Ted M. Young

Java Trainer, Coach, Live Coder, and Creator of JitterTed's TDD Game

Event Sourcing

The End of DDD Tactical Patterns?

Me: https://**ted.dev**/about

Source Code? Slides?

https://ted.dev/talks

Ted M. Young

ted@tedmyoung.com

I Can Help Your Team…

Write more Testable code
with more Effective tests

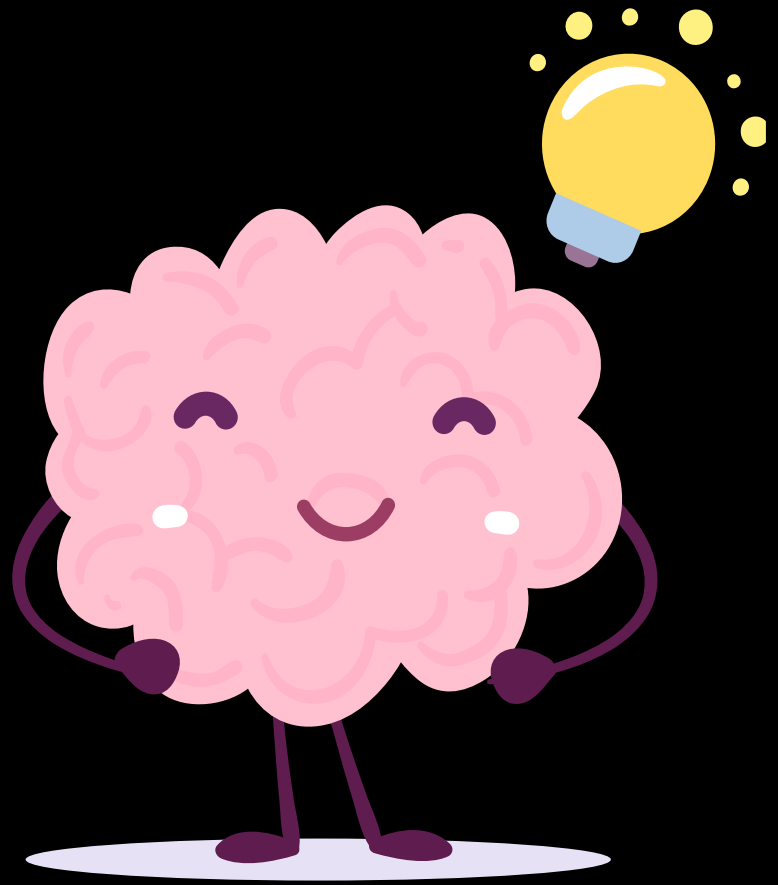
Be more productive in
Java & Spring

Effectively use
TDD

Refactor Messy Code

https://ted.dev/about

Ask Questions As You Need



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

Event Sourcing Frameworks/Libraries

.NET
Marten

PHP
EventSauce
Ecotone

JavaScript
Emmet

Java
Axoniq

What's in this TED Talk...

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

https://ted.dev/about

What is APPLICATION STATE?

Outcome of DOMAIN DECISIONS made over TIME

JitterTicket Domain

CUSTOMER

CONCERT

TICKET

https://ted.dev/about

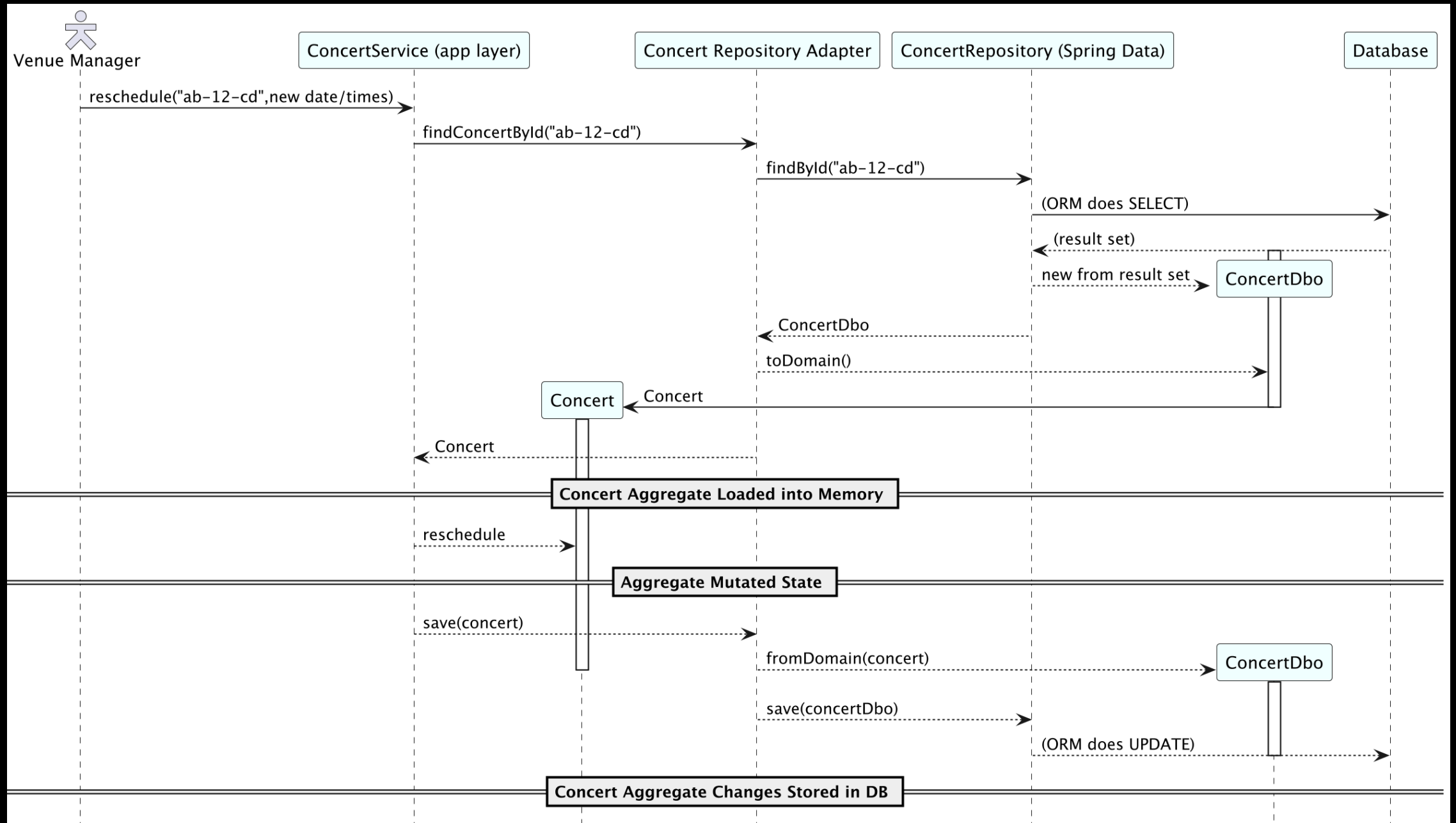
Domain Decisions

- Purchase Ticket
  - Decision (Domain Rule): enough tickets?
    - Yes -> State Change: Ticket Purchased
    - No -> No State Change
- Reschedule Concert
  - Rules: not canceled; in the future
    - Yes -> State Change: Concert Rescheduled
    - No -> No State Change (report why)

How to Persist Decisions?

