Use Marten to build event sourced apps

with no regrets (mostly)

Hannes Lowette

AXXES

Hannes Lowette

- Father of Arne, Joren & Marit
- Fiancé of Barbara
- Principal Consultant @ Axxes
- .NET backend dev since 2005

- Linebreaker
- Microsoft MVP
- NServiceBus Champ

AXXES.

Agenda

1 _ Event Sourcing

2 _ CQRS Architecture

3 _ Marten

4 _ Patterns & Features

Event Sourcing

AXXES

What database engine is most popular?

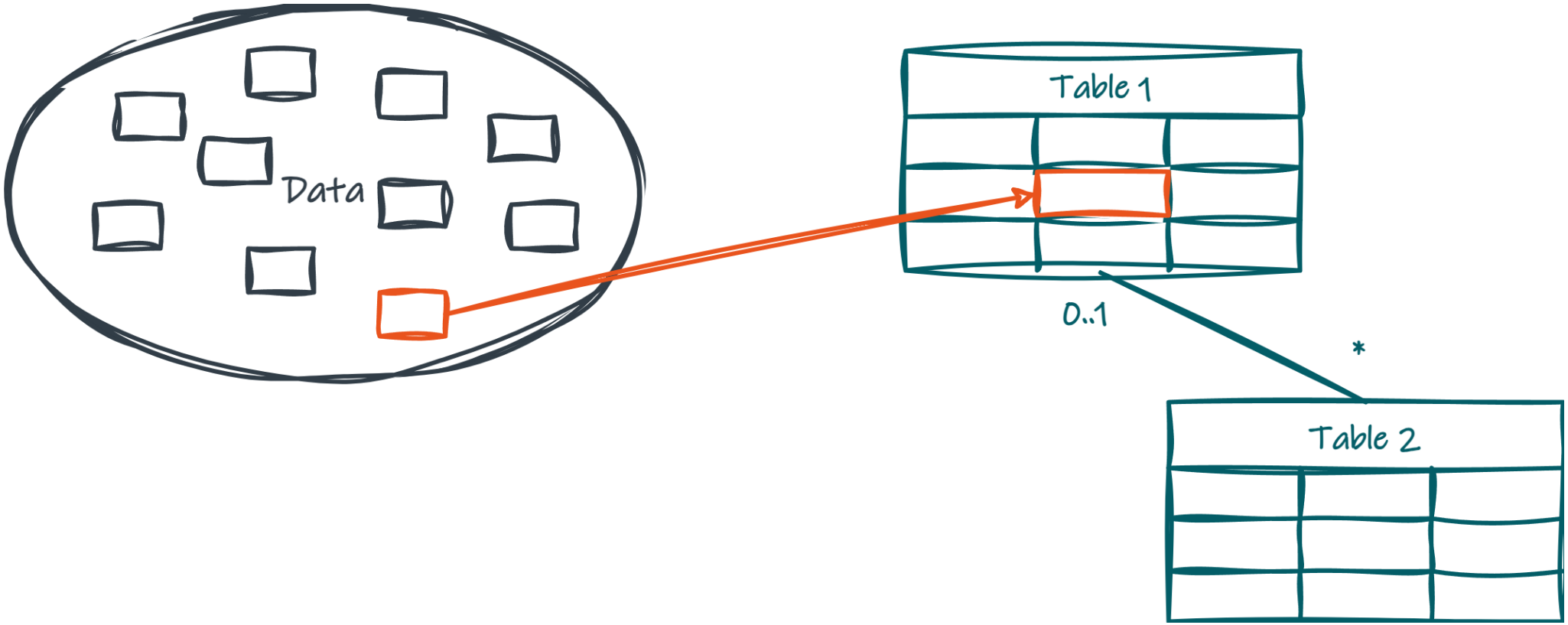
424 systems in ranking, September 2025

Rank Sep 2025	Rank Aug 2025	Rank Sep 2024	DBMS	Database Model	Score Sep 2025	Score Aug 2025	Score Sep 2024
1.	1.	1.	Oracle	Relational, Multi-model	1170.62	-50.08	-115.97
2.	2.	2.	MySQL	Relational, Multi-model	891.77	-23.69	-137.72
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	717.32	-36.84	-90.45
4.	4.	4.	PostgreSQL	Relational, Multi-model	657.17	-14.08	+12.81
5.	5.	5.	MongoDB	Document, Multi-model	380.50	-15.08	-29.74
6.	6.	↑ 7.	Snowflake	Relational	190.19	+11.29	+56.47
7.	7.	↓ 6.	Redis	Key-value, Multi-model	145.17	-2.02	-4.25
8.	8.	↑ 9.	IBM Db2	Relational, Multi-model	124.19	-3.12	+1.14
9.	9.	↑ 14.	Databricks	Multi-model	124.06	+8.25	+39.82
10.	10.	↓ 8.	Elasticsearch	Multi-model	118.26	+3.99	-10.53

AXXES

Source: https://db-engines.com/

Normalization



AXXES

~ $12k in 2025

(adjusted for inflation)

AXXES

~ 5 minutes

~ 5 seconds

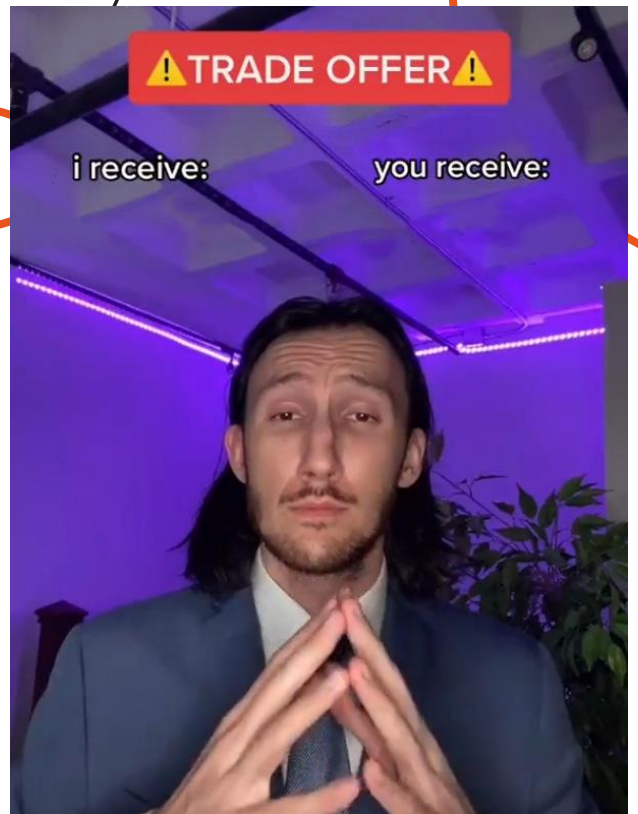
1/3 of the homepage

210 requests 9.6 MB transferred 22.2 MB resources Finish: 4.26 s

AXXES

What did we win?

- Data integrity
- CUD consistency
- Disk space

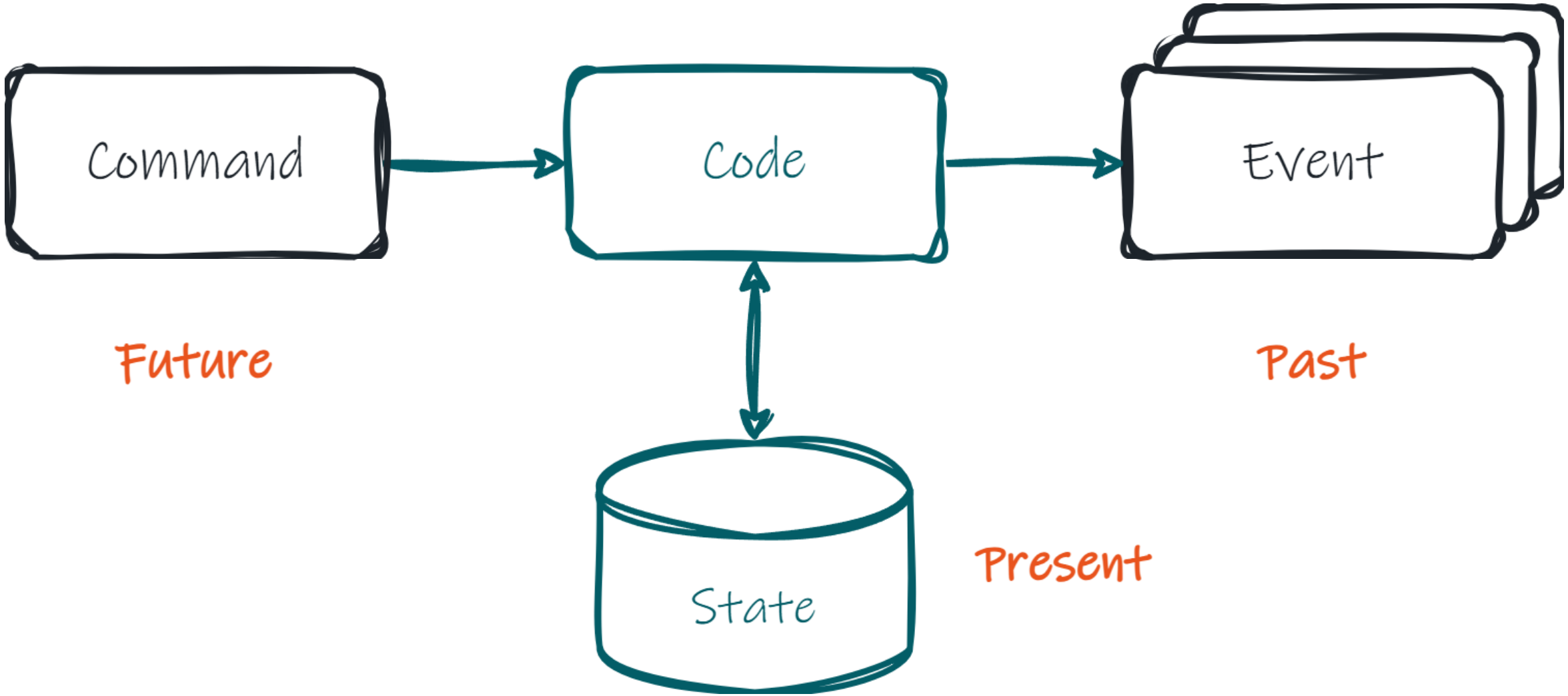


At what cost?

