Outbox

Борис Кузоваткин

2024

Что такое Outbox

Outbox –

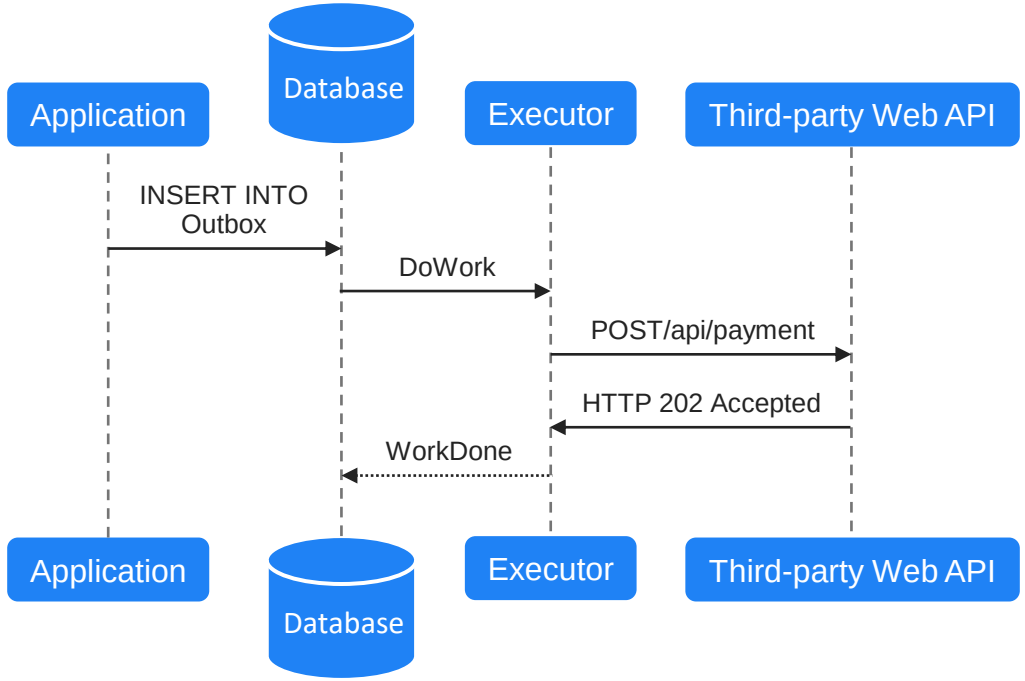
архитектурный паттерн
для надёжного обмена сообщениями
между микросервисами и другими
компонентами системы

- Простой и очевидный паттерн
- Основные элементы:

Источник: здесь происходит событие

Продьюсер: читает базу и шлёт консьюмеру

Консьюмер: система, потребляющая сообщения



Разберём **алгоритм**

Сервис записывает событие в таблицу Outbox в той же транзакции, что и изменение состояния

01

Разберём **алгоритм**

Сервис записывает событие в таблицу Outbox в той же транзакции, что и изменение состояния

01

Отдельный процесс или сервис читает события из таблицы Outbox и отправляет их в целевую систему (например, в очередь сообщений)

02

Разберём **алгоритм**

Сервис записывает событие в таблицу Outbox в той же транзакции, что и изменение состояния

01

Отдельный процесс или сервис читает события из таблицы Outbox и отправляет их в целевую систему (например, в очередь сообщений)

02

После успешной отправки события из таблицы Outbox удаляются или маркируются как обработанные

03

Разберём **алгоритм**

Сервис записывает событие в таблицу Outbox в той же транзакции, что и изменение состояния

01

Отдельный процесс или сервис читает события из таблицы Outbox и отправляет их в целевую систему (например, в очередь сообщений)

02

После успешной отправки события из таблицы Outbox удаляются или маркируются как обработанные

03

Успешный успех!

04

Спинофф: **гарантии доставки**

- **At-most-once**
  сообщения доставляются 1 раз... **или нет**
- **At-least-once**
  сообщения доставляются **1 и более раз**
- **Exactly-once**
  сообщения доставляются **строго 1 раз**
- **Гарантированный порядок сообщений**

01

Давайте напишем код!

Код

1 usage

```
private async Task SendMessages(CancellationToken cancellationToken)
{
    while (!cancellationToken.IsCancellationRequested)
    {
        var lastProcessedId = await GetLastProcessedIdAsync();

        var messages = await GetMessagesAsync(lastProcessedId);

        foreach (var message in messages)
        {
            await SendToKafkaAsync(message);
            await UpdateLastProcessedIdAsync(message.Id);
        }

        await Task.Delay(TimeSpan.FromSeconds(5), cancellationToken);
    }
}
```

1 usage

```
private async Task SendToKafkaAsync(MyMessage message)
{
    var kafkaMessage = new Message<Null, string> { Value = message.Data };
    var deliveryResult = await _producer.ProduceAsync(kafkaTopic, kafkaMessage);

    Console.WriteLine($"Сообщение для записи ID {message.Id} отправлено в Kafka");
}
```

Увы, это не работает

- Не все сообщения доставляются
- Часть сообщений дублируется
- Порядок может нарушаться

Почему не работают аутбоксы

- База – **не брокер очередей**
- В базе **нет гарантий порядка «сообщений»**
- Гарантии порядка **гарантируют блокировки**
- Брокеры очередей **быстрее rdbms** (потому что проще)
- Несколько продюсеров **усиливают блокировки**
- Несколько консюмеров **усиливают проблему порядка и гарантий доставки**

01. Код не работает

- **Порядок транзакций не совпадает с сиквенсом**
  - Потеря сообщений с долгими транзакциями
  - Нарушения порядка
- **Отправка сообщений по одному**
- **Отправка в 1 поток**

02

Давайте перепишем код!

Код

1 usage

```
private async Task SendMessages(CancellationToken cancellationToken)
{
    while (!cancellationToken.IsCancellationRequested)
    {
        var messages = await GetPendingMessagesAsync(batchSize);

        if (messages.Any())
        {
            await SendBatchToKafkaAsync(messages);
            await MarkMessagesAsSentAsync(messages.Select(m => m.Id));
        }

        await Task.Delay(TimeSpan.FromSeconds(5), cancellationToken);
    }
}
```

1 usage

```
private async Task SendBatchToKafkaAsync(ICollection<MyMessage> messages)
{
    var tasks = messages.Select(message =>
    {
        var kafkaMessage = new Message<Null, string> { Value = message.Data };
        return _producer.ProduceAsync(kafkaTopic, kafkaMessage);
    });

    await Task.WhenAll(tasks);
    Console.WriteLine($"{messages.Count} сообщений отправлено в Kafka.");
}
```

02. Код не работает

- Ура! Починили пропажу
- Не починили порядок
- Не починили дубли
- Получили зависшие сообщения
- Получили пухнущий индекс
- Теперь каждое сообщение нужно апдейтить

03

Давайте перепишем код!

Код

```
1 usage
private async Task SendMessages(CancellationToken cancellationToken)
{
    while (!cancellationToken.IsCancellationRequested)
    {
        await using var connection = new SqlConnection(connectionString);
        await connection.OpenAsync(cancellationToken);

        await using var transaction = connection.BeginTransaction();
        var messages = (await GetPendingMessagesAsync(connection, transaction, batchSize)).AsList();

        if (messages.Count != 0)
        {
            await SendBatchToKafkaAsync(messages);
            await MarkMessagesAsSentAsync(connection, transaction, messages.Select(m => m.Id));

            transaction.Commit();
        }

        await Task.Delay(TimeSpan.FromSeconds(5), cancellationToken);
    }
}
```

03. Код не работает

- Нет никакой транзакции между базой и Kafka
- Ничего не починили
- Получили долгие транзакции

