Hello.
An example scenario
Imagine you’re a happy developer
### Shopping Cart

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ROCKLINE BASKET COFFEE FILTERS (8-12 Cup Basket) 700 Filters</strong> by</td>
<td>$7.91</td>
<td>1</td>
</tr>
<tr>
<td>Connoisseur</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In Stock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eligible for FREE Shipping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>□ This is a gift Learn more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>Save for later</td>
<td></td>
</tr>
<tr>
<td><strong>Hamilton Beach 46201 12 Cup Digital Coffeemaker, Stainless Steel</strong> by</td>
<td>$39.00</td>
<td>1</td>
</tr>
<tr>
<td>Hamilton Beach</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In Stock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eligible for FREE Shipping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>□ This is a gift Learn more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>Save for later</td>
<td></td>
</tr>
<tr>
<td><strong>Cuisinart DGB-900BC Grind &amp; Brew Thermal 12-Cup Automatic Coffee...</strong> by</td>
<td>$159.00</td>
<td>2</td>
</tr>
<tr>
<td>Cuisinart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>In Stock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Eligible for FREE Shipping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>□ This is a gift Learn more</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>Save for later</td>
<td></td>
</tr>
</tbody>
</table>

**Subtotal (4 items):** $364.91
Need those Reports...
You got it.
You work furiously...
### Simple Cart Schema

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>user_id</td>
<td>integer</td>
</tr>
<tr>
<td>sku</td>
<td>character varying(25)</td>
</tr>
<tr>
<td>quantity</td>
<td>integer</td>
</tr>
</tbody>
</table>

Indexes:
- "shopping_cart_sku_index" btree (index)
- "shopping_cart_user_id_index" btree (index)
A Quick, Simple, Task
Except, There’s No Data
A Year With Event Sourcing and CQRS

Steve Pember
CTO, ThirdChannel
steve@thirdchannel.com

Software Architecture Conf, 2016
Agenda

- CQRS
- Event Sourcing
- This Tech @ ThirdChannel
- Demo
CQRS... ?
Command - Query
Responsibility Segregation
CRUD vs CQRS
CRUD / MVC
Command  
Query  
Responsibility  
Segregation
It all started with...
“Some objects are not defined primarily by their attributes. They represent a thread of identity that runs through time and often across distinct representations. Sometimes such an object must be matched with another object even though attributes differ. An object must be distinguished from other objects even though they might have the same attributes.”

–Eric Evans
Domain Driven Design

- Ubiquitous Language
Domain Driven Design

• Ubiquitous Language

• Entities / Value Objects
Entity:
when you care about identity

Who Am I?
Value
Objects:
don’t care
about
individual
Domain Driven Design

- Ubiquitous Language
- Entities / Value Objects
- Aggregates
Group of Entities with one Root
Domain Driven Design

- Ubiquitous Language
- Entities / Value Objects
- Aggregates
- Bounded Context
Objects outside of the Aggregate can permanently hold references, but only to Roots, and only by ID.

Objects outside of the Aggregate may not hold references to 'inner' objects.
Objects inside an Aggregate may hold refs to other Roots
Deleting an Aggregate also deletes dependents.
Isolate related objects into Modules
Furthermore, No direct comms across boundaries
Domain Driven Design

- Ubiquitous Language
- Entities / Value Objects
- Aggregates
- Bounded Context
- Domain Events
Some Mechanism For Passing Events (e.g. Direct, Pub/Sub, Broker)

- User Module
- Address Module
- Orders Module
- Cart Module
- Products Module
- Billing Module
Events are Transactionally Safe
Back to CQRS...
CQRS is an Evolution on DDD
Basis: Two Objects where there was One
But allows for Interesting and efficient Architectures
Multiple Query Caches derived from events
<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROCKLINE BASKET COFFEE FILTERS (8-12 Cup Basket) 700 Filters by Connoisseur</td>
<td>$7.91</td>
<td>1</td>
</tr>
<tr>
<td>Hamilton Beach 46201 12 Cup Digital Coffeemaker, Stainless Steel by Hamilton Beach</td>
<td>$39.00</td>
<td>1</td>
</tr>
<tr>
<td>Cuisinart DGB-900BC Grind &amp; Brew Thermal 12-Cup Automatic Coffee... by Cuisinart</td>
<td>$159.00</td>
<td>2</td>
</tr>
</tbody>
</table>

Subtotal (4 items): $364.91
When should you use CQRS?
With Microservices
Complex Domains
High Performant Systems
Task-Based UIs
Create Issue

Issue Type: Story

Summary: As a User, this form is too big.

Assignee: Automatic

Component/s:

Labels:

Sprint:

Story Points:

Description:

Create
console.log(this.model);

should remove this.

Other than the console log, looks fine to me.
However...
The creators caution against using CQRS
Ok, so how about Event Sourcing?
What do we do with our Domain Events?
Event Sourcing

- Alternative Storage Pattern - Save Deltas
Store **Deltas** instead of Current State
Objects are backed - ‘sourced’ - by events
Event Sourcing

- Alternative Storage Pattern - Save Deltas
- Event vs Aggregate
Event: Something which has occurred in the system
Aggregate: from DDD!
Event Sourcing

• Alternative Storage Pattern - Save Deltas

• Event vs Aggregate

• Purely Additive
There is no Delete
Event Sourcing

- Alternative Storage Pattern - Save Deltas
- Event vs Aggregate
- Purely Additive
- It’s Old
Event Sourcing

- Alternative Storage Pattern - Save Deltas
- Event vs Aggregate
- Purely Additive
- It’s Old
- Event Stream $\rightarrow$ Transient Models
All Objects are Transient Derivatives of the Event Stream
Some Caveats
I’m sure you have some doubts.
Wouldn’t it take up a large amount of storage space?
Wouldn’t ES add overhead?
Try to avoid ‘heavy’ persistence mechanisms
The toughest thing, though:
ES Can be difficult for junior developers
So Why Would I Want Event Sourcing?
So Why ES?

