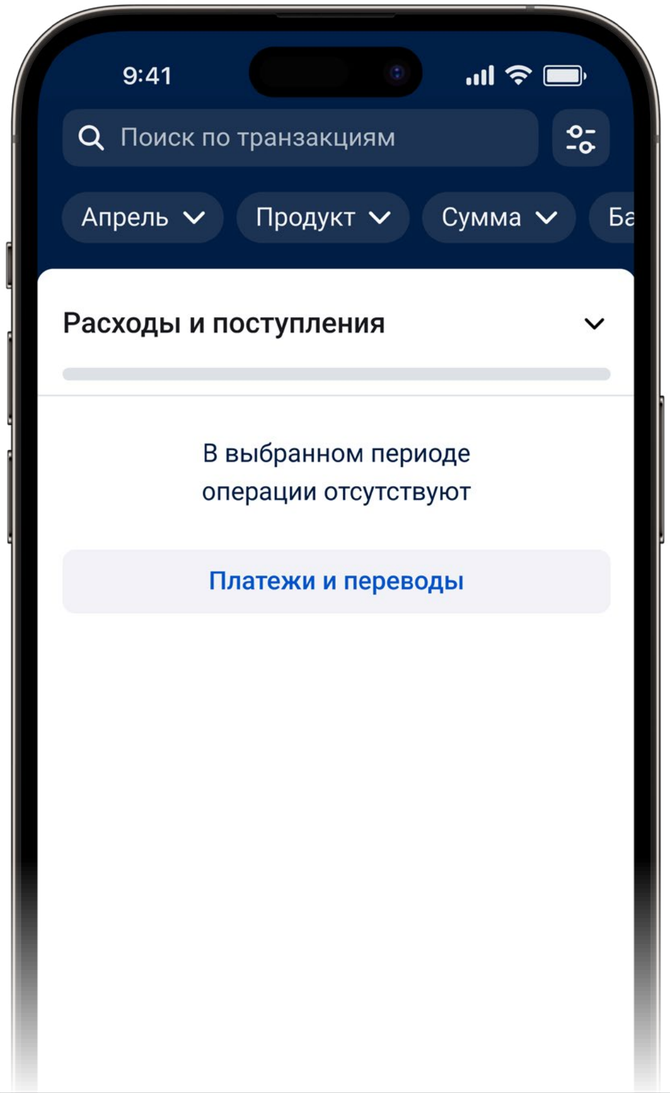


SSTable 1	
error	10:00

SSTable 2	
tombstone	11:00

SSTable 3	
created	10:05

Удаляем ошибочную запись



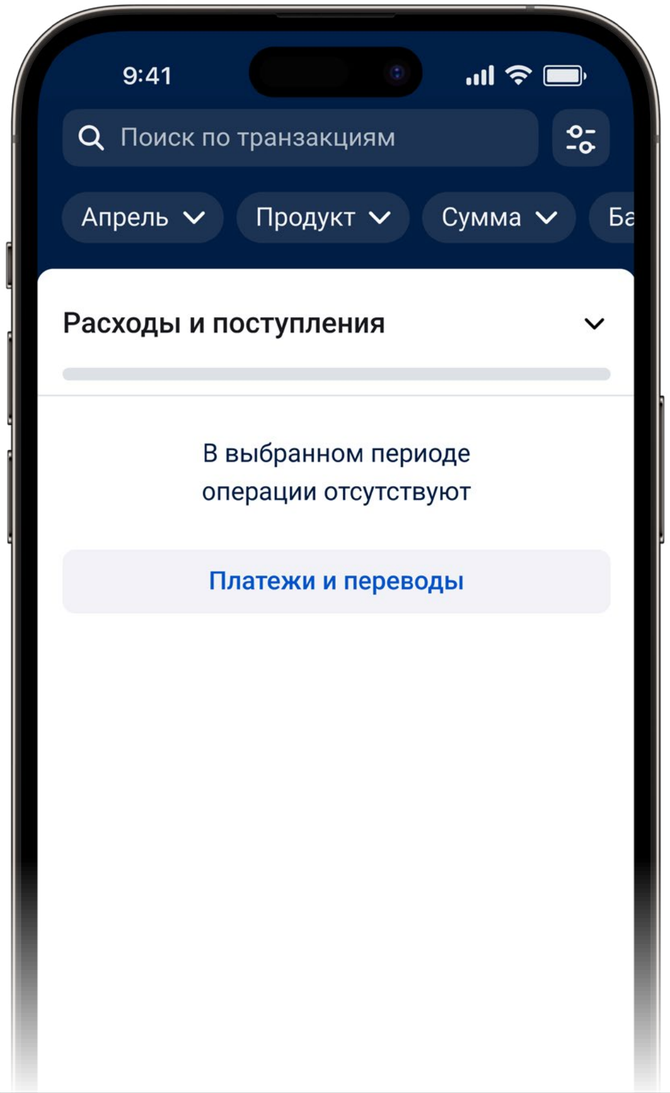
Агрегация
status = **null**

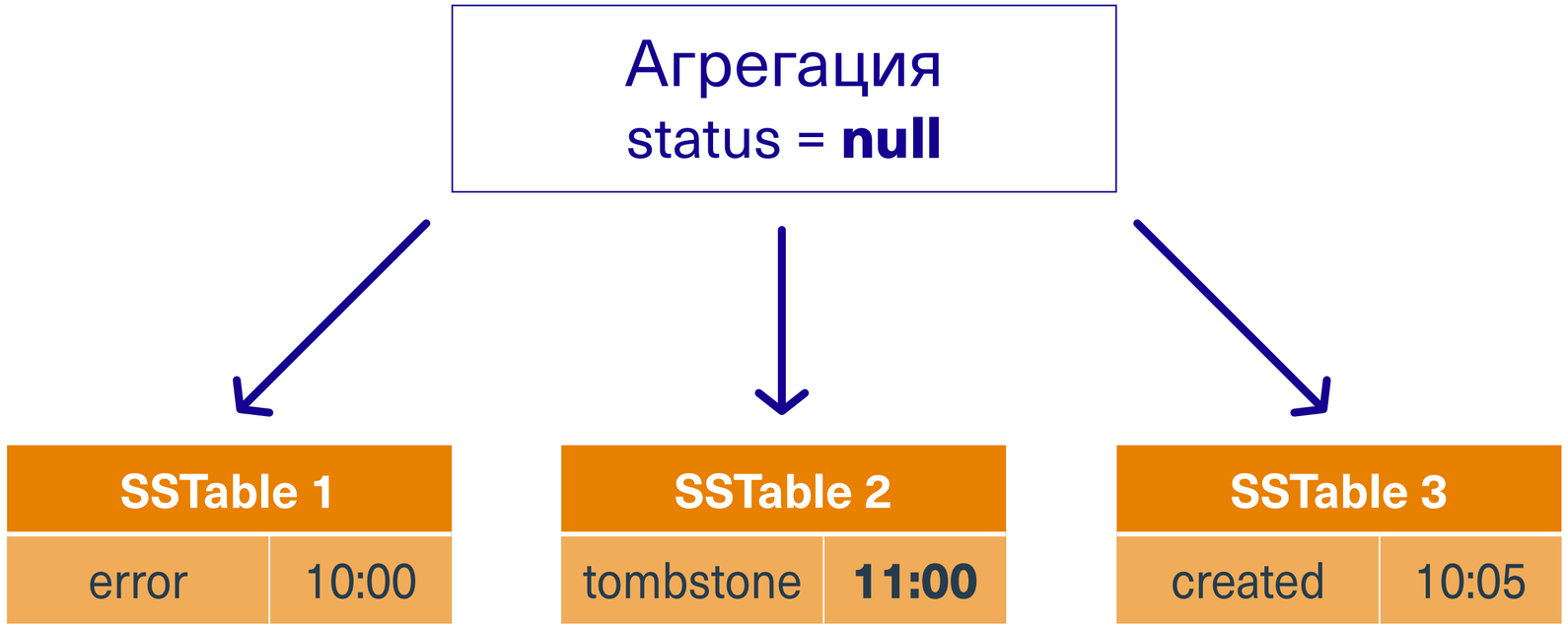
SSTable 1	
error	10:00

SSTable 2	
tombstone	**11:00**

SSTable 3	
created	10:05

Удаляем ошибочную запись

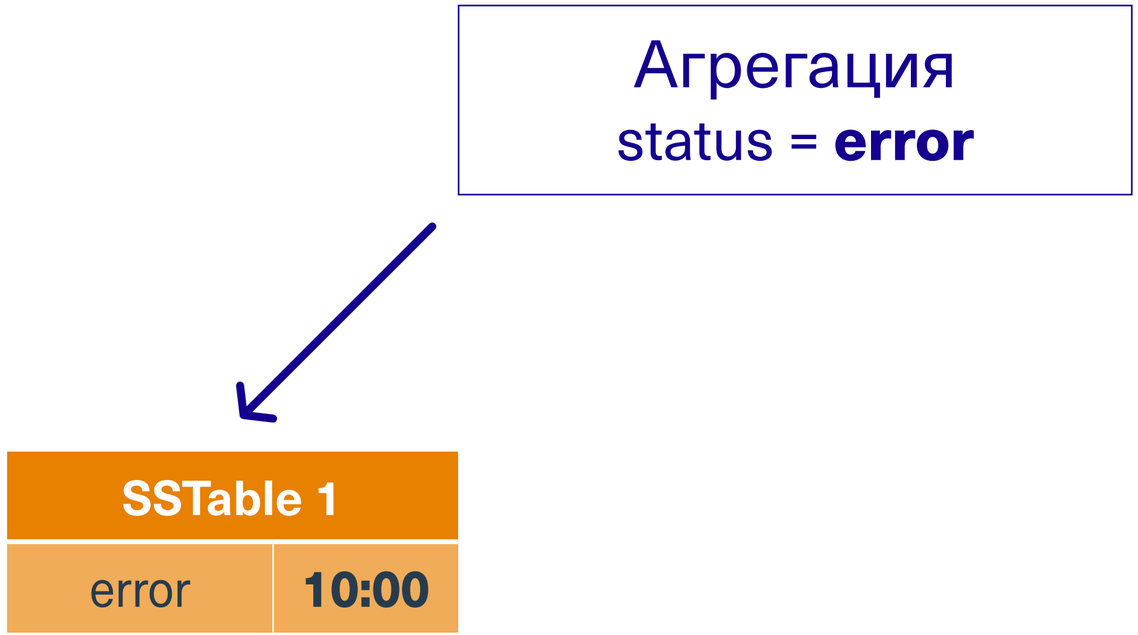




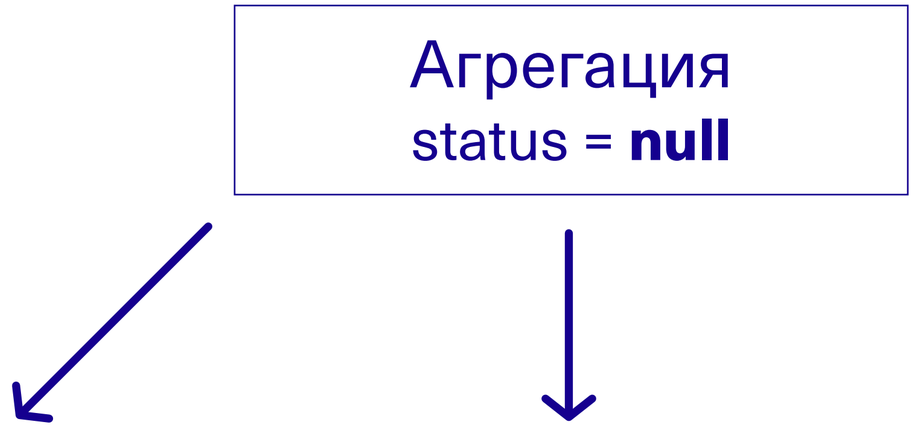
https://medium.com/@jeeyoungk/why-i-love-databases-1d4cc433685f

“You may be able to delete a future write with an old **doomstone**. This is colloquially referred as a doomstone. Hilarious as it is, it’s a real problem.”

