DTCC

2016中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2016

数 据 定 义 未 来

SequeMedia 盛拓传媒　IT168.com　ChinaUnix　ITPUB

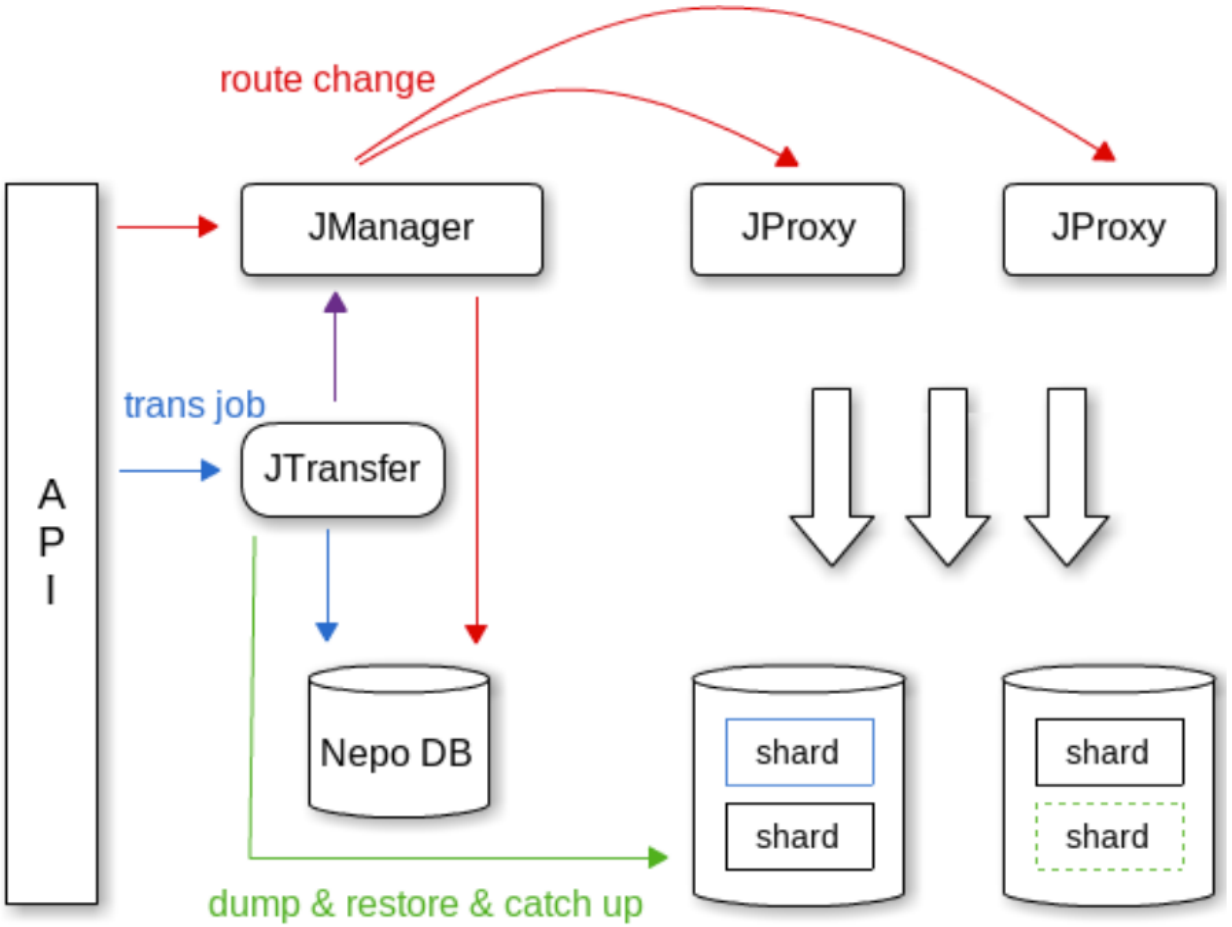
the era of databases

- SQL / RDBMS
  - Wide adoption,great for apps
  - Poor horizontal scalability
- NoSQL / SQL Middleware
  - Scalable,available
  - Manual Join, No Transaction
- NewSQL
  - Scalable, available, consistent
  - Full Transaction

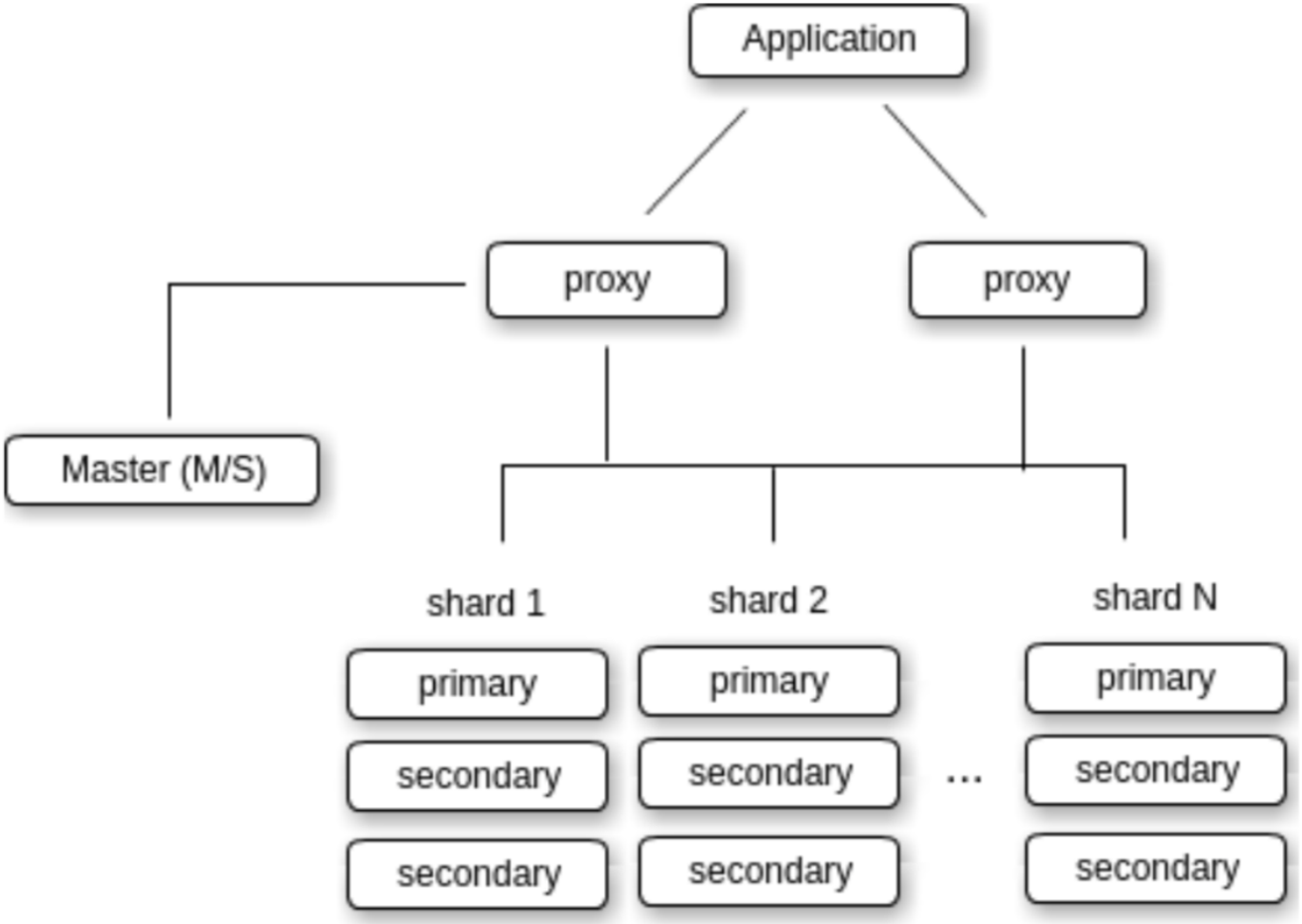
databases at Google

- **2004: BigTable**
  - eventually consistent NoSQL
- **2006: Megastore (on top of BigTable)**
  - transactional, slow, complex
- **2012: Spanner (+F1 on top of it)**
  - semi-relational, fully transactional

Problems of Middleware



Same As NoSQL

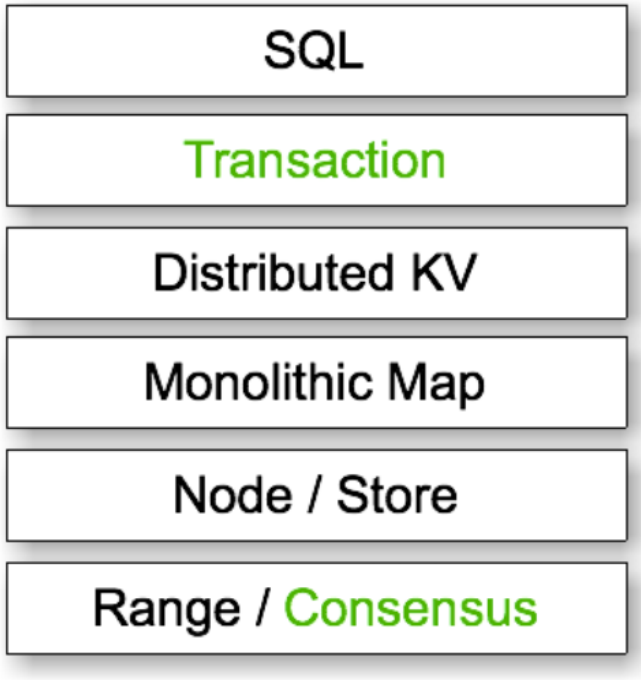


the real needs of applications

- **Apps shouldn't worry about:**
  - Sharding
  - Failover
  - Eventual consistency
- **Apps need transactions**
- **Apps need SQL**
- **Apps need ACID**

Architecture

- **The whole space is divided into ranges**
- **Using RSM to ensure consistency**
- **Node is consisted of ranges**
- **Monolithic Map is consisted of nodes**
- **Transaction is on top of the Monolithic Map**



Distributed Transaction

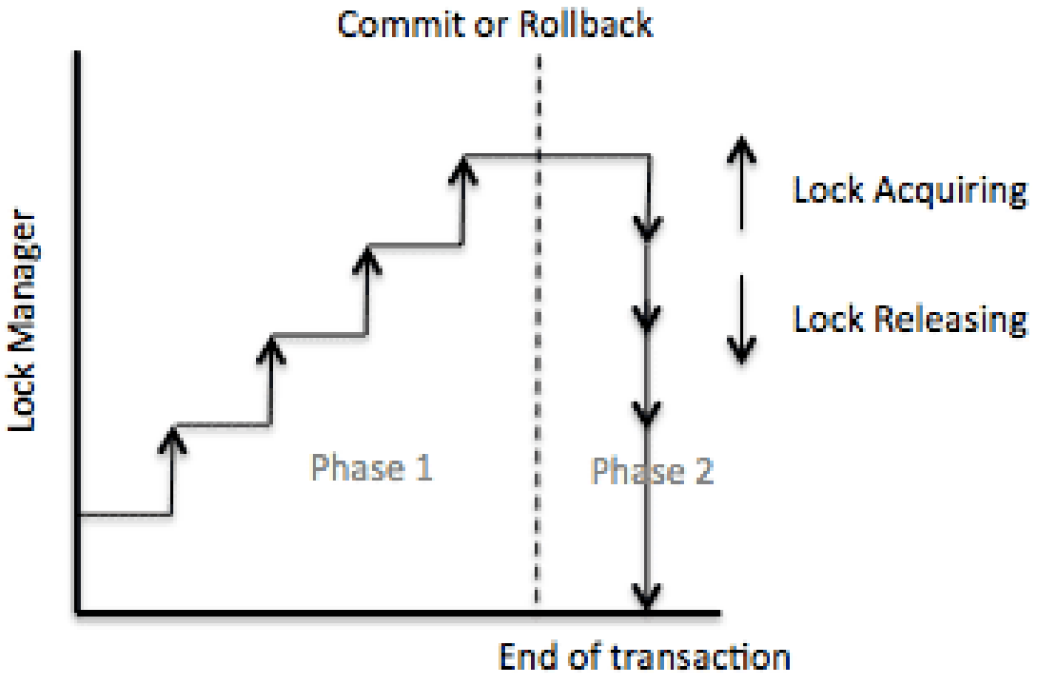
- At lease support snapshot isolation
- Using variation of two phase commit
- Using two phase lock or lock free algorithm
- All the modification of data is just a single version of that data
- Using timestamp as a part of the key
- Commit timestamp may increase as transaction runs

KV Primitives

- Get(key)
- Put(key, value)
- ConditionalPut(key, value, expValue)
- Scan(startKey, endKey)
- Del(key)

Two Phase Locking

- **lock is processed through two phases**
  - expanding phase
  - modification of data
  - shrinking phase

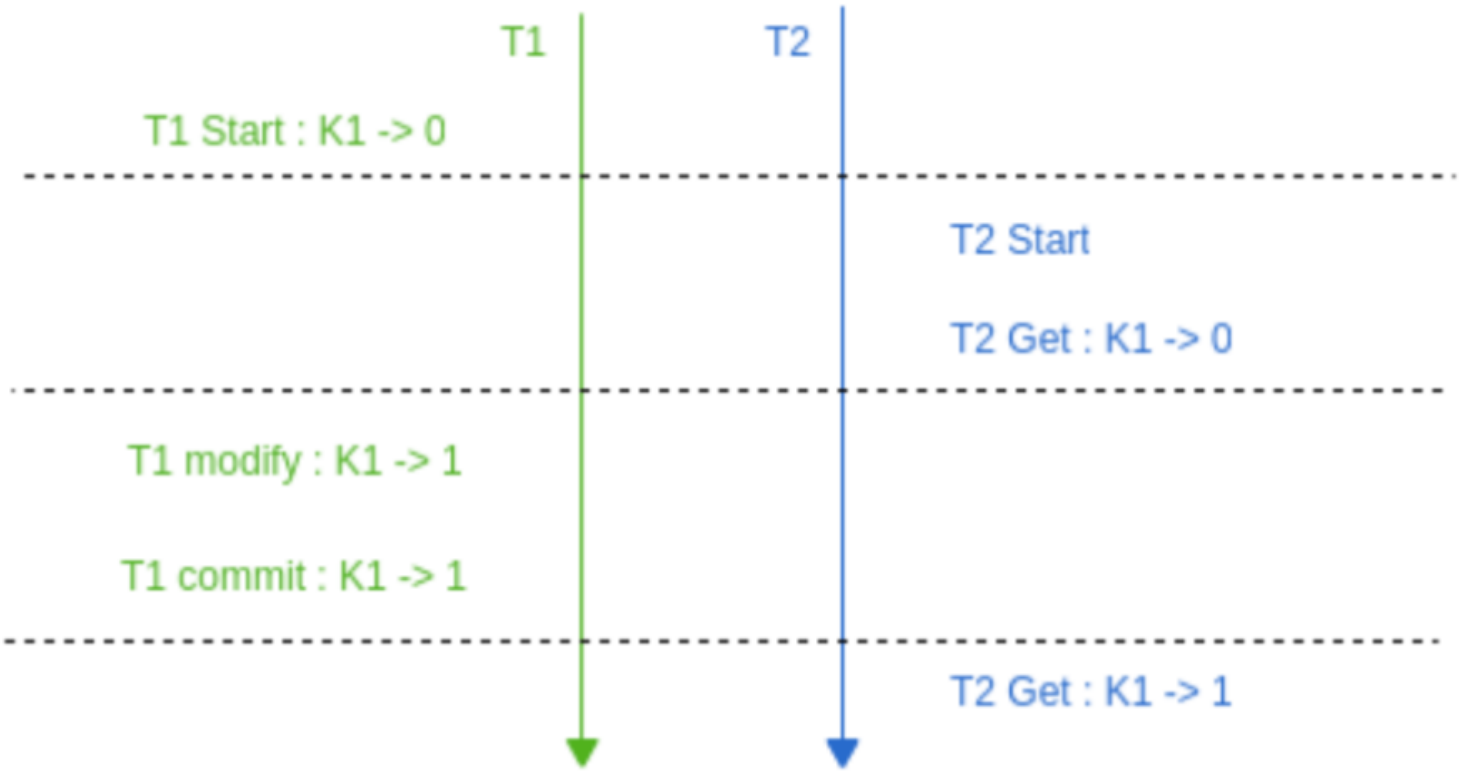


About Timestamp

- **Absolute time in distributed system is extremely hard**
  - NTP is obviously not workable
  - TrueTime API
  - Hyper logic clock
- **We all need to deal with the uncertainty of time**

Time Matters

A simple case which breaks the repeatable read isolation



An Example

Transfers 7 dollars from Bob to Joe

key	*bal:data*	*bal:lock*	*bal:write*
Bob	*6:* *5:* $10	*6:* *5:*	*6:* data @ 5 *5:*
Joe	*6:* *5:* $2	*6:* *5:*	*6:* data @ 5 *5:*

1. Initial state: Joe's account contains $2 dollars, Bob's $10.

An Example

Bob	*7:* **$3** *6:* *5:* $10	*7:* **I am primary** *6:* *5:*	*7:* *6:* data @ 5 *5:*
Joe	*6:* *5:* $2	*6:* *5:*	*6:* data @ 5 *5:*

2. The transfer transaction begins by locking Bob's account balance by writing the lock column. This lock is the primary for the transaction. The transaction also writes data at its start timestamp, 7.

An Example

Bob	*7:* $3 *6:* *5:* $10	*7:* I am primary *6:* *5:*	*7:* *6:* data @ 5 *5:*
Joe	***7:* $9** *6:* *5:* $2	***7:* primary @ Bob.bal** *6:* *5:*	*7:* *6:* data @ 5 *5:*

3. The transaction now locks Joe's account and writes Joe's new balance (again, at the start timestamp). The lock is a secondary for the transaction and contains a reference to the primary lock (stored in row "Bob," column "bal"); in case this lock is stranded due to a crash, a transaction that wishes to clean up the lock needs the location of the primary to synchronize the cleanup.

An Example

Bob	*8:* *7:* $3 *6:* *5:* $10	*8:* ***7:*** *6:* *5:*	*8:* **data @ 7** *7:* *6:* data @ 5 *5:*
Joe	*7:* $9 *6:* *5:* $2	*7:* primary @ Bob.bal *6:* *5:*	*7:* *6:*data @ 5 *5:*

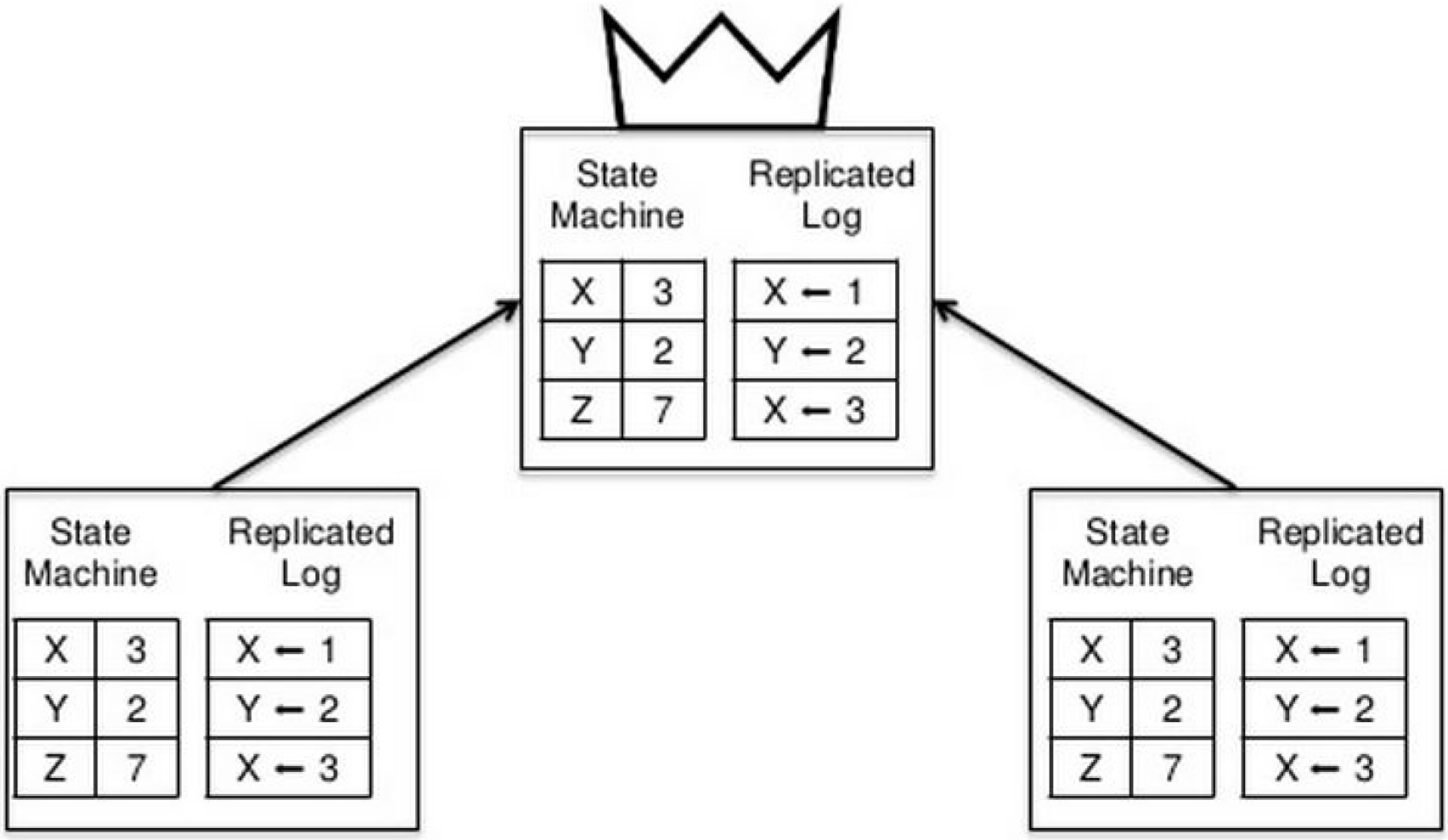
4. The transaction has now reached the commit point: it erases the primary lock and replaces it with a write record at a new timestamp (called the commit timestamp): 8. The write record contains a pointer to the timestamp where the data is stored. Future readers of the column “bal” in row “Bob” will now see the value $3.

An Example

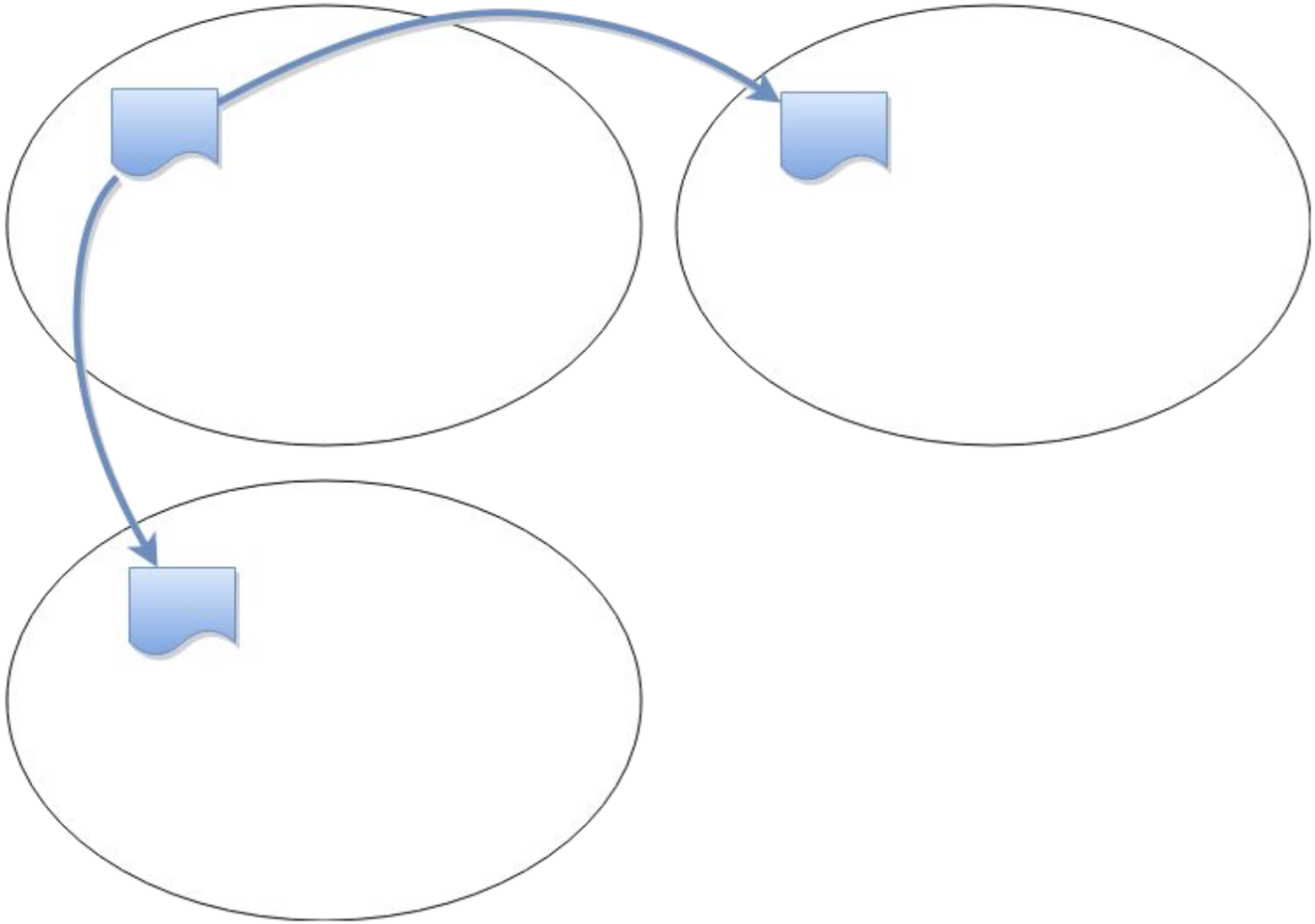
Bob	*8:* *7:* $3 *6:* *5:* $10	*8:* *7:* *6:* *5:*	*8:* data @ 7 *7:* *6:* data @ 5 *5:*
Joe	*8:* *7:* $9 *6:* *5:*$2	*8:* ***7:*** *6:* *5:*	***8:* data @ 7** *7:* *6:* data @ 5 *5:*

5. The transaction completes by adding write records and deleting locks at the secondary cells. In this case, there is only one secondary: Joe.

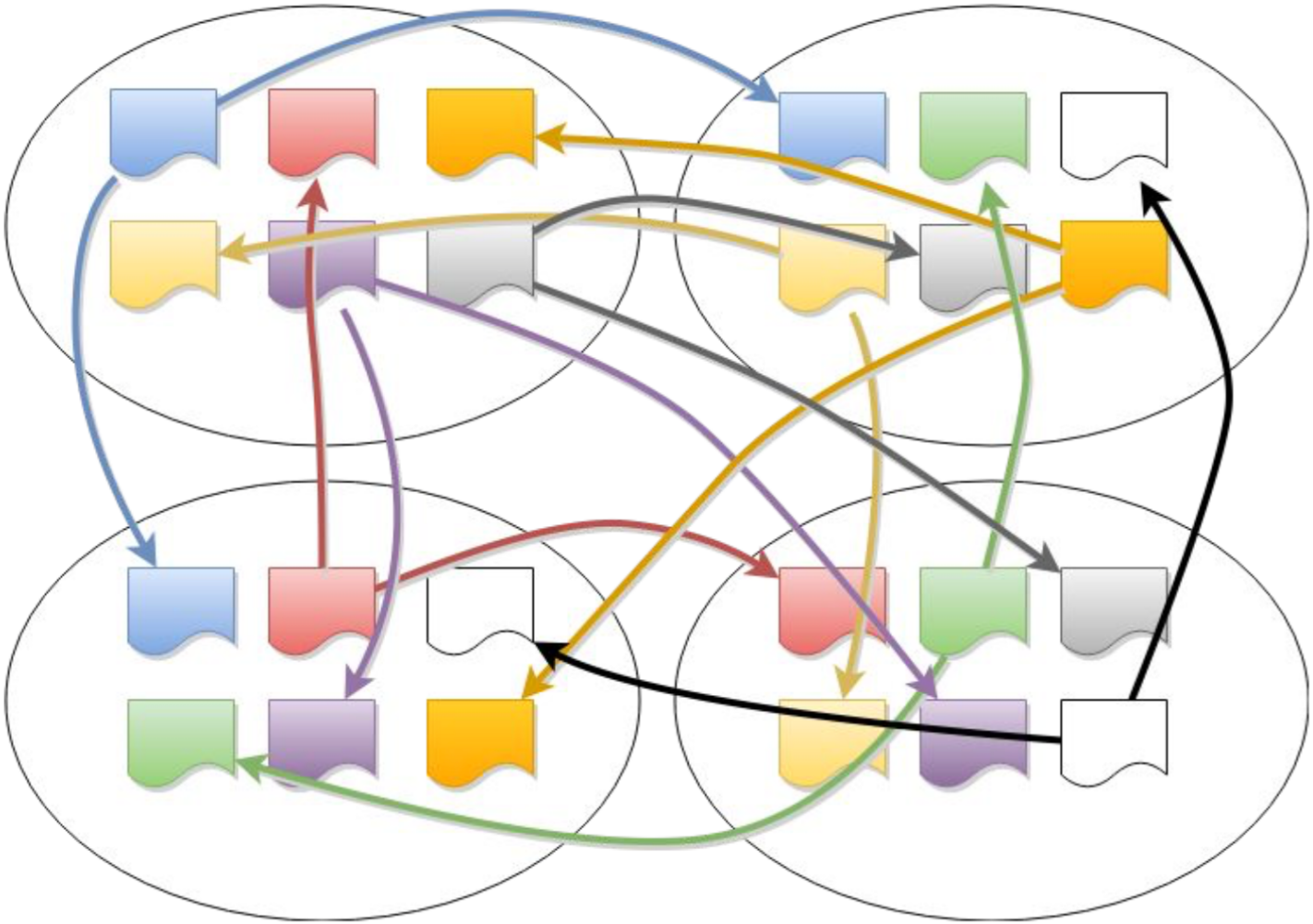
About Consensus



About Consensus



About Consensus



SQL layer on top of kv

- **All tables have a primary key**
- **One key/value pair per column**
- **key structure**
  - /<table>/<index>/<pkey>/<column>

An example

CREATE TABLE test (id integer primary key, name varchar, age int);

INSERT INTO test values(1, “Tom”, 28), (2, “Mike”, 30);

KEY	Value
/test/primary/1/name	“Tom”
/test/primary/1/age	28
/test/primary/2/name	“Mike”
/test/primary/2/age	30

THANKS

SequeMedia 盛拓传媒 IT168.com ChinaUnix ITPUB