
Яндекс

Яндекс Облако

Распределённые транзакции в Yandex Database

Семён Чечеринда, ведущий разработчик

О чём я расскажу

01 | Какие распределённые транзакции нам нужны

02 | Как мы реализовали распределённые транзакции в YDB

03 | Как масштабируется наше решение

04 | Каков overhead у запросов в нашем решении

05 | Доступные уровни изоляции для транзакций

01

Какие распределённые транзакции нам нужны

Зачем?

- Высокие требования внутри Яндекса к отказоустойчивости сервисов
- Сервисы Яндекса должны переживать выход из строя одного из ДЦ
- Опыт разработки и эксплуатации своей кросс-ДЦ noSQL БД
- Eventual consistency не подходит под наши задачи

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
- Consistency
- Isolation
- Durability

ACID

- Atomicity
- Consistency
- Isolation
- Durability

Уровни изоляции

	Аномалии конкурентного доступа				
	Dirty read	Unrepeatable read	Lost updates	Phantoms	Write skew
Read uncommitted	✓	✓	✓	✓	✓
Read committed	✗	✓	✓	✓	✓
Repeatable read	✗	✗	✗	✓	✓
Snapshot isolation	✗	✗	✗	✗	✓
Serializable	✗	✗	✗	✗	✗

Strict Serializable

Serializable + Linearizable

Уровни изоляции

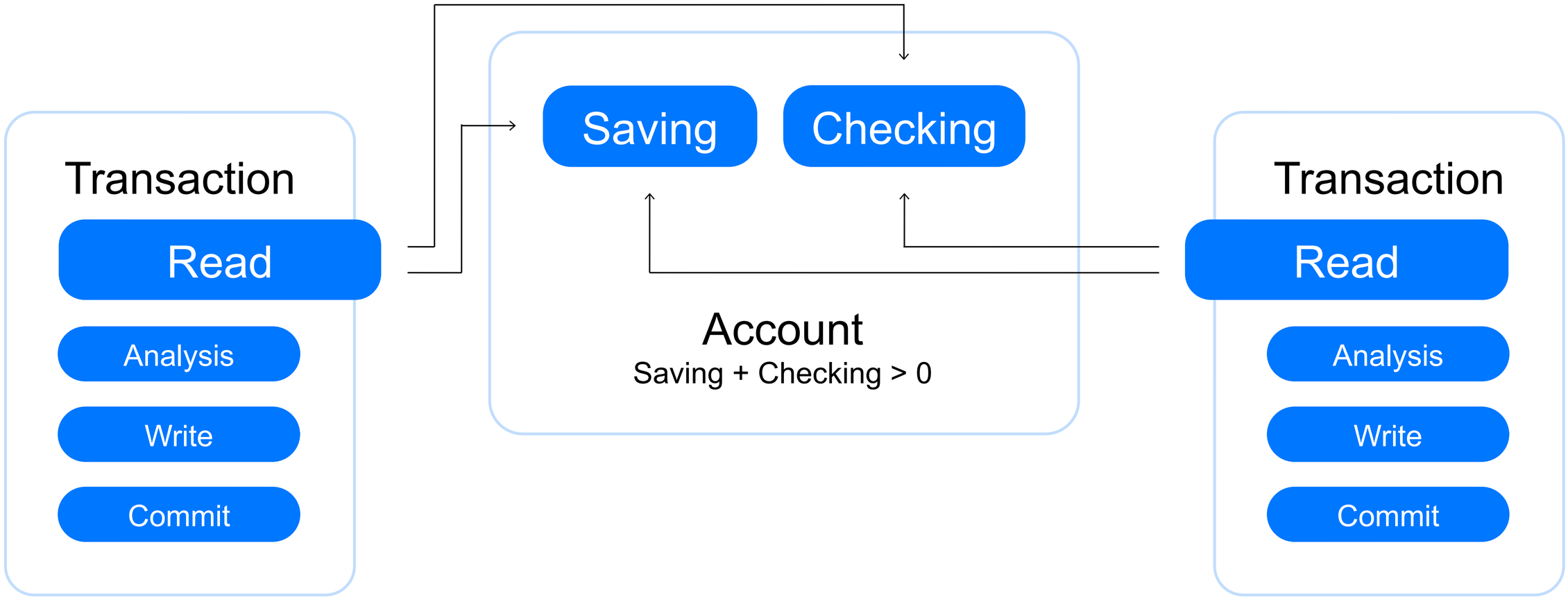
	Аномалии конкурентного доступа				
	Dirty read	Unrepeatable read	Lost updates	Phantoms	Write skew
Read uncommitted	✓	✓	✓	✓	✓
Read committed	✗	✓	✓	✓	✓
Repeatable read	✗	✗	✗	✓	✓
Snapshot isolation	✗	✗	✗	✗	✓
Serializable	✗	✗	✗	✗	✗
Strict Serializable	Serializable + Linearizable				

Уровни изоляции

	Аномалии конкурентного доступа				
	Dirty read	Unrepeatable read	Lost updates	Phantoms	Write skew
Read uncommitted	✓	✓	✓	✓	✓
Read committed	✗	✓	✓	✓	✓
Repeatable read	✗	✗	✗	✓	✓
Snapshot isolation	✗	✗	✗	✗	✓
Serializable	✗	✗	✗	✗	✗
Strict Serializable	Serializable + Linearizable				

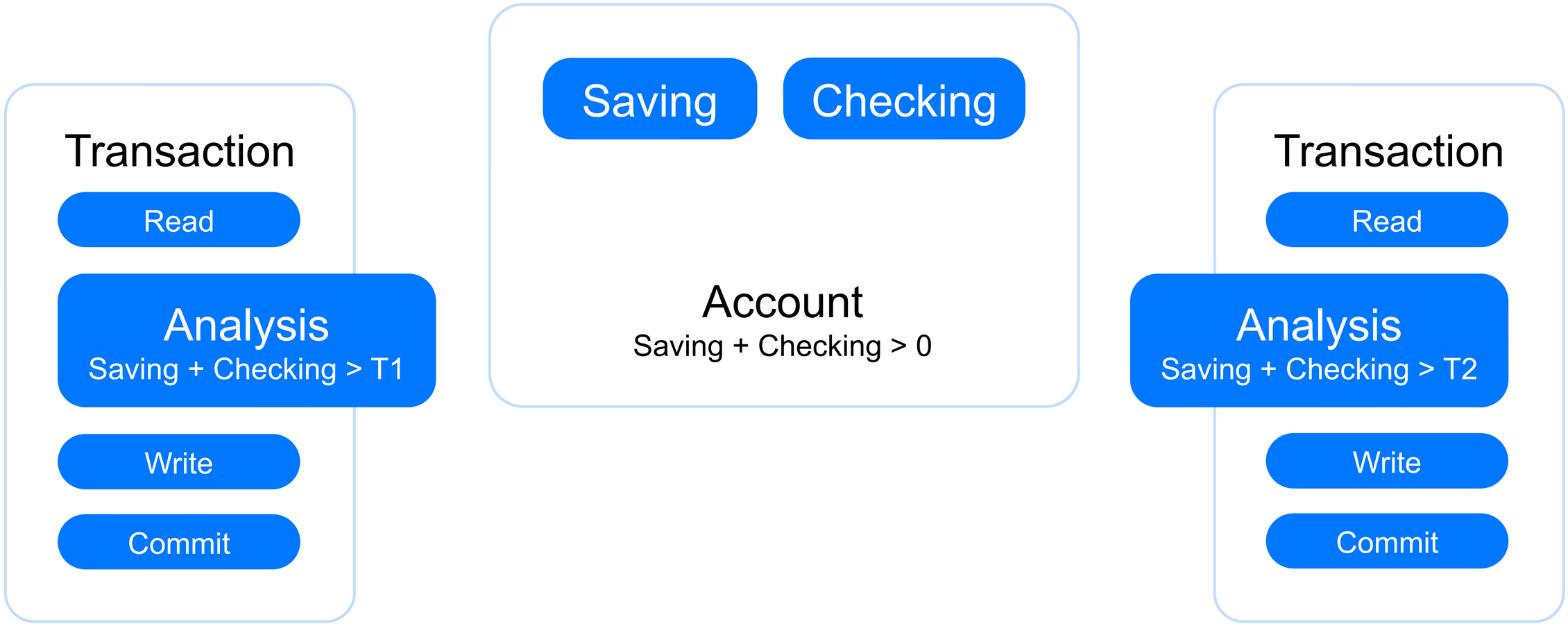
Пример аномального поведения

Write skews допустимая аномалия на уровне изоляции snapshot



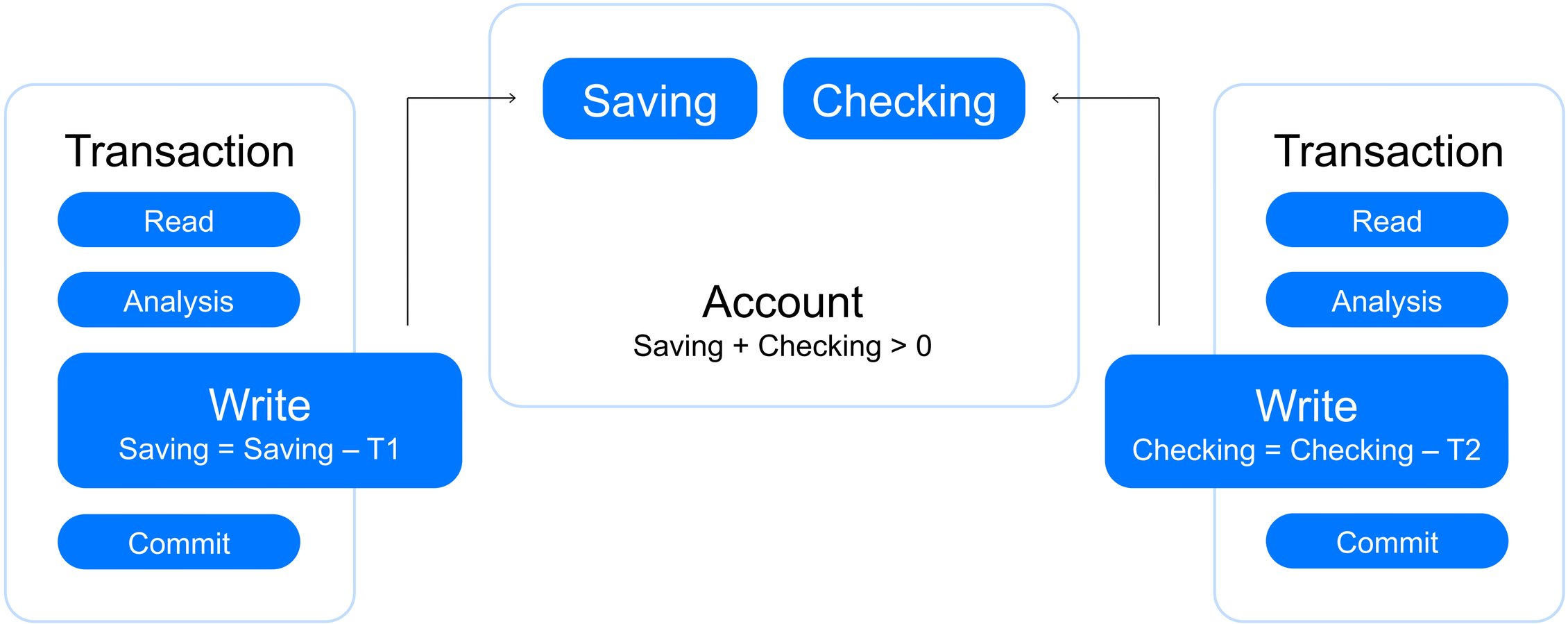
Пример аномального поведения

Write skews допустимая аномалия на уровне изоляции snapshot



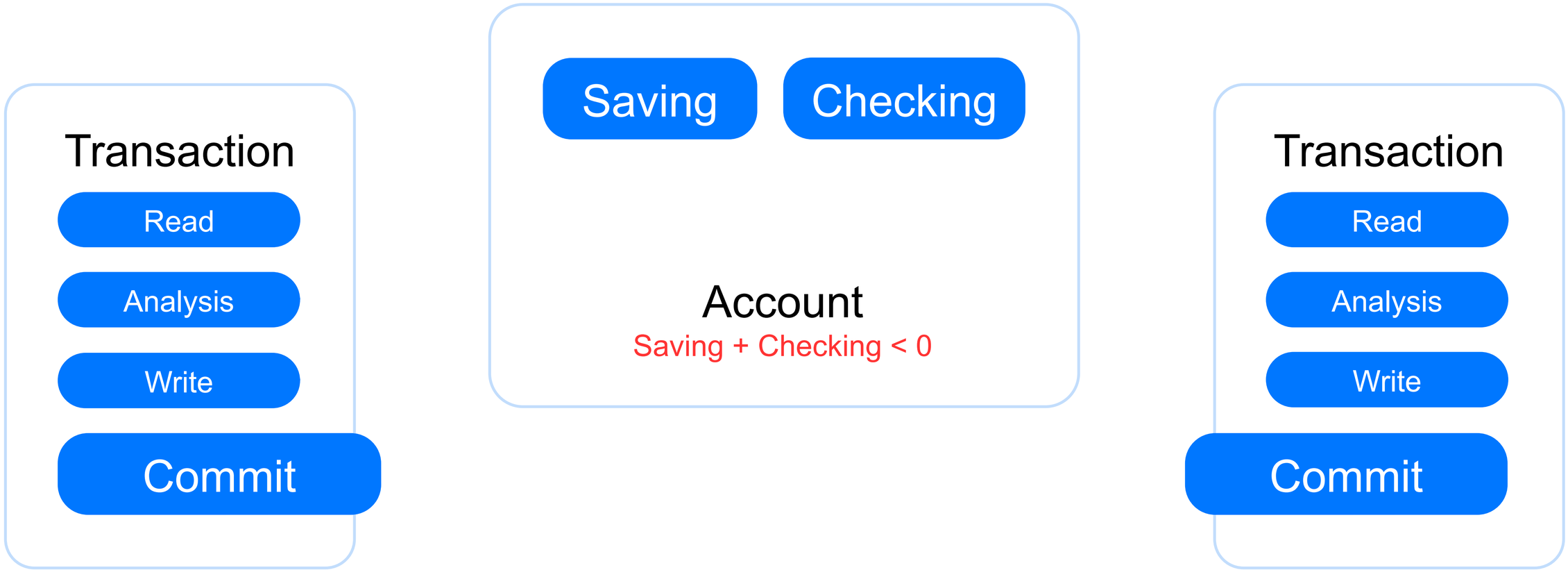
Пример аномального поведения

Write skews допустимая аномалия на уровне изоляции snapshot



Пример аномального поведения

Write skews допустимая аномалия на уровне изоляции snapshot



Пример аномального поведения

Snapshot isolation

- Обе транзакции выполняются
- Анализ данных в каждой из транзакций не покажет нарушение инварианта
- Инвариант нарушится после конкурентного выполнения транзакций

Serializable isolation

- Транзакции выполняются последовательно
- Анализ данных в каждой из транзакций будет корректным
- Инвариант не нарушится

Yandex Database (YDB)

