#gotober

What do you mean, Backwards Compatibility?

Trisha Gee, Java Driver Developer

@trisha_gee

mongoDB

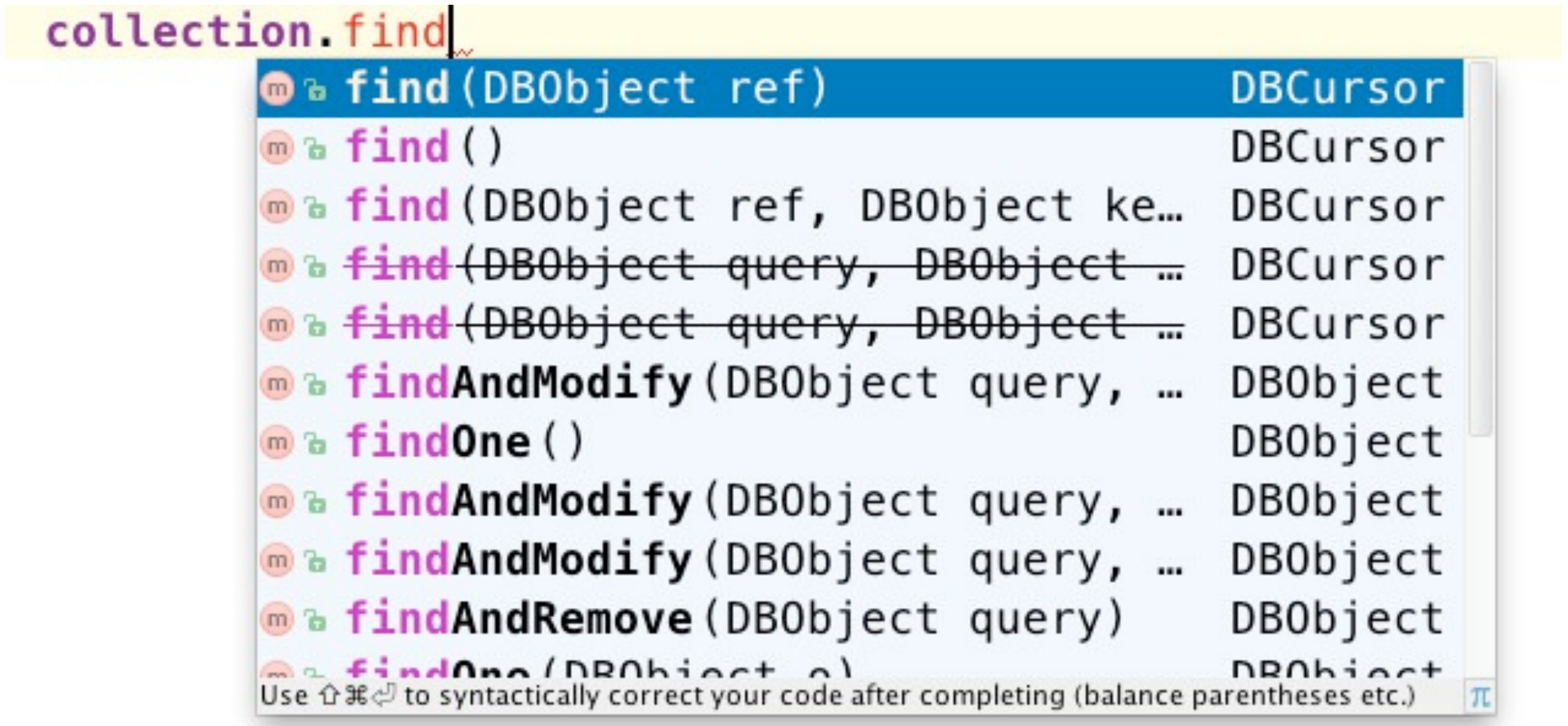
What’s the problem?