Удаляем ошибочную запись правильно



Удаляем ошибочную запись правильно



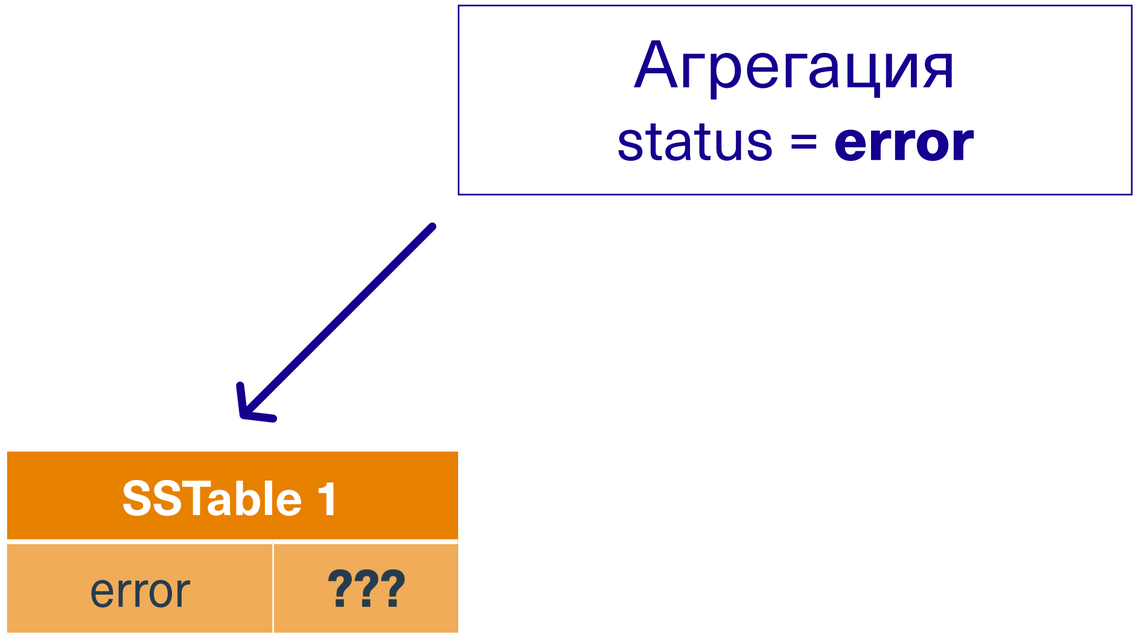

```
DELETE status
FROM conf.timeline
USING TIMESTAMP
1711868400001
WHERE client_id = 1
AND year = 2024
AND month = 4
AND date = ‘2024-04-25’
AND op_id = 333
```

SSTable 1	
error	10:00

SSTable 2	
tombstone	**10:00**

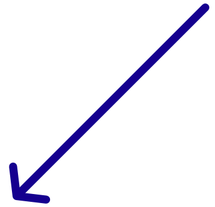
При совпадении временных меток приоритет отдаётся tombstone

Как узнать timestamp



Как узнать timestamp

Агрегация
status = **error**



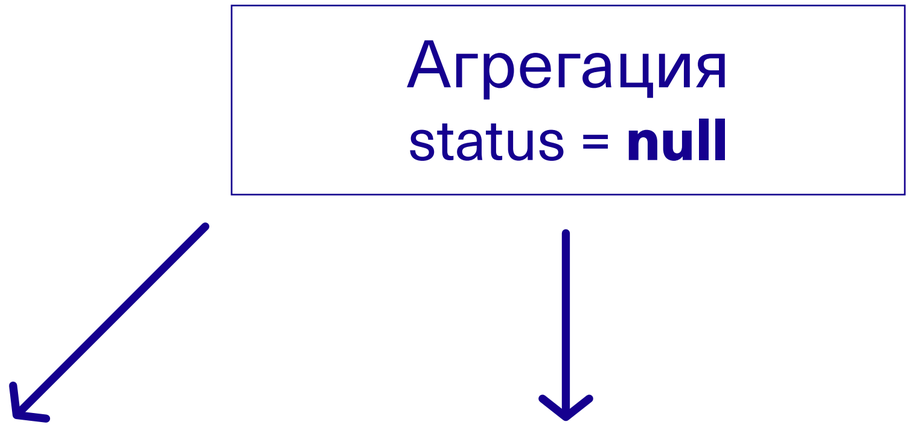
```
SELECT status,
WRITETIME(status)
FROM conf.timeline
WHERE client_id = 1
AND year = 2024
AND month = 4
AND date = '2024-04-25'
AND op_id = 333
```

SSTable 1	
error	**???**

```
 status |    writetime(status)
--------+--------------------
 error  |     1713456291673174
```

Как узнать timestamp

А если уже удалили?




```
SELECT status,
WRITETIME(status)
FROM conf.timeline
WHERE client_id = 1
AND year = 2024
AND month = 4
AND date = '2024-04-25'
AND op_id = 333
```

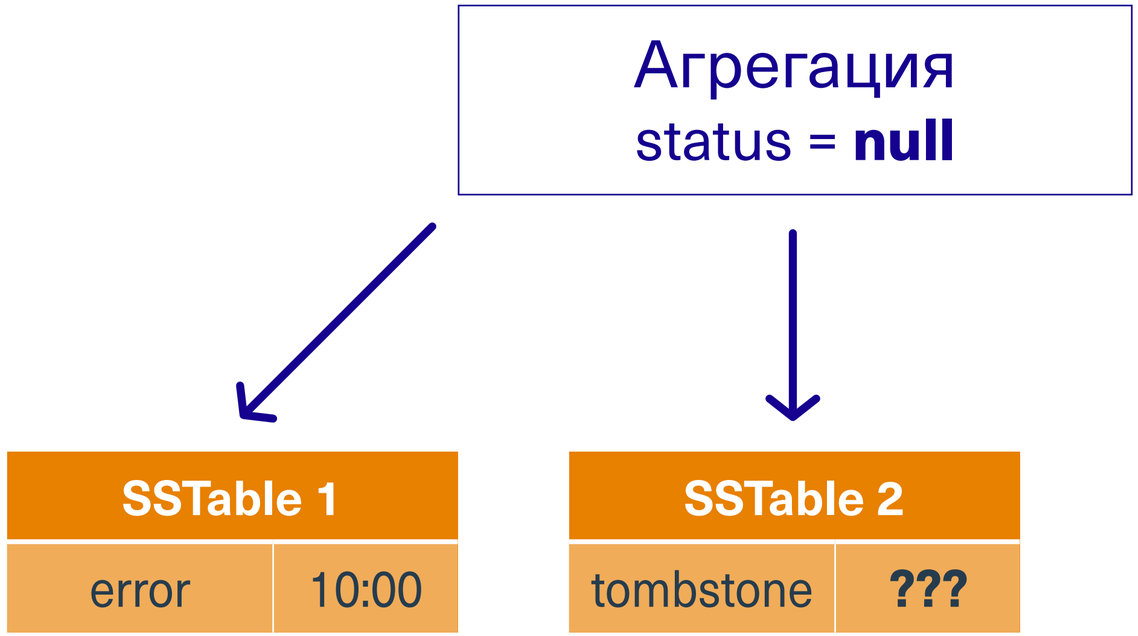
SSTable 1	
error	10:00

SSTable 2	
tombstone	**???**

```
status | writetime(status)
-------+------------------
null   |              null
```

Как узнать timestamp

А если уже удалили?




```
sstabledump lb-6-big-Data.db
[
{“key“: “1:2024“,
 “cells“: [[“2024-04-25:1:status»,1711962663,1711868400001,“d“]]}
]
```

Tombstone требуют внимания

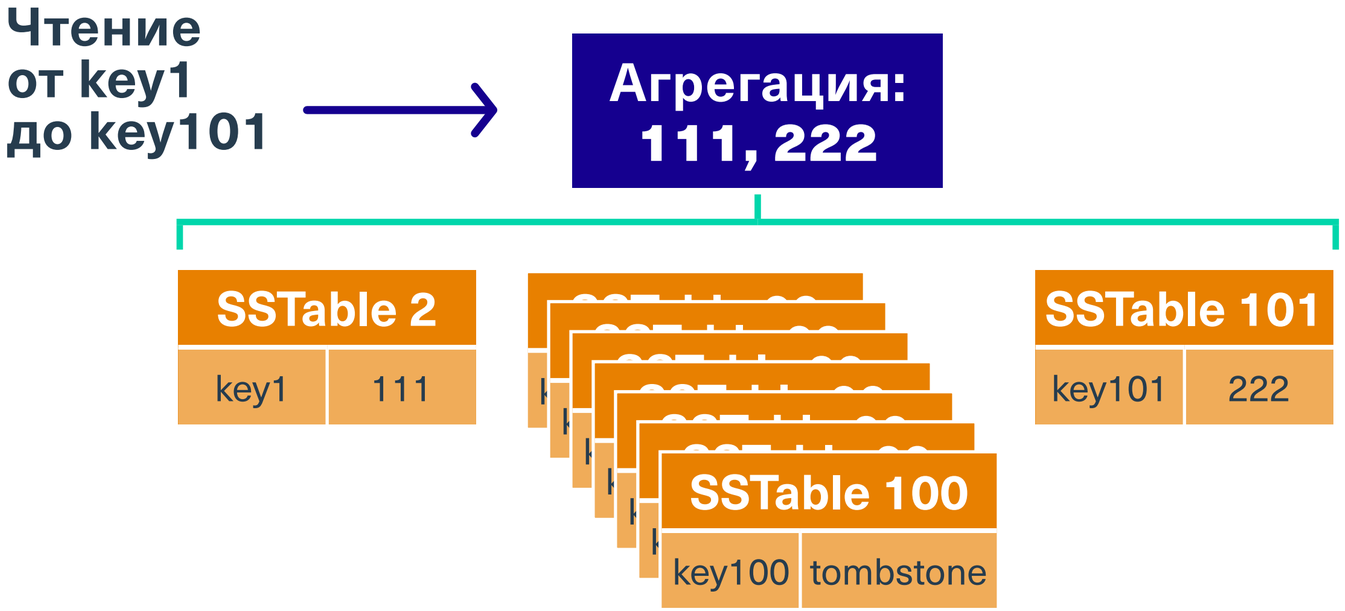
Создаются при

- Удалении строки
- Удалении ячейки
- Удалении диапазона ячеек
- Удалении элемента коллекций
- Вставке значения null