- Надёжное хранение данных с избыточностью и автоматической репликацией
- Отказоустойчивость, автоматическое восстановление от сбоев
- Распределённые ACID-транзакции с serializable-уровнем изоляции транзакций
- Горизонтальная масштабируемость до тысяч нод

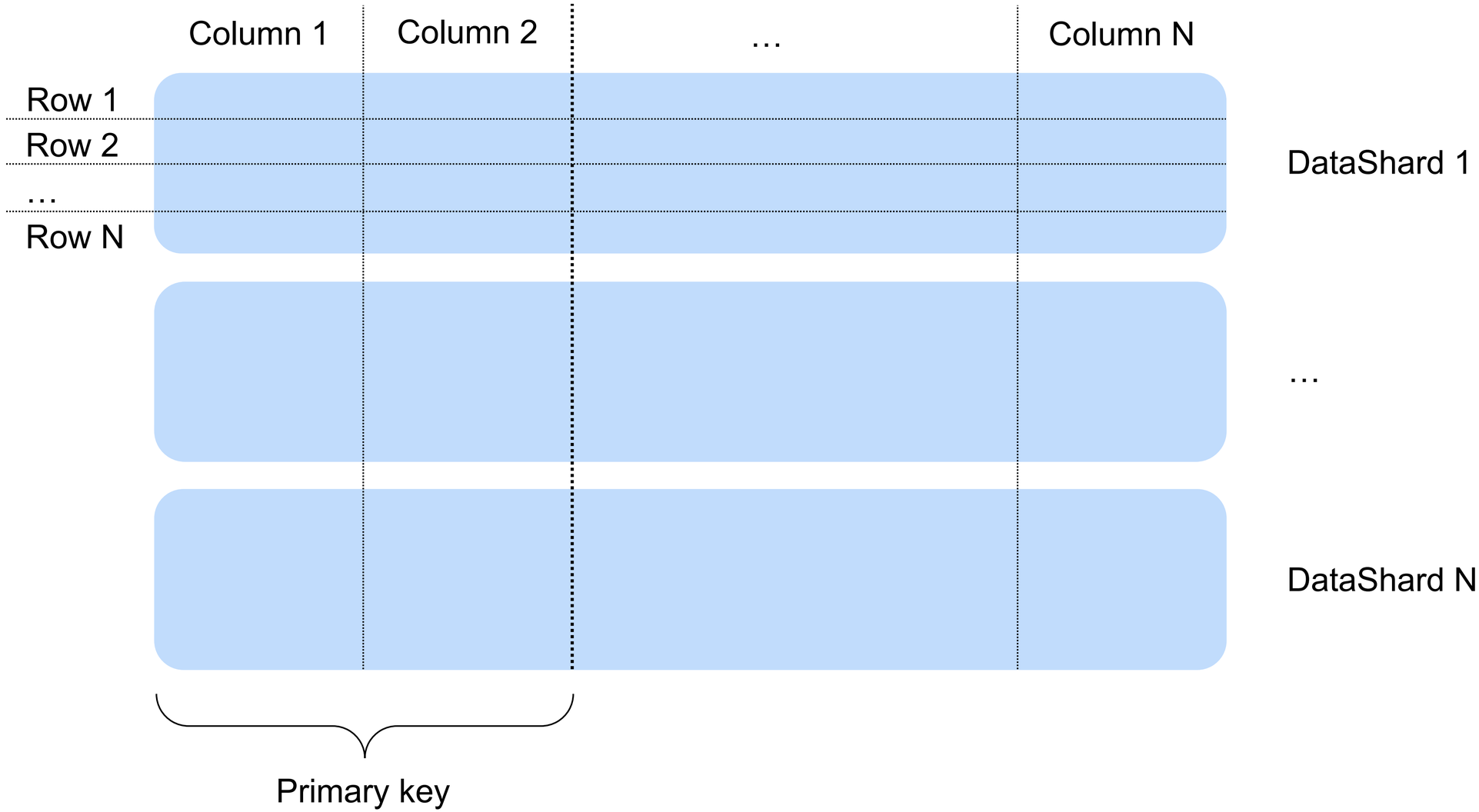
02

Как мы реализовали распределённые транзакции в YDB

Как мы реализовали распределённые транзакции в YDB

- Кто участвует в распределённых транзакциях
- Свойства и ограничения атомарного распределённого запроса
- Конкурентное выполнение распределённых транзакций

Таблицы


Column 1
Column 2
…
Column N
Row 1
Row 2
…
Row N
DataShard 1
…
DataShard N
Primary key

YDB Tablet

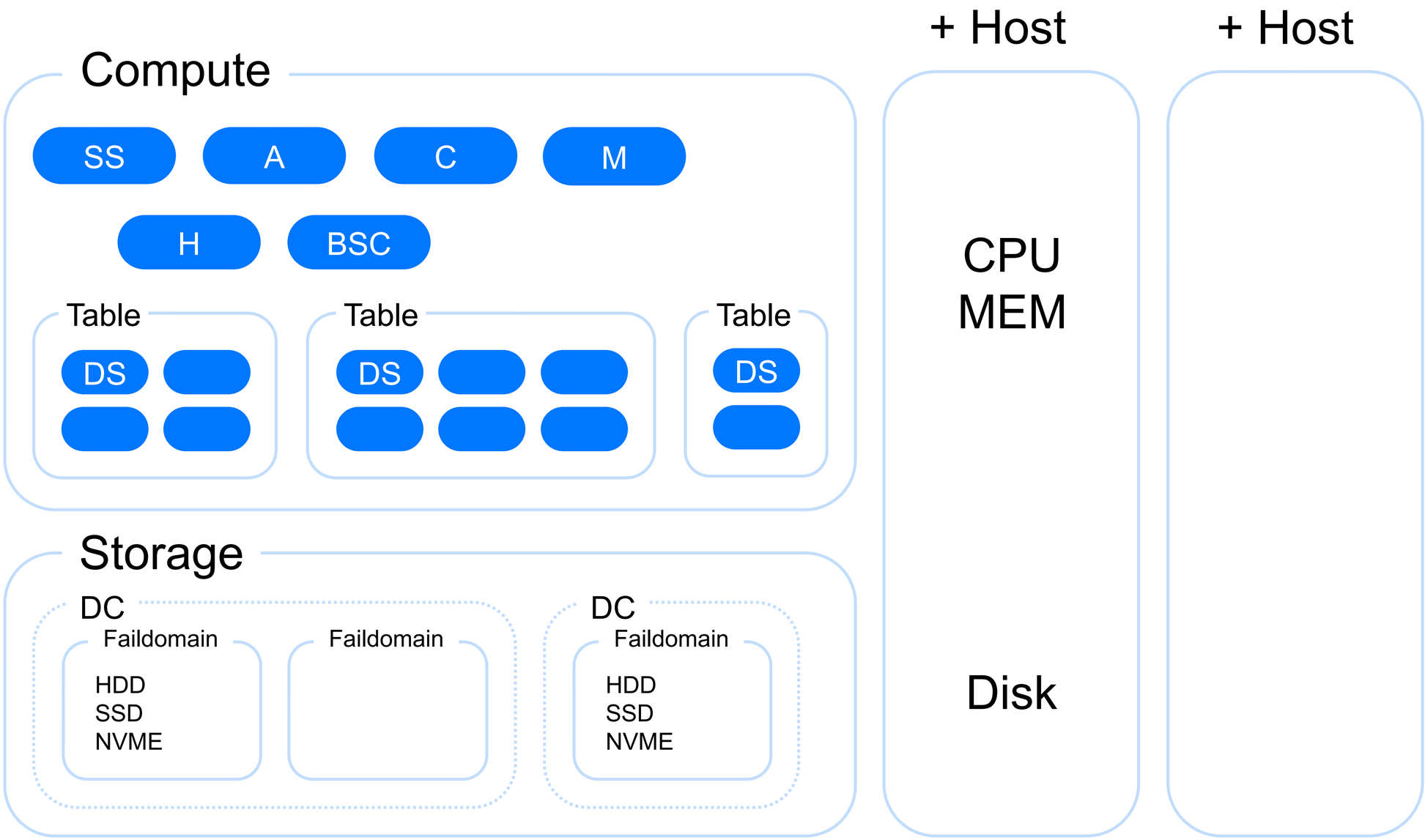
- Является актором, но имеет персистентное состояние
- Работает однопоточно
- Хранит свое состояние и данные в Distributed Storage
- Единица отказоустойчивости системы
- Всегда работает только один экземпляр
- Поднимается на любой ноде кластера
- На кластере миллионы разных экземпляров tablet

Tablet — универсальный строительный блок системы

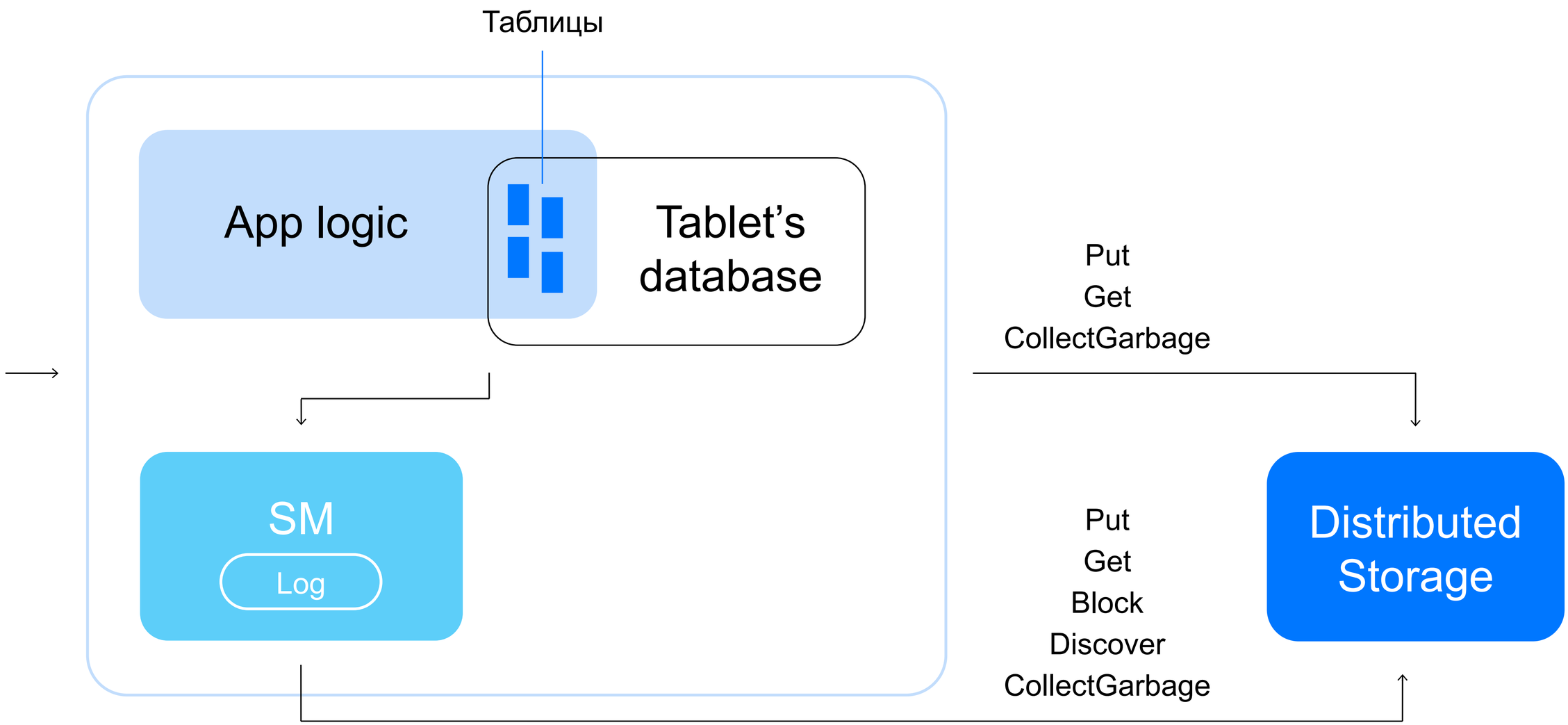
С использованием tablet построены разные логические сущности

- Пользовательские таблицы
- Очереди
- Хранилище time-series данных мониторинга
- Системные tablet, необходимые для работы YDB
- Сетевые диски облака

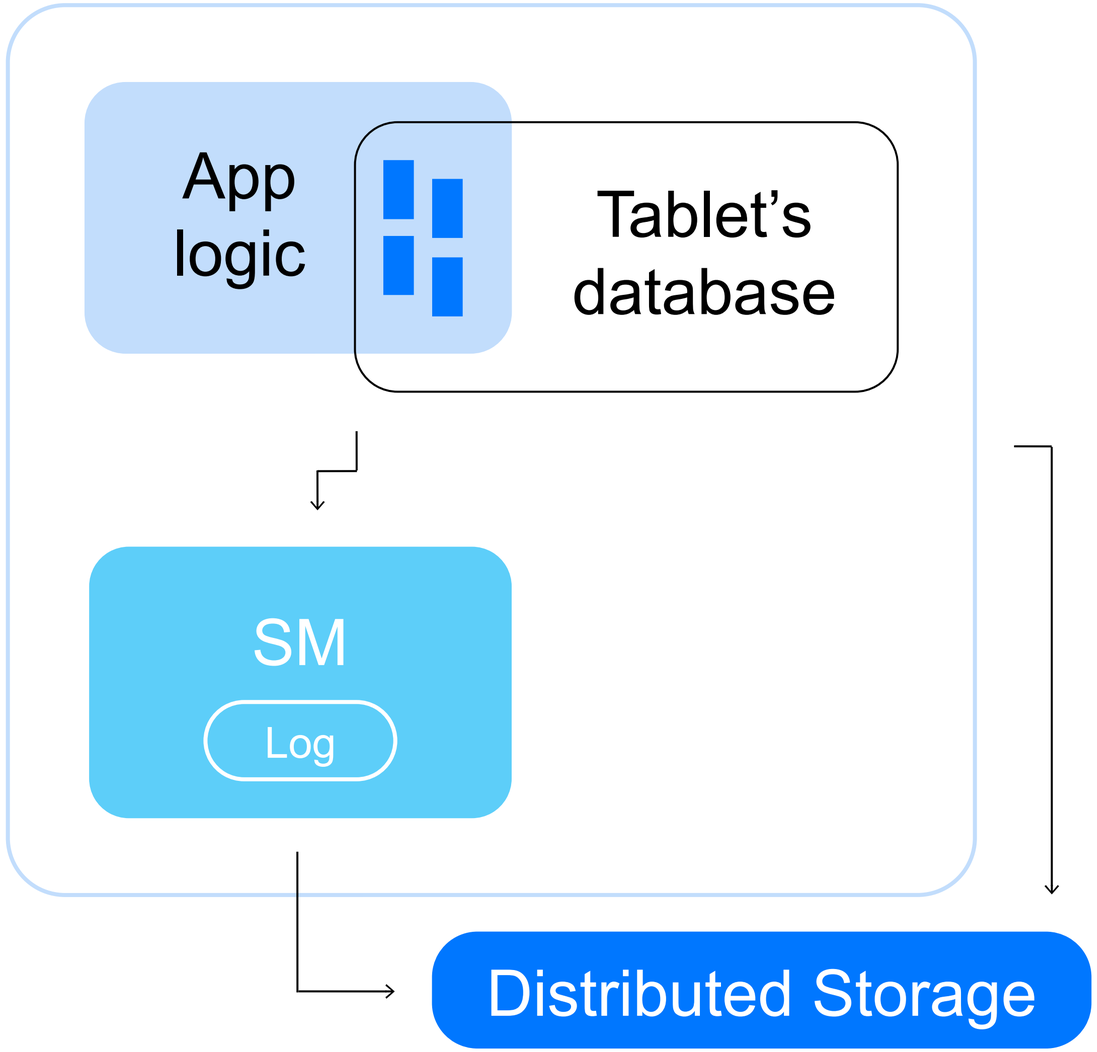
Overview


+ Host
+ Host
Compute
SS
A
C
M
H
BSC
Table
DS
Table
DS
Table
DS
Storage
DC
Faildomain
HDD
SSD
NVME
Faildomain
DC
Faildomain
HDD
SSD
NVME
CPU
MEM
Disk

