RED HAT
DEVELOPERS

Living without REST: Event-Driven Architecture

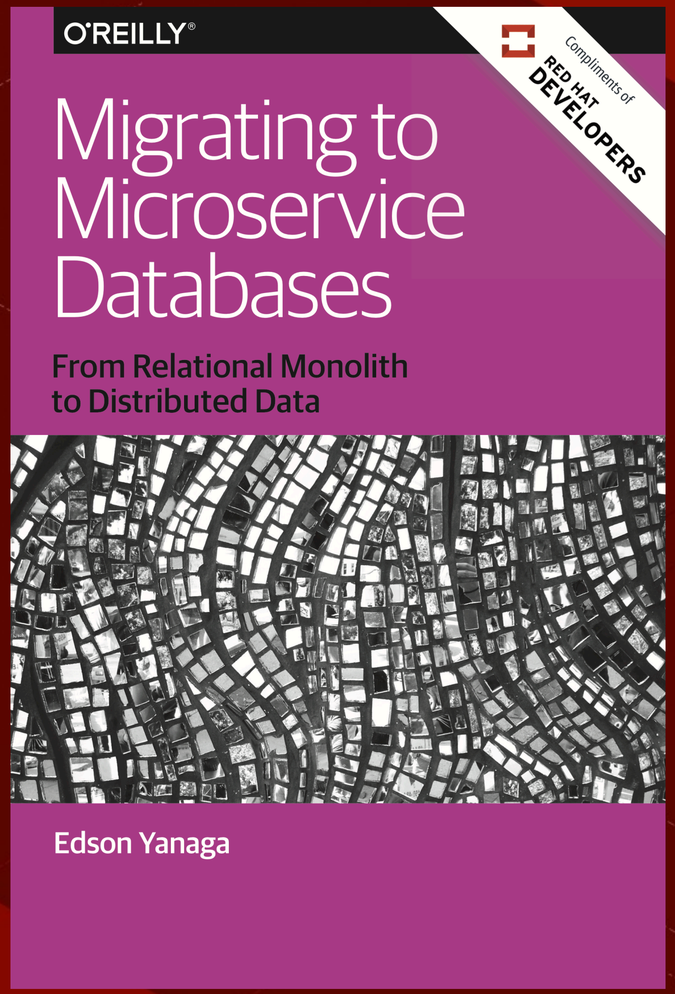
Edson Yanaga
Director of Developer Experience
@yanaga

@yanaga

Edson Yanaga

MVP
Microsoft®
Most Valuable
Professional

THEDEVCONF
ROCKSTAR
THEDEVCONF



https://bit.ly/mono2microdb

Code is easy, state is hard

Latency
Availability
Performance

Cache and/or Polling

Eventual Consistency

First, a look back at the past…

How was data managed 10 years ago?

Terrified about Entity Beans

Hibernate to the rescue!

Replacing XML with @Annotations

POJOs as an (Anemic) Domain Model

Event Sourcing

Account

ID	CUSTOMER_ID	BALANCE
1001	990	1000
1002	991	0
1003	991	-500
1004	992	300

Transactions

ID	ACCOUNT_ID	TIMESTAMP	OP	AMOUNT
1	1001	1234567890	C	1000
2	1002	1234567891	C	200
3	1001	1234567900	D	300
4	1001	1234567995	D	150

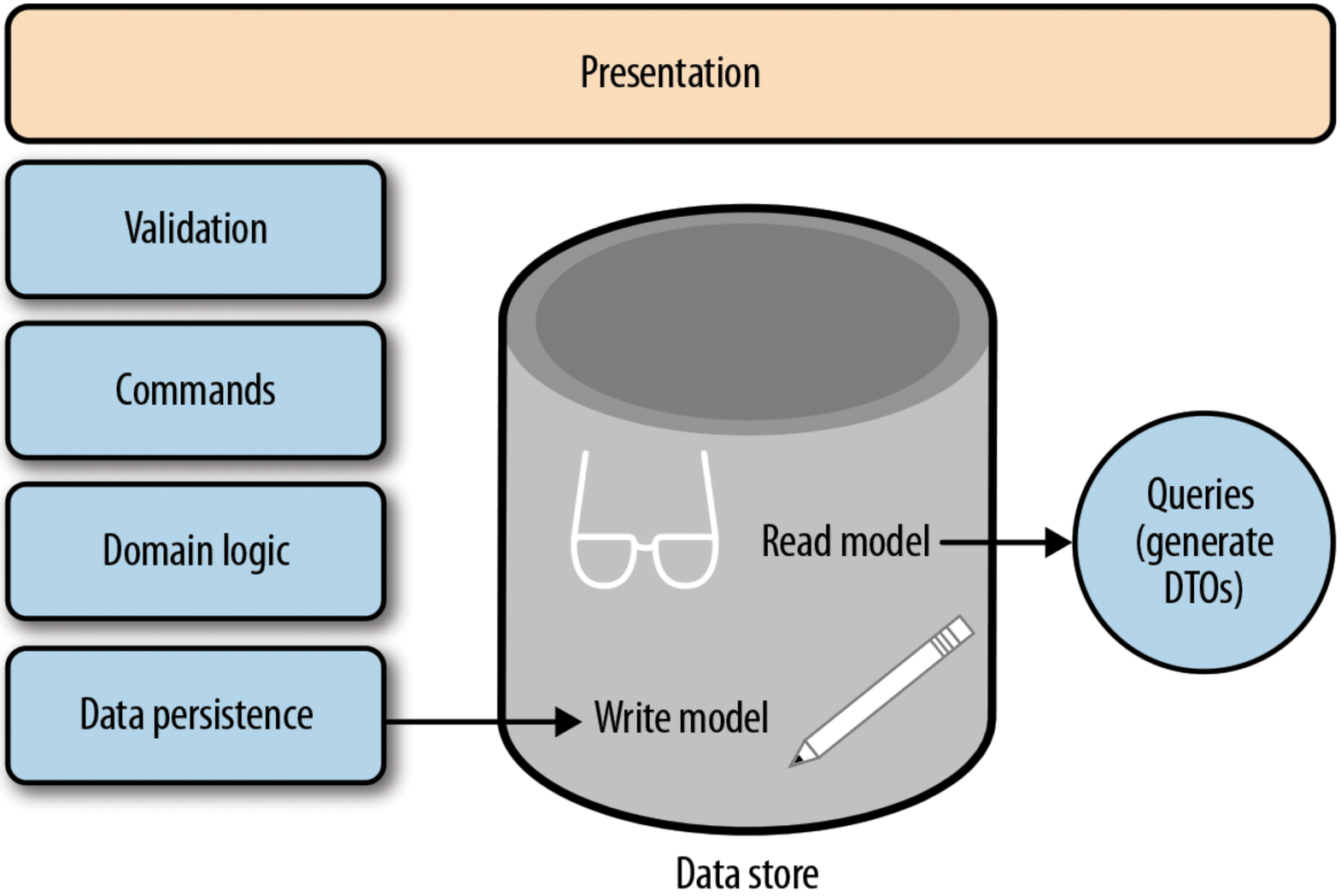
Enables you to think in the **Events** that happened in the system

CQS
(Command-Query Separation)

“Asking a question should not change the answer”
(Bertrand Meyer)

CQRS
(Command Query Responsibility Segregation)

CQRS (Command Query Responsibility Segregation)



ID	NAME	PHONE	ADDRESS	BIRTH
1	Burr	222-222-2323	901 South St	12/12/1968
2	Edson	222-333-3434	112 North Dr	03/03/1978
3	John	111-456-4545	666 Iron St	06/06/1966
4	Doe	333-789-7890	777 Boeing Dr	07/07/1977

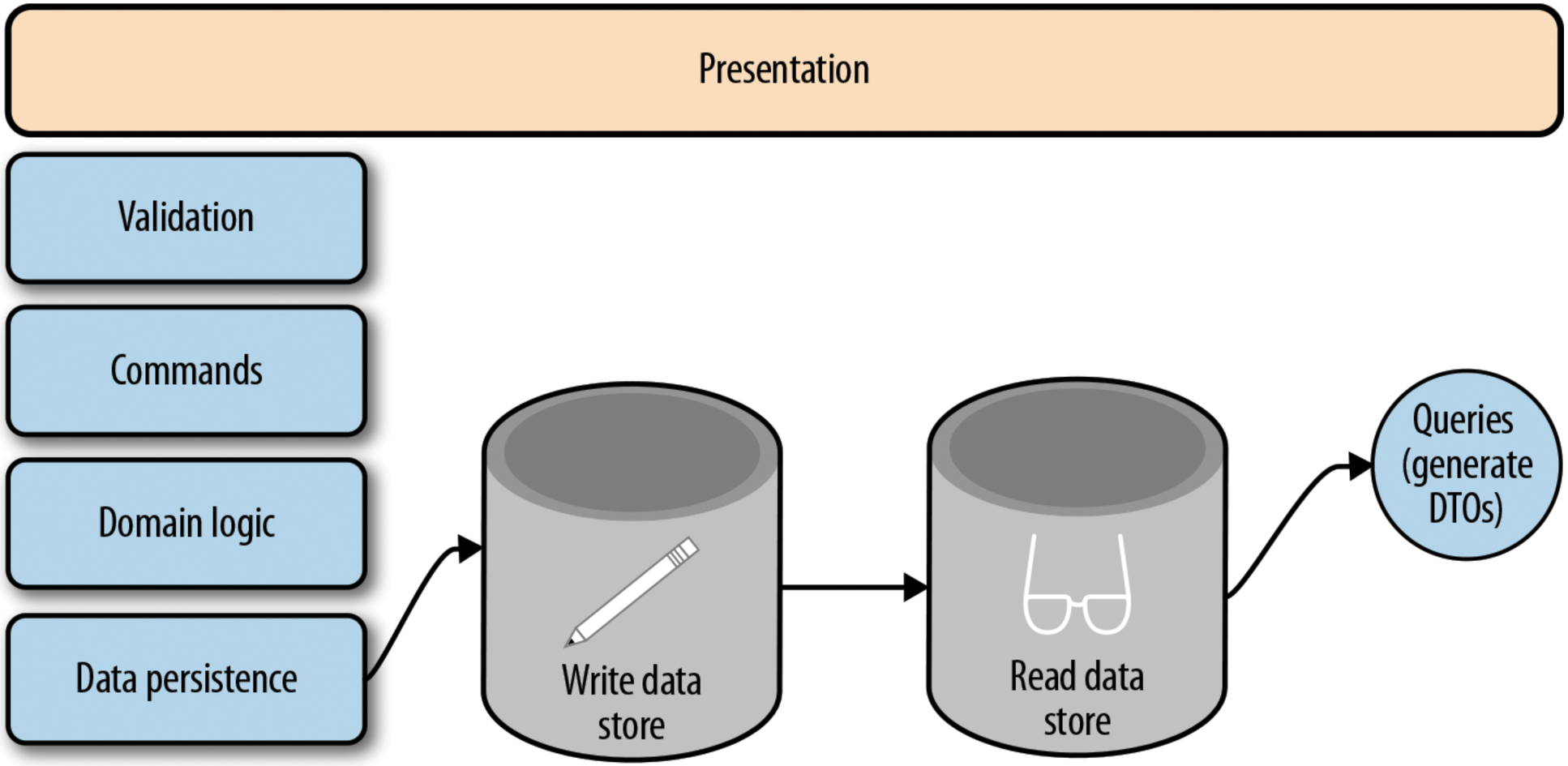
```
INSERT INTO CUSTOMER(ID,NAME,PHONE,ADDRESS,BIRTH);

SELECT * FROM CUSTOMER;
```

```
SELECT ID, NAME, PHONE FROM CUSTOMER;
```

```
SELECT ID, NAME, ADDRESS FROM CUSTOMER;
```

CQRS with separate data stores



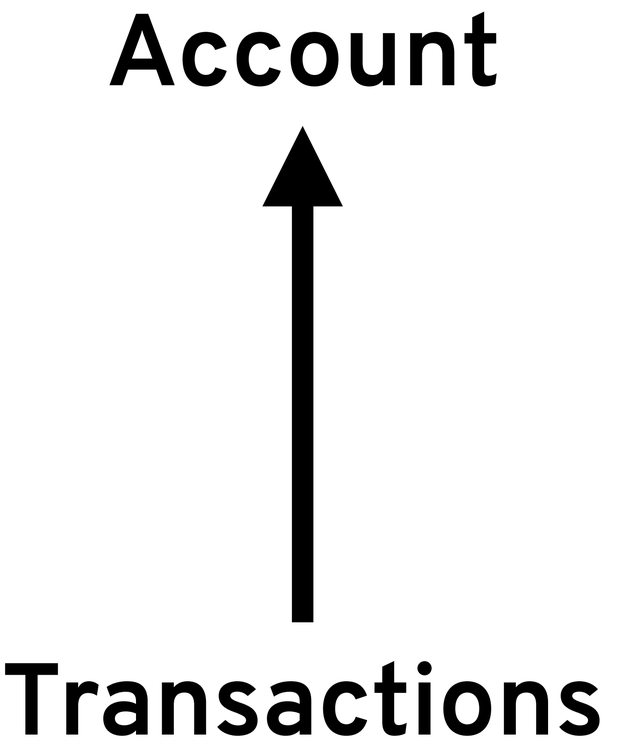
```
SELECT ID, NAME, AGE, AVG_BILL
  FROM CUSTOMER_REPORT_VIEW;
```

```
SELECT ID, PHONE, LAST_PAYMENT_AMOUNT
  FROM CUSTOMER_BILLING_VIEW;
```

CQRS & Event Sourcing

ID	CUSTOMER_ID	BALANCE
1001	990	1000
1002	991	0
1003	991	-500
1004	992	300

ID	ACCOUNT_ID	TIMESTAMP	OP	AMOUNT
1	1001	1234567890	C	1000
2	1002	1234567891	C	200
3	1001	1234567900	D	300
4	1001	1234567995	D	150



ID	CUSTOMER_ID	BALANCE
1001	990	1000
1002	991	0
1003	991	-500
1004	992	300

ID	ACCOUNT_ID	TIMESTAMP	OP	AMOUNT
1	1001	1234567890	C	1000
2	1002	1234567891	C	200
3	1001	1234567900	D	300
4	1001	1234567995	D	150

READ MODEL

Account

Transactions

WRITE MODEL

Why CQRS?

Performance

Distribution
Availability
Integration
Analytics

Canonical Source of Information

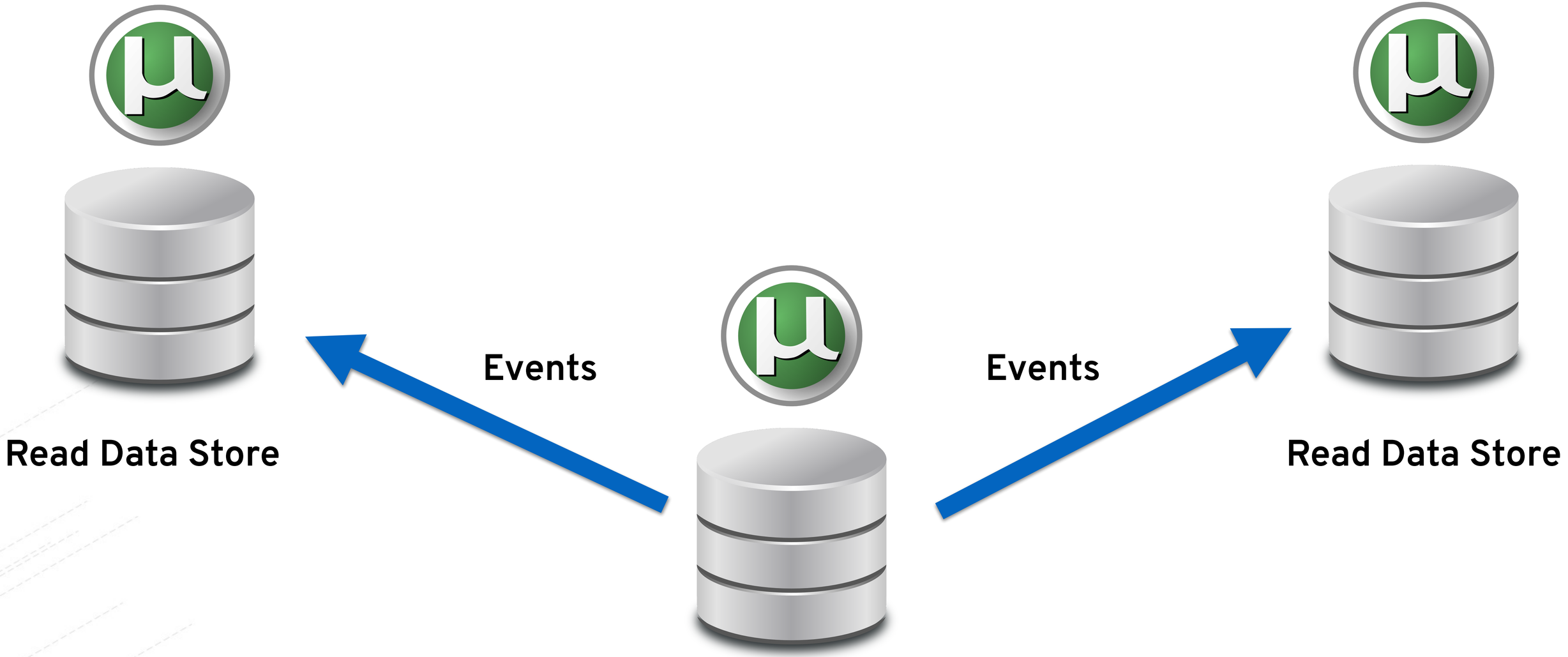
(Write Data Store)

Read Data Store

Canonical Source of Information

(Write Data Store)

Read Data Store


Events
Events
Read Data Store
Read Data Store
Canonical Source of Information
(Write Data Store)

Now, Back to the Future!

Event-Driven Architecture

Events
are facts that **happened** in the system

Low-level Events
vs
Domain-level Events

Key Technologies

ActiveMQ Artemis
Kafka
Camel
Debezium
Quarkus

bit.ly/eda-tutorial

Join
developers.redhat.com

Feedback welcome!
@yanaga

RED HAT DEVELOPERS

RED HAT
DEVELOPERS

Thank you!


plus.google.com/+RedHat


facebook.com/redhatinc


linkedin.com/company/red-hat


twitter.com/RedHatNews


youtube.com/user/RedHatVideos