How to avoid **screwing up** your business application

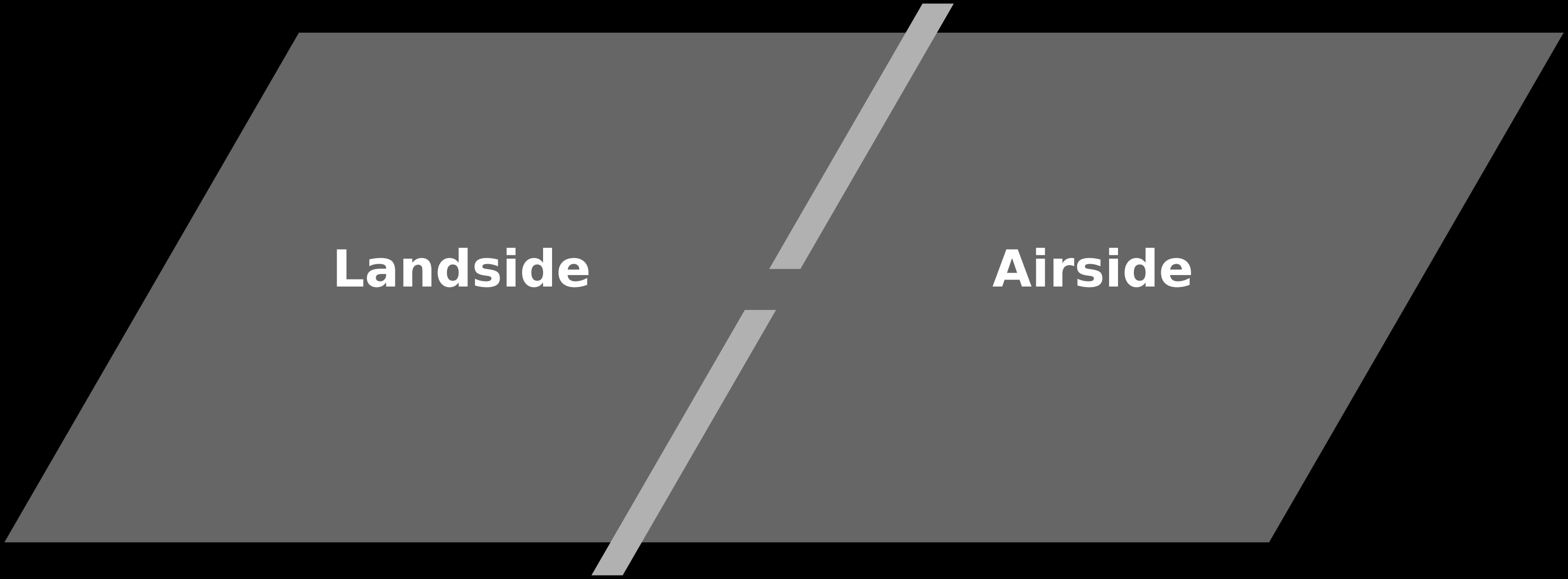
masak

YAPC::EU 2011

airports

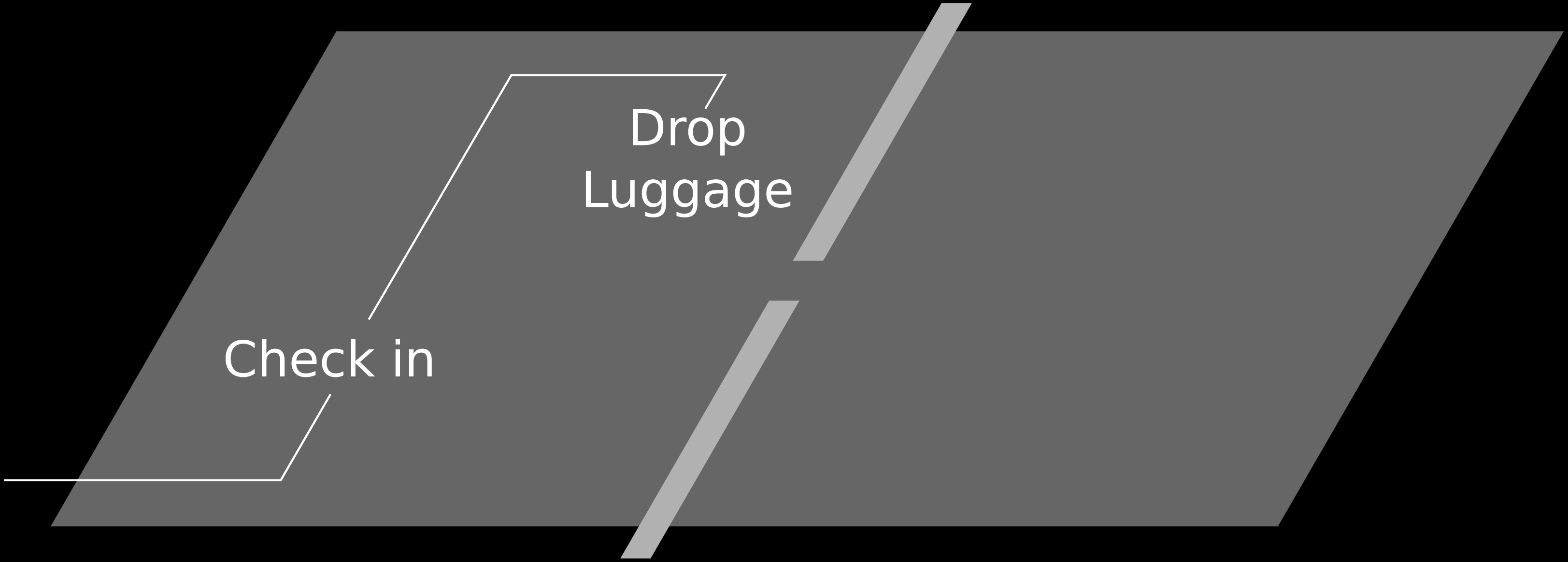
(hi, I'm masak)

