MongoDB Multi-Document ACID Transactions:

Deep Dive

Sheeri Cabral & Aly Cabral: MongoDB Product Engineering
@sheeri @Aly_Cabral

Agenda

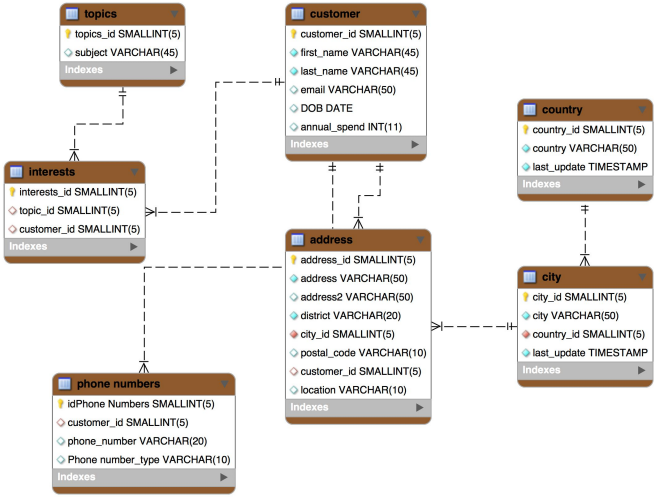
45 minutes

Part 1 (previously): Getting Started

- Why transactions, design goals, and use cases
- Transactions sessions and syntax
- Code samples

Part 2 (today): Deep Dive

- Best practices for developers
- Best practices for DBAs
- Tips and tricks



Tabular (Relational) Database

Related data split across multiple records and tables.
Multi-record transactions essential

```
{
  "_id" : ObjectId("5ad88534e3632e1a35a58d00"),
  "name" : {
    "first" : "John",
    "last" : "Doe" },
  "address" : [
    { "location" : "work",
      "address" : {
        "street" : "16 Hatfields",
        "city" : "London",
        "postal_code" : "SE1 8DJ"},
      "geo" : { "type" : "Point", "coord" : [
          51.5065752,-0.109081]}},
+   {...}
  ],
  "phone" : [
    { "location" : "work",
      "number" : "+44-1234567890"},
+   {...}
  ],
  "dob" : ISODate("1977-04-01T05:00:00Z"),
  "retirement_fund" : NumberDecimal("1292815.75")
}
```

Document Database

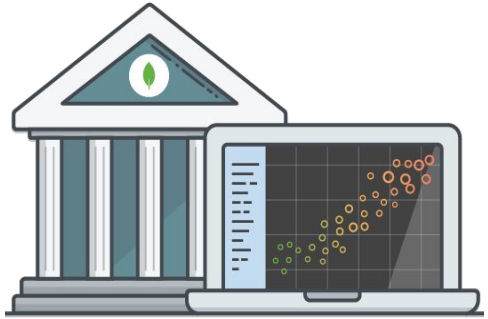
Related data contained in a single, rich document.
Transaction scoped to the document

mongoDB

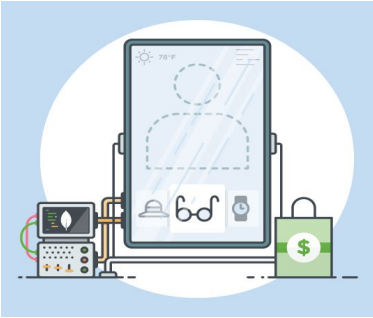
Transactions Simplify Development

- Don’t need to worry about interim states
  - Either a customer is signed-up or they are not
- Rollback/Abort is handled automatically if any single statement fails
- Point-in-time consistency across reads/writes

Transactions Use Cases



Many to Many
Relationships
Positions and Trades



Event Processing
Account Creation

Application Event Logging
Application Auditing

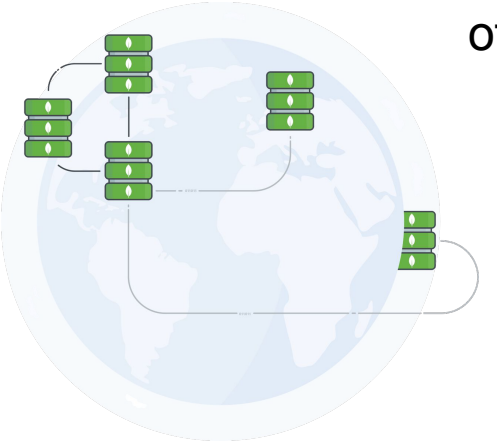
“ACID transactions are a key capability for business critical transactional systems, specifically around commerce processing. ***No other database*** *has both the power of NoSQL and cross collection ACID transaction support.*

This combination will make it easy for developers to write mission critical applications leveraging the power of MongoDB.”

Dharmesh Panchmatia, Director of E-commerce

ACID transactional guarantees
of relational databases

Developer productivity
of document databases



Freedom to **Run Anywhere**

Scale-out, data locality, and resilience of distributed systems

The Journey to Transactions

Major engineering investment over 3+ years touching every part of the server and drivers

- Storage layer
- Replication consensus protocol
- Sharding architecture
- Consistency and durability guarantees
- Global logical clock
- Cluster metadata management
- Exposed to drivers through API enhancements

Delivering on the Transactions Roadmap

MongoDB 3.0	MongoDB 3.2	MongoDB 3.4	MongoDB 3.6	MongoDB 4.0	MongoDB 4.2
New Storage engine (WiredTiger)	Enhanced replication protocol: stricter consistency & durability	Shard membership awareness	Consistent secondary reads in sharded clusters	Replica Set Transactions	Distributed Transactions
	WiredTiger default storage engine		Logical sessions	Make catalog timestamp-aware	Oplog applier prepare support
	Config server manageability improvements		Retryable writes	Snapshot reads	Distributed commit protocol
	Read concern “majority”		Causal Consistency	Recoverable rollback via WT checkpoints	Global point-in-time reads
			Cluster-wide logical clock	Recover to a timestamp	More extensive WiredTiger repair
			Storage API to changes to use timestamps	Sharded catalog improvements	Transaction manager
			Read concern majority feature always available		
			Collection catalog versioning		
			UUIDs in sharding		
			Fast in-place updates to large documents in WT		

mongoDB

MongoDB Transactions are ACID Compliant

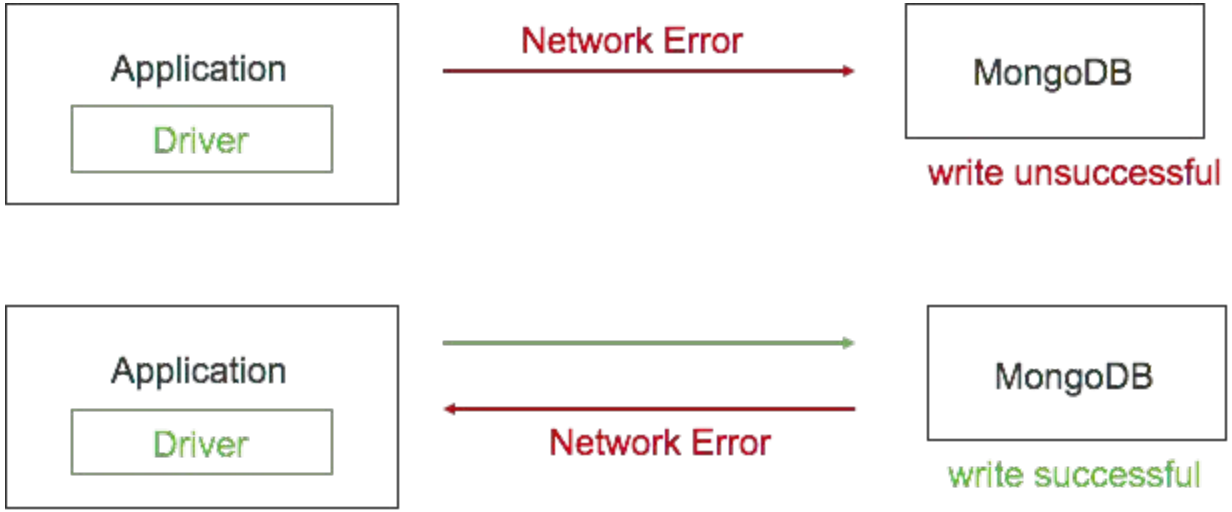
(yes, even across shards)

New API in MongoDB 4.2!

```
def insertDocuments(s):
    s.client.db.col.insert_one({'abc': 1}, session=s)
    s.client.db.col.insert_one({'xyz': 999}, session=s)

with client.start_session() as s:
    s.with_transaction(insertDocuments);
```

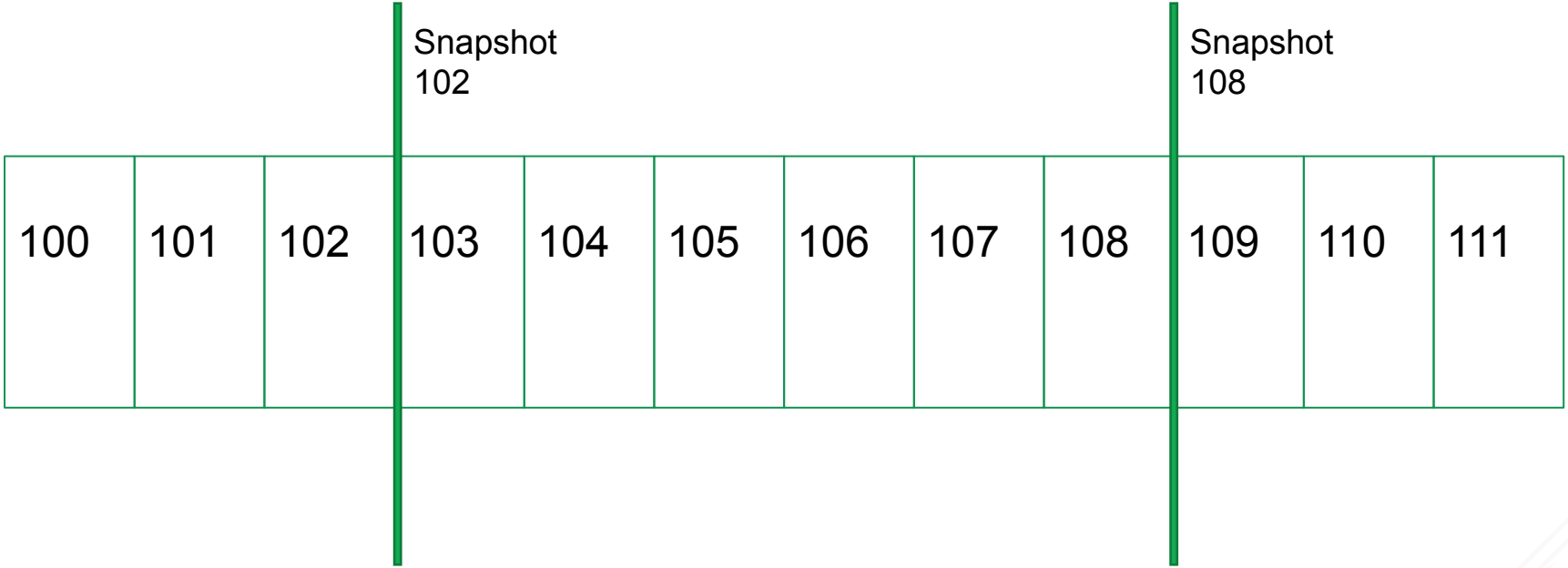
mongoDB


Application
Driver
Network Error
MongoDB
write unsuccessful
Application
Driver
Network Error
MongoDB
write successful

1) All data modeling rules still apply
2) Transactions should not be the most common operation (sanity check for rule 1)
3) Pass in the session to all statements
4) Implement retry logic, transactions can always abort
5) Don’t unnecessarily leave snapshots open
6) To trigger write conflicts, make sure you’re doing writes
7) Plan for DDL Operations

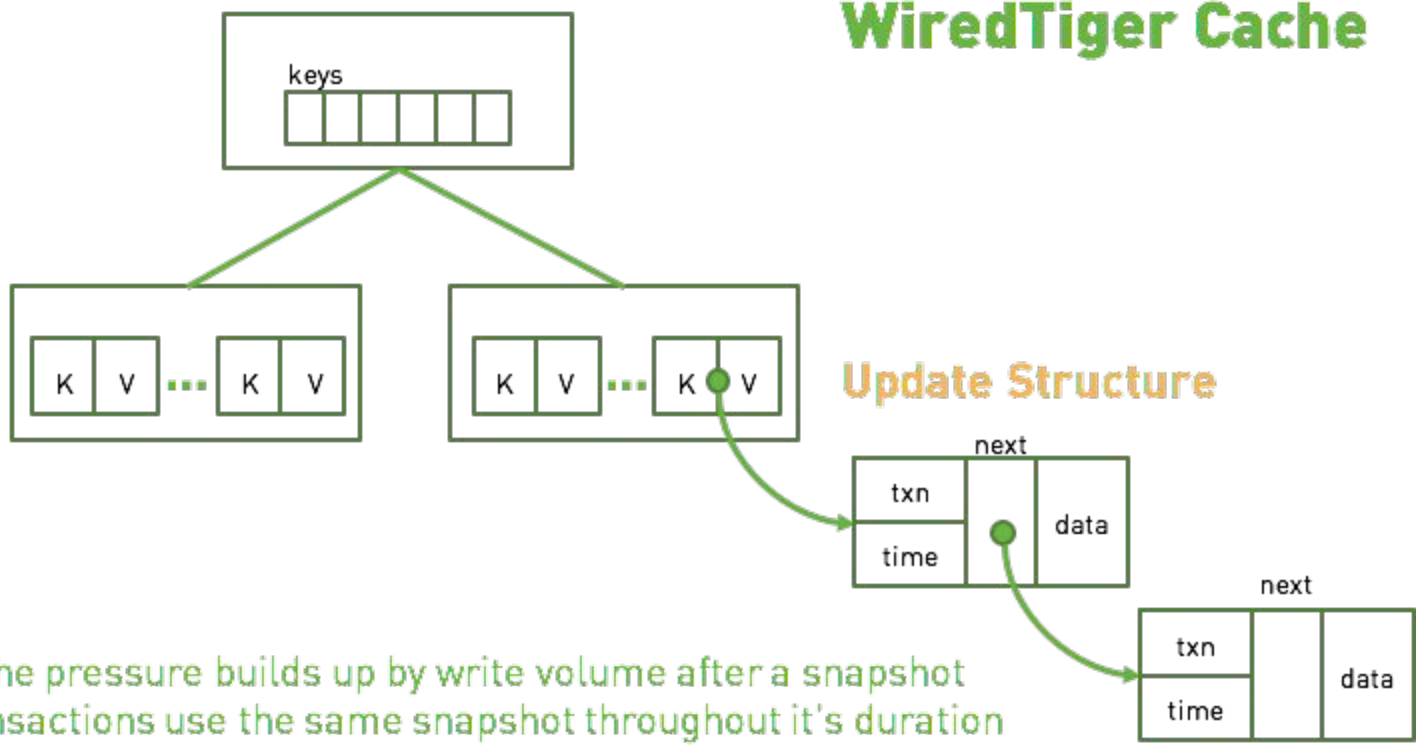
Operations Best Practices

Snapshot Isolation



mongoDB

WiredTiger Cache



1) Cache pressure builds up by write volume after a snapshot
2) Transactions use the same snapshot throughout it's duration
3) The update structure can only be cleaned up after the snapshot is evicted

Transactions Larger than 16MB

MongoDB 4.0:

```
{ <statement 1>,
  <statement 2>,
  <statement 3>,
  <statement 4>}
```

MongoDB 4.2:

```
{<statement 1>},
{<statement 2>},
{<statement 3>},
{<statement 4>}
```

This does not mean you can have infinitely large transactions --

WT still has to maintain the history since the snapshot time

Transactions automatically abort after 60 seconds – TUNABLE. **transactionLifetimeLimitSeconds**

Mutable Shard Key Values

Tiered storage: Aging out older documents to low cost storage shard

Global-redistribution: Rehoming documents to a new region

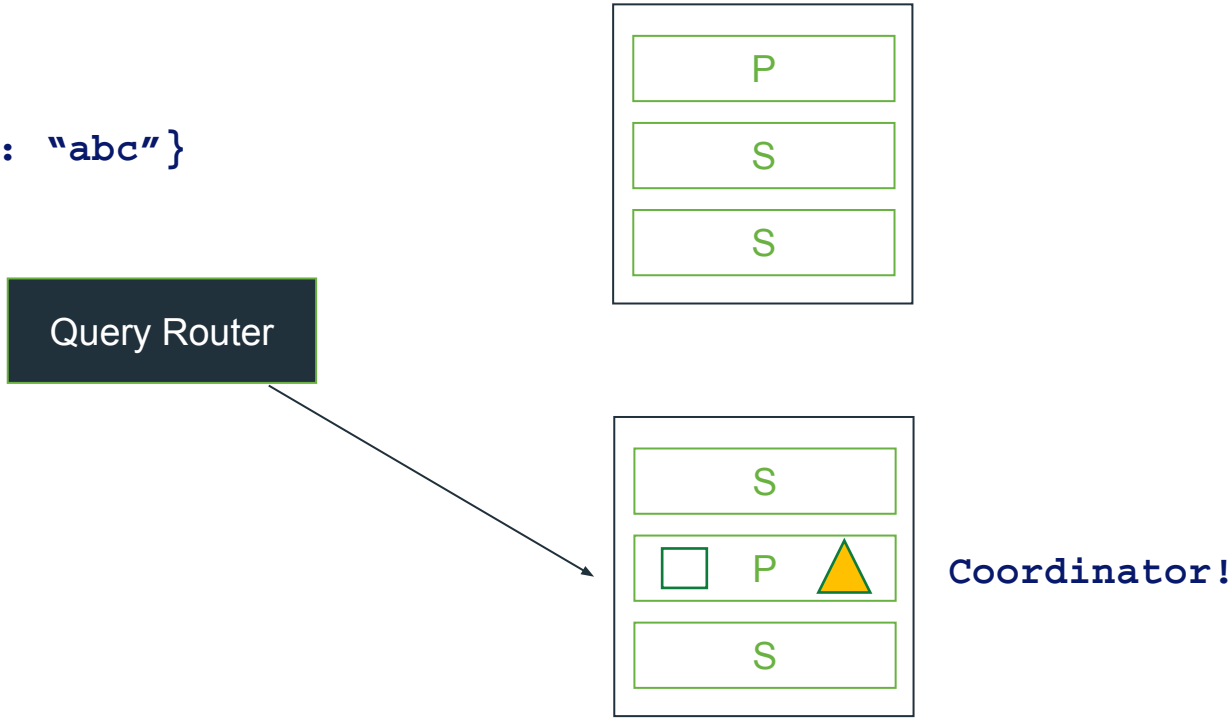
Participant

any shard that executes operations on behalf of a given transaction

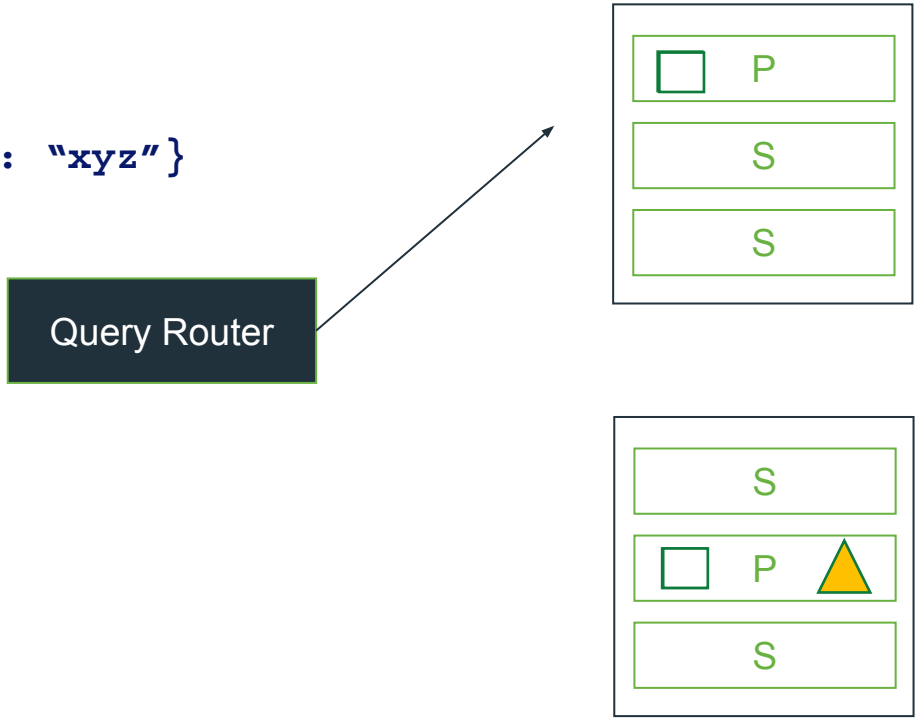
Coordinator

A single shard that is responsible for coordinating the commit across shards for a given transaction

```
Insert {_id : "abc"}
```

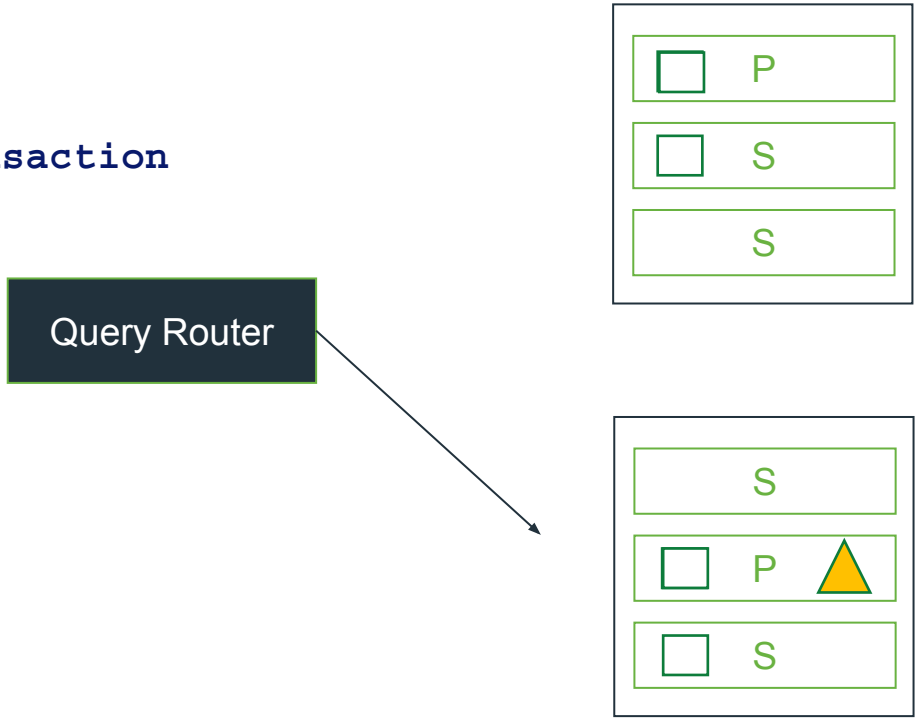


```
Insert {_id : "xyz"}
```



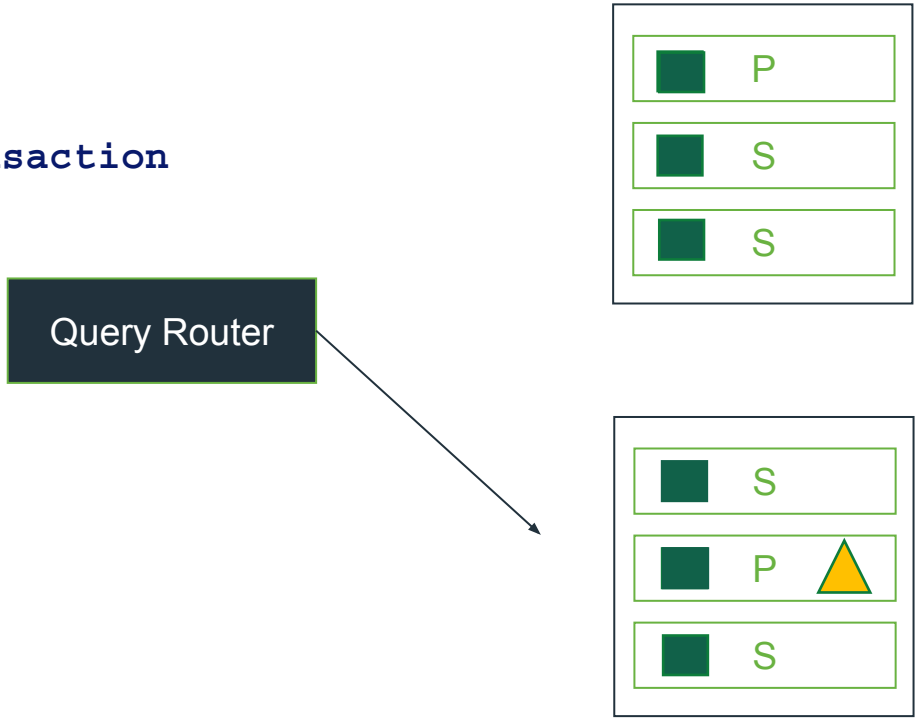
Coordinator!

Commit transaction



Coordinator!

Commit transaction

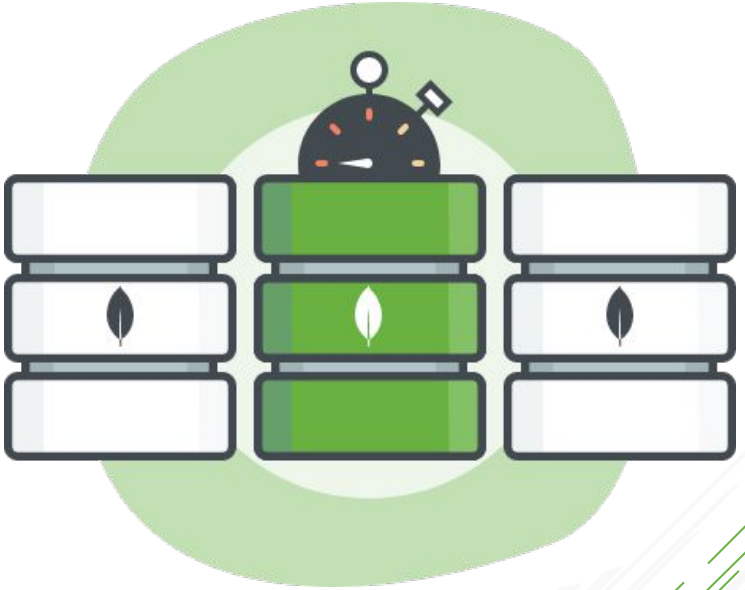


Coordinator!

Our Journey to Distributed ACID Transactions

Starting in 2015, major engineering investment touching every part of the server and drivers

- Storage layer
- Replication consensus protocol
- Sharding architecture
- Consistency and durability guarantees
- Global logical clock
- Cluster metadata management
- Exposed to drivers through API enhancements

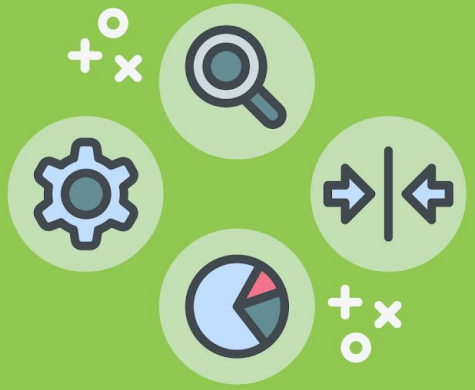


Beware cheap imitations…..

	Other distributed DBs	mongoDB
Applies transactions to base data & indexes	✗	✓
Enforces isolation and consistency server-side	✗ *Must be controlled by the client*	✓
Conversational via the drivers	✗ *Stored procedures*	✓
Interleave reads and writes	✗	✓
Guarantees enforced across partitions/shards	✗	✓
Supported by all mainstream drivers	✗	✓

New Online Course

M042: New Features and Tools in MongoDB 4.2



- Free ▪ Online ▪ 9 Chapters
- Register Now! - https://university.mongodb.com/courses/M042/about

mongoDB

Get Started

- ❑ Spin up on MongoDB Atlas
  - ❑ $200 credit! MONGODB4DOT2
- ❑ Review the transactions documentation
- ❑ Download the Guide to What's New
- ❑ Tune in to the Deep Dive on Oct 23

mongoDB

Q&A