mongoDB

# Distributed Consensus in MongoDB

Spencer T Brody

Senior Software Engineer at MongoDB

@stbrody

# Agenda

- Introduction to consensus
- Leader-based replicated state machine
- Elections and data replication in MongoDB
- Improvements coming in MongoDB 3.2

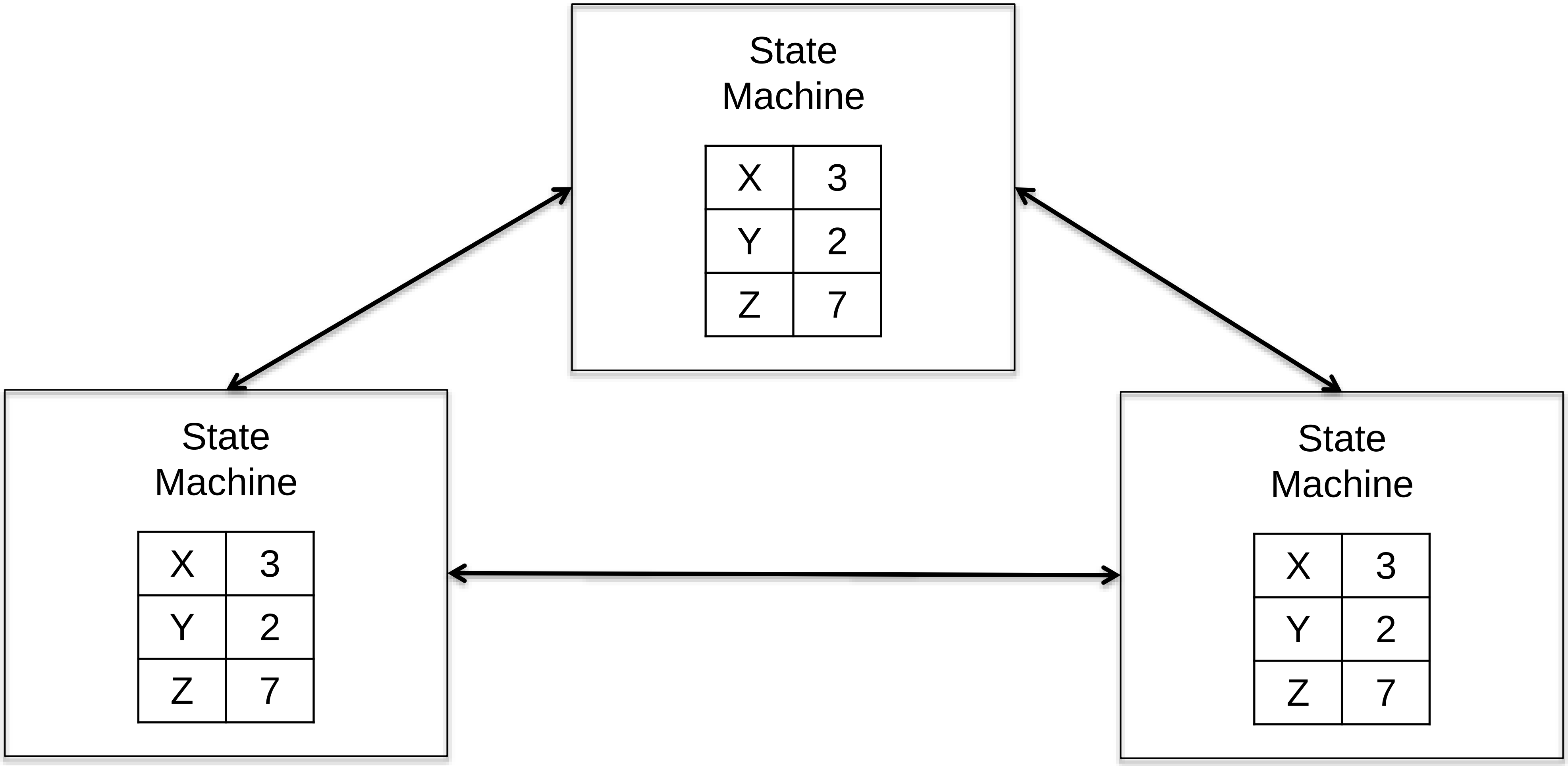
# Why use Replication?