The Domain

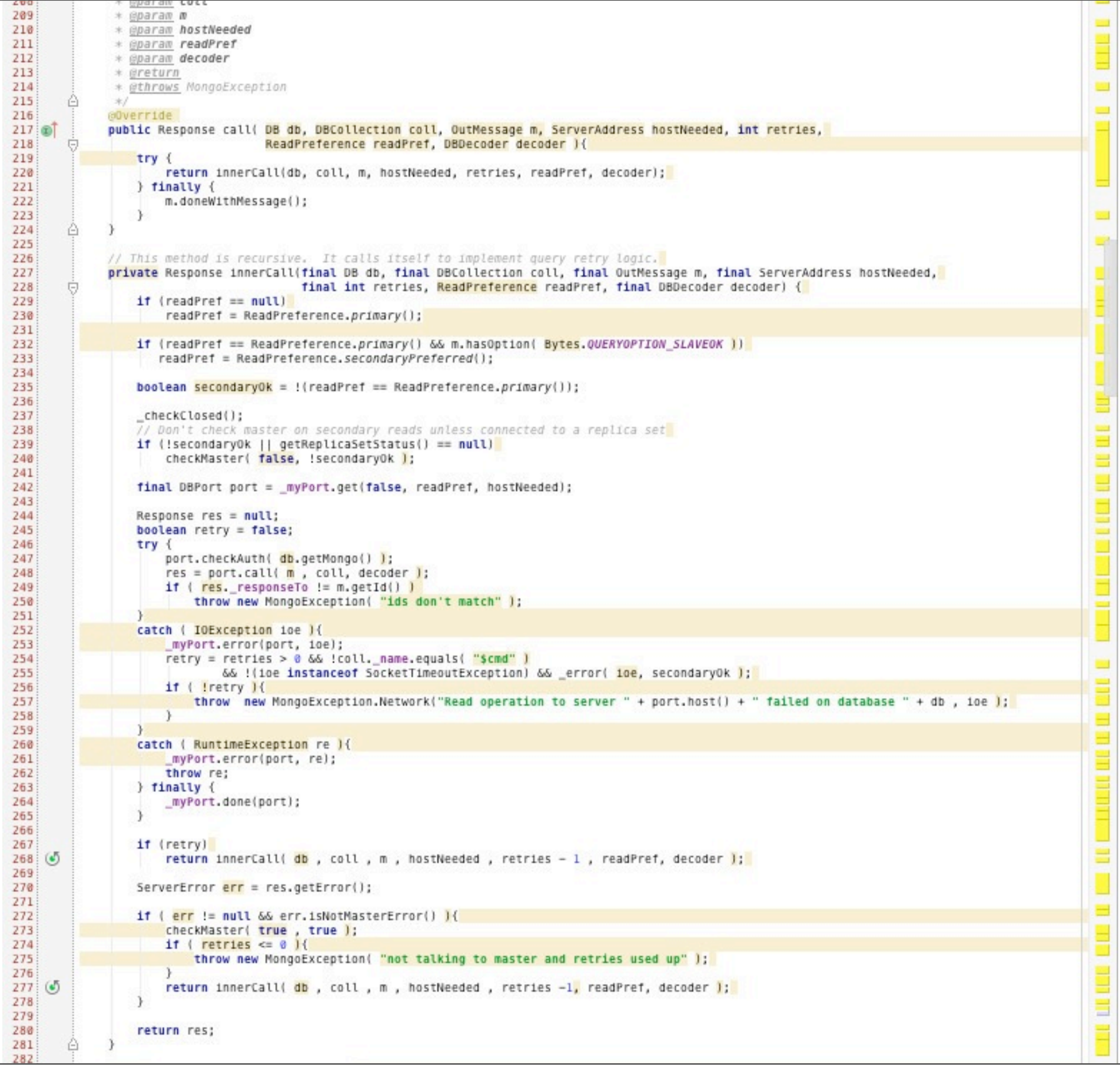
MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