YDB Tablet

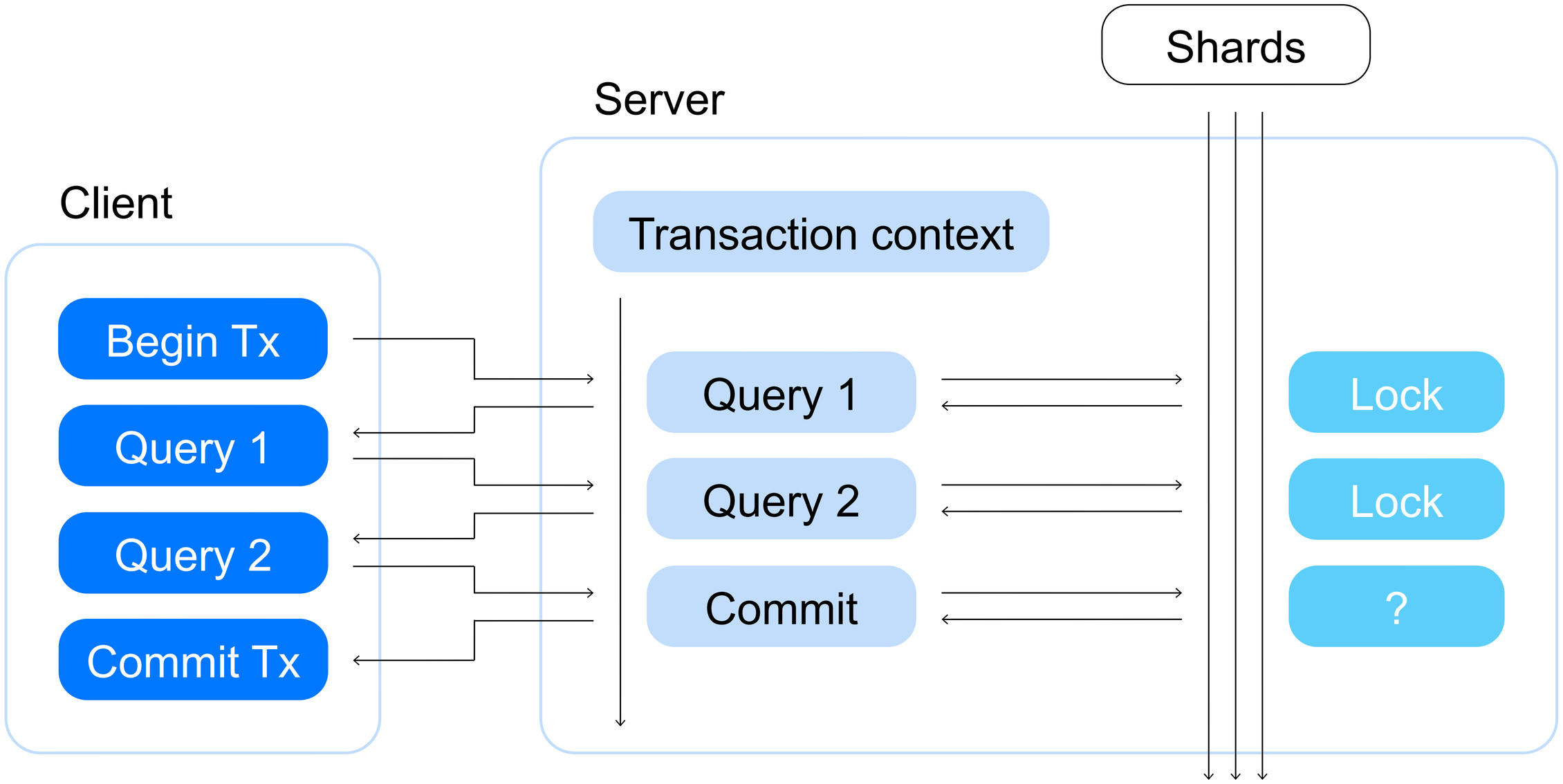

Таблицы
App logic
Tablet's database
SM
Log
Put
Get
CollectGarbage
Put
Get
Block
Discover
CollectGarbage
Distributed Storage

Datashard как tablet



- Надёжность и доступность — это свойства таблетки
- Транзакционное изменение своих данных
  - Однопоточная, strict serializable
  - Одношардовые запросы
- Примитив масштабируемости таблицы
  - Количество шардов в таблице меняется в зависимости от её размера и нагрузки

Общий вид


Shards
Server
Client
Transaction context
Begin Tx
Query 1
Query 2
Commit Tx
Query 1
Query 2
Commit
Lock
Lock
?

Атомарный распределённый запрос

- Атомарная операция над шардами
- Выполнение операции детерминировано
- Известны read / write set’ы ключей операции

Конкурентное выполнение

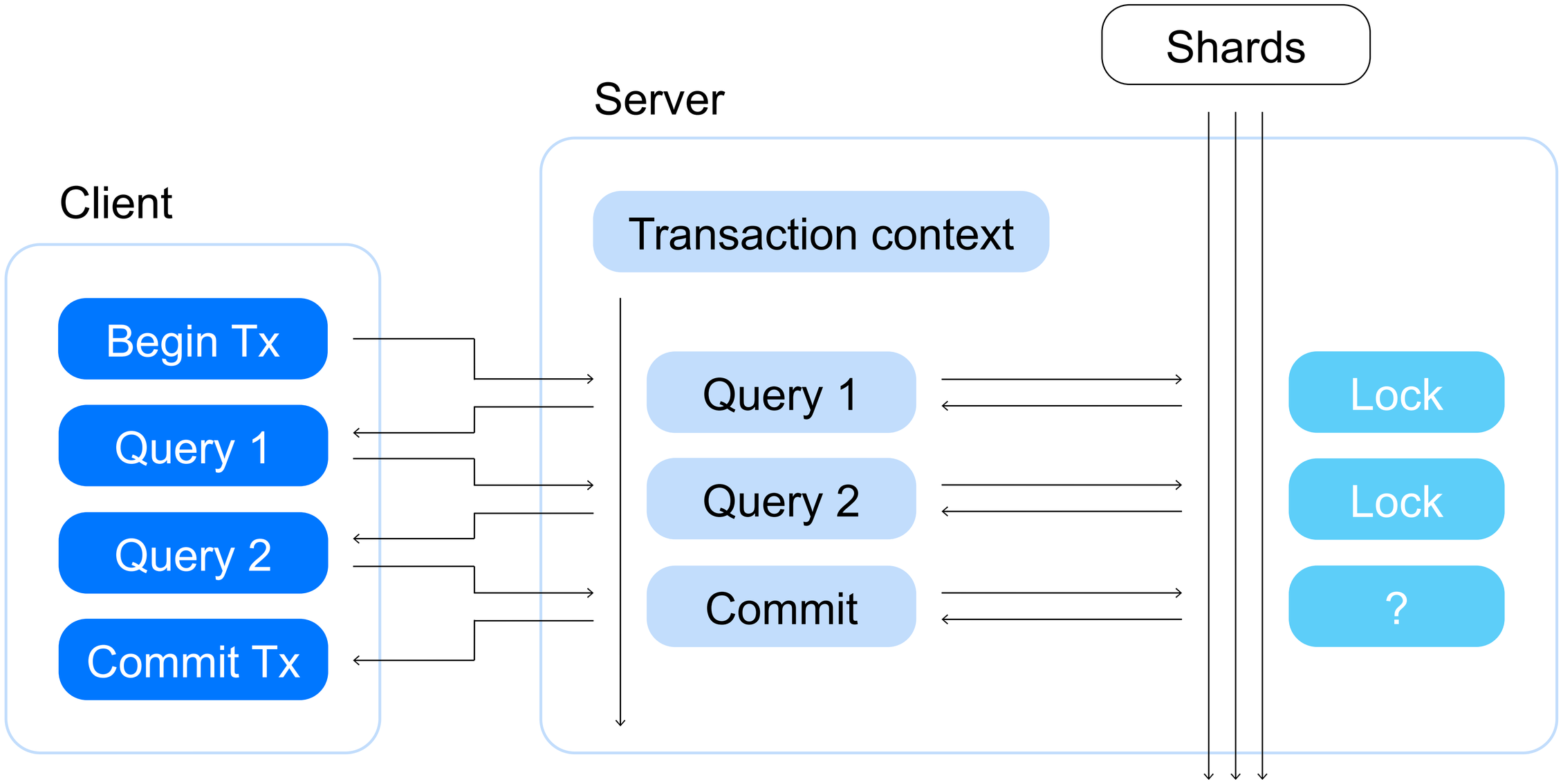
Calvin: Fast Distributed Transactions for Partitioned Database Systems by Daniel J. Abadi, Alexander Thomson

**Если каждая транзакция выполняется детерминировано,
то их упорядоченное выполнение также детерминировано**

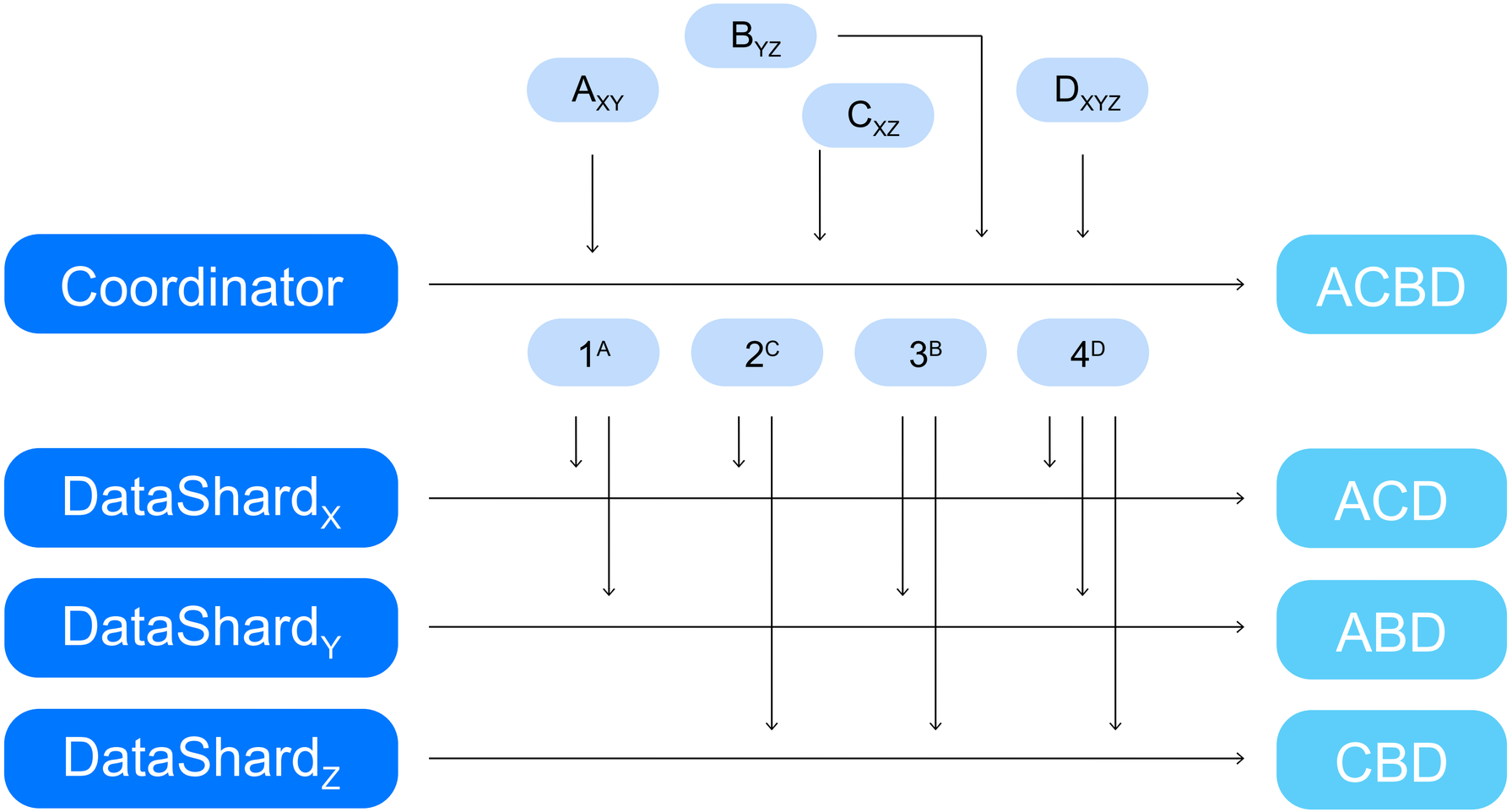
Детерминированное выполнение распределённых транзакций

- Определить глобальный порядок выполнения транзакций
- Выполнить на шардах релевантные им транзакции
в соответствии с их глобальным порядком выполнения

Общий вид


Shards
Server
Client
Transaction context
Begin Tx
Query 1
Query 2
Commit Tx
Query 1
Query 2
Commit
Lock
Lock
?

