EXIGEN services

CQRS - инновационное решение проблем современных Enterprise приложений.

Андрей Ломакин (lomakin.andrey@gmail.com)
Артем Оробец (enisher@gmail.com)

EXIGEN services

Содержание

1. Почему CQRS ?
   a) История возникновения современной архитектуры
   b) Проблемы CRUD
   c) CQRS как решение проблем
2. Архитектура CQRS приложений
3. Реализация CQRS на основе Axon framework
4. Event Store от Exigen Services
5. CQRS приложения разрабатываемые в Exigen Services

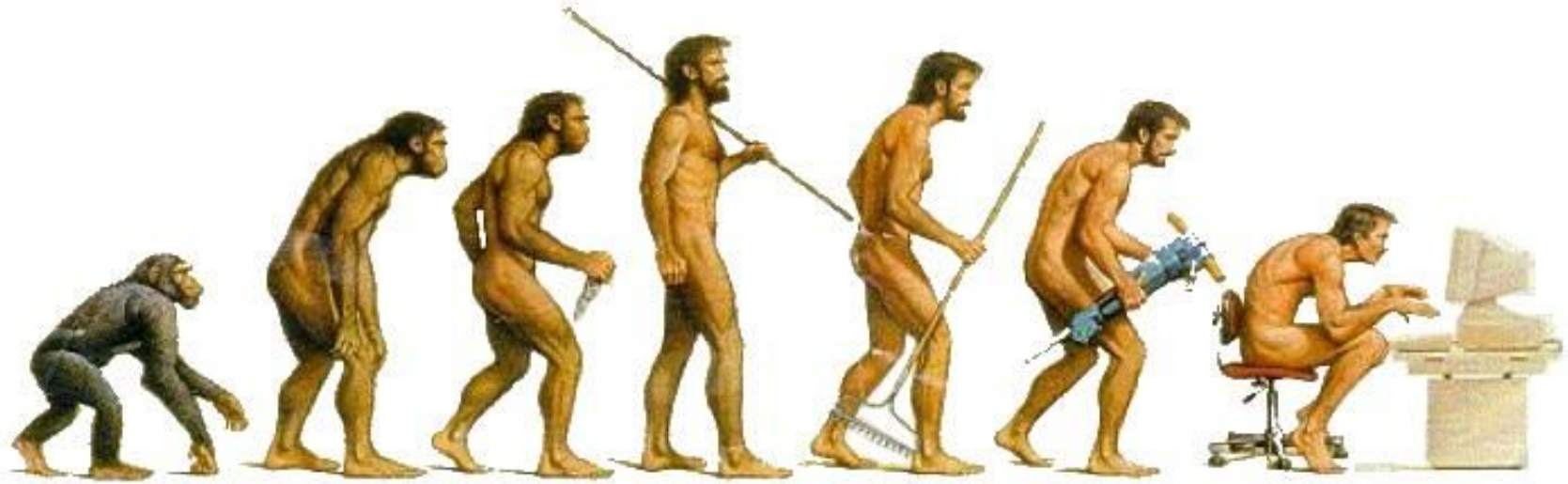
Почему CQRS ?

