

# CQRS AND EVENT SOURCING



@metehan\_gltkn



Metehan Gultekin

Java Developer at @Definex

Co-organizer at @Folksdev



@metehan\_gltkn



@mgmetehan



@mgmetehan

C Q R S

Command  
(Komut)

Query  
(Sorgu)

Responsibility  
(Sorumluluk)

Segregation  
(Ayrıştırma)

CQRS == CQS  
??

- ★ Bir fonksiyon ya veriyi değiştirmeli (command) ya da veriyi döndürmeli (query), ancak her ikisini aynı anda yapmamalıdır.
- ★ 1988 yılında tanıtılmıştır.

# CQRS vs CRUD

*read*  
Query == /GET

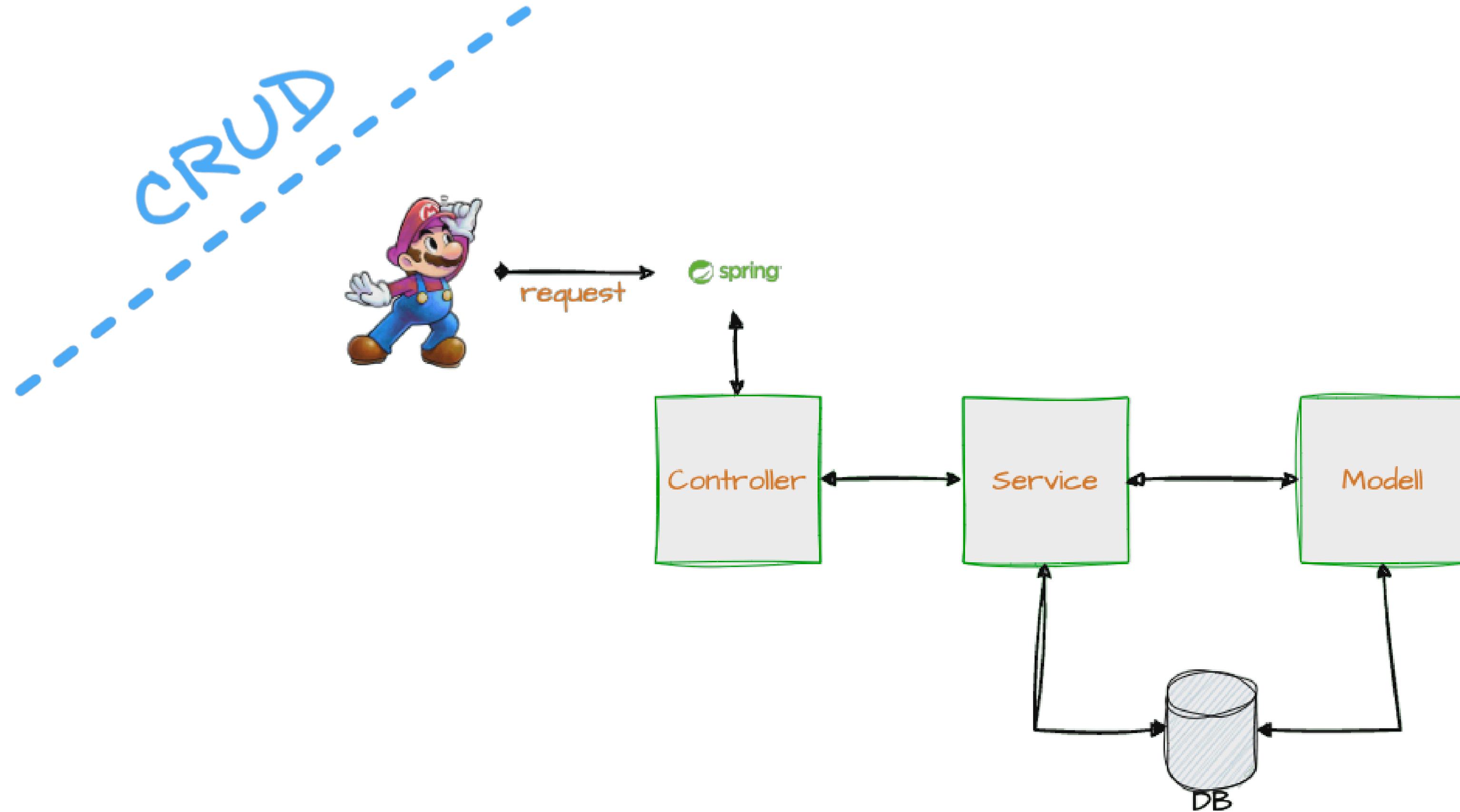
*write*  
Command == /PUT  
/DELETE

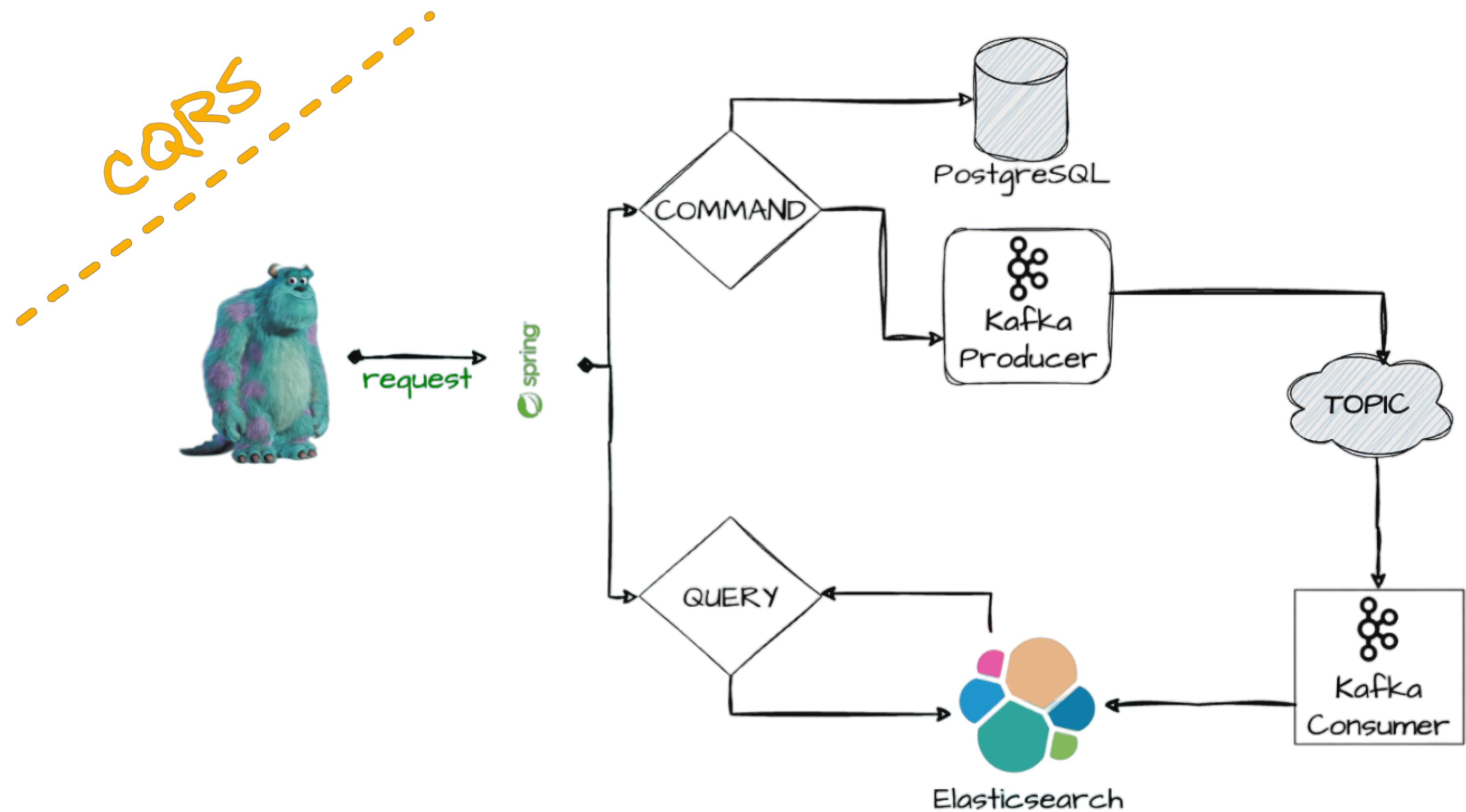
Create == /POST

Read == /GET