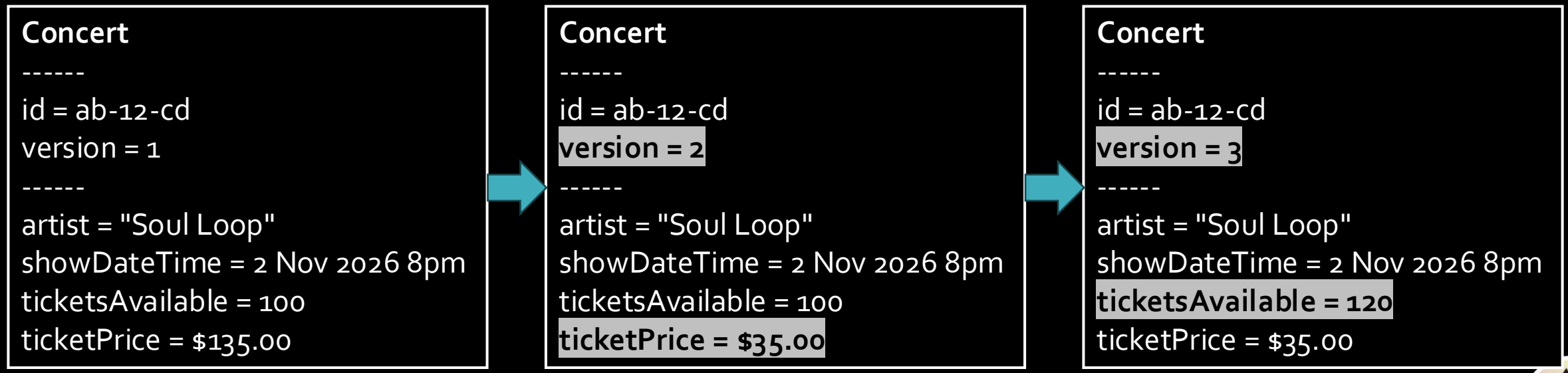
Often: Map Objects to Database Tables (ORM)

Adapter-Driven ORM Persistence



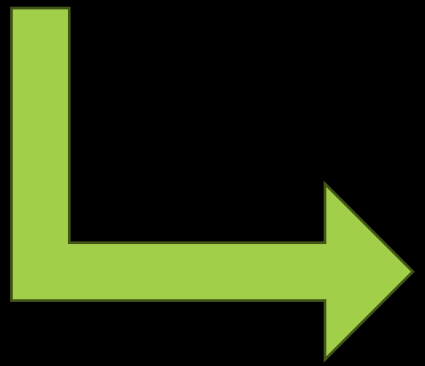
Storing Data as "Latest State"

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



Storing Data as Changes in State

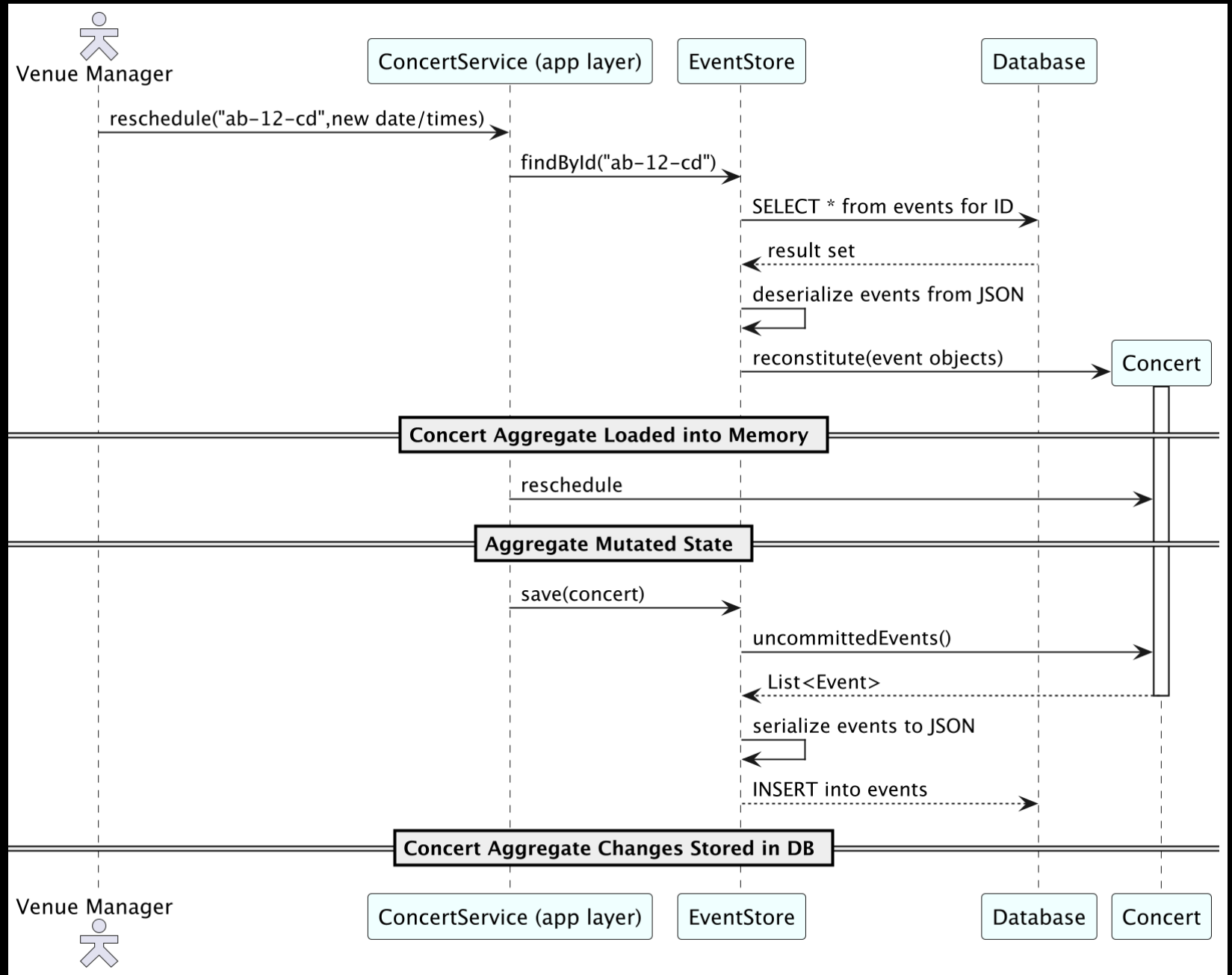
Concert Scheduled:	**23, ab-12-cd, "Soul Loop", 2 Nov 2026, 20:00, 100, 135.00**
Ticket Price Changed:	**41, ab-12-cd, 35.00**
Capacity Increased:	**73, ab-12-cd, 120**