Tombstone требуют внимания

Создаются при

- Удалении строки
- Удалении ячейки
- Удалении диапазона ячеек
- Удалении элемента коллекций
- Вставке значения null



Занимают место на диске

Влияют на IO и загружаются в память при запросах по диапазону ключей и full scan

Tombstone требуют внимания

tombstone_warn_threshold: 1000

tombstone_failure_threshold: 100000

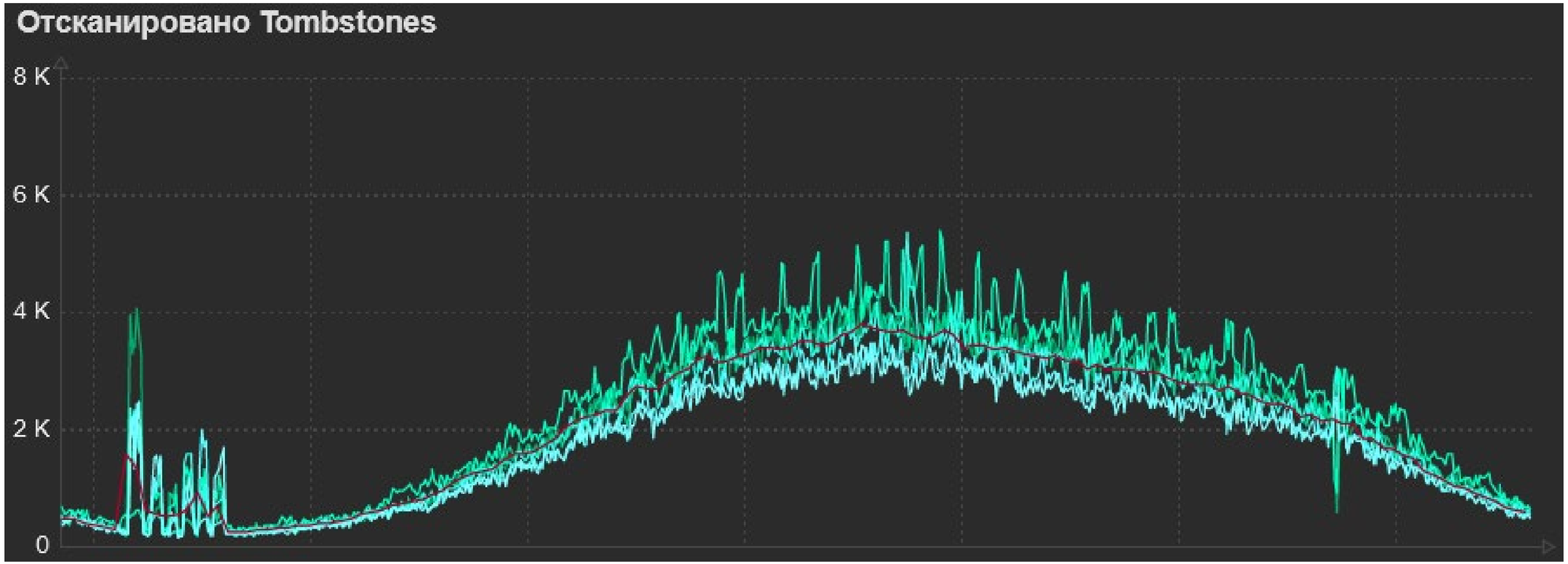
Tombstone требуют внимания

tombstone_warn_threshold: 1000

tombstone_failure_threshold: 100000

```
nodetool tablestats conf.timeline
 ...
Average tombstones per slice (last five minutes): 1.0
Maximum tombstones per slice (last five minutes): 1
```

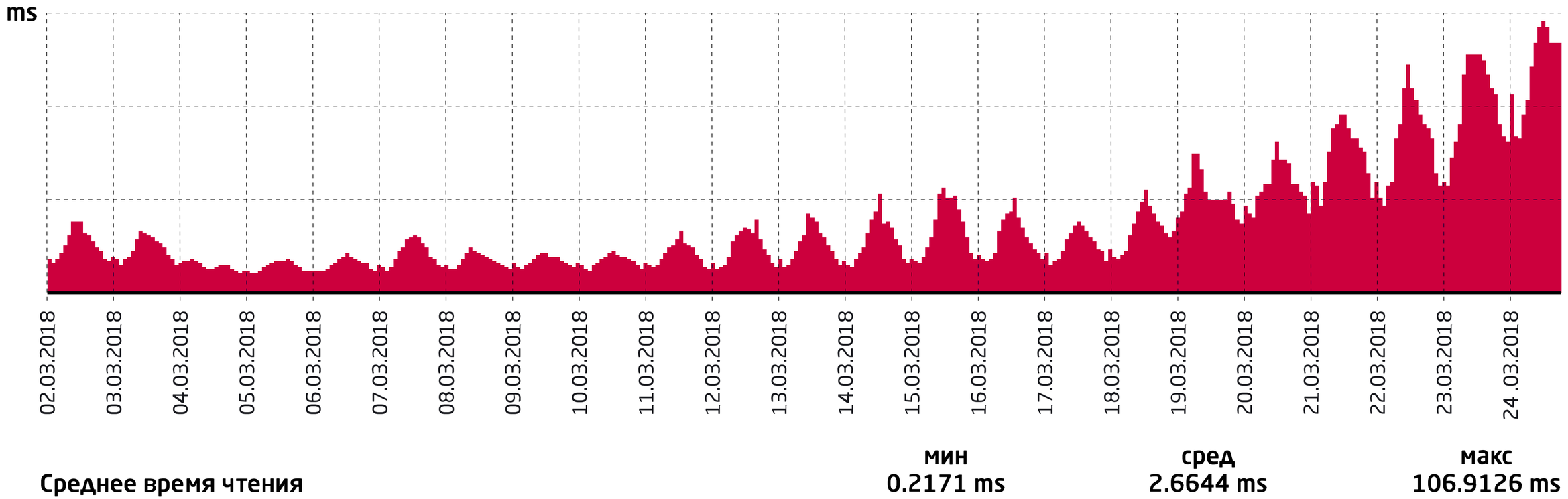
Tombstone требуют внимания



TombstoneScannedHistogram — количество tombstone обработанных при чтении из таблицы

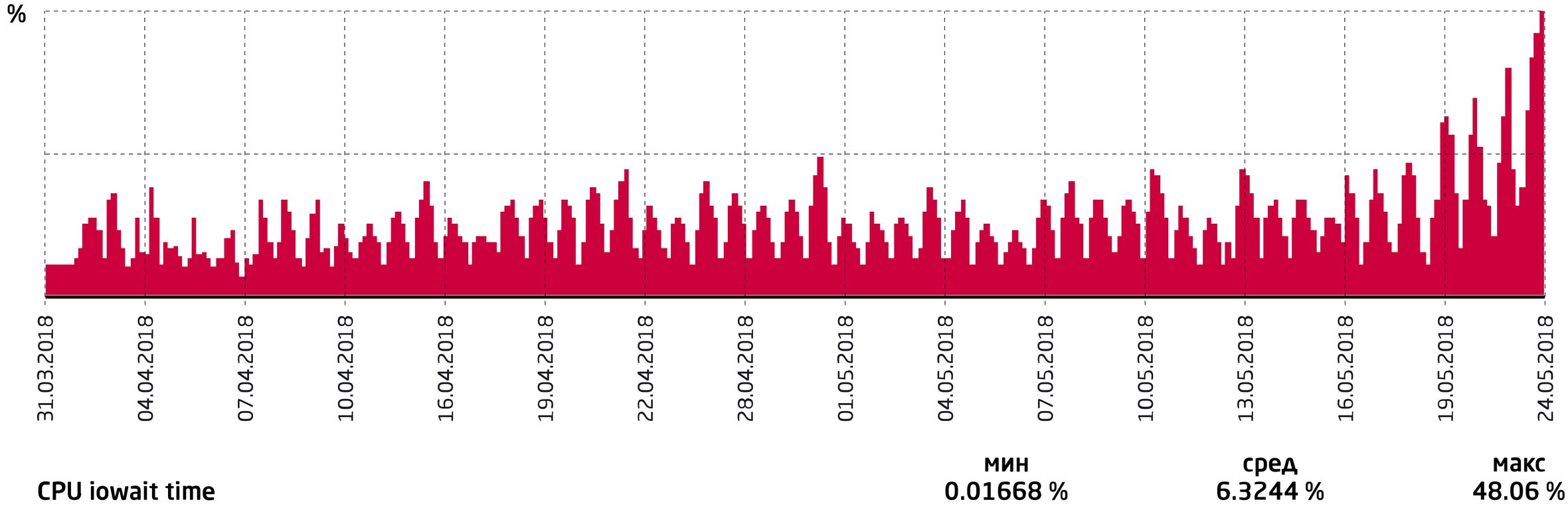
Новая проблема

- Время чтения увеличивается



Новая проблема

- Время чтения увеличивается
- А ещё растёт iowait



Новая проблема

- Время чтения увеличивается
- А ещё растёт iowait
- Упёрлись в ограничения дисков

Новая проблема

- Время чтения увеличивается
- А ещё растёт iowait
- Упёрлись в ограничения дисков
- Времени на обновление железа нет
- Ожидается пиковая нагрузка