Airport

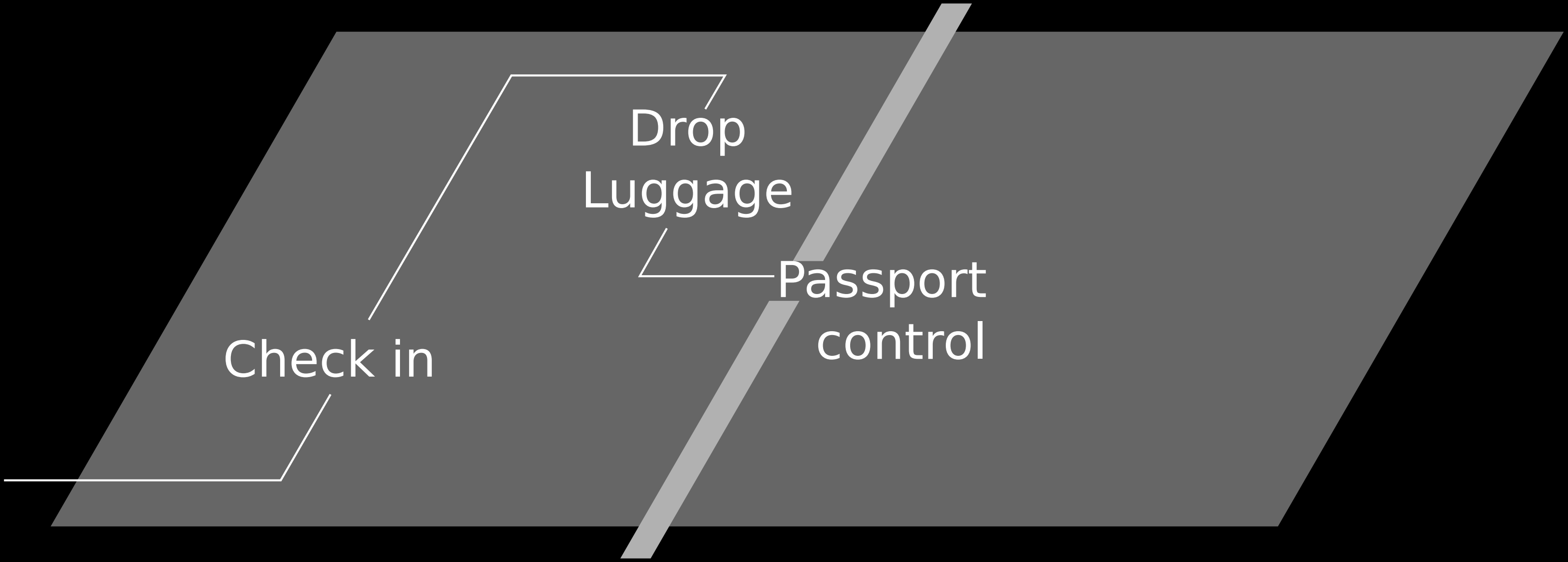


Airport

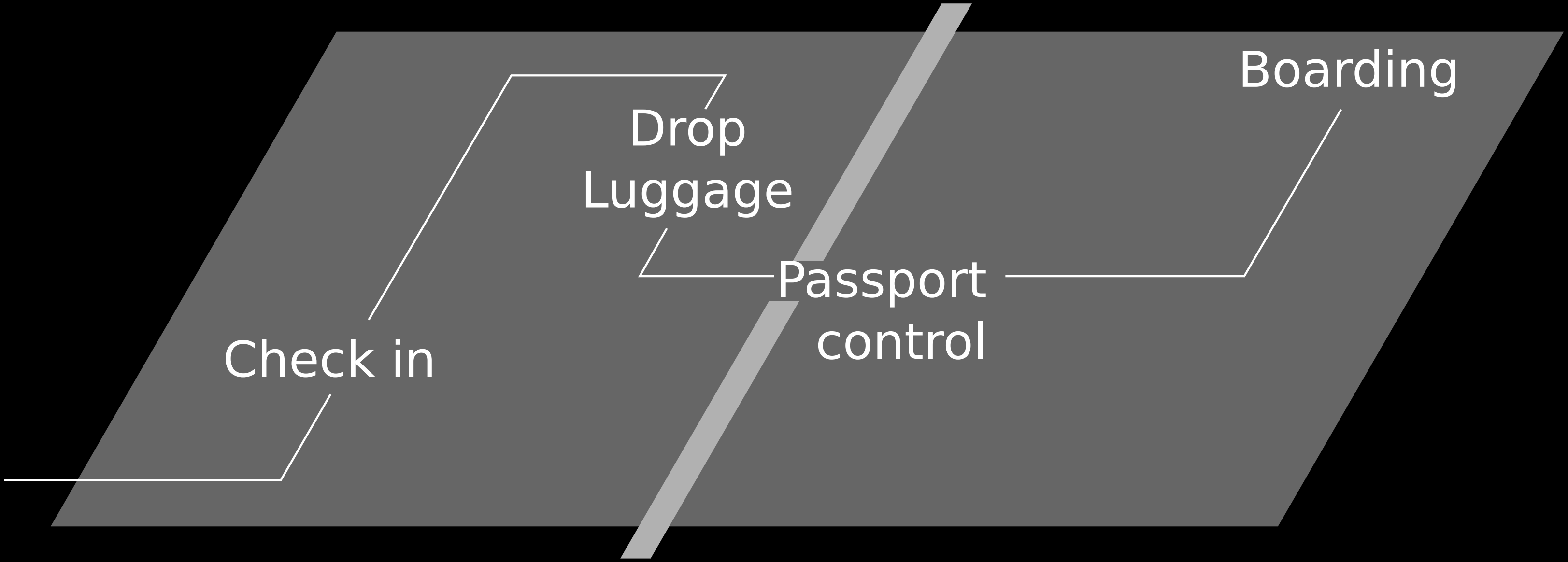
Airport



Airport



Airport



post-hypnotic suggestion

it's ok

model

to have

more than one

traditionally

data

nouns

Passenger

Flight

Luggage

normalized

DDD

domain model

Passenger

```
book
check in
security-clear
board
```

Flight

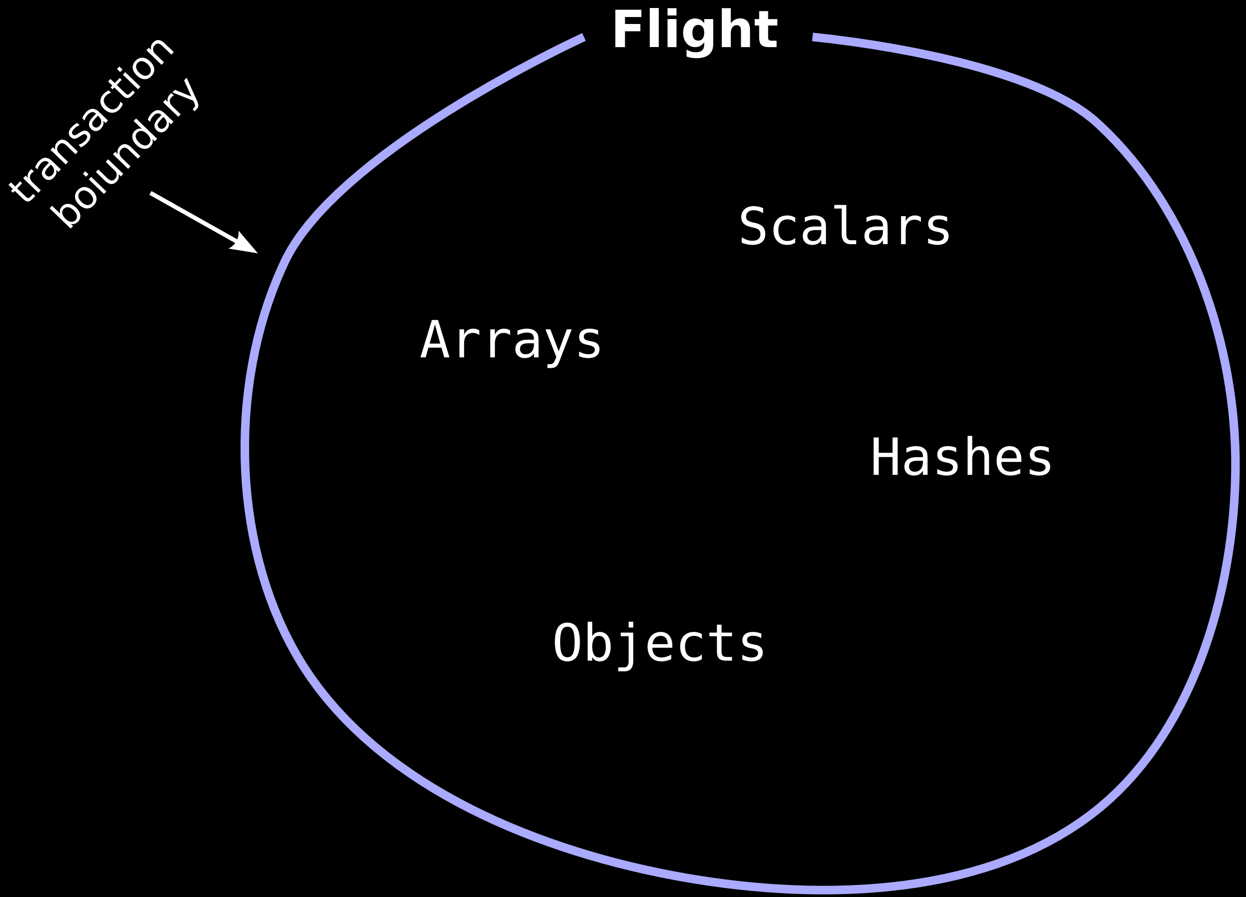
```
register
take off
land
```

Luggage

```
register
```

focus on the verbs

aggregate

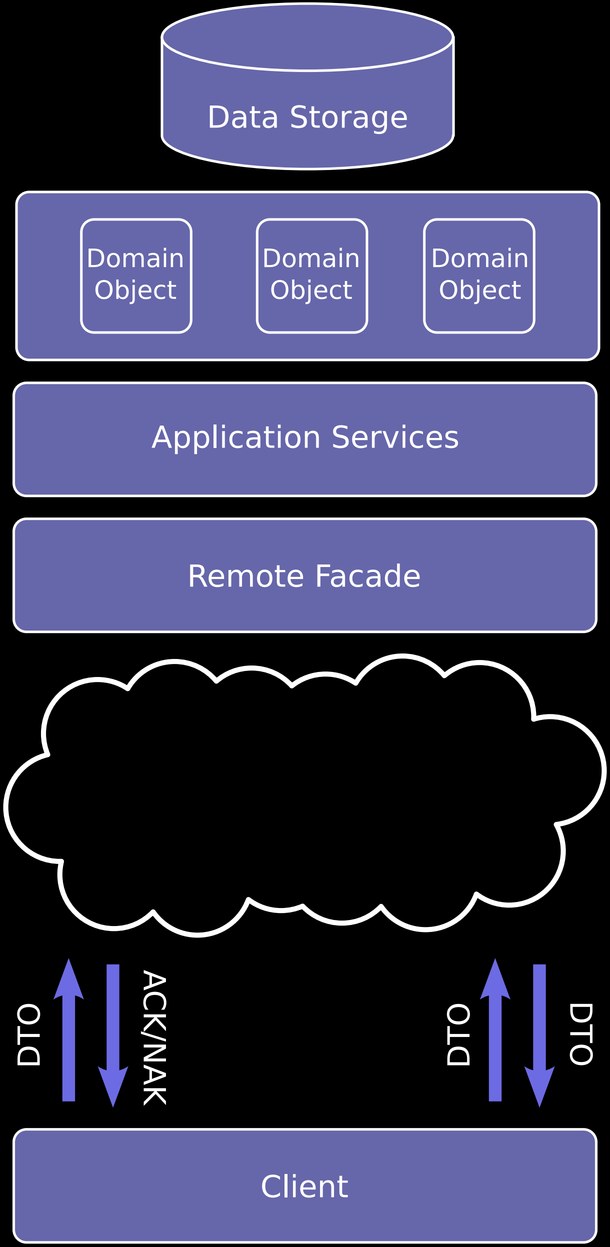

