

Ted M. Young

Java Trainer, Coach, & Live Coder



Event-Sourcing from Scratch

How Should We Persist Data?

Me: <https://ted.dev/about>

BlueSky: [@ted.dev](#)

Twitch: <https://JitterTed.Stream>

YouTube: <https://JitterTed.TV>

Source Code? Slides? Go here:

<https://ted.dev/talks>

Ted M. Young
ted@tedmyoung.com

I Can Help Your Team...

Write more Testable code
with more Effective tests

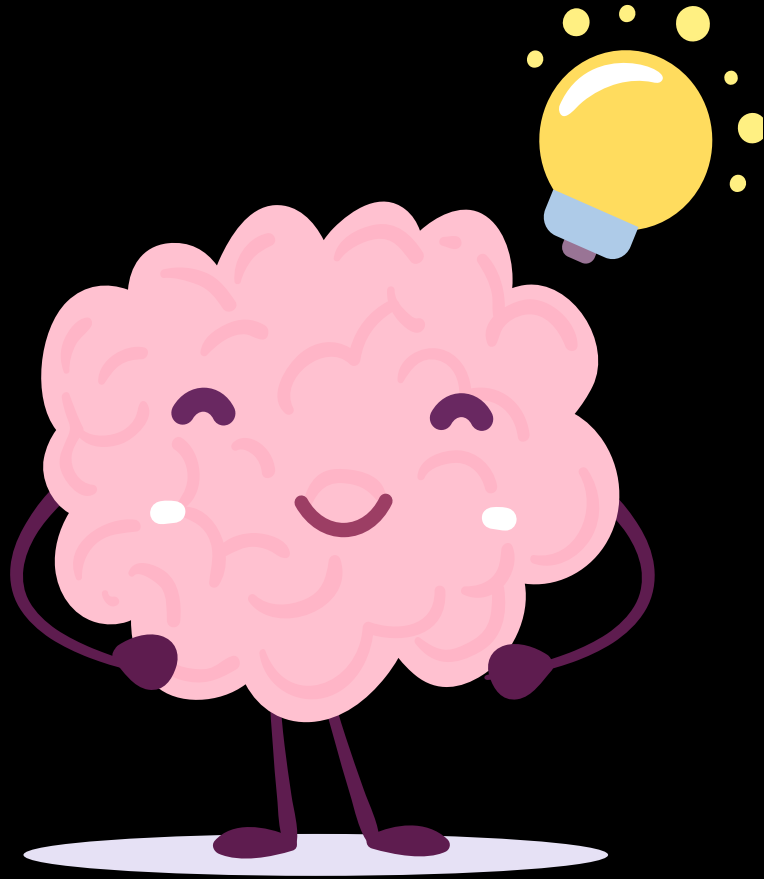
Be more productive in
Java & Spring

Effectively use
TDD

**Refactor
Messy
Code**



Ask Questions As You Need



I may defer the
answer if I'm going
to cover it later!



Event Sourcing Frameworks/Libraries

Java:
AxonIQ

.NET:
Marten

PHP:
EventSauce





What's in this TED Talk...

1. How do we persist data?
2. Define terms, clear up misconceptions
3. JitterTix: Concert Event Ticketing domain
4. Dig into code (warning: generics!) & **tests**
5. Related topics to explore on your own:
 - CQRS, Event Modeling, Event Storming
 - Versioning & Schema Migration
 - Performance & Snapshotting
 - Dynamic Consistency Boundary & Decider & GDPR



How do we Persist Data?

Often: Map Objects to Database Tables (ORM)

Mapping Objects from/to Database



r/ExperiencedDevs
u/Fuzzy_World427 · 10h

DDD: How do you map DTOs when entities have private setters?

Hey all,

I'm running into trouble mapping DTOs into aggregates. My entities all have **private setters** (to protect invariants), but this makes mapping tricky.

I've seen different approaches:

- Passing the whole DTO into the aggregate root constructor (but then the domain knows about DTOs).
- Using mapper/extension classes (cleaner, but can't touch private setters).
- Factory methods (same issue).
- Even AutoMapper struggles with private setters without ugly hacks.

So how do you usually handle mapping DTOs to aggregates when private setters are involved?