Координация транзакции

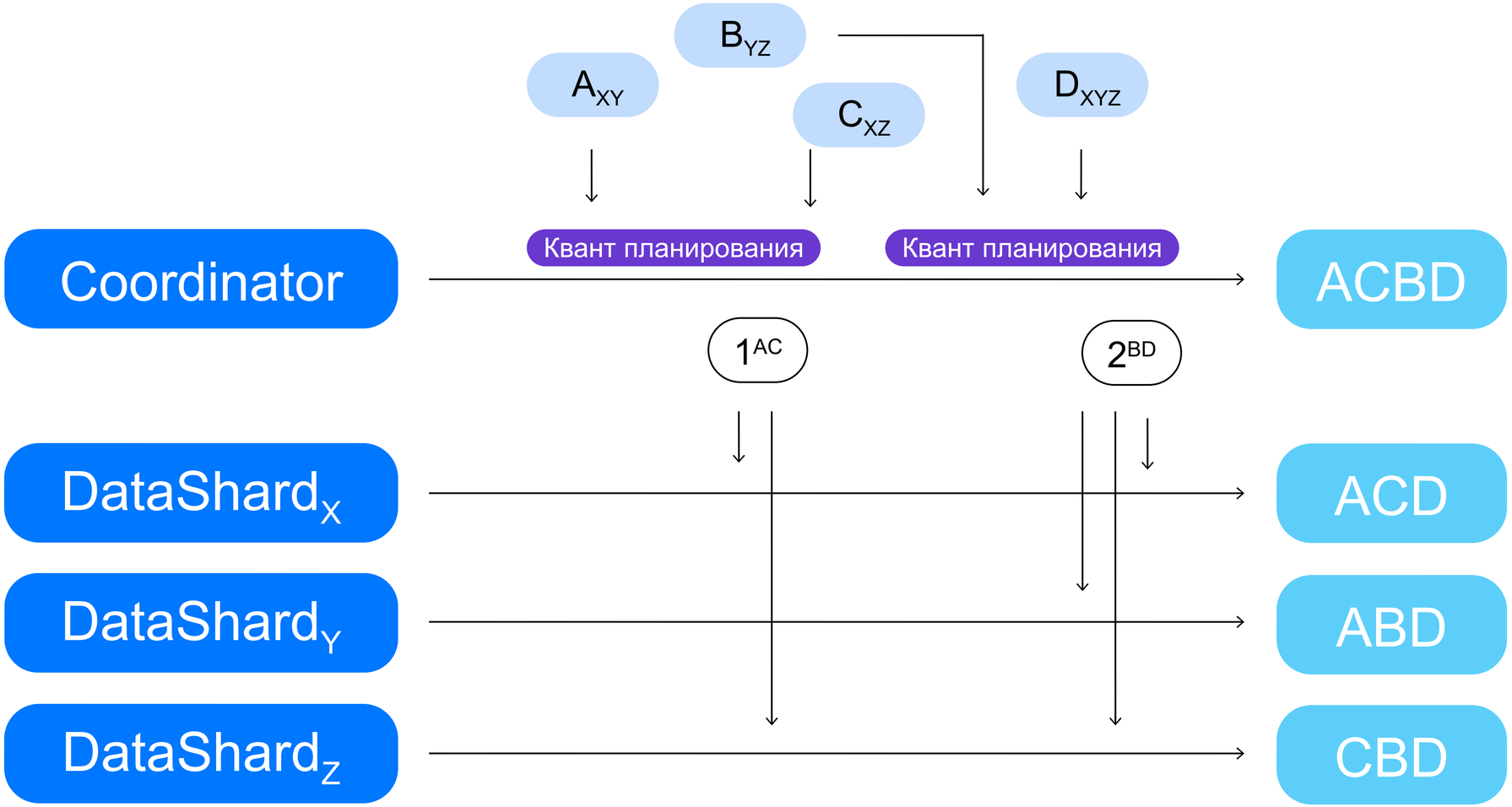


A, B, C, D —транзакции в системе
X, Y, Z — даташарды таблиц
1, 2, 3, 4 — порядок выполнения транзакций

03

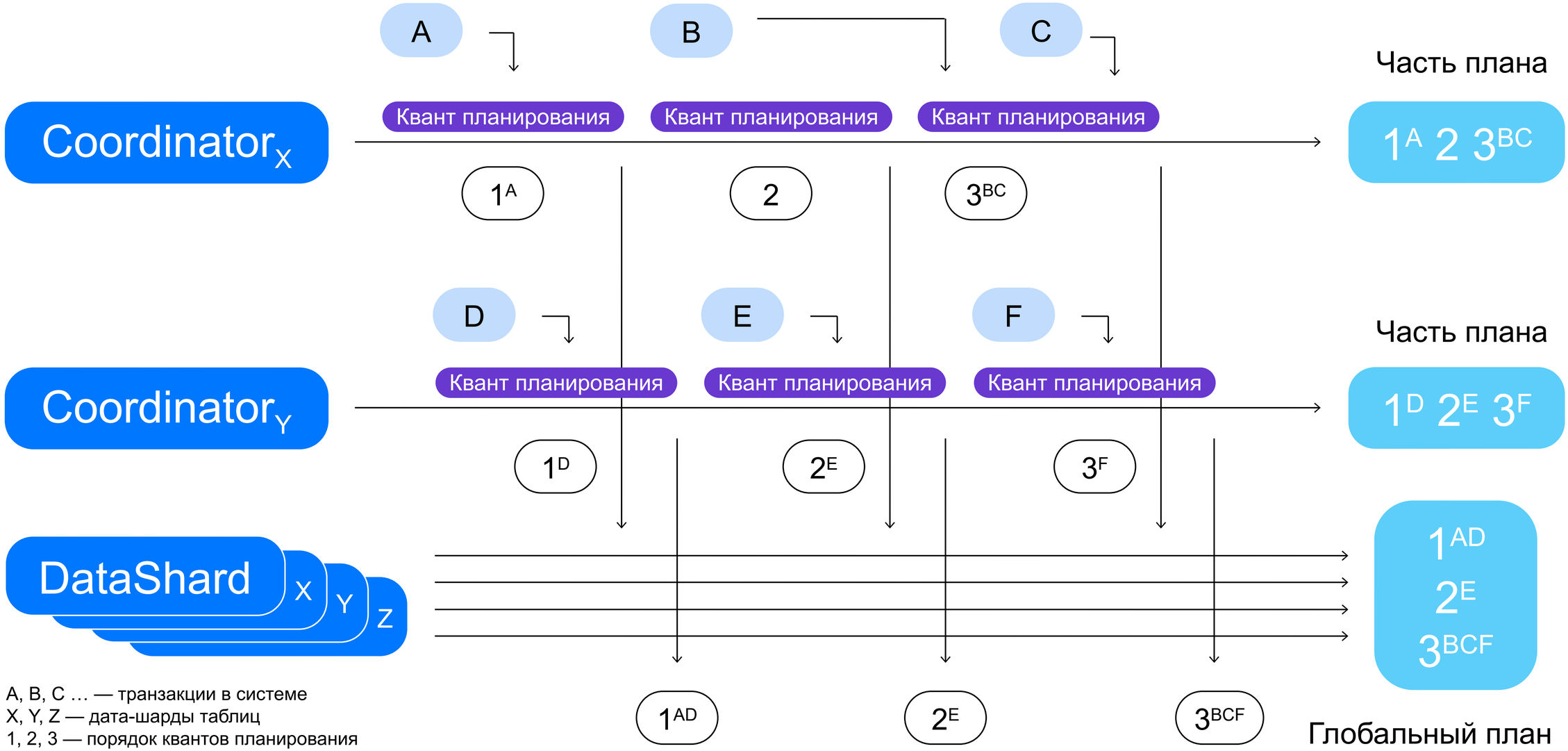
Как масштабировать наше решение

Кванты планирования



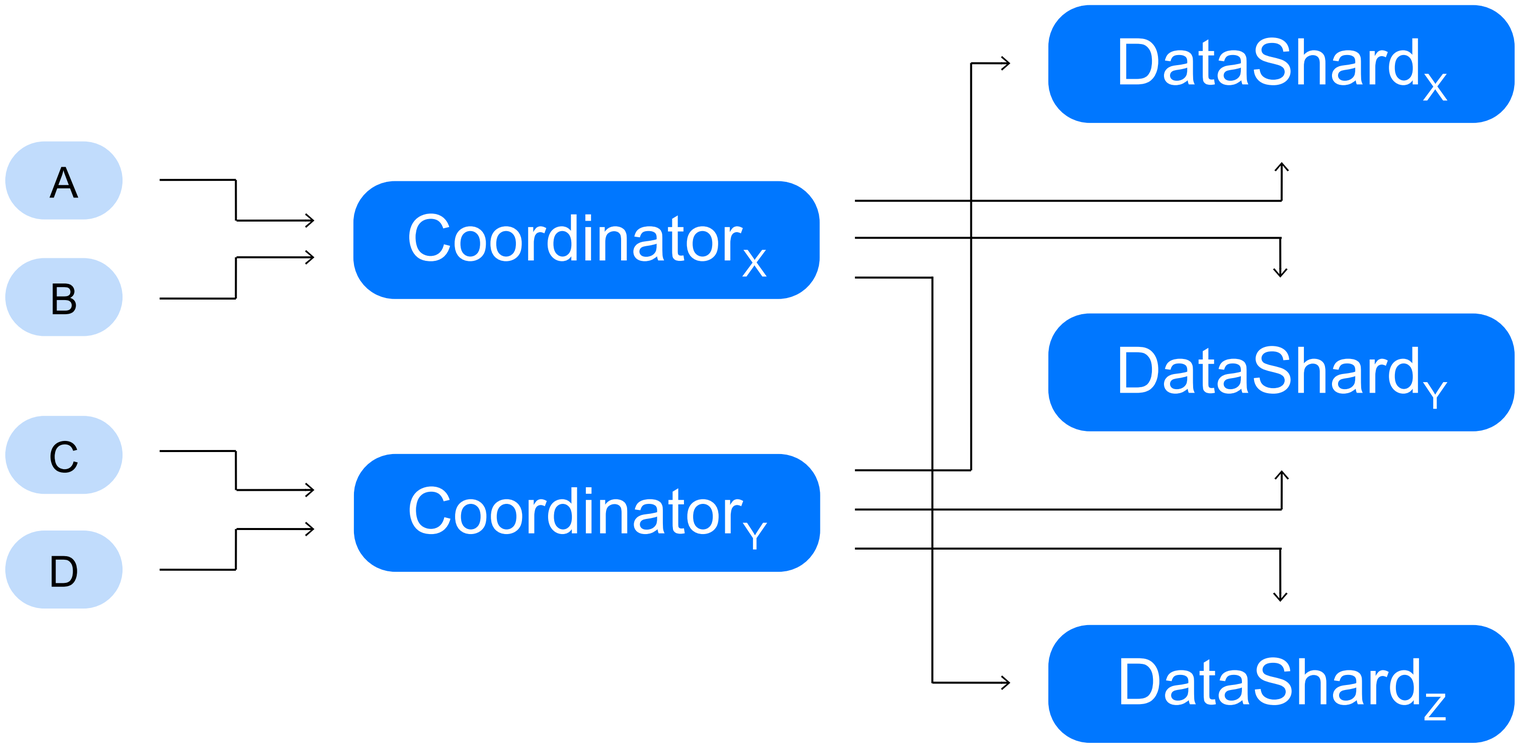
A, B, C, D — транзакции в системе
X, Y, Z — дата-шарды таблиц
1, 2, 3, 4 — порядок выполнения транзакций

Шардированный координатор

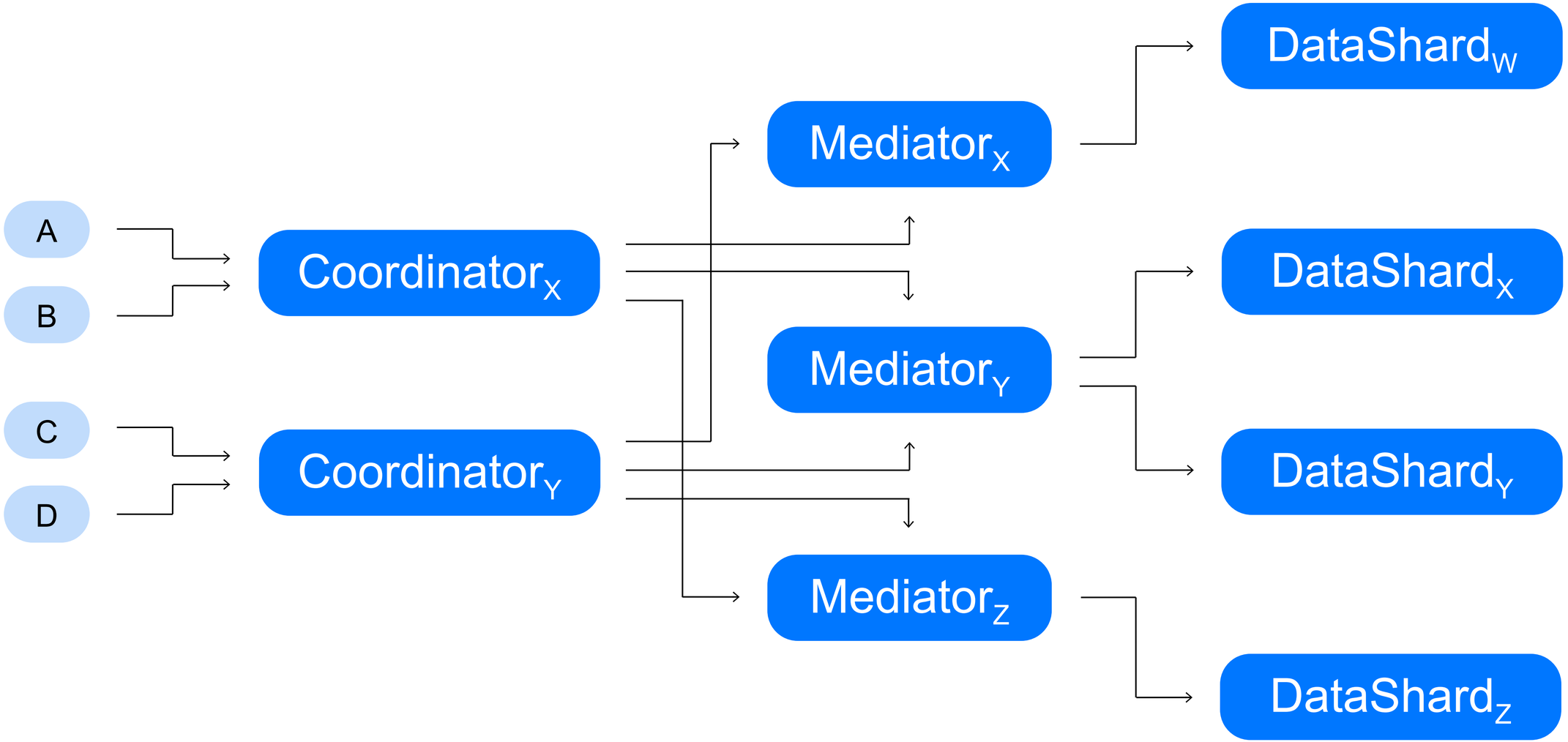


A, B, C … — транзакции в системе
X, Y, Z — дата-шарды таблиц
1, 2, 3 — порядок квантов планирования