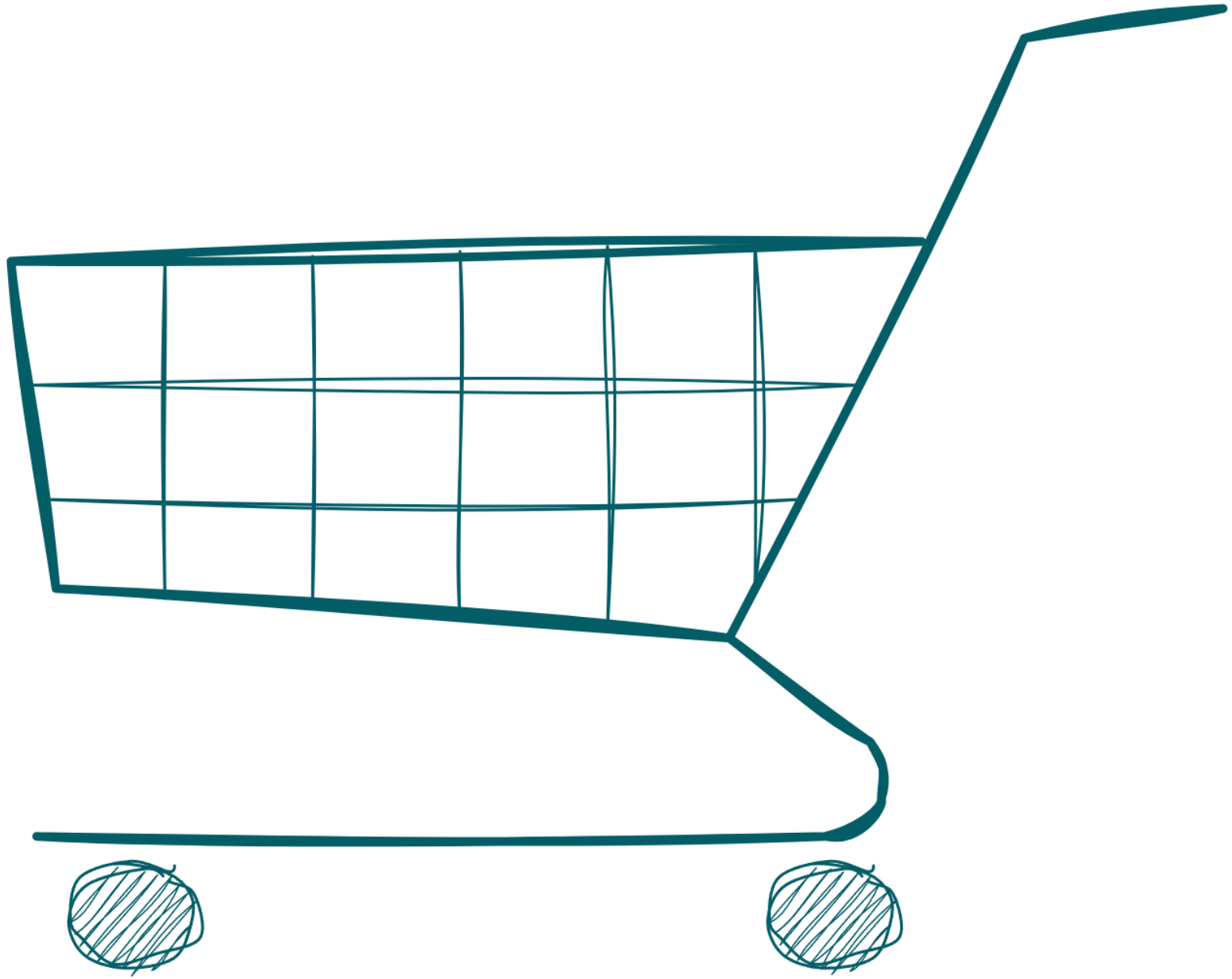
- Joins (performance)
- Database locks
- Loss of history