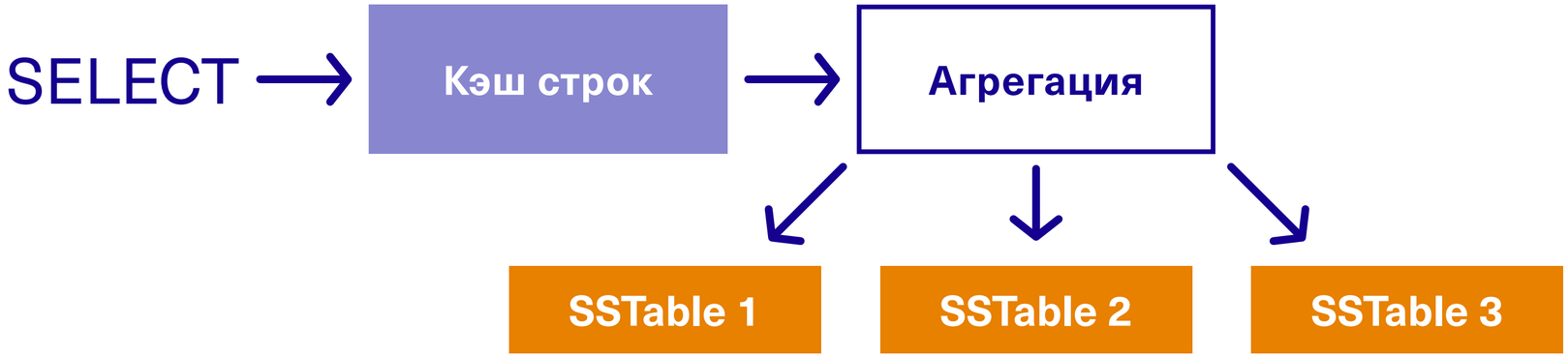
Как разгрузить диск?

- Full scan-ов нет
- Большого количества tombstone нет
- Есть кейсы когда мы много обновляем и читаем

Как разгрузить диск?

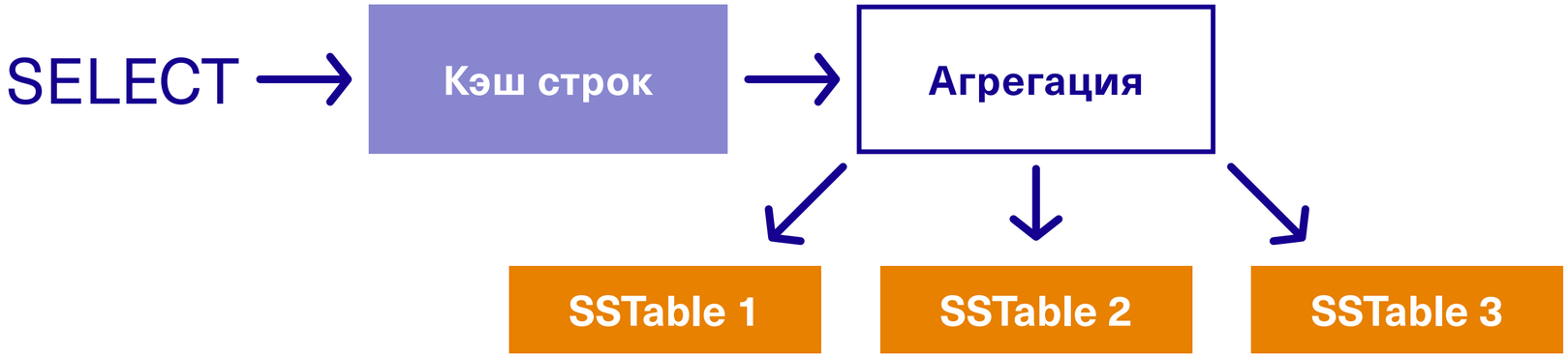
- Full scan-ов нет
- Большого количества tombstone нет
- Есть кейсы когда мы много обновляем и читаем

Кэш строк



rows_per_partition — сколько первых строк партиции кэшировать

Кэш строк

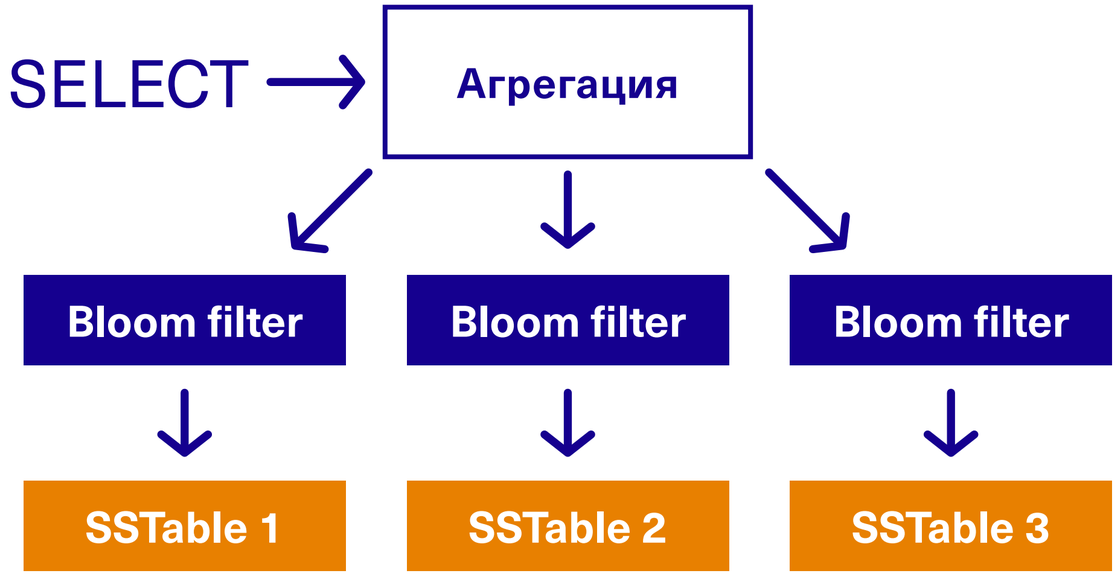


rows_per_partition — сколько первых строк партиции кэшировать

- Кэширует только первые строки
- Сбрасывается при записи
- Utilizing appropriate OS page cache will result in better performance than using row caching