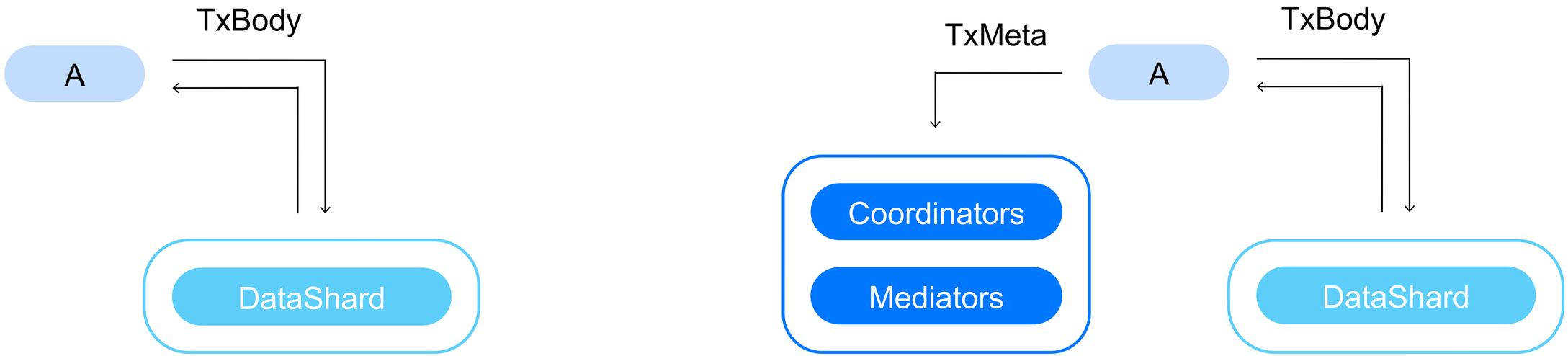
Количество связей Coordinator–DataShard



Медиатор


A
B
C
D
Coordinator$_X$
Coordinator$_Y$
Mediator$_X$
Mediator$_Y$
Mediator$_Z$
DataShard$_W$
DataShard$_X$
DataShard$_Y$
DataShard$_Z$

Разделить data и meta-информацию



04

Каков overhead запросов у нашего решения

Overhead

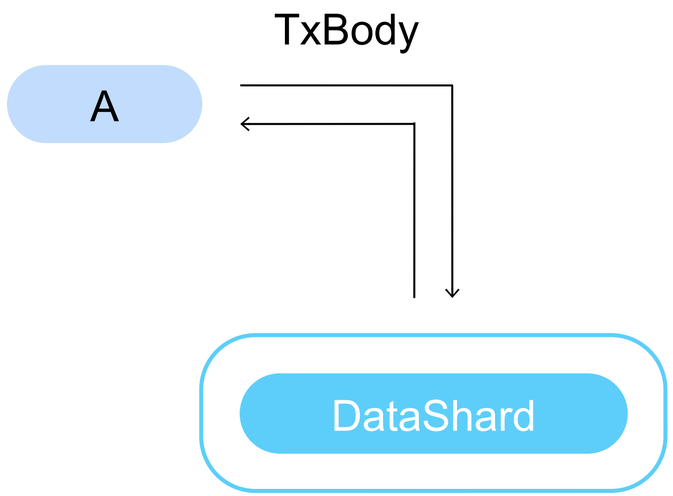
Игнорируем все, что связанно с самим запросом, считаем только overhead

- Время выполнения самого запроса не учитываем
- Все метаданные транзакций в памяти, их чтение бесплатно
- Одновременно выполняющиеся запросы считаем как один

Классы запросов

1. Read only, one shard — RO Immediate
2. Write only, one shard — WO Immediate
3. Read only / write only, multi shard — RO/WO
4. Read write, multi shard — RW

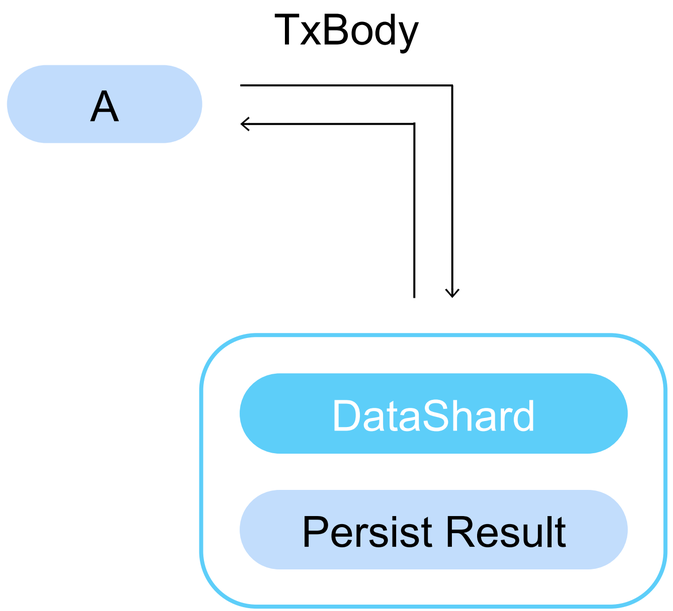
RO immediate



Итого

› 1 RTT

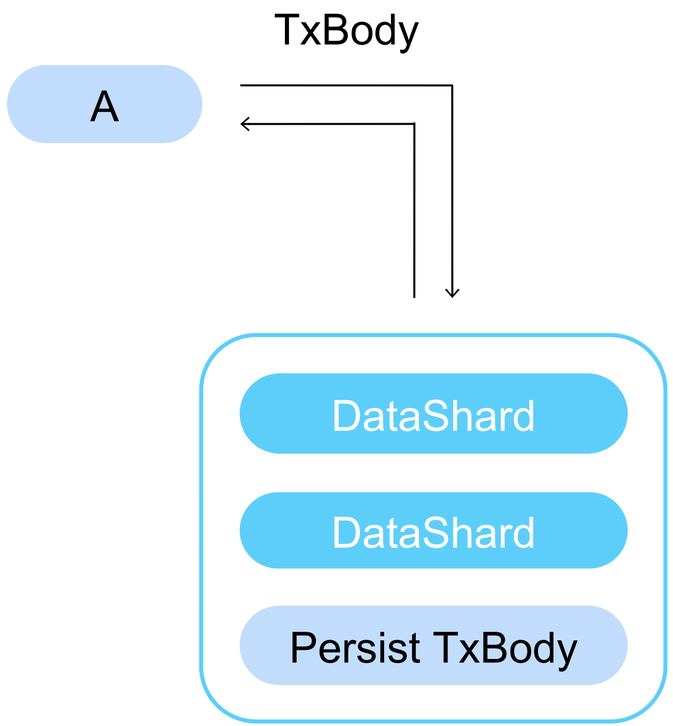
WO immediate



Итого

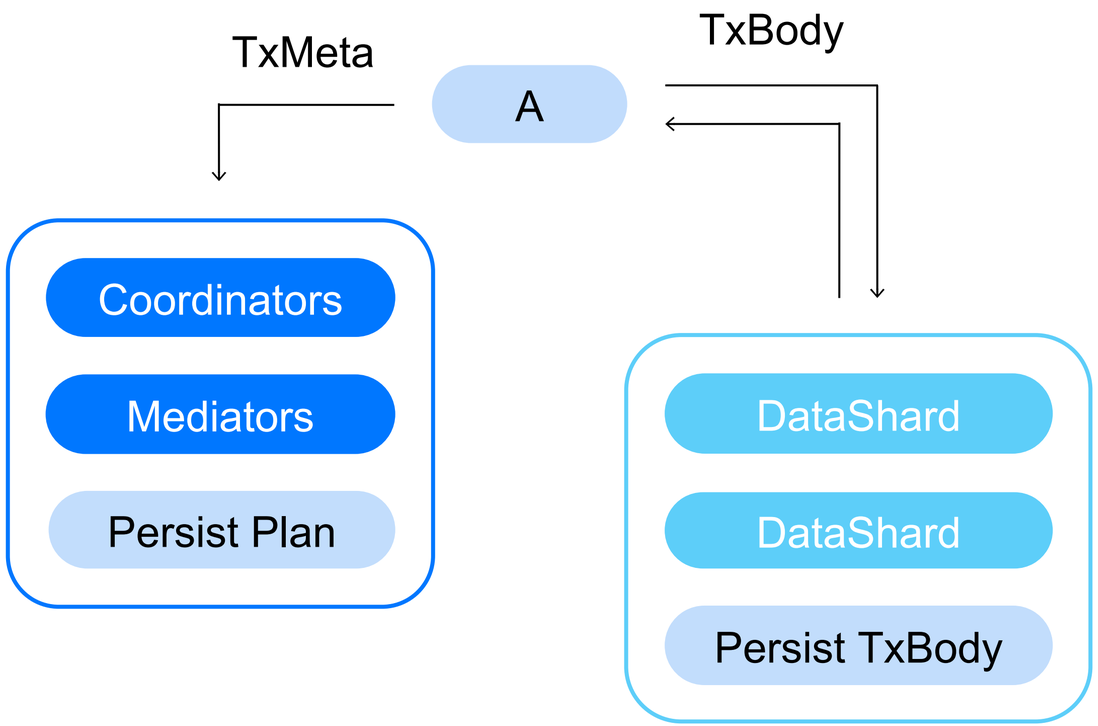
- 1 RTT
- 1 write

RO / WO



- 1 RTT
- 1 write

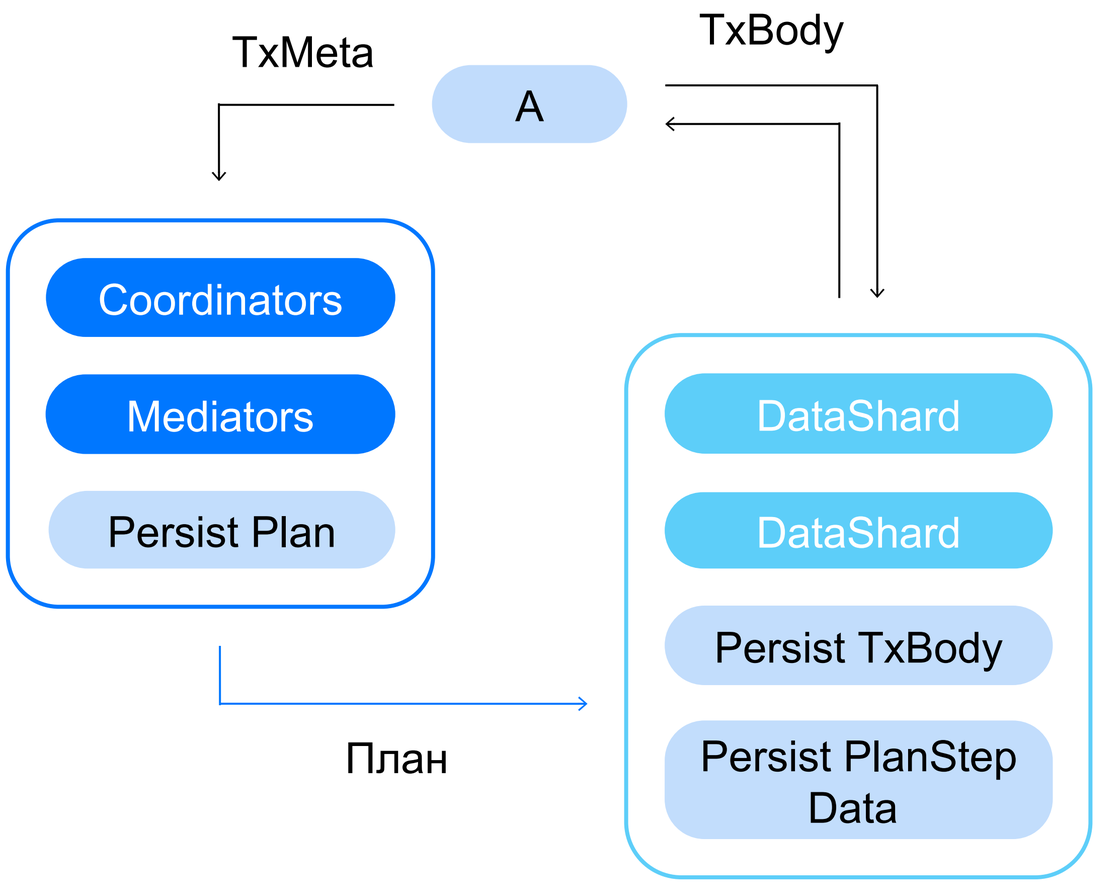
RO / WO



- 1 RTT
- 1 write

- 0.5 RTT
- 1 plan batch time
- 1 write
- 0.5 RTT

RO / WO



- 1 RTT
- 1 write

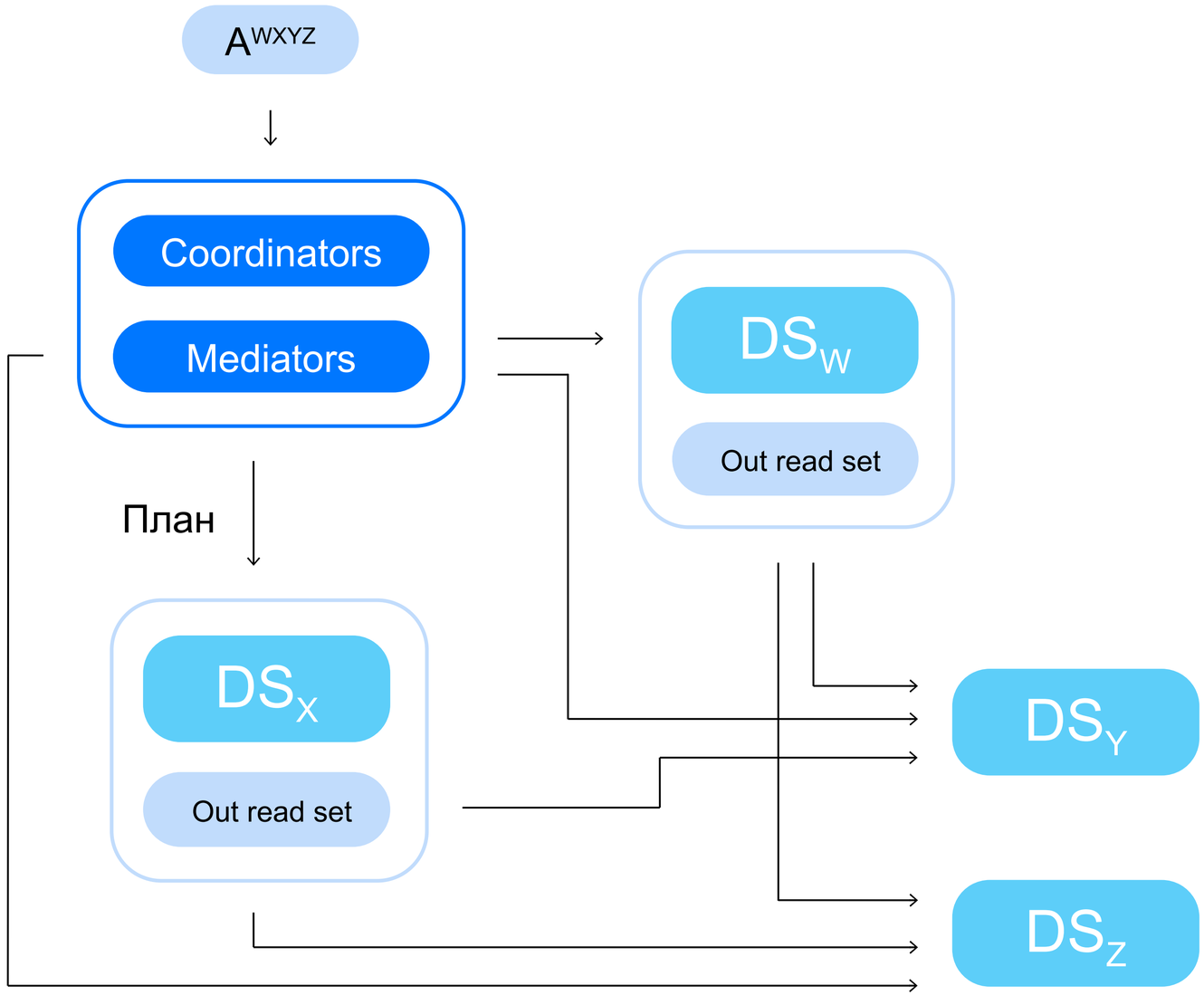
- 0.5 RTT
- 1 plan batch time
- 1 write
- 0.5 RTT

