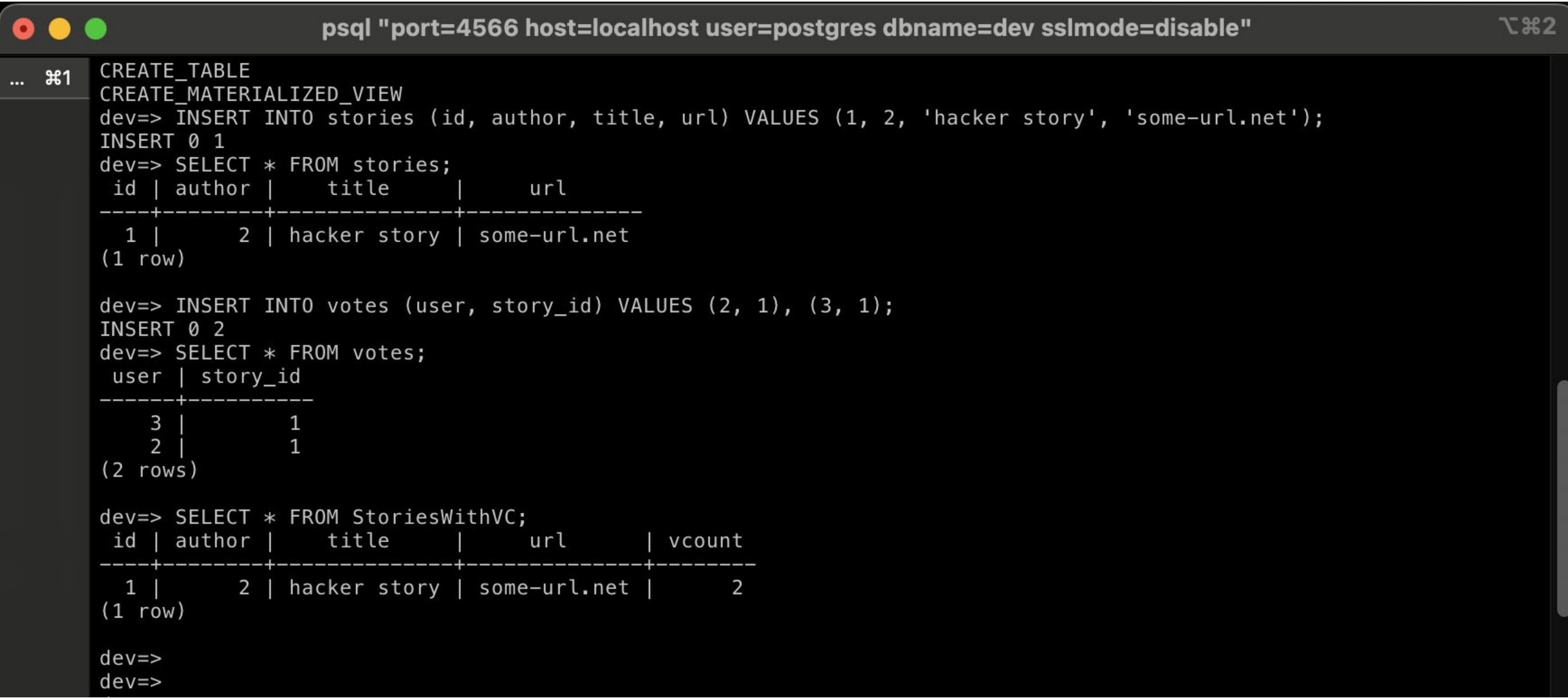
The wonderful life of a SQL query in a streaming database

RisingWave Labs

- Creator of the RisingWave Database
- OLAP streaming queries
- Incremental updates on materialized views

Views and Materialized Views

- What is a view?
- What is a MV?
- How do you calc views traditionally?
  - Full rebuild is expensive
- Incremental updates
  - Run aggregate on diff
  - Run background job that detects changes in base table
  - Triggers that fire if there is a change to the underlying table
- RisingWave: Incremental updates



psql "port=4566 host=localhost user=postgres dbname=dev sslmode=disable"
⌥⌘2
... ⌘1

```
CREATE_TABLE
CREATE_MATERIALIZED_VIEW
dev=> INSERT INTO stories (id, author, title, url) VALUES (1, 2, 'hacker story', 'some-url.net');
INSERT 0 1
dev=> SELECT * FROM stories;
 id | author |    title     |     url
----+--------+--------------+--------------
  1 |      2 | hacker story | some-url.net
(1 row)

dev=> INSERT INTO votes (user, story_id) VALUES (2, 1), (3, 1);
INSERT 0 2
dev=> SELECT * FROM votes;
 user | story_id
------+----------
    3 |        1
    2 |        1
(2 rows)

dev=> SELECT * FROM StoriesWithVC;
 id | author |    title     |     url      | vcount
----+--------+--------------+--------------+--------
  1 |      2 | hacker story | some-url.net |      2
(1 row)

dev=>
dev=>
```

Streaming graph

```
CREATE TABLE stories (id int, author int, title text, url text);
CREATE TABLE votes (user int, story_id int);

CREATE MATERIALIZED VIEW StoriesWithVC AS
SELECT id, author, title, url, vcount
FROM stories
JOIN ( SELECT story_id, COUNT(*) AS vcount FROM votes GROUP BY story_id) as VoteCount
on VoteCount.story_id = stories.id;
```

```
EXPLAIN CREATE MATERIALIZED VIEW StoriesWithVC AS
SELECT id, author, title, url, vcount
FROM stories JOIN (SELECT story_id, COUNT(*) AS vcount FROM votes GROUP BY story_id) AS VoteCount
ON VoteCount.story_id = stories.id;
-------------------------------------------------------------------------------------------------
 StreamMaterialize { columns: [id, author, title, url, vcount, ...] }
 └─StreamExchange
   └─StreamHashJoin { type: Inner, predicate: stories.id = votes.story_id }
     ├─StreamExchange { dist: HashShard(stories.id) }
     │ └─StreamTableScan { table: stories, columns: [id, author, title, url, _row_id] }
     └─StreamHashAgg { group_key: [votes.story_id], aggs: [count] }
       └─StreamExchange { dist: HashShard(votes.story_id) }
         └─StreamTableScan { table: votes, columns: [story_id, _row_id] }
```

```
EXPLAIN CREATE MATERIALIZED VIEW StoriesWithVC AS
SELECT id, author, title, url, vcount
FROM stories JOIN (SELECT story_id, COUNT(*) AS vcount FROM votes GROUP BY story_id) AS VoteCount
ON VoteCount.story_id = stories.id;
-------------------------------------------------------------------------------------------------
 StreamMaterialize { columns: [id, author, title, url, vcount, ...] }
 └─StreamExchange
   └─StreamHashJoin { type: Inner, predicate: stories.id = votes.story_id }
     ├─StreamExchange { dist: HashShard(stories.id) }
     │ └─StreamTableScan { table: stories, columns: [id, author, title, url, _row_id] }
     └─StreamHashAgg { group_key: [votes.story_id], aggs: [count] }
       └─StreamExchange { dist: HashShard(votes.story_id) }
         └─StreamTableScan { table: votes, columns: [story_id, _row_id] }
```

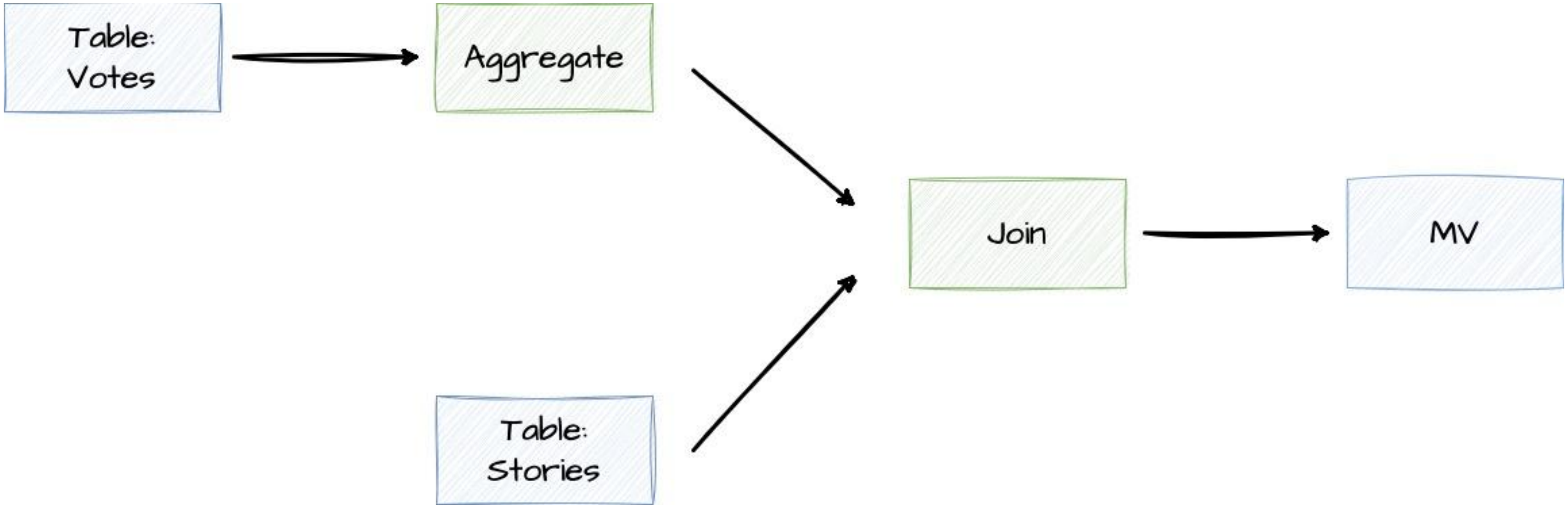
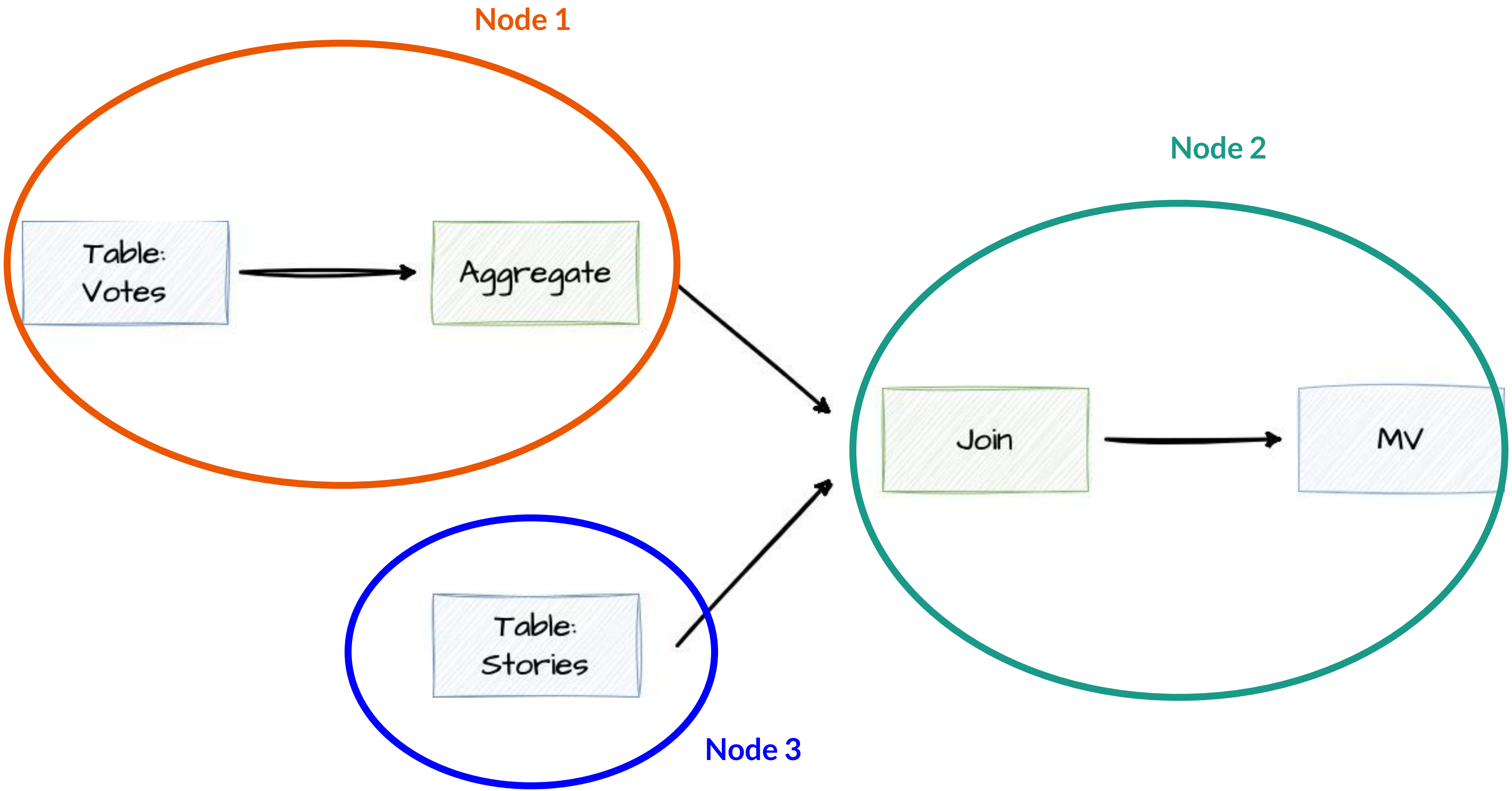


Table: Votes
Aggregate
Join
MV
Table: Stories



Node 1

Node 2

Node 3

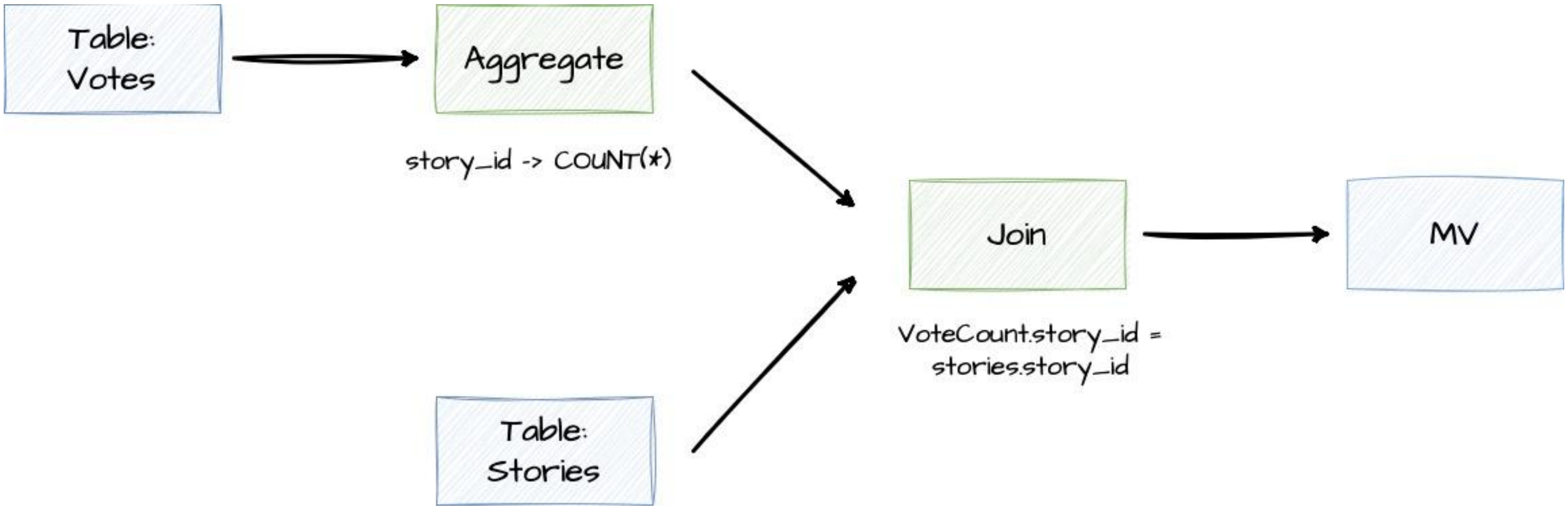


Table: Votes

Aggregate

story_id -> COUNT(*)

Table: Stories

Join

VoteCount.story_id = stories.story_id

MV

user	story_id
1234	1003
2345	1004
9999	1003

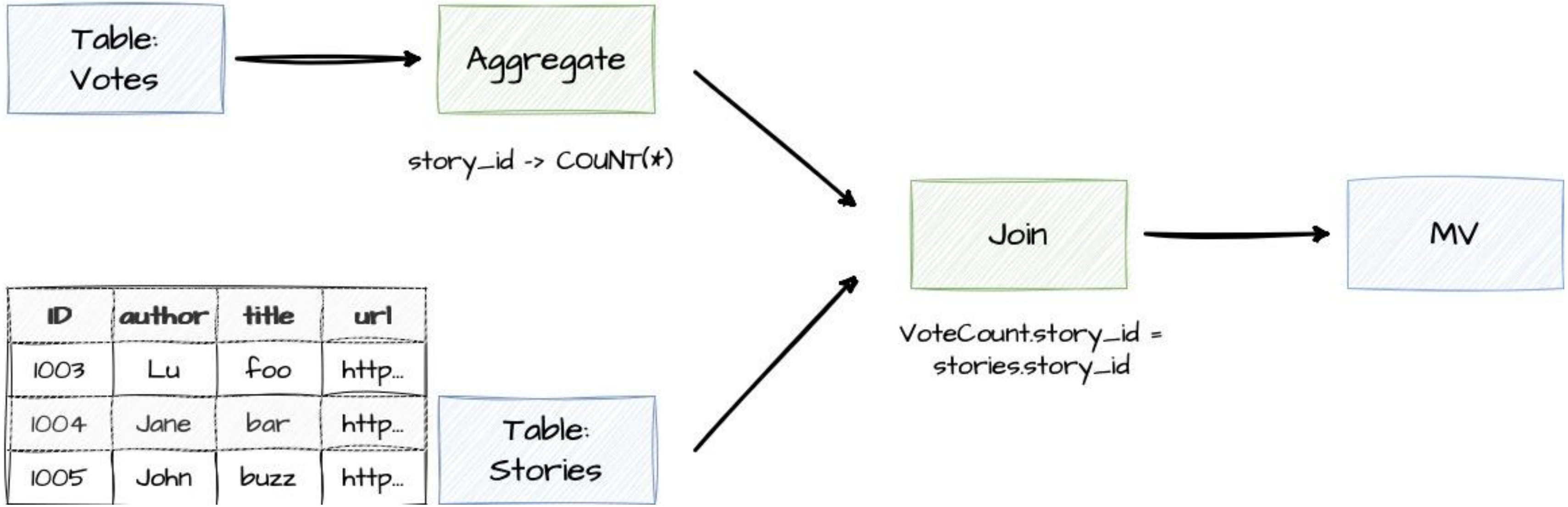


Table: Votes

Aggregate

story_id -> COUNT(*)