- Data redundancy
- High availability

# What is Consensus?

- Getting multiple processes/servers to agree on something
- Must handle a wide range of failure modes
  - Disk failure
  - Network partitions
  - Machine freezes
  - Clock skews

# Basic consensus



State Machine

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

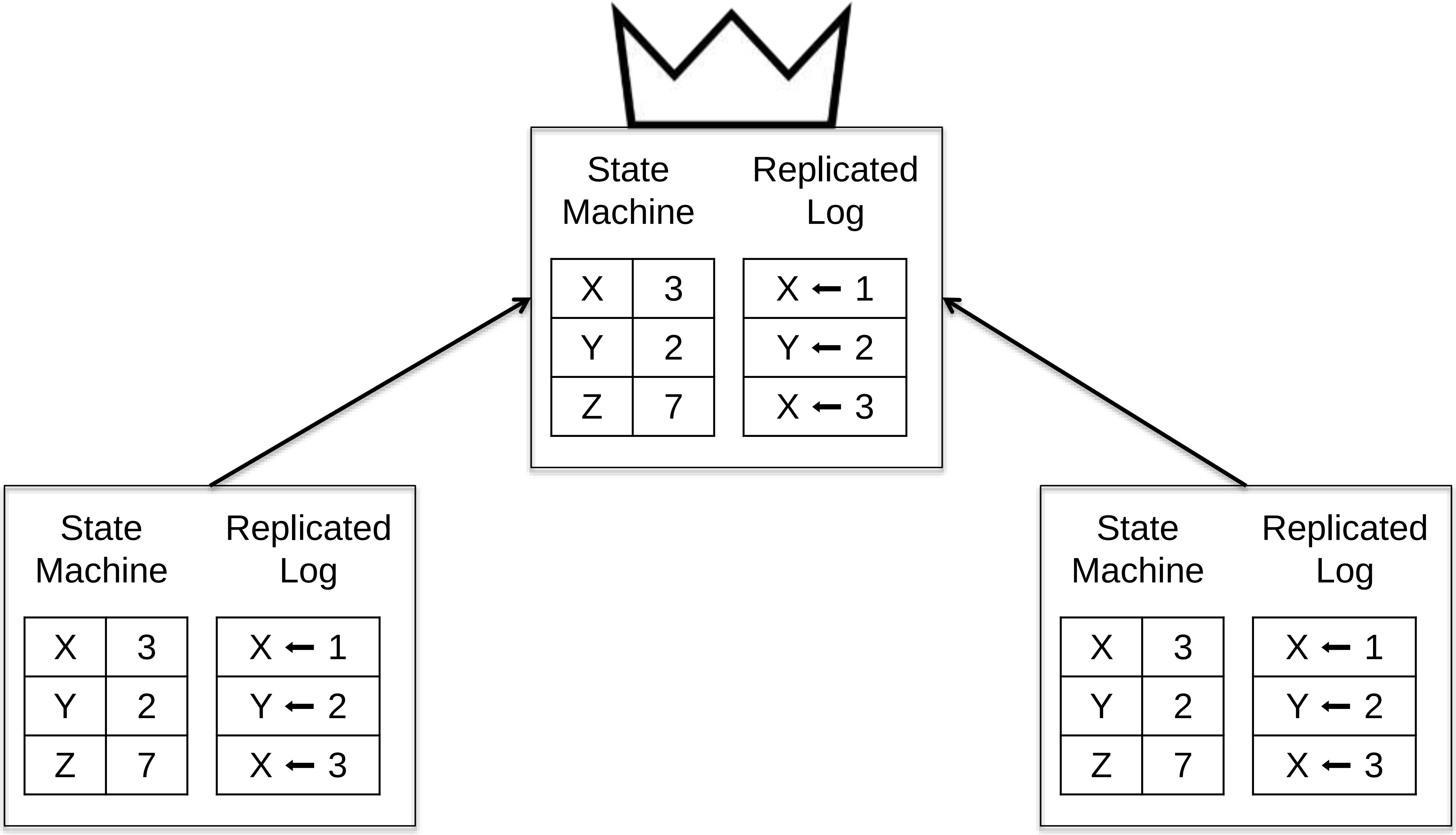
State Machine

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

State Machine

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

# Leader Based Consensus



State Machine

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

Replicated Log

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

