СУБД: индивидуальный пошив и подгонка по фигуре.

Банальности, проверенные опытом

(C) Дельгядо Филипп, 2016, «Интерактивные технологии и системы»

Базы данных

Велосипеды

Обычный проект

Мало денег

(c) Дельгядо Филипп, 2016

Обычный проект

Мало денег

Обычные программисты

(c) Дельгядо Филипп, 2016

Обычный проект

Мало денег

Обычные программисты

Сжатые сроки

(c) Дельгядо Филипп, 2016

Обычный проект

Мало денег

Обычные программисты

Сжатые сроки

Небольшая команда

(c) Дельгядо Филипп, 2016

Обычный проект

Мало денег

Обычные программисты

Сжатые сроки

Небольшая команда

Не highload

(c) Дельгядо Филипп, 2016

Задача

Платежная система (аналог Яндекс.Денег)

- понятный ограниченный рынок
- небольшая команда

Уже почти год в продакшене

Активно расширяемся)

(c) Дельгядо Филипп, 2016

Что такое платежная система

Много видов платежных операций

сквозные, прямые, возвраты, рекуррентные

(c) Дельгядо Филипп, 2016

Что такое платежная система

Много видов платежных операций
сквозные, прямые, возвраты, рекуррентные

Счета пользователей, персональные данные

(c) Дельгядо Филипп, 2016

Что такое платежная система

Много видов платежных операций
сквозные, прямые, возвраты, рекуррентные

Счета пользователей, персональные данные

Идентификация, регистрация, проверка пользователей

(c) Дельгядо Филипп, 2016

Что такое платежная система

Много видов платежных операций
    сквозные, прямые, возвраты, рекуррентные

Счета пользователей, персональные данные

Идентификация, регистрация, проверка пользователей

Требования контролирующих органов
    PCI DSS, 115-ФЗ, 161-ФЗ, 152-ФЗ, 222-ФЗ

(c) Дельгядо Филипп, 2016

Особенности разработки

Надежность

(c) Дельгядо Филипп, 2016

Особенности разработки

Надежность

Предсказуемость

(c) Дельгядо Филипп, 2016

Особенности разработки

Надежность

Предсказуемость

Простота

(c) Дельгядо Филипп, 2016

Особенности разработки

Надежность

Предсказуемость

Простота

Pessimism Driven Development

(c) Дельгядо Филипп, 2016

В результате получаем

Два ДЦ (в одном городе)

40+ сервисов

- Из них 20+ шлюзов

7 БД PostgreSQL

И все это дублируется

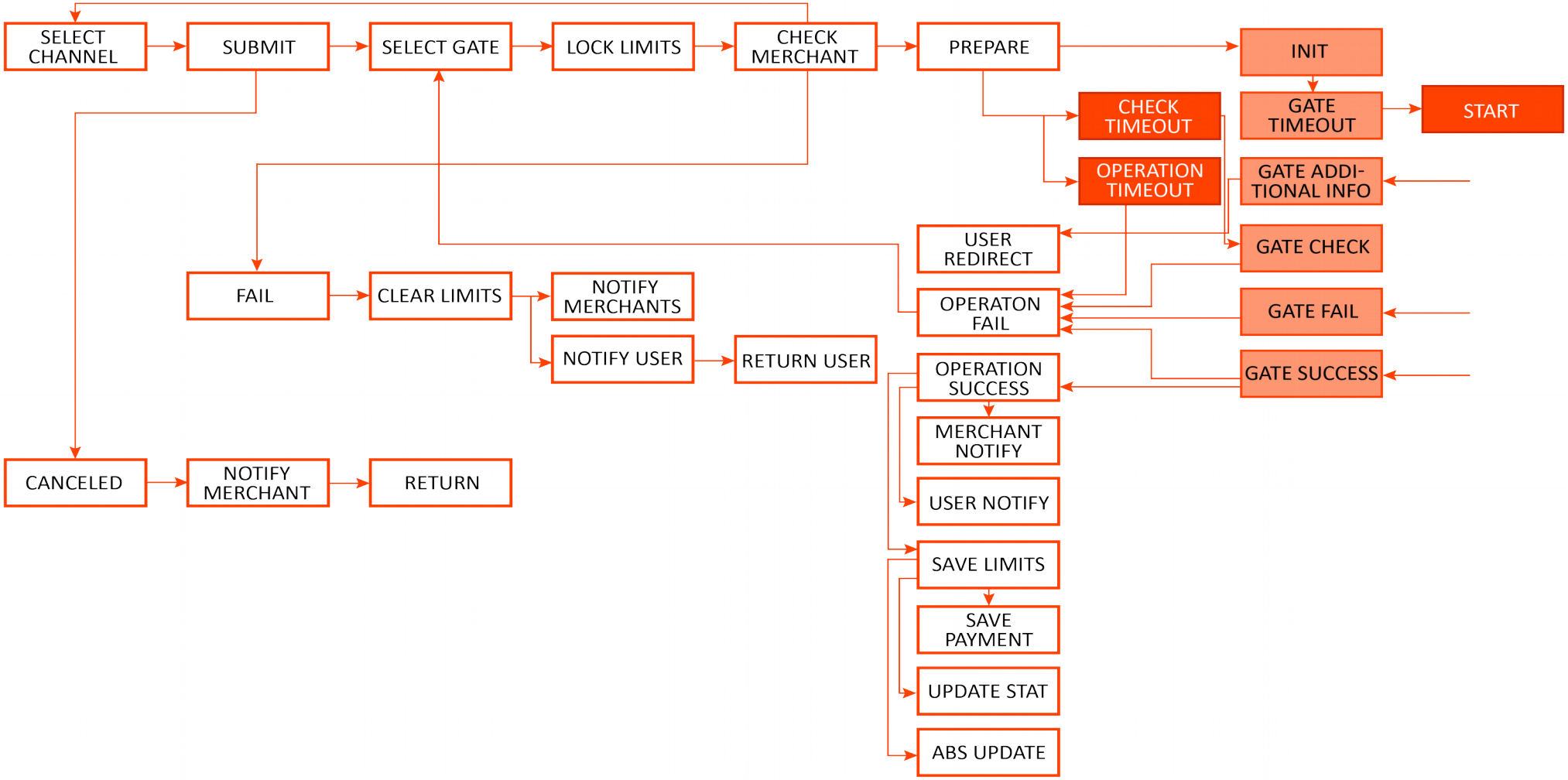
(c) Дельгядо Филипп, 2016

Процесс платежа

Когда обычных очередей не хватает

(c) Дельгядо Филипп, 2016

Что такое платеж



(c) Дельгядо Филипп, 2016

Отдельный шаг в платеже

Порождается другими шагами или внешними событиями

Порождает другие шаги

Меняет СУБД

Выполняется атомарно

Может выполняться отложенно

Идемпотентен

(c) Дельгядо Филипп, 2016

При этом

Шаги не должны мешать друг другу

Одновременные события в платеже должны ждать друг друга

(c) Дельгядо Филипп, 2016

При этом

Шаги не должны мешать друг другу

- Одновременные события в платеже должны ждать друг друга

Шаг обязательно должен быть обработан

- Даже если сервер упал

(c) Дельгядо Филипп, 2016

При этом

Шаги не должны мешать друг другу
- Одновременные события в платеже должны ждать друг друга

Шаг обязательно должен быть обработан
- Даже если сервер упал

Нужно уметь обрабатывать ошибки
- Логика обработки может быть сложной

(c) Дельгядо Филипп, 2016

При этом

Шаги не должны мешать друг другу
- Одновременные события в платеже должны ждать друг друга

Шаг обязательно должен быть обработан
- Даже если сервер упал

Нужно уметь обрабатывать ошибки
- Логика обработки может быть сложной

И нужно обработать примерно 100 платежей в секунду
- Т.е. где-то 1000 шагов в секунду