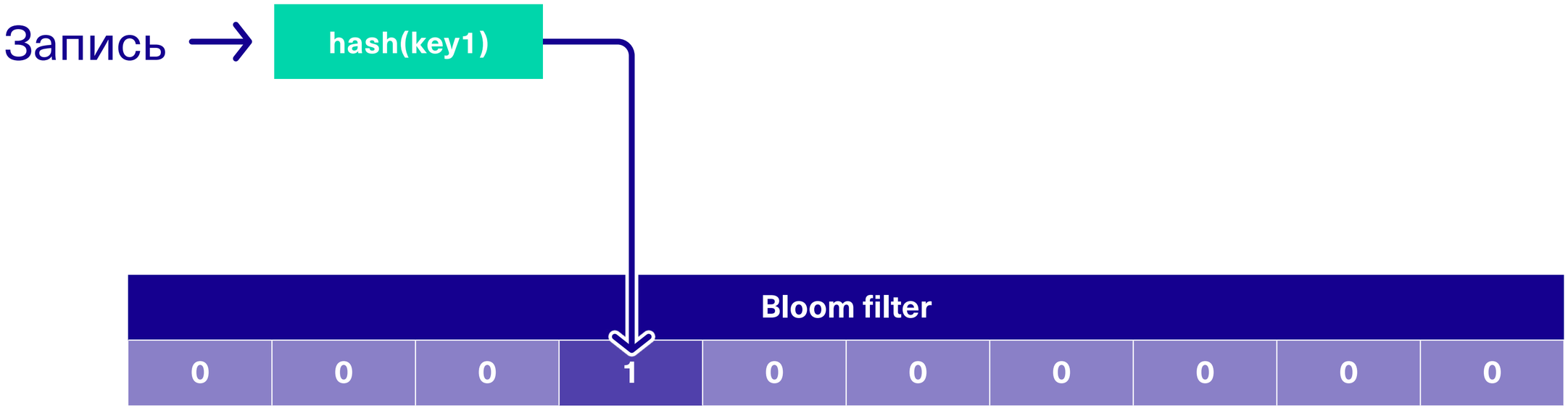
Bloom filter

Bloom filter

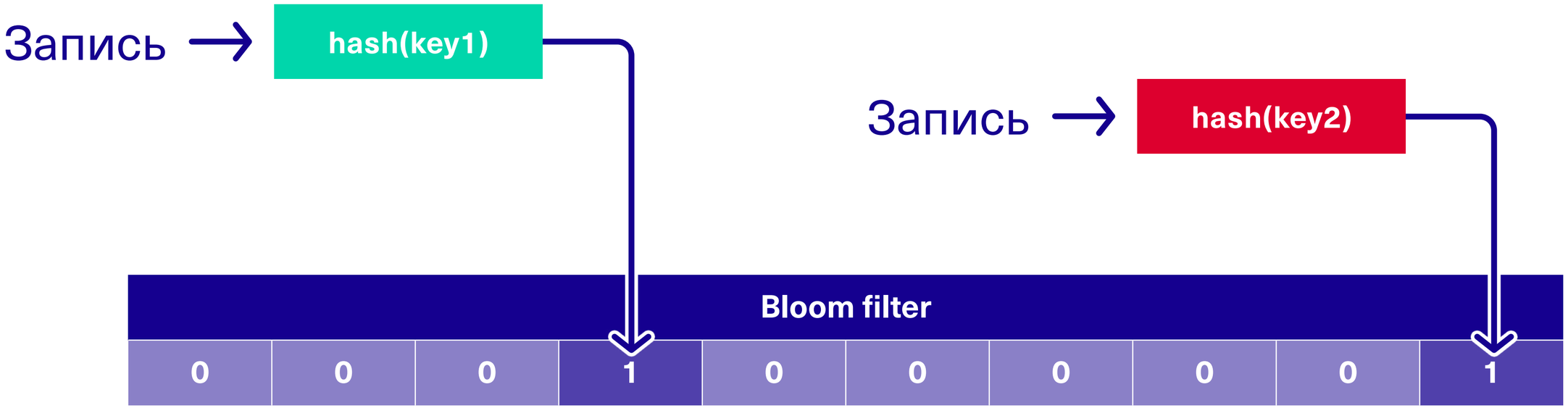


Bloom filter говорит **«ключа точно нет»** или **«ключ возможно присутствует»**

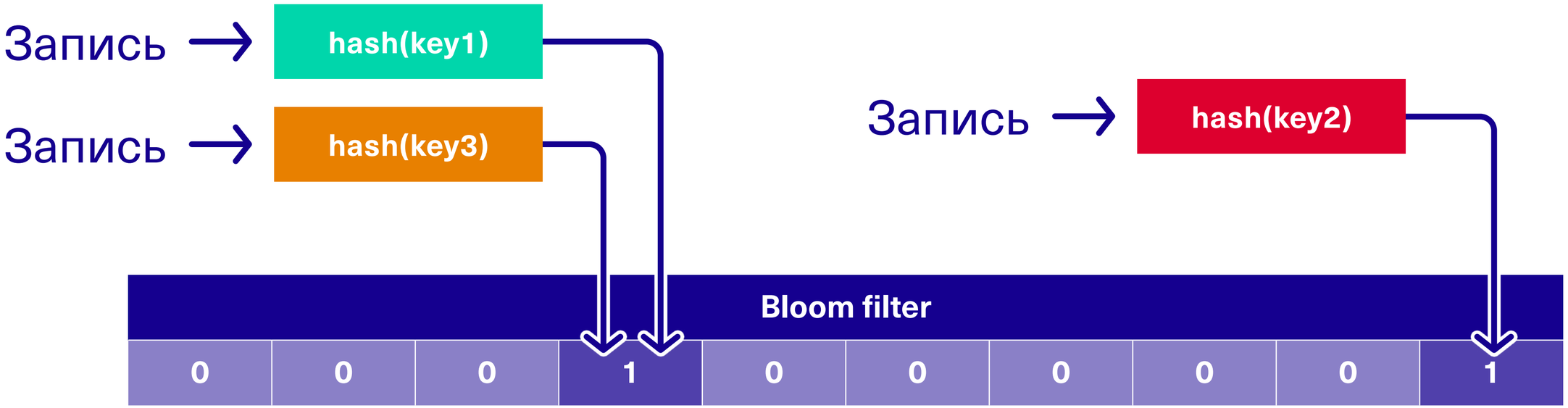
Bloom filter



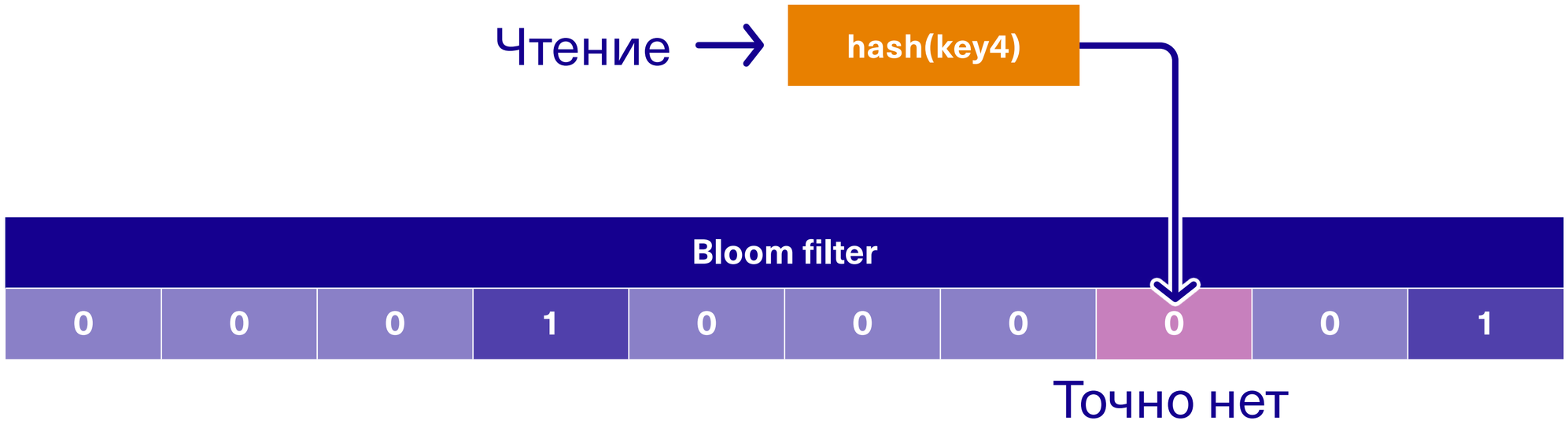
Bloom filter



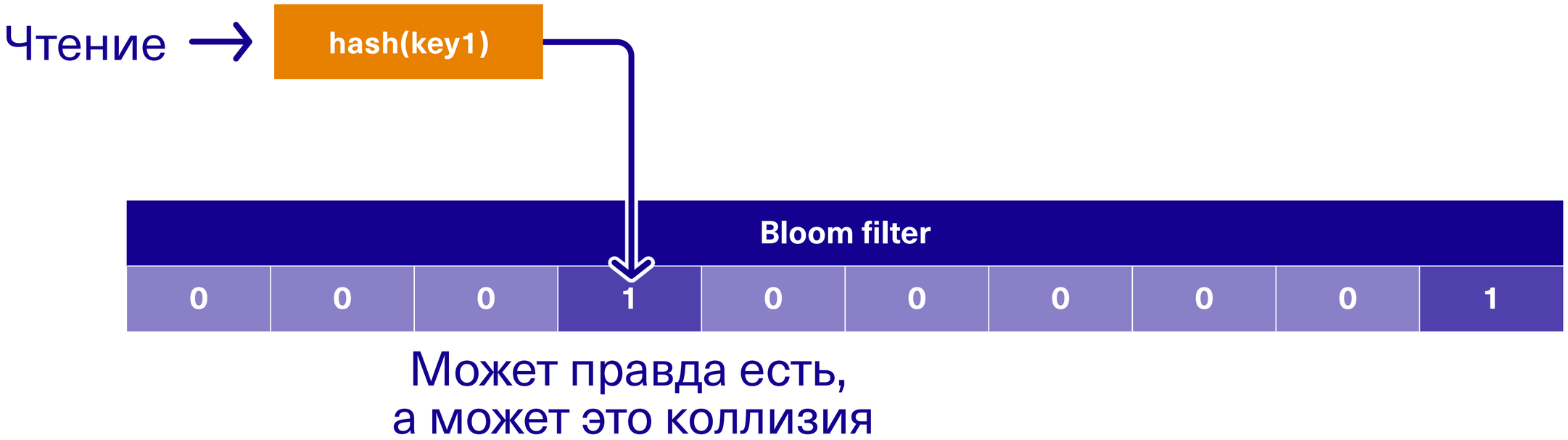
Bloom filter



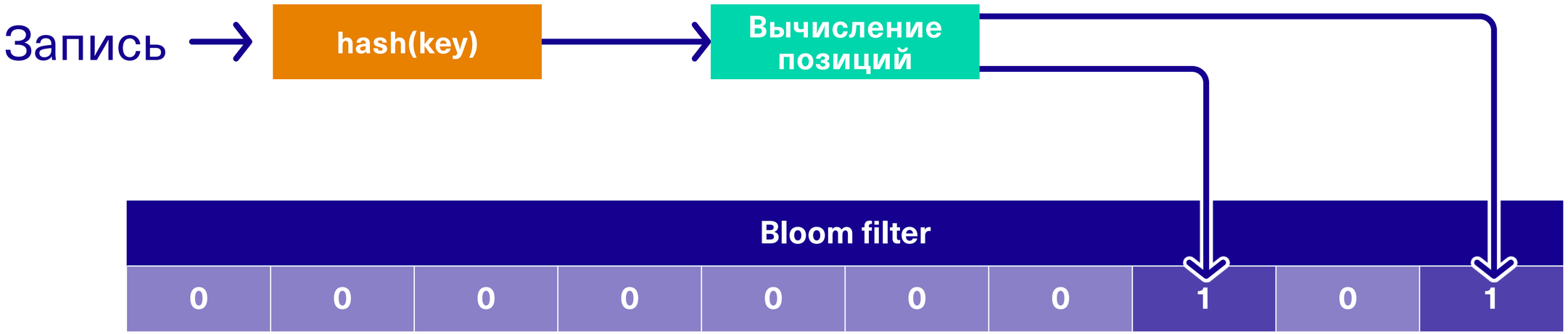
Bloom filter



Bloom filter

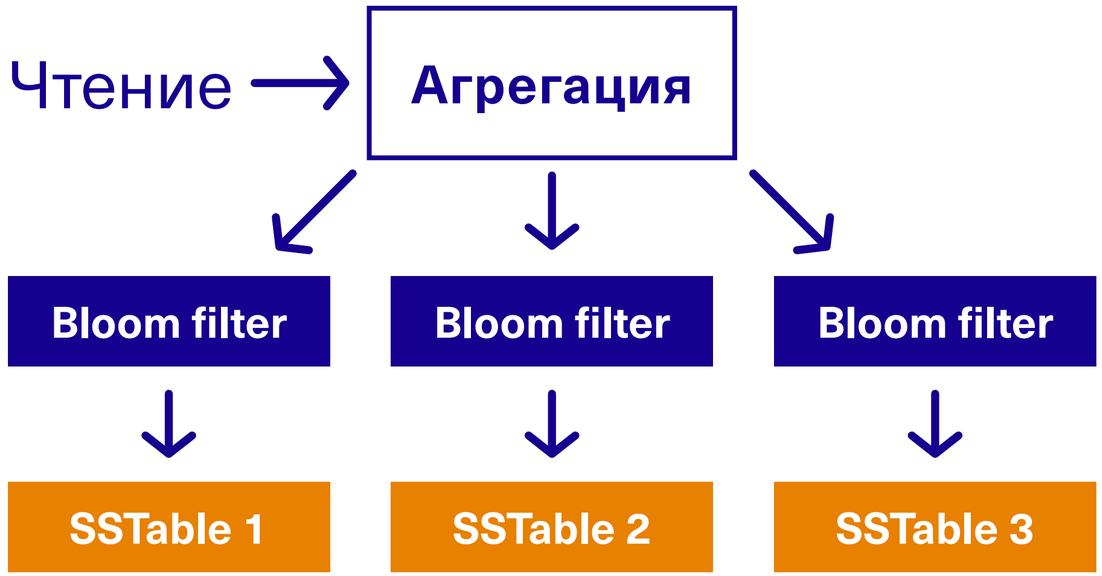


Bloom filter



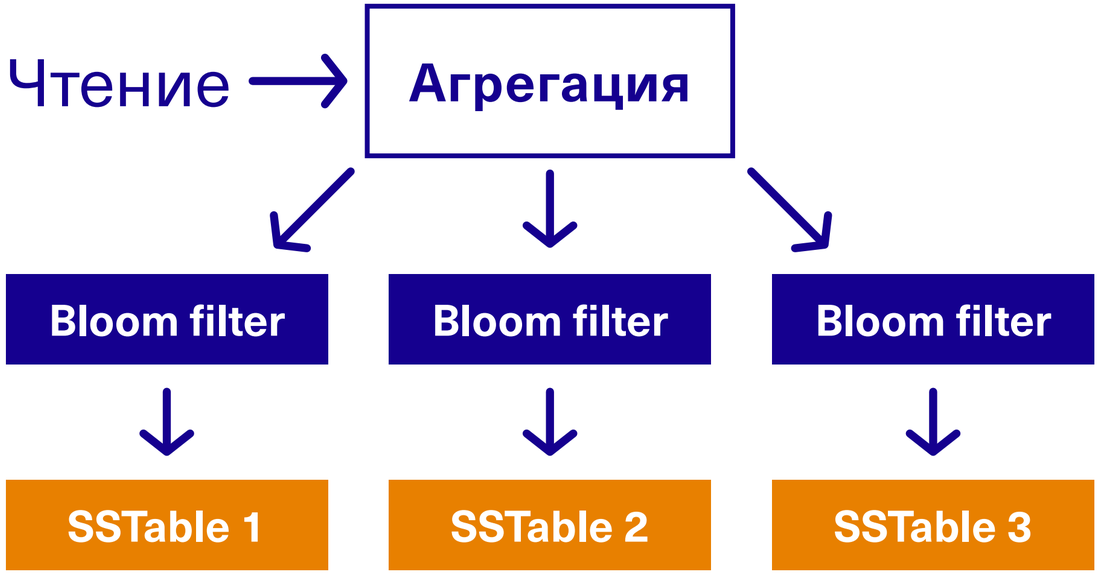
Bloom filter

bloom_filter_fp_chance — вероятность ошибки (от 0 до 1, рекомендуется от 0.01 до 0.1)



Bloom filter

bloom_filter_fp_chance — вероятность ошибки (от 0 до 1, рекомендуется от 0.01 до 0.1)




```
nodetool tablestats conf.timeline
 ...
Bloom filter false positives: 16721
Bloom filter false ratio: 0.0390
Bloom filter space used: 9392731
```

Minor compaction

Много обновляем и читаем, значит compaction должен помочь

Minor compaction

Много обновляем и читаем, значит compaction должен помочь

Minor compaction

Много обновляем и читаем, значит compaction должен помочь

SSTable 1 SSTable 3

Minor compaction

Много обновляем и читаем, значит compaction должен помочь

Minor compaction

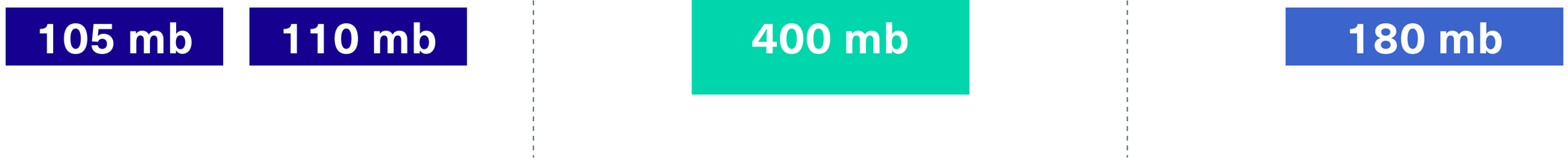
Много обновляем и читаем, значит compaction должен помочь