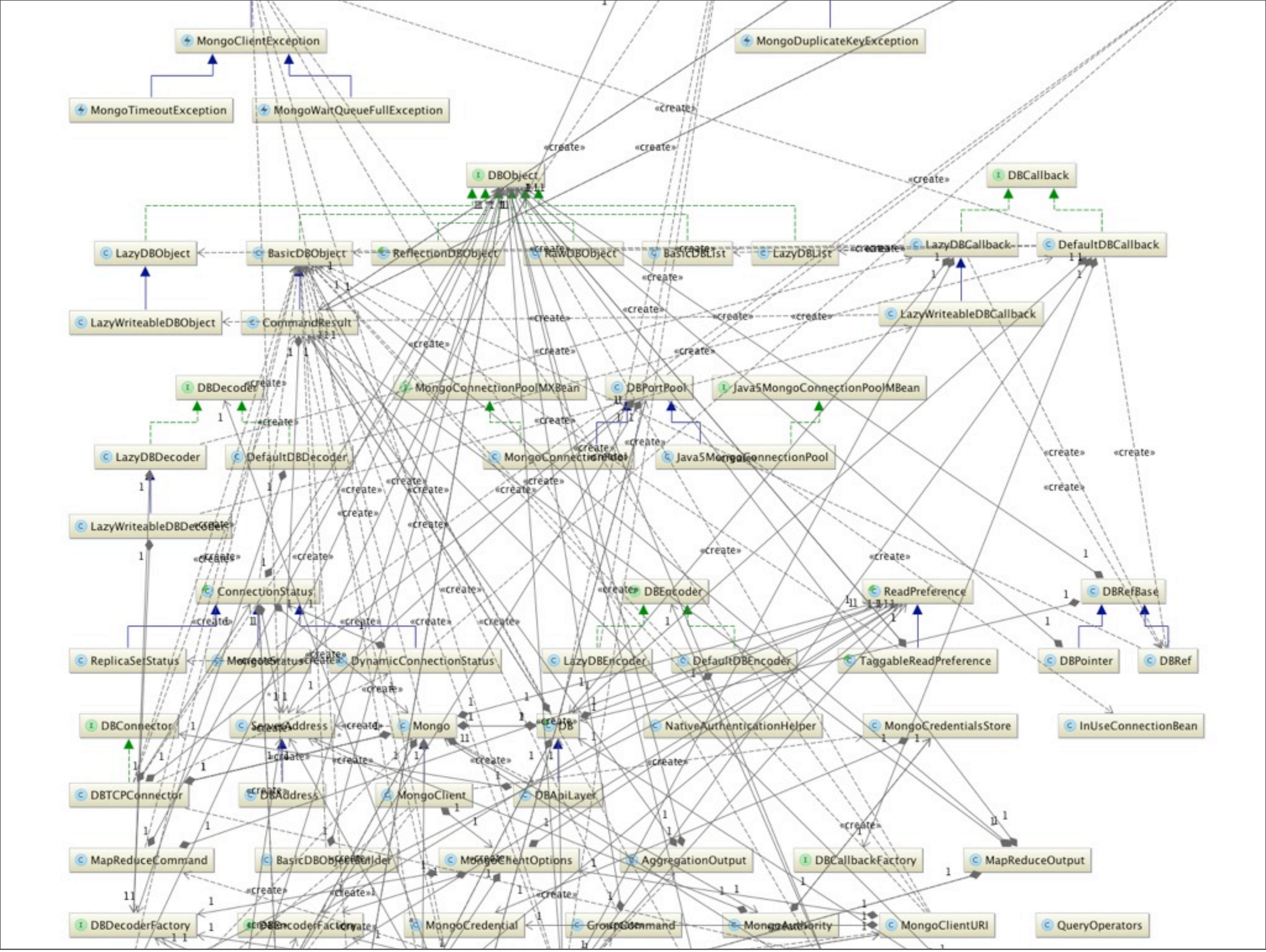
The Driver



```
collection.find
find(DBObject ref)                       DBCursor
find()                                   DBCursor
find(DBObject ref, DBObject ke…          DBCursor
~~find(DBObject query, DBObject …~~      DBCursor
~~find(DBObject query, DBObject …~~      DBCursor
findAndModify(DBObject query, …          DBObject
findOne()                                DBObject
findAndModify(DBObject query, …          DBObject
findAndModify(DBObject query, …          DBObject
findAndRemove(DBObject query)            DBObject
findOne(DBObject o)                      DBObject
Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.)   π
```



```java
     * @param coll
     * @param m
     * @param hostNeeded
     * @param readPref
     * @param decoder
     * @return
     * @throws MongoException
     */
    @Override
    public Response call( DB db, DBCollection coll, OutMessage m, ServerAddress hostNeeded, int retries,
                          ReadPreference readPref, DBDecoder decoder ){
        try {
            return innerCall(db, coll, m, hostNeeded, retries, readPref, decoder);
        } finally {
            m.doneWithMessage();
        }
    }

    // This method is recursive.  It calls itself to implement query retry logic.
    private Response innerCall(final DB db, final DBCollection coll, final OutMessage m, final ServerAddress hostNeeded,
                               final int retries, ReadPreference readPref, final DBDecoder decoder) {
        if (readPref == null)
            readPref = ReadPreference.primary();

        if (readPref == ReadPreference.primary() && m.hasOption( Bytes.QUERYOPTION_SLAVEOK ))
           readPref = ReadPreference.secondaryPreferred();

