
Яндекс

Яндекс Облако

Yandex Database:
обзор сервиса и решаемые задачи

Сергей Пучин

Содержание

Решаемые задачи

- Online-нагрузка (OLTP)
  - Чтение / запись
  - Большой TPS, низкие задержки

- Транзакционное выполнение
- Диалект SQL для написания запросов

SQL

- Отказоустойчивость
- Высокая доступность
- Масштабируемость по нагрузке

NoSQL

Yandex Database

> Yandex Database — это геораспределённая
> база данных, предоставляющая

- Надежное хранение данных с автоматической репликацией
- Механизм распределенных ACID-транзакций
  со строгой консистентностью
- Высокую пропускную способность при малом времени
  отклика
- Автоматическое восстановление после сбоев
- Горизонтальную масштабируемость до тысяч нод
- Декларативный язык запросов YQL

Пример использования: Турбо-страницы поиска Яндекс

- Хранение метаданных для картинок в документах
  - Получение метаданных для документа
  - Обновление устаревших метаданных
  - Добавление метаданных для новых документов
- Read-Write транзакции
- Терабайты данных
- 50K+ TPS
- Задержки не более 100 ms для 99% запросов

Пример использования: Яндекс.Коллекции

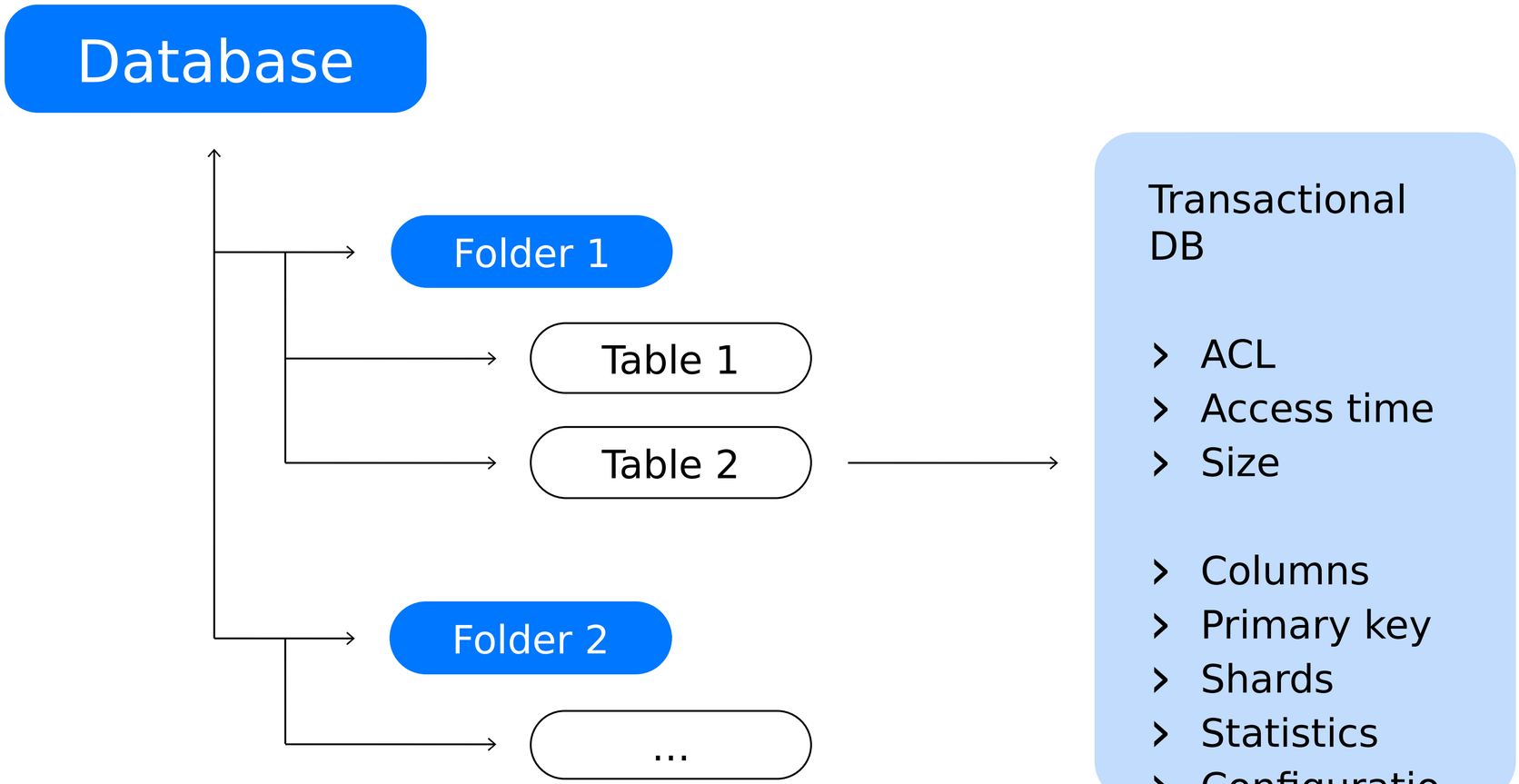
- Хранение истории о лентах рекомендаций для пользователей
  - Получение последних записей из истории пользователя
  - Добавление новых рекомендаций в историю
- Необходимость переживать потерю одного из дата-центров
- Read-Write транзакции для получения и обновления истории
- Терабайты данных и 1K+ TPS
- Задержки не более 50 ms для 99% запросов