AXXES

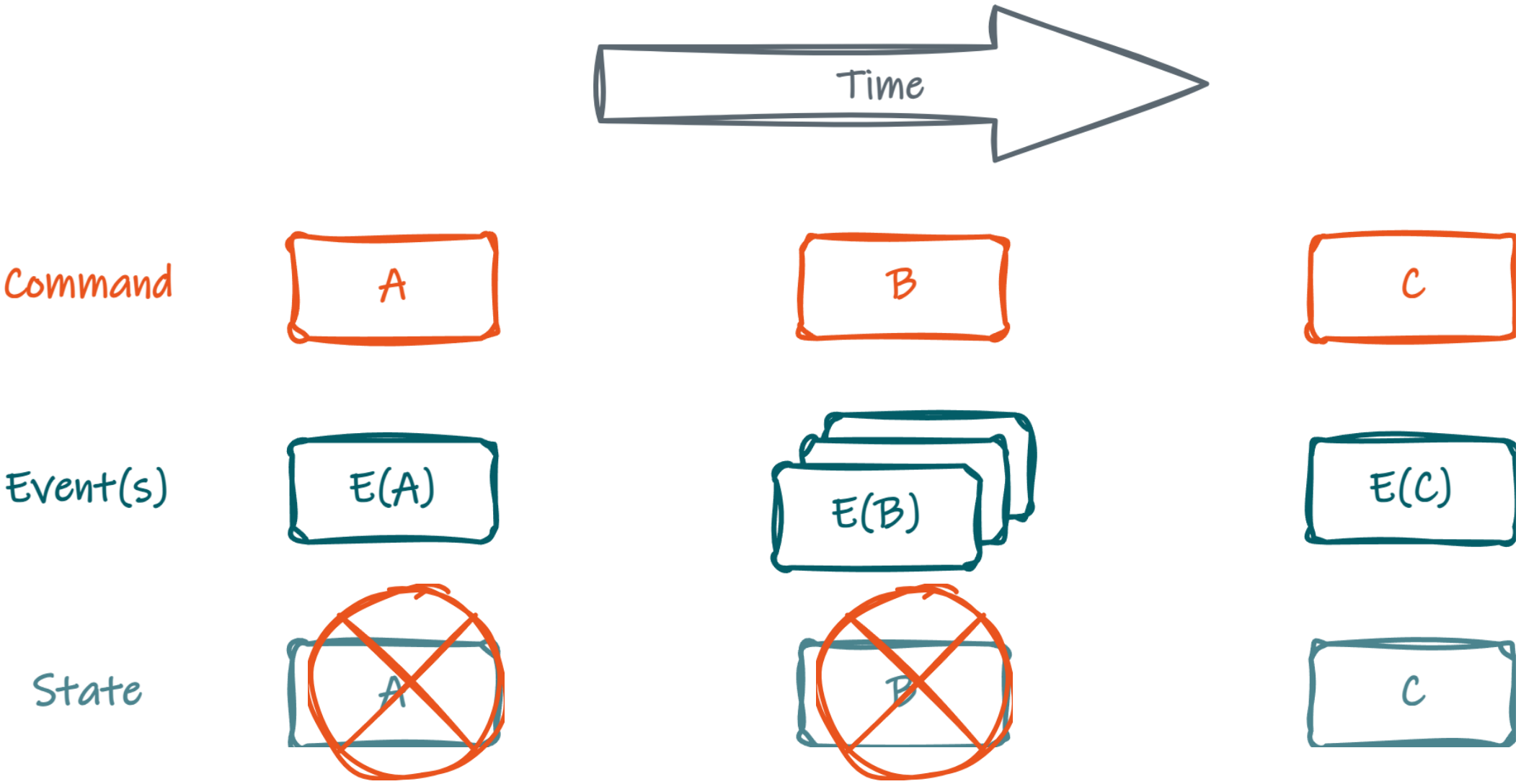

Event Sourcing

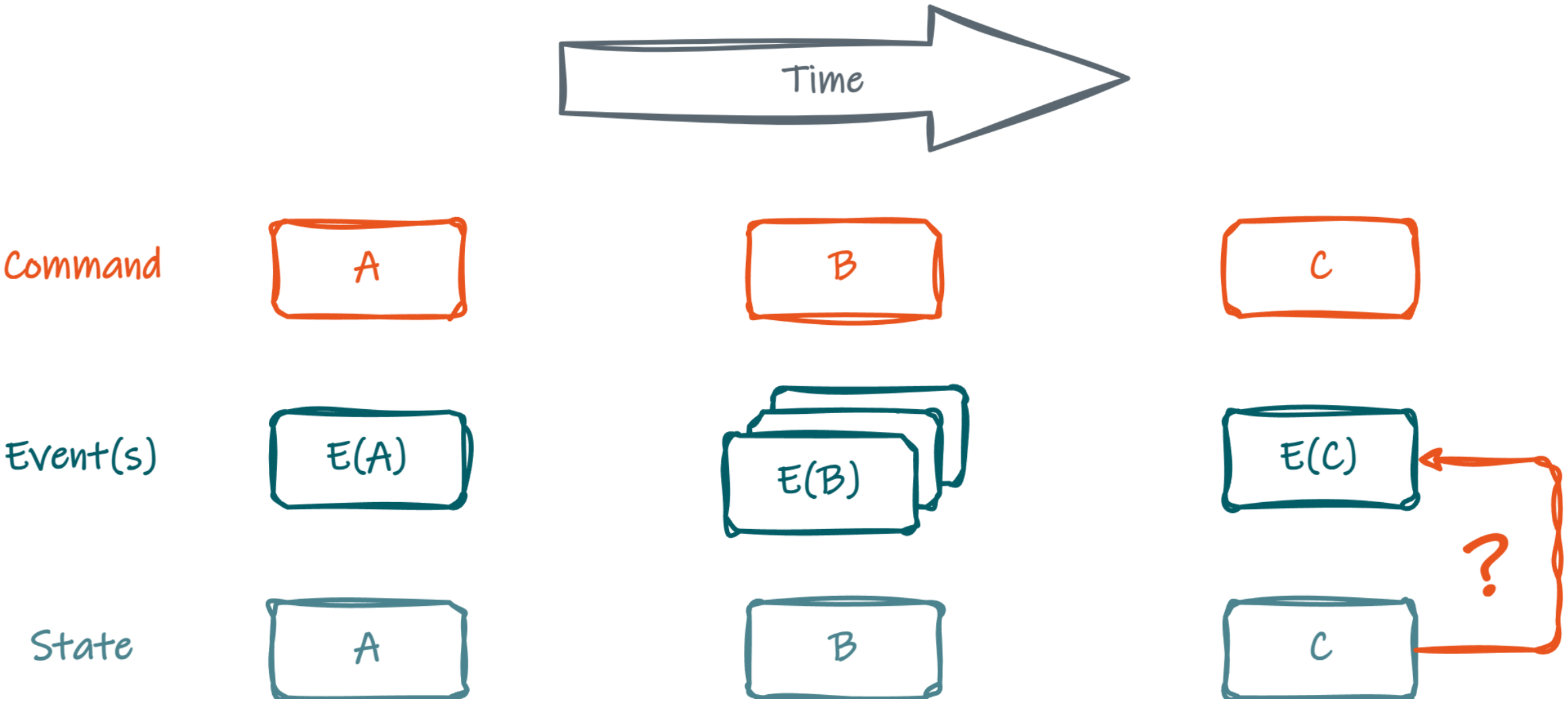


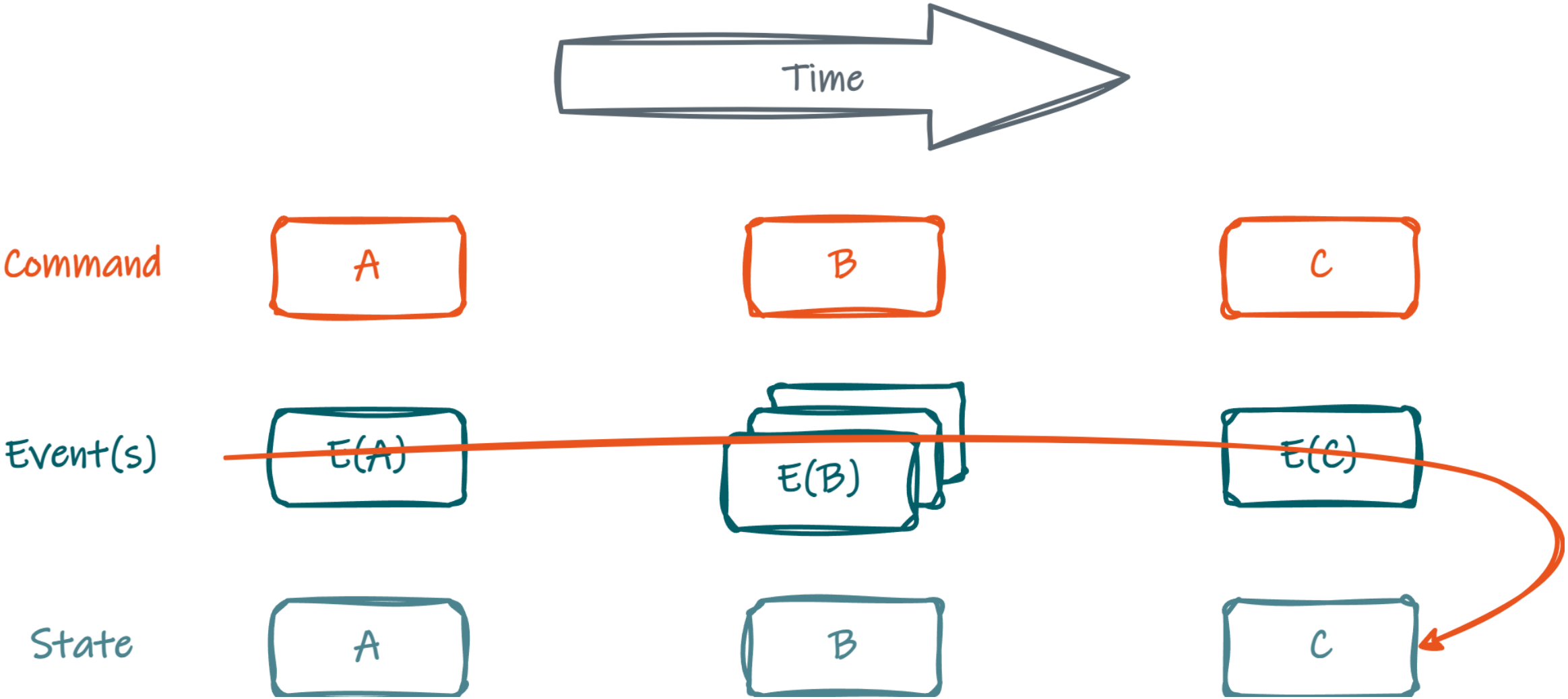
AXXES

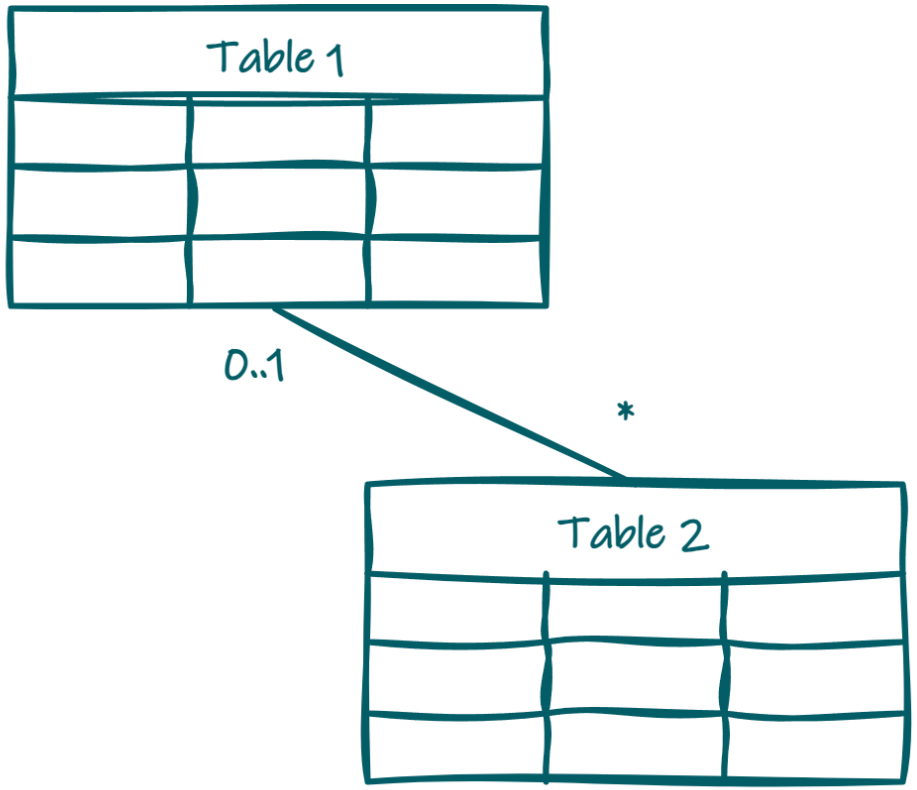


AXXES

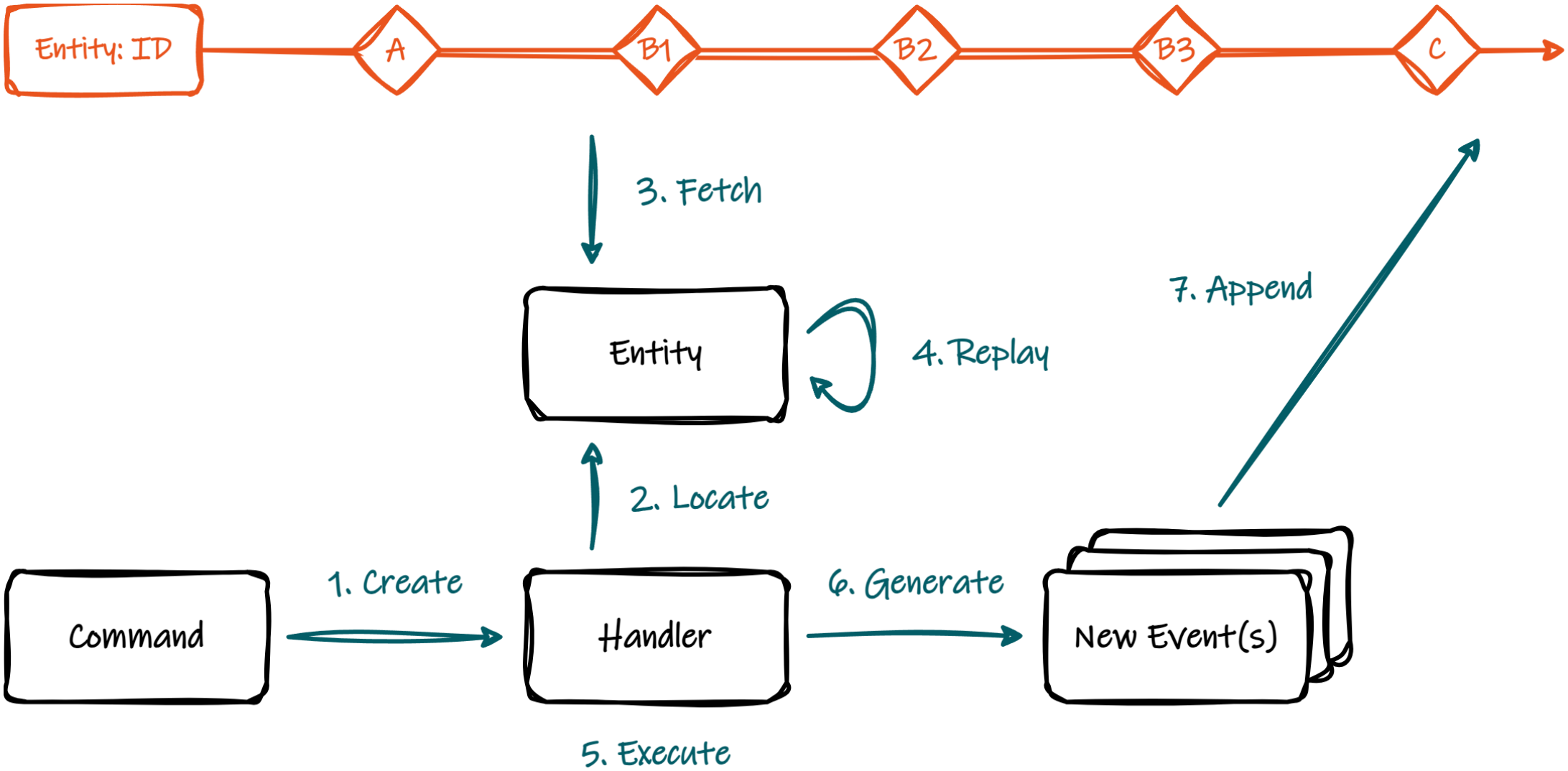

Time
Command
A
B
C
Event(s)
E(A)
E(B)
E(C)
State
A
B
C


Time
Command
A
B
C
Event(s)
E(A)
E(B)
E(C)
?
State
A
B
C

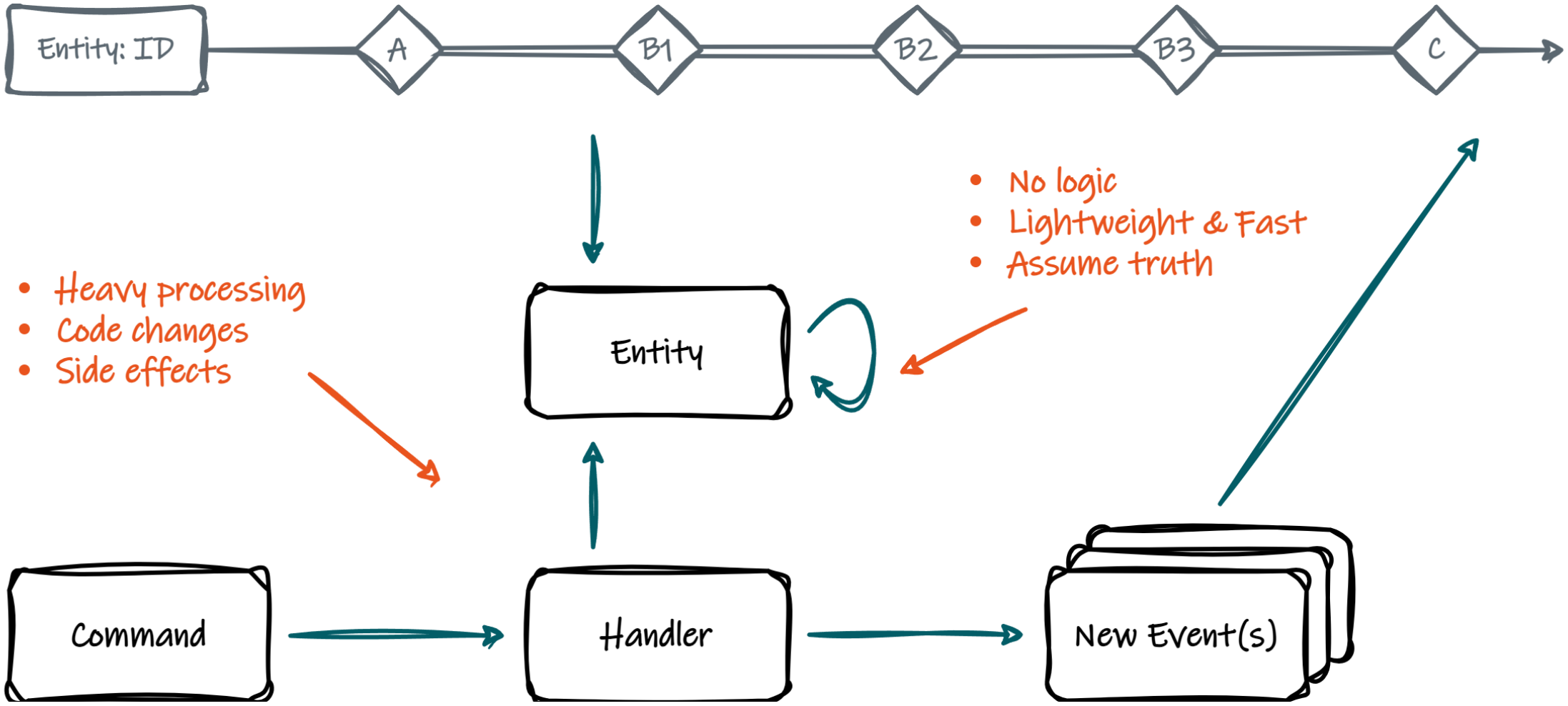

Time
Command
A
B
C
Event(s)
E(A)
E(B)
E(C)
State
A