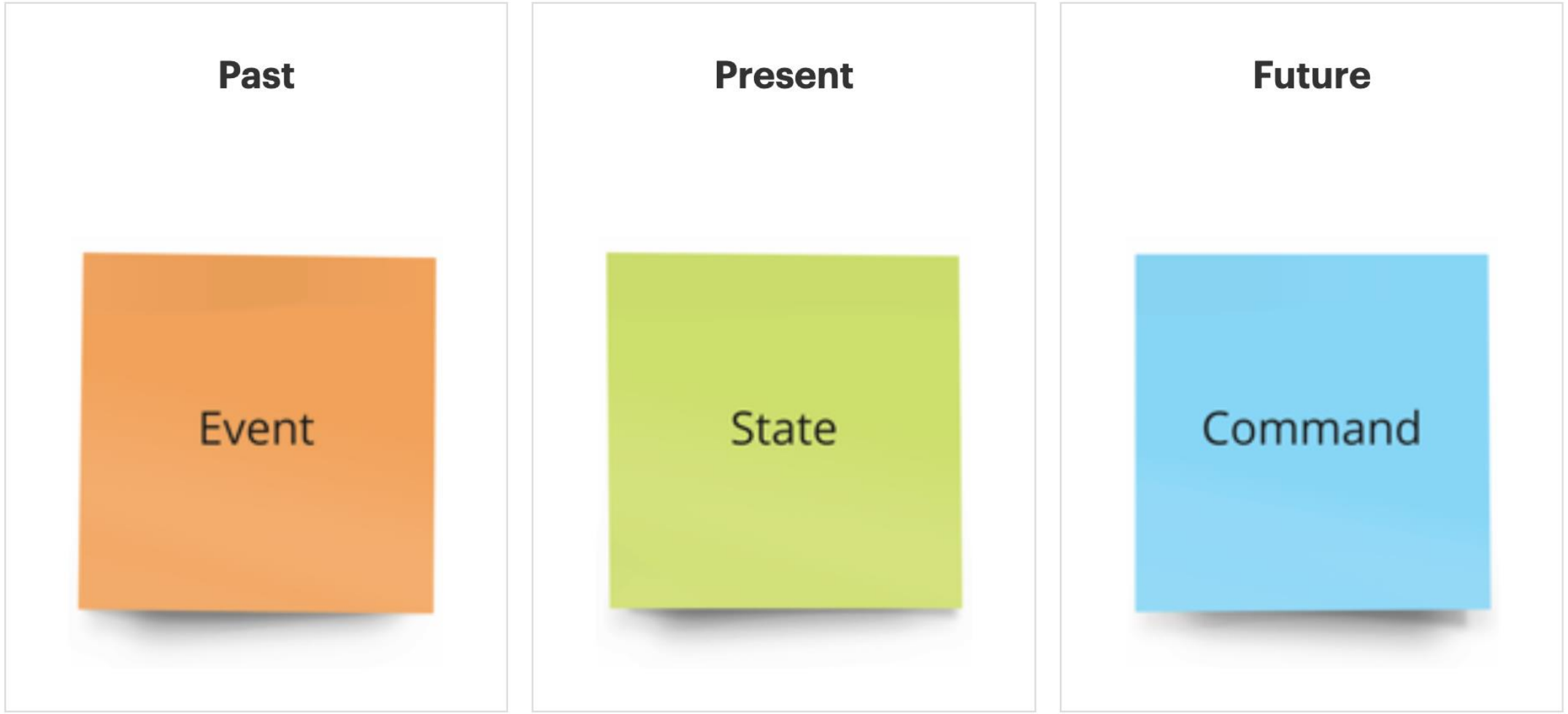
Concert
---
id = ab-12-cd
---
artist = "Soul Loop"
showDateTime = 2 Nov 2026 8pm
ticketsAvailable = 120
ticketPrice = $35.00

https://ted.dev/about

Event-Sourced Persistence



Fundamentals



https://ted.dev/about

Definitions: EVENT (the past)

a *fact,* something that *happened*

Customer Registered

Concert Scheduled

Ticket Price Changed

Ticket Purchased

https://ted.dev/about

Definition: STATE (now)

Context needed by the application to *DECIDE*

Concert
---
id = ab-12-cd
version = 1
---
artist = "Soul Loop"
showDateTime = 2 Nov 2025 8pm
ticketsAvailable = 100
ticketPrice = $135.00

Definition: COMMAND (future)