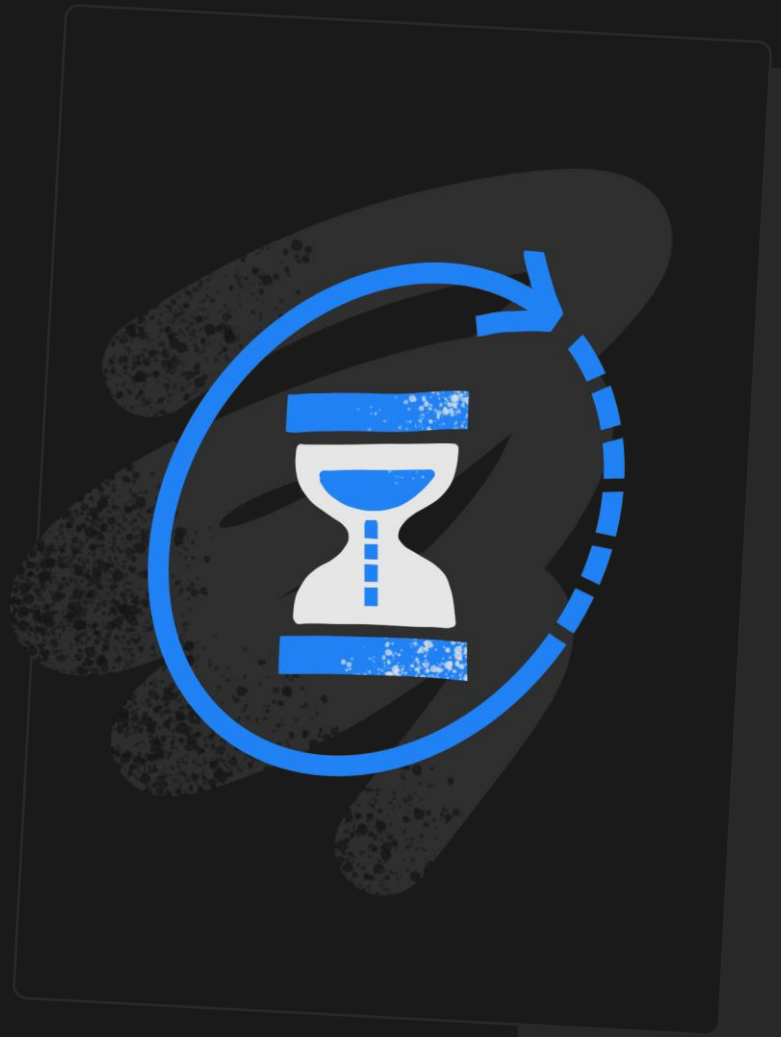
04

Давайте перепишем код!

Код

```
1 usage
private async Task SendMessages(CancellationToken cancellationToken)
{
    while (!cancellationToken.IsCancellationRequested)
    {
        var connection = new SqlConnection(connectionString);
        await connection.OpenAsync(cancellationToken);

        var messages = await GetAndMarkMessagesAsSendingAsync(connection, batchSize);

        if (messages.Count != 0)
        {
            await SendBatchToKafkaAsync(messages);
            await MarkMessagesAsSentAsync(connection, messages.Select(m => m.Id));
        }

        await Task.Delay(TimeSpan.FromSeconds(5), cancellationToken);
    }
}
```

04. Код не работает



- Починили долгие транзакции
- Получили x2 записи
- Теперь ещё и зависшие в sending

Ничего не работает!

- Починить дубли так и не получилось
- Помечать приходится каждое сообщение, а то и по 2 раза
- Что делать с зависшими сообщениями?
- Таблица аутбокса пухнет, индекс болеет
- Автовакуум приходит, и всё вообще встаёт

А ещё будет болеть…

- Вставка\чтение outbox может блокировать всю базу
- Составной ключ сообщения

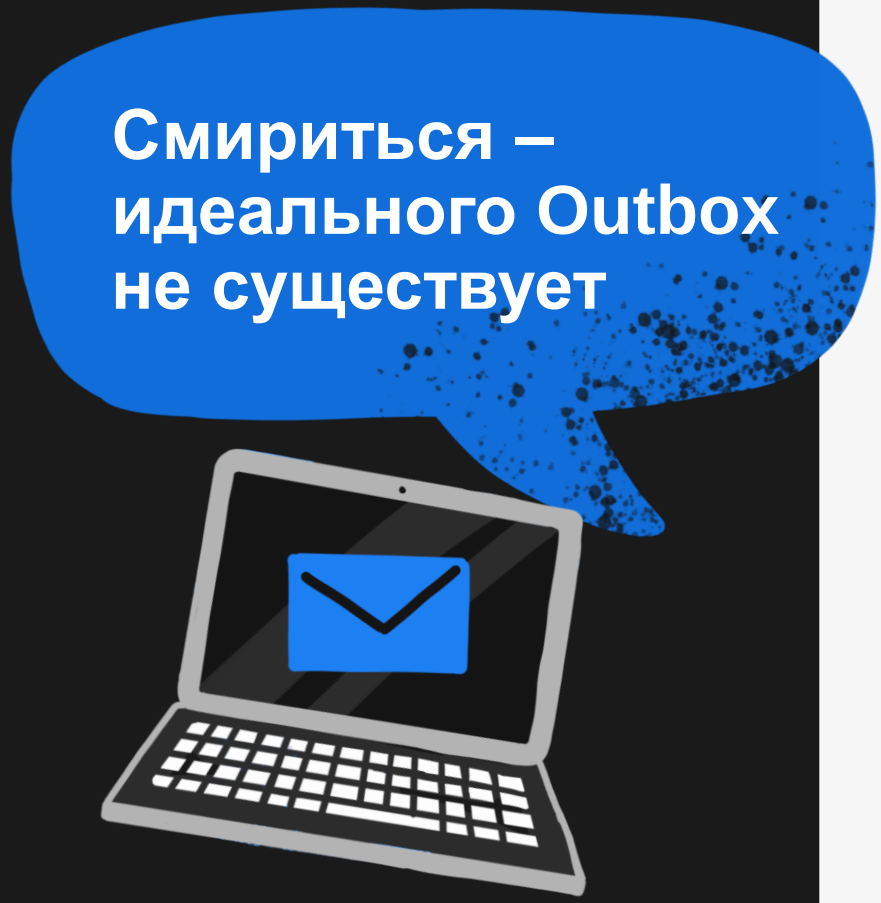
А ещё будет болеть…

- Проблема холодного старта
- Если приёмник умер – в базе копится огромный лаг

А ещё будет болеть…

- Клиенты Kafka могут терять offset

Что со всем этим делать



- Проблему дублей **нельзя решить на отправителе**
- Сообщения всё равно **будут теряться** по той или иной причине
- База **будет падать** и без Outbox
- Порядок можно гарантировать **только в 1 очереди**
- При определенном уровне нагрузки **падает абсолютно всё**

Как чинить дубли

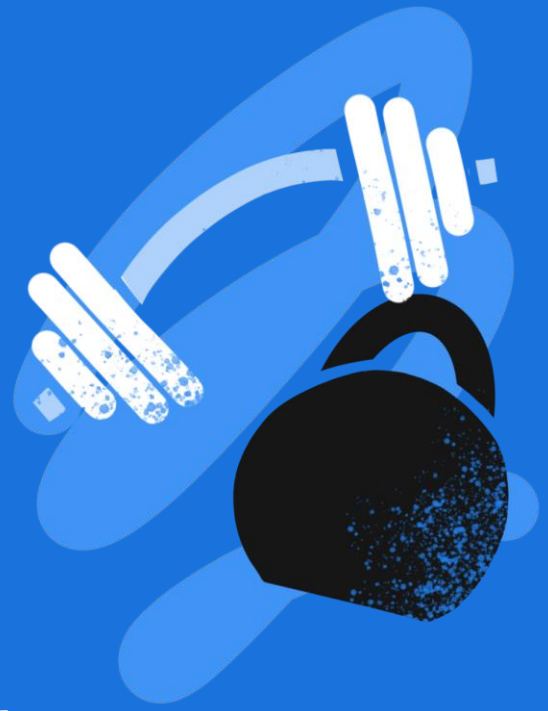
- Дубли – **это нормально**
- Сделать уникальный ключ сообщения
  - UUID – плохо для деревьев
  - UUIDv7 – лучше для деревьев
  - Sequence – зайки на лужайке
- Делаем сообщения идемпотентными для клиента

Как чинить зависшие сообщения

- Watchdog
- Не процессим очередь, пока не отправлена текущая пачка
- Метрики и алерты
- Тулинг допроливок, сдвигов оффеса и т.д.

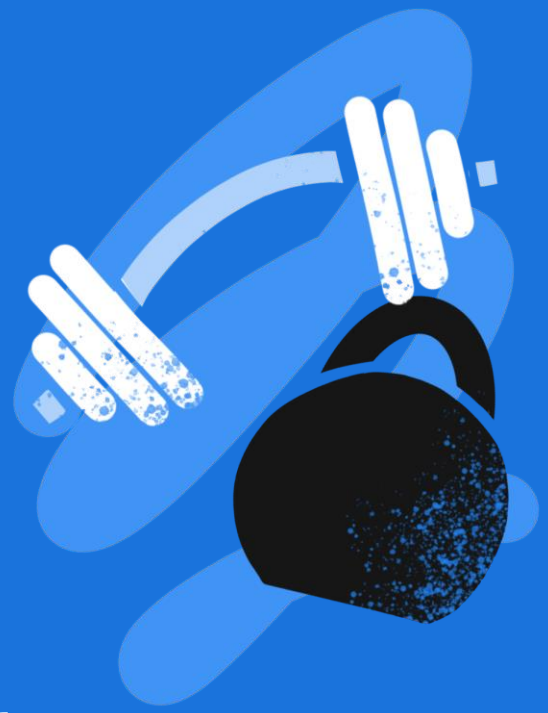
Как бороться со смертью

- Со смертью базы – никак
- Не хранить сообщения в «очереди» базы
- Не лить сразу в базу-приёмник
- Пользоваться простыми, стабильными брокерами
- Тулинг балковой заливки



Как бороться с нагрузкой

- Очевидно, **писать более простой код**

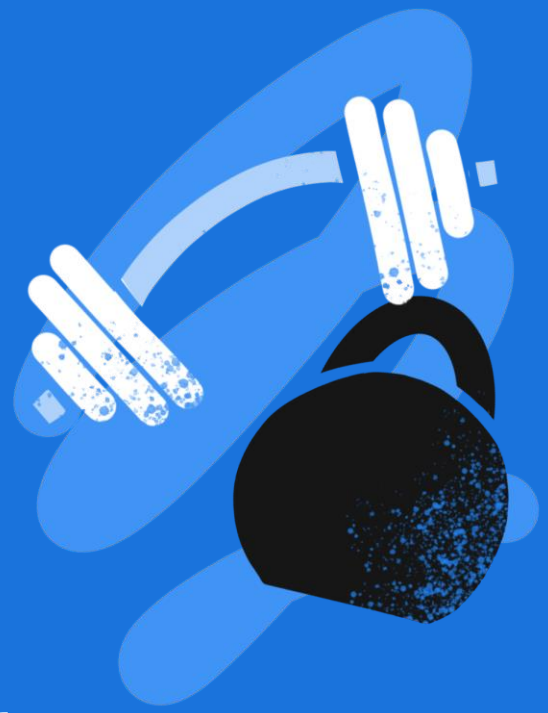


Как бороться с нагрузкой

- Очевидно, писать **более простой код**
- **Партиционирование**
  - Range Partitioning

Range Partitioning

```
2
3   CREATE TABLE outbox
4   (
5       id         BIGSERIAL NOT NULL,
6       entity_id  BIGINT    NOT NULL,
7       payload    JSONB     NOT NULL,
8       created_at TIMESTAMP NOT NULL,
9       PRIMARY KEY (id, created_at)
10  ) PARTITION BY RANGE (created_at);
11
12
13  CREATE TABLE outbox_2024_01 PARTITION OF outbox
14      FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');
15
16  CREATE TABLE outbox_2024_02 PARTITION OF outbox
17      FOR VALUES FROM ('2024-02-01') TO ('2024-03-01');
```

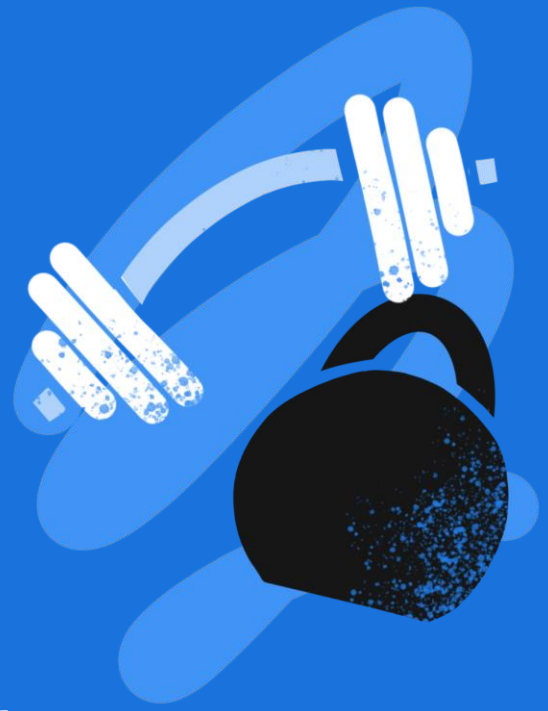


Как бороться с нагрузкой

- Очевидно, писать **более простой код**
- Партиционирование
  - Range Partitioning
  - Hash Partitioning

Hash Partitioning

```
20
21  CREATE TABLE outbox
22  (
23      id         BIGSERIAL NOT NULL,
24      entity_id  BIGINT    NOT NULL,
25      payload    JSONB     NOT NULL,
26      created_at TIMESTAMP NOT NULL,
27      PRIMARY KEY (id)
28  ) PARTITION BY HASH (entity_id);
29
30
31  CREATE TABLE outbox_p0 PARTITION OF outbox
32      FOR VALUES WITH (MODULUS 4, REMAINDER 0);
33
34  CREATE TABLE outbox_p1 PARTITION OF outbox
35      FOR VALUES WITH (MODULUS 4, REMAINDER 1);
36
37  CREATE TABLE outbox_p2 PARTITION OF outbox
38      FOR VALUES WITH (MODULUS 4, REMAINDER 2);
39
40  CREATE TABLE outbox_p3 PARTITION OF outbox
41      FOR VALUES WITH (MODULUS 4, REMAINDER 3);
42
```

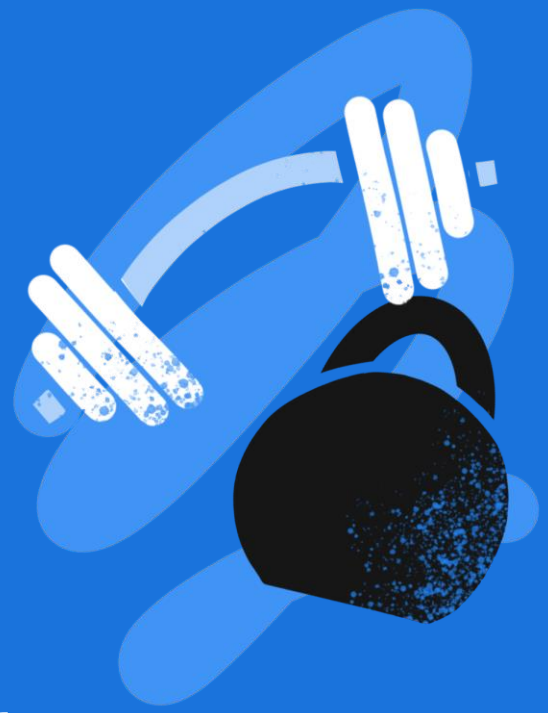


Как бороться с нагрузкой

- Очевидно, писать **более простой код**
- Партиционирование
  - Range Partitioning
  - Hash Partitioning
  - List Partitioning

List Partitioning

```
43
44  CREATE TABLE outbox
45  (
46      id          BIGSERIAL NOT NULL,
47      entity_id   BIGINT    NOT NULL,
48      entity_type INT       NOT NULL,
49      payload     JSONB     NOT NULL,
50      created_at  TIMESTAMP NOT NULL,
51      PRIMARY KEY (id)
52  ) PARTITION BY LIST (entity_type);
53
54
55  CREATE TABLE outbox_type_A PARTITION OF outbox
56      FOR VALUES IN (1);
57
58  CREATE TABLE outbox_type_B PARTITION OF outbox
59      FOR VALUES IN (2);
60
61  CREATE TABLE outbox_type_C PARTITION OF outbox
62      FOR VALUES IN (3);
63
64  CREATE TABLE outbox_type_default PARTITION OF outbox
65      FOR VALUES IN (DEFAULT);
66
```

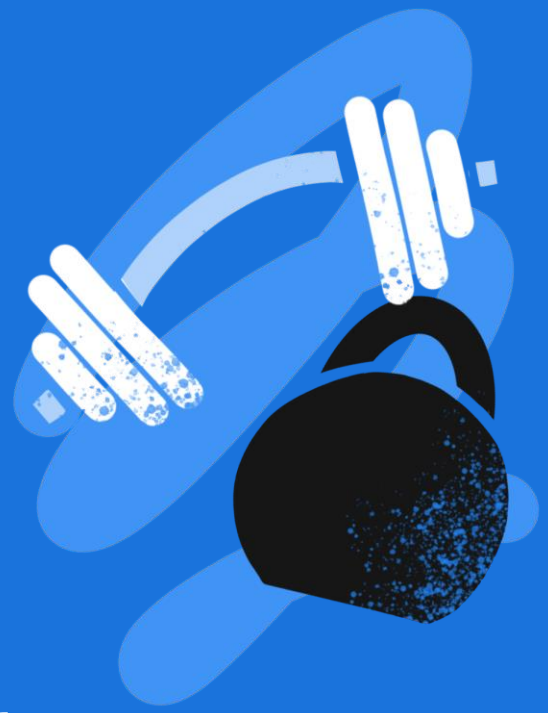


Как бороться с нагрузкой

- Очевидно, писать **более простой код**
- Партиционирование
  - Range Partitioning
  - Hash Partitioning
  - List Partitioning
  - Composite Partitioning

Composite Partitioning

```sql

CREATE TABLE outbox
(
    id          BIGSERIAL NOT NULL,
    entity_id   BIGINT    NOT NULL,
    entity_type INT       NOT NULL,
    payload     JSONB     NOT NULL,
    created_at  TIMESTAMP NOT NULL,
    PRIMARY KEY (id)
) PARTITION BY LIST (entity_type);

-- Партиции по entity_type
CREATE TABLE outbox_type_A PARTITION OF outbox
    FOR VALUES IN (1)
    PARTITION BY RANGE (created_at);

CREATE TABLE outbox_type_B PARTITION OF outbox
    FOR VALUES IN (2)
    PARTITION BY RANGE (created_at);

-- Партиции по диапазонам дат для entity_type = A
CREATE TABLE outbox_type_1_2024_01 PARTITION OF outbox_type_A
    FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');

CREATE TABLE outbox_type_1_2024_02 PARTITION OF outbox_type_A
    FOR VALUES FROM ('2024-02-01') TO ('2024-03-01');

-- Партиции по диапазонам дат для entity_type = B
CREATE TABLE outbox_type_2_2024_01 PARTITION OF outbox_type_B
    FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');

CREATE TABLE outbox_type_2_2024_02 PARTITION OF outbox_type_B
    FOR VALUES FROM ('2024-02-01') TO ('2024-03-01');

-- Партиции по диапазонам дат для других entity_type
CREATE TABLE outbox_type_default_2024_01 PARTITION OF outbox_type_default
    FOR VALUES FROM ('2024-01-01') TO ('2024-02-01');

CREATE TABLE outbox_type_default_2024_02 PARTITION OF outbox_type_default
    FOR VALUES FROM ('2024-02-01') TO ('2024-03-01');

```



Как бороться с нагрузкой

- Очевидно, писать **более простой код**
- Партиционирование
  - Range Partitioning
  - Hash Partitioning
  - List Partitioning
  - Composite Partitioning
- **Шардировать базу**
  - Горизонтально
  - Вертикально

Про что мы **не поговорили**

- Готовые решения
- Оркестрация сотен Outbox
- Пайплайны из Outbox

Спасибо!

Борис Кузоваткин

2024