I. Почему CQRS ?

EXIGEN services

С чего всё началось

Поговорим про эволюцию разработки Enterprise приложений



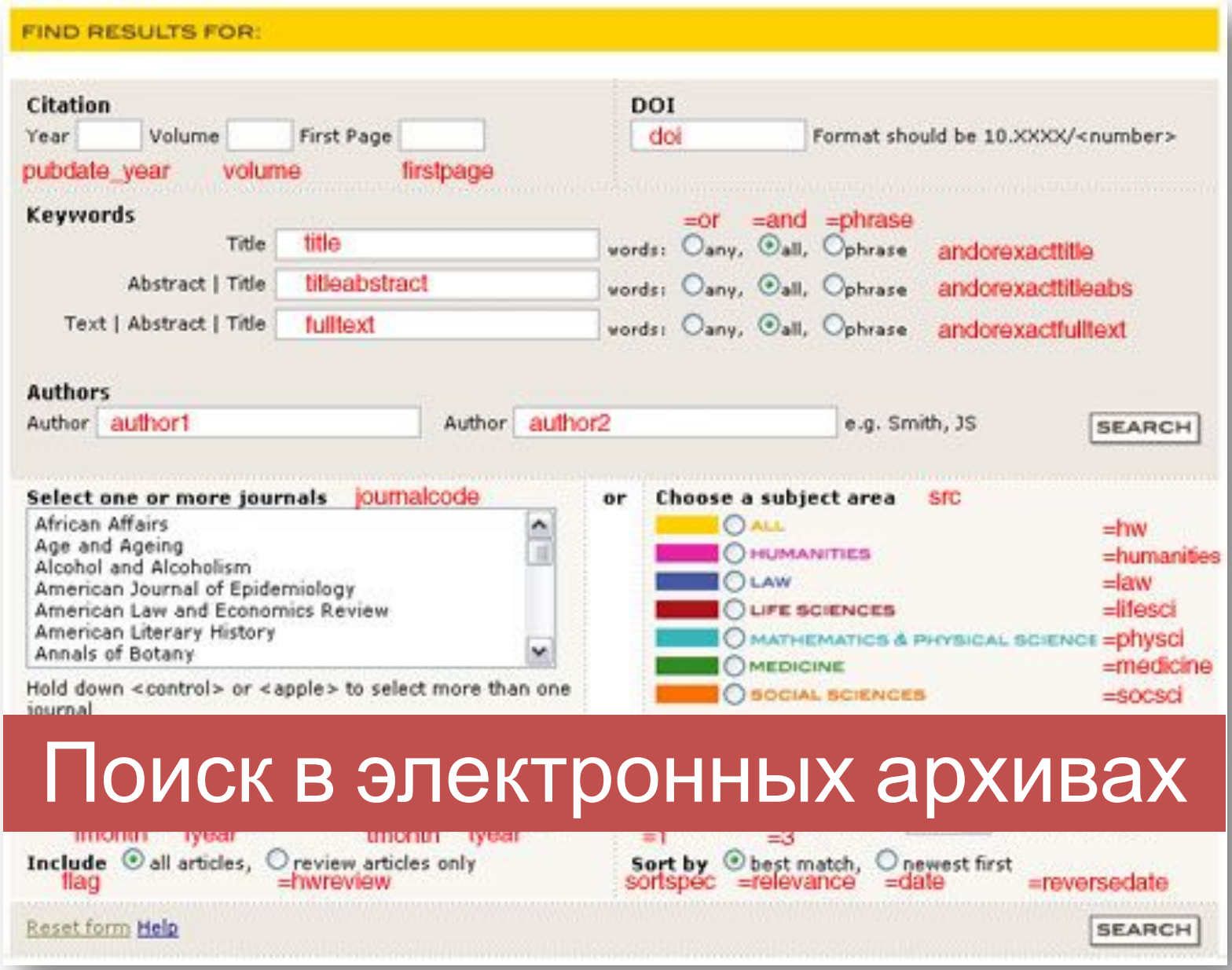
Эволюция представления документов

Эволюция представления документов

Эра электронных архивов

EXIGEN services

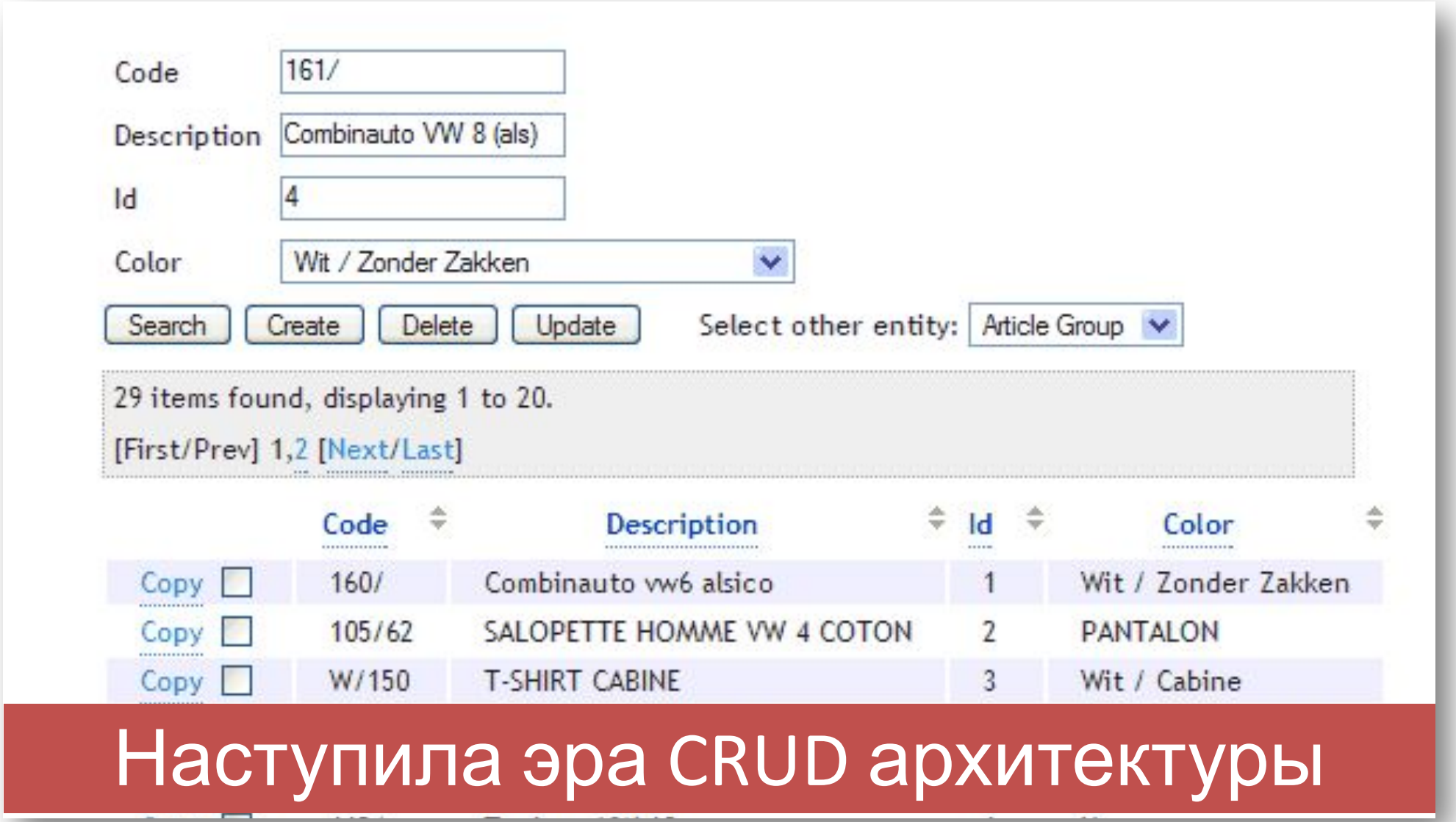
Эра электронных архивов



Поиск в электронных архивах

EXIGEN services

Эра электронных архивов

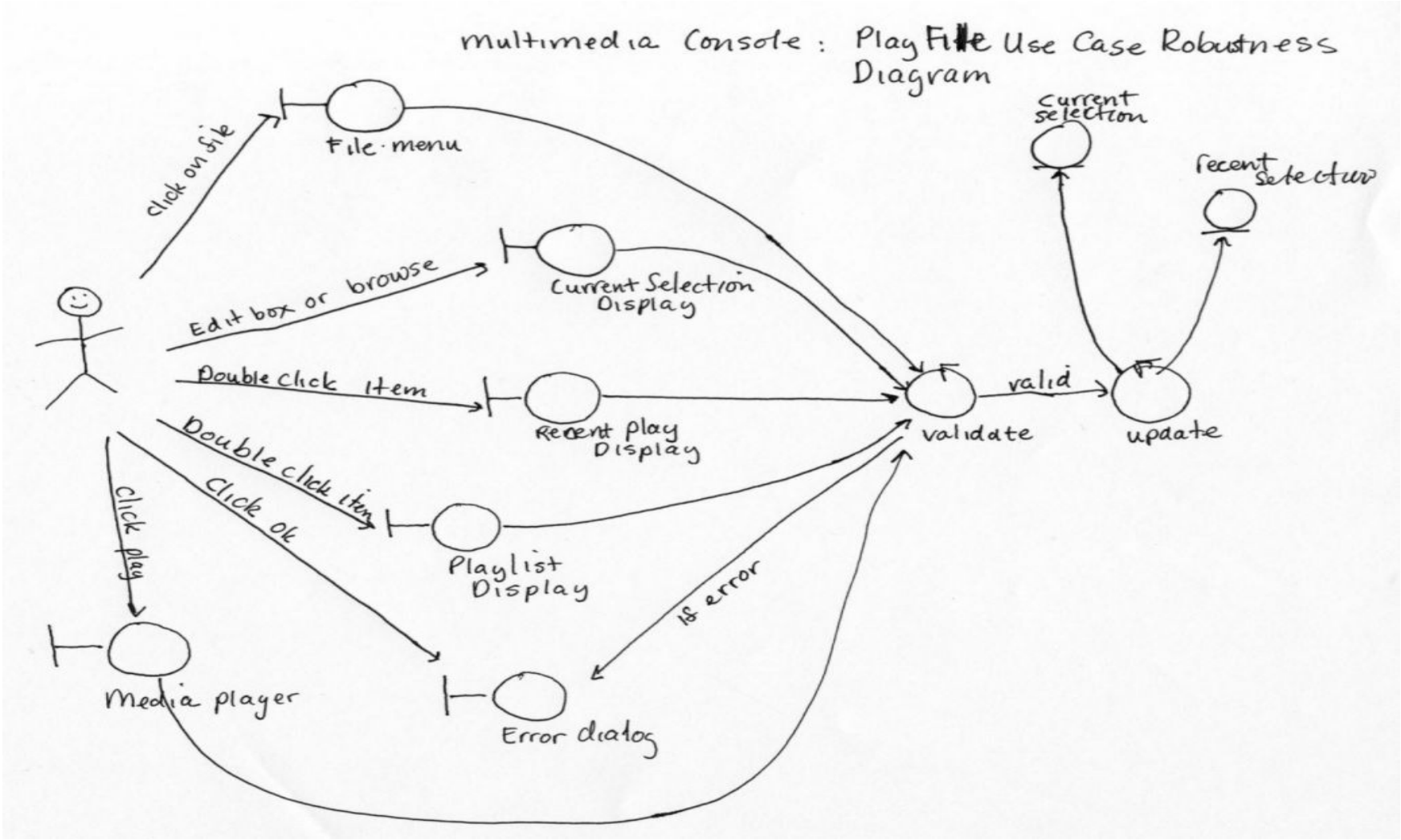


Наступила эра CRUD архитектуры

EXIGEN services

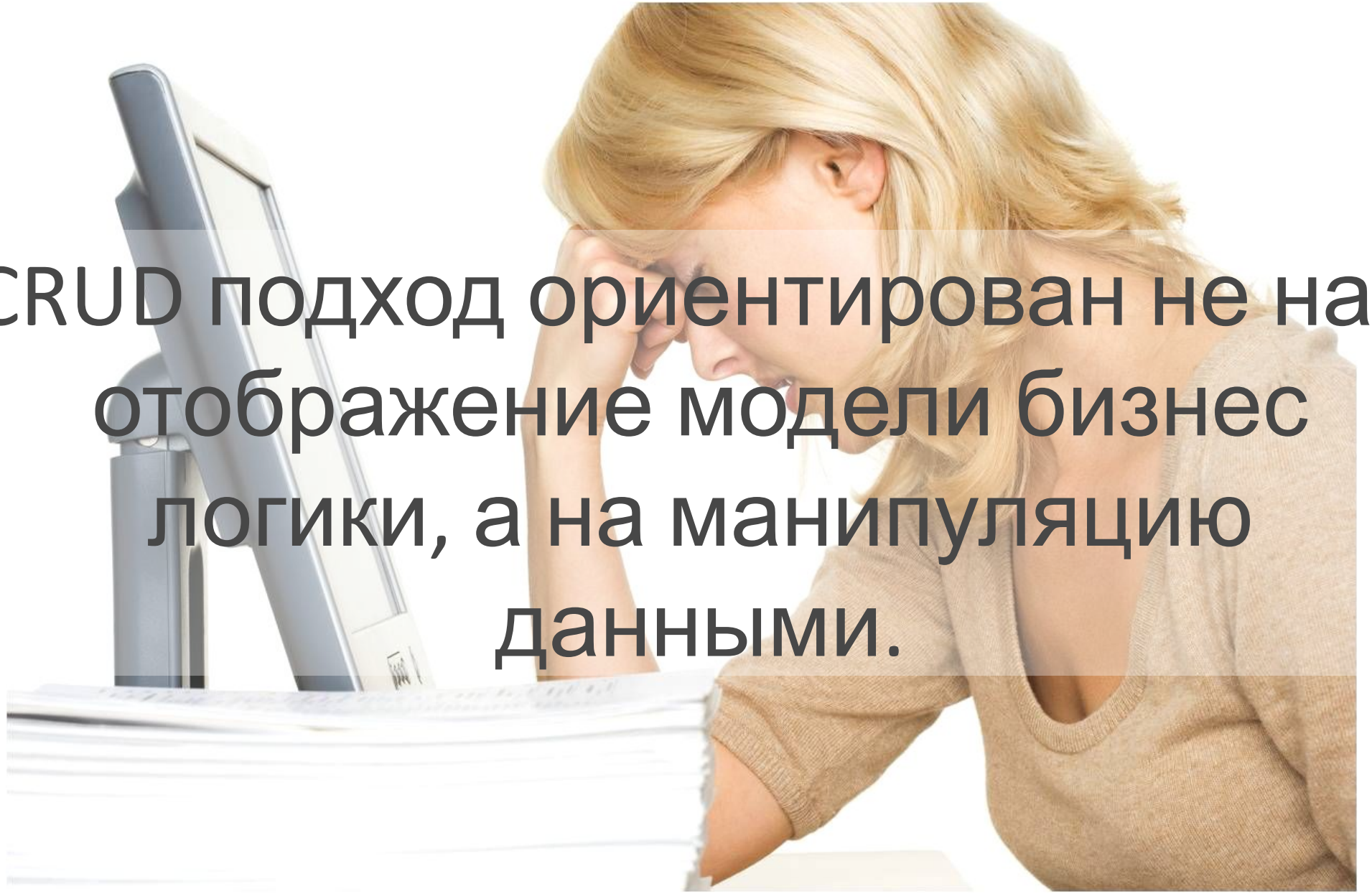
Эволюция обработки данных

Эра бизнес процессов



EXIGEN services

Однако…



CRUD подход ориентирован не на отображение модели бизнес логики, а на манипуляцию данными.

Основные проблемы CRUD

II. Основные проблемы CRUD

EXIGEN services

Проблема 1. Использование JavaBean

....

JavaBean – *“Reusable software components that can be manipulated visually in a builder tool”*.

EXIGEN services

… приводит к …

1. Нарушению инкапсуляции бизнес объектов.
2. Ухудшению читаемости кода
3. Трудности поддержки
4. Вся бизнес логика переносится в методы сервисов.

Проблема 2. Оптимизация производительности и её последствия.

EXIGEN services

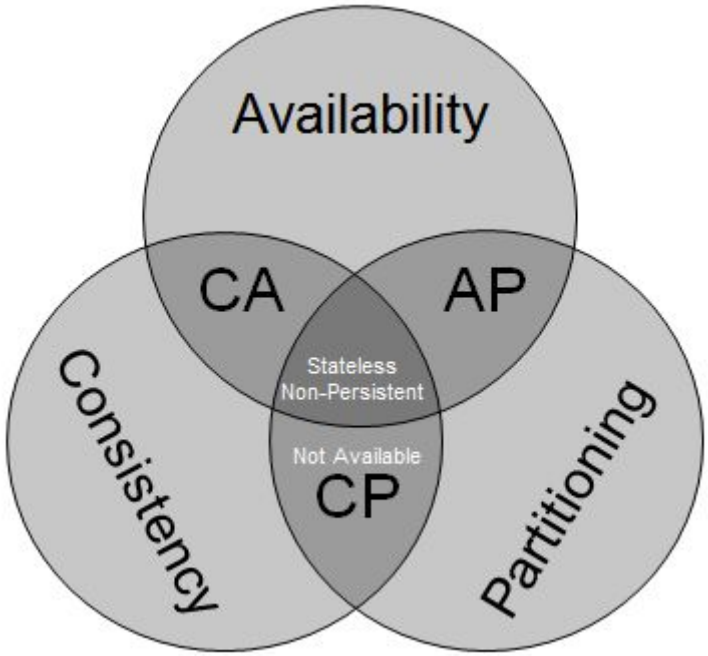
Использование ORM Tool вместе с денормализацией размывает

Domain Model

EXIGEN services

Проблема 3. Проблема масштабируемости

В теореме CAP при обработке бизнес данных мы всегда выбираем целостность данных (C)