Update == /PUT

Delete == /DELETE





## CQRS'nin avantajları:

- \* Çeşitli Çözümler Sunma Esnekliği
- \* Hata ve Kesinti İzolasyonu
- \* Daha İyi Güvenlik Kontrolü
- \* Read Model Optimizasyonu
- \* Farklı Domain Logic Üzerinde Çalışma
- \* Bakım Kolaylığı
- \* Geliştirme Hızı
- \* Bekleme Süresi Olmadan İşlemlerin Yürütülmesi
- \* Bağımsız Ölçekleme

## CQRS'nin dezavantajları:

- \* Karmaşıklık ve Öğrenme Eğrisi
- \* Artan Geliştirme Zamanı
- \* Veri Senkronizasyonu Zorlukları
- \* İki Veritabanı Kullanımı & Maliyet Artışı
- \* Eğitim ve Adaptasyon Zorlukları
- \* İş Süreçleri Arasındaki Kopukluk
- \* Artan İletişim İhtiyacı

CQRS'i

Ne Zaman

Kullanmalıyım

ve

Kullanmamalıyım

?





# CQRS Ne Zaman Kullanılmalı?

- ⌘ Karmaşık İş Mantığı
- ⌘ Farklı İhtiyaçlara Sahip Modeller
- ⌘ Yüksek Trafikli Sistemler
- ⌘ Farklı Veritabanı ve Teknoloji Kullanımı
- ⌘ Bağımsız Geliştirme ve Dağıtım
- ⌘ Uygulama Güvenilirliği
- ⌘ Event Sourcing Kullanımı

# CQRS'ı Ne Zaman Kullanmamalıyız?

- ❖ Proje Karmaşıklığı ve Ölçeği Küçükse
- ❖ Ekip Deneyimi ve Eğitimi Yetersizse
- ❖ Ekstra Karmaşıklık İstenmiyorsa
- ❖ Maliyet ve Kaynak Sınırlamaları
- ❖ Hızlı Geliştirme ve Prototip İhtiyacı
- ❖ Veri Tutarlılığı Mutlak Öncelik Değilse
- ❖ Event Sourcing Gereksinimi Yoksa
- ❖ Performans Kritik Değilse

