
Яндекс

Яндекс

Yandex Database: распределенные запросы в облаках

Сергей Пучин

BC BackendConf РИТ 2019

Содержание

Решаемые задачи

- OLTP нагрузка
  - Чтение / запись
  - Транзакционное выполнение
  - Низкие задержки
  - Большой TPS
- Использование SQL-like языка запросов
- Отказоустойчивость
- Высокая доступность
- Масштабируемость по нагрузке

Yandex Database

- *Yandex Database* - это георасперделенная 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

Use case: Турбо-страницы

- 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

Use case: Коллекции

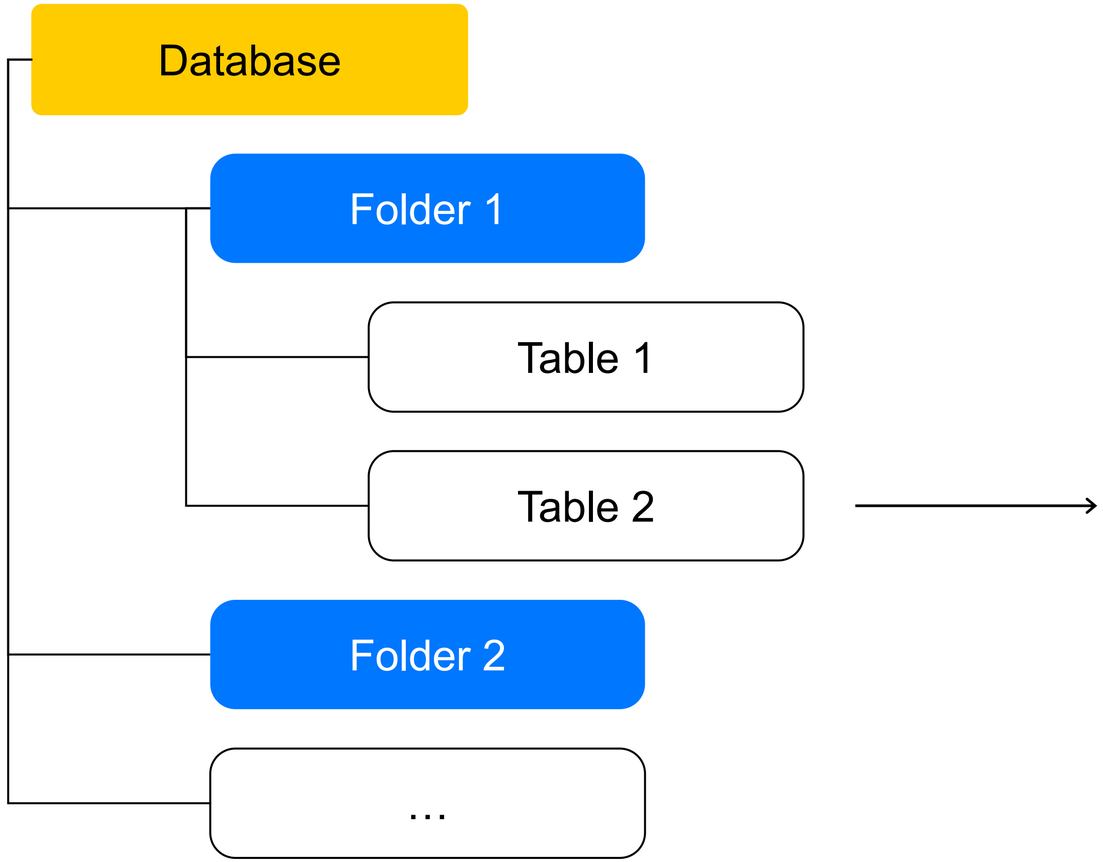
- Хранение истории о лентах рекомендаций для пользователей
  - Получение последних записей из истории пользователя
  - Добавление новых рекомендаций в историю
- Необходимость переживать потерю одного из датацентров
- Read-Write транзакции для получения и обновления истории
- Терабайты данных
- *1K*+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

