Skillbox

Работа с событиями. Command & Query Responsibility Segregation (CQRS)

Андрей Гордиенков

Solution Architect

ABAX

В прошлом уроке

- Event Sourcing — концепция, применение
- Особенности реализации

В этом уроке

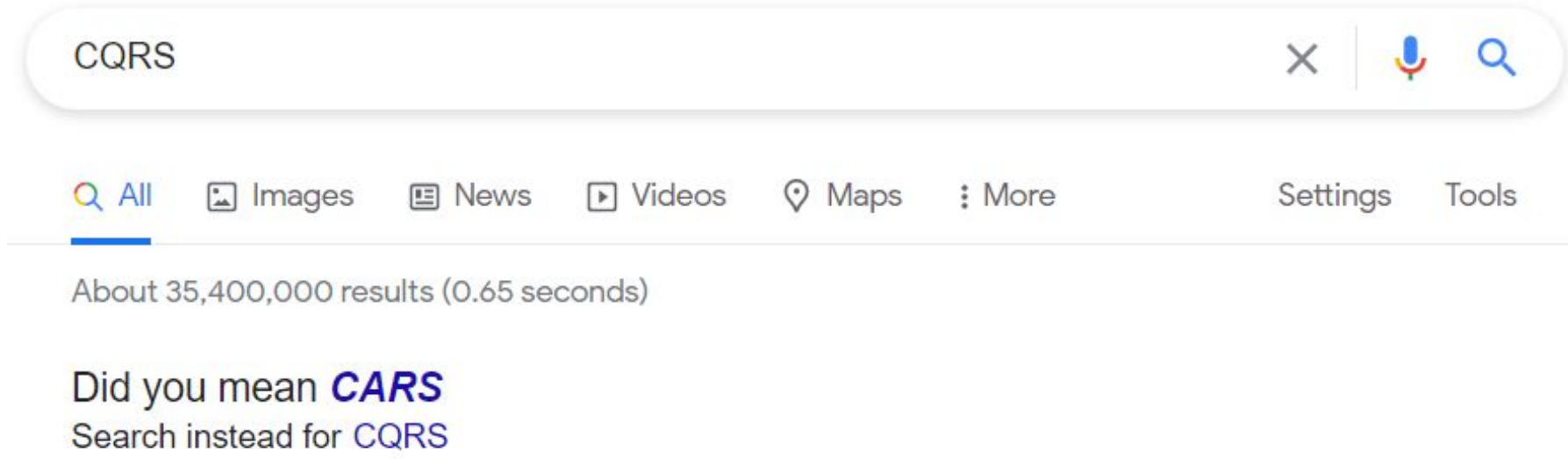
- Что такое CQRS, какую проблему решает?
- Связь с Event Sourcing

Немного истории

1985 год, Бертран Мейер (Bertran Meyer) описал концепцию Command-Query Separation (CQS).

- Команды делают изменения и не возвращают данные
- Запросы только возвращают данные

~ 2010 год, Грег Янг (Greg Young) предложил и описал CQRS-концепт на основе CQS, который очень быстро стал популярен.



