Technische Hochschule Ingolstadt

Fakultät für Elektrotechnik und Informatik

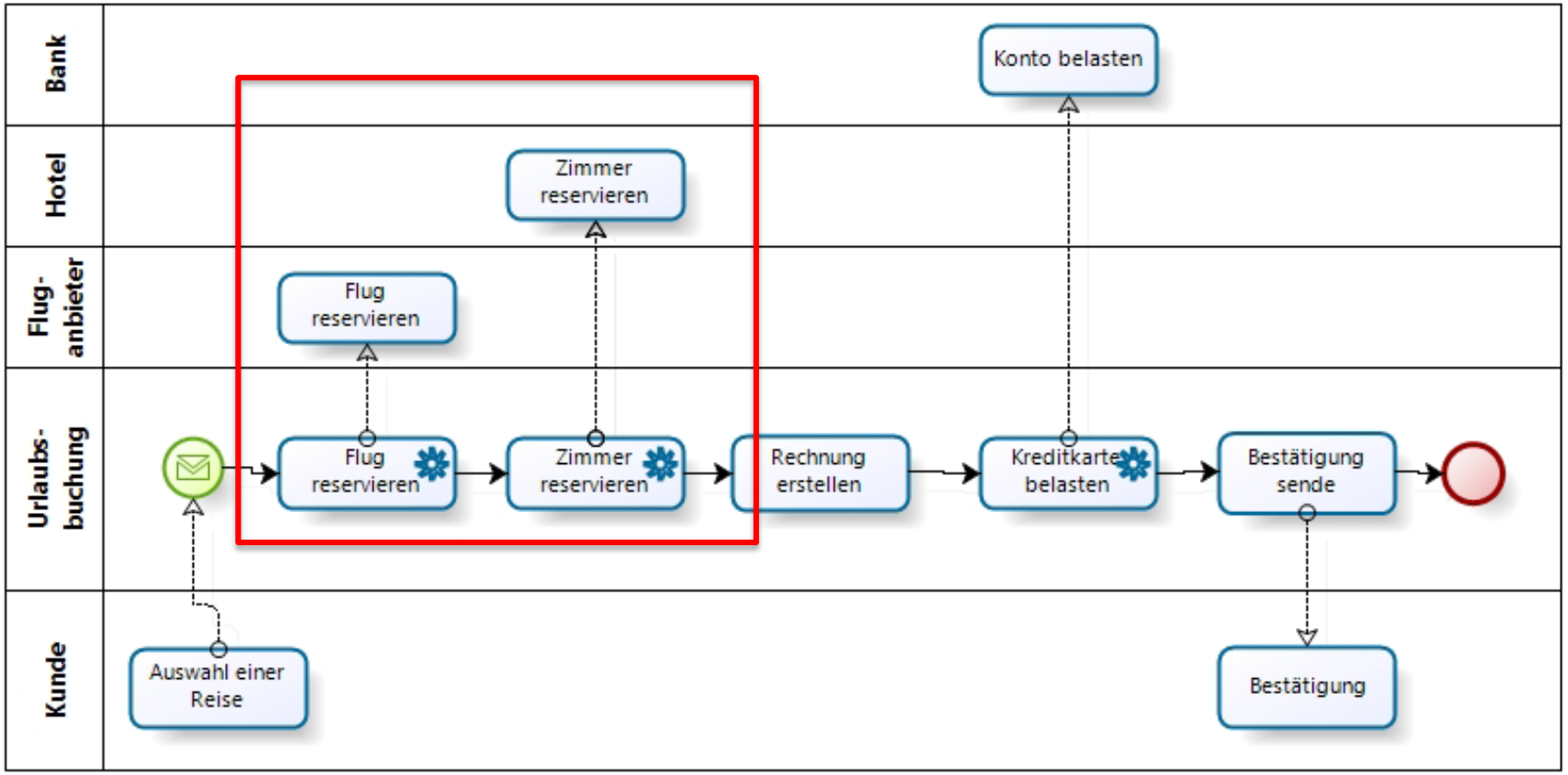
Zukunft in Bewegung

IT-Integrations- und Migrationstechnologien

Transaktionen im Kontext von Microservices

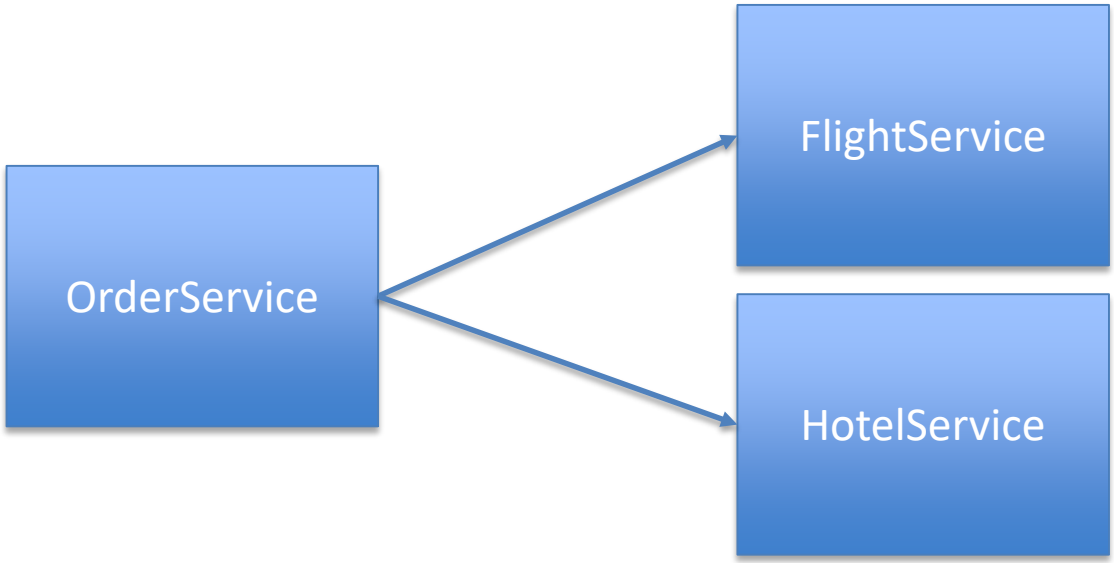
Prof. Dr. Bernd Hafenrichter 26.11.2023

Motivation



- Was passiert wenn die Kreditkarte des Kunden nicht gedeckt ist?
- Was passiert wenn die Zimmerreservierung fehl schlägt?

Motivation



Problem:

- Wie kann eine Transaktion über mehrere Services hinweg implementiert werden?
- Ein 2-Phase-Commit-Protokoll ist aufgrund der schlechten Skalierbarkeit und zentralem Transaktionskoordinator keine option

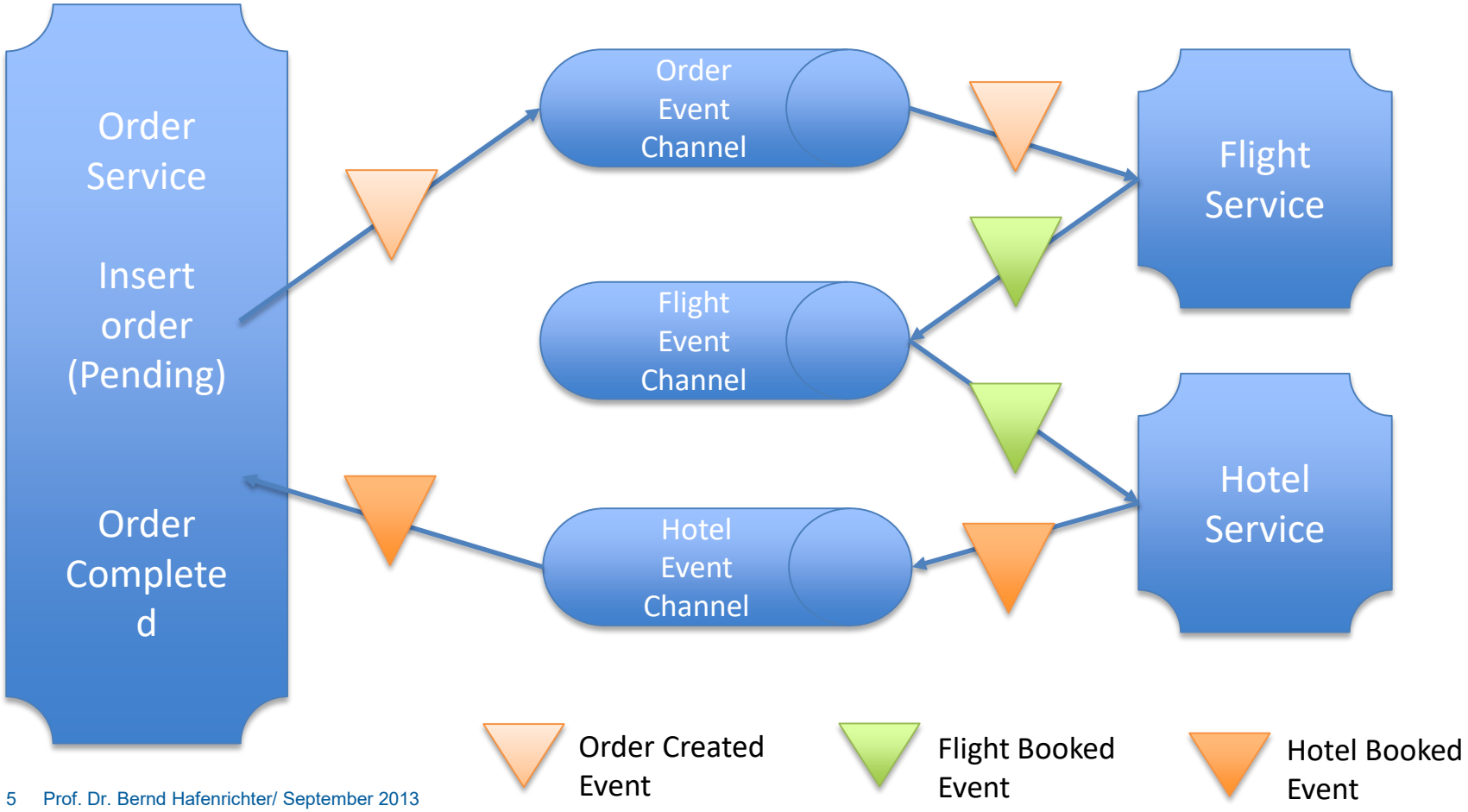
Motivation

Das Saga-Pattern:

- Implementiere eine verteilte Transaktion auf Basis von lokalen Transaktionen
- Eine Saga ist eine Folge von lokalen Transaktionen.
- Jede lokale Transaktion aktualisiert die Datenbank und veröffentlicht eine Nachricht oder ein Ereignis, um die nächste lokale Transaktion in der Saga auszulösen.
- Wenn eine lokale Transaktion fehlschlägt führt die Saga eine Reihe von Ausgleichstransaktionen aus, die die von den vorangegangenen lokalen Transaktionen vorgenommenen Änderungen rückgängig machen.

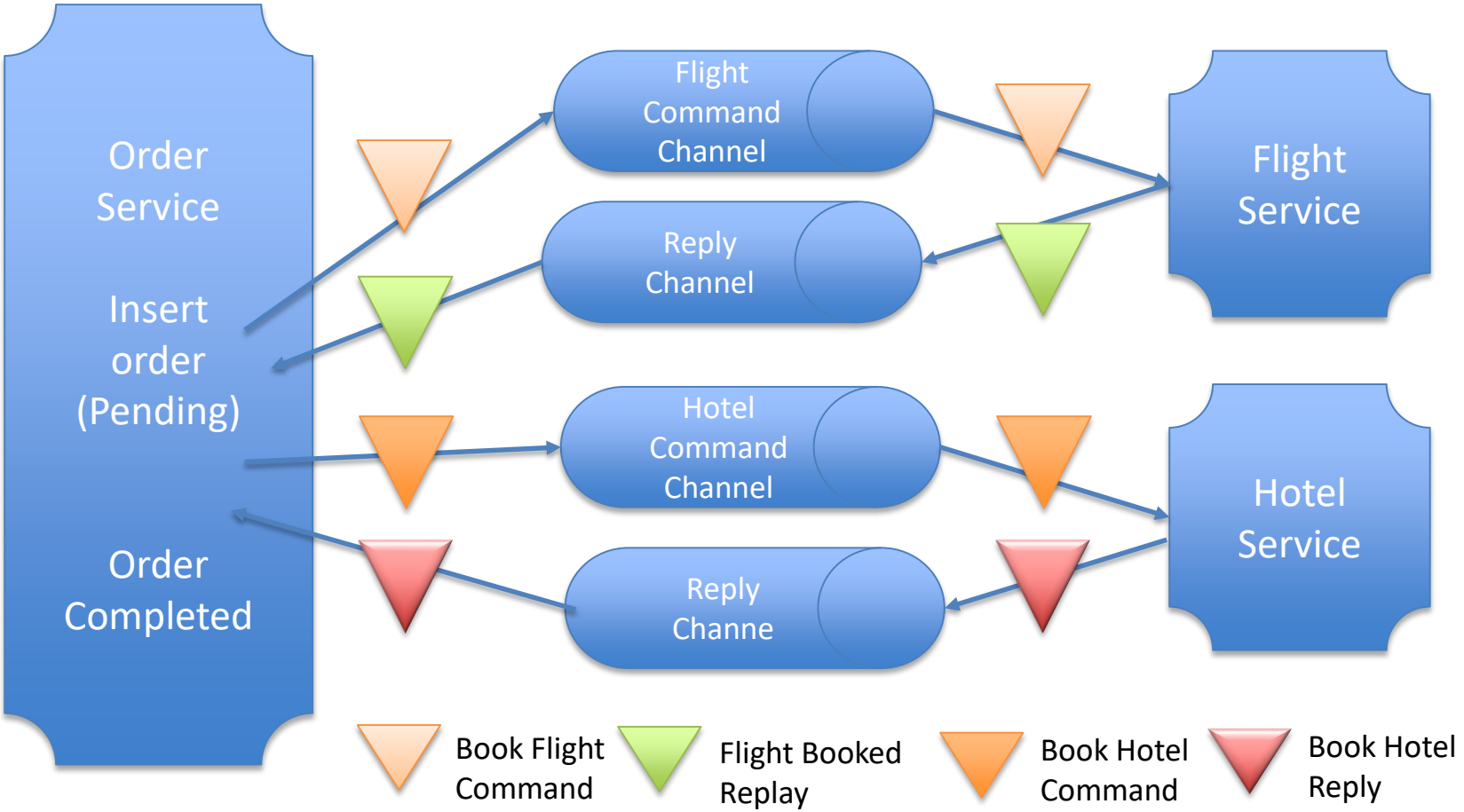
Motivation

Choreografie - jede lokale Transaktion veröffentlicht Domänenereignisse, die lokale Transaktionen in anderen Diensten auslösen



Motivation

Orchestrierung - ein Orchestrator (Objekt) teilt den Teilnehmern mit, welche lokalen Transaktionen ausgeführt werden sollen

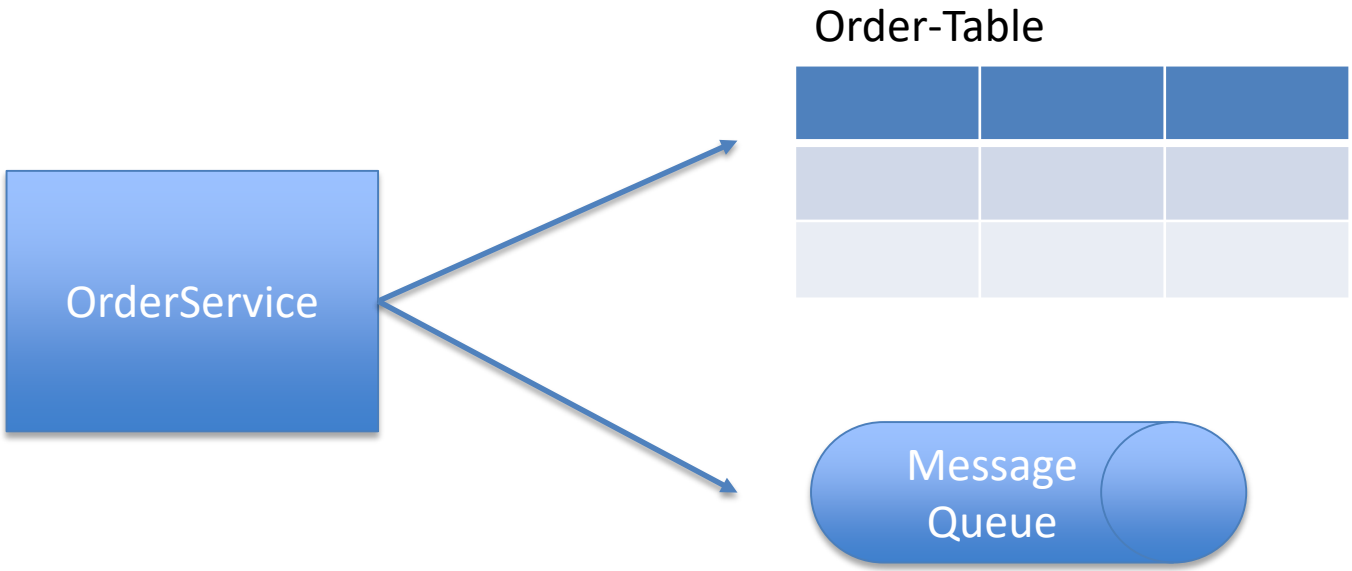


Motivation

Bewertung:

- Das Pattern ermöglicht es einer Anwendung, die Datenkonsistenz über mehrere Dienste hinweg aufrechtzuerhalten, ohne verteilte Transaktionen zu verwenden.
- Das Programmiermodell ist komplexer. So muss ein Entwickler beispielsweise kompensierende Transaktionen entwerfen, die explizit Änderungen rückgängig machen, die zu einem früheren Zeitpunkt in einer Saga vorgenommen wurden.
- Um zuverlässig zu sein, muss ein Dienst seine Datenbank atomar aktualisieren und eine Nachricht/Ereignis veröffentlichen. Er kann nicht den traditionellen Mechanismus einer verteilten Transaktion verwenden, die sich über die Datenbank und den Message Broker erstreckt.
    - Mögliche Lösungen: Event sourcing, Transactional Outbox

Motivation



Problem:

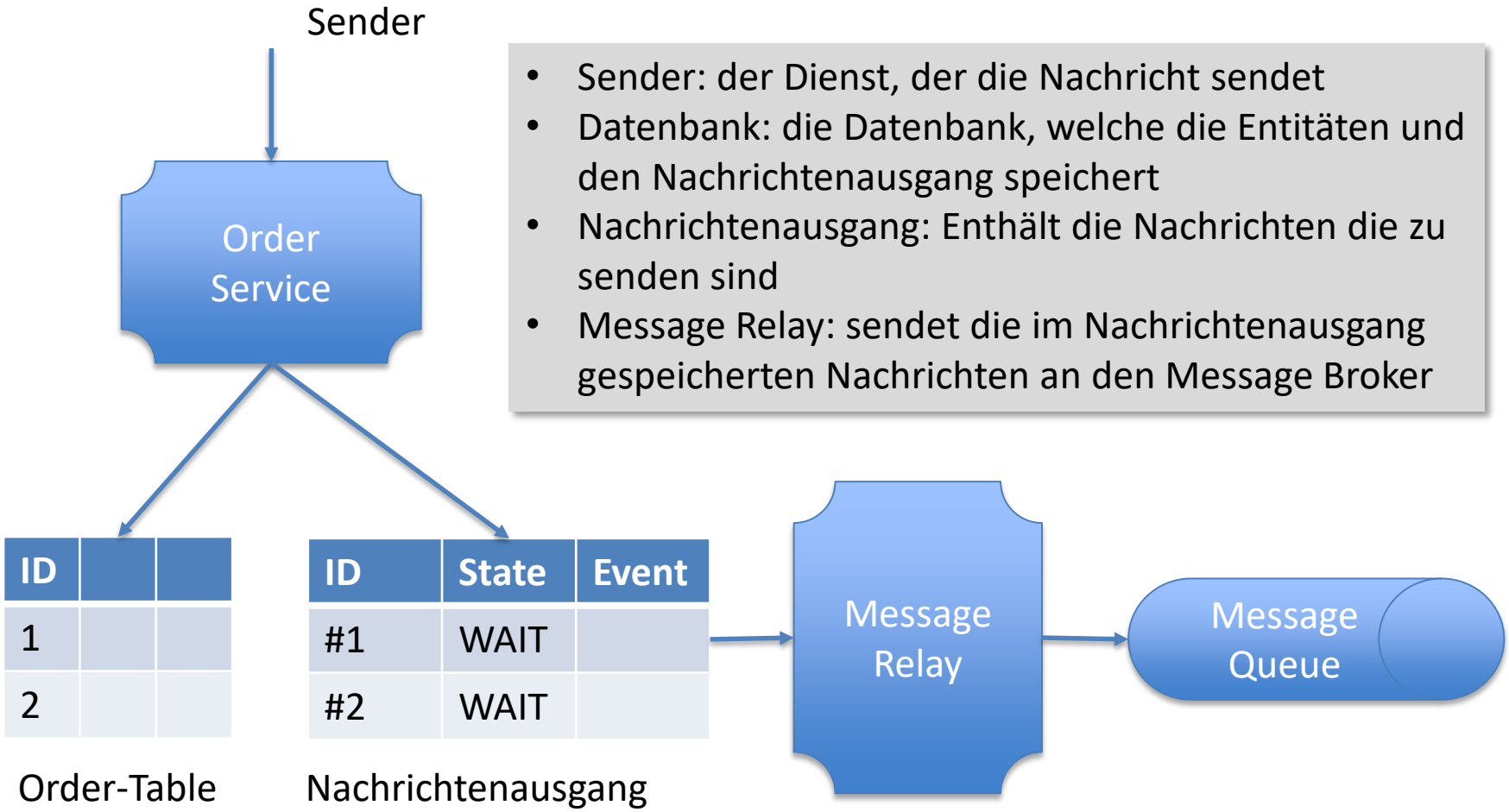
- Wie kann man die Datenbank atomar aktualisieren und Nachrichten an einen Message Broker senden?

Motivation

Das Transactional-Outbox-Pattern:

- Die Lösung besteht darin, dass der Dienst, der die Nachricht sendet, die Nachricht zunächst in der Datenbank als Teil der Transaktion speichert, mit der die Entitäten aktualisiert werden.
- Ein separater Prozess (Message Relay) sendet dann die Nachrichten an den Message Broker.

Motivation



- Sender: der Dienst, der die Nachricht sendet
- Datenbank: die Datenbank, welche die Entitäten und den Nachrichtenausgang speichert
- Nachrichtenausgang: Enthält die Nachrichten die zu senden sind
- Message Relay: sendet die im Nachrichtenausgang gespeicherten Nachrichten an den Message Broker

Motivation

Bewertung:

- 2PC wird nicht verwendet
- Die Nachrichten werden garantiert nur dann gesendet, wenn die Datenbanktransaktion bestätigt wird.
- Die Nachrichten werden in der Reihenfolge an den Message Broker gesendet, in der sie von der Anwendung gesendet wurden.

Probleme

- Das Message Relay kann eine Nachricht mehr als einmal veröffentlichen. Es kann z. B. abstürzen, nachdem es eine Nachricht veröffentlicht hat, aber bevor es die Tatsache, dass es dies getan hat, aufzeichnet. Beim Neustart wird die Nachricht dann erneut veröffentlicht. (Idempotenz ist wichtig)