B
C



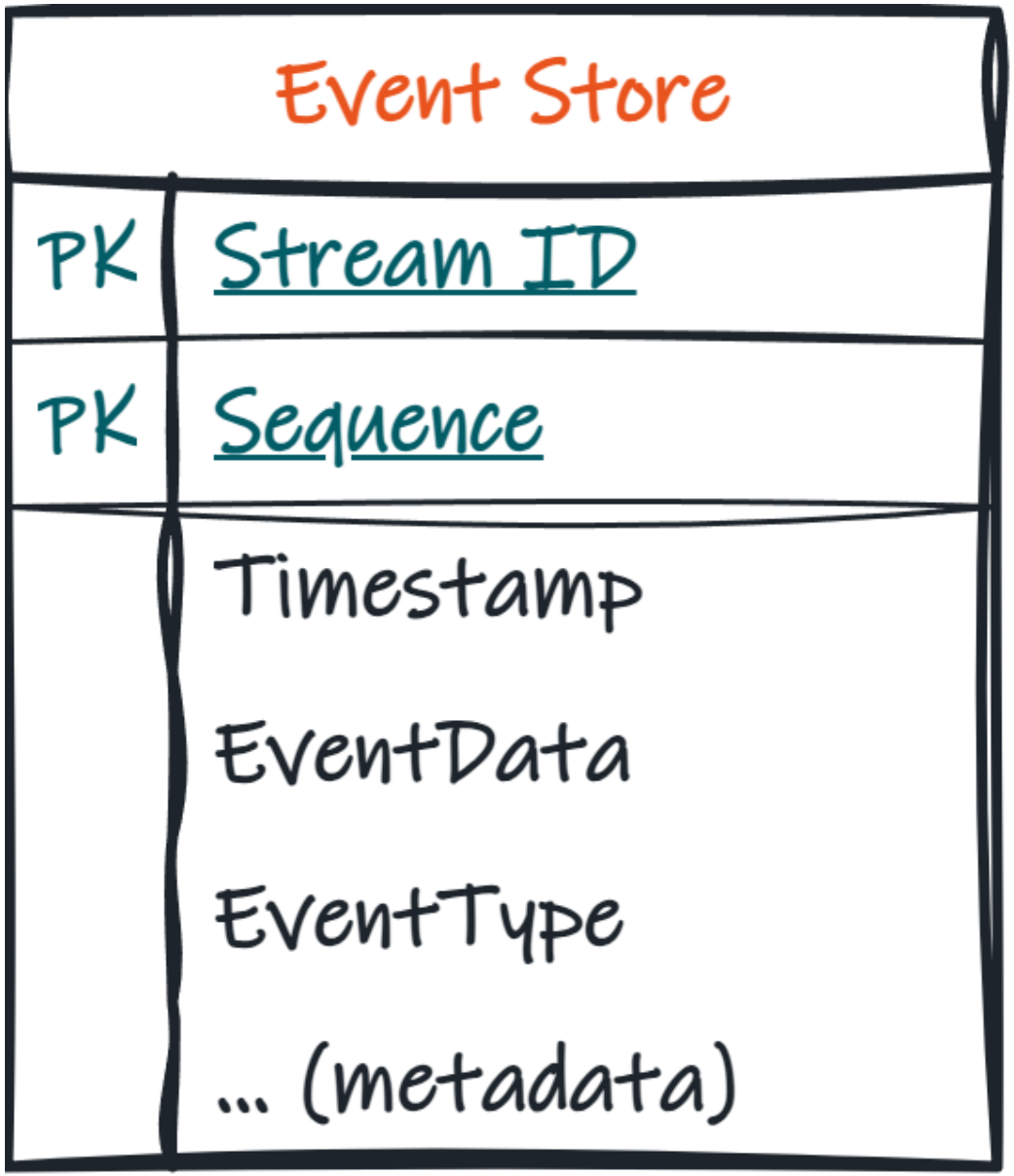
AXXES


Entity: ID
A
B1
B2
B3
C
3. Fetch
Entity
4. Replay
7. Append
2. Locate
Command
1. Create
Handler
6. Generate
New Event(s)
5. Execute

AXXES


Entity: ID
A
B1
B2
B3
C
Entity
Command
Handler
New Event(s)
Heavy processing
Code changes
Side effects
No logic
Lightweight & Fast
Assume truth

AXXES



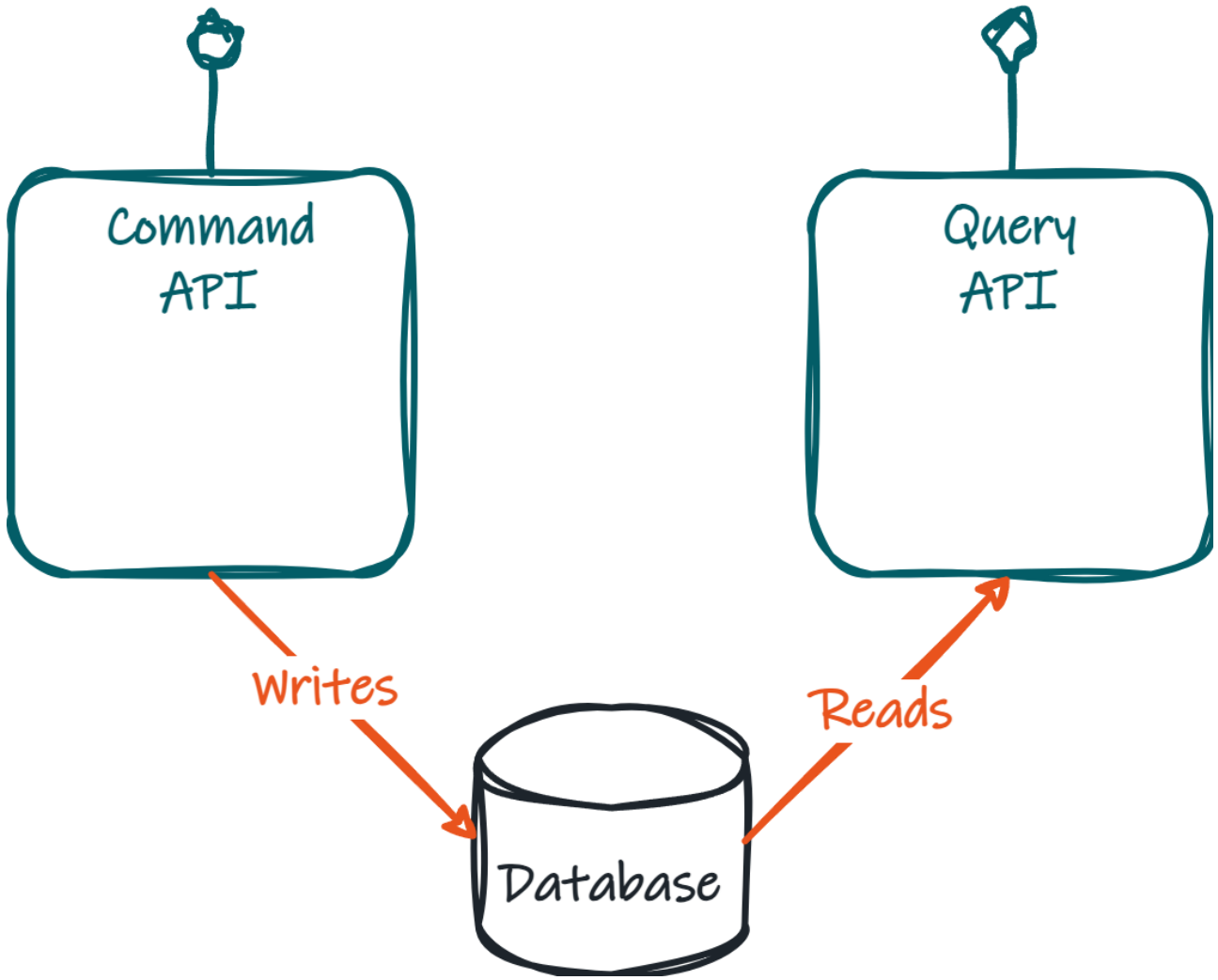
- Append only (no locks)
- No joins
- Full entity history intact

