بسم الله الرحمن الرحیم

تکنولوژی کامپیوتر

جلسه‌ی سیزدهم

رپلیکیشن - همانندسازی

جلسه گذشته

اتکاپذیری، مقیاس‌پذیری و نگهداری‌پذیری

جلسه جدید

رپلیکیشن

- Replication is keeping an entire copy of the data on multiple machines
- We will assume that all of data can be stored in a single machine

چرا رپلیکیشن؟

- مقیاس پذیری؟
  - *در افزایش چه لود پارامتری؟ با چه تغییری؟ روی چه پرفورمنس متریکی؟*
- اتکا پذیری؟
- نگهداری‌پذیری؟
- همچنین کمک به بهبود ریسپانس‌تایم؟ چطوری؟؟

اگه داده‌مون تغییر نکنه

روش‌های رپلیکیشن

- Single Leader Replication
- Multi-Leader Replication
- Leaderless Replication (جلسه‌ی بعد)

SINGLE LEADER REPLICATION

Leader / Follower

- Replica / Node
- Leader (Or Master / Primary)
  - *One of replicas will be Leader*
  - *All writes must go through the leader*
- Follower
  - *Catch data from leader*
  - *reads can come from any replica*

What happen on write?

- On writes, leader sends change log information to all followers
  - ***Synchronous*** *- clear success/failure; clients must wait for all replicas*
  - ***Asynchronous*** *- lowest latency; no guarantee of durability after successful write*
- Sync vs Async? Pros and Cons
  - *Single synchronous follower improves durability*
  - *Semi-Sync replication*

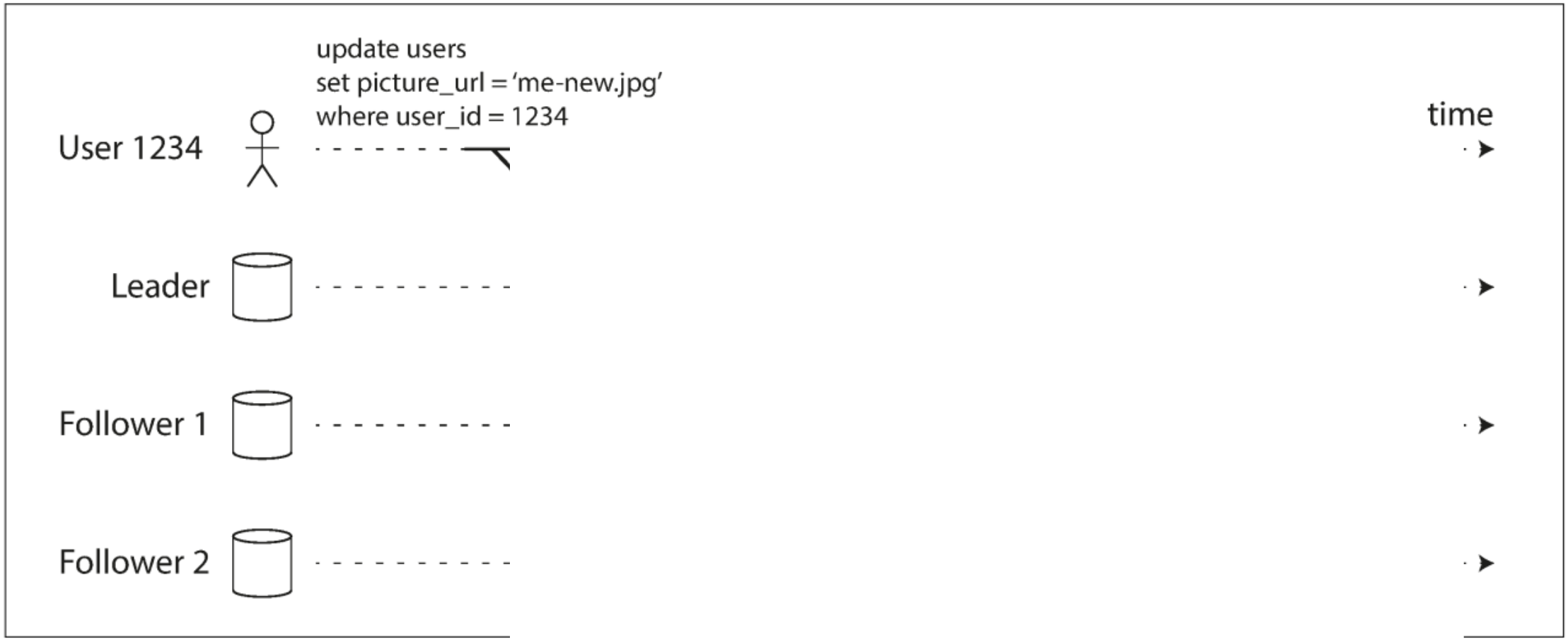


Figure 5-2. Leader-based replication with one synchronous and one asynchronous follower.

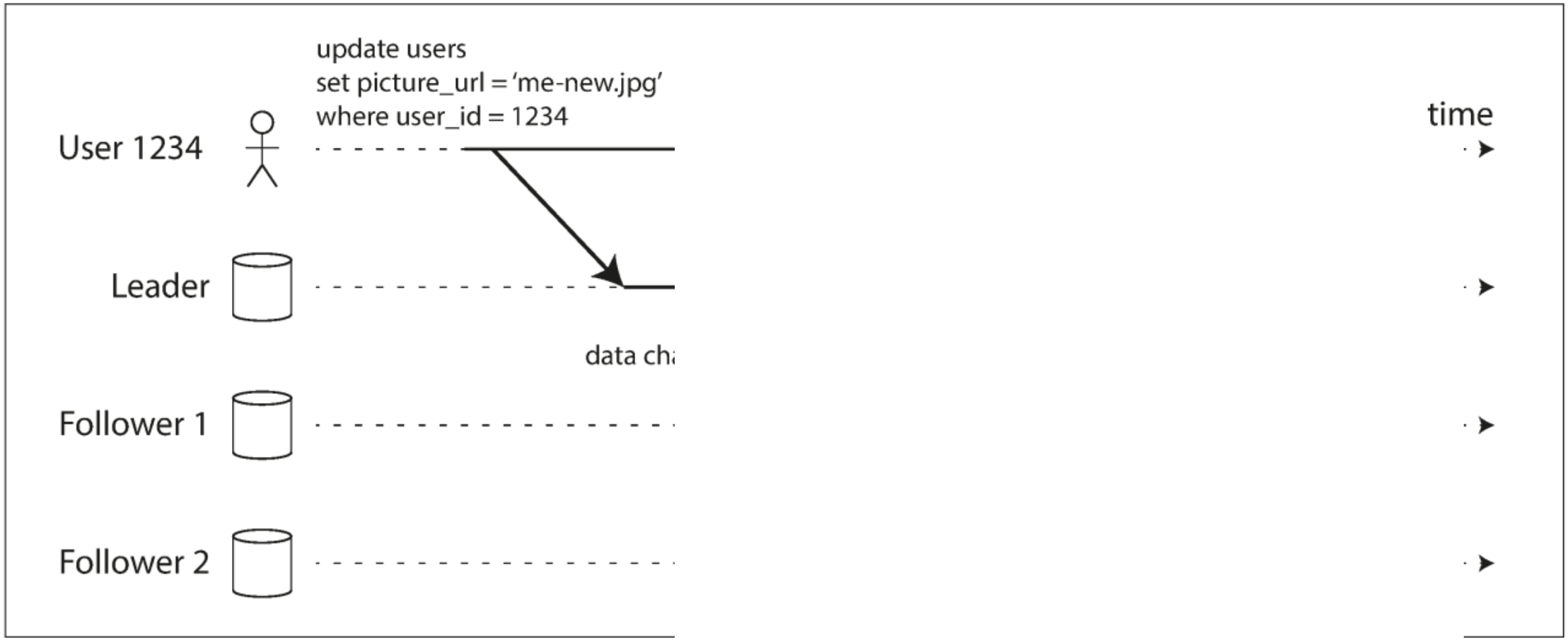


Figure 5-2. Leader-based replication with one synchronous and one asynchronous follower.

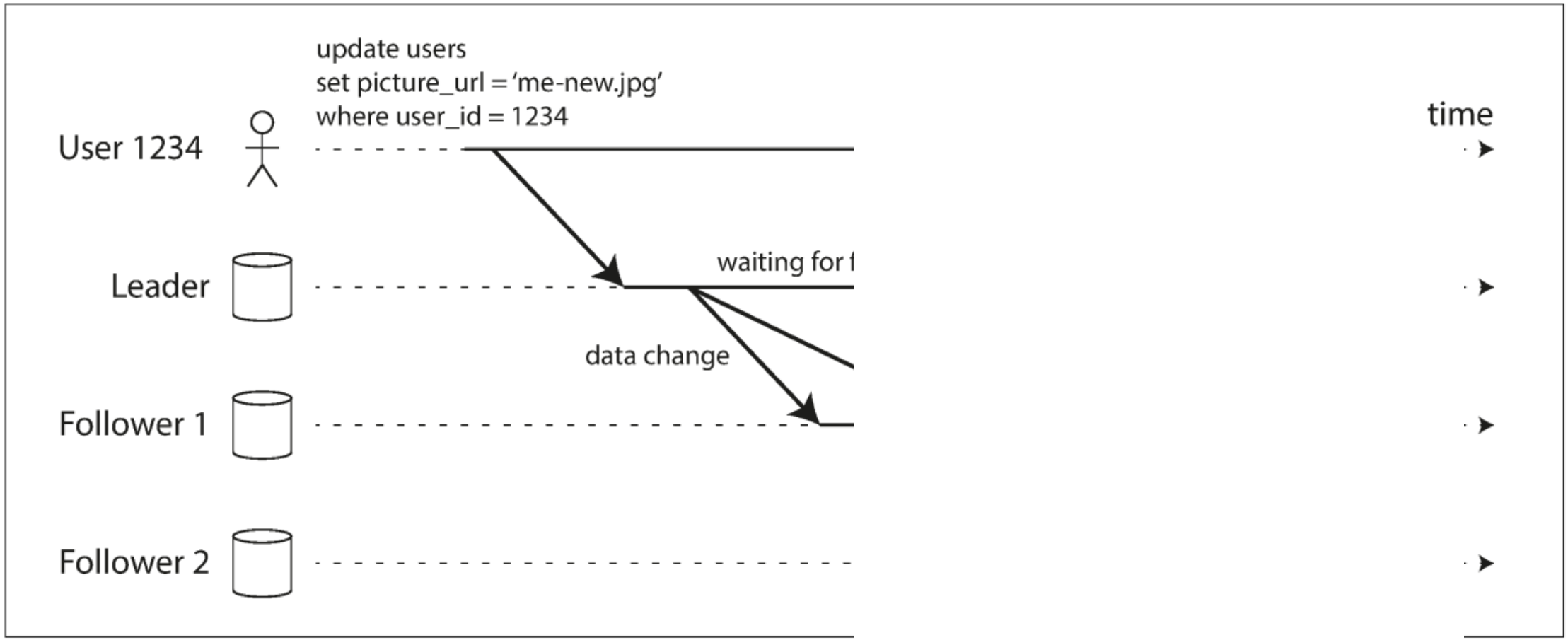


Figure 5-2. Leader-based replication with one synchronous and one asynchronous follower.

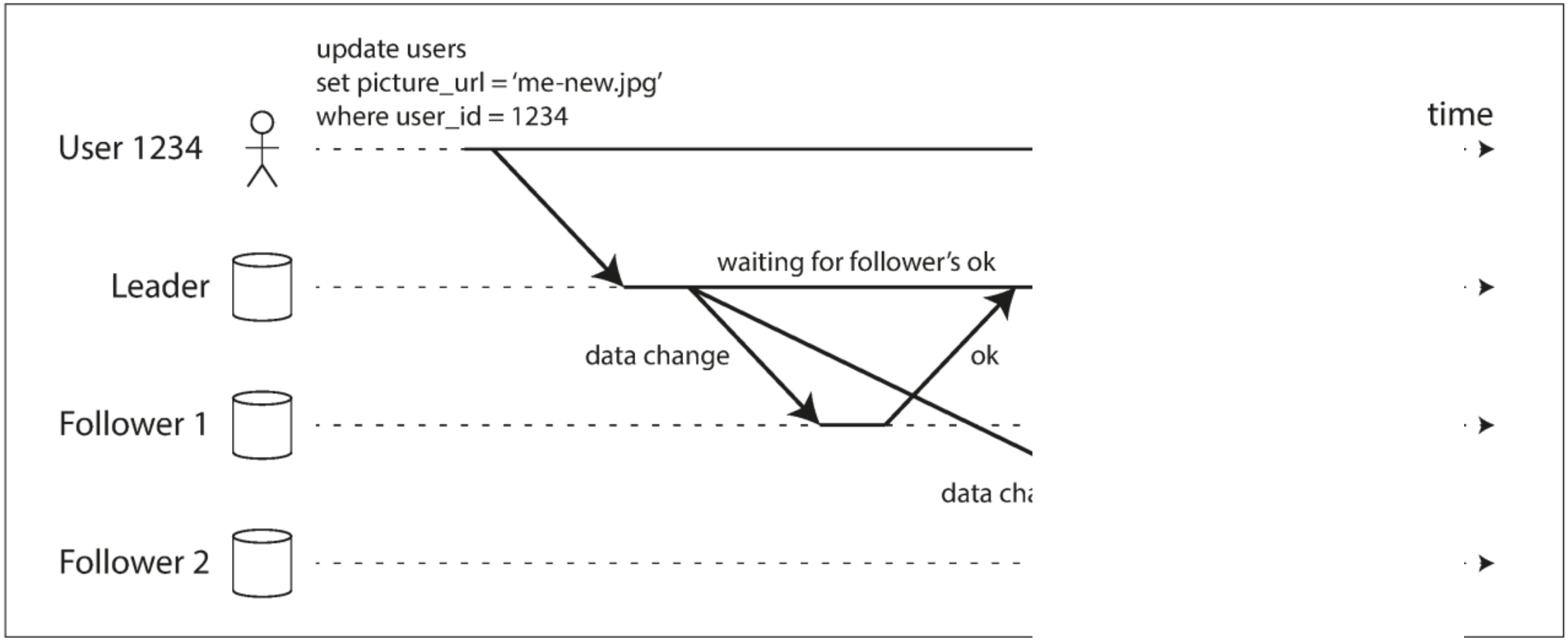


Figure 5-2. Leader-based replication with one synchronous and one asynchronous follower.

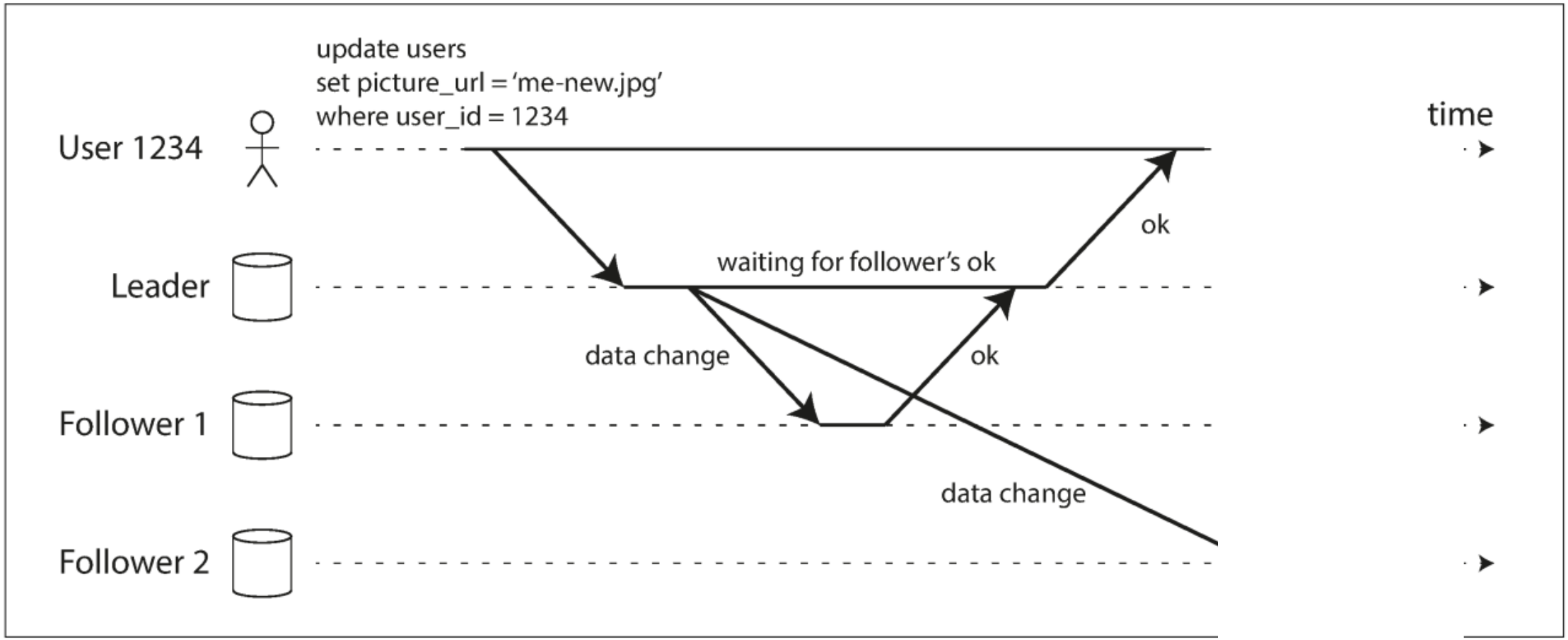


Figure 5-2. Leader-based replication with one synchronous and one asynchronous follower.

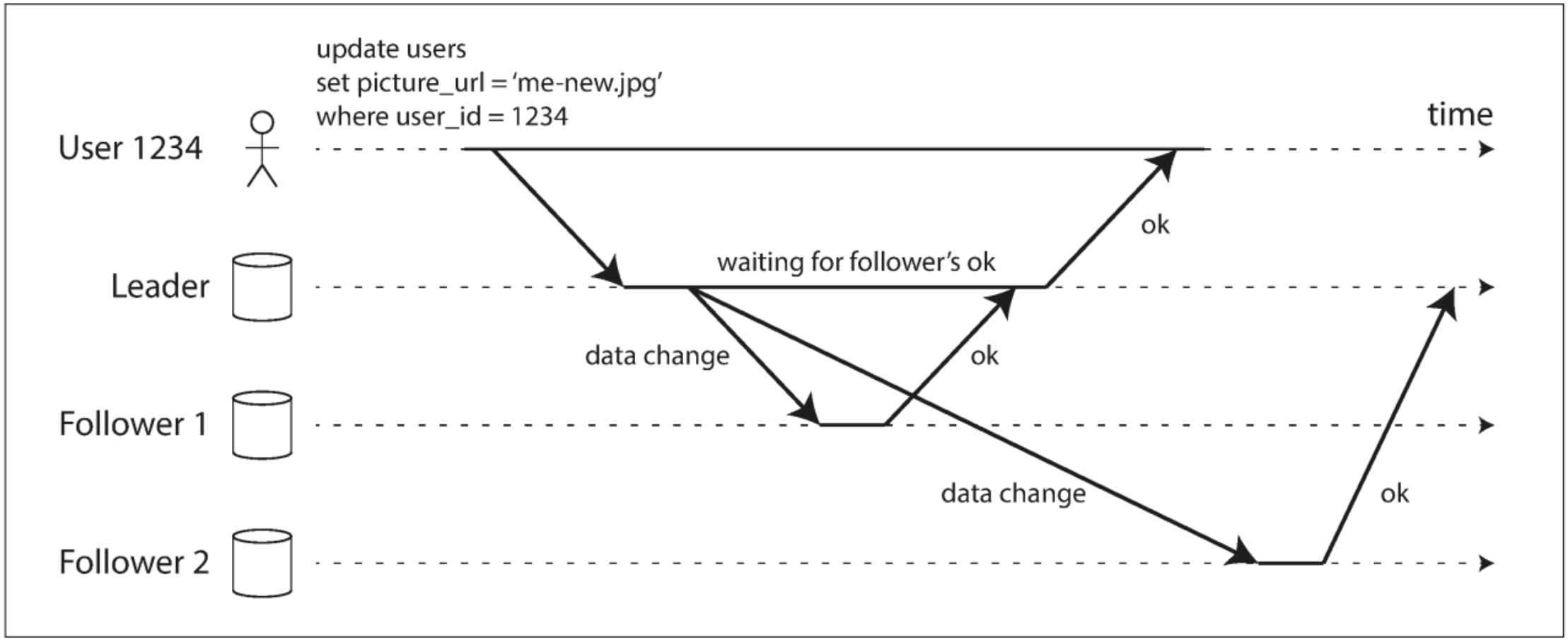


Figure 5-2. Leader-based replication with one synchronous and one asynchronous follower.

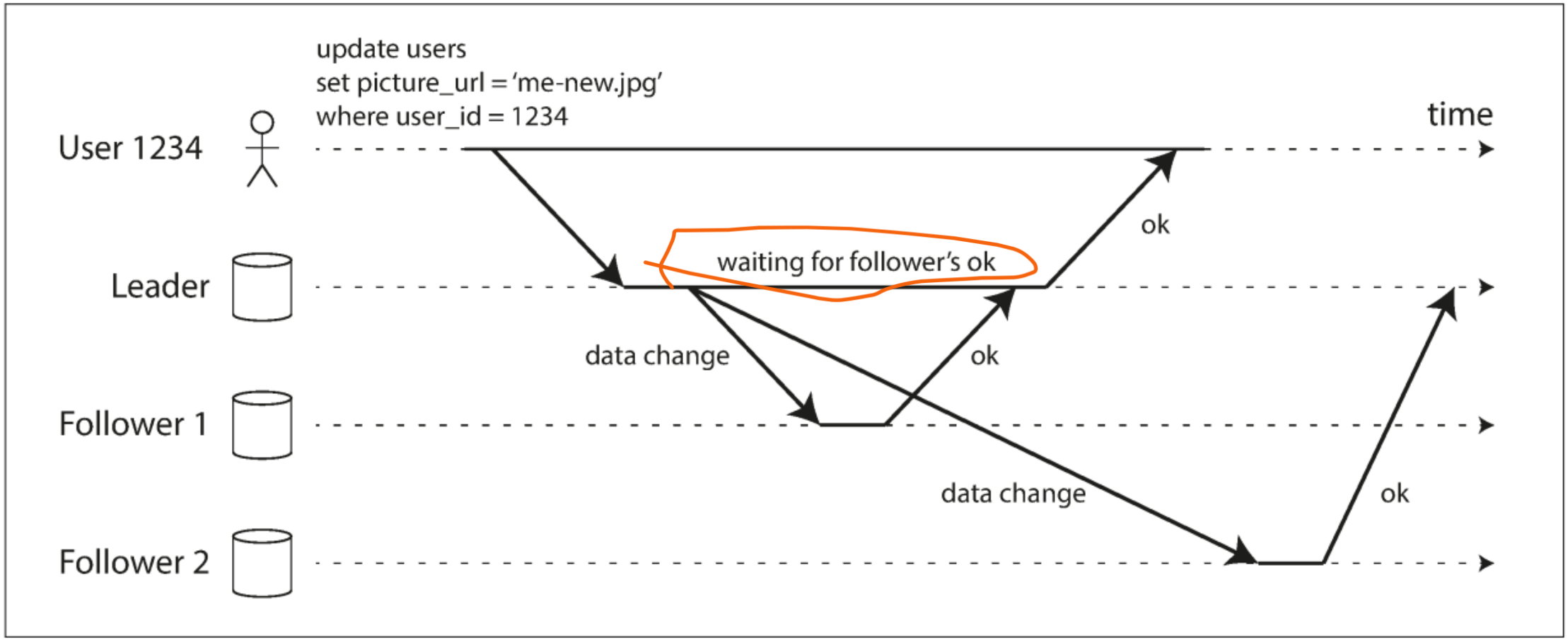


Figure 5-2. Leader-based replication with one synchronous and one asynchronous follower.

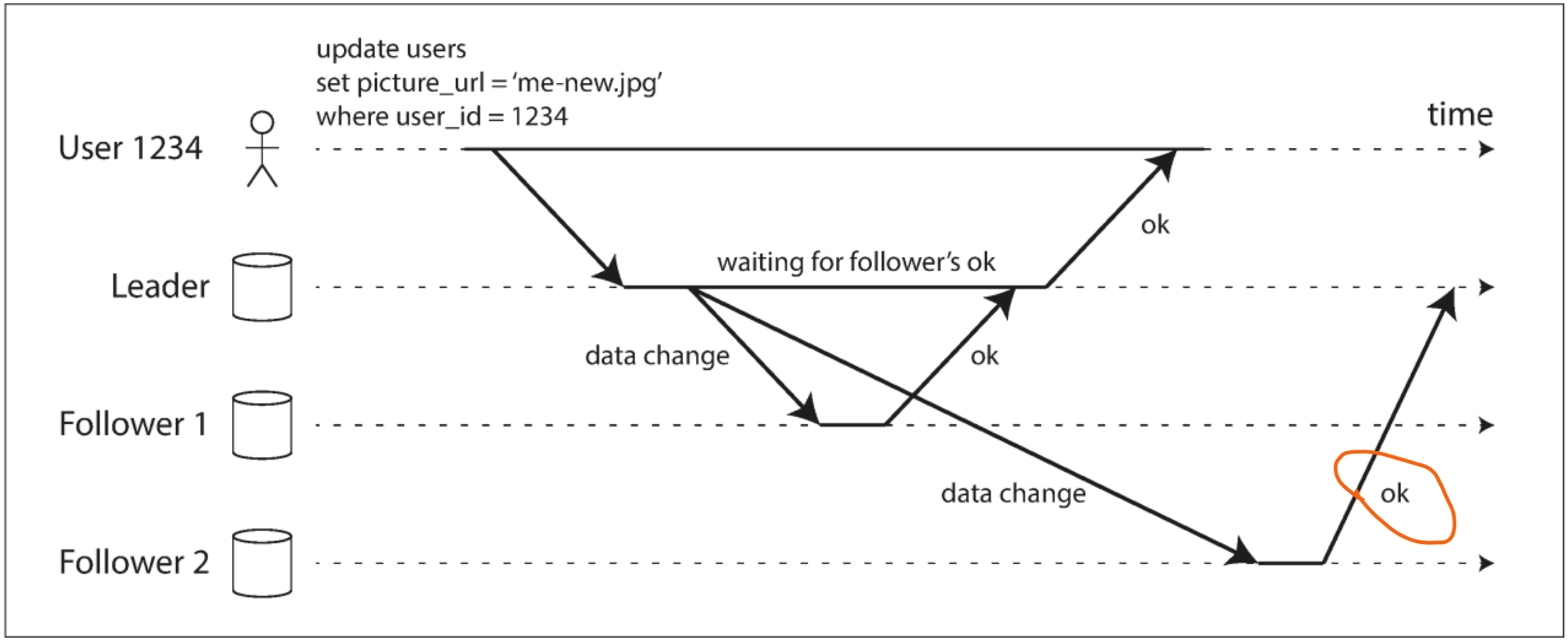


Figure 5-2. Leader-based replication with one synchronous and one asynchronous follower.

What happen if a replica fails?

- Leader
- Sync Replica
- Async Replica

How to add new replica?

- Why it is a problem?
- Snapshot / Backup of data?
  - *Some models:*
    - MVCC (Multi-Version Concurrency Control) model
      - *Most of SQL dbs*
    - Copy on write + Log-Based
      - *Redis*
    - …

Leader Failure - Failover

- If leader fails -> promote one of the followers to be leader.
- Is it easy?
- How we determine the leader fails?
  - *Who determines?*
  - *Timeout? How long?*

Leader Failure - Failover

- How to choose the new leader? Which node is up to date?
- How to reconfigure system for new leader?
  - *Writes*
  - *Other replicas?*
- What happen to writes that may not in new leader?
  - *The Github failure example*

Leader Failure - Failover

- What happen when the failed leader come back?
  - *Split Brain problem!*
- Manual operation or automatic operation?

Replication Log

- Statement based replication
  - repeat the original command
  - nondeterministic commands!
    - RAND(), NOW()
    - Side effects, …

Replication Log

- Write-ahead log (WAL) shipping
  - send detailed physical write change log
  - فلان بایت از فلان بلاک از بهمان فایل رو از این به اون تغییر دادم.
  - Problem with version upgrade!?

Replication Log

- Logical (row based) log shipping
  - send row level key and data
  - *On insert?*
  - *On update?*
  - *On delete?*
- Logical Replication Vs. Physical Replication?

Replication Log

- Trigger-based replication
  - *Manually codded*
  - *Why we can need it?*
  - *Flexible*
  - *Higher overhead*
  - *Risk bugs!*
- Postgresql Example

REPLICATION LAG

Replication Lag

- How Long?
  - *Seconds?*
  - *Minutes?*
  - *Hours / Days ??!*

Consistency

- Strong Consistency
- Eventual Consistency

Read your own writes

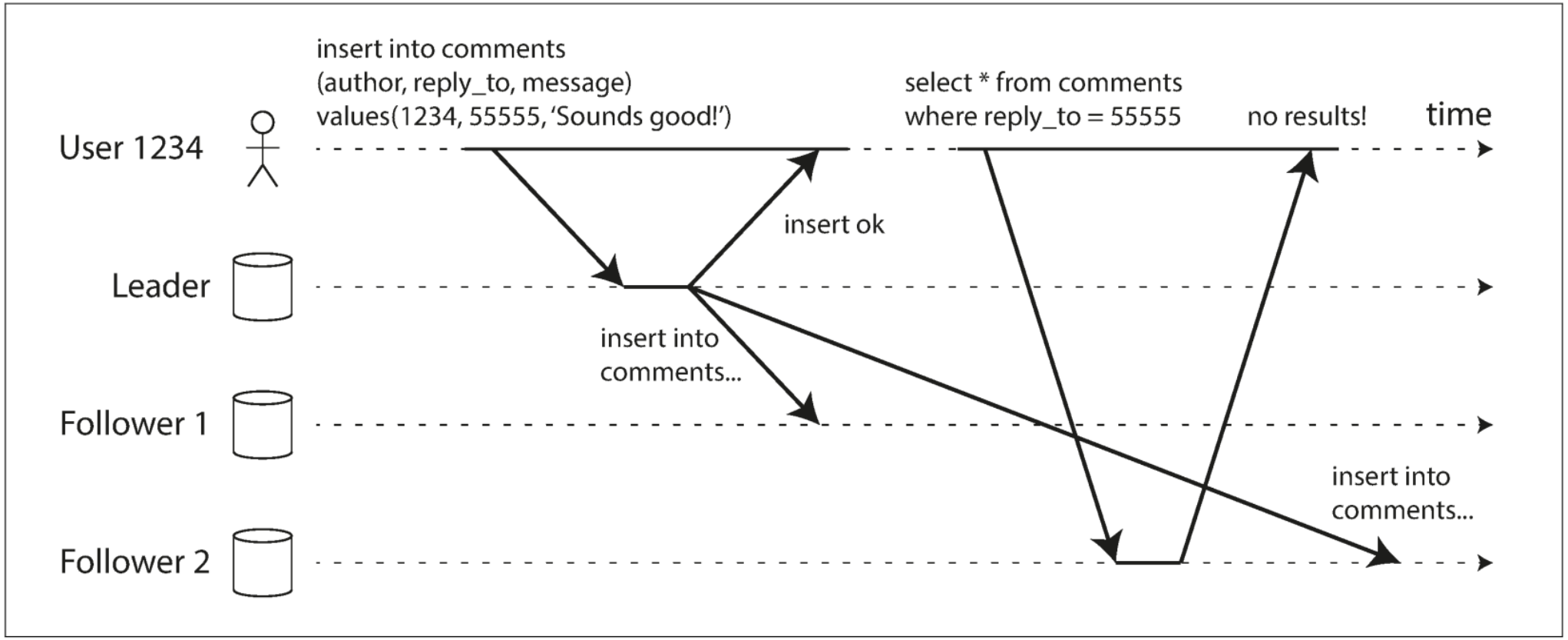


Figure 5-3. A user makes a write, followed by a read from a stale replica. To prevent this anomaly, we need read-after-write consistency.

Read-After-Write Consistency

- Or read-your-writes consistency
- Solutions?
  - *if few things modifiable, read them all from the leader*
  - *Track last update and read everything from leader for a short period*
  - *Save last timestamp of write*
- Cross-device read-after-write consistency?

Monotonic reads

- Comments of a site
- user makes two reads, the second is from a more lagged replica
  - *This will look to the user like information went backward in time*
- Solution?
  - *Each user, read from same replica…*

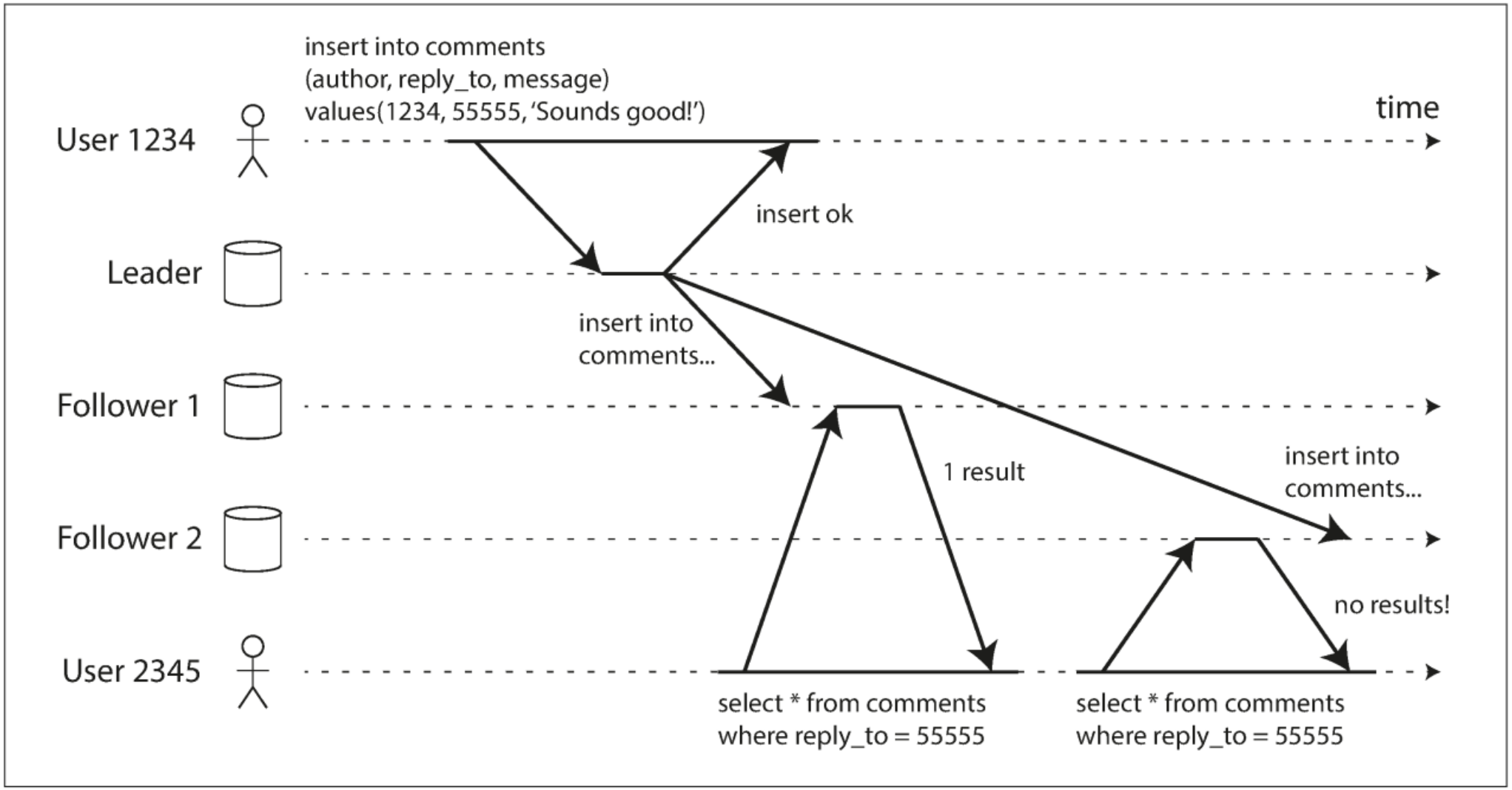


Figure 5-4. A user first reads from a fresh replica, then from a stale replica. Time appears to go backward. To prevent this anomaly, we need monotonic reads.

Monotonic reads

- Solution?
  - *Each user, read from same replica…*

Consistent prefix reads

- Assume we have multiple databases / partitions

Mr. Poons
How far into the future can you see, Mrs. Cake?

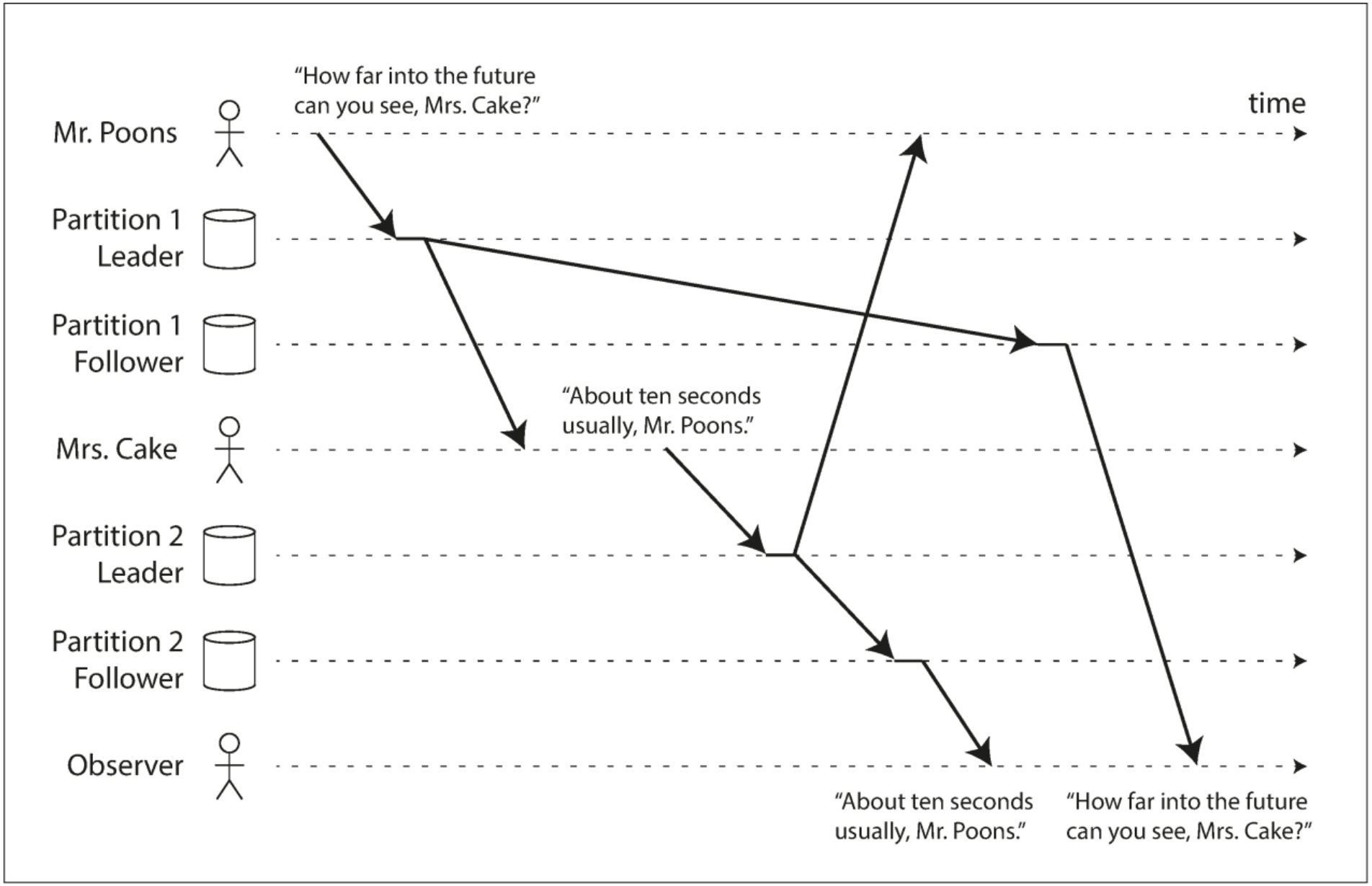
Mrs. Cake
About ten seconds usually, Mr. Poons.

Consistent prefix reads

- Now a third-person, read data and observe following:

Mrs. Cake
About ten seconds usually, Mr. Poons.

Mr. Poons
How far into the future can you see, Mrs. Cake?


"How far into the future can you see, Mrs. Cake?"
time
Mr. Poons
Partition 1 Leader
Partition 1 Follower
"About ten seconds usually, Mr. Poons."
Mrs. Cake
Partition 2 Leader
Partition 2 Follower
Observer
"About ten seconds usually, Mr. Poons."
"How far into the future can you see, Mrs. Cake?"

Solutions for Replication Lag?

- Transactions?

MULTI-LEADER REPLICATION

Multi-Leader Replication

- each leader also acts as a follower to the other leaders
- Use cases?
  - *Multi-data center operation*
  - *Clients with offline operation*
  - *Collaborative editing*

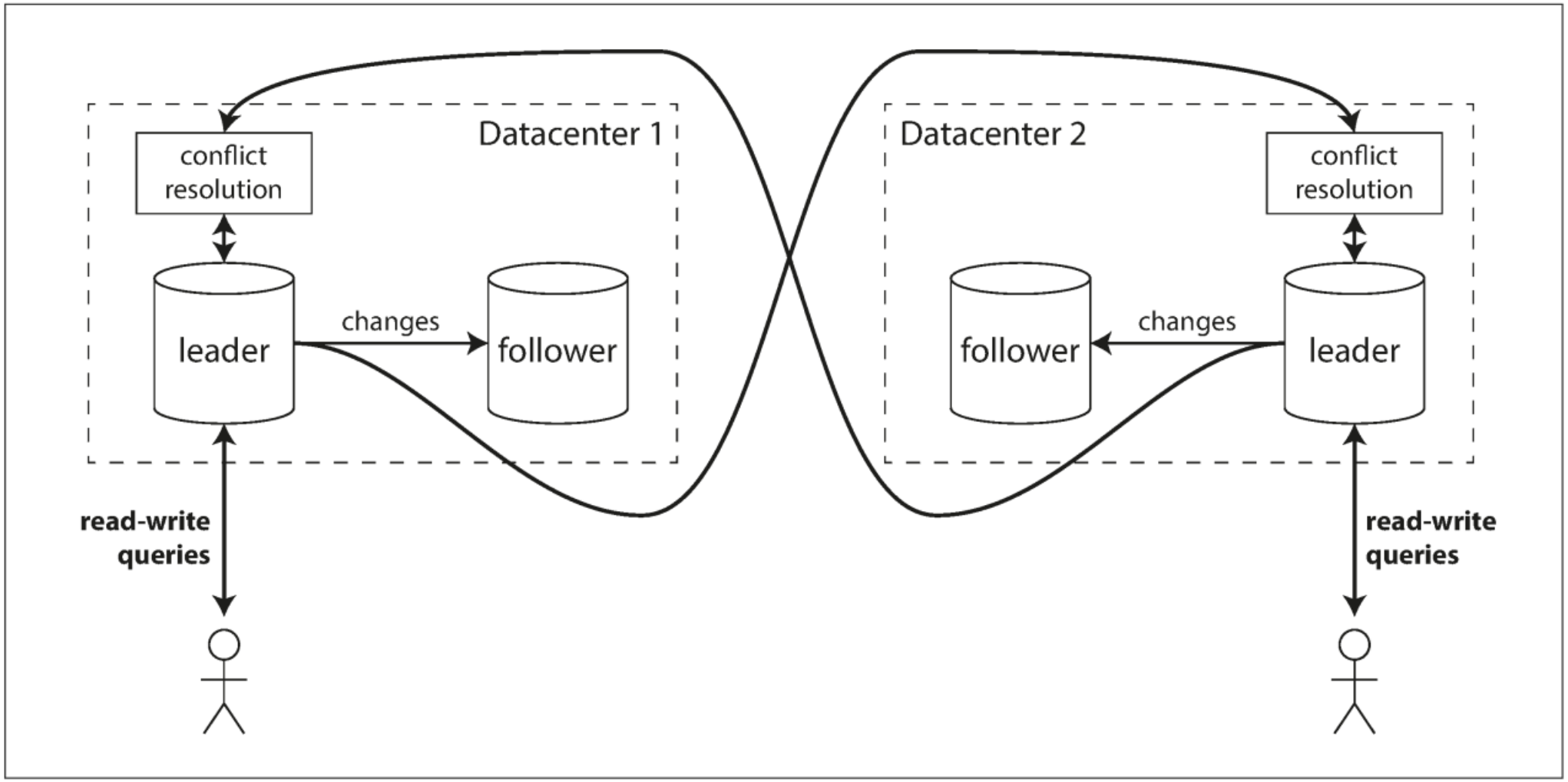


Figure 5-6. Multi-leader replication across multiple datacenters.

Handling Write Conflicts

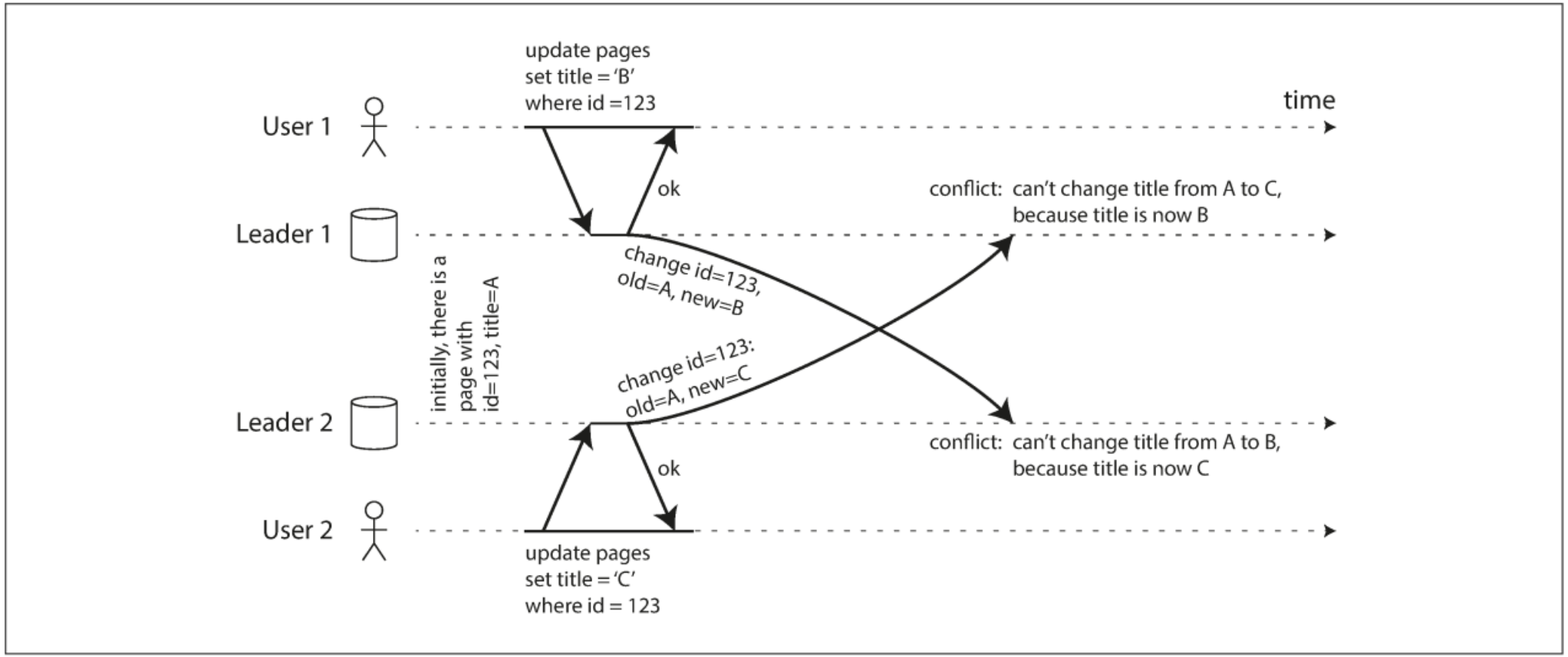


Figure 5-7. A write conflict caused by two leaders concurrently updating the same record.

Handling Write Conflicts

- Downside of multi-leader is that write conflicts can happen, requiring conflict resolution
- Why it's hard and have bad User Experience?
  - Conflicts are generally detected asynchronously, so too late to prompt user

Handling Write Conflicts

- Conflict avoidance
  - *Assigning each record a home datacenter works until a failure or they move*
  - *If you need to change the designated leader for a record -> conflict may happen ☹*

جلسه‌ی بعد