Request **to change state**

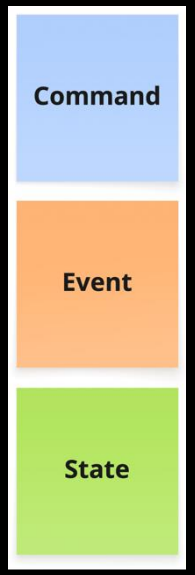
Register New Customer

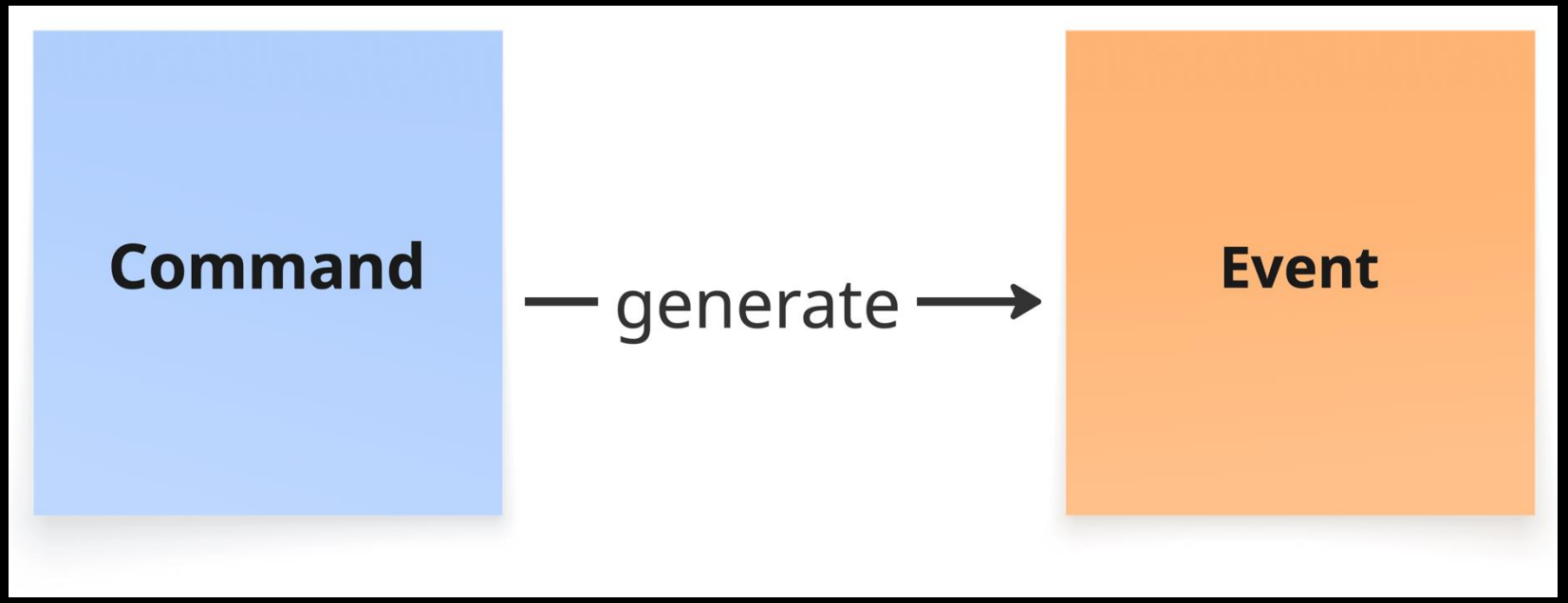
Schedule Concert

Change Ticket Price

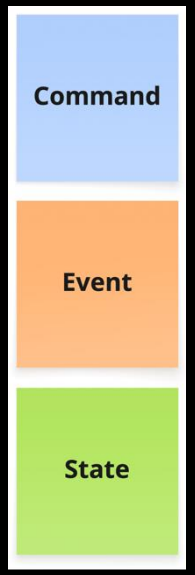
Purchase Tickets

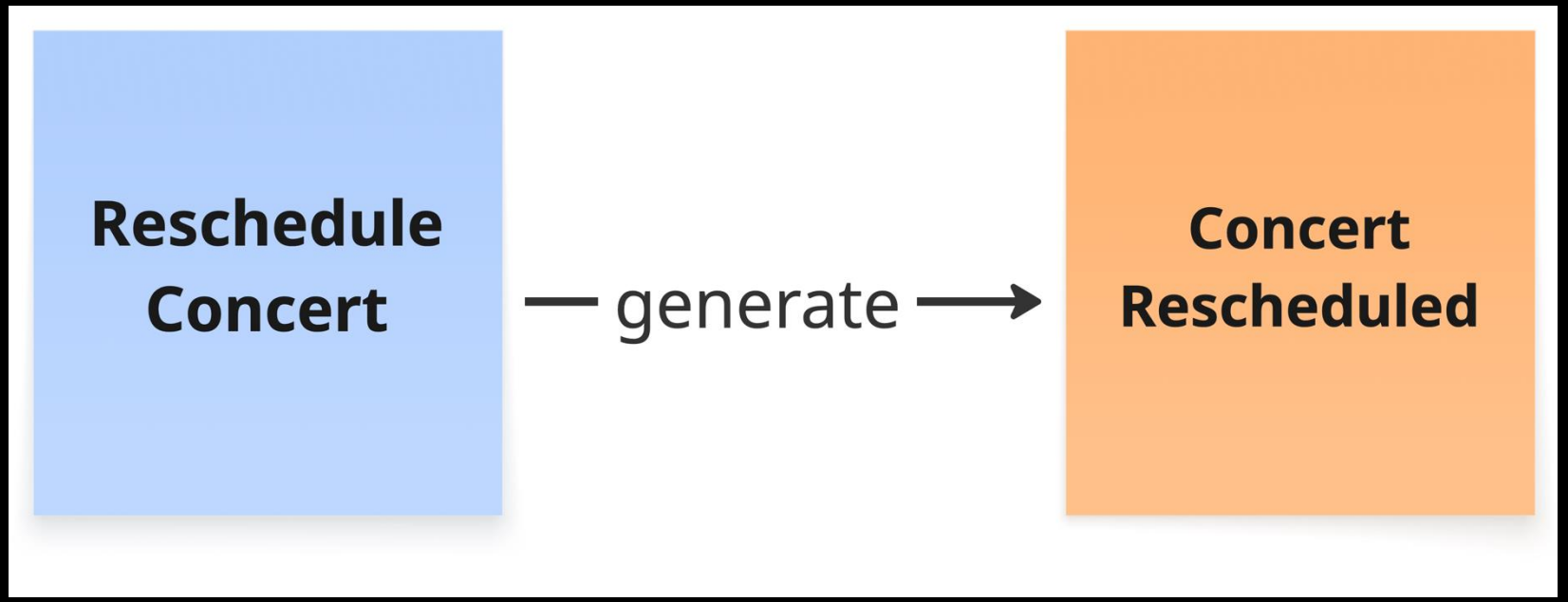
Commands Generate Events



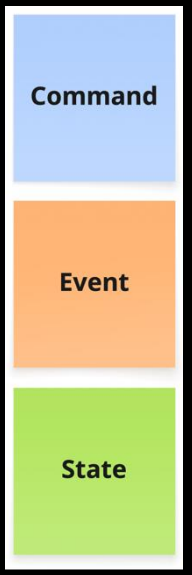


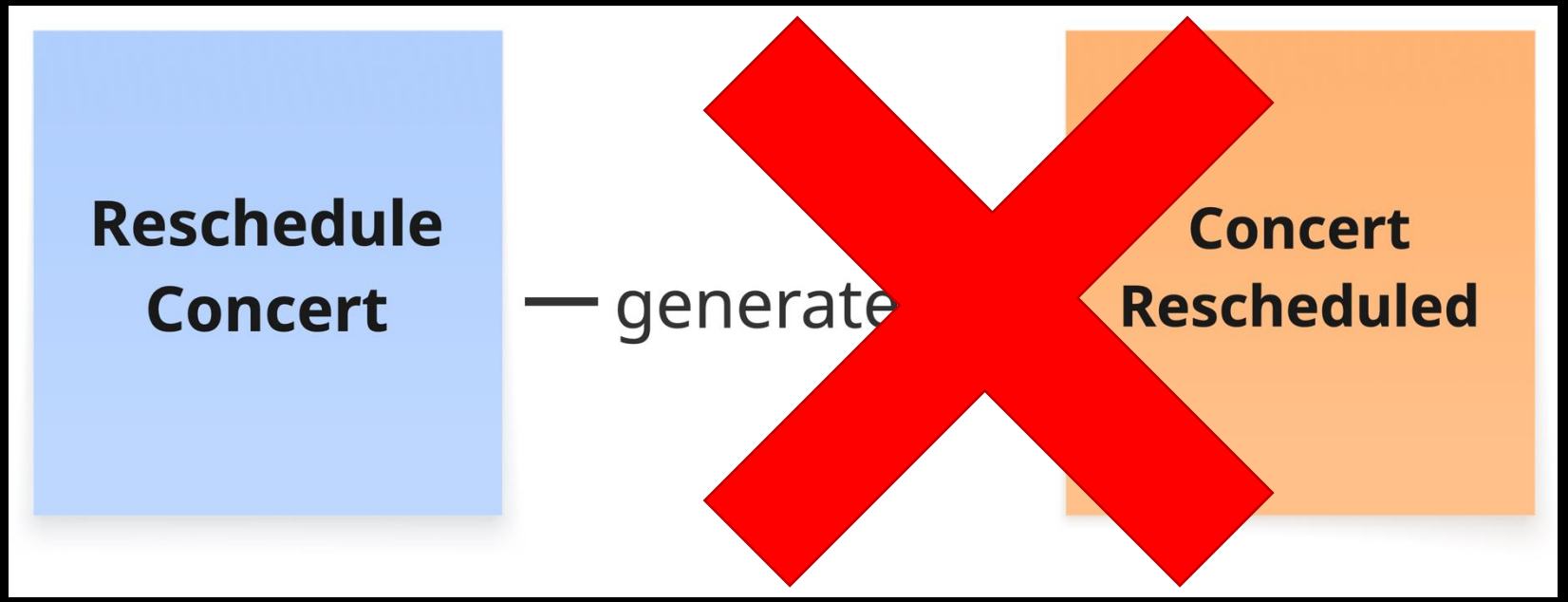
Commands Generate Events





...unless It's Not Allowed (Rule)