AXXES

CQRS Architecture

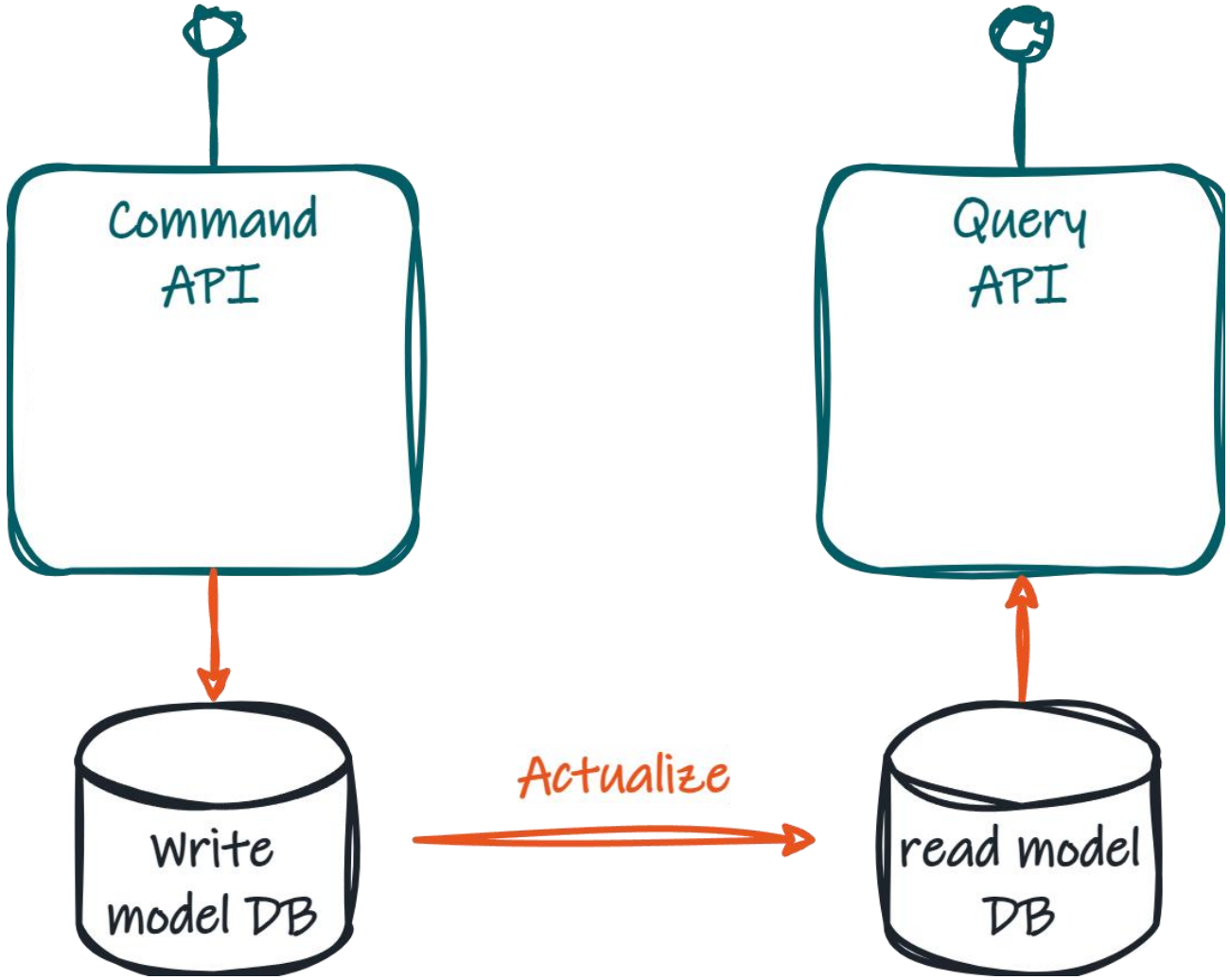
AXXES

CQS



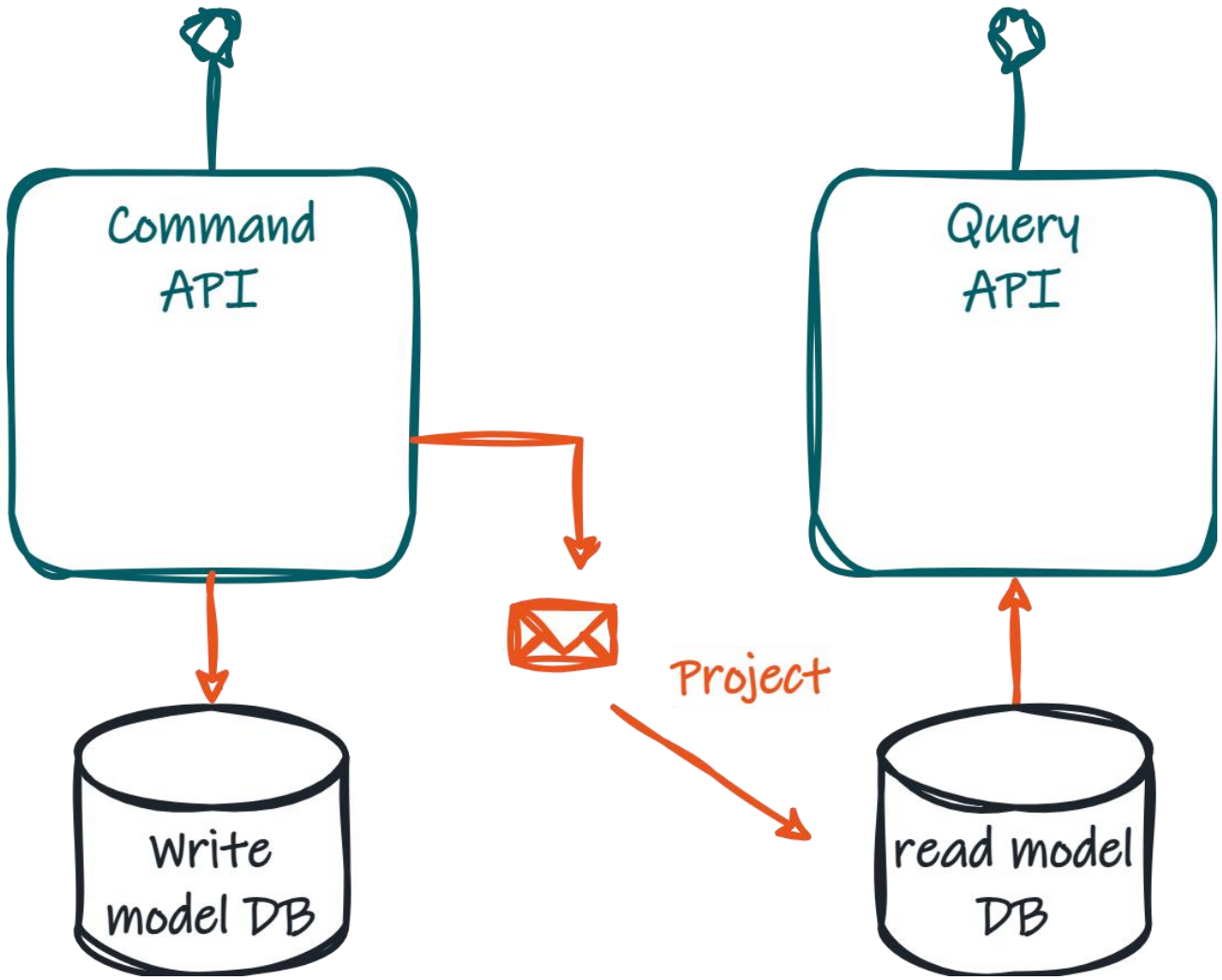
AXXES

CQRS


Command
API
Query
API
Write
model DB
Actualize
read model
DB

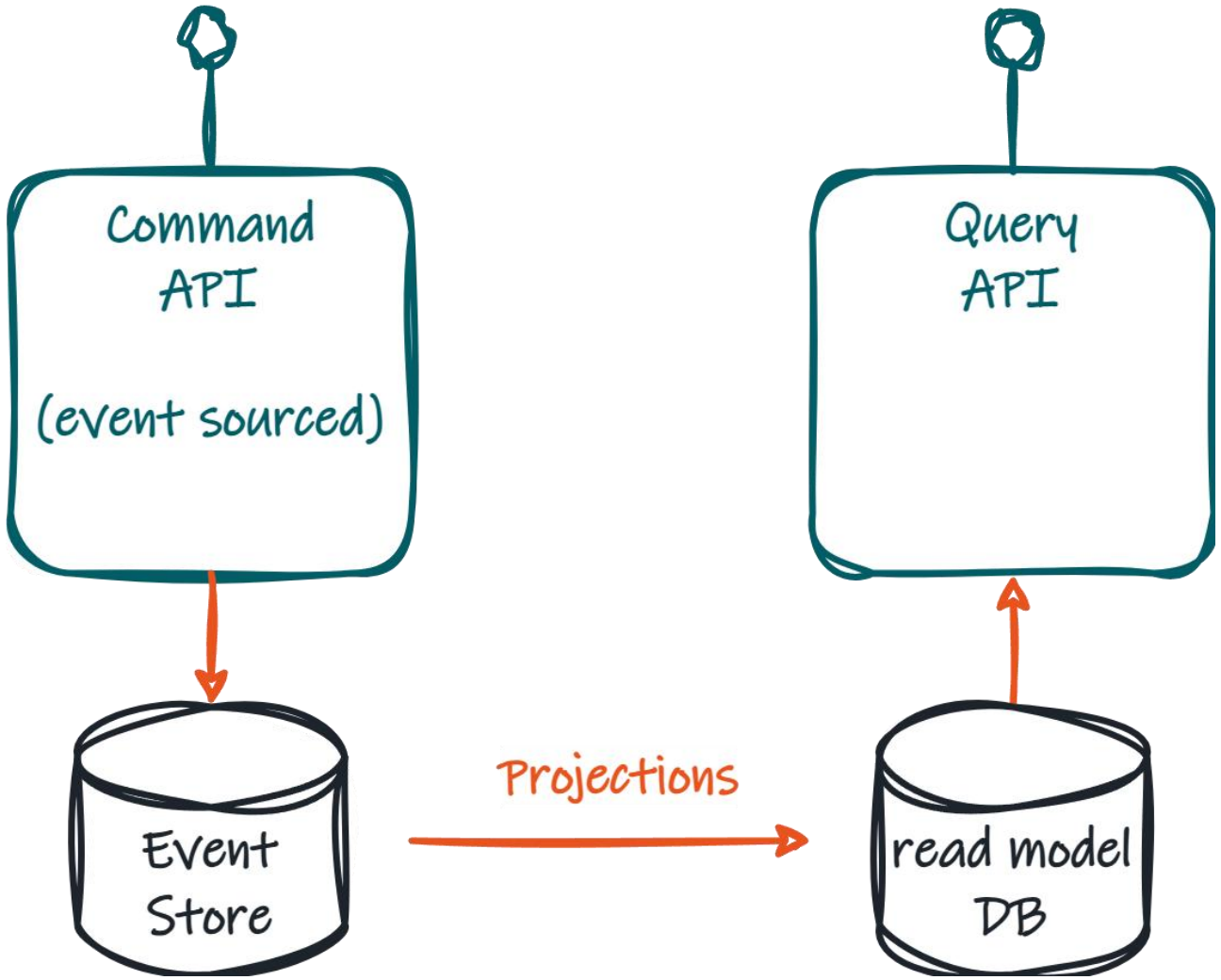
AXXES

CQRS

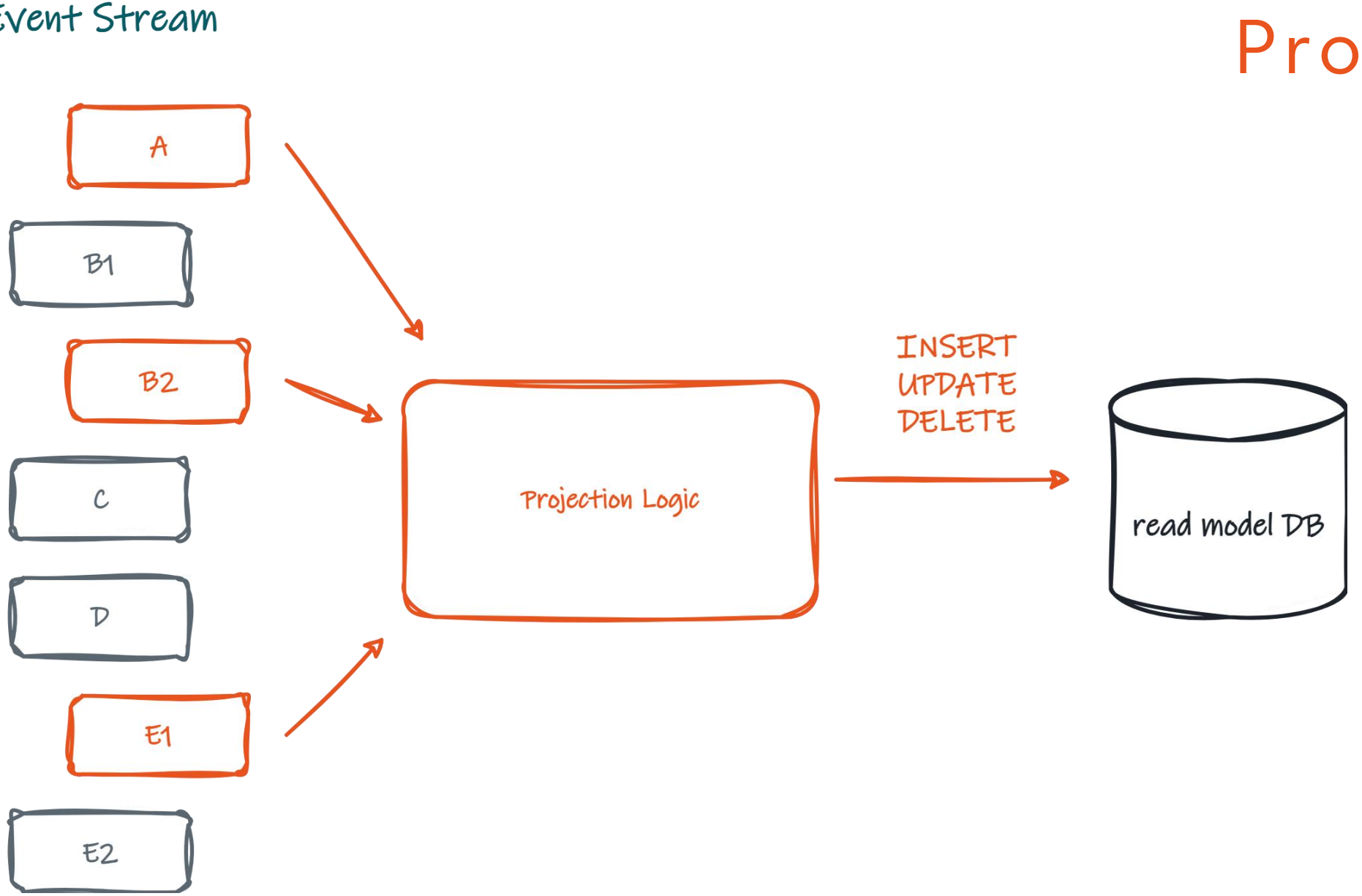


AXXES

CQRS


Command
API
(event sourced)
Query
API
Event
Store
Projections
read model
DB

Projections

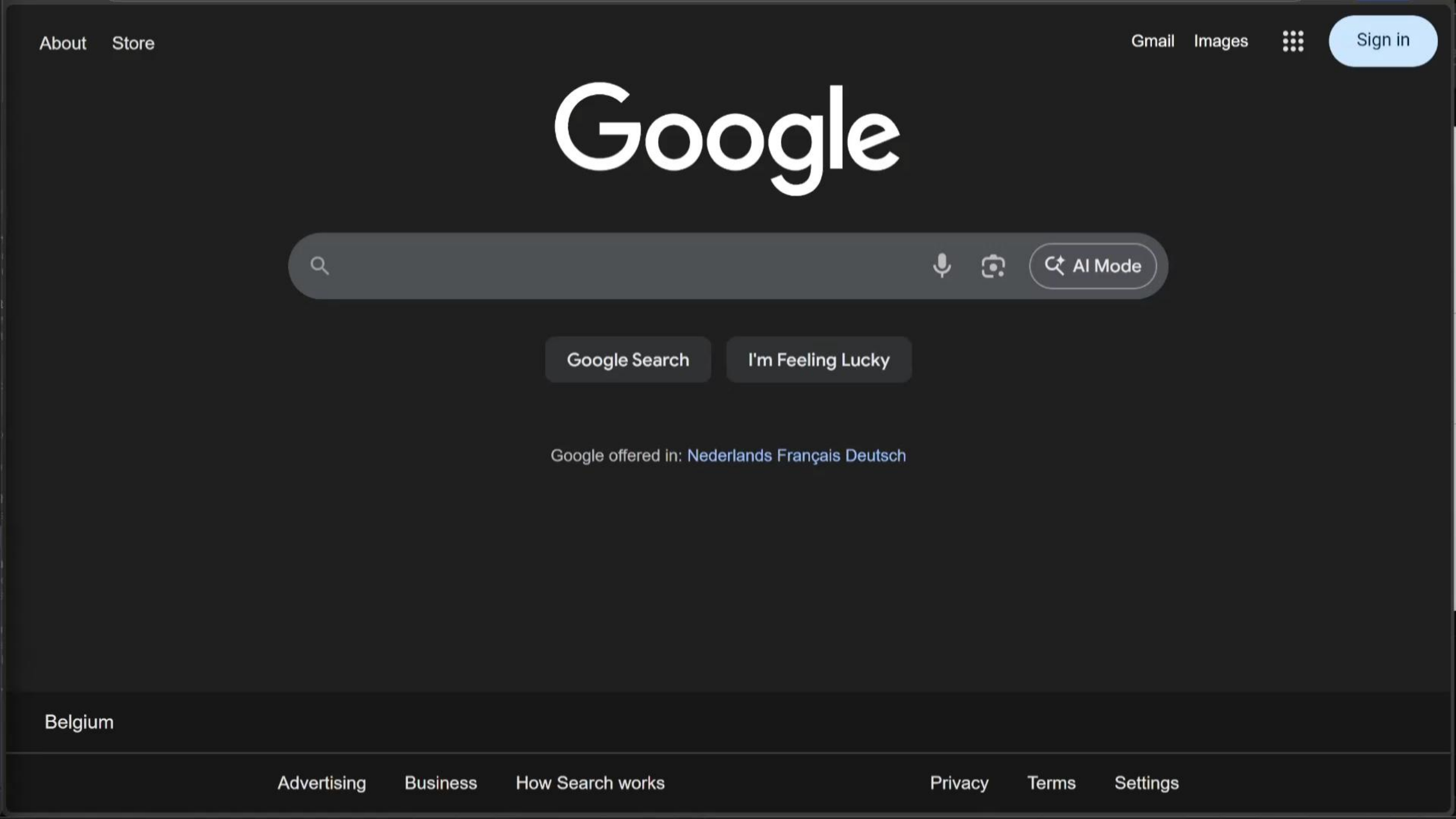


AXXES

Marten

AXXES

AXXES

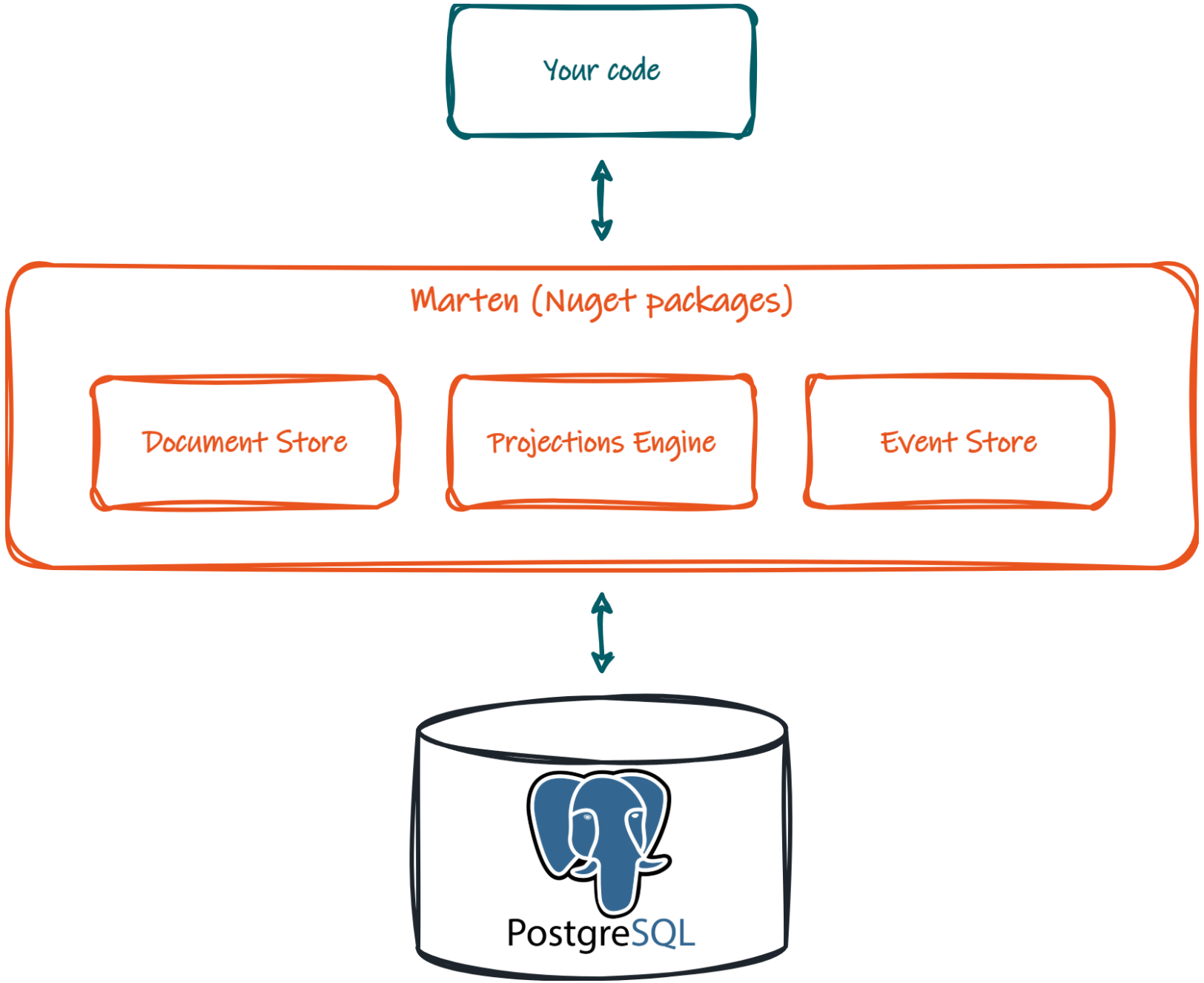

About
Store
Gmail
Images
Sign in
Google
AI Mode
Google Search
I'm Feeling Lucky
Google offered in: Nederlands Français Deutsch
Belgium
Advertising
Business
How Search works
Privacy
Terms
Settings

Up to 53 cm
~ 1.5 kg

63 cm average
Up to 2 kg
150 cm wingspan

Martens and fishers: Primarily target raven nests and eggs.

AXXES


Your code
Marten (Nuget packages)
Document Store
Projections Engine
Event Store
PostgreSQL

AXXES

In a nutshell

Leverages Postgres:
- ACID compliant
- JSONB for fast document performance
- Multi-node sync

Features:
- Document Store
- Multi-tenant
- Event Store & Projections engine
- Production ready
- Integrates with Wolverine (critter stack)

https://martendb.io/

AXXES

Free & open source (with support plans)

Basic

Individuals and small teams

- ✓ Response within 48 hours
- ✓ Up to 2 critical incidents *
- ✓ Private Discord channel

$ 3000 / year

Contact Us

Standard

Growing businesses

- ✓ Response within 36 hours
- ✓ Up to 5 critical incidents *
- ✓ Up to 2 non-critical incidents **
- ✓ Private Discord channel
- ✓ Private issue tracking board

$ 6000 / year

Contact Us

Premium

Larger companies with mission critical projects

- ✓ Response within 24 hours
- ✓ Unlimited critical incidents *
- ✓ Up to 4 non-critical incidents **
- ✓ Private Discord channel
- ✓ Private issue tracking board
- ✓ Email support

$ 15000 / year

Contact Us

* Software defects in the supported libraries which severely affect customer production system.

** Technical questions and expert advice.

https://jasperfx.net/support-plans/

Marten

nuget.org

Version: Latest stable 8.16.1 | Install

Package source mapping is off. Configure

Options

Description

.NET Transactional Document DB and Event Store on PostgreSQL

Version:	8.16.1
Owner(s):	jeremydmiller, mysticmind, oskar.dudycz
Author(s):	Jeremy D. Miller,Babu Annamalai,Jaedyn Tonee
License:	MIT
Downloads:	13,123,453
Date published:	Thursday, 20 November 2025 (20/11/2025)
Project URL:	https://martendb.io/
Report Abuse:	https://www.nuget.org/packages/Marten/8.16.1/ReportAbuse

Dependencies

- net8.0
- net9.0
- net10.0
  - FSharp.Core (>= 9.0.100)
  - JasperFx (>= 1.10.1)
  - JasperFx.Events (>= 1.12.1)
  - JasperFx.RuntimeCompiler (>= 4.2.0)
  - Microsoft.Extensions.Hosting.Abstractions (>= 10.0.0)