Use case: 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

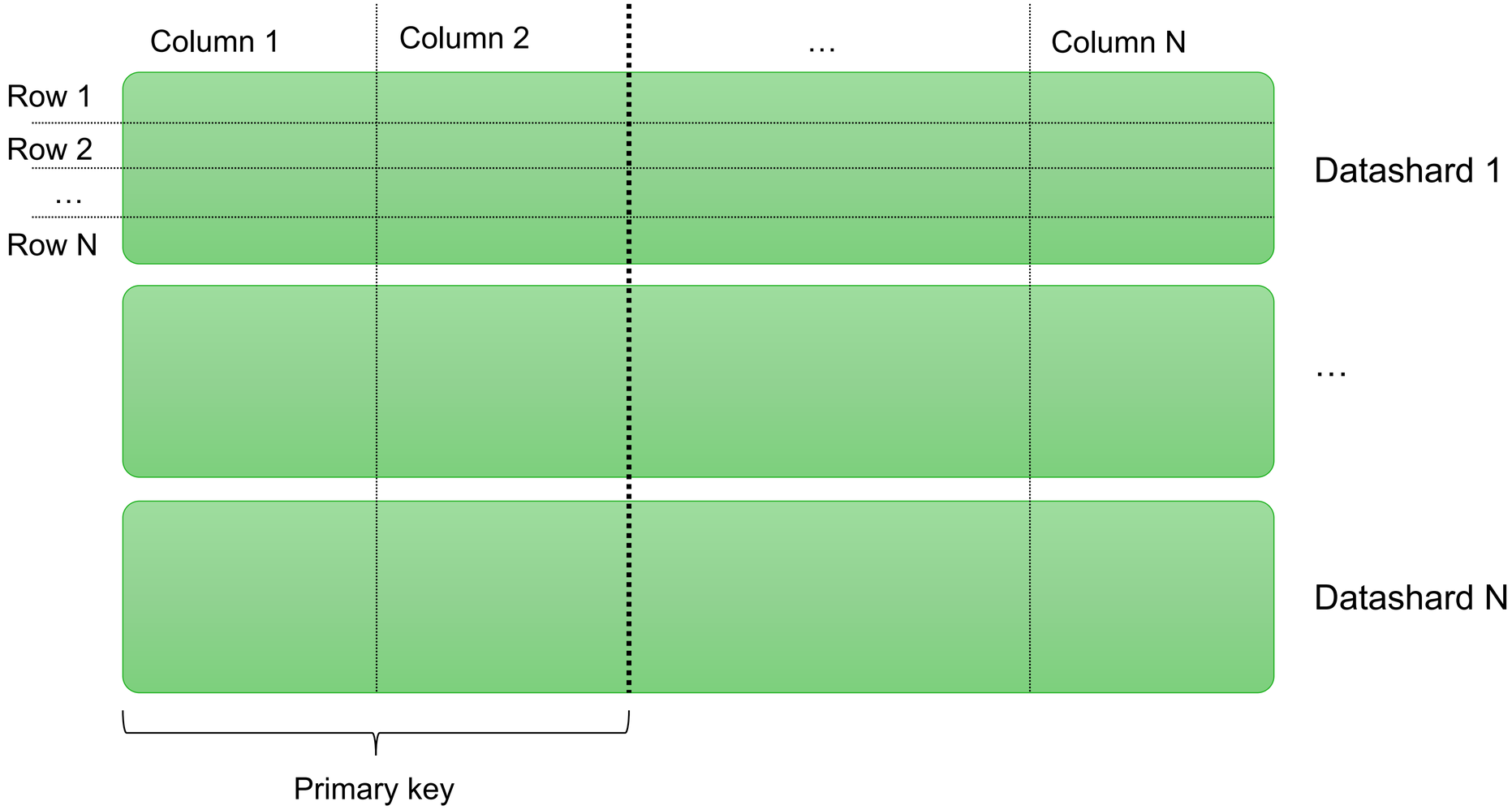


Metadata

- ACL
- Access time
- Size

- Columns
- Primary key
- Shards
- Statistics
- Configuration

Таблицы


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

Способы шардирования

- Uniform Partitioning
  - Разбиение на заданное количество шардов по равномерно распределенному 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

Yandex Database

Распределенные запросы

Клиентское взаимодействие

Yandex Database – сервис Яндекс Облака

- Data plane
  - GRPC + Proto3 API

- Клиентские библиотеки для популярных языков программирования
  - Python
  - Go
  - Java

Yandex Query Language (YQL)

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

- Специализированные DML конструкции:
  - UPSERT / REPLACE
  - UPDATE ON
  - DELETE ON

Yandex Query Language (YQL)

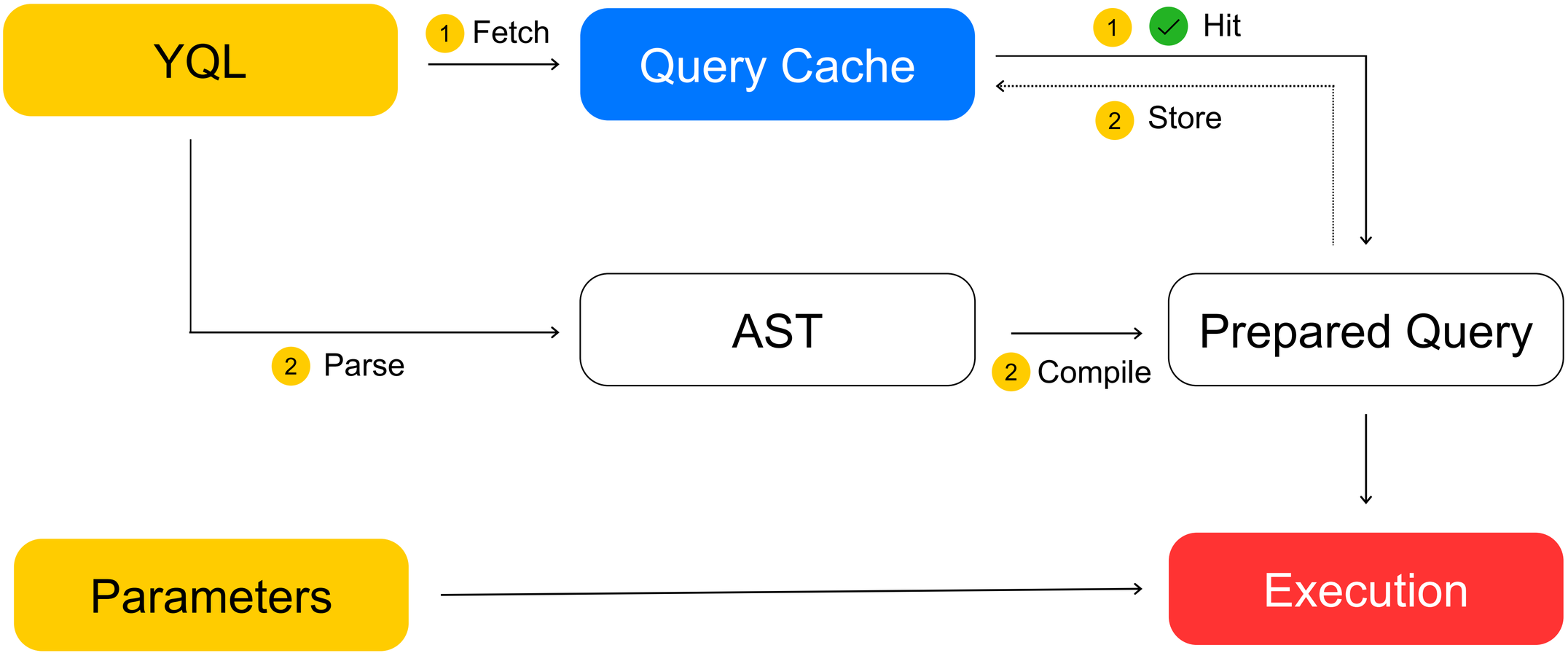
```
DECLARE $date AS Date;

DECLARE $series_id AS Uint64;


$title = (

    SELECT title FROM `series` WHERE series_id = $series_id

);


UPSERT INTO `dates`(date, title) VALUES

    ($date, $title);
```

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



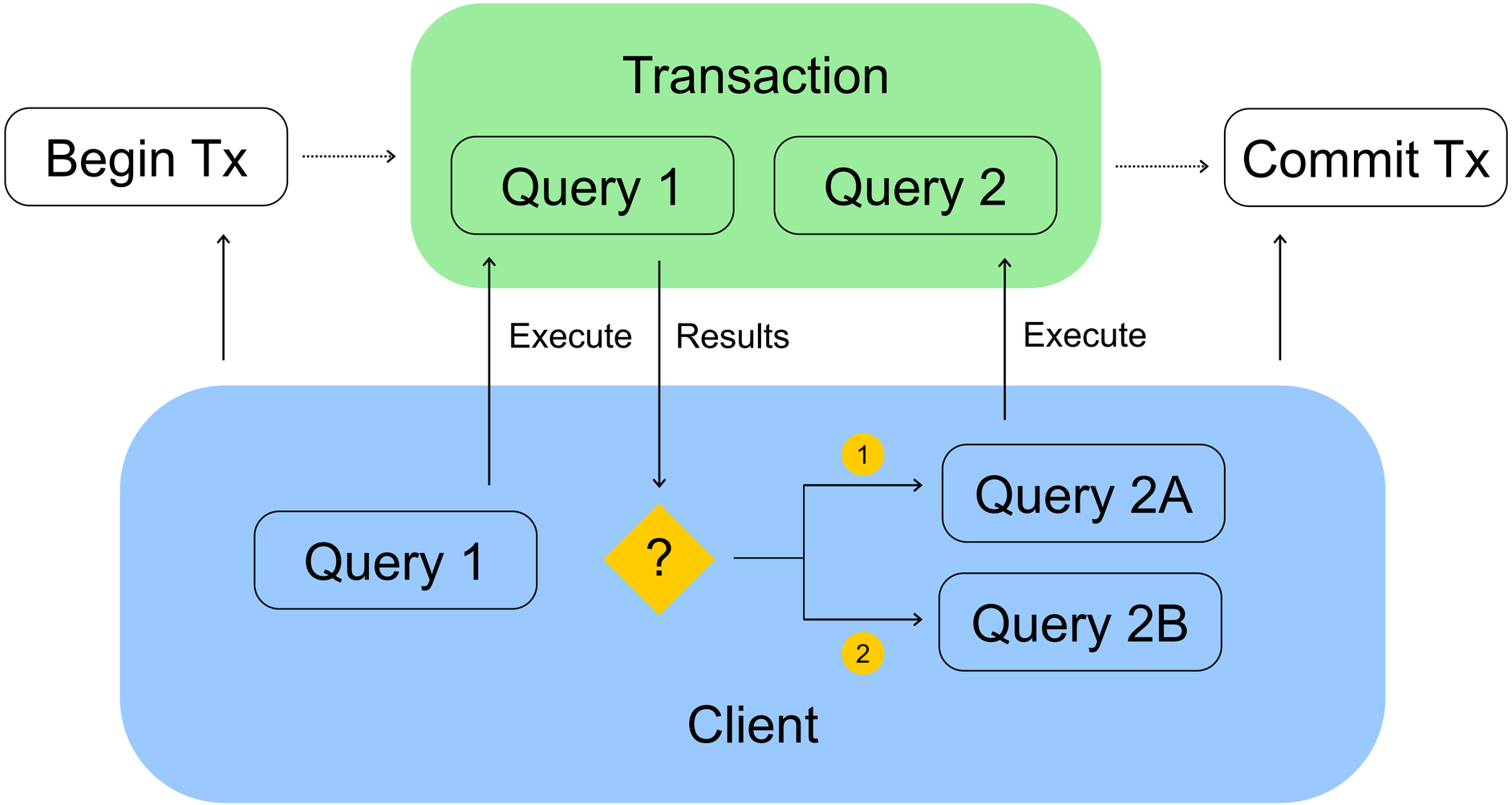
Транзакции

- 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



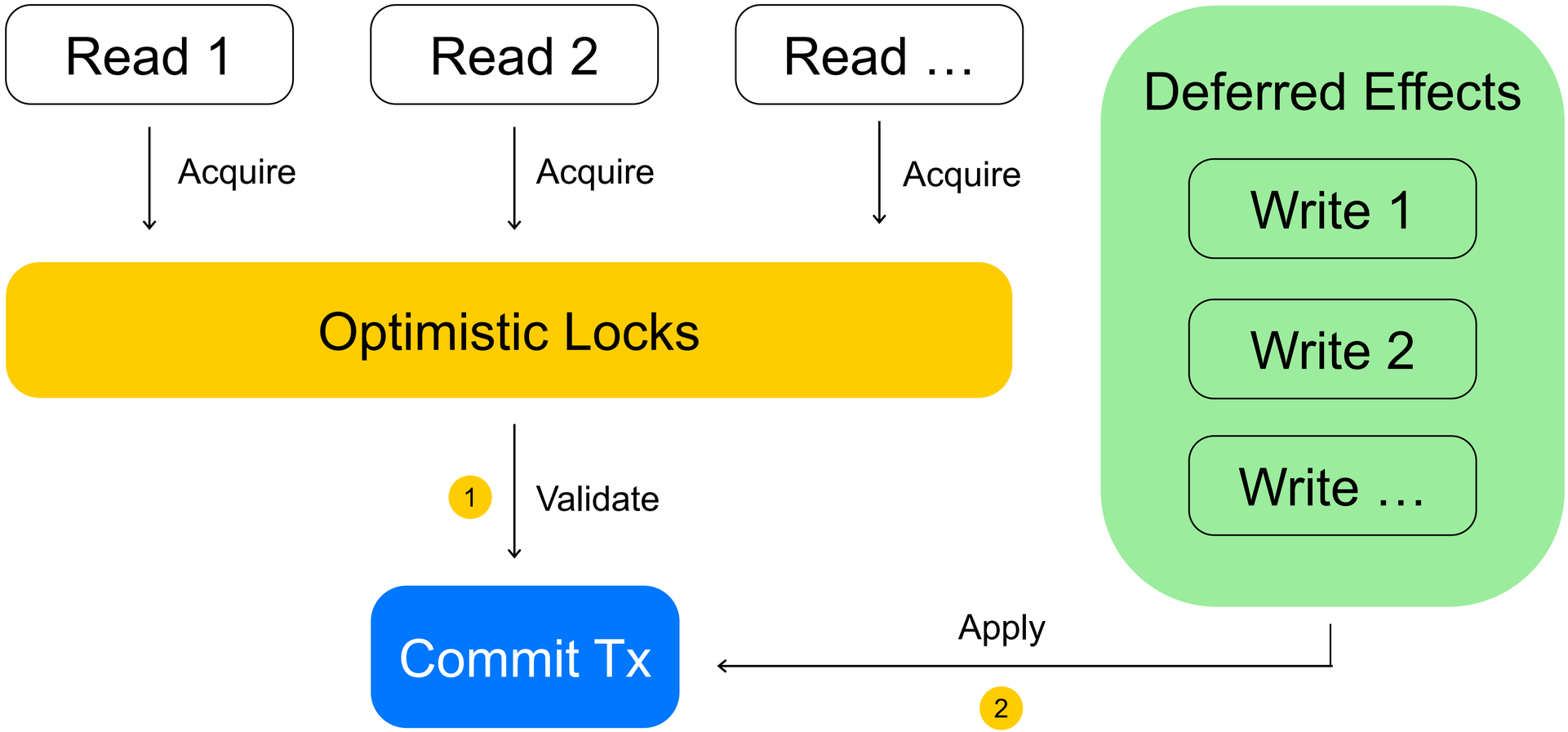
Оптимистичные блокировки

- 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



Yandex Database

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
- Размер ответа запросов
  - Размер ответа запроса не должен превышать *50 MB*
  - Если в ответе запроса содержится больше *1000* строчек, он будет обрезан, с установкой флага *Truncated*

Время жизни транзакции

Минимизируйте время жизни транзакции, это позволит уменьшить вероятность ее 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

Таймауты запросов

Указывайте значения таймаутов 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* – георасределенная отказоустойчивая база данных с механизмом строго консистентных транзакций**

- Подходит для решения задач, требующих надежного хранения данных и высокой доступности
- Использует YQL в качестве языка запросов
- Предоставляется как сервис в Яндекс Облаке

Ссылки

- Yandex Database Private Preview:
  https://cloud.yandex.ru/services/ydb

- Python SDK
  https://github.com/yandex-cloud/ydb-python-sdk
- Go SDK
  https://github.com/yandex-cloud/ydb-go-sdk
- Java SDK
  https://github.com/yandex-cloud/ydb-java-sdk

Яндекс

Спасибо

Сергей Пучин

Технический лидер, команда
распределенных запросов в YDB

 spuchin@yandex-team.ru

 @spuchin