• More than the Perfect Audit Log
Audit logs tell the History

Events tell the intent of History
Also, You’re now a Time Traveller
So Why ES?

- More than the Perfect Audit Log
- Only Structural Model That Does Not Lose Information
NEVER DELETE!!!
So Why ES?

- More than the Perfect Audit Log
- Only Structural Model That Does Not Lose Information
- Ideal for Business Analysis
Analysis of the Event Stream is a Projection
Current State is Boring

- Mine ALL ProductRemoved events
- Find ALL ProductAdded events followed by RemovedEvents for same product within 5 minutes
- Find Average Duration between ProductAdd and OrderPlaced
Assignment History:

1-31-2015 18:56:02: Status changed to ACTIVE by System
1-31-2015 18:56:02: User Signed Contract
1-30-2015 21:06:05: Status changed to ACCEPTED by Sarah
1-30-2015 21:05:57: Role changed to AGENT by System
1-30-2015 21:05:57: Status changed to WAIT_LISTED by System
1-30-2015 21:05:57: Status changed to ACCEPTED by Sarah
1-30-2015 21:05:56: Status changed to INTERVIEWED by Sarah
1-28-2015 14:31:13: Status changed to NEEDS_INTERVIEW by Sarah
1-28-2015 14:31:12: Status changed to REVIEWED by Sarah
1-27-2015 19:44:12: Status changed to PENDING by System
1-27-2015 19:44:12: Role changed to CANDIDATE by System
1-27-2015 19:44:12: User added to program Retail Intelligence
Future Proof your Data!
Remember the Sad Developer?
With Event Sourcing...
You can see everything
Business Folks Love A Good Report

HA HA!

BUSINESS!
Still...
Businesses depend on **Time Series Data**
Mechanics and Demo
Pure Event Sourcing
3 Base Objects
Aggregate

id: UUID
revision: int
type: String

applyNew(): List<Event>
loadHistorical(): List<Event>
loadFromSnapshot(): Snapshot
Event

id: UUID
revision: int
type: String
aggregateld: UUID
date: Date
userId: String/UUID
data: Text / String / JSON

loadDataFromString(): String
process(): Aggregate
<table>
<thead>
<tr>
<th>Snapshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>id: UUID</td>
</tr>
<tr>
<td>aggregatedId: UUID</td>
</tr>
<tr>
<td>revision: int</td>
</tr>
<tr>
<td>date: Date</td>
</tr>
<tr>
<td>data: Text / String / JSON</td>
</tr>
</tbody>
</table>
Successful Commands Result in new Events on the Aggregate
Loading a Query builds up Aggregates from the Events
Tying It All Together
CQRS & ES
The Event Stream is the Source of Truth.
Pre-Build Query Objects from the Event Stream
Questions So Far?
What have we learned @ThirdChannel?
There are few available tools ...
Apache Kafka
A high-throughput distributed messaging system.
Akka & Akka Persistence
...and we used none of them
Event Source Library

This is a small library for working with Event Sourced aggregates and events. It provides functionality...
Commands & Queries immediately felt like a perfect fit.
...and make sure you append your class names with ‘Command’ and ‘Query’
Query - based Views are an efficiency gain
Query **Read Models** should be able to be freely trashed and rebuilt
We’ve still not run into Aggregate write contention
Save Every Event
You really don’t need snapshots (at first)
Be Wary of Editing an Event
Small Events > Large Events
This is Not...Great.

class MyMassive20PropertiesUpdatedEvent extends Event<String>

    @EventData String firstName
    @EventData String lastName
    @EventData String description
    @EventData String email
    @EventData String productSKU
    @EventData int count
    @EventData boolean active
    @EventData UUID userUUID
    @EventData UUID orderUUID
    @EventData int test
    @EventData int foo
    @EventData int blah
    @EventData int cost
    @EventData int var1
    @EventData int var2
    @EventData int var3
    @EventData int var4
    @EventData int var5
    @EventData int var6
    @EventData int var7

    @Override
    void restoreData(Map data) {
A Bit Better

```java
@CompileStatic
class SurveyQuestionsRemovedEvent extends Event<Survey> {

    @EventData
    List<UUID> questionUUIDs = []

    @Override
    void restoreData(Map data) {
        questionUUIDs = data.questionUUIDs?.collect(UUID::toString);
    }
```
This next bit is important
Pure Event Sourcing Is Not Always The Best Choice
Include a ‘dateEffective’ field on the Event table

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>uuid</td>
</tr>
<tr>
<td>aggregate_id</td>
<td>uuid</td>
</tr>
<tr>
<td>revision</td>
<td>integer</td>
</tr>
<tr>
<td>date</td>
<td>timestamp with time zone</td>
</tr>
<tr>
<td>date_effective</td>
<td>timestamp with time zone</td>
</tr>
<tr>
<td>clazz</td>
<td>character varying(256)</td>
</tr>
<tr>
<td>user_id</td>
<td>character varying(50)</td>
</tr>
<tr>
<td>data</td>
<td>jsonb</td>
</tr>
</tbody>
</table>
Add Current-State, Indexed, Columns to your Aggregates
And Make Many of Them
Demo Time
Thank You!
Image Credits

- Mad Developers: Last Week Tonight
- Obama Thumbs up: http://amiloszportraits.com/barack-obama-thumbs-up-meme
- Star Trek face palm: http://tvtropes.org/pmwiki/pmwiki.php/FacePalm/LiveActionTV
- Obama, Nye, and Tyson: en.wikipedia.org/wiki/Selfie