State Machine

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

Replicated Log

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

State Machine

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

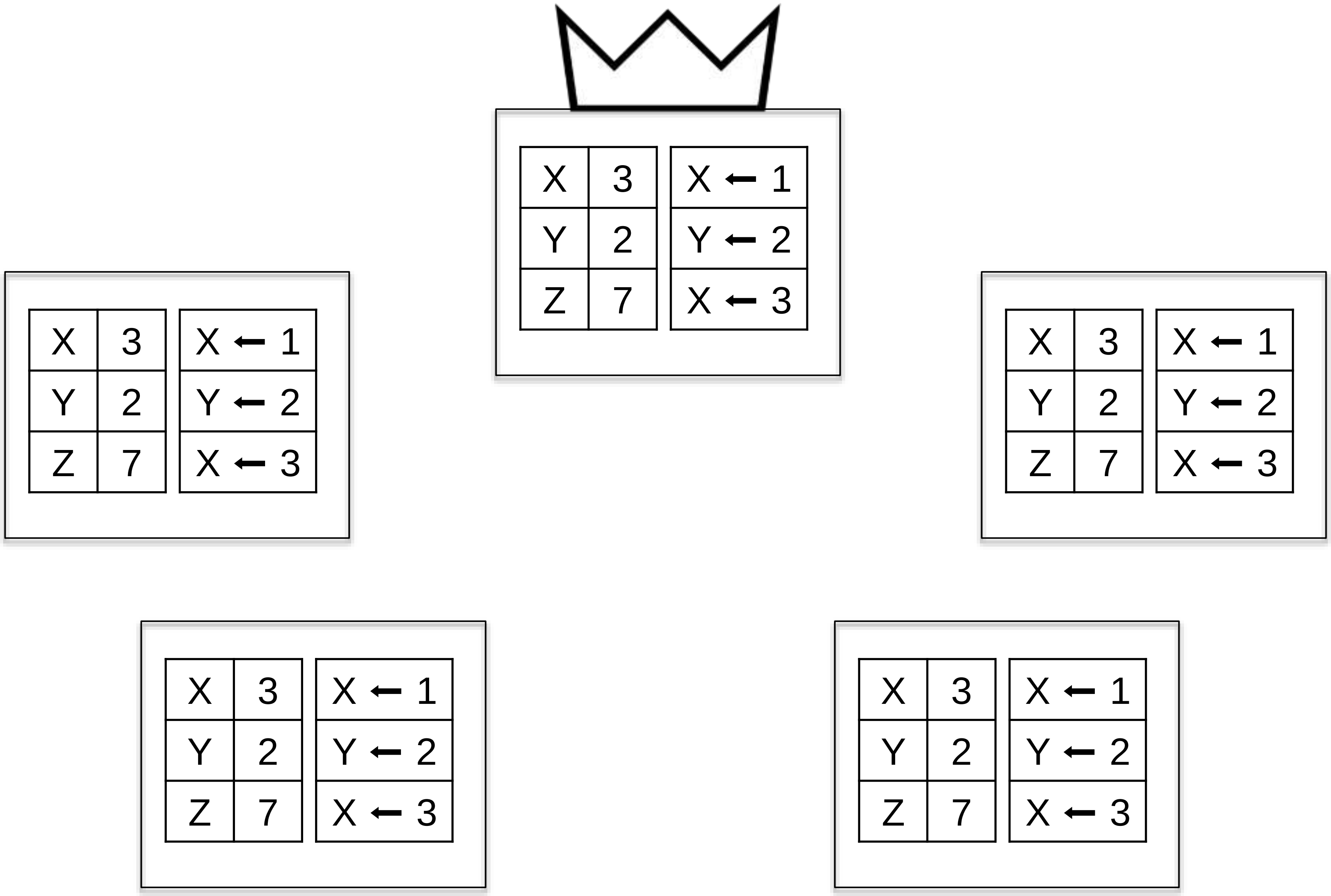
Replicated Log

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

# Agenda

- Introduction to consensus
- Leader-based replicated state machine
- Elections and data replication in MongoDB
- Improvements coming in MongoDB 3.2

# Elections



| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

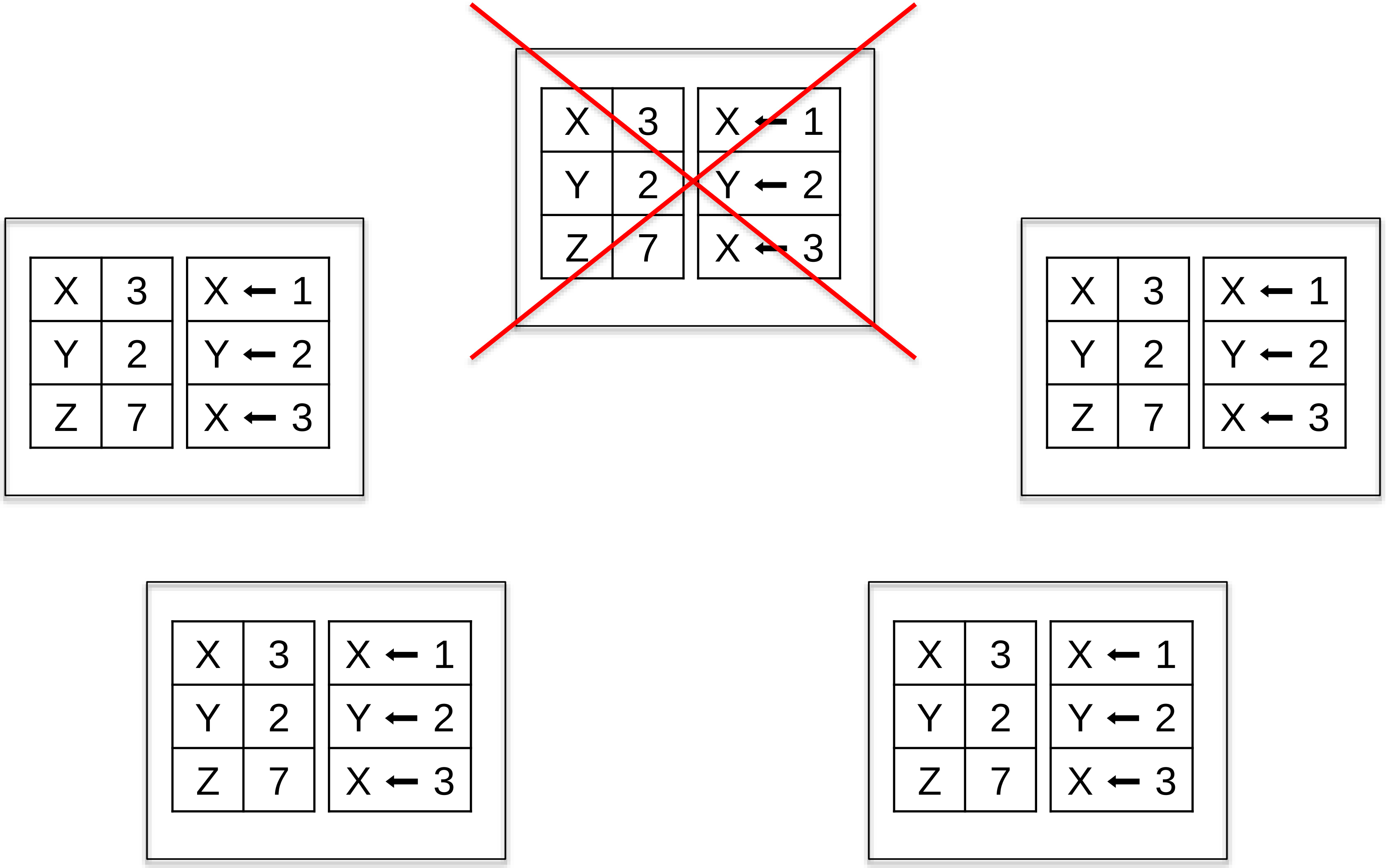
| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

# Elections



| | | |
|---|---|---|
| ~~X~~ | ~~3~~ | ~~X ← 1~~ |
| ~~Y~~ | ~~2~~ | ~~Y ← 2~~ |
| ~~Z~~ | ~~7~~ | ~~X ← 3~~ |

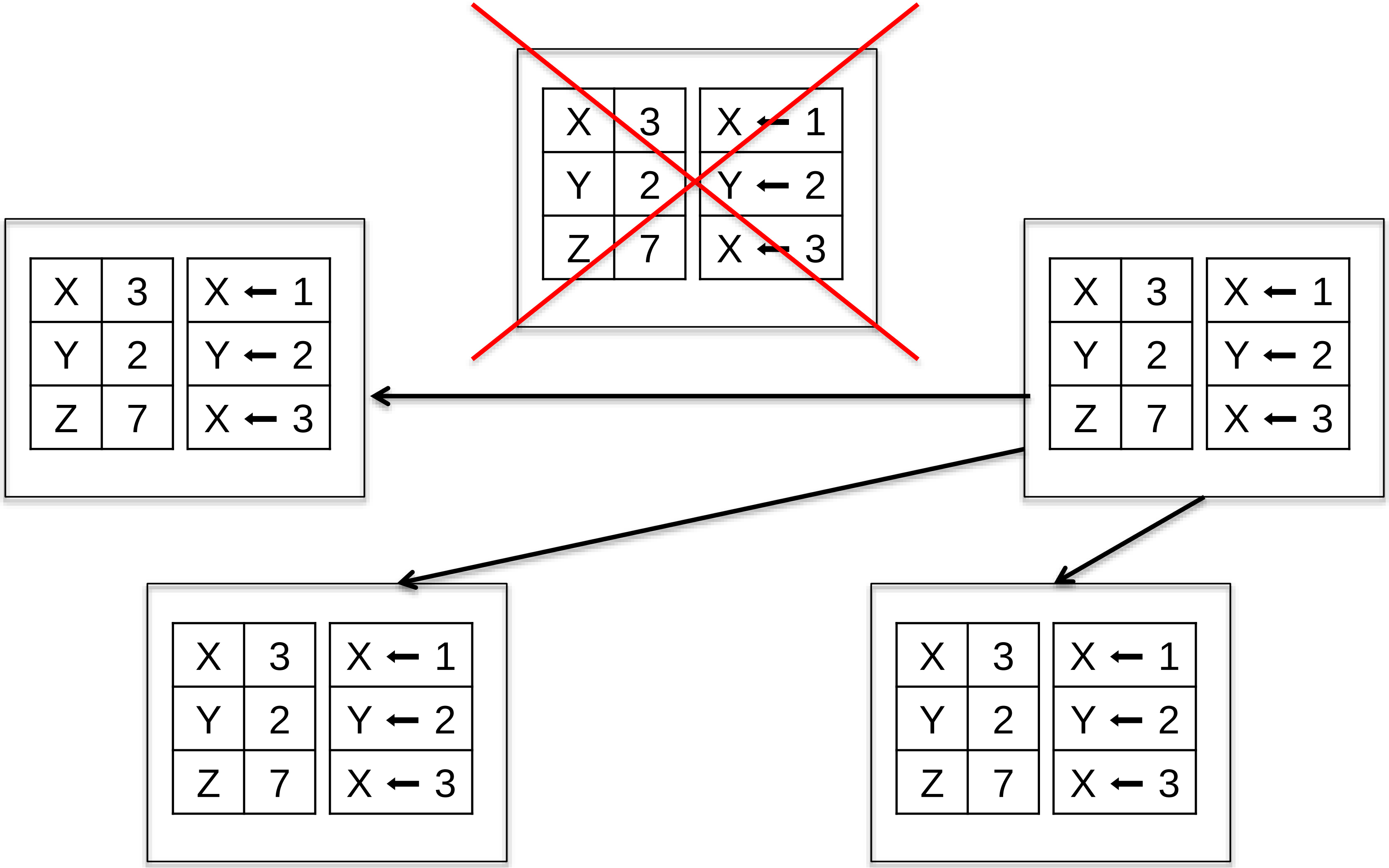
| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

# Elections



| | | | |
|---|---|---|---|
| X | 3 | | X ← 1 |
| Y | 2 | | Y ← 2 |
| Z | 7 | | X ← 3 |

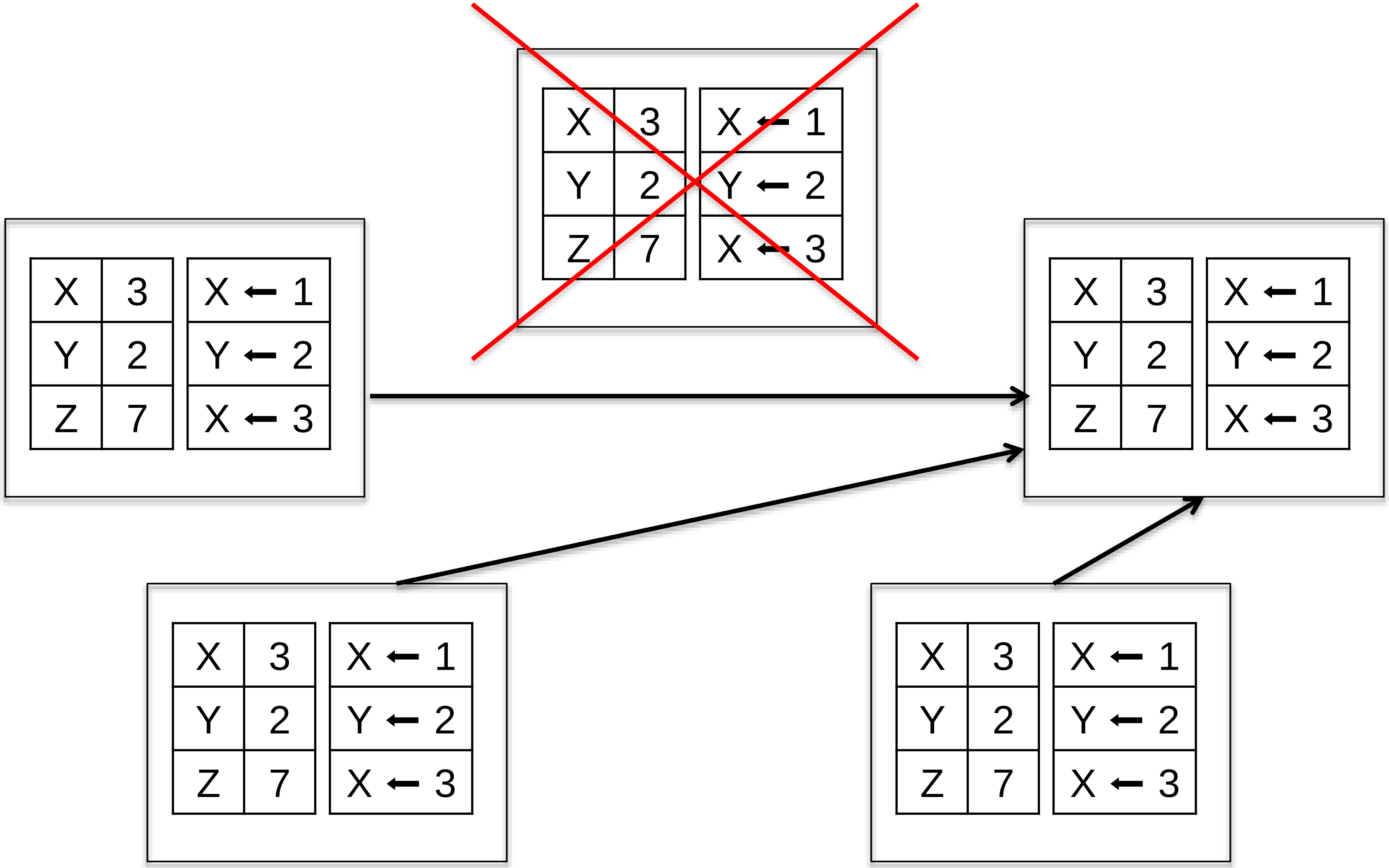
| | | | |
|---|---|---|---|
| X | 3 | | X ← 1 |
| Y | 2 | | Y ← 2 |
| Z | 7 | | X ← 3 |

| | | | |
|---|---|---|---|
| X | 3 | | X ← 1 |
| Y | 2 | | Y ← 2 |
| Z | 7 | | X ← 3 |

| | | | |
|---|---|---|---|
| X | 3 | | X ← 1 |
| Y | 2 | | Y ← 2 |
| Z | 7 | | X ← 3 |

| | | | |
|---|---|---|---|
| X | 3 | | X ← 1 |
| Y | 2 | | Y ← 2 |
| Z | 7 | | X ← 3 |

# Elections



|  |  |  |
|---|---|---|
| ~~X~~ | ~~3~~ | ~~X ← 1~~ |
| ~~Y~~ | ~~2~~ | ~~Y ← 2~~ |
| ~~Z~~ | ~~7~~ | ~~X ← 3~~ |

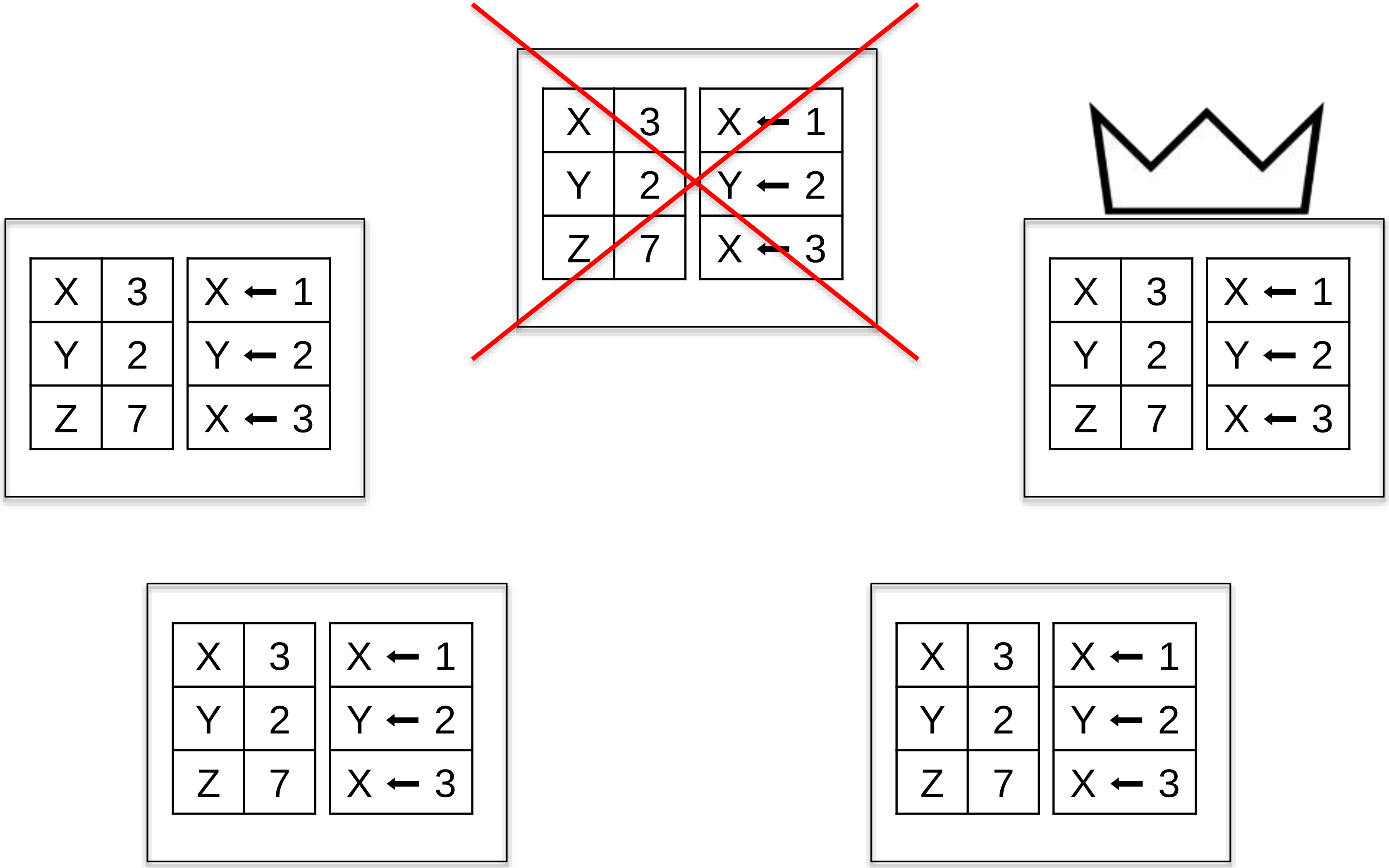
|  |  |  |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

|  |  |  |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

|  |  |  |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

|  |  |  |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

# Elections



| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

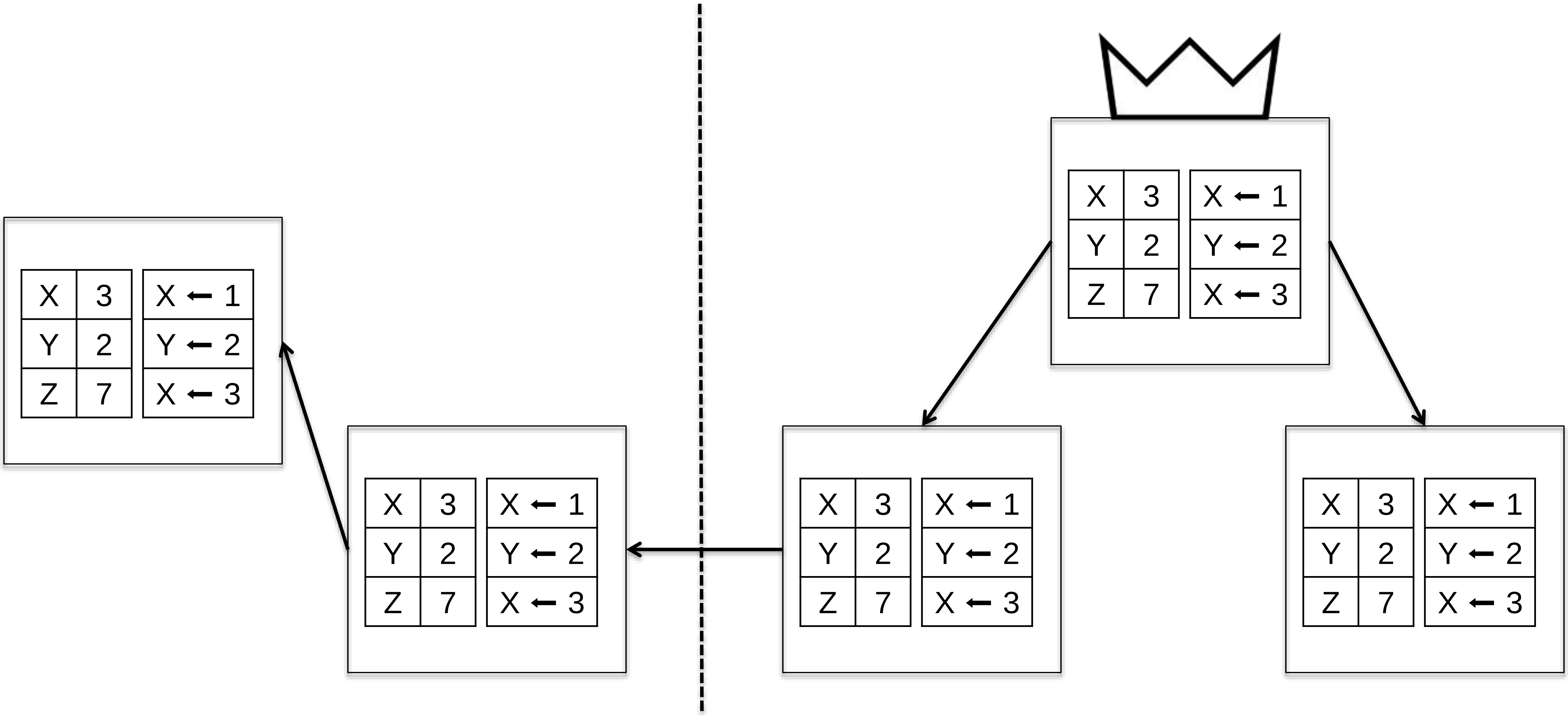
| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

# Data Replication



| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

| | |
|---|---|
| X | 3 |
| Y | 2 |
| Z | 7 |

| |
|---|
| X ← 1 |
| Y ← 2 |
| X ← 3 |

# Agenda

- Introduction to consensus
- Leader-based replicated state machine
- Elections and data replication in MongoDB
- Improvements coming in MongoDB 3.2

# Agenda

- Introduction to consensus
- Leader-based replicated state machine
- Elections and data replication in MongoDB
- Improvements coming in MongoDB 3.2
  - Goals and inspiration from Raft Consensus Algorithm
  - Preventing double voting
  - Monitoring node status
  - Calling for elections

# Goals for MongoDB 3.2

- Decrease failover time
- Speed up detection and resolution of false primary situations

# Finding Inspiration in Raft

- “In Search of an Understandable Consensus Algorithm” by Diego Ongaro: https://ramcloud.stanford.edu/raft.pdf
- Designed to address the shortcomings of Paxos
  - Easier to understand
  - Easier to implement in real applications
- Provably correct
- Remarkably similar to what we’re doing already

# Raft Concepts

- Term (election) IDs
- Monitoring node status using existing data replication channel
- Asymmetric election timeouts

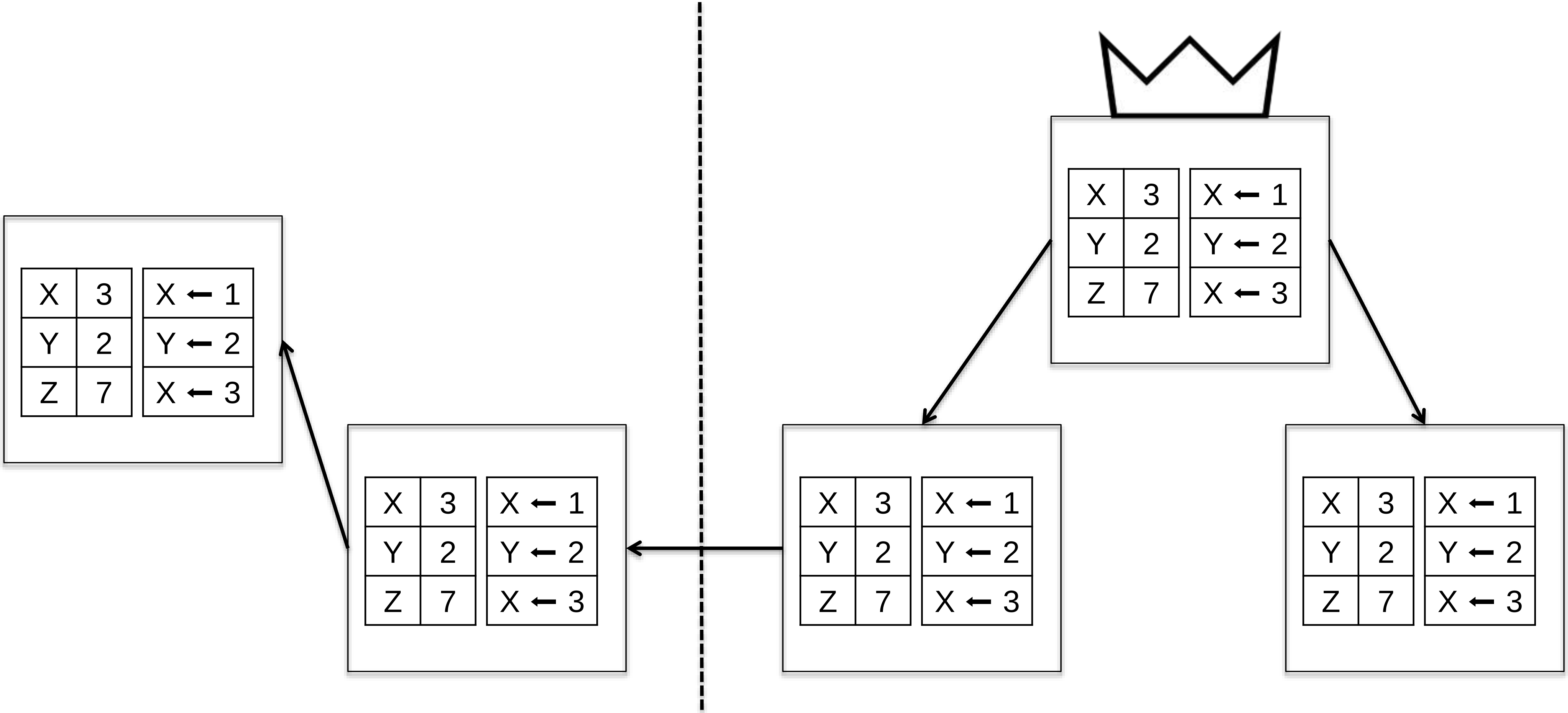
# Preventing Double Voting

- Can’t vote for 2 nodes in the same election
- Pre-3.2: 30 second vote timeout
- Post-3.2: Term IDs
- Term:
    - Monotonically increasing ID
    - Incremented on every election *attempt*
    - Lets voters distinguish elections so they can vote twice quickly in different elections.

# Monitoring Node Status

- Pre-3.2: Heartbeats
  - Sent every two seconds from every node to every other node
  - Volume increases quadratically as nodes are added to the replica set
- Post-3.2: Extra metadata sent via existing data replication channel
  - Utilizes chained replication
  - Faster heartbeats = faster elections and detection of false primaries

# Data Replication



| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

| | | |
|---|---|---|
| X | 3 | X ← 1 |
| Y | 2 | Y ← 2 |
| Z | 7 | X ← 3 |

# Determining When To Call For An Election

- Tradeoff between failover time and spurious failovers
- Node calls for an election when it hasn’t heard from the primary within the election timeout
- Starting in 3.2:
  - Election timeout is configurable
  - Election timeout is varied randomly for each node
  - Varying timeouts help reduce tied votes
  - Fewer tied votes = faster failover

# Conclusion

- MongoDB 3.2 will have
    - Faster failovers
    - Faster error detection
    - More control to prevent spurious failovers
- This means your systems are
    - More stable
    - More resilient to failure
    - Easier to maintain

# Questions?