Skillbox

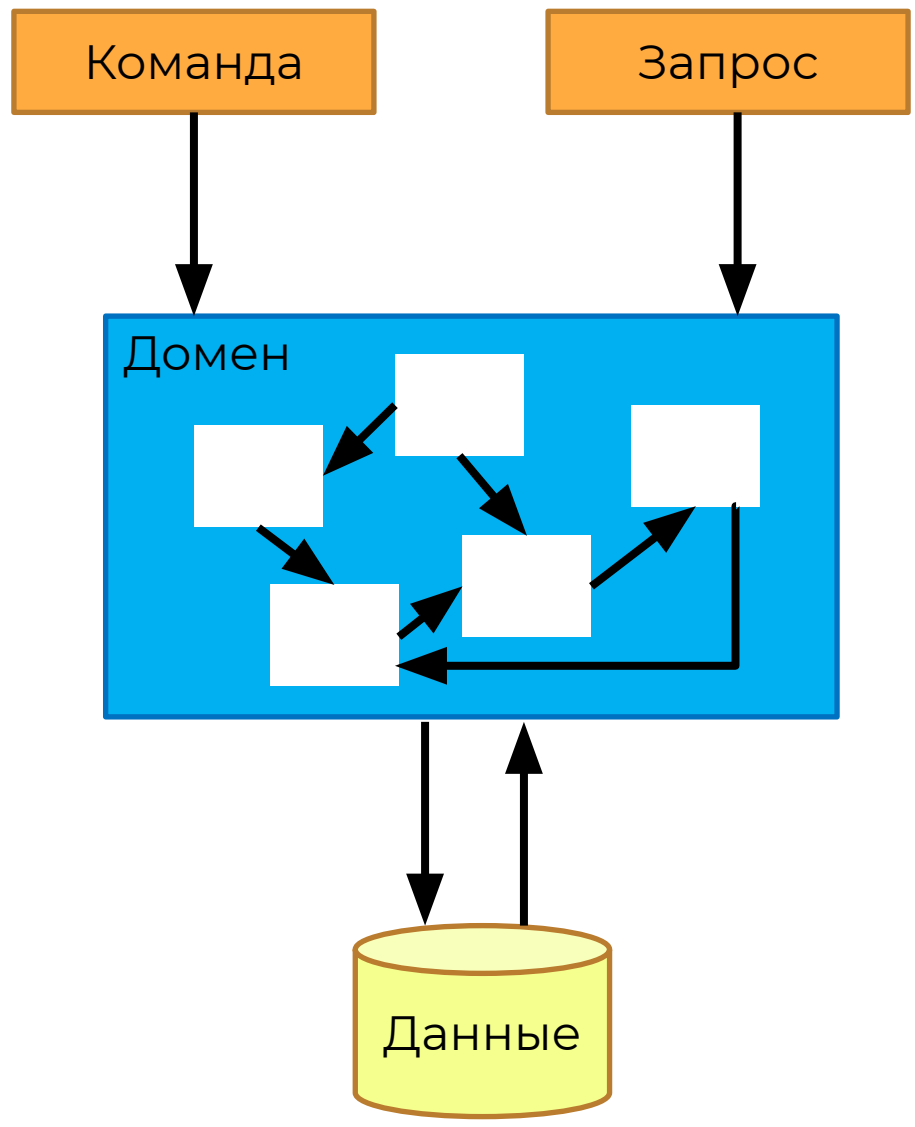
CQS vs CQRS

CQS: команды и запросы обрабатываются одним объектом.

CQRS: команды и запросы обрабатываются разными объектами.

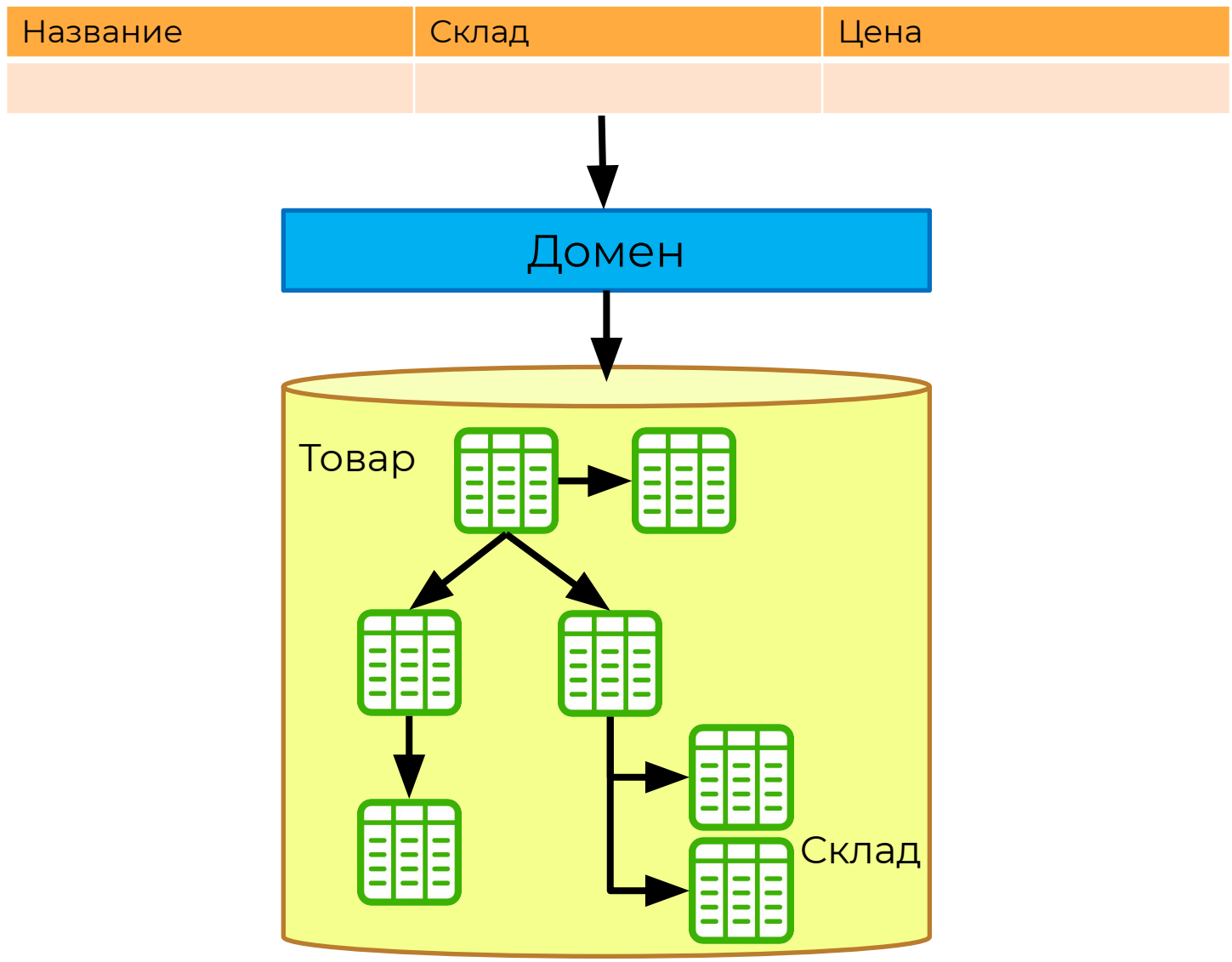
Какую проблему решаем?