EXIGEN services

Проблема 4: В реальной жизни не бывает конфликтов модификации данных

- В бизнес процессах происходящих в реальной жизни не бывает конфликтов модификации данных.
- Если возникает такой конфликт значит это проблема в реализации бизнес логики.
- CRUD не учитывает этого.

CQRS как решение

III. CQRS как решение

CQRS - Command Query Responsibility Segregation

EXIGEN services

Проблема 1: Использование JavaBean

Использование JavaBean остаётся для отображения данных на стороне обработки запросов, но JavaBean != Domain Entity.

EXIGEN services

Проблема 2: Оптимизация производительности и её последствия.

Денормализация данных выполняется только на стороне обработки запросов.

EXIGEN services

Данные на стороне запросов

Каждая таблица – денормализованное представление данных на экране пользователя

Нет необходимости применять SQL базы данных.

Высокопроизводительные альтернативы - Apache Cassandra, HBase, Hypertable ….

EXIGEN services

Проблема 3: Проблема масштабируемости

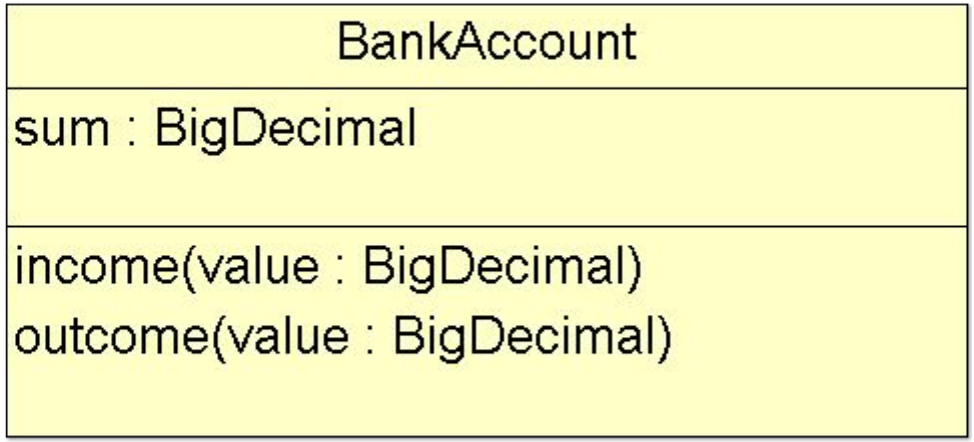
- Целостность данных нужна только на стороне обработки бизнес логики.
- На стороне обработки запросов мы можем использовать eventual consistency

EXIGEN services

Проблема 4: В реальной жизни не бывает конфликтов модификации данных

Два способа представления состояния объекта

1. В виде значений переменных внутри объекта.