Flight
transaction boiundary
Scalars
Arrays
Hashes
Objects

bounded context

Passenger tracking

Luggage tracking

so, traditionally


Data Storage
Domain Object
Domain Object
Domain Object
Application Services
Remote Facade
DTO
ACK/NAK
DTO
DTO
Client

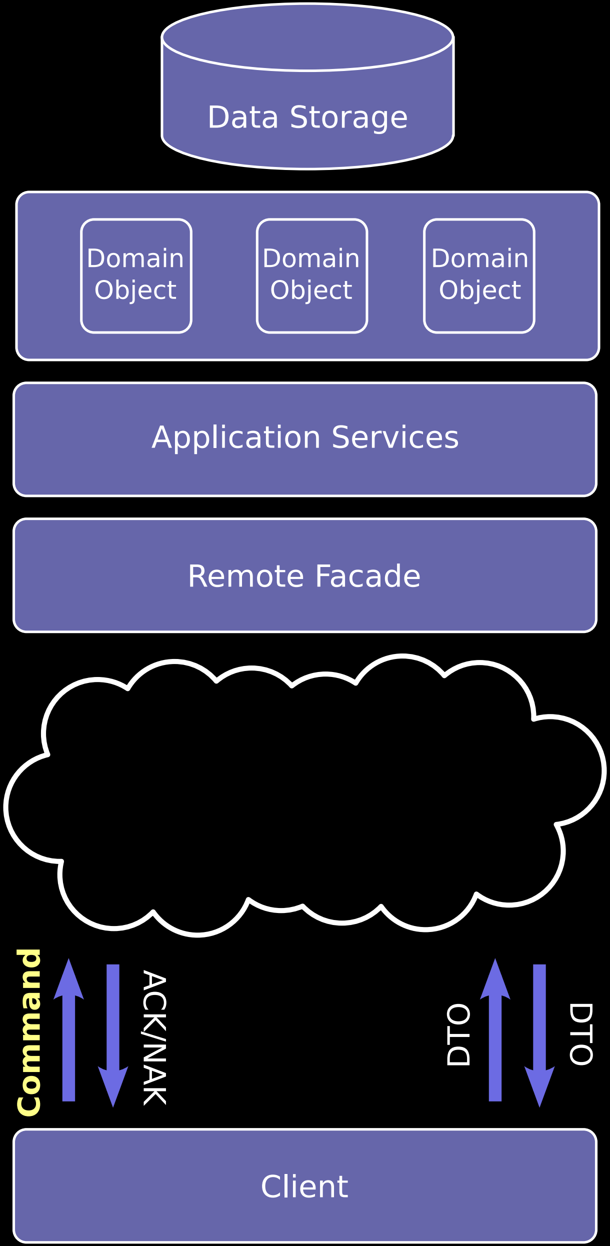
```
PassangerService

void PutPassengerInFirstClass(PassengerId)
Passenger GetPassenger(PassengerId)
ArrayRef[Passenger] GetPassengersWithName(Name)
ArrayRef[Passenger] GetFirstClassPassengers()
void ChangePassengerLocale(PassengerId, NewLocale)
