이사	대표
	

Eventuate Platform

난 비즈니스 로직에 집중, 분산 데이터 관리는 맡긴다.

소속
Git 활용 및 표준프레임워크 MSA

발표자
Sooamazing

발표일시
2025년 05월 17일

OSSCA

목차

미리 보기

Refactoring to Microservices

Transaction Messaging

DDD

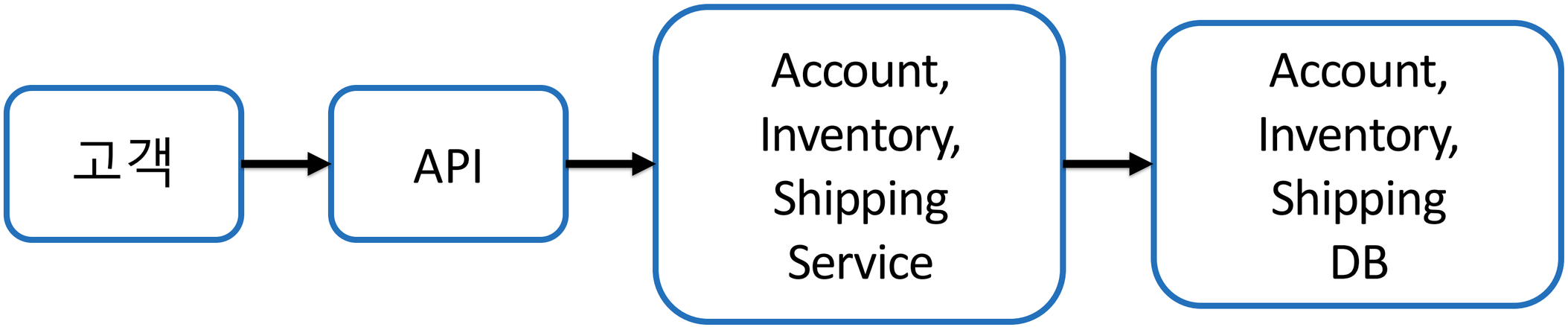
CQRS

SAGA

Loose Coupling

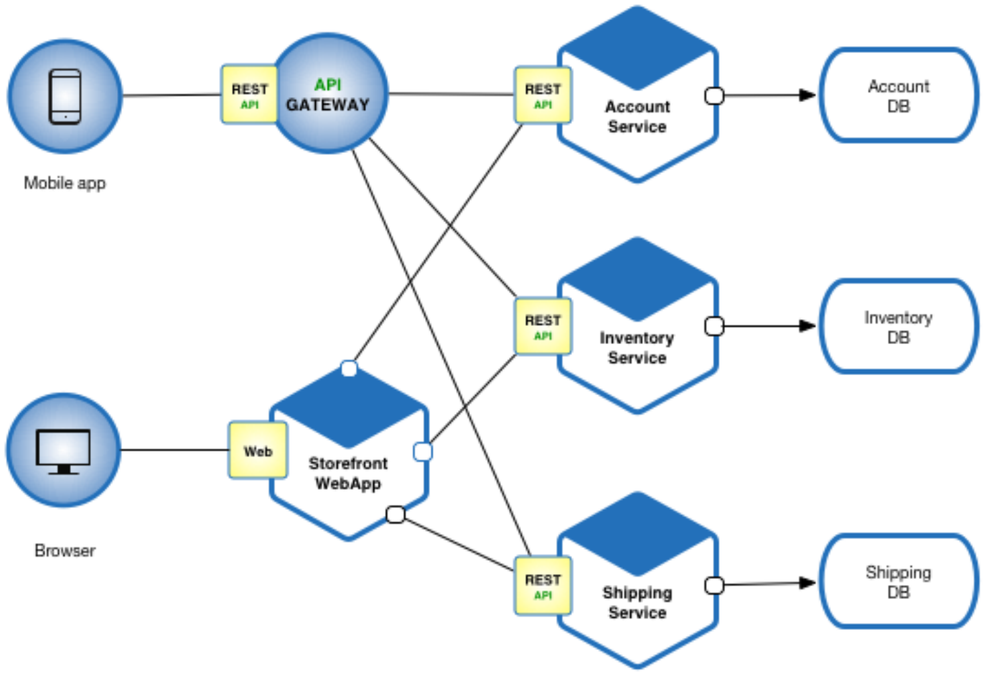
Event Sourcing

현재



전자상거래 애플리케이션 아키텍처 예시

도입



전자상거래 애플리케이션의 마이크로 서비스 아키텍처 예시

01. Loose coupling – 결합을 느슨하게

- Design-time coupling: 한 서비스에 변경이 필요할 때, 다른 서비스를 변경하는 것. → 느슨한 결합으로 영향 줄이기
- Runtime coupling: 서비스 하나의 장애가 다른 서비스의 요청 응답에도 영향을 미치는 것. → 느슨한 결합으로 영향 줄이기

02. 분산 데이터 관리

- 각 서비스에 가장 적합한 유형의 데이터베이스 선택 가능
- 여러 서비스에 걸친 요청에서 데이터 일관성 유지
  - 트랜잭션 구현 – Saga 패턴
  - 쿼리 구현 – API composition, CQRS

출처: Eventuate explained using microservices patterns / 마이크로서비스 아키텍처 패턴 / Pattern: Database per service

소개

Eventuate Platform
개발자가 비즈니스 로직에 집중할 수 있도록
분산 데이터 관리 문제를 해결한다.

