



Как шардировать базу данных и не пожалеть об этом



Иван Дедов

Ведущий разработчик информационных систем



Для кого этот доклад?



- ⚙ Вы — разработчик, отвечающий за систему, хранящую большой объём данных
- 👁 Вы хотите увидеть кейсы применения технологий на практике
- 🎓 Вам интересно узнать что-то новое

ПЛАН

База о шардировании базы

Создаём шардированную базу данных с нуля

Шардируем то, что уже работает

Ломаем шардированием то, что уже работает

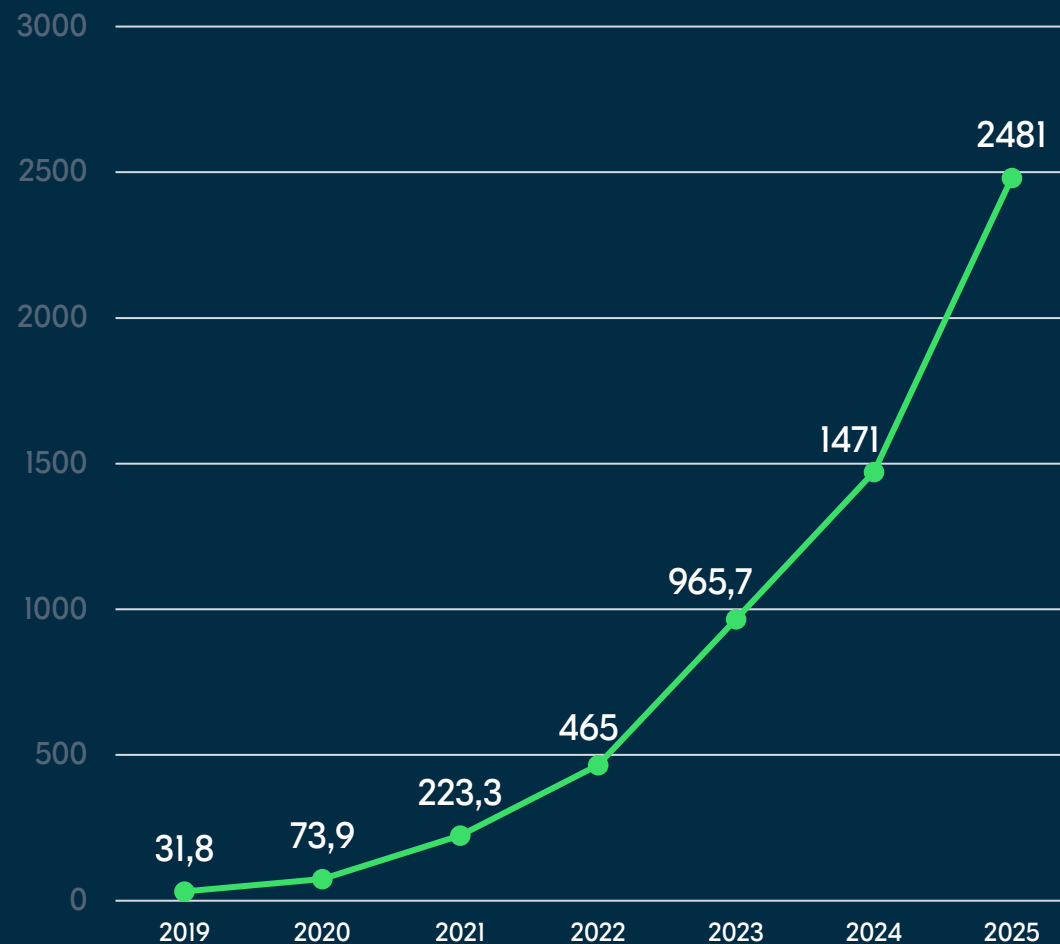
Делаем выводы



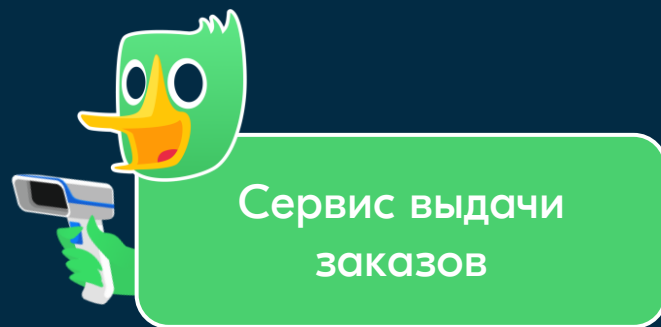
С чего всё начиналось

- ↑ Число заказов и клиентов маркетплейса постоянно растёт
- ↑ Объёмы хранимых и обрабатываемых данных увеличиваются
- 🔊 Архитектурный комитет Ozon принял решение об **обязательном шардировании баз данных**

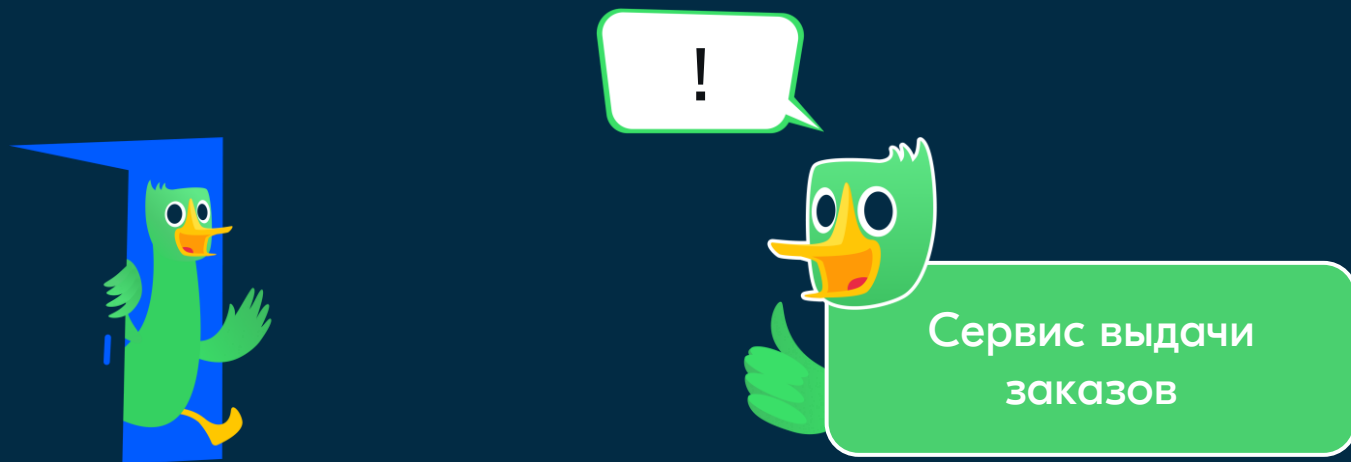
Количество доставленных заказов, млн



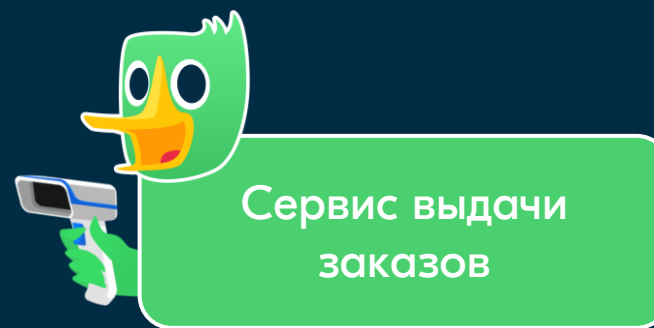
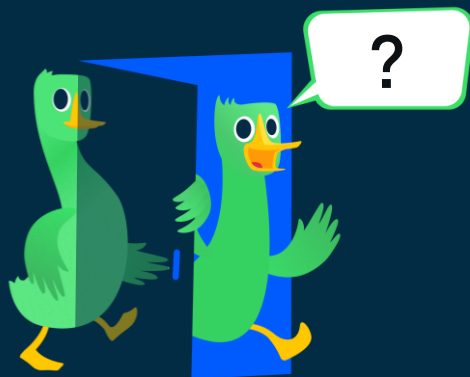
Масштабирование системы