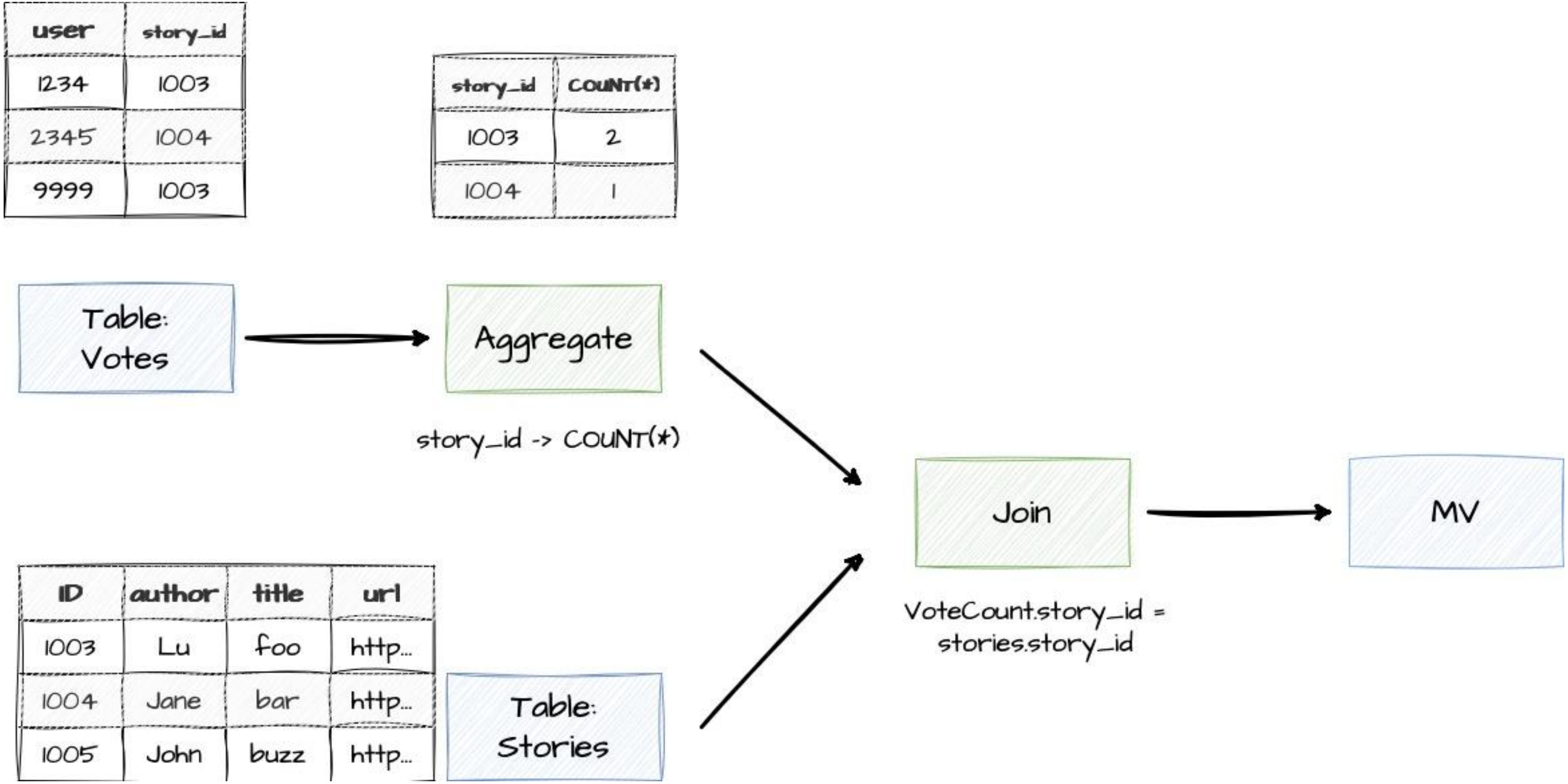
Table: Stories

Join

VoteCount.story_id = stories.story_id

MV

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...



user	story_id
1234	1003
2345	1004
9999	1003

story_id	COUNT(*)
1003	2
1004	1

Table: Votes

Aggregate

story_id -> COUNT(*)

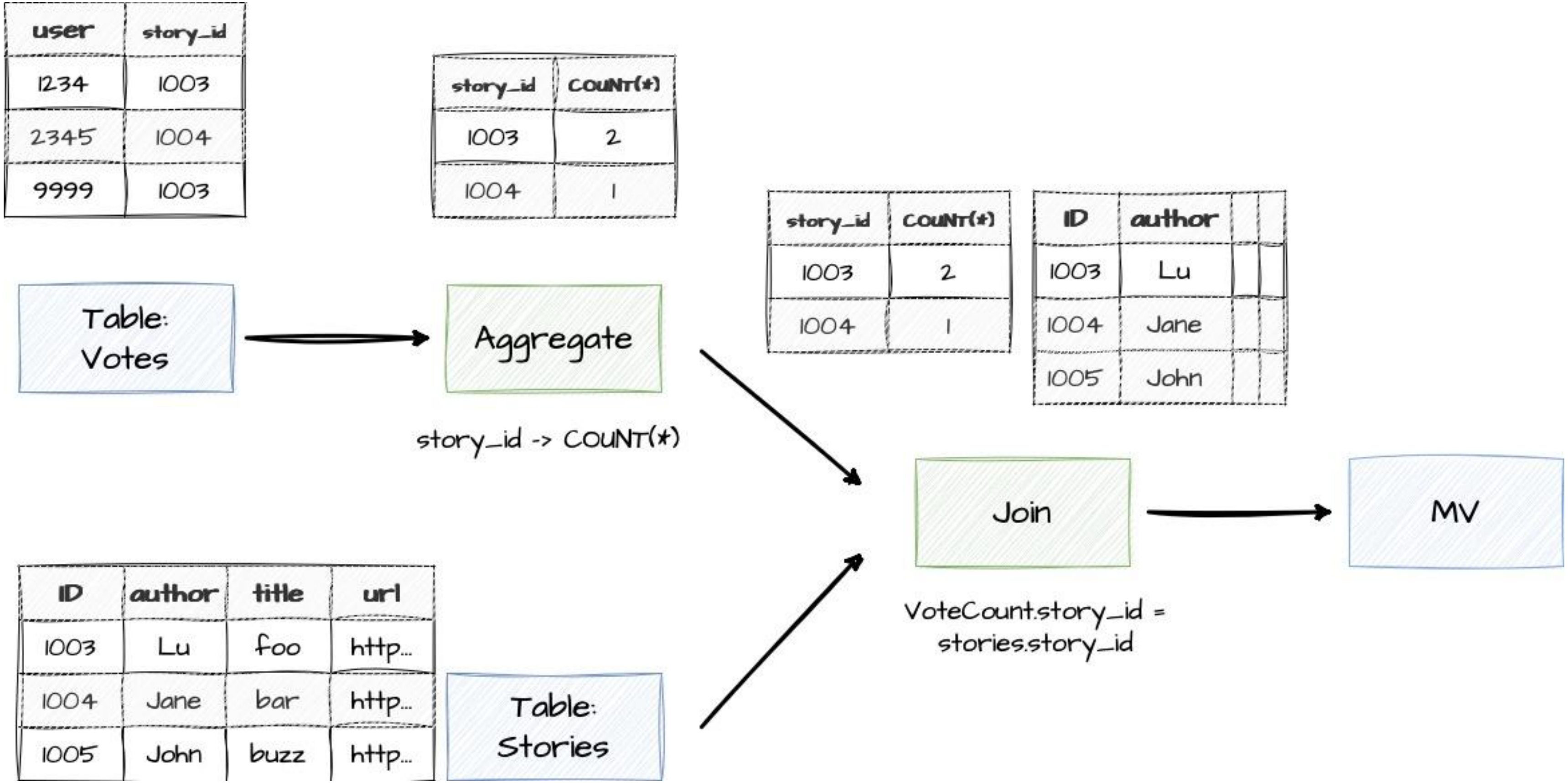
Join

VoteCount.story_id = stories.story_id

MV

Table: Stories

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...



user	story_id
1234	1003
2345	1004
9999	1003

story_id	COUNT(*)
1003	2
1004	1

Table: Votes

Aggregate

story_id -> COUNT(*)

story_id	COUNT(*)
1003	2
1004	1

ID	author		
1003	Lu		
1004	Jane		
1005	John		

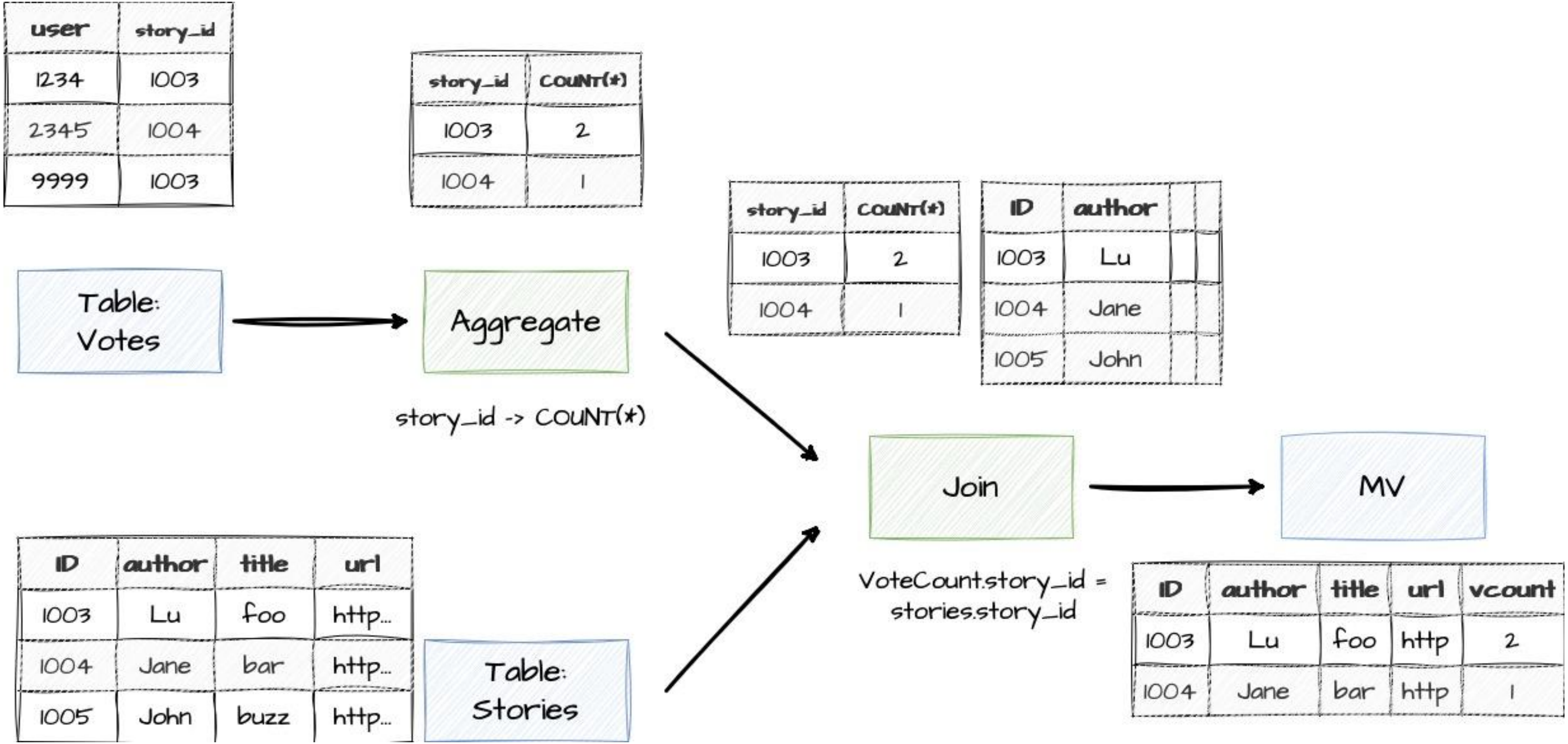
Join

VoteCount.story_id =
stories.story_id

MV

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

Table: Stories



user	story_id
1234	1003
2345	1004
9999	1003

Table: Votes

Aggregate

story_id -> COUNT(*)

story_id	COUNT(*)
1003	2
1004	1

story_id	COUNT(*)
1003	2
1004	1

ID	author		
1003	Lu		
1004	Jane		
1005	John		

Join

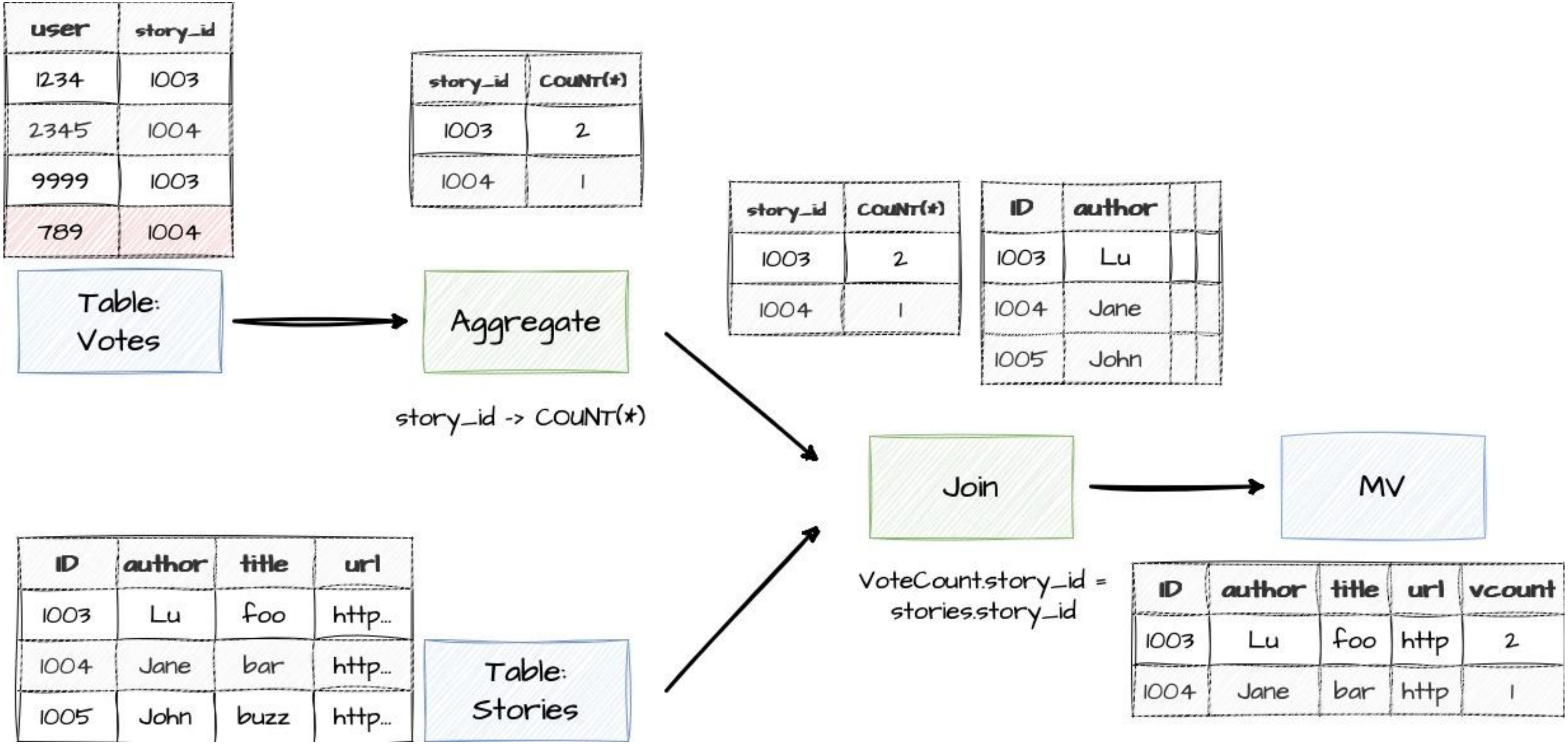
VoteCount.story_id =
stories.story_id

MV

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

Table: Stories

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	1



user	story_id
1234	1003
2345	1004
9999	1003
789	1004

Table: Votes

story_id	COUNT(*)
1003	2
1004	1

Aggregate

story_id -> COUNT(*)