void RegisterPassenger(Name, SSN, FlightId)
void EditPassengerDetails(PassengerDetails)
```

PassangerWriteService

```
void PutPassengerInFirstClass(PassengerId)
void ChangePassengerLocale(PassengerId, NewLocale)
void RegisterPassenger(Name, SSN, FlightId)
void EditPassengerDetails(PassengerDetails)
```

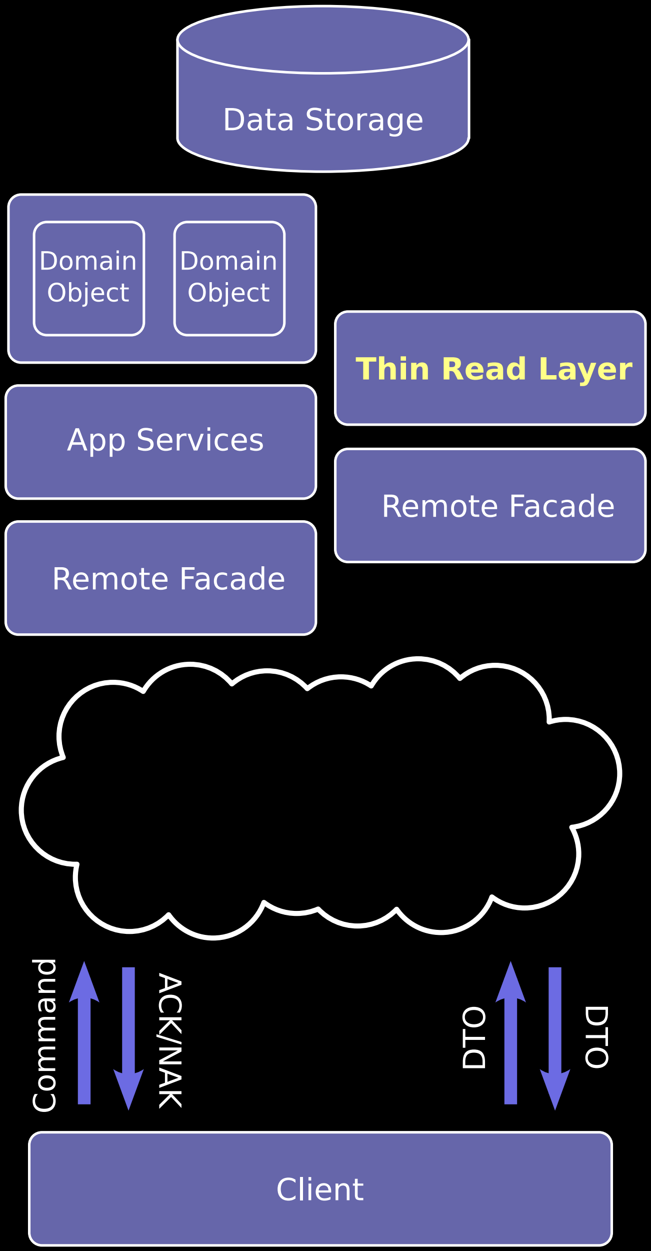
PassangerReadService

```
Passenger GetPassenger(PassengerId)
ArrayRef[Passenger] GetPassengersWithName(Name)
ArrayRef[Passenger] GetFirstClassPassengers()
```


Data Storage
Domain Object
Domain Object
Domain Object
Application Services
Remote Facade
Command
ACK/NAK
DTO
DTO
Client

the end

the end?


Data Storage
Domain Object
Domain Object
Thin Read Layer
App Services
Remote Facade
Remote Facade
Command
ACK/NAK
DTO
DTO
Client

hm…

read-side/write-side

be normal

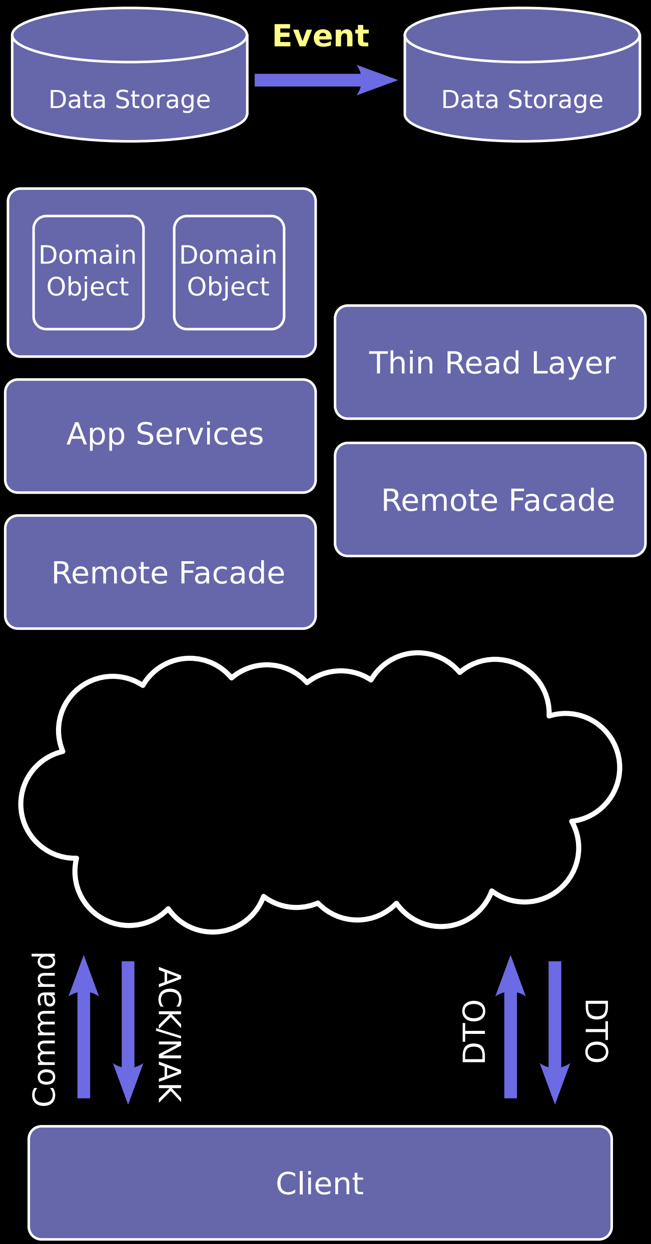
why?