        boolean secondaryOk = !(readPref == ReadPreference.primary());

        _checkClosed();
        // Don't check master on secondary reads unless connected to a replica set
        if (!secondaryOk || getReplicaSetStatus() == null)
            checkMaster( false, !secondaryOk );

        final DBPort port = _myPort.get(false, readPref, hostNeeded);

        Response res = null;
        boolean retry = false;
        try {
            port.checkAuth( db.getMongo() );
            res = port.call( m , coll, decoder );
            if ( res._responseTo != m.getId() )
                throw new MongoException( "ids don't match" );
        }
        catch ( IOException ioe ){
            _myPort.error(port, ioe);
            retry = retries > 0 && !coll._name.equals( "$cmd" )
                    && !(ioe instanceof SocketTimeoutException) && _error( ioe, secondaryOk );
            if ( !retry ){
                throw  new MongoException.Network("Read operation to server " + port.host() + " failed on database " + db , ioe );
            }
        }
        catch ( RuntimeException re ){
            _myPort.error(port, re);
            throw re;
        } finally {
            _myPort.done(port);
        }

        if (retry)
            return innerCall( db , coll , m , hostNeeded , retries - 1 , readPref, decoder );

        ServerError err = res.getError();

        if ( err != null && err.isNotMasterError() ){
            checkMaster( true , true );
            if ( retries <= 0 ){
                throw new MongoException( "not talking to master and retries used up" );
            }
            return innerCall( db , coll , m , hostNeeded , retries -1, readPref, decoder );
        }

        return res;
    }
```