마이크로 서비스 개발을 위한 오픈 소스 플랫폼

데이터 일관성 유지
Saga 패턴 사용

CQRS 쿼리 구현

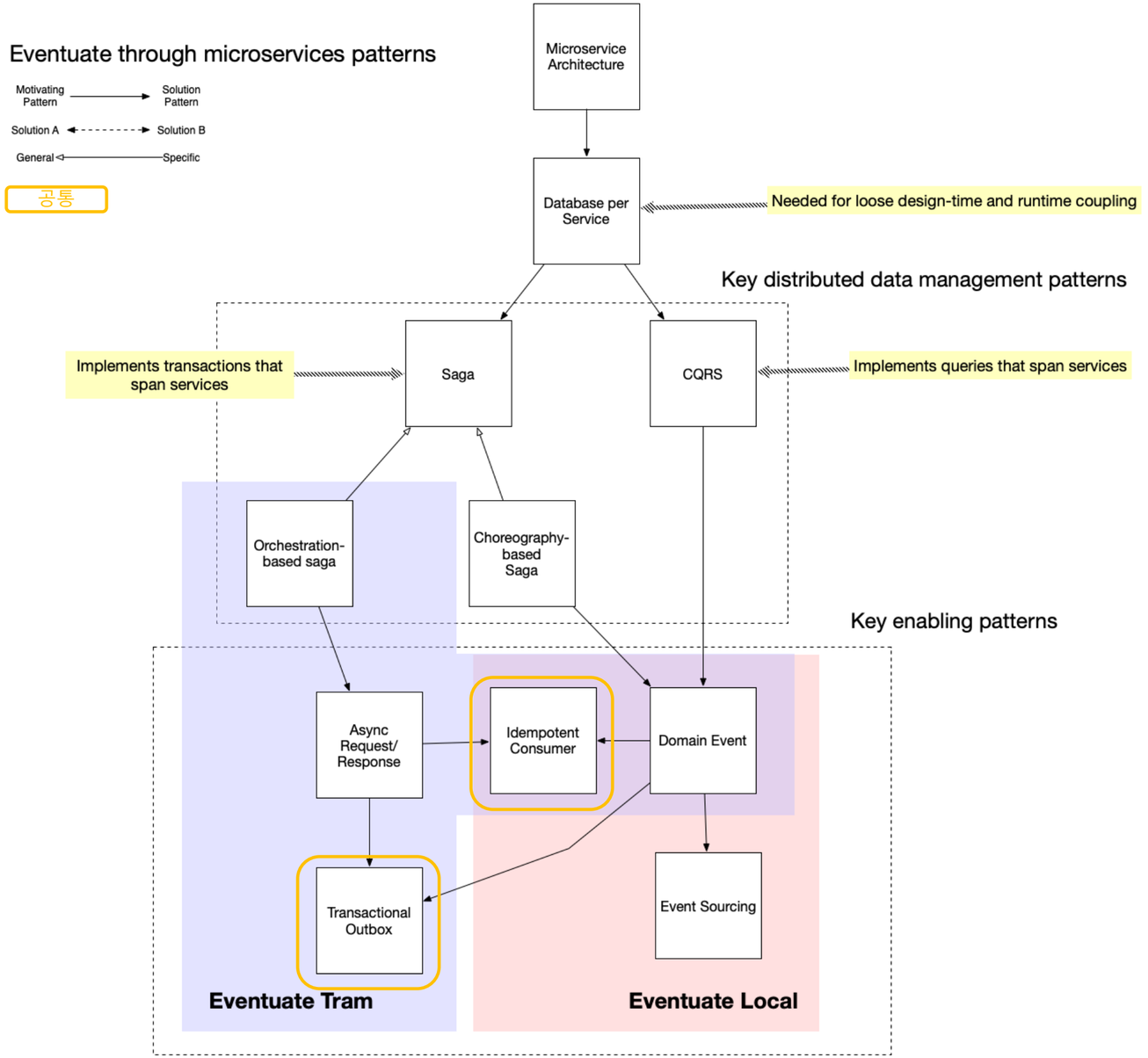
검증된 기술 활용
MySQL, Apache Kafka

Transaction Messaging
Transaction Outbox / Event Sourcing 패턴 사용

마이크로 서비스를 위한 설계

출처: eventuate

구조



출처: [Eventuate explained using microservices patterns]

비교

Eventuate Tram

- 기존 방식(예: JPA/JDBC 및 Entity Framework)의 지속성을 사용하는 서비스를 위한 프레임워크.
- 비스니스 로직을 다시 작성하지 않고도 Spring Boot, Mincronaut, Quark 및 .NET 마이크로 서비스에 쉽게 추가할 수 있고, 트랜잭션 메시징을 제공함.
- 일반적인 지속성(regular persistence)을 사용하고 이벤트를 명시적으로 게시할 수 있음.

VS

Eventuate Local

- 이벤트 소싱 프레임워크.
- 이벤트 소싱은 이벤트 중심 비즈니스 로직 및 지속성 프로그래밍 모델로 데이터 변경 시 이벤트 자동 발행, 모든 업데이트에 대한 안정적인 감사, 시간 쿼리 기본 지원 등의 장점을 제공함.
- 익숙한 데이터 베이스를 계속 사용할 수 있음.
- Java, Scale, Spring 프레임워크 등 다양한 언어와 프레임워크를 위한 이벤트 저장소와 클라이언트 라이브러리로 구성됨.

출처: eventuate / About Eventuate

Saga

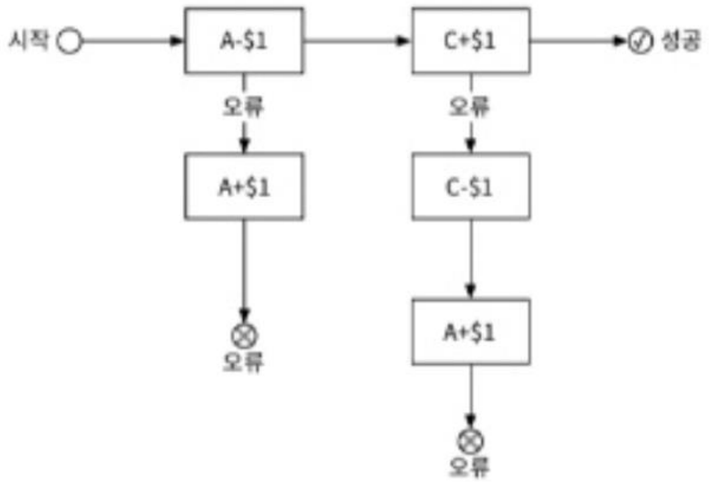


그림 12.13 사가 작업 흐름

A 계좌에서 C 계좌로 $1 이체하는 상황

01. 개념

여러 서비스에 걸친 트랜잭션을 구현합니다. Saga는 해당하는 서비스가 메시지를 교환하는 방식으로 협력하는 일련의 로컬 트랜잭션으로 결과적 일관성에 의존합니다.

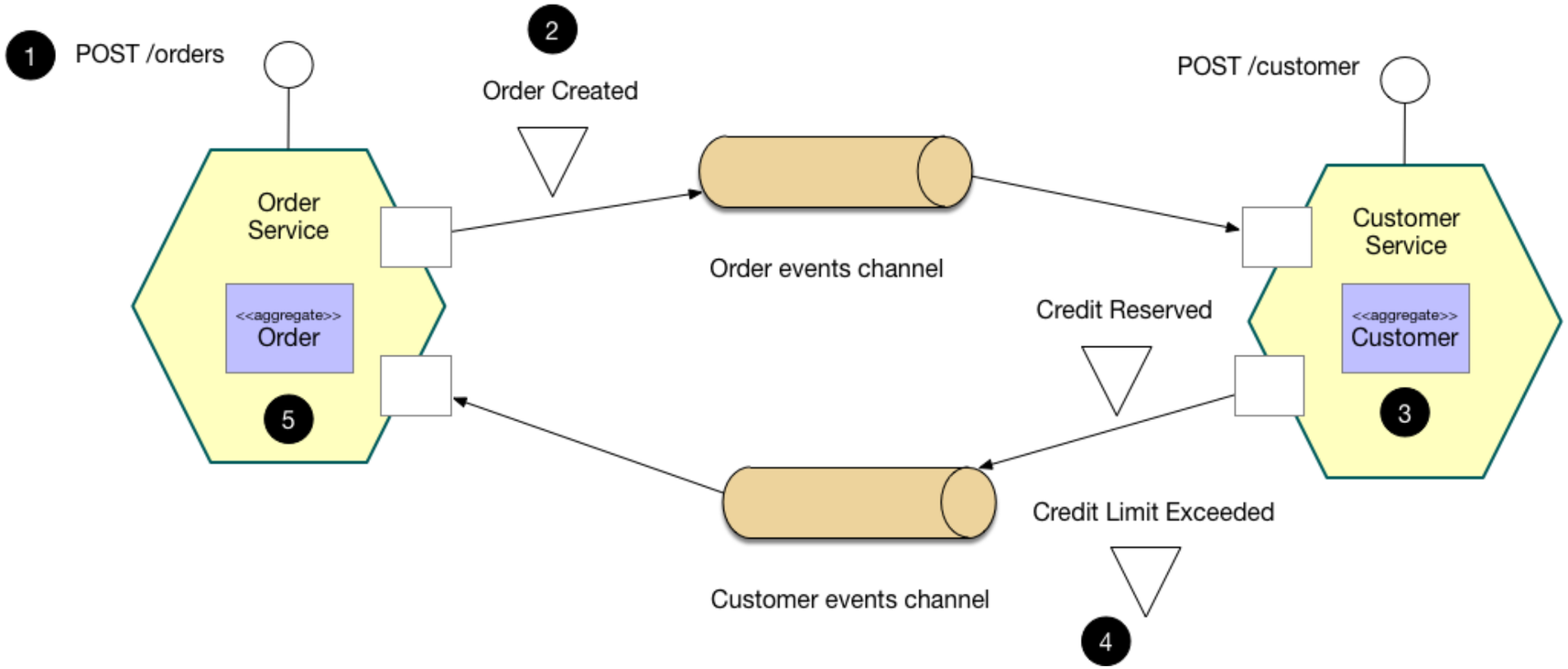
02. 동작

1. 모든 연산은 순서대로 정렬된다. 각 연산은 자기 데이터베이스에 독립 트랜잭션으로 실행된다.
2. 연산은 첫번째부터 마지막까지 순서대로 실행된다. 한 연산이 완료되면 다음 연산이 개시된다.
3. 연산이 실패하면 전체 프로세스는 실패한 연산부터 맨 처음 연산까지 역순으로 보상 트랜잭션을 통해 롤백된다. 따라서 n개의 연산을 실행하는 분산 트랜잭션은 보상 트랜잭션을 위한 n개 연산까지 총 2n개의 연산을 준비해야 한다.

출처: Pattern: Saga / 가상 면접 사례로 배우는 대규모 시스템 설계 기초 2 | 알렉스 쉬 p. 418~419

Saga

03-1. Choreography-based saga (분산 조율, 안무)

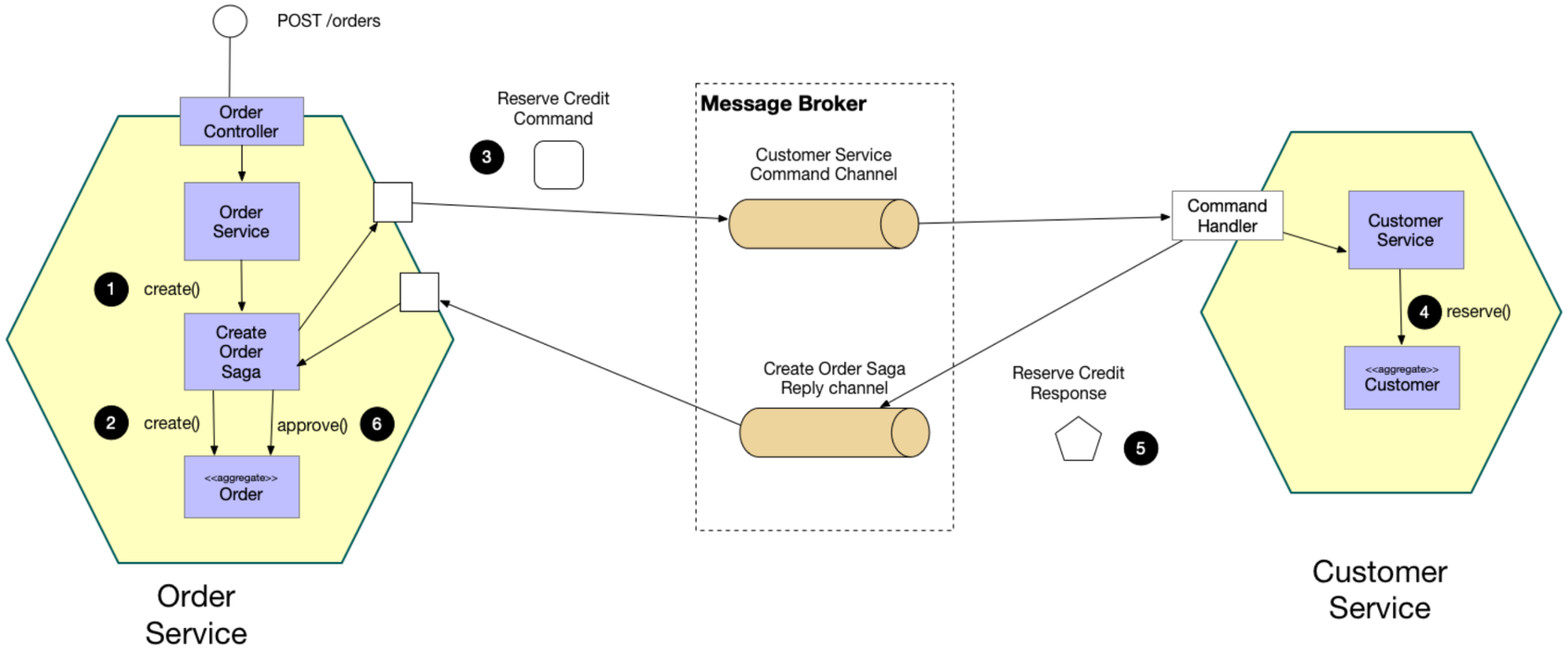


각 도메인 이벤트 구독 / 탈중앙화된 조율 방식

출처:Pattern: Saga / 가상 면접 사례로 배우는 대규모 시스템 설계 기초 2 | 알렉스 쉬 p. 418~419

Saga

03-2. Orchestration-based saga(중앙 집중형)

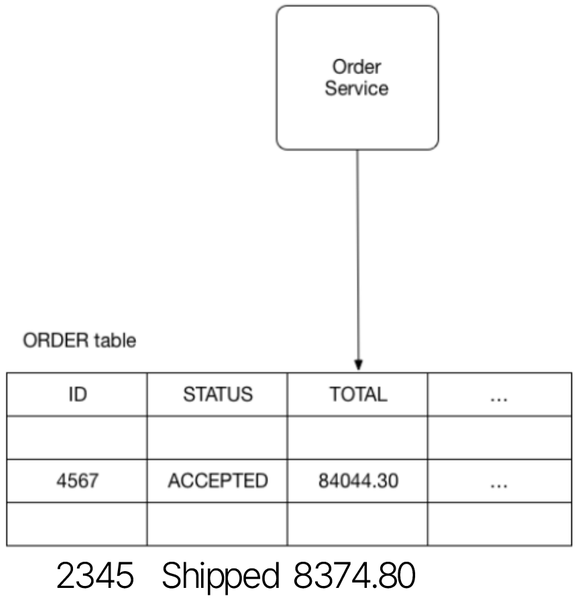


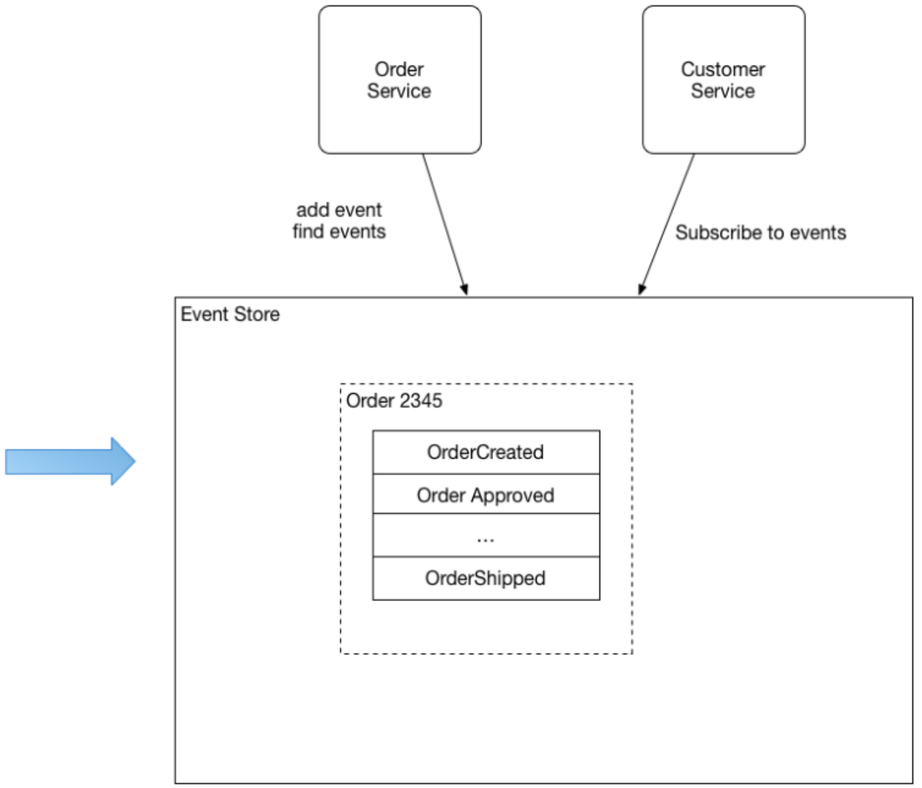
하나의 조정자(coordinator) 존재 / 복잡한 상황 처리

출처:Pattern: Saga / 가상 면접 사례로 배우는 대규모 시스템 설계 기초 2 | 알렉스 쉬 p. 418~419

Event Sourcing

01. 개념

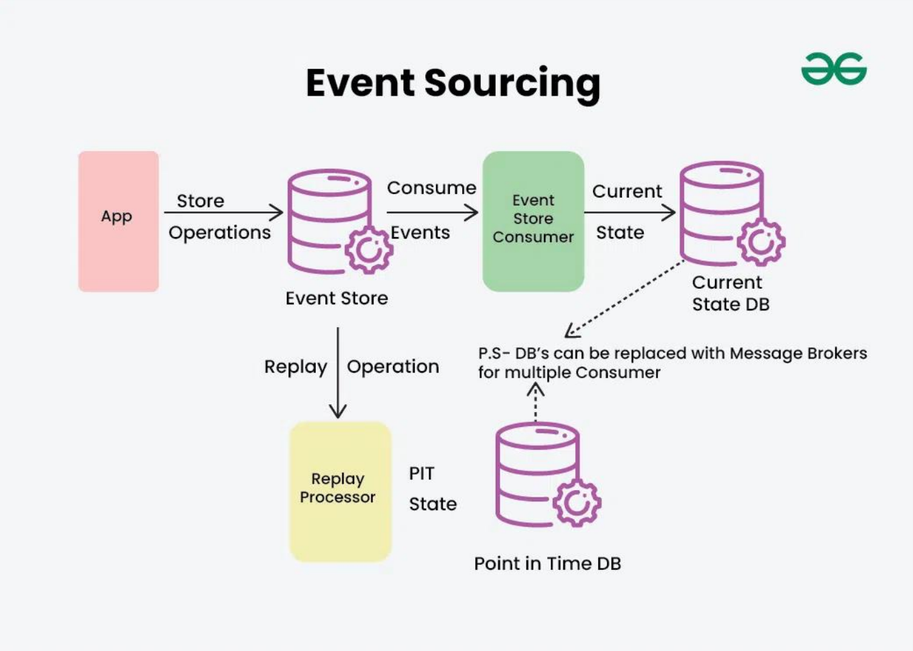




최신 상태만 저장하는 대신 모든 변경 사항을 일련의 이벤트로 기록해 데이터를 저장하는 방식

출처: Pattern: Event sourcing / greeks for greeks - Event Sourcing Pattern

Event Sourcing



02. 동작

- 상태 대신 이벤트 캡처: 모든 시스템 수정 사항은 최종 상태(예: 주문 완료)를 보존하는 대신 이벤트(예: 주문 생성됨, 품목 추가됨, 주문 완료)로 문서화됩니다. 모든 이벤트는 고유한 동작 또는 수정 사항을 나타냅니다.
- 이벤트 순차 저장: 모든 이벤트는 발생한 정확한 순서대로 순차적으로 저장됩니다.
- 이벤트 재생을 통한 상태 재구성: 현재 상태를 알아야 할 때 시스템은 과거 이벤트를 모두 재생하거나 처리해 상태를 처음부터 구축합니다.
- 새 이벤트 처리: UPDATE / DELETE가 따로 없이 새 이벤트를 생성하는 것으로 처리합니다.

출처: greeks for greeks - Event Sourcing Pattern

핵심어

Refactoring to Microservices

Transaction Messaging

DDD

CQRS

SAGA

Loose Coupling

Event Sourcing

문제

Eventuate Platform은
엔티티의 변경 이력 등 감사를
위한 기능은 지원하지 않는다.

○ X

문제

SAGA 패턴은 여러 서비스에 걸친 쿼리를 구현한 패턴이다.

○ X

참고

더 알아보고 싶다면,

Lose Coupling

- 요즘 IT - 느슨한 결합(loosely coupled) 원칙을 활용한 소프트웨어 설계

Eventuate 사용 예제

- Eventuate - Eventuate example microservices applications

Saga 패턴

- Medium - Saga pattern: Choreography and Orchestration
- Medium - Microservices Patterns: The Saga Pattern

Event Sourcing

- 강남언니 - [SaaS] 시간여행이 가능한 시스템 아키텍처
- greeks for greeks - Event Sourcing Pattern

질문

보충 / 질문 시간

의견대환영~!

관련 주제 발표 일정을 참고해 주세요.

- 5/10 CQRS(Command Query responsibility Segregation)
- 5/13 Domain-Driven Design(DDD)
- 5/22 Transaction Management