# CQRS'in Kullanılabileceği Senaryolar

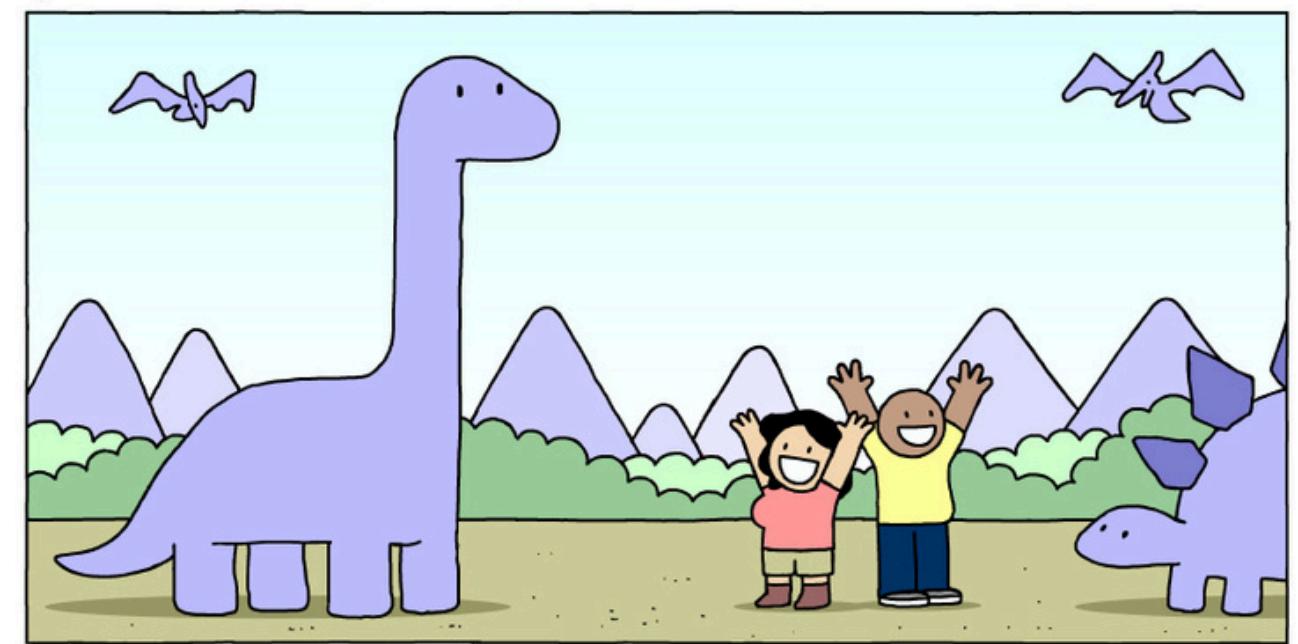
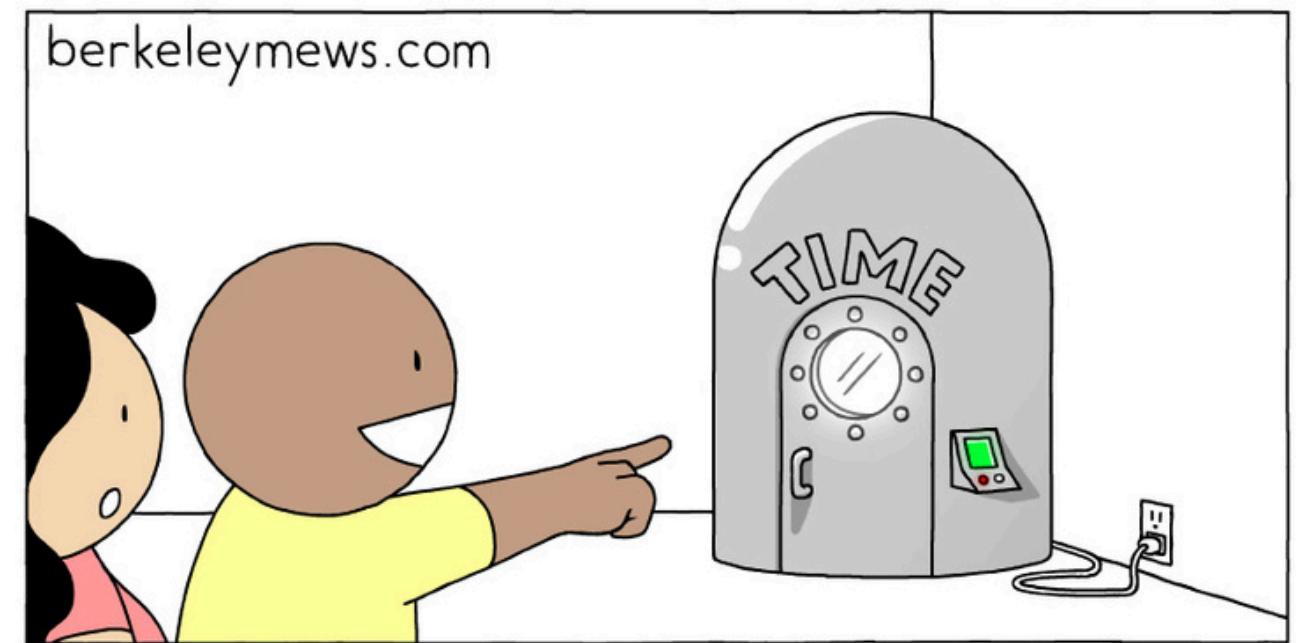
→ Netflix

→ E-Ticaret Platformlar

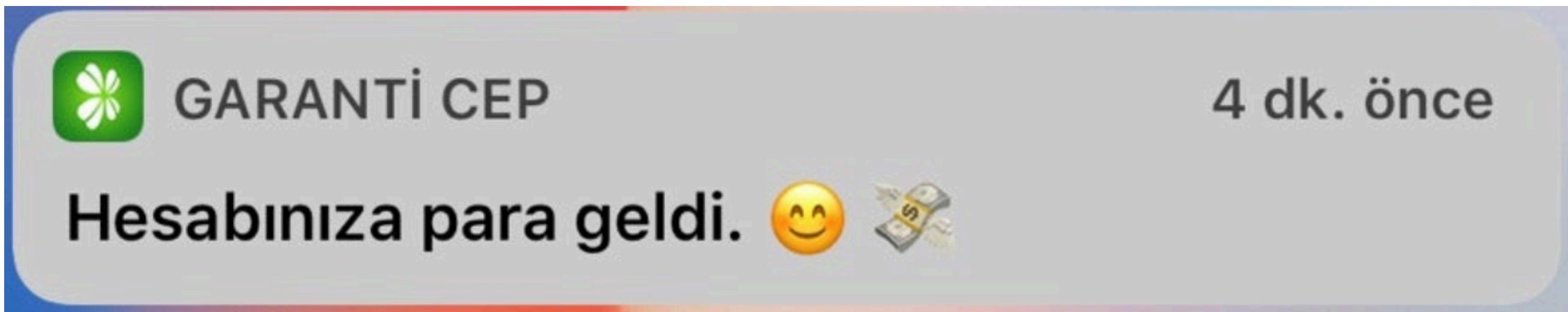
→ Uber

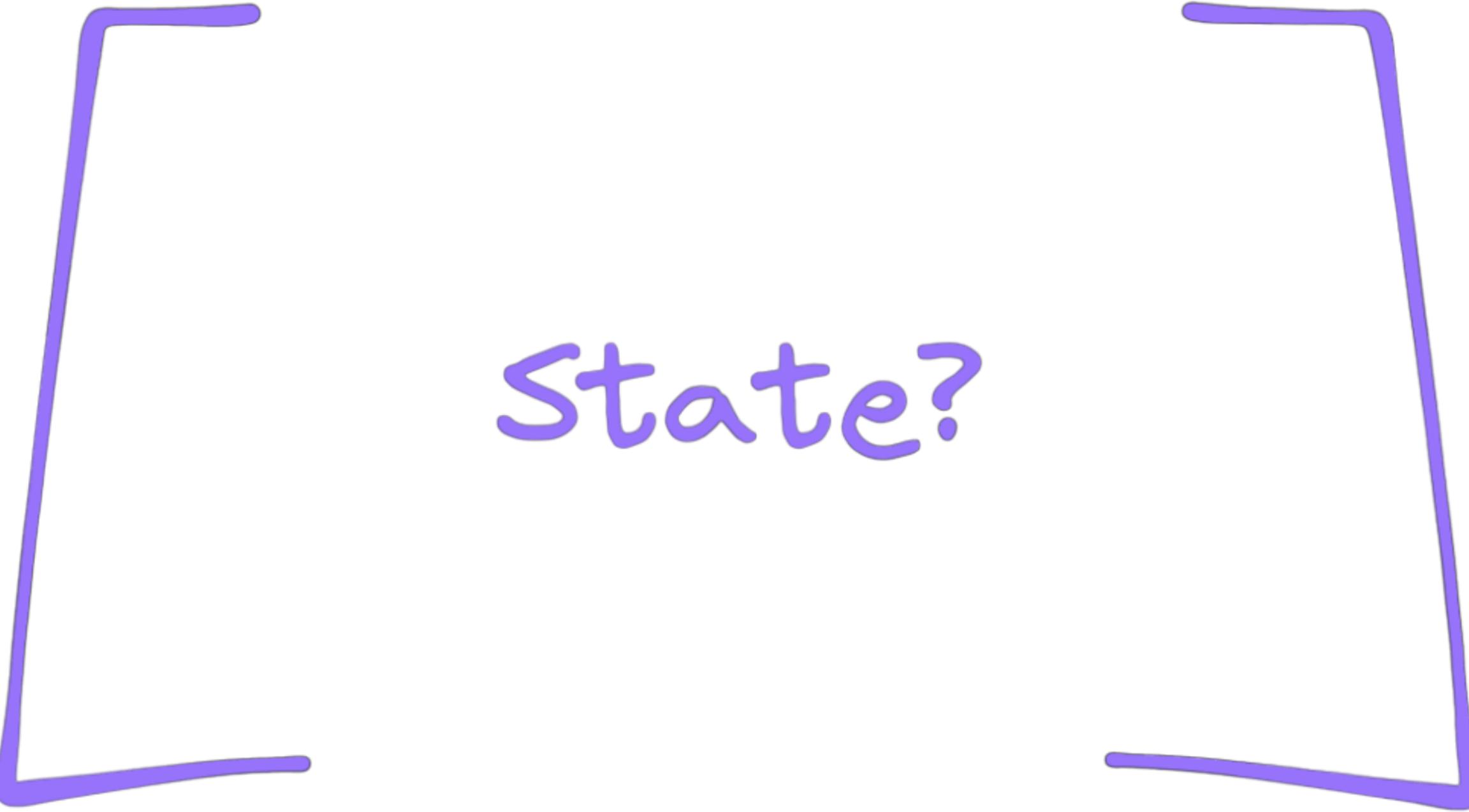
→ Twitter

Event Sourcing  
Olay Tabanlı Mimari



# Event?





State?

Aggregate?

Total Item Stock  
8

Items Created  
8

t1

$\frac{t_1}{T}$



Total Item Stock  
7

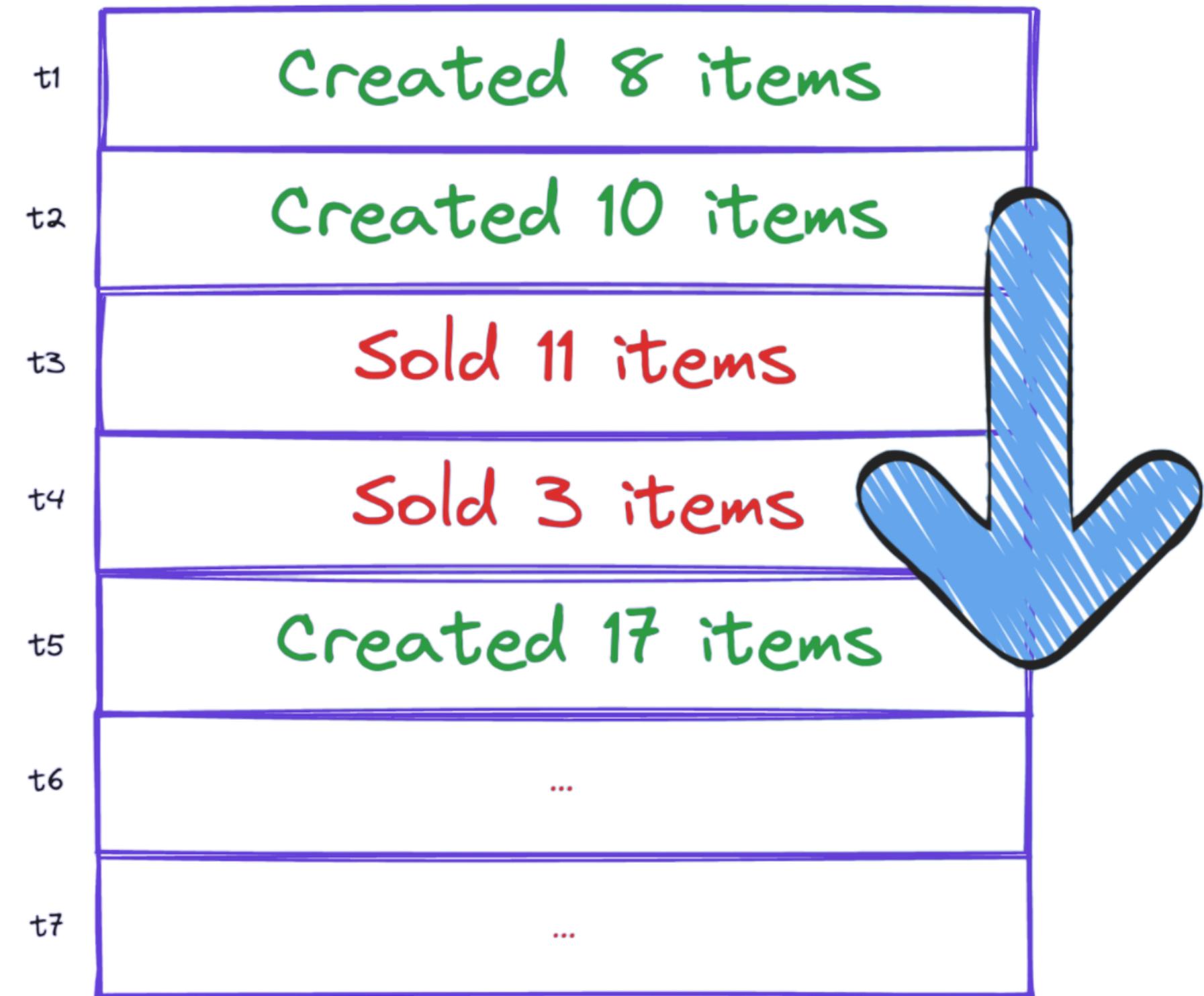
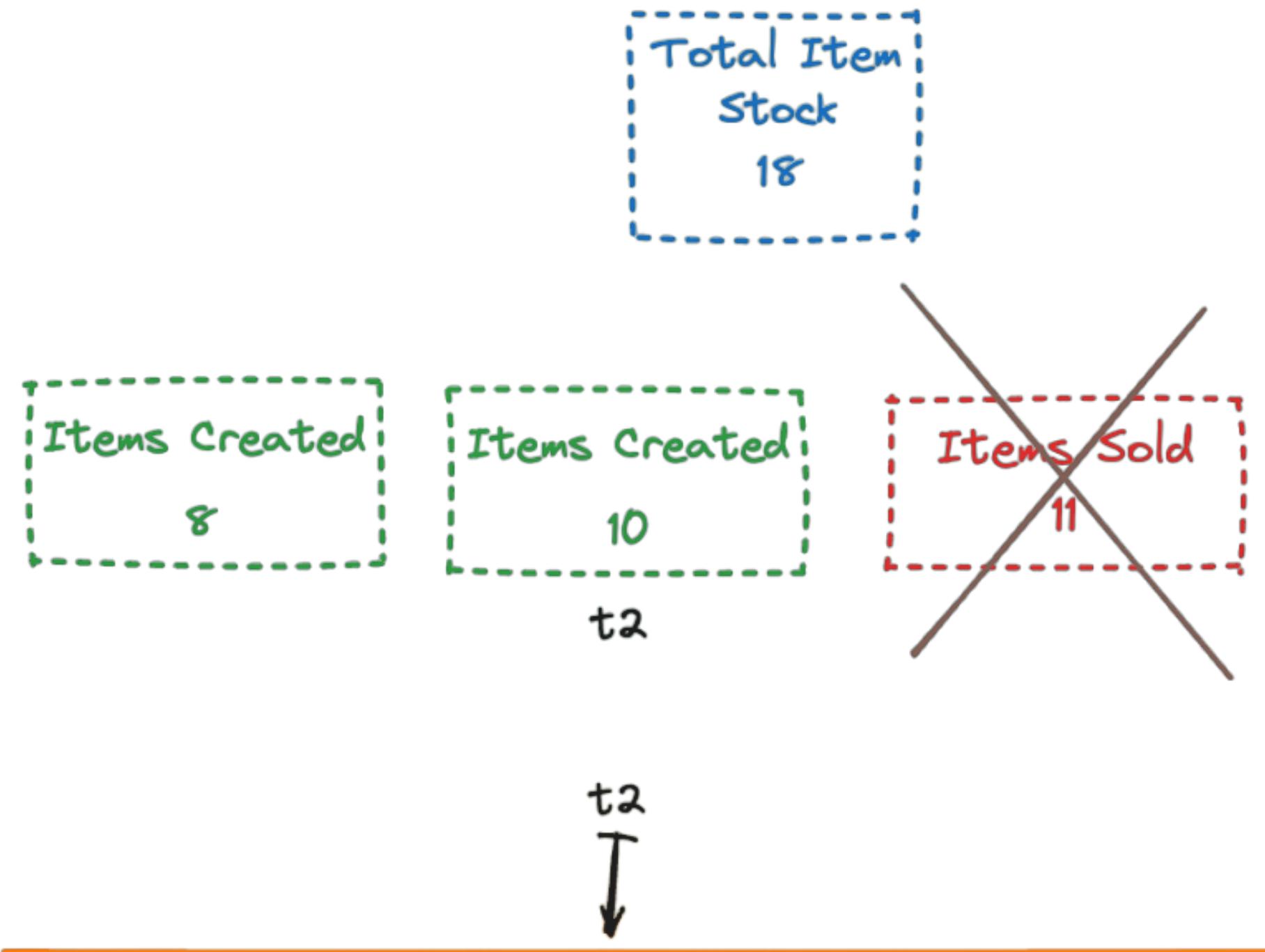
Items Created  
8

Items Created  
10

Items Sold  
11

t3

t3  
↓



Event Sourcing

==

Event-Driven Architecture

??

Event Sourcing, uygulamanın durumunu her durum değişikliğini temsil eden event aracılığıyla kaydeden bir mimari desen

Event-Driven Architecture ise bileşenler arasında event'lerin yaylanması ve dinlenmesine dayanan bir iletişim prensibidir.

# Entities vs Events

Order Created

id	order_name	order_state	create_date	update_date
1	Mac M2	Created	t1	-

t1

event_id	event_type	entity_type	entity_id	event_data	event_date
80	Order Created	Order	101	{...}	t1

Order Approved

id	order_name	order_state	create_date	update_date
1	Mac M2	Approved	t1	t2

t2

event_id	event_type	entity_type	entity_id	event_data	event_date
80	Order Created	Order	101	{...}	t1
81	Order Approved	Order	101	{...}	t2

## Order Shipped

id	order_name	order_state	create_date	update_date
1	Mac M2	Shipped	t1	t3

t3

event_id	event_type	entity_type	entity_id	event_data	event_date
80	Order Created	Order	101	{...}	t1
81	Order Approved	Order	101	{...}	t2
82	Order Shipped	Order	101	{...}	t3
...	...	...	...	...	...
...	...	...	...	...	...

# Event Sourcing'in avantajları

- »» Geçmiş İzleme ve Denetleme
- »» Hata Ayıklama Kolaylığı
- »» Paralel İşleme Yeteneği
- »» Durum Geri Yükleme Yeteneği
- »» İş Zekası ve Raporlama
- »» İleriye Dönük Uyumluluk

## Event Sourcing'in dezavantajları

- »» Karmaşıklık
- »» Öğrenme Eğrisi
- »» Performans
- »» Veritabanı Boyutu
- »» Analistik Zorluklar
- »» Sistem Yeniden İnşası

# Event Sourcing'in Kullanılabileceği Senaryolar

- \* Bankacılık ve Finansal İşlemler
- \* E-Ticaret Sistemleri
- \* Lojistik ve Tedarik Zinciri Yönetimi



*That's all folks!*



@metehan\_gltkn



@mgtmetehan



@mgtmetehan

# Referanslar

- CQRS Documents — Greg Young
- Clarified CQRS — Udi Dahan
- CQRS — Martin Fowler
- CQS — Martin Fowler
- Implementing DDD — Vaughn Vernon
- Azure Cloud CQRS pattern
- Command and Query Responsibility Segregation (CQRS) pattern — Microsoft Docs
- Software Architect's Handbook — Joseph Ingino
- Event Sourcing — Martin Fowler
- Code with Shadman — CQRS