Friday, 18 October 13

Why is this my problem?

What do we want to do?

Design Goals

- Intuitive API
- Consistency
- Understandable exceptions
- Cleaner design
- Test friendly
- Backwards compatible
- http://is.gd/java3mongodb

Happy Users

Three Types Of Users

Three Types Of Users

1. Java Developers

Three Types Of Users

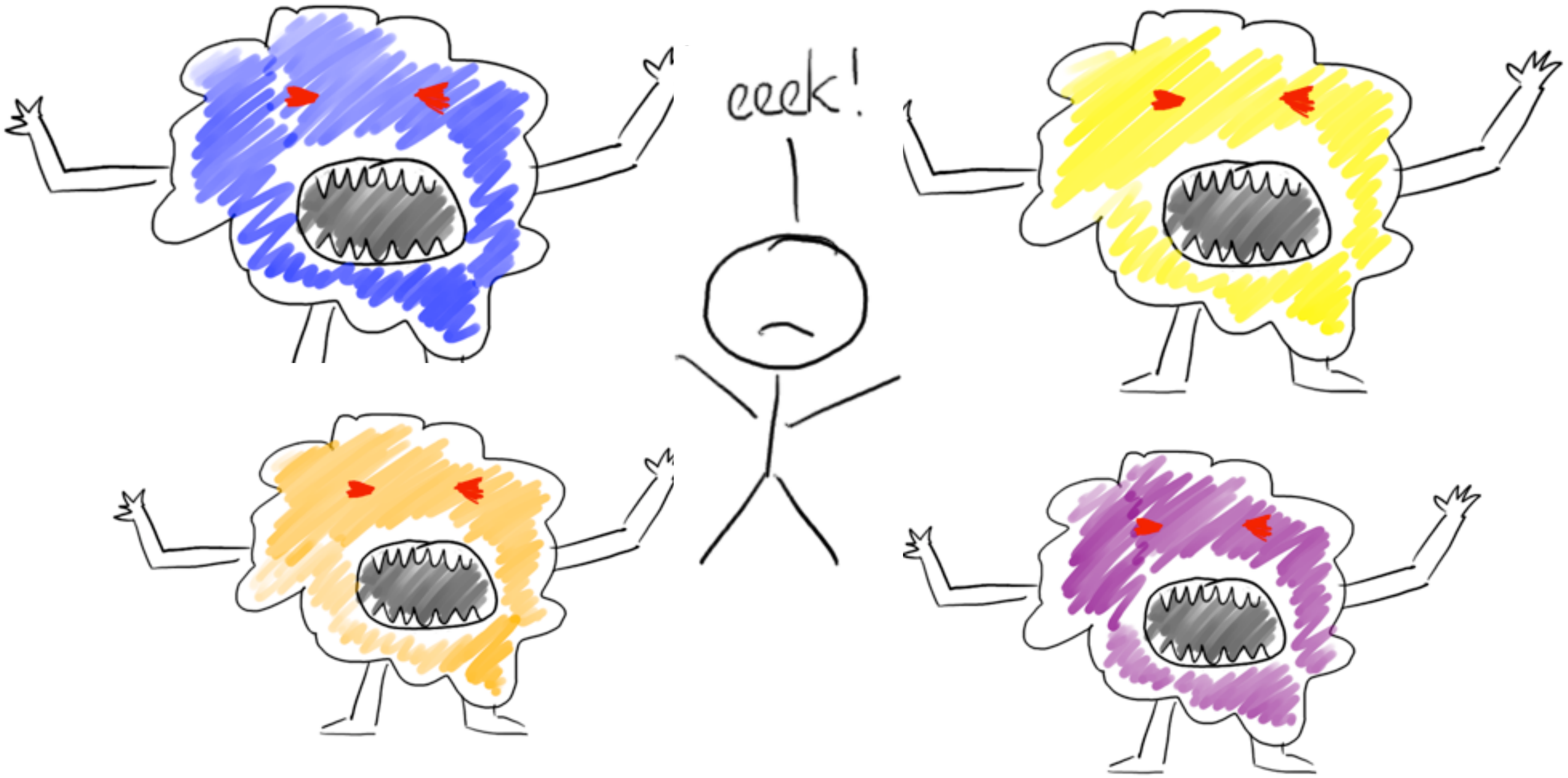
1. Java Developers
2. ODMs / other drivers / third parties

