Skillbox

Управление транзакциями

Андрей Гордиенков

Solution Architect

ABAX

В прошлом и в этом уроке

В прошлом уроке:

- сущность транзакций
- ACID- и BASE-гарантии
- распределённые транзакции

В этом уроке:

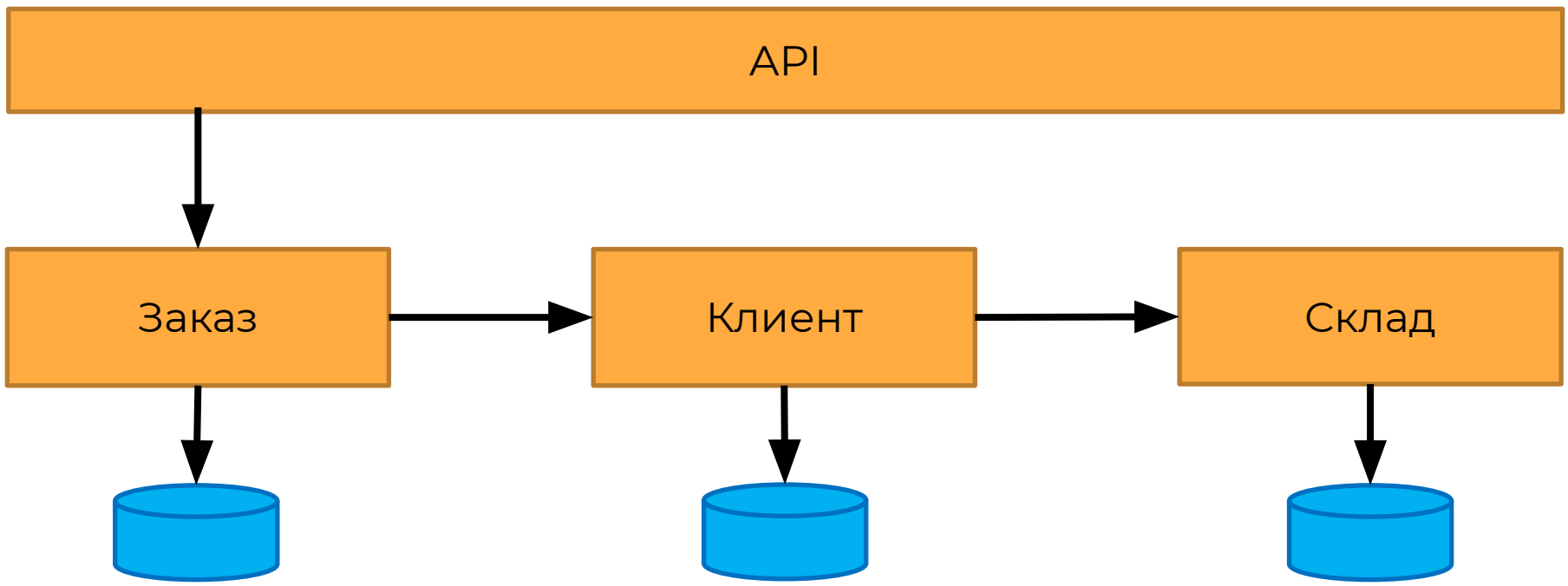
- шаблон «Сага» для распределённых транзакций
- ограничения и применимость шаблона

Skillbox

Шаблон «Сага»

Андрей Гордиенков
Solution Architect
ABAX

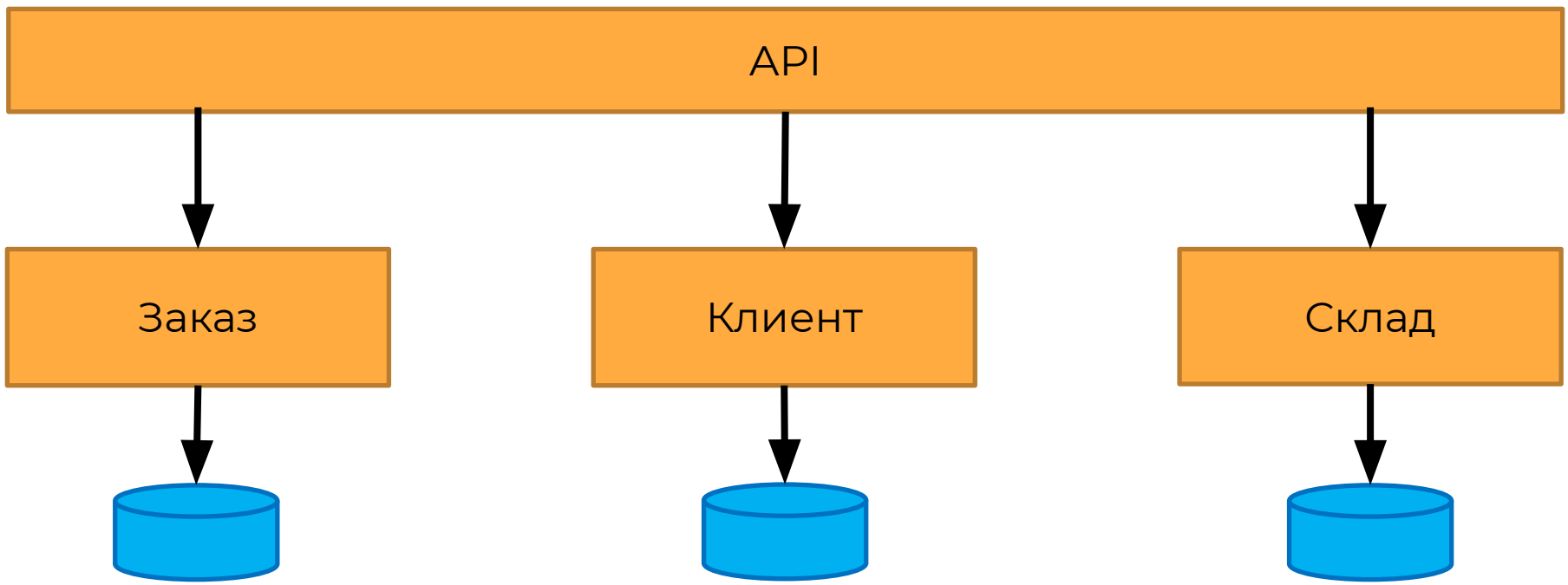
Проблема



2 Phase Commit нельзя применить из-за особенностей реализации.

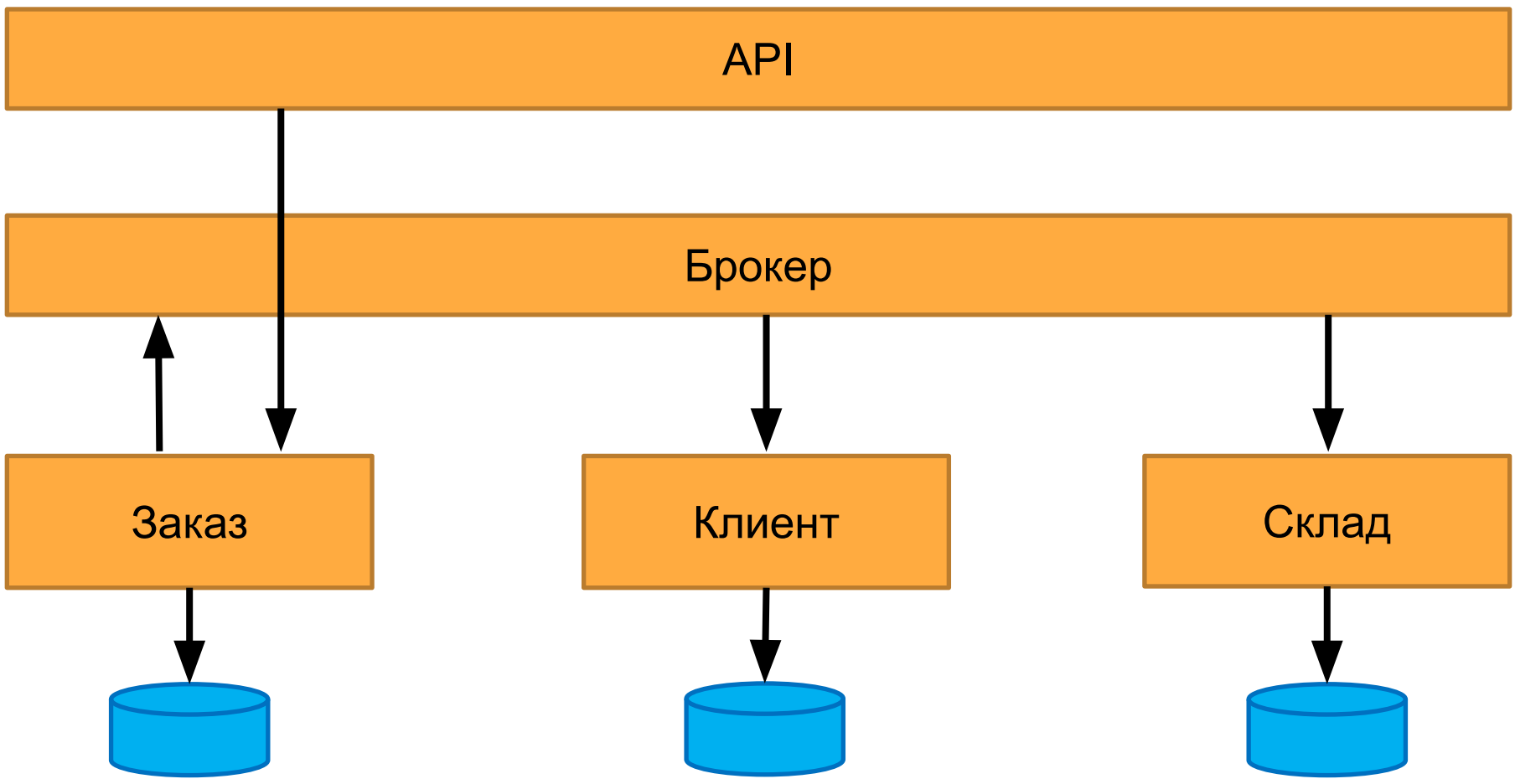
- Время для согласования: зависимые сервисы могут требовать дополнительные данные
- Возможное отсутствие хранилищ для сервисов

Проблема



Минус: слишком много ответственности в API-слое.

Проблема



Брокер создаёт много рисков.

Skillbox

«Сага»

Последовательность локальных транзакций с компенсационным механизмом. Каждый следующий шаг может продолжаться только после успешности предыдущего.

Первое упоминание — в 1987 году в публикации «Долгоживущие транзакции» Гектора Гарсия-Молины (Hector Garcia-Molina) и Кеннета Салема (Kenneth Salem).

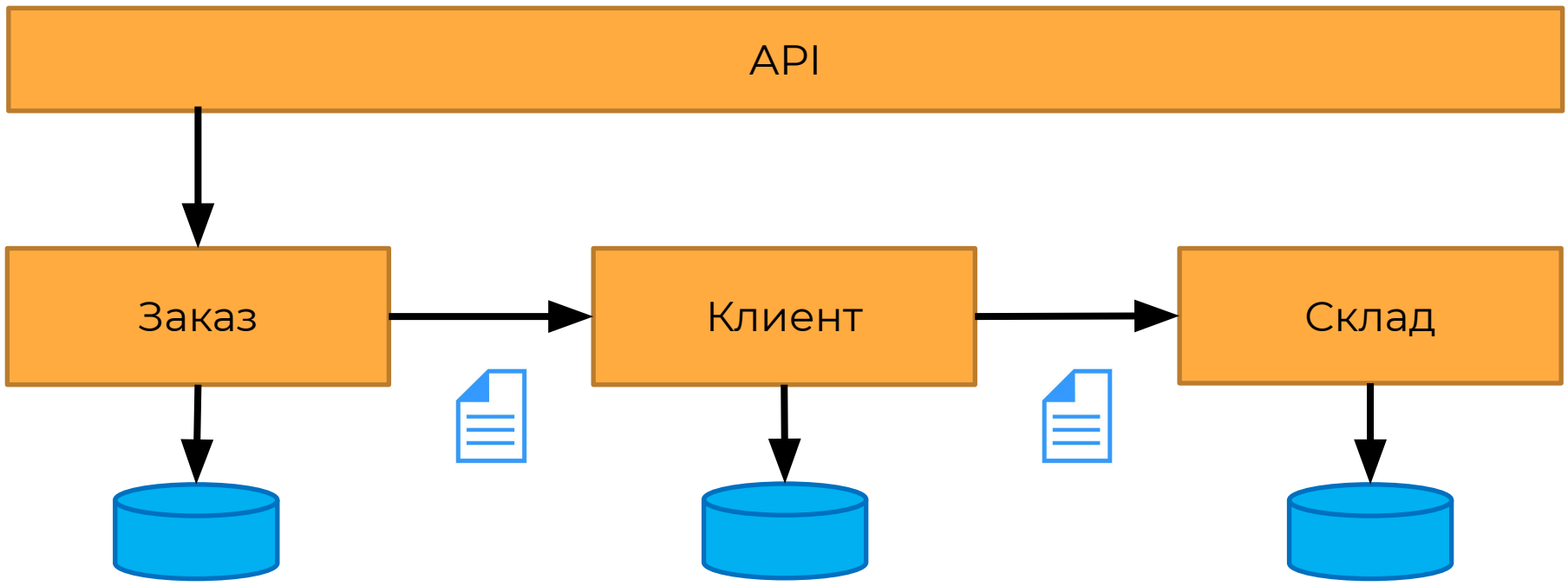
По своей сути «Сага» — это конечный автомат.

Применимость

Обычно используется, когда на каждом шаге требуется принятие решения человеком, то есть наиболее очевидное применение — документооборот с визированием решений.

Основная особенность: процесс «засыпает» после каждого шага, пока не появится какое-либо решение.

Проблема



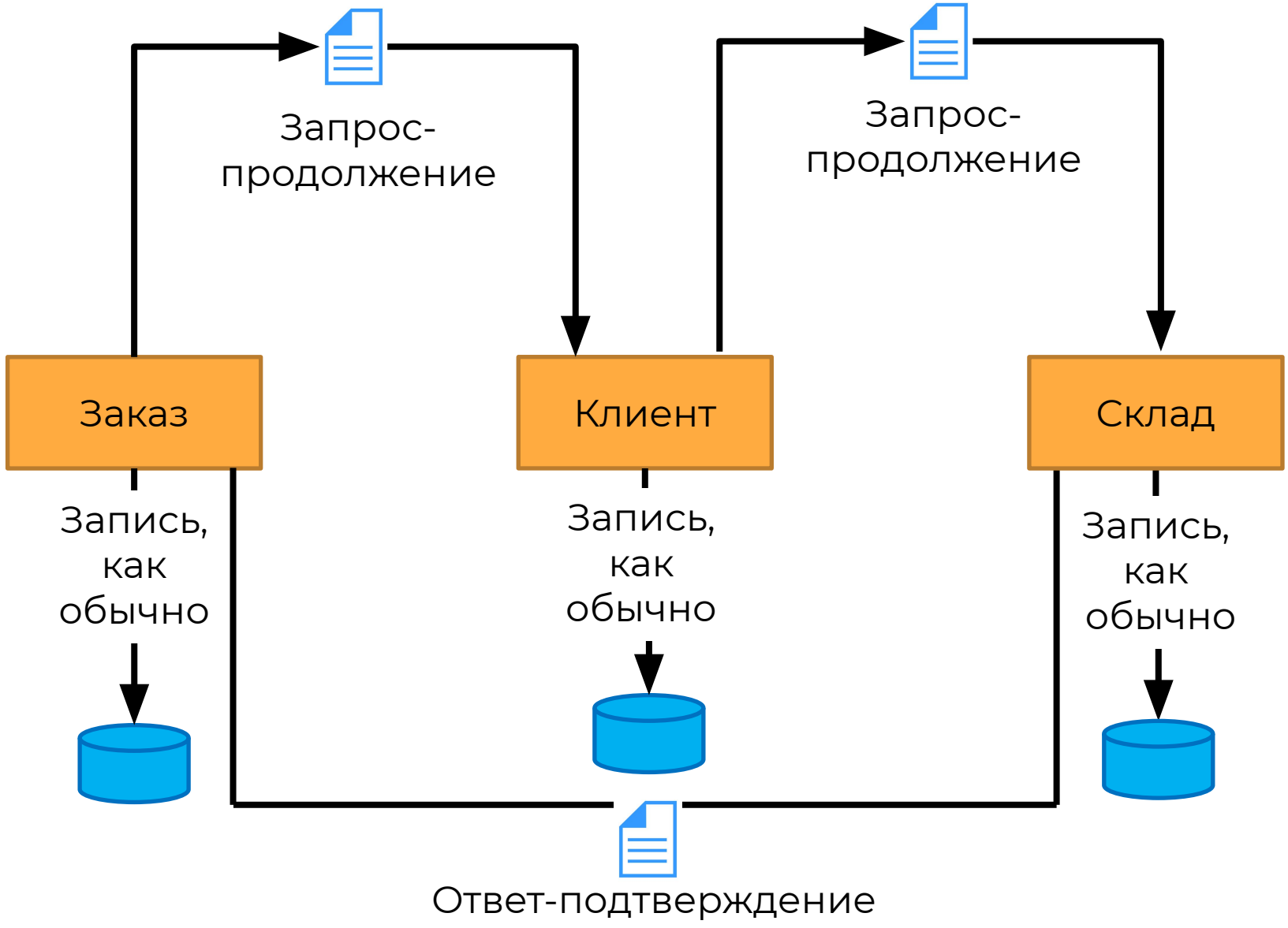
Специальное сообщение (команда) с данными для транзакций. Последний участник обработки подтверждает успешность инициатору «Саги».

Реализация

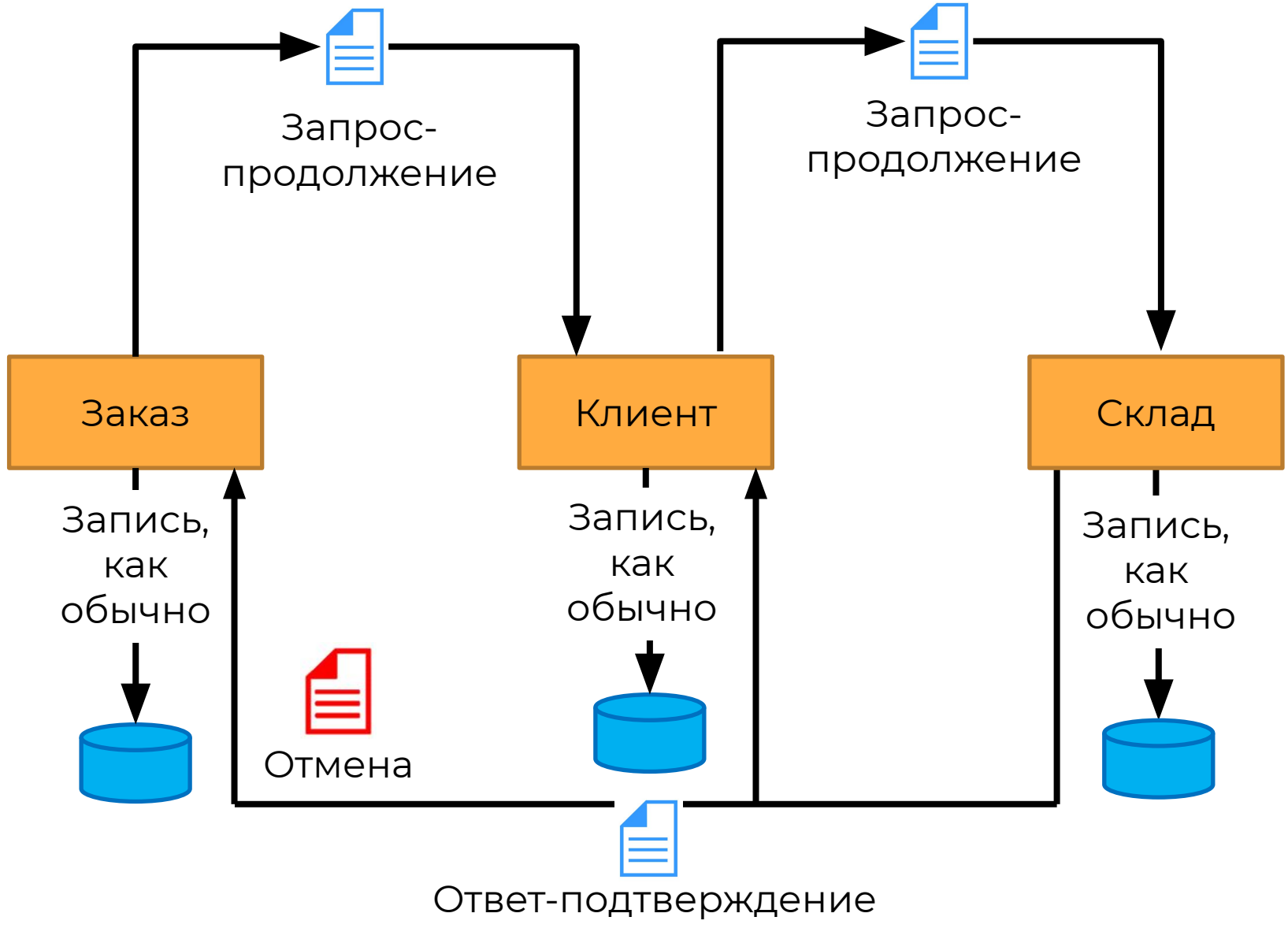
Варианты:

- Хореография — сервисы сами знают, куда дальше послать запрос.
- Оркестрация — существует брокер сообщений, который организует коммуникацию между сервисами.

Хореография



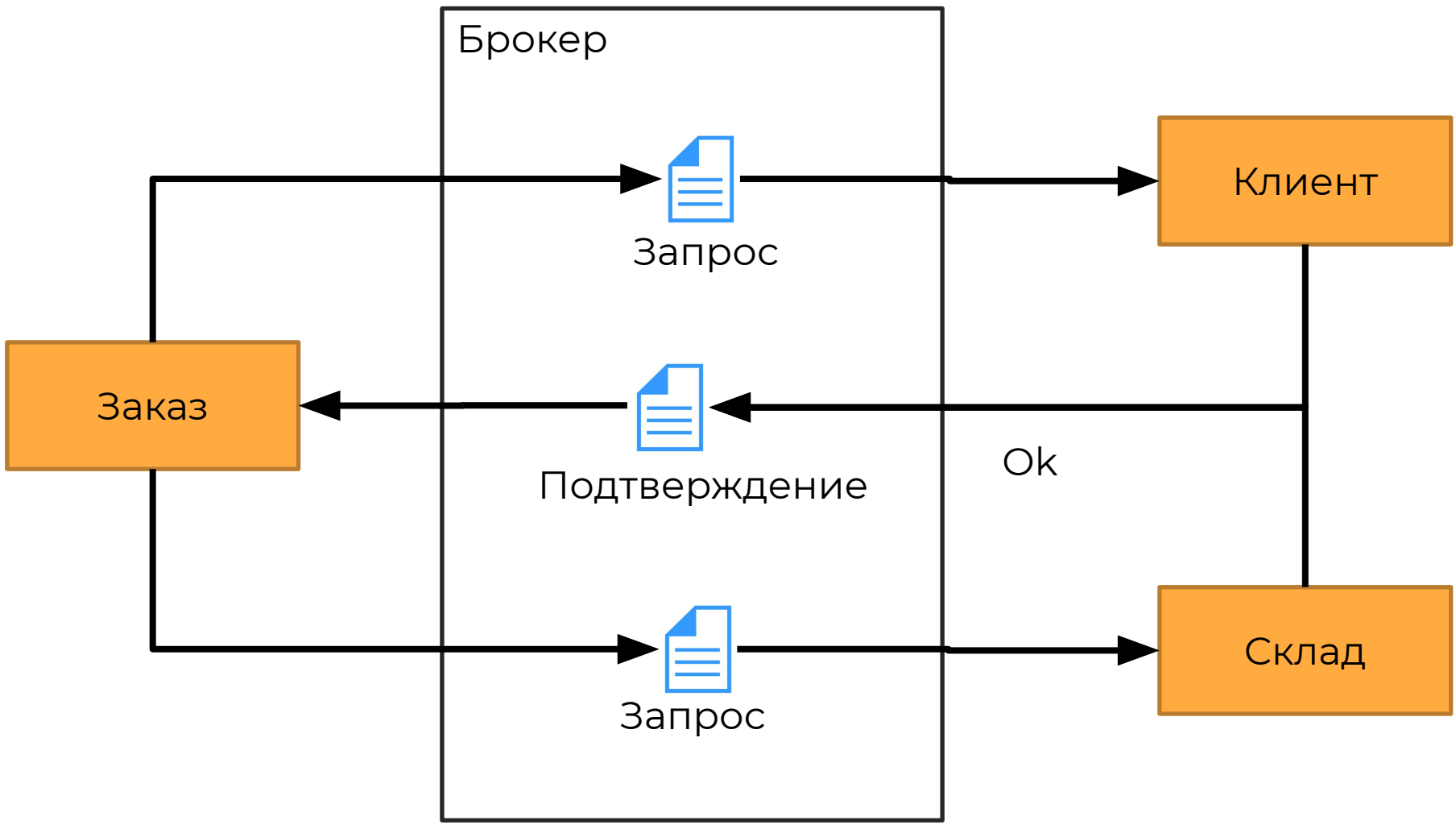
Хореография



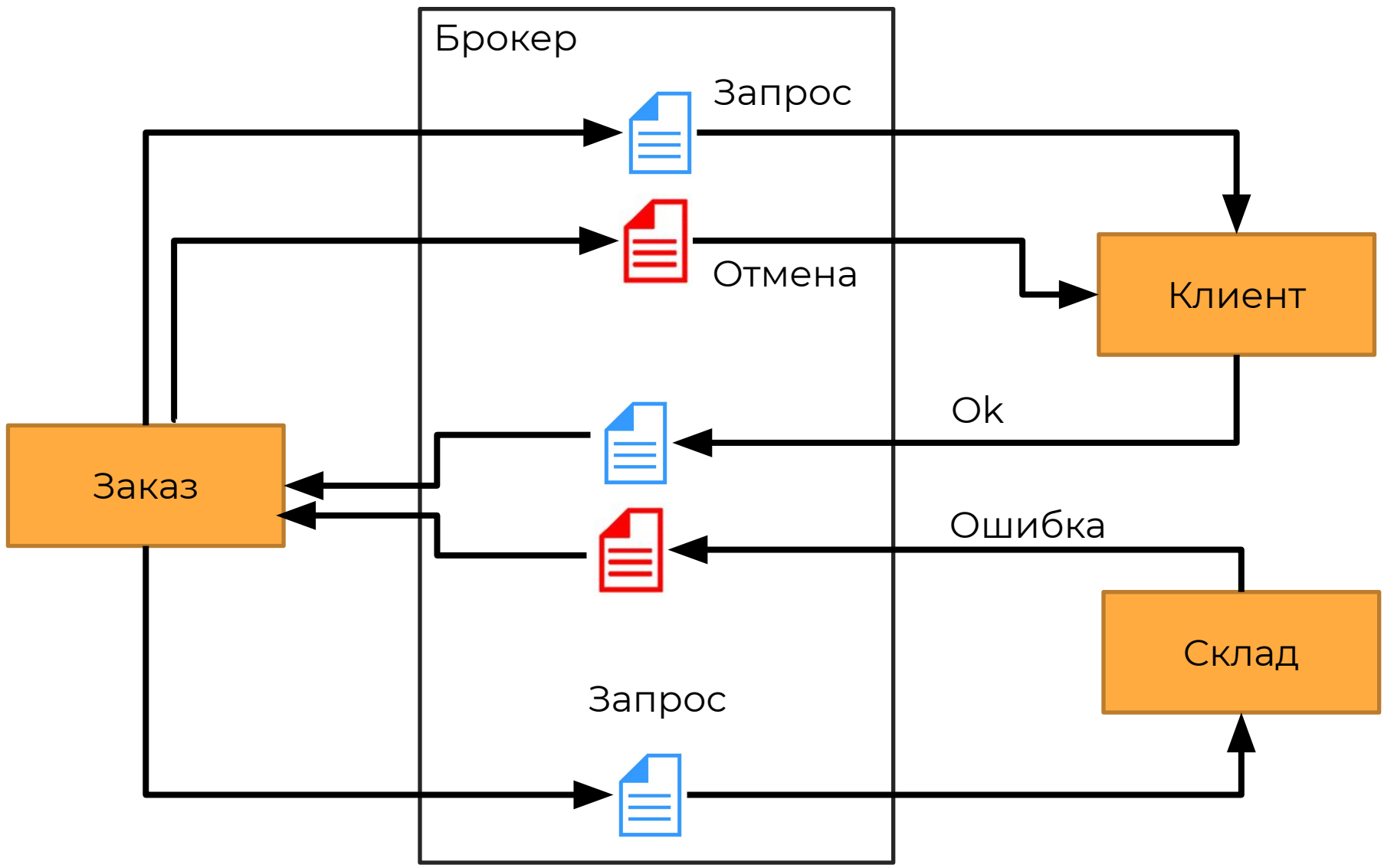
Отмена операции:

- должен существовать механизм оповещения всех об ошибке
- сервисы должны знать предыдущие шаги

Оркестрация



Оркестрация



Отмена операции:

- брокер должен знать, что отменить
- инициатор посылает команду отмены

Советы

- Избегайте циклических зависимостей между сервисами.
- Не позволяйте зависимым сервисам в транзакции вызывать брокера напрямую.
- Всегда создавайте уникальный ID транзакции для облегчения отслеживания статуса и зависимостей.
- Используйте асинхронные операции.
- Сохраняйте запрос для повторных попыток.
- Запрос должен содержать как можно больше данных для успешного выполнения.

Требования

1. Каждая операция должна иметь вариант для отмены изменений.
2. Каждая транзакция должна иметь глобальный идентификатор.
3. С самого начала дизайн-система и модель данных должны поддерживать компенсационные команды (отмены).
4. Сервис должен атомарно применять изменения и уведомлять об изменениях.
5. Команды со всеми параметрами должны сохраняться для возможности повторного выполнения и реализации отмены.

Сложности

- Сложность координации долгоживущих транзакций, если они состоят более чем из 3-4 шагов.
- Сложность отслеживания зависимостей, как у любой системы, построенной на событиях.
- Возможность циклических зависимостей между сервисами и событиями.
- Сложность тестирования.
- При любом подходе надо гарантировать доставку сообщений об успешности либо отмене операции.
- Нужны дополнительные усилия для решения проблемы чтения «грязных» данных.

Альтернатива

- Принять более слабые модели согласованности данных.
- Использовать новые компенсационные команды, а не отменять старые — например, как это сделано в подходе Event-Sourcing.

Самое важное

Избегайте распределённых транзакций. Использовать, только когда все остальные варианты не подходят.

«Сага» проста и привлекательна только как концепт, на практике всё выходит очень сложно и неочевидно.

Выводы

- Распределённые транзакции сложны в реализации и использовании.
- Шаблон «Сага» не решает проблем распределённых транзакций.

Что дальше?

- Уровни согласованности данных в базах данных и почему важно их понимать.
- Согласованность данных и производительность приложения.

Skillbox

Спасибо за внимание!