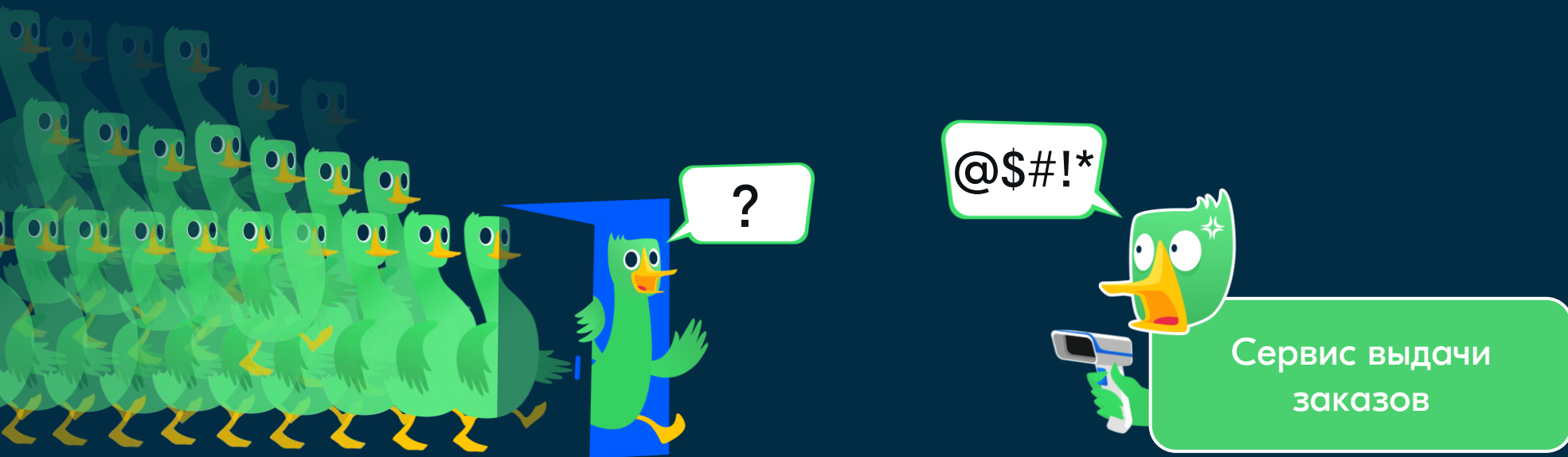
Масштабирование системы



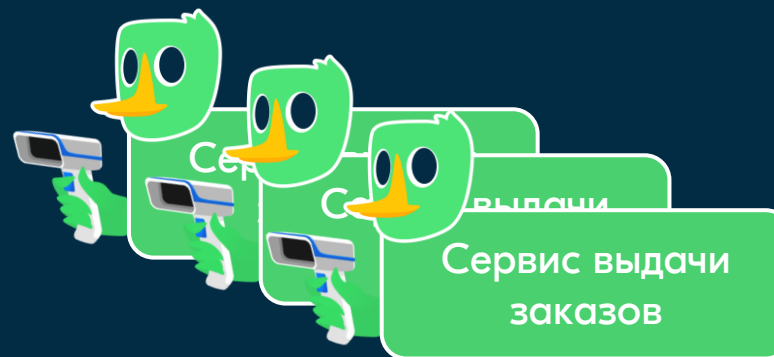
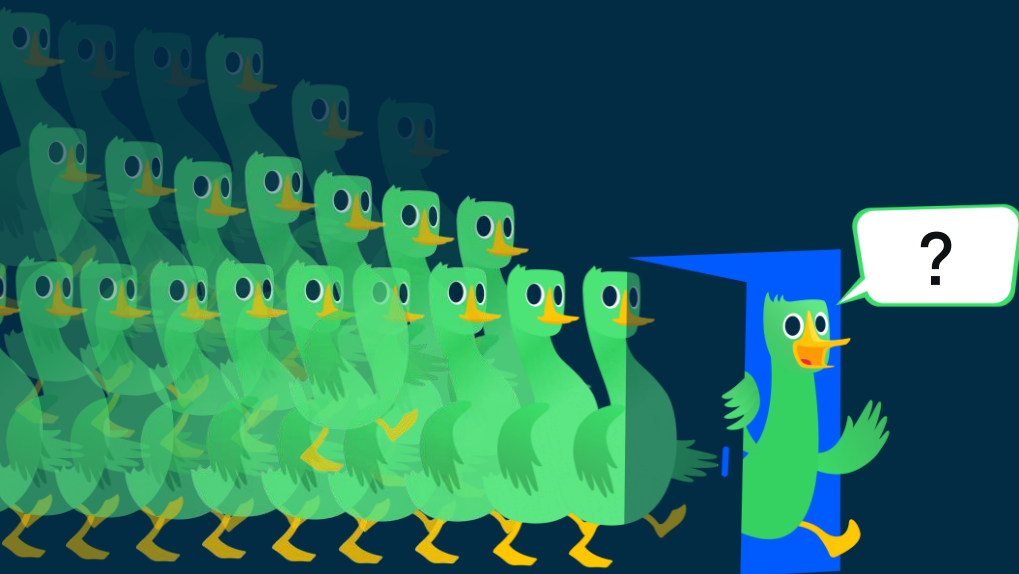
Масштабирование системы



Масштабирование системы



Масштабирование системы



горизонтальное масштабирование

VS

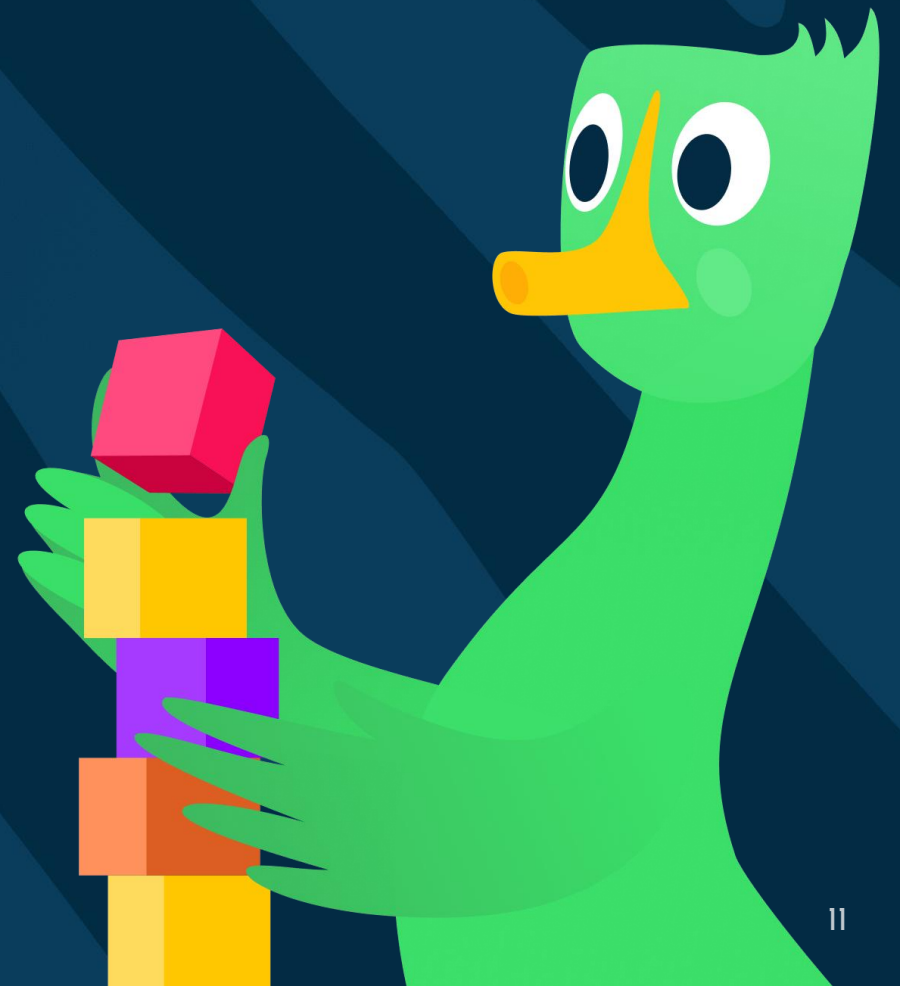
вертикальное масштабирование



Масштабирование системы



Шардирование — техника горизонтального масштабирования баз данных



Создаём шардированную базу данных

order-db @ server
public.order

order_id	client_id	target_store	price	currency	is_prepaid
1	9000	МОСКВА_43	499.5	RUB	● true
2	8000	МОСКВА_43	1500	RUB	false
3	7000	ЕРЕВАН_23	6500	AMD	● true
4	9000	ВЛАДИВОСТОК_2000	775.9	RUB	false

```
SELECT *  
FROM order  
WHERE order_id = 2;
```

```
SELECT *  
FROM order  
WHERE client_id = 9000;
```

```
SELECT price  
      , currency  
FROM order  
WHERE is_prepaid;
```

Создаём шардированную базу данных

order-db @ server
public.order

order_id	client_id	target_store	price	currency	is_prepaid
1	9000	МОСКВА_43	499.5	RUB	● true
2	8000	МОСКВА_43	1500	RUB	false
3	7000	ЕРЕВАН_23	6500	AMD	● true
4	9000	ВЛАДИВОСТОК_2000	775.9	RUB	false

```
SELECT *  
FROM order  
WHERE order_id = 2;
```

```
SELECT *  
FROM order  
WHERE client_id = 9000;
```

```
SELECT price  
      , currency  
FROM order  
WHERE is_prepaid;
```

Создаём шардированную базу данных

order-db @ server
public.order

order_id	client_id	target_store	price	currency	is_prepaid
1	9000	МОСКВА_43	499.5	RUB	● true
2	8000	МОСКВА_43	1500	RUB	false
3	7000	ЕРЕВАН_23	6500	AMD	● true
4	9000	ВЛАДИВОСТОК_2000	775.9	RUB	false

```
SELECT *  
FROM order  
WHERE order_id = 2;
```

```
SELECT *  
FROM order  
WHERE client_id = 9000;
```

```
SELECT price  
      , currency  
FROM order  
WHERE is_prepaid;
```

Создаём шардированную базу данных

order-db @ server
public.order

order_id	client_id	target_store	price	currency	is_prepaid
1	9000	МОСКВА_43	499.5	RUB	● true
2	8000	МОСКВА_43	1500	RUB	false
3	7000	ЕРЕВАН_23	6500	AMD	● true
4	9000	ВЛАДИВОСТОК_2000	775.9	RUB	false

```
SELECT *  
FROM order  
WHERE order_id = 2;
```

```
SELECT *  
FROM order  
WHERE client_id = 9000;
```

```
SELECT price  
      , currency  
FROM order  
WHERE is_prepaid;
```

Создаём шардированную базу данных

order-db-1 @ server-1
public.order

order_id	client_id	target_store	price	currency	is_prepaid
1	9000	МОСКВА_43	499.5	RUB	● true
2	8000	МОСКВА_43	1500	RUB	false

order-db-2 @ server-2
public.order

order_id	client_id	target_store	price	currency	is_prepaid
3	7000	ЕРЕВАН_23	6500	AMD	● true
4	9000	ВЛАДИВОСТОК_2000	775.9	RUB	false

```
SELECT *  
FROM order  
WHERE order_id = 2;
```

```
SELECT *  
FROM order  
WHERE client_id = 9000;
```

```
SELECT price  
      , currency  
FROM order  
WHERE is_prepaid;
```

Создаём шардированную базу данных

order-db-1 @ server-1
public.order

order_id	client_id	target_store	price	currency	is_prepaid
1	9000	МОСКВА_43	499.5	RUB	● true
2	8000	МОСКВА_43	1500	RUB	false

order-db-2 @ server-2
public.order

order_id	client_id	target_store	price	currency	is_prepaid
3	7000	ЕРЕВАН_23	6500	AMD	● true
4	9000	ВЛАДИВОСТОК_2000	775.9	RUB	false

```
SELECT *  
FROM order  
WHERE order_id = 2;
```

order-db-1

```
SELECT *  
FROM order  
WHERE client_id = 9000;
```

```
SELECT price  
      , currency  
FROM order  
WHERE is_prepaid;
```

Создаём шардированную базу данных

order-db-1 @ server-1
public.order

order_id	client_id	target_store	price	currency	is_prepaid
1	9000	МОСКВА_43	499.5	RUB	● true
2	8000	МОСКВА_43	1500	RUB	false

order-db-2 @ server-2
public.order

order_id	client_id	target_store	price	currency	is_prepaid
3	7000	ЕРЕВАН_23	6500	AMD	● true
4	9000	ВЛАДИВОСТОК_2000	775.9	RUB	false

```
SELECT *  
FROM order  
WHERE order_id = 2;
```

```
SELECT *  
FROM order  
WHERE client_id = 9000;
```

order-db-1
order-db-2

```
SELECT price  
      , currency  
FROM order  
WHERE is_prepaid;
```

Создаём шардированную базу данных

order-db-1 @ server-1
public.order

order_id	client_id	target_store	price	currency	is_prepaid
1	9000	МОСКВА_43	499.5	RUB	● true
2	8000	МОСКВА_43	1500	RUB	false

order-db-2 @ server-2
public.order

order_id	client_id	target_store	price	currency	is_prepaid
3	7000	ЕРЕВАН_23	6500	AMD	● true
4	9000	ВЛАДИВОСТОК_2000	775.9	RUB	false

```
SELECT *  
FROM order  
WHERE order_id = 2;
```

```
SELECT *  
FROM order  
WHERE client_id = 9000;
```

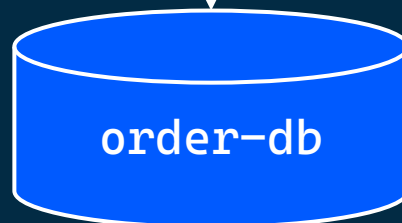
```
SELECT price  
      , currency  
FROM order  
WHERE is_prepaid;
```

order-db-1
order-db-2

Как теперь работать с данными?

order_id	client_id	target_store	price	currency	is_prepaid
100500	999999	ПЕТРОЗАВОДСК_13	600	RUB	● true

```
public async Task AddOrder(OrderModel order) {  
    const string sql = "INSERT INTO order ( ... ) ... ";  
    var parameters = new { OrderId = order.Id, ... };  
    var addOrderCommand = new CommandDefinition(sql, parameters);  
    await using var connection = await _database.GetConnectionAsync();  
    await connection.ExecuteAsync(addOrderCommand);  
}
```

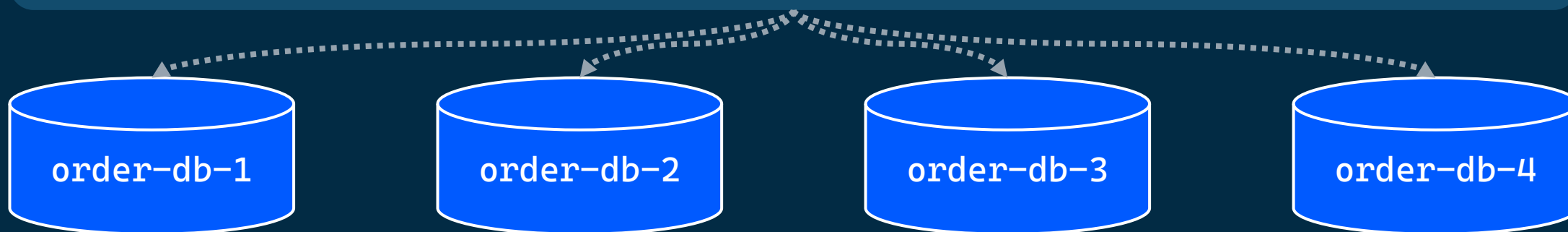


Как теперь работать с данными?

order_id	client_id	target_store	price	currency	is_prepaid
100500	999999	ПЕТРОЗАВОДСК_13	600	RUB	● true



```
public async Task AddOrder(OrderModel order) {  
    const string sql = "INSERT INTO order ( ... ) ... ";  
    var parameters = new { OrderId = order.Id, ... };  
    var addOrderCommand = new CommandDefinition(sql, parameters);  
    await using var connection = await _database.GetConnectionAsync();  
    await connection.ExecuteAsync(addOrderCommand);  
}
```

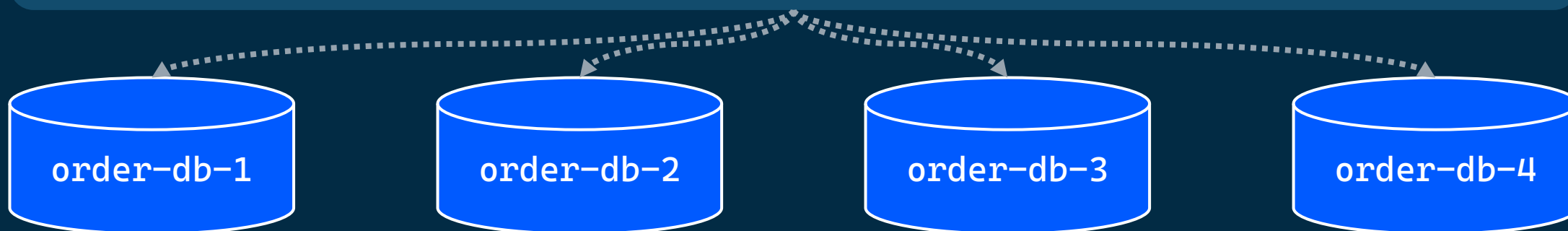


Как теперь работать с данными?

order_id	client_id	target_store	price	currency	is_prepaid
100500	999999	ПЕТРОЗАВОДСК_13	600	RUB	● true



```
public async Task AddOrders(OrderModel[] orders) {  
    const string sql = "INSERT INTO order ( ... ) ... ";  
    var parameters = new { OrderIds = orders.Select(ord => ord.Id), ... };  
    var addOrdersCommand = new CommandDefinition(sql, parameters);  
    await using var connection = await _database.GetConnectionAsync();  
    await connection.ExecuteAsync(addOrdersCommand);  
}
```



Маршрутизация запросов

T – это:

order_id	client_id	target_store	price	currency	is_prepaid
100500	999999	ПЕТРОЗАВОДСК_13	600	RUB	● true

```
interface IShardingRule<T> {  
    ShardId GetShardId(T obj, ShardCollection allShards);  
}
```

```
interface IShardedDataContext<T> {  
    IShardingRule<T> ShardingRule { get; }  
}
```

Маршрутизация запросов

```
interface IShardingRule<T> {  
    ShardId GetShardId(T obj, ShardCollection allShards);  
}
```

```
interface IShardedDataContext<T> {  
    IShardingRule<T> ShardingRule { get; }  
  
    Task<DbConnection> GetConnectionAsync(  
        T obj,  
        CancellationToken token);  
}
```

Маршрутизация запросов

```
interface IShardingRule<T> {  
    ShardId GetShardId(T obj, ShardCollection allShards);  
}  
  
interface IShardedDataContext<T> {  
    IShardingRule<T> ShardingRule { get; }  
  
    Task<DbConnection> GetConnectionAsync(  
        T obj,  
        CancellationToken token);  
  
    IShardedQueryScope<T> GetShardedQueryScope(  
        IEnumerable<T> objects);  
}
```

Маршрутизация запросов

```
internal sealed class ShardedQueryScope<T> {  
    private readonly IEnumerable<T> _objects;  
    private readonly IShardingRule<T> _shardingRule;  
    private readonly List<IQueryStep<T>> _steps;  
  
    public ShardedQueryScope<T> Append(IQueryStep<T> query) {  
        _steps.Add(query); return this;  
    }  
  
    public Task Execute(Cancellation_token token) { ... }  
}
```

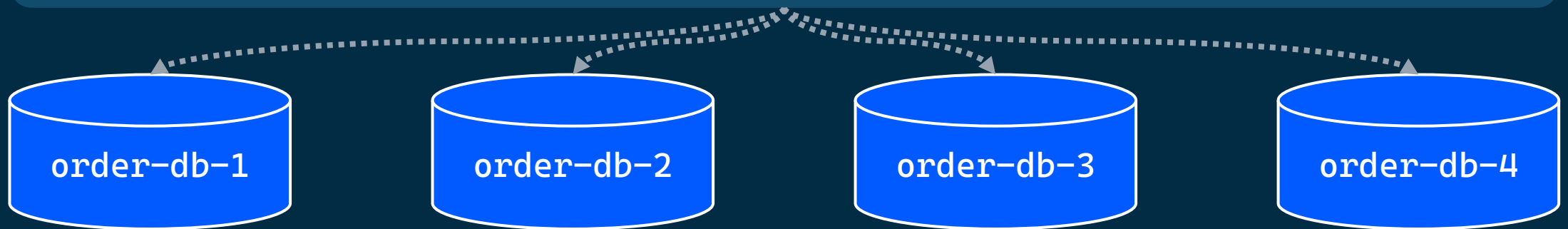
Маршрутизация запросов

```
internal sealed class ShardedQueryScope<T> {  
    private readonly IEnumerable<T> _objects;  
    private readonly IShardingRule<T> _shardingRule;  
    private readonly List<IQueryStep<T>> _steps;  
    private readonly SemaphoreSlim _semaphore;  
  
    public Task Execute(CancellationToken token) => Task.WhenAll(  
        _objects  
        .GroupBy(_shardingRule.GetShardId)  
        .Select(async shardObjects => {  
            await _semaphore.WaitAsync(token);  
            try {  
                foreach (var step in _steps) {  
                    await step.Perform(shardObjects, token);  
                }  
            }  
            finally { _semaphore.Release(); }  
        }));  
}
```

Маршрутизация запросов

order_id	client_id	target_store	price	currency	is_prepaid
100500	999999	ПЕТРОЗАВОДСК_13	600	RUB	● true

```
public async Task AddOrders(OrderModel[] orders, CancellationToken token) {  
    await _dataContext  
    /* 1 */ .GetShardedQueryScope(orders)  
    /* 2 */ .Append(new ExecutableQuery("INSERT INTO order ( ... ) ..."))  
    /* 3 */ .Execute(token);  
}
```



ПЛАН

База о шардировании базы

Создаём шардированную базу данных с нуля

Шардируем то, что уже работает

Ломаем шардированием то, что уже работает

Делаем выводы



Микросервис хранения атрибутов заказов и товаров

Контекст

Количество заказов на Ozon быстро растёт

Микросервис выдачи с ПВЗ использует монолитную БД MS SQL



Текущие проблемы

При отказе единственного и не зависящего от нашей команды источника данных **останавливаются все выдачи с ПВЗ**

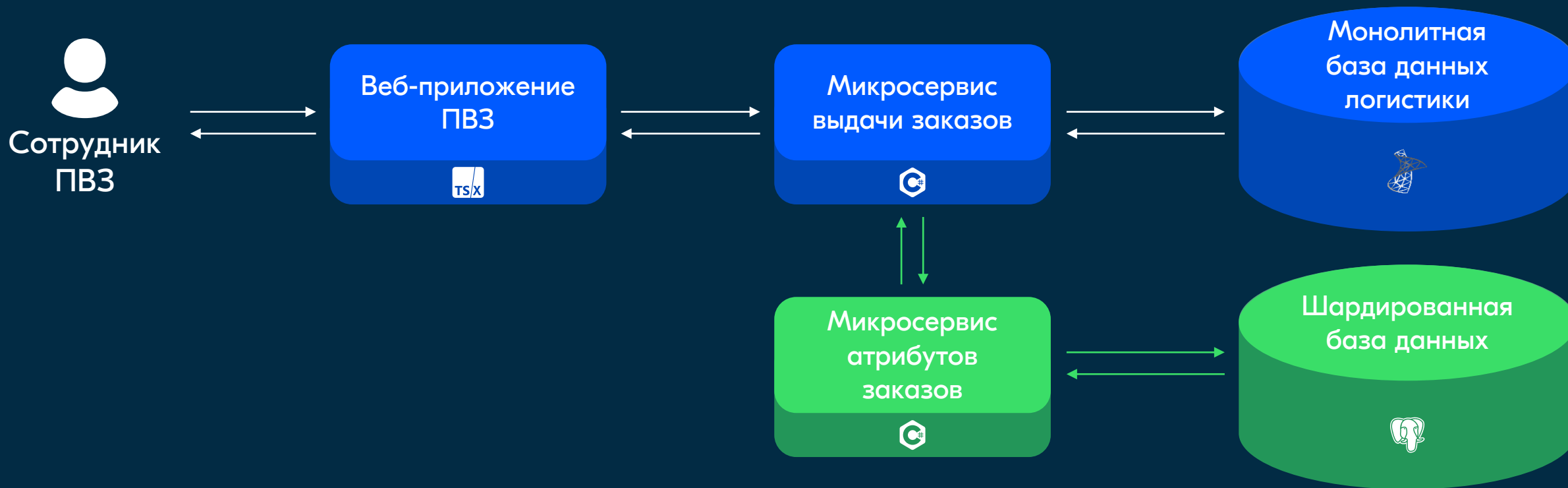


Ожидаемый результат

Новый **горизонтально масштабируемый** и **отказоустойчивый** источник данных для микросервиса выдачи



Микросервис хранения атрибутов заказов и товаров



Создание нового сервиса-источника данных

- Продумать процесс перевода потребителя с монолитной БД на шардированную

```
var oldInfo = await GetFromSingleDb( ... );  
var newInfo = await GetFromShardedDb( ... );  
  
CompareAndLogDifferences(oldInfo, newInfo);  
  
return _sourceDatabase is Source.SingleDb  
    ? oldInfo  
    : newInfo;
```



Создание нового сервиса-источника данных

- Продумать процесс перевода потребителя с монолитной БД на шардированную
- Определить ключ шардирования БД на основании требований и сценариев использования микросервиса

? Найди товары данного клиента

Микросервис выдачи заказов на ПВЗ

? Покажи подробную информацию о заказах клиента

? Выдай/аннулируй товары клиента

? Создай отправление с указанными атрибутами

Асинхронное чтение топиков Kafka

? Смени атрибуты/статус указанного отправления

⚠ В нашем случае в приоритете находится оптимизация читающей нагрузки



Создание нового сервиса-источника данных

- Продумать процесс перевода потребителя с монолитной БД на шардированную
- Определить ключ шардирования БД на основании требований и сценариев использования микросервиса

? Найди товары данного клиента **на указанном ПВЗ**

Микросервис выдачи заказов на ПВЗ

? Покажи подробную информацию о заказах клиента **на данном ПВЗ**

? Выдай/аннулируй товары клиента **на этом ПВЗ**

? Создай отправление с указанными атрибутами
(в том числе **целевой ПВЗ**)

Асинхронное чтение топиков Kafka

? Смени атрибуты/статус указанного отправления

⚠ В нашем случае в приоритете находится оптимизация **читающей нагрузки**



Создание нового сервиса-источника данных

- Продумать процесс перевода потребителя с монолитной БД на шардированную
- Определить ключ шардирования БД на основании требований и сценариев использования микросервиса
- Спроектировать архитектуру базы данных
- Написать код
- Запустить новый микросервис в Production



Создание нового сервиса-источника данных

- Продумать процесс перевода потребителя с монолитной БД на шардированную
- Определить ключ шардирования БД на основании требований и сценариев использования микросервиса
- Спроектировать архитектуру базы данных
- Написать код
- Запустить новый микросервис в Production
- Переключить выборки на новый источник данных



Результаты шардирования

~7 месяцев

занял весь процесс шардирования
(от анализа требований
до полного перехода на новую БД)

100 мс

99-я перцентиль времени
ответа нового сервиса



>400 RPS

снято с монолитной
логистической базы
данных MS SQL

При необходимости масштабирования в новую базу
можно быстро добавить новые шарды

Повышена отказоустойчивость и стабильность
микросервиса выдачи заказов

Применимость подхода



01

Кардинально пересматриваются
бизнесовые сущности и процессы



02

Новый источник данных будет
использоваться во многих системах



ПЛАН

База о шардировании базы

Создаём шардированную базу данных с нуля

Шардируем то, что уже работает

Ломаем шардированием то, что уже работает

Делаем выводы



Микросервис адресного хранения ПВЗ

Контекст

Сервис хранит информацию о местоположении товаров внутри ПВЗ

Сеть ПВЗ Ozon растёт, как и число товаров клиентов



Текущие проблемы

Сильно растёт объём базы данных и используемых ей ресурсов

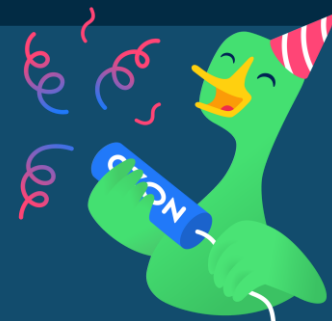
Запросы начинают выполняться медленнее



Ожидаемый результат

Ускорено исполнение запросов

Обеспечена отказоустойчивость и горизонтальная масштабируемость



Микросервис адресного хранения ПВЗ



Смена источника данных внутри микросервиса

- Определить ключ шардирования БД на основании требований и сценариев использования микросервиса
 - ❓ Найди ячейку, на которой лежит указанный предмет
 - ❓ Положи предмет на заданную ячейку
 - ❓ Удали предмет с указанной ячейки



Смена источника данных внутри микросервиса

- Определить ключ шардирования БД на основании требований и сценариев использования микросервиса
 - ❓ Найди ячейку ПВЗ, на которой лежит указанный предмет
 - ❓ Положи предмет на заданную ячейку в указанном ПВЗ
 - ❓ Удали предмет с указанной ячейки ПВЗ



Смена источника данных внутри микросервиса

- Определить ключ шардирования БД на основании требований и сценариев использования микросервиса
 - ❓ Найди ячейку ПВЗ, на которой лежит указанный предмет
 - ❓ Положи предмет на заданную ячейку в указанном ПВЗ
 - ❓ Удали предмет с указанной ячейки ПВЗ
- Продумать процесс перевода микросервиса на шардированную БД
 - ⚠ В сервисе постоянно есть и **читающая**, и **пишущая нагрузка**
 - ⚠ И пишущая, и читающая нагрузки **синхронные**
 - ⚠ Данные о местоположении предметов должны быть **консистентны**



Смена источника данных внутри микросервиса

- Продумать процесс перевода микросервиса на шардированную БД
 - ⚠ В сервисе постоянно есть и читающая, и пишущая нагрузка
 - ⚠ Пишущая нагрузка синхронная
 - ⚠ Данные о местоположении предметов должны быть консистентны

Как записывать данные в обе базы,
не теряя консистентности?



Смена источника данных внутри микросервиса

- Продумать процесс перевода микросервиса на шардированную БД
 - ⚠ В сервисе постоянно есть и читающая, и пишущая нагрузка
 - ⚠ Пишущая нагрузка синхронная
 - ⚠ Данные о местоположении предметов должны быть консистентны

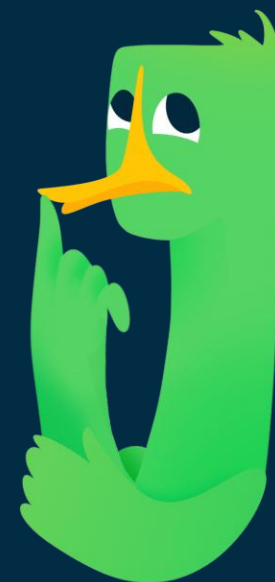
Как понять, что между данными есть расхождения,
и как их исправлять?



Смена источника данных внутри микросервиса

- Продумать процесс перевода микросервиса на шардированную БД
 - ⚠ В сервисе постоянно есть и читающая, и пишущая нагрузка
 - ⚠ Пишущая нагрузка синхронная
 - ⚠ Данные о местоположении предметов должны быть консистентны

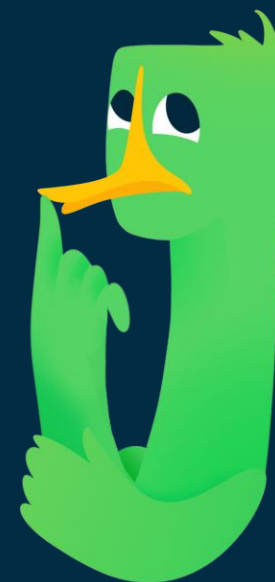
ПВЗ не работают ночью...



Смена источника данных внутри микросервиса

- Продумать процесс перевода микросервиса на шардированную БД
 - ⚠ В сервисе постоянно есть и читающая, и пишущая нагрузка
 - ⚠ Пишущая нагрузка синхронная
 - ⚠ Данные о местоположении предметов должны быть консистентны

ПВЗ не работают ночью...
Мы знаем часовой пояс каждого ПВЗ...



Смена источника данных внутри микросервиса

- Продумать процесс перевода микросервиса на шардированную БД
 - ⚠ В сервисе постоянно есть и читающая, и пишущая нагрузка
 - ⚠ Пишущая нагрузка синхронная
 - ⚠ Данные о местоположении предметов должны быть консистентны

ПВЗ не работают ночью...
Мы знаем часовой пояс каждого ПВЗ...
Миграция данных по одному ПВЗ будет быстрая...



Смена источника данных внутри микросервиса

- Продумать процесс перевода микросервиса на шардированную БД
 - ⚠ В сервисе постоянно есть и читающая, и пишущая нагрузка
 - ⚠ Пишущая нагрузка синхронная
 - ⚠ Данные о местоположении предметов должны быть консистентны

При переносе запрещаем редактировать,
копируем данные, после чего всегда
используем шардированную БД!



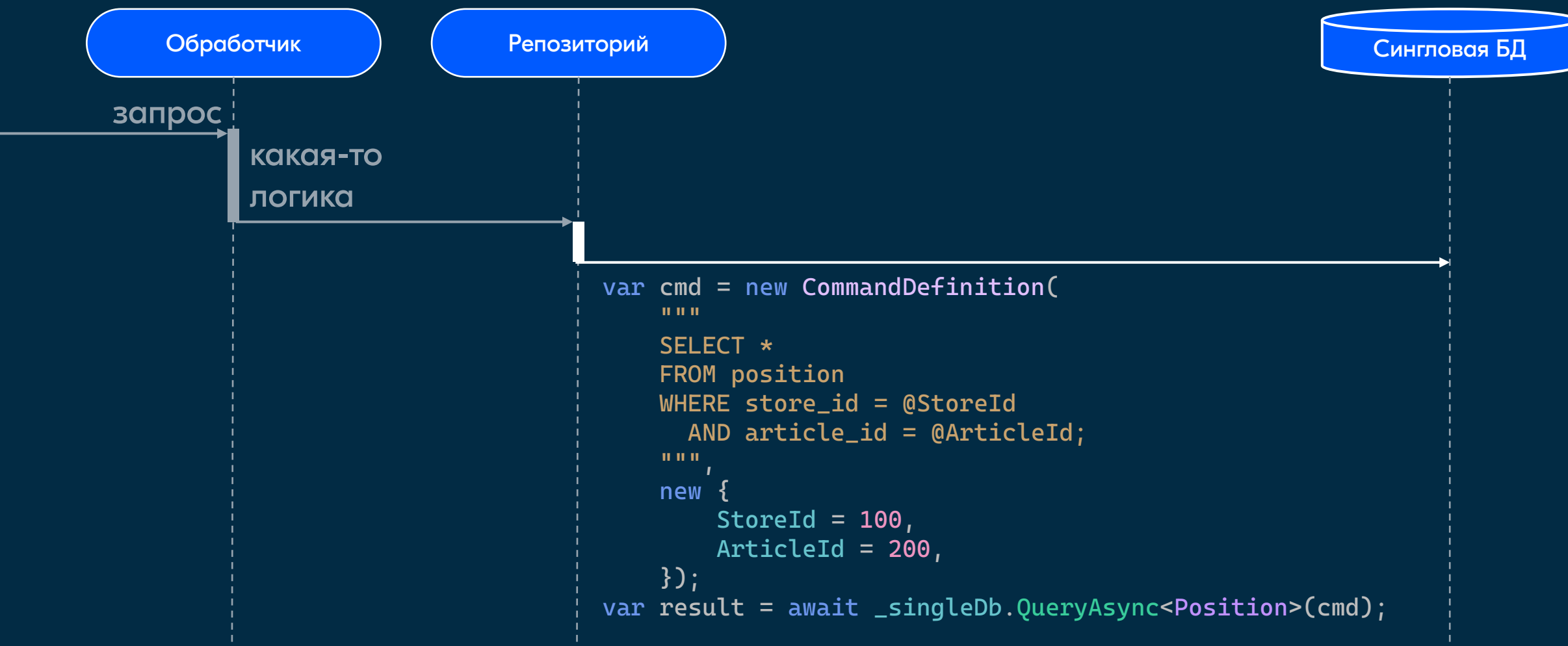
Смена источника данных внутри микросервиса



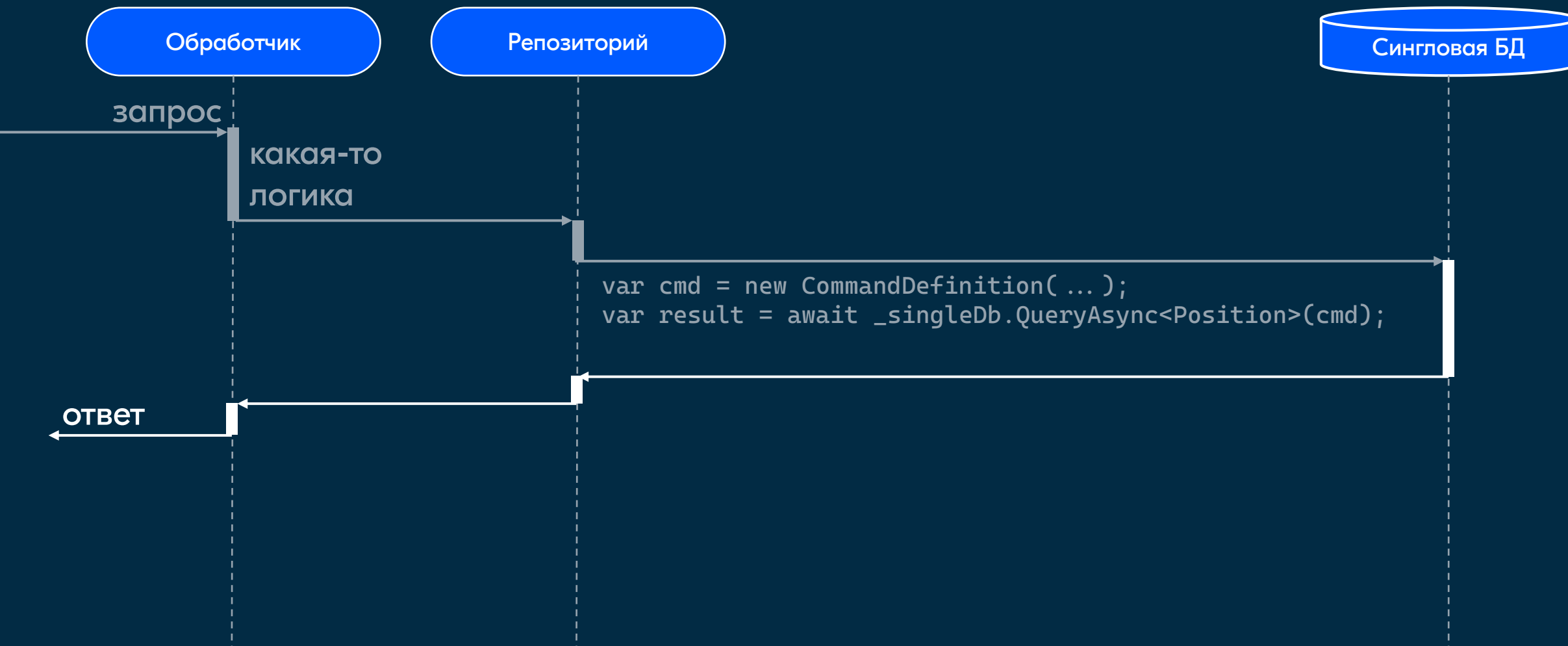
Смена источника данных внутри микросервиса



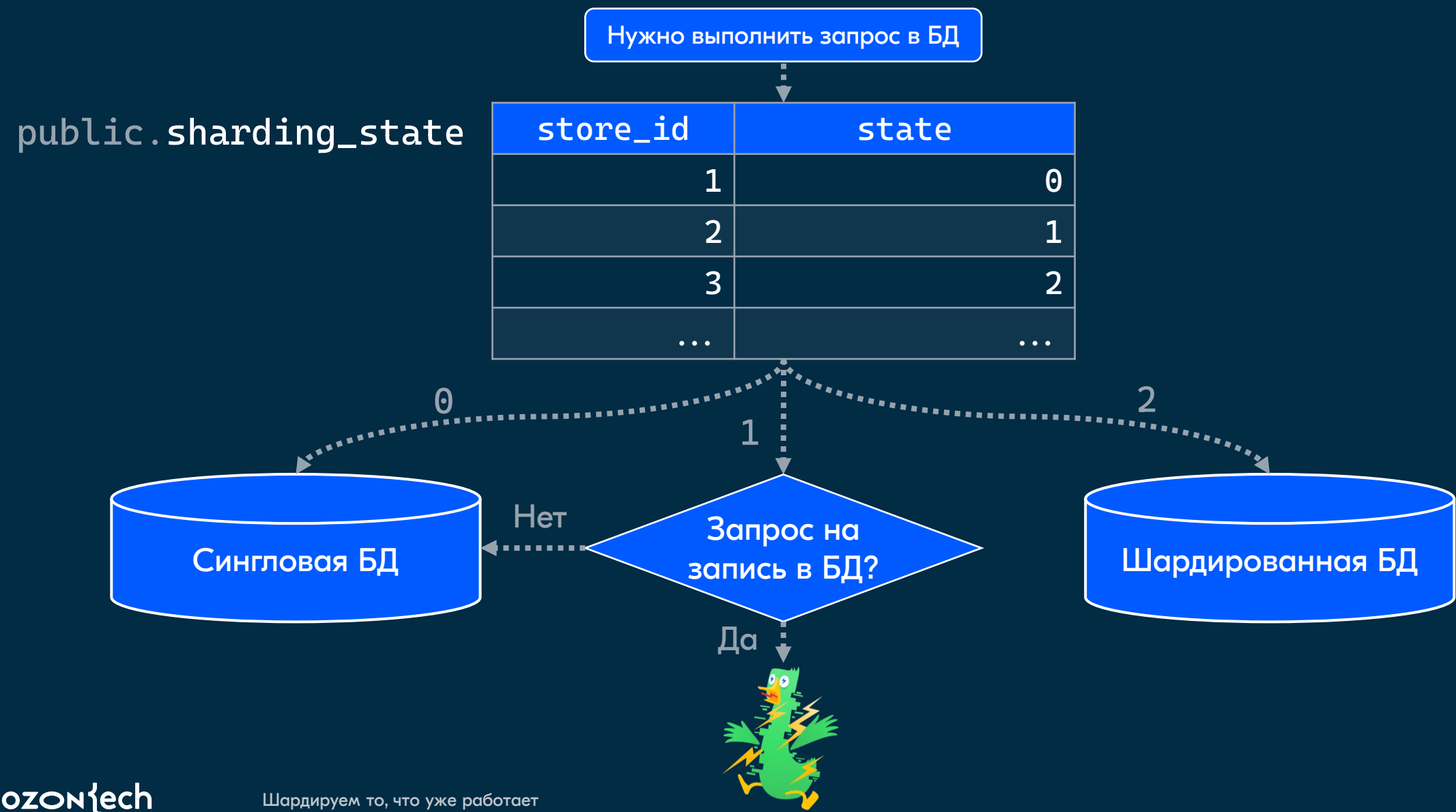
Смена источника данных внутри микросервиса



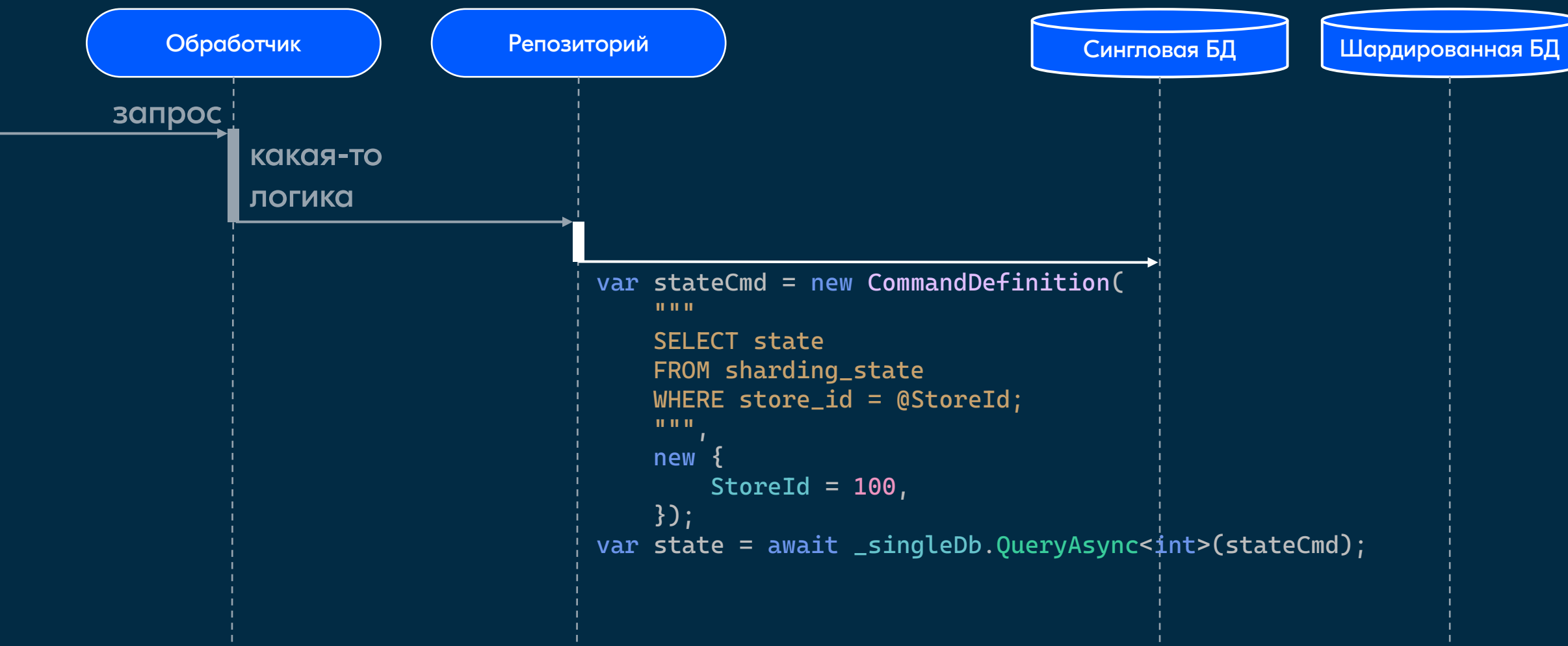
Смена источника данных внутри микросервиса



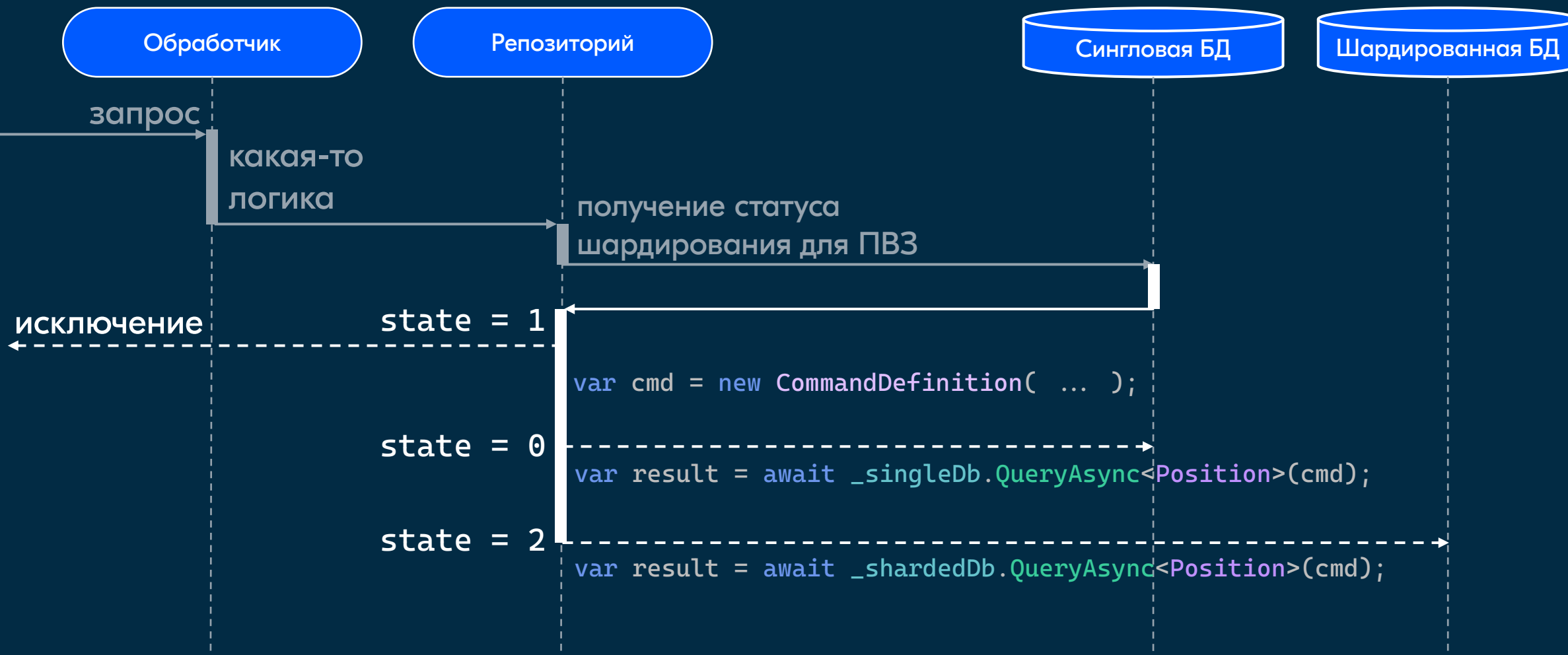
Смена источника данных внутри микросервиса



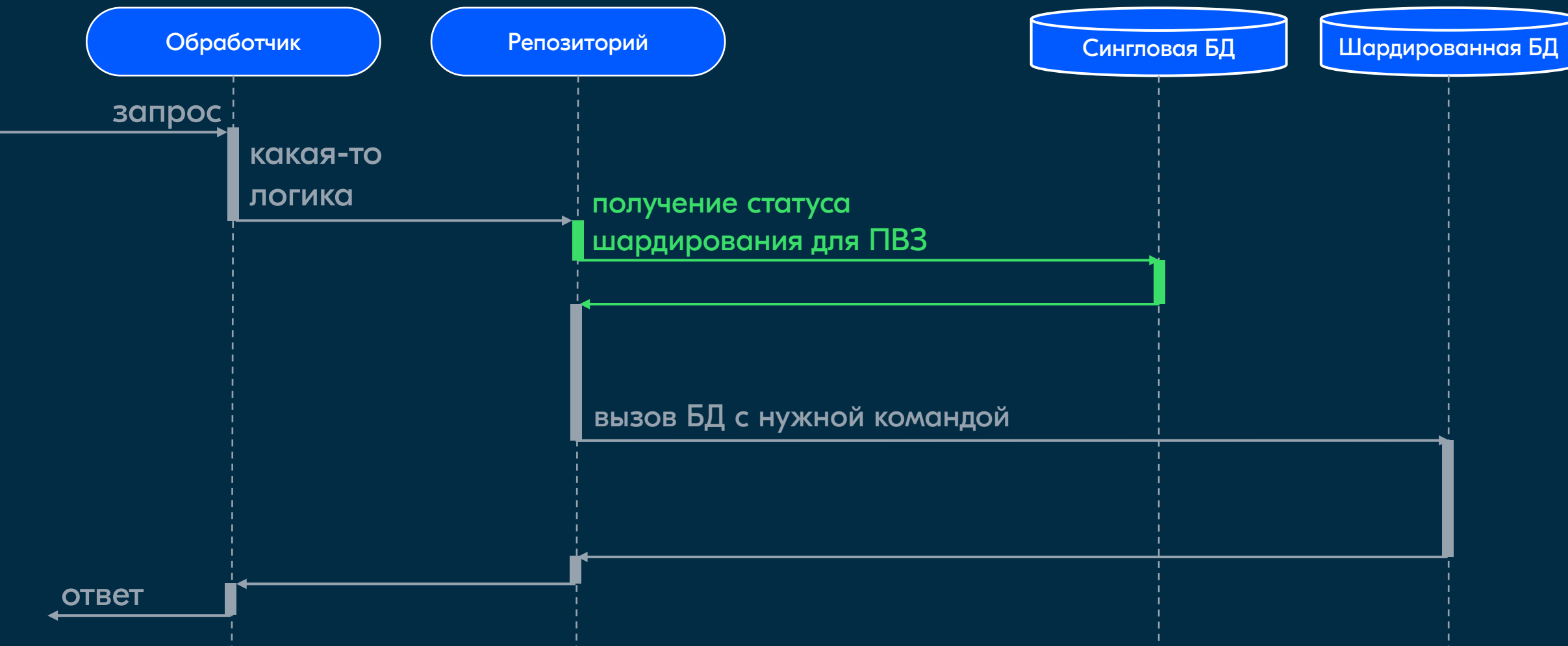
Смена источника данных внутри микросервиса



Смена источника данных внутри микросервиса



Смена источника данных внутри микросервиса



Поэтапная раскатка шардированной БД на все ПВЗ

Пн	Вт	Ср	Чт	Пт	Сб	Вс
1	2	3 Проверили на одном ПВЗ	4 Раскатка на следующие 5 ПВЗ	5	6	7
8 Каждый день раскатывали функционал ещё на 5—10 тысяч ПВЗ	9	10	11	12	13	14
15	16	17 	18 Перенос завершён на всех ПВЗ	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Результаты шардирования

48 мс

99-я перцентиль времени
ответа сервиса



Используемый объем диска

198 → **60** GB на 1 реплику

792 → **728** GB на кластер

6 400 RPS

выдерживает микросервис
в ходе нагрузочных тестов



Используемая оперативная память

15 → **4** GB на 1 реплику

60 → **48** GB на кластер

Результаты шардирования

List of all user indexes (every 30min) ⓘ

IndexName 🚫	Index si: 🚫	Vali 🚫	TableName 🚫	Table size 🚫
public.PK_position	5.47 GiB	On	public.position	1.76 GiB
public.IX_position_article_id_store_id	4.31 GiB	On	public.position	1.76 GiB
public.IX__empty_element_sort_optimized	1.79 GiB	On	public.element	1.96 GiB
public.idx__position__store_id_element_id	1.70 GiB	On	public.position	1.76 GiB
public.idx__position__store_id_element_id_number	1.69 GiB	On	public.position	1.76 GiB
public.idx__position__store_id_group_key	1.63 GiB	On	public.position	1.76 GiB
public.IX_position_element_id_number	1.29 GiB	On	public.position	1.76 GiB
public.IX__empty_element_sort	1.24 GiB	On	public.element	1.96 GiB
public.IX_element_relation_parent_element_id	1.01 GiB	On	public.element_relation	1.73 GiB
public.PK_element_relation	805 MiB	On	public.element_relation	1.73 GiB
public.PK_element	793 MiB	On	public.element	1.96 GiB
public.IX_element_relation_child_element_id	782 MiB	On	public.element_relation	1.73 GiB
public.IX_position_element_id_article_id_not_null	545 MiB	On	public.position	1.76 GiB
public.IX_position_element_id	545 MiB	On	public.position	1.76 GiB
public.IX_element_store_id_element_type_id_is_active_attribute	381 MiB	On	public.element	1.96 GiB
public.IX_element_store_id	343 MiB	On	public.element	1.96 GiB

Результаты шардирования

List of all user indexes (every 30min) ⓘ

IndexName ⚙	Index si: ⚙	Vali ⚙	TableName ⚙	Table size ⚙
bucket_0.IX_position_article_id_store_id	21.7 MiB	On	bucket_0.position	28.6 MiB
bucket_0.position_pkey	20.3 MiB	On	bucket_0.position	28.6 MiB
bucket_0.idx__position__store_id_element_id	14.8 MiB	On	bucket_0.position	28.6 MiB
bucket_0.idx__position__store_id_group_key	14.7 MiB	On	bucket_0.position	28.6 MiB
bucket_0.IX__empty_element_sort_optimized	12 MiB	On	bucket_0.element	9.67 MiB
bucket_0.idx__position__store_id_element_id_number	10.8 MiB	On	bucket_0.position	28.6 MiB
bucket_0.IX_position_element_id_number	9.71 MiB	On	bucket_0.position	28.6 MiB
bucket_0.IX__empty_element_sort	6.29 MiB	On	bucket_0.element	9.67 MiB
bucket_0.element_pkey	5.25 MiB	On	bucket_0.element	9.67 MiB
bucket_0.element_relation_pkey	5.19 MiB	On	bucket_0.element_relation	9.44 MiB
bucket_0.IX_element_relation_child_element_id	5.19 MiB	On	bucket_0.element_relation	9.44 MiB
bucket_0.IX_position_element_id	4.16 MiB	On	bucket_0.position	28.6 MiB
bucket_0.IX_position_element_id_article_id_not_null	4.16 MiB	On	bucket_0.position	28.6 MiB
bucket_0.IX_element_store_id_element_type_is_active_attribute	1.88 MiB	On	bucket_0.element	9.67 MiB
bucket_0.IX_element_store_id	1.65 MiB	On	bucket_0.element	9.67 MiB
bucket_0.IX_element_relation_parent_element_id	1.61 MiB	On	bucket_0.element_relation	9.44 MiB
bucket_0.IX_element_element_type	1.14 MiB	On	bucket_0.element	9.67 MiB

Результаты шардирования

buffers_alloc: было



buffers_alloc: стало



Применимость подхода



01

Можно быстро перенести данные, если система может позволить себе временную недоступность на запись



02

Миграцию данных между БД можно проводить постепенно, разделив данные на небольшие части



ПЛАН

База о шардировании базы

Создаём шардированную базу данных с нуля

Шардируем то, что уже работает

Ломаем шардированием то, что уже работает

Делаем выводы



Микросервис антифрода при выдаче заказов

Контекст

Надёжное хранилище данных для запрета конфликтующих действий

Данные хранятся максимум два дня, в большинстве случаев — несколько часов



Текущие проблемы

При недоступности БД микросервиса **останавливаются все выдачи на ПВЗ**



Ожидаемый результат

Путём разделения одного источника данных на несколько мы позволим проводить хотя бы часть выдач, не останавливая работу всей сети ПВЗ целиком



Микросервис антифрода при выдаче заказов

→ ~8 000 RPS

Микросервис
выдачи заказов



```
SetGiveoutStartedRequest {  
  repeated int64 posting_ids = 1;  
}
```

Микросервис
отмены заказов



```
GetPostingStatusRequest {  
  repeated int64 posting_ids = 1;  
}
```

Микросервис
поддержки ПВЗ



```
RemovePostingStatusRequest {  
  repeated int64 posting_ids = 1;  
}
```



Микросервис антифрода

giveout_started

posting_id	created_at
1	2026-06-13 21:10:00+00:00
2	2026-06-11 16:08:53+00:00
5	2026-06-12 18:56:23+00:00

return_started

posting_id	created_at
4	2026-06-08 08:01:07+00:00
8	2026-06-09 13:58:12+00:00

(Очередная) смена источника данных внутри микросервиса

- Продумать процесс перевода потребителя с монолитной БД на шардированную
- Определить ключ шардирования
- Спроектировать архитектуру базы данных
- Написать код
- Запустить шардированную базу в Production
- Переключить выборки на новый источник данных



(Очередная) смена источника данных внутри микросервиса

- Продумать процесс перевода потребителя с монолитной БД на шардированную
- Определить ключ шардирования
- ✓ Спроектировать архитектуру базы данных
- Написать код
- Запустить шардированную базу в Production
- Переключить выборки на новый источник данных



(Очередная) смена источника данных внутри микросервиса

- Продумать процесс перевода потребителя с монолитной БД на шардированную
- ❗ Определить ключ шардирования
- ✅ Спроектировать архитектуру базы данных
- Написать код
- Запустить шардированную базу в Production
- Переключить выборки на новый источник данных



(Очередная) смена источника данных внутри микросервиса

- Продумать процесс перевода потребителя с монолитной БД на шардированную
- ✓ Определить ключ шардирования
- ✓ Спроектировать архитектуру базы данных
- Написать код
- Запустить шардированную базу в Production
- Переключить выборки на новый источник данных



(Очередная) смена источника данных внутри микросервиса

- ❓ Продумать процесс перевода потребителя с монолитной БД на шардированную
- ✓ Определить ключ шардирования
- ✓ Спроектировать архитектуру базы данных
- Написать код
- Запустить шардированную базу в Production
- Переключить выборки на новый источник данных



(Очередная) смена источника данных внутри микросервиса

❓ Продумать процесс перевода потребителя с монолитной БД на шардированную

⚠ В сервисе постоянно есть и **читающая**, и **пишущая нагрузка**

⚠ И пишущая, и читающая нагрузки **синхронные**

⚠ **Не получится переносить данные частями** — нет осмысленного разделения по ПВЗ

```
var singleTask = ExecuteSingleQuery(...);  
var shardedTask = ExecuteShardedQuery(...);  
await Task.WhenAll(singleTask, shardedTask);  
  
return await singleTask;
```



(Очередная) смена источника данных внутри микросервиса

- ✓ Продумать процесс перевода потребителя с монолитной БД на шардированную
- ✓ Определить ключ шардирования
- ✓ Спроектировать архитектуру базы данных
 - Написать код
 - Запустить шардированную базу в Production
 - Переключить выборки на новый источник данных



(Очередная) смена источника данных внутри микросервиса

- ✓ Продумать процесс перевода потребителя с монолитной БД на шардированную
- ✓ Определить ключ шардирования
- ✓ Спроектировать архитектуру базы данных
- ✓ Написать код
- ✓ Запустить шардированную базу в Production
- Переключить выборки на новый источник данных



(Очередная) смена источника данных внутри микросервиса

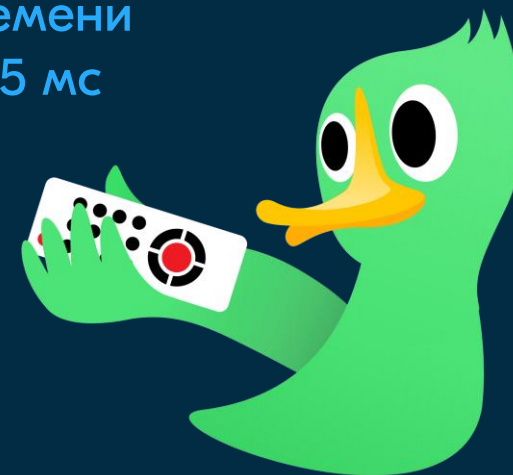
? Переключить выборки на новый источник данных



~7 500 RPS отрабатывают
менее чем за 100 мс

0.9-я квантиль времени
ответа сервиса ~15 мс

100% запросов
получают
код ответа «OK»



(Очередная) смена источника данных внутри микросервиса

✖ Переключить выборки на новый источник данных



Более половины запросов отвечают дольше 2 сек

0.9-я квантиль времени ответа превысила 5 сек



Смотрим на графики

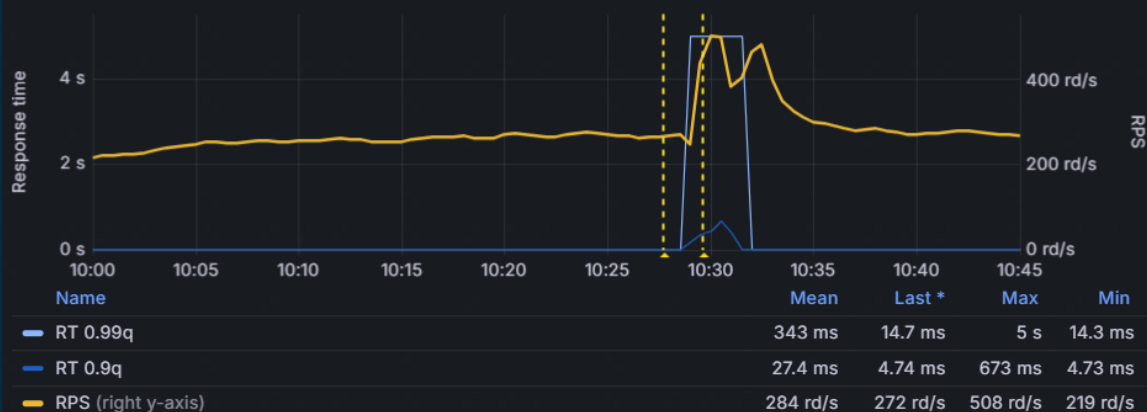
pvz-api-pms-watchdog: response time (0.9, 0.99q) + RPS



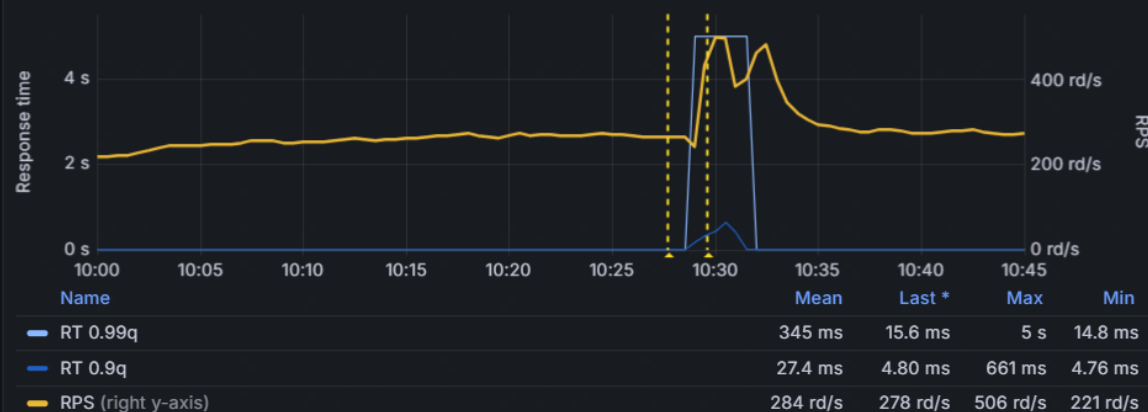
pvz-api-pms-watchdog-db-1: response time (0.9, 0.99q) + RPS



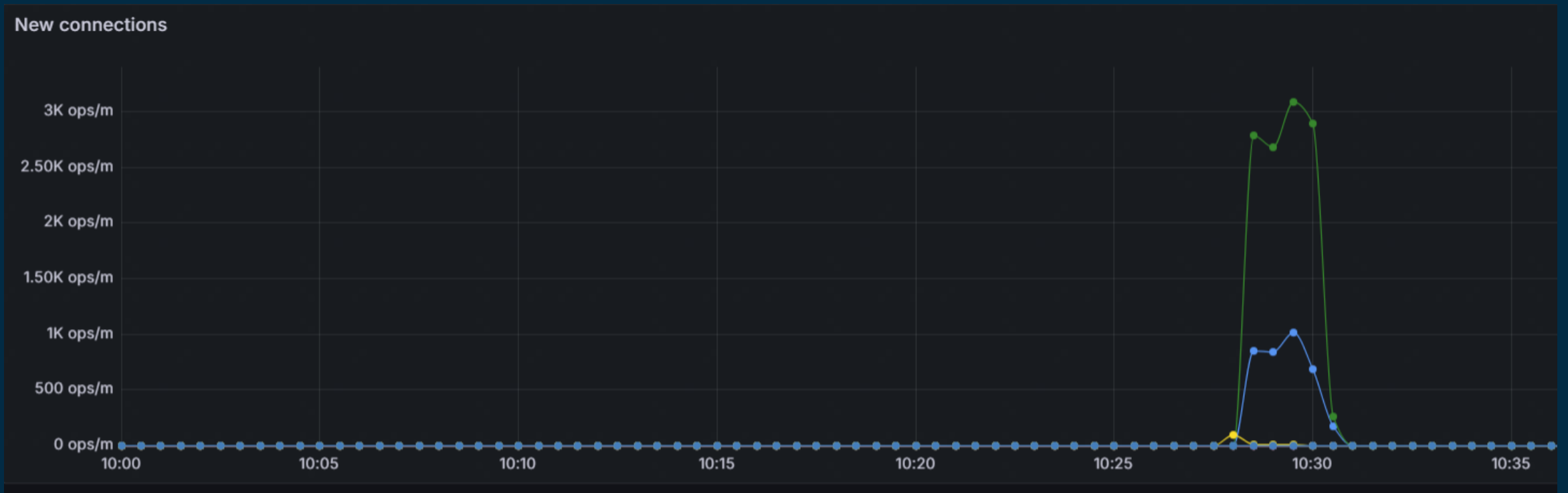
pvz-api-pms-watchdog-db-2: response time (0.9, 0.99q) + RPS



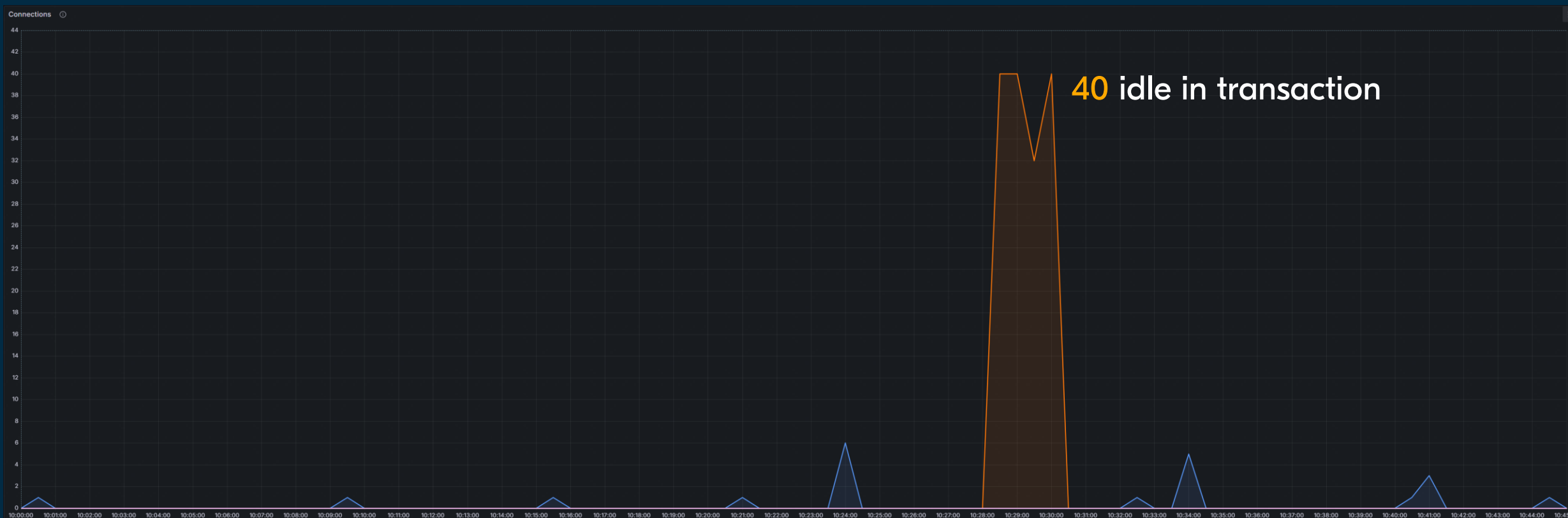
pvz-api-pms-watchdog-db-3: response time (0.9, 0.99q) + RPS



Смотрим на графики



Смотрим на графики



В чём проблема?



→ ~8 000 RPS

 Микросервис антифрода

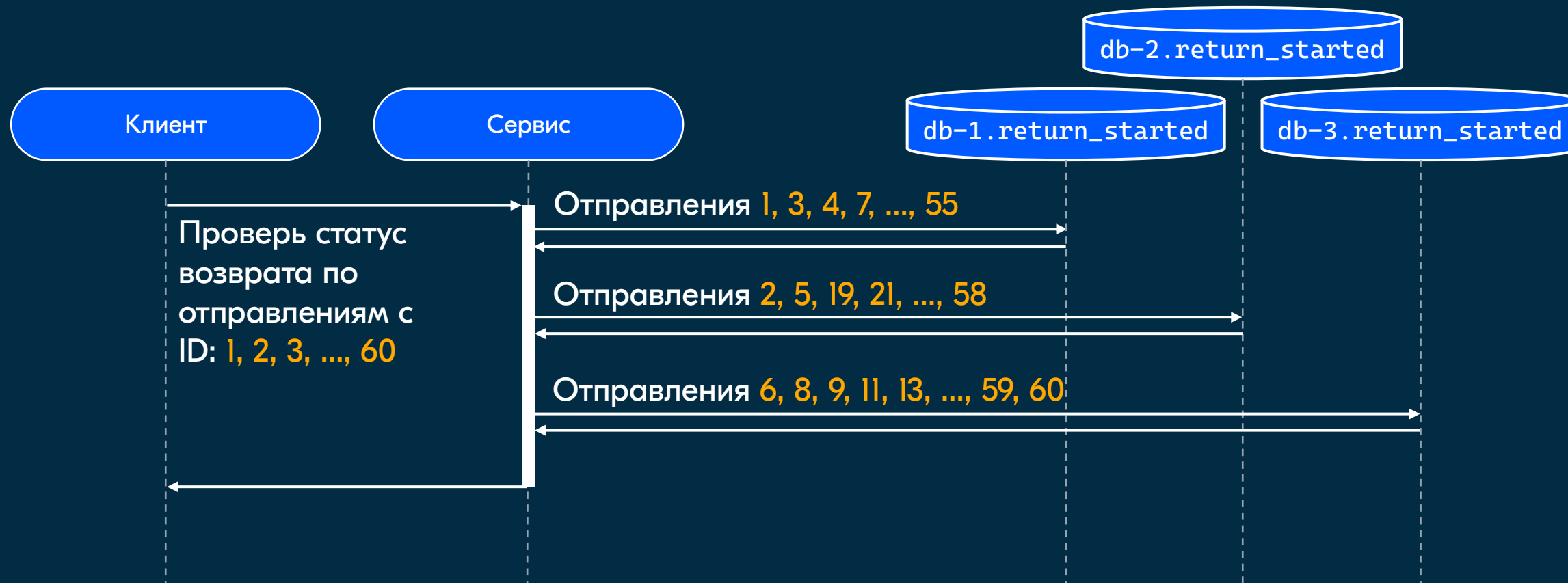
giveout_started

posting_id	created_at
1	2026-06-13 21:10:00+00:00
2	2026-06-11 16:08:53+00:00
5	2026-06-12 18:56:23+00:00

return_started

posting_id	created_at
4	2026-06-08 08:01:07+00:00
8	2026-06-09 13:58:12+00:00

В чём проблема?





Что пошло
не так?

01

База данных сервиса не являлась
долгосрочным хранилищем данных



02

В запросах нельзя было выделить
осмысленного ключа шардирования



03

Распараллеливание запросов
не всегда приводит к увеличению
производительности



Когда в руках шардирование,
все проблемы кажутся
сингловыми базами



ПЛАН

База о шардировании базы

Создаём шардированную базу данных с нуля

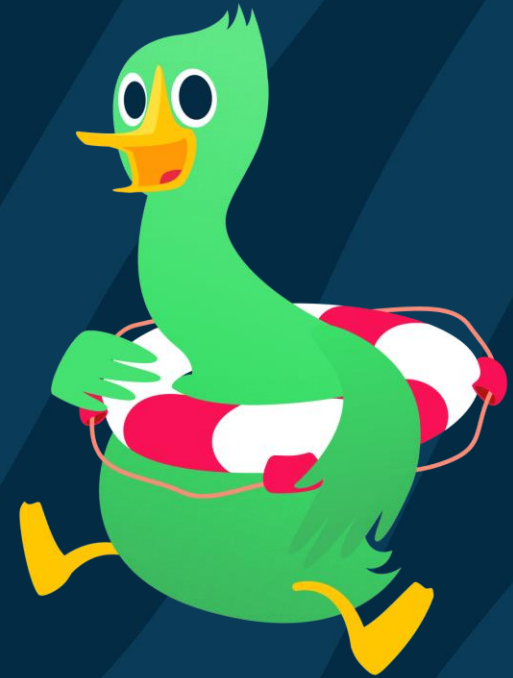
Шардируем то, что уже работает

Ломаем шардированием то, что уже работает

Делаем выводы



Шардирование применяется
для горизонтального
масштабирования баз данных,
что повышает производительность
и отказоустойчивость системы



Шардирование зачастую **не требуется**
применять для баз, хранящих малый
объём данных



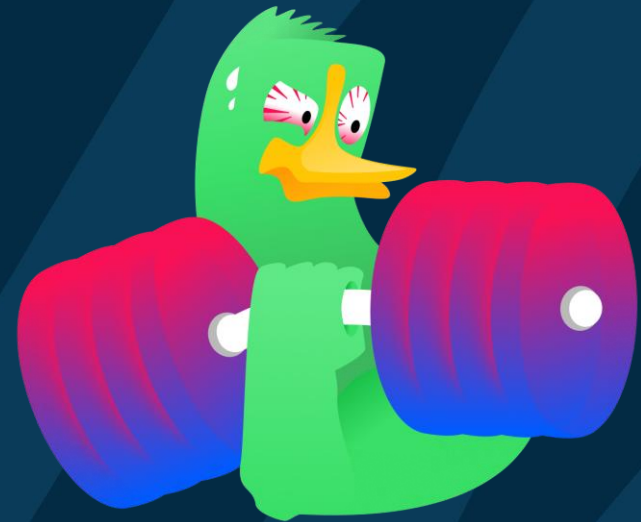
При проектировании архитектуры
важно правильно выбирать
ключ шардирования



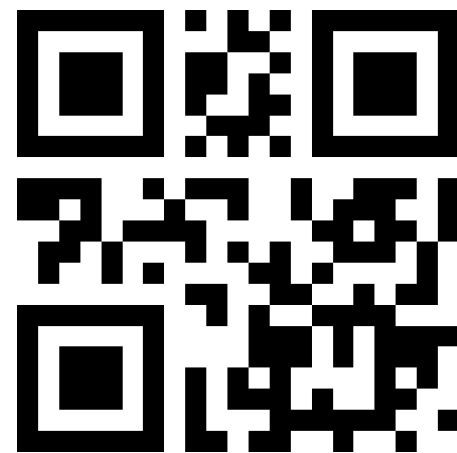
Нужно **тщательно продумывать**
способ перехода со старого
источника данных на новый



Перед внедрением базы данных
в рабочую среду обязательно
провести нагрузочное
тестирование



Вопросы



t.me/dedov_ivan

ozon{tech



Иван Дедов

indedov@gmail.com

idedov@ozon.ru