Three Types Of Users

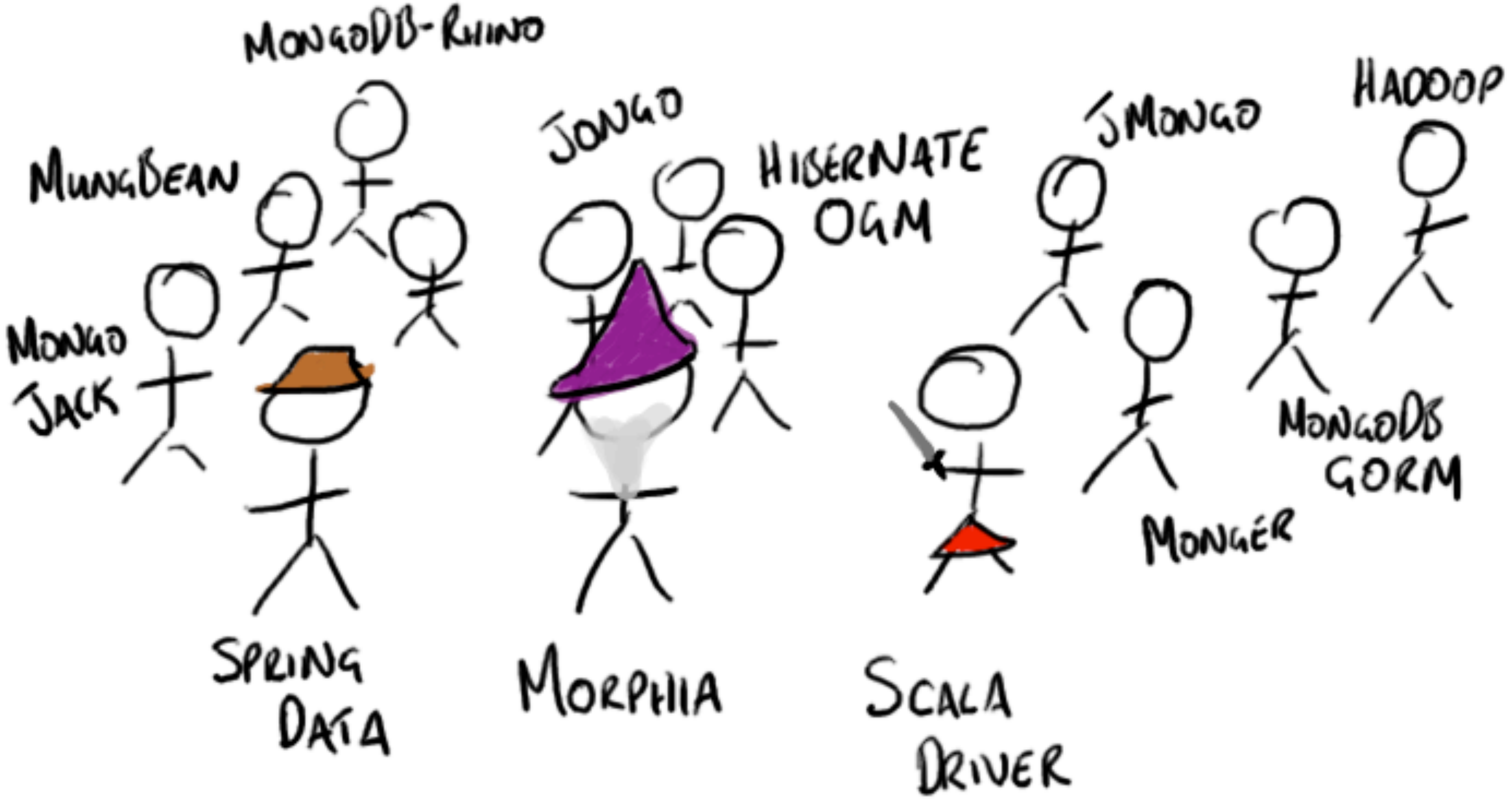
1. Java Developers
2. ODMs / other drivers / third parties
3. Contributors

What’s stopping us?