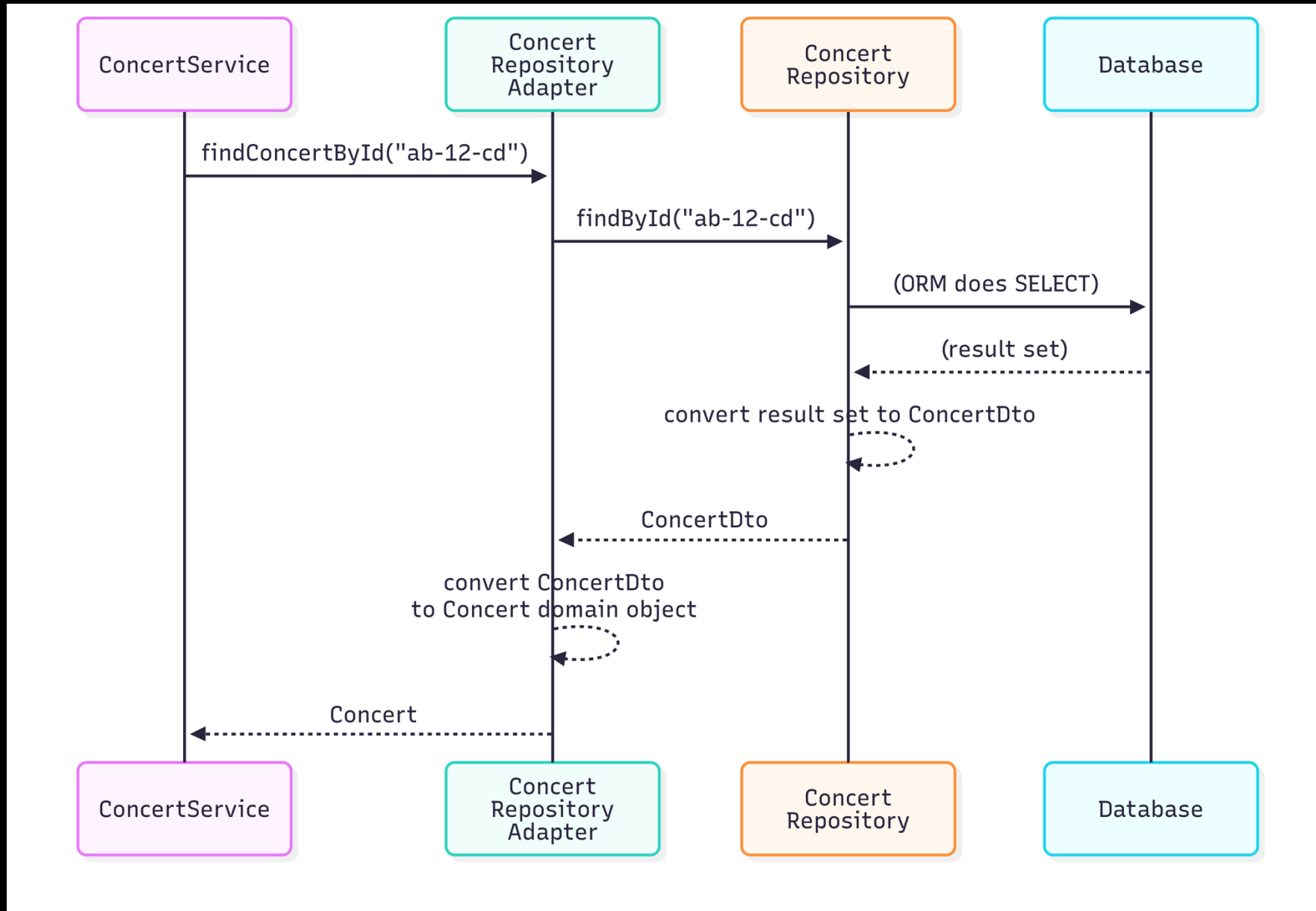
24



51

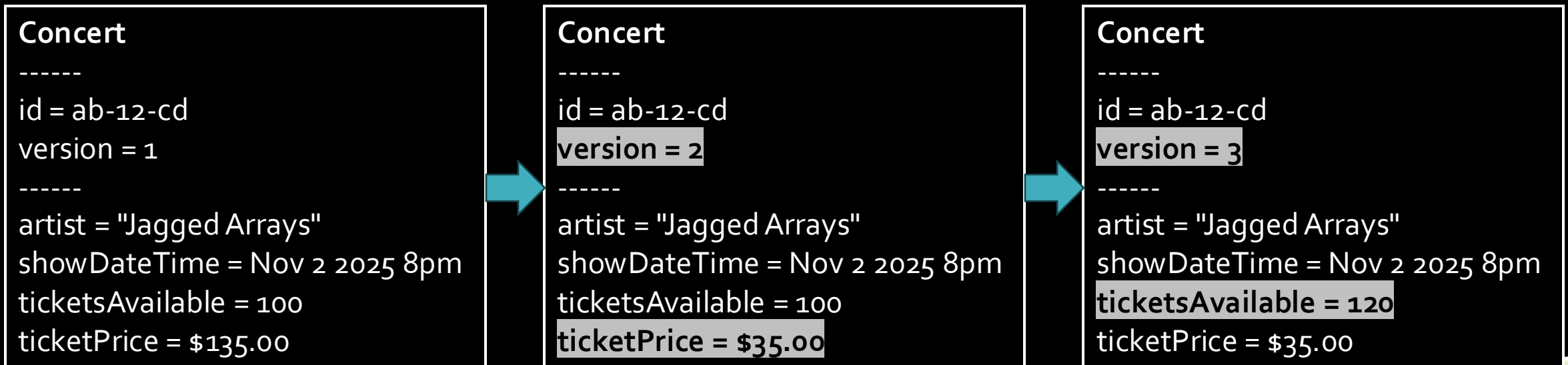


Adapter-Driven Persistence



Storing Data as "CRUD" or "State"

- "State-Sourced" / "CRUD-Sourced"
- No history, throws away old information
- No knowledge of how/why state changed



Fundamentals

Past

Event

Present

State

Future

Command



Definitions: EVENT (the past)

a fact, something that happened

CustomerRegistered

ConcertScheduled

TicketPurchased

TicketTransferred



Definition: STATE (now)

Data model needed by the application.

Concert

id = ab-12-cd

version = 1

artist = "Jagged Arrays"

showDateTime = Nov 2 2025 8pm

ticketsAvailable = 100

ticketPrice = \$135.00



Definition: COMMAND (future)

Request to change state

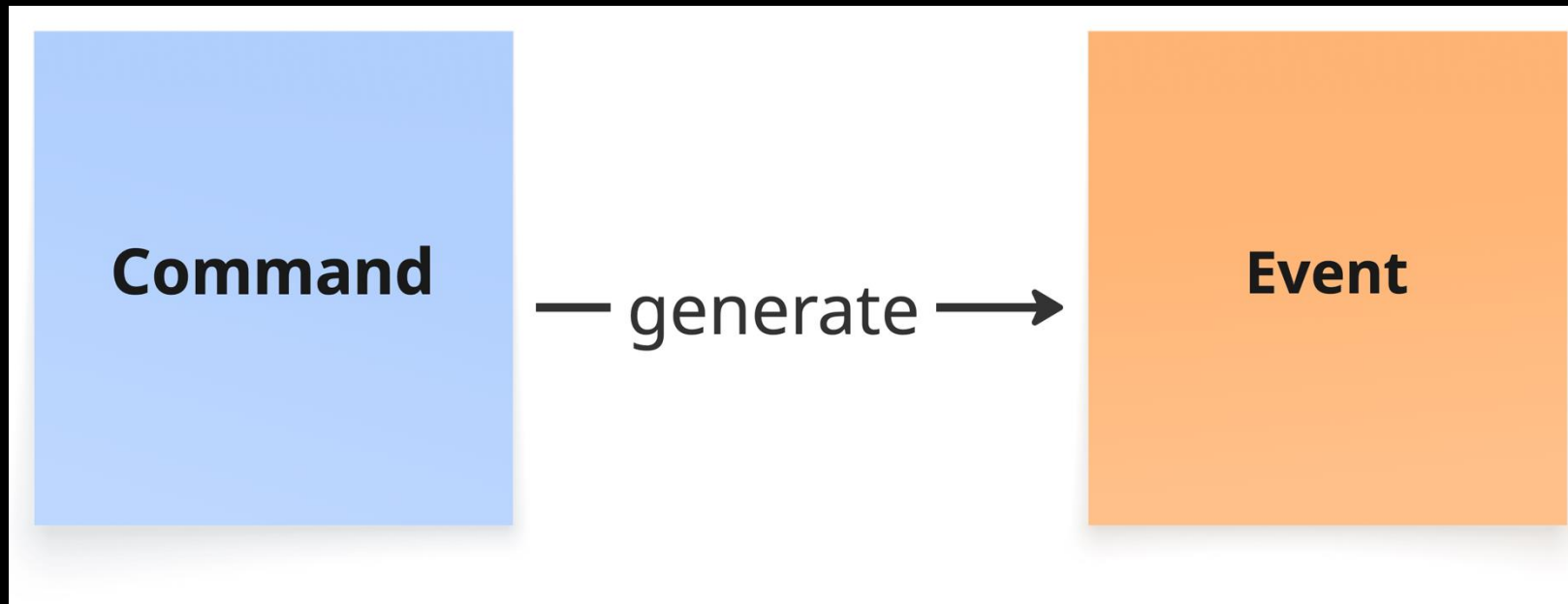
schedule

reschedule

purchaseTickets



Commands Generate Events



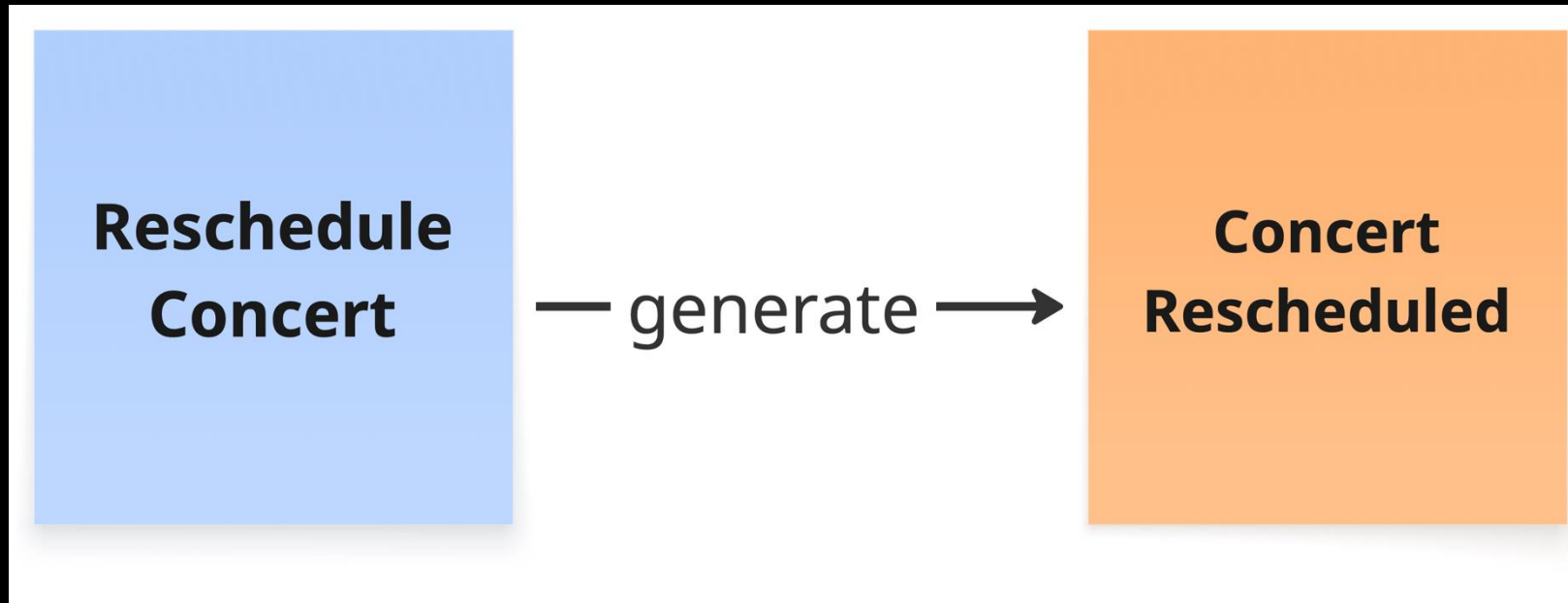
Command

Event

State



Commands Generate Events



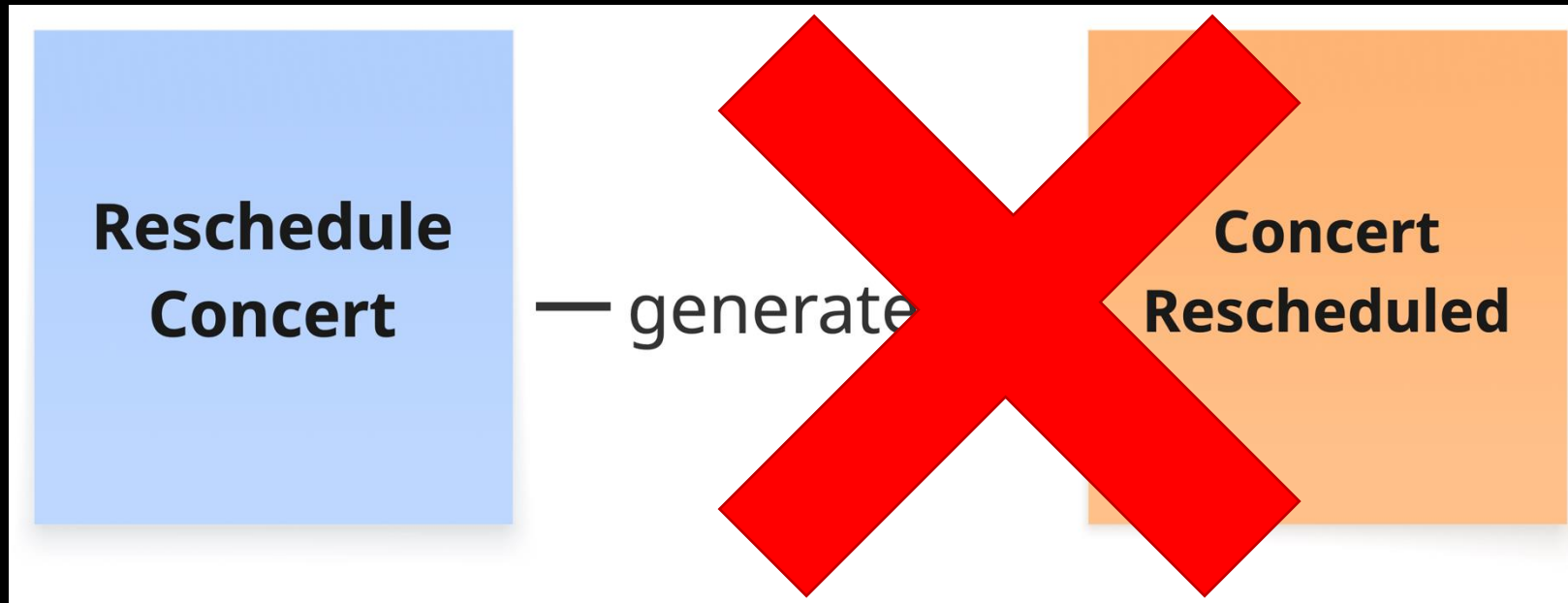
Command

Event

State



...unless It's Not Allowed



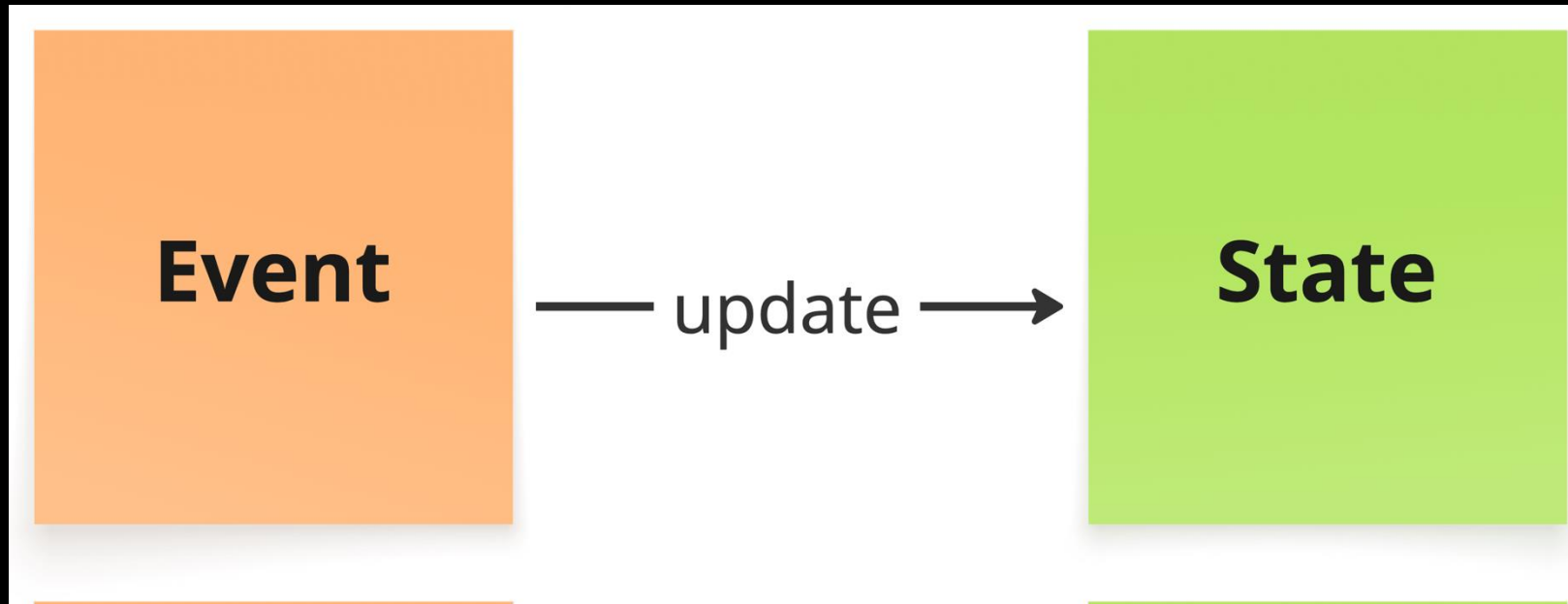
Command

Event

State



Events Update State



Command

Event

State



Events Update State (always!)



Command

Event

State



Events *Project* to State

Command

Event

State

**Concert
Scheduled**

Sonic Waves
Oct 30, 2025
\$135, 100 tix

+

**Ticket Price
Changed**

\$35

+

**Concert
Rescheduled**

Nov 2, 2025

— projected →

Concert

Sonic Waves
Nov 2, 2025
\$35, 100 tix



Event-Sourcing PROJECTION

A function that constructs a *data model* by sequentially *replaying events* from the *event store*.



Event Store

An append-only "database" that records all events generated by the application.



Benefits of Event-Sourcing

Ask Questions You Didn't Know You Had

How many customers transfer tickets?

Immutable Events

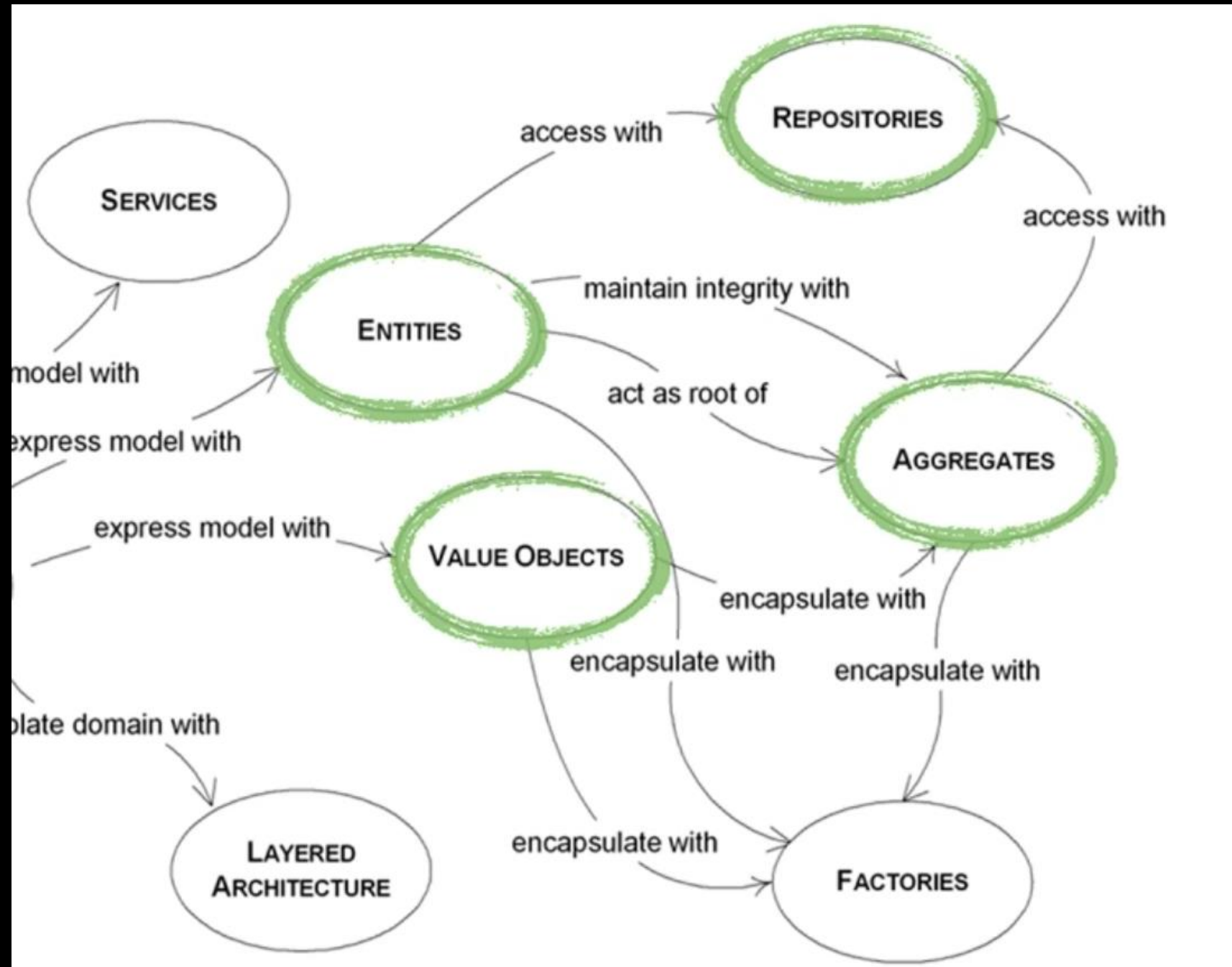
Never lose information: there is no DELETE

Replayability

Time Travel: see state of system in the past

Domain-Driven Design

Tactical Patterns



Entity

Unique (Has Identity), History, and Attributes

Value Object

Identified Only by its Attributes (immutable)

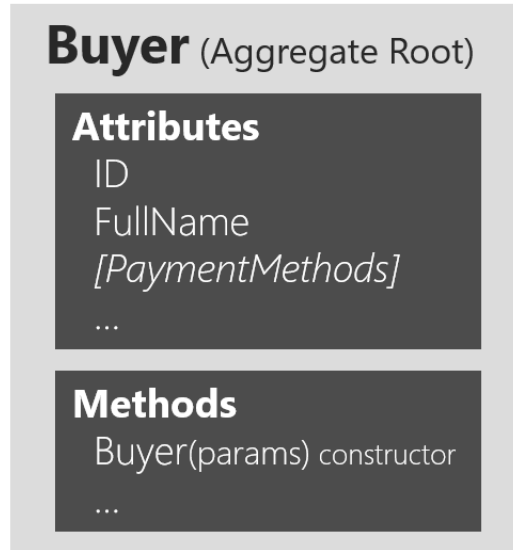
Quantify, describe, and measure

Aggregate

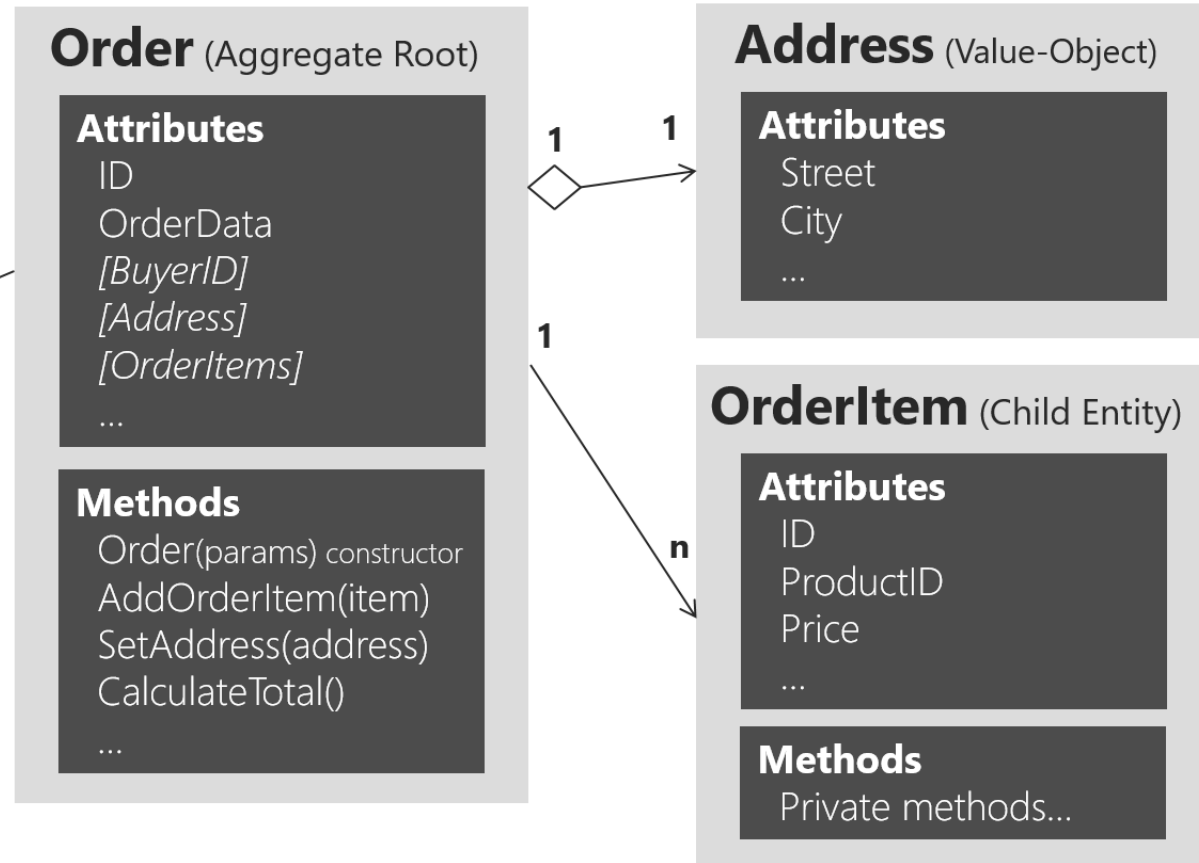
Consistency [transactional] Boundary
(Enforced by "Aggregate Root" Entity)

Aggregate Pattern

Buyer Aggregate (One entity)

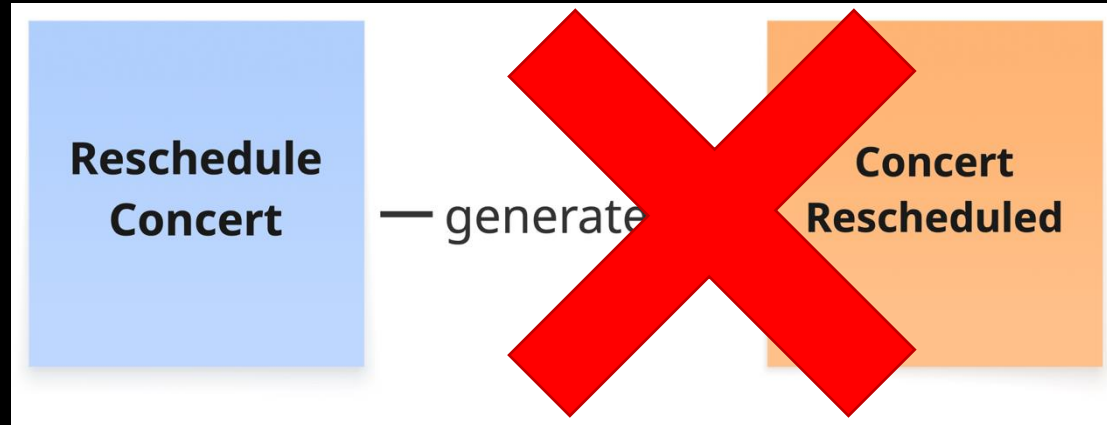


Order Aggregate (Multiple entities and Value-Object)



Repository

Used for Storing and Retrieving Aggregates



Aggregate & Projections

Aggregates in Event-Sourcing is a *Projection*
and the *Decider*

Misconceptions

Event-Driven Architecture

Different Kinds of Events

CQRS

Command-Query Responsibility Segregation

Does Not Require Event-Sourcing!

JitterTix Concert Ticketing

The Event-Sourced Ticketing System

Explore Concert Tickets

Event Viewer

Copyright © 2025 JitterTed Productions. All rights reserved.

<https://ted.dev/about>



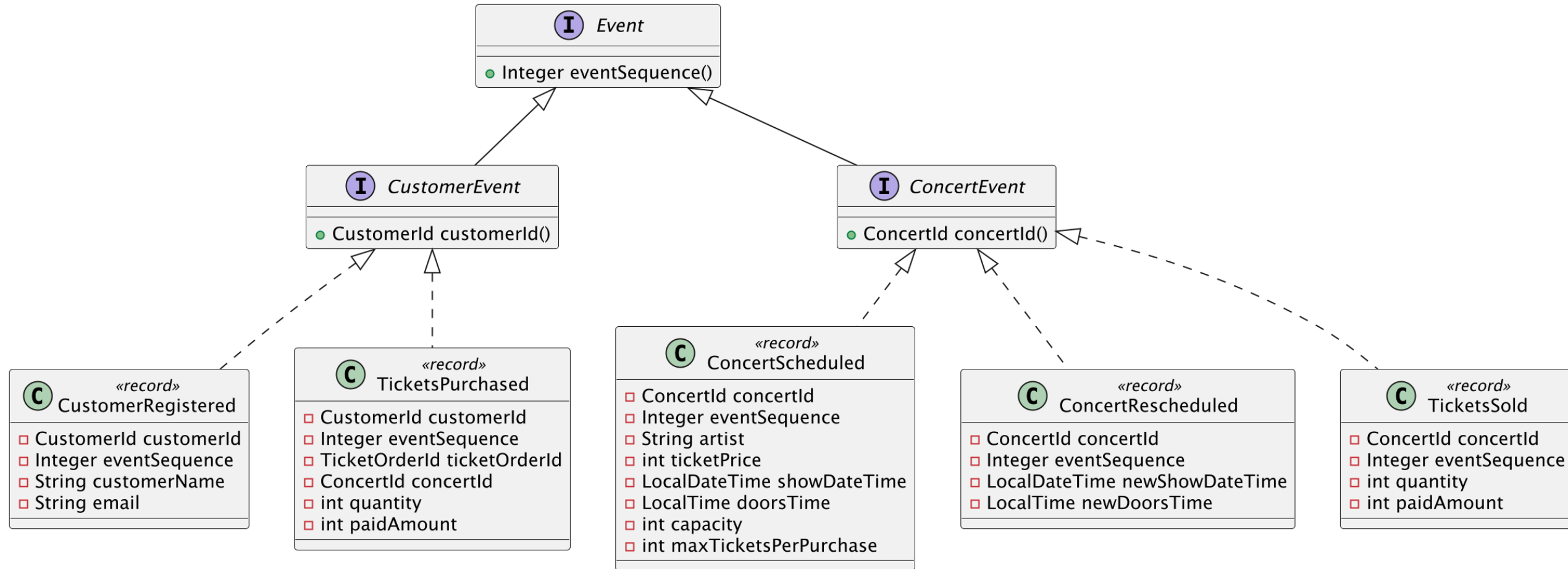
JitterTicket Design

tldraw

JitterTix: Concert Ticketing System

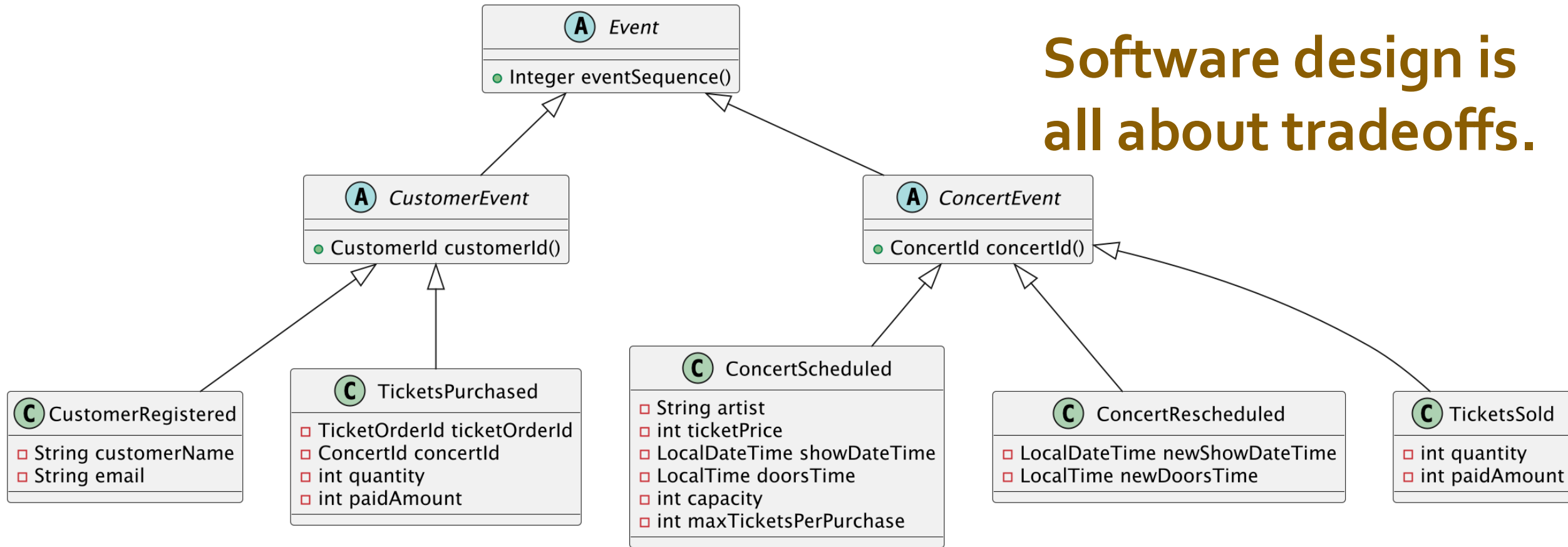
Diving Into the Code

Events as Java Records



Events as Java Classes

Software design is all about tradeoffs.



Code Walkthrough

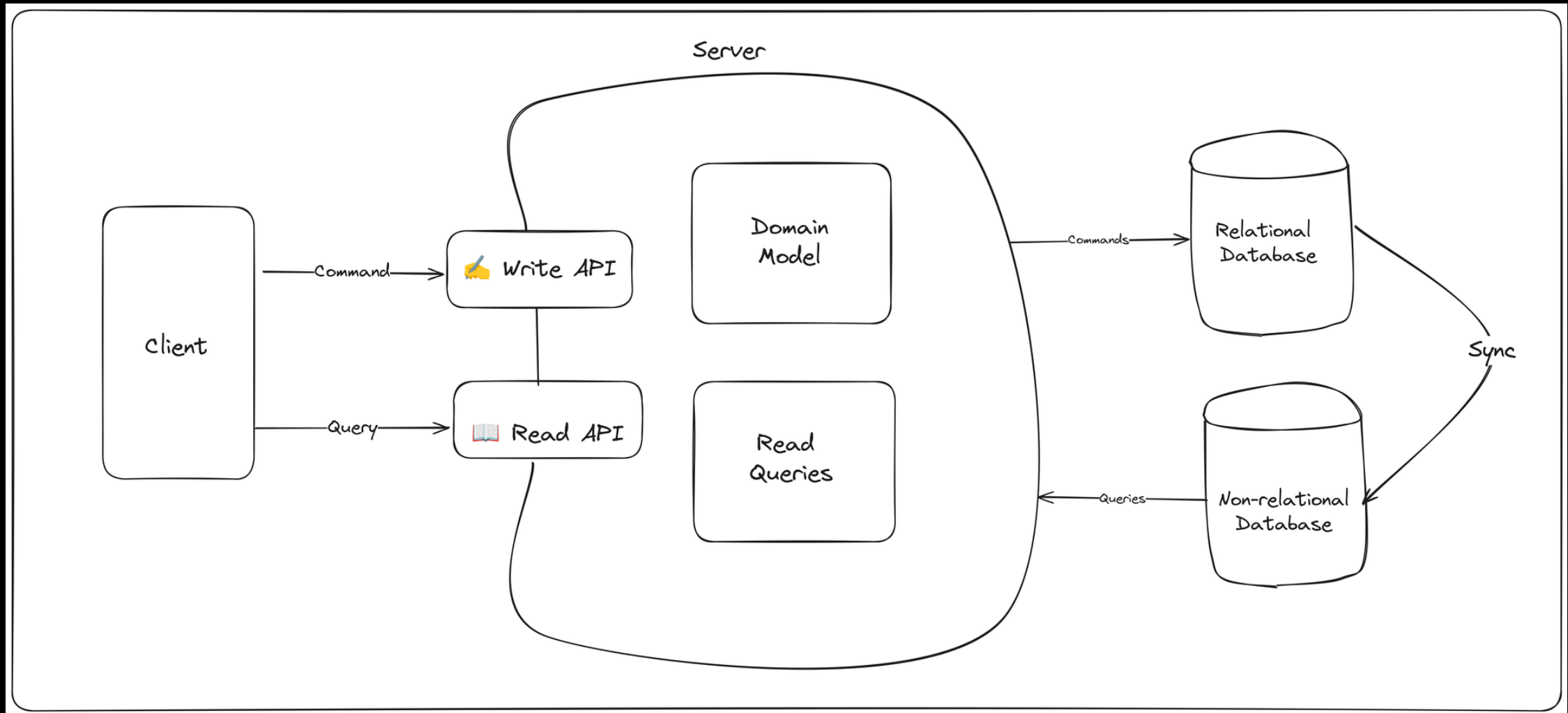
Versioning Event Logs (Streams)

(see Greg Young's Versioning in an Event-Sourced System)