  - Newtonsoft.Json (>= 13.0.3)
  - Npgsql.Json.NET (>= 9.0.2)
  - Polly.Core (>= 8.5.2)
  - Weasel.Postgresql (>= 8.3.0)

Registration

```
// Register Marten
builder.Services.AddMarten(options =>
    {
        // Give it a connection string
        var connStr = builder.Configuration.GetConnectionString("MartenDB")!;
        options.Connection(connStr);

        // Optional, defaults to "public"
        options.DatabaseSchemaName = "martendemo";

        // Further Marten config goes here
    })
    // Pick a default session type
    .UseLightweightSessions()
    // Start an asynchronous projection daemon
    .AddAsyncDaemon(DaemonMode.Solo);
```

AXXES

Commands

```
record LogTicket(Guid TicketId, string Title, string Description);

record ProposeResolution(Guid TicketId, string Comment);

record SelectResolution(Guid TicketId, string Resolution);

record CloseTicket(Guid TicketId);
```

Events

```
record TicketLogged(string Title, string Description);

record ResolutionProposed(string Comment);

record ResolutionSelected(string Resolution);

record TicketClosed;
```

Root Entity

```
record Ticket(
    Guid Id,
    string Title,
    string Description,
    string[] Comments,
    string? Resolution,
    bool IsClosed)
{
    public static Ticket Create(IEvent<LogTicket> logged)
        => new(logged.StreamId, logged.Data.Title, logged.Data.Description, [], null, false);

    public static Ticket Apply(ResolutionProposed proposed, Ticket previous)
        => previous with { Comments = previous.Comments.Append(proposed.Comment).ToArray() };

    public static Ticket Apply(ResolutionSelected selected, Ticket previous)
        => previous with { Resolution = selected.Resolution };

    public static Ticket Apply(TicketClosed closed, Ticket previous)
        => previous with { IsClosed = true };
}
```

AXXES

Start a stream

```
[HttpPost("log")]
public async Task<IActionResult> LogTicket(
    [FromServices] IDocumentSession session,
    [FromBody] LogTicket command)
{
    var startEvent = new TicketLogged(command.Title, command.Description);

    session.Events.StartStream<Ticket>(command.TicketId, startEvent);

    await session.SaveChangesAsync();

    return Ok();
}
```

AXXES

Use an existing stream

```
[HttpPost("close")]
public async Task<IActionResult> CloseTicket(
    [FromServices] IDocumentSession session,
    [FromBody] CloseTicket command)
{
    // Fetches all events and projects into a Ticket entity
    var stream = await session.Events.FetchForWriting<Ticket>(command.TicketId);
    var ticket = stream.Aggregate;

    // Check the command's validity and append new events accordingly
    if (ticket?.Resolution is not null)
        stream.AppendOne(new TicketClosed());

    await session.SaveChangesAsync();

    return Ok();
}
```

AXXES

Make a new projection

```
record OpenTicket(Guid Id, string Title, bool Resolved);

class OpenTicketProjection : SingleStreamProjection<OpenTicket, Guid>
{
    public OpenTicketProjection()
    {
        DeleteEvent<TicketClosed>();
    }

    public OpenTicket Create(IEvent<LogTicket> logged)
        => new(logged.StreamId, logged.Data.Title, false);

    public OpenTicket Apply(ResolutionSelected resolved, OpenTicket previous)
        => previous with { Resolved = true };
}
```

AXXES

Register a projection

```
options.Projections
    .Add<OpenTicketProjection>(ProjectionLifecycle.Async);
```

Async required Async Daemon to be active
(runs in a background worker)

AXXES

Tuning projections

Immediate

- 100% consistent
- No delays

Eventual

- Pre-processed
- No slow-down of write

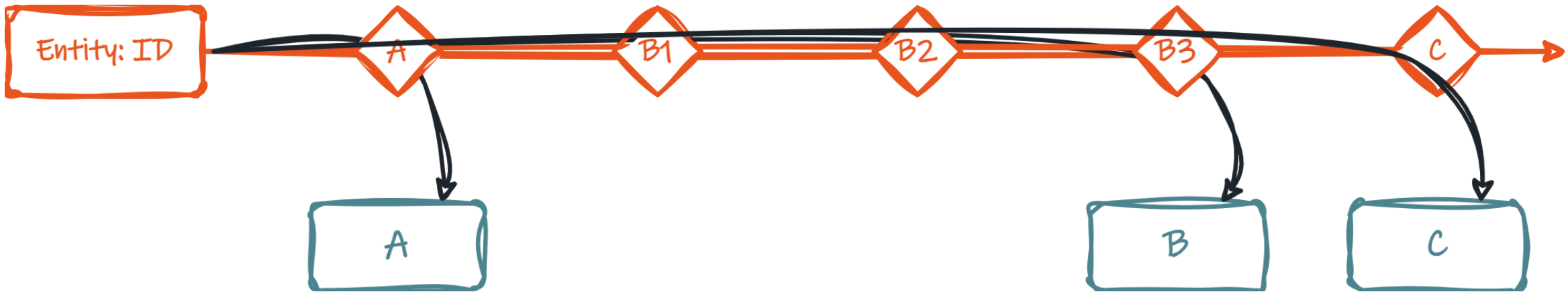
On-demand

- Occasional queries
- No wasted processing

Patterns & Features

AXXES

Point-in-time recovery



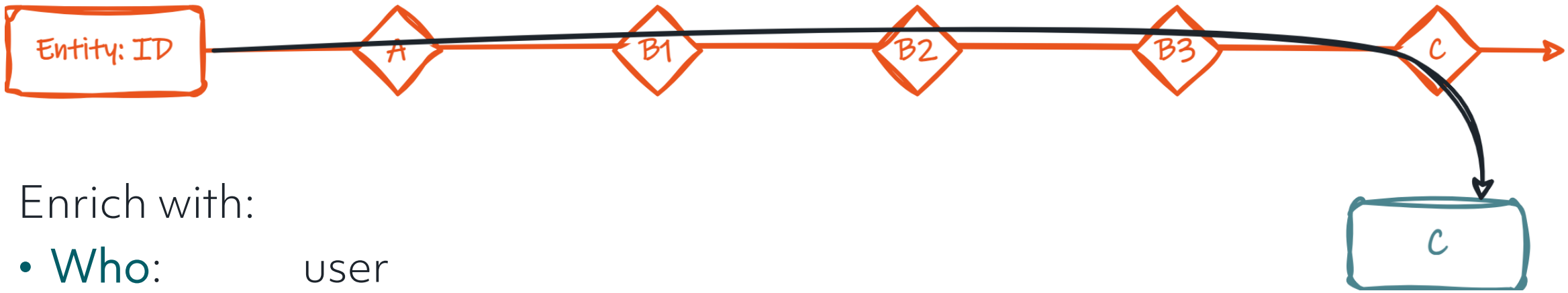
AXXES

Point-in-time recovery

```
[HttpGet("{id}/version/{version}")]
public async Task<IActionResult> GetTicket(
    [FromServices] IDocumentSession session, Guid id, long version)
{
    var ticket = await session.Events.AggregateStreamAsync<Ticket>(id, version: version);

    return ticket == null
        ? NotFound()
        : Ok(ticket);
}

[HttpGet("{id}/timestamp/{timestamp}")]
public async Task<IActionResult> GetTicket(
    [FromServices] IDocumentSession session, Guid id, DateTimeOffset timestamp)
{
    var ticket = await session.Events.AggregateStreamAsync<Ticket>(id, timestamp: timestamp);

    return ticket == null
        ? NotFound()
        : Ok(ticket);
}
```

AXXES

Audit Log?



Enrich with:

- Who: user
- Why: command
- When: timestamp
- Metadata: correlation etc.

→ Command log

AXXES

Versioning

Backwards compatibility

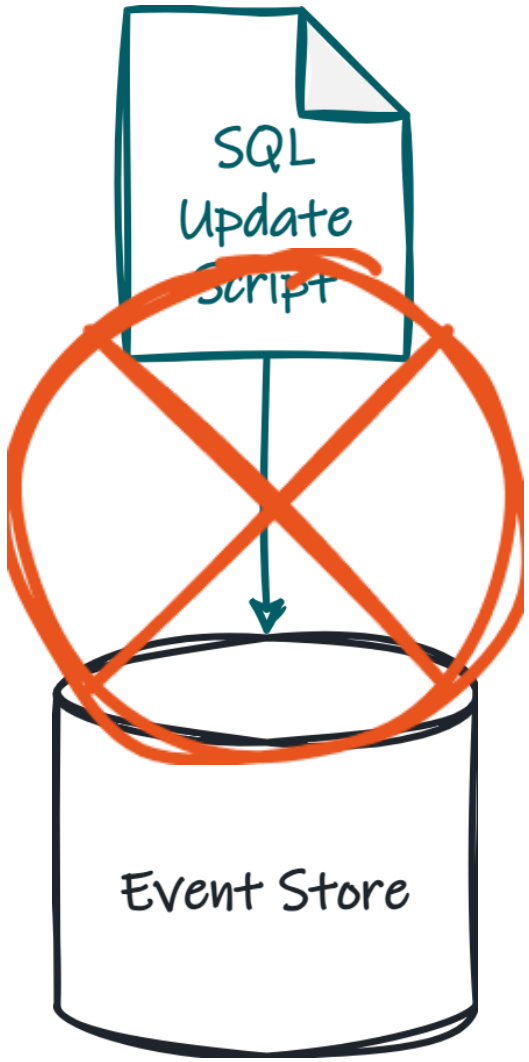
- Deserialize old versions correctly
- Default values