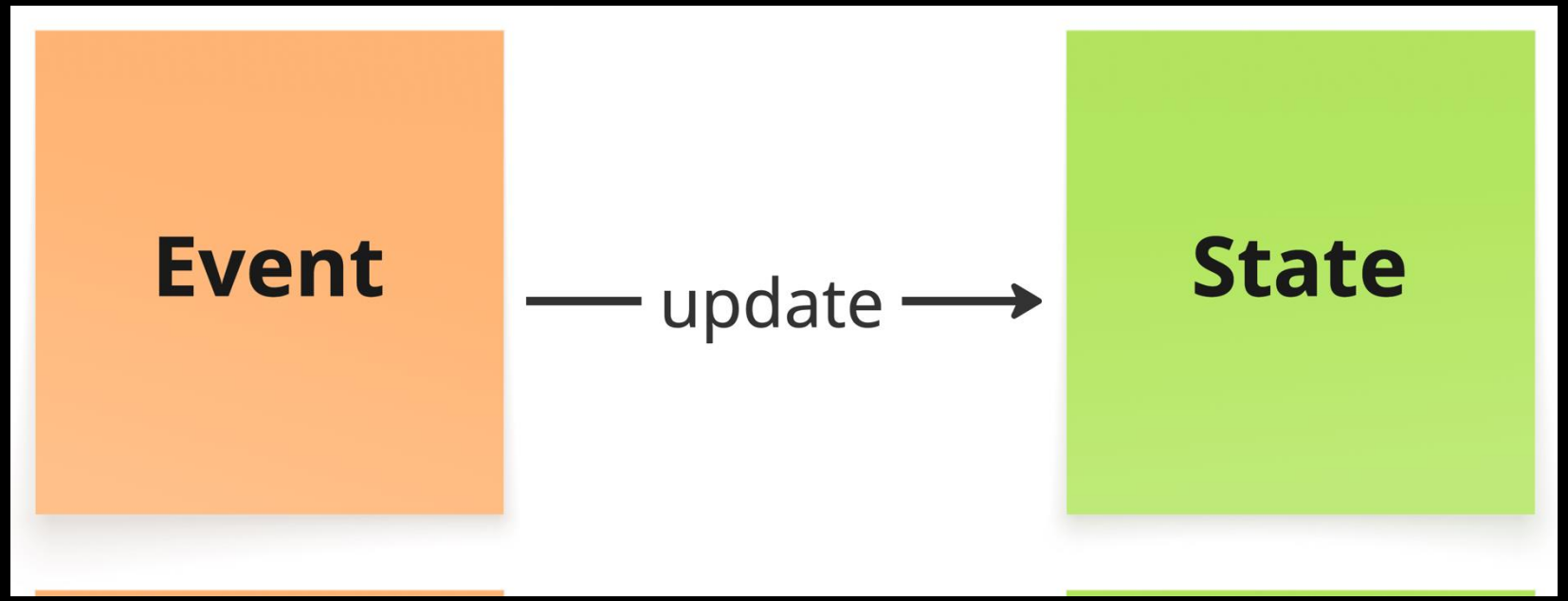


Events Update State

Command

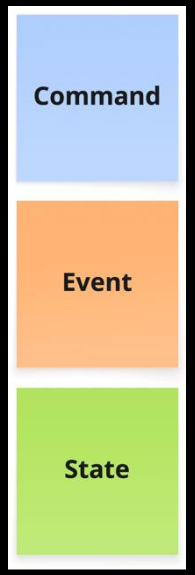
Event

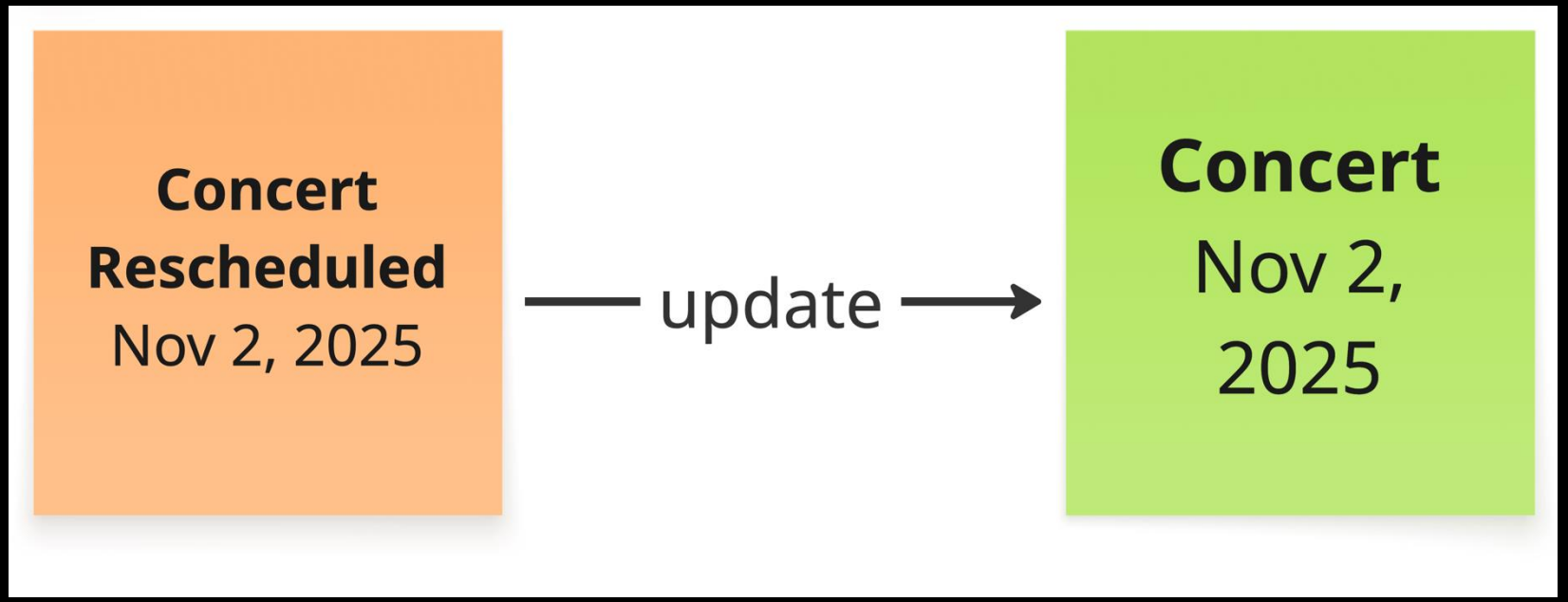
State



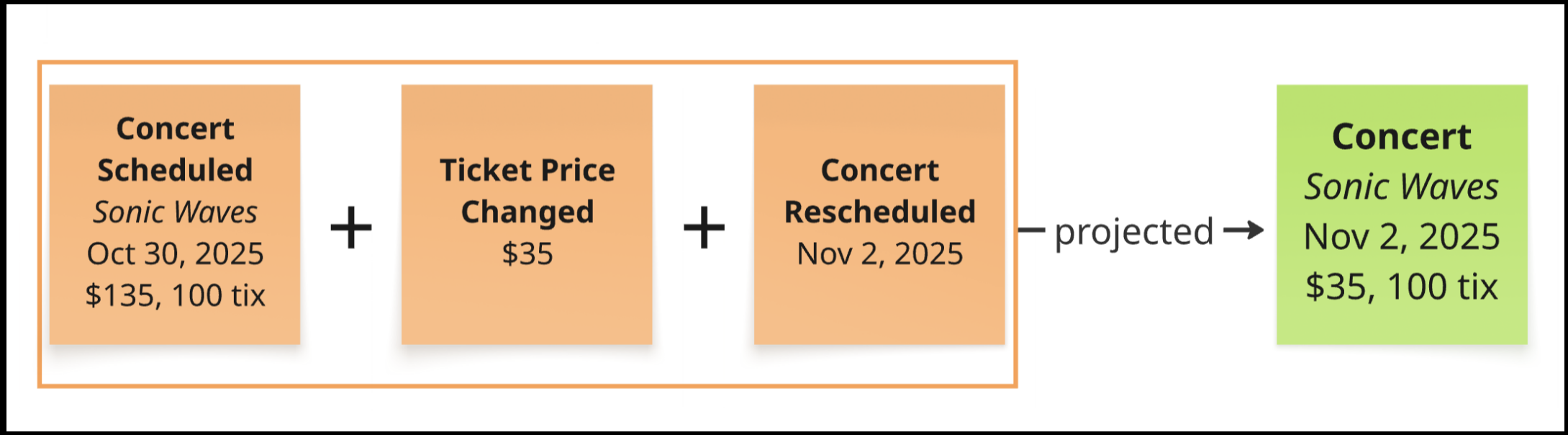
https://ted.dev/about

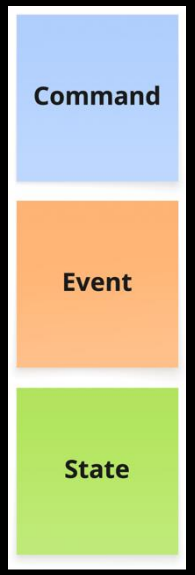
Events Update State (always!)





Events *Project* to State





Event-Sourcing PROJECTOR

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

Produces a PROJECTION

Projections Can Provide

- Current STATE (context) to DECIDE
- Customized Views for UI
- Database tables for integration

One Concept: Many Uses

Event Store

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

Events = Command + State + Decider

```
List<Event> decide(State, CommandParams)
```

```
record Aggregate(State state) {
  List<Event> decide(CommandParams commandParams) {
    if (state allows commandParams) {
      return List.of(new Event(data));
    }
    return emptyList();
  }
}
```

Events = Command + State + Decider

```
record Aggregate(State state) {
  private List<Event> uncommittedEvents = new ArrayList<>();

  void decide(CommandParams commandParams) {
    if (state allows commandParams) {
      uncommittedEvents.add(new Event(data));
    }
  }

  Stream<Event> uncommittedEvents() {
    return uncommittedEvents.stream());
}
```

Events = Command + State + Decider

```
private String artist;
private int ticketPrice;
private LocalDateTime showDateTime;
private LocalTime doorsTime;
private int capacity;
private int maxTicketsPerPurchase;
private int availableTicketCount;
private boolean ticketsOnSale = true;
```

```
public void rescheduleTo(LocalDateTime newShowDateTime, LocalTime newDoorsTime) {
    if (newShowDateTime.isAfter(showDateTime) && capacity - availableTicketCount == 0) {
        ConcertRescheduled concertRescheduled =
                new ConcertRescheduled(getId(), null, newShowDateTime, newDoorsTime);
        enqueue(concertRescheduled);
    }
}
```

Benefits of Event-Sourcing

Ask Questions You Didn't Know You Had

How many customers transfer tickets?

How often are we rescheduling concerts?