story_id	COUNT(*)
1003	2
1004	1

ID	author		
1003	Lu		
1004	Jane		
1005	John		

Join

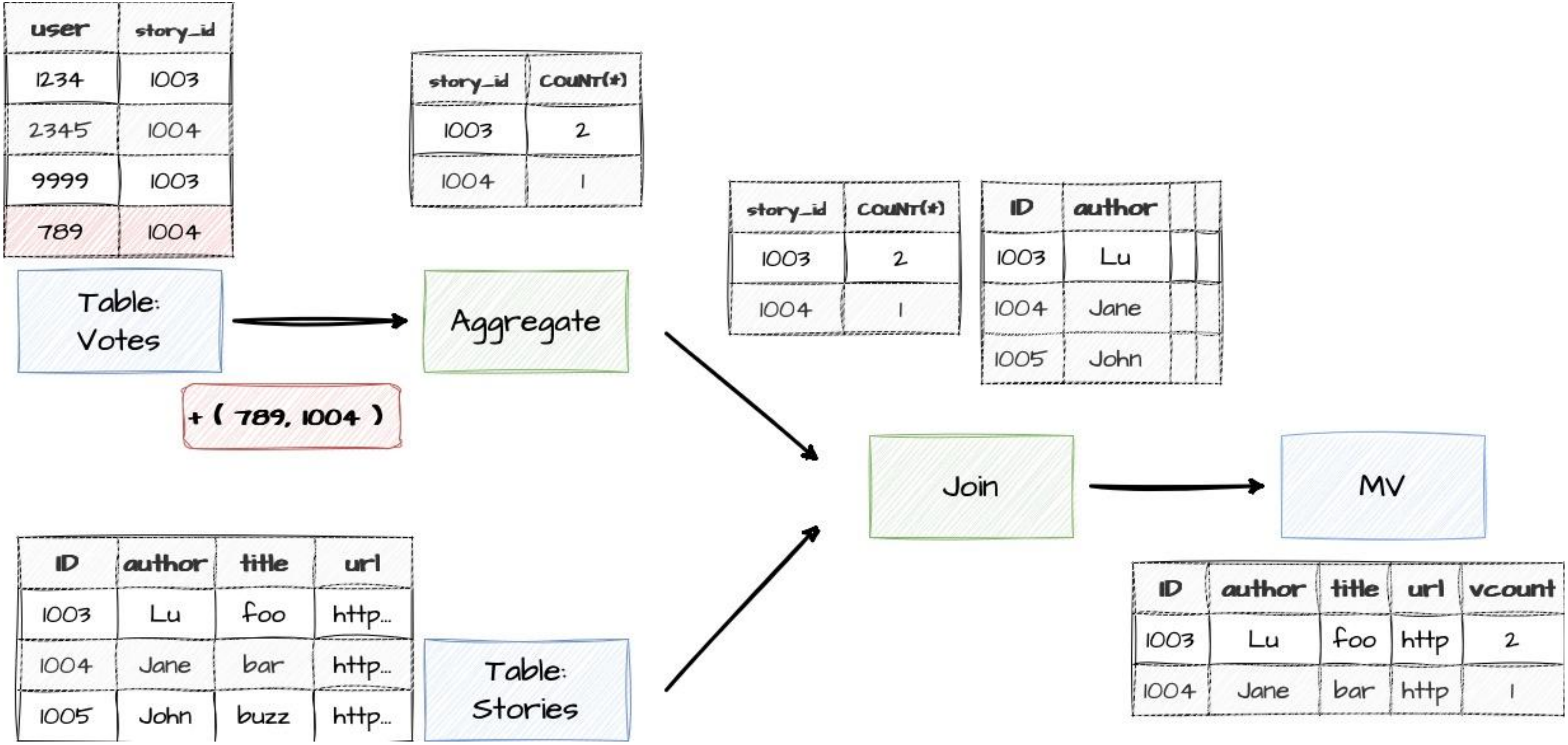
VoteCount.story_id =
stories.story_id

MV

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

Table: Stories

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	1



user	story_id
1234	1003
2345	1004
9999	1003
789	1004

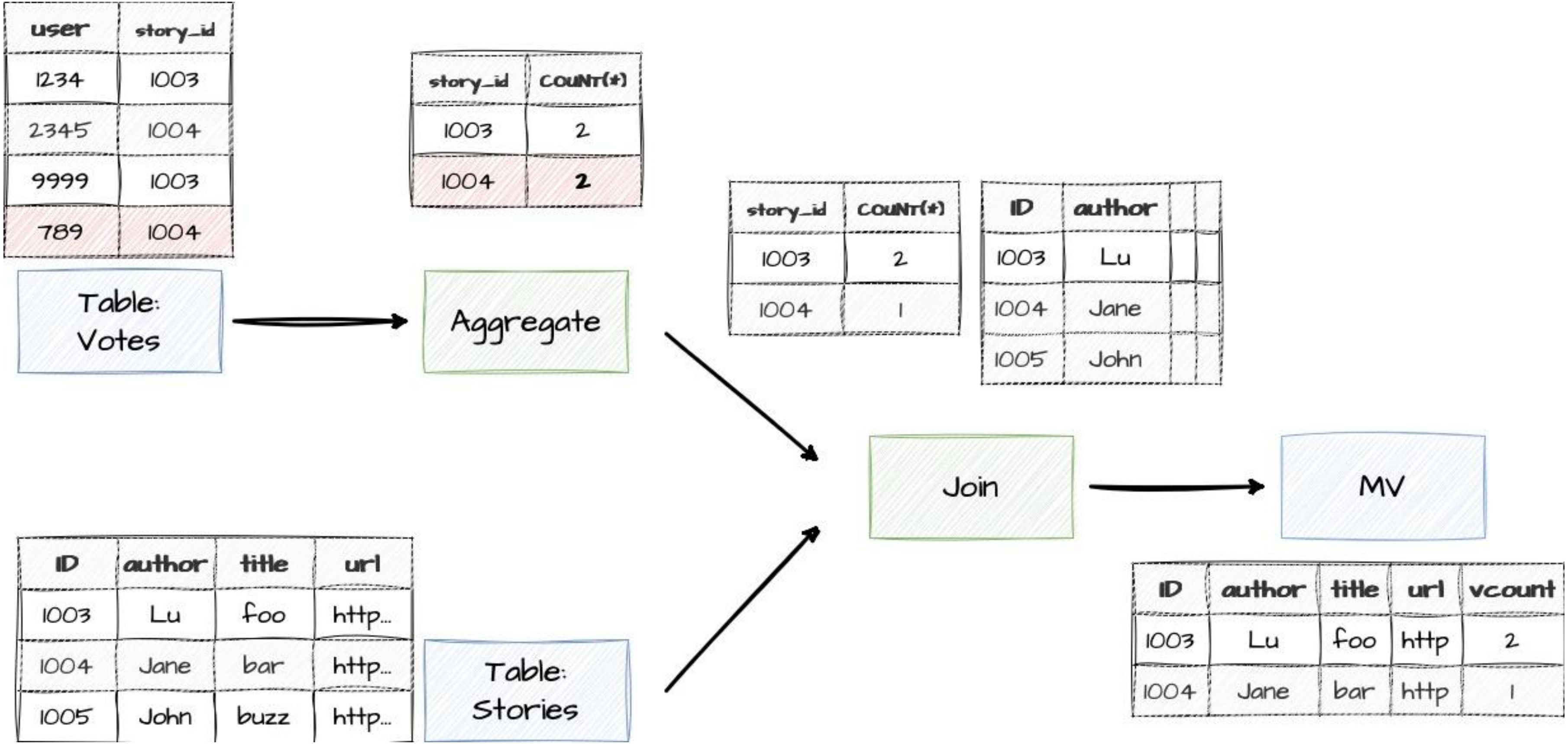
story_id	COUNT(*)
1003	2
1004	1

story_id	COUNT(*)
1003	2
1004	1

ID	author		
1003	Lu		
1004	Jane		
1005	John		

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	1



user	story_id
1234	1003
2345	1004
9999	1003
789	1004

Table: Votes

Aggregate

story_id	COUNT(*)
1003	2
1004	**2**

story_id	COUNT(*)
1003	2
1004	1

ID	author		
1003	Lu		
1004	Jane		
1005	John		

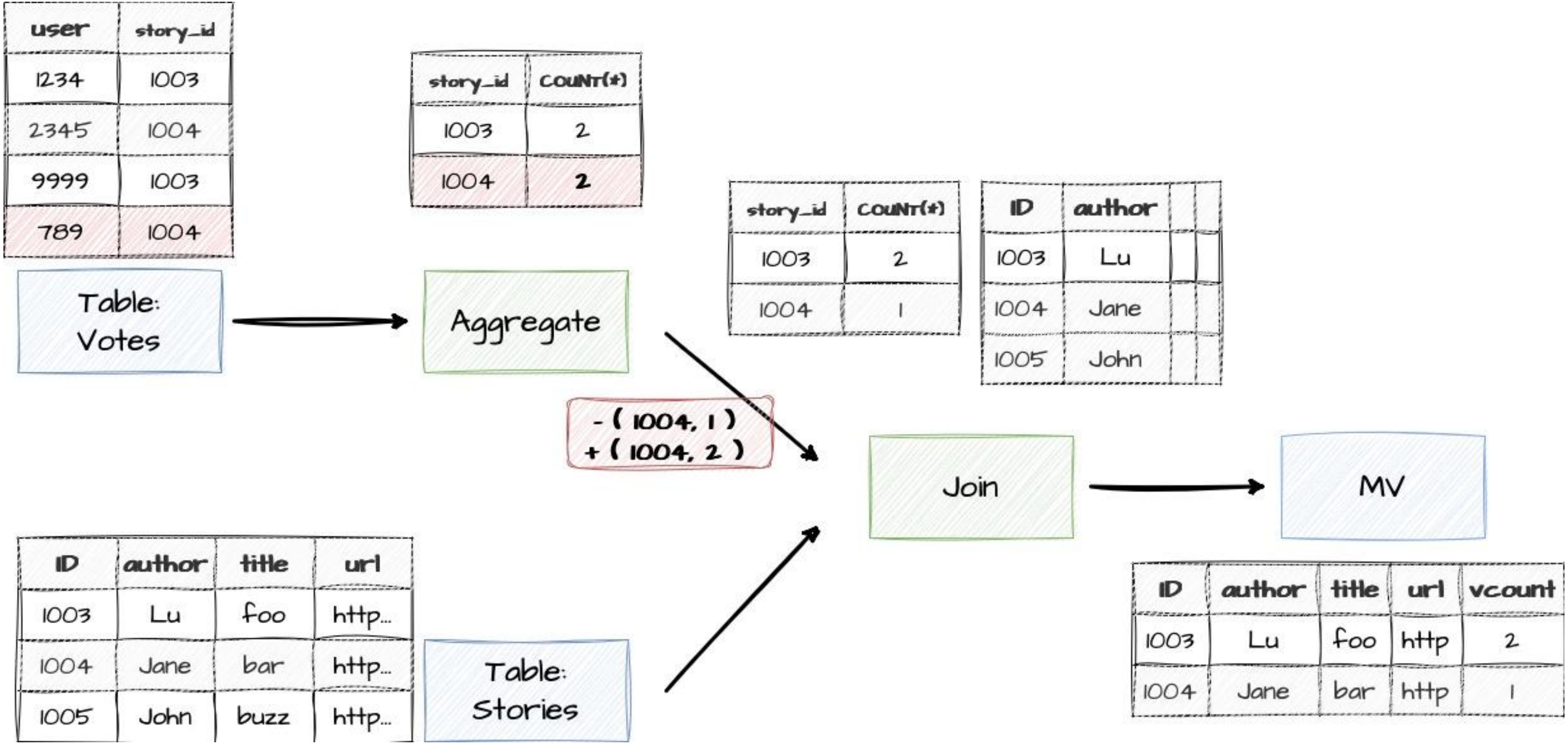
ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

Table: Stories

Join

MV

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	1



user	story_id
1234	1003
2345	1004
9999	1003
789	1004

Table: Votes

Aggregate

story_id	COUNT(*)
1003	2
1004	2

- (1004, 1)
+ (1004, 2)

story_id	COUNT(*)
1003	2
1004	1

ID	author		
1003	Lu		
1004	Jane		
1005	John		

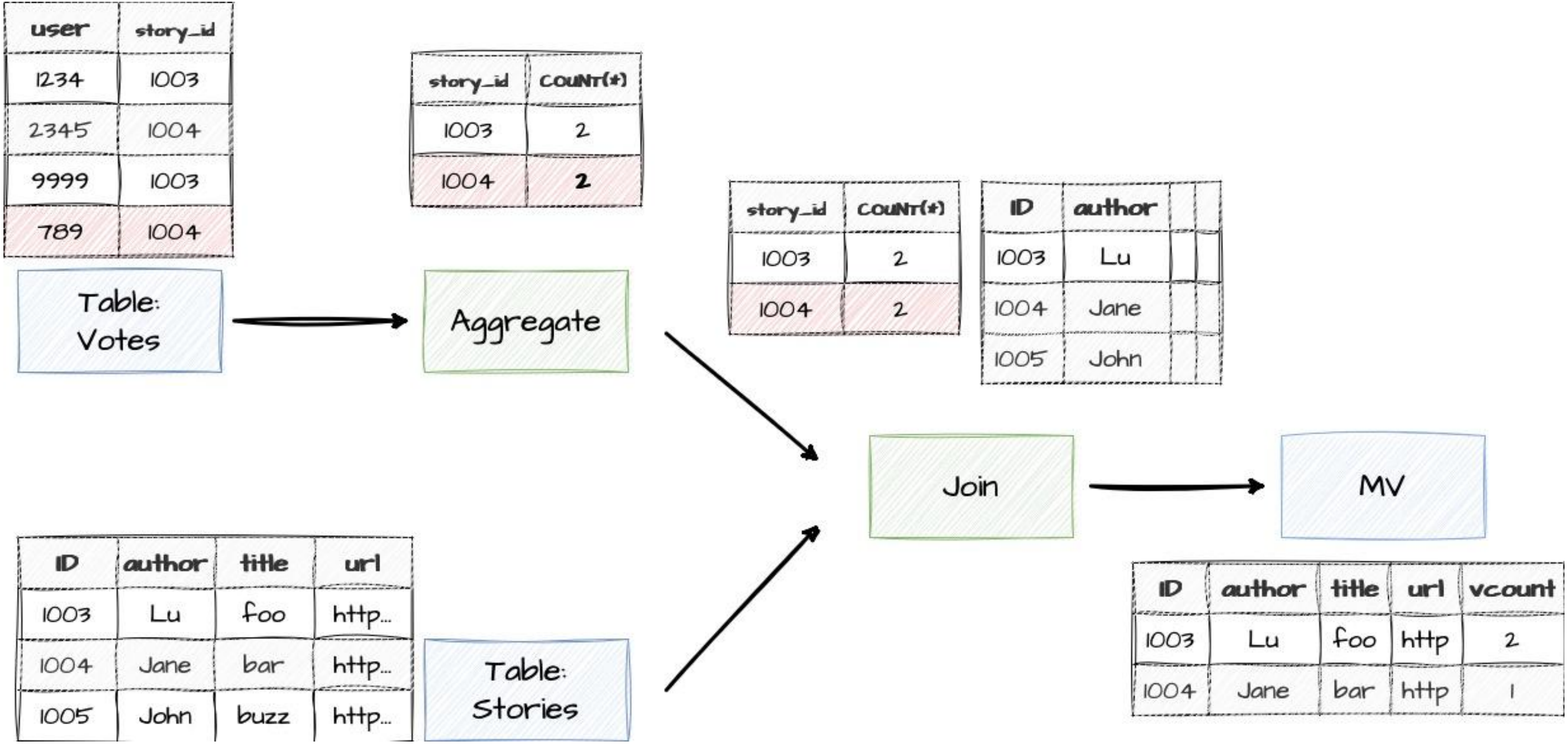
Join

MV

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	1

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

Table: Stories



user	story_id
1234	1003
2345	1004
9999	1003
789	1004

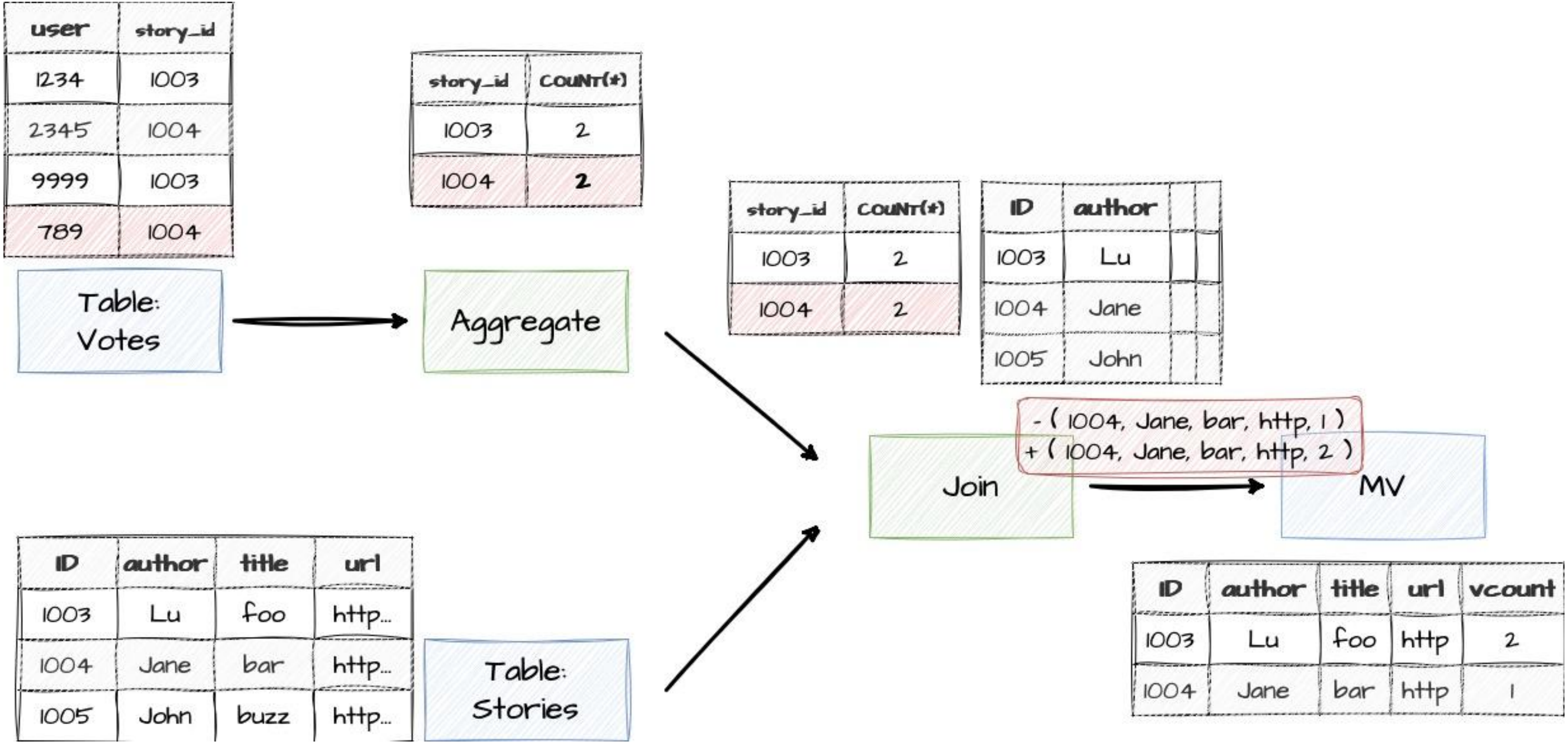
story_id	COUNT(*)
1003	2
1004	**2**

story_id	COUNT(*)
1003	2
1004	2

ID	author		
1003	Lu		
1004	Jane		
1005	John		

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	1



user	story_id
1234	1003
2345	1004
9999	1003
789	1004

story_id	COUNT(*)
1003	2
1004	2

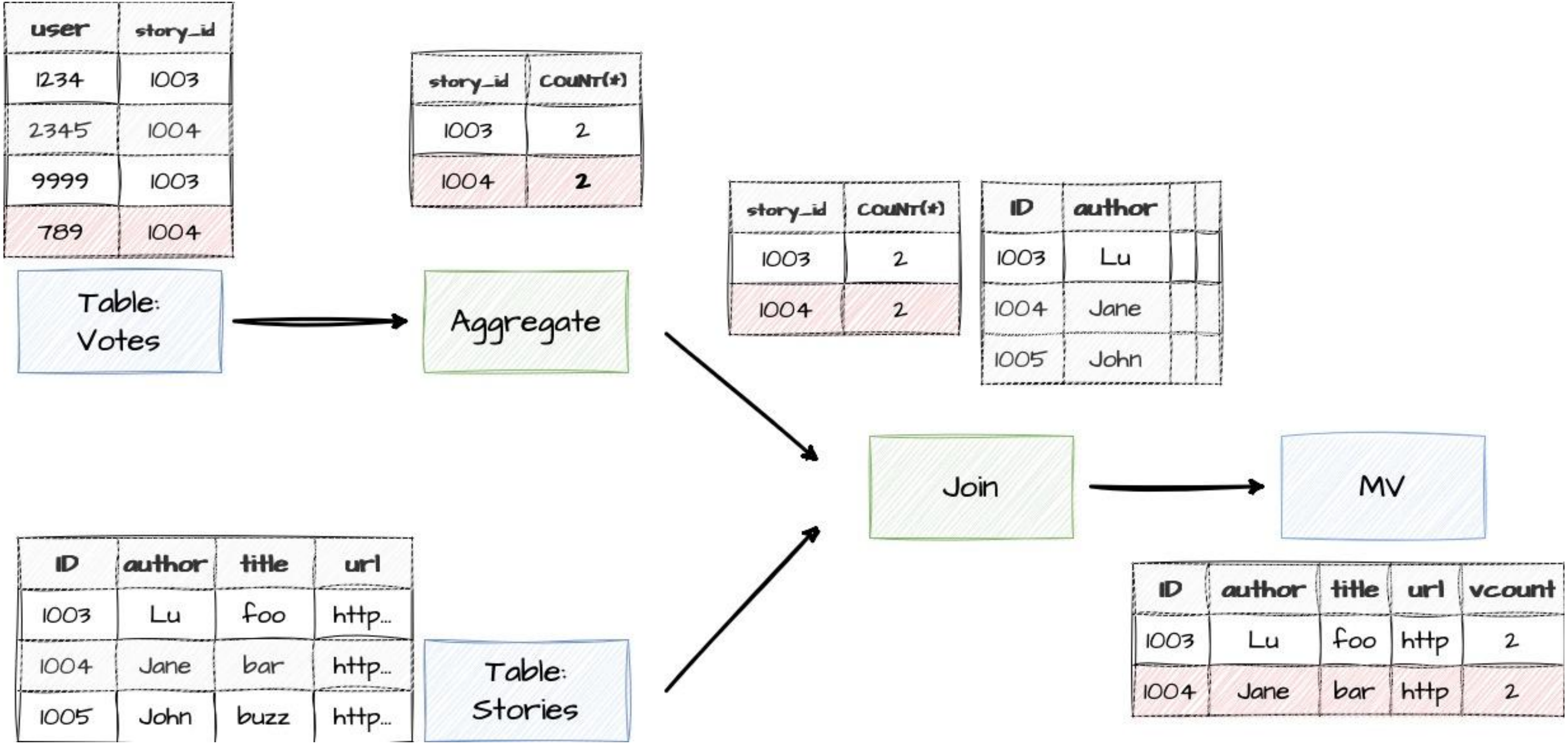
story_id	COUNT(*)
1003	2
1004	2

ID	author		
1003	Lu		
1004	Jane		
1005	John		

- (1004, Jane, bar, http, 1)
+ (1004, Jane, bar, http, 2)