Split into multiple events

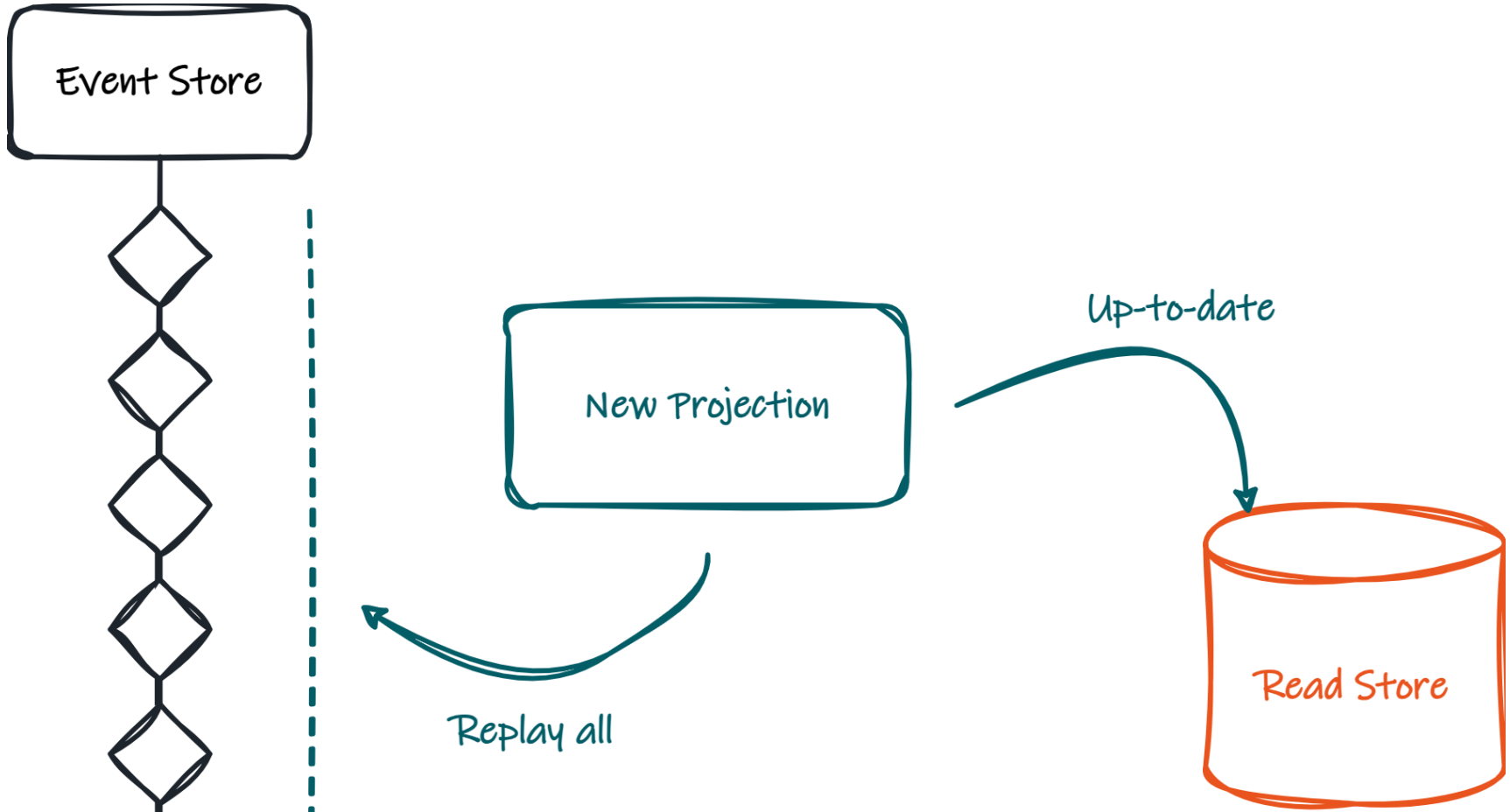
- If this makes sense in the domain

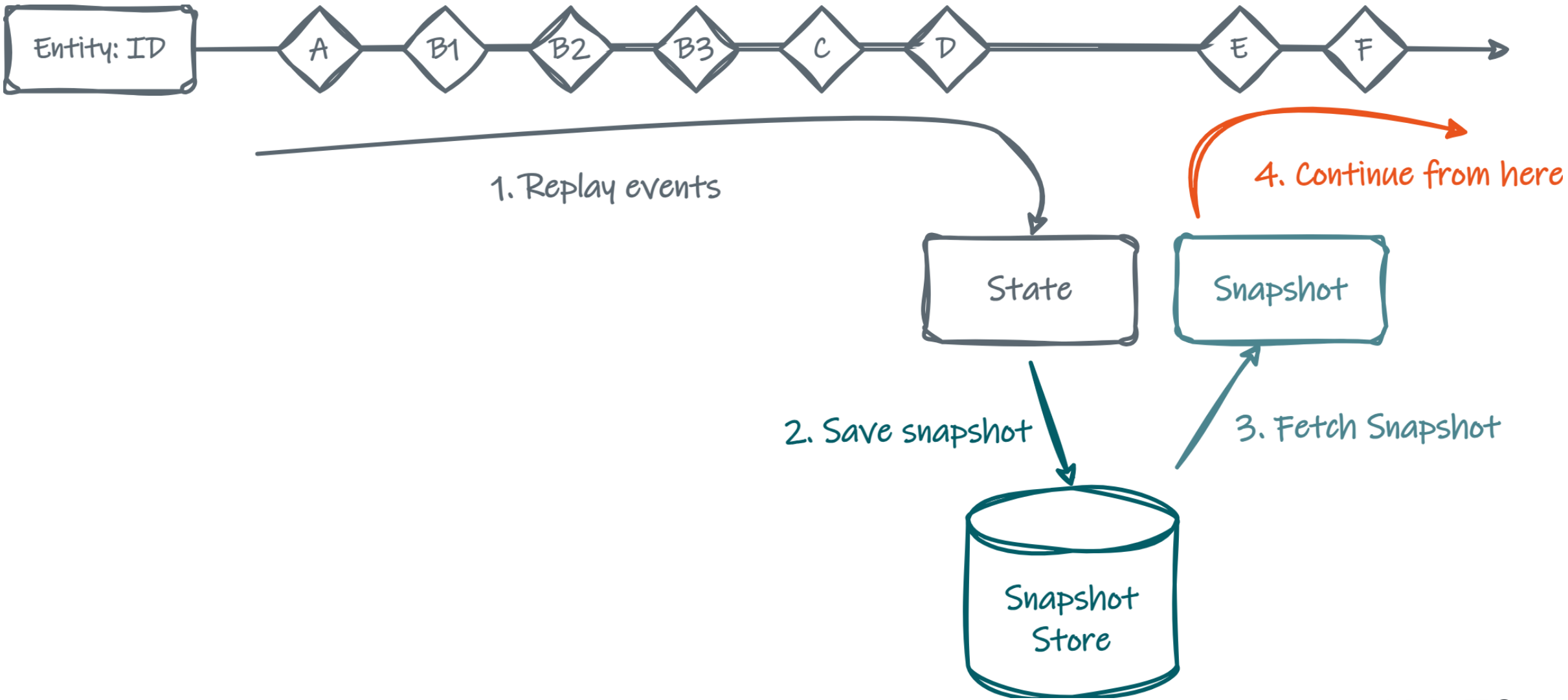
Different versions

- Upcast old events
- Let aggregates handle both

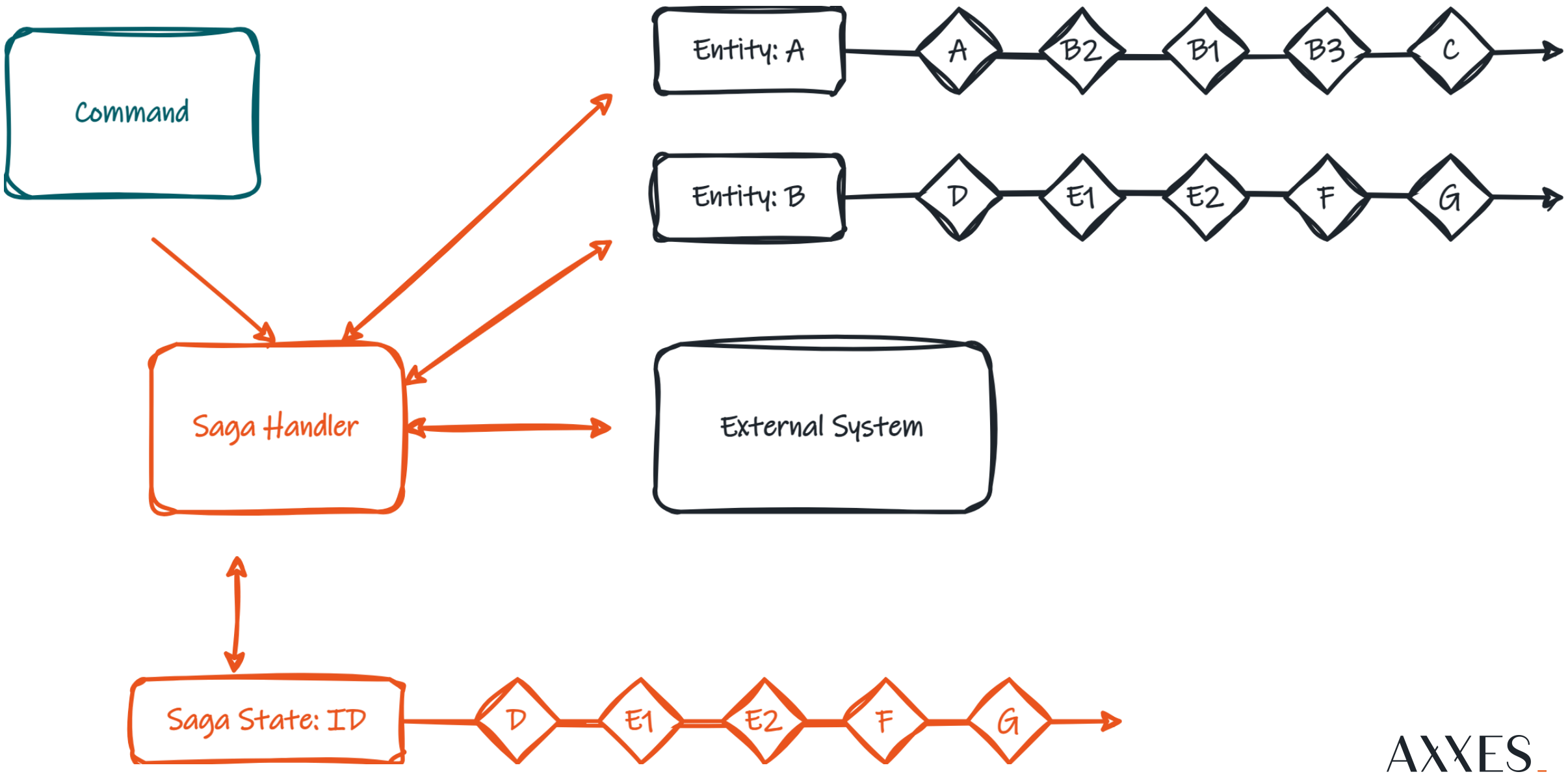


AXXES

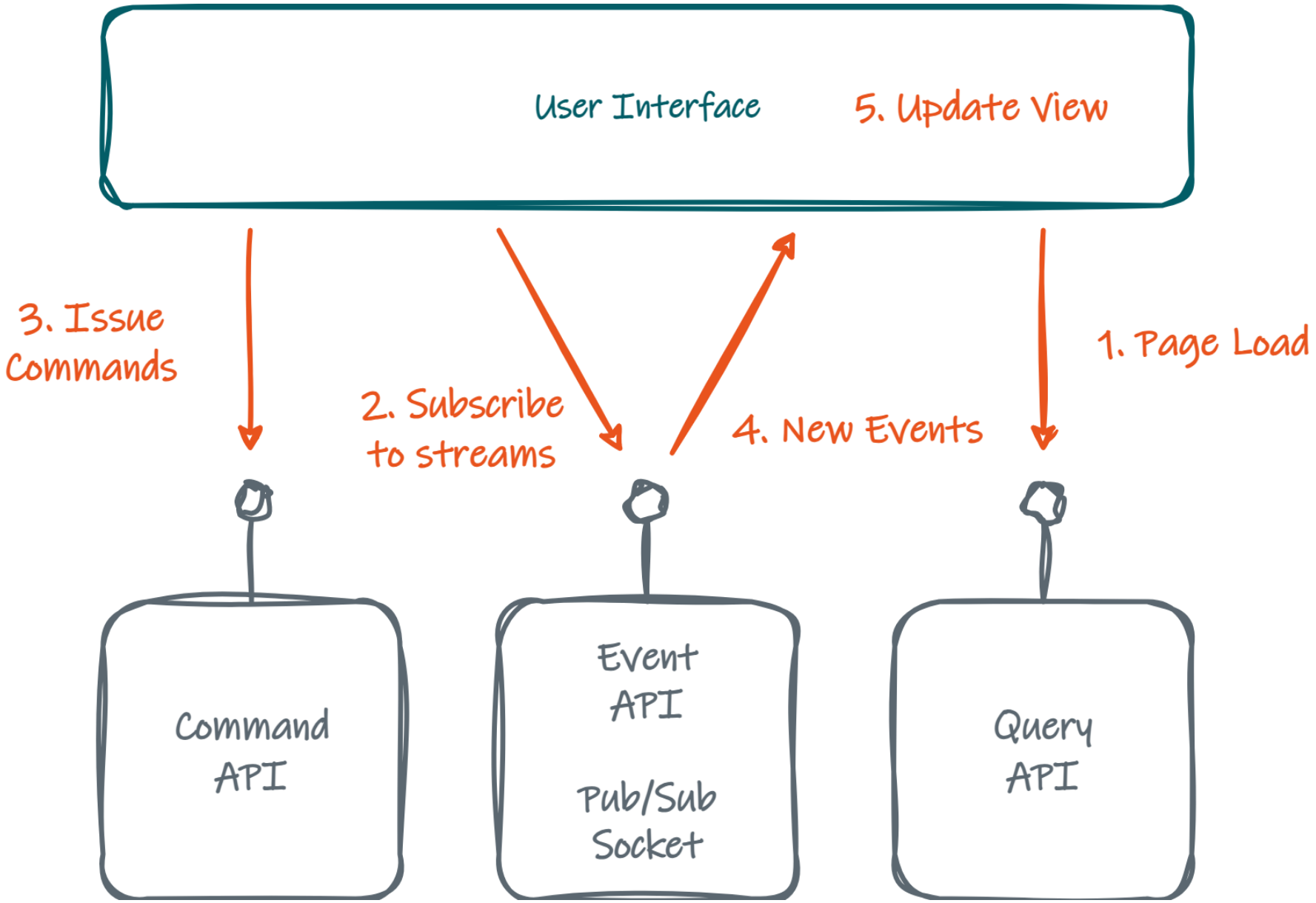

Event Store
New Projection
Up-to-date
Read Store
Replay all


Entity: ID
A
B1
B2
B3
C
D
E
F
1. Replay events
4. Continue from here
State
Snapshot
2. Save snapshot
3. Fetch Snapshot
Snapshot Store

AXXES


Command
Entity: A
A
B2
B1
B3
C
Entity: B
D
E1
E2
F
G
Saga Handler
External System
Saga State: ID
D
E1
E2
F
G

AXXES


User Interface
5. Update View
3. Issue Commands
2. Subscribe to streams
4. New Events
1. Page Load
Command API
Event API
Pub/Sub Socket
Query API

AXXES

Further reading

- Marten documentation: https://martendb.io/events/

- Critter Stack Discord: https://discord.com/invite/WMxrvegf8H
  (please don't spam questions before researching)

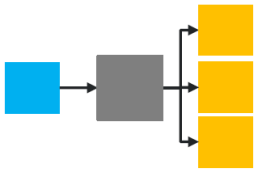
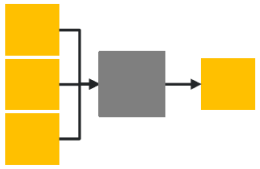
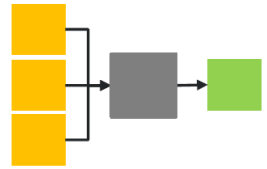
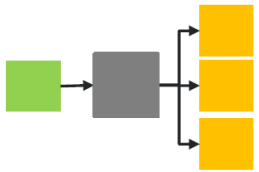
- Jeremy's blog: https://jeremydmiller.com/
- Oskar's blog: https://event-driven.io/en/
- Yves Goeleven: https://www.linkedin.com/in/goeleven/
- Greg Young's CQRS paper:
  https://cqrs.wordpress.com/wp-content/uploads/2010/11/cqrs_documents.pdf

AXXES

Yves Goeleven: https://www.goeleven.com/talks/techorama-2023-be/fantastic-9.pdf

My fantastic 9

Feel free to steal them

In\Out	Command	Event	State
Command	Delegation 	Aggregate Root 	Downstream Activity 
Event	Reaction 	Event Stream Processing 	Projection 
State	Task processing 	Event Generator 	State Transformation 

When would you use Event Sourcing?

AXXES

Thank you!

@hanneslowette.net

linkedin.com/in/hanneslowette

dometrain.com/author/hannes-lowette/

AXXES