- 0.5 RTT
- 1 write
- 0.5 RTT

Итого

- 6 RTT + 1 plan batch time + 3 diskIO

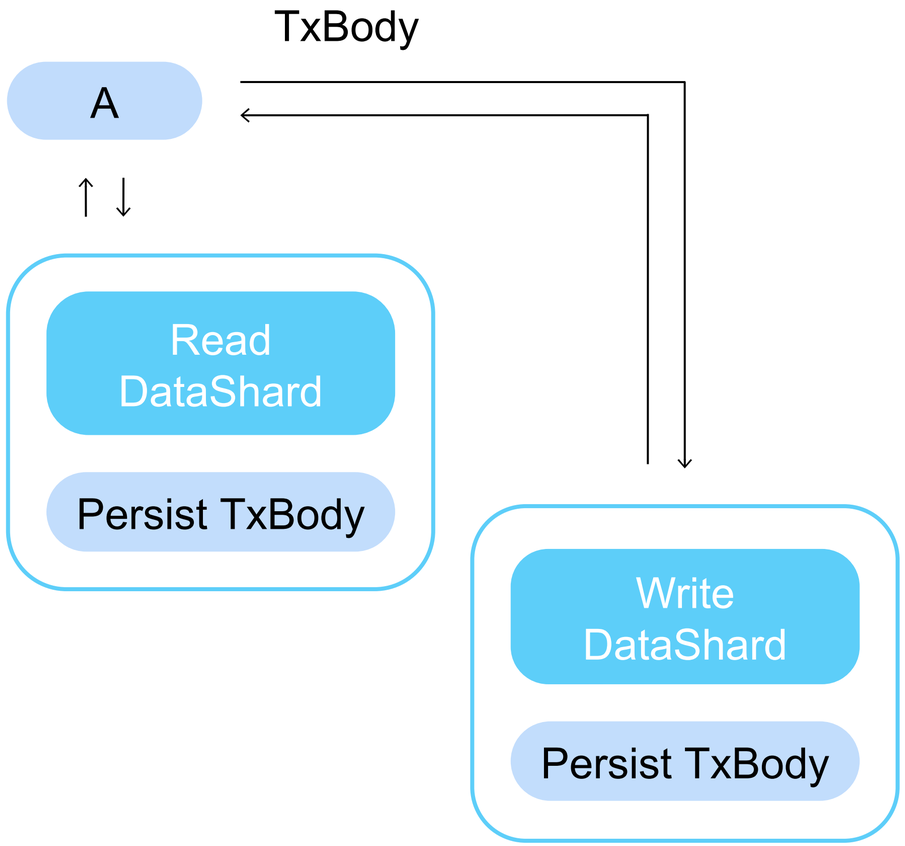
RW



Область применения

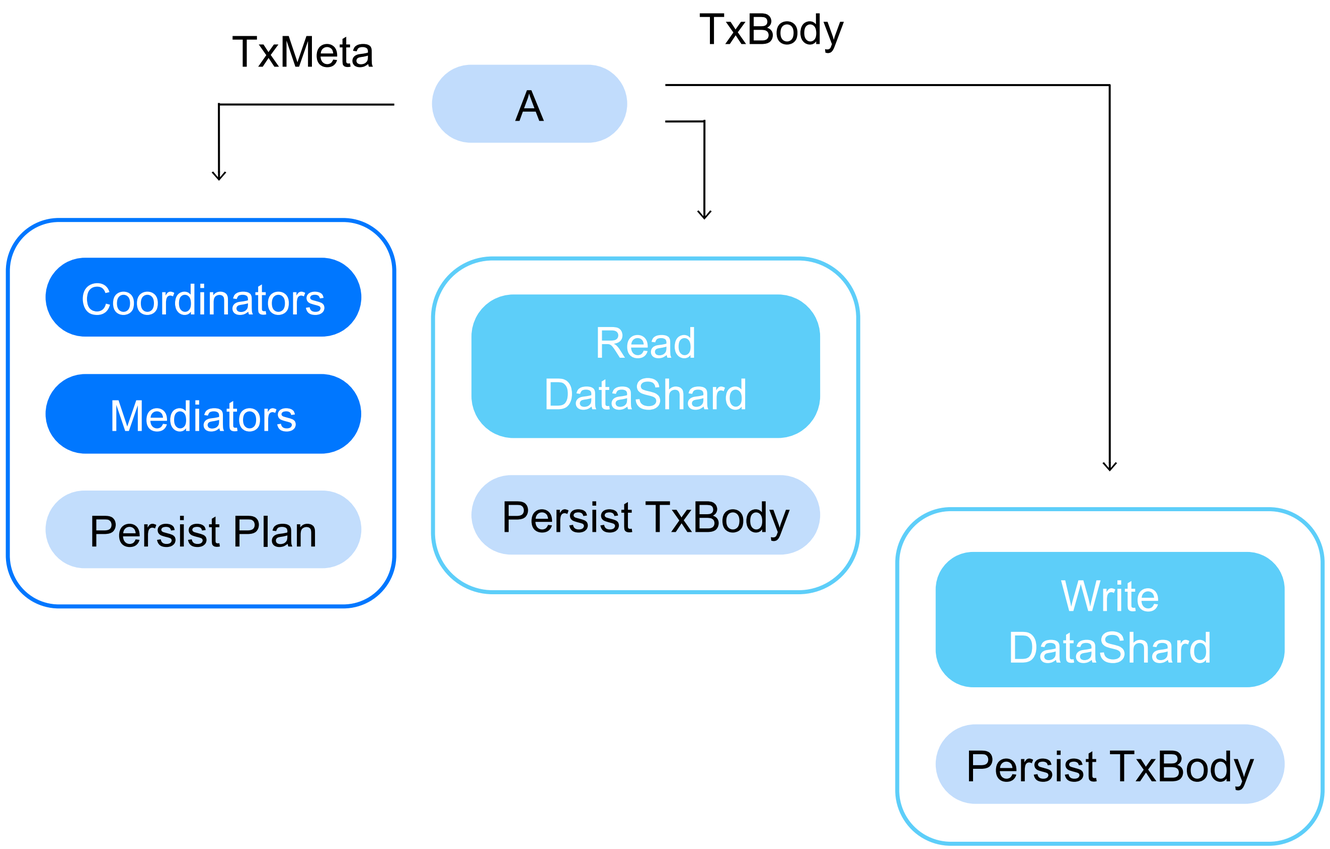
- Высокая конкуренция на данных

Транзакция RW



- 1 RTT
- 1 write

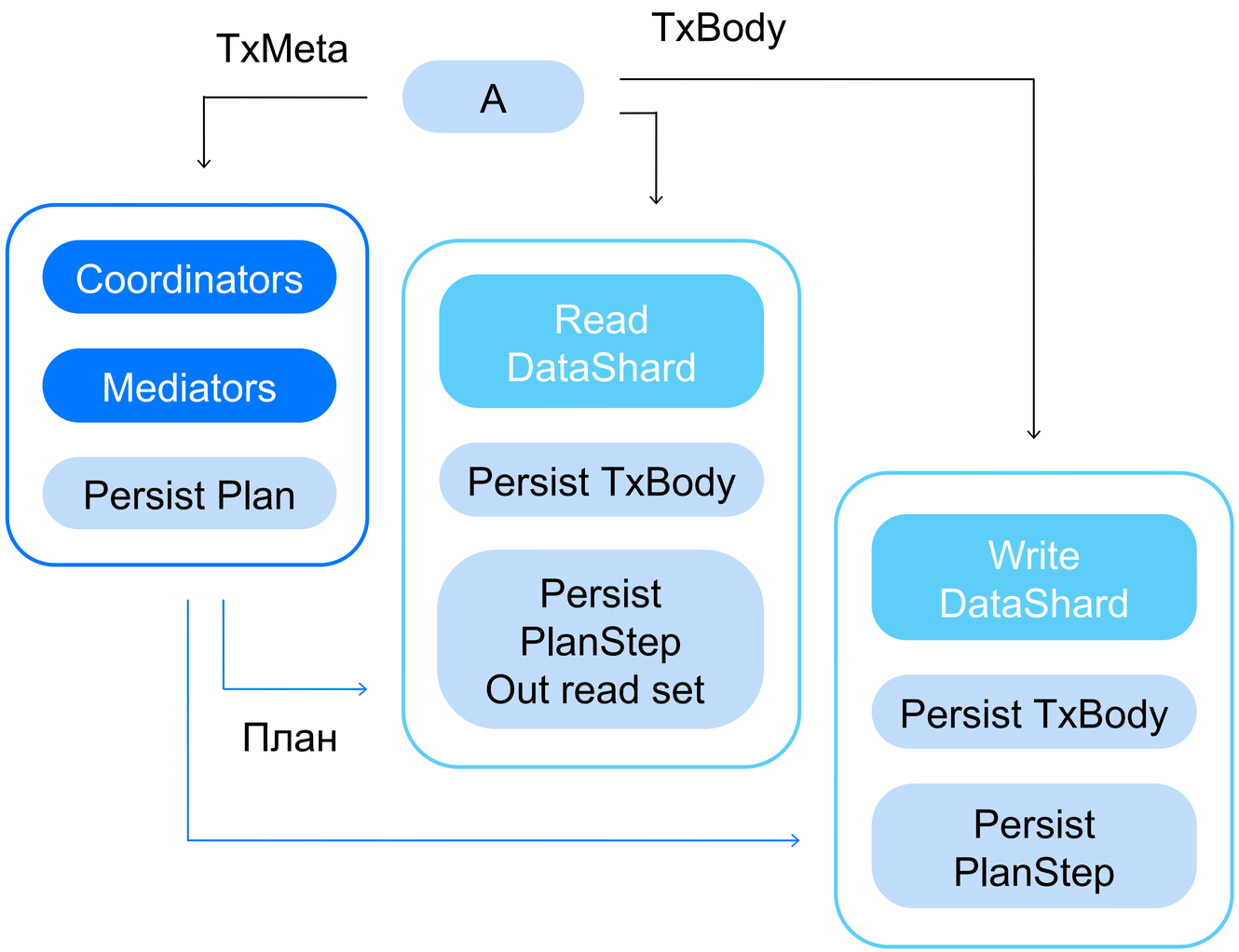
RW



- 1 RTT
- 1 write

- 0.5 RTT
- 1 plan batch time
- 1 write
- 0.5 RTT

RW

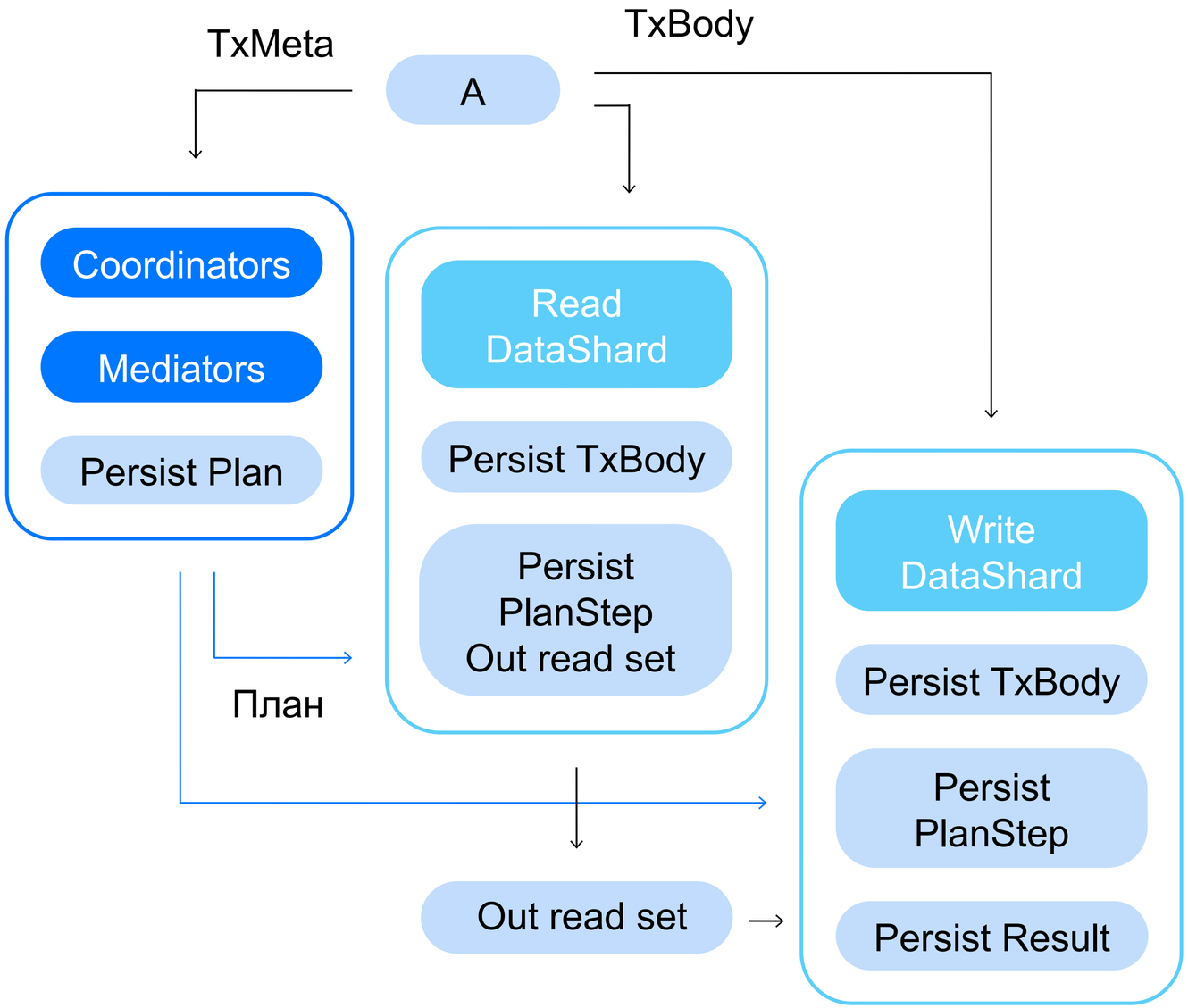


- 1 RTT
- 1 write

- 0.5 RTT
- 1 plan batch time
- 1 write
- 0.5 RTT

- 0.5 RTT
- 1 write

RW



- 1 RTT
- 1 write

- 0.5 RTT
- 1 plan batch time
- 1 write
- 0.5 RTT

- 0.5 RTT
- 1 write

- 0.5 RTT
- 1 write
- 0.5 RTT

Итого

- 7.5 RTT + 1 plan batch time + 4 diskIO

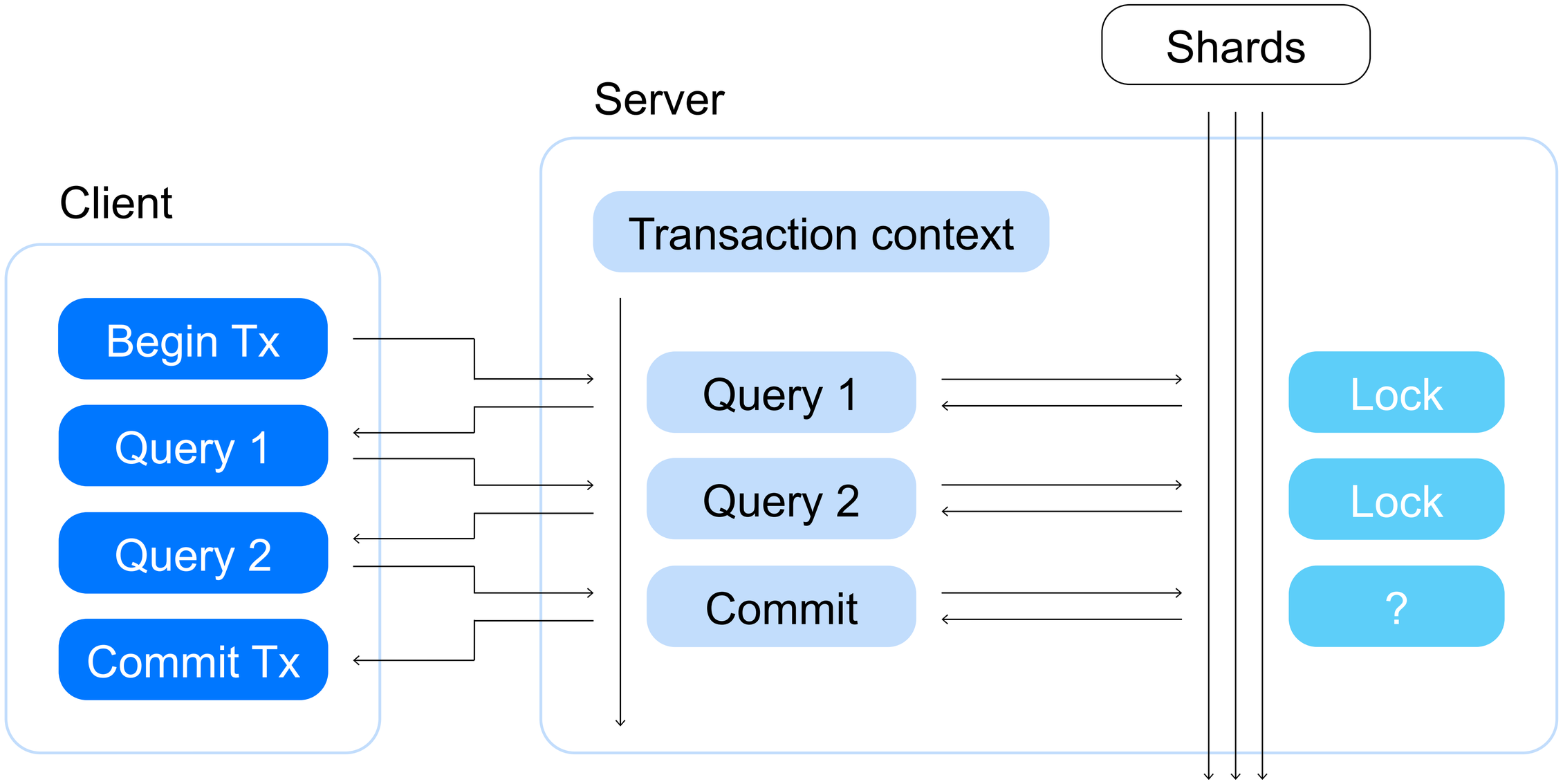
Overview

1. Read only, one shard — RO Immediate
   — **1 RTT**