Immutable Events

Never lose information: there is no DELETE

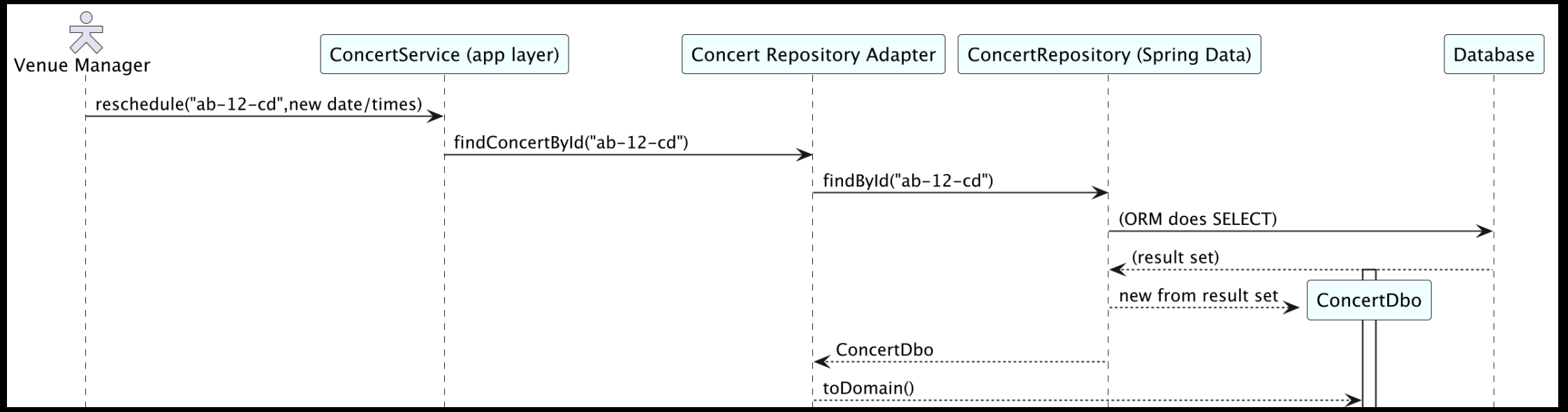
Fewer Concepts

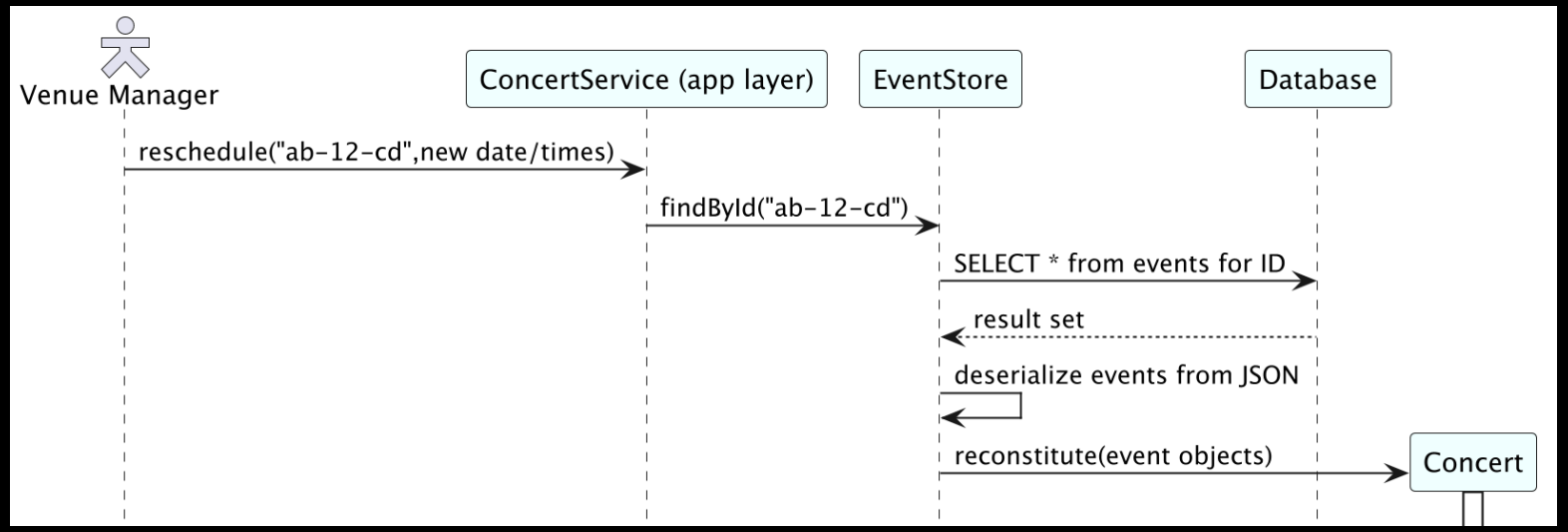
Commands, Events, Projections

Less complexity than ORMs

No awkward mapping, lazy loading, or dirty-checking

Fewer Concepts

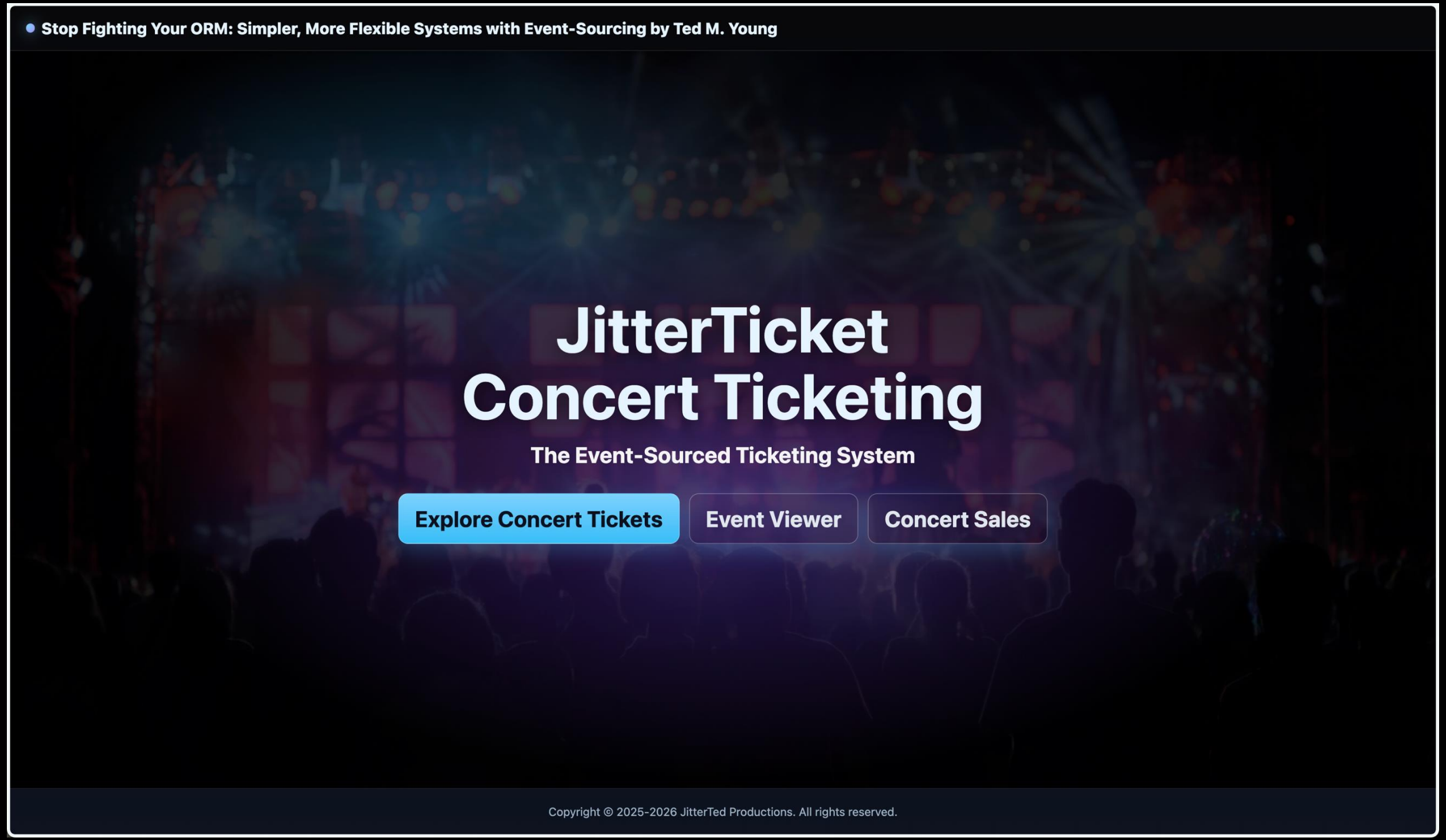




https://ted.dev/about

Replayability

Time Travel: see state of system in the past