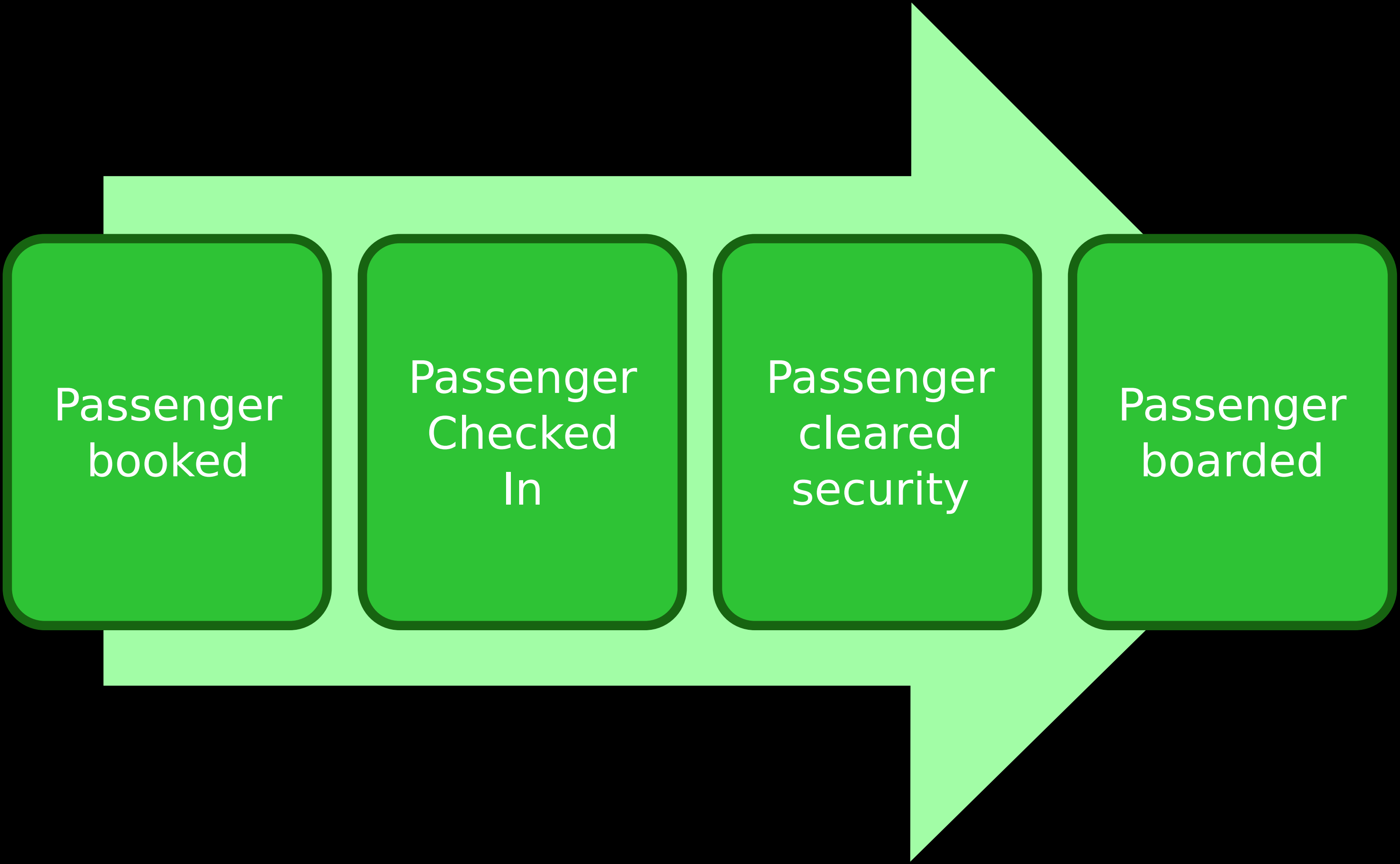
Databases intended for
online transaction processing (OLTP)
are typically more normalized than
databases intended for
online analytical processing (OLAP).

- Wikipedia

reads are common

optimize for reads

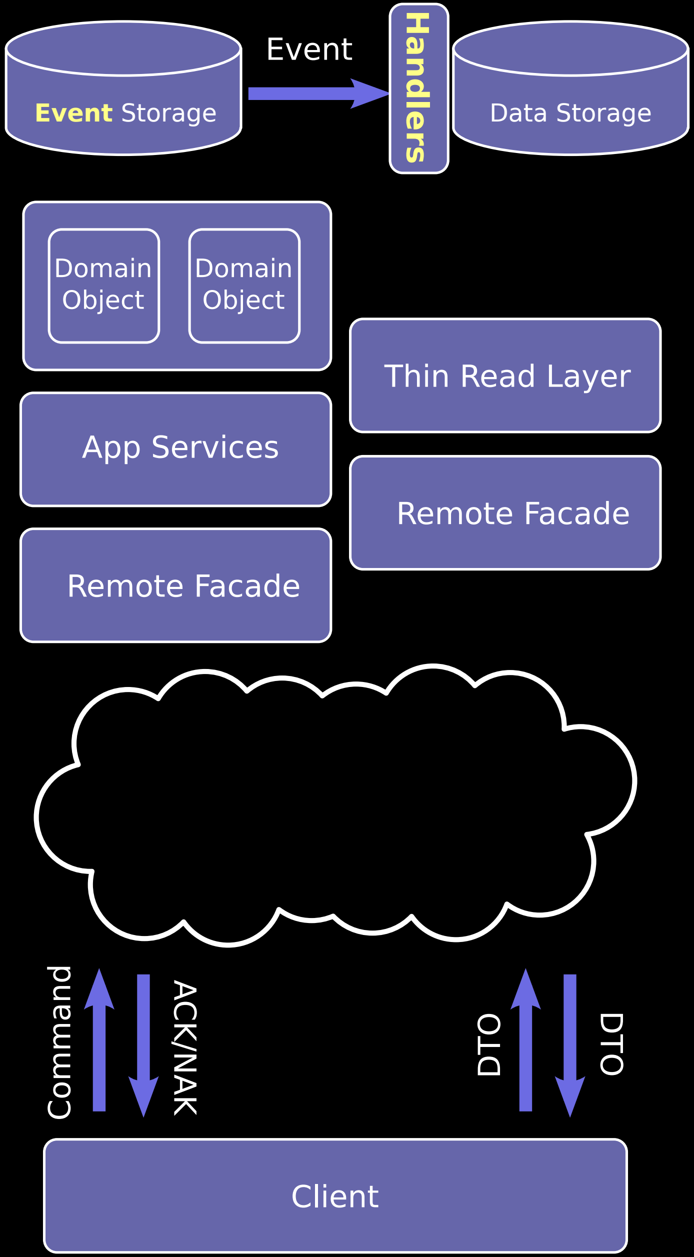

Data Storage
Event
Data Storage
Domain Object
Domain Object
App Services
Remote Facade
Thin Read Layer
Remote Facade
Command
ACK/NAK
DTO
DTO
Client


Passenger booked
Passenger Checked In
Passenger cleared security
Passenger boarded

```
sum = foldl (+) 0
```

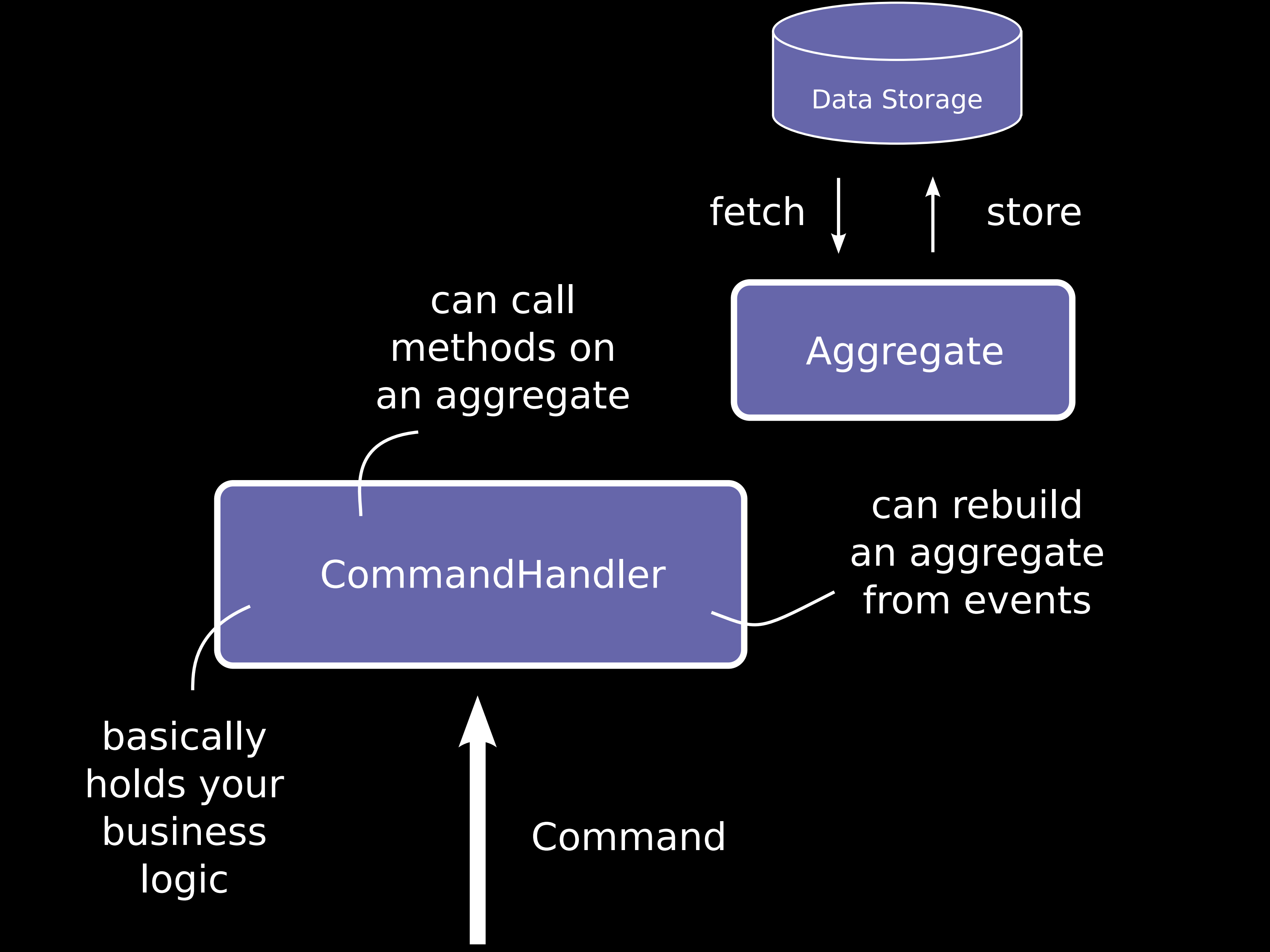
state = foldl apply empty

customize your read-side


Event Storage
Event
Handlers
Data Storage
Domain Object
Domain Object
App Services
Remote Facade
Thin Read Layer
Remote Facade
Command
ACK/NAK
DTO
DTO
Client

current state isn't always enough

need prediction FAIL

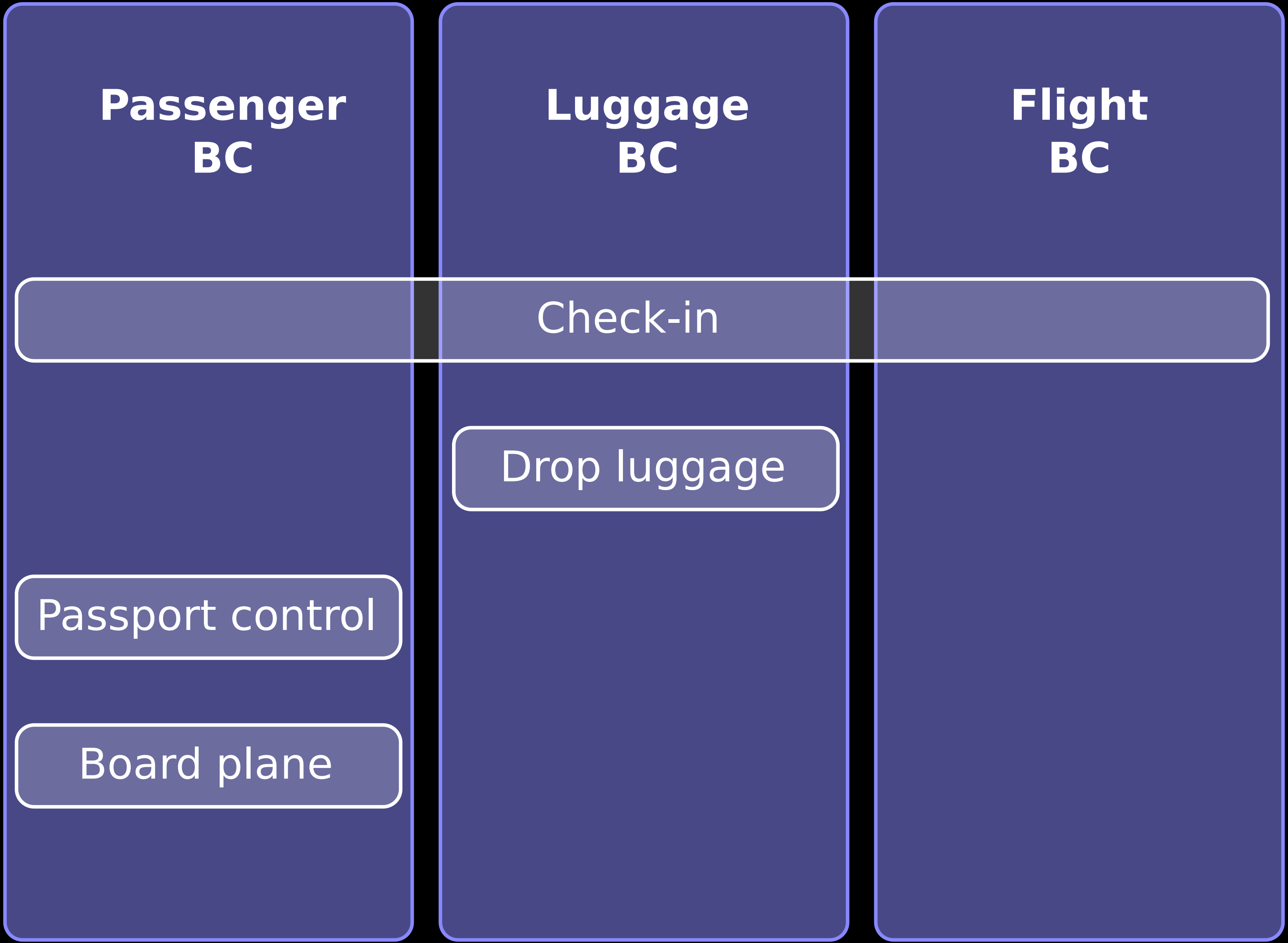

Data Storage
fetch
store
Aggregate
can call methods on an aggregate
CommandHandler
can rebuild an aggregate from events
basically holds your business logic
Command

Events table

Column name	Column type
AggregateId	Guid
Data	Blob
Version	Int

Aggregates table

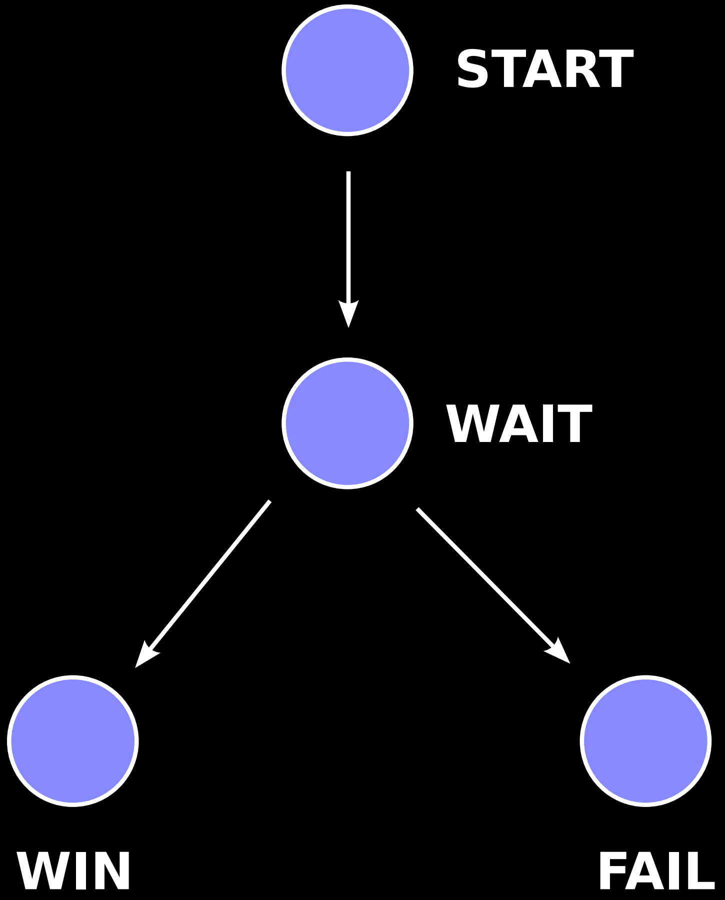
Column name	Column type
AggregateId	Guid
Type	Varchar
Version	Int


Passenger BC
Luggage BC
Flight BC
Check-in
Drop luggage
Passport control
Board plane

problem

consistency

saga


START
WAIT
WIN
FAIL

testing

Given

an aggregate in a certain state

When

an action performed on the aggregate

Then

a number of consequences

Given

a number of events

When

an action performed on the aggregate

Then

a number of consequences

Given

`ArrayRef[Event]`

When

an action performed on the aggregate

Then

a number of consequences

Given

`ArrayRef[Event]`

When

a command performed on the aggregate

Then

a number of consequences

Given

`ArrayRef[Event]`

When

`Command`

Then

a number of consequences

Given

`ArrayRef[Event]`

When

`Command`

Then

a number of events

Given

`ArrayRef[Event]`

When

`Command`

Then

`ArrayRef[Event]`

Given

```
ArrayRef[Event]
```

When

```
Command
```

Then

```
ArrayRef[Event] | Exception
```

team independence

agile

outsourcing

summary

more than one model

aggregates

CQRS

read side/write side

event sourcing

thank you