2. Write only, one shard — WO Immediate
   — **2 RTT + 1 diskIO**
3. Read only / write only, multi shard — RO/WO
   — **6 RTT + 1 plan batch time + 3 diskIO**
4. Read write, multi shard — RW
   — **7.5 RTT + 1 plan batch time + 4 diskIO**

05

YQL-транзакции

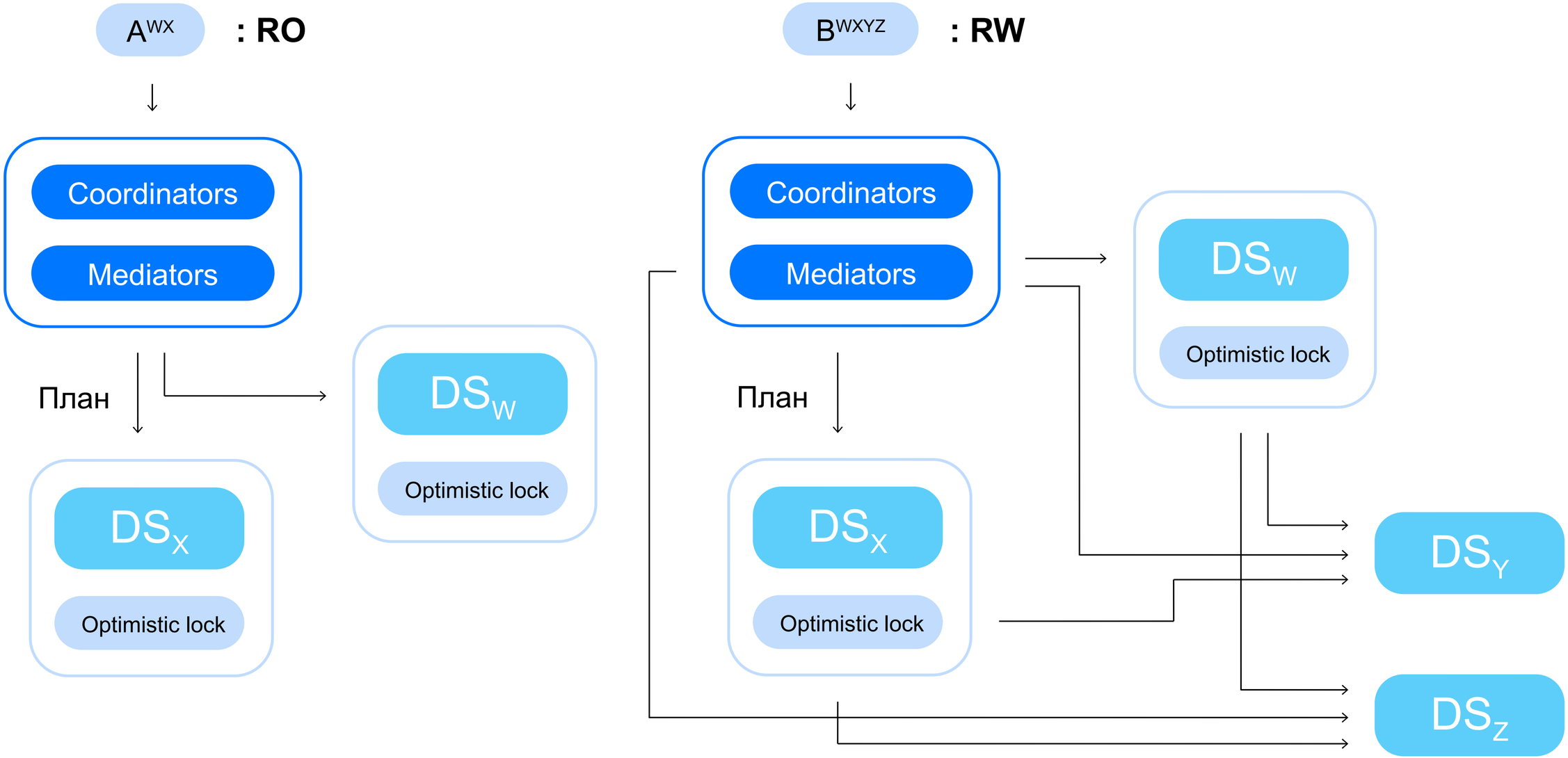
Общий вид


Shards
Server
Client
Transaction context
Begin Tx
Query 1
Query 2
Commit Tx
Query 1
Query 2
Commit
Lock
Lock
?

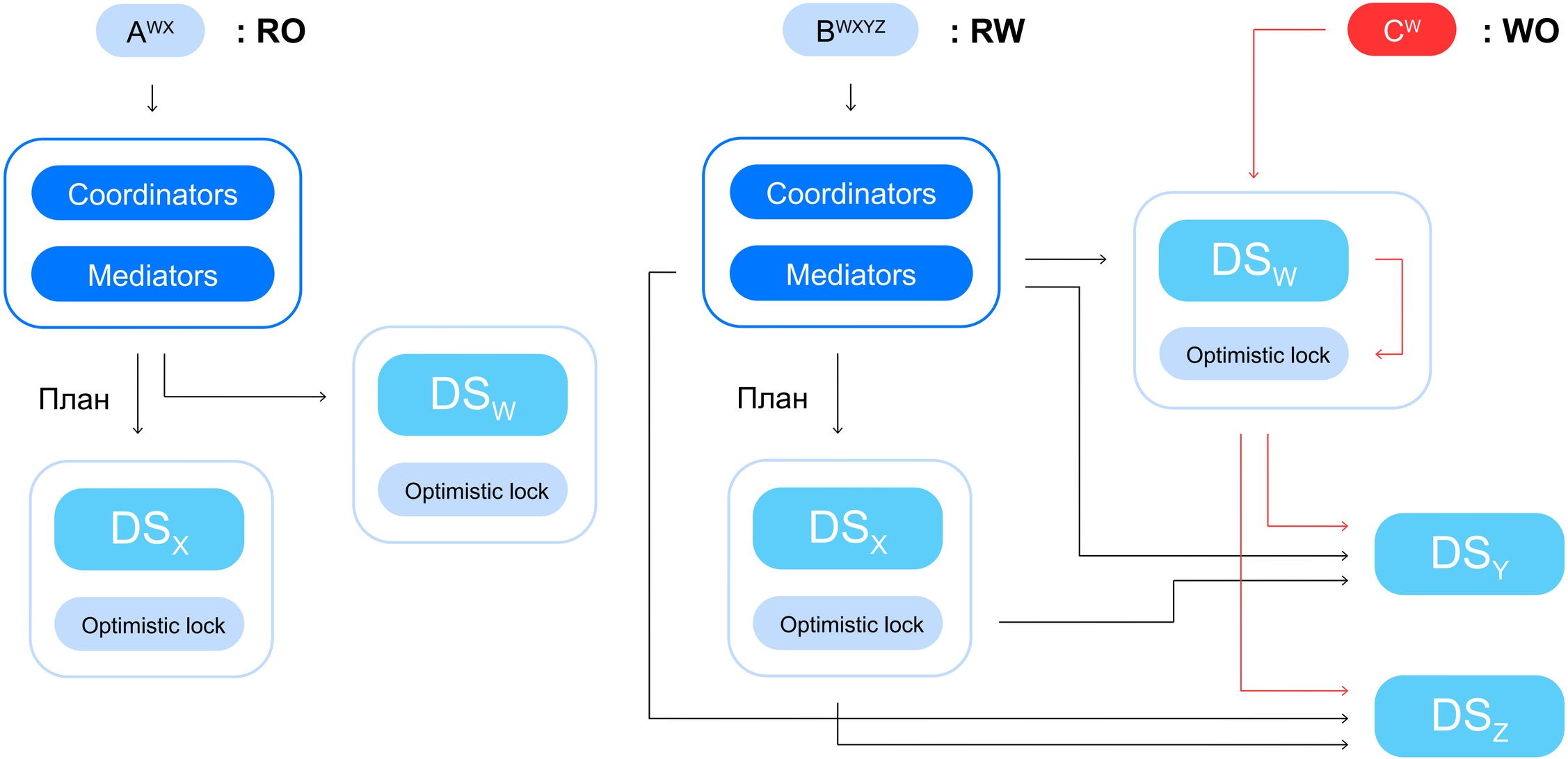
YQL-транзакции

- Несколько отдельных атомарных запросов в рамках одной YQL-транзакции
- Набор read / write set’ов ключей каждого запроса определен
- Каждый запрос чтения захватывает оптимистичную блокировку
- Изменения применяются в последнем запросе на коммите транзакции
- Взятые блокировки проверяются в последнем запросе на коммите транзакции