Пример использования: Яндекс.Облако

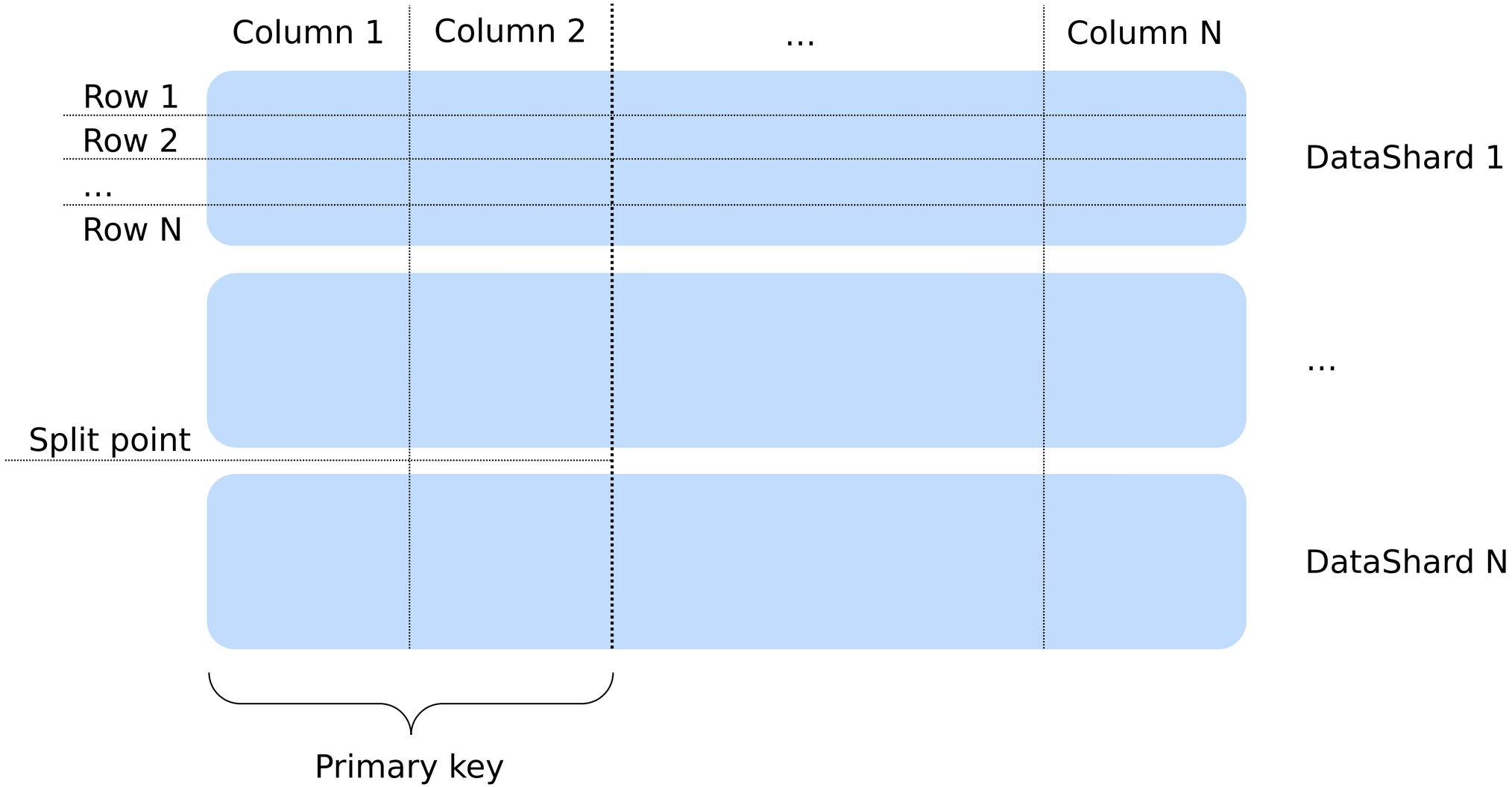
Основное хранилище метаданных для Яндекс Облака

- Системные сервисы (IaaS / PaaS)
- Пользовательские сервисы для управления данными
  - Yandex Object Store
  - Yandex Message Queue

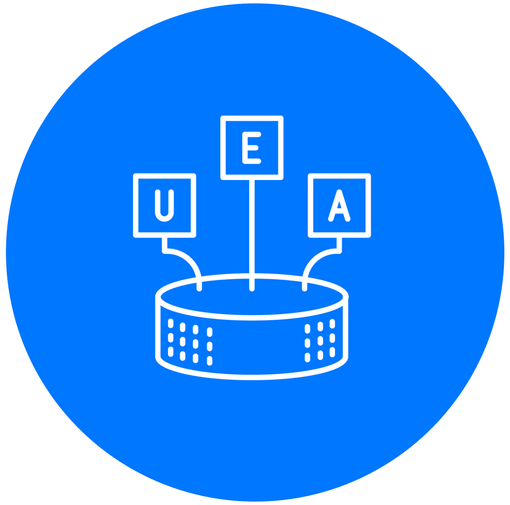
Схема базы данных



Таблицы


Column 1
Column 2
...
Column N
Row 1
Row 2
...
Row N
DataShard 1
...
Split point
DataShard N
Primary key

Способы шардирования



- Uniform Partitioning
  - Разбиение на заданное количество шардов по равномерно распределённому числовому ключу
  - Пример: использование хеша от ключа для шардирования
- Explicit Partitioning
  - Явное указание границ шардов, в том числе для составных ключей
- Auto Partitioning
  - Автоматический split / merge по размеру данных

Распределённые запросы

Yandex Query Language (YQL)

- Диалект SQL
- Строгая типизация
- Нативная поддержка составных типов
- Именованные подзапросы
- Явная параметризация

- Специализированные DML-конструкции:
  - UPSERT / REPLACE
  - UPDATE ON (Update by list of key/values)
  - DELETE ON (Delete by list of keys)

Yandex Query Language (YQL)

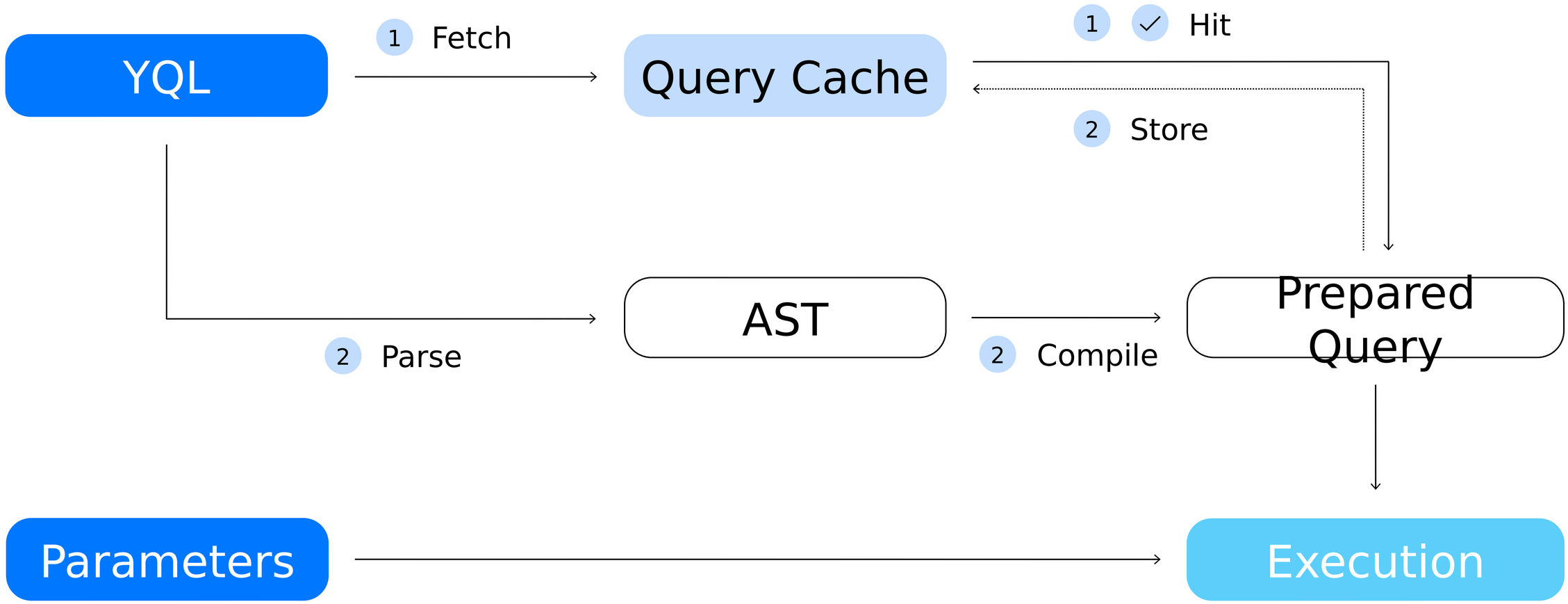
```
DECLARE $name AS String;

DECLARE $groupIds AS List<Struct<GroupId: Uint64>>;


$userId = (

    SELECT Id FROM `Users` VIEW ByName WHERE Name = $name

);


UPSERT INTO `Groups`

SELECT GroupId, $userId AS UserId FROM AS_TABLE($groupIds);
```

Подготовленные запросы

Компиляция запроса — дорогая операция

- Параметризация запросов — ключ к эффективному выполнению
- Для параметров запроса явно задается их тип
- Подготовленный запрос может использоваться с произвольными значениями параметров
- Подготовка может быть выполнена явно (прогрев), либо при первом выполнении запроса
- Результат подготовки сохраняется в кэше запросов

Подготовленные запросы



Транзакции

Распределенные ACID-транзакции

- Уровень изоляции: Serializable
- Транзакции выполняются логически последовательно
- Все изменения надежно сохранены при успешном коммите

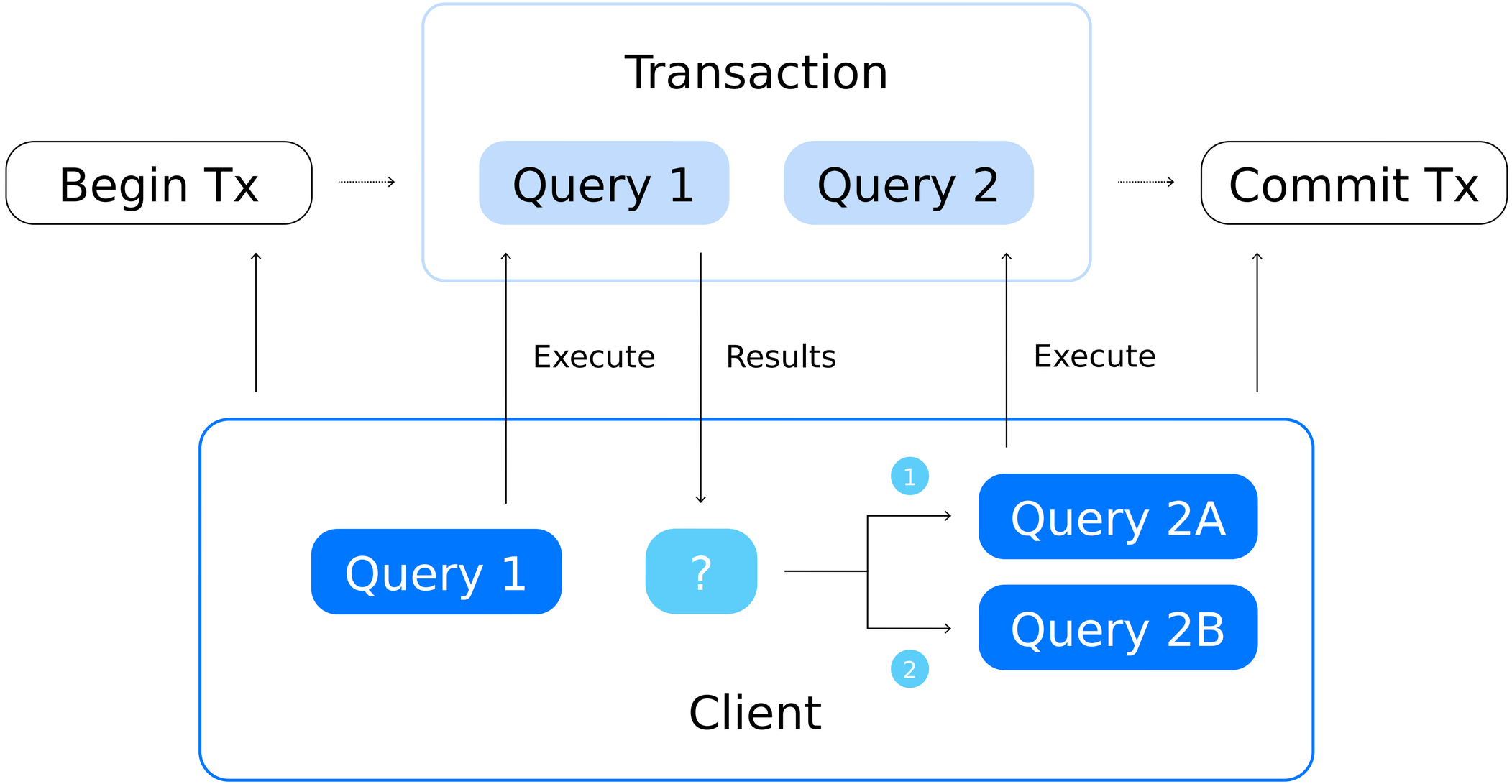
Отложенные изменения

- Изменения применяются на коммите транзакции
- Транзакции не видят собственных изменений

Открытые транзакции

- Транзакция может состоять из нескольких отдельных запросов
- Очередной запрос в транзакцию может зависеть от результата выполнения предыдущих и клиентской логики
- Ошибка выполнения запроса инвалидирует транзакцию
- Транзакция начинается вызовом/ флагом BEGIN
- Транзакция завершается вызовом / флагом COMMIT или вызовом ROLLBACK

Открытые транзакции



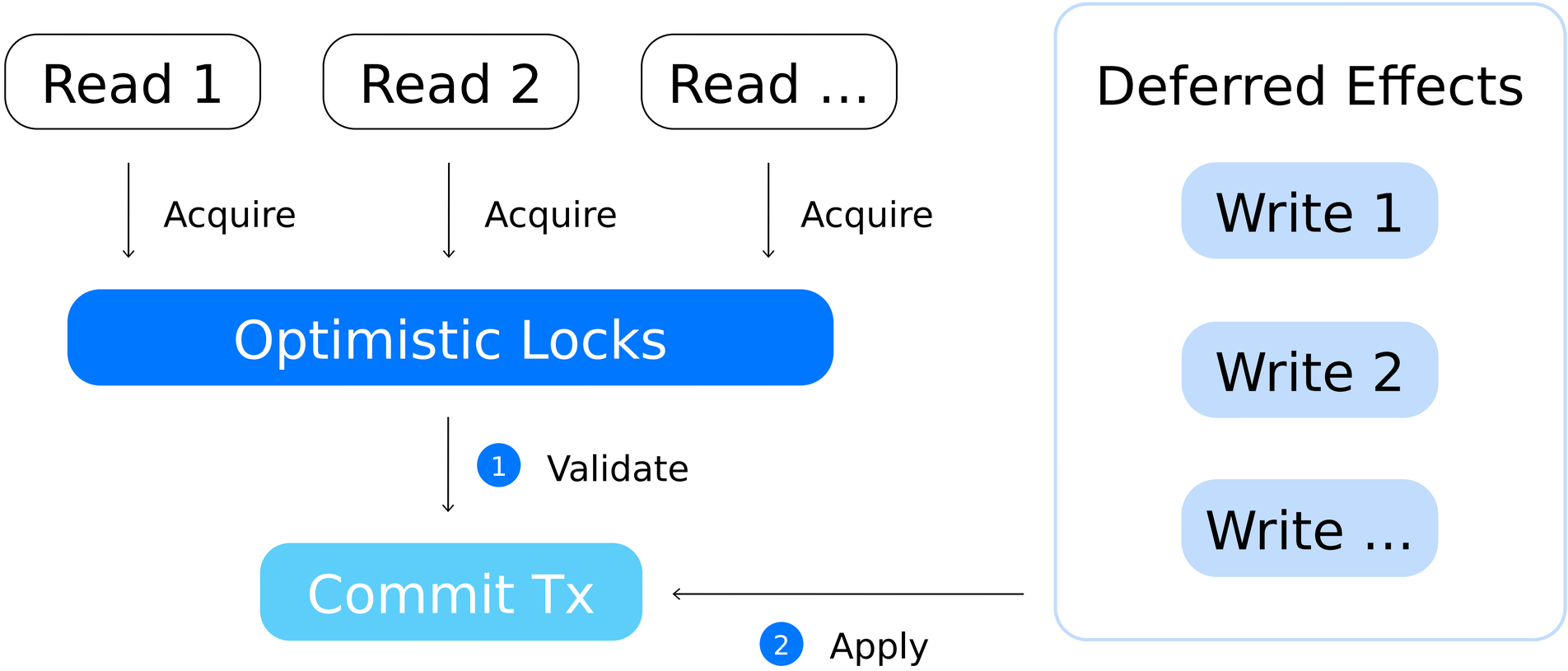
Оптимистичные блокировки

- Каждое из чтений в транзакции захватывает блокировку (lock)
- Блокировка берется на весь диапазон читаемого PK
- Блокировка не препятствует чтению и записи данных другими запросами
- При любом изменении по ключу из диапазона блокировки она инвалидируется
- Все блокировки сохраняются до момента коммита транзакции

Оптимистичные блокировки

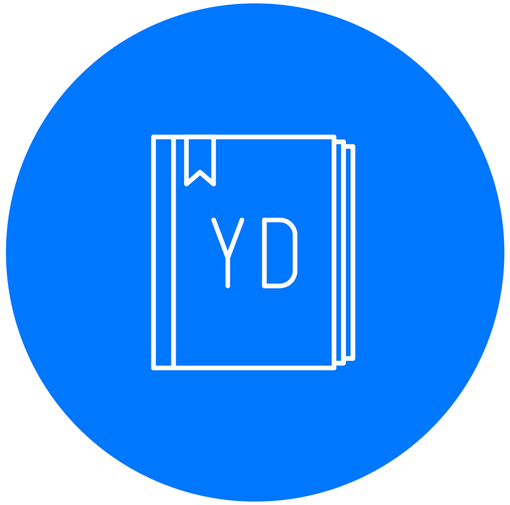
- При коммите транзакции:
  - Все блокировки транзакции валидируются
  - В случае удачной валидации применяются отложенные изменения транзакции
  - В случае неудачи транзакция откатывается
- В механизме оптимистичных блокировок выигрывают записи
- Гарантируется прогресс по крайней мере одной конкурентной транзакции

Оптимистичные блокировки



Клиентское взаимодействие

Клиентское взаимодействие



Yandex Database — сервис Яндекс.Облака

- Data plane
  - GRPC + Protobuf 3 API
- Клиентские библиотеки для популярных языков программирования
  - Python
  - Go
  - Java

Сервисы

- Discovery
  - Поиск хостов и клиентская балансировка
- Scheme
  - Общие операции над элементами схемы
- Table
  - Операции над таблицами
  - Выполнение транзакций
- Operation
  - Управление долгоживущими операциями

Сессии

> Сессия — логическая абстракция подключения к БД

- Сессия хранит текущее пользовательское состояние:
  - Активные транзакции
  - Кэш запросов
- Все операции с данными происходят в контексте сессии
- Сессии должны переиспользоваться для достижения высокой производительности

Execute Data Query

```
message ExecuteDataQueryRequest {
    Ydb.Operations.OperationParams operation_params = 6; // Timeouts
    string session_id = 1; // Session identificator
    Query query = 3; // YQL query or query ID
    map<string, TypedValue> parameters = 4; // Query parameters
    QueryCachePolicy query_cache_policy = 5; // Store in query cache
    TransactionControl tx_control = 2; // Transaction selector
    StatsCollectionMode collect_stats = 7; // Enable stats collection
}
```

Transaction Control

```
message TransactionControl {
    oneof tx_selector {
        string tx_id = 1; // Use existing transaction
        TransactionSettings begin_tx = 2; // Begin new transaction with specified
          // settings
    }
    bool commit_tx = 10; // Commit existing transaction
}
```

Execute Query Result

```
message ExecuteQueryResult {
    repeated Ydb.ResultSet result_sets = 1; // Result sets (for each result)
    TransactionMeta tx_meta = 2; // Transaction meta information (id)
    QueryMeta query_meta = 3; // Query meta information (id, parameter types)
    Ydb.TableStats.QueryStats query_stats = 4; // Query execution statistics
}
```

Result Set

```
message ResultSet {
    repeated Column columns = 1; // Metadata of columns
    repeated Value rows = 2; // Rows of table, each row is a struct
    bool truncated = 3; // Flag indicates if the result was truncated
}
```

Эффективные транзакции

Размер транзакций

Эффективность выполнения транзакции
в первую очередь зависит от количества
и сложности выполняемых операций

Количество операций в транзакции

- Убедитесь что транзакция содержит только те операции, которые логически должны быть выполнены атомарно

Объем данных, затрагиваемых транзакцией

- Минимизируйте объем данных, читаемых / записываемых
в транзакции
- Избегайте больших сканов

Размер транзакций

- Количество шардов, затрагиваемых транзакцией (ширина)
  - Одношардовые транзакции наиболее эффективны
  - Обращайтесь к таблицам по префиксу PK или вторичного индекса, это позволит не задействовать лишние шарды
  - В случае операции JOIN убедитесь, что заданы предикаты
    на индексы левой и правой таблиц

Время жизни транзакции

> Минимизируйте время жизни транзакции, это позволит уменьшить вероятность её отката из-за инвалидации блокировок

- Минимизируйте клиентское взаимодействие с транзакцией
  - Используйте YQL для выражения логики над данными
  - Избегайте долгих клиентских вычислений в открытой транзакции
- Использование флагов BEGIN / COMMIT предпочтительнее явных вызовов
  - Отсутствие лишних хопов до кластера
  - Более эффективное выполнение последнего запроса в транзакции

Конкурентность по ключам

Высокая конкурентность по ключам может
ограничить масштабируемость системы

- Запросы в каждом из шардов выполняются последовательно
  - Шардируйте таблицы при высокой нагрузке
  - Распределяйте нагрузку по ключам таблицы
- При наличии конфликтов по ключам чтения / записи, только часть транзакций будут завершаться успешно
  - Избегайте больших чтений и высокой конкурентности по отдельным ключам таблицы

Повторы транзакций

Транзакции могут завершаться неуспешно
из-за инвалидации блокировок или других
серверных отказов

- Клиентское приложение ответственно за повтор неуспешных транзакций
- Клиентские SDK предоставляют функциональность для реакции на стандартные серверные ошибки
- Предпочитайте идемпотентные транзакции для упрощения логики повтора транзакций

Тайм-ауты запросов

Указывайте значения тайм-аутов для запросов, это позволит серверу не тратить ресурсы на их выполнение, если результат уже не важен для клиента

- Сервер попытается прервать выполнение запроса после истечения указанного временного интервала
- Позволяет увеличить устойчивость системы
к всплескам нагрузки и сбоям

Заключение

Yandex Database — геораспределенная отказоустойчивая база данных с механизмом строго консистентных транзакций

- Направлена на решение задач, требующих надежного хранения данных и высокой доступности
- Предоставляет SQL-like интерфейс для выполнения запросов
- Горизонтально масштабируется по нагрузке
- Предоставляется как сервис в Яндекс.Облаке

Ссылки

Yandex Database Private Preview	cloud.yandex.ru/services/ydb
Python SDK	github.com/yandex-cloud/ydb-python-sdk
Go SDK	github.com/yandex-cloud/ydb-go-sdk
Java SDK	github.com/yandex-cloud/ydb-java-sdk

Яндекс Облако

Спасибо!

Сергей Пучин

Руководитель группы распределенных запросов YDB

spuchin@yandex-team.ru

@spuchin