Size-Tiered Compaction https://youtu.be/TyTXOjFMi7k?si=Hsid0evoaOzkEy7o

Leveled Compaction https://youtu.be/6yJEwqseMY4?si=VaOEeNLFm_IbFcoR

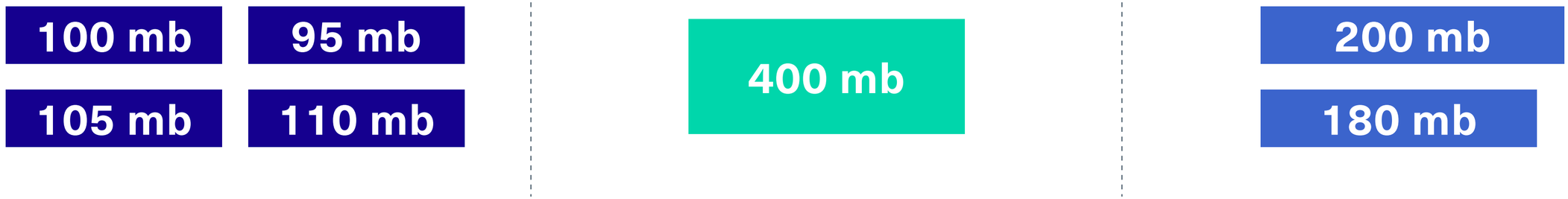
Time Window Compaction https://youtu.be/P8jPlppa_8I?si=bVw8EVt3puEVn-MP

Size Tiered Compaction Strategy (STCS)



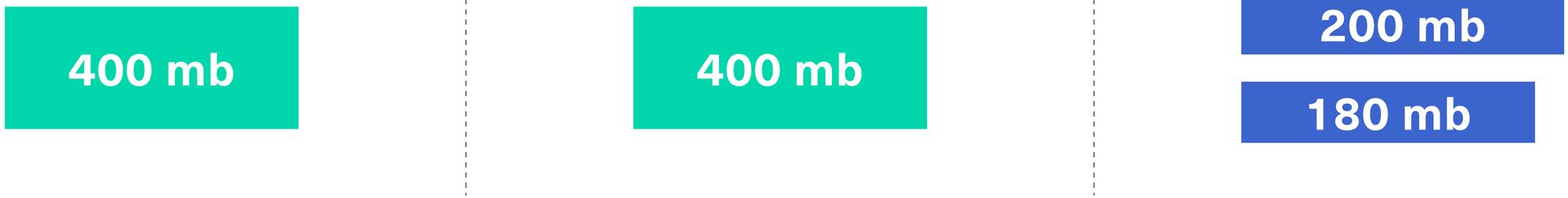
- Стратегия по умолчанию
- Объединяет близкие по размеру SSTable в одну
- Объединяет когда набралось нужное количество SSTable
- Оптимальна при активной записи (и если другие стратегии не подходят)

Size Tiered Compaction Strategy (STCS)



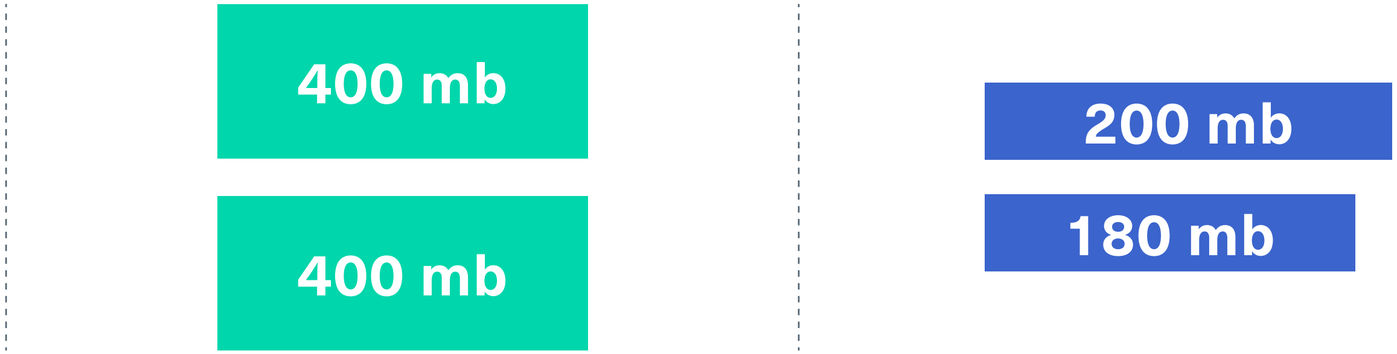
- Стратегия по умолчанию
- Объединяет близкие по размеру SSTable в одну
- Объединяет когда набралось нужное количество SSTable
- Оптимальна при активной записи (и если другие стратегии не подходят)

Size Tiered Compaction Strategy (STCS)



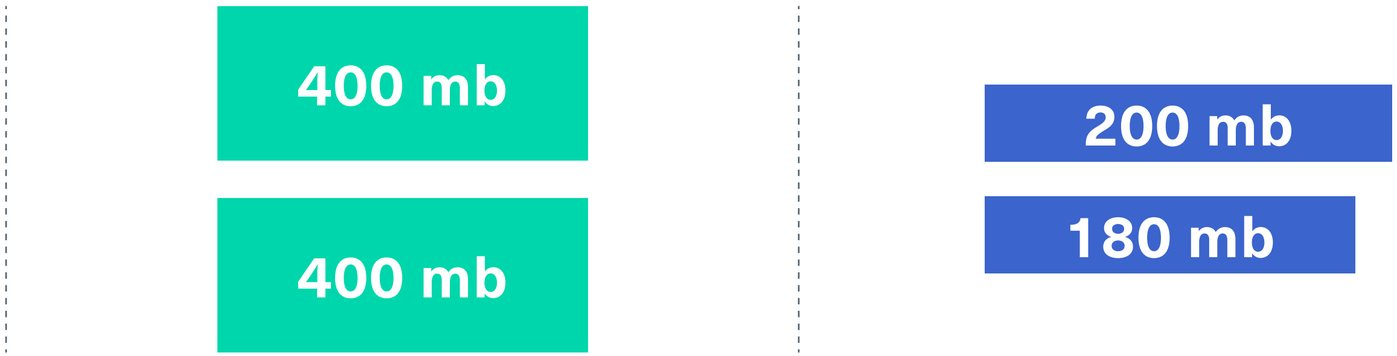
- Стратегия по умолчанию
- Объединяет близкие по размеру SSTable в одну
- Объединяет когда набралось нужное количество SSTable
- Оптимальна при активной записи (и если другие стратегии не подходят)

Size Tiered Compaction Strategy (STCS)



- Стратегия по умолчанию
- Объединяет близкие по размеру SSTable в одну
- Объединяет когда набралось нужное количество SSTable
- Оптимальна при активной записи (и если другие стратегии не подходят)

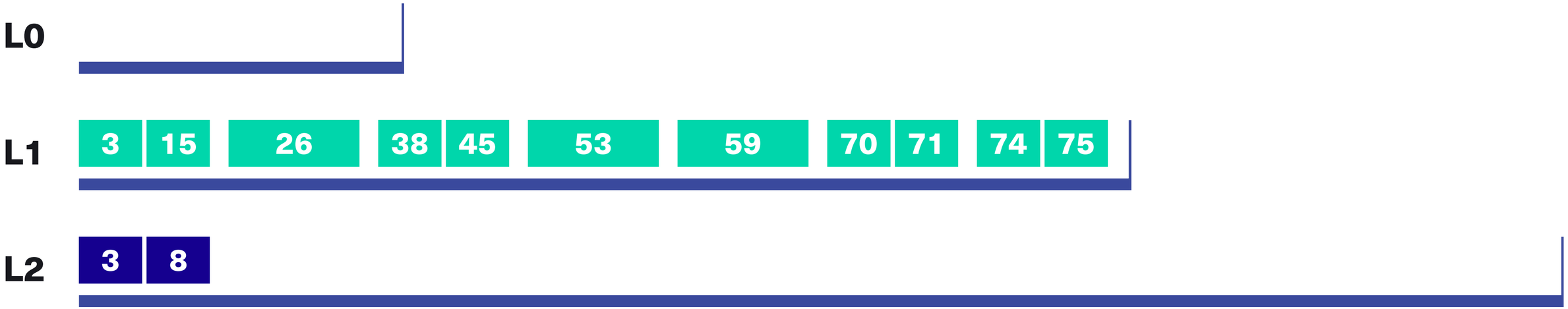
Size Tiered Compaction Strategy (STCS)



- Стратегия по умолчанию
- Объединяет близкие по размеру SSTable в одну
- Объединяет когда набралось нужное количество SSTable
- Оптимальна при активной записи (и если другие стратегии не подходят)

```
ALTER TABLE tablename WITH compaction =
  {‘class’ : ‘SizeTieredCompactionStrategy’}
```

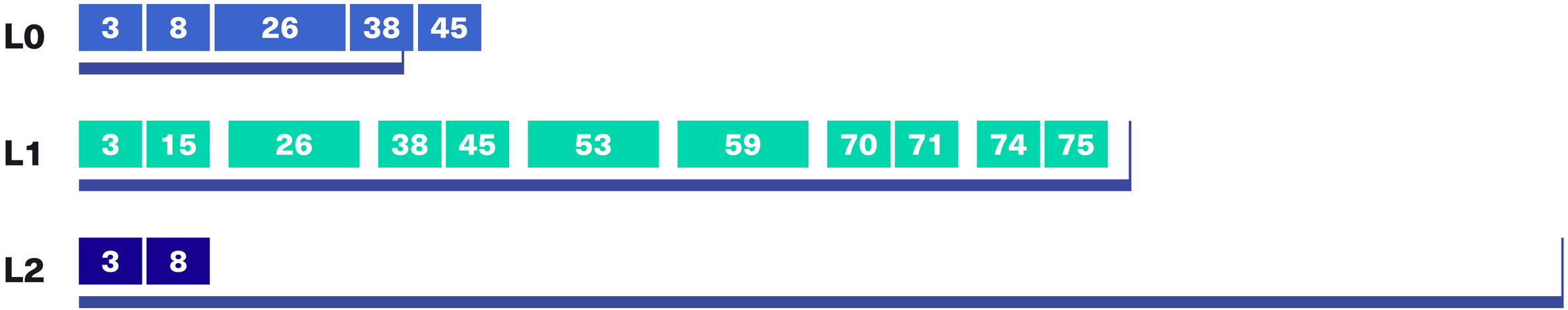
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

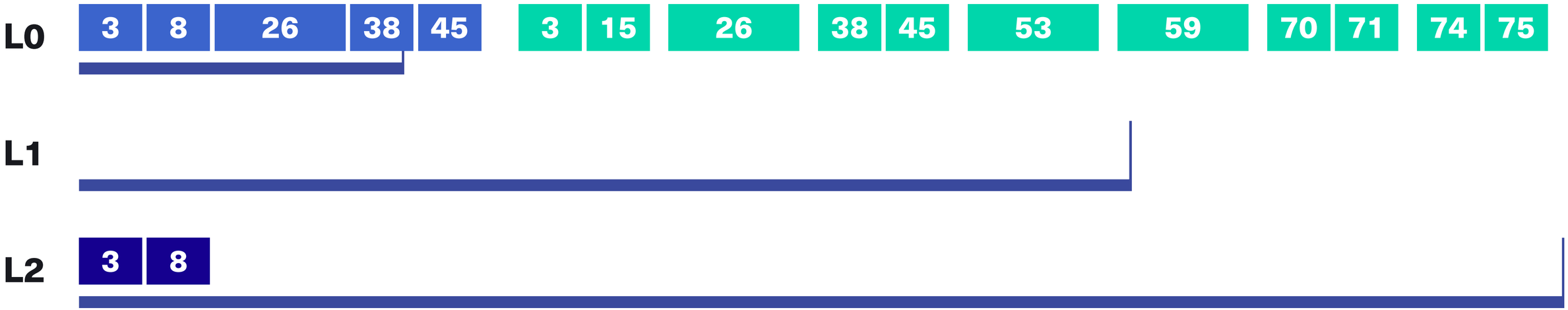
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

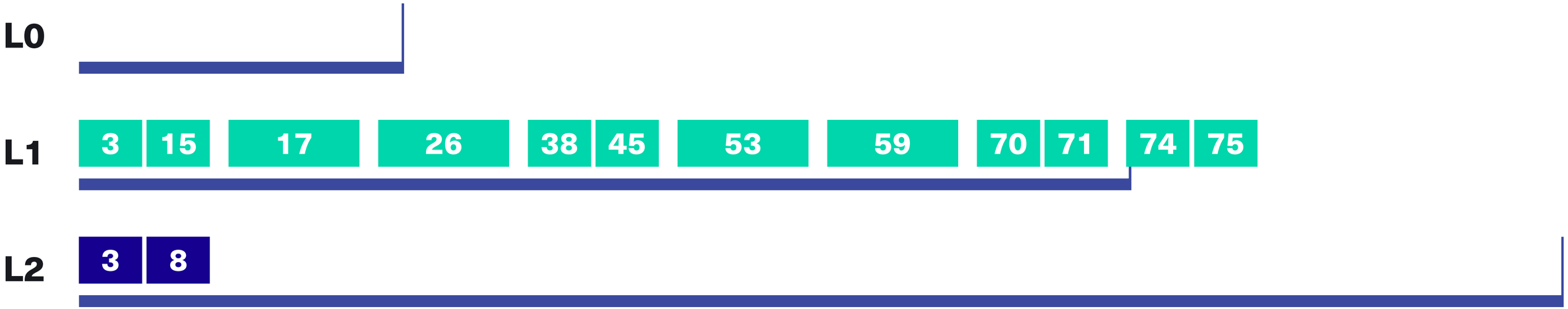
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

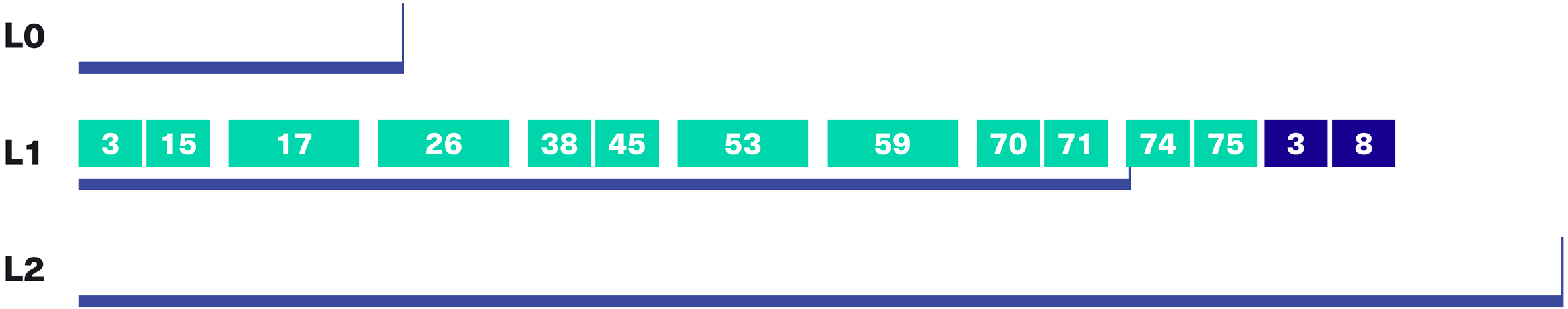
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

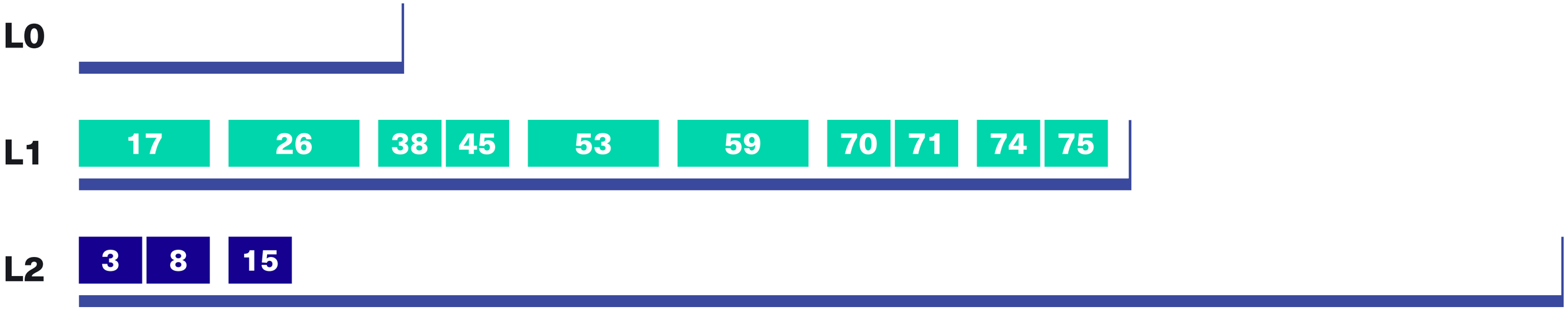
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

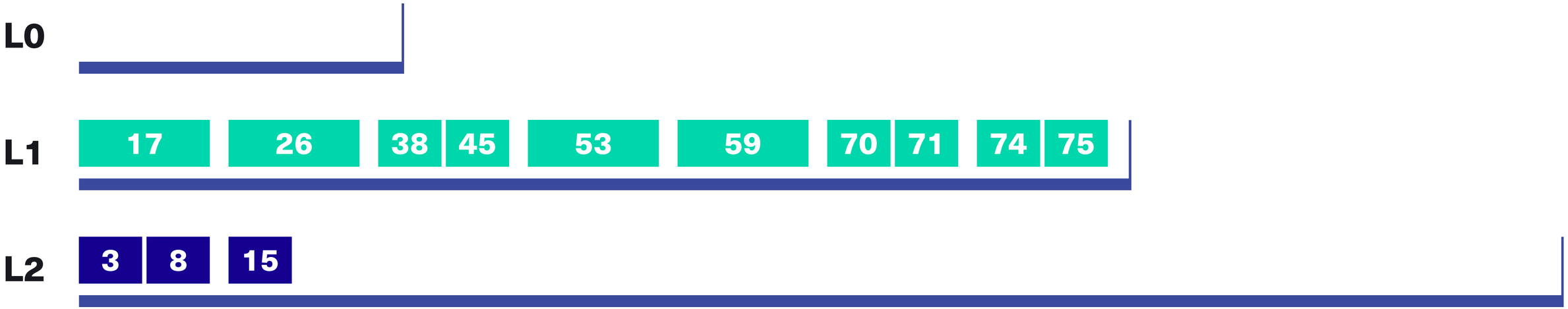
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

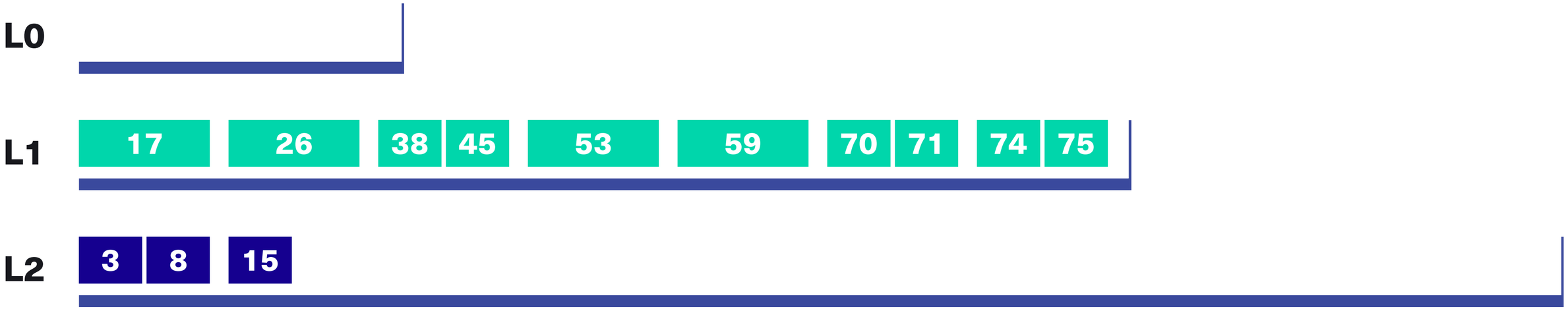
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет
- Оптимальна при интенсивных чтениях обновляемых данных

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

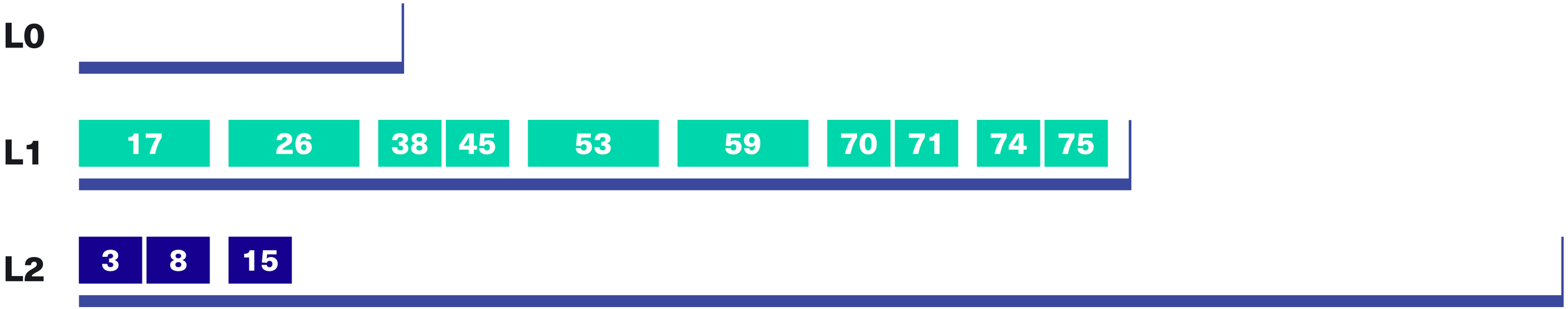
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет
- Оптимальна при интенсивных чтениях обновляемых данных
- Нагружает IO

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

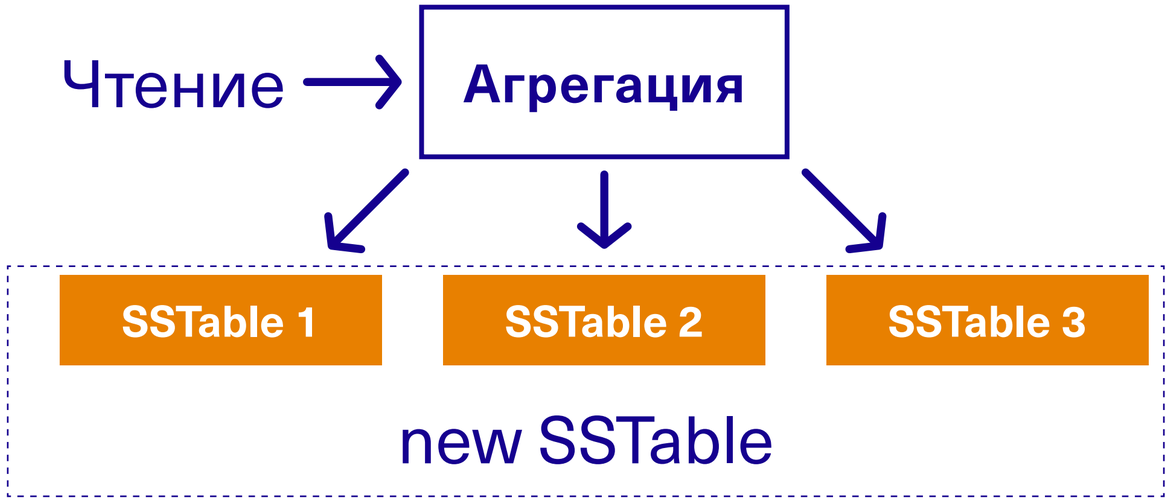
Leveled Compaction Strategy (LCS)



- Каждый уровень в 10 раз больше предыдущего
- Дублирования ключей внутри уровня нет
- Оптимальна при интенсивных чтениях обновляемых данных
- Нагружает IO — в нашем кейсе нет запаса IO

```
ALTER TABLE tablename WITH
  compaction = { ‘class’ :  ‘LeveledCompactionStrategy’}
```

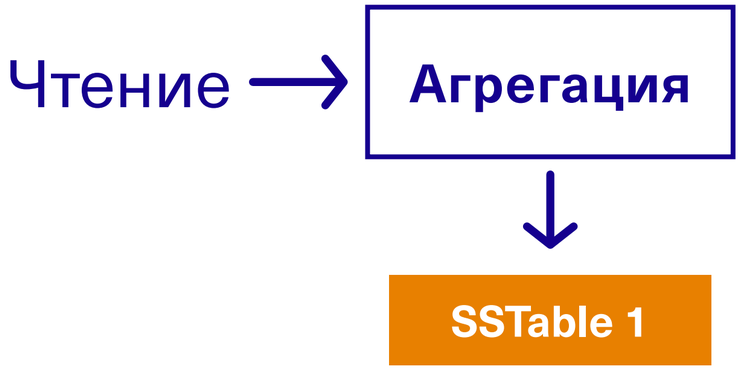
Major compaction



- Мёржит все SSTable в одну
- Дублирования ключа (устаревших версий данных) нет
- Запускается по команде

```
nodetool compact keyspace table
```

Major compaction



- Мёржит все SSTable в одну
- Дублирования ключа (устаревших версий данных) нет
- Запускается по команде

```
nodetool compact keyspace table
```

Major compaction

Небезопасное решение:

- Большая нагрузка на CPU, IO
- Требует много места на диске

WARNING

Major compaction

Небезопасное решение:

- Большая нагрузка на CPU, IO
- Требует много места на диске
- Отрабатывает долго

WARNING

Major compaction

Небезопасное решение:

- Большая нагрузка на CPU, IO
- Требует много места на диске
- Отрабатывает долго
- Большой SSTable может мешать Size Tiered Compaction Strategy

WARNING

Major compaction

Небезопасное решение:

- Большая нагрузка на CPU, IO
- Требует много места на диске
- Отрабатывает долго
- Большой SSTable может мешать Size Tiered Compaction Strategy

WARNING

1 TB

Major compaction

Небезопасное решение:

- Большая нагрузка на CPU, IO
- Требует много места на диске
- Отрабатывает долго
- Большой SSTable может мешать Size Tiered Compaction Strategy

WARNING

Major compaction

Небезопасное решение:

- Большая нагрузка на CPU, IO
- Требует много места на диске
- Отрабатывает долго
- Большой SSTable может мешать Size Tiered Compaction Strategy

WARNING

100 mb | 95 mb | 105 mb | 110 mb

200 mb | 180 mb

1 TB

Major compaction

Небезопасное решение:

- Большая нагрузка на CPU, IO
- Требует много места на диске
- Отрабатывает долго
- Большой SSTable может мешать Size Tiered Compaction Strategy

WARNING

400 mb

200 mb

180 mb

1 TB

Выводы

Apache Cassandra полезный инструмент:

- Настраиваемые согласованность и доступность
- Высокая скорость записи из коробки

Выводы

Apache Cassandra полезный инструмент:

- Настраиваемые согласованность и доступность
- Высокая скорость записи из коробки

Необходимо понимать как она работает под капотом:

- У using timestamp есть нюансы
- Tombstone имеют свою цену

Выводы

Apache Cassandra полезный инструмент:

- Настраиваемые согласованность и доступность
- Высокая скорость записи из коробки

Необходимо понимать как она работает под капотом:

- У using timestamp есть нюансы
- Tombstone имеют свою цену

Нагрузку на IO можно снизить:

- Кэши
- Bloom filter
- Compaction strategies

ПОЧТА БАНК

Спасибо за внимание!