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	1



user	story_id
1234	1003
2345	1004
9999	1003
789	1004

story_id	COUNT(*)
1003	2
1004	**2**

story_id	COUNT(*)
1003	2
1004	2

ID	author
1003	Lu
1004	Jane
1005	John

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	2

Distributed systems

Challenges and opportunities in a distributed setup

- Opportunity:
  - Execute in parallel
- Challenges:
  - **Recovery:** Trying not to lose data when a node crashes
  - **Scalability:** Adding/removing nodes if you have more/less workloads

Distributed systems Parallelism

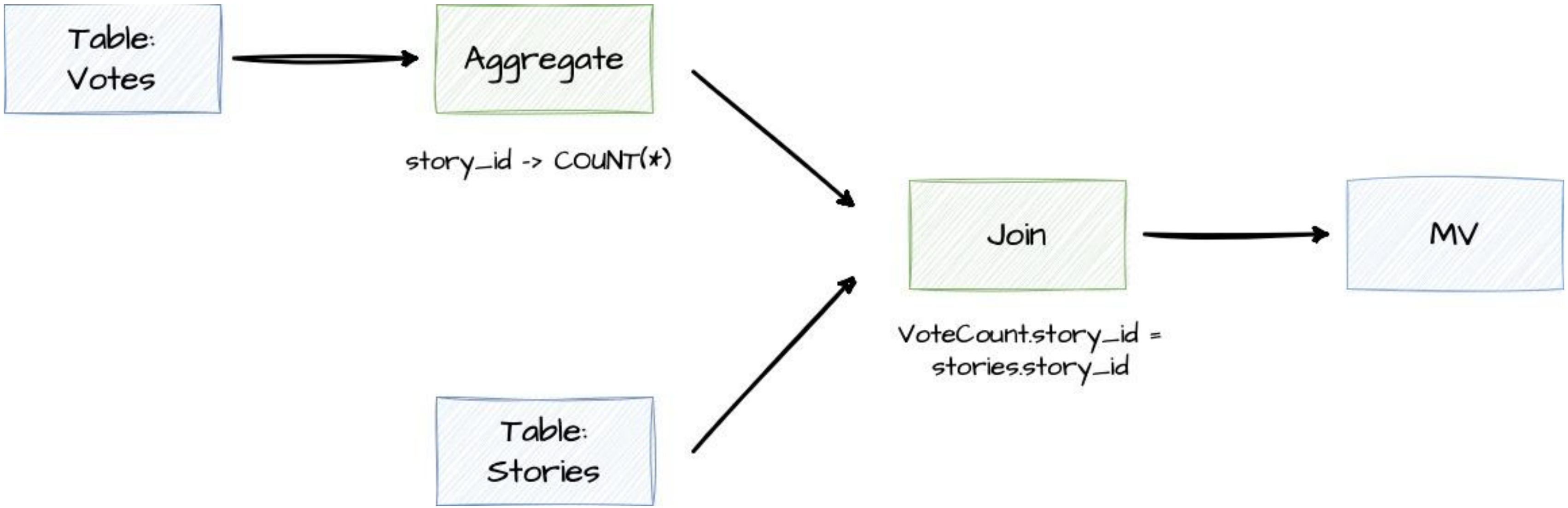
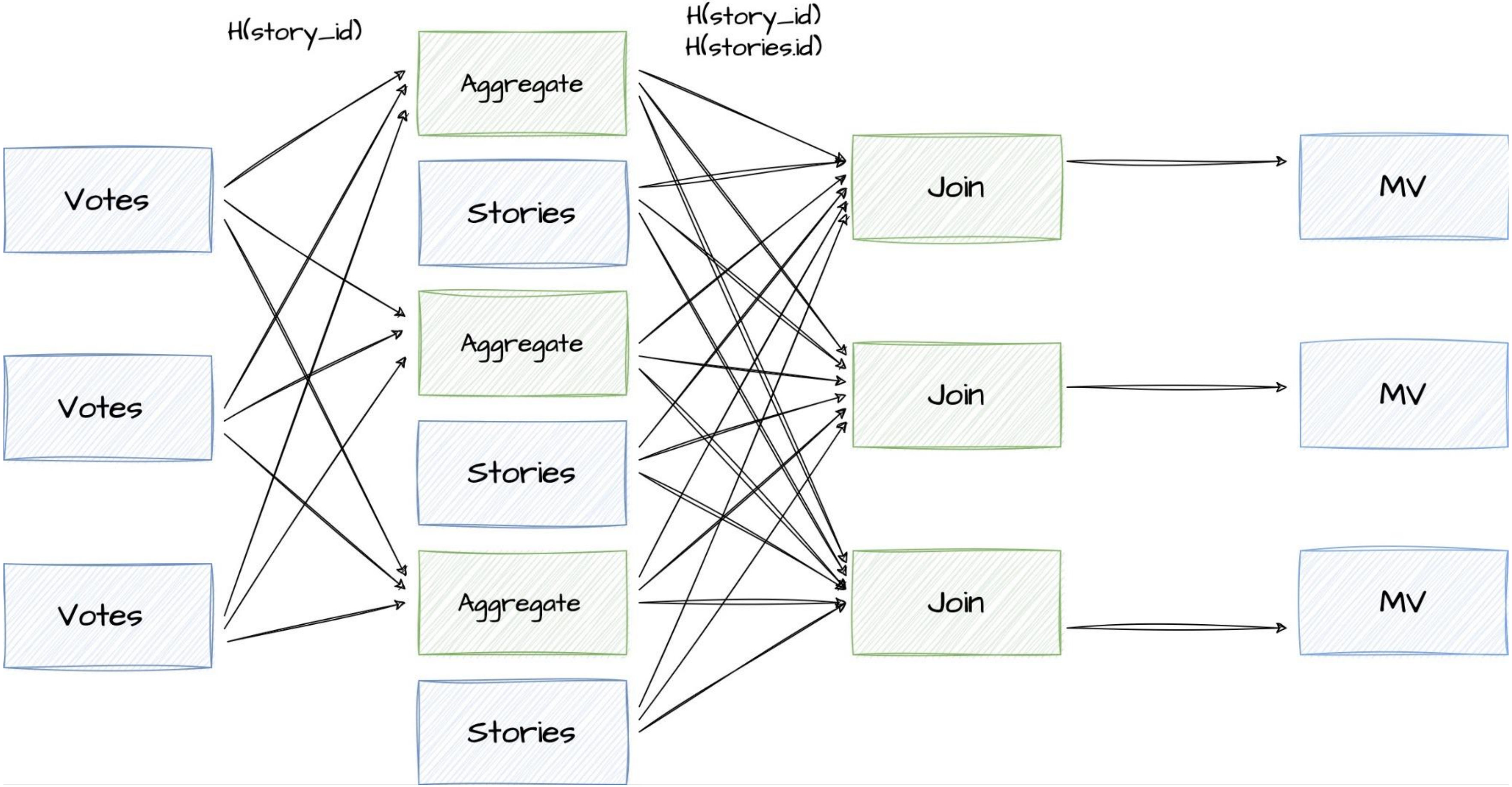
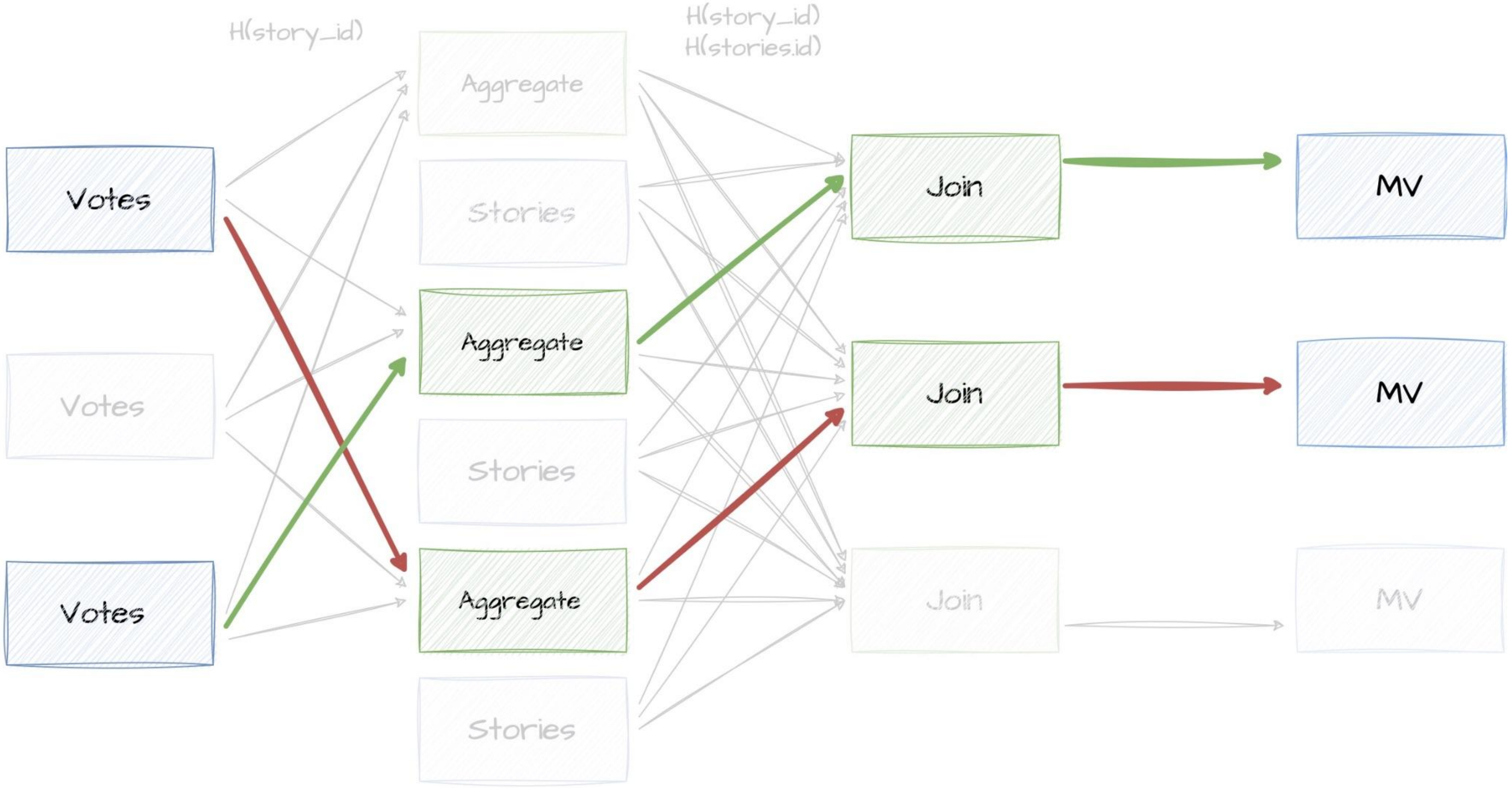
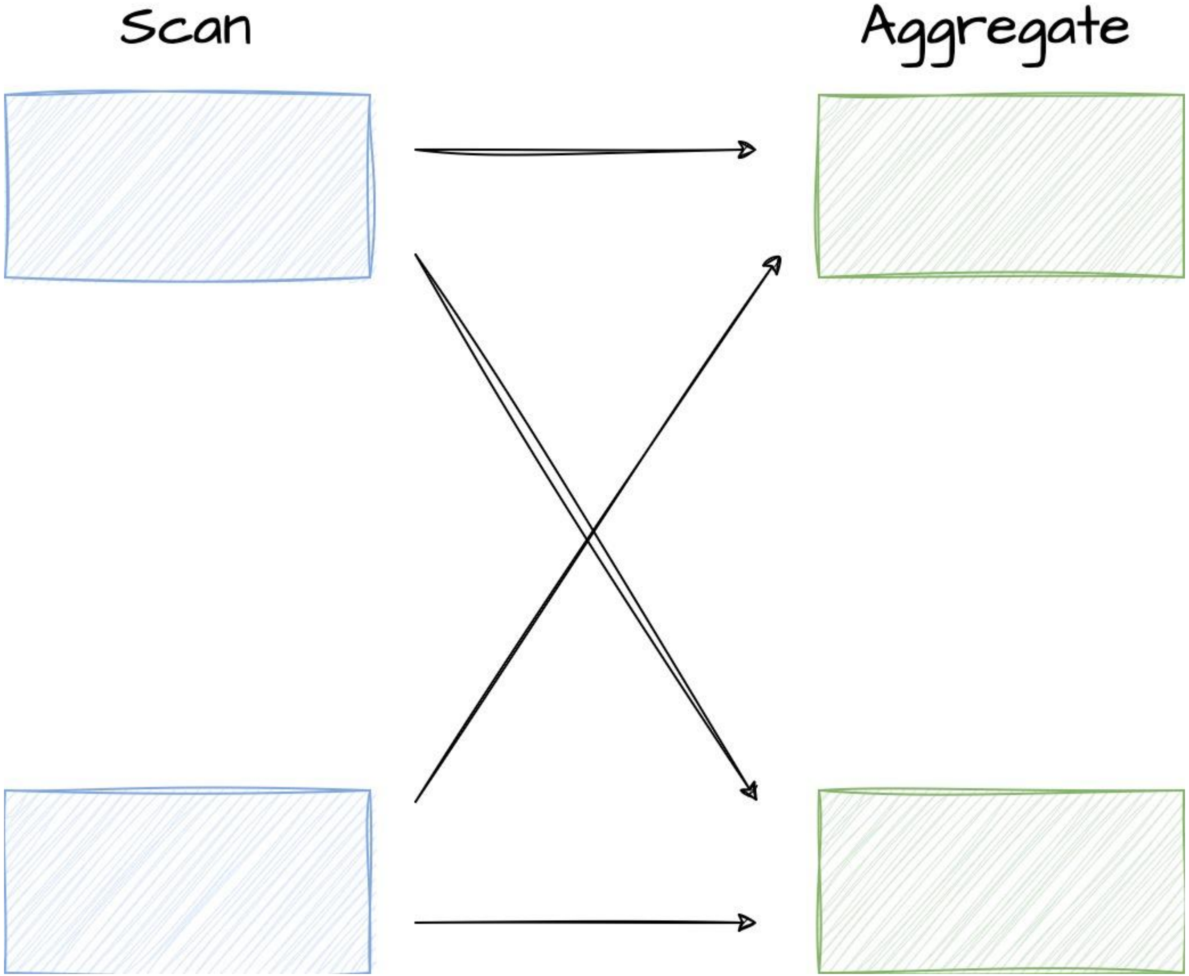
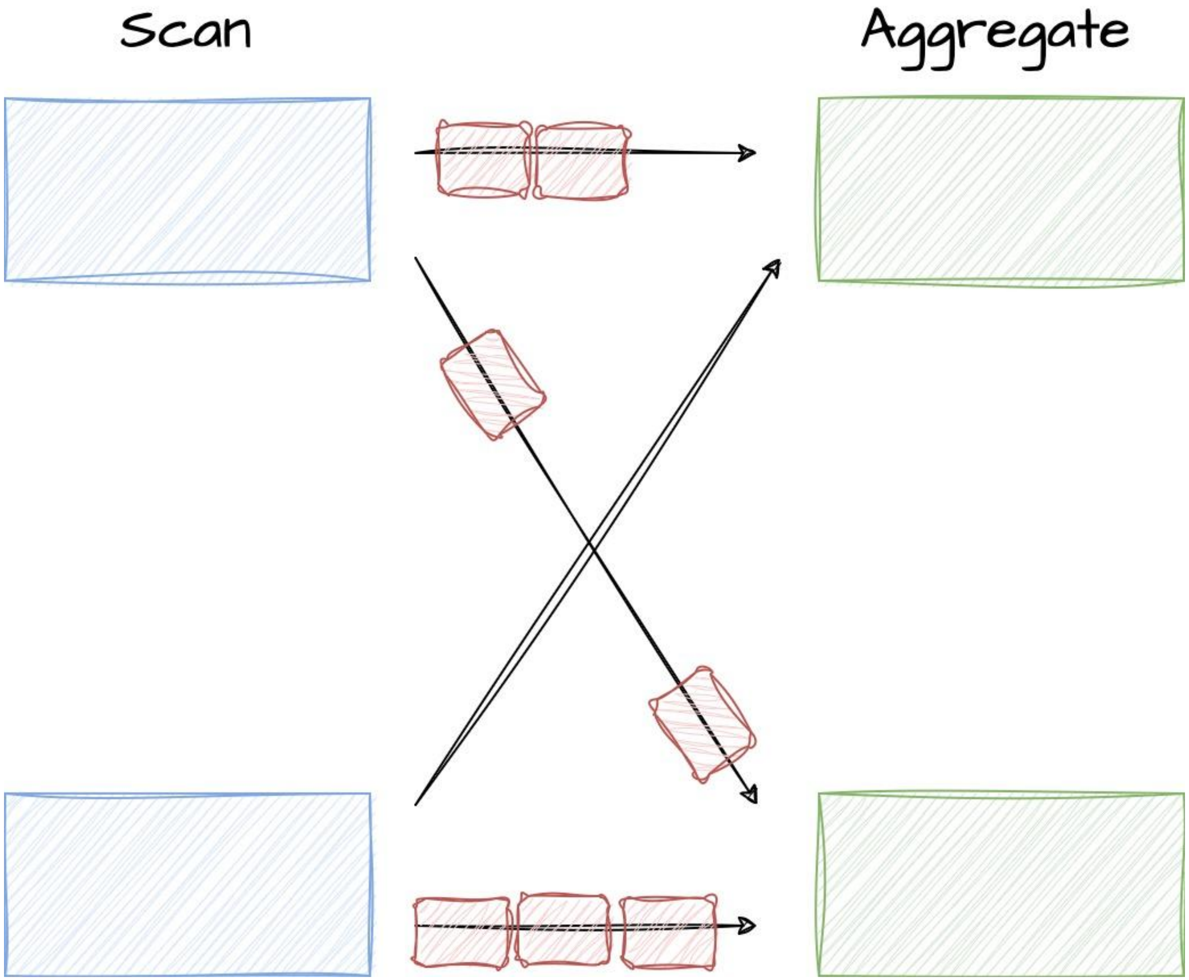