https://ted.dev/about

JitterTicket Flows

tldraw

JitterTicket: Concert Ticketing System

Diving Into the Code

Let's talk about Performance

Similar, if not BETTER than ORM

Misconceptions

Event-Driven Architecture

Often Integrating across Context Boundaries

Event Streaming

High Volume and/or Rate of Events

CQRS

Command-Query Responsibility Segregation

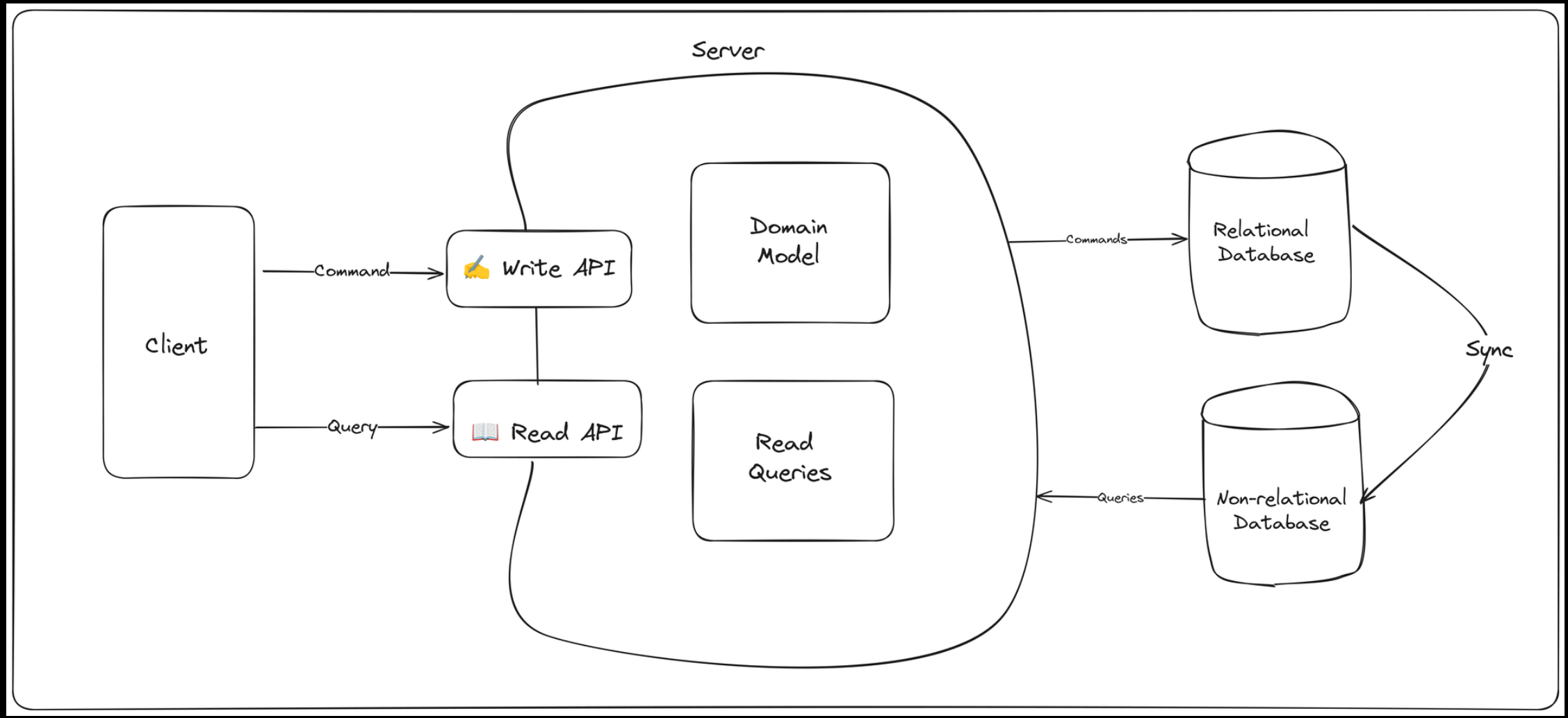
Does Not Require Event-Sourcing!