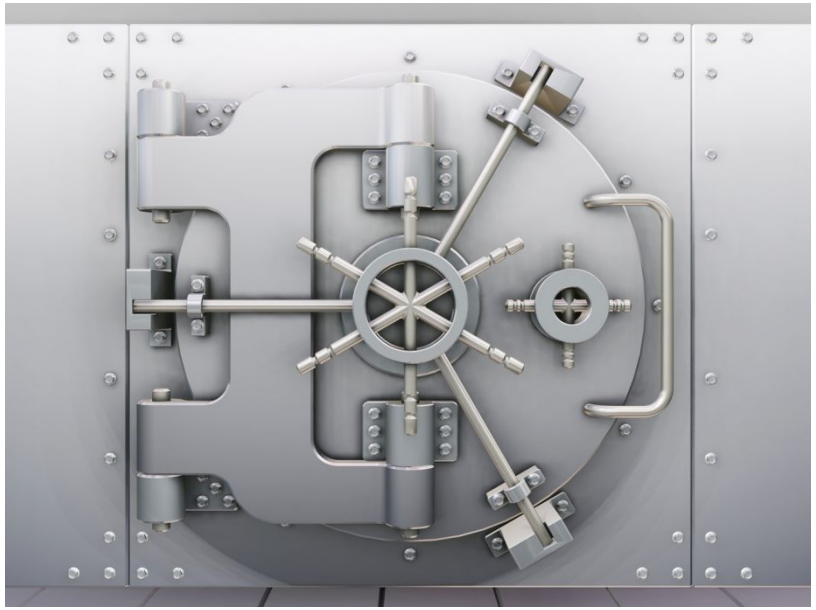


Два подхода представления объектов

2. В виде последовательности событий

+11.2 M $

+ 5.3 M $



Sum: 8.3 M $

- 8.2 M $

EXIGEN services

Два подхода представления объектов

- Каждый агрегат – это пакет событий.
- Нет необходимости использовать реляционные базы данных.
- База должна обладать ACID свойствами.

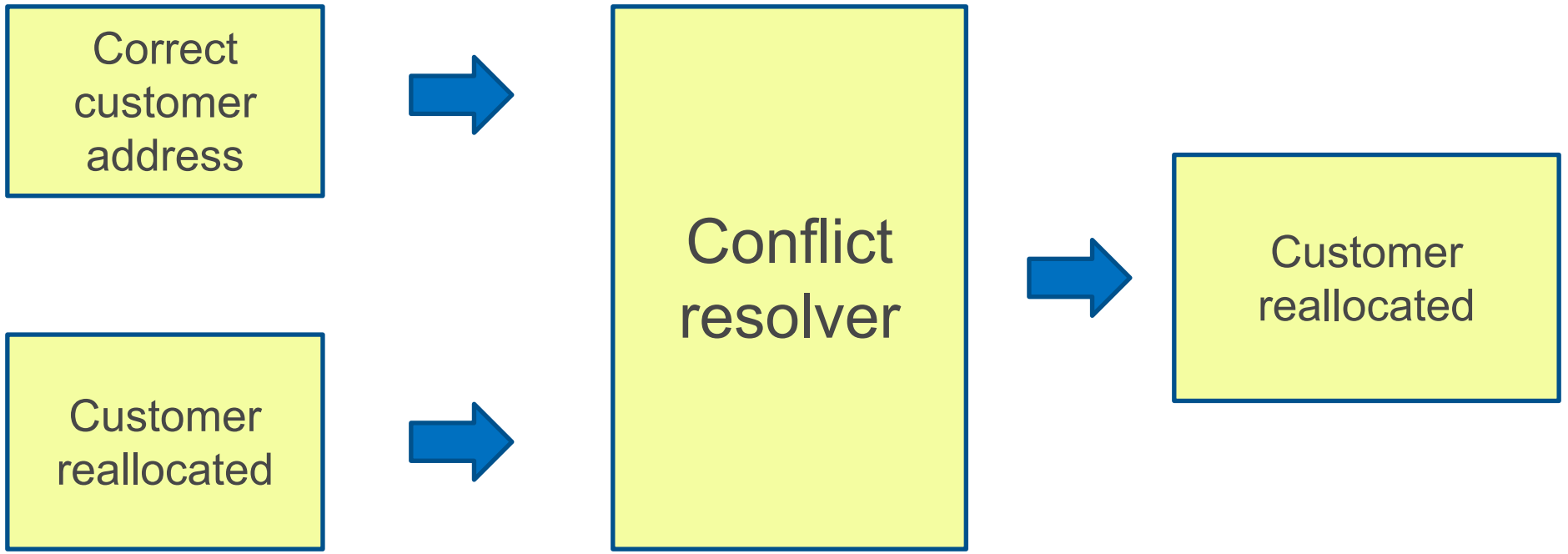
EXIGEN services

Два подхода представления объектов

Преимущества:

- удобный мониторинг изменений состояния системы
- возможность отката состояния системы до любого момента времени
- удобный механизм репликации данных и разрешения конфликтов

Разрешение конфликтов



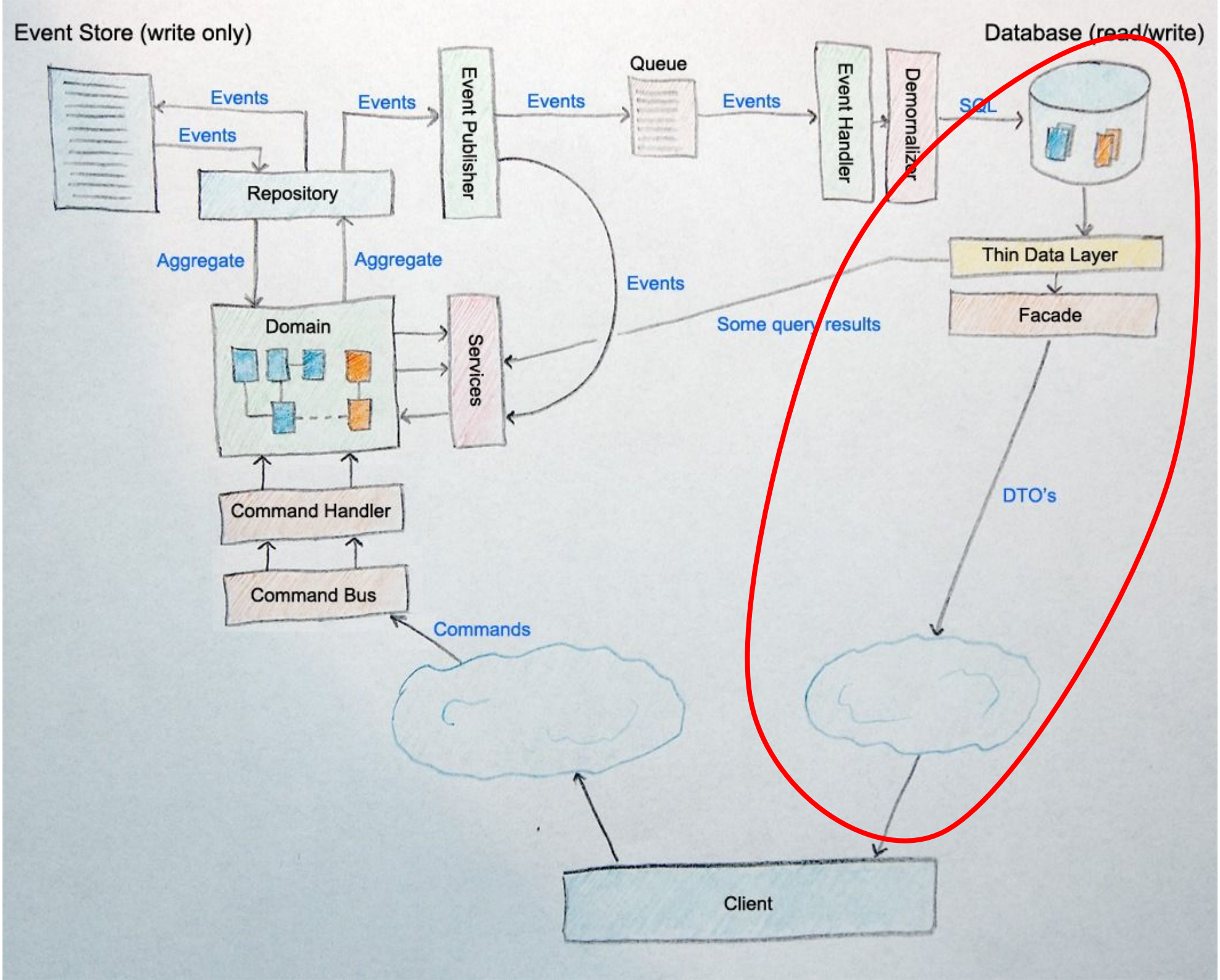
IV. Архитектура CQRS приложений

EXIGEN services

CQRS – это только подход

CQRS это только подход,

как вы его реализуете, зависит только от вас.