(c) Дельгядо Филипп, 2016

Нужна очередь событий

Надежная, с гарантией доставки и обработки

1000 событий в секунду

Последовательная обработка внутри очереди

Отложенные события

Поддержка

Работа с очередью в бизнес-транзакции

(c) Дельгядо Филипп, 2016

Нужна очередь событий

Очень не хочется ХА

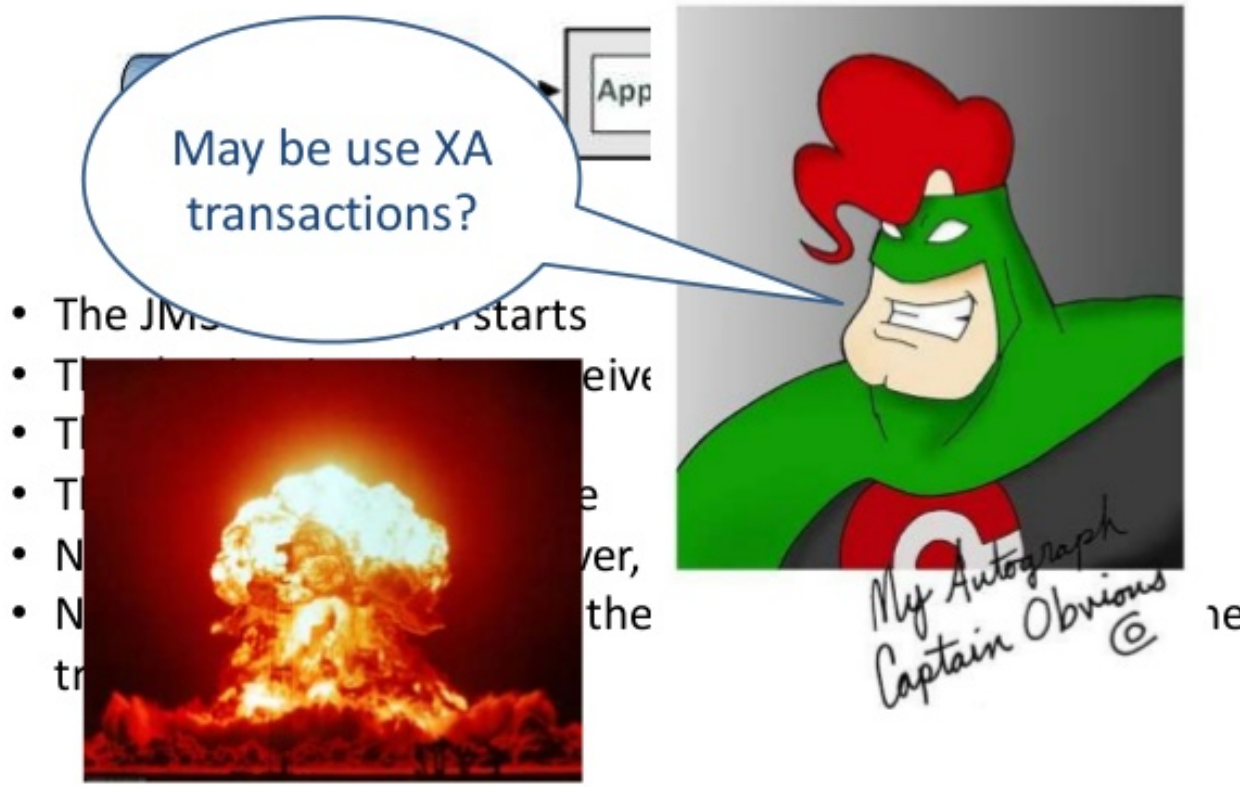
(c) Дельгядо Филипп, 2016

Нужна очередь событий

Очень не хочется XA

«XA транзакции, наверное, самое большое enterprise зло, которое было изобретено»

(C) Николай Алименков



(c) Дельгядо Филипп, 2016

Нужна очередь событий

Очень не хочется ХА

Очень не хочется распределенных блокировок

(c) Дельгядо Филипп, 2016

Нужна очередь событий

Очень не хочется ХА

Очень не хочется распределенных блокировок

Не хочется писать много кода

Не хочется сложных решений

(c) Дельгядо Филипп, 2016

ActiveMQ?

Pros:

- Есть message groups
- Есть переотправка
- Написана на Java

(c) Дельгядо Филипп, 2016

ActiveMQ?

Pros:

- Есть message groups
- Есть переотправка
- Написана на Java

Cons:

- Плохой опыт использования
- Проблемы с надежностью
- Нет внятной поддержки
- Сложно с бизнес-транзакциями

(c) Дельгядо Филипп, 2016

Надежность ActiveMQ

https://ibmadvantage.com/2014/05/30/wmqvsamq/

«*Failover: ActiveMQ lost or duplicated messages during “power outage” and “network failure” scenarios. This is unacceptable in enterprise environments.*»

(c) Дельгядо Филипп, 2016

Проблемы с надежностью

https://ibmadvantage.com/2014/05/30/wmqvsamq/

«Failover: ActiveMQ lost or duplicated messages during “power outage” and “network failure” scenarios. This is unacceptable in enterprise environments.»

https://erikwramner.wordpress.com/2015/09/18/beware-of-activemq-with-jdbc-persistence/

«In other words all messages waiting for redelivery are lost at failover unless the job scheduler store is highly available, which means a highly available KahaDB store. Installations that use JDBC don’t do HA KahaDB»

(c) Дельгядо Филипп, 2016

PGQ

2008 год, Skytools (Skype)

Использует особенности реализации транзакций внутри PostgreSQL

Активно используется для логической репликации

- эффективная пакетная обработка
- Topic queue

(c) Дельгядо Филипп, 2016

PGQ?

Pros:

надежная

на PostgreSQL

поддержка

(c) Дельгядо Филипп, 2016

PGQ?

Pros:

надежная
на PostgreSQL
поддержка

Cons:

рассчитана на пакетную обработку
нет message groups
work queue приделана «сбоку»
скупая документация
сложная логика

(c) Дельгядо Филипп, 2016

RabbitMQ?

Pros:

- Говорят быстрый
- Модный
- Есть поддержка

(c) Дельгядо Филипп, 2016

RabbitMQ?

Pros:

Говорят быстрый
Модный
Есть поддержка

Cons:

exclusive queue не бывает durable
redelivery не настраиваемый
с отложенными событиями сложно
с транзакциями сложно
не Java

(c) Дельгядо Филипп, 2016

Варианты решения

Не бывает идеального решения

(c) Дельгядо Филипп, 2016

Варианты решения

Не бывает идеального решения

Но бывает

- Быстро реализуемое
- Понятное
- Удовлетворяющее задачам
- Расширяемое
- Дешевое

(c) Дельгядо Филипп, 2016

Оглянемся вокруг

В проекте уже есть PostgreSQL 9.5
с качественной поддержкой

(c) Дельгядо Филипп, 2016

Оглянемся вокруг

В проекте уже есть PostgreSQL 9.5
с качественной поддержкой

А в PostgreSQL 9.5 появился **select for update skip locked**
goo.gl/OisdFE — оригинальное письмо

(c) Дельгядо Филипп, 2016

Очередь на skip locked

Таблица очередей

queue
Payment_1
Payment_2
Payment_3
Payment_4

Таблица событий

id	queue	event
239	Payment_1	ProviderSuccess
240	Payment_1	CustomerCancel
241	Payment_2	3ds_redirect
242	Payment_1	PaymentTimeout
243	Payment_2	GateNotify
244	Payment_2	PaymentTimeout

(c) Дельгядо Филипп, 2016

Очередь на skip locked

```
select q.queue, e.event from QUEUES q join EVENTS e
order by e.id asc Limit 1
for update skip locked;
```

queue
Payment_1
Payment_2
Payment_3
Payment_4

id	queue	event
239	Payment_1	ProviderSuccess
240	Payment_1	CustomerCancel
241	Payment_2	3ds_redirect
242	Payment_1	PaymentTimeout
243	Payment_2	GateNotify
244	Payment_2	PaymentTimeout

(c) Дельгядо Филипп, 2016

Очередь на skip locked

```
select q.queue, e.event from QUEUES q join EVENTS e
order by e.id asc Limit 1
for update skip locked;
```

queue
Payment_1
Payment_2
Payment_3
Payment_4

id	queue	event
239	Payment_1	ProviderSuccess
240	Payment_1	CustomerCancel
241	Payment_2	3ds_redirect
242	Payment_1	PaymentTimeout
243	Payment_2	GateNotify
244	Payment_2	PaymentTimeout

(c) Дельгядо Филипп, 2016

Очередь на skip locked

```
select q.queue, e.event from QUEUES q join EVENTS e
order by e.id asc Limit 1
for update skip locked;
```

queue
Payment_1
Payment_2
Payment_3
Payment_4

id	queue	event
239	Payment_1	ProviderSuccess
240	Payment_1	CustomerCancel
241	Payment_2	3ds_redirect
242	Payment_1	PaymentTimeout
243	Payment_2	GateNotify
244	Payment_2	PaymentTimeout

(c) Дельгядо Филипп, 2016

Очередь на skip locked

```
select q.queue, e.event from QUEUES q join EVENTS e
order by e.id asc Limit 1
for update skip locked;
```

queue
Payment_1
Payment_2
Payment_3
Payment_4

id	queue	event
239	Payment_1	ProviderSuccess
240	Payment_1	CustomerCancel
241	Payment_2	3ds_redirect
242	Payment_1	PaymentTimeout
243	Payment_2	GateNotify
244	Payment_2	PaymentTimeout

(c) Дельгядо Филипп, 2016

Очередь на skip locked

```
select q.queue, e.event from QUEUES q join EVENTS e
order by e.id asc Limit 1
for update skip locked;
```

queue
Payment_1
Payment_2
Payment_3
Payment_4

id	queue	event
239	Payment_1	ProviderSuccess
240	Payment_1	CustomerCancel
241	Payment_2	3ds_redirect
242	Payment_1	PaymentTimeout
243	Payment_2	GateNotify
244	Payment_2	PaymentTimeout

(c) Дельгядо Филипп, 2016

Очередь на skip locked

```
select q.queue, e.event from QUEUES q join EVENTS e
order by e.id asc Limit 1
for update skip locked;
```

queue
Payment_1
Payment_2
Payment_3
Payment_4

id	queue	event
240	Payment_1	CustomerCancel
241	Payment_2	3ds_redirect
242	Payment_1	PaymentTimeout
243	Payment_2	GateNotify
244	Payment_2	PaymentTimeout

(c) Дельгядо Филипп, 2016

Очередь на skip locked

```
select q.queue, e.event from QUEUES q join EVENTS e
order by e.id asc Limit 1
for update skip locked;
```

queue
Payment_1
Payment_2
Payment_3
Payment_4

id	queue	event
240	Payment_1	CustomerCancel
241	Payment_2	3ds_redirect
242	Payment_1	PaymentTimeout
243	Payment_2	GateNotify
244	Payment_2	PaymentTimeout

(c) Дельгядо Филипп, 2016

Замечания

Одна очередь на платеж

Внутри каждого платежа обрабатываем события по одному

Добавление события ничего не блокирует

Корректно работает на всех уровнях изоляции

Работа с очередью производится в бизнес-транзакции!

(c) Дельгядо Филипп, 2016

Особенности реализации

А вот теперь про Java

(c) Дельгядо Филипп, 2016

Что мы хотели от очереди

✓Последовательная обработка внутри очереди

? Надежная, с гарантией доставки и обработки

? Работа с очередью в бизнес-транзакции

? Отложенные события

? Сложная логика обработки ошибок

? 1000 событий в секунду

(c) Дельгядо Филипп, 2016

Надежная

```
tx.start(()→
{
    event = dao.getEvent();
    if (event==null) return;
    try
    {
      …логика обработки события
      dao.remove(event);
    }
    catch (Exception e)
    {
      …обработка ошибки
    }

}
```


(c) Дельгядо Филипп, 2016

Надежная

```
tx.start(()→
{
    event = dao.getEvent();
    if (event==null) return;
    error = txAction.start (() →
    {
      …логика обработки события
      dao.remove(event);
    }
    if (error!=null)
     txError.start( ()→
     {
       …обработка ошибки
     }
}
```


(c) Дельгядо Филипп, 2016

Что мы хотели от очереди

- ✓Последовательная обработка внутри очереди
- ✓Надежная, с гарантией доставки и обработки
- ✓Работа с очередью в бизнес-транзакции
- ? Отложенные события
- ? Сложная логика обработки ошибок
- ? 1000 событий в секунду

(c) Дельгядо Филипп, 2016

Базовый запрос

```
Select q.queue, e.id, e.type, e.data
  from queues q
  join events e on q.queue=e.queue
where e.affectedFrom < now() and
   q.executor in (:executor, :ALL)
   and q.state = :ACTIVE and q.type= :SEQUENCE
order by e.id asc limit 1
for update skip locked
```


(c) Дельгядо Филипп, 2016

Отложенные события

```
Select q.queue, e.id, e.type, e.data
  from queues q
  join events e on q.queue=e.queue
where e.affectedFrom < now() and
   q.executor in (:executor, :ALL)
   and q.state = :ACTIVE and q.type= :SEQUENCE
order by e.id asc limit 1
for update skip locked
```


(c) Дельгядо Филипп, 2016

Что мы хотели от очереди

- ✓Последовательная обработка внутри очереди
- ✓Надежная, с гарантией доставки и обработки
- ✓Работа с очередью в бизнес-транзакции
- ✓Отложенные события
- ? Сложная логика обработки ошибок
- ? 1000 событий в секунду

(c) Дельгядо Филипп, 2016

Обработка ошибок

? Останавливаем платеж

? Пробуем повторить позже

? Удаляем событие и идем дальше

✓Делаем что-то сложное

(c) Дельгядо Филипп, 2016

Останавливаем платеж

```
Select q.queue, e.id, e.type, e.data
  from queues q
  join events e on q.queue=e.queue
where e.affectedFrom < now() and
   q.executor in (:executor, :ALL)
    and q.state = :ACTIVE and q.type= :SEQUENCE
order by e.id asc limit 1
for update skip locked
```


(c) Дельгядо Филипп, 2016

Обработка ошибок

✓Останавливаем платеж

? Пробуем повторить позже

? Удаляем событие и идем дальше

✓Делаем что-то сложное

(c) Дельгядо Филипп, 2016

Обработка ошибок

✓Останавливаем платеж

? Пробуем повторить позже

считаем попытки в отдельной табличке

задержку считаем как attempCount->delayInMs

? Удаляем событие и идем дальше

✓Делаем что-то сложное

(c) Дельгядо Филипп, 2016

Обработка ошибок

✓Останавливаем платеж

✓Что-то делаем и пробуем повторить позже
считаем попытки в отдельной табличке
задержку считаем как attempCount->delayInMs

? Удаляем событие и идем дальше
dao.remove(event)

✓Делаем что-то сложное

(c) Дельгядо Филипп, 2016

Что мы хотели от очереди

- ✓Последовательная обработка внутри очереди
- ✓Надежная, с гарантией доставки и обработки
- ✓Работа с очередью в бизнес-транзакции
- ✓Отложенные события
- ✓Сложная логика обработки ошибок
- ? 1000 событий в секунду

(c) Дельгядо Филипп, 2016

Что еще хочется

Привязка очереди к обработчику

exclusive queue

Последовательная и параллельная обработка

(c) Дельгядо Филипп, 2016

Привязка к обработчику

```
Select q.queue, e.id, e.type, e.data
  from queues q
  join events e on q.queue=e.queue
where e.affectedFrom < now() and
   q.executor in (:executor, :ALL)
   and q.state = :ACTIVE and q.type= :SEQUENCE
order by e.id asc limit 1
for update skip locked
```


(c) Дельгядо Филипп, 2016

Последовательная обработка

```
Select q.queue, e.id, e.type, e.data
  from queues q
  join events e on q.queue=e.queue
where e.affectedFrom < now() and
   q.executor in (:executor, :ALL)
   and q.state = :ACTIVE and q.type= :SEQUENCE
order by e.id asc limit 1
for update skip locked
```


(c) Дельгядо Филипп, 2016

Параллельная обработка

```
Select q.queue, e.id, e.type, e.data
  from queues q
  join events e on q.queue=e.queue
where e.affectedFrom < now() and
   q.executor in (:executor, :ALL)
   and q.state = :ACTIVE and q.type= :PARALLEL
order by e.id asc limit 1
for update of e skip locked
```


(c) Дельгядо Филипп, 2016

Что мы хотели от очереди

- ✓Последовательная обработка внутри очереди
- ✓Надежная, с гарантией доставки и обработки
- ✓Работа с очередью в бизнес-транзакции
- ✓Отложенные события
- ✓Сложная логика обработки ошибок
- ? 1000 событий в секунду

(c) Дельгядо Филипп, 2016

Как мы разбираем события

Читаем из очереди
в несколько потоков

(c) Дельгядо Филипп, 2016

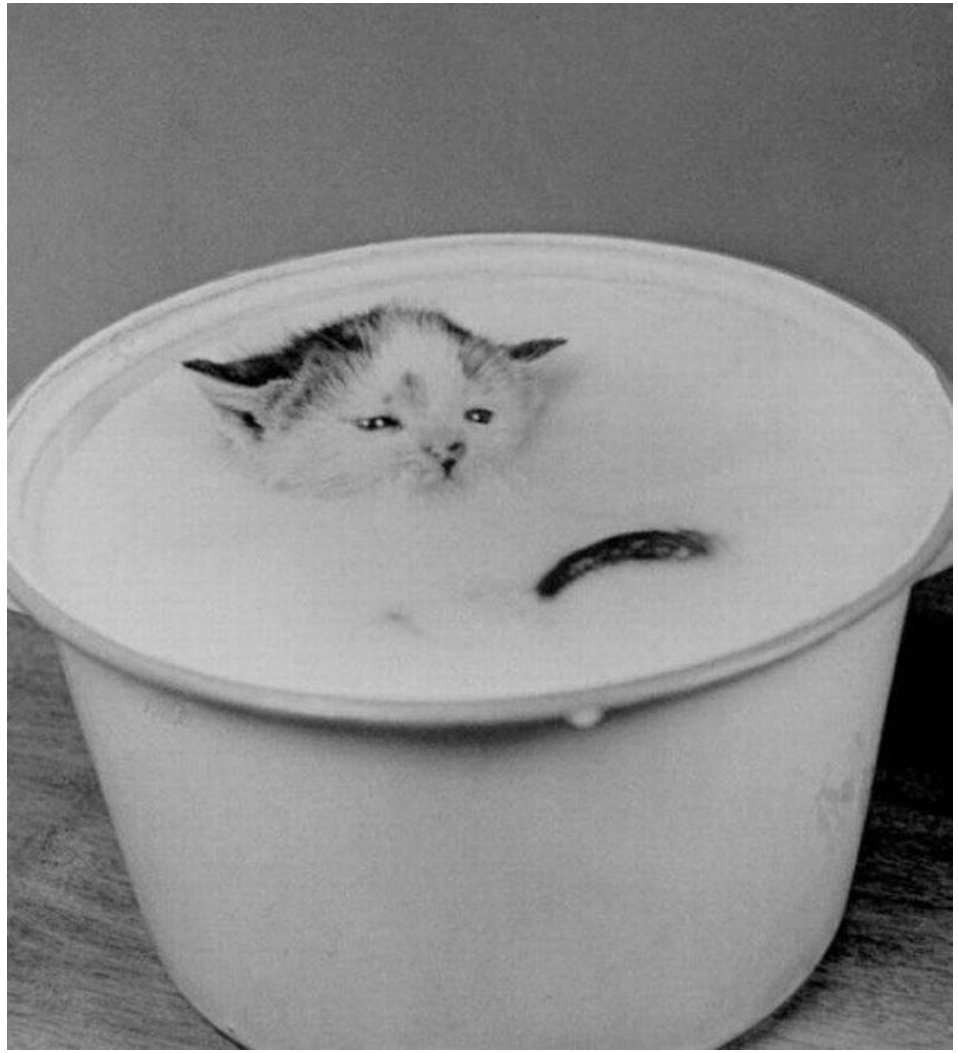
Как мы разбираем события

Потоков много,
событий мало

(c) Дельгадо Филипп, 2016

Как мы разбираем события

Событий много,
потоков не хватает



(c) Дельгядо Филипп, 2016

Уже знакомый код

```
tx.start(()→
{
    event = dao.getEvent();
    if (event==null) return;
    error = txAction.start (() →
    {
      …логика обработки события
      dao.remove(event);
    }
    if (error!=null)
     txError.start( ()→
     {
       …обработка ошибки
     }
}
```


(c) Дельгядо Филипп, 2016

Размножимся

```
tx.start(()→
{
    event = dao.getEvent();
    if (event==null) return;
    fork();
    error = txAction.start (() →
    {
      …логика обработки события
      dao.remove(event);
    }
    if (error!=null)
     txError.start( ()→
     {
       …обработка ошибки
     }
}
```


(c) Дельгядо Филипп, 2016

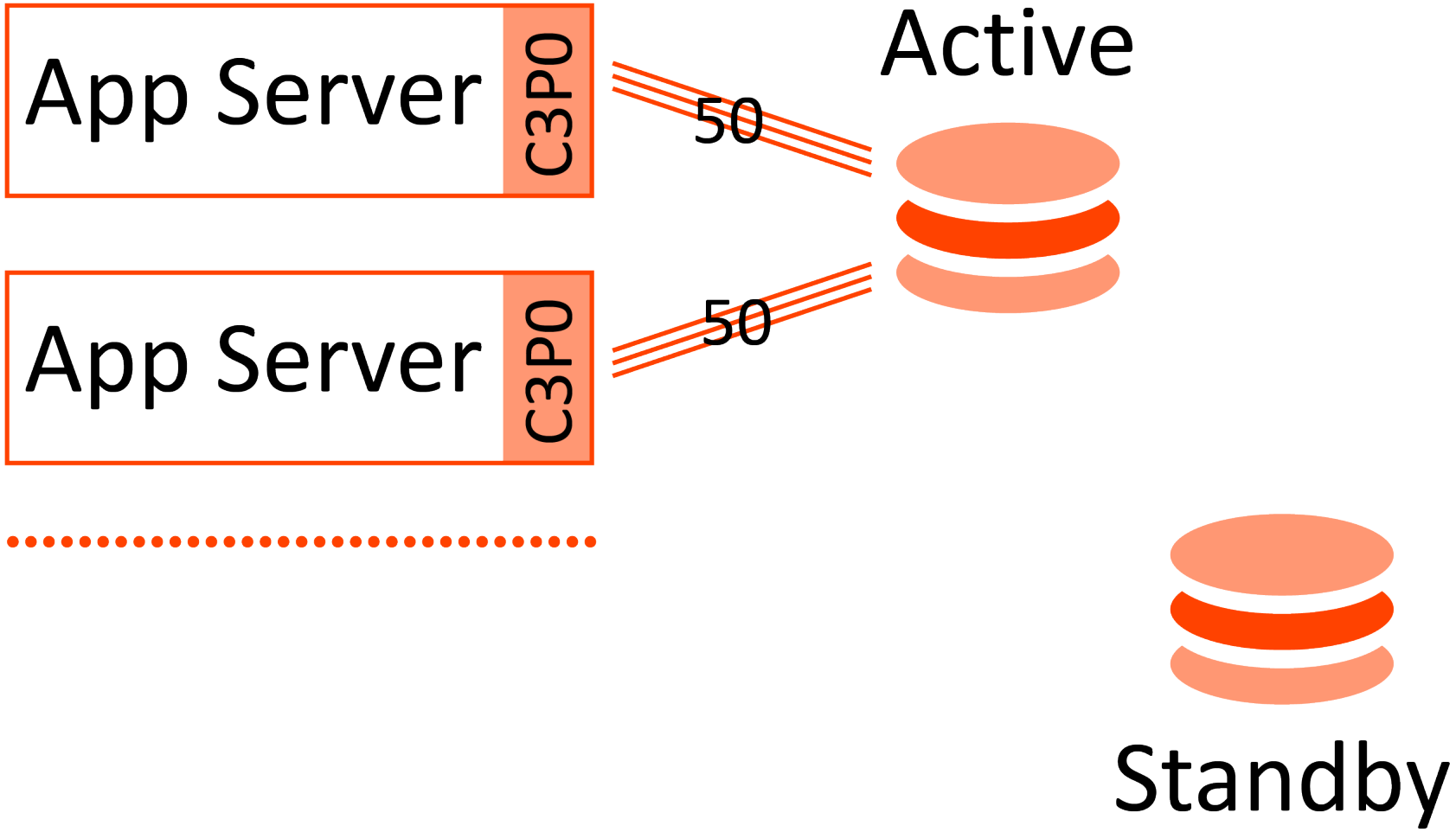
Итого

Как следствие — нагрузка быстро растет, если очередь полна и минимальная при простоях

При этом общее число обработчиков лимитируется пулом тредов (и пулом соединений)

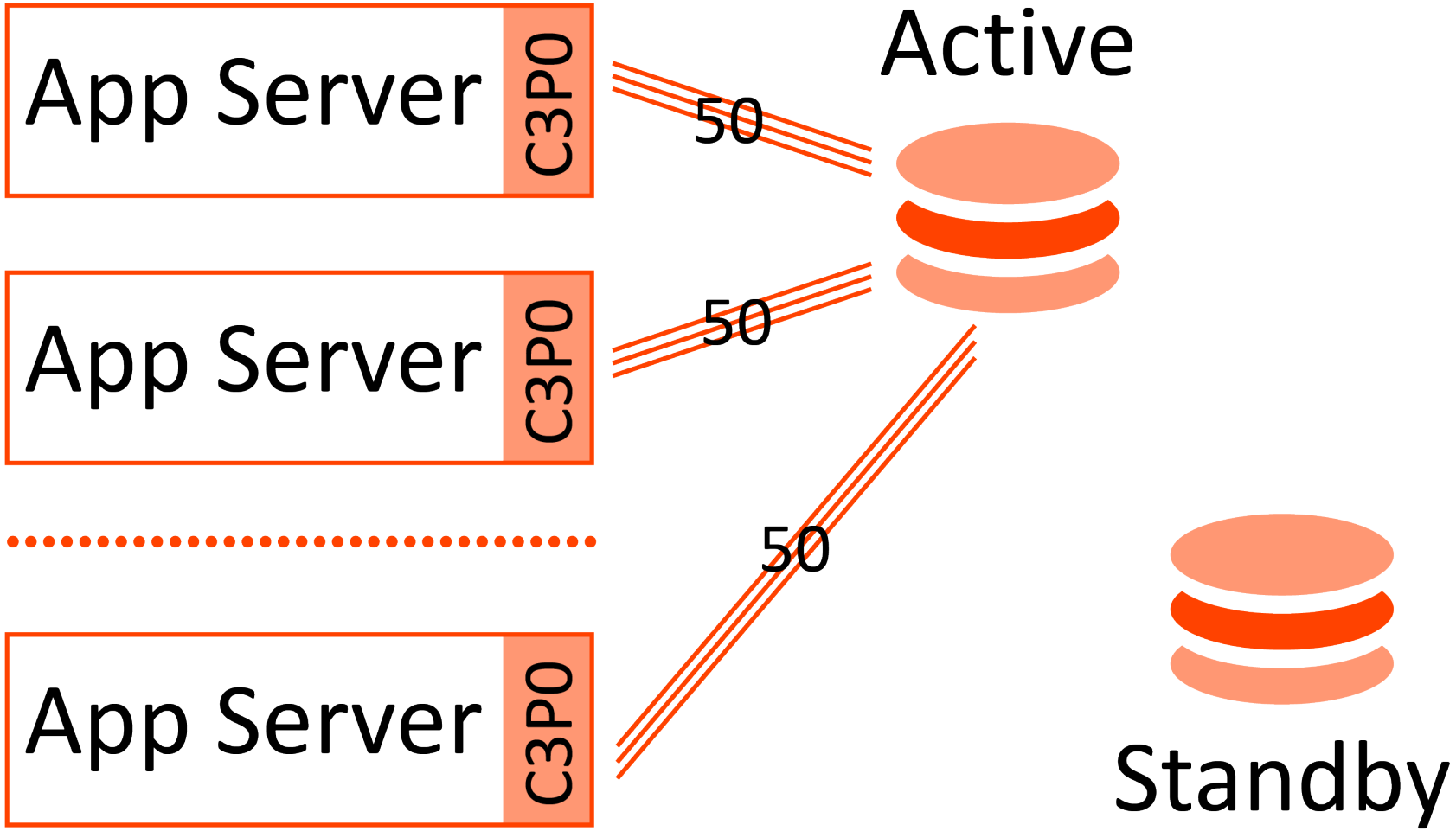
(c) Дельгядо Филипп, 2016

Пул соединений



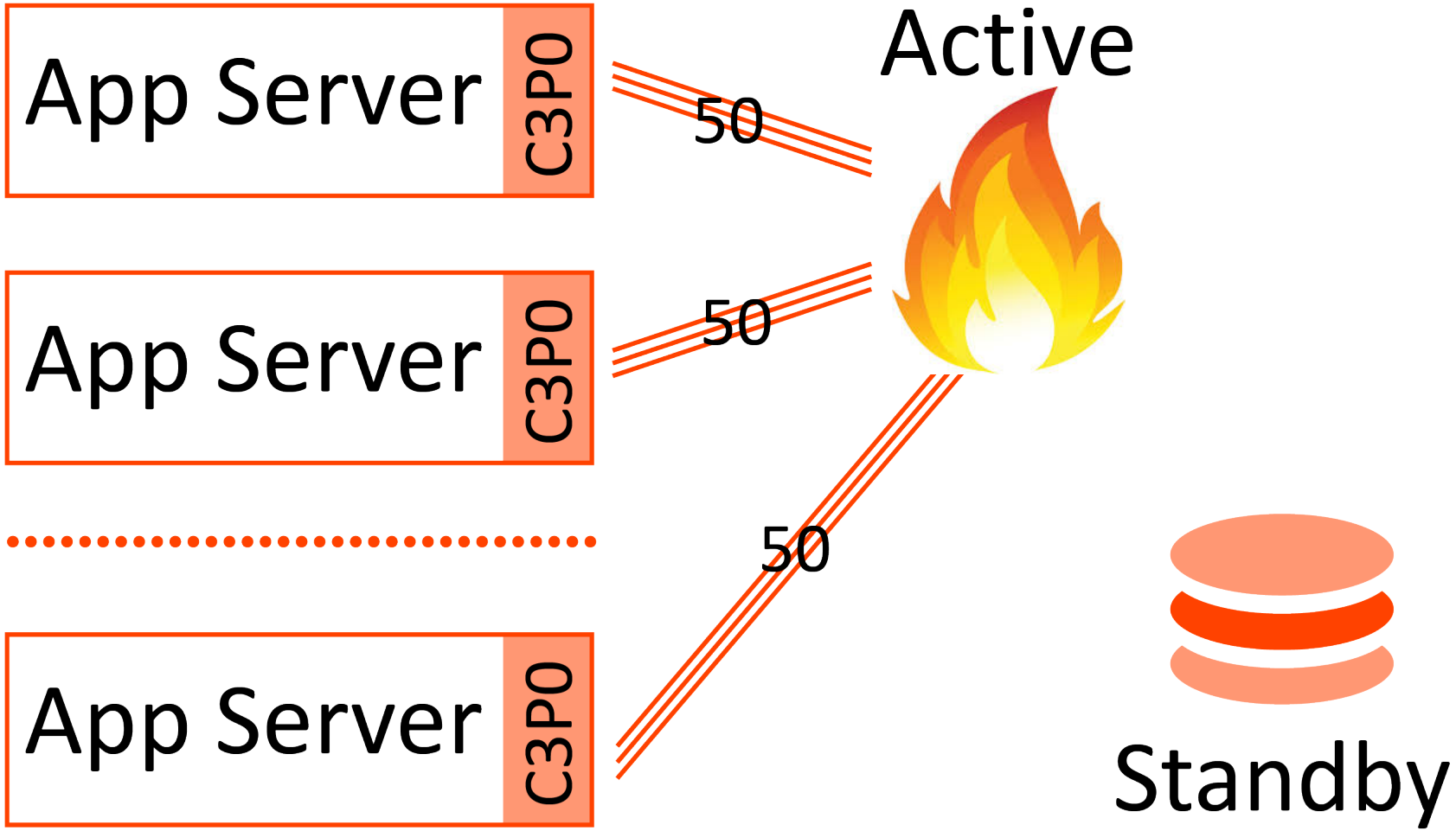
(c) Дельгядо Филипп, 2016

Пул соединений



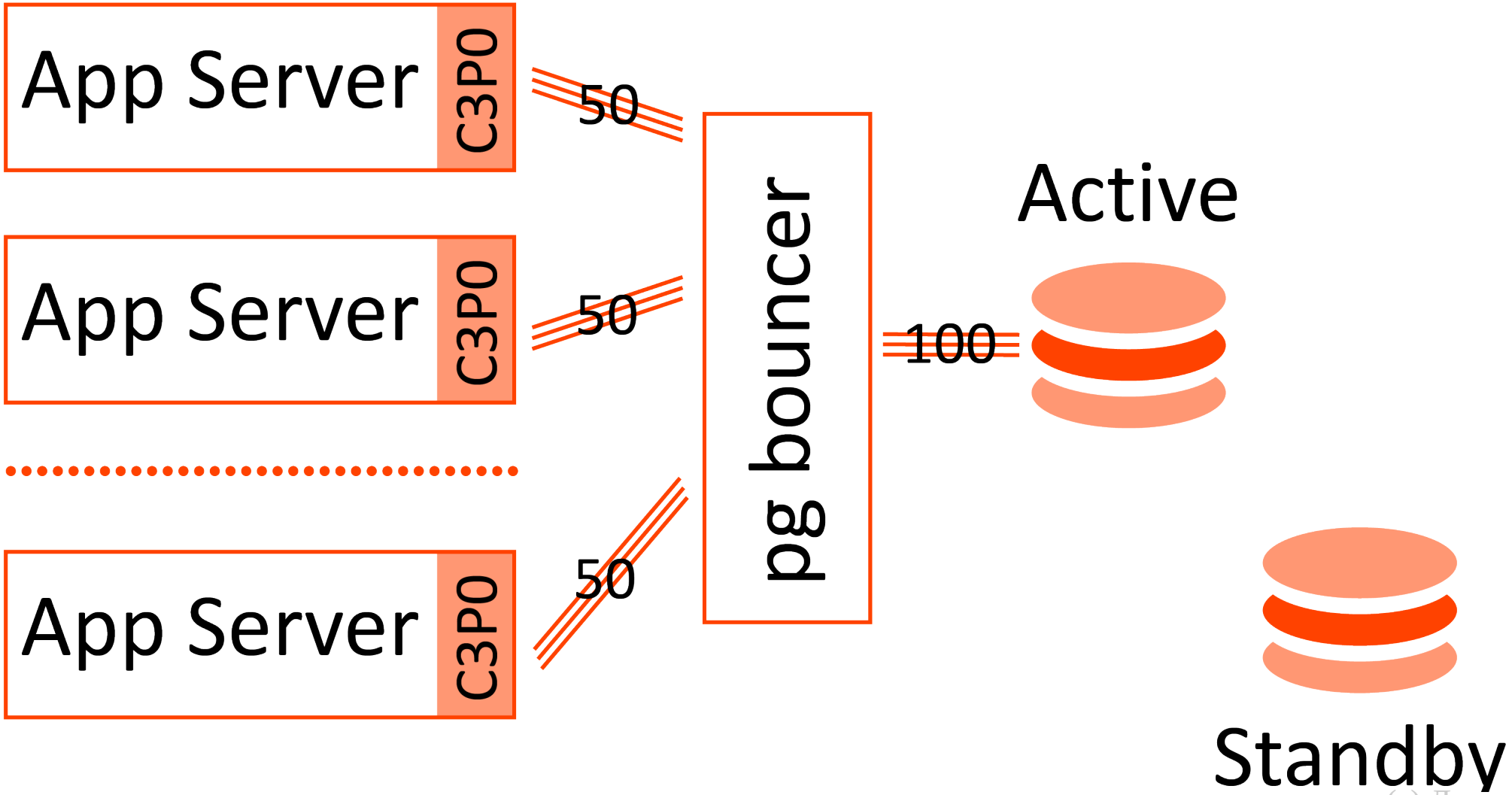
(c) Дельгядо Филипп, 2016

Пул соединений



(c) Дельгядо Филипп, 2016

Пул соединений



(c) Дельгядо Филипп, 2016

Померим нагрузку

10 000 «платежей» из 10 шагов

Каждый шаг:

- load context from DB
- 10 ms wait
- save context to DB

Настройки «из коробки»

(c) Дельгядо Филипп, 2016

Померим нагрузку

Мой ноутбук (i5, два ядра, SSD, Win10)

40 платежей в секунду

(c) Дельгядо Филипп, 2016

Померим нагрузку

Мой ноутбук (i5, два ядра, SSD, Win10)

40 платежей в секунду

Два раздельных сервера (i5, 8 ядер, SSD, Linux)

100 платежей в секунду

(c) Дельгядо Филипп, 2016

Что мы хотели от очереди

- ✓Последовательная обработка внутри очереди
- ✓Надежная, с гарантией доставки и обработки
- ✓Работа с очередью в бизнес-транзакции
- ✓Отложенные события
- ✓Сложная логика обработки ошибок
- ✓1000 событий в секунду

(c) Дельгядо Филипп, 2016

Что мы хотели от очереди

- ✓Последовательная обработка внутри очереди
- ✓Надежная, с гарантией доставки и обработки
- ✓Работа с очередью в бизнес-транзакции
- ✓Отложенные события
- ✓Сложная логика обработки ошибок
- ✓1000 событий в секунду
- Статистика и мониторинг

(c) Дельгядо Филипп, 2016

Итого

700 строчек кода

2 дня на concept proof

2 недели на разработку
(в основном на тесты и обсуждение API)

(c) Дельгядо Филипп, 2016

Технологии

PostrgeSQL

- select for update skip locked
- nested transactions

Java

- Spring Transaction Template
- Spring JDBCTemplates
- ScheduledThreadExecutor
- c3p0

(c) Дельгядо Филипп, 2016

Проблемы

Куча «условно удачных» решений:

- неравномерная балансировка нагрузки
- требуется много соединений к БД
- нельзя разменять надежность на производительность

Точно есть что-то лучше

Нужно много тестов

(c) Дельгядо Филипп, 2016

Достоинства

Точно знаем как работает

Легко добавлять функциональность

Никаких ХА

(c) Дельгядо Филипп, 2016

Доменная модель

Или про жизнь без ORM

(c) Дельгядо Филипп, 2016

Очень много таблиц…


```
"id": 22,
  "requestId": "X00054762580001",
  "workflowType": "Refillflow",
  "state": "SUCCESS",
  "createTimestamp": 1464399[illegible]0847,
  "stateTimestamp": 1464399[illegible]7557,
  "runtimeStation": "FINCHBIG",
  "merchantId": "1gateway",
  "customerId": "1",
  "merchantCustomerId": "MPersonId",
  "description": "testmmwk engine",
  "amount": {"amount": 100.00, "currencyCode": "RUB"},
  "customerFee": {"amount": 0.00, "currencyCode": "RUB"},
  "merchantFee": {"amount": 0.00, "currencyCode": "RUB"},
  "providerFee": {"amount": 0.00, "currencyCode": "RUB"},
  "bankFee": {"amount": 0.00, "currencyCode": "RUB"},
  "statusLog": ["START", "GATE_SELECT", "GATE_START", "GATE_WAIT", "TRANSACTIONS","NOTIFICATION",
"SUCCESS"],
  "gateLog": [{"operationId": 67667[illegible]357,"gateId": "pyga_ln","extOperationCode":
"WAMMO_67667[illegible]357", "extDate": "2016-05-24", "failType": null,"externalError": null,"timeout": false,
"timeoutCount": 0,"deadline": "2016-05-24T17:27:10.[illegible]Z"},
  "paymentChannel": {"channel": "RUSMOBILEAGGREGATOR", "externalAccountName": "AccName",
"externalAccountCode": "1gMegap", "postponed": false,"externalAccountCustomerId": null },
  "gateRef": {"gateId": "pyga_ln", "params": {"mode": "simple_success","payment_account":
"phone_account"
    "order": 1, "timeout": 0, "timeoutSec": 60, "deadlineTimeoutSec": 300, "test": false, "enabled": true },
  "gateWhitelist": [],
  "operationId": 67667[illegible]357,
  "outgoings": [{"operationId": 0,"active": true,"outgoing":     {"type":
"FLOW_STATUS_SUCCESS_REFILL", "url": null,"params": null}   } ],
  "acquirerTransactionCode": 67667[illegible]357,
  "emitterTransactionCode": "WAMMO_67667[illegible]357",
  "configversionStart": 224,
  "configversionGateStart": 224,
  "configversionTransactions": 224,
  "failReason": null,
  "paymentAccount": "AccName",
  "manualPaymentCompletion": null,
  "extraParameters": {"sourceIP": "0:0:0:0:0:0:0:1"},
  "startTime": "2016-05-24T14:22:49.847Z",
  "completionTime": "2016-05-24T14:22:50.238Z",
  "providerRegistryDate": "2016-05-24"
```


контекст платежа

(c) Дельгядо Филипп, 2016

Очень много таблиц…


```
"id": 22,
  "requestId": "X00547625B000",
  "workflowType": "Refill/Flow",
  "state": "SUCCESS",
  "createTimestamp": 1464099[illegible]0847,
  "stateTimestamp": 1464099[illegible]7557,
  "runtimeStation": "FINCHBOX",
  "merchantId": "1gatewb",
  "customerId": "1",
  "merchantCustomerId": "BAPersonId",
  "description": "testmob engine",
  "amount": {"amount": 100.00, "currencyCode": "RUB"},
  "customerFee": {"amount": 0.00, "currencyCode": "RUB"},
  "merchantFee": {"amount": 0.00, "currencyCode": "RUB"},
  "providerFee": {"amount": 0.00, "currencyCode": "RUB"},
  "bankFee": {"amount": 0.00, "currencyCode": "RUB"},
  "statusLog": ["START", "GATE_SELECT", "GATE_START", "GATE_WAIT", "TRANSACTIONS","NOTIFICATION",
"SUCCESS"],
  "gateLog": [{"operationId": 67607[illegible]157,"gateId": "pyga_lo","extOperationCode":
"GAMMA_67607[illegible]157", "extDate": "2016-05-24", "failType": null,"externalError": null,"timeout": false,
"timeoutCount": 0,"deadline": "2016-05-24T17:27:10.[illegible]Z"},
  "paymentChannel": {"channel": "RSYMOBILEKASSSS", "externalAccountName": "AccName",
"externalAccountCode": "[illegible]", "postponed": false,"externalAccountCustomerId": null },
  "gateRef": {"gateId": "pyga_lo", "params": {"mode": "simple_success","payment_account":
"phone_account",
    "order": 1, "timeout": 0, "timeoutSec": 60, "deadlineTimeoutSec": 300, "test": false, "enabled": true },
  "gateWhitelist": [],
  "operationId": 67607[illegible]157,
  "outgoing": [{"operationId": 0,"active": true,"outgoing": {"type":
"FLOW_STATUS_SUCCESS_REFILL", "url": null,"params": null} } },
  "acquirerTransactionCode": "67607[illegible]157",
  "emitterTransactionCode": "GAMMA_67607[illegible]157",
  "configversionStart": 224,
  "configversionGateStart": 224,
  "configversionTransactions": 224,
  "failReason": null,
  "paymentAccount": "AccName",
  "manualPaymentCompletion": null,
  "extraParameters": {"sourceIP": "0:0:0:0:0:0:0:1"},
  "startTime": "2016-05-24T14:22:49.847Z",
  "completeTime": "2016-05-24T14:22:50.238Z",
  "providerRegistryDate": "2016-05-24"
```


Много таблиц и связей

Сложно менять структуру

Дорогая поддержка

Без ОРМа не выжить

(c) Дельгядо Филипп, 2016

Сложно менять структуру

```
alter table TEST
    add column NULLABLECOLUMN varchar;
```


(c) Дельгядо Филипп, 2016

Сложно менять структуру

```
alter table TEST
    add column NULLABLECOLUMN varchar;
```

В некоторых сценариях может полностью блокировать работу с таблицей TEST на долгое время.

www.facebook.com/groups/postgresql/permalink/506304452899640/

(c) Дельгядо Филипп, 2016

Очень много таблиц…


```
"id": 22,
  "requestId": "X00054762580001",
  "workflowType": "Refillflow",
  "state": "SUCCESS",
  "createTimestamp": 1464399[illegible]0847,
  "stateTimestamp": 1464399[illegible]7557,
  "runtimeStatus": "FINISHED",
  "merchantId": "1gatewb",
  "customerId": "1",
  "merchantCustomerId": "MPersonId",
  "description": "testmob engine",
  "amount": {"amount": 100.00, "currencyCode": "RUB"},
  "customerFee": {"amount": 0.00, "currencyCode": "RUB"},
  "merchantFee": {"amount": 0.00, "currencyCode": "RUB"},
  "providerFee": {"amount": 0.00, "currencyCode": "RUB"},
  "bankFee": {"amount": 0.00, "currencyCode": "RUB"},
  "statesLog": ["START", "GATE_SELECT", "GATE_START", "GATE_WAIT", "TRANSACTIONS","NOTIFICATION",
"SUCCESS"],
  "gateLog": [{"operationId": 67667[illegible]357,"gateId": "pyga_ln","extOperationCode":
"WEMU_67667[illegible]357", "extDate": "2016-05-24", "failType": null,"externalError": null,"timeout": false,
"timeoutCount": 0,"deadline": "2016-05-24T17:27:10.[illegible]Z"},
  "paymentChannel": {"channel": "RESPONSIBLESUCCESS", "externalAccountName": "AccName",
"externalAccountCode": "qWWapp", "postponed": false,"externalAccountCustomerId": null },
  "gateRef": {"gateId": "pyga_ln", "params": {"mode": "simple_success","payment_account":
"phone_account"
    "order": 1, "timeout": 0, "timeoutSec": 60, "deadlineTimeoutSec": 300, "test": false, "enabled": true },
  "gateWhitelist": [],
  "operationId": 67667[illegible]357,
  "outgoings": [{"operationId": 0,"active": true,"outgoing":     {"type":
"FLOW_STATUS_SUCCESS_REFILL", "url": null,"params": null}   } ],
  "acquirerTransactionCode": "67667[illegible]357",
  "emitterTransactionCode": "WEMU_67667[illegible]357",
  "configureTimeStart": 224,
  "configureTimeGateStart": 224,
  "configureTimeTransactions": 224,
  "failReason": null,
  "paymentAccount": "AccName",
  "manualPaymentCompletion": null,
  "extraParameters": {"sourceIP": "0:0:0:0:0:0:0:1"},
  "startTime": "2016-05-24T14:22:49.847Z",
  "completeTime": "2016-05-24T14:22:50.238Z",
  "providerRegistryDate": "2016-05-24"
```


Или есть другой путь?

(c) Дельгядо Филипп, 2016

А что нам нужно?

Прочитать сложную сущность из БД

Сохранить ее изменения

Найти сущность по каким-то полям
обычно PK и что-то еще

(c) Дельгядо Филипп, 2016

Например, так

```
create table operations (
  id bigint not null primary key,
  userId bigint,
  state varchar,
  data text );
```


(c) Дельгядо Филипп, 2016

Что сделали?

Храним сложные сущности внутри записей

- как-то их при этом сериализуем
- например в json

Придумываем, как быстро искать сущности

- в отдельные поля выделяем то, где ищем

(c) Дельгядо Филипп, 2016

Что это дает

Простота работы с БД

(c) Дельгядо Филипп, 2016

Что это дает

Простота работы с БД

Простота обновления структуры

(c) Дельгядо Филипп, 2016

Что это дает

Простота работы с БД

Простота обновления структуры

Производительность

(c) Дельгядо Филипп, 2016

Что это дает

Простота работы с БД

Простота обновления структуры

Производительность

Атомарность изменений

(c) Дельгядо Филипп, 2016

Чем плохо

Не поддерживается ОРМами

(c) Дельгядо Филипп, 2016

Чем плохо

Не поддерживается ОРМами

Сложность администрирования

(c) Дельгядо Филипп, 2016

Чем плохо

Не поддерживается ОРМами

Сложность администрирования

Сложность для построения отчетов

(c) Дельгядо Филипп, 2016

Чем плохо

Не поддерживается ОРМами

Сложность администрирования

Сложность для построения отчетов

Атомарность изменений

(c) Дельгядо Филипп, 2016

Как хранить

PostgreSQL:

(c) Дельгядо Филипп, 2016

Как хранить

PostgreSQL:

::varchar – произвольный размер, упаковка

(c) Дельгядо Филипп, 2016

Как хранить

PostgreSQL:

::varchar – произвольный размер, упаковка

::text – синоним для varchar

(c) Дельгядо Филипп, 2016

Как хранить

PostgreSQL:

::varchar – произвольный размер, упаковка

::text – синоним для varchar

::json – валидация, расширение SQL для доступа

(c) Дельгядо Филипп, 2016

Как хранить

PostgreSQL:

::varchar – произвольный размер, упаковка

::text – синоним для varchar

::json – валидация, расширение SQL для доступа

::jsonb – эффективное бинарное хранение и доступ

(c) Дельгядо Филипп, 2016

Как хранить

PostgreSQL:

::varchar – произвольный размер, упаковка

::text – синоним для varchar

::json – валидация, расширение SQL для доступа

::jsonb – эффективное бинарное хранение и доступ

Oracle, MS SQL, MySQL — собственные решения

(c) Дельгядо Филипп, 2016

Как это выглядит в БД

```
select id,
  data->'amount'->>'amount',
  data->'amount'->>'currency'
from operations;
```


(c) Дельгядо Филипп, 2016

Как это выглядит в БД

Лучше так не делать

```
select id,
  data->'amount'->>'amount',
  data->'amount'->>'currency'
from operations;
```


(c) Дельгядо Филипп, 2016

Подстилаем соломку…

Храним номер версии сериализации

- он точно будет быстро расти

(c) Дельгядо Филипп, 2016

Как это выглядит в БД

```
create table operations (
  id bigint not null primary key,
  userId bigint,
  state varchar,
  schema bigint not null,
  data json );
```


(c) Дельгядо Филипп, 2016

Подстилаем соломку…

Храним номер версии сериализации

- он точно будет быстро расти

Прописываем приоритет индексных полей над содержимым

- или не сериализуем лишнее

(c) Дельгядо Филипп, 2016

Как это выглядит в БД

```
create table operations (
  id bigint not null primary key,
  userId bigint,
  state varchar,
  schema bigint not null,
  data json );
```


(c) Дельгядо Филипп, 2016

Подстилаем соломку…

Храним номер версии сериализации

- он точно будет быстро расти

Прописываем приоритет индексных полей над содержимым

- или не сериализуем лишнее

Заранее готовим код для обновления версий

- он точно пригодится

(c) Дельгядо Филипп, 2016

Подстилаем соломку…

Храним номер версии сериализации

- он точно будет быстро расти

Прописываем приоритет индексных полей над содержимым

- или не сериализуем лишнее

Заранее готовим код для обновления версий

- он точно пригодится

Учим десериализатор поддерживать конфликты

- а лучше бы и полиморфизм

(c) Дельгядо Филипп, 2016

Jackson

Управляется аннотациями

Работает с приватными полями

Можно фильтровать, какие атрибуты сериализовать

(c) Дельгядо Филипп, 2016

Как это выглядит в коде

```
class Operations {
  static final int SCHEMA=1;
  private long id;
  private Long userId,
  private String state,
  @JsonView(DB.Class) private Money amount,
  @JsonView(DB.Class) private List<Params> params,

  public Operations (id,userId,state) {...}
};
```


(c) Дельгядо Филипп, 2016

Jackson

Управляется аннотациями

Работает с приватными полями

Можно фильтровать, какие атрибуты сериализовать

Умеет «досериализовывать» уже созданные объекты

Умеет полимофизм

(c) Дельгядо Филипп, 2016

Плохой код в DAO

```
jdbc().query(
«select id,userId,state,data
   from operations where userId=?», uid,
(rs,i)→
{
   Operations operation;
   operation = objectMapper.readValue
         (rs.getString(4), Operations.class);
   operation.setId(rs.getLong(1));
   operation.setUserId(rs.getLong(2));
   operation.setState(rs.getString(3));
});
```


(c) Дельгядо Филипп, 2016

Приличный код

```
jdbc().query(
«select id,userId,state,data
   from operations where userId=?», uid,
(rs,i)→
{
   Operations operation;
   operation = new Operations
      (rs.getLong(1),
       rs.getLong(2),
       rs.getString(3));
   objectMapper.readerForUpdating(operation)
                              .readValue(rs.getString(4))
});
```


(c) Дельгядо Филипп, 2016

Jackson

Управляется аннотациями

Работает с приватными полями

Можно фильтровать, какие атрибуты сериализовать

Умеет «досериализовывать» уже созданные объекты

Умеет полимофизм

@JsonTypeInfo

(c) Дельгядо Филипп, 2016

Смена версий

Обычно ничего особенного делать не надо

(c) Дельгядо Филипп, 2016

Смена версий

Обычно ничего особенного делать не надо

Иногда нужно писать специальный код

- @JsonAnySetter
- @JsonSerialize/@JsonDeserialize

(c) Дельгядо Филипп, 2016

Смена версий

Обычно ничего особенного делать не надо

Иногда нужно писать специальный код

@JsonAnySetter

@JsonSerialize/@JsonDeserialize

Иногда надо хранить старые «версии» классов

```
if (schemaVersion==1) {
    OldOperations oldo = mapper.readValue(data,OldOperations.class);
    Operations op = Operations.renew(oldo);
    return op;
}
```


(c) Дельгядо Филипп, 2016

Хранение событий в очереди

В БД данные события храним как text

При получении клиентский код может сериализовать/десериализовать в json/etc

Только бизнес-логика все знает о версиях структуры

(c) Дельгядо Филипп, 2016

Бонусы

Оптимистические блокировки

(c) Дельгядо Филипп, 2016

Бонусы

Оптимистические блокировки

История изменений

(c) Дельгядо Филипп, 2016

Бонусы

Оптимистические блокировки

История изменений

Криптозащита

(c) Дельгядо Филипп, 2016

Итого

Когда имеет смысл делать велосипеды

(c) Дельгядо Филипп, 2016

Когда делаем велосипеды

Быстрее сделать, чем искать и адаптировать

(c) Дельгядо Филипп, 2016

Когда делаем велосипеды

Быстрее сделать, чем искать и адаптировать

Нужно конкретное решение и не подходят существующие

(c) Дельгядо Филипп, 2016

Когда делаем велосипеды

Быстрее сделать, чем искать и адаптировать

Нужно конкретное решение и не подходят существующие

Иной вариант увеличивает сложность системы

- не всегда хочется поднимать еще один 3d-party сервер

(c) Дельгядо Филипп, 2016

Когда делаем велосипеды

Быстрее сделать, чем искать и адаптировать

Нужно конкретное решение и не подходят существующие

Иной вариант увеличивает сложность системы

- не всегда хочется поднимать еще один 3d-party сервер

Нужно мотивировать разработчиков

(c) Дельгядо Филипп, 2016

Как делаем велосипеды

Точно знаем, что делаем и зачем

(c) Дельгядо Филипп, 2016

Как делаем велосипеды

Точно знаем, что делаем и зачем

- Что нам нужно сделать
- Когда нужно остановиться
- Что точно не делаем

(c) Дельгядо Филипп, 2016

Как делаем велосипеды

Точно знаем, что делаем и зачем

- Что нам нужно сделать
- Когда нужно остановиться
- Что точно не делаем

KISS & YAGNI

(c) Дельгядо Филипп, 2016

Как делаем велосипеды

Точно знаем, что делаем и зачем

- Что нам нужно сделать
- Когда нужно остановиться
- Что точно не делаем

KISS & YAGNI

Учет рисков

- Должно быть запасное решение

(c) Дельгядо Филипп, 2016

Как делаем велосипеды

Точно знаем, что делаем и зачем

- Что нам нужно сделать
- Когда нужно остановиться
- Что точно не делаем

KISS & YAGNI

Учет рисков

- Должно быть запасное решение

Знание

- Базовые знания, документация

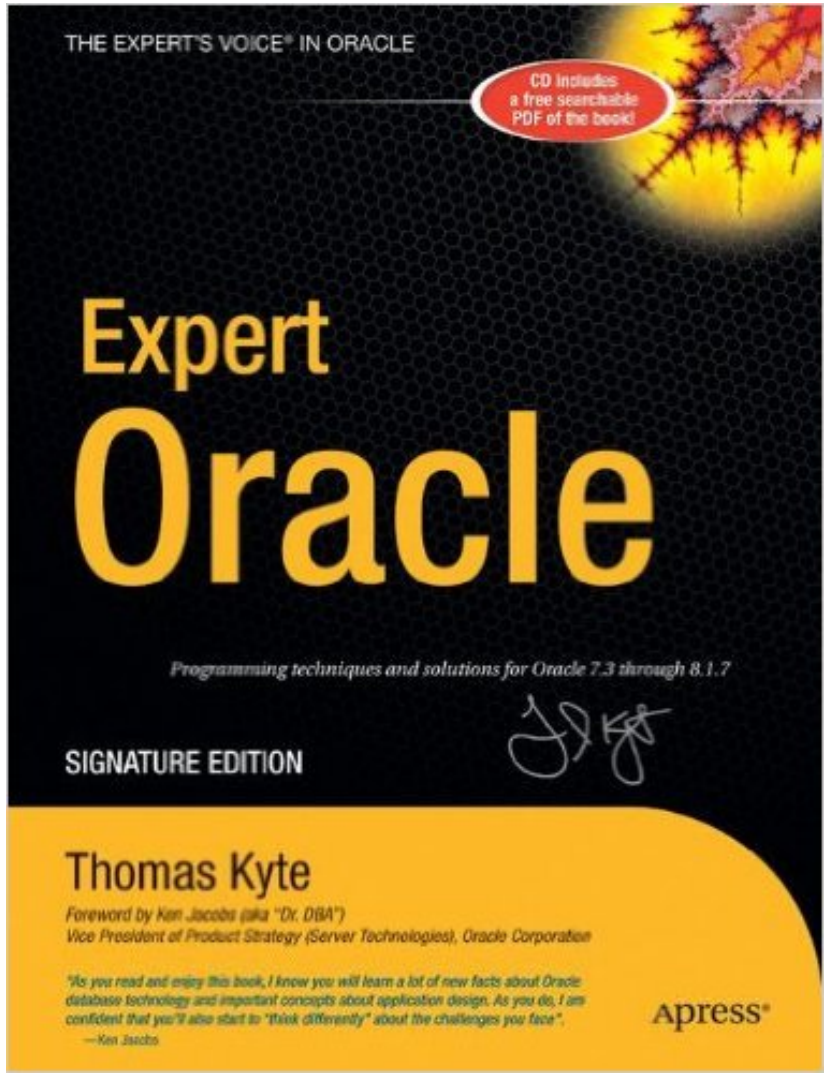
(c) Дельгядо Филипп, 2016

Литература

Том Кайт «Oracle для профессионалов»

Первые 200 страниц

http://www.amazon.in/dp/1590595254



(c) Дельгядо Филипп, 2016

Литература

Том Кайт «Oracle для профессионалов»

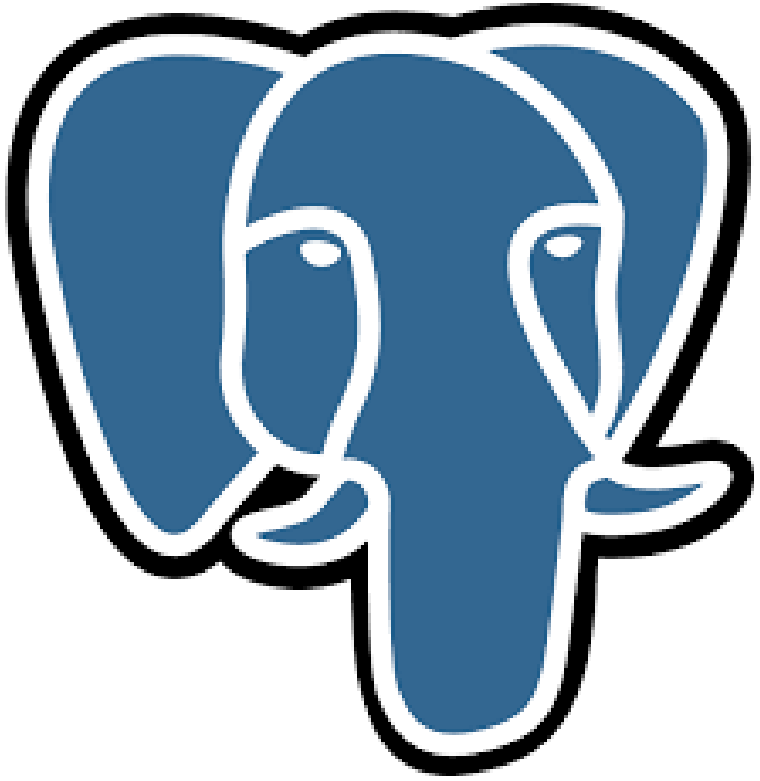
Первые 200 страниц

http://www.amazon.in/dp/1590595254

Документация по PostgreSQL

https://postgrespro.ru/docs/postgresql/9.5/

https://www.postgresql.org/docs/current/static/



(c) Дельгядо Филипп, 2016

Вопросы и контакты

Дельгядо Филипп,

dph.main@gmail.com
phd@itasystems.ru
vk.com/dphil

ИТИС Интерактивные Технологии и Системы

http://itasystems.ru/

(c) Дельгядо Филипп, 2016

Kafka?

Pros:

- Очень быстрая
- Крутая
- Есть partition

Cons:

- Сложно с отложенными событиями
- Сложно сделать redelivery
- Сложно с бизнес-транзакциями

(c) Дельгядо Филипп, 2016

Akka?

Pros:

- Модно
- Есть акторы

Cons:

- Нет гарантий доставки
- JDBC Persistance довольно наколенный
- Сложно найти разработчиков

(c) Дельгядо Филипп, 2016