Eventbrite

07.23.18

Understanding and Applying CQRS

Vinicius Feitosa Pacheco

@ViniciusPach

Microservice Patterns and Best Practices
Vinicius Feitosa Pacheco
January 2018

http://bit.ly/microservice-book

@ViniciusPach

Agenda

1. What is CQRS?
2. When can we use CQRS ?
3. Understanding CQRS
4. Applying CQRS
5. Common Mistakes

1

What is CQRS?

Command

Query

Responsibility

Segregation

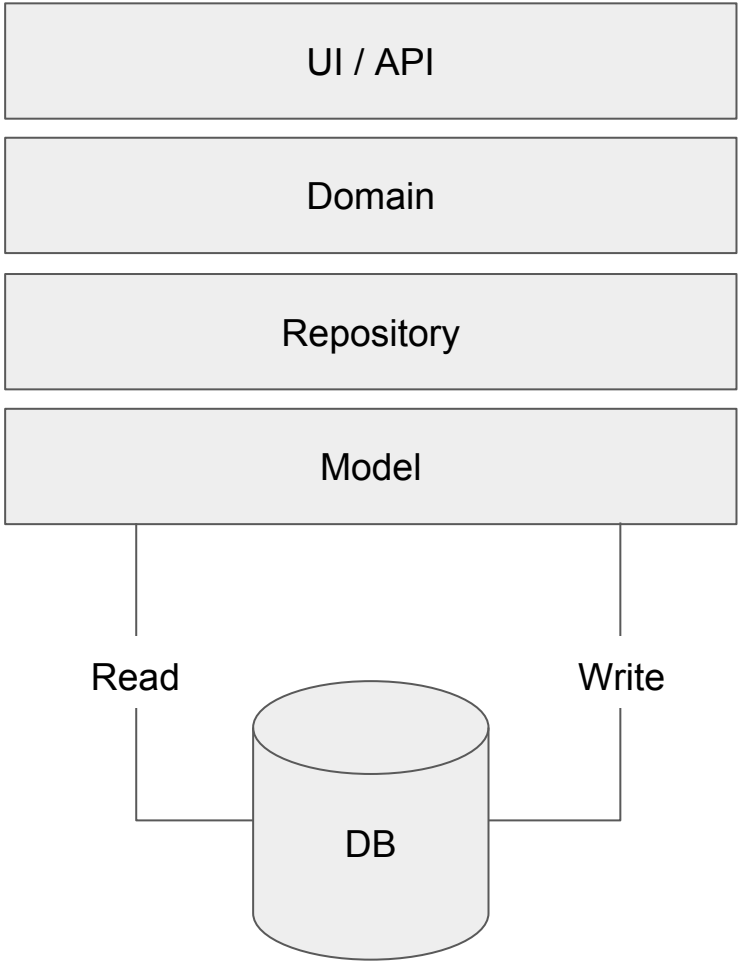
> CQRS is simply the creation of two objects where there was previously only one. The separation occurs based upon whether the methods are a command or a query...

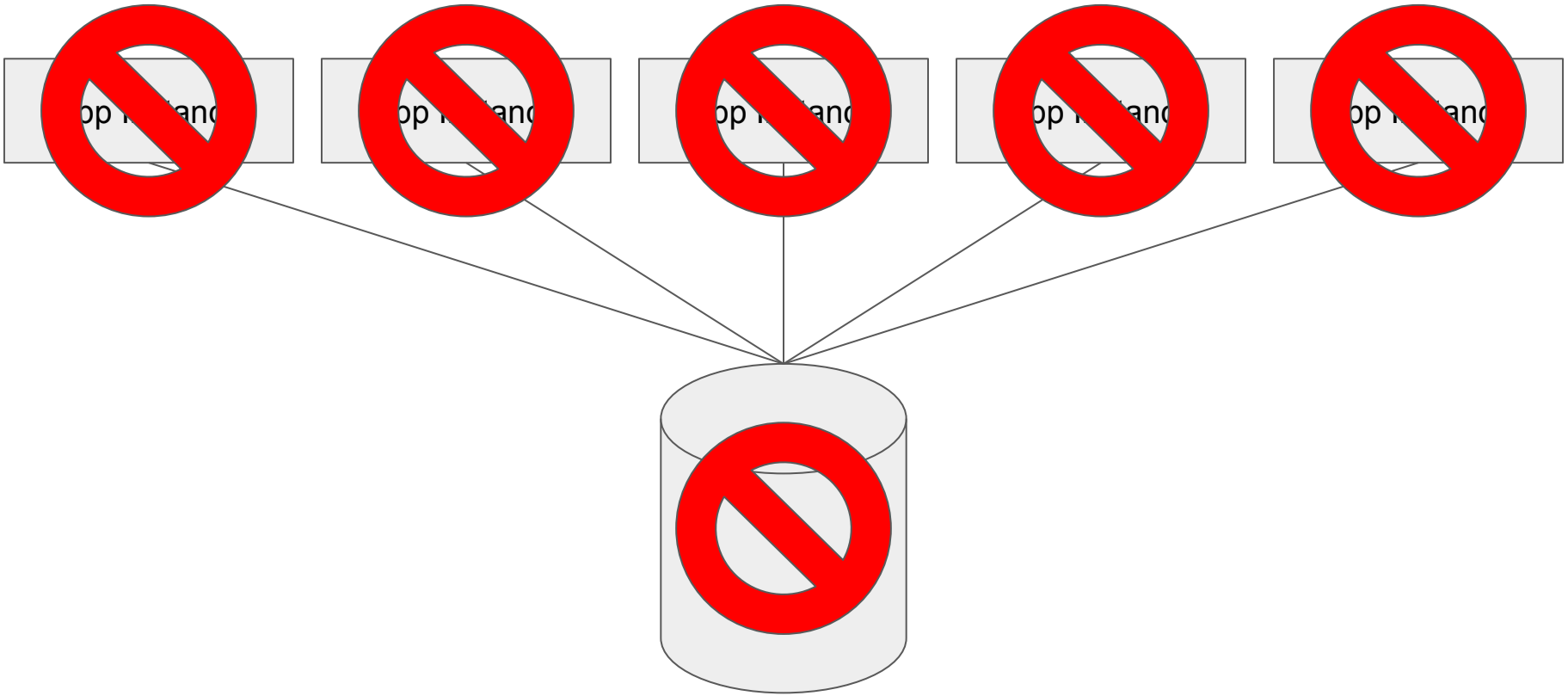
Young, Greg (2010)

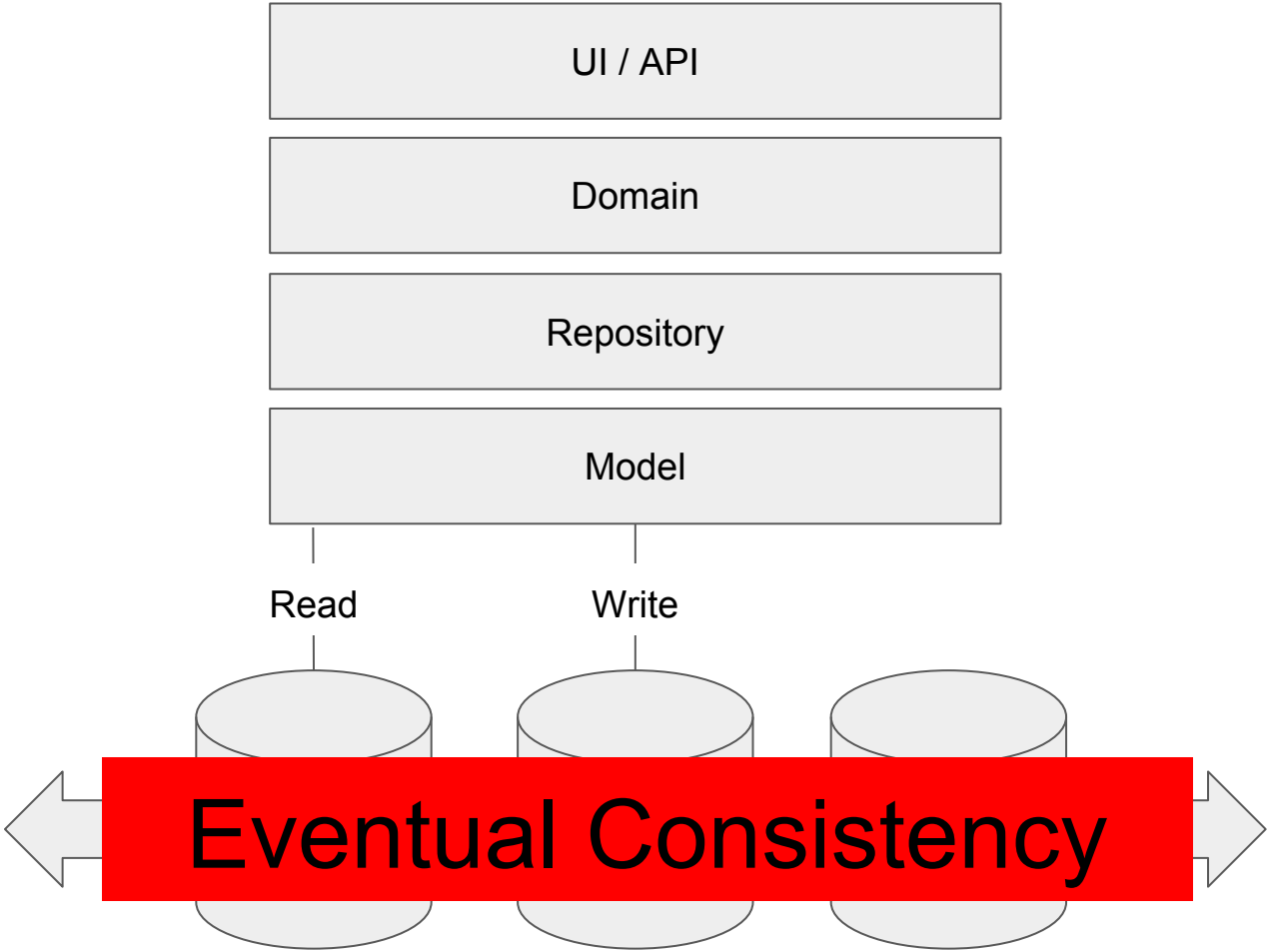
2

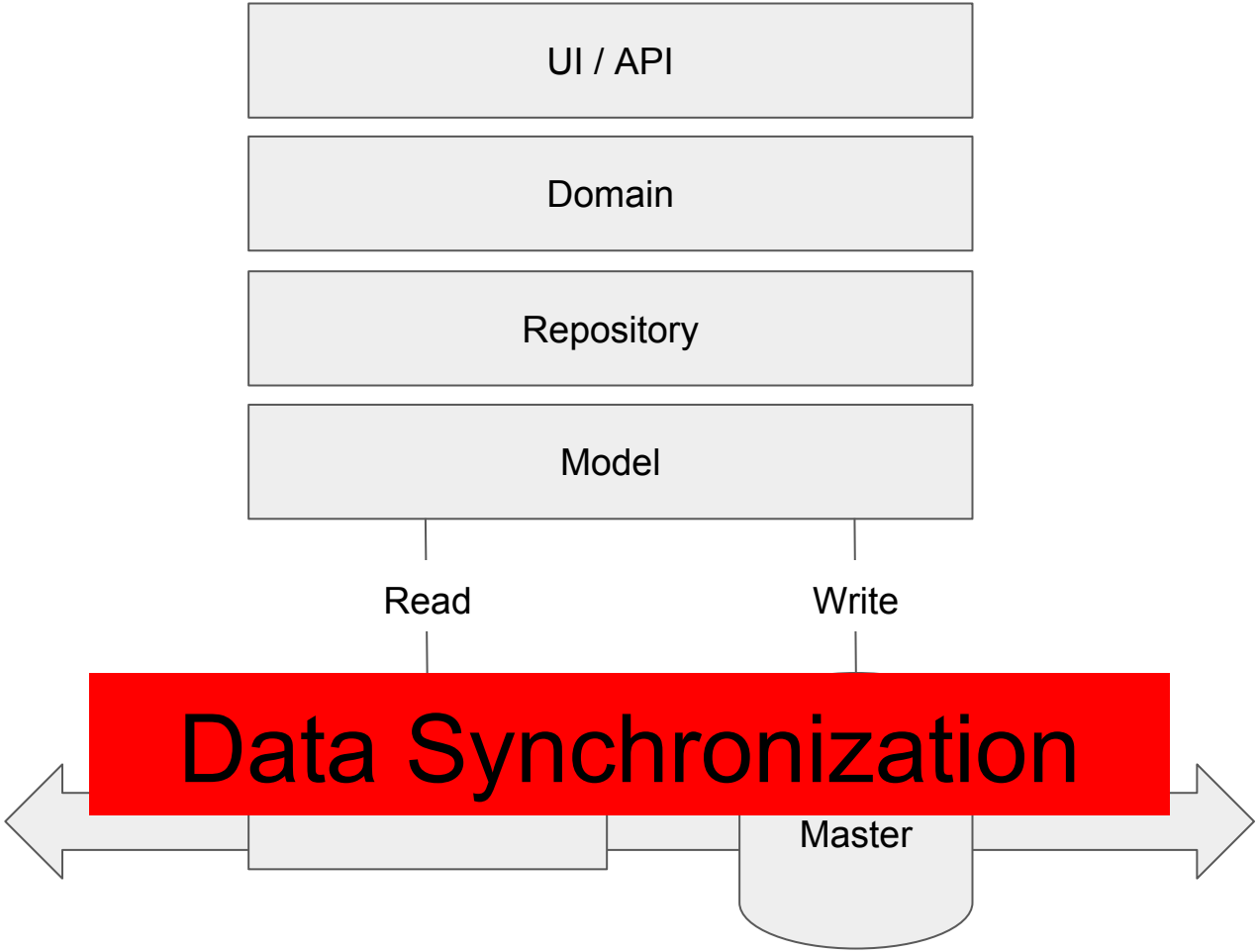
When can we use CQRS ?

The “Normal” Architecture


UI / API
Domain
Repository
Model
Read
Write
DB




UI / API
Domain
Repository
Model
Read
Write
Eventual Consistency


UI / API
Domain
Repository
Model
Read
Write
Data Synchronization
Master

When CQRS is util?

- When my storage is a bottleneck.
- When the application has complex queries and these queries could be optimized.
- When a big number of users are updating a small data set and the data could be outdated.

3

Understanding CQRS

CQRS

- Query Stack – Operations that retrieve information from the data in the application.
- Command Stack – Operations that modify the State of the data in the application.

3.1

QueryStack

QueryStack

- It is simpler than the CommandStack.
- It is a synchronous layer that retrieves data from a denormalized reading.
- Presumes “flat” queries.

3.2

CommandStack

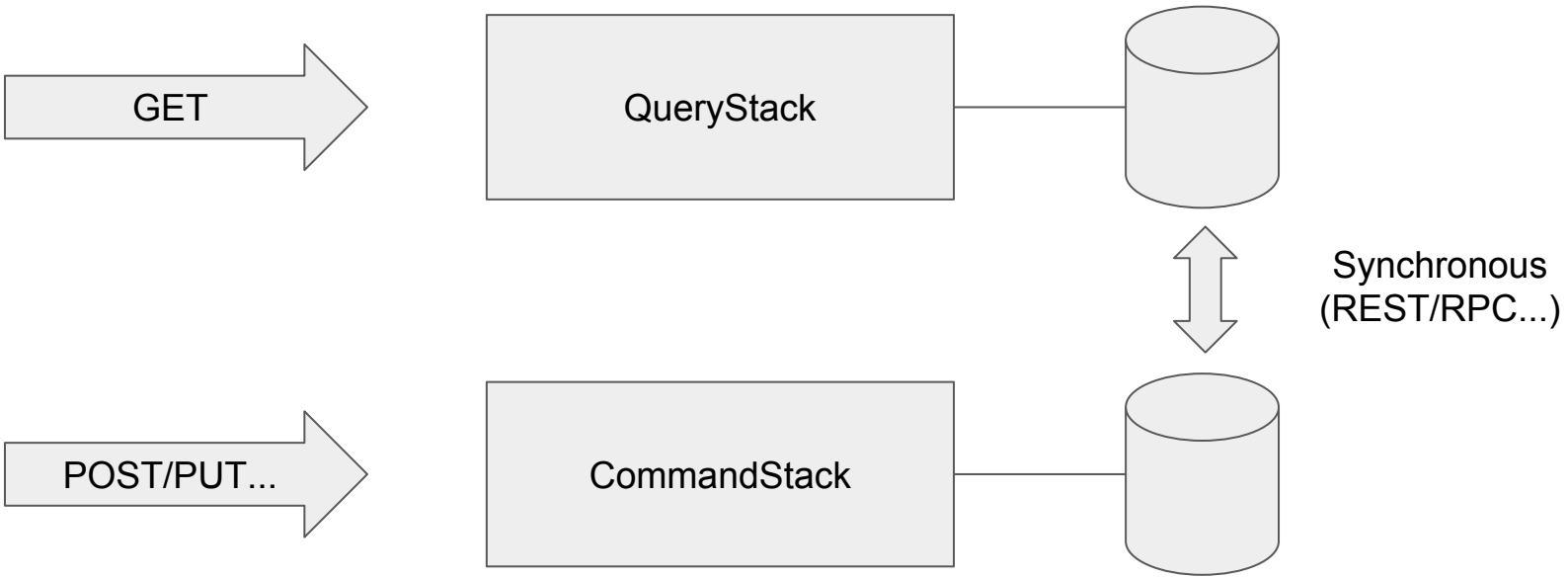
CommandStack

- Potentially asynchronous.
- Has *behavior-centric* approach where all business intention is initially triggered by the client as a use case.
- The Commands are declared in an imperative fashion and are raised asynchronously.
- The handlers returns success or failure.
- A command updates the state of an entity and raises an event that will update the data needed in the database reading.

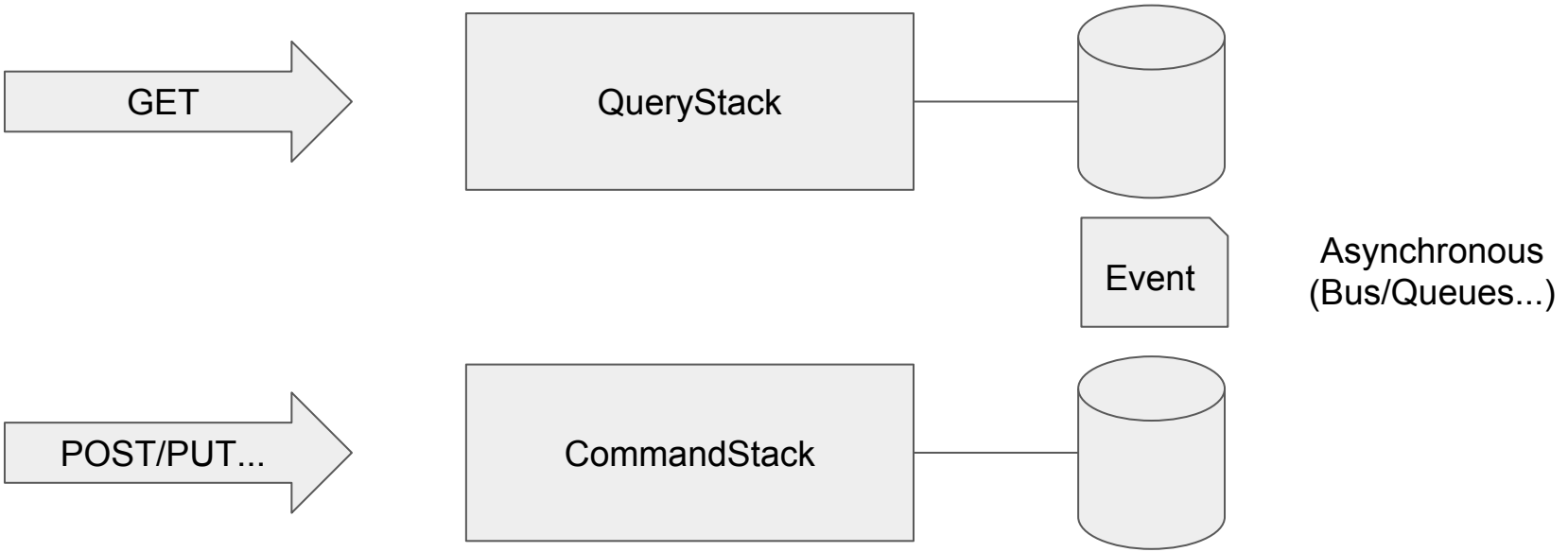
3.3

Synchronization

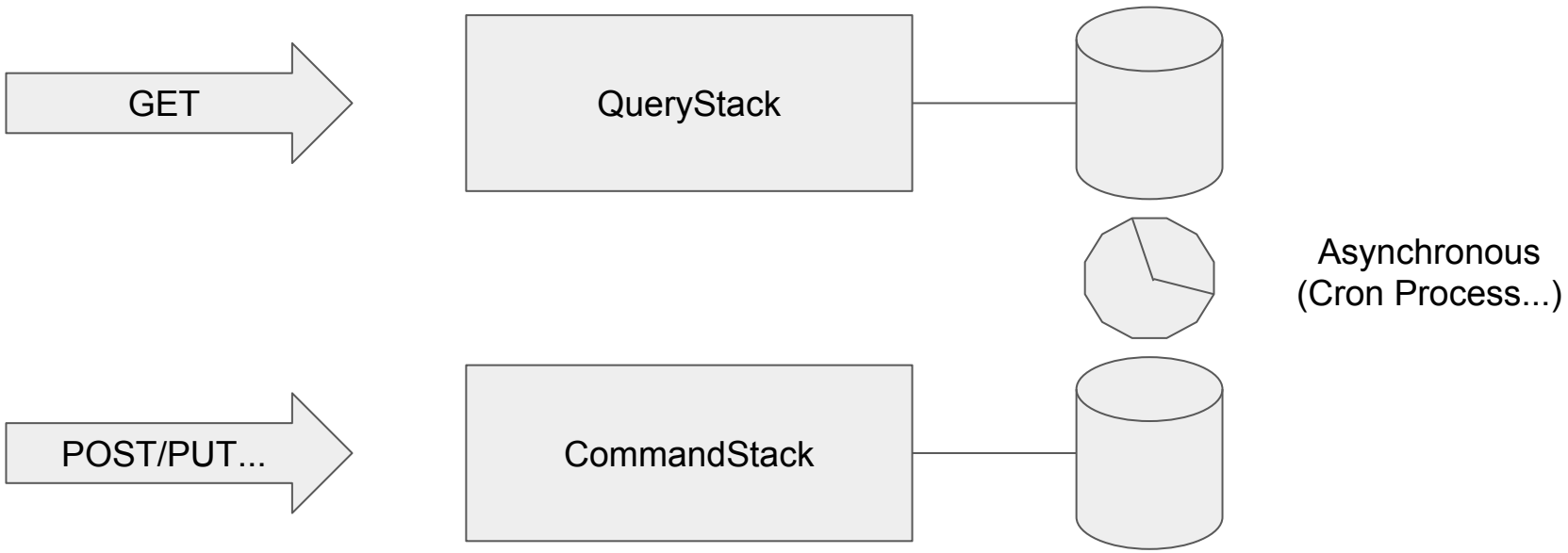
Synchronization: Automatic updating



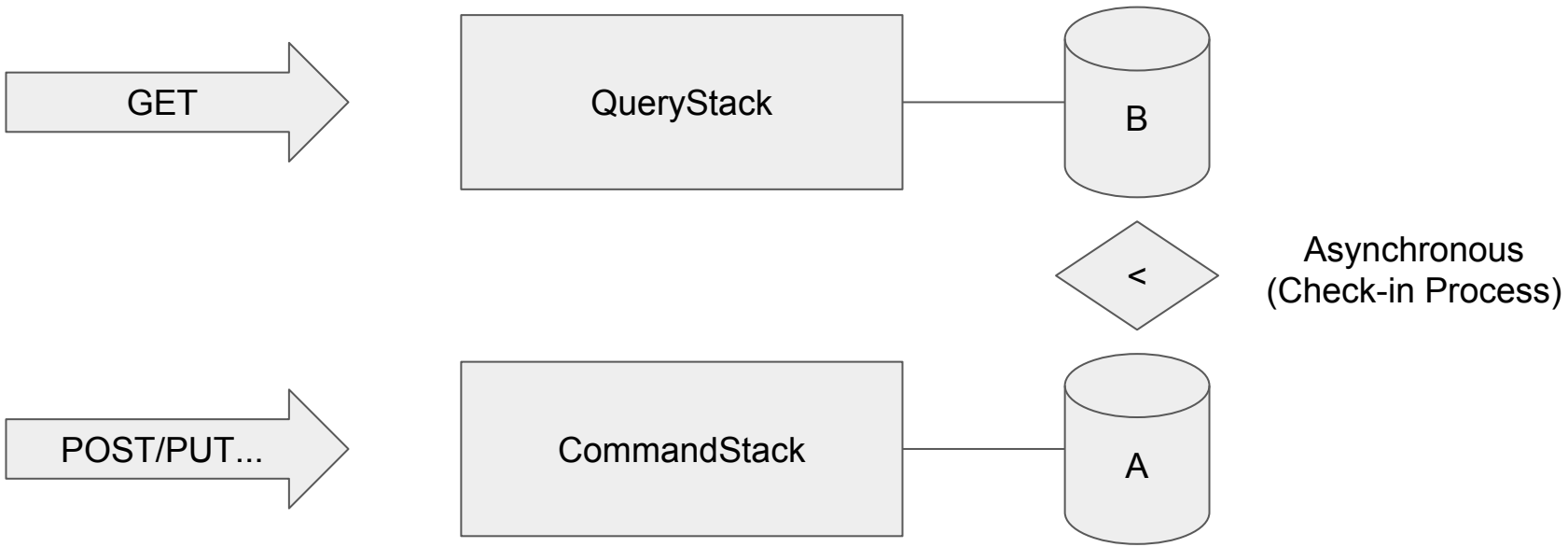
Synchronization: Update possible



Synchronization: Controlled update

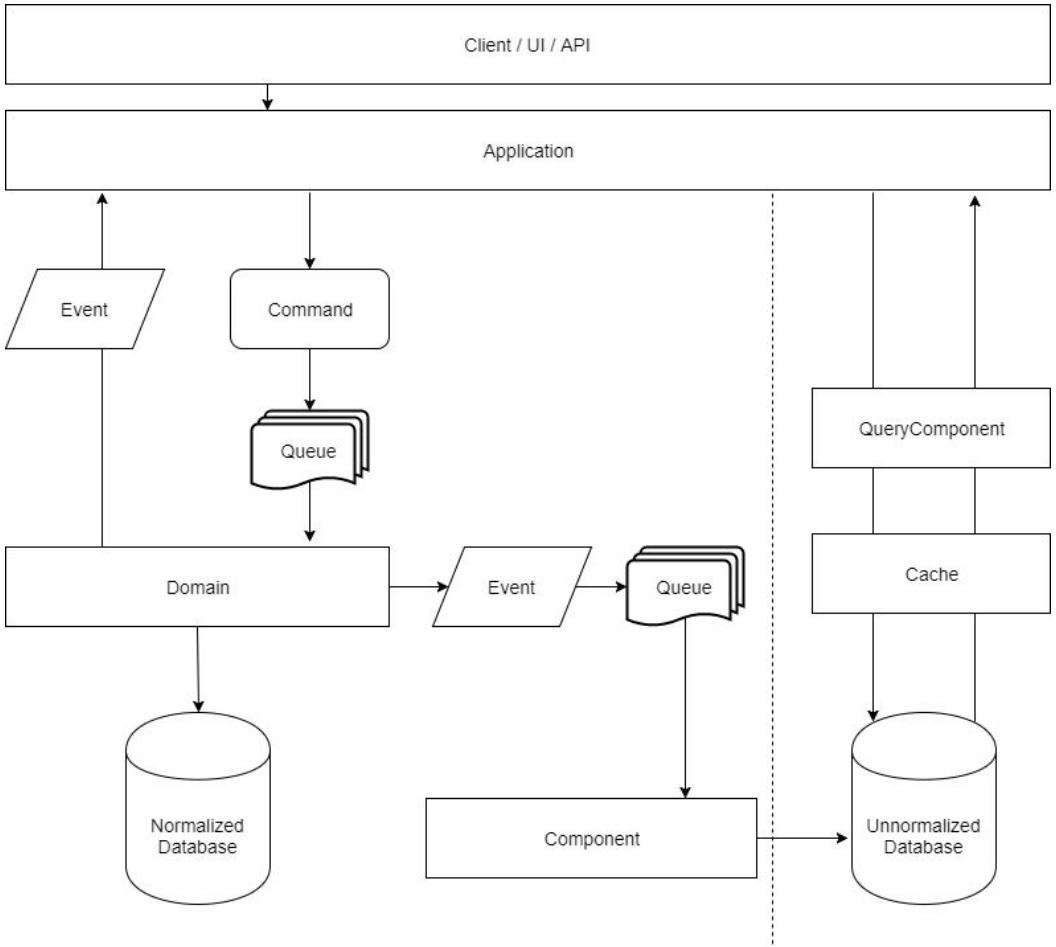


Synchronization: Update on demand



3.4

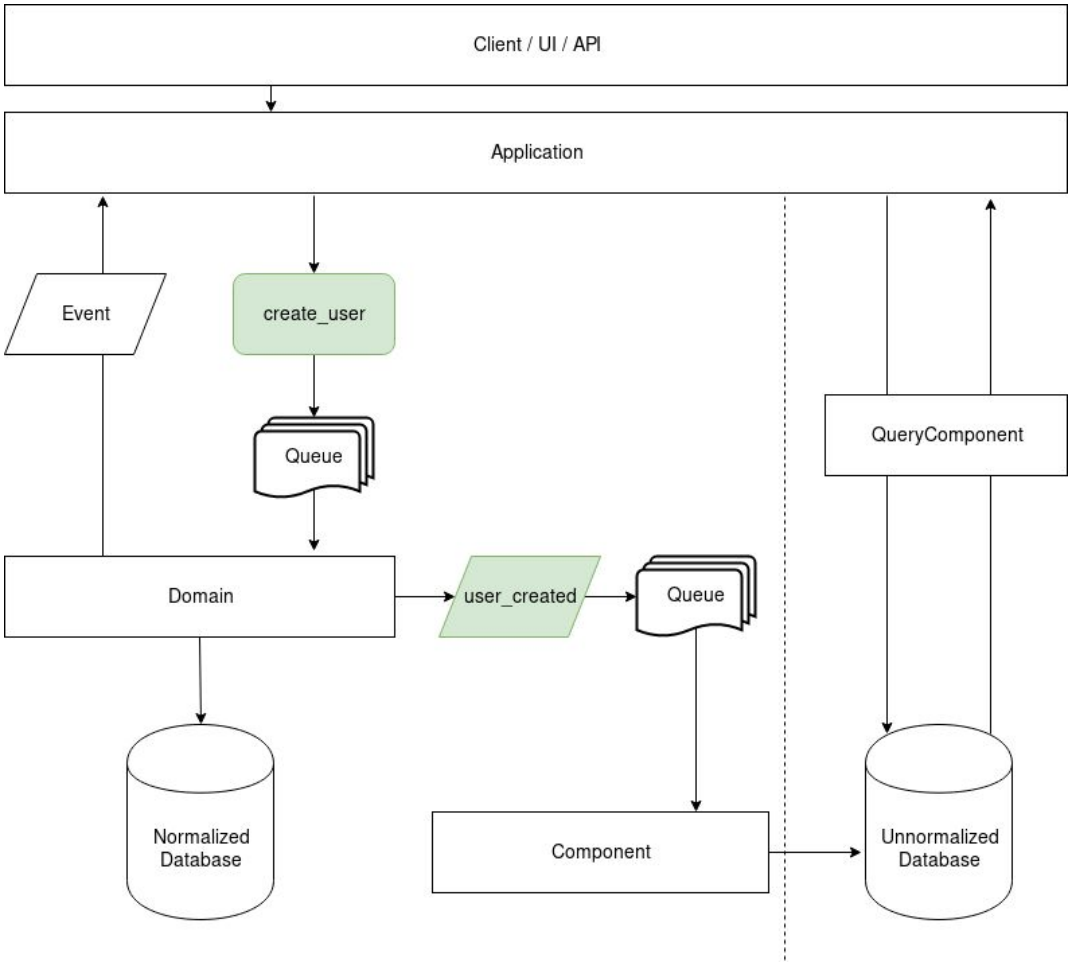
Queueing


Client / UI / API
Application
Event
Command
Queue
QueryComponent
Domain
Event
Queue
Cache
Normalized Database
Component
Unnormalized Database

4

Applying CQRS

https://github.com/viniciusfeitosa/europython2018


Client / UI / API
Application
Event
create_user
Queue
QueryComponent
Domain
user_created
Queue
Normalized Database
Component
Unnormalized Database

5

Common Mistakes

Mistake 1:

CQRS and Event Sourcing must be implemented together

Mistake 2:

CQRS requires eventual consistency

Mistake 3:

CQRS depends on Queues and Message Brokers

Mistake 4:

CQRS is easy

Mistake 5:

CQRS is architecture

“ CQRS is the best thing since sliced bread!

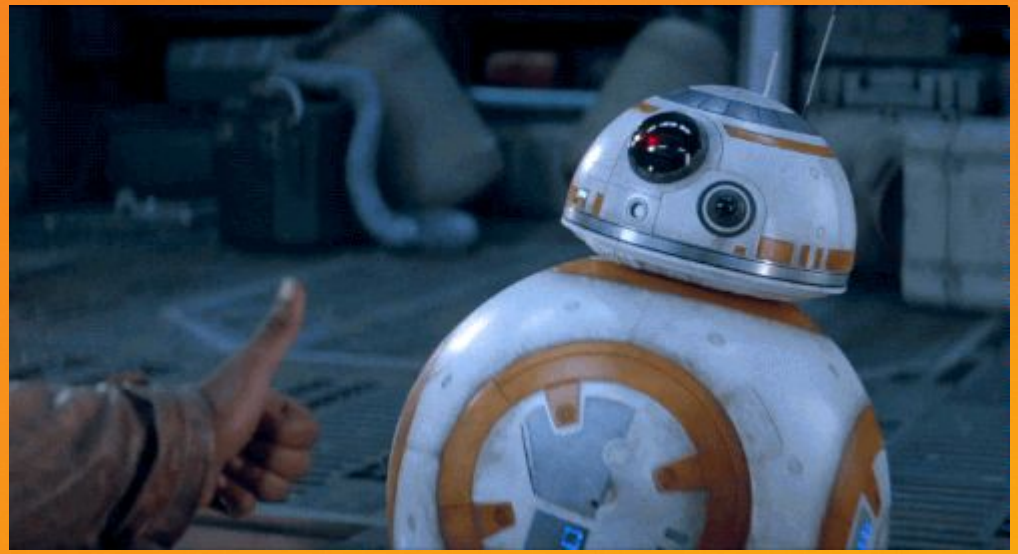
Pacheco, Vinicius (Today)

Microservice Patterns and Best Practices
Vinicius Feitosa Pacheco
January 2018

http://bit.ly/microservice-book

@ViniciusPach

“ May the CQRS be with you



Kenobi, Vinicius (2018)

Thank you