Event Store (write only)
Database (read/write)
Events
Events
Events
Events
Event Publisher
Events
Queue
Events
Event Handler
Demormalizer
SQL
Repository
Aggregate
Aggregate
Thin Data Layer
Events
Facade
Some query results
Domain
Services
DTO's
Command Handler
Command Bus
Commands
Client

EXIGEN services

Queries

Многослойная архитектура

Запрос к DB

Конвертирование в доменную модель

Конвертирование в DTO

Передача клиенту

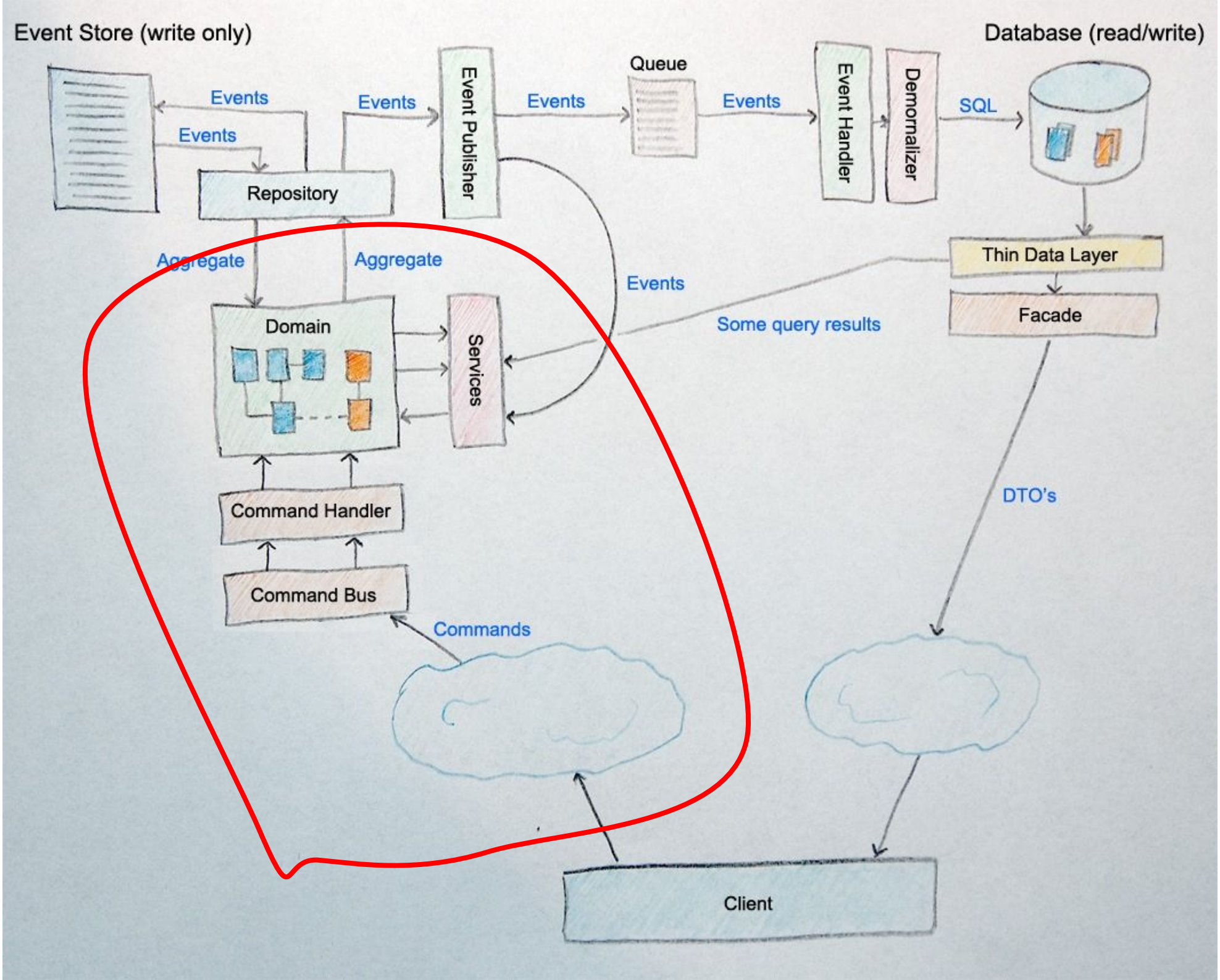
CQRS

Запрос к DB

Конвертирование в доменную модель

Конвертирование в DTO

Передача клиенту


Event Store (write only)
Database (read/write)
Events
Events
Events
Event Publisher
Events
Queue
Events
Event Handler
Demormalizer
SQL
Repository
Aggregate
Aggregate
Thin Data Layer
Events
Facade
Some query results
Domain
Services
DTO's
Command Handler
Command Bus
Commands
Client

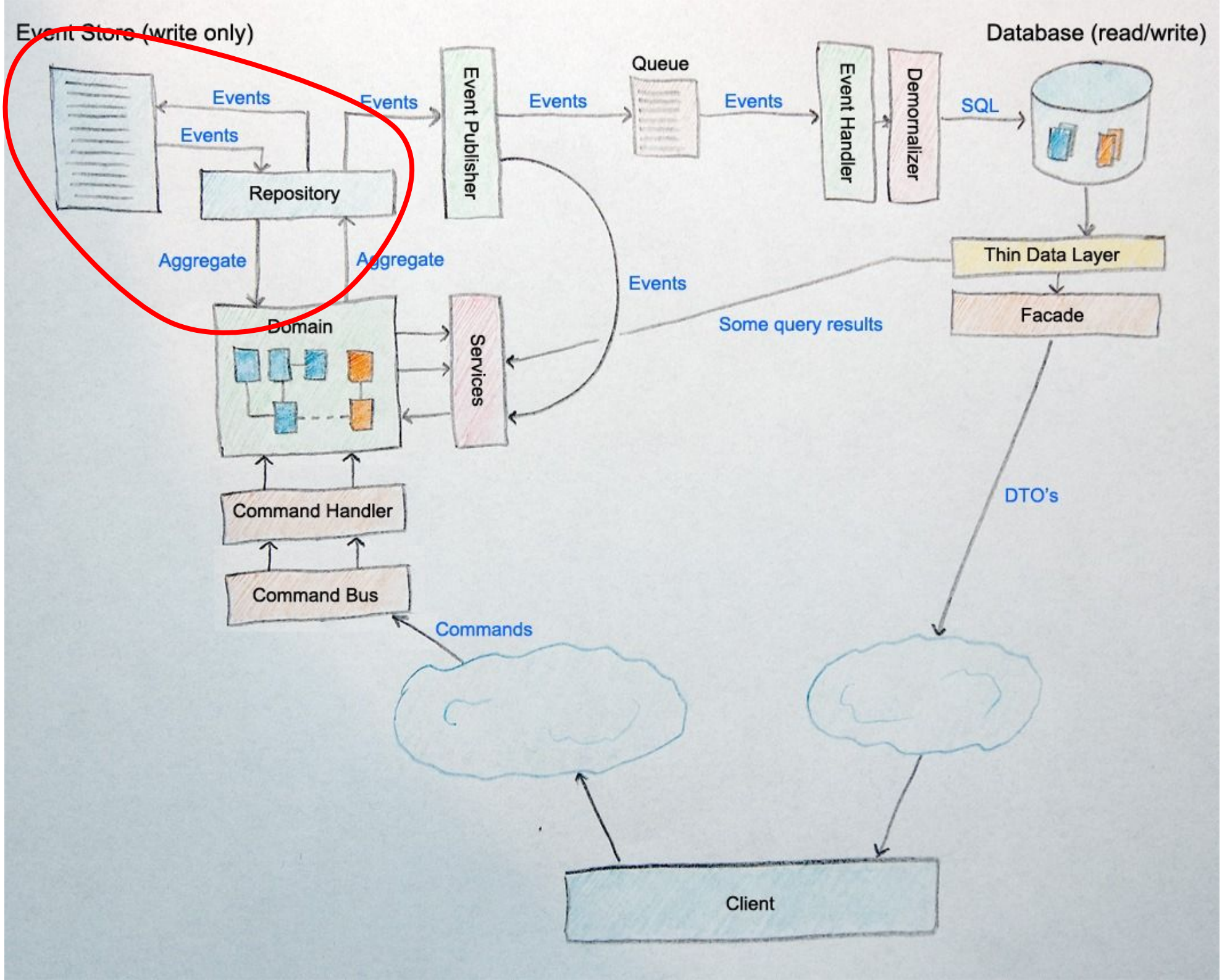
EXIGEN services

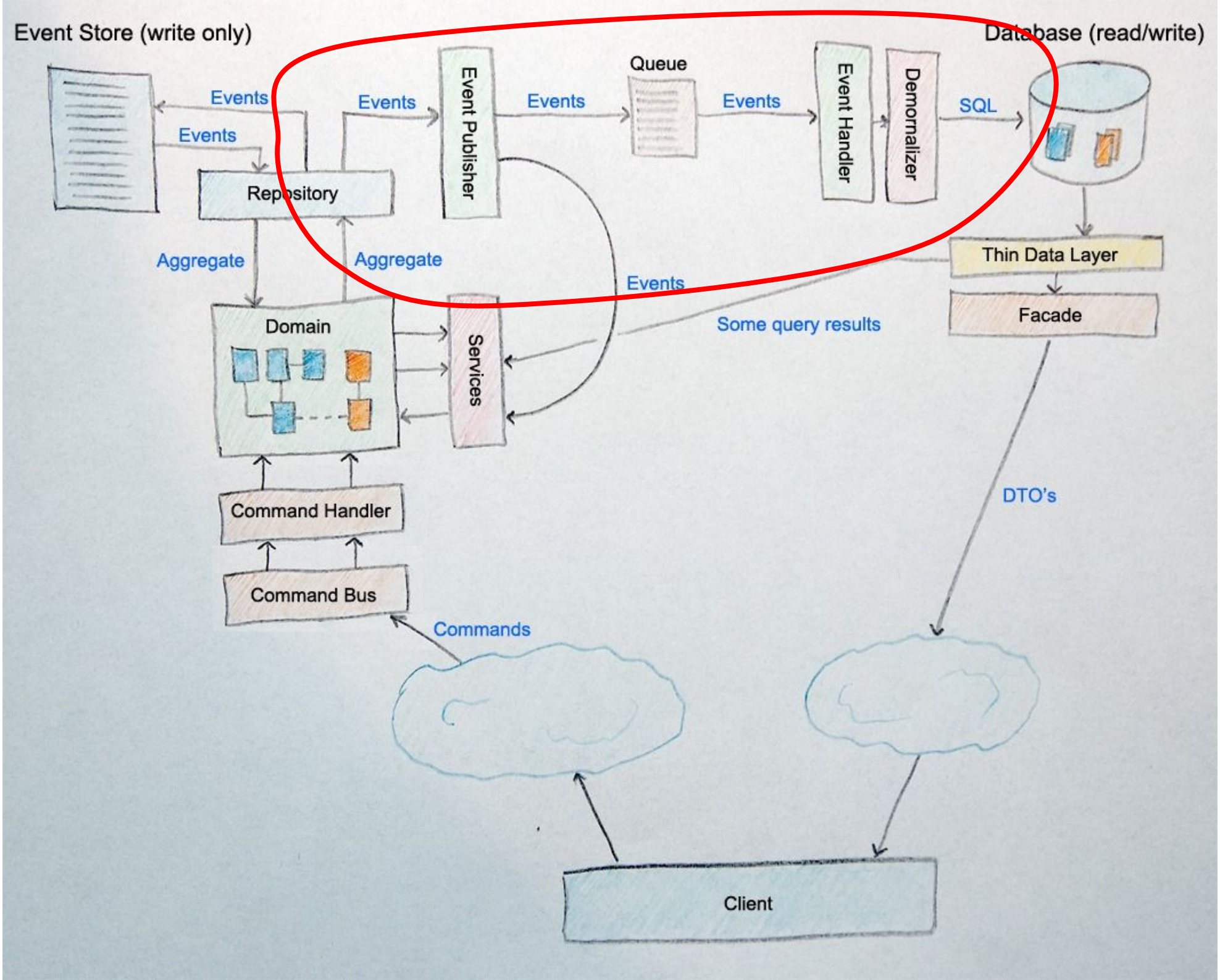
Commands

Команда – представляет собой отражение бизнес действия, действия в котором заинтересован пользователь приложения.

Преимущество использования команд:

1. Ориентация на бизнес проблемы пользователя.
2. Удобный механизм мониторинга и масштабирования


Event Store (write only)
Database (read/write)
Events
Events
Events
Events
Event Publisher
Events
Queue
Events
Event Handler
Demormalizer
SQL
Repository
Aggregate
Aggregate
Thin Data Layer
Events
Facade
Some query results
Domain
Services
DTO's
Command Handler
Command Bus
Commands
Client


Event Store (write only)
Database (read/write)
Events
Events
Events
Events
Event Publisher
Events
Queue
Events
Event Handler
Demormalizer
SQL
Repository
Aggregate
Aggregate
Thin Data Layer
Events
Facade
Some query results
Domain
Services
DTO's
Command Handler
Command Bus
Commands
Client

EXIGEN services

Event bus

- Репликация данных между компонентом обработки бизнес логики и компонентом селективных выборок.
- В обычном приложении количество селективных запросов на порядок больше количества запросов на изменение данных.
- Один бизнес компонент - множество query компонент

EXIGEN services

….преимущества…..

1. Хорошие условия для реализации DDD
2. Использование CEP
3. Готовность к облачным вычислениям
4. Простота распределения обязанностей между узконаправленными командами

V. Реализация CQRS

EXIGEN services

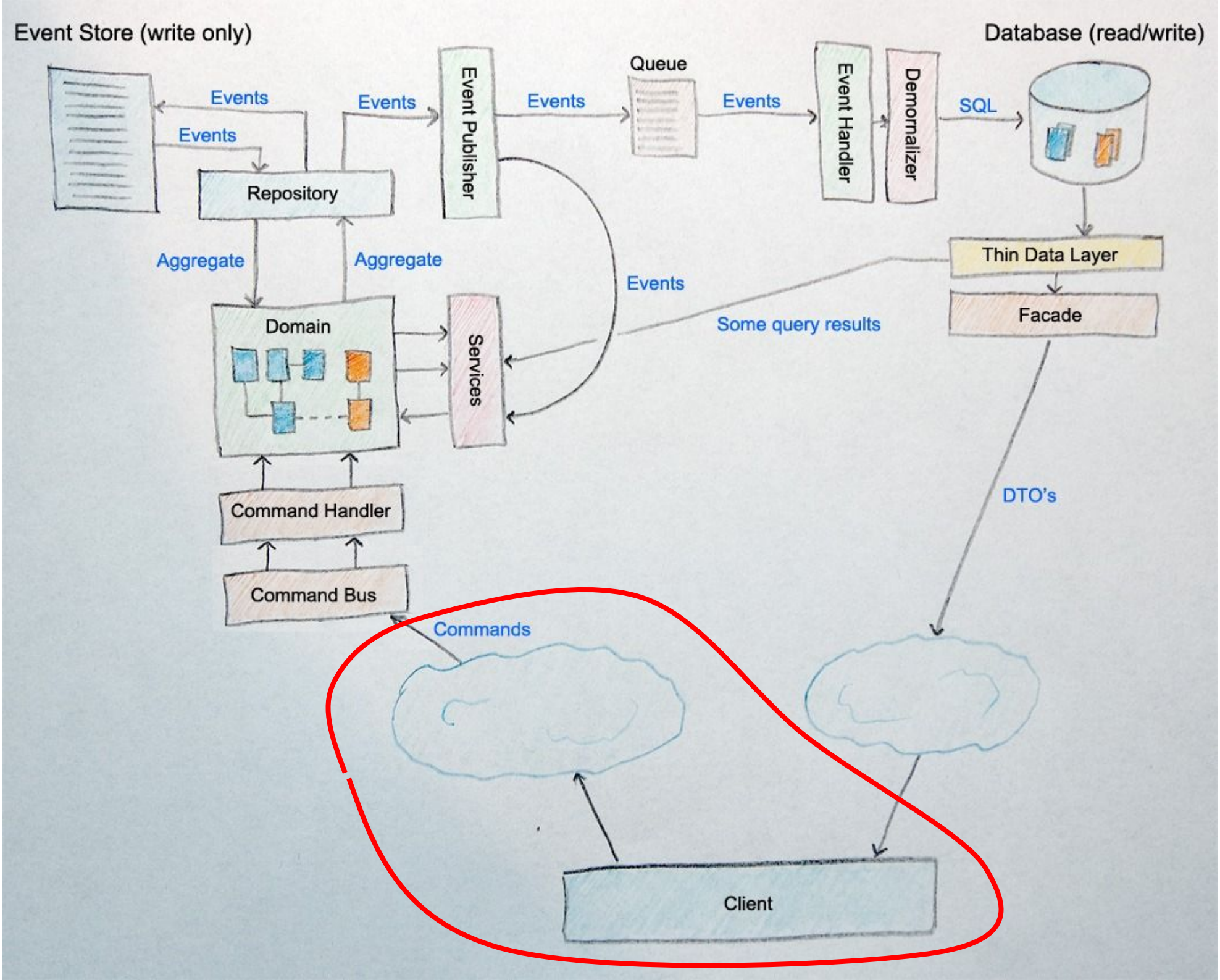
Axon framework

Axon framework - самый популярный и наиболее функциональный.

EXIGEN services

Пример CQRS приложения

Address Book –
управление списком
адресов


Event Store (write only)
Database (read/write)
Events
Events
Events
Events
Event Publisher
Events
Queue
Events
Event Handler
Demormalizer
SQL
Repository
Aggregate
Aggregate
Thin Data Layer
Events
Facade
Some query results
Domain
Services
DTO's
Command Handler
Command Bus
Commands
Client

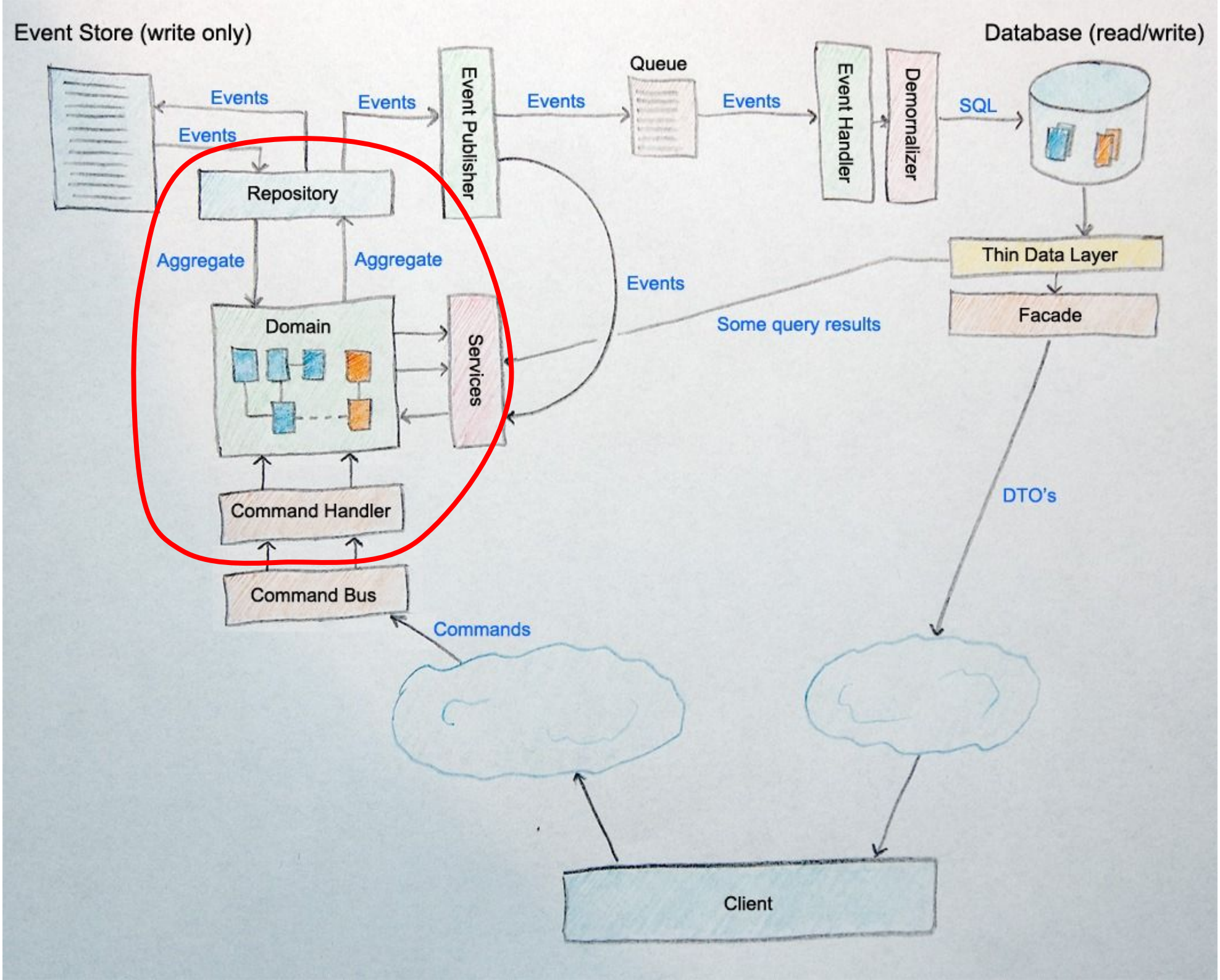
EXIGEN services

Создание и отправка команды

```
@RequestMapping(value = "{identifier}/address/new", method = RequestMethod.POST)
public String formNewAddressSubmit(@ModelAttribute("address") @Valid AddressEntry address,
                                   BindingResult bindingResult) {
    if (!bindingResult.hasErrors()) {
        RegisterAddressCommand command = new RegisterAddressCommand();
        command.setAddressType(address.getAddressType());
        command.setCity(address.getCity());
        command.setContactId(address.getIdentifier());
        command.setStreetAndNumber(address.getStreetAndNumber());
        command.setZipCode(address.getZipCode());

        commandBus.dispatch(command);

        return "redirect:/contacts/" + address.getIdentifier();
    }
    return "contacts/address";
}
```


Event Store (write only)
Database (read/write)
Events
Events
Events
Events
Event Publisher
Events
Queue
Events
Event Handler
Demormalizer
SQL
Repository
Aggregate
Aggregate
Domain
Services
Events
Some query results
Thin Data Layer
Facade
DTO's
Command Handler
Command Bus
Commands
Client

EXIGEN services

Обработка команды

```
@CommandHandler
public void handle(RegisterAddressCommand command) {
    Address address = new Address(
            command.getStreetAndNumber(),
            command.getZipCode(),
            command.getCity()
        );

    Contact contact = repository.load(new StringAggregateIdentifier(command.getContactId()));
    contact.registerAddress(command.getAddressType(), address);
}
```

EXIGEN services

Бизнес действие

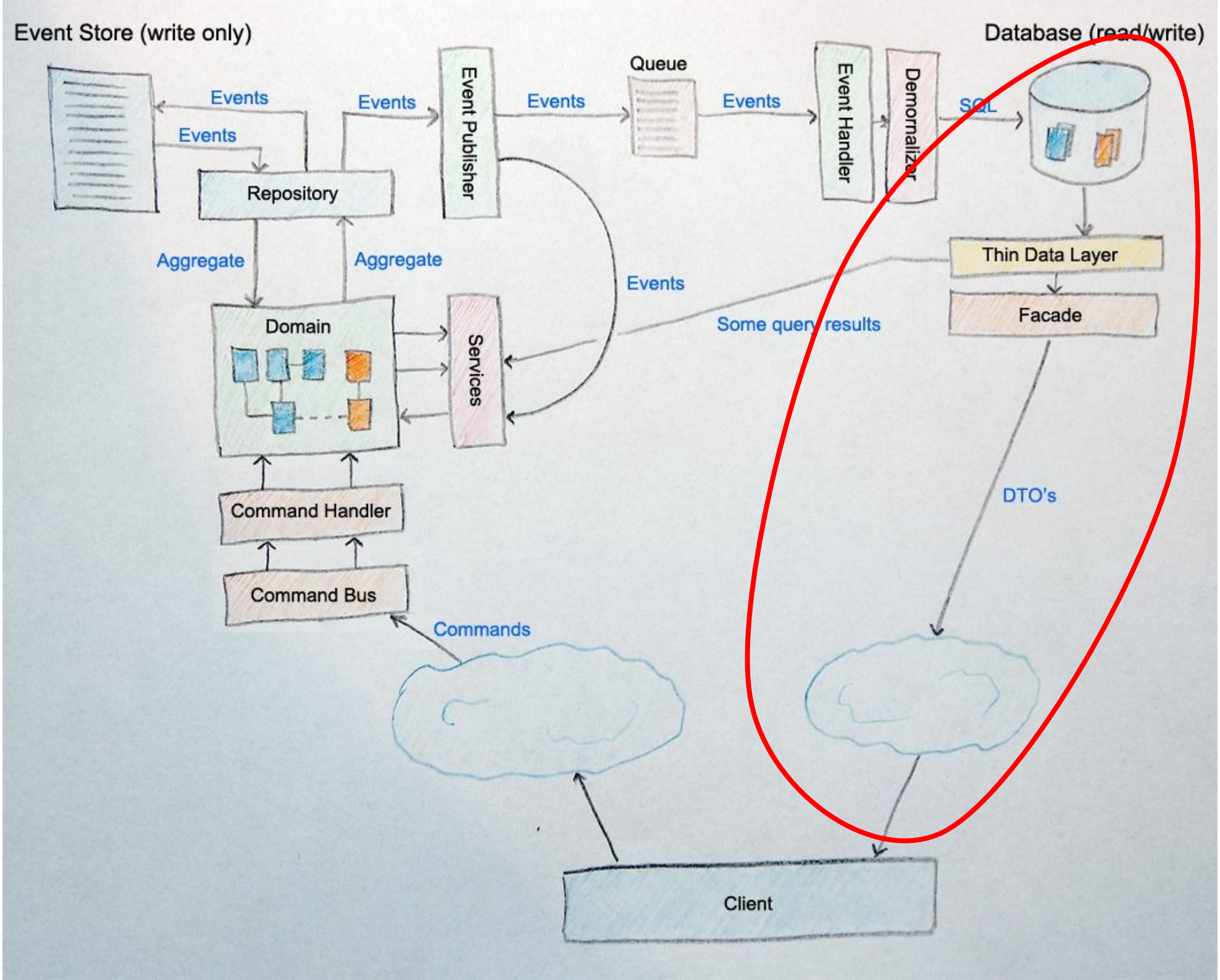
```
public void registerAddress(AddressType type, Address address) {
    if (addresses.containsKey(type)) {
        apply(new AddressChangedEvent(type, address));
    } else {
        apply(new AddressAddedEvent(type, address));
    }
}
```

EXIGEN services

Обработка события изменения состояния

```
@EventHandler
public void handleAddressChangedEvent(AddressChangedEvent event) {
    AddressEntry entry = (AddressEntry) entityManager.createQuery(
            "SELECT e from AddressEntry e WHERE e.identifier = :id and e.addressType = :type")
            .setParameter("id", event.getContactIdentifier())
            .setParameter("type", event.getType())
            .getSingleResult();

    entry.setStreetAndNumber(event.getAddress().getStreetAndNumber());
    entry.setZipCode(event.getAddress().getZipCode());
    entry.setCity(event.getAddress().getCity());
    entityManager.persist(entry);
}
```


Event Store (write only)
Database (read/write)
Events
Events
Events
Events
Event Publisher
Events
Queue
Events
Event Handler
Demormalizer
SQL
Repository
Aggregate
Aggregate
Thin Data Layer
Events
Facade
Some query results
Domain
Services
DTO's
Command Handler
Command Bus
Commands
Client

EXIGEN services

Запросная часть

```
public List<AddressEntry> findAllAddressesForContact(String contactIdentifier) {
    return entityManager.createQuery("SELECT e FROM AddressEntry e WHERE e.identifier = :id")
            .setParameter("id", contactIdentifier)
            .setMaxResults(10)
            .getResultList();
}
```

EXIGEN services

Простота настройки

```
<context:annotation-config/>

<axon:annotation-config command-bus="commandBus" executor="taskExecutor"/>


<axon:command-bus id="commandBus">
    <axon:interceptors>
        <bean class="org.axonframework.commandhandling.interceptors.SpringTransactionalInterceptor">
            <property name="transactionManager" ref="transactionManager"/>
        </bean>
    </axon:interceptors>
</axon:command-bus>
```

EXIGEN services

Axon – repository и event store.

Repository		 JPA	
Event Store	 File System	 JPA	 Mongo DB

EXIGEN services

Event Stores – pros and cons

JPA Event Store

транзакционный но медленный

Mongo DB, File system

нет поддержки ACID свойств
большая скорость обработки данных

EXIGEN services

Orient DB Event store

OrientDB®

- Поддержка транзакционности.
- Очень большая скорость чтения и записи данных.
- Поддержка кластеризации.

EXIGEN services

Orient DB Event store

Результаты тестов на производительность (транзакций в секунду, Pentium Duo Core E520 2,5 GHz 2 Gb, 7200 RPM) :

Test	File system	JPA	Orient DB
Small transactions	3330	946	3146
Big transactions	8506	1582	3354

Примеры реализации в ExigenServices

Примеры реализации внутри компании

- Автоматизация выдачи залогов и учёт ценностей в ломбардах.
- Учёт нарядов выполненных работ в Днепропетровской Торгово-Промышленной Палате

EXIGEN services

Итоги

CQRS – подход, обеспечивающий реализацию современных требований к корпоративным системам с точки зрения их масштабируемости, мониторинга, построения сложной бизнес логики.

Вопрос

EXIGEN services

Авторы

- Артём Оробец. twitter: @Dr_EniSh , enisher@gmail.com, skype: dr_enish
- Андрей Ломакин. twitter: @Andrey_Lomakin , lomakin.andrey@gmail.com , skype: lomakin_andrey

EXIGEN services

Источники

1. First CQRS introduction http://www.infoq.com/presentations/greg-young-unshackle-qcon08
2. CQRS architecture overview - http://elegantcode.com/2009/11/11/cqrs-la-greg-young/
3. Greg Young blog - http://codebetter.com/gregyoung/
4. Race conditions does not exist http://www.udidahan.com/2010/08/31/race-conditions-dont-exist/
5. Domain Driven Design Aggregator - http://domaindrivendesign.org/
6. Axon framework home page - http://code.google.com/p/axonframework/
7. Mark Nijhof blog http://cre8ivethought.com/blog