- **Copy-Replace**
- **Split-Stream**
- **Join-Stream**
- **Copy-Transform**



CQRS – Separated Data Models

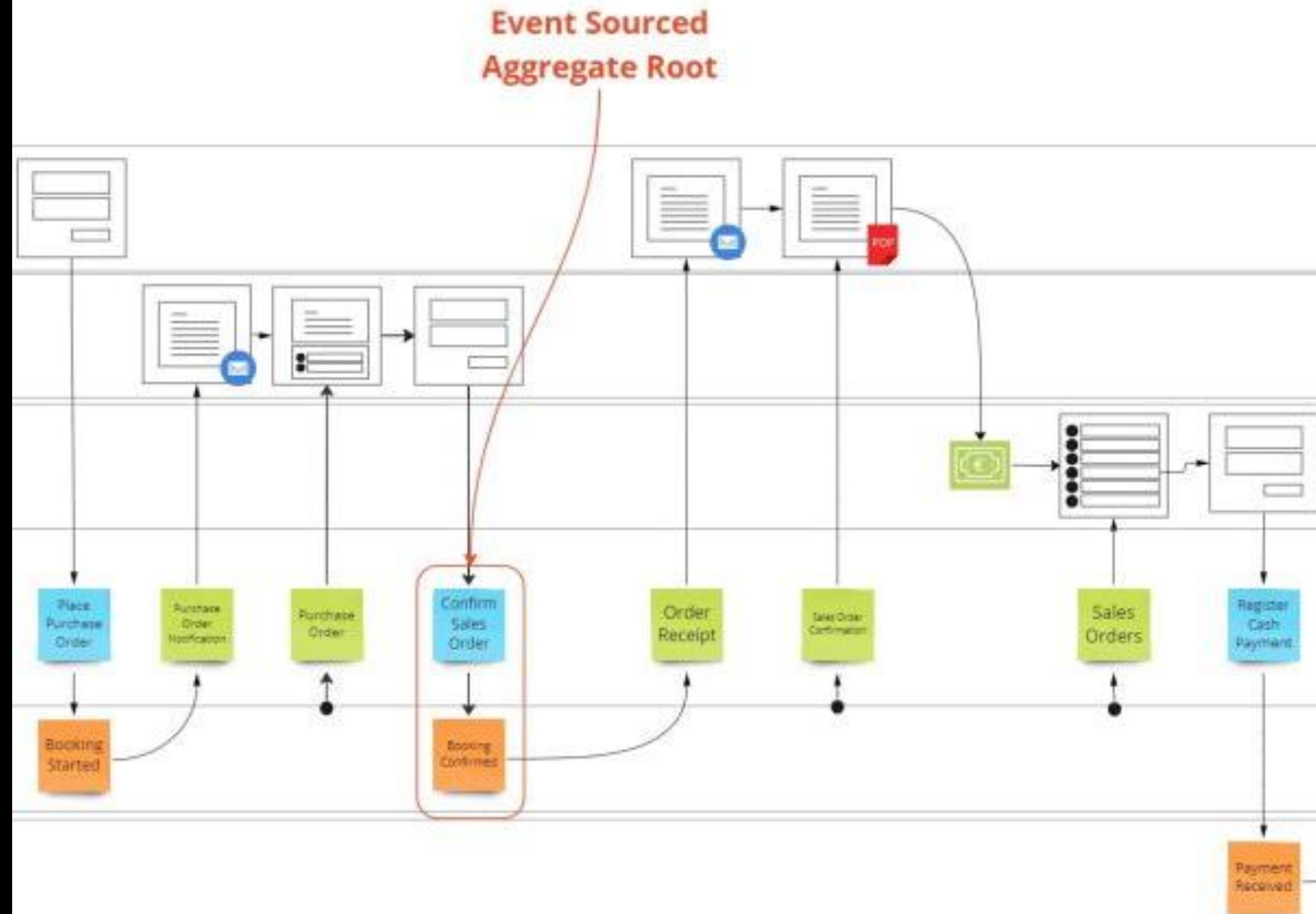


Event Modeling

Command

Event

State





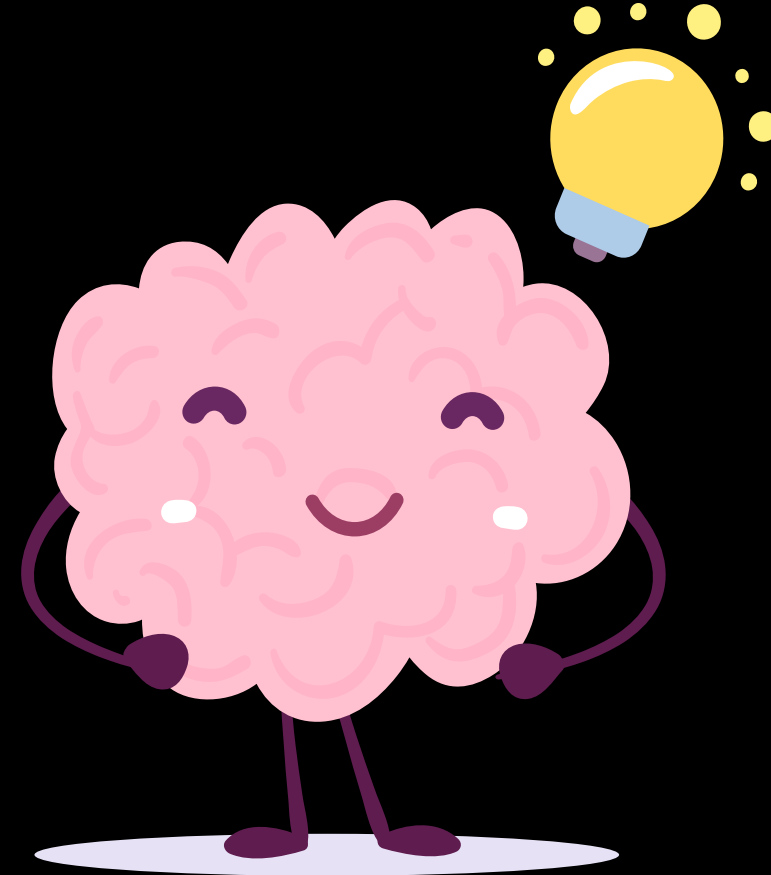
Ted M. Young

Java Trainer, Coach, & Live Coder

Get in touch: ted@tedmyoung.com

About me: <https://ted.dev/about>

What other questions do you have?



Ted M. Young

Java Trainer, Coach, & Live Coder

Get in touch: ted@tedmyoung.com

About me: <https://ted.dev/about>

Thank You...

Source Code? Slides?

<https://ted.dev/talks/>



**Don't forget
to rate this
session!**