Table: Votes
Aggregate
story_id -> COUNT(*)
Join
VoteCount.story_id = stories.story_id
MV
Table: Stories


H(story_id)
H(story_id)
H(stories.id)
Votes
Votes
Votes
Aggregate
Stories
Aggregate
Stories
Aggregate
Stories
Join
Join
Join
MV
MV
MV

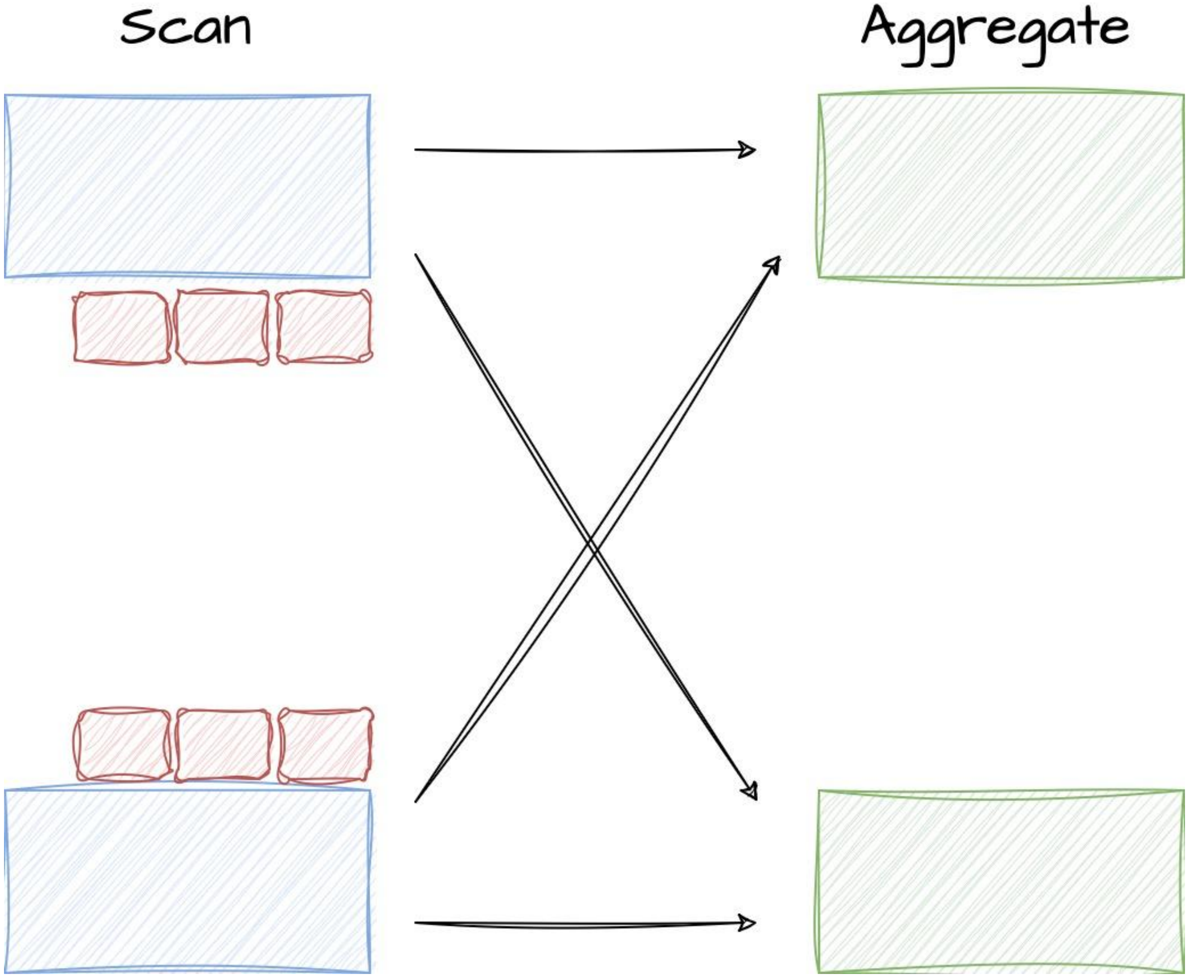

H(story_id)
H(story_id)
H(stories.id)
Votes
Votes
Votes
Aggregate
Stories
Aggregate
Stories
Aggregate
Stories
Join
Join
Join
MV
MV
MV

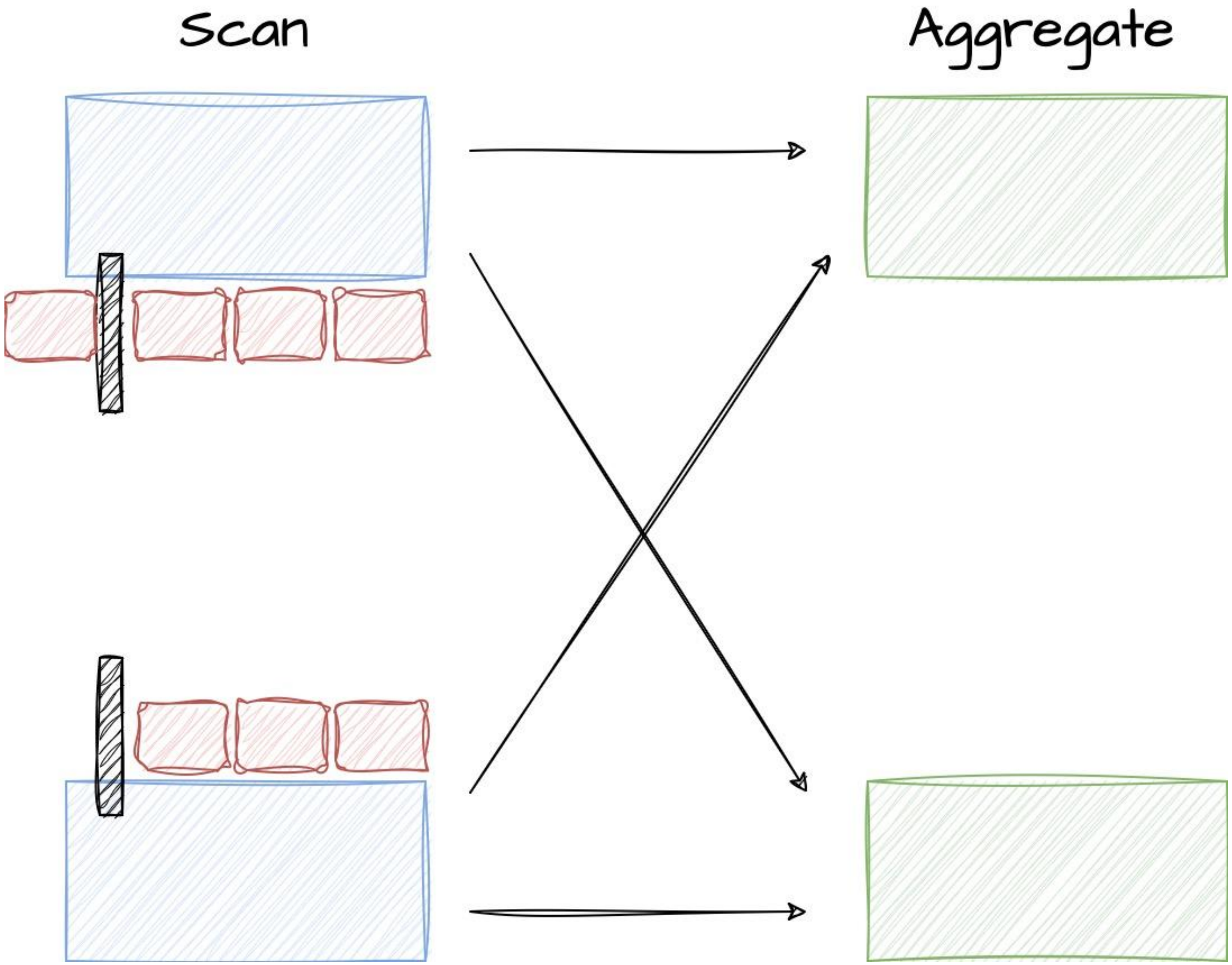

Scan
Aggregate

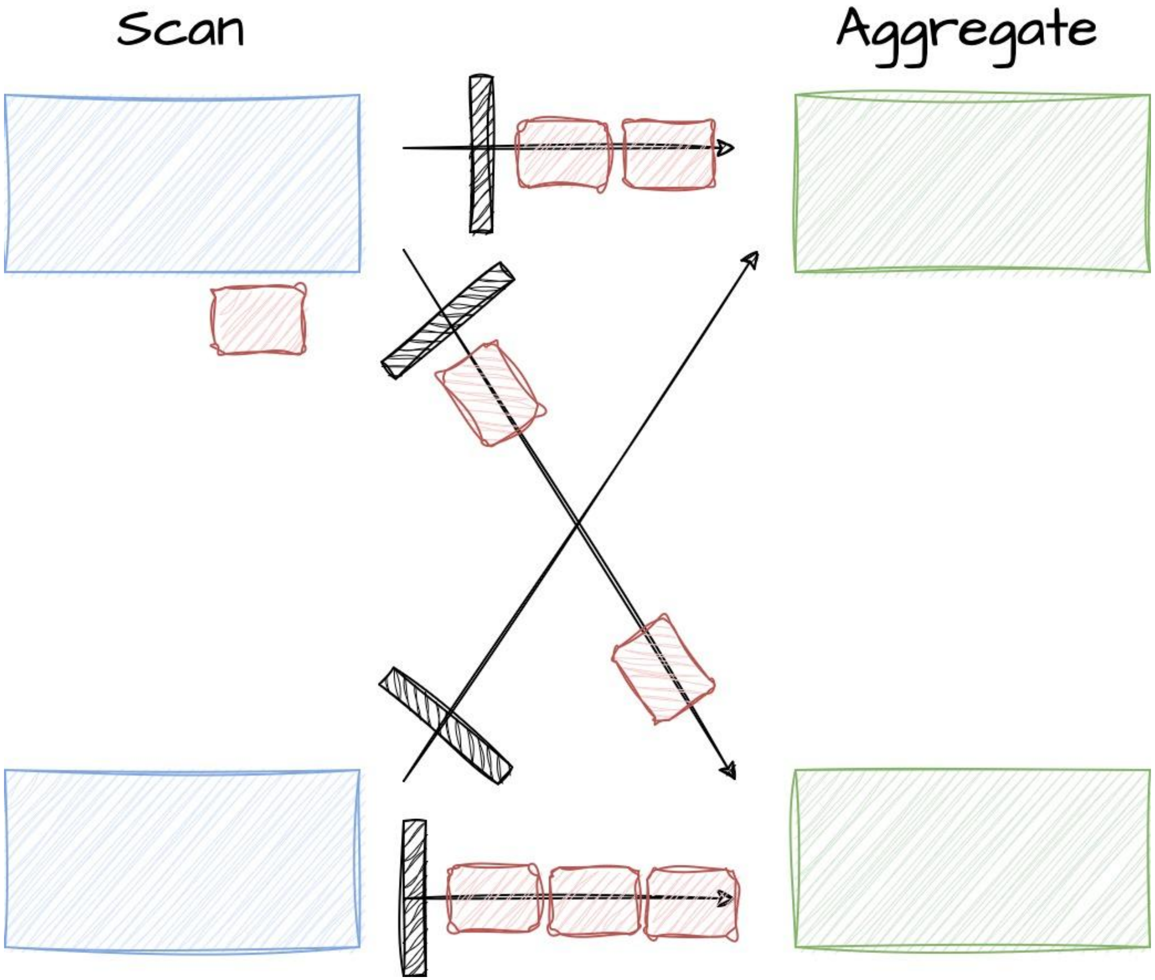


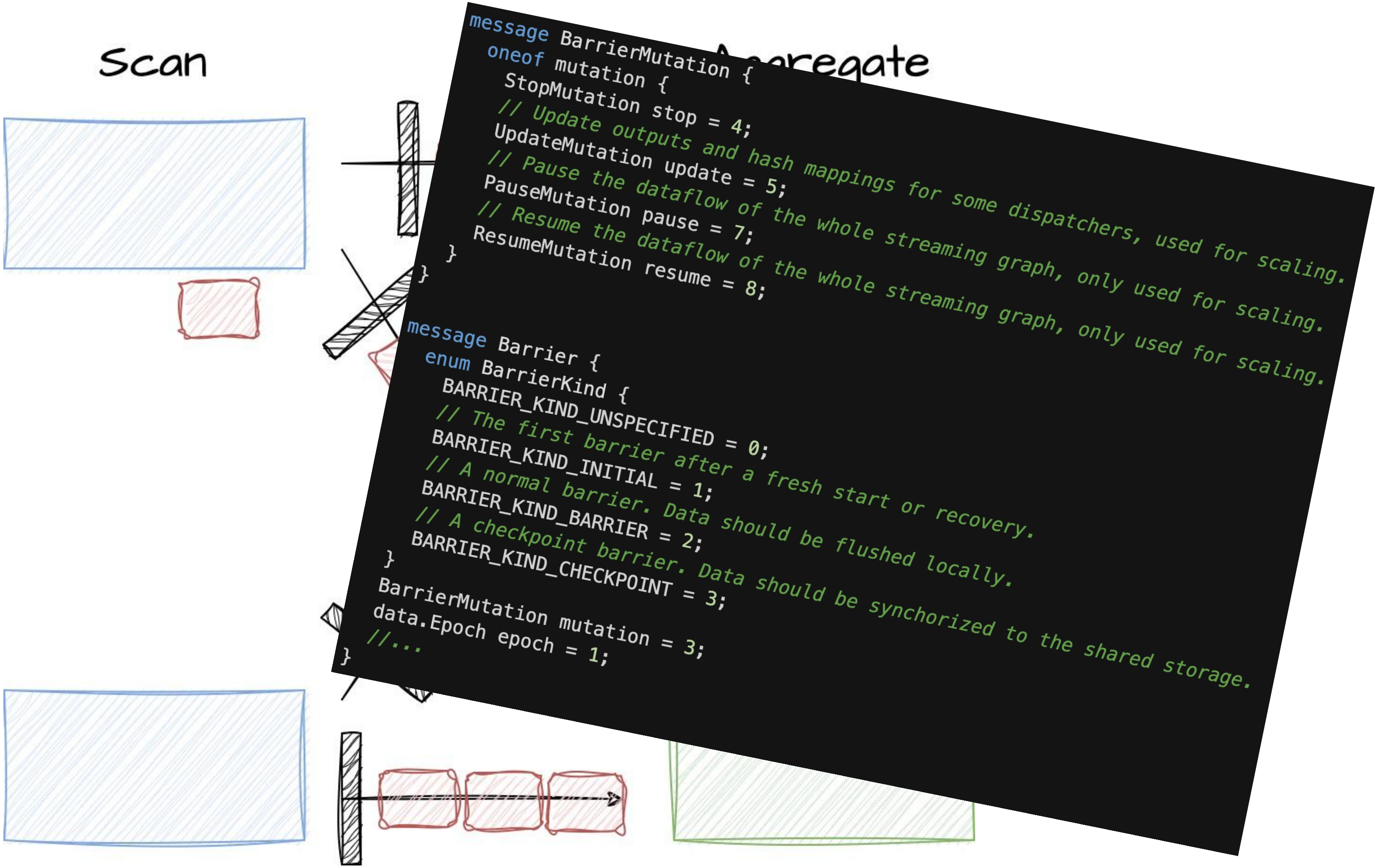
Scan

Aggregate


Scan
Aggregate


Scan
Aggregate

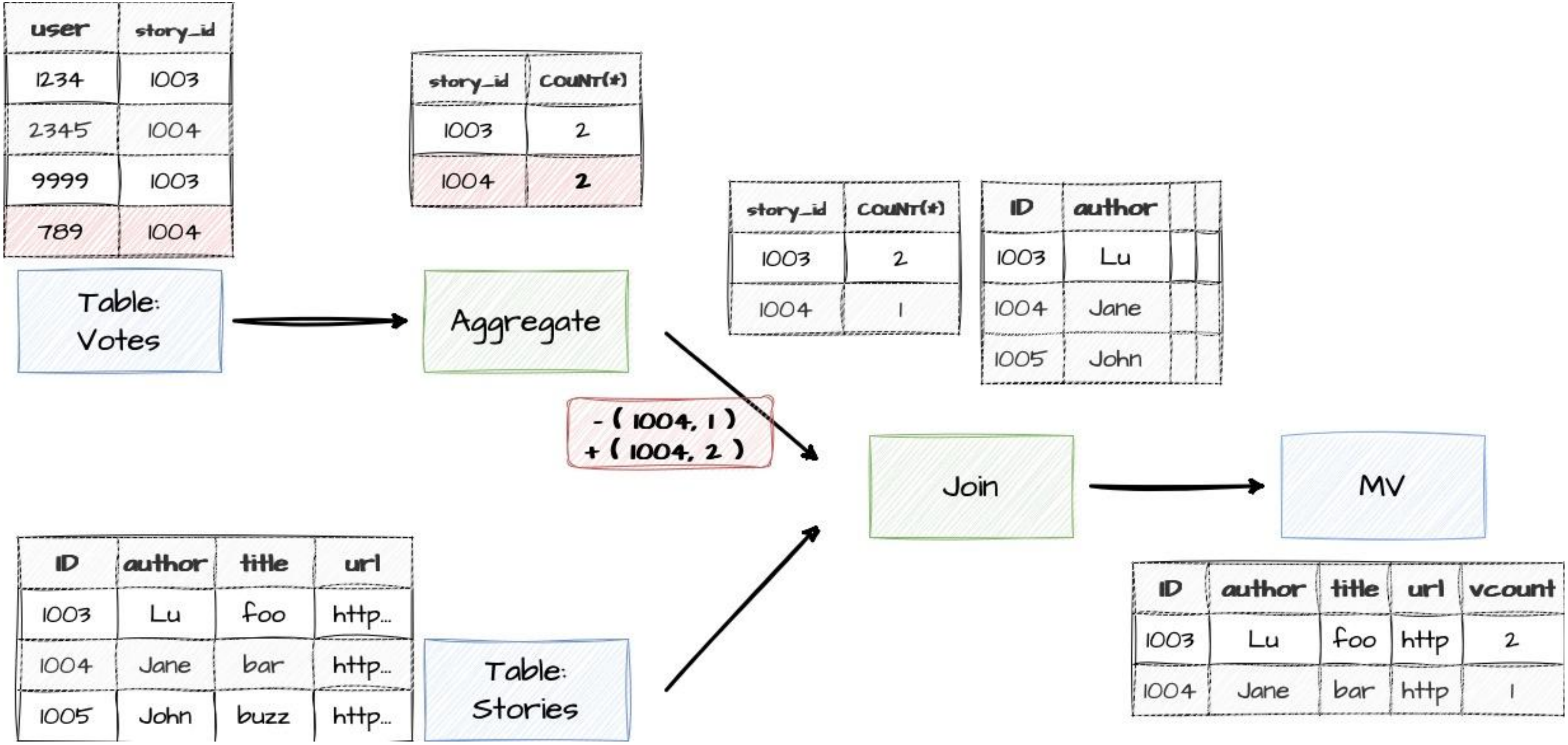

Scan
Aggregate



```
message BarrierMutation {
  oneof mutation {
    StopMutation stop = 4;
    // Update outputs and hash mappings for some dispatchers, used for scaling.
    UpdateMutation update = 5;
    // Pause the dataflow of the whole streaming graph, only used for scaling.
    PauseMutation pause = 7;
    // Resume the dataflow of the whole streaming graph, only used for scaling.
    ResumeMutation resume = 8;
  }
}

message Barrier {
  enum BarrierKind {
    BARRIER_KIND_UNSPECIFIED = 0;
    // The first barrier after a fresh start or recovery.
    BARRIER_KIND_INITIAL = 1;
    // A normal barrier. Data should be flushed locally.
    BARRIER_KIND_BARRIER = 2;
    // A checkpoint barrier. Data should be synchorized to the shared storage.
    BARRIER_KIND_CHECKPOINT = 3;
  }
  BarrierMutation mutation = 3;
  data.Epoch epoch = 1;
  //...
}
```

Distributed systems Recovery



user	story_id
1234	1003
2345	1004
9999	1003
789	1004

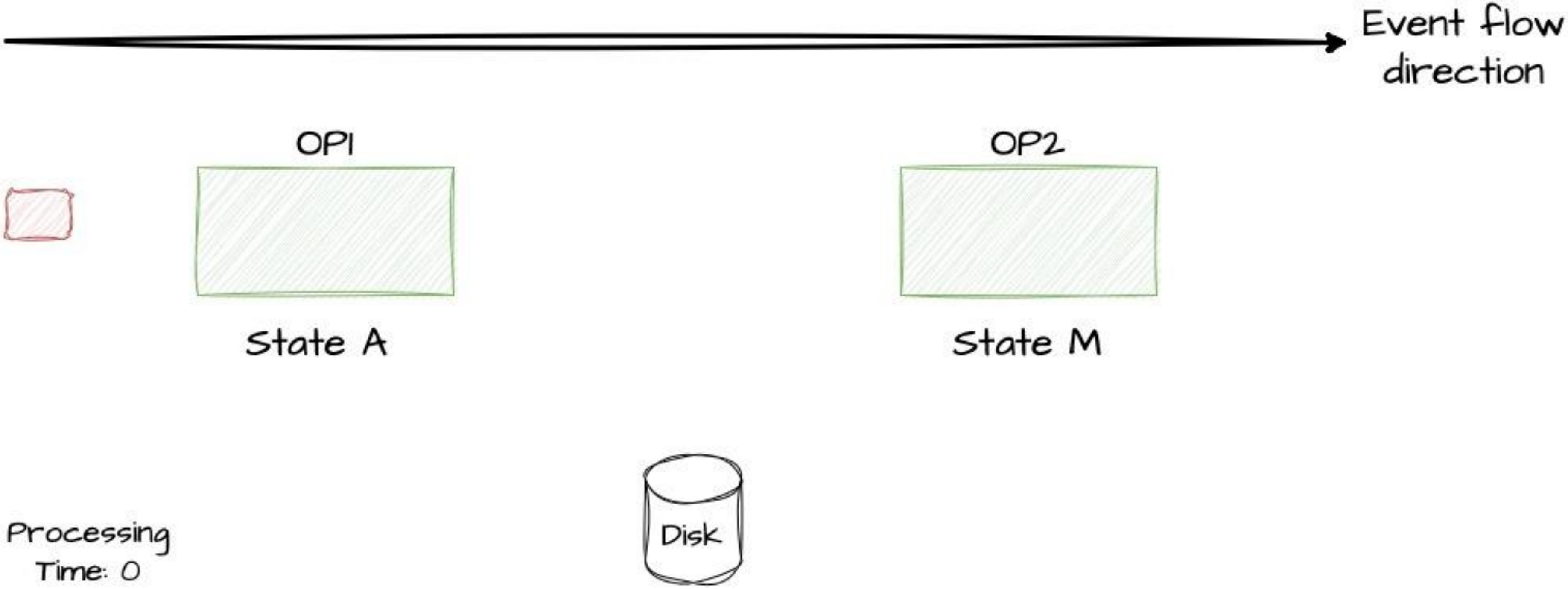
story_id	COUNT(*)
1003	2
1004	2

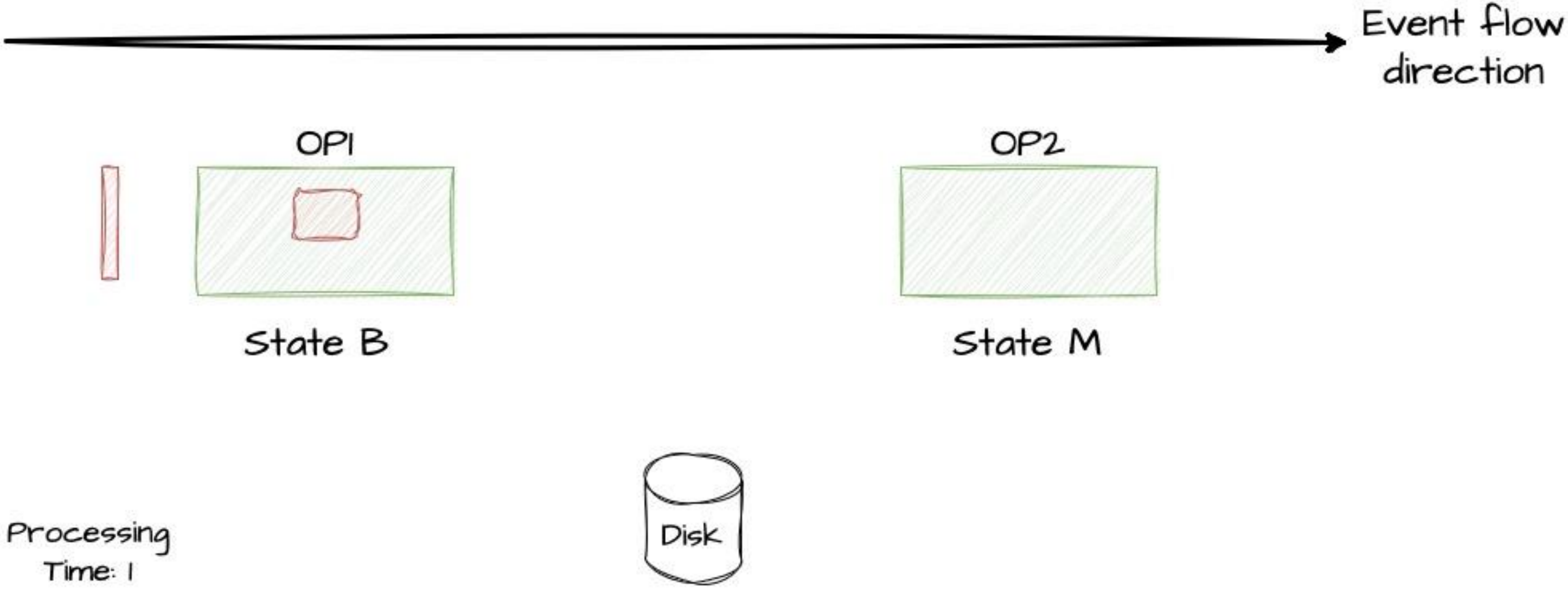
story_id	COUNT(*)
1003	2
1004	1

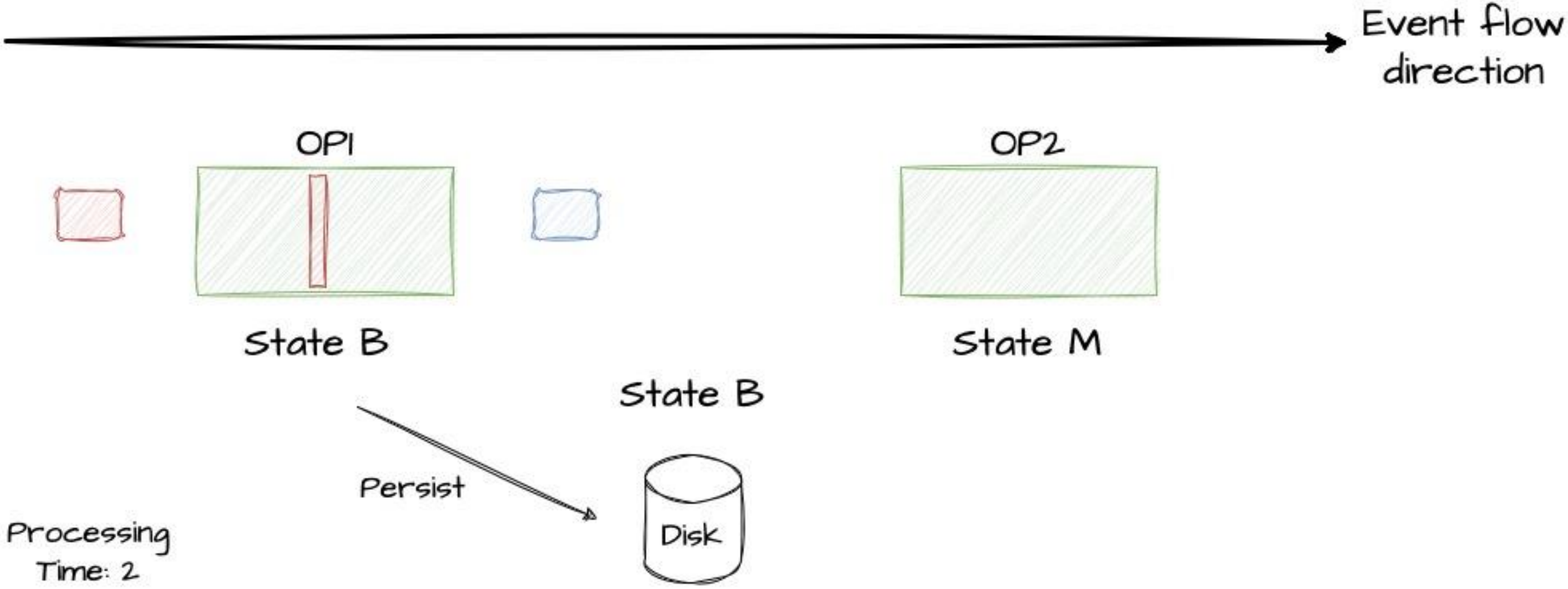
ID	author		
1003	Lu		
1004	Jane		
1005	John		

ID	author	title	url
1003	Lu	foo	http...
1004	Jane	bar	http...
1005	John	buzz	http...

ID	author	title	url	vcount
1003	Lu	foo	http	2
1004	Jane	bar	http	1


Event flow direction
OP1
State A
OP2
State M
Disk
Processing Time: 0


Event flow direction
OP1
OP2
State B
State M
Disk
Processing Time: 1



Event flow direction

OP1

State B

OP2

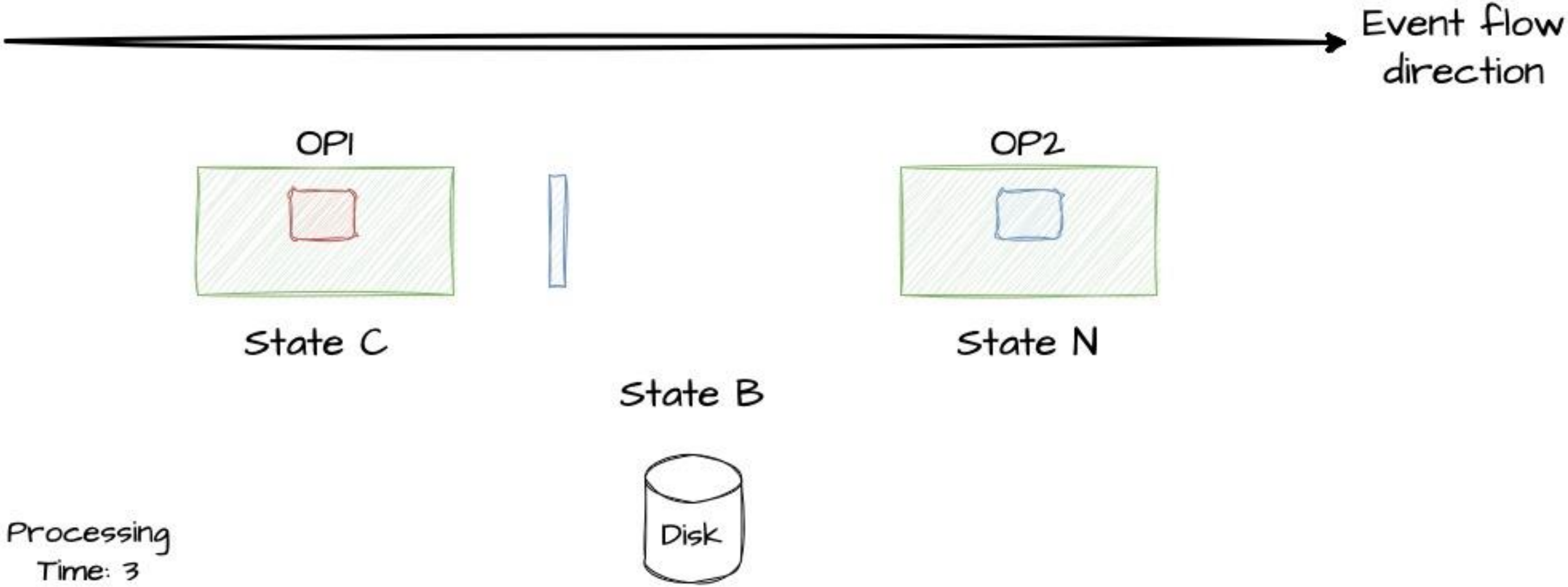
State M

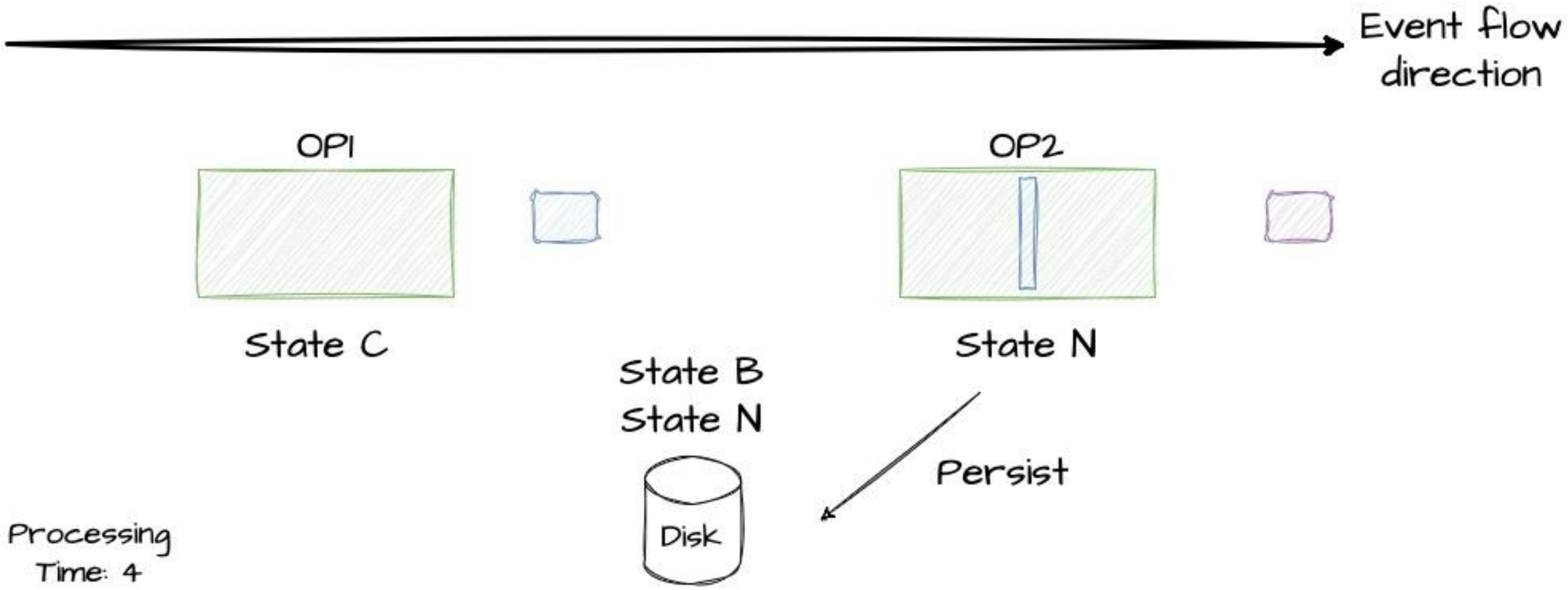
State B

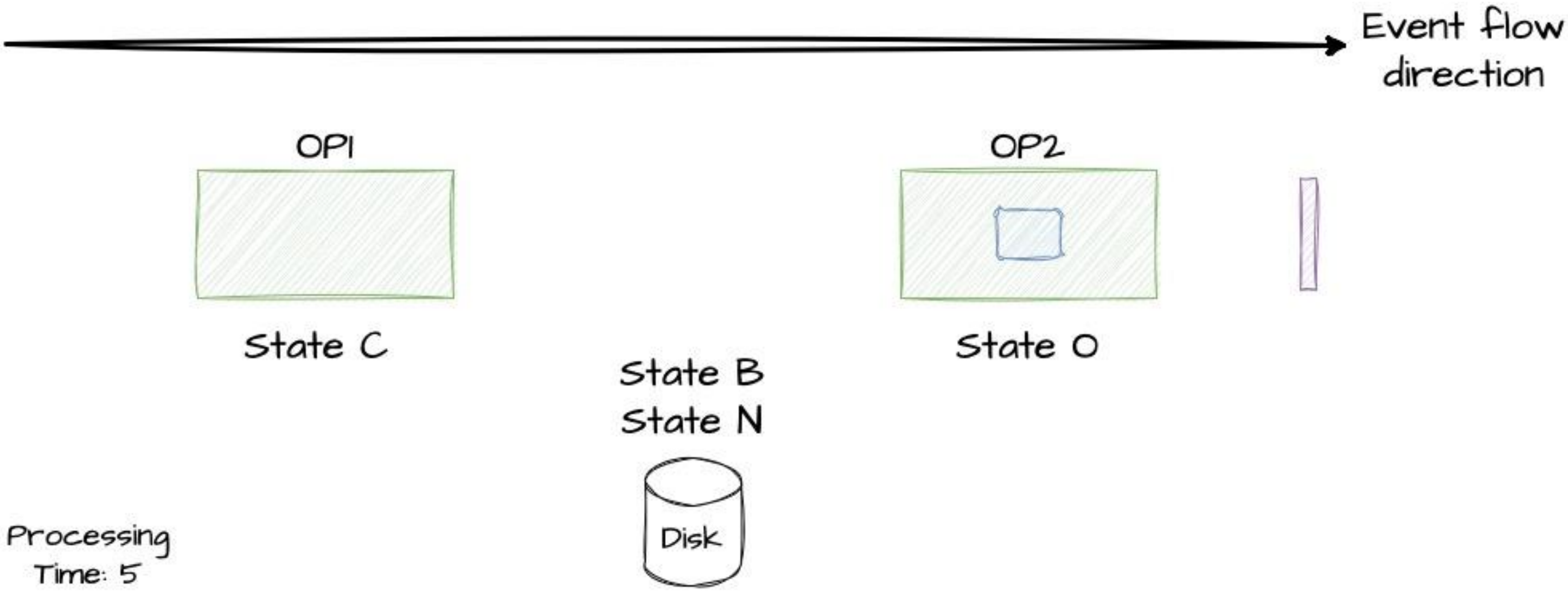
Persist

Disk

Processing Time: 2


Event flow direction
OP1
OP2
State C
State N
State B
Disk
Processing Time: 3


Event flow direction
OP1
OP2
State C
State N
State B
State N
Persist
Disk
Processing Time: 4



Event flow direction

OP1

State C

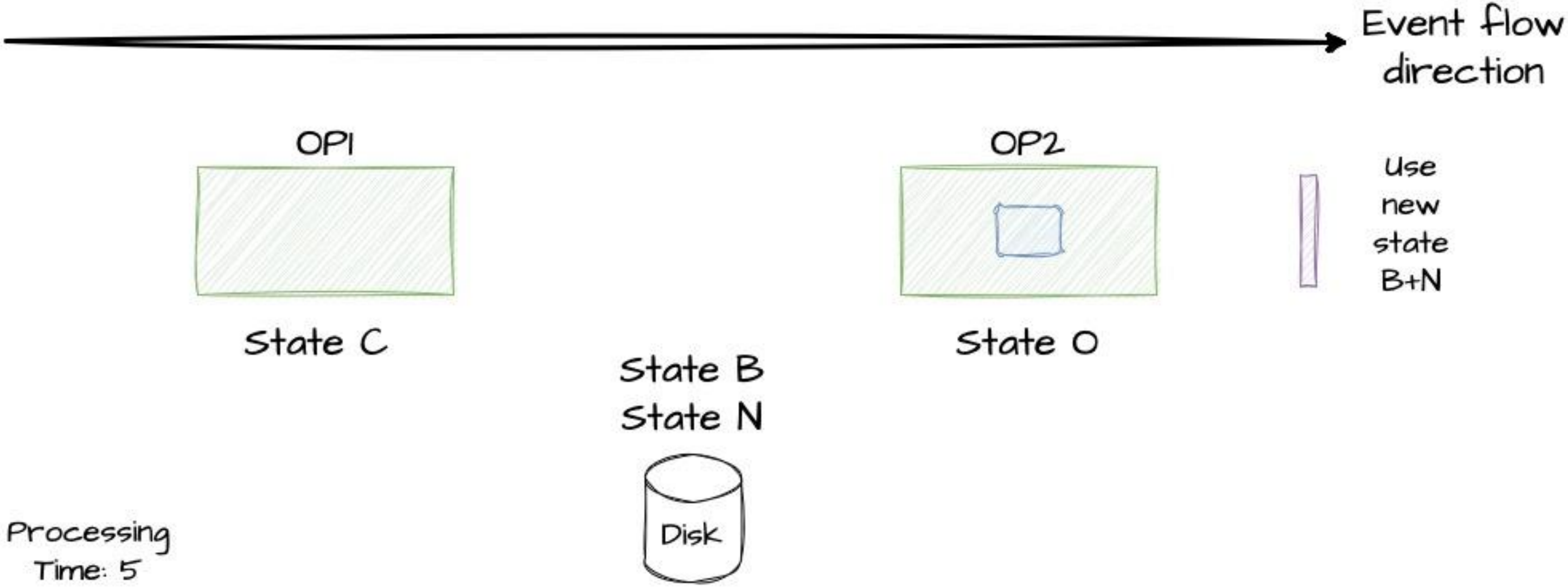
OP2

State O

State B
State N

Disk

Processing Time: 5


Event flow direction
OP1
OP2
Use new state B+N
State C
State O
State B
State N
Disk
Processing Time: 5

Distributed systems
Scaling

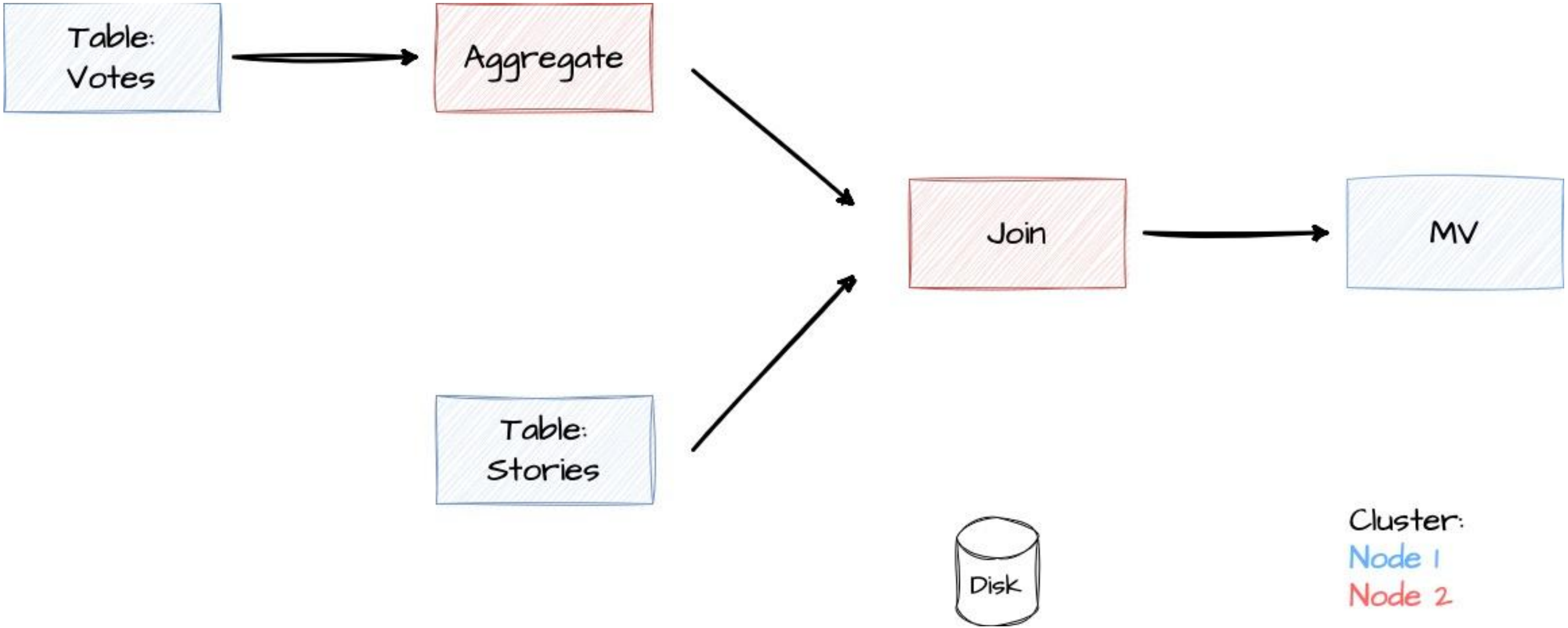


Table:
Votes
Aggregate
Table:
Stories
Join
MV
Disk
Cluster:
Node 1
Node 2

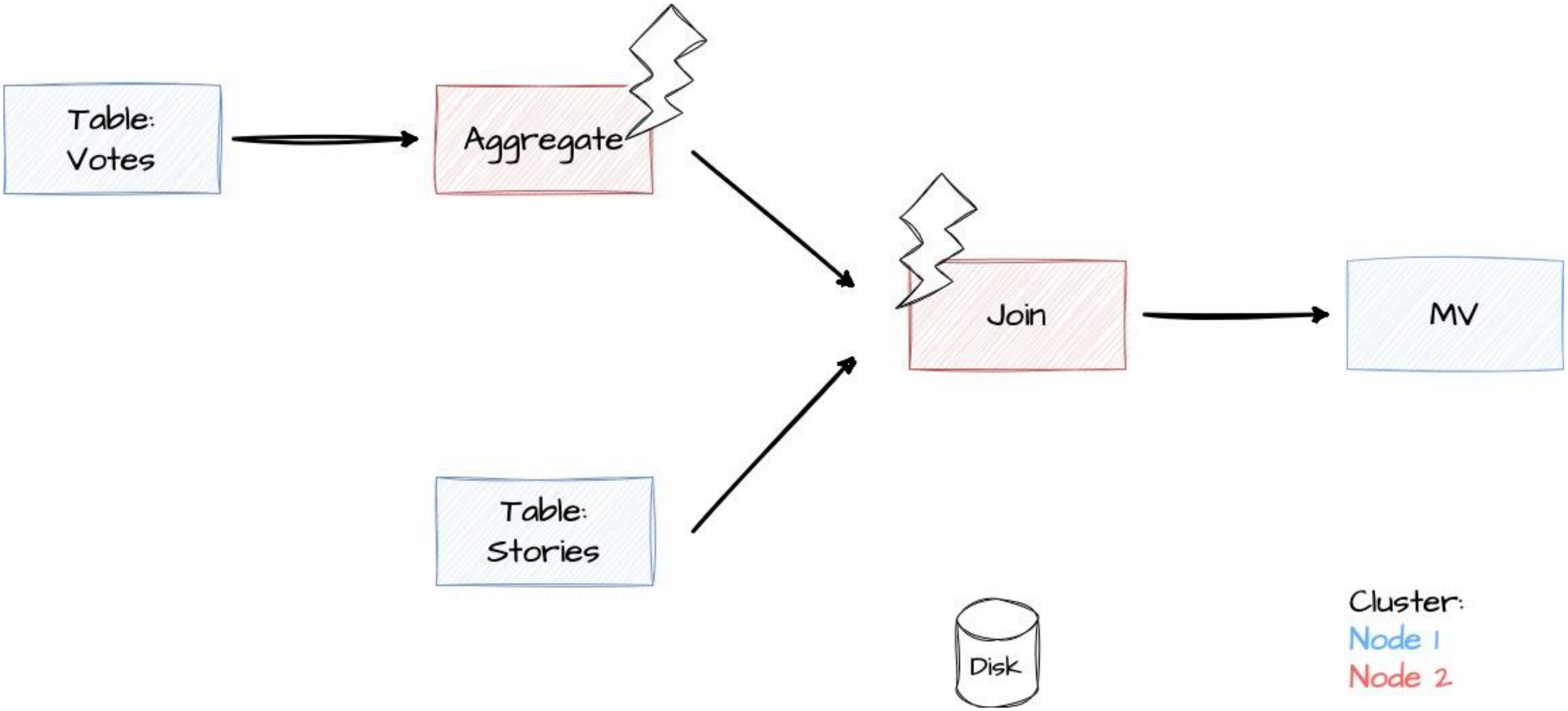


Table:
Votes
Aggregate
Join
MV
Table:
Stories
Disk
Cluster:
Node 1
Node 2

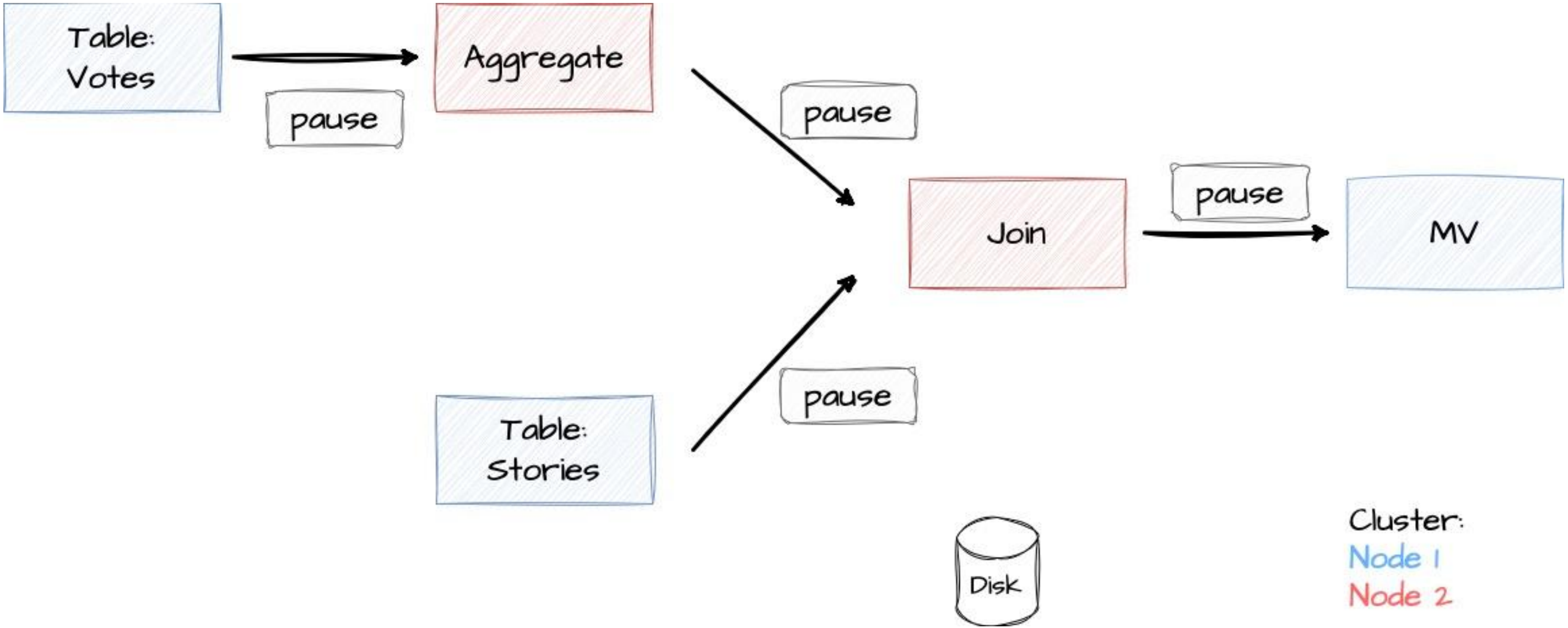
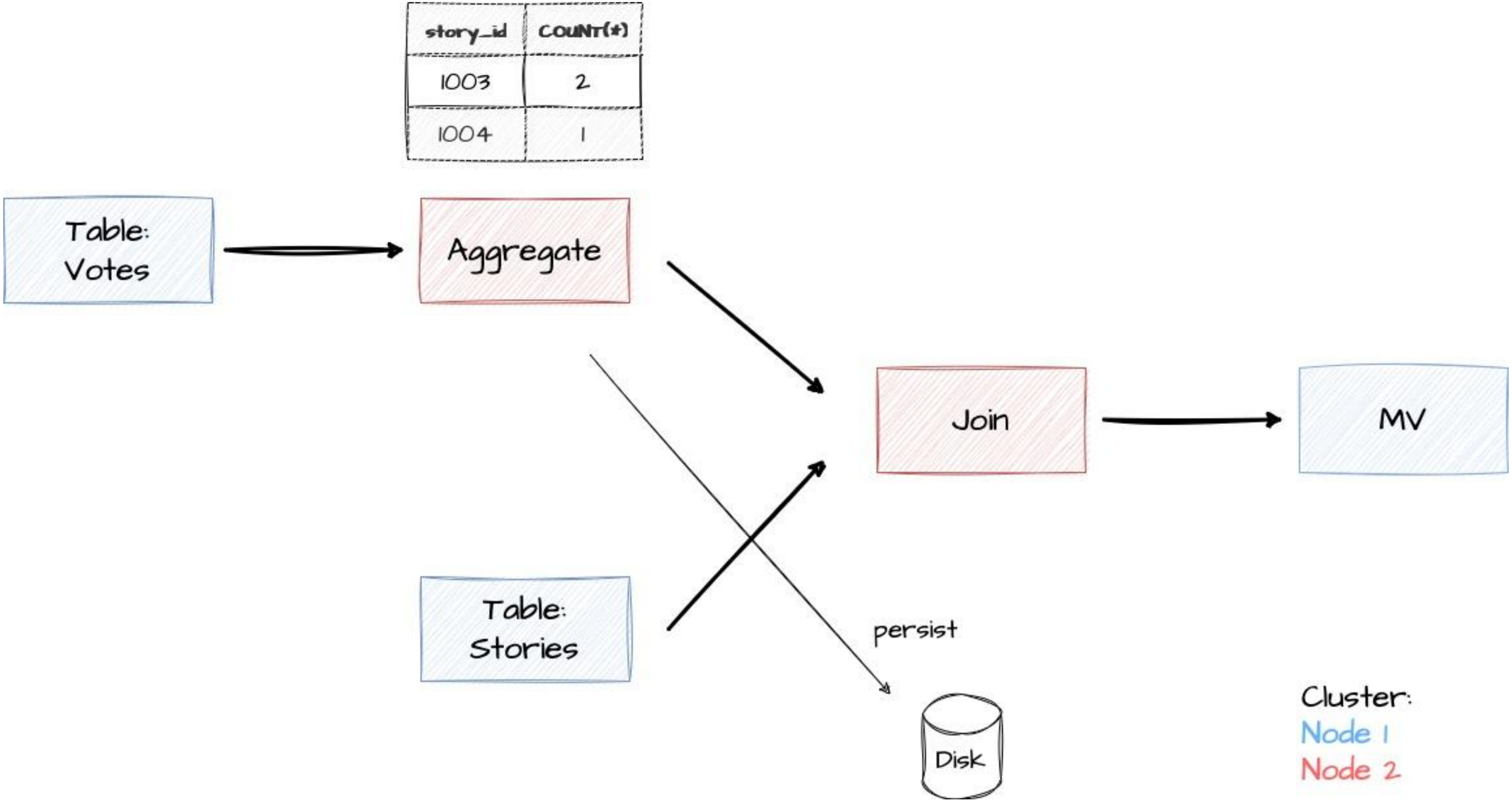
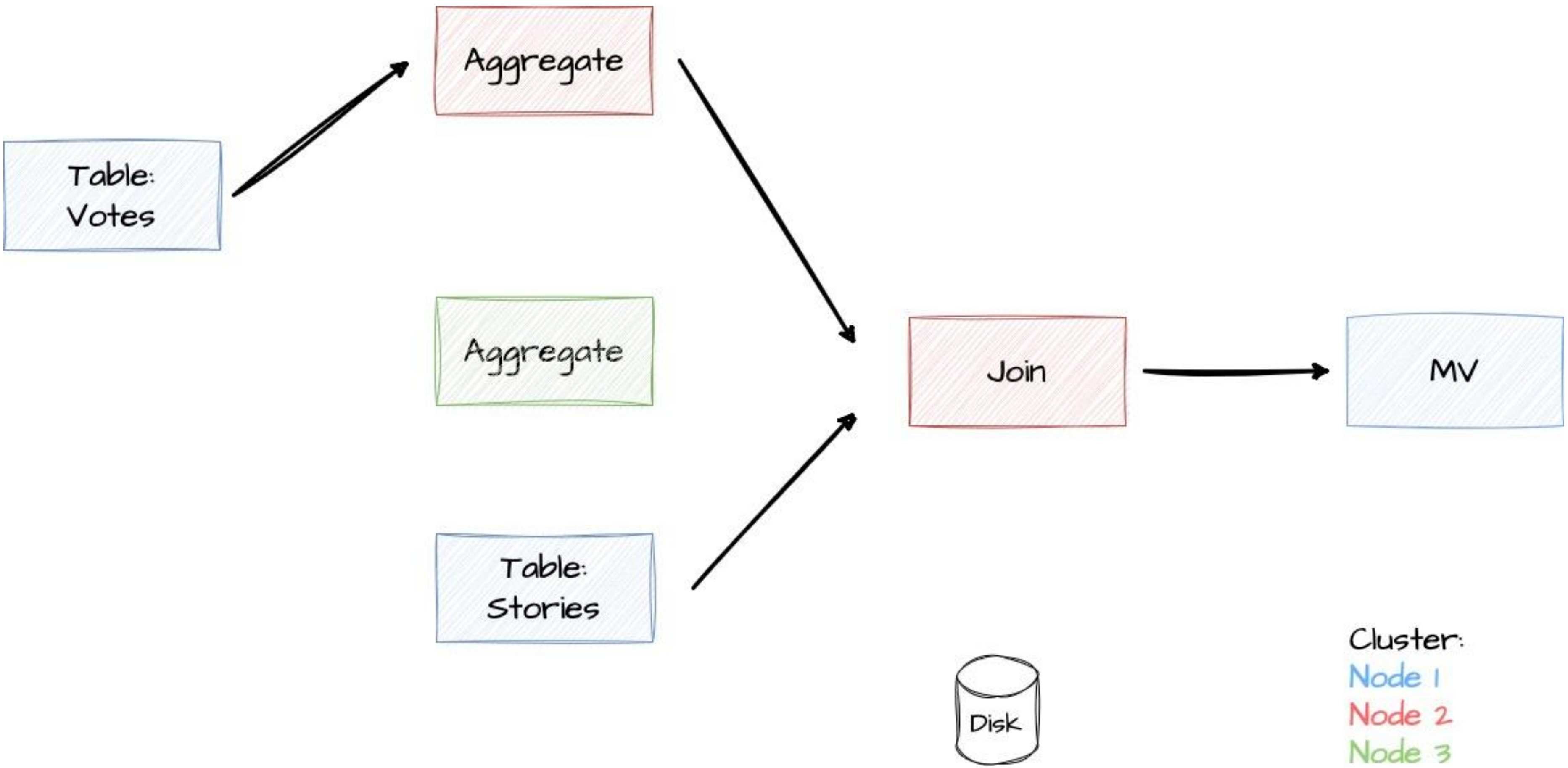
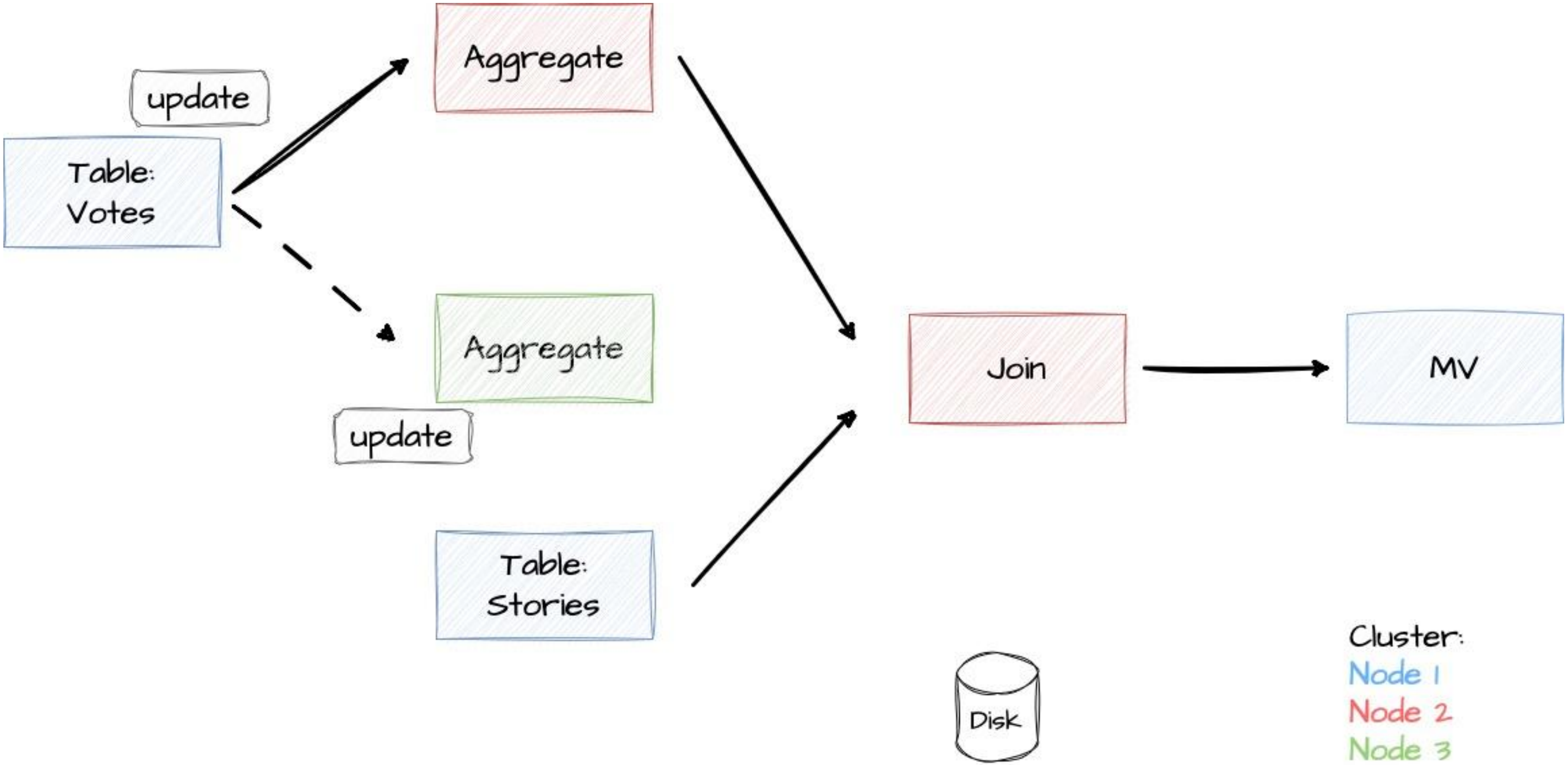


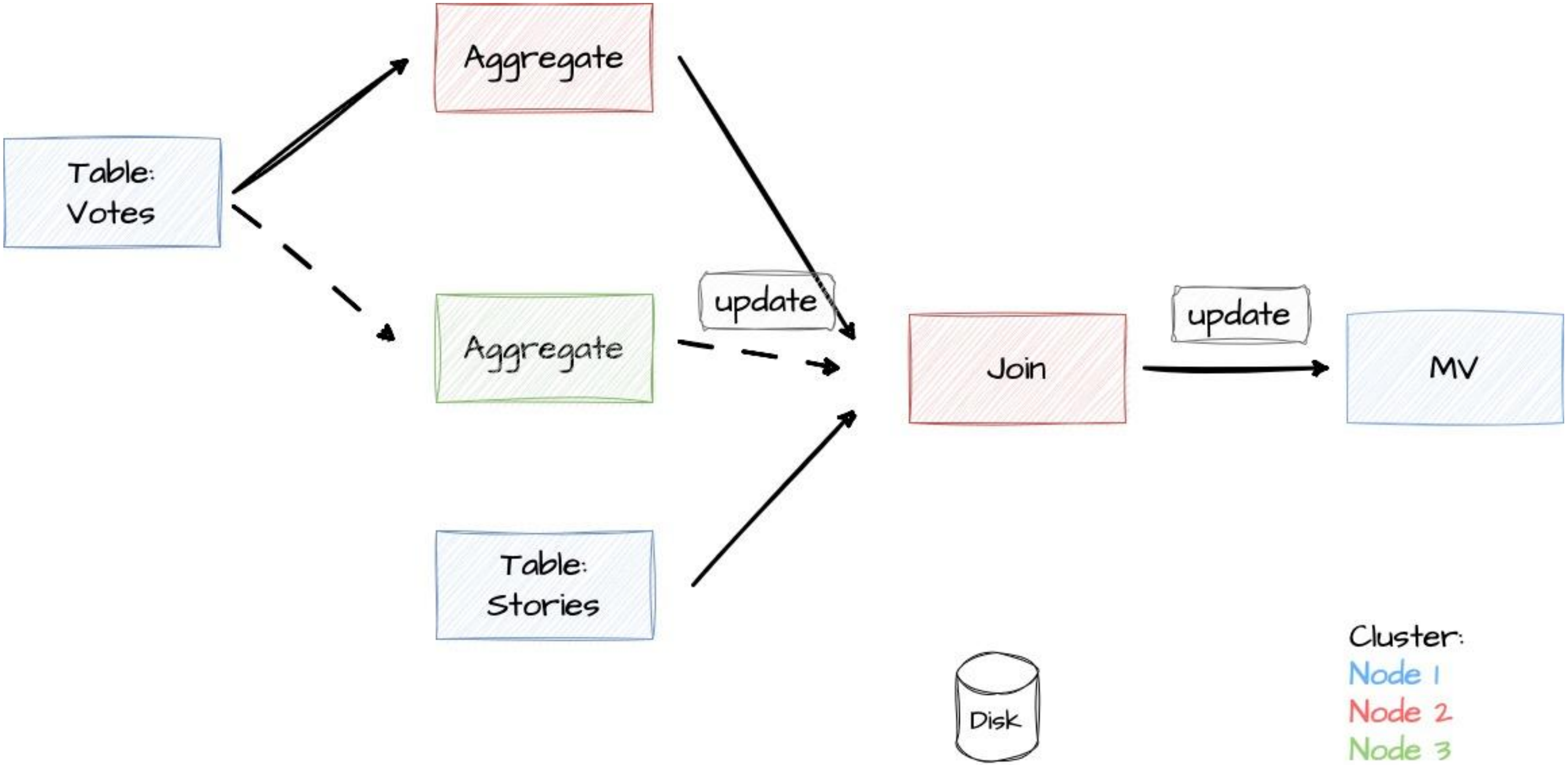
Table: Votes
pause
Aggregate
pause
Join
pause
MV
pause
Table: Stories
Disk
Cluster:
Node 1
Node 2

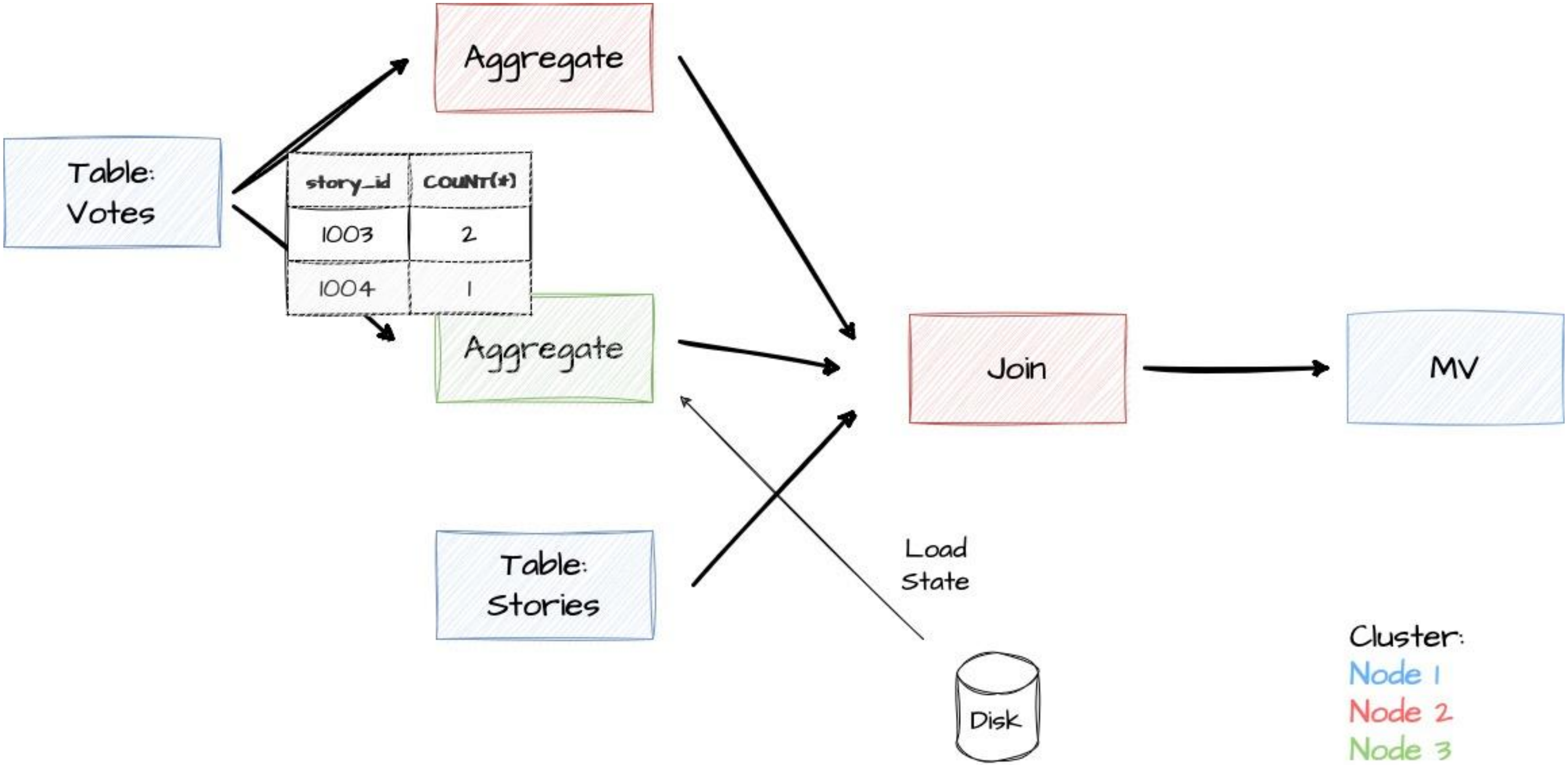


story_id	COUNT(*)
1003	2
1004	1

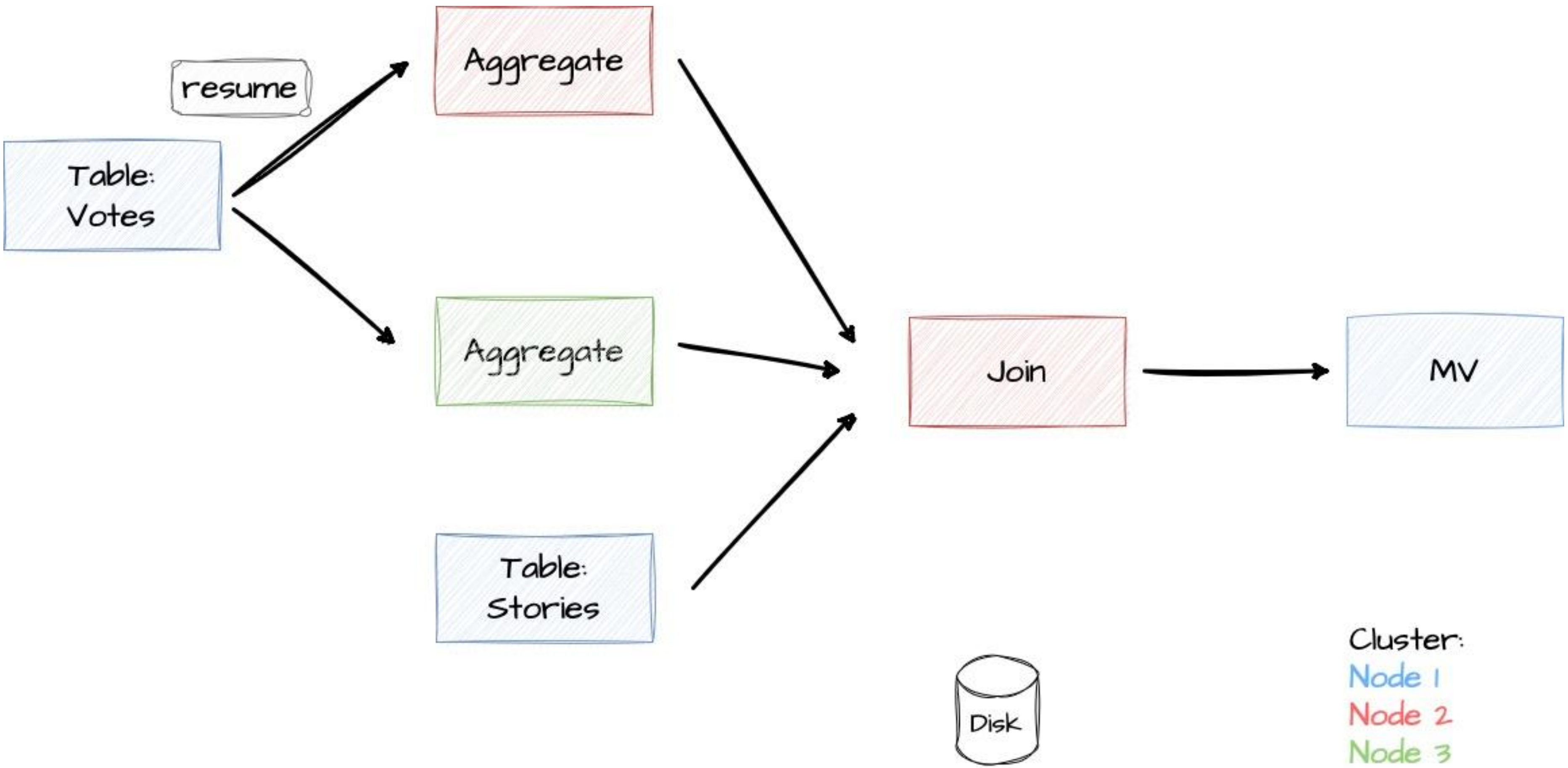

Aggregate
Table: Votes
Aggregate
Join
MV
Table: Stories
Disk
Cluster:
Node 1
Node 2
Node 3


update
Table:
Votes
Aggregate
Aggregate
update
Table:
Stories
Join
MV
Disk
Cluster:
Node 1
Node 2
Node 3


Aggregate
Table: Votes
update
Aggregate
Join
update
MV
Table: Stories
Disk
Cluster:
Node 1
Node 2
Node 3



story_id	COUNT(*)
1003	2
1004	1


resume
Table:
Votes
Aggregate
Aggregate
Table:
Stories
Join
MV
Disk
Cluster:
Node 1
Node 2
Node 3

Thank you!

Try RisingWave:

Ask questions: