31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Event Sourcing in einer Microservice-Architektur

Ein Erfahrungsbericht aus der Werkhalle

Michael Omann

Senacor Technologies AG

Event Sourcing in einer Microservice-Architektur

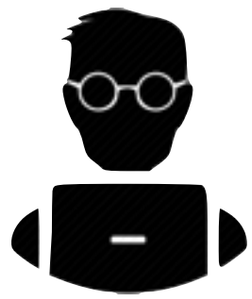
Michael Omann

SENACOR

Michael Omann

Architect bei Senacor

Softwareentwickler!



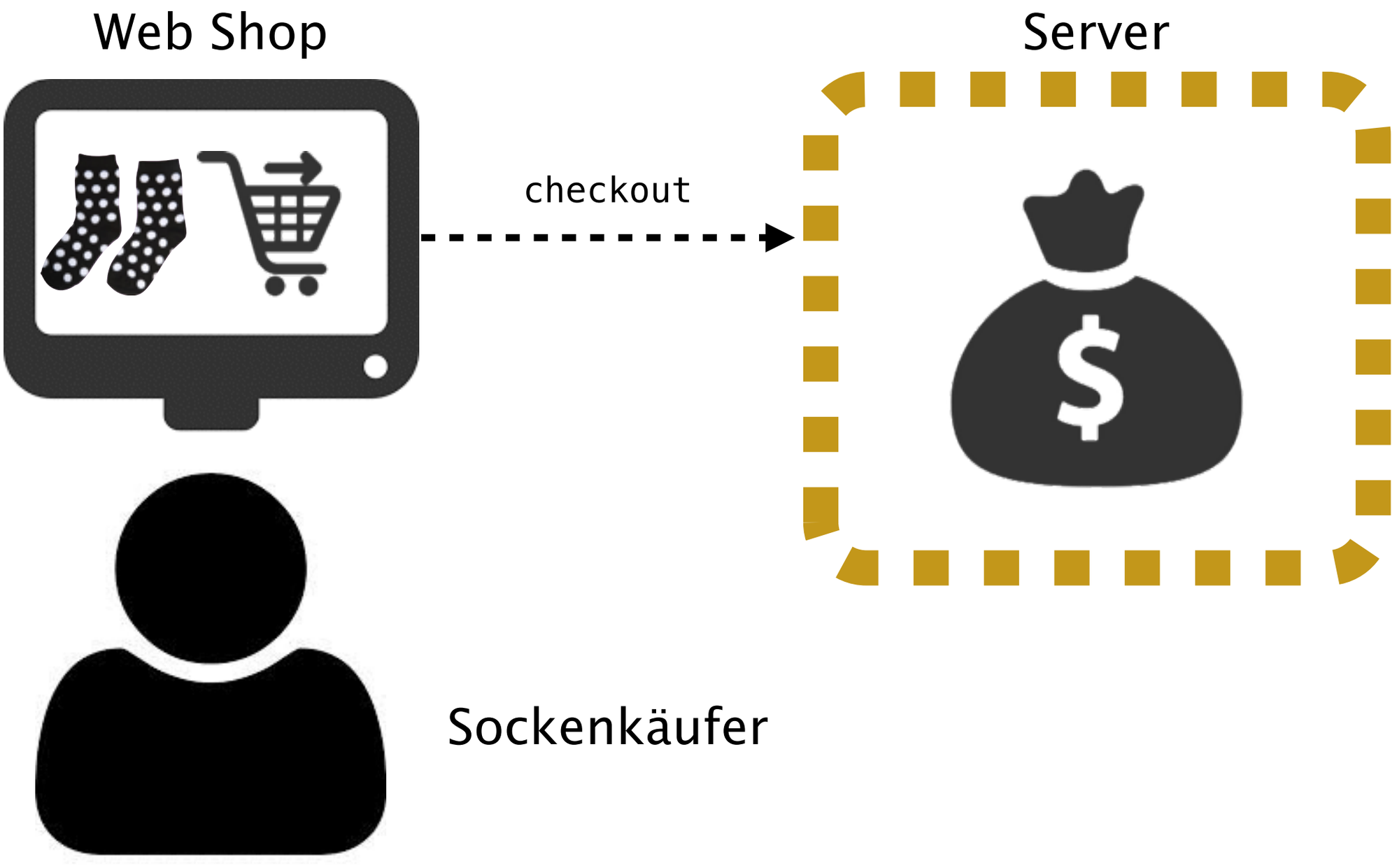
Meine Branchen und Kunden:

No to NoSQL!

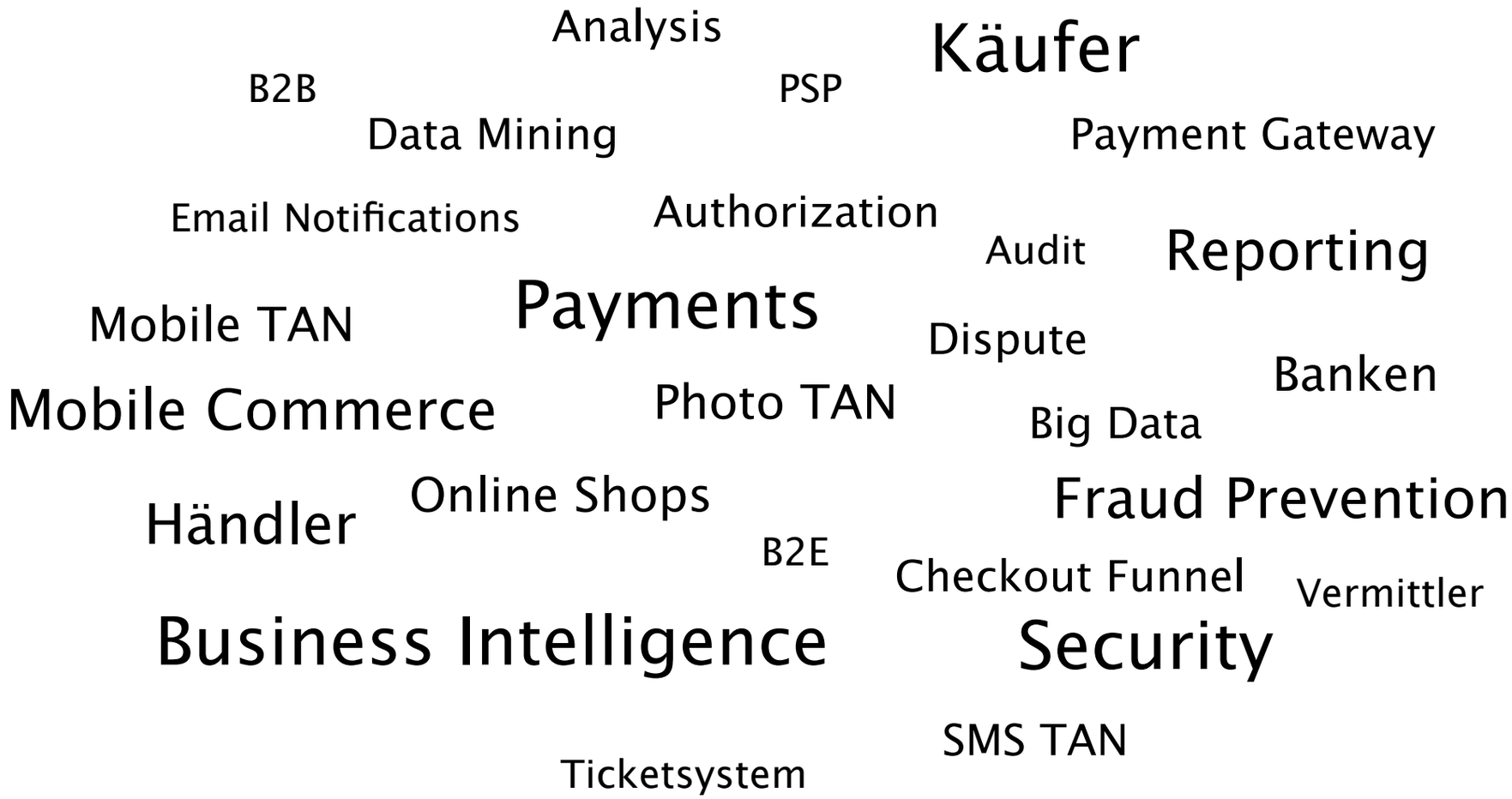
- Retail Banking,
- Commercial Banking,
- Investment Banking
- Banking, Banking, Banking,…

eCommerce Payment Solution

eCommerce Payment Solution?



Anforderungen



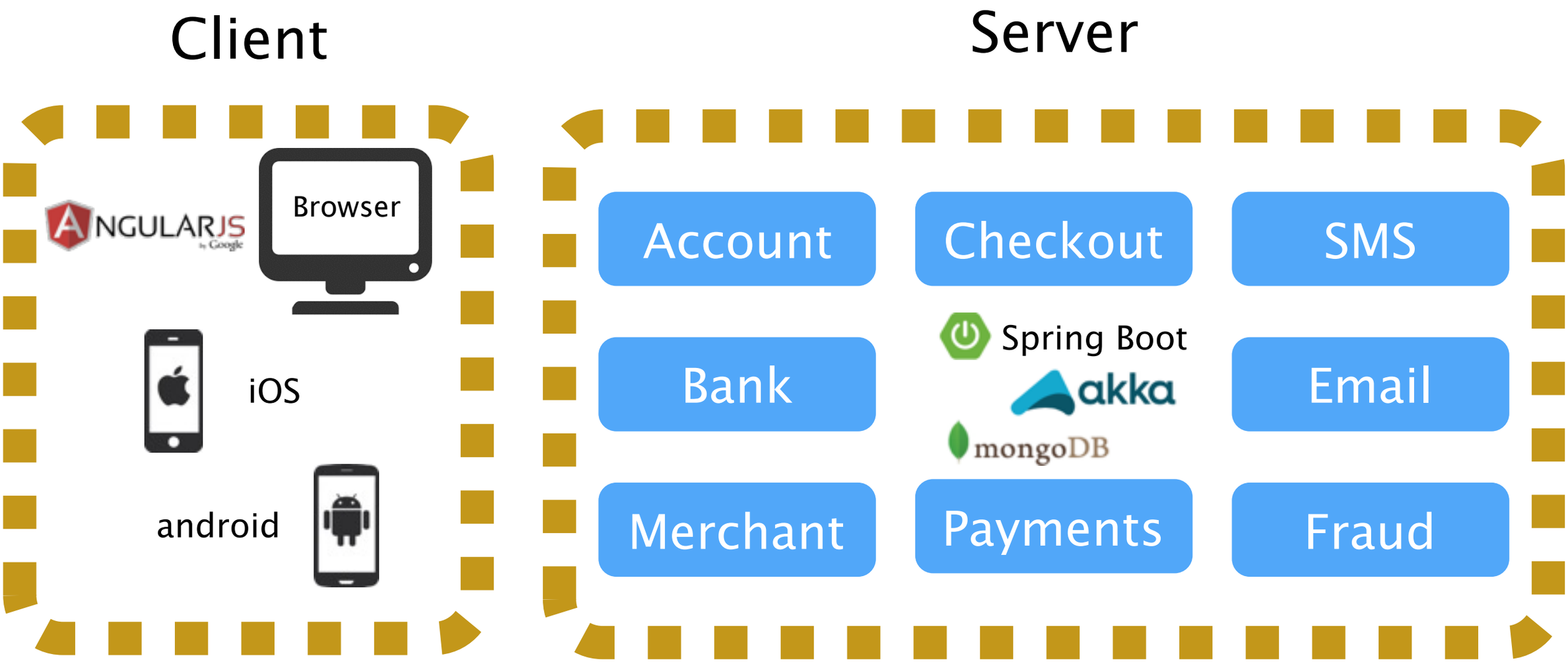
Stammdatenverwaltung

Zahlungssicherheit

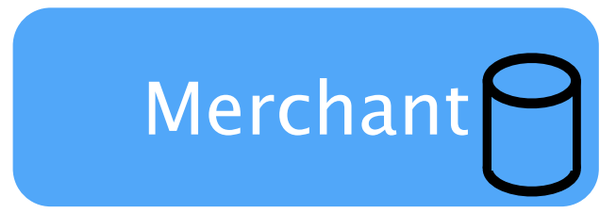
Notifications

Business Intelligence

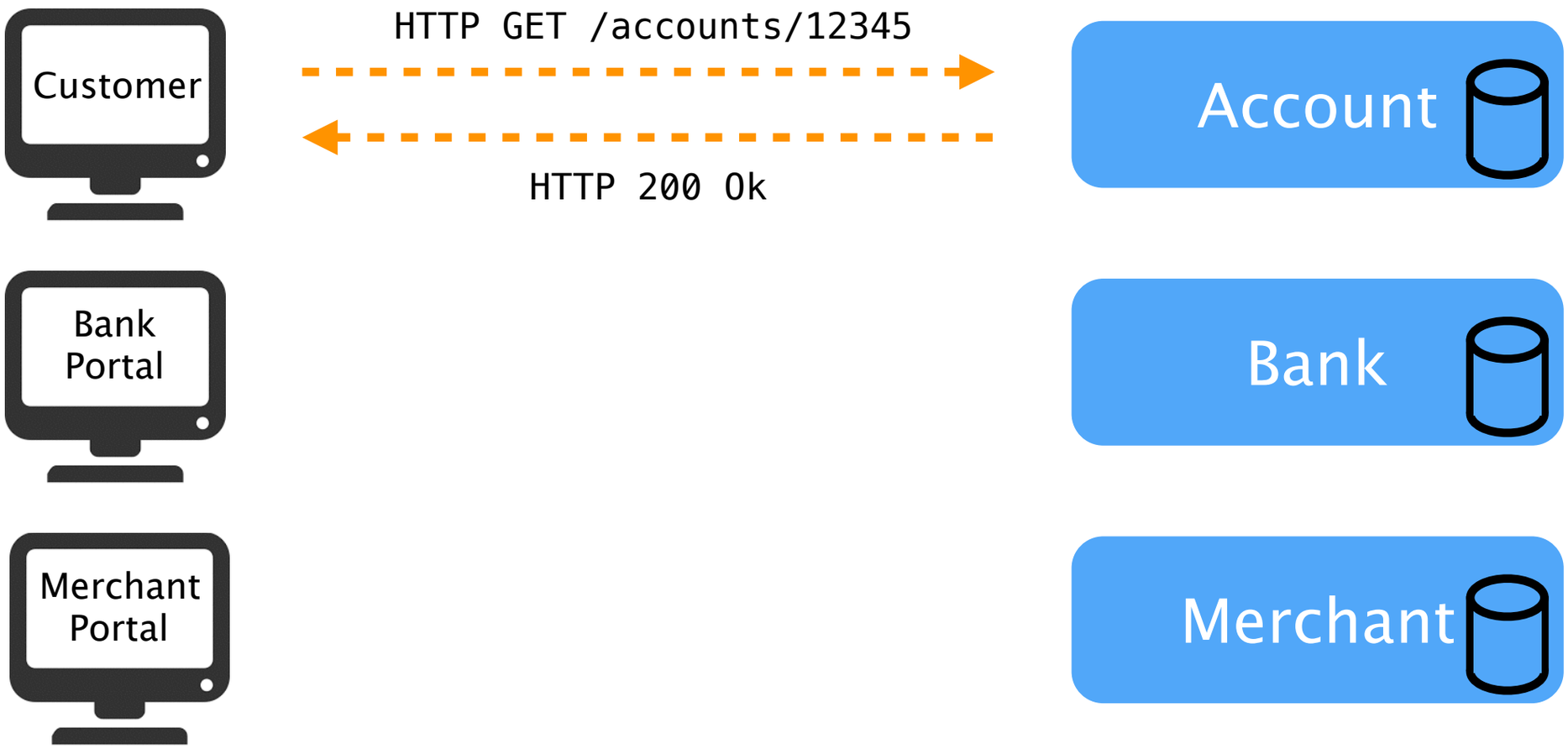
Microservice-Architektur



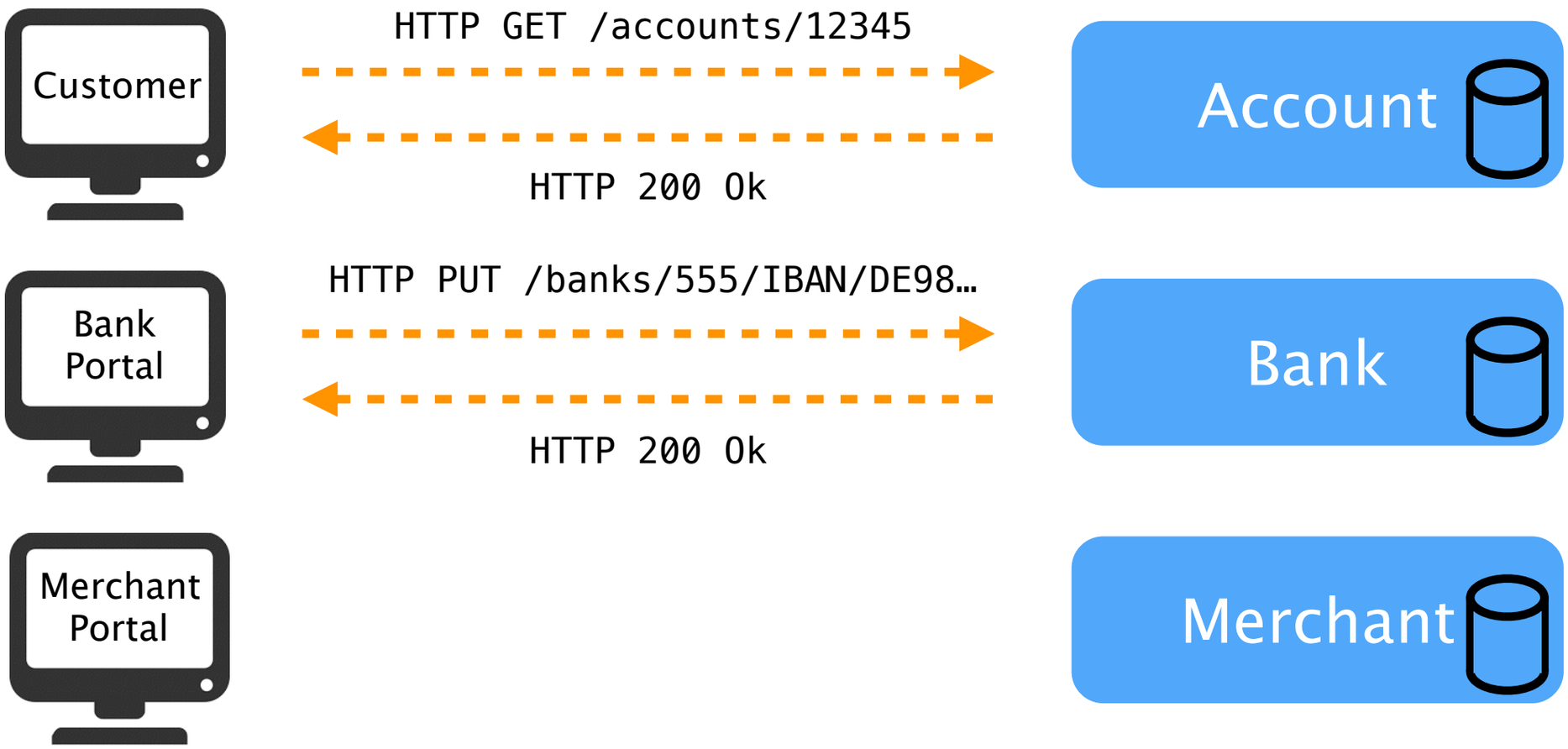
Stammdatenverwaltung



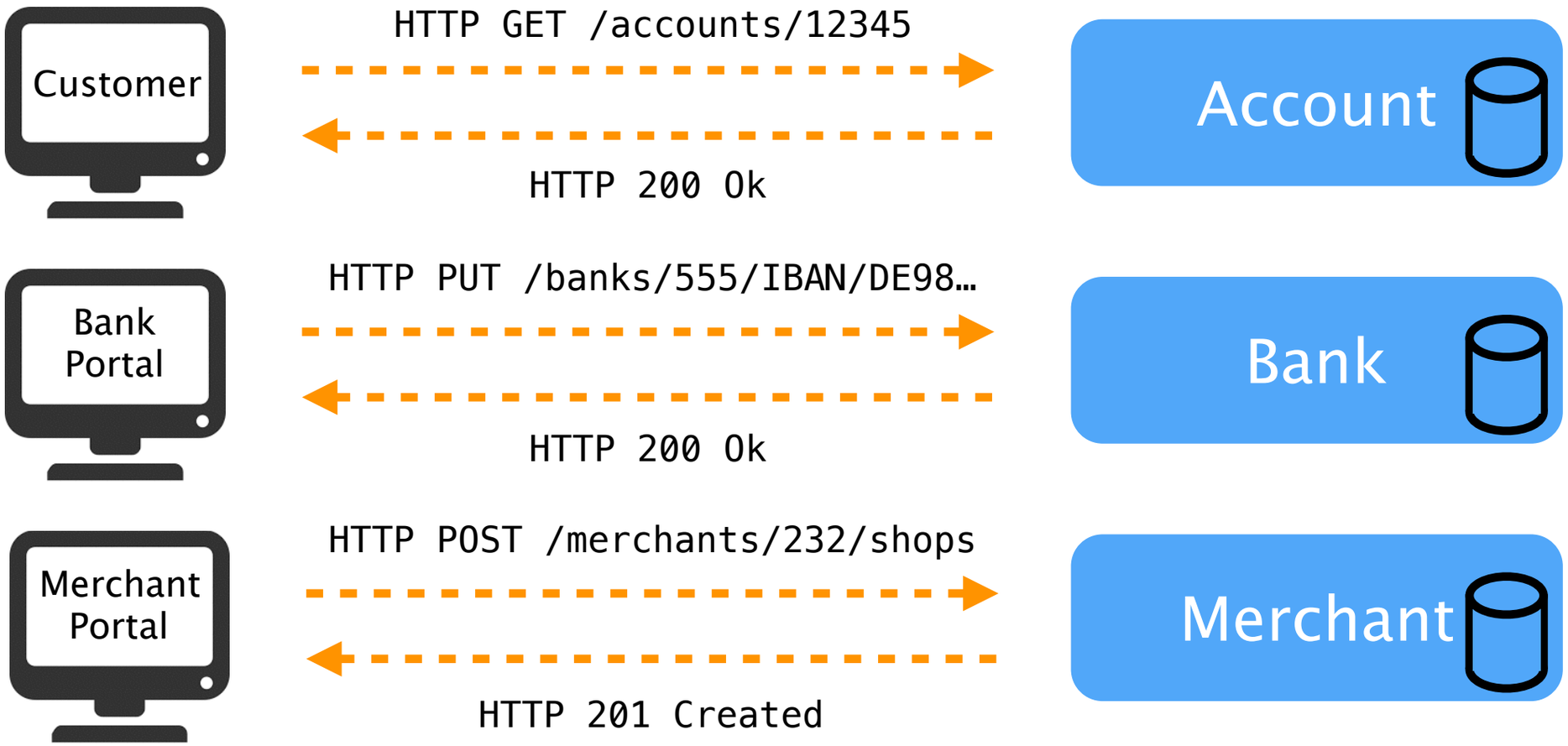
Stammdatenverwaltung



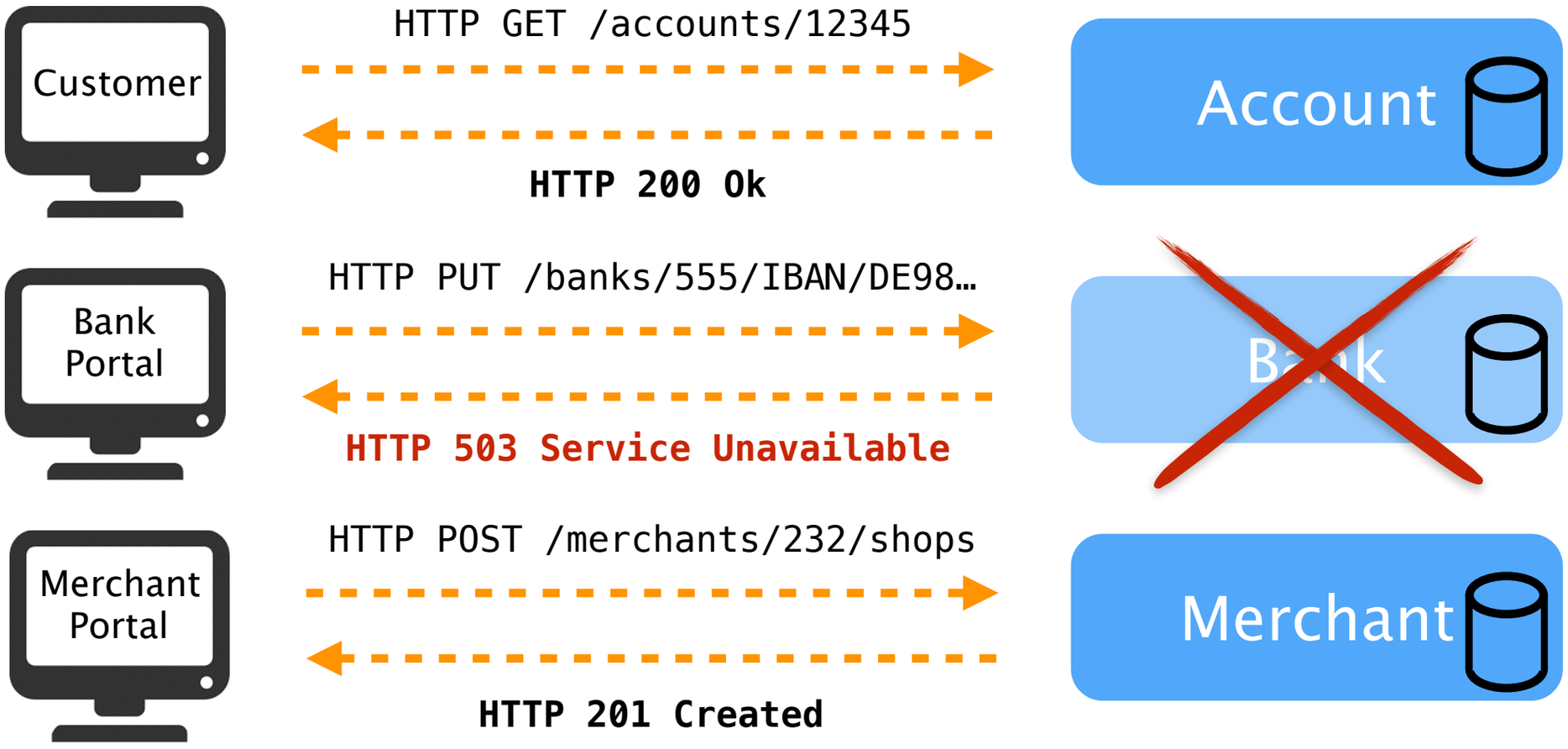
Stammdatenverwaltung



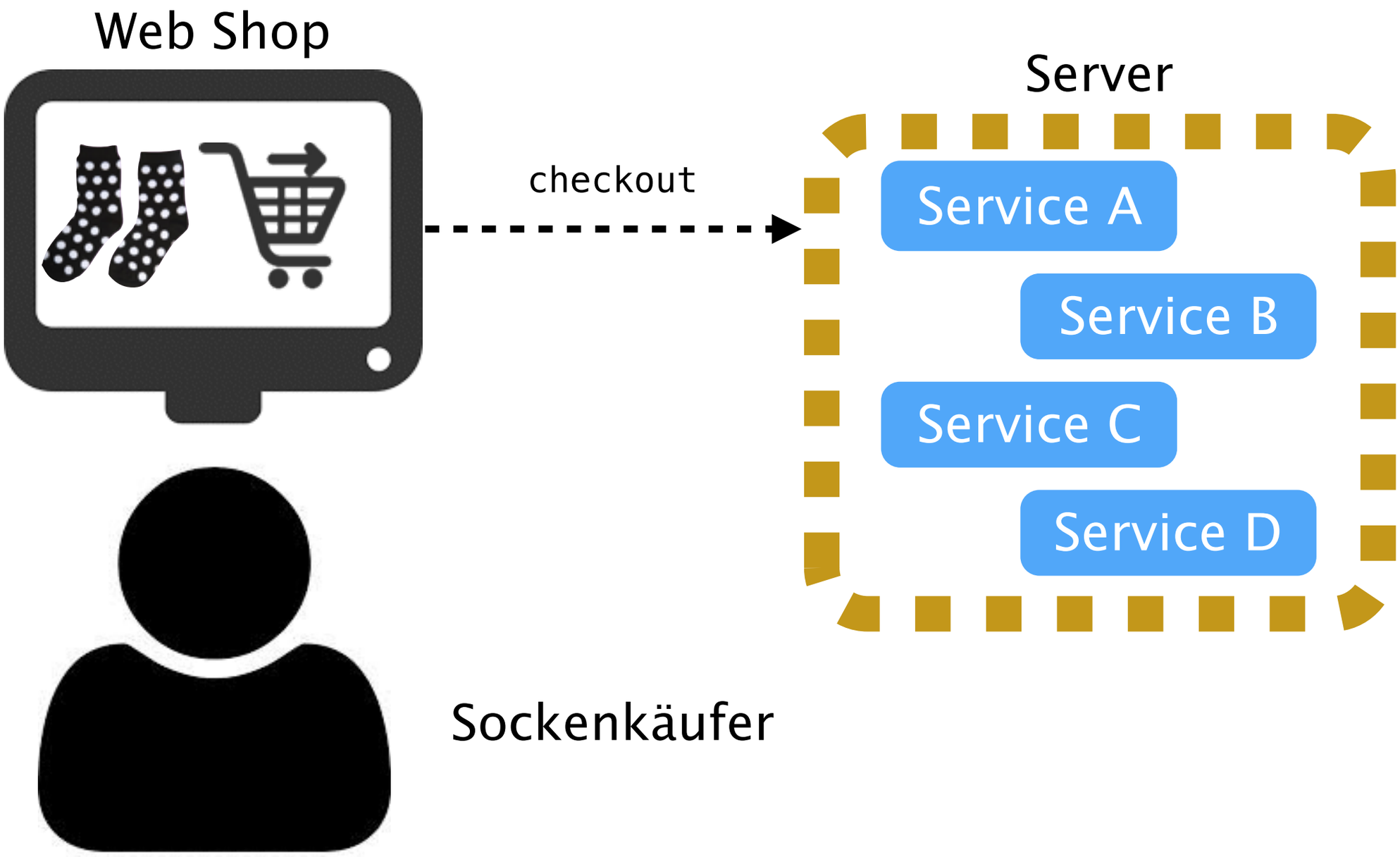
Stammdatenverwaltung



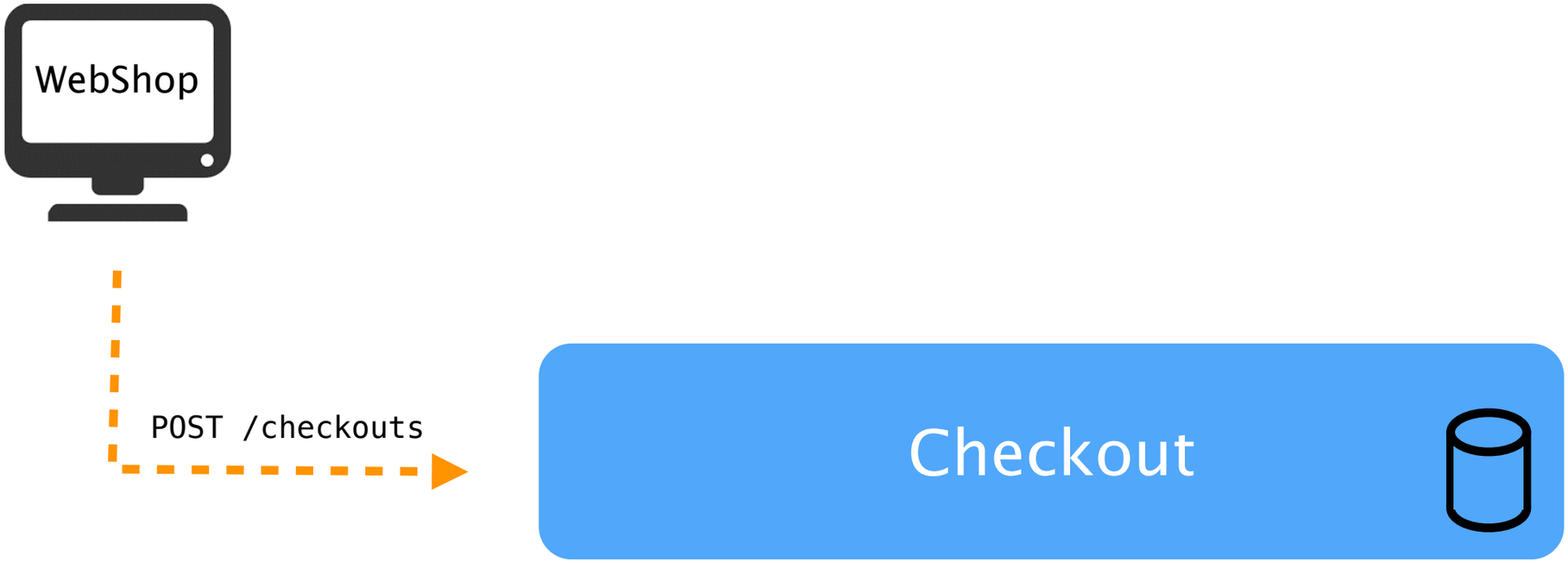
Stammdatenverwaltung



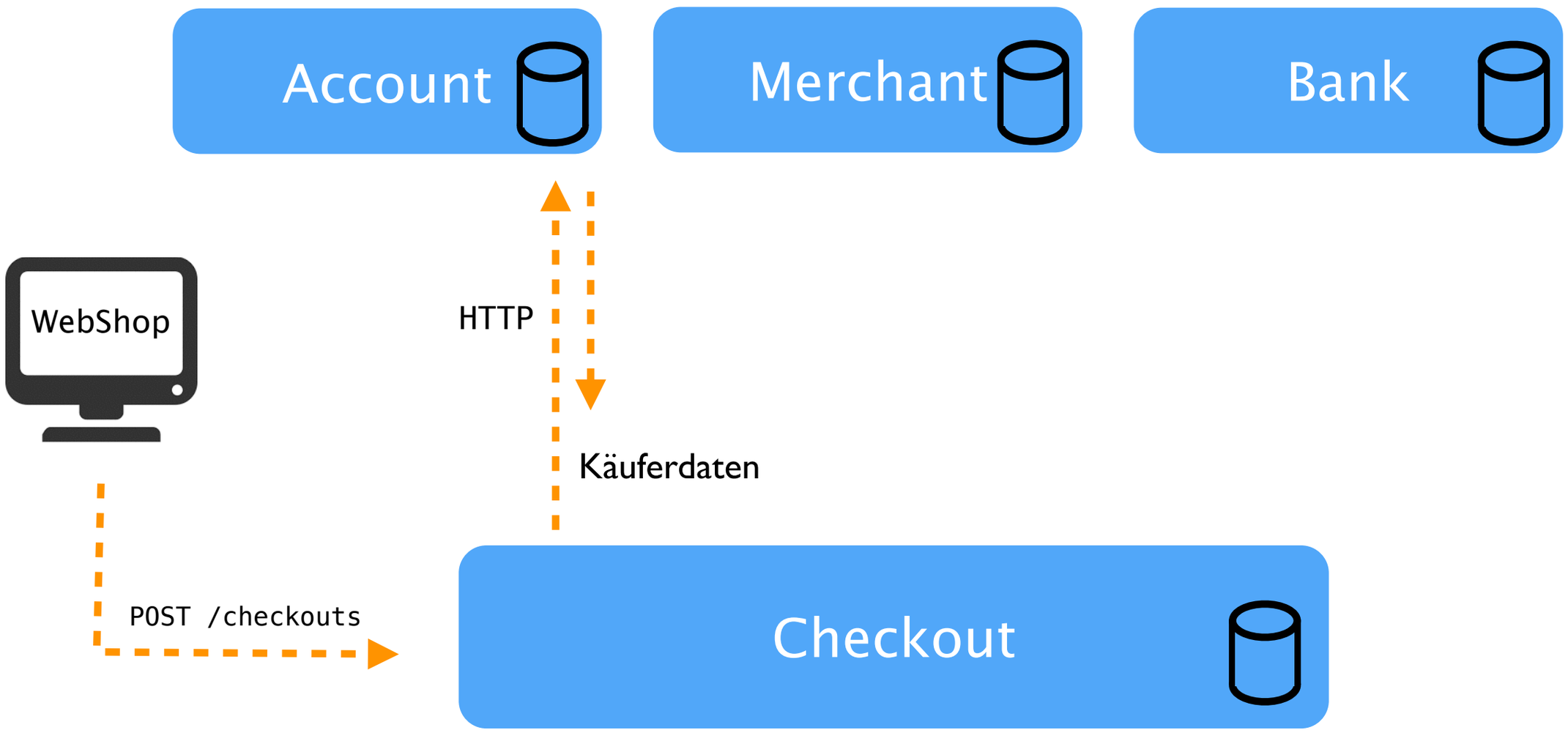
Zahlungsinitierung



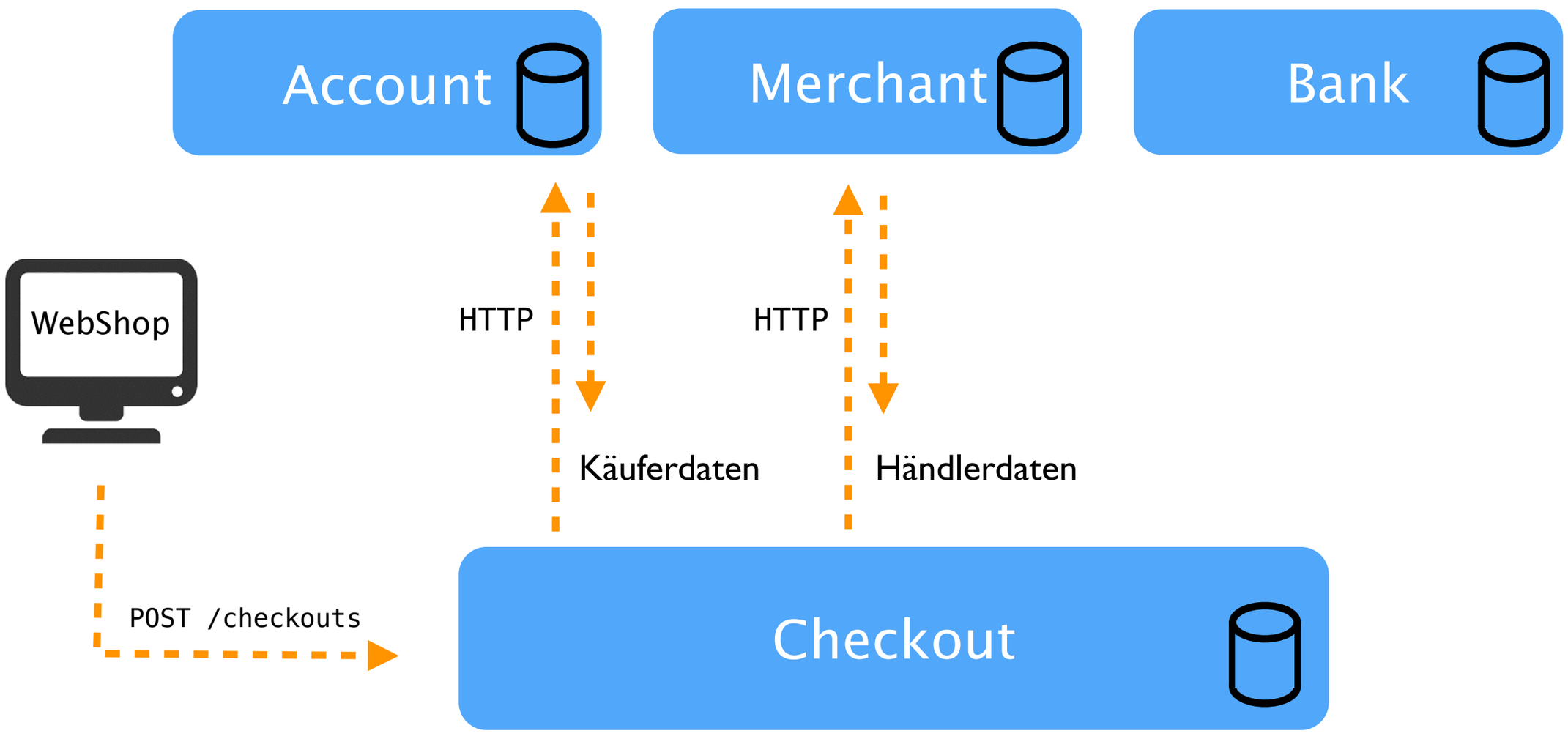
Zahlungsinitierung



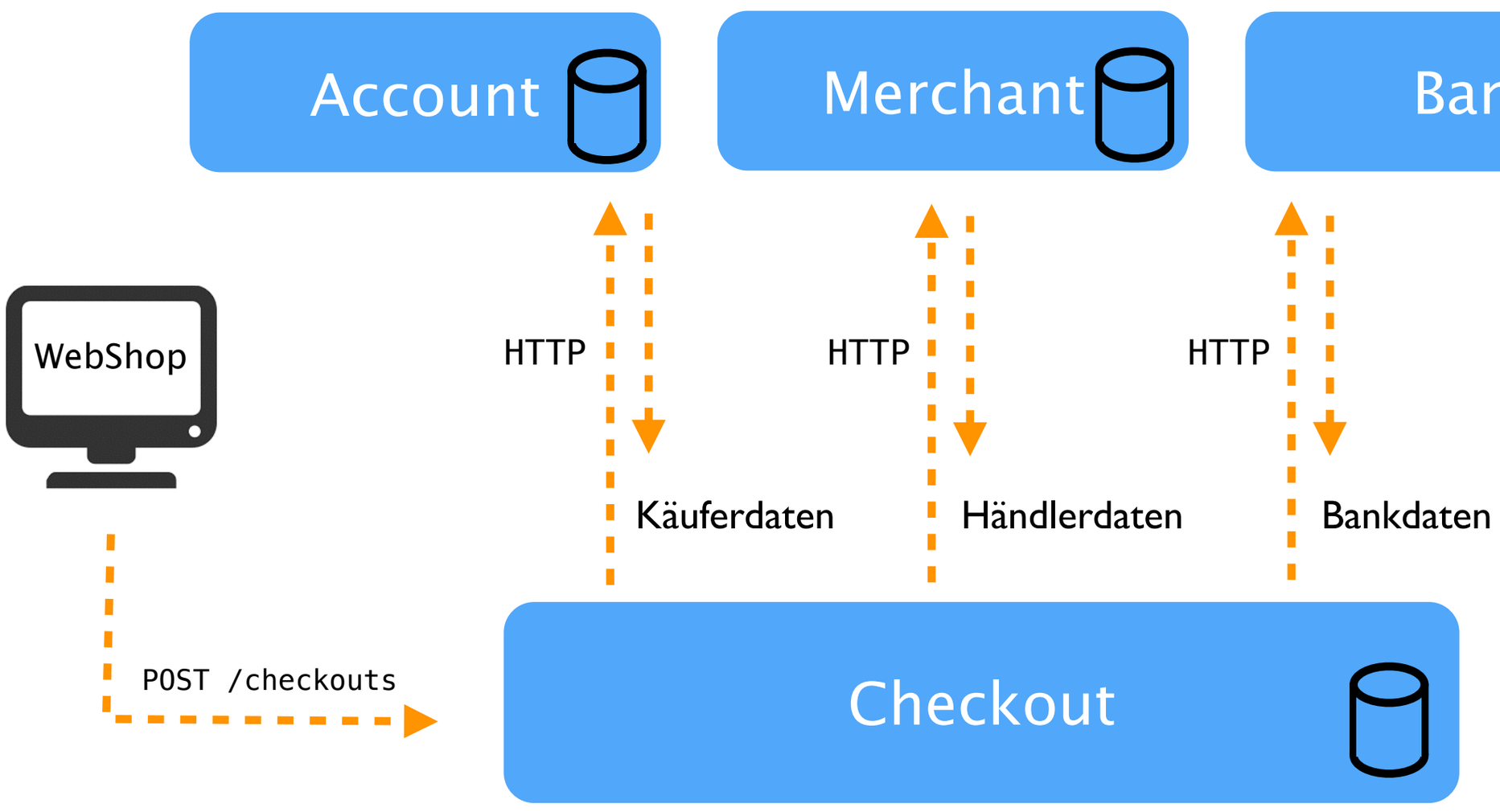
Zahlungsinitierung



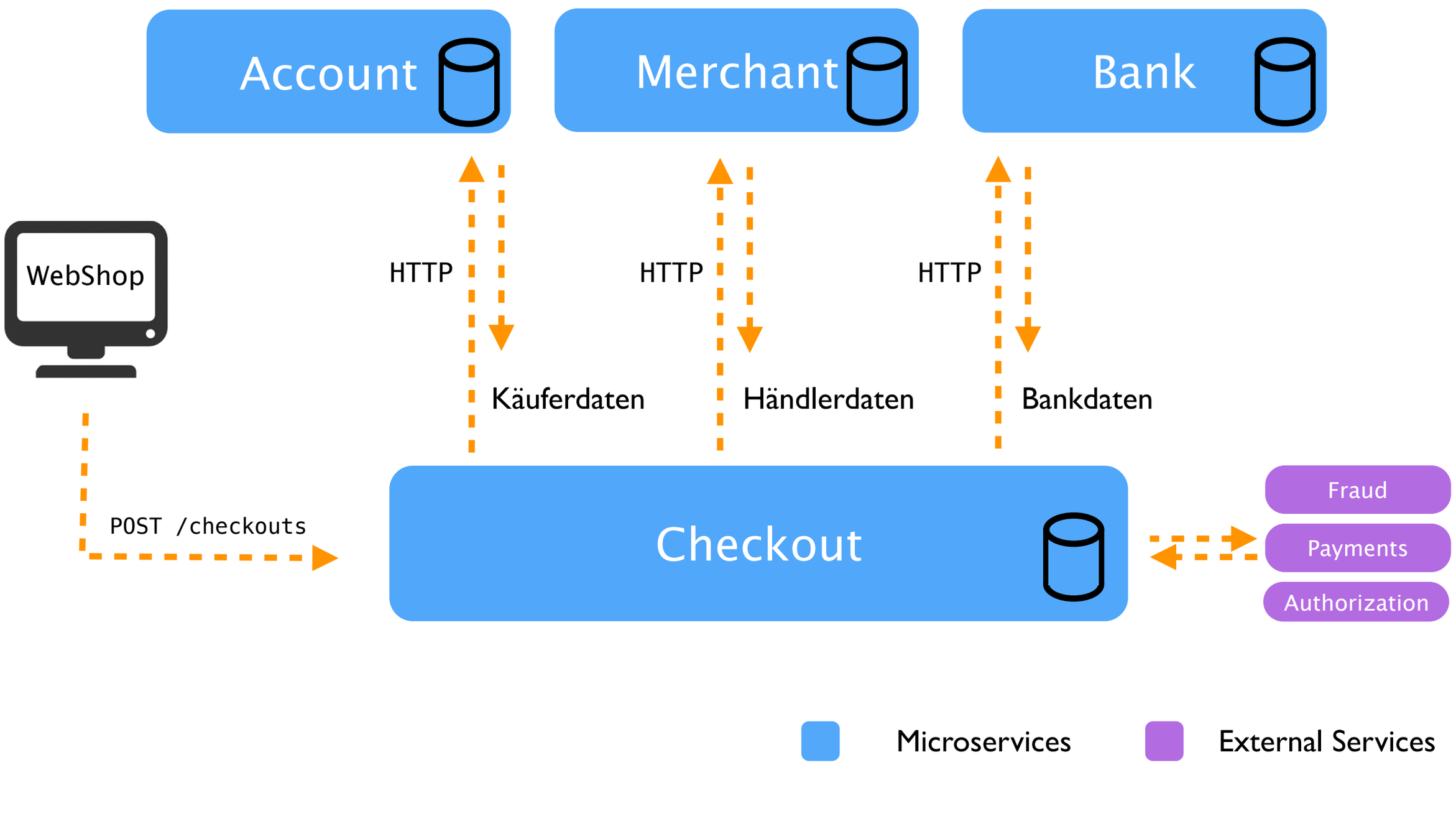
Zahlungsinitierung



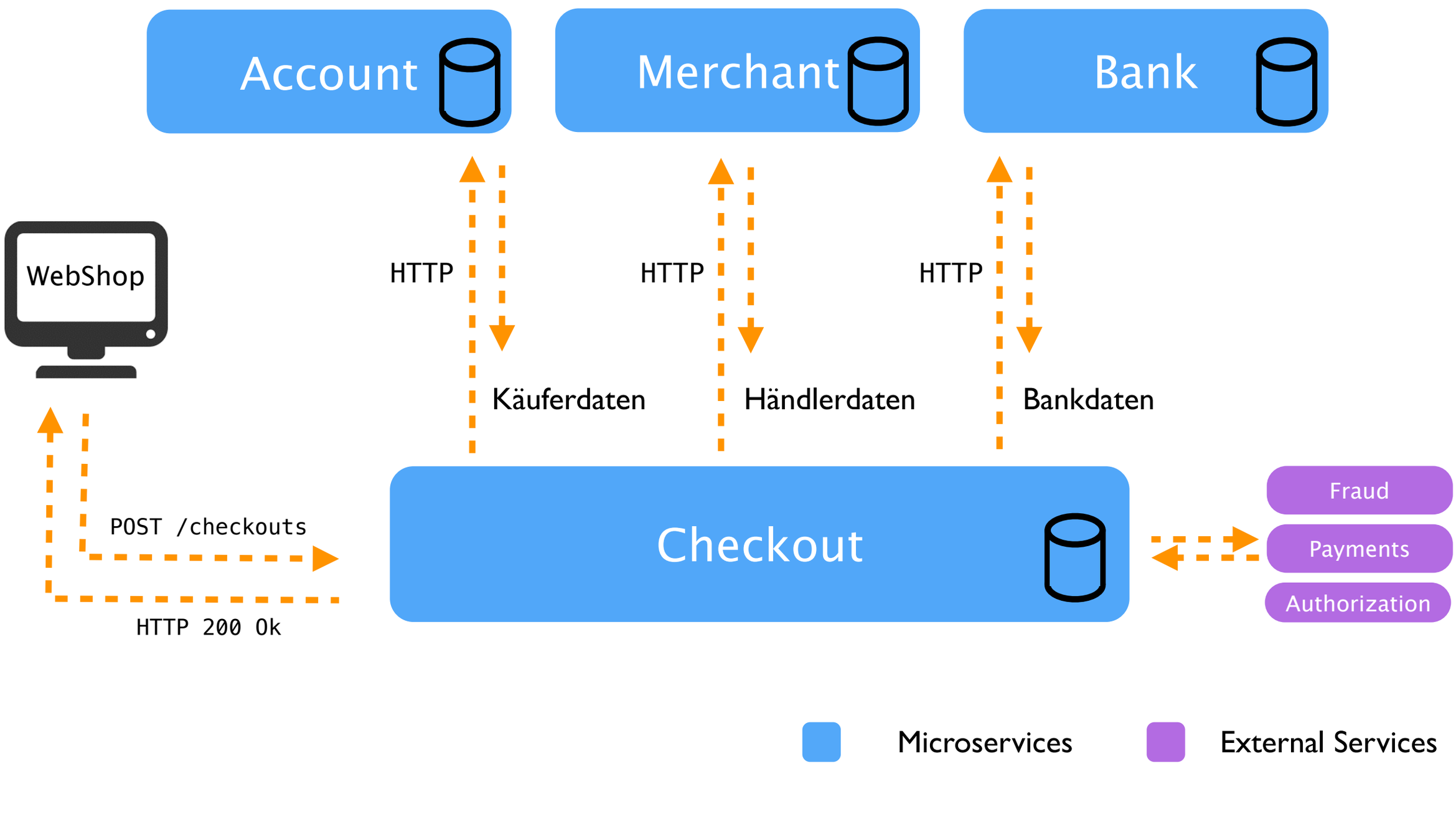
Zahlungsinitierung



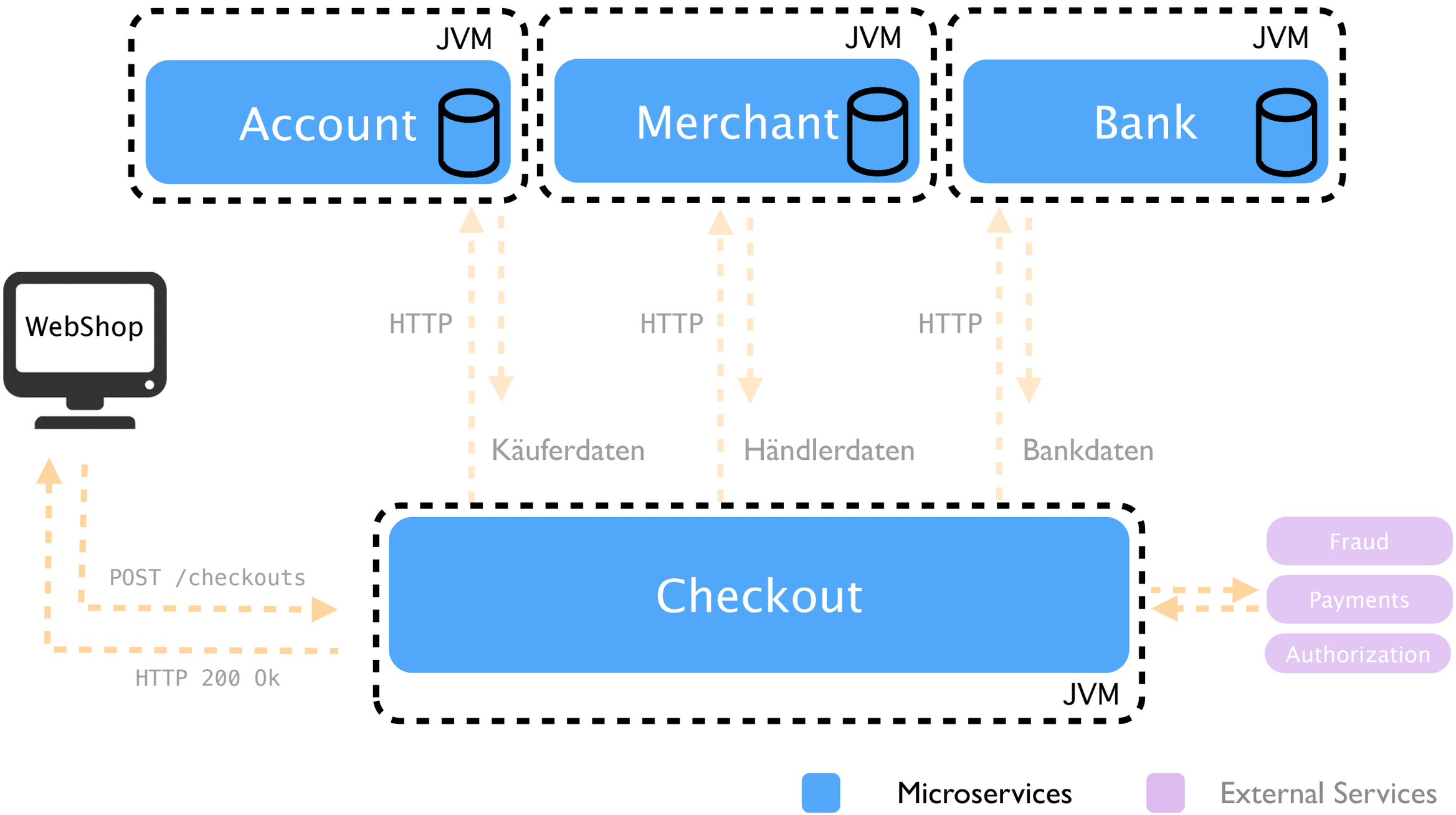
Zahlungsinitierung



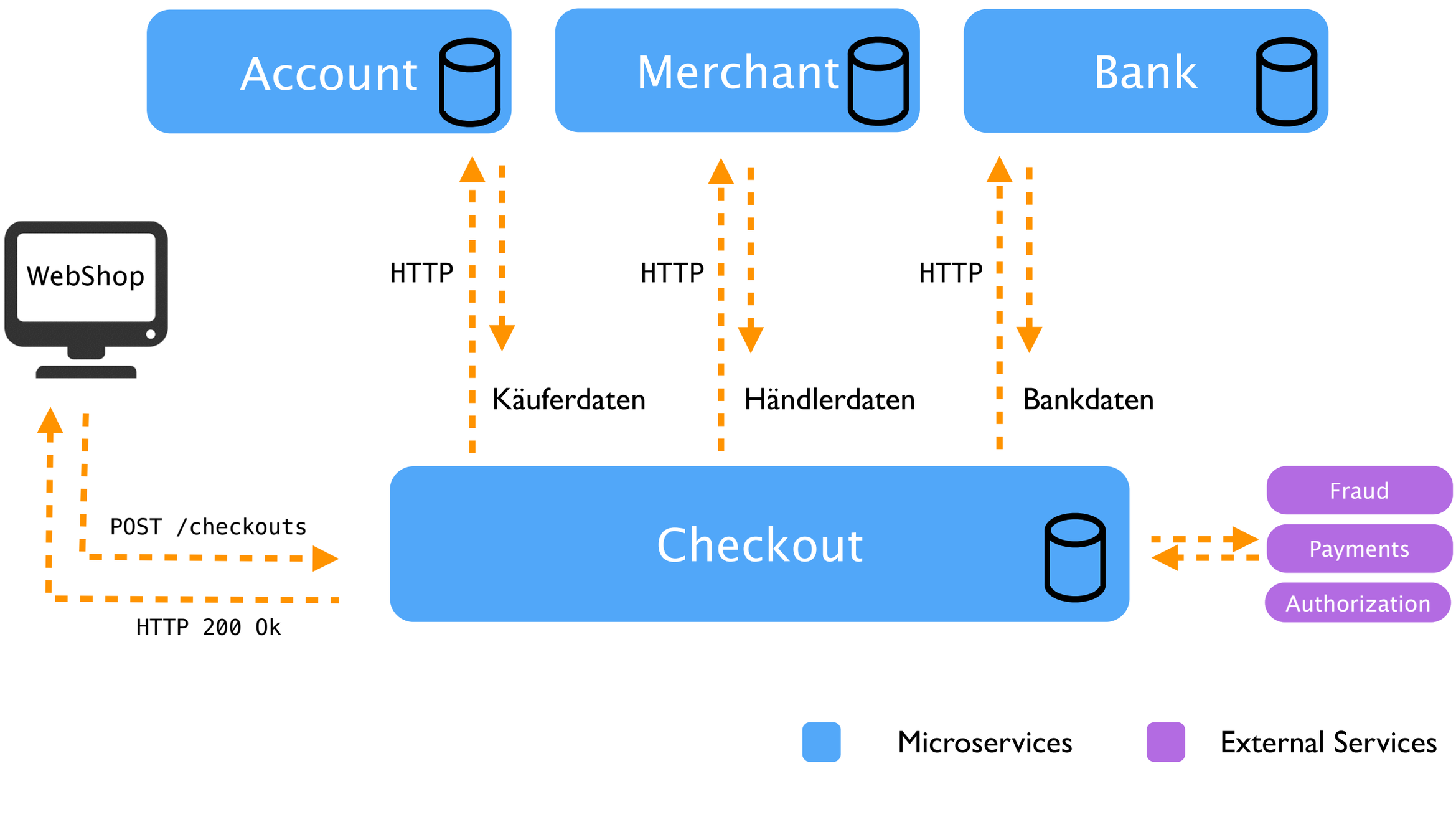
Zahlungsinitierung



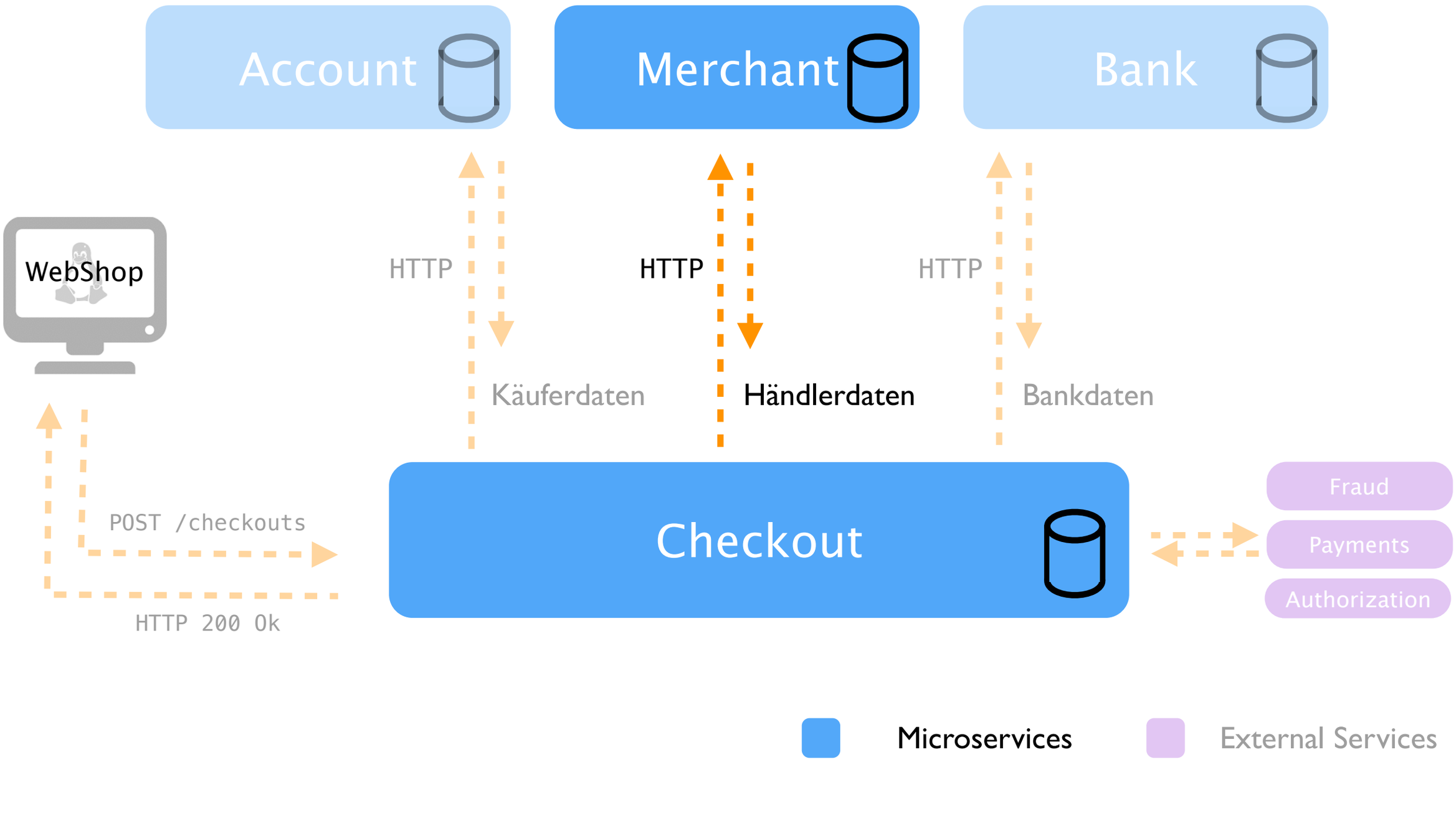
Zahlungsinitierung



Zahlungsinitierung

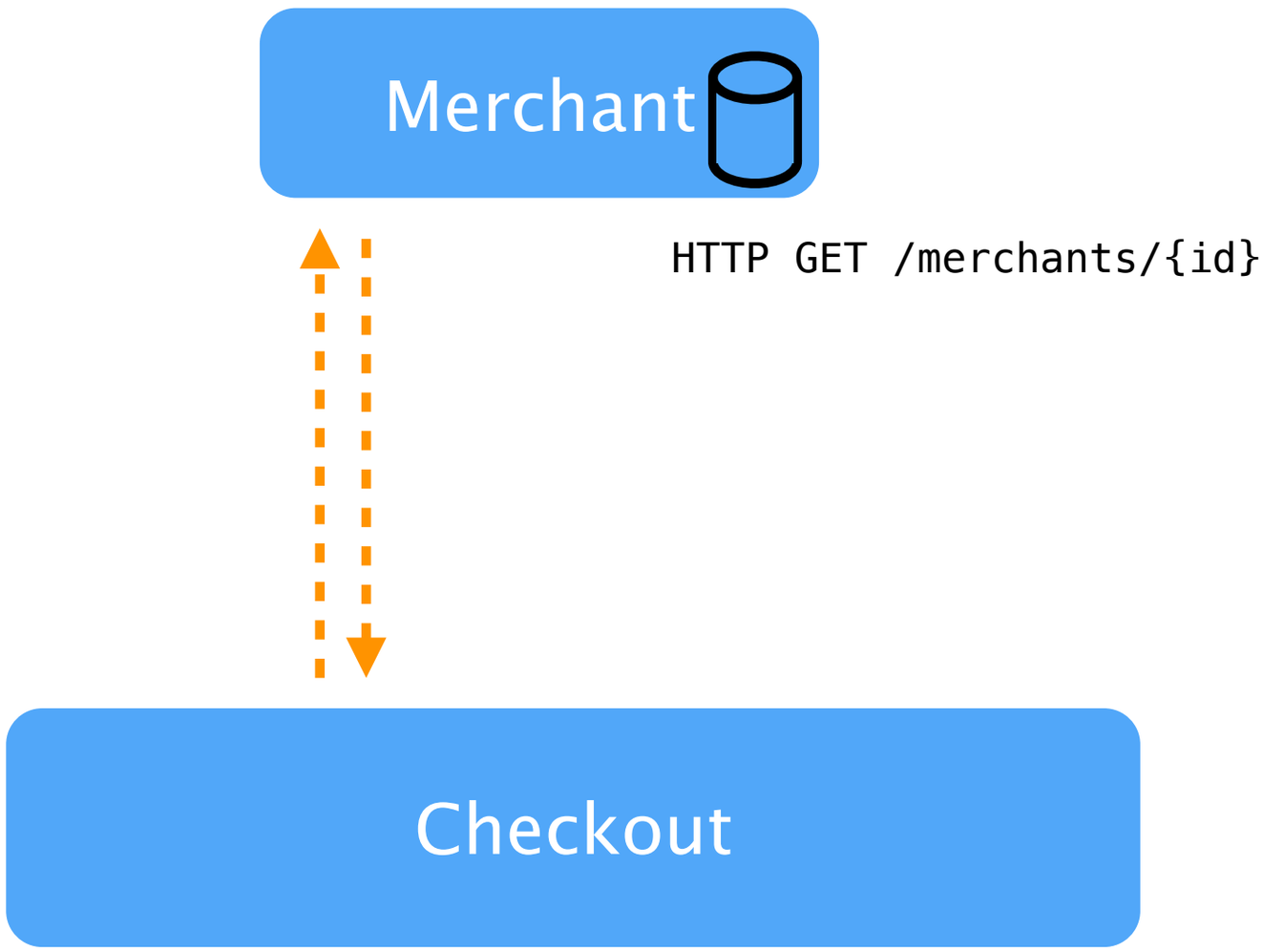


Zahlungsinitierung

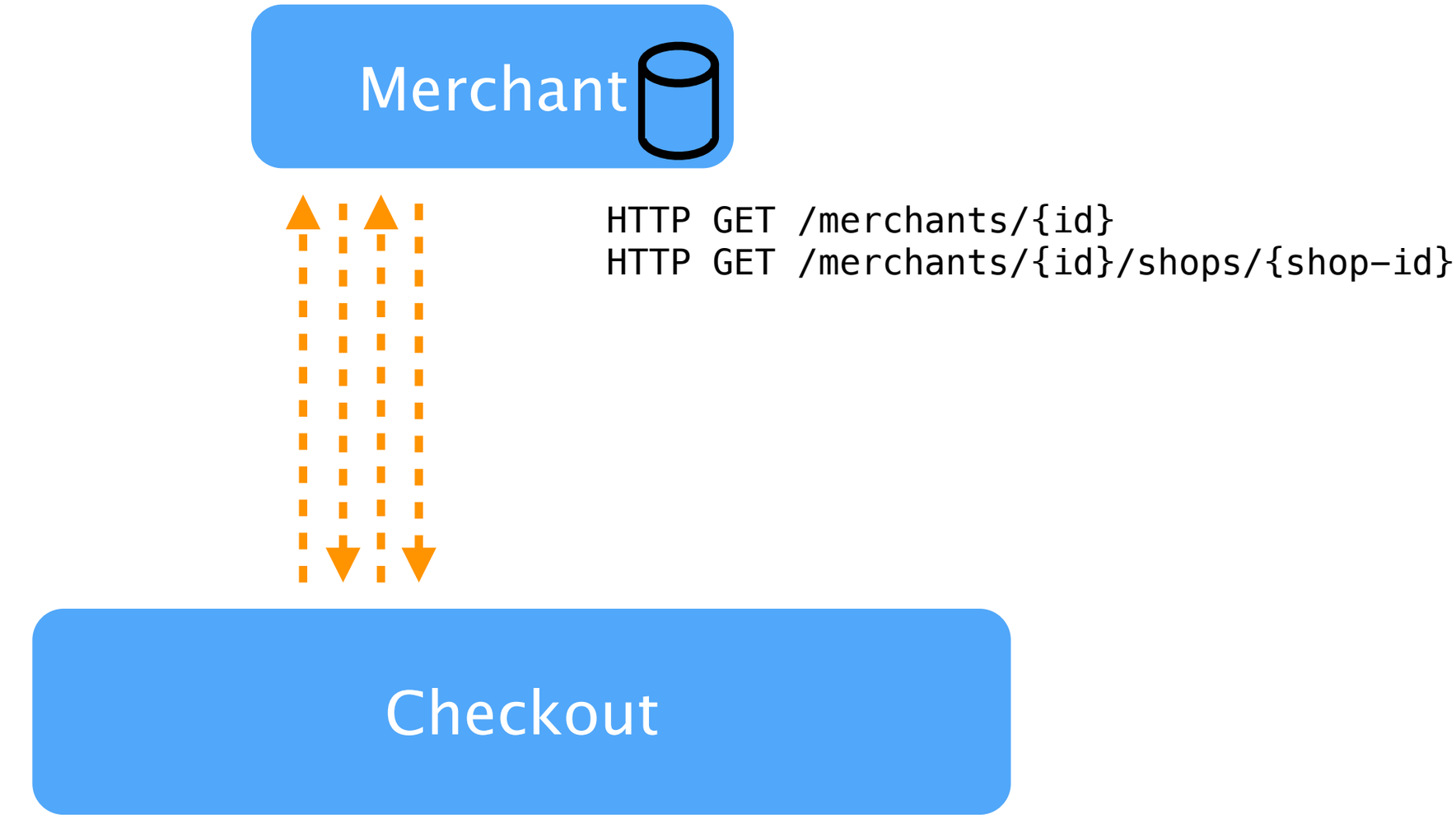


Service2Service Kommunikation

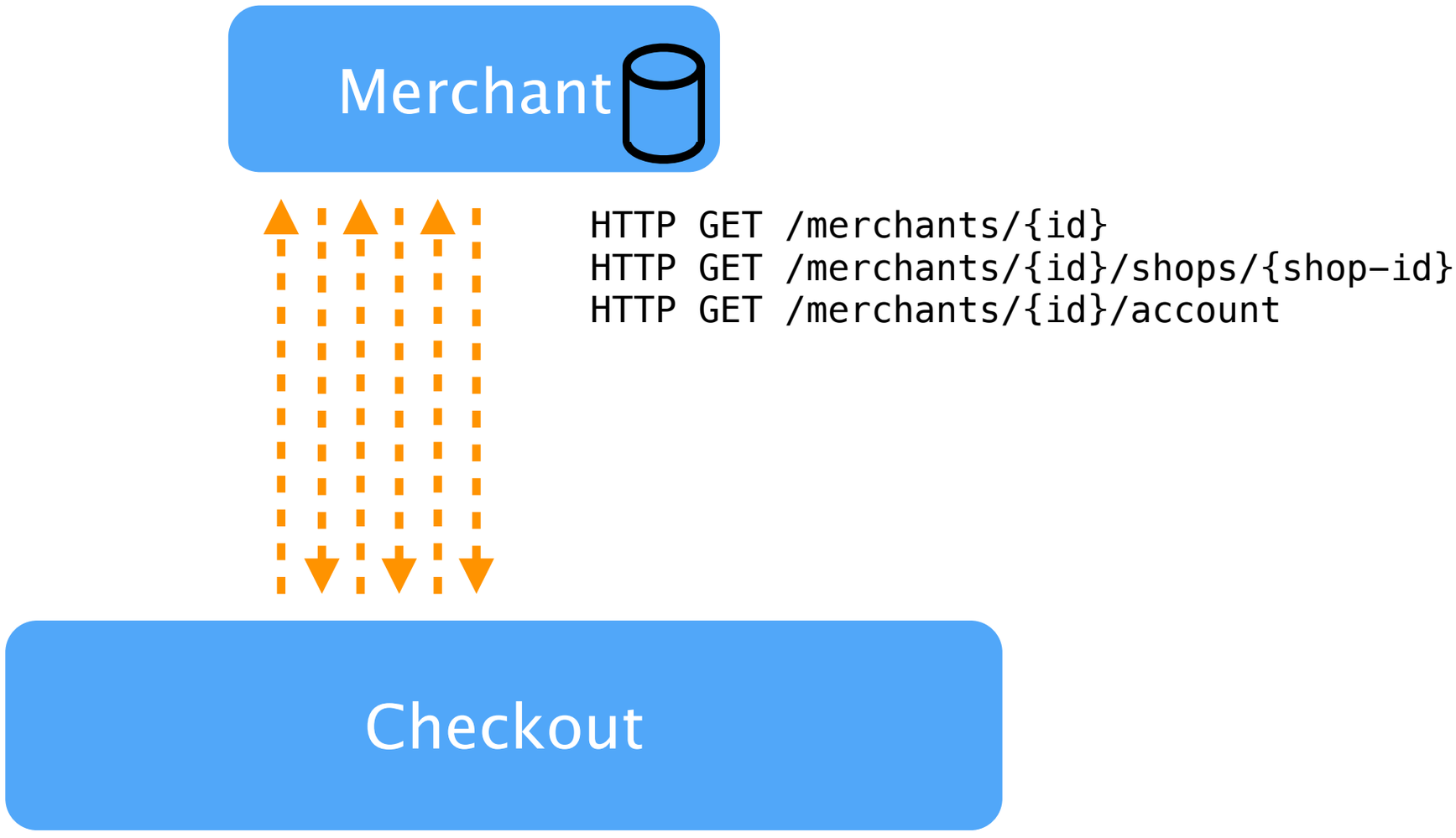
Service2Service Kommunikation



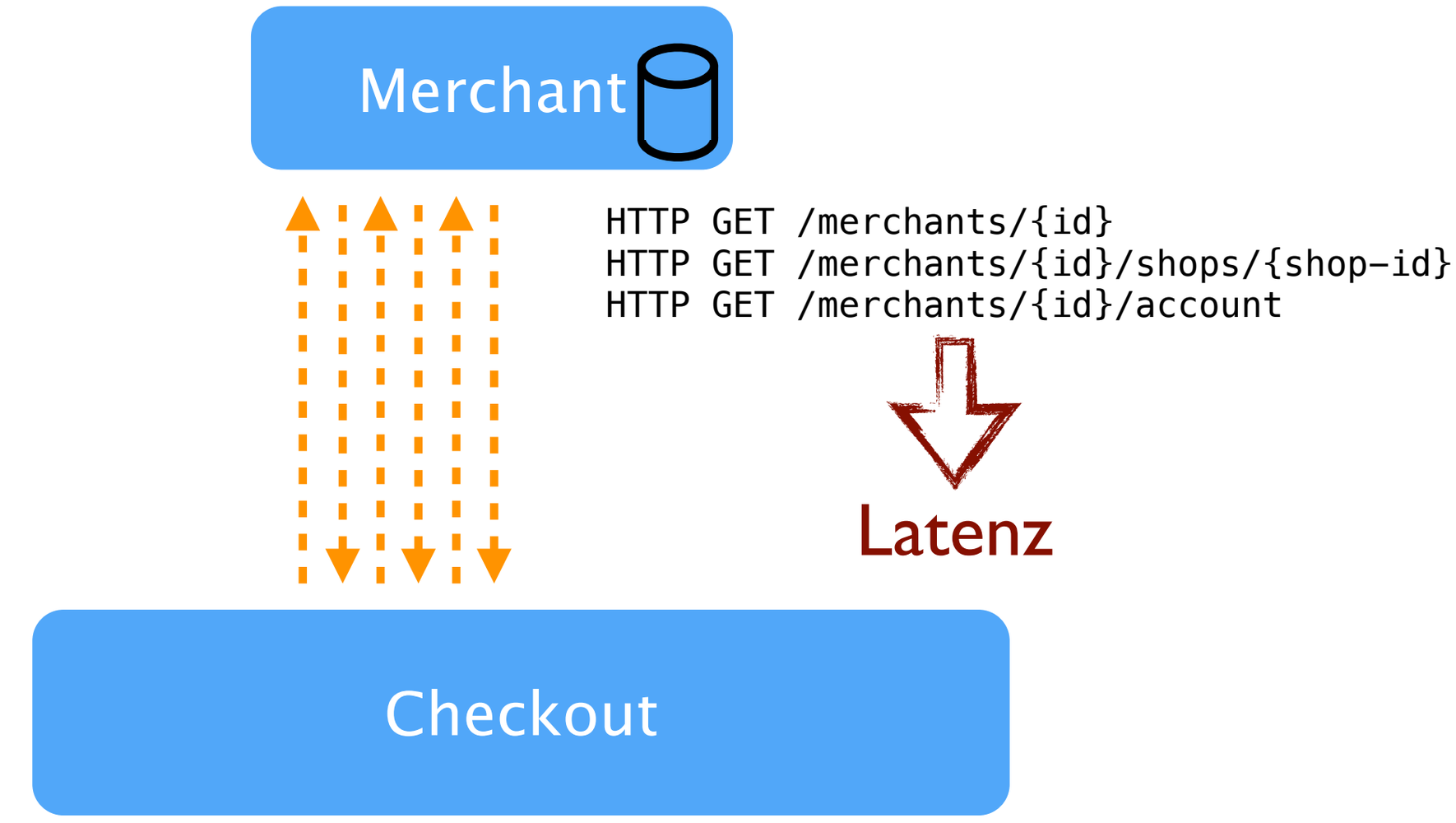
Service2Service Kommunikation



Service2Service Kommunikation



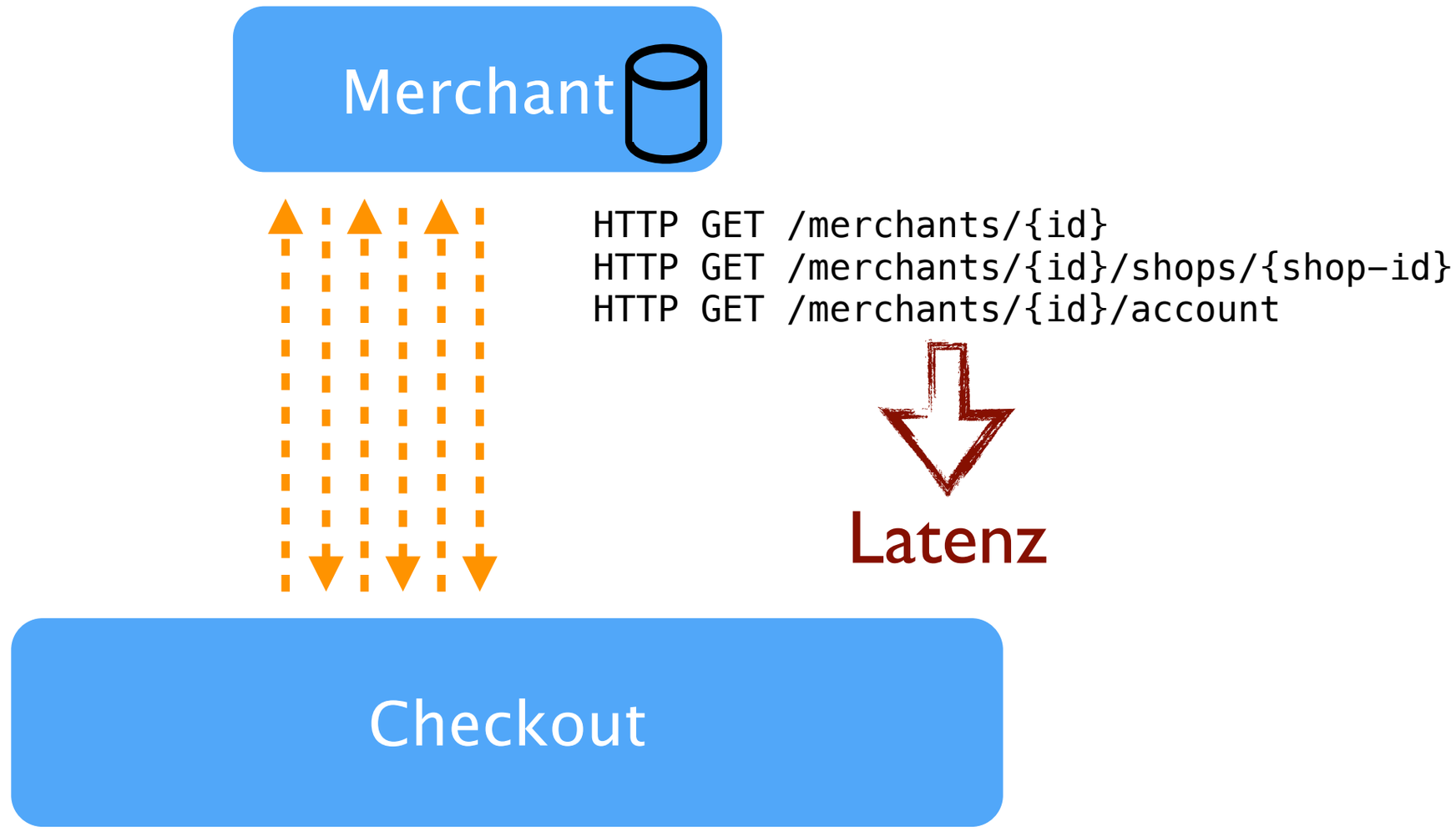
Service2Service Kommunikation



Service2Service Kommunikation

Timeouts?

Http 5xx?

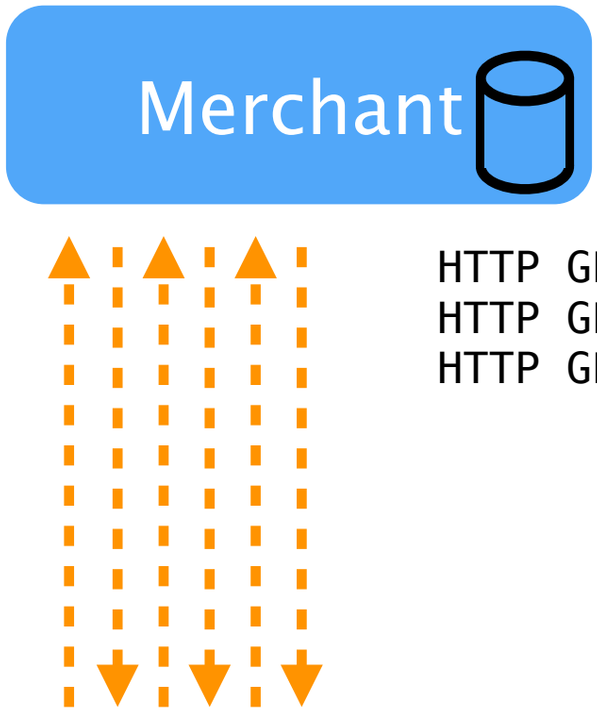


Service2Service Kommunikation

Timeouts?

Http 5xx?

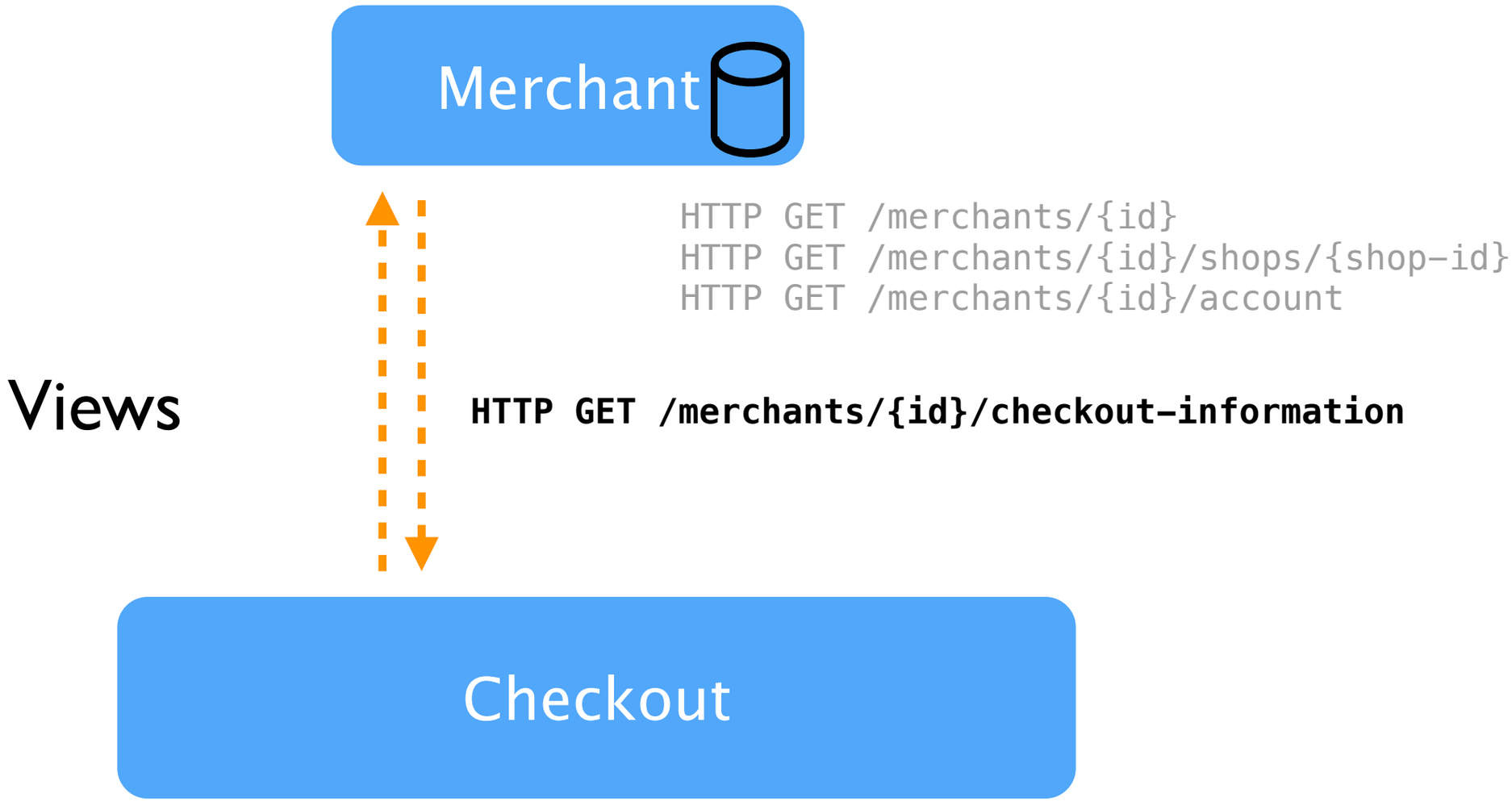
Resiliency




```
HTTP GET /merchants/{id}
HTTP GET /merchants/{id}/shops/{shop-id}
HTTP GET /merchants/{id}/account
```

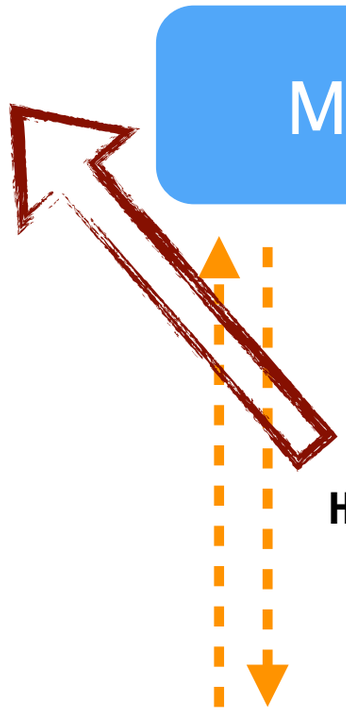
Latenz

Service2Service Kommunikation



Service2Service Kommunikation

Service-spezifische URL-Endpoints



Merchant

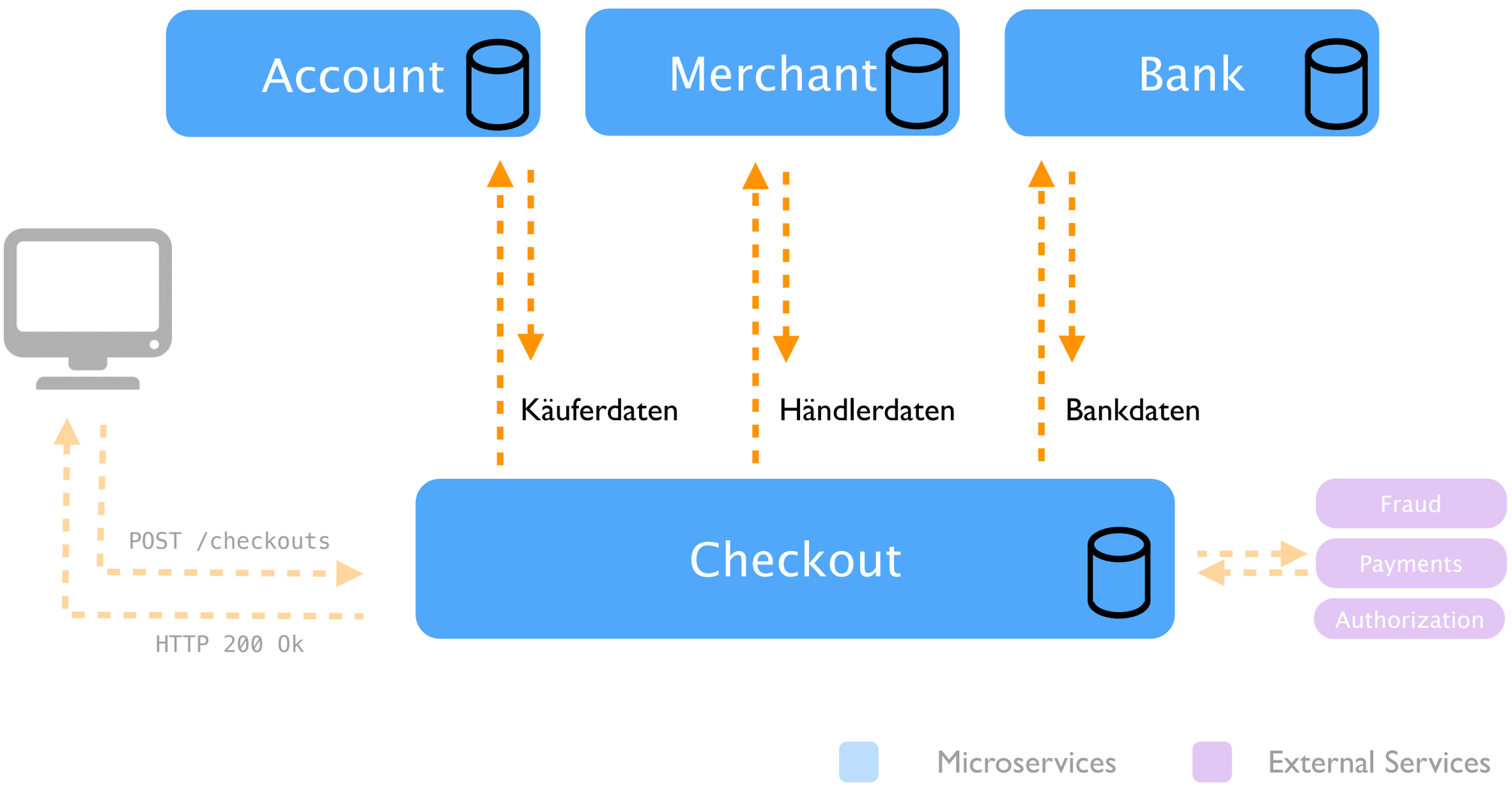
```
HTTP GET /merchants/{id}
HTTP GET /merchants/{id}/shops/{shop-id}
HTTP GET /merchants/{id}/account
```

Views

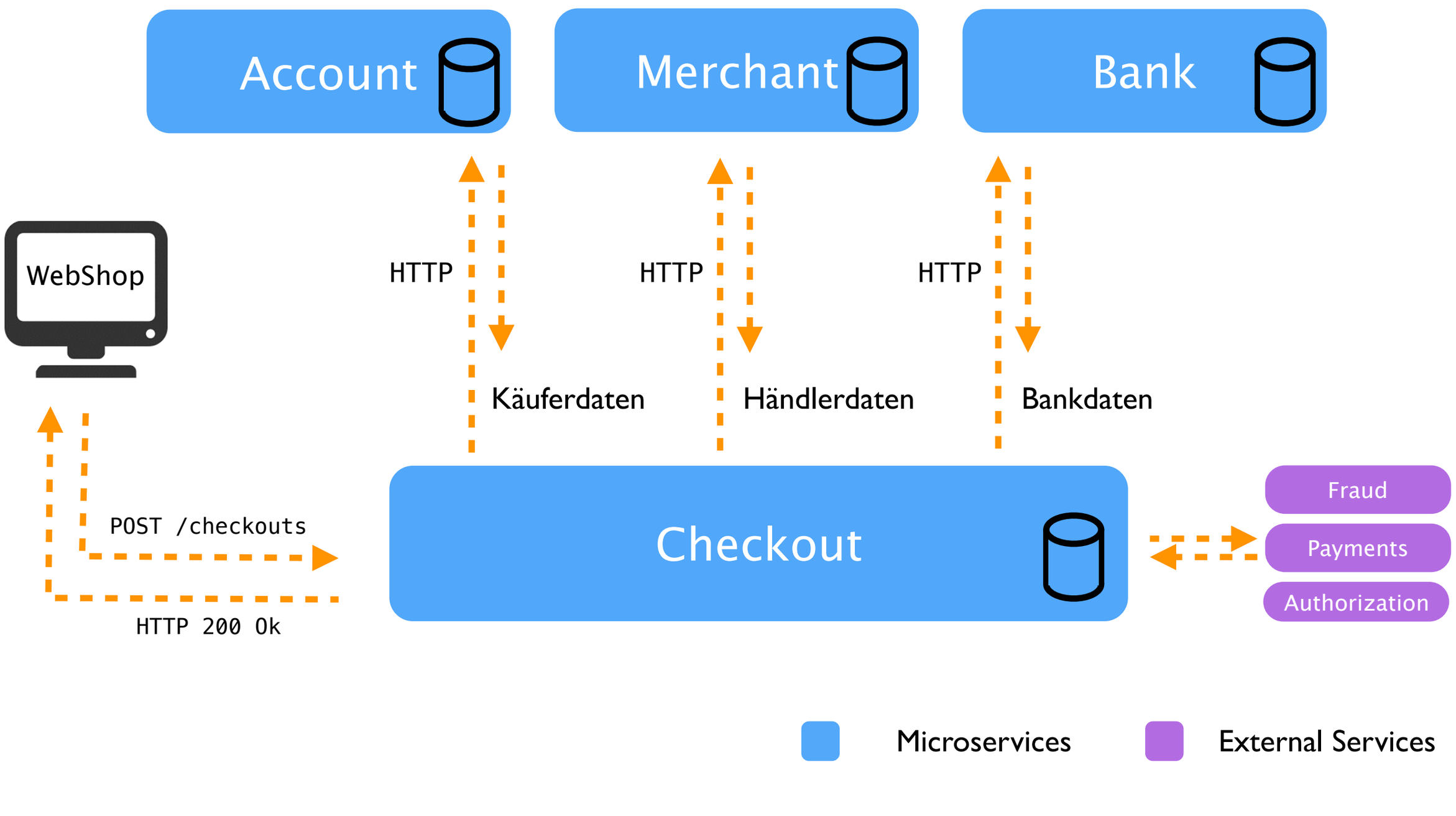
`HTTP GET /merchants/{id}/checkout-information`

Checkout

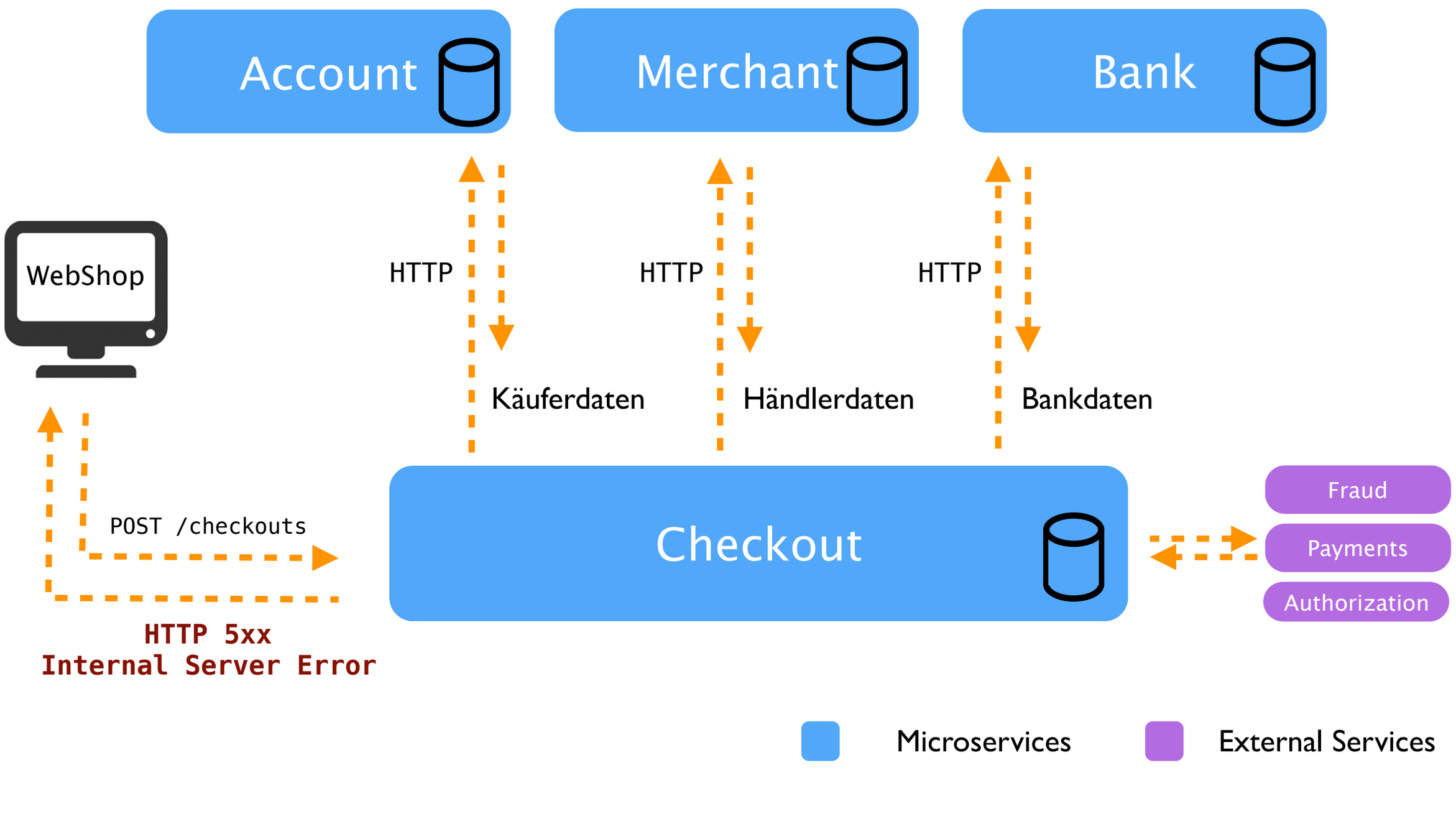
Zahlungsinitierung



Tests?


Account
Merchant
Bank
WebShop
HTTP
HTTP
HTTP
Käuferdaten
Händlerdaten
Bankdaten
POST /checkouts
HTTP 200 Ok
Checkout
Fraud
Payments
Authorization
Microservices
External Services

Tests?


Account
Merchant
Bank
WebShop
HTTP
HTTP
HTTP
Käuferdaten
Händlerdaten
Bankdaten
POST /checkouts
HTTP 5xx
Internal Server Error
Checkout
Fraud
Payments
Authorization
Microservices
External Services

Fail!

service-e2e-test

Verfügbarkeit

low availability
low traffic

Account

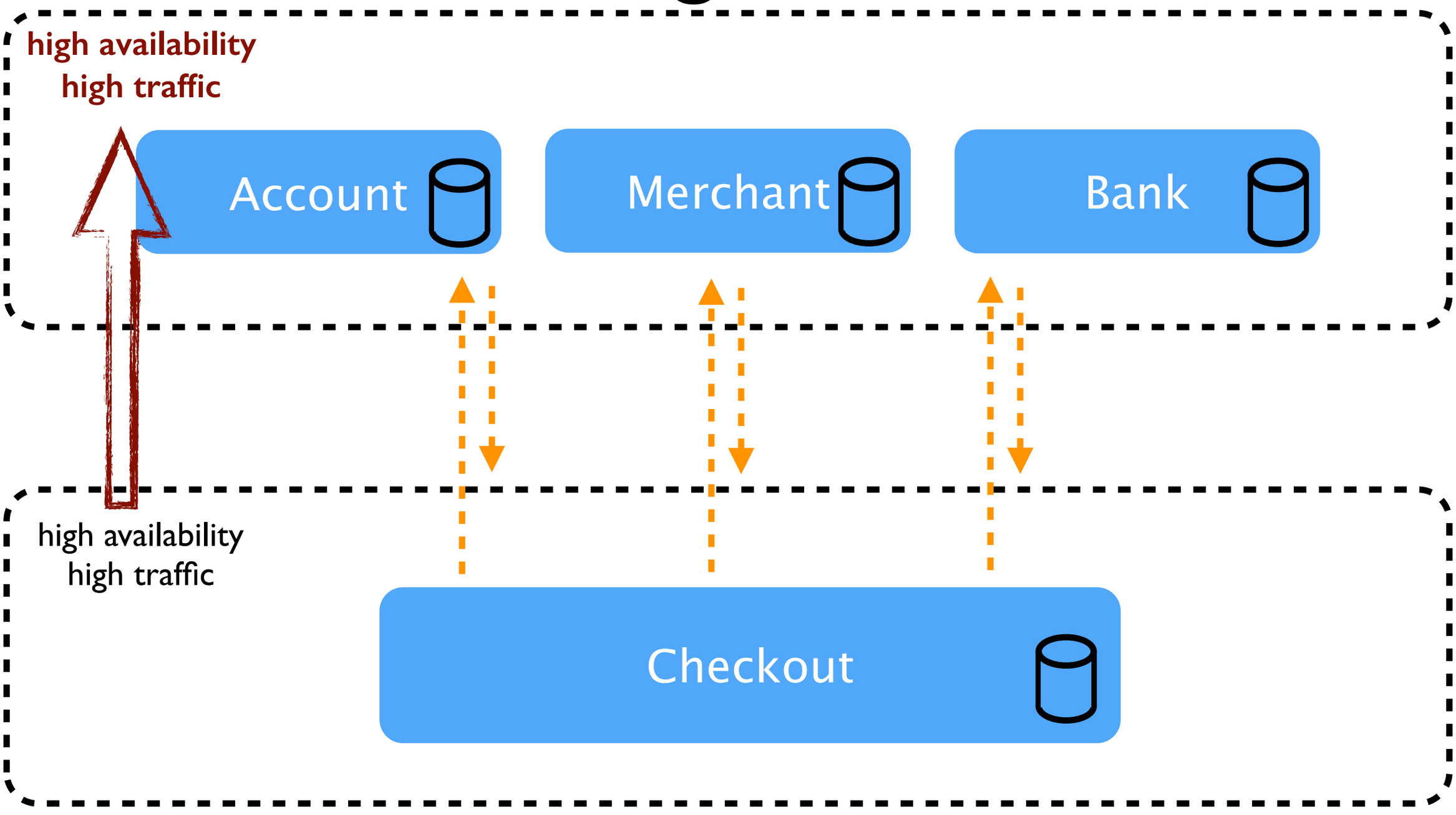
Merchant

Bank

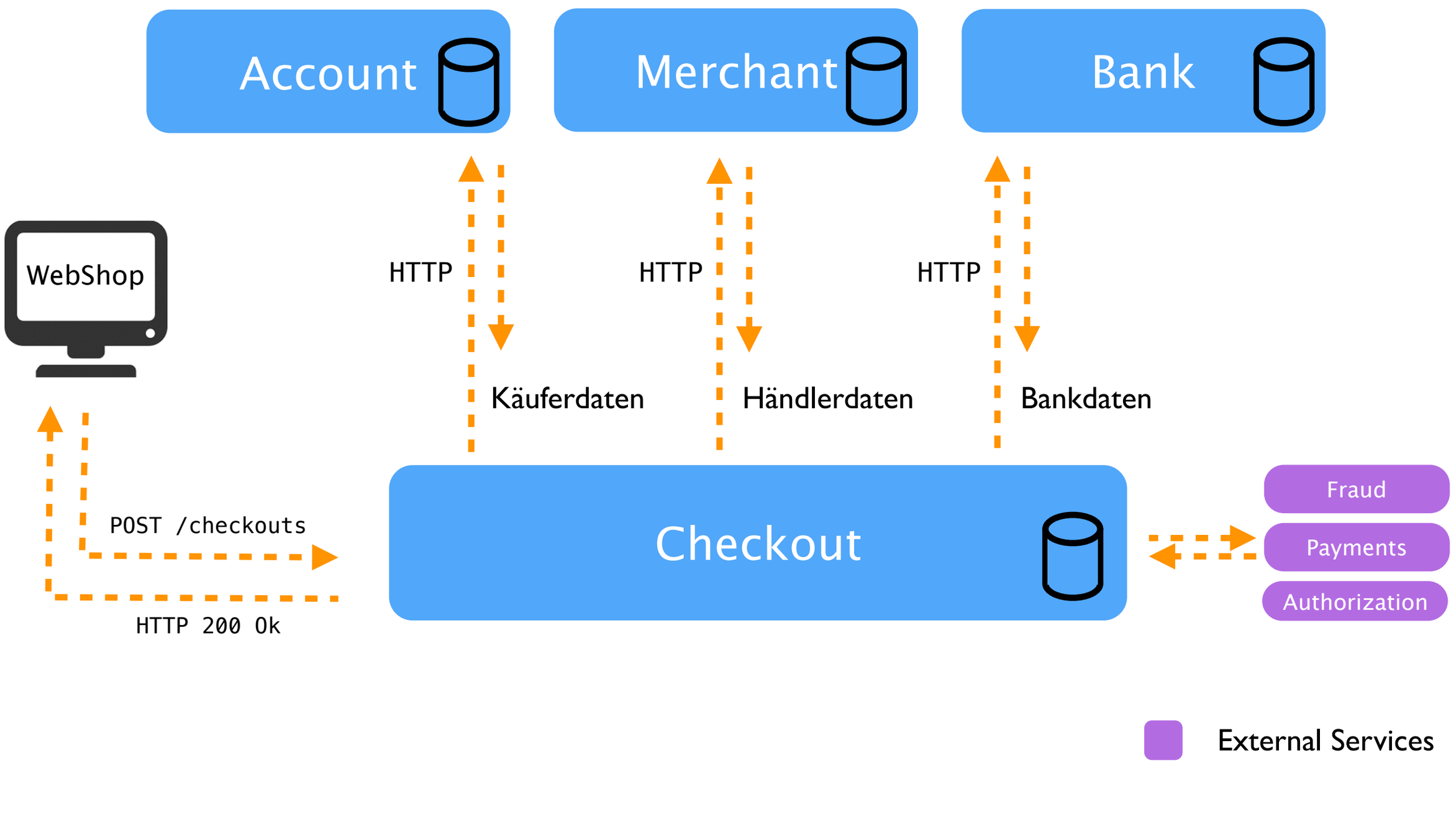
high availability
high traffic

Checkout

Verfügbarkeit



:-(


Account
Merchant
Bank
WebShop
HTTP
HTTP
HTTP
Käuferdaten
Händlerdaten
Bankdaten
POST /checkouts
HTTP 200 Ok
Checkout
Fraud
Payments
Authorization
External Services

Event Sourcing

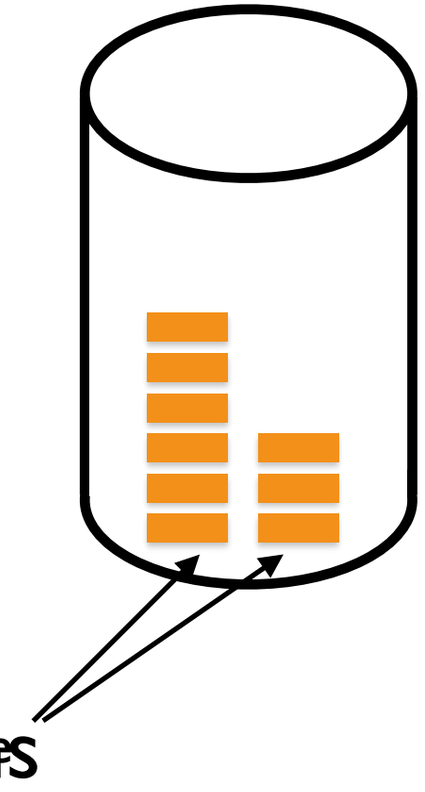
Account Events

Phone Number Added

Address Updated

Account Limit Updated

Event Store



Event Streams

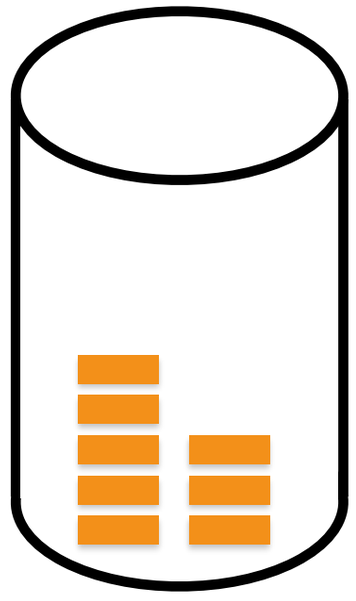
Account Events

Phone Number Added

Address Updated

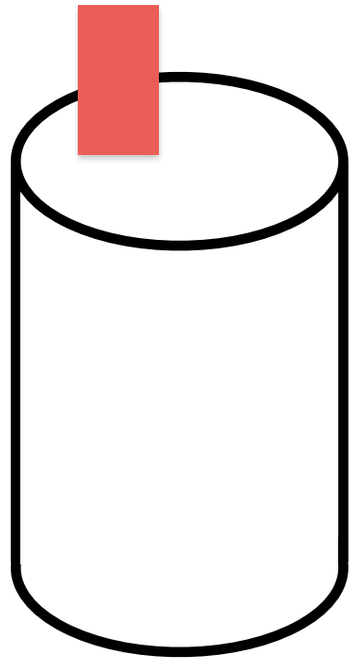
Account Limit Updated

Event Store



```
“Account” : {
      “id: “0123456789”
      “name: “Mike”
      “lastName: “Magic”

      “limit” : 2000
      “address: {…}
      “phoneNumber: “+49 126 555 555
      …
}
```

reduce



Current State

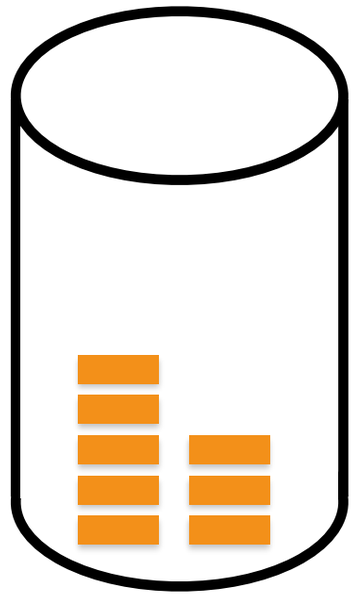
Account Events

Phone Number Added

Address Updated

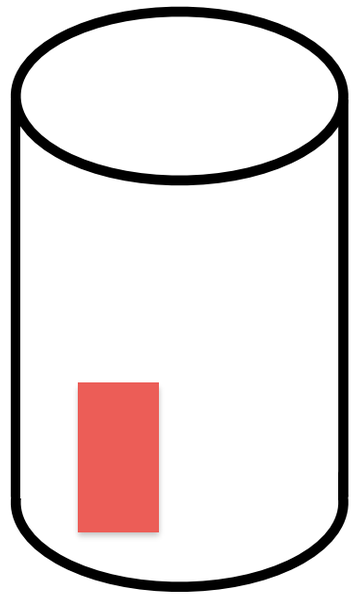
Account Limit Updated

Event Store



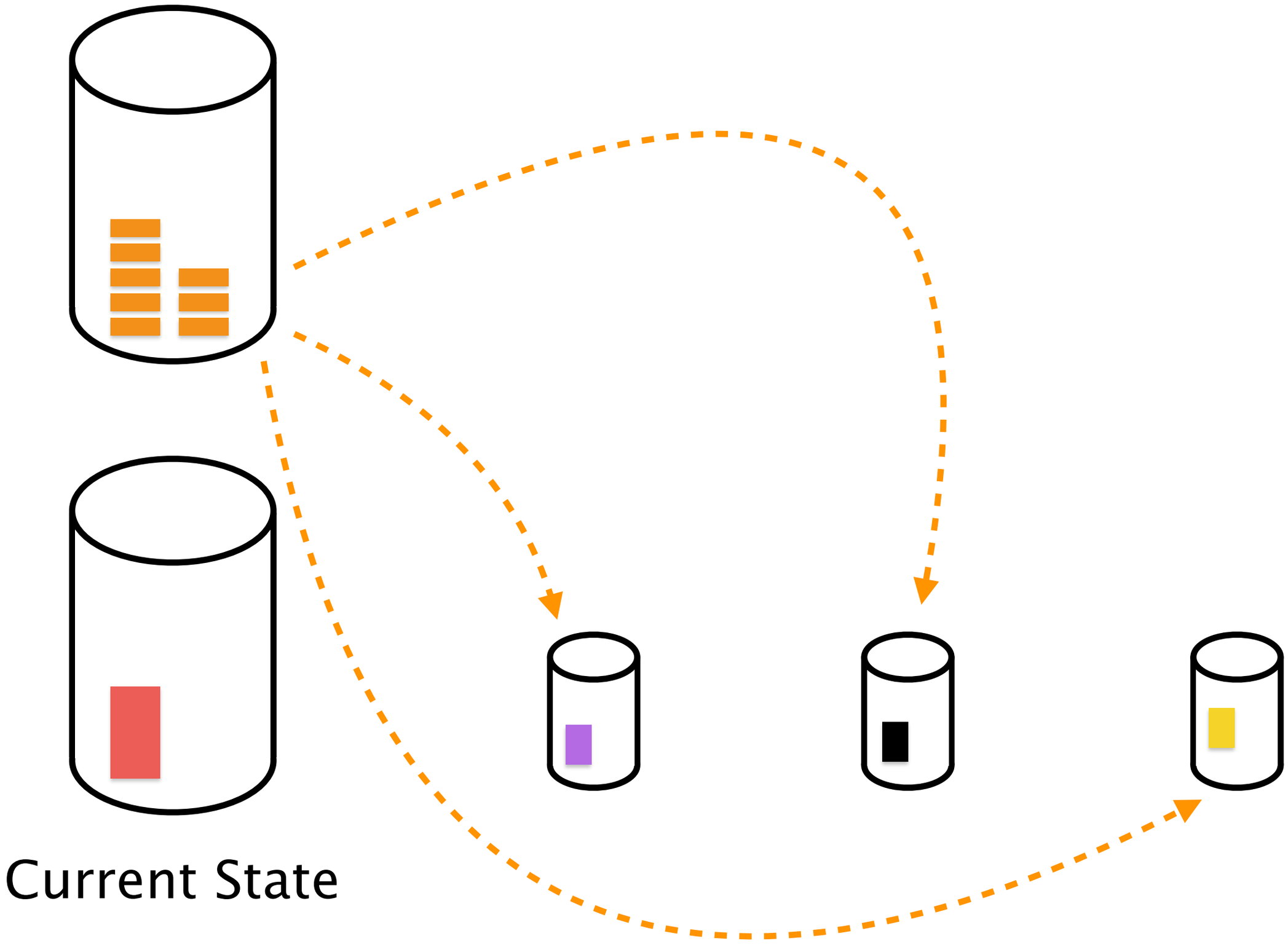
```
“Account” : {
      “id: “0123456789”
      “name: “Mike”
      “lastName: “Magic”

      “limit” : 2000
      “address: {…}
      “phoneNumber: “+49 126 555 555
      …
}
```

reduce



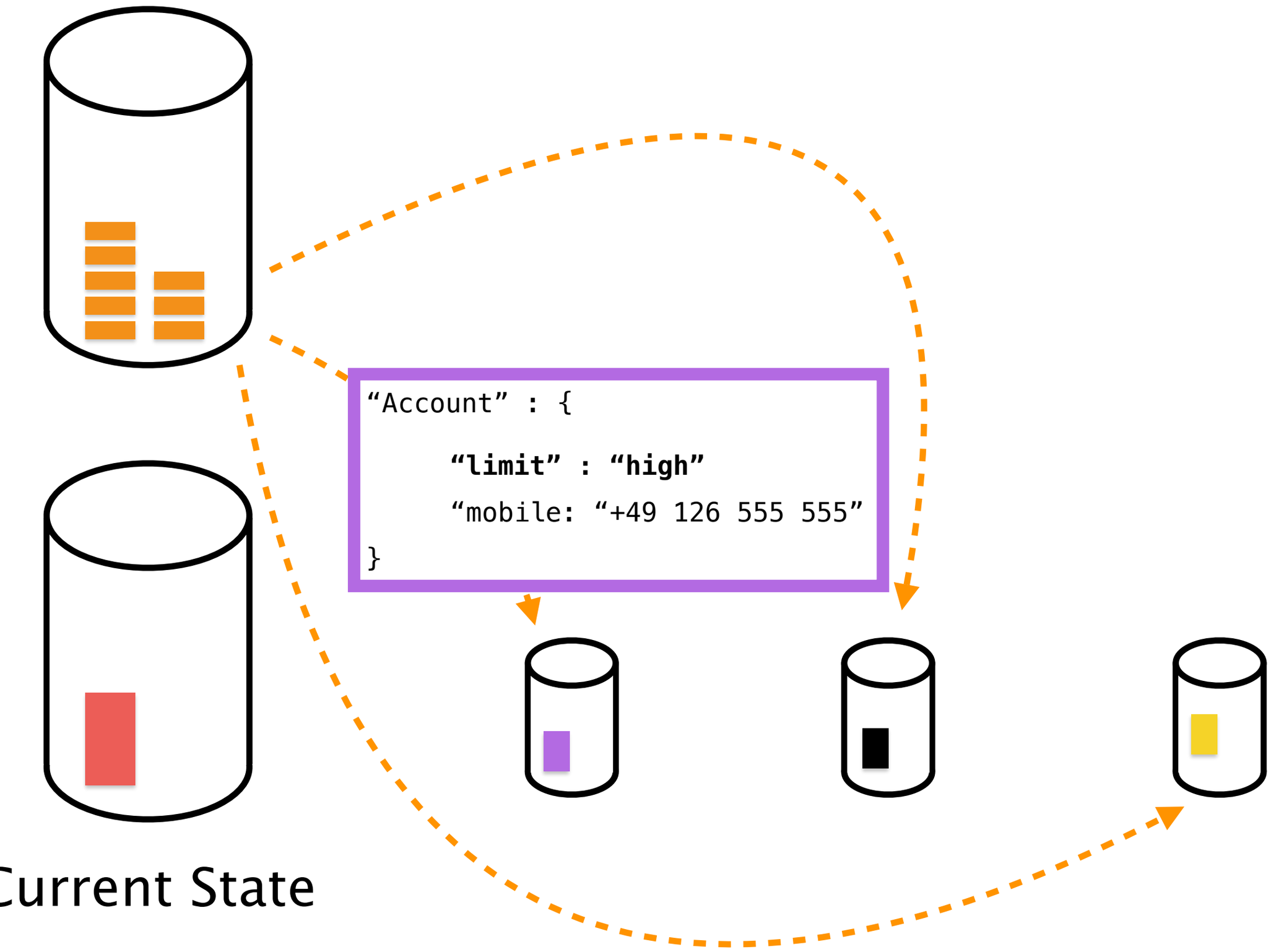
Current State

Event Store



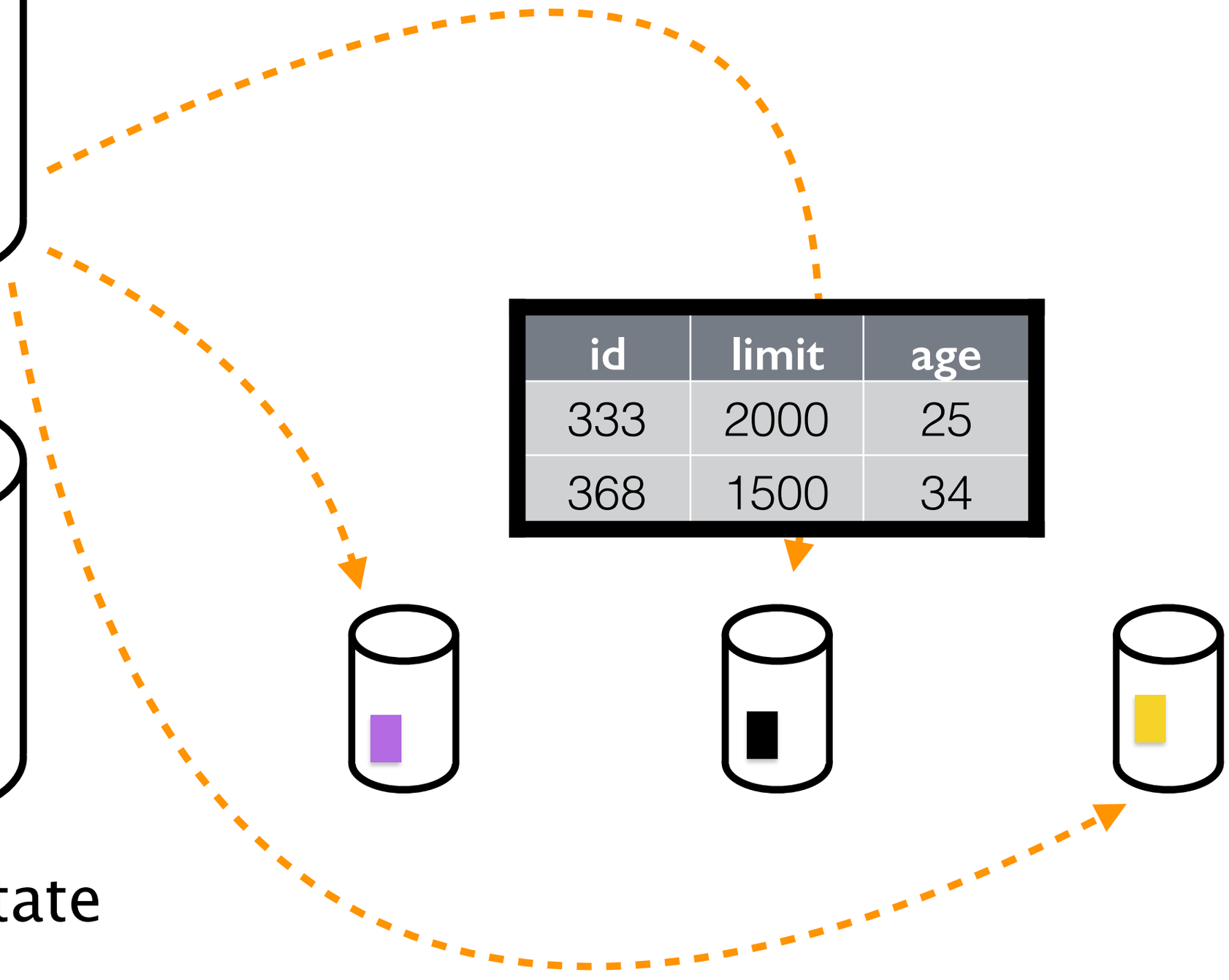
Current State

Event Store



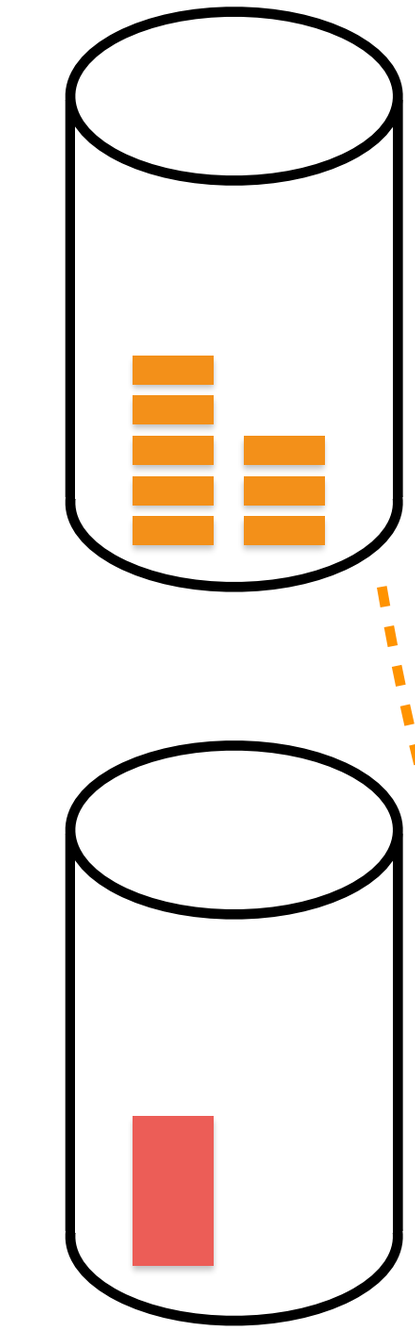
Current State

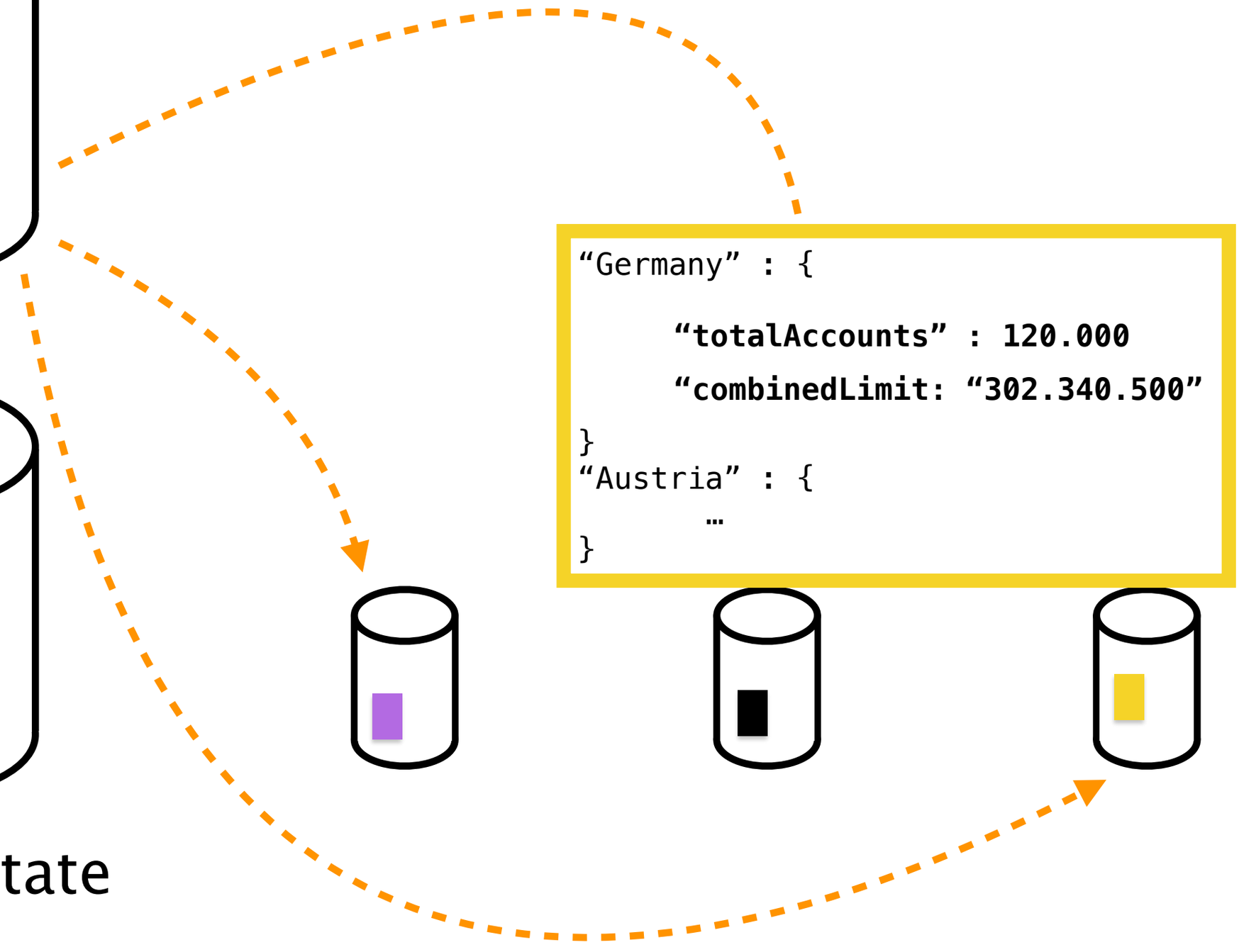
Event Store



Current State

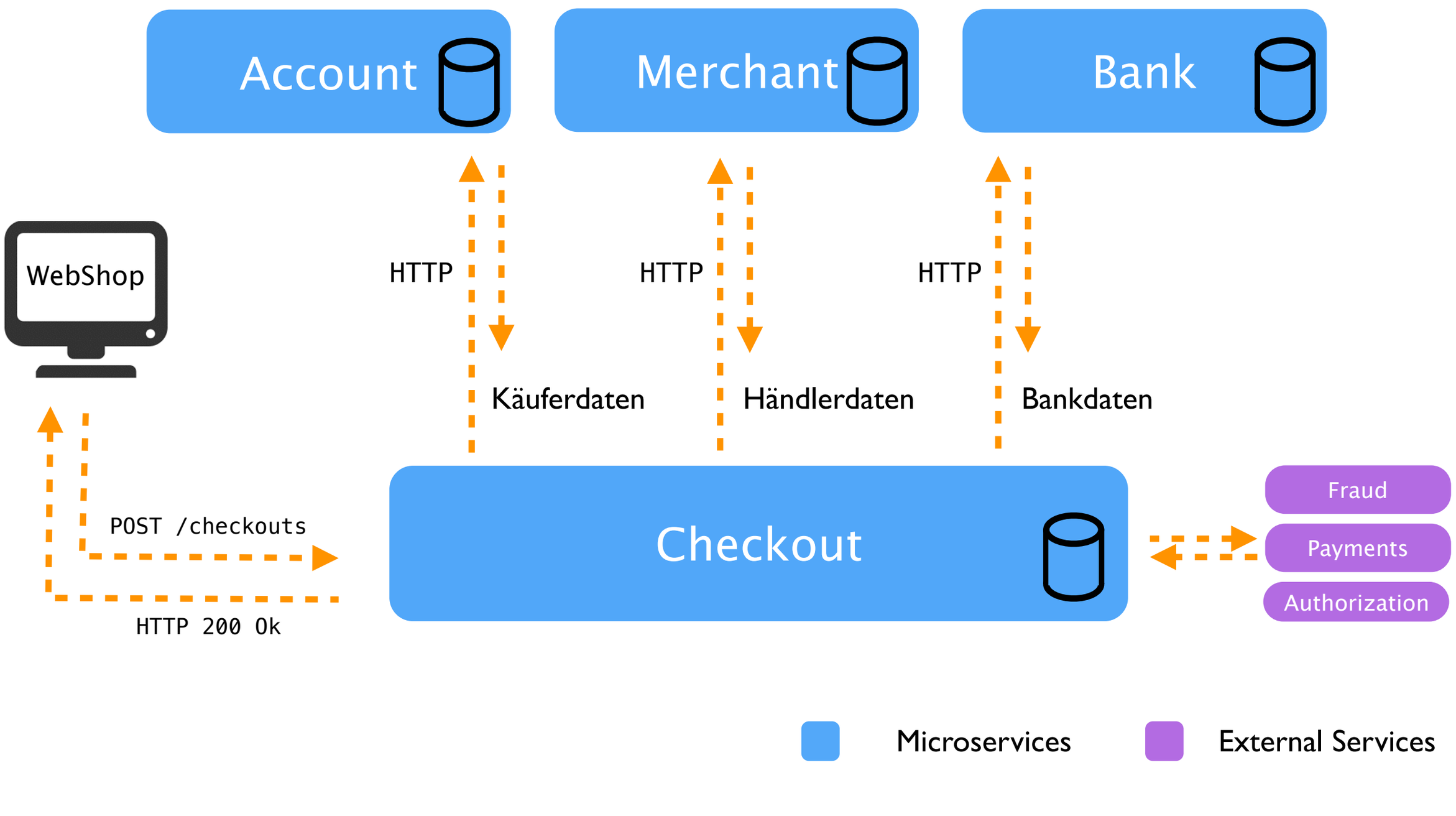
Event Store



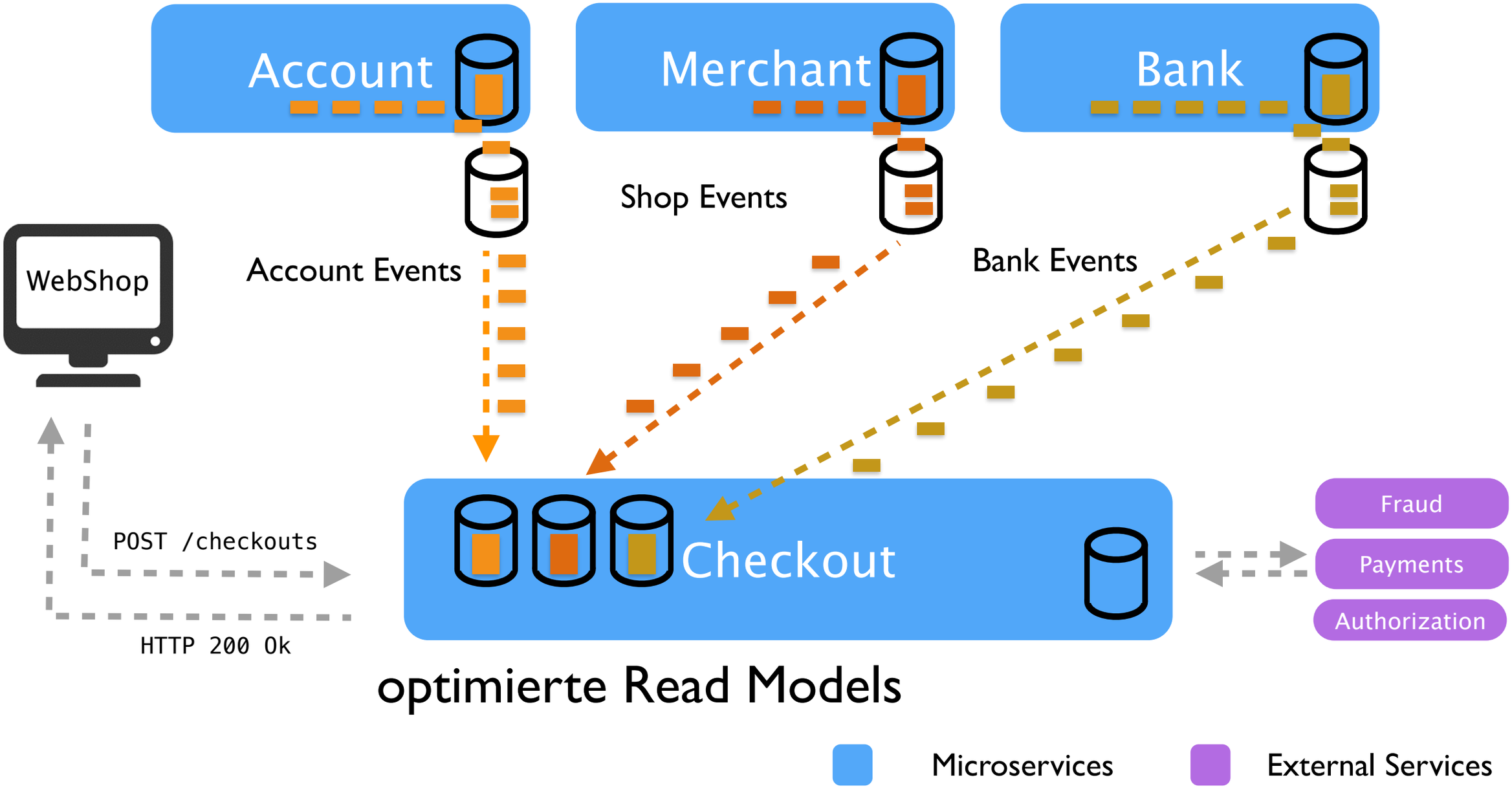


Current State

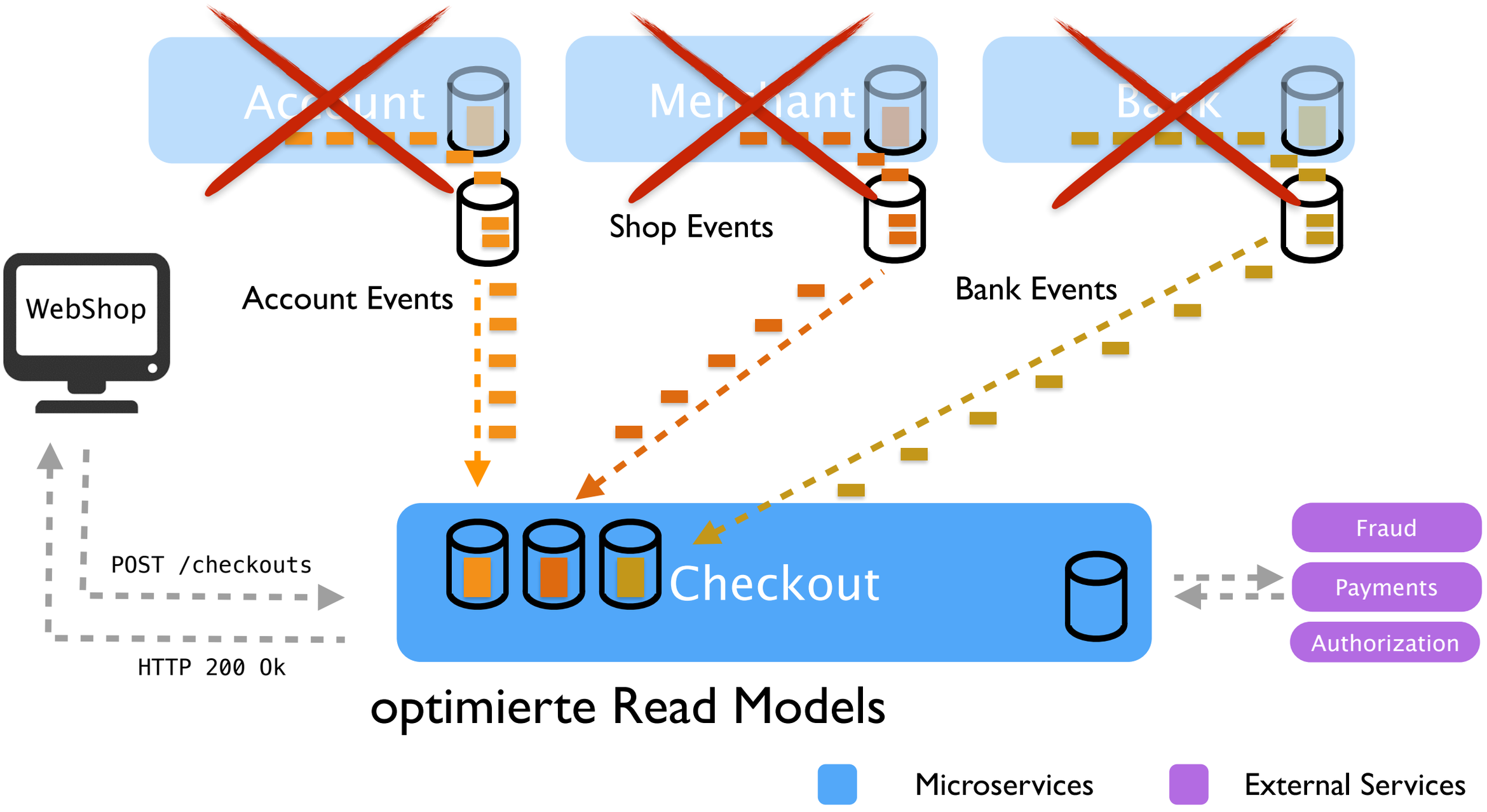
Event Sourcing?



Event Sourcing

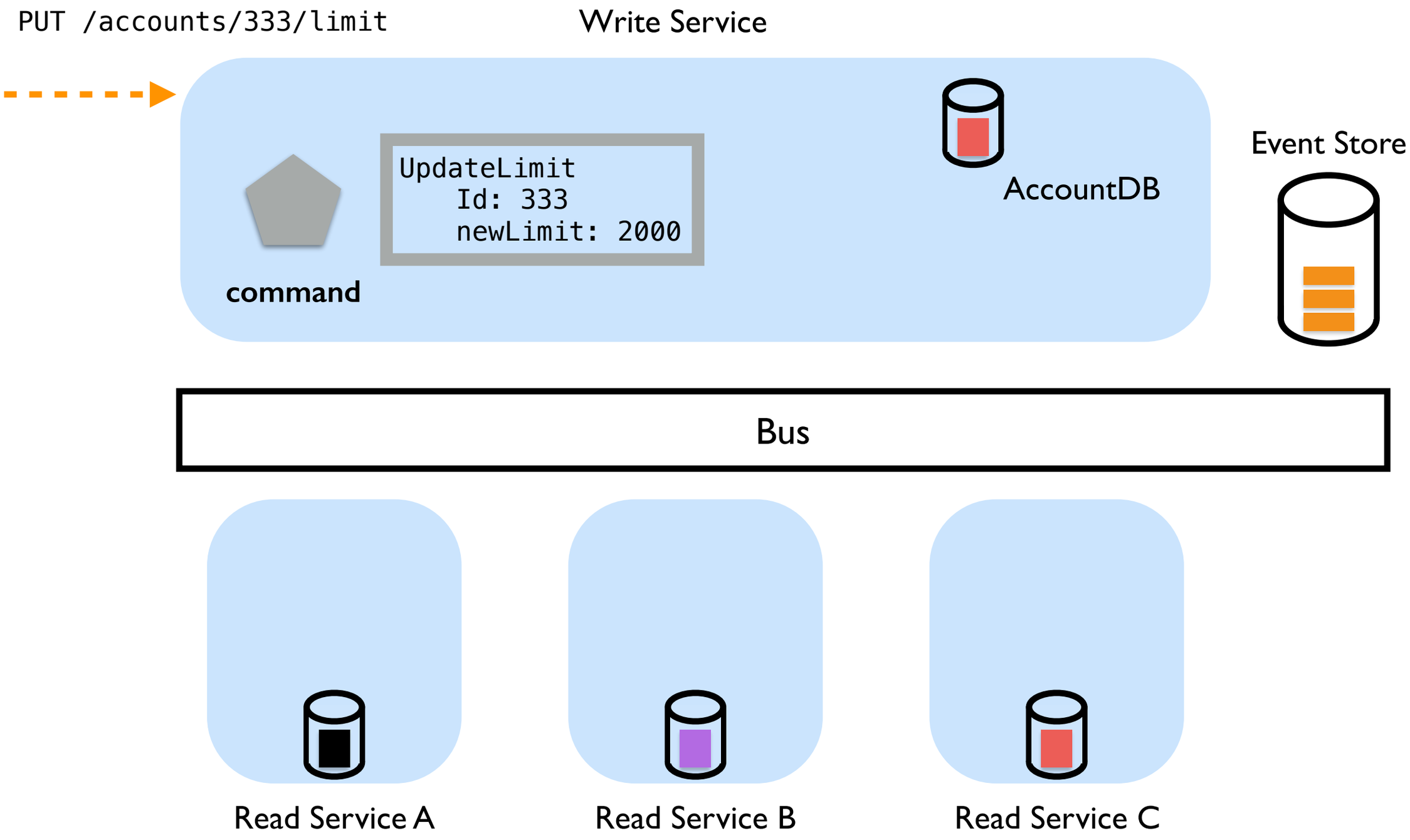


Event Sourcing



Implementierung?

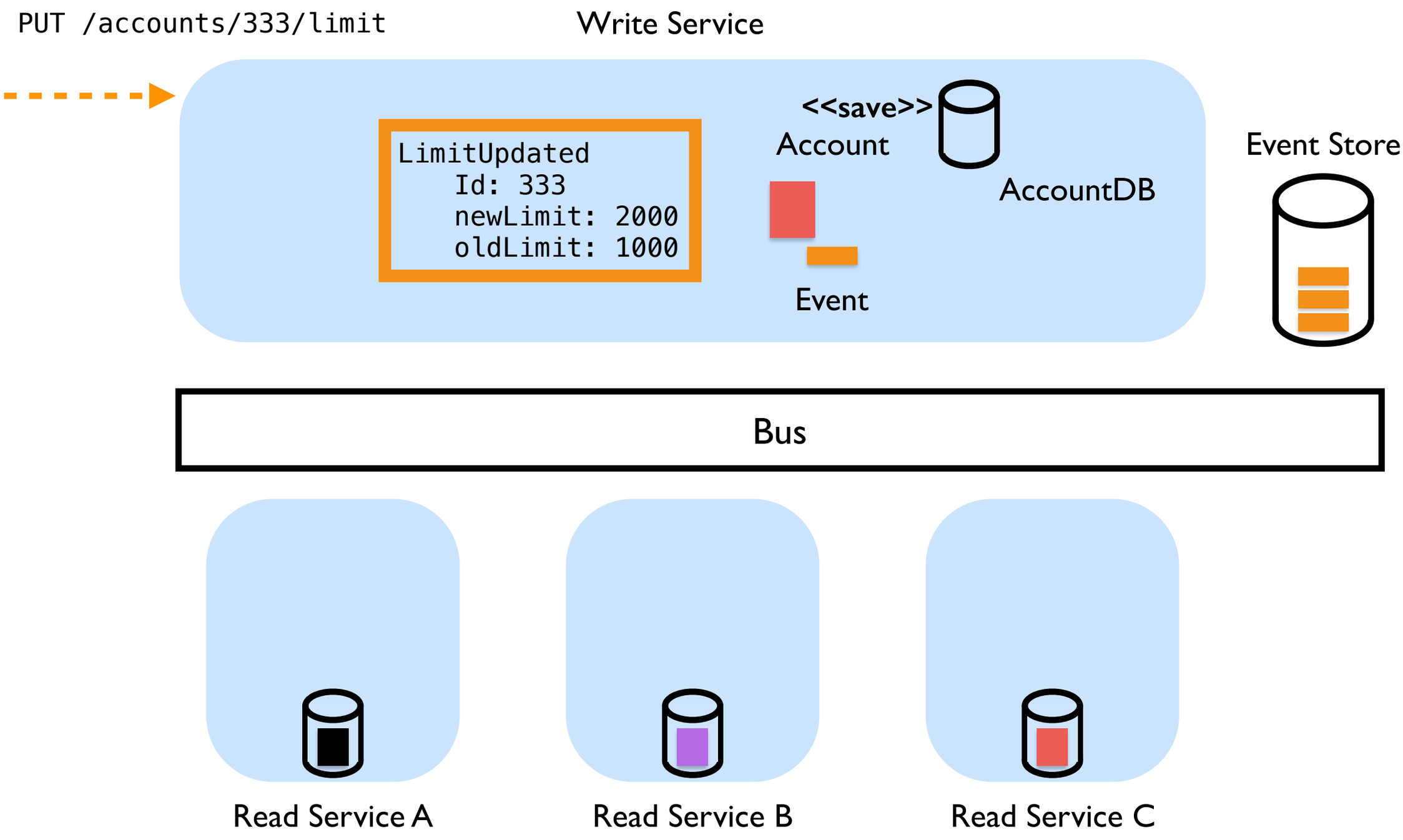
“Standardansatz”



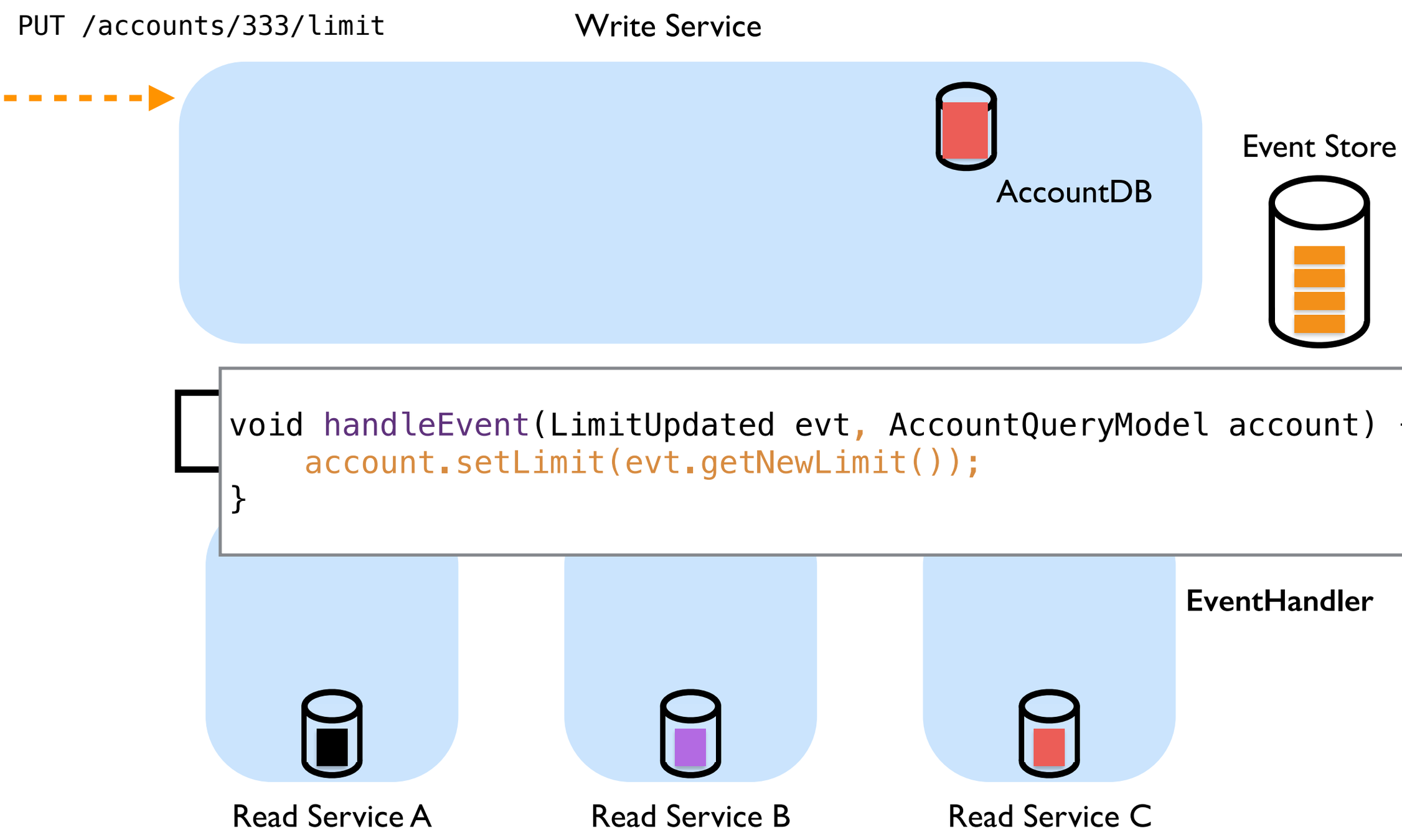
“Standardansatz”


```
Event handleCommand(UpdateLimit cmd, Account account) {
    if (account.isLocked)
        throw new IllegalStateException("Account locked");
    if (cmd.getLimit() > MAX_LIMIT)
        throw new UnsupportedOperationException("Limit exceeded")
    ...
    ...
    ...
    return new LimitUpdated(cmd.getId(), cmd.getLimit(), account.getLimit());
}
```

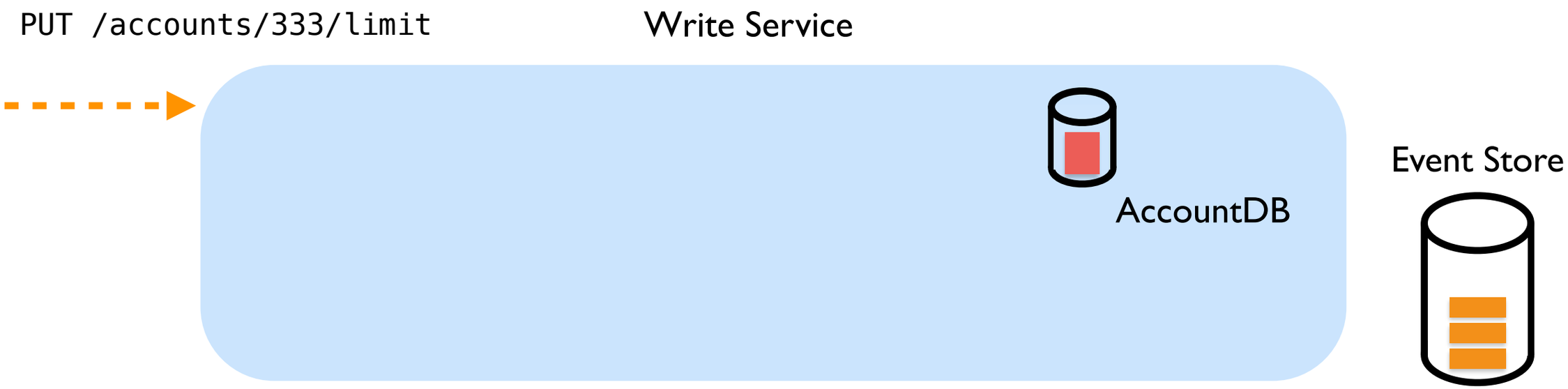
“Standardansatz”



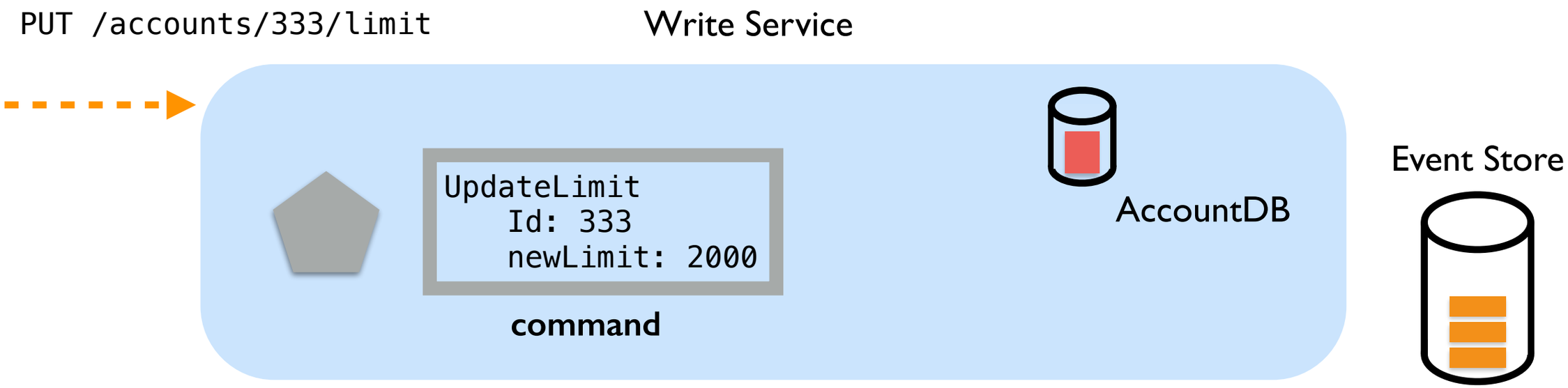
“Standardansatz”



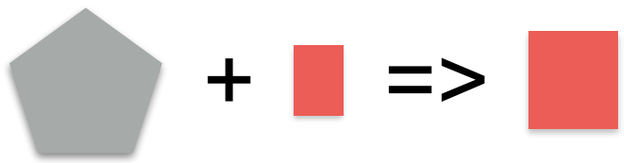
Write Service



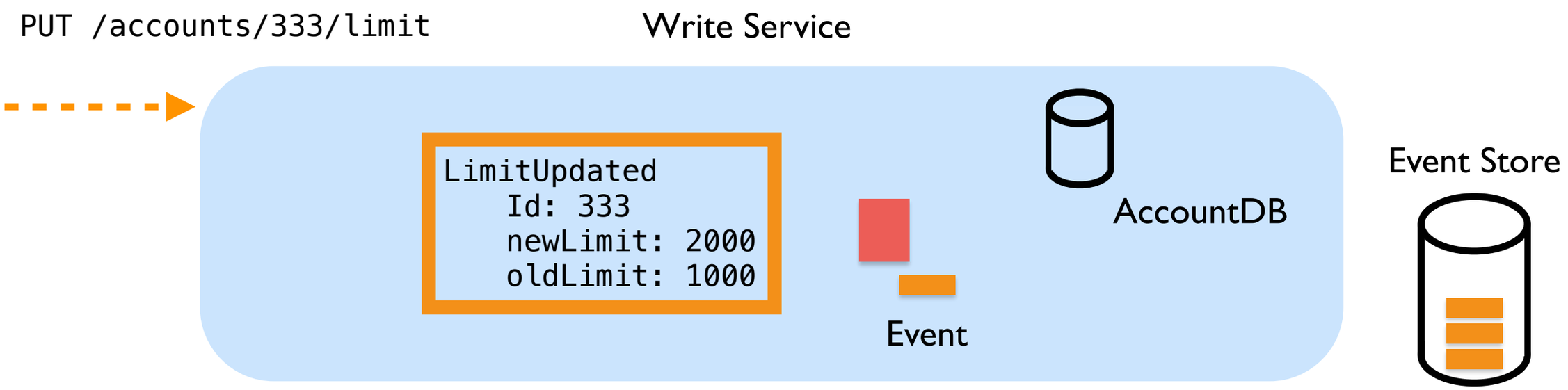
Write Service



Write Service

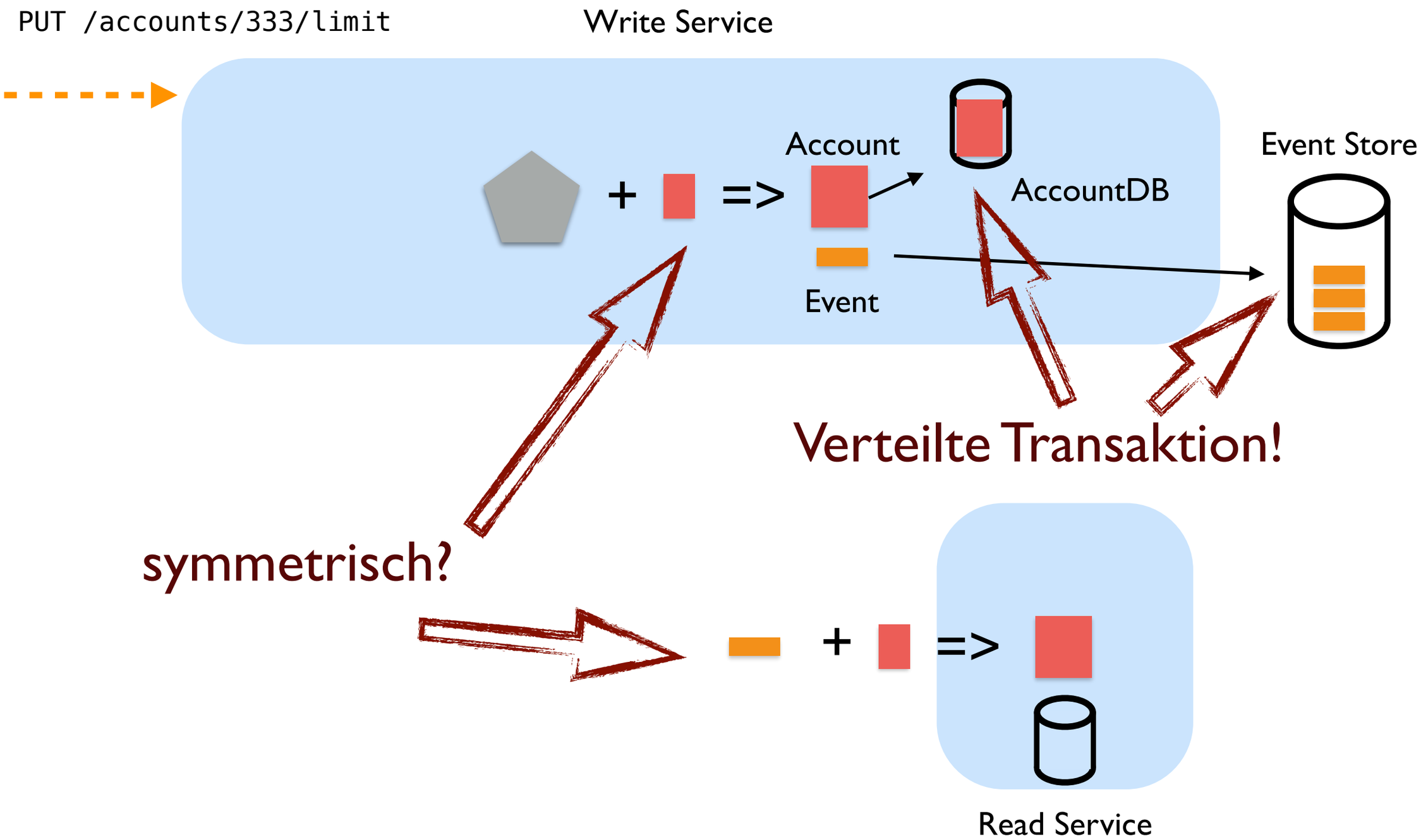
1. Command ändert Account State 

Write Service

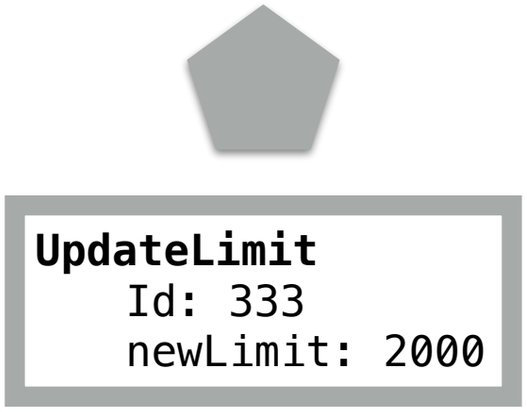


1. Command ändert Account State
2. Ableitung eines Events `diff( ■ , ■ ) =>` ▬
3. Speichern in AccountDB
4. Speichern in Event Store

Write Service



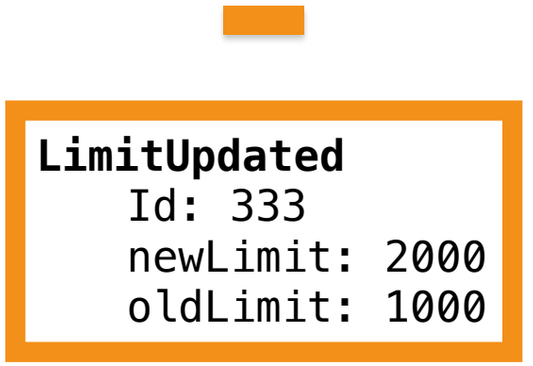
Commands vs Events



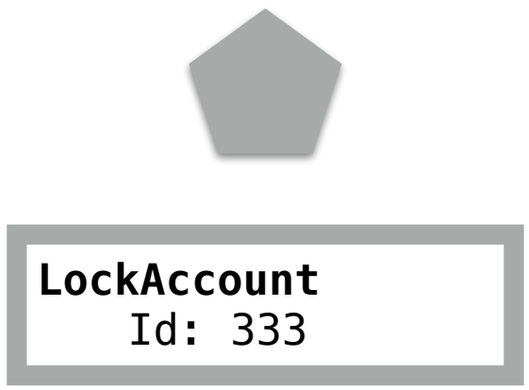
Was möchte der Client?

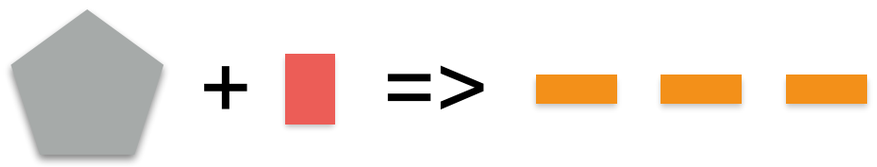
Wie interagiert der Client mit dem Server?

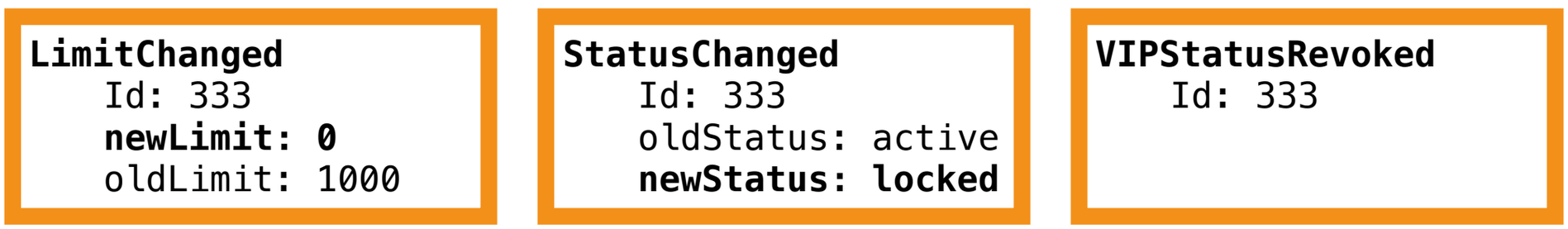
Welche Zustandsänderung(en)
wurde(n) durch den Command ausgelöst?



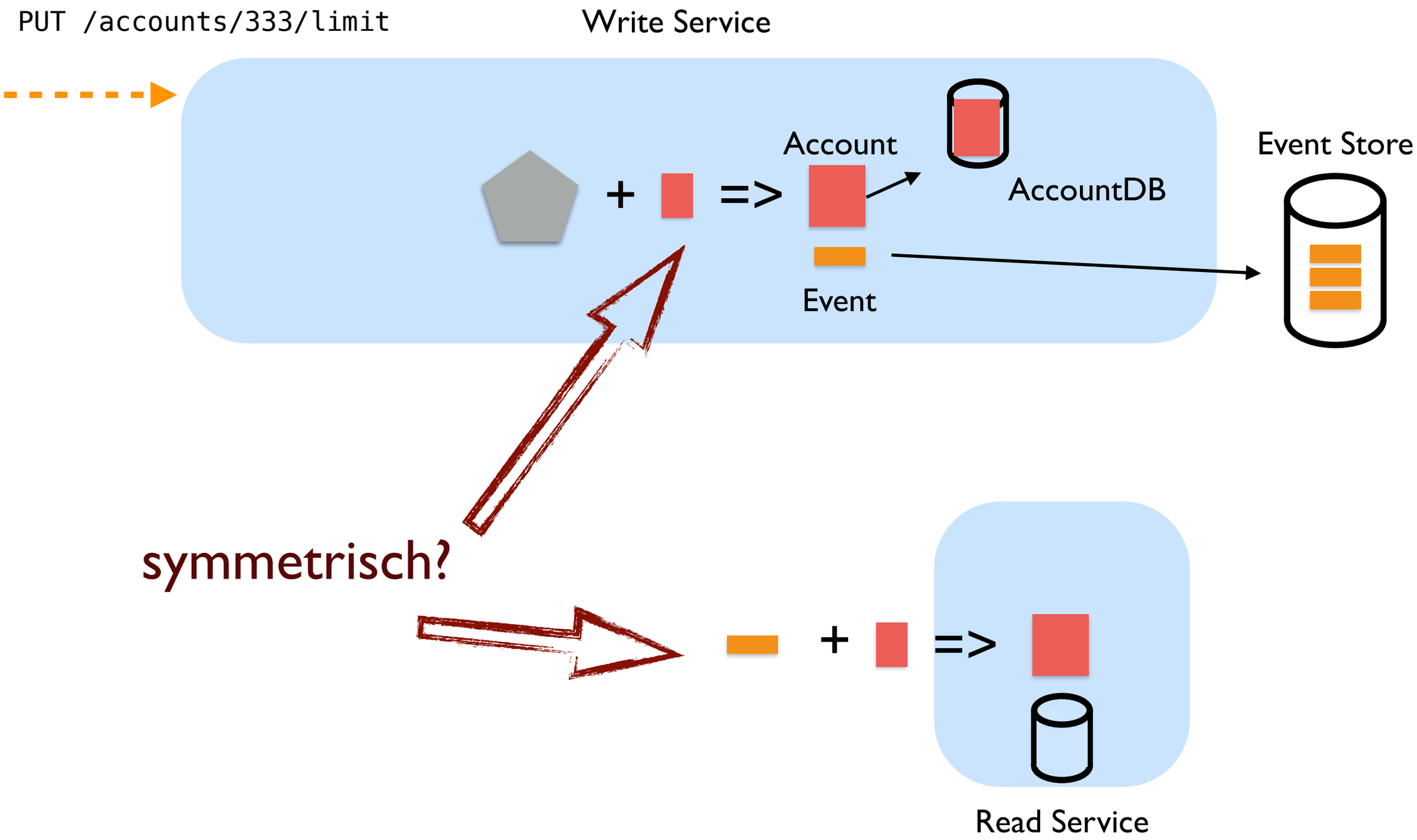
Commands vs Events



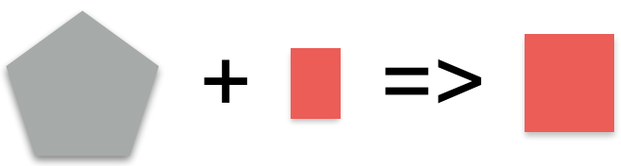




Write Service

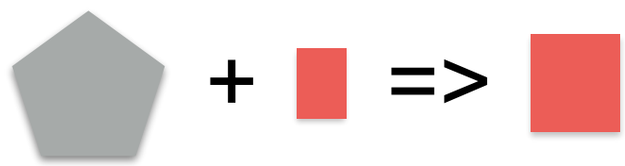


Commands dürfen keine Zustandsänderung bewirken!

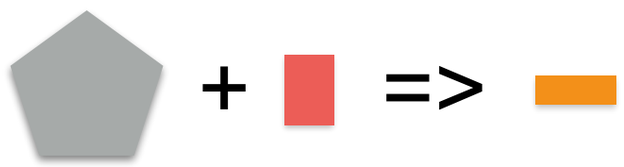


:-(

Events leiten sich aus Commands + Current State ab

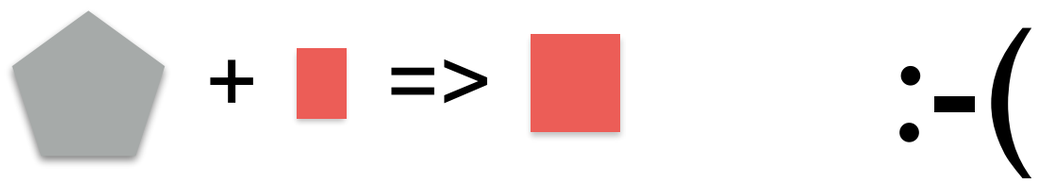


:-(

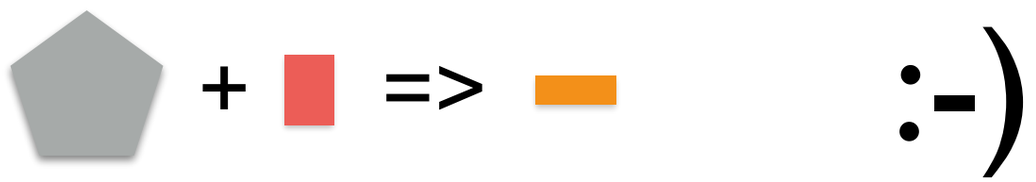


:-)

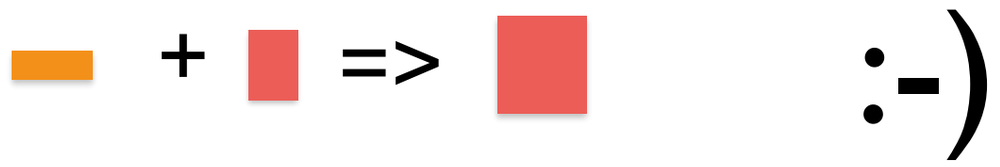
Events ändern den Zustand



:-(

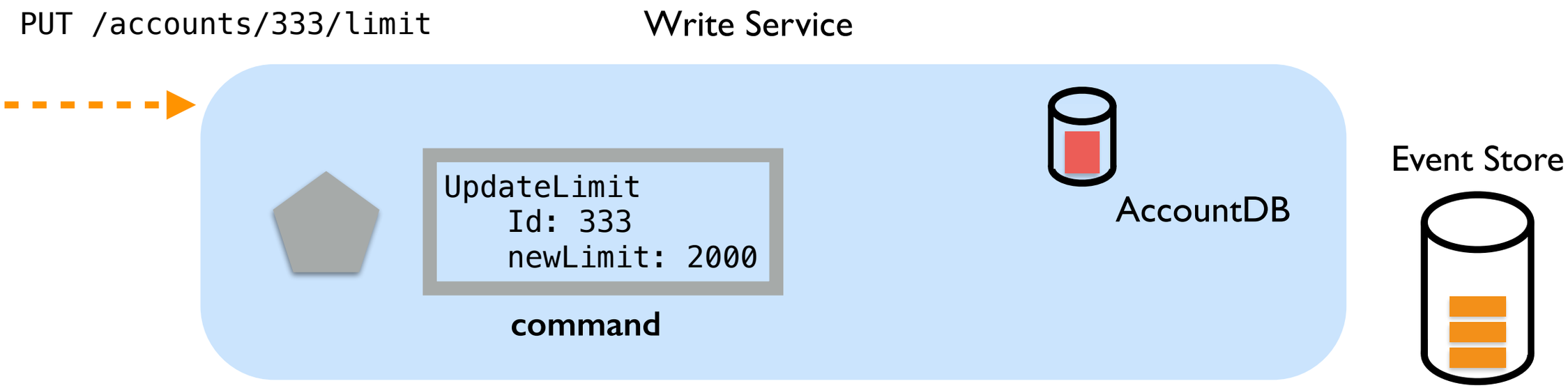


:-)



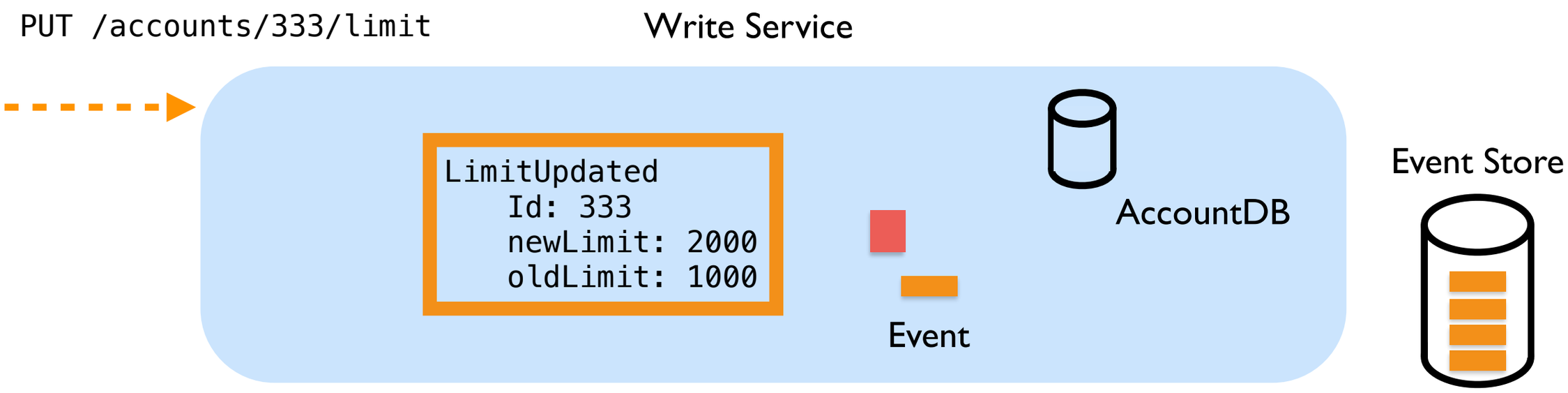
:-)

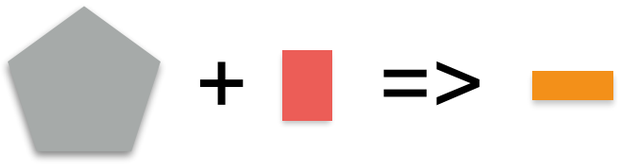
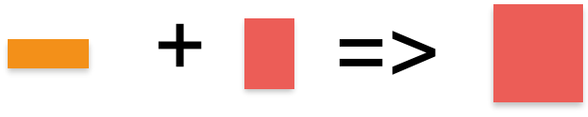
Write Service



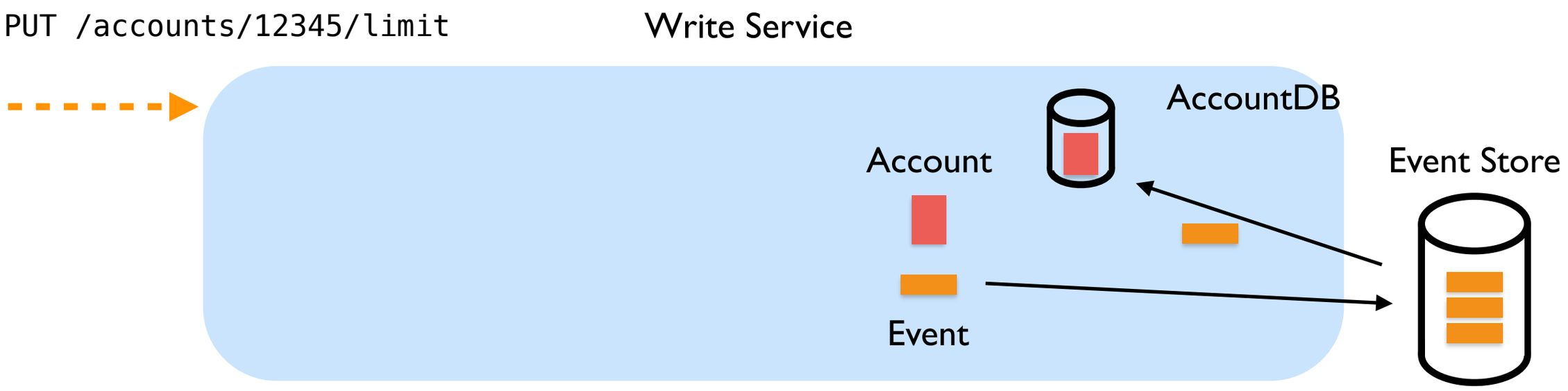
Write Service

Write Service



1. Event aus Command angeleitet 
2. Speichern in Event Store
3. Publishen der Events zur AccountDB
4. Events ändern Current State 

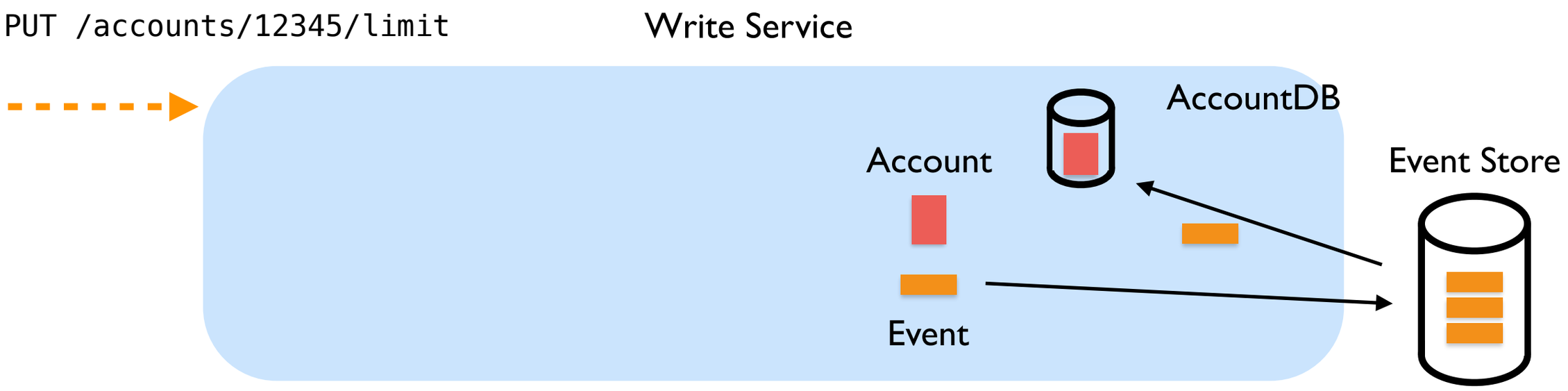
Write Service



AccountDB als ‘Projection’ der Events

Keine verteilten Transaktionen

Write Service



AccountDB als ‘Projection’ der Events

Keine verteilten Transaktionen

Race Condition!

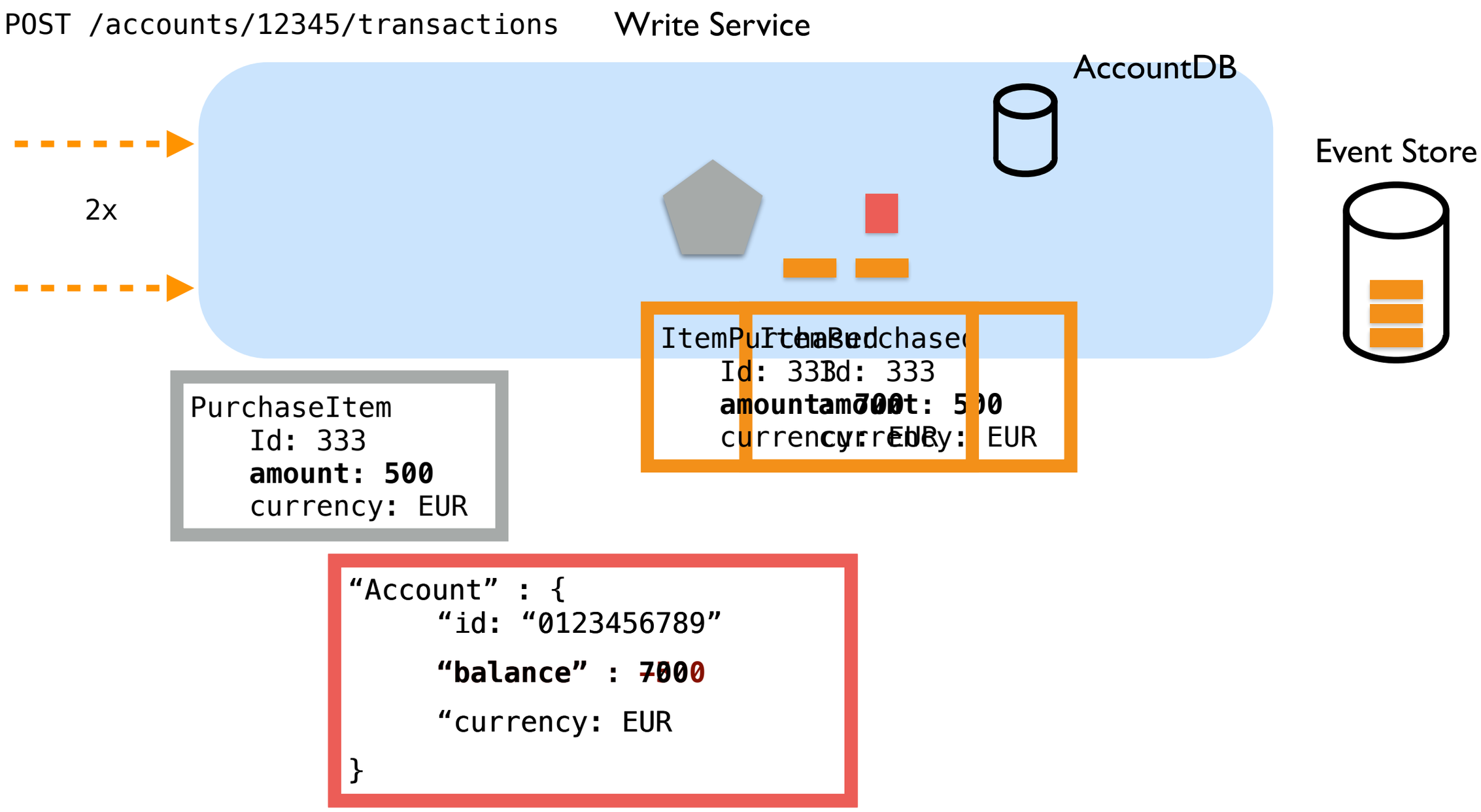
Race Condition


```
PurchaseItem
    Id: 333
    amount: 500
    currency: EUR
```

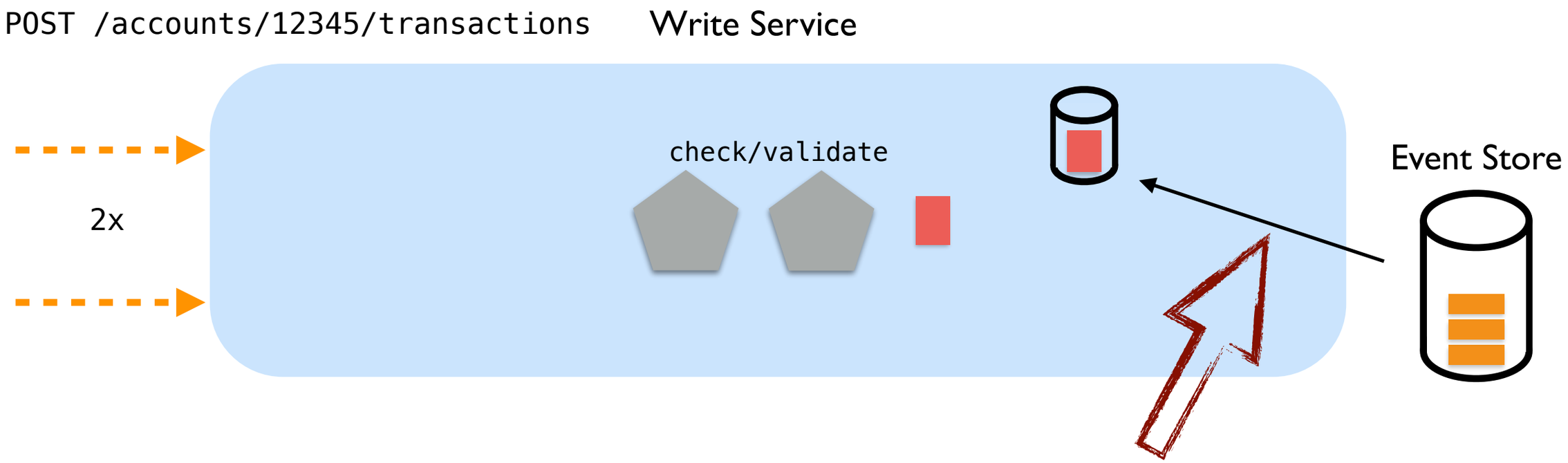
```
PurchaseItem
    Id: 333
    amount: 700
    currency: EUR
```

```
“Account” : {
     “id: “0123456789”
     “balance” : 700
     “currency: EUR
}
```

Race Condition

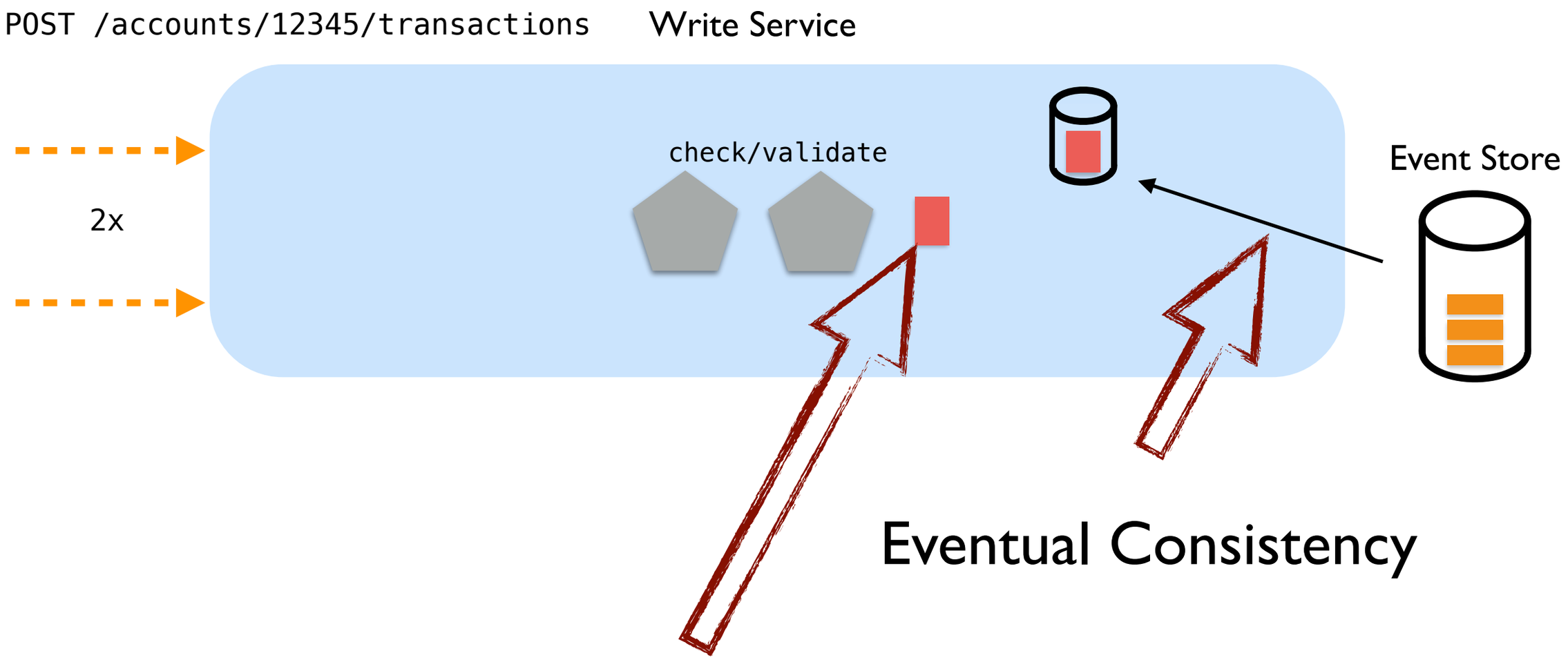


Race Condition



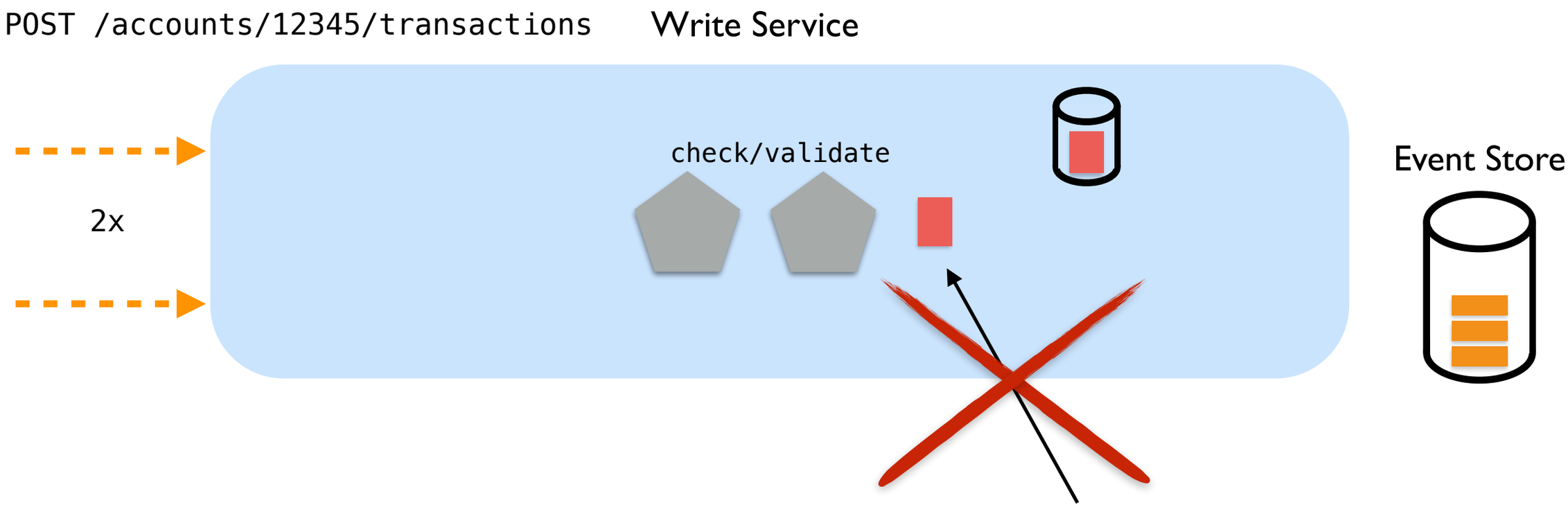
Eventual Consistency

Race Condition



Validierung gegen out-of-date Account

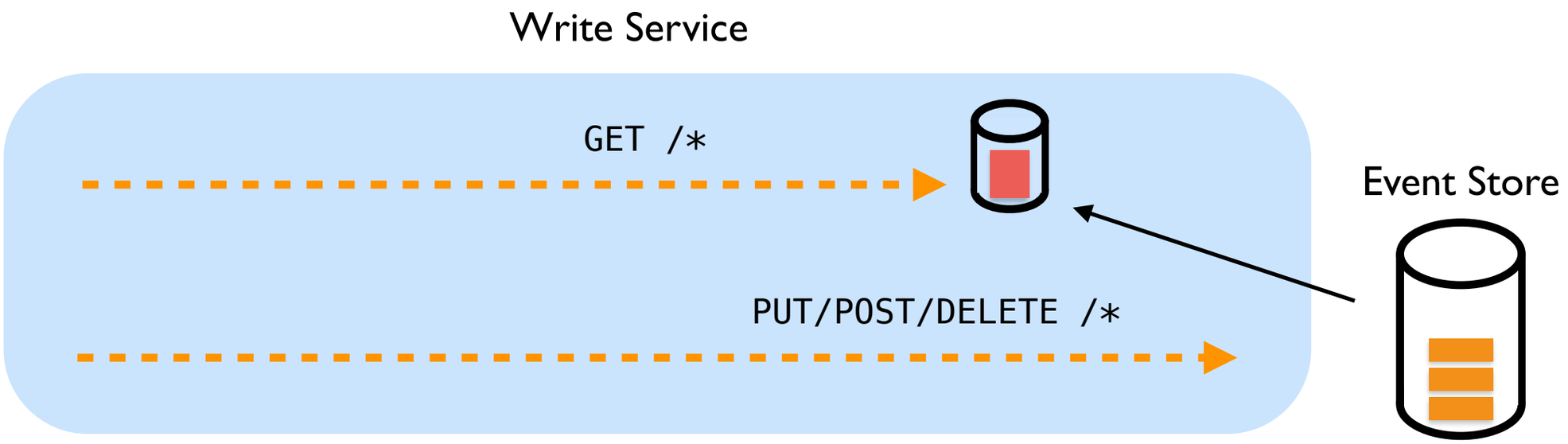
Race Condition



Current State nicht für Plausibilisierung verwenden

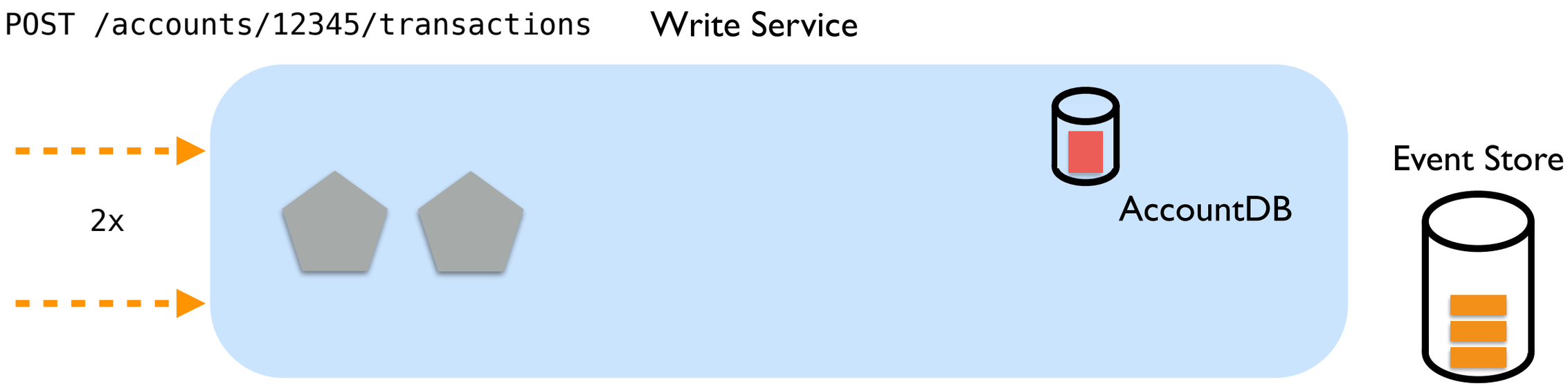
AccountDB Optimierung für lesenden Zugriff

Read - Write

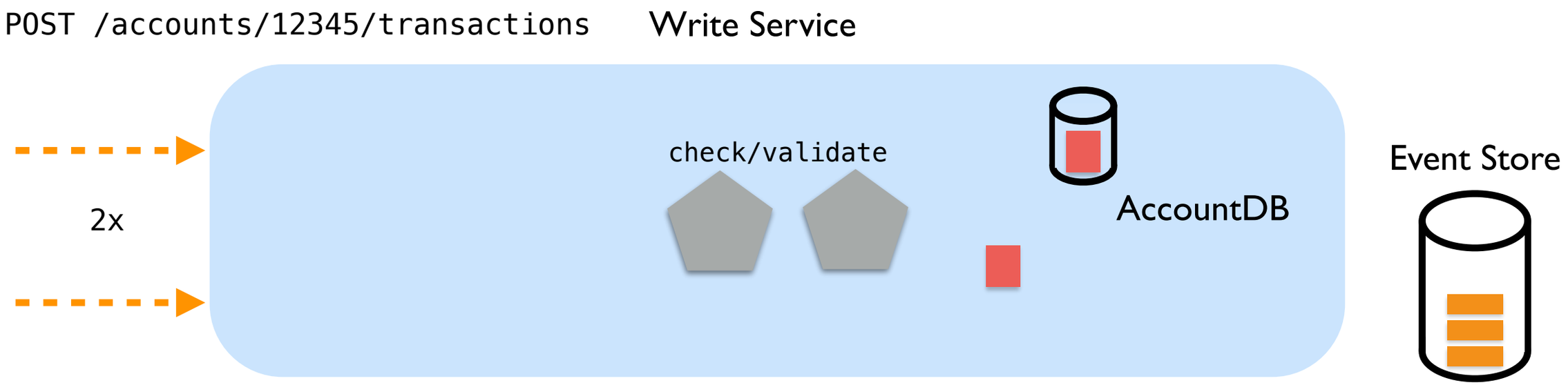


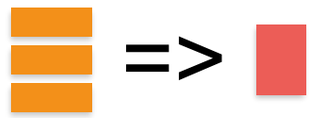
für lesenden Zugriff optimiert

Write Service 3

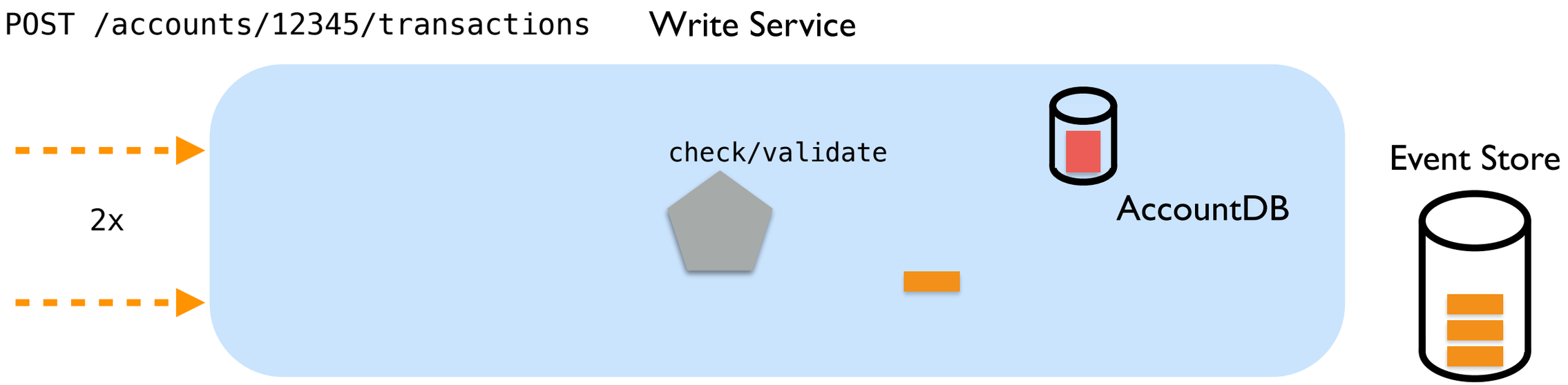


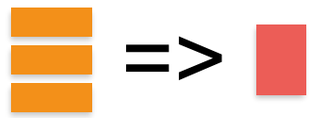
Write Service 3



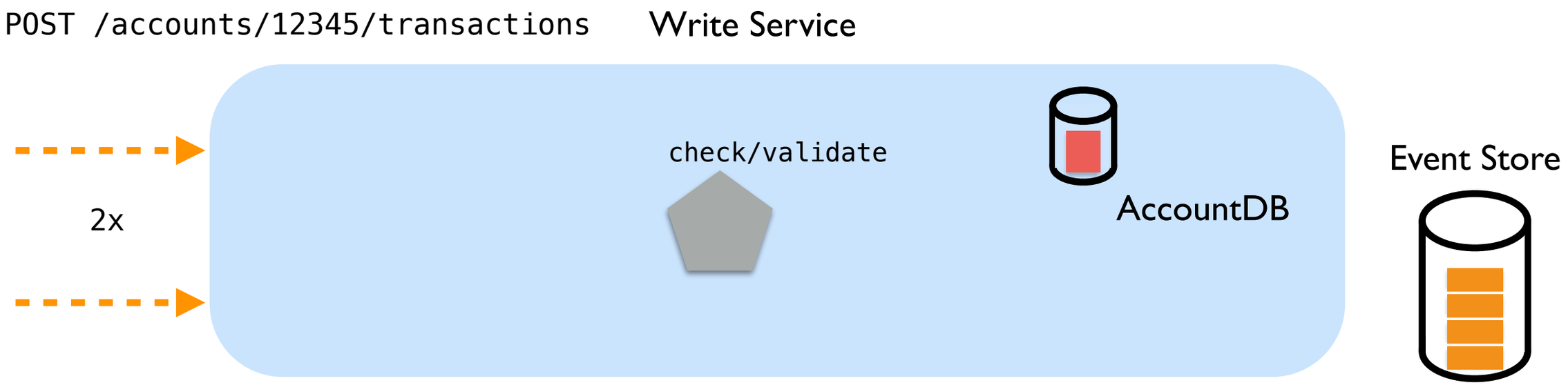
1. Erzeugung des current state aus Events  =>

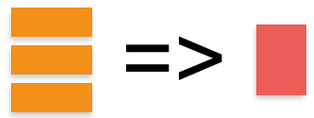
Write Service 3



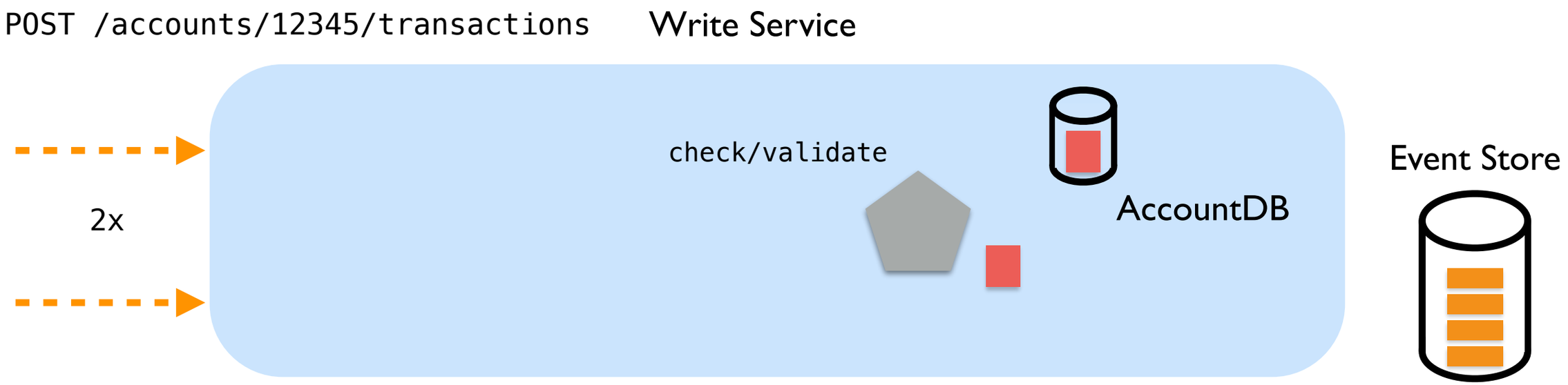
1. Erzeugung des current state aus Events  =>

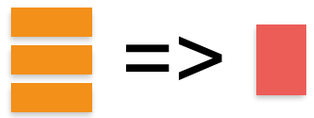
Write Service 3



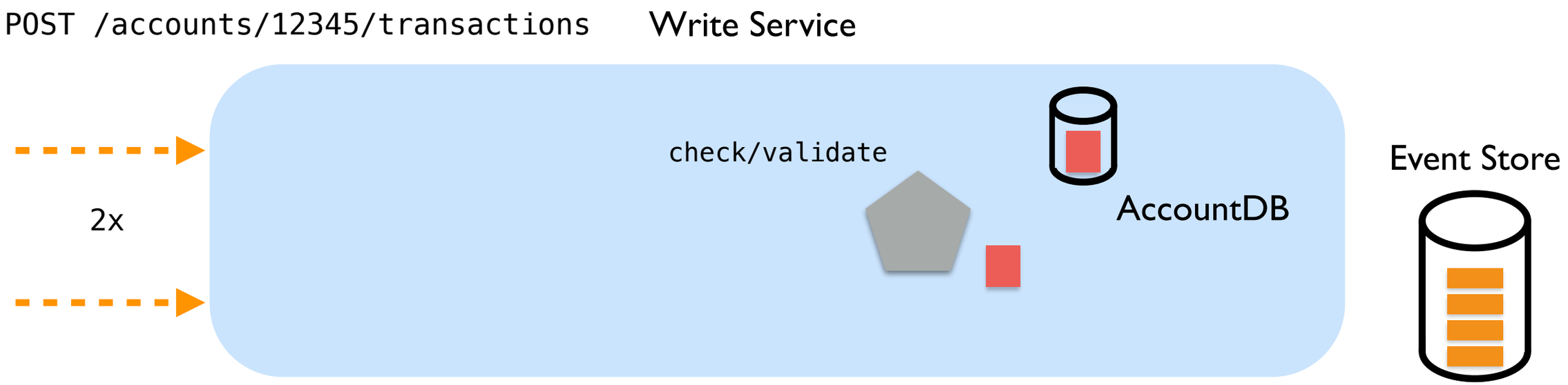
1. Erzeugung des current state aus Events  =>

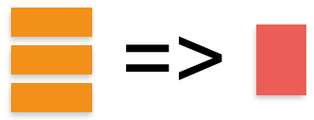
Write Service 3

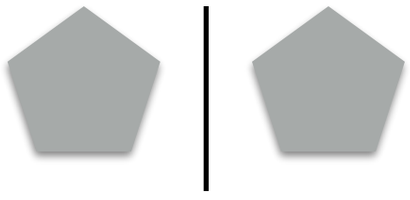


1. Erzeugung des current state aus Events  =>

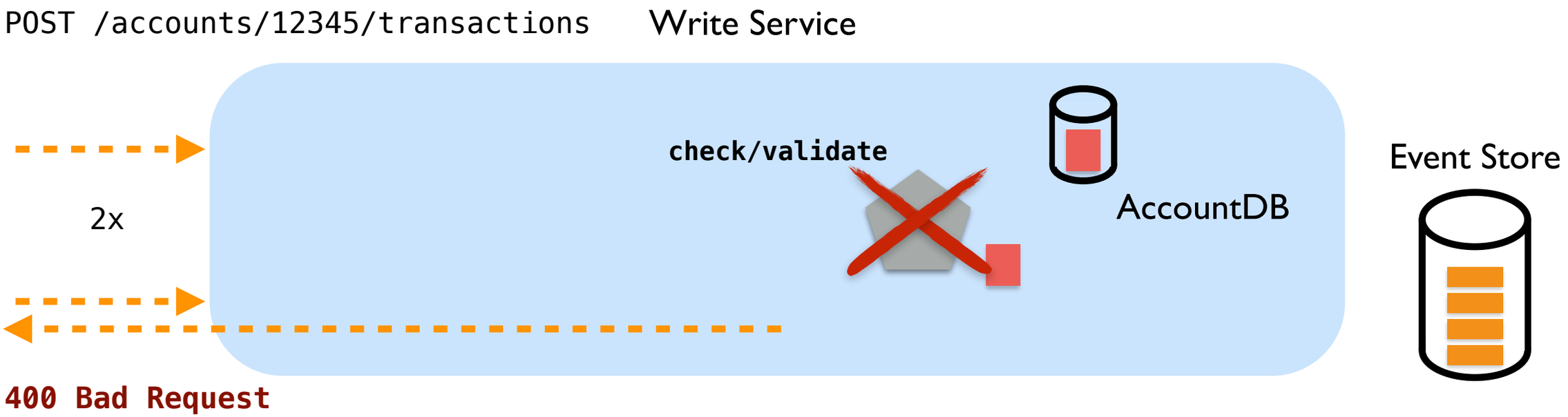
Write Service 3

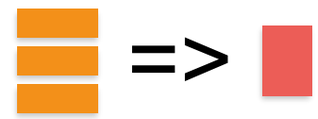


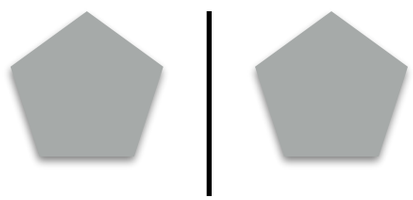
1. Erzeugung des current state aus Events 

2. Sequenzielles Validieren/Plausibilisieren 

Write Service 3

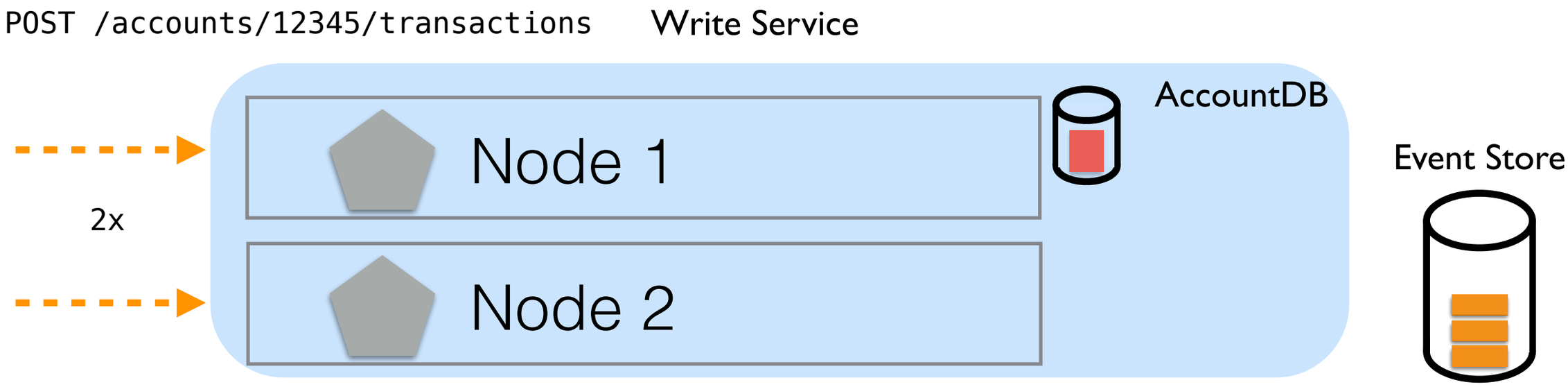


1. Erzeugung des current state aus Events 

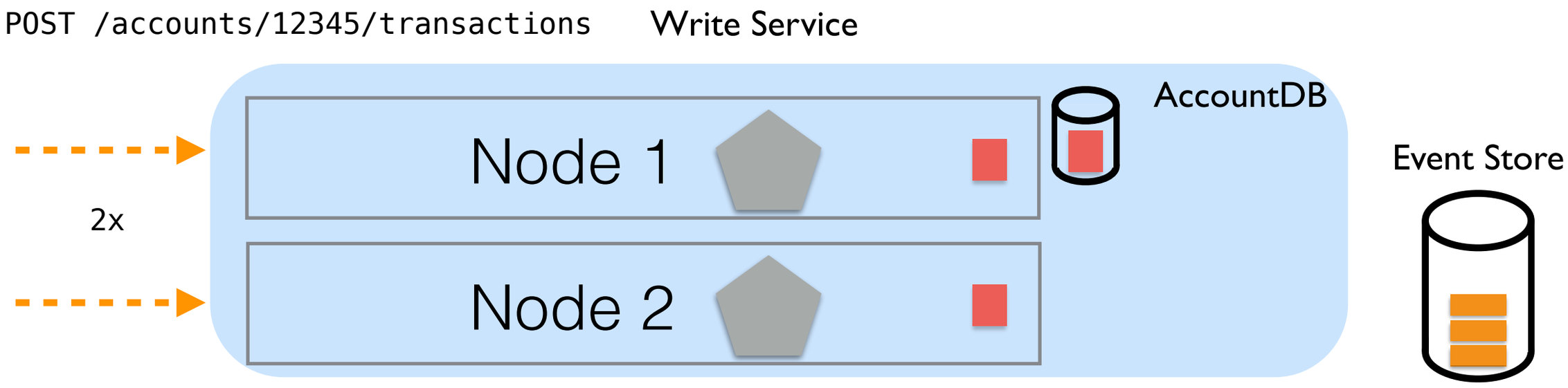
2. Sequenzielles Validieren/Plausibilisieren 

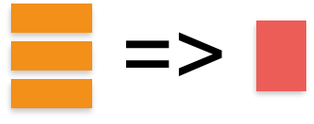
Event Sourcing im Cluster?

Cluster


POST /accounts/12345/transactions
Write Service
2x
Node 1
Node 2
AccountDB
Event Store

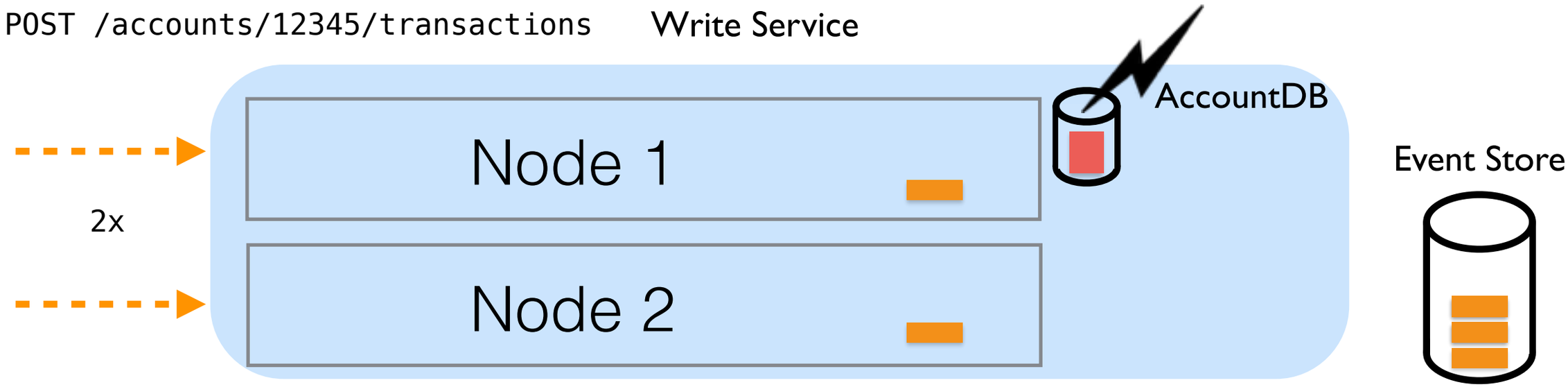
Cluster



Erzeugung des current state aus Events 

Validierung auf unterschiedlichen Knoten im Cluster

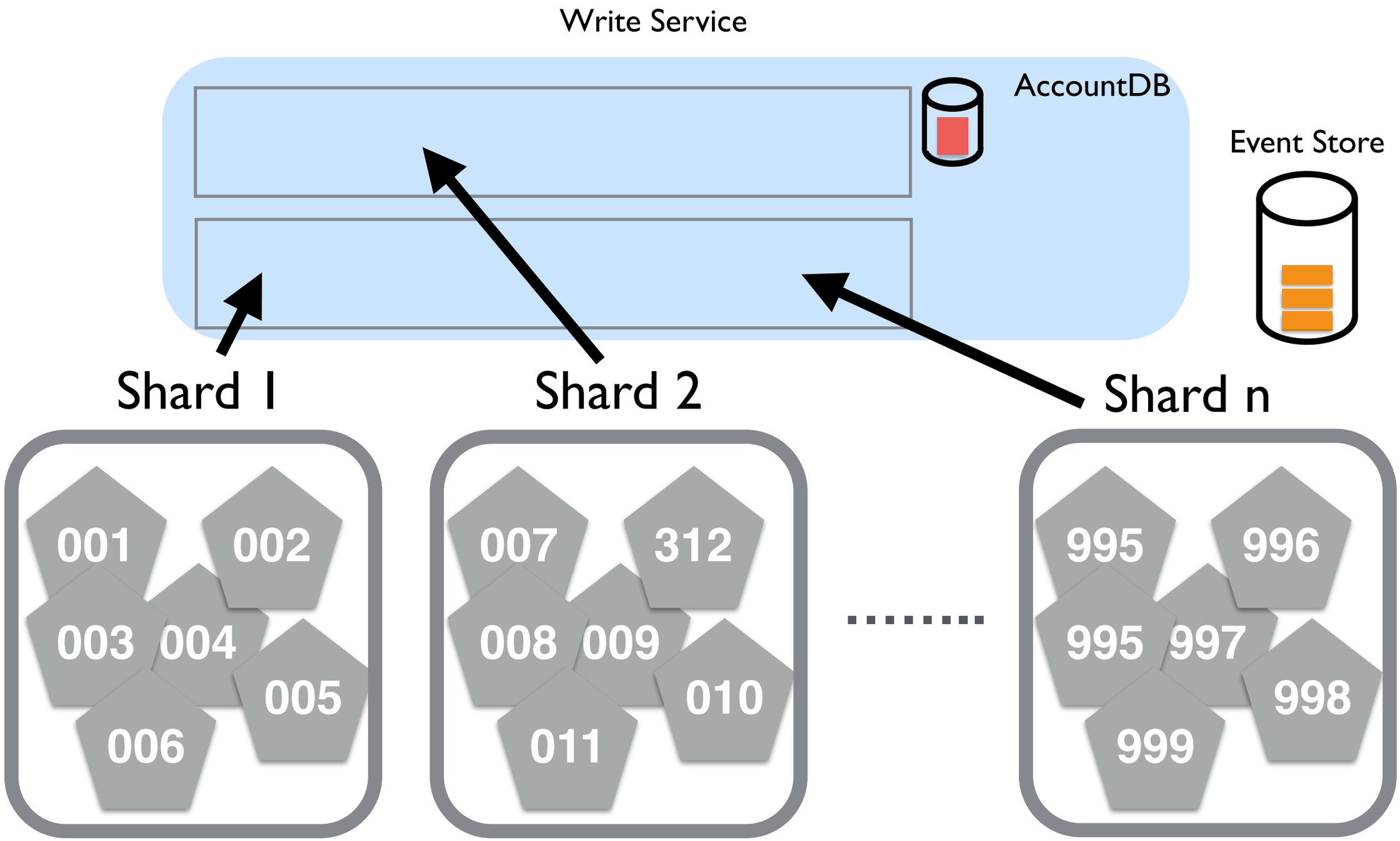
Cluster



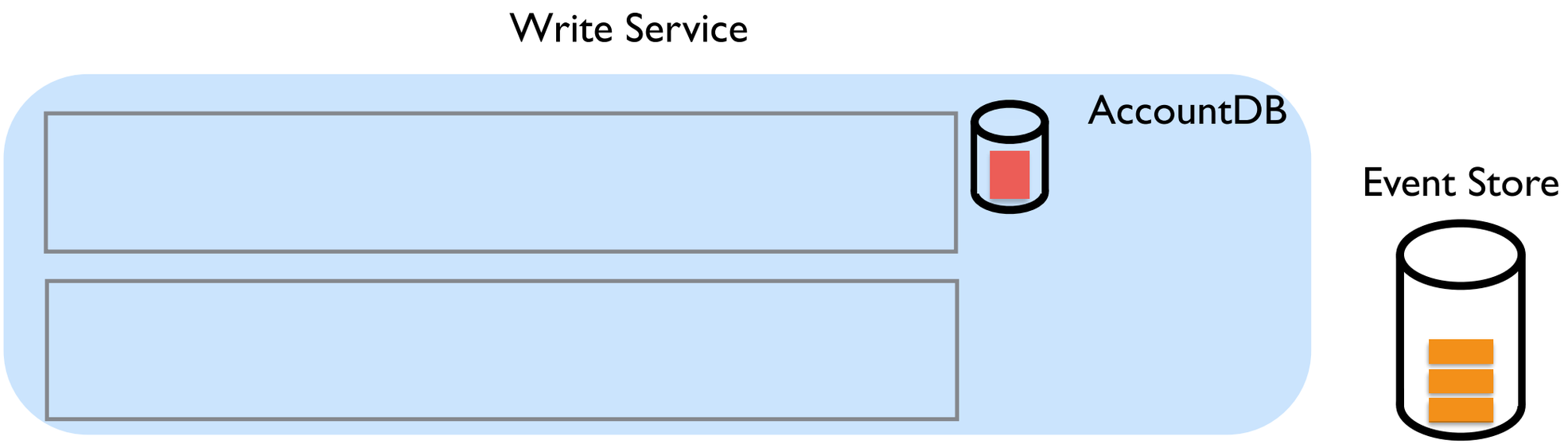

```
“Account” : {
      “id: “0123456789”
      “balance” : -500
      “currency: EUR
}
```

Cluster Sharding

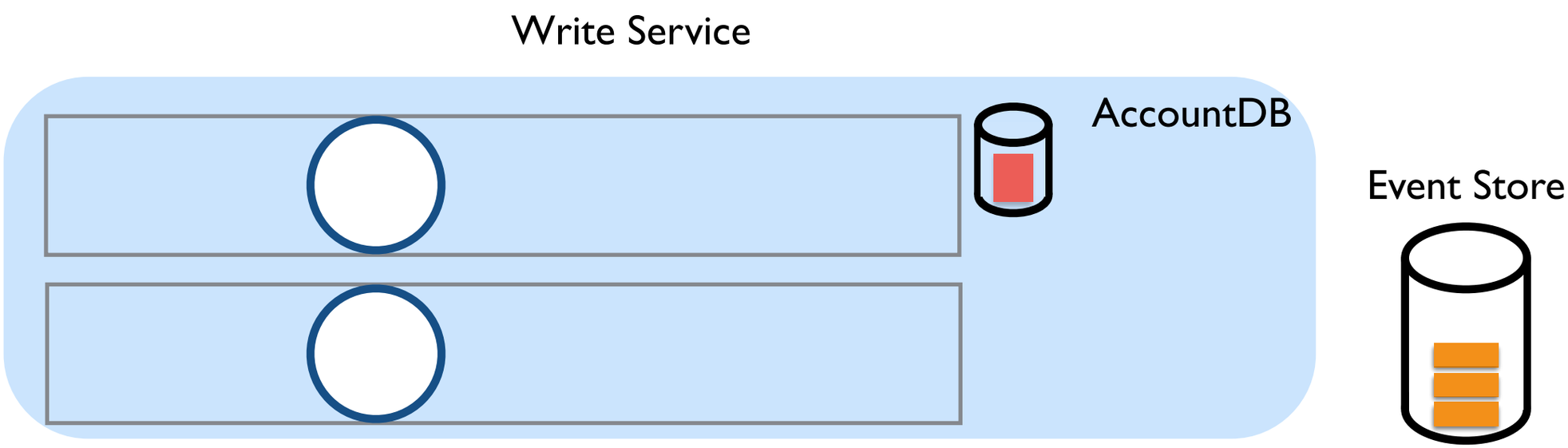
Cluster Sharding

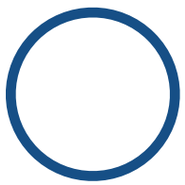


Cluster Sharding mit Akka



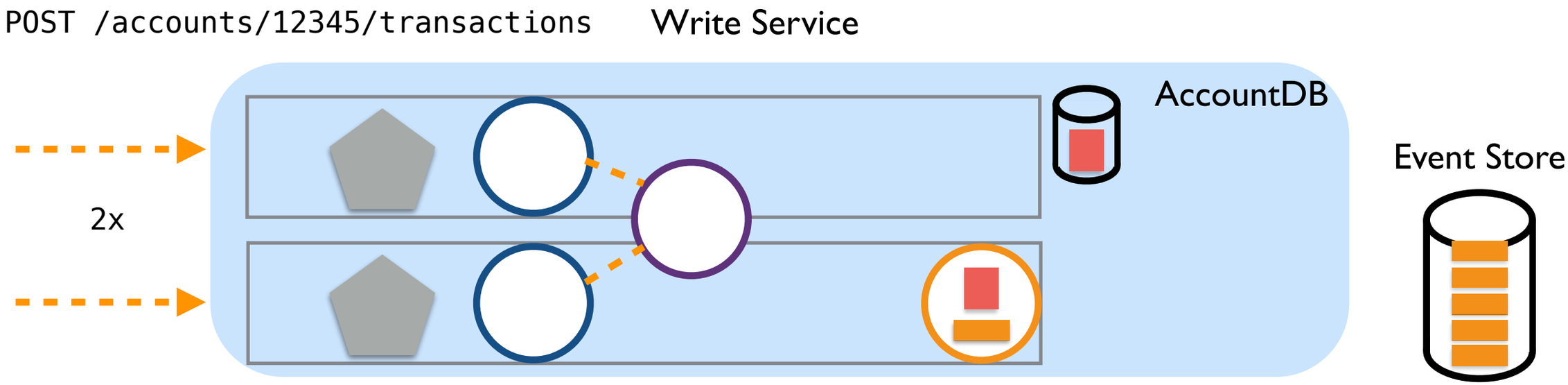
Cluster Sharding mit Akka





ShardRegion (Actor)

Cluster Sharding mit Akka

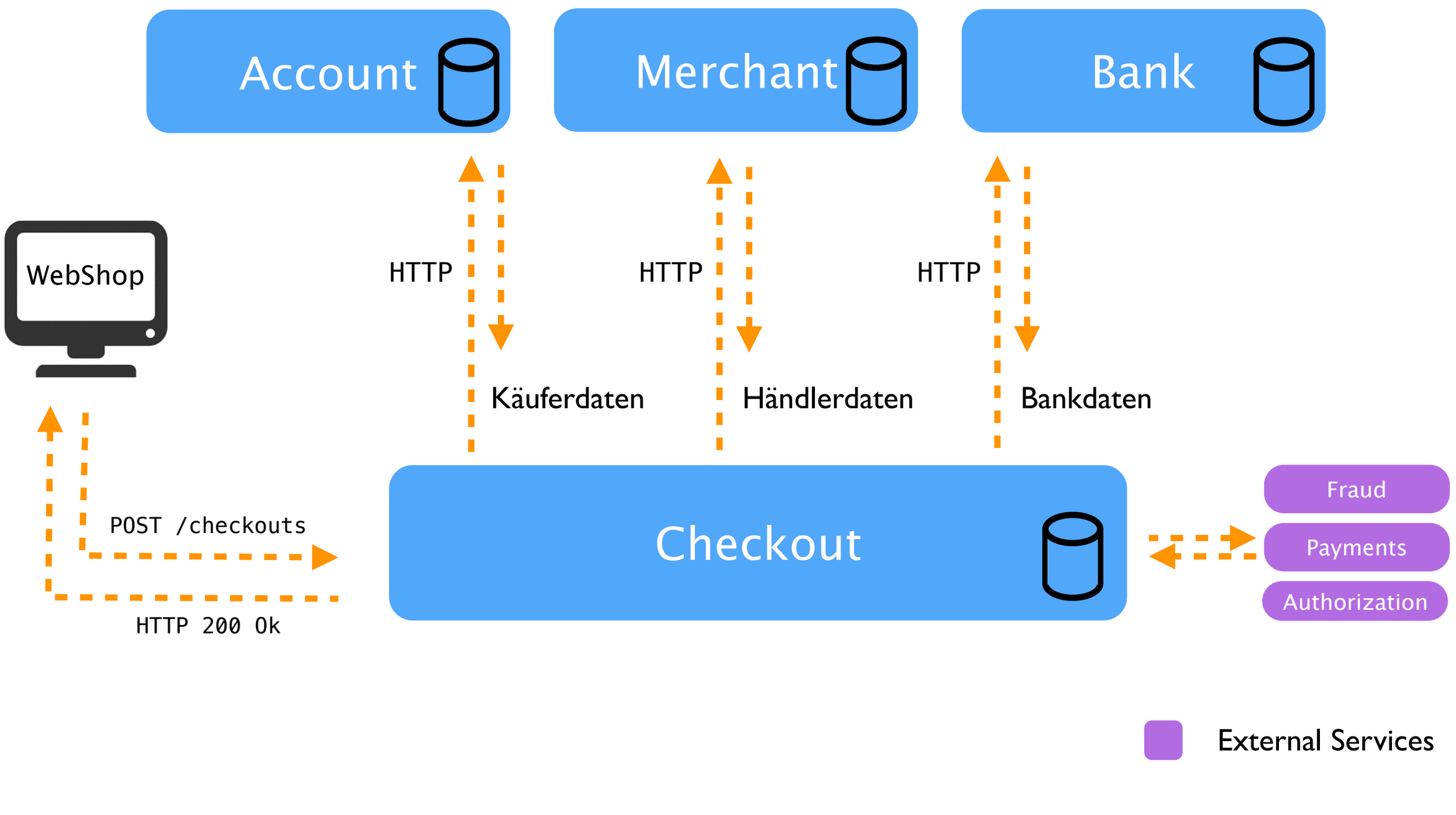


ShardRegion (Actor)

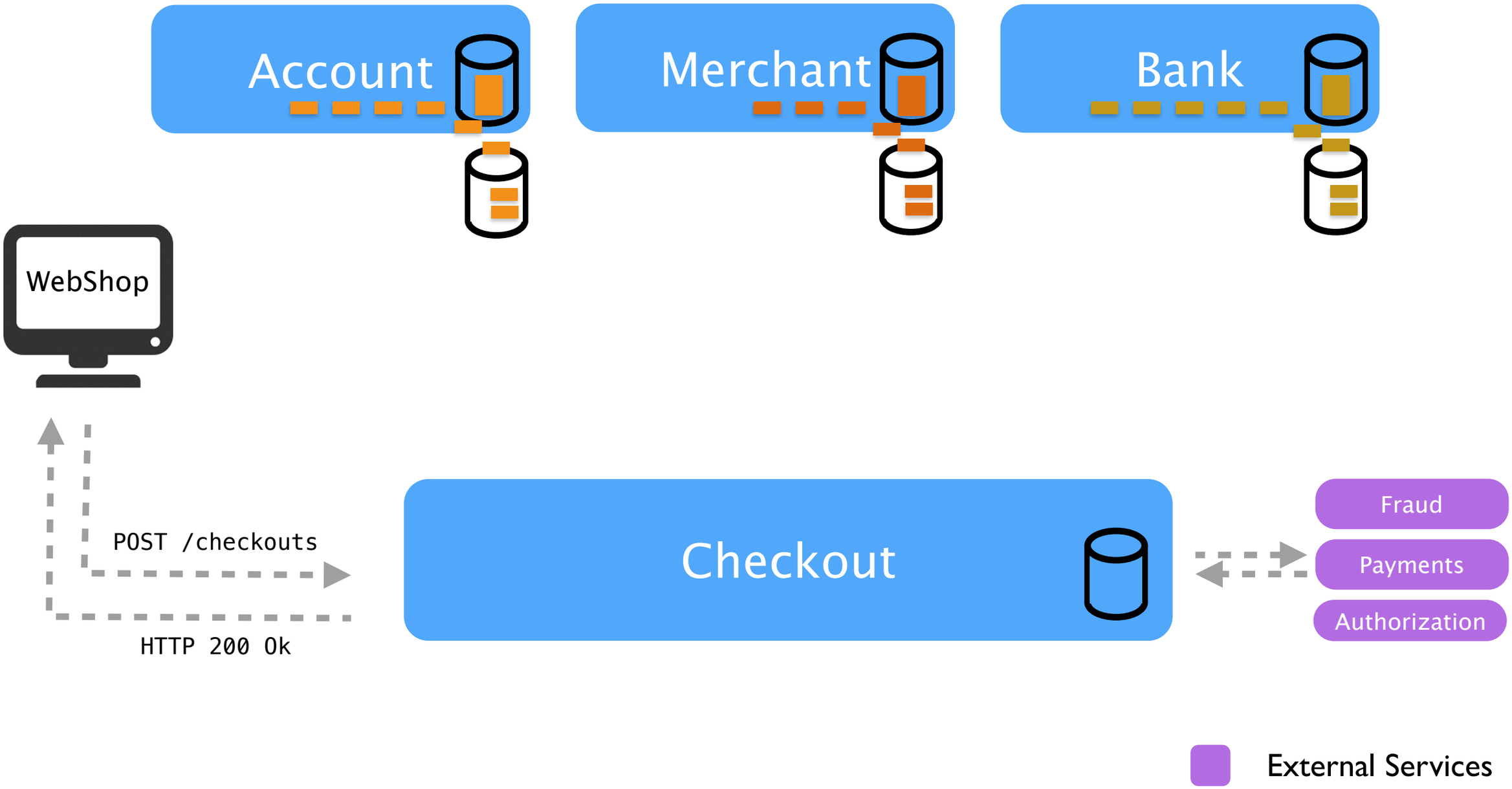
ShardCoordinator (Actor, ClusterSingleton)

PersistentActor

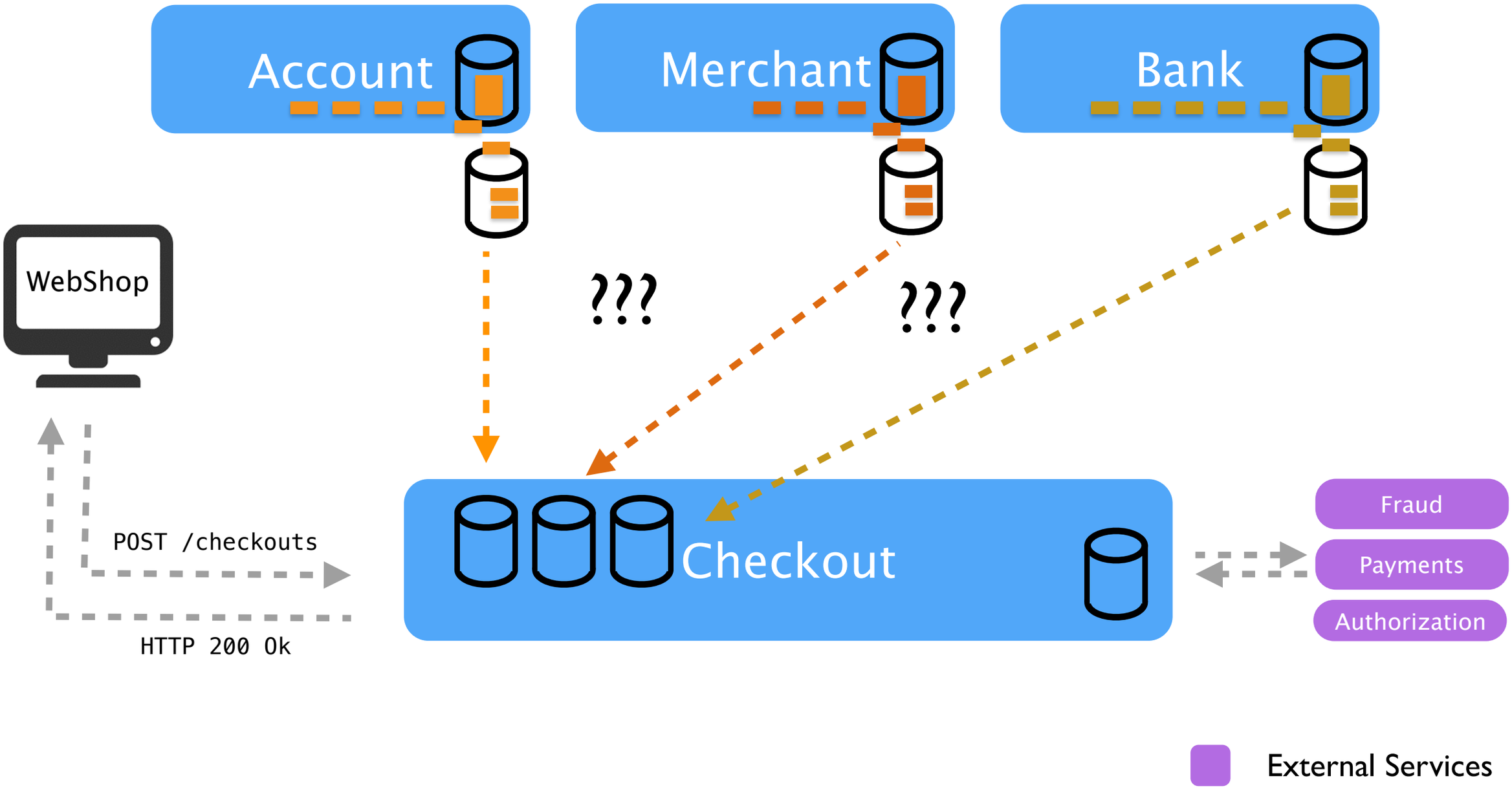
:-(


Account
Merchant
Bank
WebShop
HTTP
HTTP
HTTP
Käuferdaten
Händlerdaten
Bankdaten
POST /checkouts
HTTP 200 Ok
Checkout
Fraud
Payments
Authorization
External Services

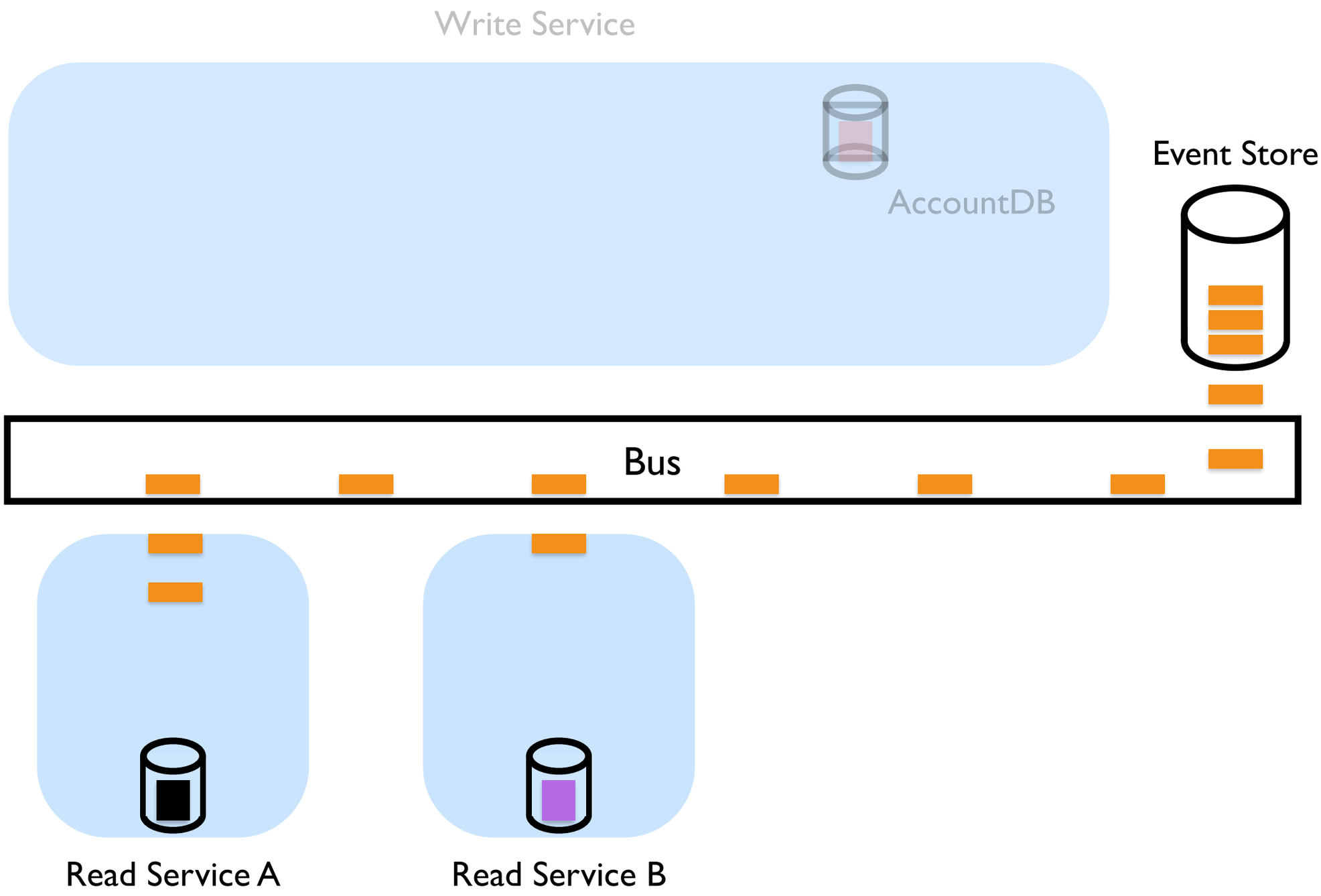
Stammdaten Eventsourced



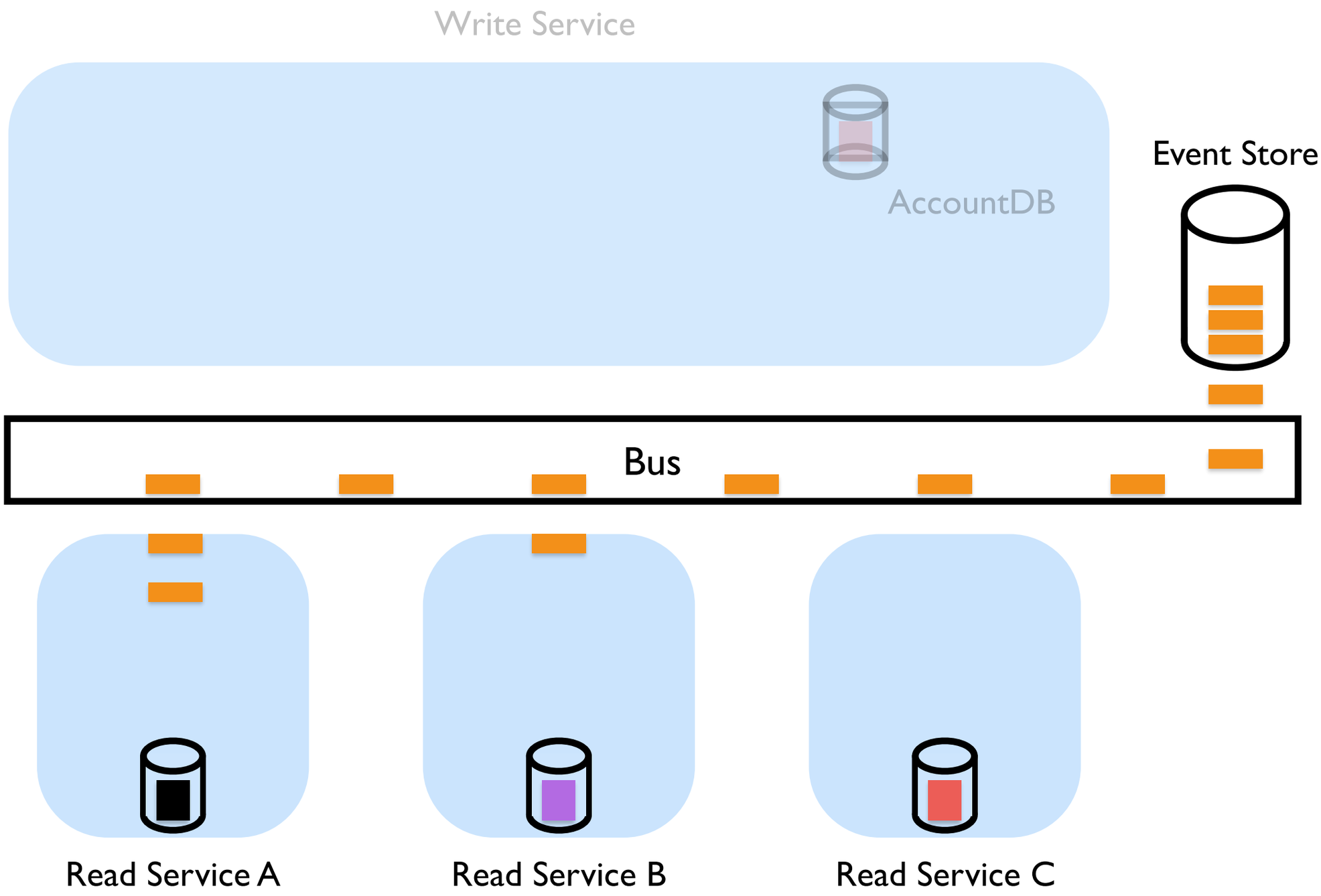
Event Publishing?



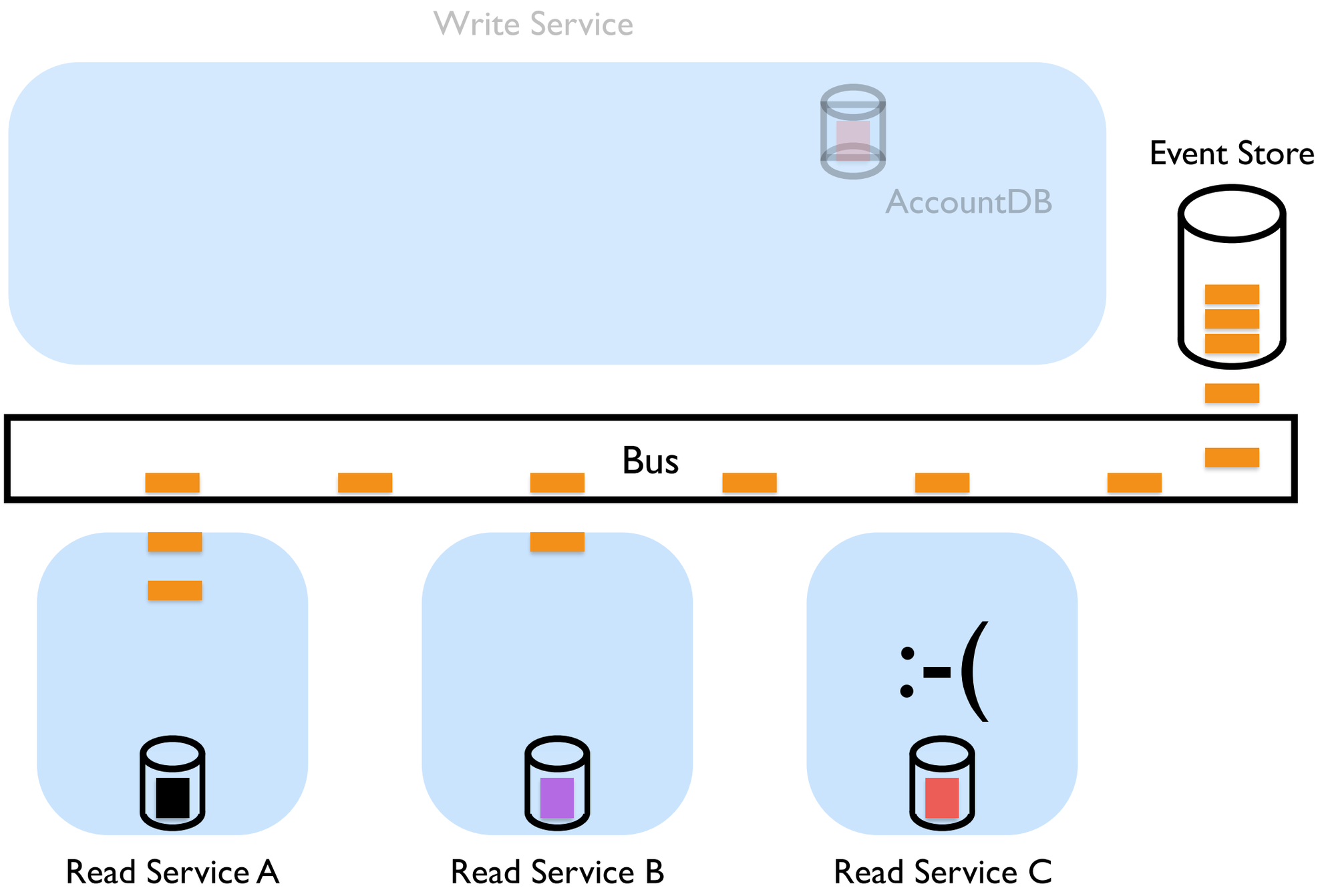
Pub-Sub mit Bus



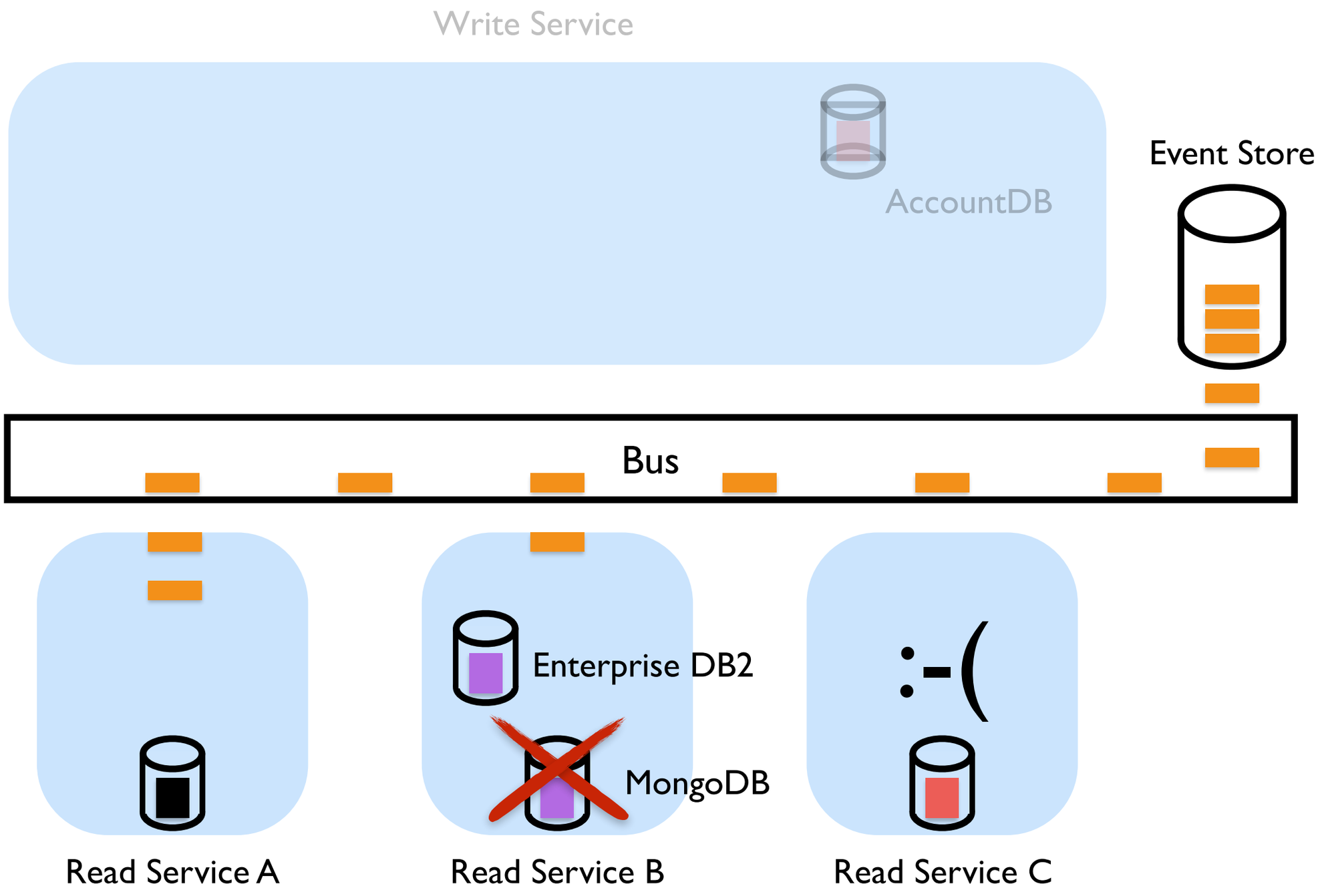
Pub-Sub mit Bus



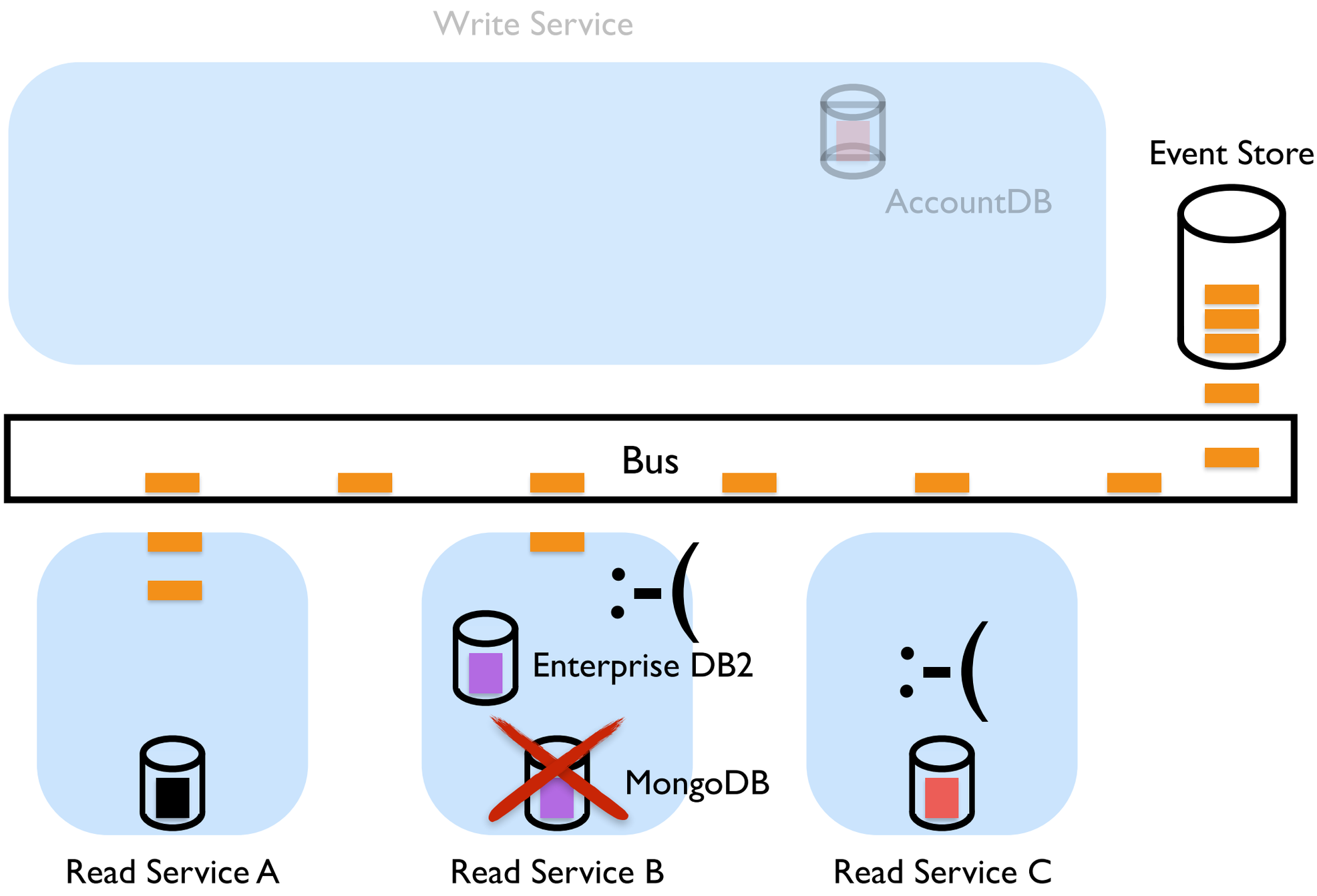
Pub-Sub mit Bus



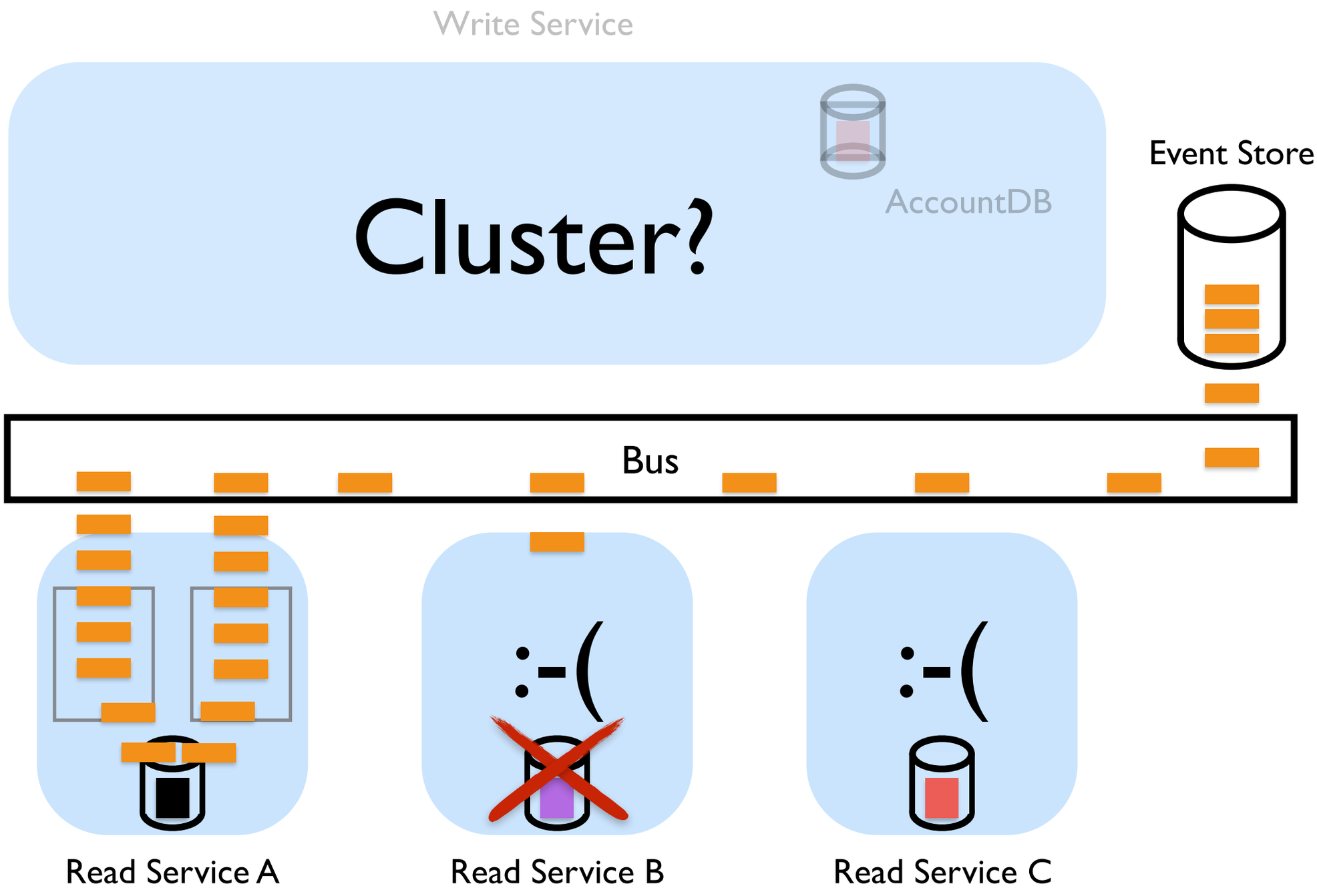
Pub-Sub mit Bus



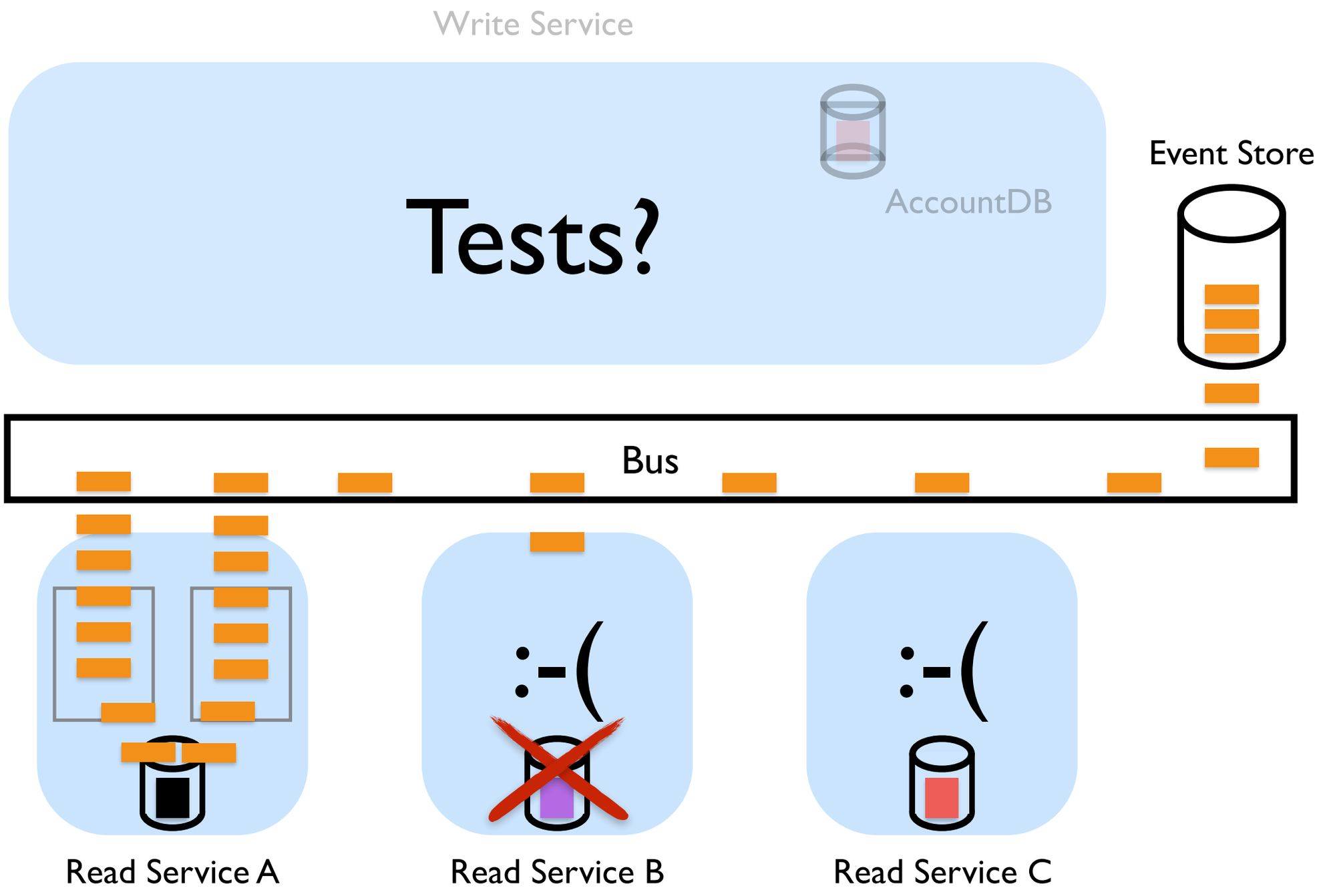
Pub-Sub mit Bus



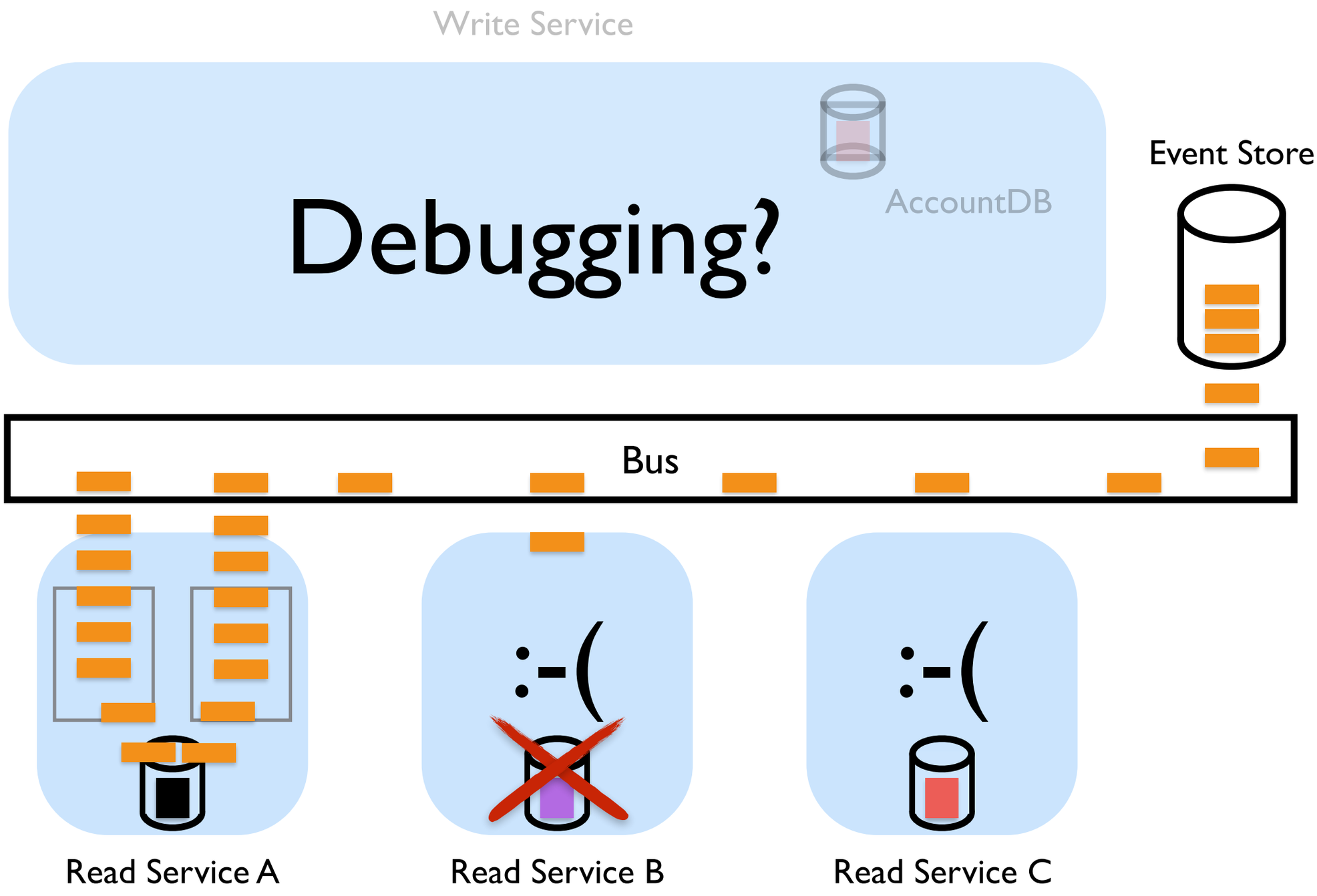
Pub-Sub mit Bus



Pub-Sub mit Bus

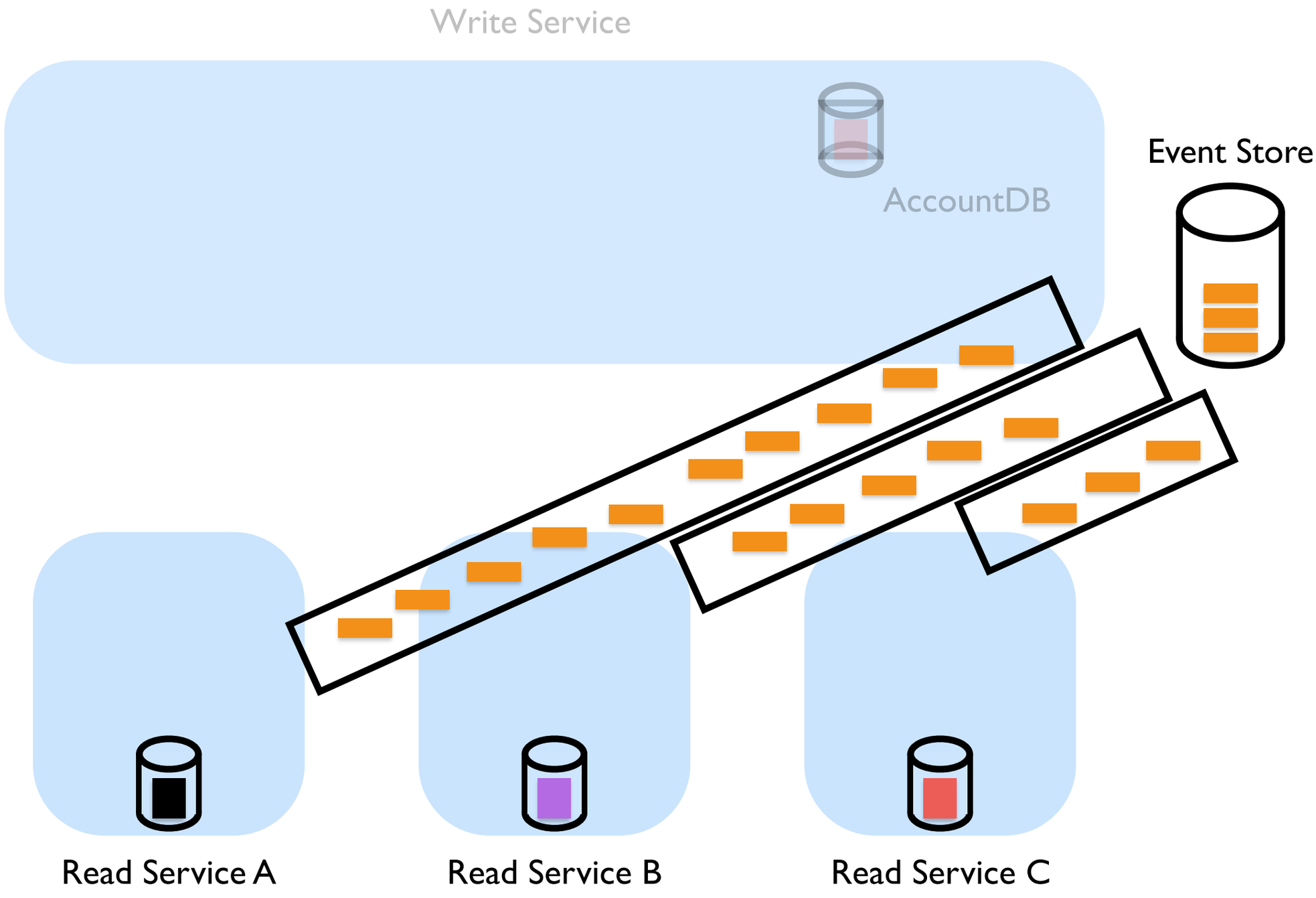


Pub-Sub mit Bus

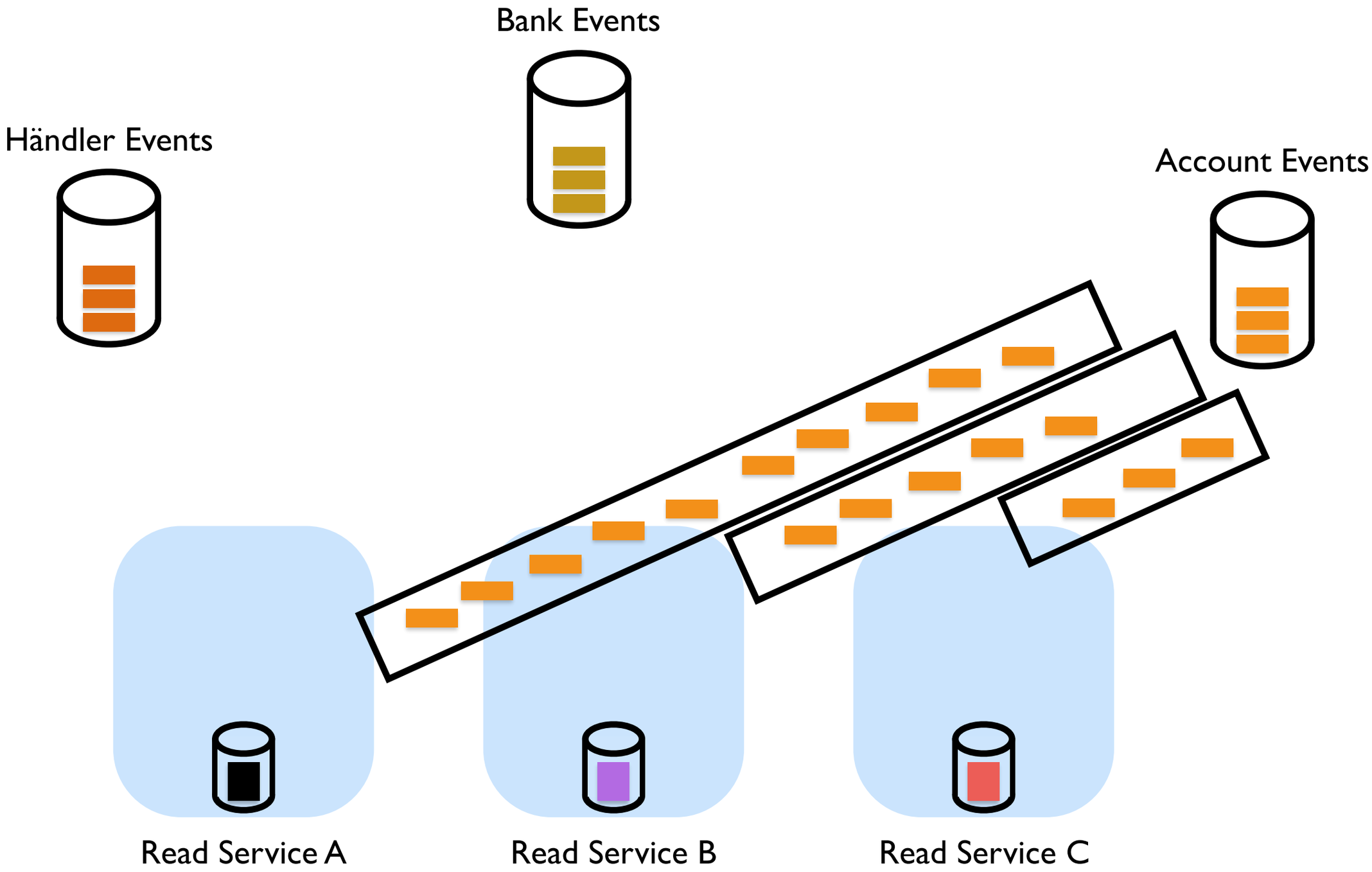


Queues!

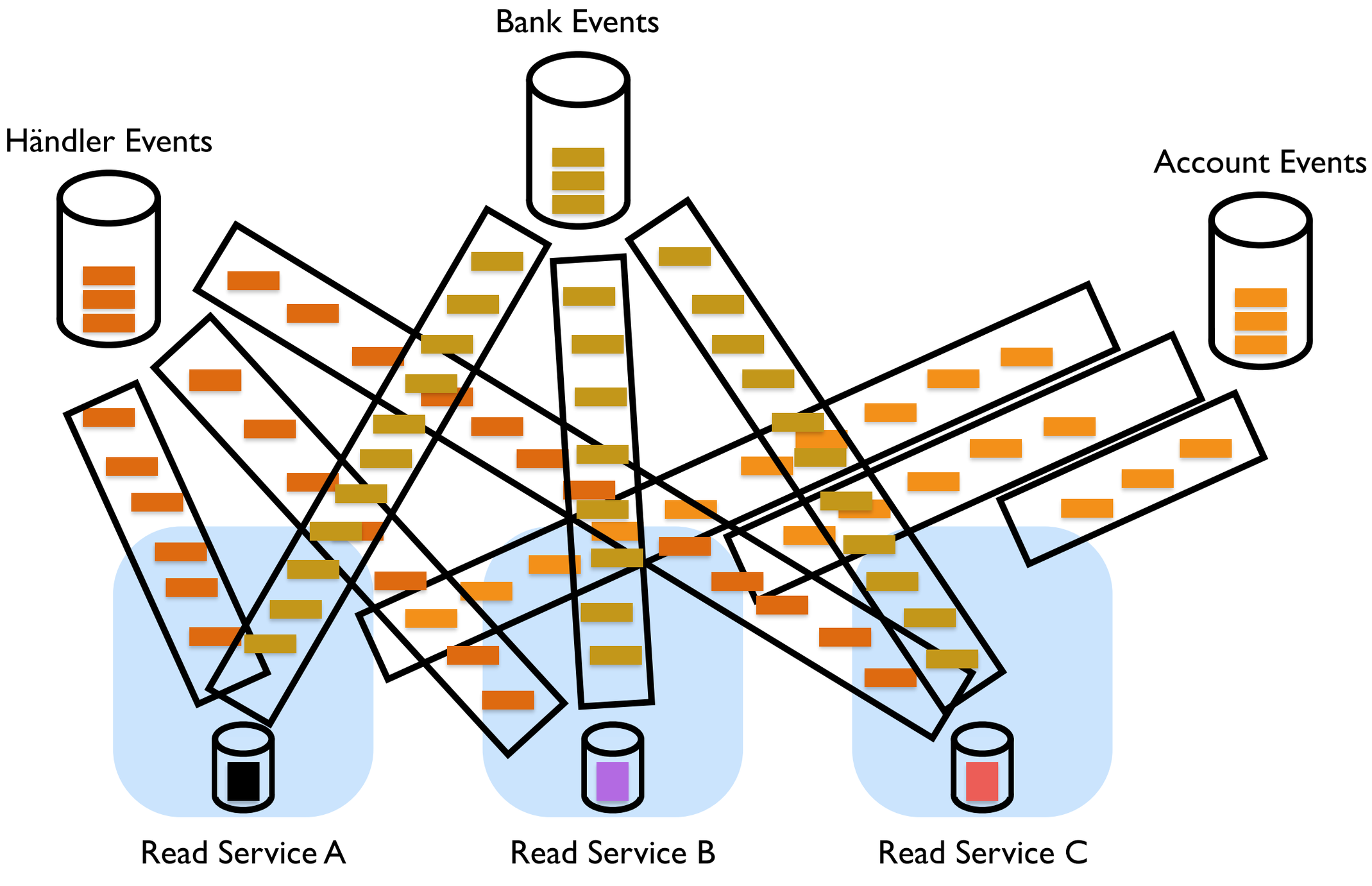
Point2Point mit Queues



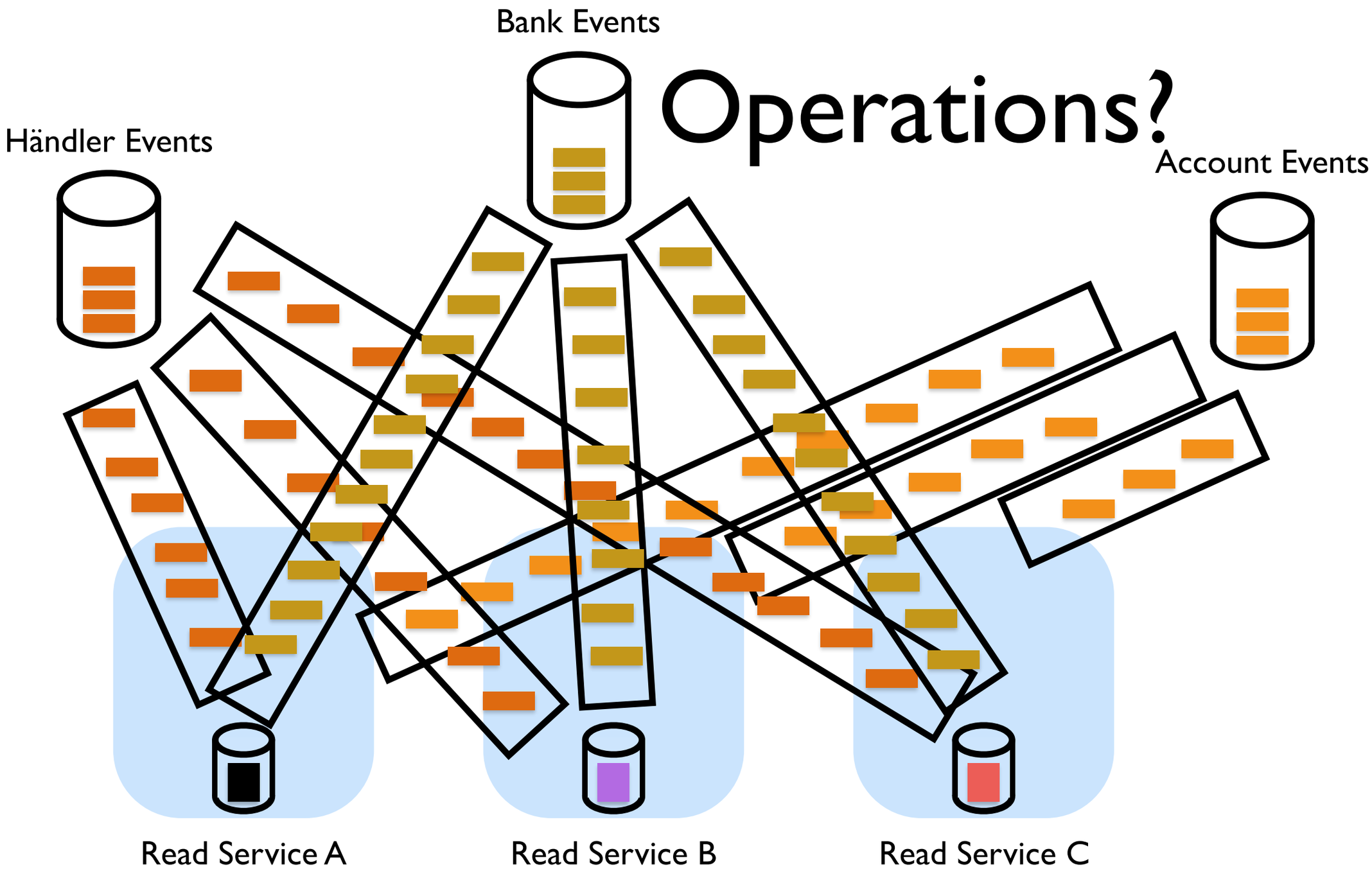
Point2Point mit Queues



:-o

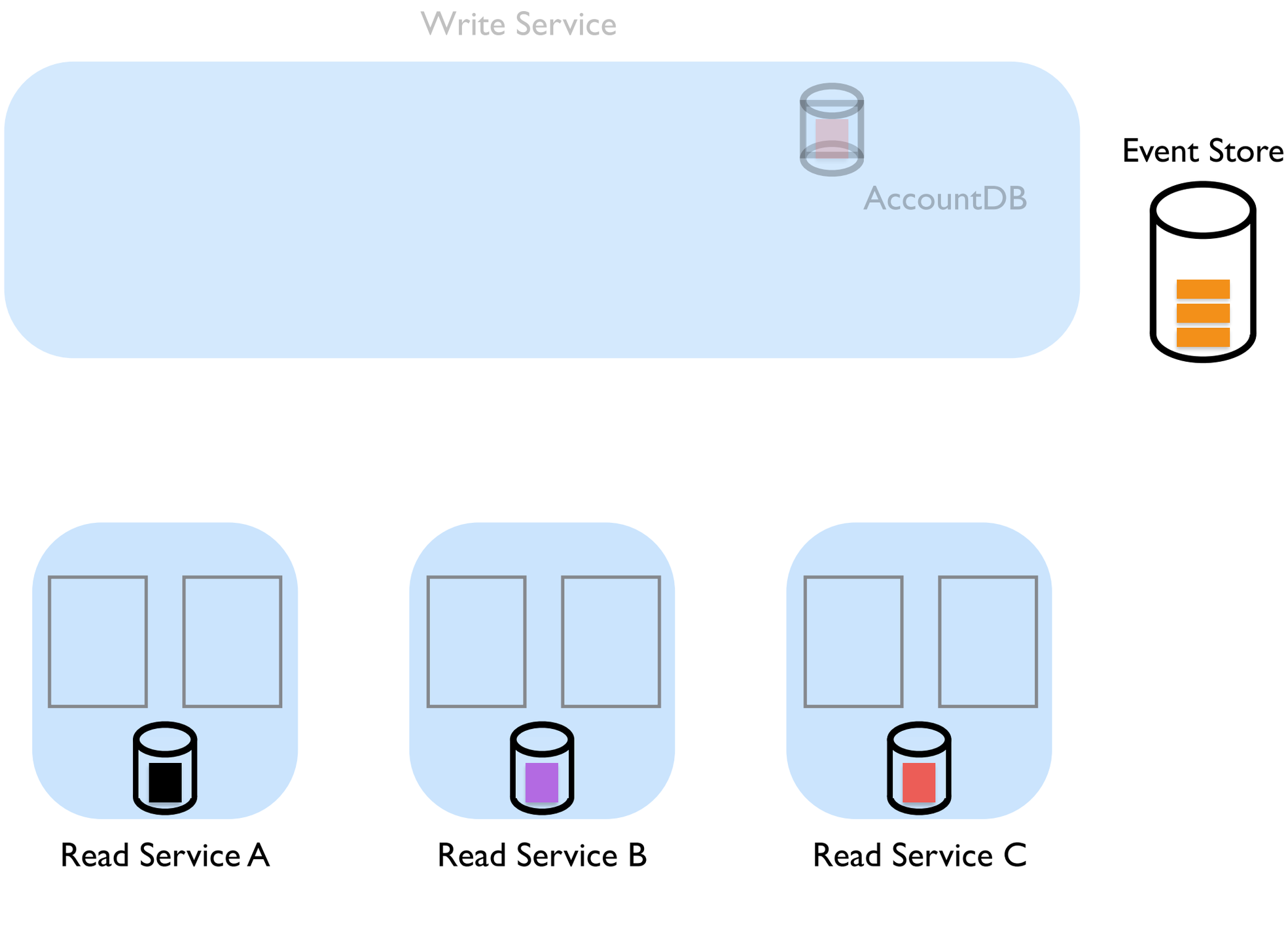

Bank Events
Händler Events
Account Events
Read Service A
Read Service B
Read Service C

:-o

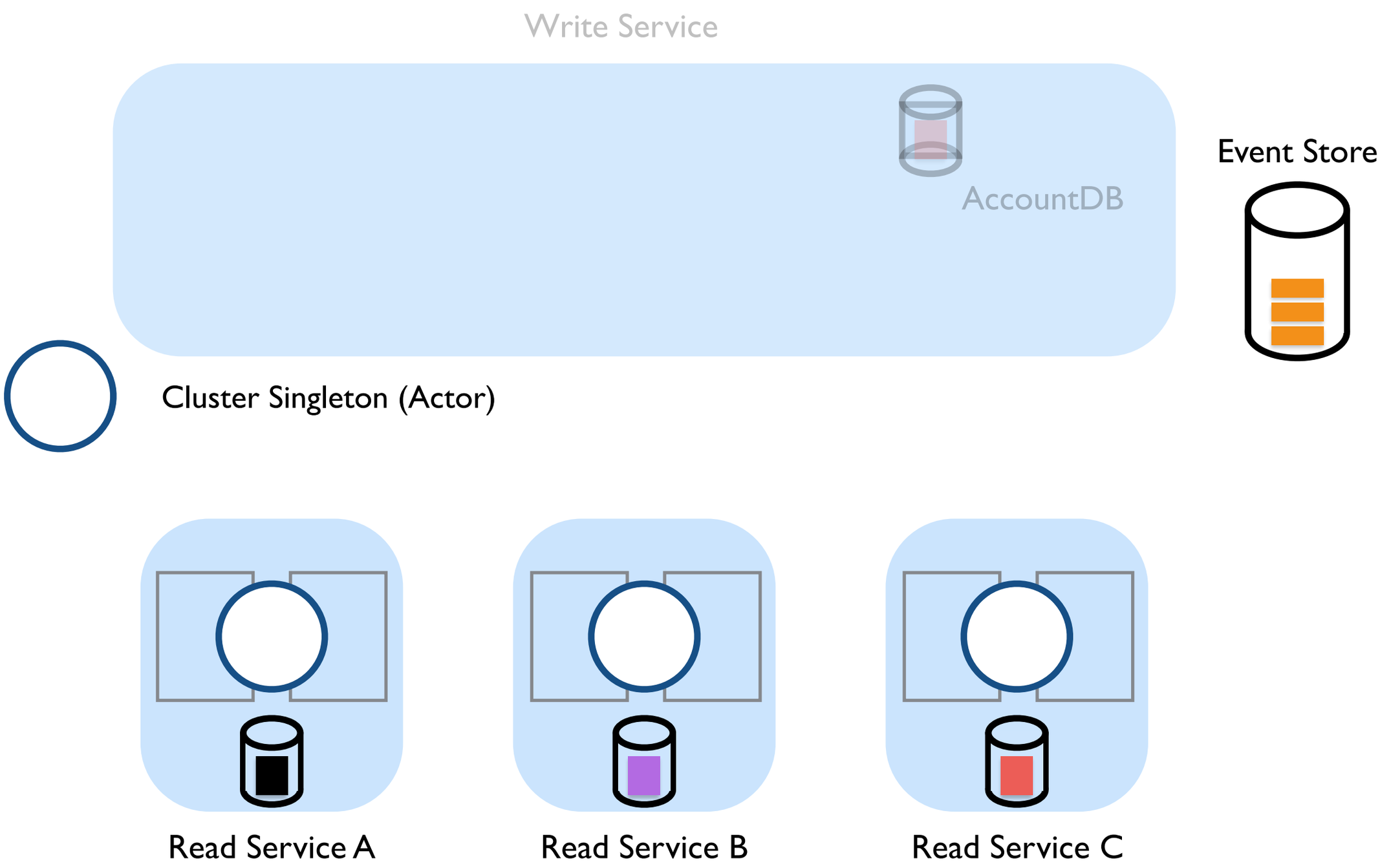

Bank Events
Operations?
Händler Events
Account Events
Read Service A
Read Service B
Read Service C

Poll???

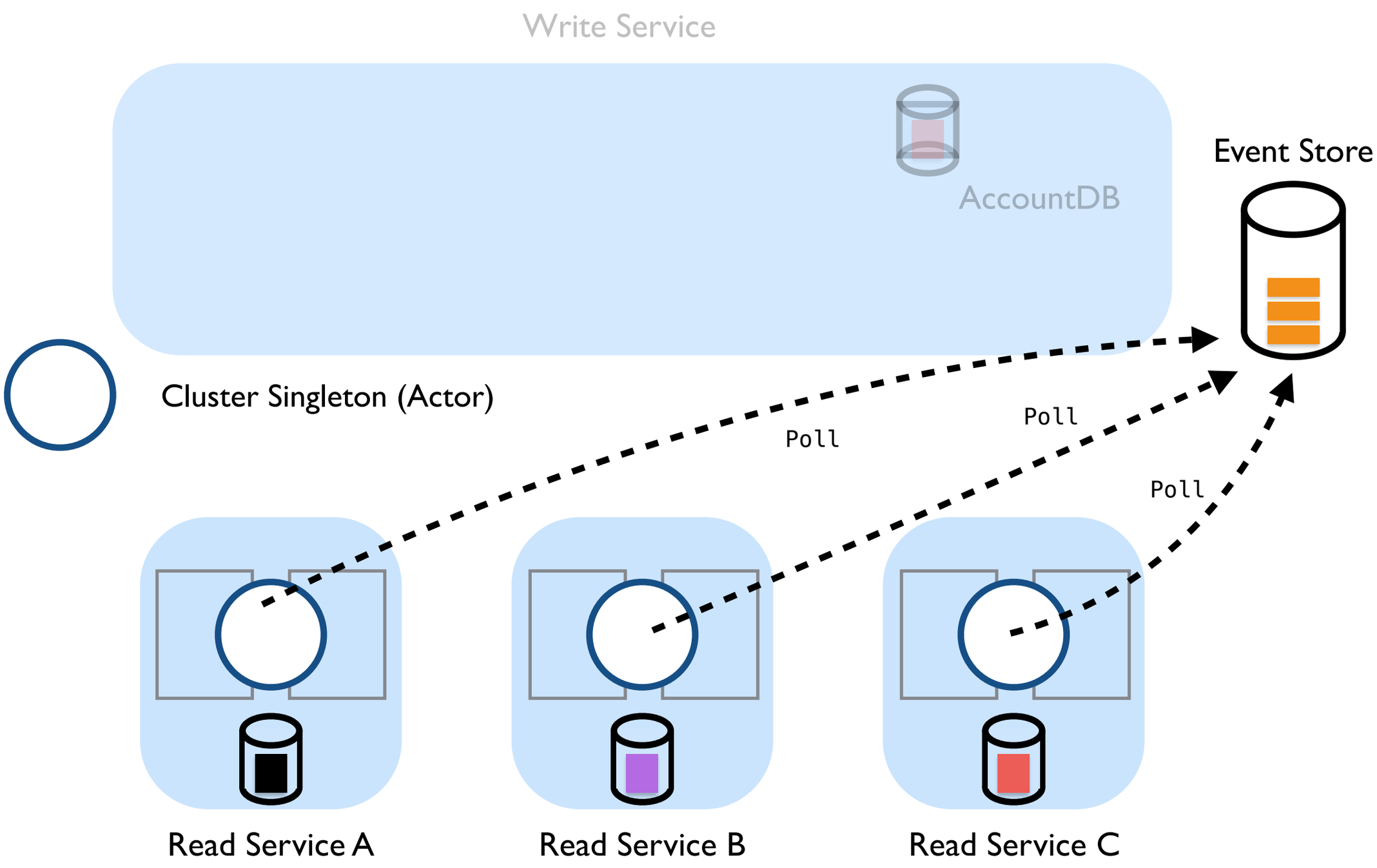
Polling mit Akka

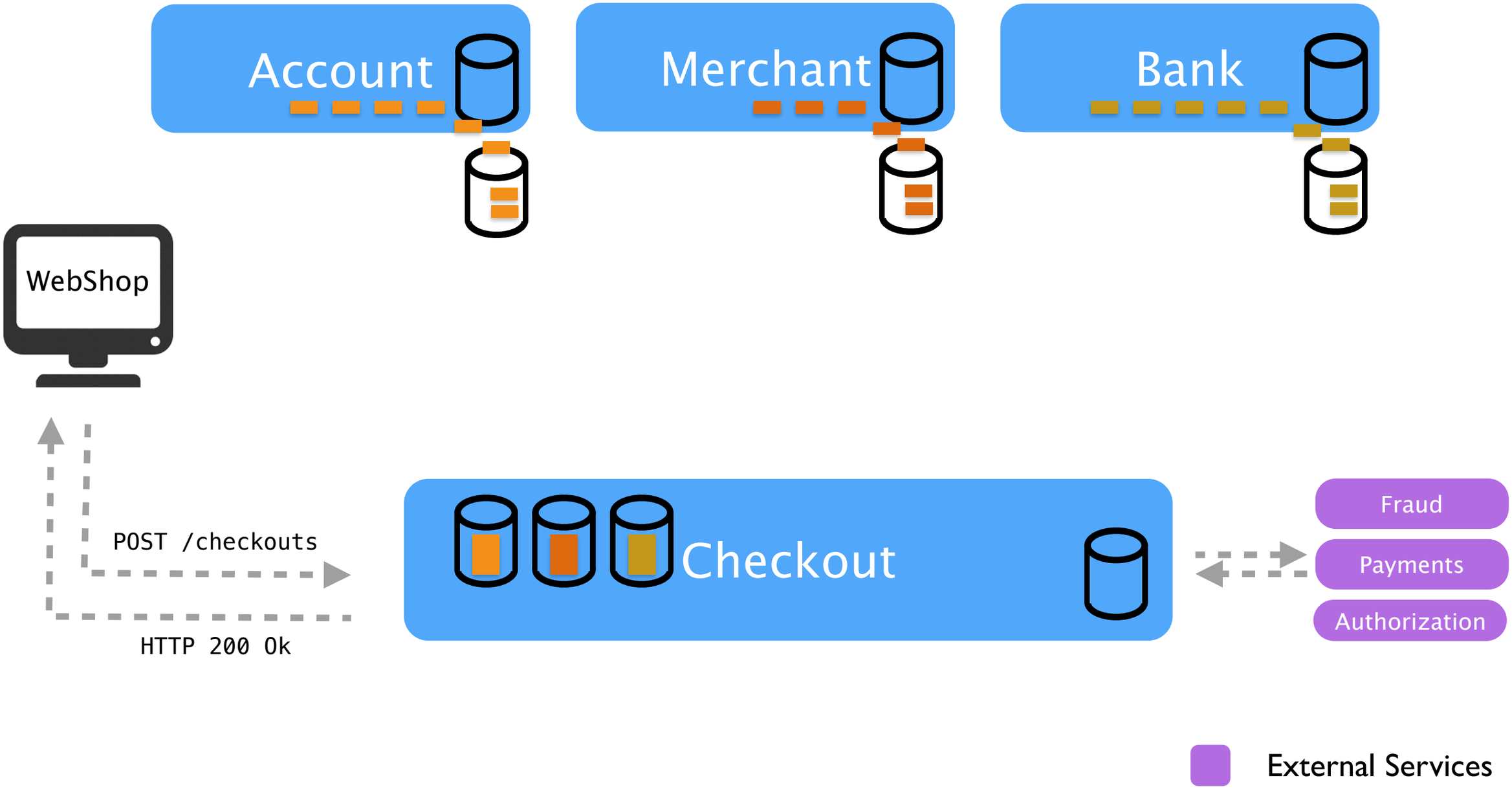


Polling mit Akka

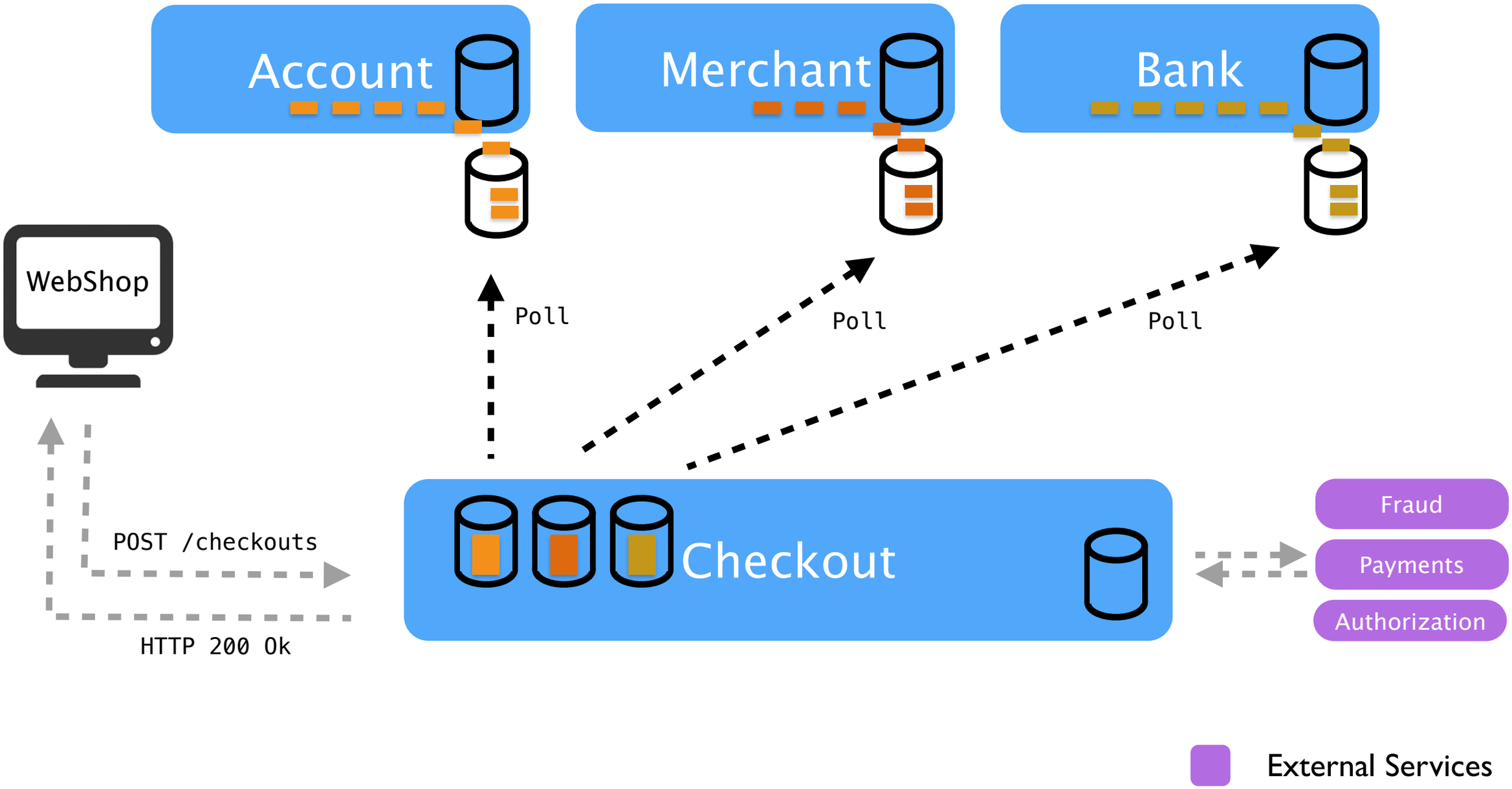


Polling mit Akka

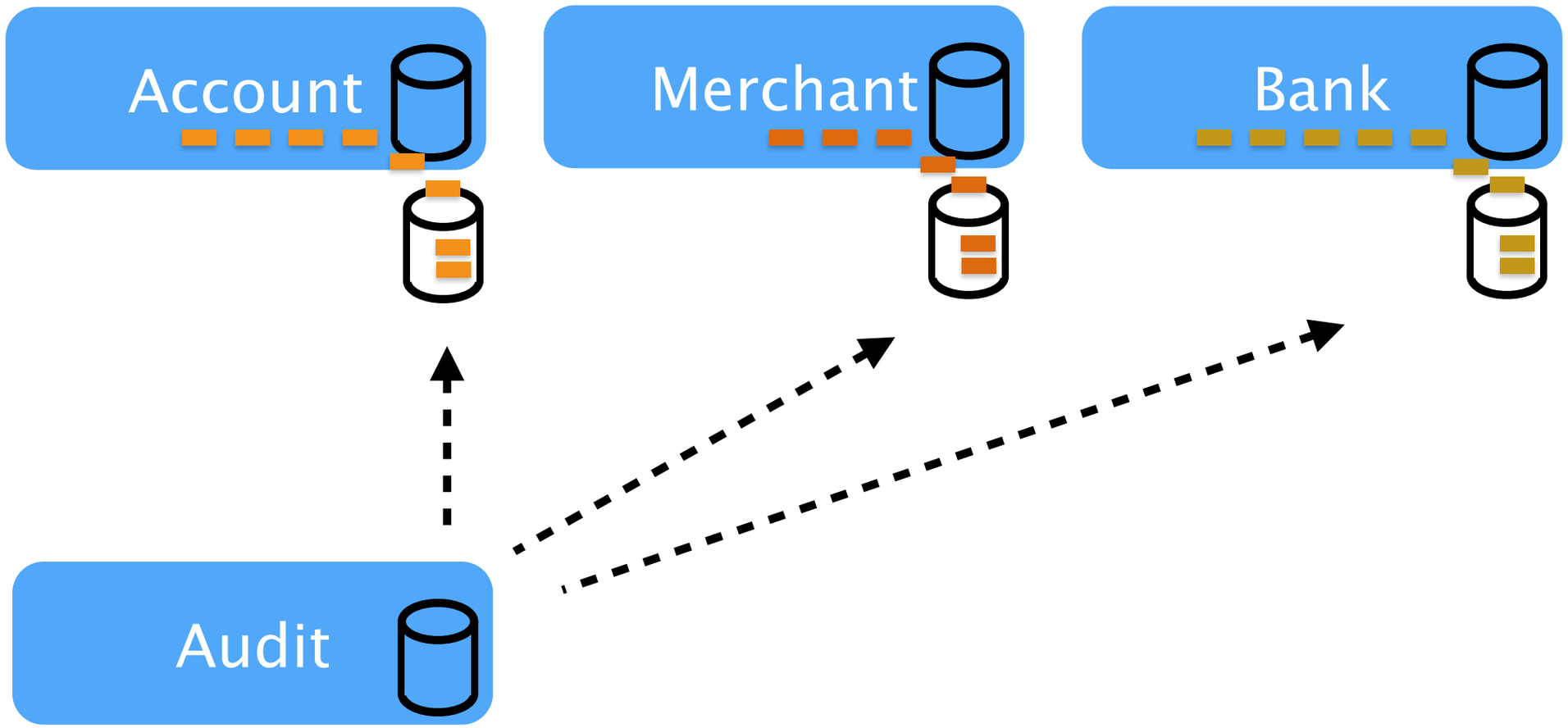



Account
Merchant
Bank
WebShop
POST /checkouts
HTTP 200 Ok
Checkout
Fraud
Payments
Authorization
External Services

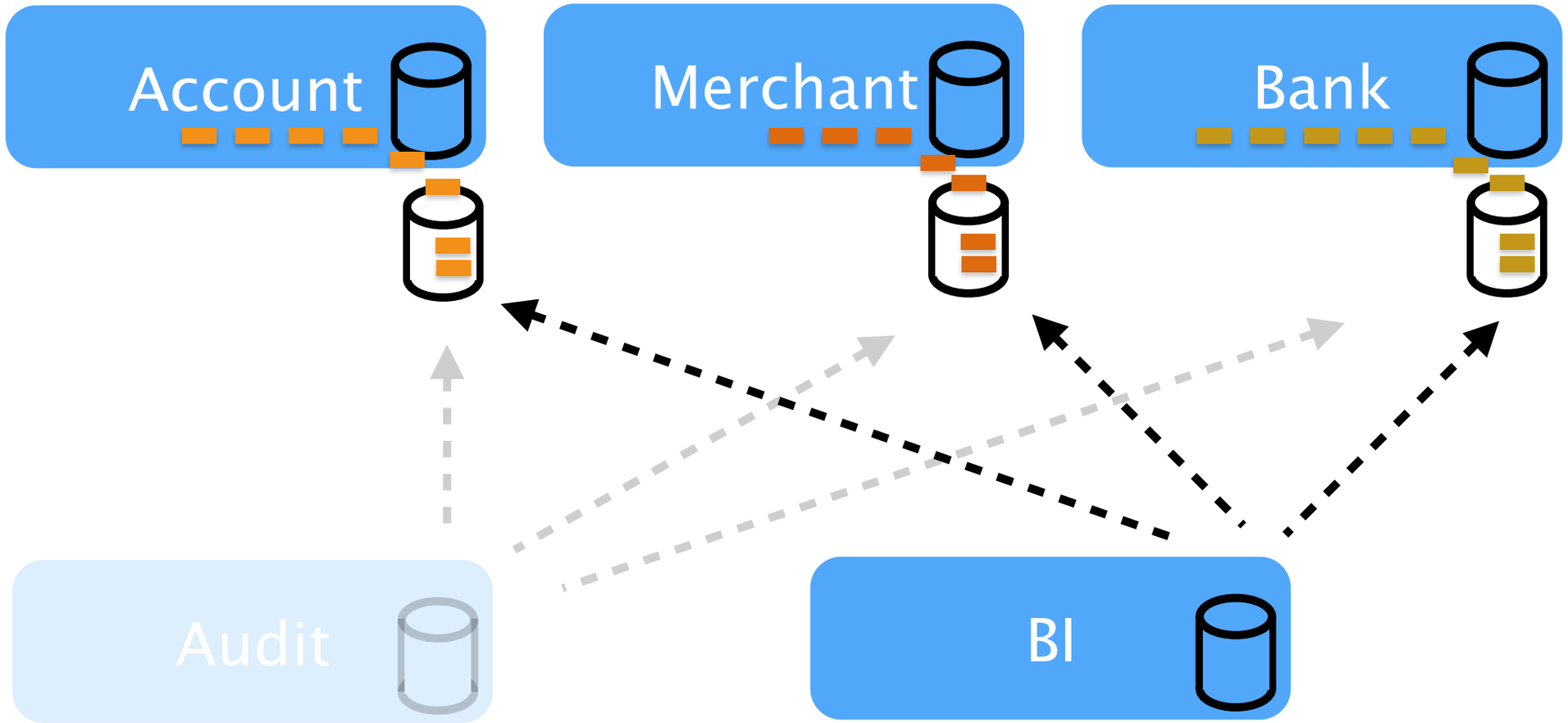
Poll for the win



Auditlog



Business Intelligence



Danke!

Michael Omann
michael.omann@senacor.com

SENACOR

Fragen?

Michael Omann
michael.omann@senacor.com

SENACOR

Danke!

Michael Omann
michael.omann@senacor.com

SENACOR