Получение данных идёт через все сложные правила и связи доменной логики.



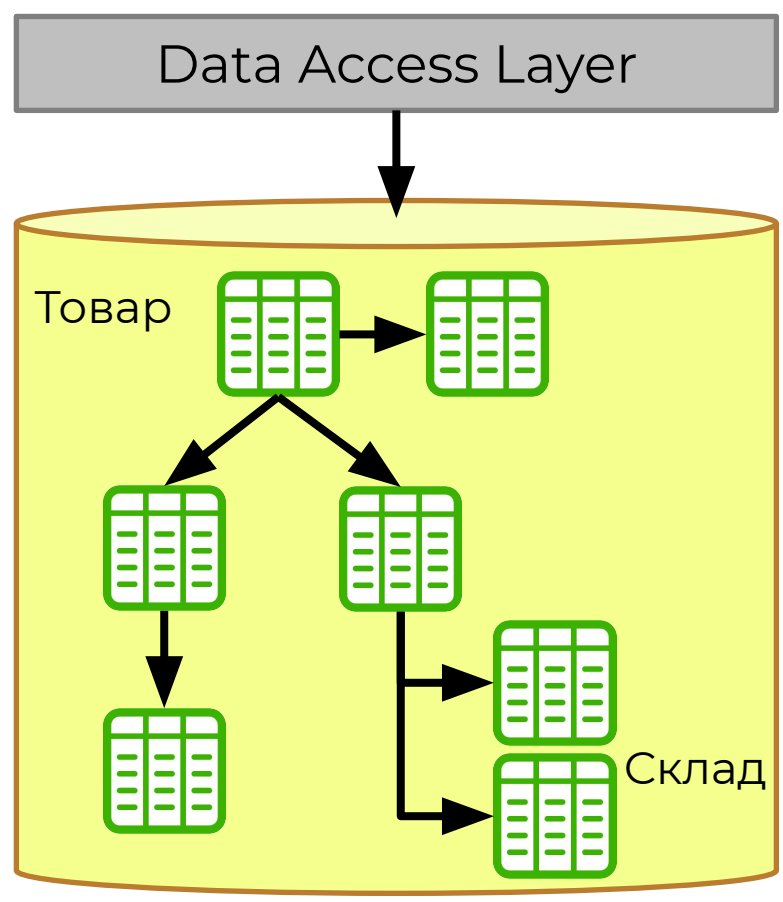
Какую проблему решаем?

Необходимо запросить и реконструировать значительно больший объём данных, чем покажем.