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

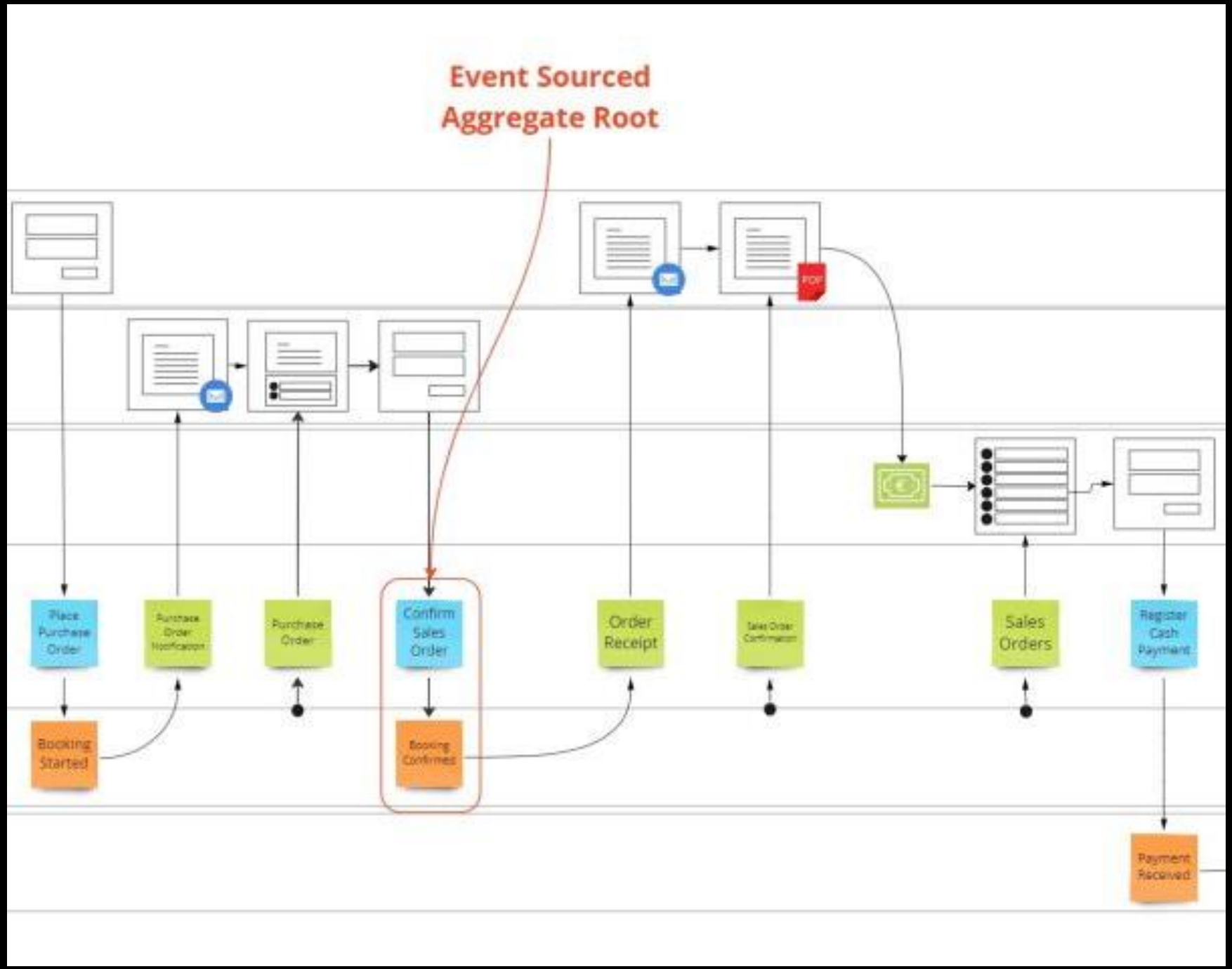


Event Modeling

Command

Event

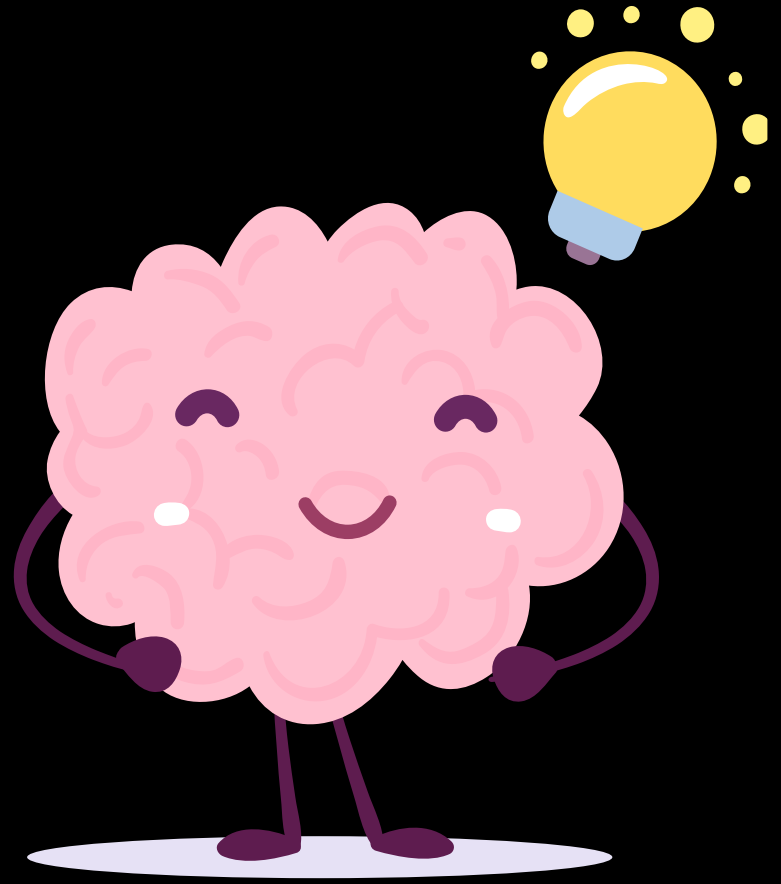
State



Ted M. Young
Java Trainer, Coach, Live Coder,
and Creator of JitterTed's TDD Game

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,
and Creator of JitterTed's TDD Game

Get in touch: ted@tedmyoung.com
About me: https://ted.dev/about

Thank You...

Source Code? Slides?
https://ted.dev/talks/

Please provide feedback here: