Data-Intensive Applications

THE BIG IDEAS BEHIND RELIABLE, SCALABLE, AND MAINTAINABLE SYSTEMS

Chapter 5 : Replication

Why you want to store data across multiple machines?

- Scalability :

  If your data volume, read load or write load grows bigger than a single machine can handle

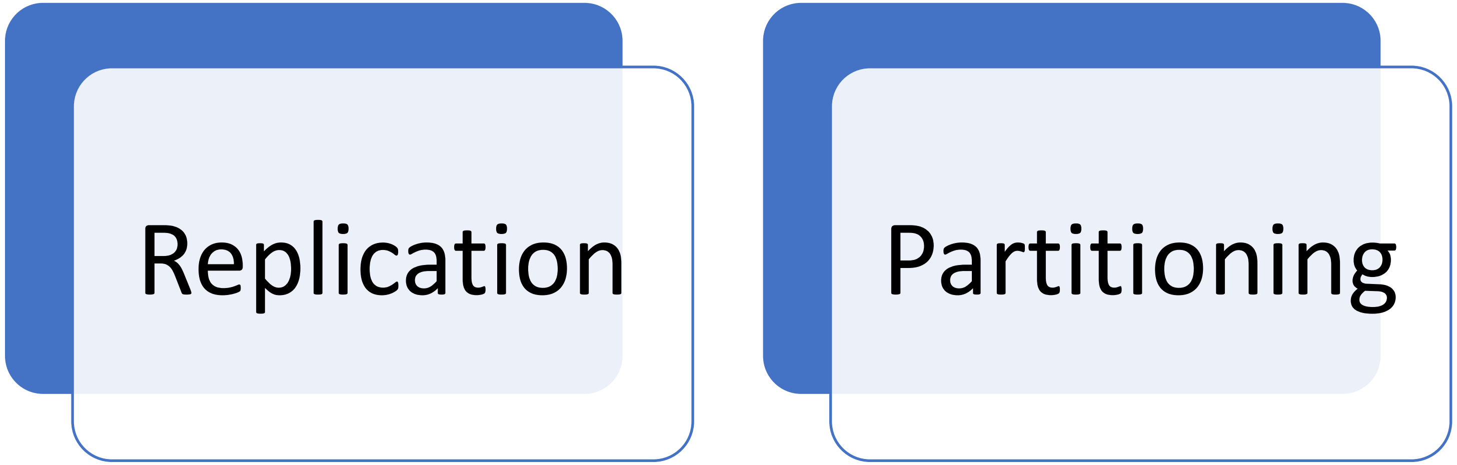
- Fault tolerance/High Availability:

  If one machine crashes or some failure occurs, you can enable redundancy

- Latency:

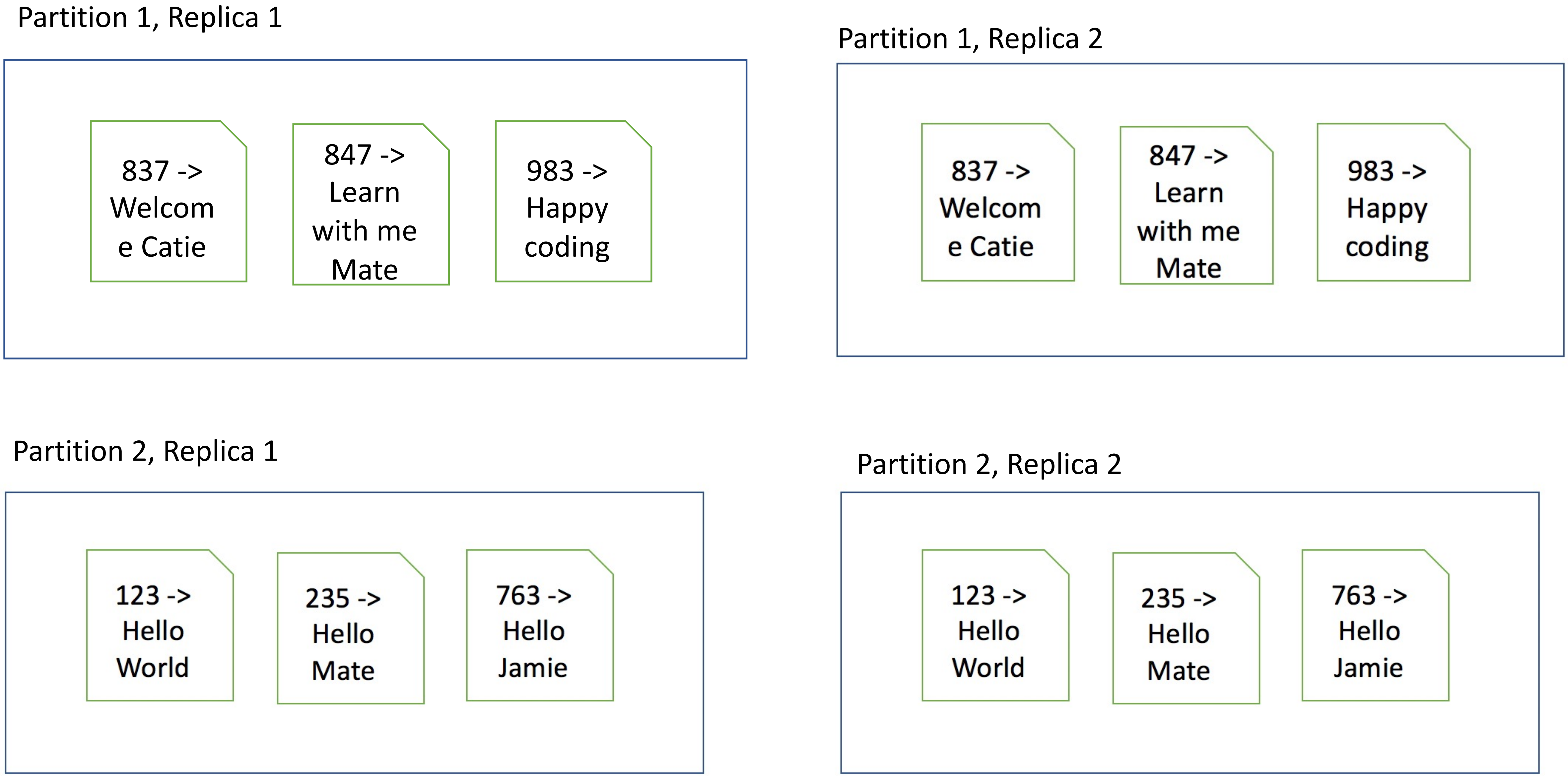
  Users are spread across different locations, serve from location that is geographically close to the users

How data is distributed across multiple machines?



Replication

Partitioning



Partition 1, Replica 1

837 -> Welcome Catie

847 -> Learn with me Mate

983 -> Happy coding

Partition 1, Replica 2

837 -> Welcome Catie

847 -> Learn with me Mate

983 -> Happy coding

Partition 2, Replica 1

123 -> Hello World

235 -> Hello Mate

763 -> Hello Jamie

Partition 2, Replica 2

123 -> Hello World

235 -> Hello Mate

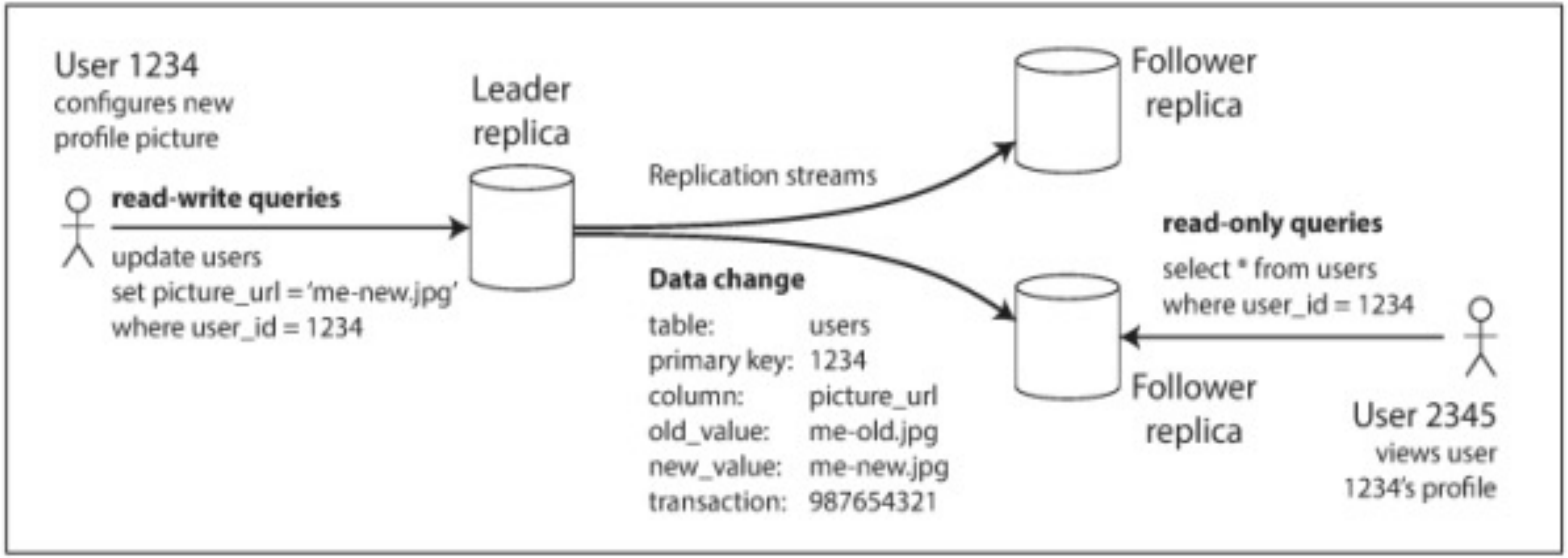
763 -> Hello Jamie

Replication

- Keep data geographically close to users -> Latency
- Fault tolerance – Availability
- Scale out for read and write queries

- Problem with replication: Handling changes to replicated data
- Approaches:
  - Single leader
  - Multi leader
  - Leaderless

Leaders and Followers



Leader based replication, active/passive or master-slave replication

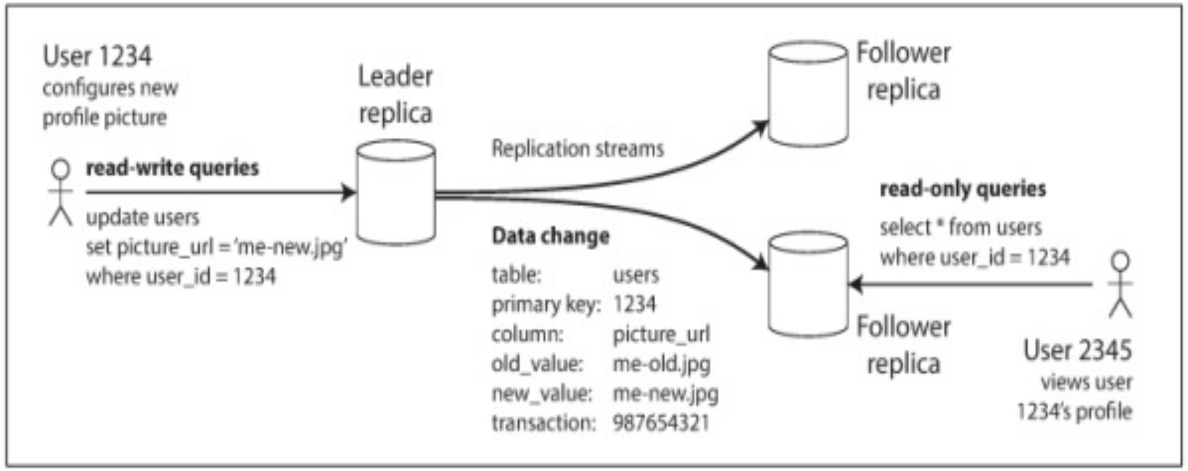
Synchronous vs Asynchronous Replication

- Synchronous
  - Follower is guaranteed to have an up-to-date copy of data that is consistent with the leader.
  - If leader suddenly fails, data will be still available on follower.
  - If sync follower does not respond, leader must block all writes, till the sync replica is available.
  - Generally sync replication in database means- one follower is sync and other followers are async – Semi synchronous
- Asynchronous
  - If leader fails and is not recoverable, writes get lost, but it guarantees that leader can continue processing writes.

How can we set up the followers?

- Take consistent snapshot of leader’s database at some point in time.
- Copy the snapshot to the new follower node.
- Follower connects to the leader and requests all the data changes that have happened since the snapshot was taken. Mysql – binlog coordinates..
- Now when follower has caught up, it can process the data changes.

How can we achieve high availability using leader-based replication?



```
update users
set picture_url = 'me-new.jpg'
where user_id = 1234
```

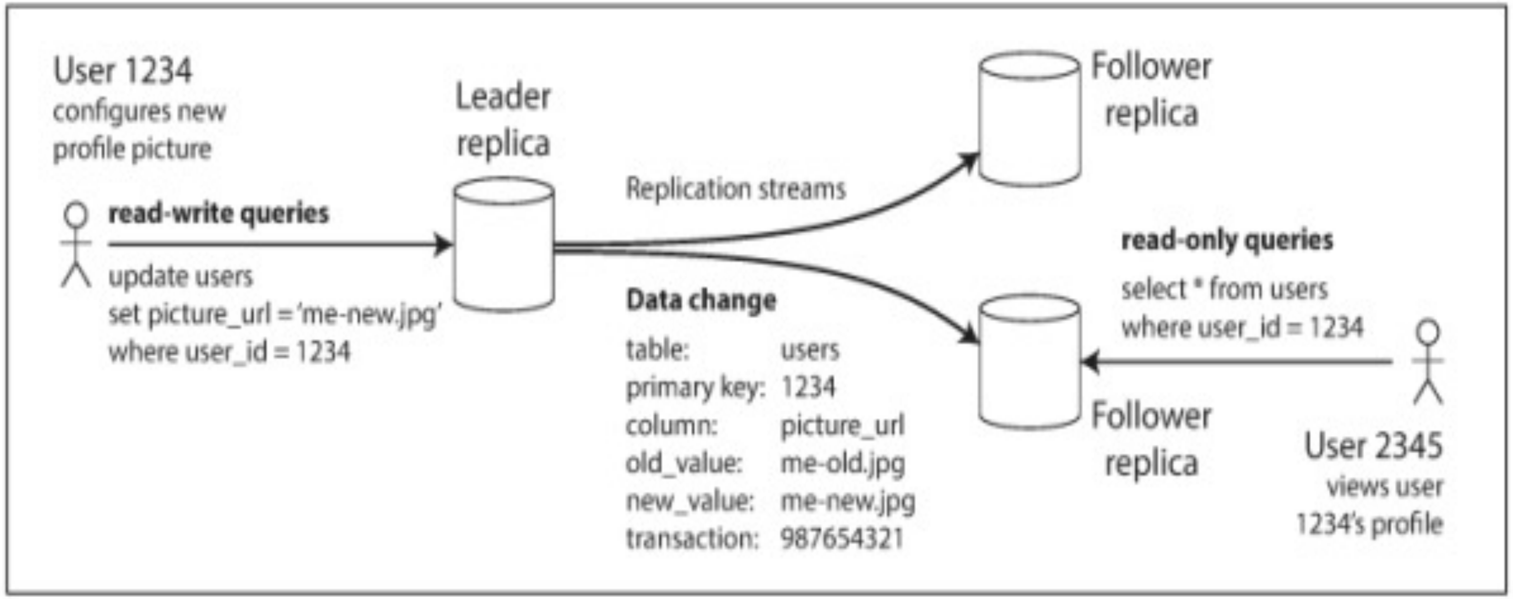
```
table:        users
primary key:  1234
column:       picture_url
old_value:    me-old.jpg
new_value:    me-new.jpg
transaction:  987654321
```

```
select * from users
where user_id = 1234
```

- **Follower failure – Catch up Recovery**
  - Local disk – contains log

- **Leader failure –Failover**
  - One of the follower needs to be promoted to the new leader
  - Client needs to be reconfigured to send writes to the new leader
  - Other followers start consuming data changes from new leader
  - Automatic failover process:
    1. Determining that leader has failed - using timeouts
    2. Choosing a new leader – Consensus problem
    3. Reconfiguring the system to use the new leader

Issues with automatic failover



- If async replication – new leader might not be up-to date. If former leader joins back the cluster, then conflicting writes
- Discarding writes is dangerous.
- Split brain

Implementation of Replication logs

- Statement based replication logs
- Write-ahead log(WAL)
- Logical(row-based) log replication
- Trigger-based replication

Statement based replication

- Leader logs every request that it executes and sends that statement to its followers.
- Issues:
  - Non-deterministic functions like now(), rand() can generate different results on different replicas.
  - If statements are using auto increment column, then multiple concurrent statements can be an issue.
  - Statements that trigger procedures etc. can result in different outcomes.

Write-ahead Logs

- In case of log-structured engines, this WAL is the main place for storage.
- In case of B-trees, which overwrites individual disk blocks, every modification is written to WAL first.
- Log describes data on very low level, it contains details of which bytes were changed in which disk blocks

Logical(row-based) log replication

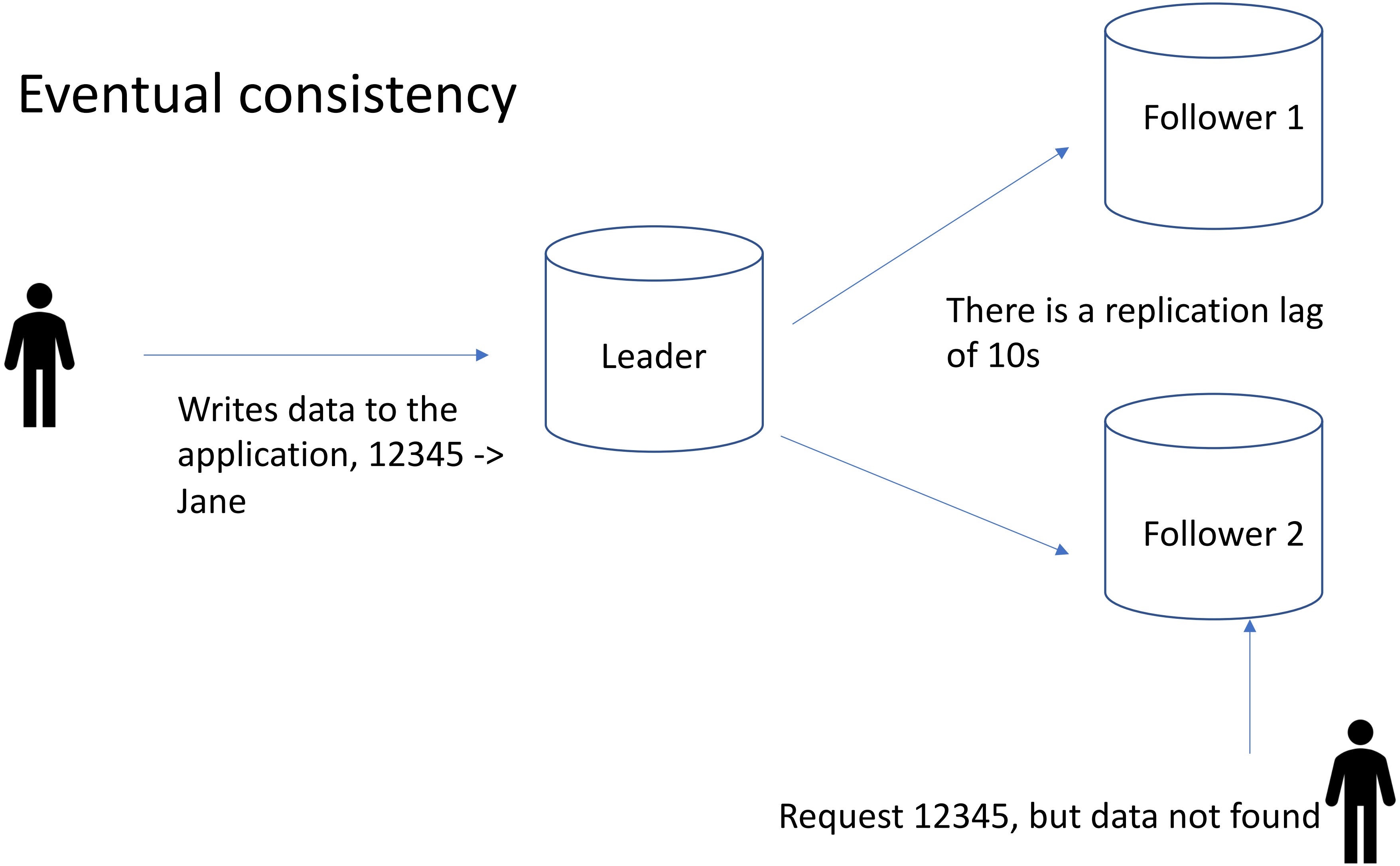
- Different log formats for replication and storage engines – logical log- distinguished from storage engine’s (physical) data representation.
- Logical log contains:
  - Inserted row – log contains new values of all columns
  - Deleted row – Primary key, and old value of all columns
  - Updated row – Primary key and new values of all columns
  - Then a log indicating – transaction was committed.

Trigger-based replication

Trigger – code that is automatically executed when a data change occurs in database system.

Problems with replication lag

- Eventual consistency



Follower 1

Leader

There is a replication lag of 10s

Follower 2

Writes data to the application, 12345 -> Jane

Request 12345, but data not found

Approaches to Replication Lag

- Reading your own writes
- Read-after-write consistency/ read-your-writes consistency
- Monotonic reads
- Consistent Prefix reads

Summary

- ✓ What is Replication?
- ✓ Why Replication is needed?
- ✓ Single Leader Replication
- ✓ Replication Lag

Thank You!