Object Relational Mapping

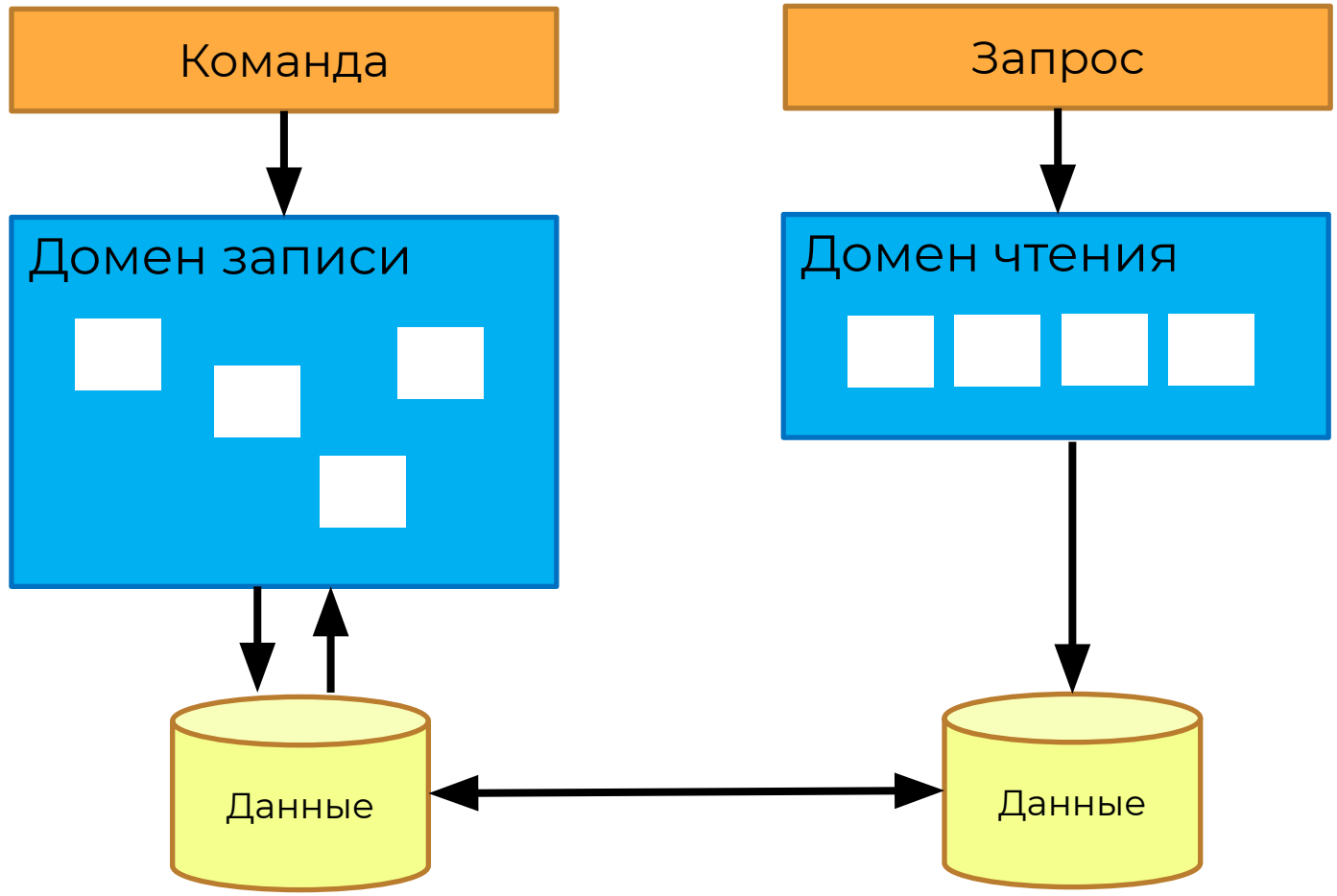
- Огромное количество служебного кода, который перекладывает данные из одного представления в другое
- Ручная оптимизация ORM для получения нужной проекции данных



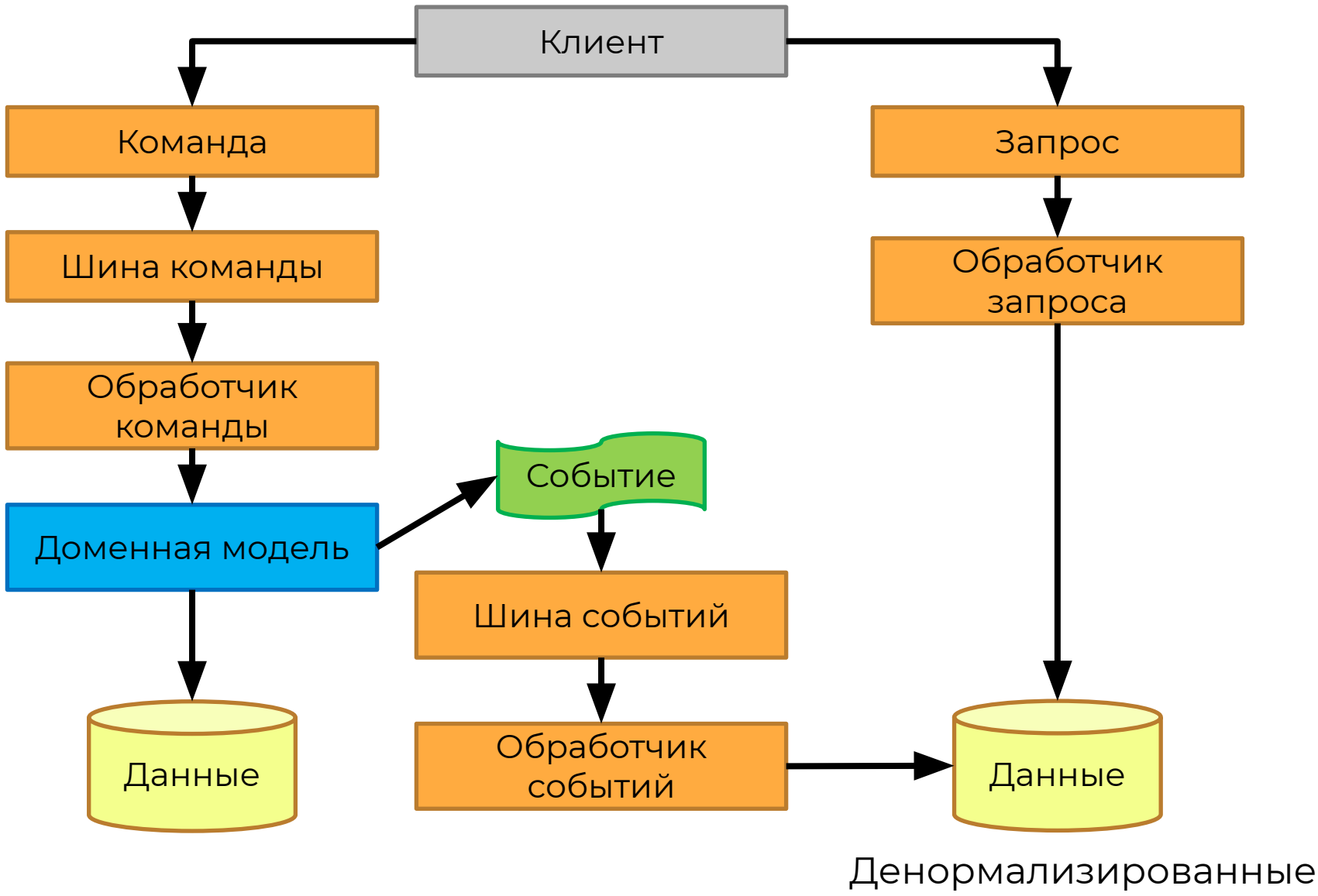
Skillbox

Разделение команд и запросов

