МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Масштабирование БД

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Как сделать БД эффективнее?

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

1. Избавиться от лишних запросов

Лишние запросы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Запросы в цикле вместо массовых запросов
2. Джойны на стороне кода

ORM: проблема N+1

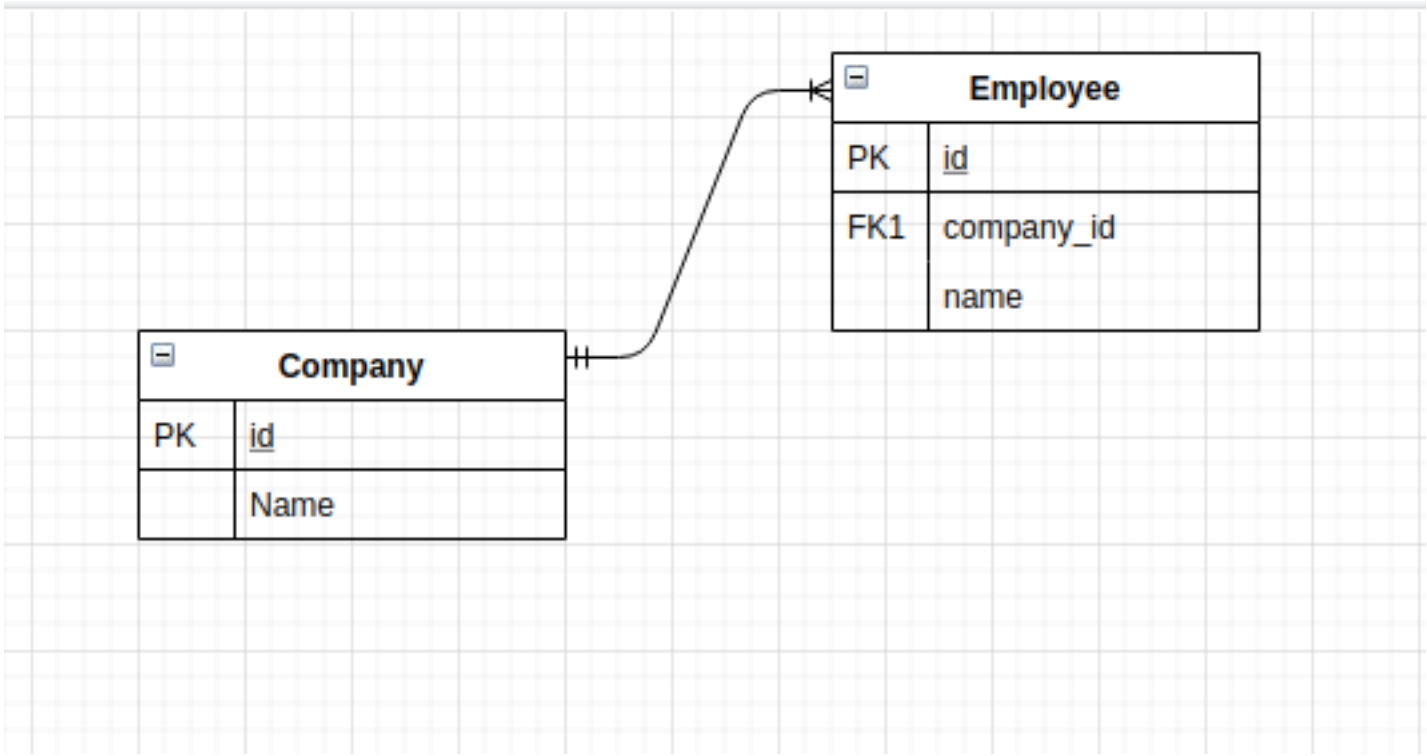
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

ORM — библиотеки, которые помогают работать с БД в терминах языка программирования без использования **SQL**.

Примеры **ORM**: **Hibernate**, **NHibernate**, **SQLAlchemy**.

ORM: проблема N+1

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



ORM: проблема N+1

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
var employees = Employees.GetByName(‘Ivan’);
foreach (var employee in employees)
{
    Console.WriteLine(employee.company.name);
}
```

Проблема N+1: SQL-запросы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
SELECT * FROM Employee WHERE name = ‘Ivan’;

SELECT * FROM Company WHERE id = 14;

SELECT * FROM Company WHERE id = 26;

SELECT * FROM Company WHERE id = 99;

SELECT * FROM Company WHERE id = 156;
```

Проблема N+1: единый SQL-запрос

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
SELECT *
FROM Employee e
JOIN Company c ON e.company_id = c.id
WHERE e.name = ‘Ivan’;
```

ORM: проблема N+1

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
var employees = Employees.Prefetch(Companies).GetByName('Ivan');

foreach (var employee in employees)

{

    Console.WriteLine(employee.company.name);

}
```

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

2. Оптимизация запросов

Оптимизация запросов

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

EXPLAIN (ANALYZE)

EXPLAIN ANALYZE

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
EXPLAIN ANALYZE SELECT *
FROM tenk1 t1, tenk2 t2
WHERE t1.unique1 < 10 AND t1.unique2 = t2.unique2;
```

EXPLAIN ANALYZE

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
QUERY PLAN
-------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=4.65..118.62 rows=10 width=488) (actual time=0.128..0.377 rows=10 loops=1)
   ->  Bitmap Heap Scan on tenk1 t1  (cost=4.36..39.47 rows=10 width=244) (actual time=0.057..0.121 rows=10
loops=1)
         Recheck Cond: (unique1 < 10)
         ->  Bitmap Index Scan on tenk1_unique1  (cost=0.00..4.36 rows=10 width=0) (actual time=0.024..0.024
rows=10 loops=1)
               Index Cond: (unique1 < 10)
   ->  Index Scan using tenk2_unique2 on tenk2 t2  (cost=0.29..7.91 rows=1 width=244) (actual time=0.021..0.022
rows=1 loops=10)
         Index Cond: (unique2 = t1.unique2)
 Planning time: 0.181 ms
 Execution time: 0.501 ms
```

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

3. Индексы

Индексы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Индекс — структура, содержащая данные из таблицы БД, оптимизированные под поиск по определенному запросу.

Индексы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
SELECT *
FROM Employee
WHERE first_name = ‘Ivan’;
```

Индексы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
SELECT *
FROM Employee
WHERE first_name = ‘Ivan’
      AND last_name = ‘Ivanov’;
```

Индексы из нескольких колонок

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Для некоторых запросов нужны индексы из нескольких колонок.

Такие **индексы** работают как телефонная книга:

- **помогают найти** всех Ивановых
- **помогают найти** всех Ивановых Иванов
- не помогают найти всех Иванов Ивановичей

Индексы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Индексы ускоряют получение данных, но замедляют вставку, изменение и удаление, потому что данные нужно менять сразу в нескольких местах.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

4. Денормализация

Нормальные формы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Все данные хранятся в своих таблицах, в БД отсутствует дублирование данных.

Денормализация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Намеренное приведение структуры базы данных в состояние, не соответствующее критериям нормализации, обычно проводимое с целью ускорения операций чтения из базы за счет добавления избыточных данных.

Запрос к нормализованной БД

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
SELECT e.id, c.name
FROM Employee e
JOIN Company c ON e.company_id = c.id
WHERE e.name = ‘Ivan’;
```

Нормализованная БД

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Employee

id
name
company_id

Денормализуем БД

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Employee

id
name
company_id
company_name

Запрос к денормализованной БД

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
SELECT id, company_name
FROM DenormalizedEmployee
WHERE name = ‘Ivan’;
```

Materialized views

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Материализованные представления — физические объекты базы данных, содержащие результаты выполнения запросов.

Запросы к **материализованным представлениям** выполняются очень быстро.

Materialized views

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Материализованные представления так же, как и индексы, замедляют изменение данных в БД и занимают дополнительное место на диске.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

5. Репликация

Репликация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Хранение копий одних и тех же данных на нескольких машинах, соединенных с помощью сети.

Цели репликации

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. **Географическое распределение**
2. **Повышение надежности**
3. **Повышение пропускной способности**

Виды репликации

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. С одним ведущим узлом
2. С несколькими ведущими узлами
3. Без ведущих узлов

Ведущие и ведомые узлы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

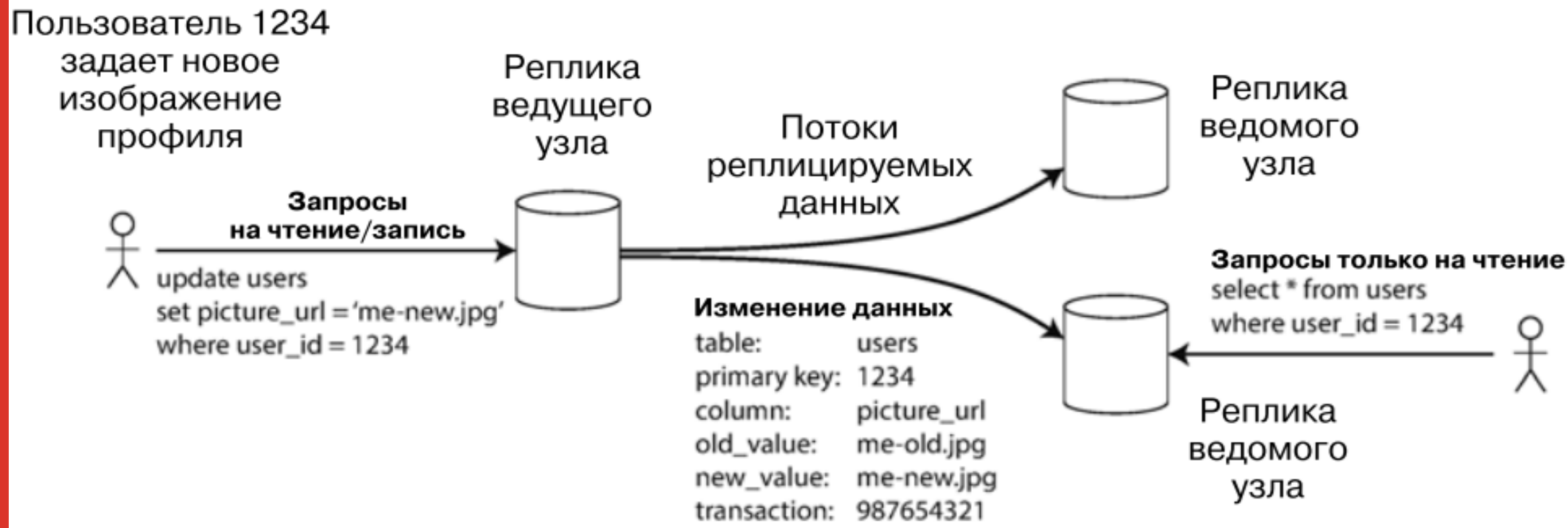
Ведущий узел (*leader, master*) — принимает запросы на запись.

Ведомый узел (*follower, slave*) — принимает от ведущего узла поток изменений и обновляет свою копию данных.

И ведущий, и ведомые узлы могут принимать запросы на чтение.

Ведущие и ведомые узлы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



Синхронная и асинхронная репликация

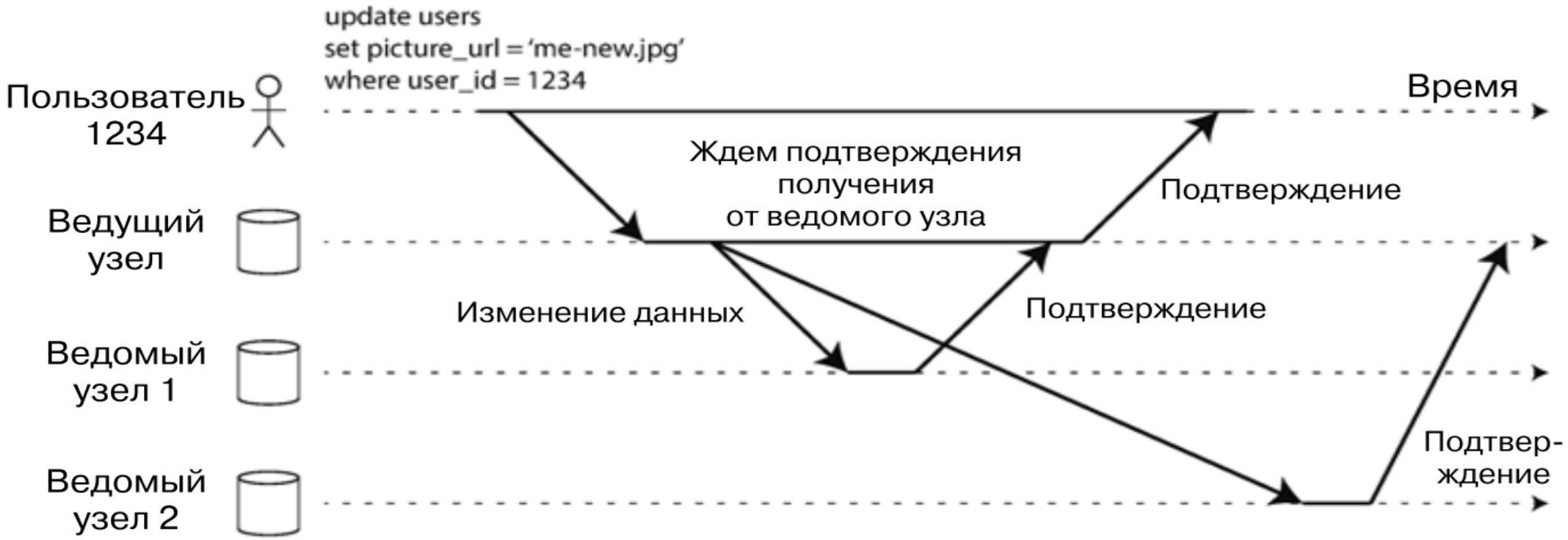
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Синхронная репликация — ведущий узел ждет подтверждения от ведомого прежде чем подтвердить клиенту сохранение данных и сделать их видимыми.

Асинхронная репликация — ведущий узел отправляет сообщение ведомому, но не ждет ответа.

Синхронная и асинхронная репликация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



Синхронная и асинхронная репликация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Преимущество синхронной репликации: данные на реплике гарантированно актуальны и согласованы с ведущим узлом.

Недостаток синхронной репликации: если реплика недоступна, приходится блокировать все попытки записи на ведущий узел.

Восстановление после отказов: ведомый узел

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Ведомый узел хранит журнал полученных изменений в данных.

После сбоя он может определить, какую последнюю операцию он обработал, и запросить у ведущего узла все более новые операции.

Восстановление после отказов: ведущий узел

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Понять, что ведущий узел отказал
2. Выбрать новый ведущий узел
3. Переключить систему на использование нового ведущего узла

Проблемы при восстановлении

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Не все операции успели реплицироваться
2. Рассинхронизация с внешними системами
3. Несколько узлов могут начать считать себя ведущими
4. Непонятно, сколько ждать прежде чем объявить ведущий узел недоступным

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Реализация журналов репликации

Операторная репликация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Ведущий узел записывает в журнал все входящие запросы на запись (**INSERT, UPDATE, DELETE**) и отправляет этот журнал всем ведомым узлам.

Ведомые узлы выполняют эти запросы так, будто они пришли от клиента.

Плюсы операторной репликации

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Просто и очевидно.

Минусы операторной репликации

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1) **Недетерминированные функции** (NOW() или RAND()) будут генерировать разные результаты для каждой реплики
2) Если запросы зависят от существующих значений (UPDATE … WHERE, столбец с автоинкрементом), то нужно гарантировать одинаковый порядок выполнения запросов

WAL

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Write-ahead log — журнал, в который БД заносит изменения в данных до того, как внести их в сами данными. Благодаря **WAL**, можно не сохранять данные на диск после каждого изменения, но иметь возможность восстановить актуальное состояние данных после сбоя.

Репликация с помощью переноса WAL

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Ведущий узел записывает WAL не только на диск, но и отправляет его по сети ведомым узлам.

Ведомые узлы создают у себя точные копии тех же структур данных, что и на ведущем.

Репликация с помощью переноса WAL

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

WAL описывает данные на уровне байтов на дисковых блоках. Это тесно связывает репликацию и хранение данных.

Из-за этого невозможно использовать различные версии БД на разных узлах. При обновлении требуется останавливать всю систему

Логическая репликация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Логический журнал содержит операции записи в БД на уровне строк:

- Значения всех столбцов при добавлении строк
- Информация, необходимая для идентификации, при удалении строки
- Информация, необходимая для идентификации, и новые значения столбцов при обновлении строки

Логическая репликация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Поскольку логический журнал расцеплен с внутренним устройством подсистемы хранения, оказывается проще поддерживать его обратную совместимость

Триггерная репликация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Самый гибкий способ.

С помощью триггеров изменяемые данные заносятся в отдельную таблицу. Оттуда их читает внешний процесс и переносит в другую БД.

Внешний процесс может производить любые действия с данными: например, выбрать только часть данных, или писать их в БД другого типа.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Проблемы задержки репликации

Проблемы задержки репликации

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Репликация помогает масштабировать и ускорить запросы на чтение.

Но при чтении данных с асинхронных реплик есть вероятность получить устаревшую информацию, если этот узел запаздывает.

Eventual consistency

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Когда узел запаздывает, возникает **несогласованность**: состояние системы различается, когда данные читаются с актуального ведущего узла и когда они читаются с отстающего ведомого узла.

Такая ситуация называется ***конечной согласованностью***, потому что рано или поздно ведомый узел получит данные.

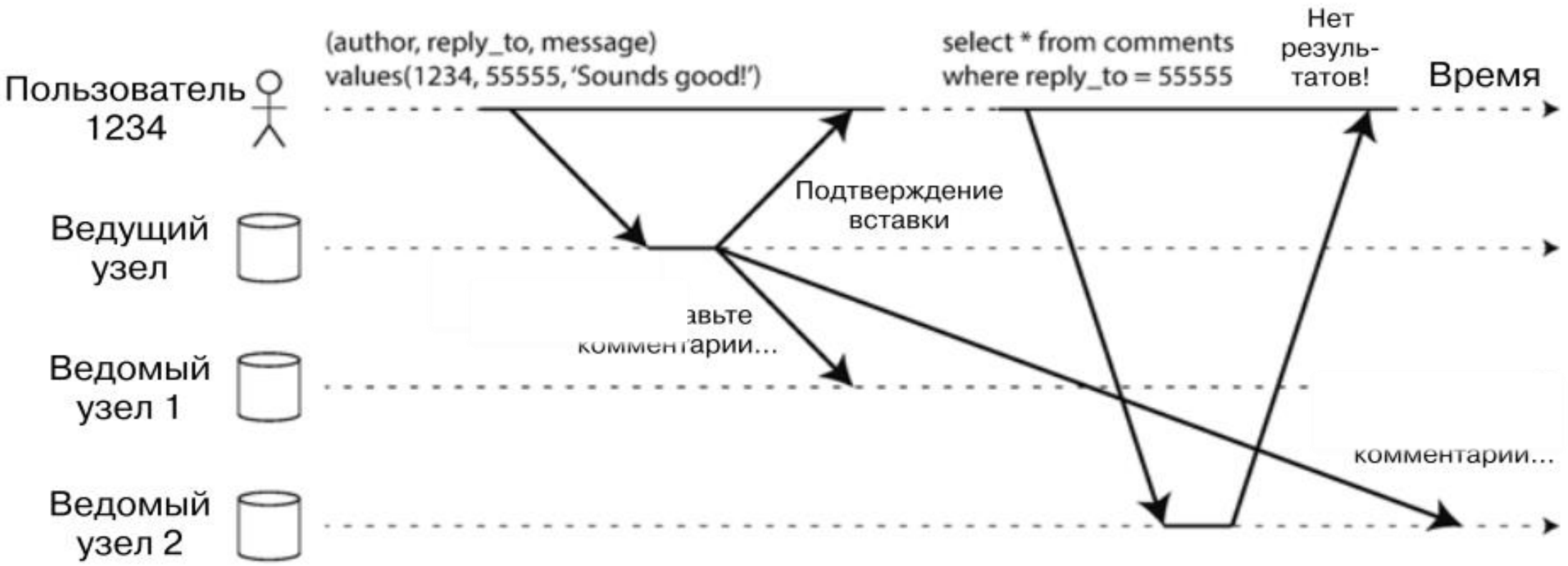
Чтение своих записей

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Проблема: **пользователь отправляет данные, но не видит, что они сохранились**

Чтение своих записей

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



Решения проблемы чтения своих записей

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Данные, которые пользователь может менять, читаются с ведущего узла, остальные с ведомых.
- После того, как пользователь что-то поменяет, в течение одной минуты читать данные только с ведущего узла.
- Запоминать id или время последнего изменения от пользователя и взаимодействовать только с репликами, на которые эти данные уже распространились.

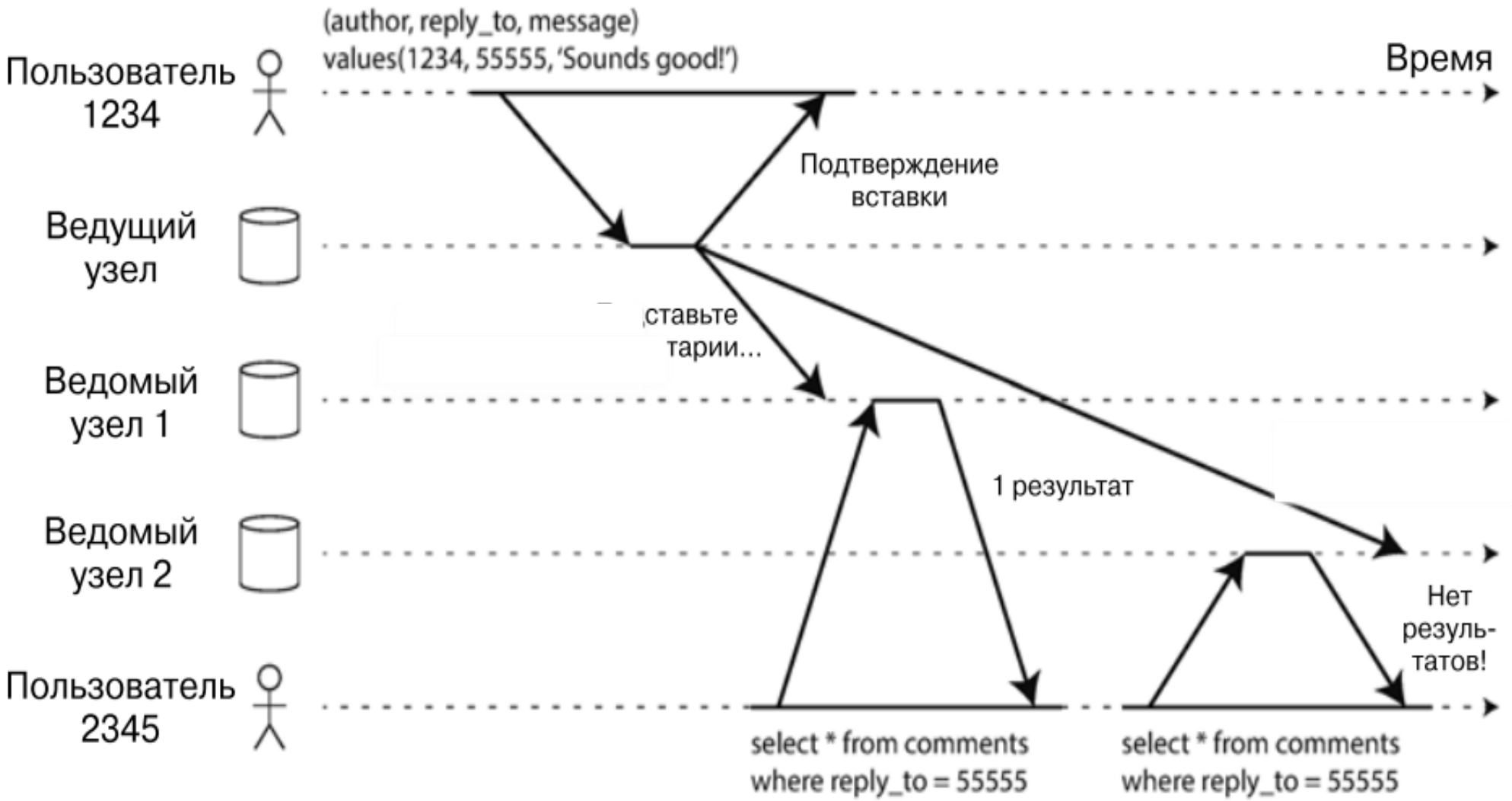
Монотонные чтения

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Проблема: **пользователь видит данные, потом они пропадают, а потом снова появляются**.

Монотонные чтения

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ



Решение проблемы монотонных чтений

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Прикреплять пользователя к определенной реплике и читать данные только из нее.

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Репликация с несколькими ведущими узлами

Несколько ведущих узлов

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

В схеме с несколькими ведущими узлами есть несколько узлов, которые поддерживают запись на них, а потом распространяют данные на другие узлы.

Каждый ведущий является ведомым для других ведущих узлов.

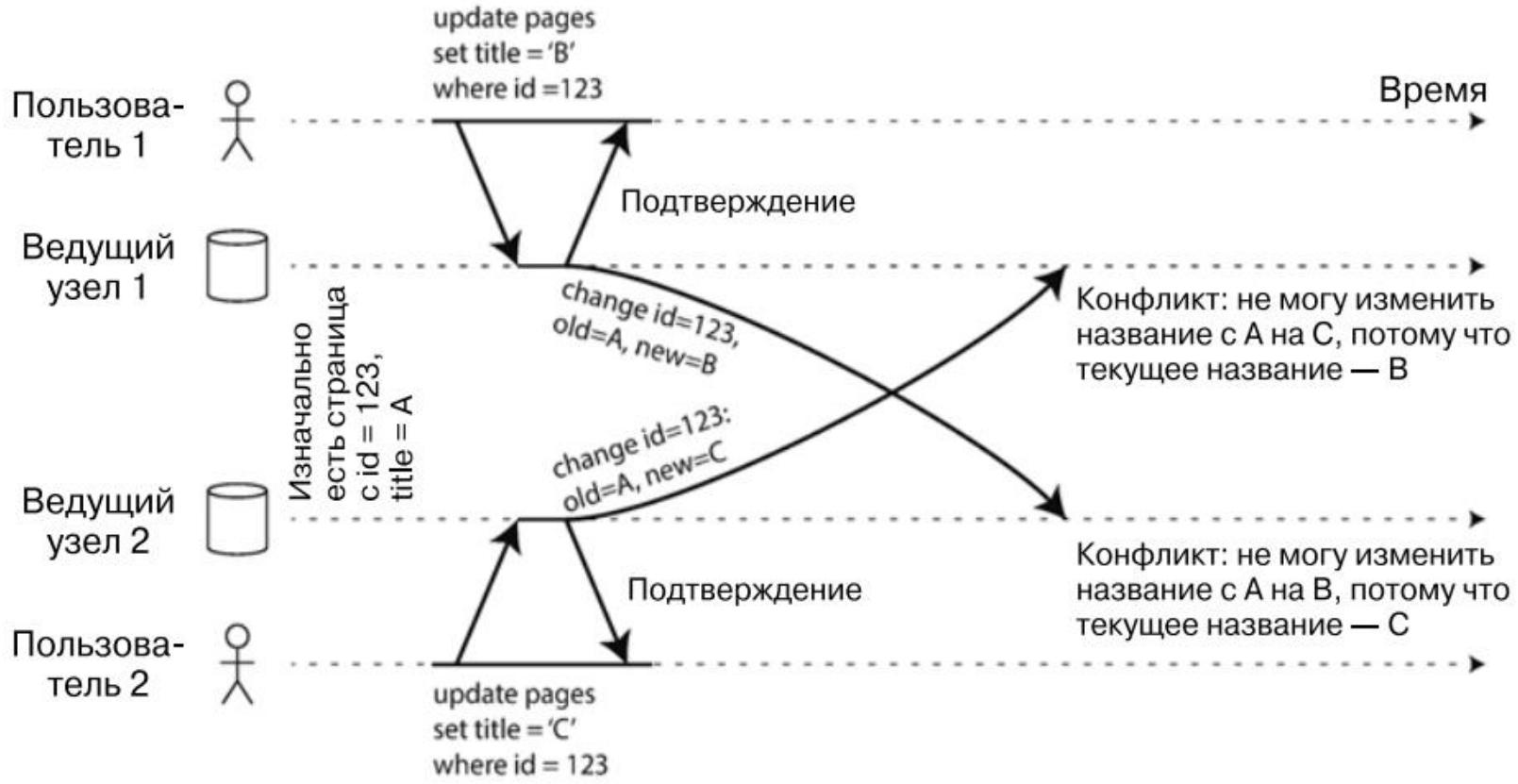
Проблема нескольких ведущих узлов

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Если одни и те же данные были одновременно отредактированы в нескольких ведущих узлах, возникает конфликт.

Проблема нескольких ведущих узлов

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



Предотвращение конфликтов

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Для каждого объекта выбирается «родной» узел, изменение этого объекта происходит только через этот узел.

Сходимость к согласованному состоянию

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

В каждой реплике операции применяются в собственном порядке. Поэтому непонятно, каким должно быть финальное значение. Но каждая реплика должна быть согласованной в конечном итоге, поэтому должен быть алгоритм для получения итогового состояния.

Сходимость к согласованному состоянию

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Last write wins: каждой операции присваивается метка времени или случайный идентификатор, победитель выбирается по наибольшему значению
2. Каждой реплике присваивается свой номер, данные из реплики с большим номером имеют больший приоритет
3. Значения объединяются в одно общее значение (например, B/C)
4. Все конфликтующие значения сохраняются в БД, при чтении данных приложение выбирает победителя или предлагает пользователю выбрать его

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

6. Шардинг

Шардинг

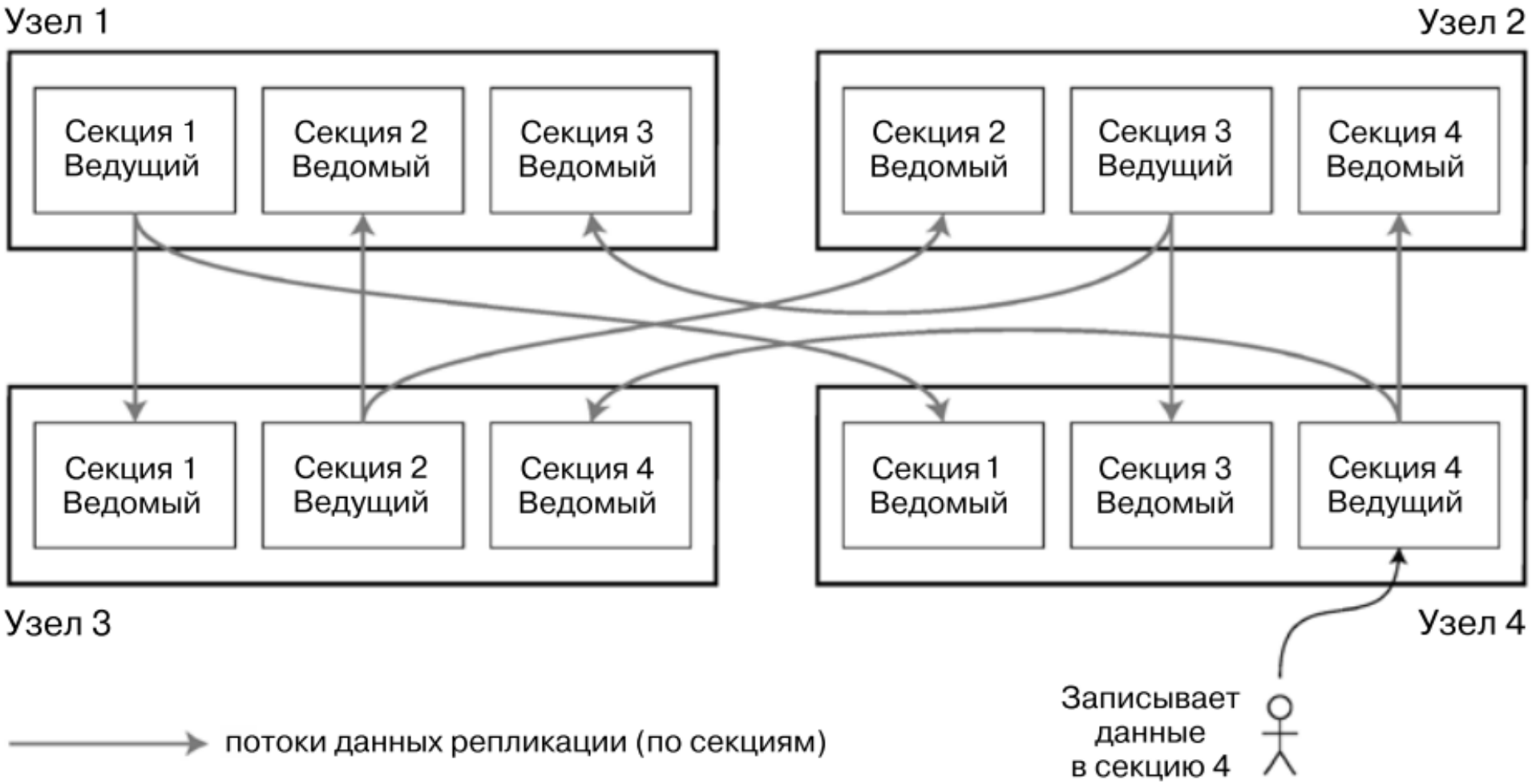
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Шардинг (секционирование) — разделение данных между несколькими узлами.

Шардинг позволяет работать с большими наборами данных, которые не помещаются на один узел.

Шардинг + репликация

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



Распределение данных

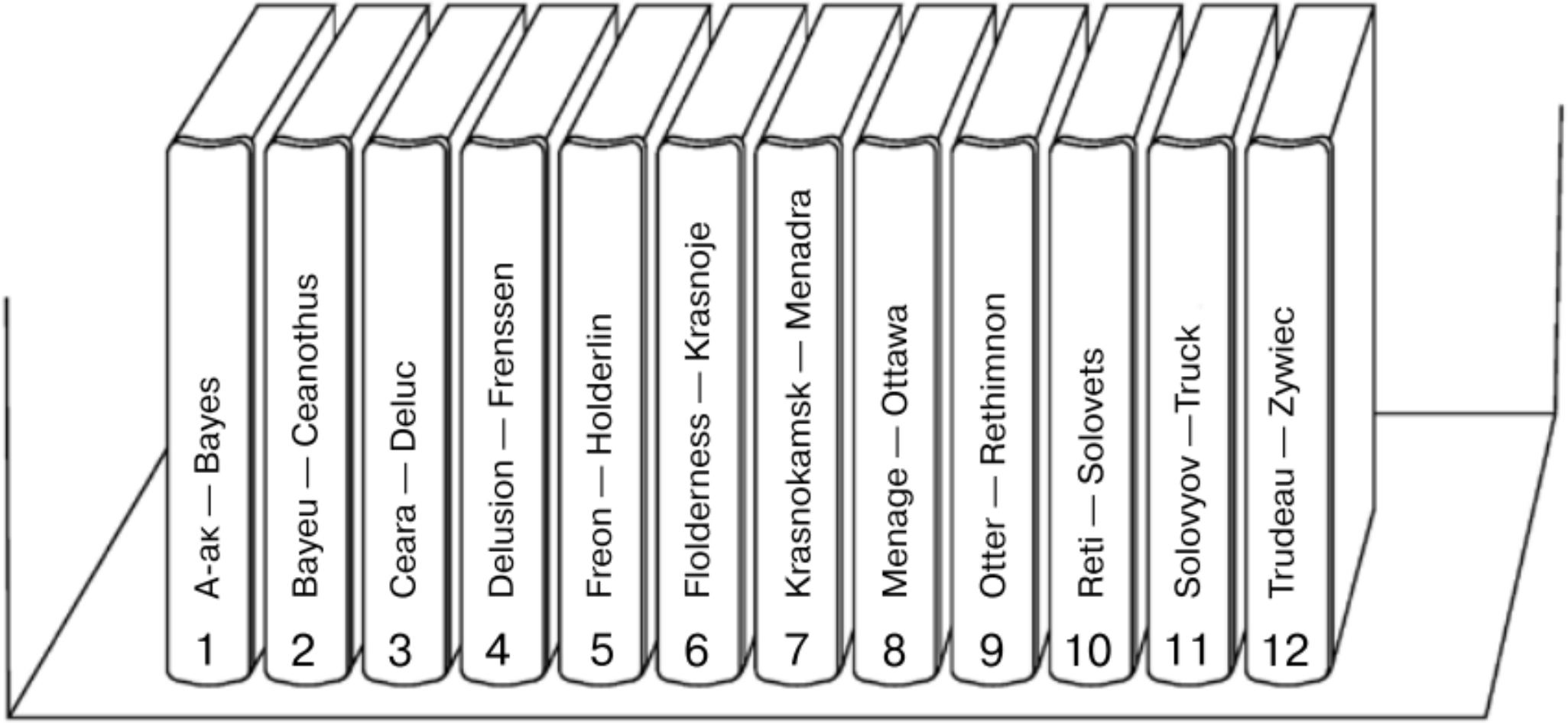
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Задача шардинга — максимально равномерно распределить данные по узлам.

Если взять 10 узлов так и распределить данные так, чтобы каждый получал 1/10 всей нагрузки, мы теоретически увеличим пропускную способность системы в 10 раз.

Секционирование по диапазонам ключа

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ



Секционирование по хешу ключа

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Хеш-функция получает на вход асимметричные значения и выдает равномерно распределенные значения, которые можно использовать для определения секции.

Плюс: гарантировано равномерное распределение

Минус: невозможно эффективно выполнить запрос по диапазону

Вторичные индексы

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Вторичный индекс необходим, когда записи выбираются не по первичному ключу, а по другим полям.

Два вида секционирования со вторичными индексами:

- **По документам**
- **По термам**

Секционирование по документам

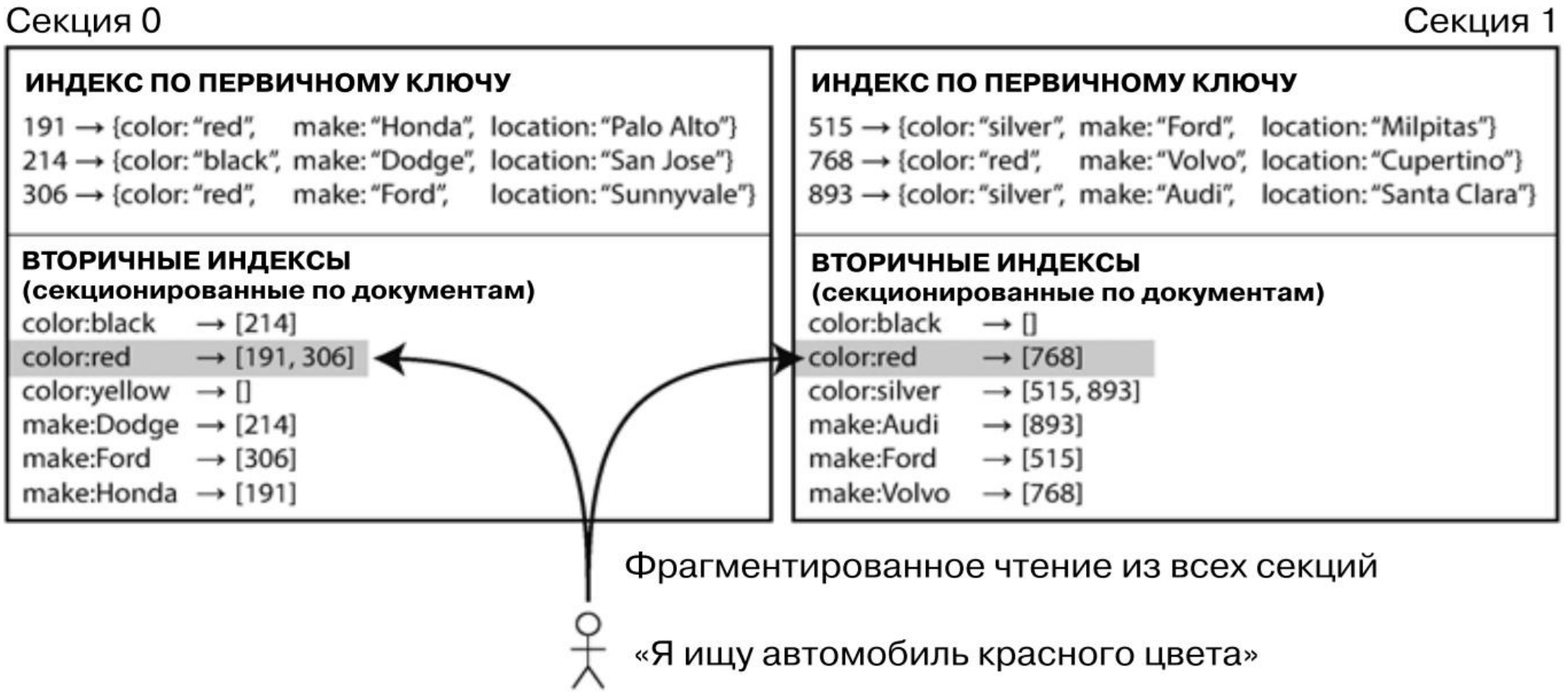
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Каждая секция поддерживает локальные индексы с хранящимися в них записями.

При поиске отправляются запросы на все секции, а затем результаты выполнения запросов объединяются.

Секционирование по документам

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



Секционирование по термам

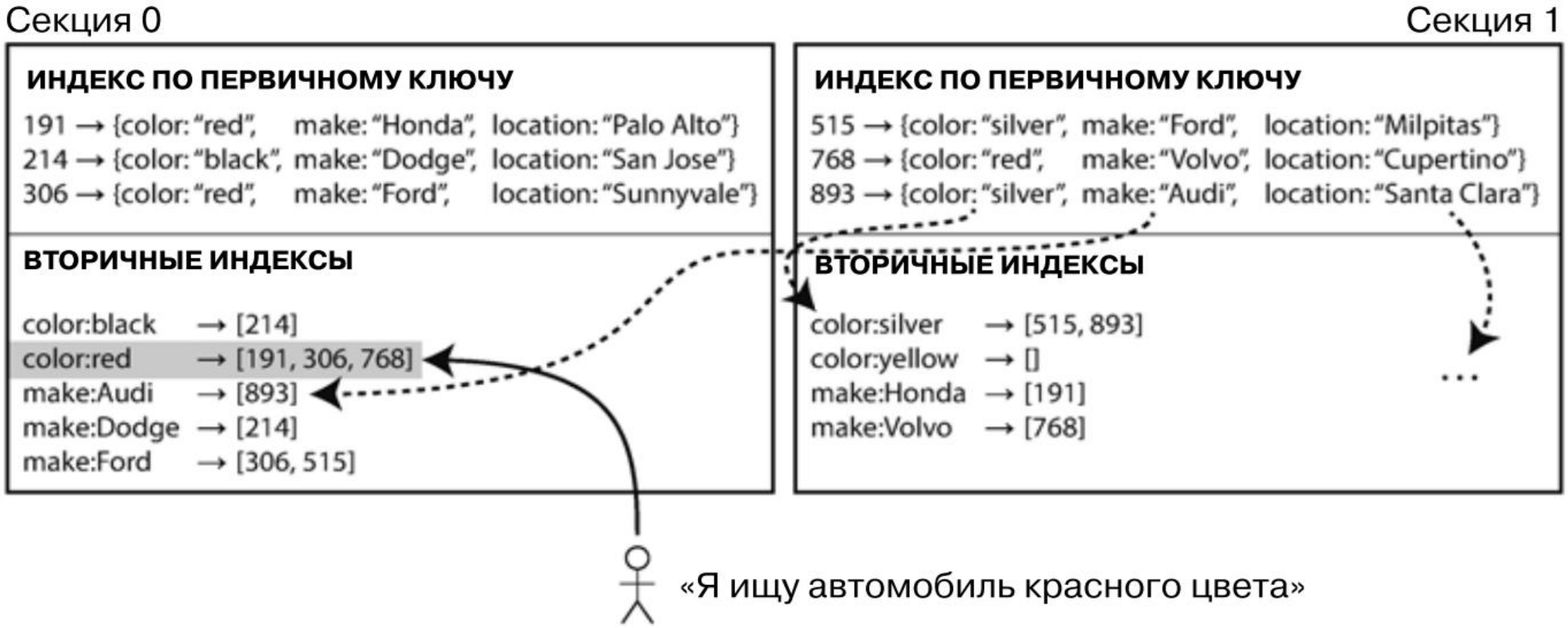
МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Строится глобальный индекс, содержащий значения из всех секций. Чтобы он не стал узким местом, он тоже секционируется.

Это упрощает чтение, но усложняет запись, поскольку одно изменение может затрагивать несколько индексов, находящихся в разных секциях.

Секционирование по термам

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ



Перебалансировка данных

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Со временем:

- **Данных становится больше**
- **Появляются новые узлы**
- **Узлы выходят из строя**

Из-за этого требуется перемещение данных и перенаправление запросов между узлами.

Перебалансировка: хеширование по модулю N

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Если у нас есть десять секций, можно выбирать нужную по формуле: hash(key) % 10.

Но в таком случае при изменении количества узлов большинство ключей придется перенести в другой узел.

Поэтому используются диапазоны значений ключа.

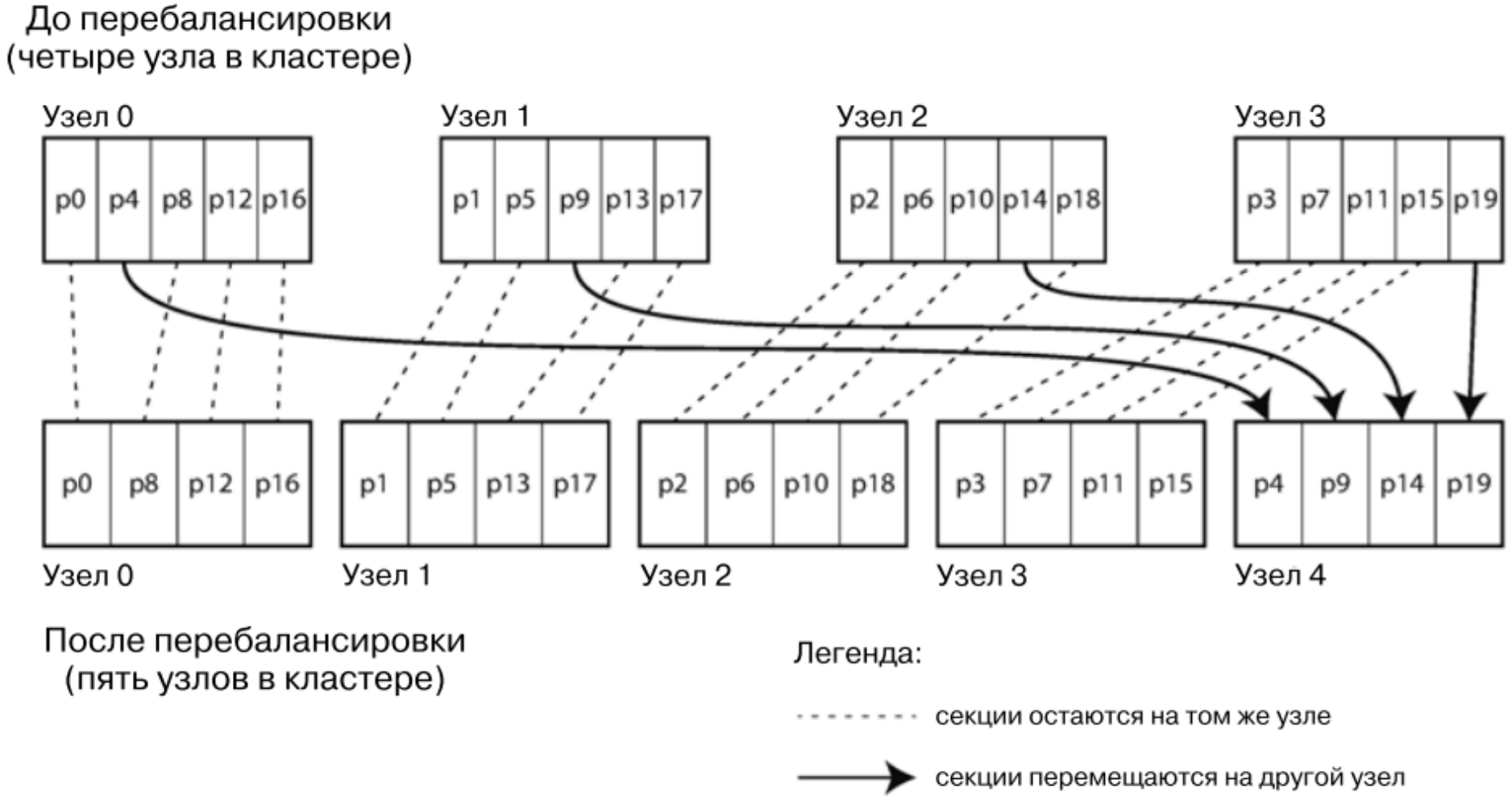
Перебалансировка: заранее созданные секции

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Можно заранее создать больше секций, чем узлов, и разместить на каждом узле несколько секций.

Тогда при добавлении нового узла можно перенести часть существующих секций на новый узел.

Перебалансировка: заранее созданные секции



Плюсы заранее созданных секций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Количество секций и распределение ключей между секциями не меняется
- Во время переноса данных используется старое распределение секций

Минусы заранее созданных секций

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

- Тяжело подобрать идеальный размер секции
- Тяжело определить границы между секциями, чтобы данные равномерно распределились между ними

Перебалансировка: динамическое секционирование

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

БД сама отслеживает размер секции и разделяет ее на две при достижении определенного размера (или сливает две секции в одну при удалении данных).

После разделения секций одну из них можно перенести на другой узел для балансировки нагрузки.

Перебалансировка: пропорциональное секционирование

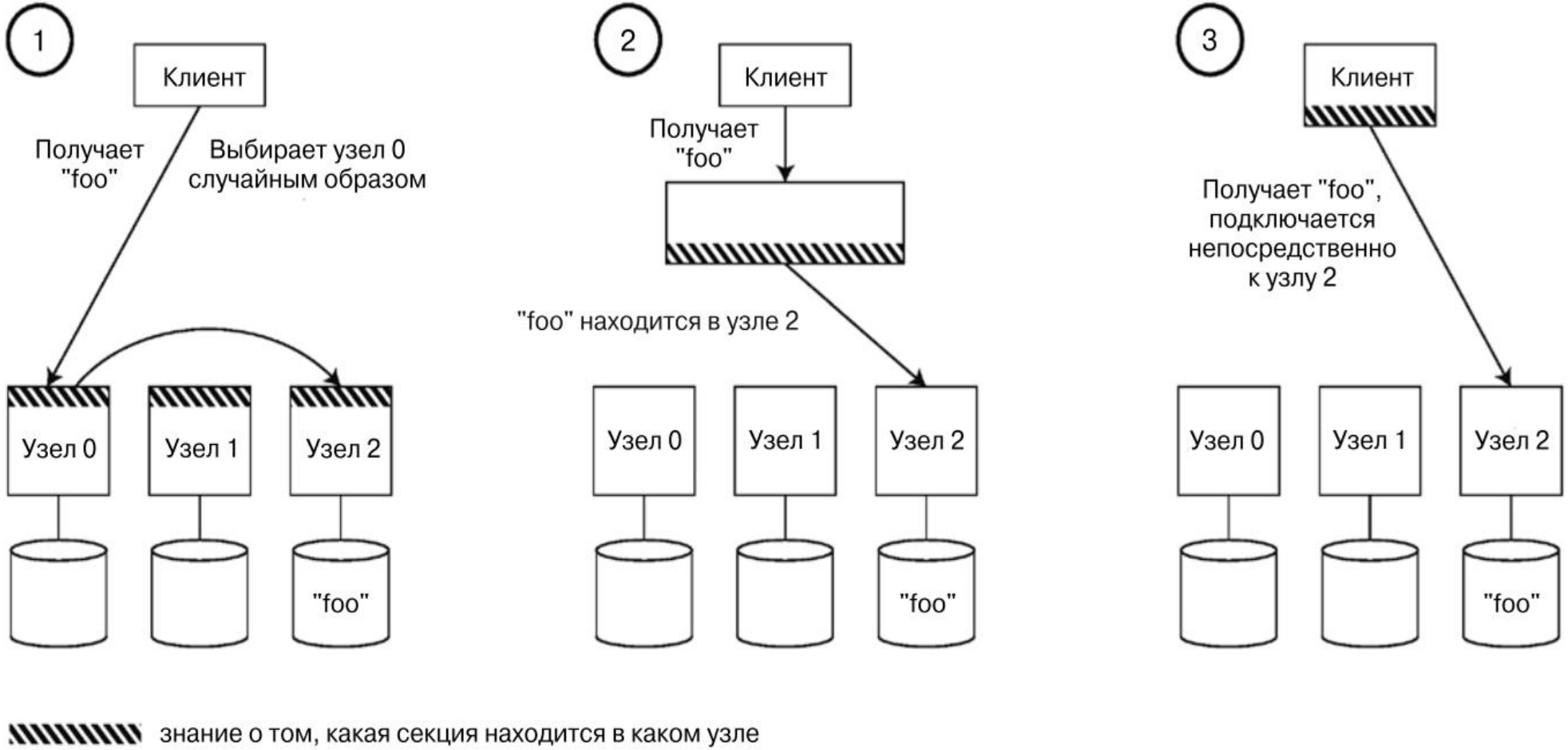
На каждом узле хранится фиксированное количество секций.

Новый узел выбирает случайные существующие секции и забирает половину данных оттуда.

Маршрутизация запросов

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Как клиент понимает, к какому узлу ему подключиться?


1
Клиент
Получает "foo"
Выбирает узел 0 случайным образом
Узел 0
Узел 1
Узел 2
"foo"
2
Клиент
Получает "foo"
"foo" находится в узле 2
Узел 0
Узел 1
Узел 2
"foo"
3
Клиент
Получает "foo", подключается непосредственно к узлу 2
Узел 0
Узел 1
Узел 2
"foo"
знание о том, какая секция находится в каком узле

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Ссылки

Литература

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Ссылки

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Шардинг: паттерны и антипаттерны
https://habr.com/ru/company/oleg-bunin/blog/313366/

МОСКОВСКИЙ
ГОРОДСКОЙ
УНИВЕРСИТЕТ
МГПУ

СПАСИБО ЗА ВНИМАНИЕ