YQL-транзакции



YQL-транзакции



05

Доступные уровни изоляции для транзакций

Уровень изоляции

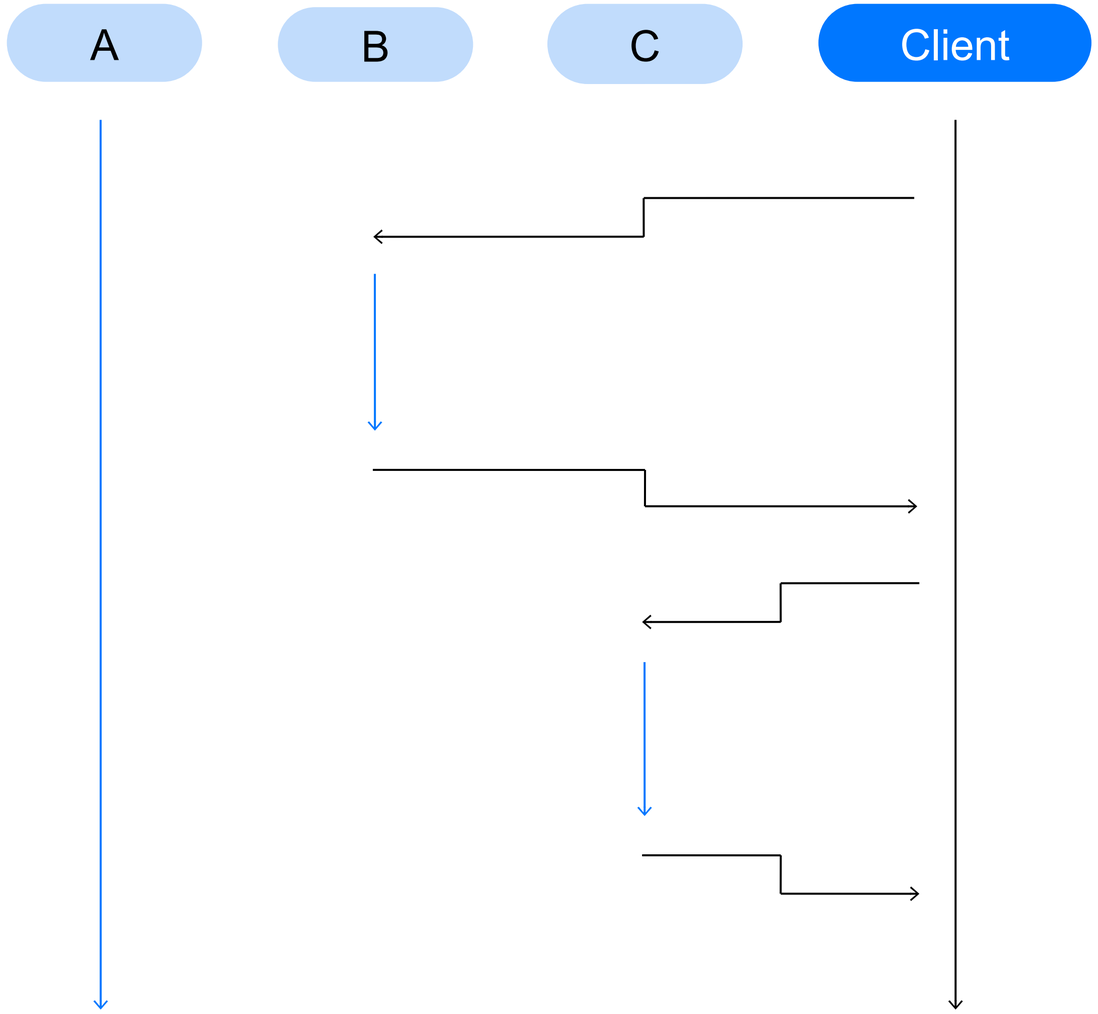
Serializable — default isolation level

- Координируемые и immediate-транзакции

Strict serializable — maximum isolation level

- Все транзакции координируемы

Strict serializable

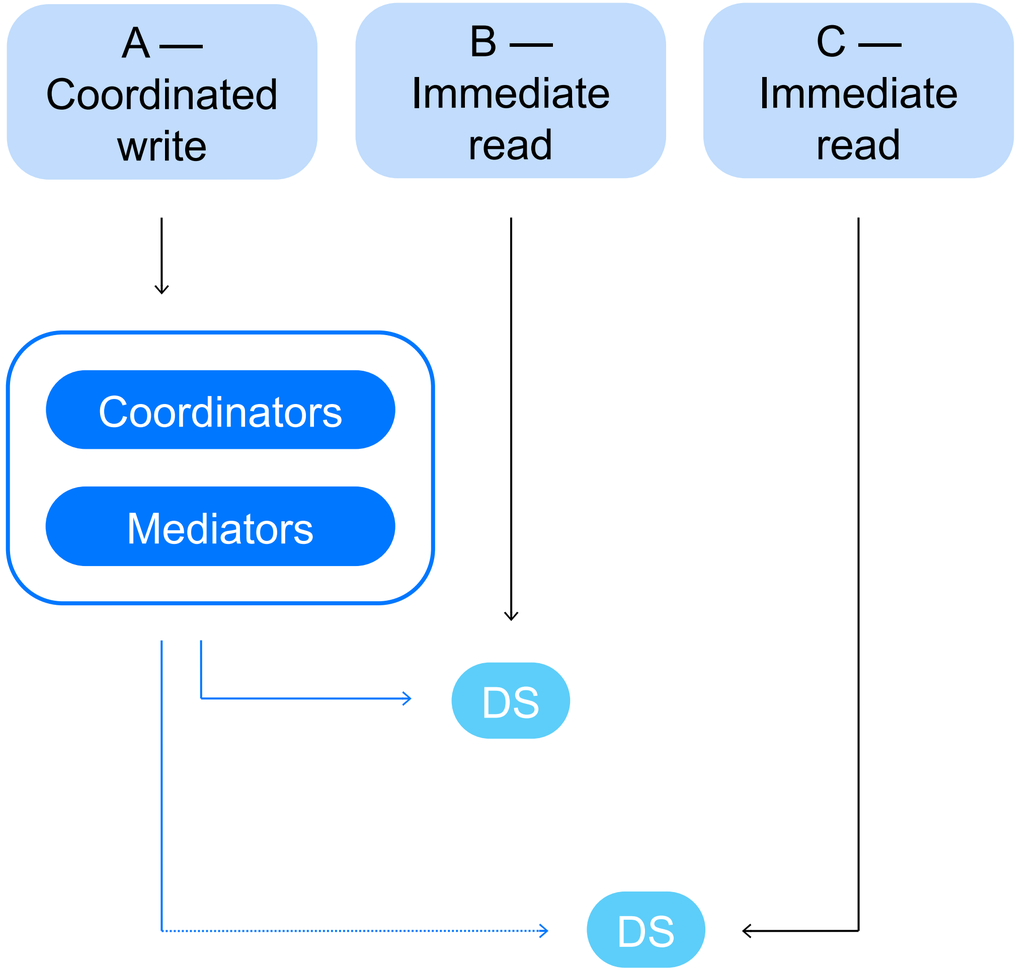


- Транзакция A в процессе выполнения
- Транзакции B и C выполняются последовательно

Strict serializable

- Порядок между B и C определен однозначно
- Если B видит изменения A, то C также должно их видеть

Serializable



Serializable

- Порядок между B и C существует, но не определен заранее
- Возможно, что B видит изменения A, в то время как C не видит изменений A
- Возможный порядок выполнения: C A B

Yandex Database (YDB)

- Надёжное хранение данных с избыточностью и автоматической репликацией
- Отказоустойчивость, автоматическое восстановление от сбоев
- Распределённые ACID-транзакции с serializable-уровнем изоляции транзакций
- Высокая пропускная способность при малом времени отклика
- Горизонтальная масштабируемость до тысяч нод

Яндекс Облако

Спасибо!

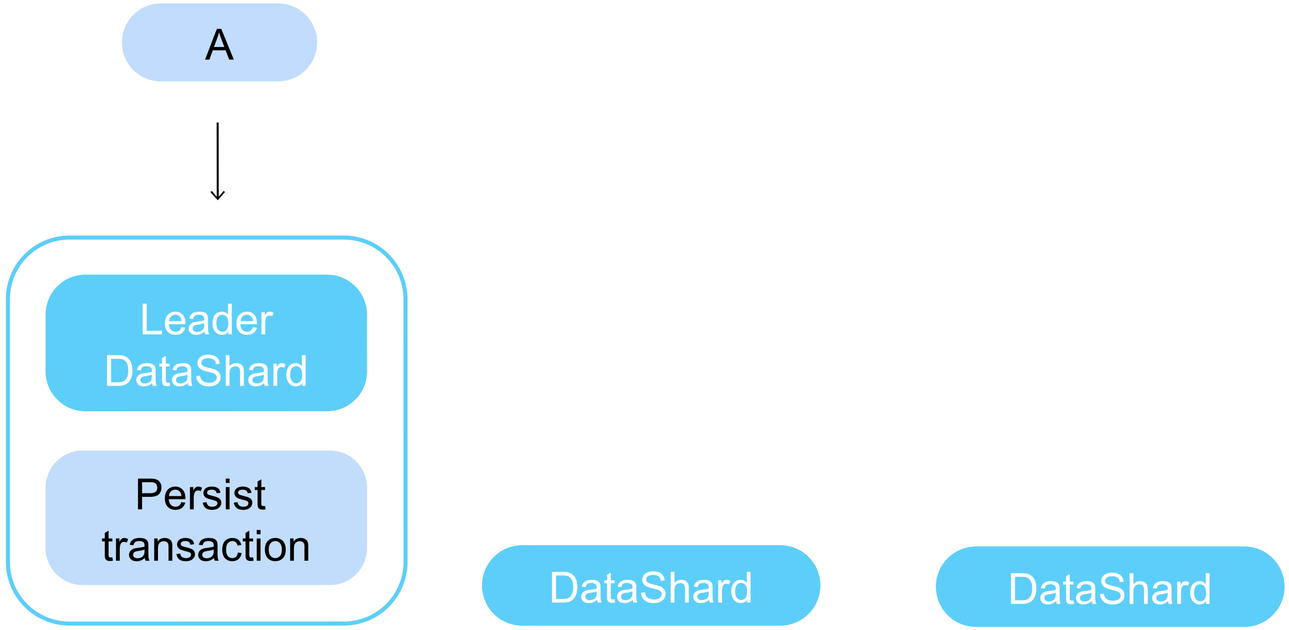
Семён Чечеринда

Ведущий разработчик

svc@yandex-team.ru

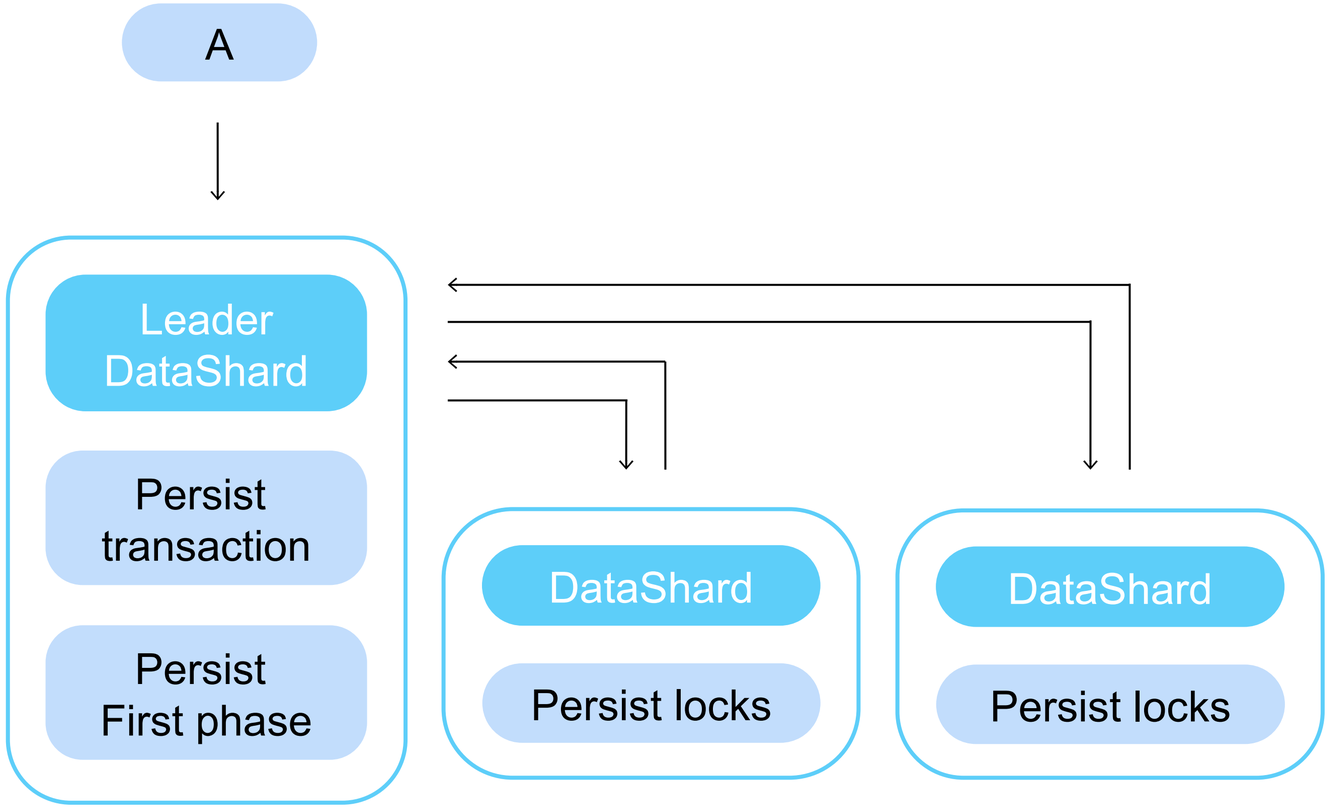
cloud.yandex.ru/services/ydb

What if two-phase commit



- 0.5 RTT
- 1 write

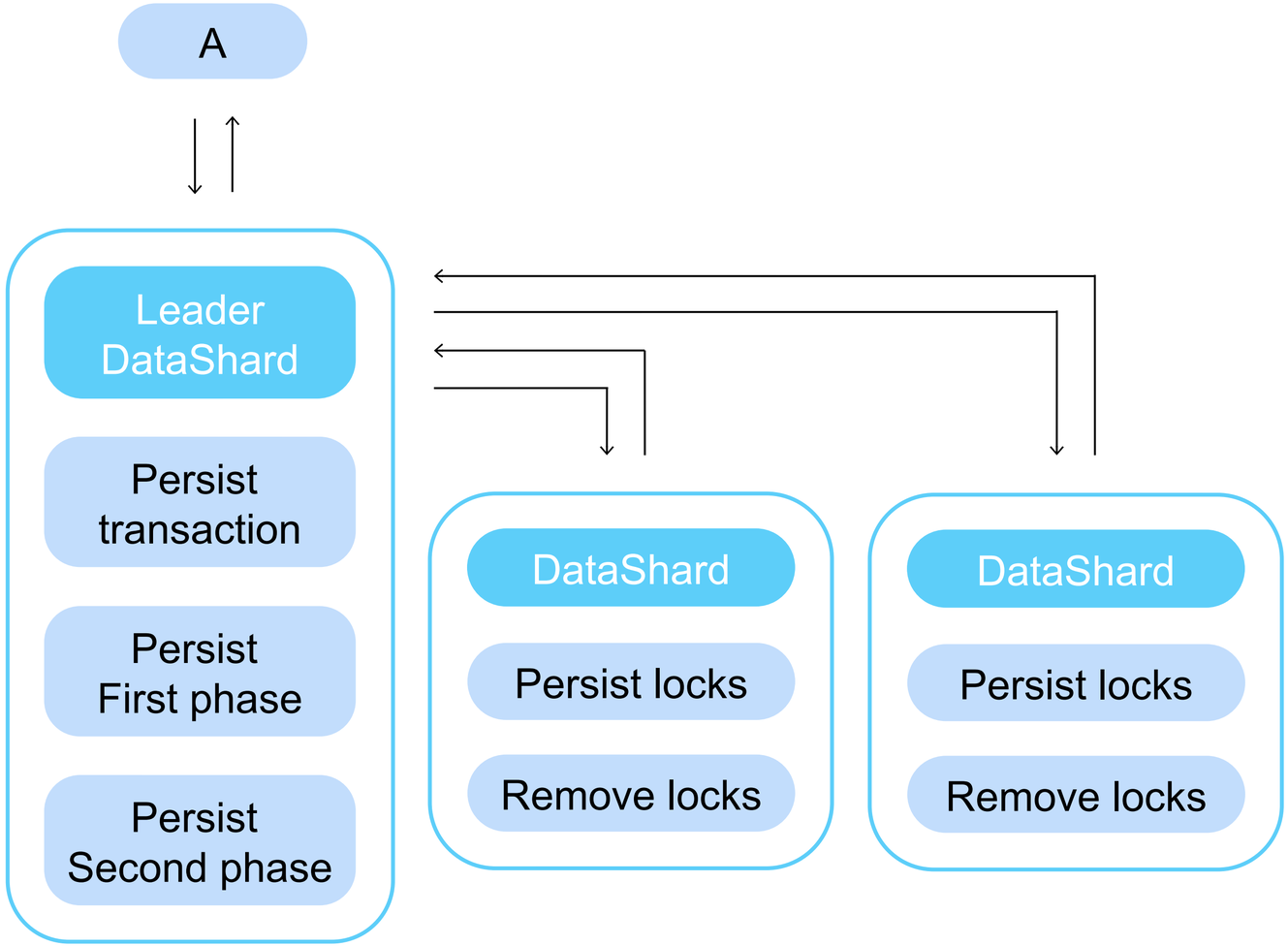
What if two-phase commit



- 0.5 RTT
- 1 write

- 1 RTT
- 1 write
- 1 write

What if two-phase commit



- 0.5 RTT
- 1 write

- 1 RTT
- 1 write
- 1 write

- 1 RTT
- 1 write
- 0.5 RTT

Итого

- 7 RTT + 4 diskIO