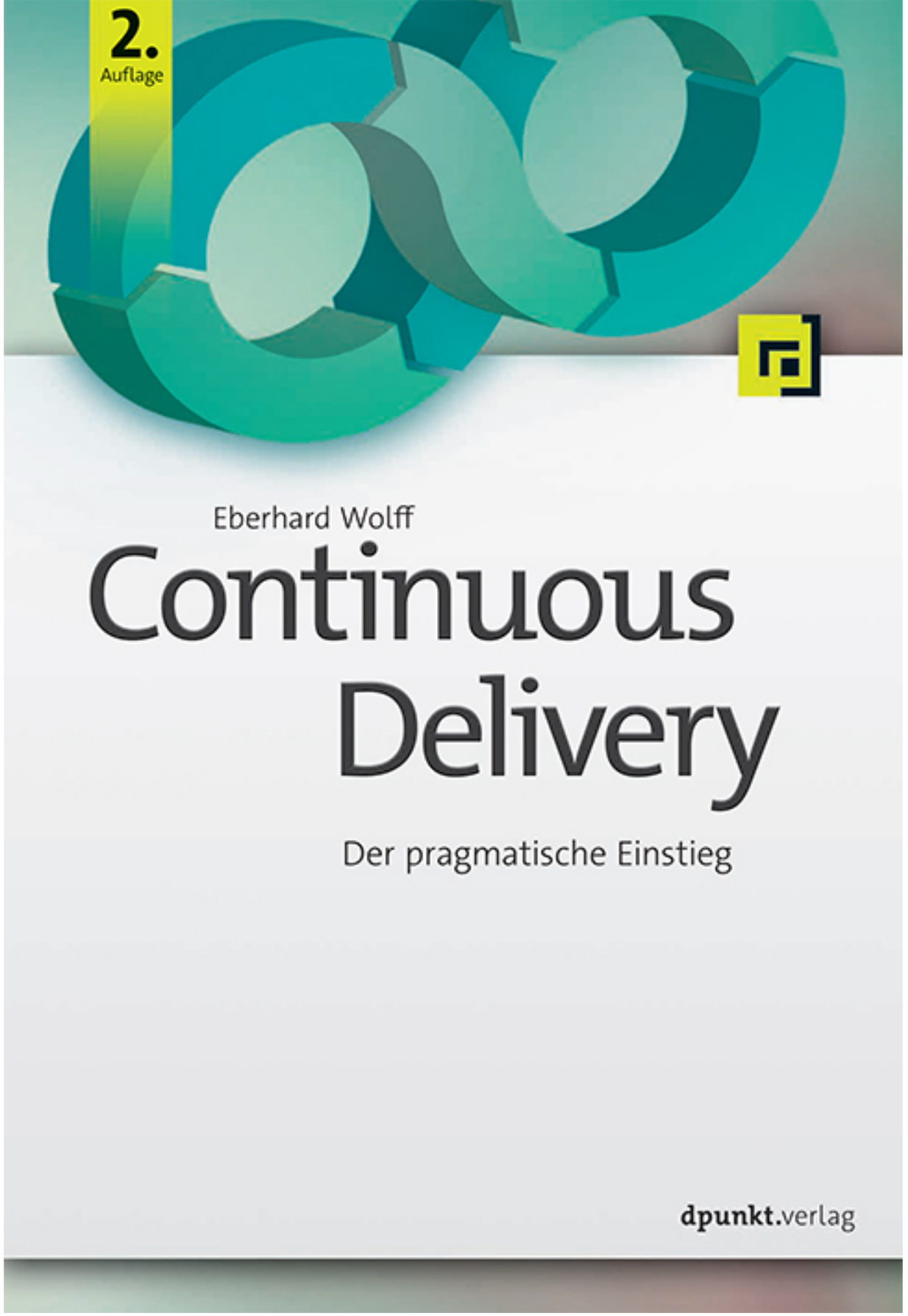
Data Architecture – Not Just for Microservices

Eberhard Wolff
@ewolff
Fellow



http://continuous-delivery-buch.de/

Eberhard Wolff

Microservices

Grundlagen flexibler Softwarearchitekturen

dpunkt.verlag

http://microservices-buch.de/

Microservices

Flexible Software Architectures

Eberhard Wolff

http://microservices-book.com/

Eberhard Wolff

Microservices Primer

A Short Overview

FREE!!!!

innoQ

http://microservices-book.com/primer.html

Classic Data Architecture

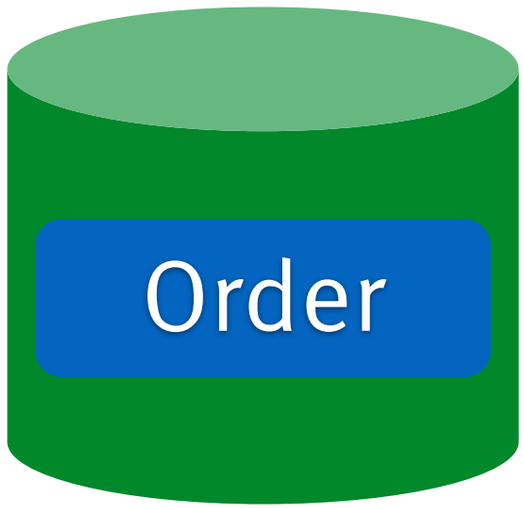
- Centralized databases
- ...or services that provide data
- Ensures consistency across systems
- ...for data model
- ...and updates to data
- Reuse

Classic Data Architecture

Billing

Order Process

CRM



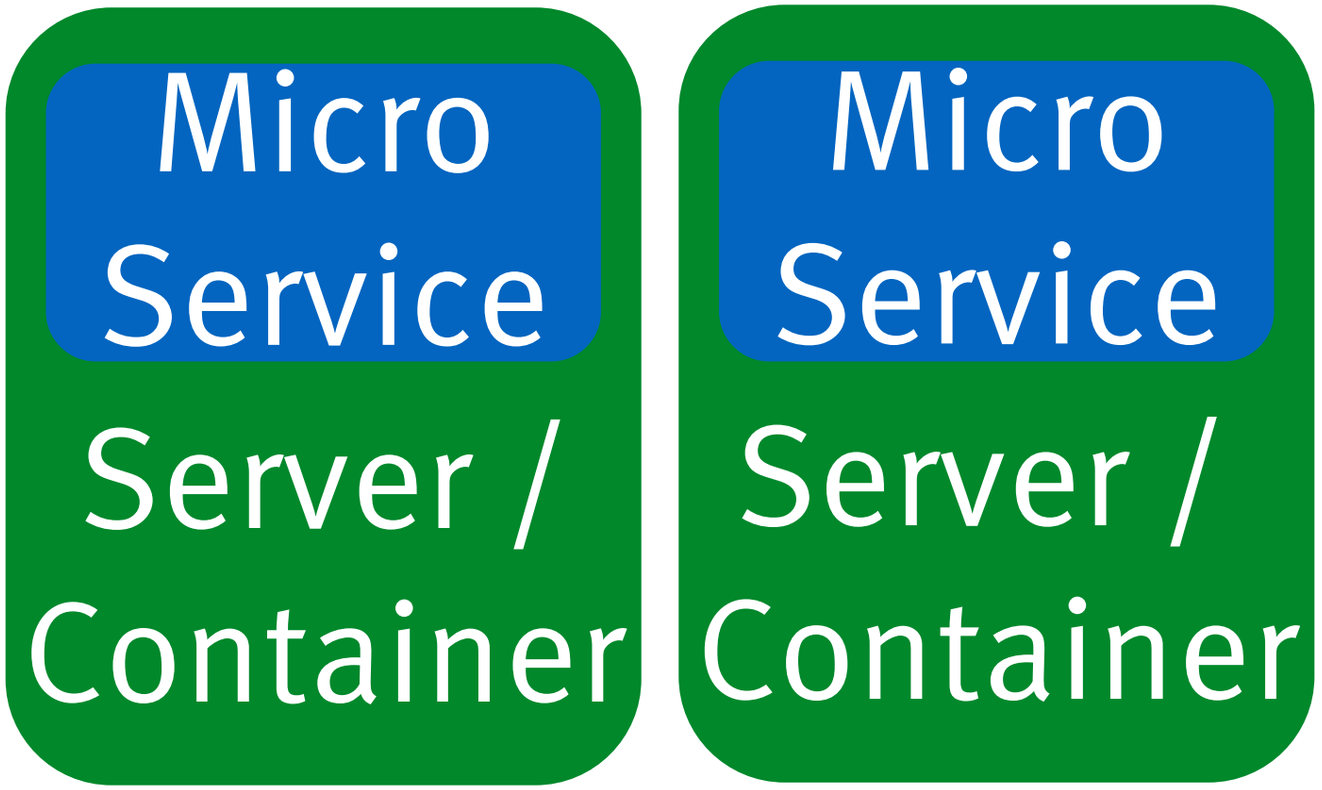
Who is using a centralized database?

Who likes the centralized database?

Microservices: Definition

- No consistent definition
- Microservices are modules
- Independent deployment units
- E.g. processes, Docker container
- Microservice owned by one team

Microservices: Definition



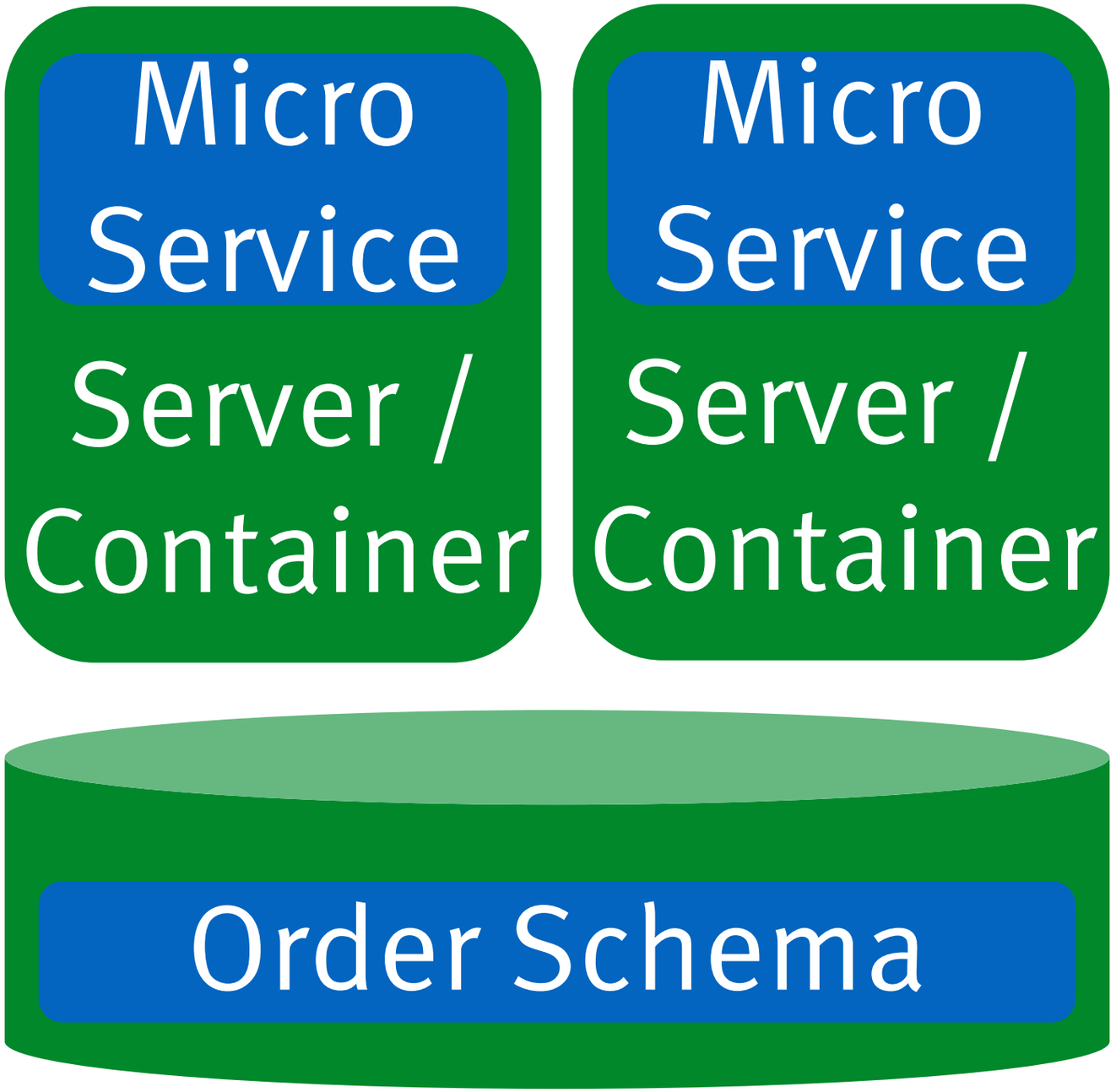
Why Microservices?

- Develop a feature
- ...bring it into production
- ...with no coordination

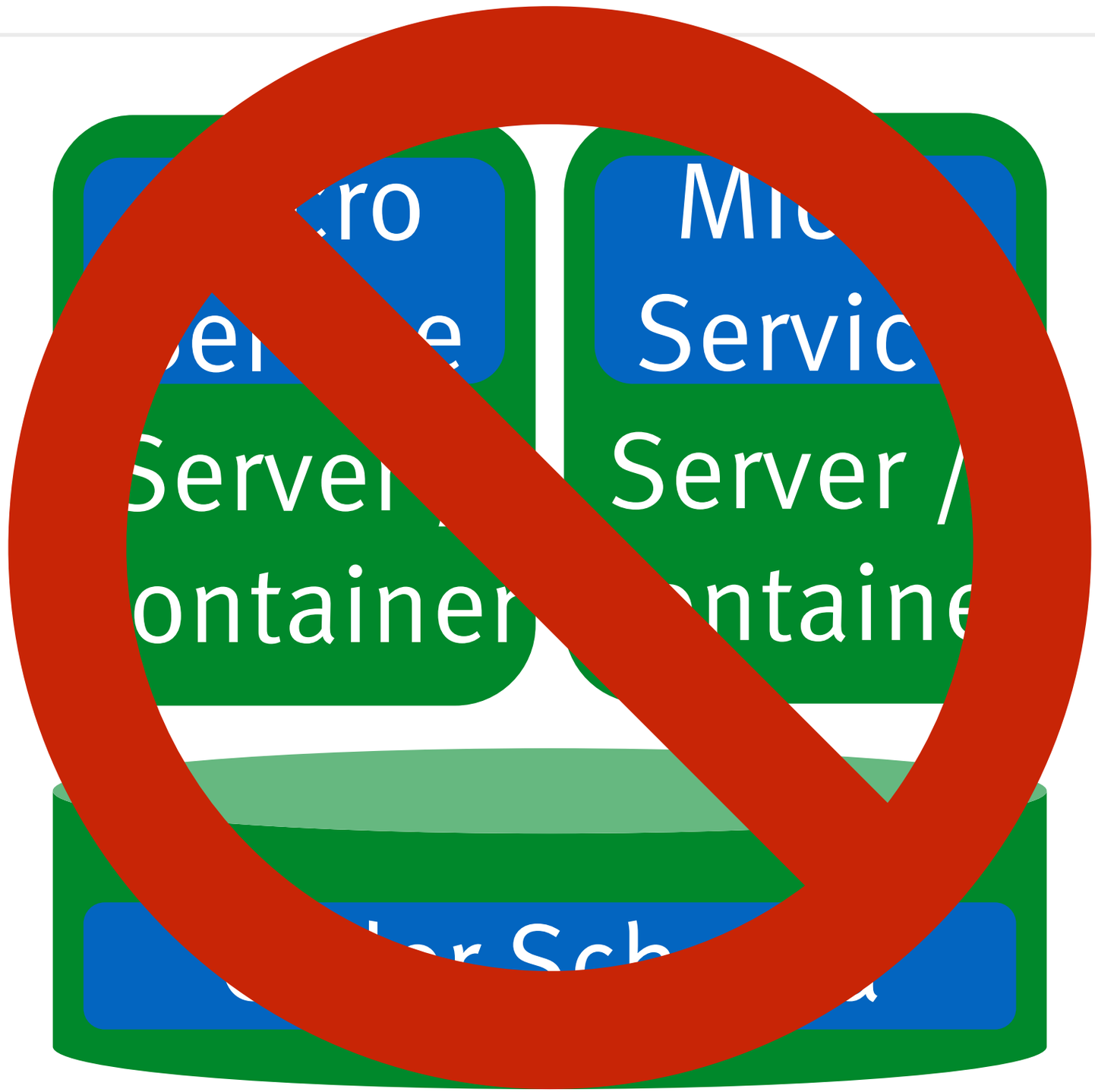
- Independent scaling
- Free choice of technology
- Robustness
- Security

Microservices aim for decoupling

Microservices & Data



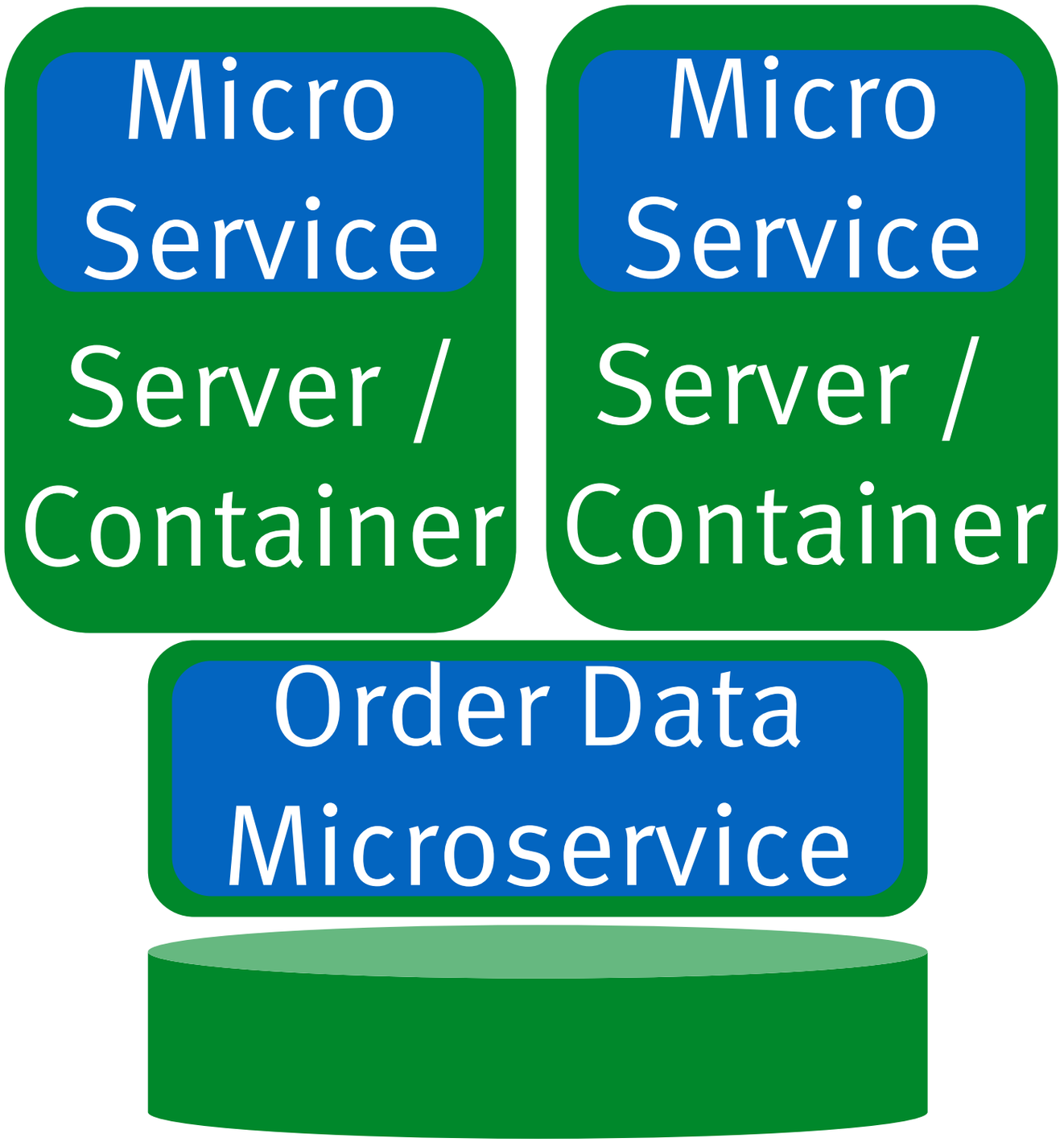
Microservices & Data



Microservices & Data

- Decoupling for data, too
- Separate data storage

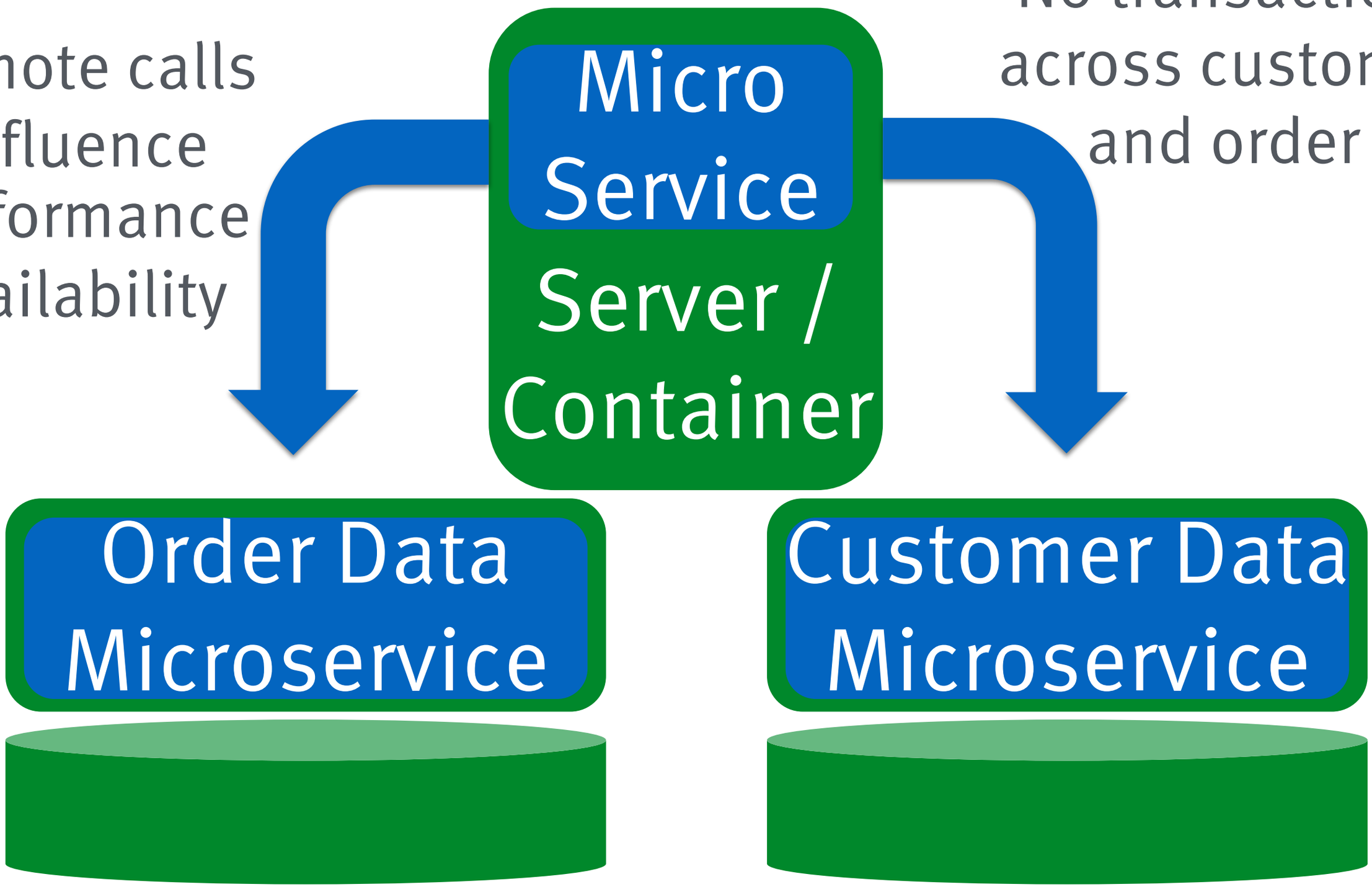
Data Microservices



Data Microservices

Remote calls influence performance availability

No transaction across customer and order



Data Microservice

- Change two microservices if new feature requires change to data schema
- But: data in one place
- No consistency issues

Data microservice limits decoupling.

Encapsulation

- Information hiding
- Hide the internal data structure
- Provide access only through a well defined interface
- Data and databases should not be exported

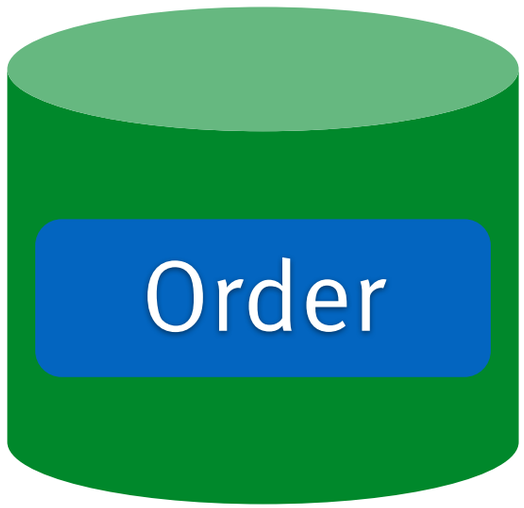
Violates Encapsulation

Billing

Order Process

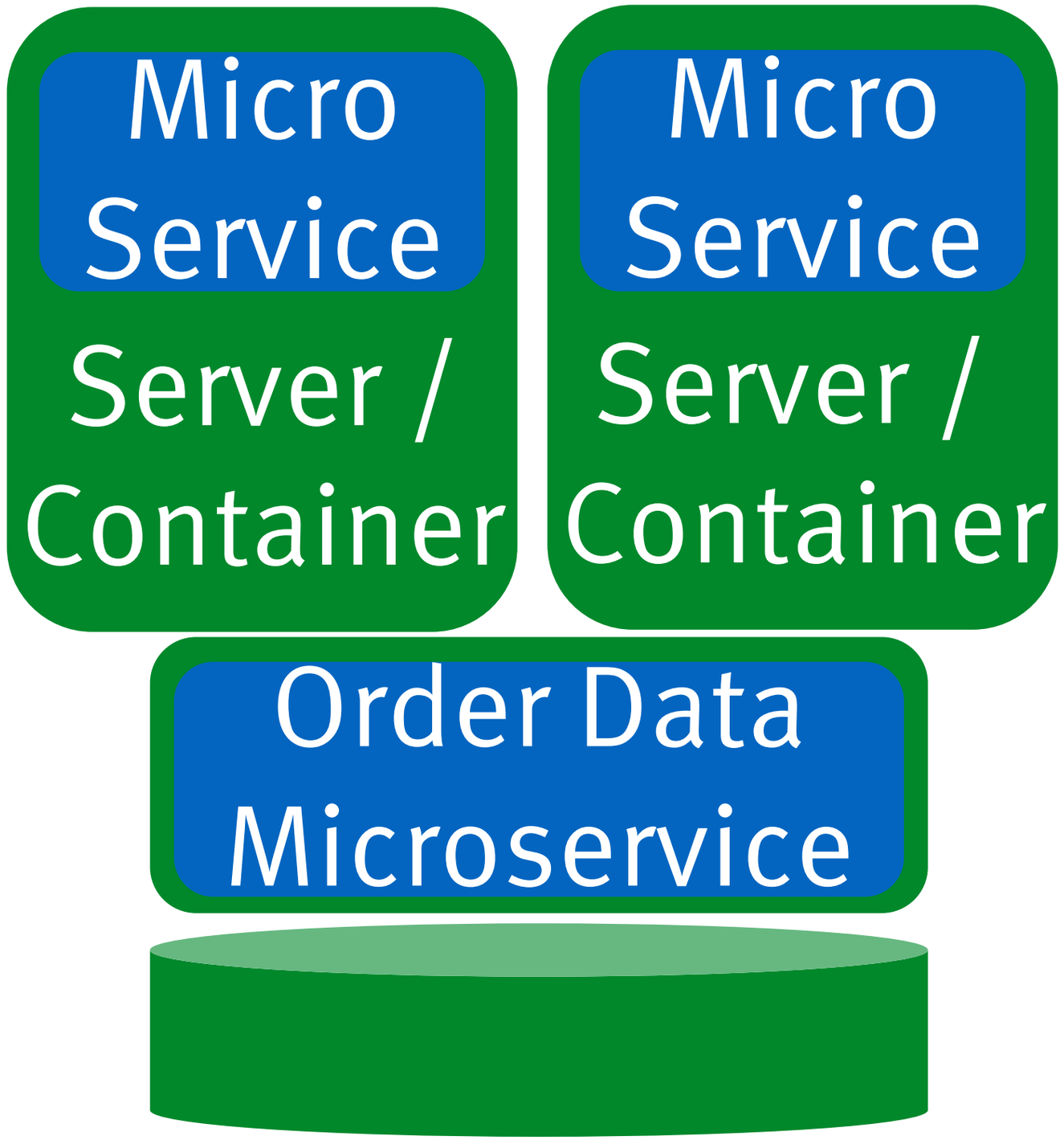
Logic

CRM



Shared data model

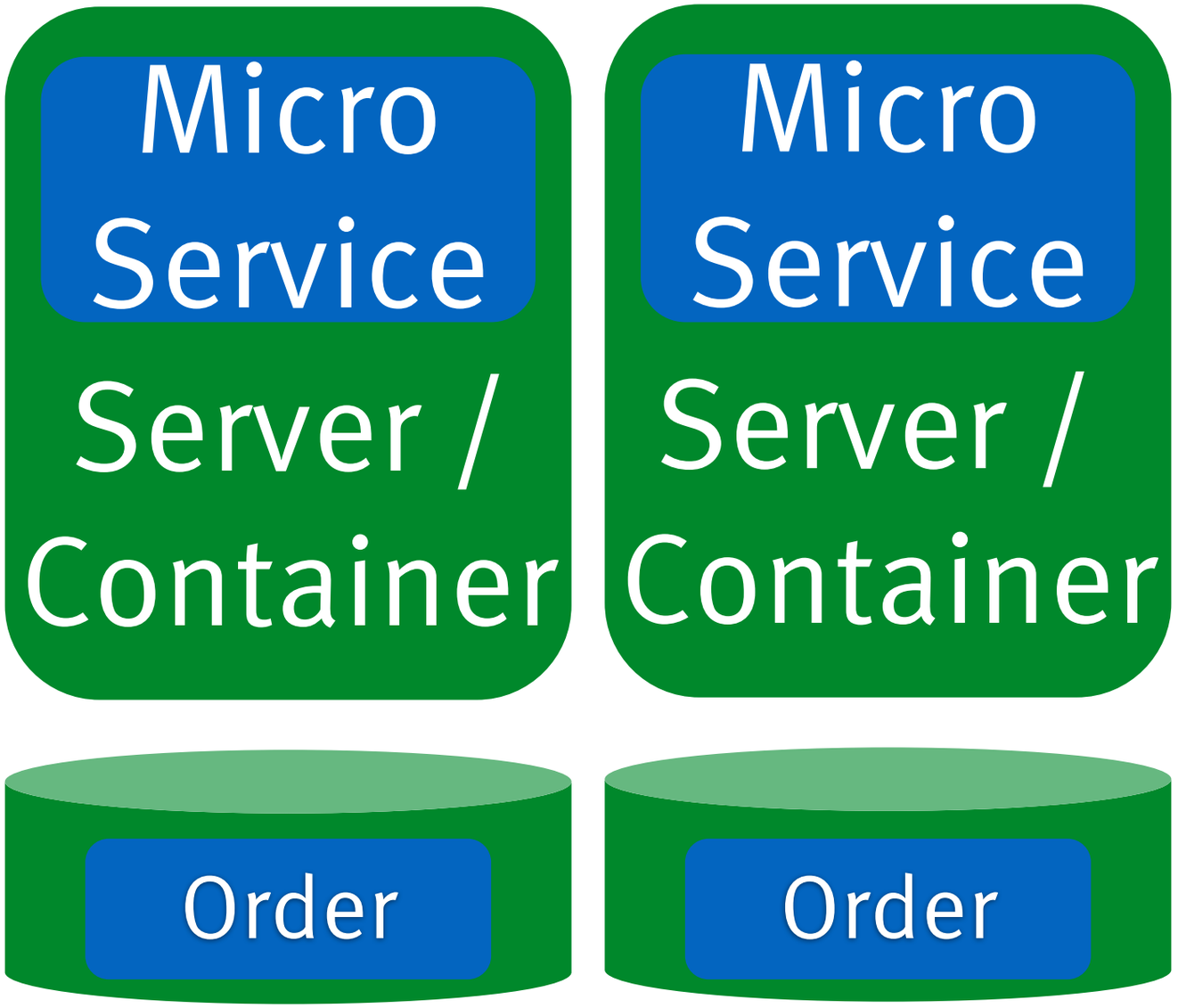
Violates Encapsulation



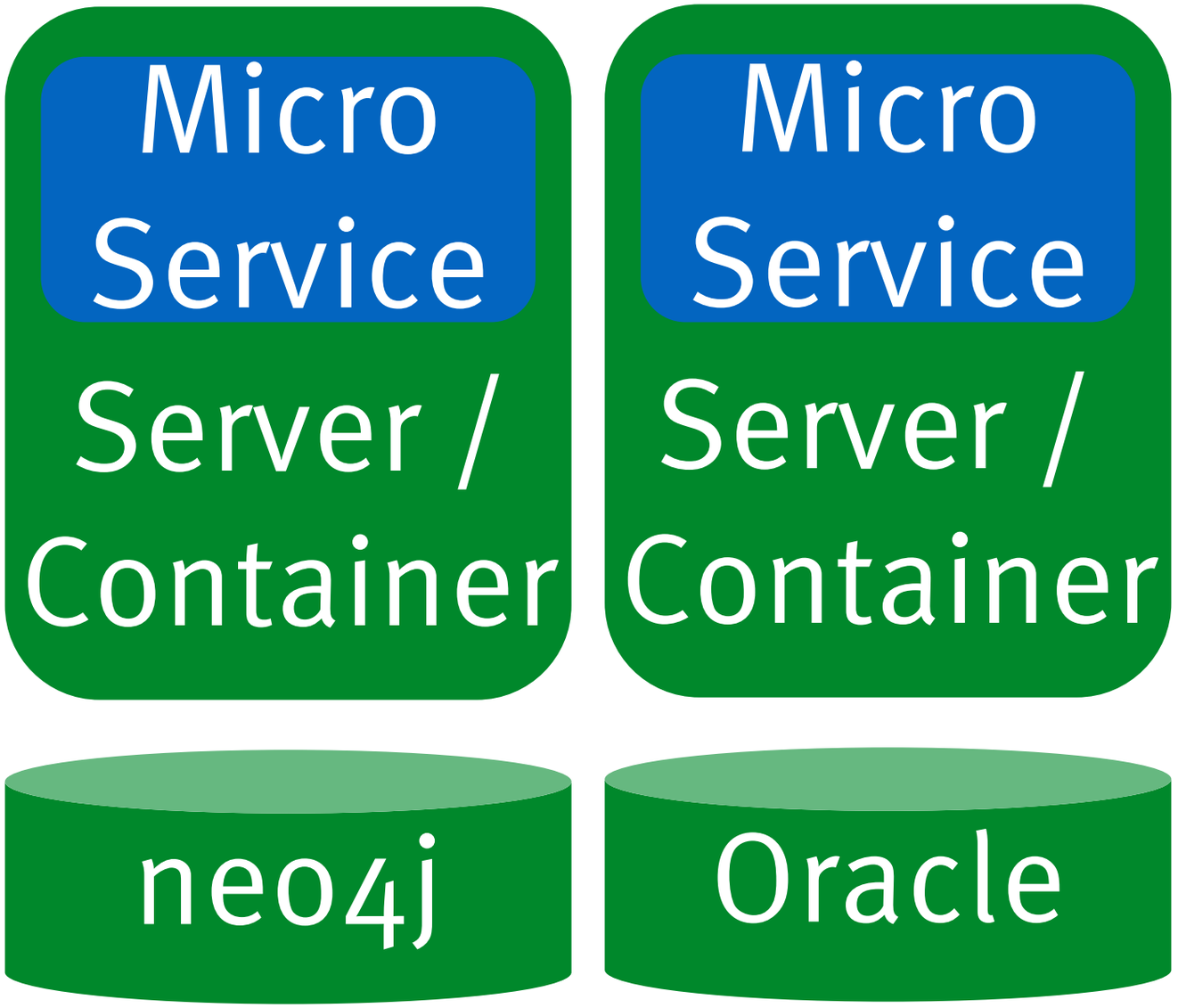
Logic

Shared data model

Separate Databases



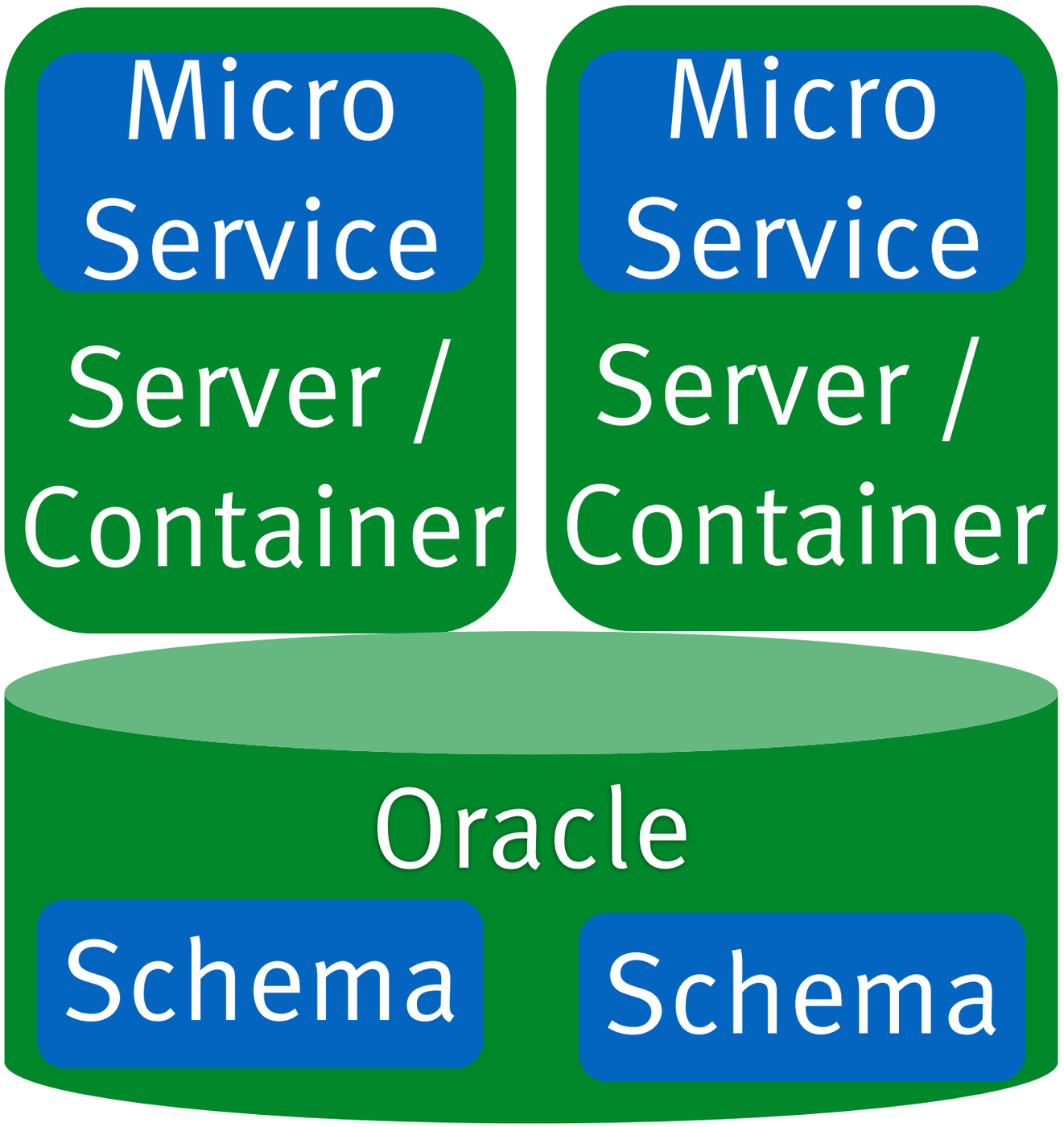
Different Databases



Different Databases

- “Polyglot persistence”
- Use the best tool for the job
- Technology freedom
  - advantage of microservices
- ...but extra effort
- Backup, disaster recovery etc.
- Not as easy as e.g. different frameworks

Separate Schema



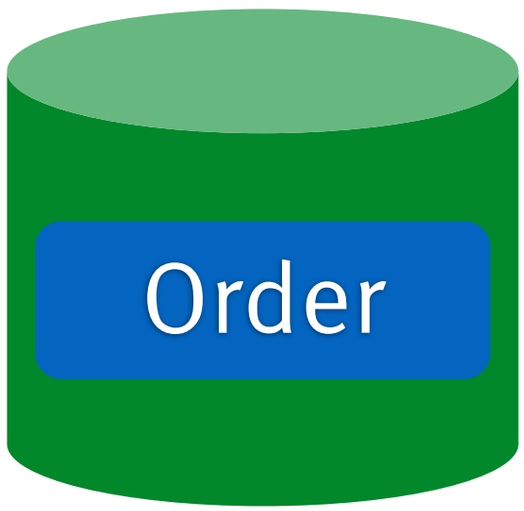
Separate Schemas

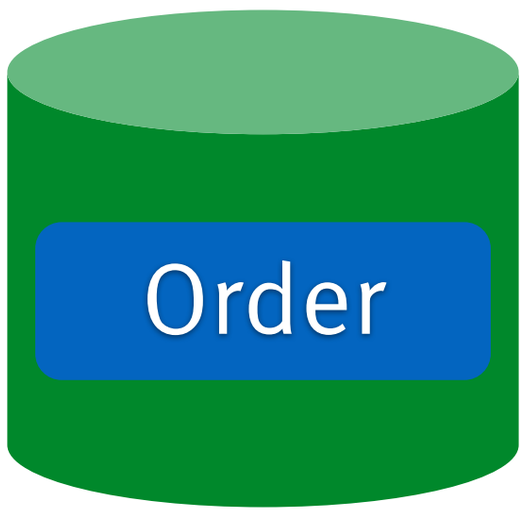
- Less effort
- Decoupled data models
- ...but limited independent scaling and robustness

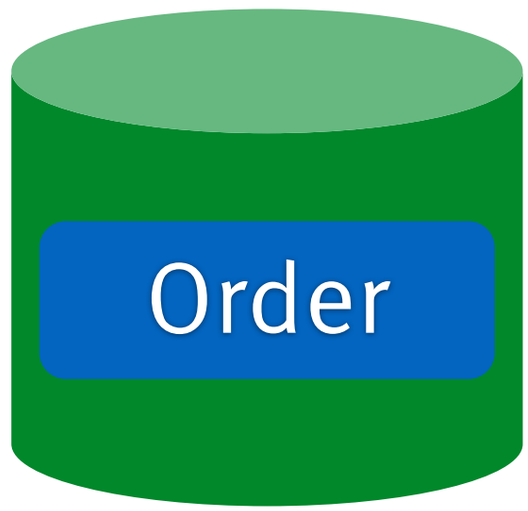
Billing

Order Process

CRM







Redundancy!!!

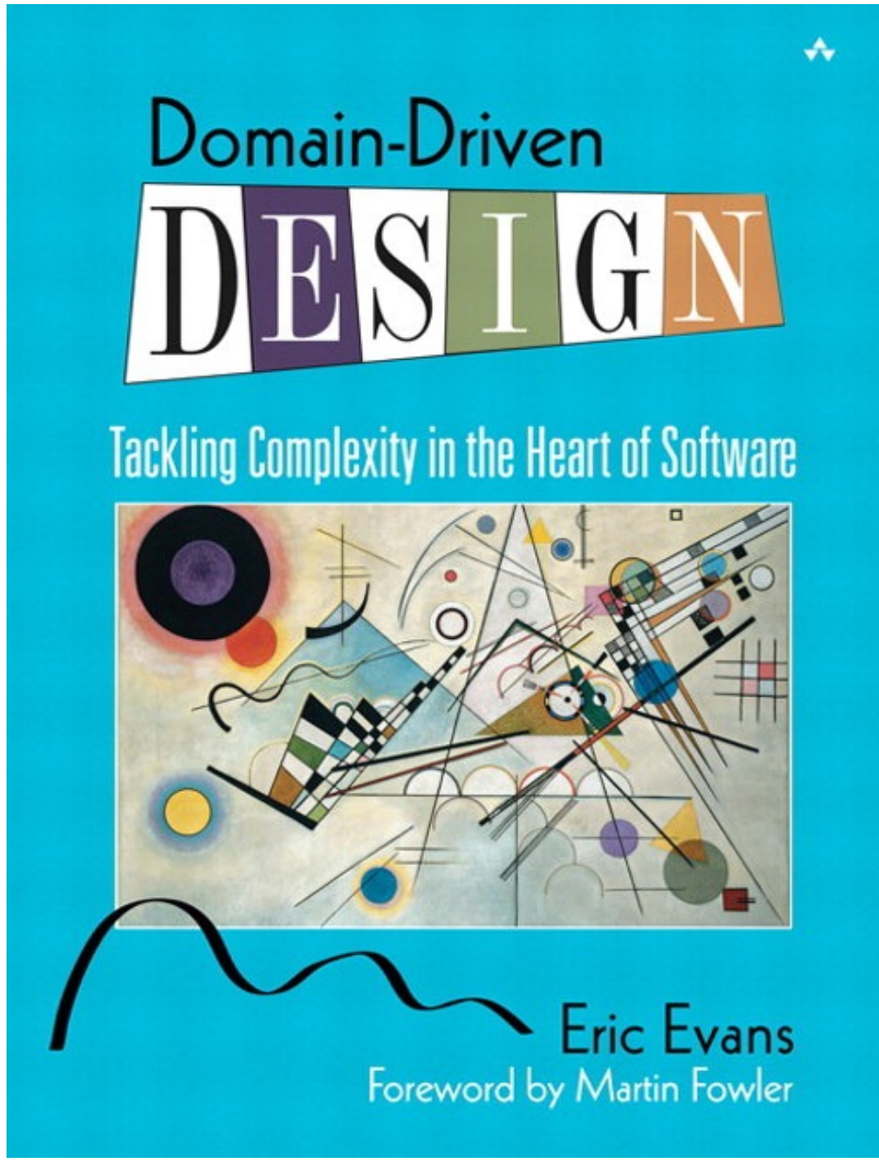
THE END IS NEAR


WHAT IF I TOLD YOU
THERE IS NO REDUNDANT DATA HERE?
memegenerator.net

Domain-driven Design

Domain-driven Design

- 2004
- Still very relevant
- By Eric Evans
- Focus on part IV
- Free reference: http://domainlanguage.com/ddd/reference/



Order

Order #

Shipping address

Tracking #

Items

Item Categories

Priority shipping

Customs #

Account #

Credit card #

…

My Domain Model is a mess!

Bounded Context

- Domain model is only valid for one context
- There is no universal data model!
- See all failed SOA attempts

Tracking

Order

Shipping address

Tracking #

Priority shipping

Order

Order #

Shipping address

Tracking #

Items

Item Categories

Priority shipping

Customs #

Account #

Credit card #

…

Recommen-dations

Order

Item Categories

Customs

Order

Customs #

Payment

Order

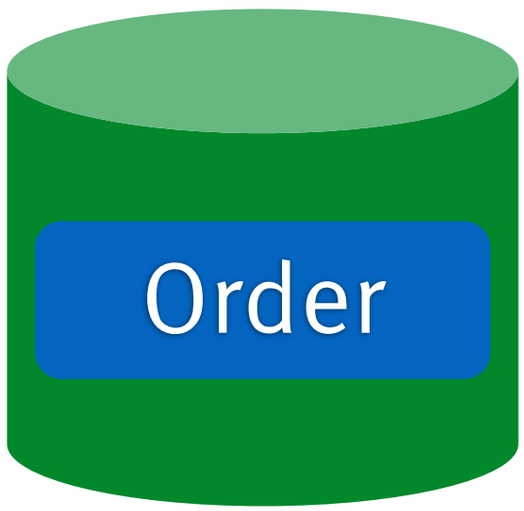
Account #

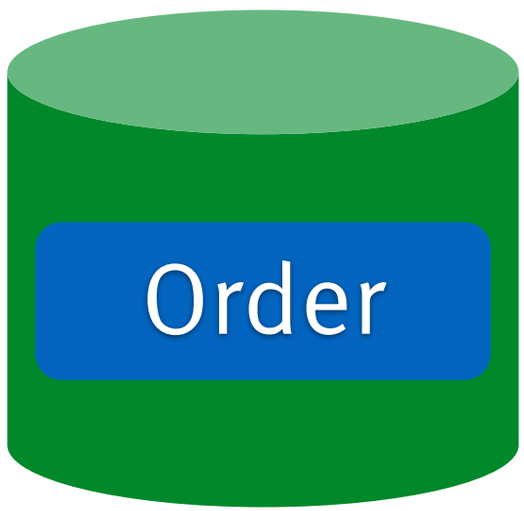
Credit card #

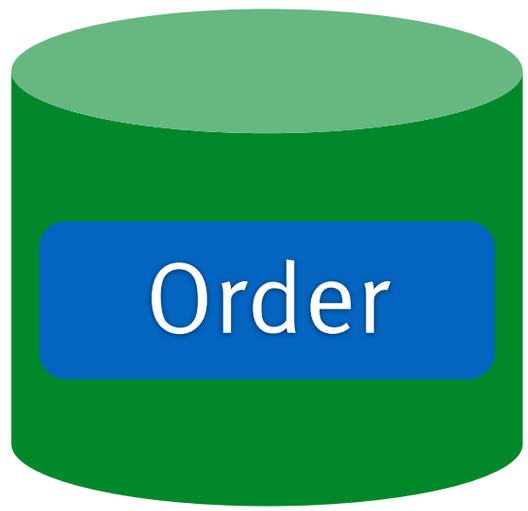
Billing

Order Process

CRM







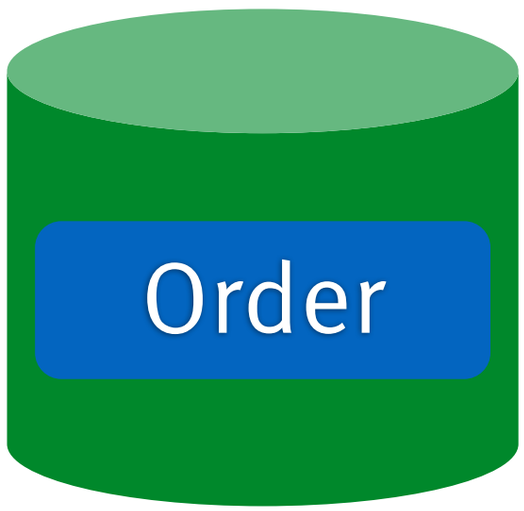
Bounded Context

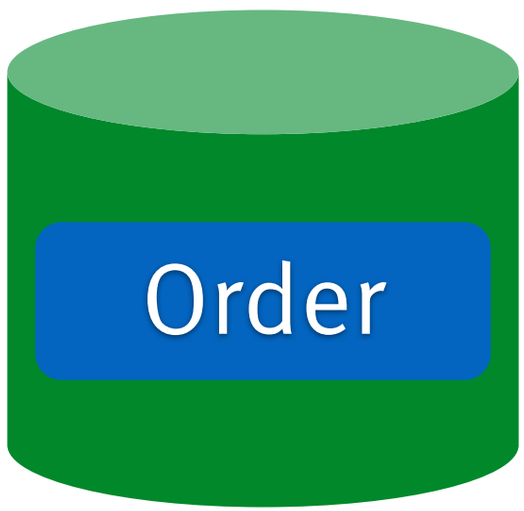
- Microservice = Bounded Contexts
- Changes for new features are local
- ...even if data models need to be changed

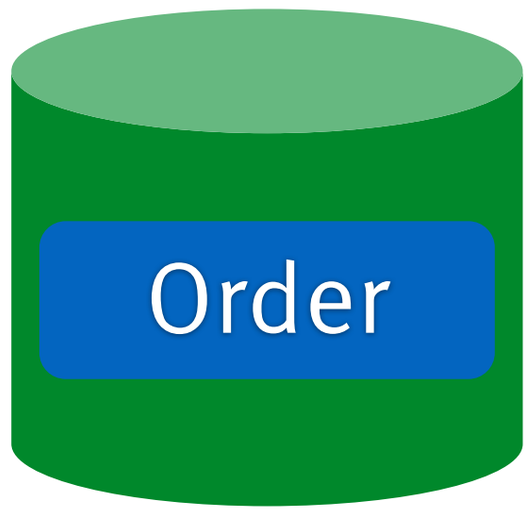
Billing

Order Process

CRM







Redundancy?

Redundancy?

Not really

Bounded Context

What about basic data of an order?

Strategic Design

- How do BOUNDED CONTEXTS relate to each other?
- Context can have relationships
- DDD defines several relationship patterns

Shared Kernel

- Subset of a model
- ...that two teams share
- Eric Evans: Including code and database
- Microservices: Just sharing a model

Anti-corruption Layer

- Don’t let e.g. a legacy model influence a new model
- Isolate model by additional layer
- No need to modify the old system

Context Relationships

- Team = Deployment Unit = Bounded Context
- Context Relationships define how Bounded Context are used...
- ...and how much teams need to collaborate

Coordination Effort

Shared BOUNDED CONTEXT

SHARED KERNEL

CUSTOMER / SUPPLIER

ANTICORRUPTION LAYER

CONFORMIST

SEPARATE WAYS

Context Map

Context Map

- Show the different Bounded Context
- ...and the relation to each other
- Bounded Context might be microservices
- ...or communication links

Registration

Basic Customer Data

Basic Customer Data

Order Process

Customer Order Data

Customer Order Data

Customer Order Data

Anticorruption Layer

Delivery

Customer Order Data

Billing

Mainframe Customer Data

Basic Customer Data + Customer Order Data = Shared Kernel

Billing

Additional data

Order Process

Additional data

CRM

Additional data

Order Data

Shared Kernel Order

Centralized Shared Kernel

- Ensures consistency
- ...but needs to be called for a lot of operations
- Resilience / performance / transactions
- Have one master as the source of truth

Billing

Order Process

CRM

Additional data

Shared Kernel Order

Additional data

Shared Kernel Order

Additional data

Shared Kernel Order

Decentralized Shared Kernel

- Might be inconsistent
- ...but all data for all requests is available in the local database

- Better resilience...
- ...and performance

How to Replicate Data?

Database Replication

- Built into the database
- Replicate schema across database instances

- But: Microservices have separated schemas
- Every Microservice might have different data
- ...so database replication is not a good fit

Replication with Events

Events

- Later addition to Domain-driven Design
- Events with a business meaning
- Decouple time:
  Asynchronous
- Decouple logic:
  System can handle event as it pleases

New Order Event

Billing

Order Process

CRM

Additional data

Shared Kernel Order

Additional data

Shared Kernel Order

Additional data

Shared Kernel Order

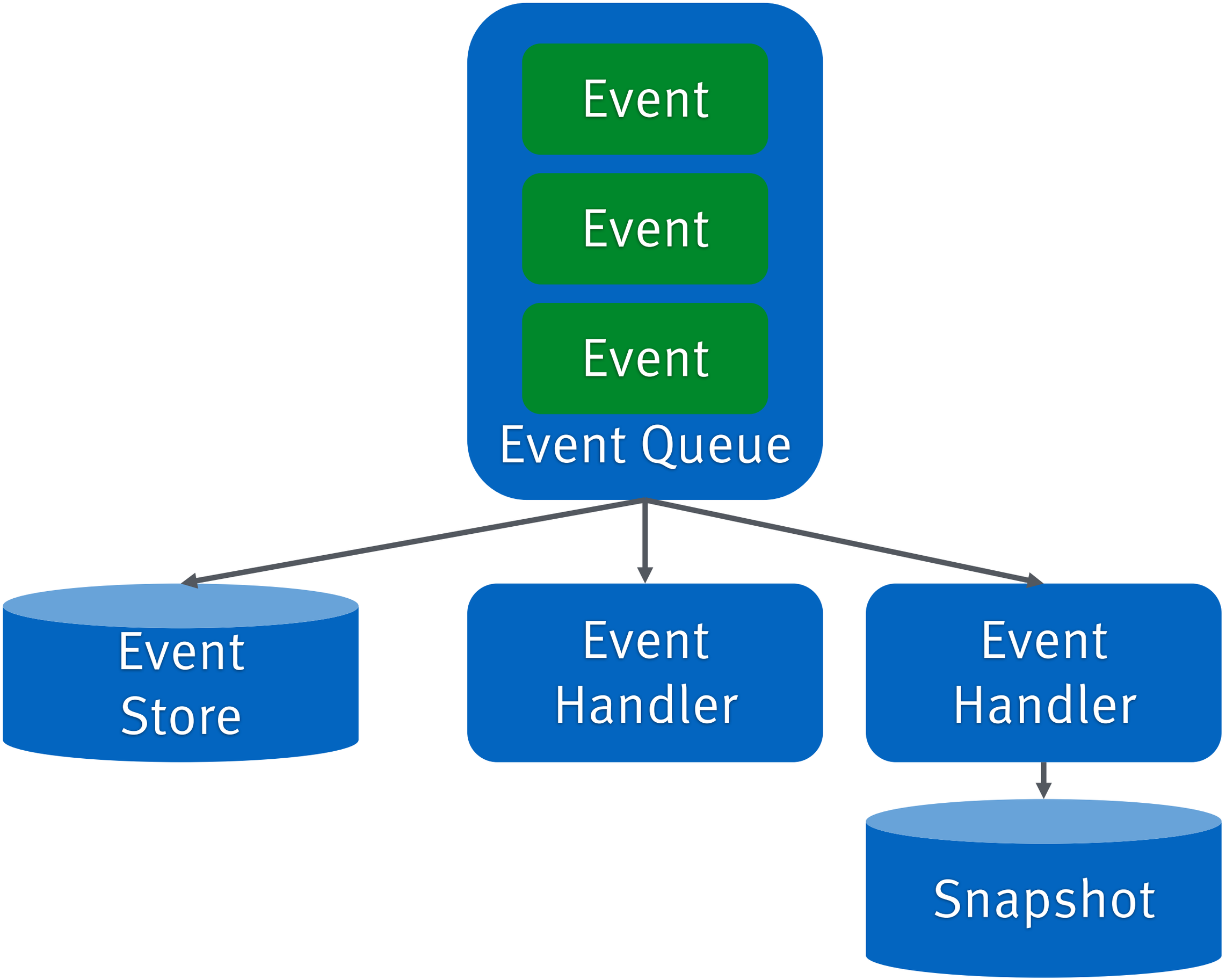
Events & Data Replication

- Events lead to data replication
- i.e. each system stores information it received in an event
- Data stored in separate schema
- Very decoupled
- Hard to repair inconsistencies

More Fun With Events

Event Sourcing

- Internal Structure for Microservice with events
- Current state result of all events
- Calculate state on the fly?


Event
Event
Event
Event Queue
Event Store
Event Handler
Event Handler
Snapshot

Event Sourcing

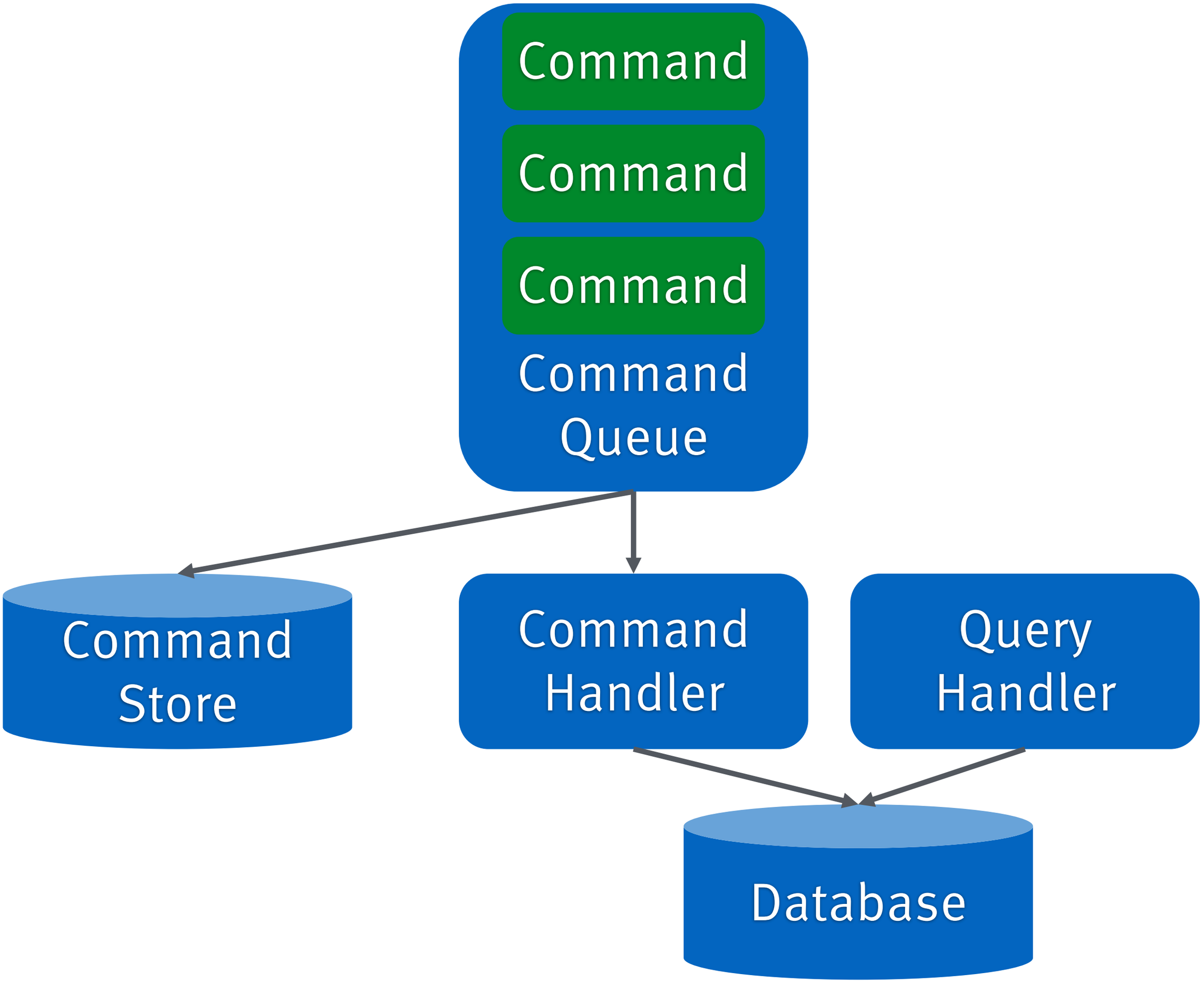
- Event store and snapshot help to repair inconsistencies
- Event-based architecture in microservices

CQRS

- Command – Query Responsibility Segregation
- Commands change data
- Query provide data
- Implement in separate modules
- ...or even microservices
- ...with potentially different Bounded Contexts

Commands vs Events

- Command: Change that data!
- Event: Something has happened
- Component decides if data should be changed


Command
Command
Command
Command Queue
Command Store
Command Handler
Query Handler
Database

Batch Replication

Batch

- Get all data
- Provide API
- ...to decouple schema
- Copy interesting data into local database

Billing

Order Process

CRM

Additional data

Shared Kernel Order

Batch

API

Additional data

Shared Kernel Order

API

Batch

Additional data

Shared Kernel Order

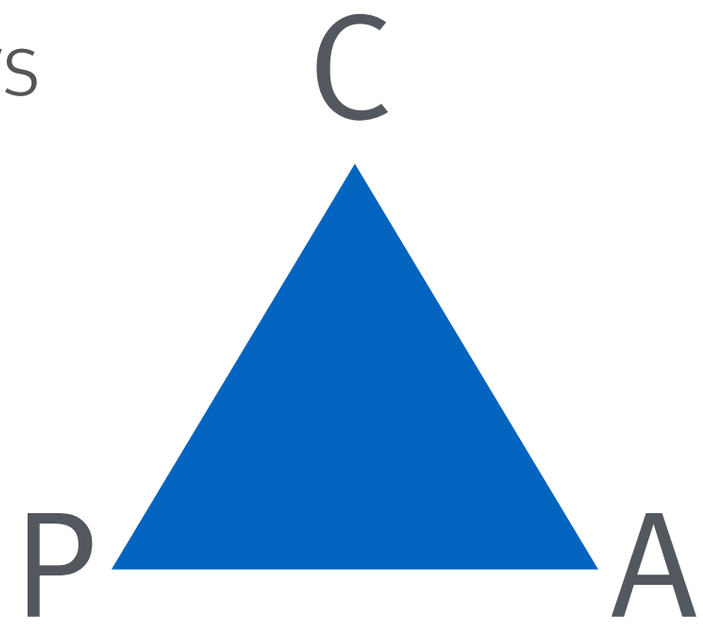
Batch & Data Replication

- Easy to repair inconsistencies
- Batch run at specific points
- i.e. updates take time
- Data not consistent across microservices

CAP: Challenge for Replication

CAP Theorem

- Consistency
  - All nodes see the same data
- Availability
  - Node failures do not prevent survivors from operating
- Partition Tolerance
  - System continues to operate despite arbitrary message loss



CAP Theorem: P

- Network partitions do occur
- Even with highly available network hardware
- Also: very slow response = partition

- Need to deal with P

CAP Theorem: C or A?

- Node cannot access other nodes
- Might have missed updates

- A, not C:
  Answer with a potentially wrong answer
- C, not A:
  Don’t answer – the answer might be wrong

New Order Event

Billing

Order Process

CRM

inconsistent

or unavailable

Additional data

Shared Kernel Order

Additional data

Shared Kernel Order

Additional data

Shared Kernel Order

Conclusion

Classic: Centralized Database

Microservices: private database decoupling

Data Microservices: Consistent but resilience / performance / transactions / decoupling?

Schema per Microservice: Simple infrastructure

Database per Microservice: Polyglot Persistence

Redundancy?

Redundant Data or Bounded Context?

Context Map and Context Relations

Replication

e.g. Shared Kernel

Database Replication

Batch

CQRS

Event Sourcing

Events

CAP

Decentralize data!