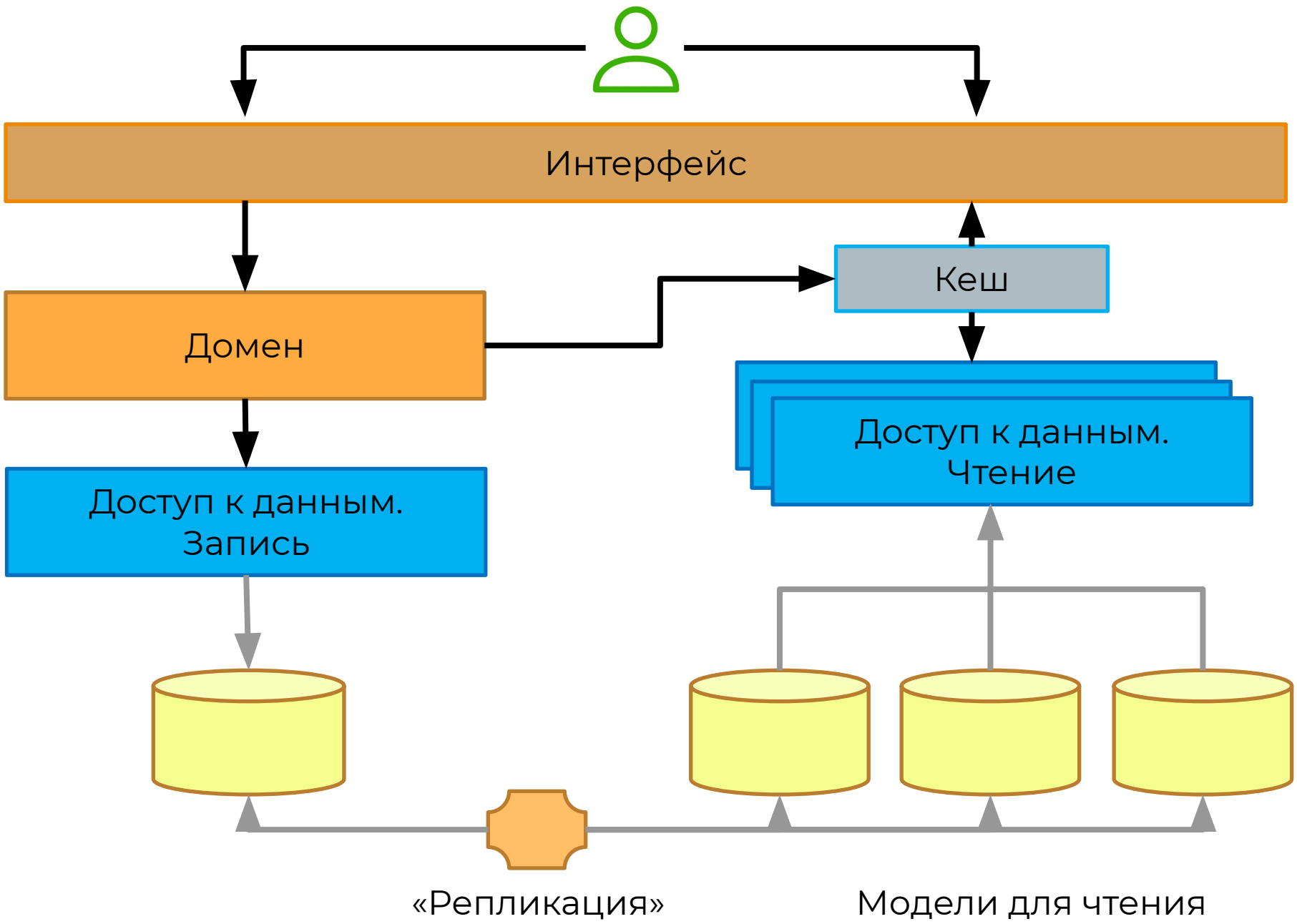
Получение данных идёт через все сложные правила и связи доменной логики.



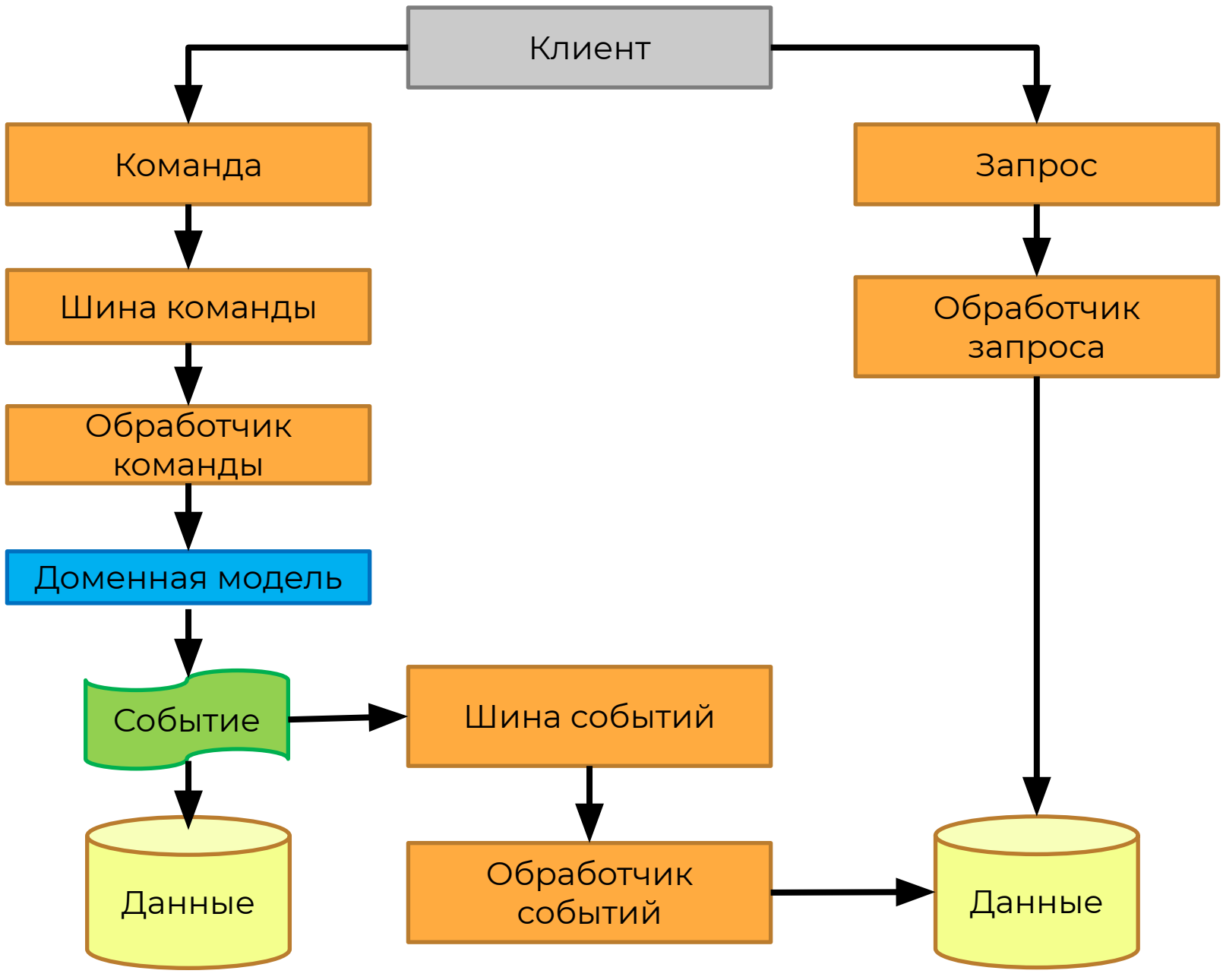
Детальное представление



Event Sourcing



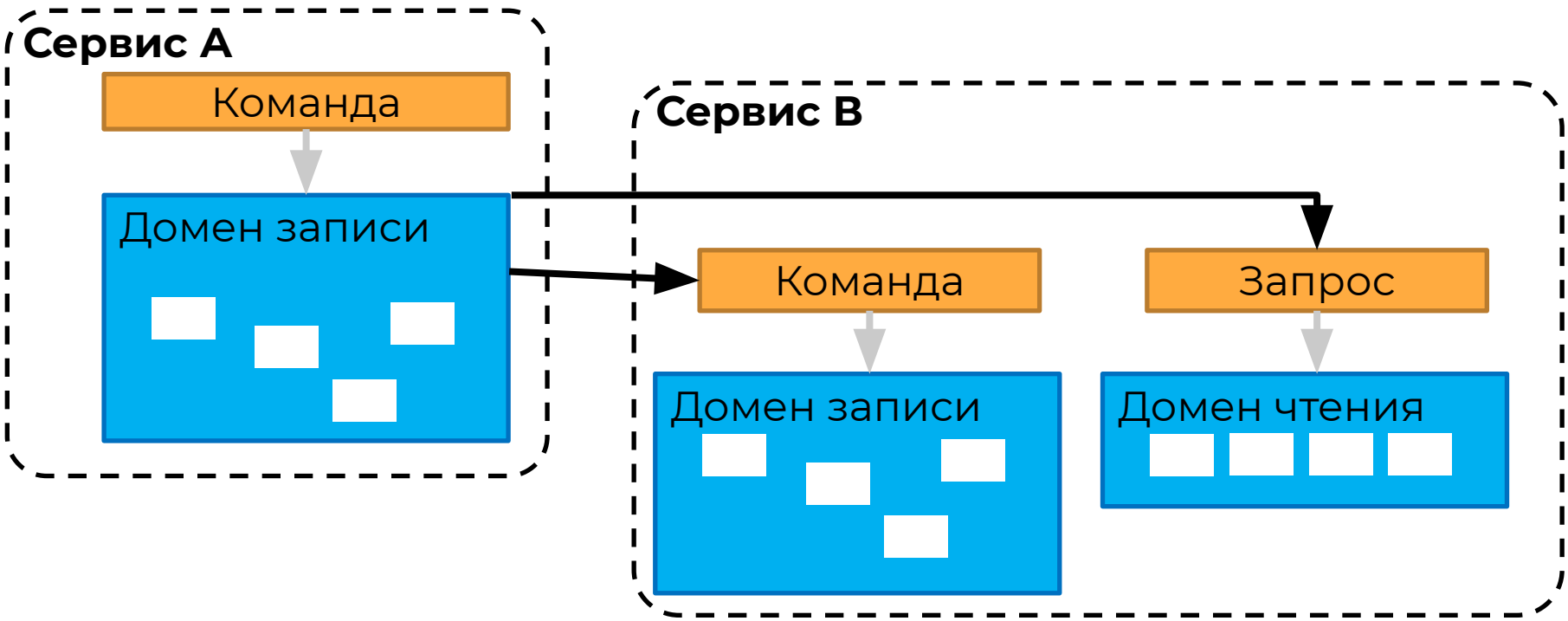
CQRS + Event Sourcing



Нюансы

Для расчётов внутренних данных, необходимых для обработки команды, не стоит использовать новые команды и сообщения. Это только усложнит ваш код.

События должны информировать о значимых для бизнеса вещах и сообщать другим сервисам, что что-то произошло.

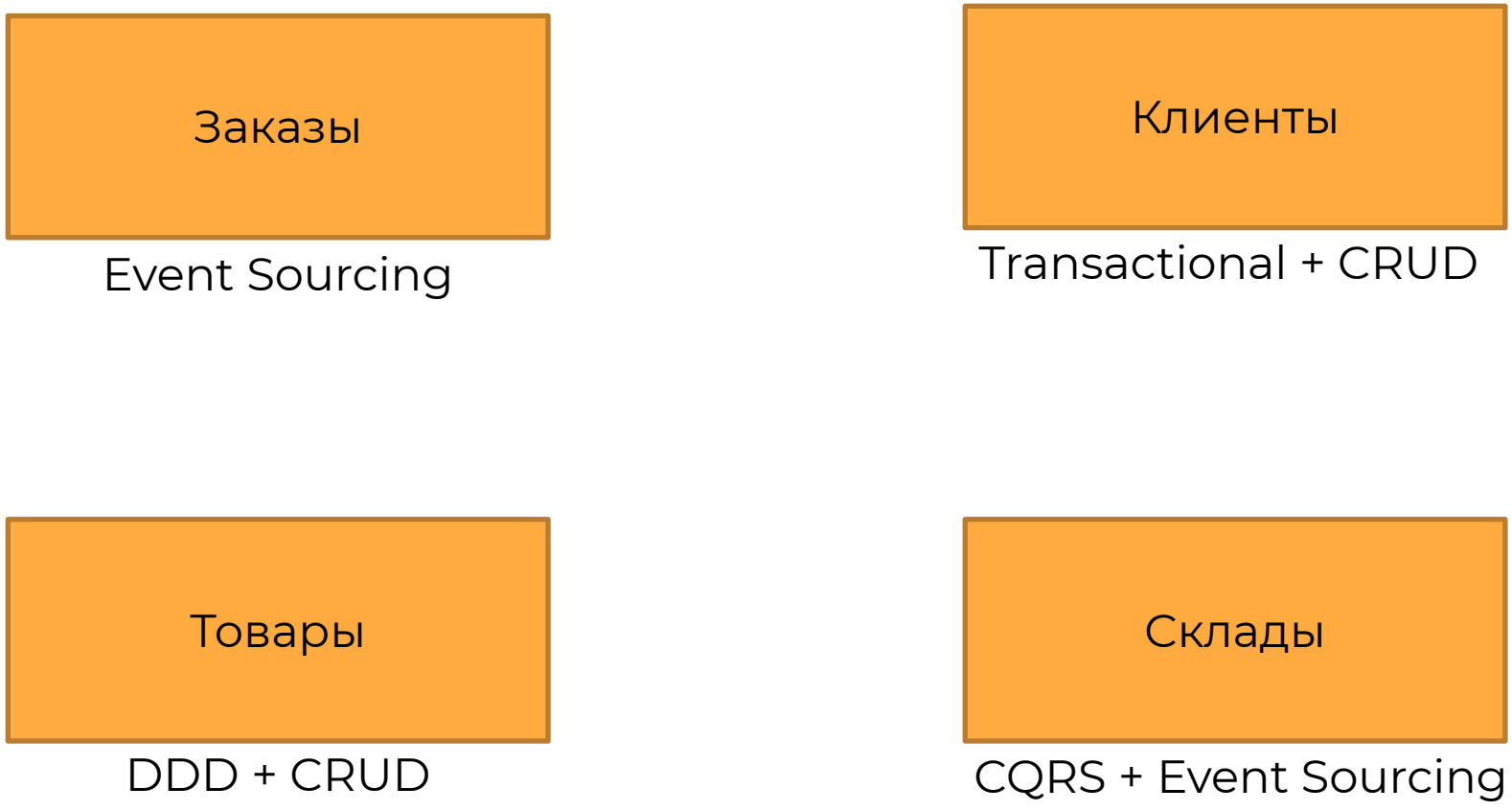


Применимость

- Сервисы (системы), где происходит работа с массивом данных, а не с единичными записями
- Система не обязана реализовывать Domain-Driven Design или строиться на событиях
- Чтение данных преобладает над записью, тогда можно получить серьёзный прирост производительности
- Для средних и крупных систем, где требуются скорость и гибкость

Памятка

Каждый сервис может реализовывать свой архитектурный стиль.
Применять CQRS для всей системы часто нецелесообразно.



Технические сложности

- Синхронизация данных между базой записи и базами чтения.
  Вы должны понимать и принимать, что согласованность данных определяется как Eventually Consistent
- Требуется больше места для хранения данных
- Проблема «чтения своих данных» сохраняется
- Поддержка версионности сообщений

Выводы

- CQRS показывает лучшие результаты в связке с Event Sourcing и Domain-Driven Design
- Подходит для средних и крупных приложений со сложной доменной логикой, где требуется скорость работы и где согласованность данных некритична
- Упрощает работу с чтением данных

Skillbox

Что дальше?

Разбор вопросов применения, заблуждений и реализации архитектур: Event Sourcing, CQRS и Event-Driven.

Skillbox

Спасибо за внимание!