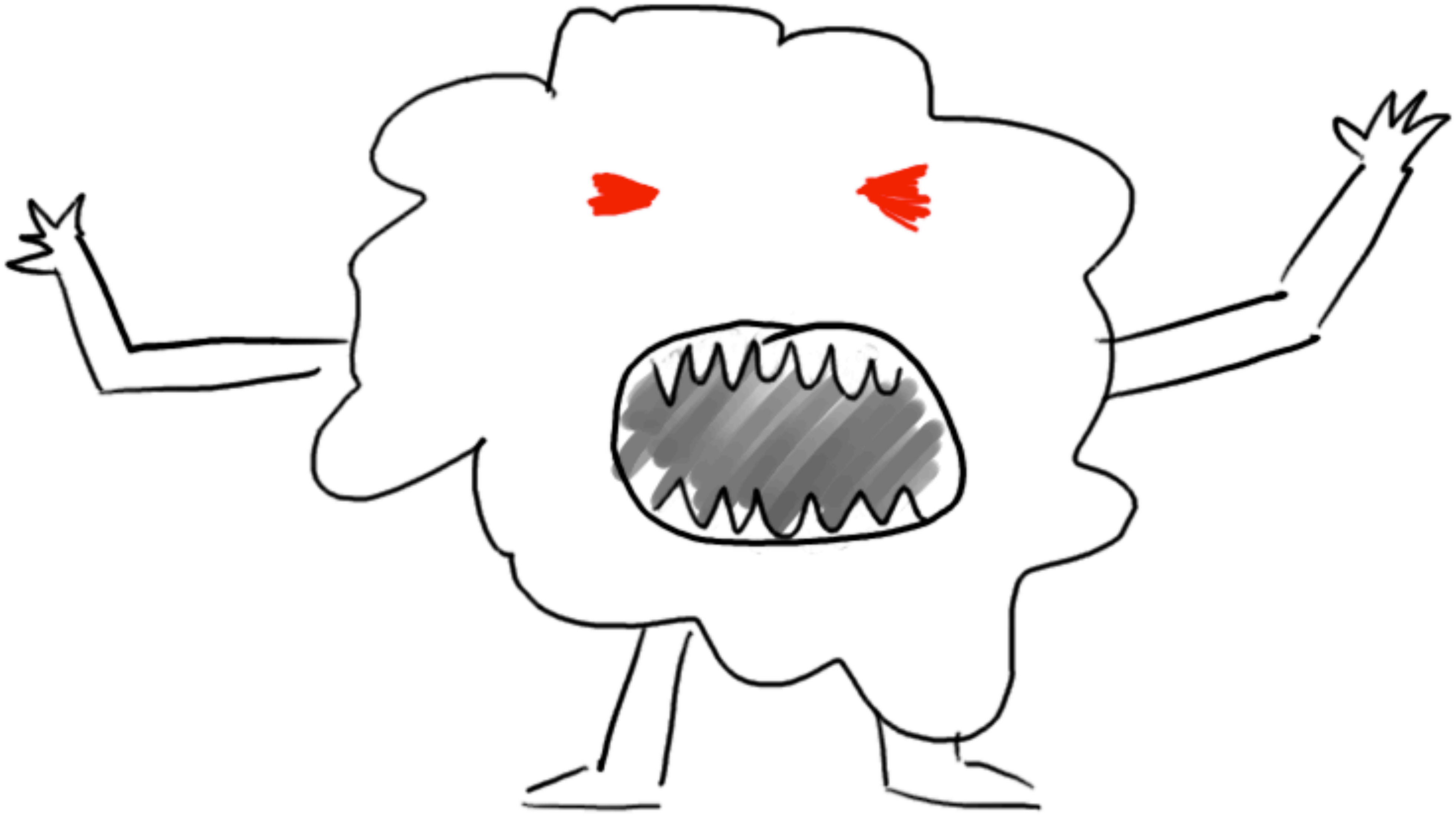
mongoDB



Lots of unknowns


MongoDB-Rhino
MungBean
Jongo
Hibernate OGM
JMongo
Hadoop
Mongo Jack
MongoDB GORM
Monger
Spring Data
Morphia
Scala Driver

Lots of APIs

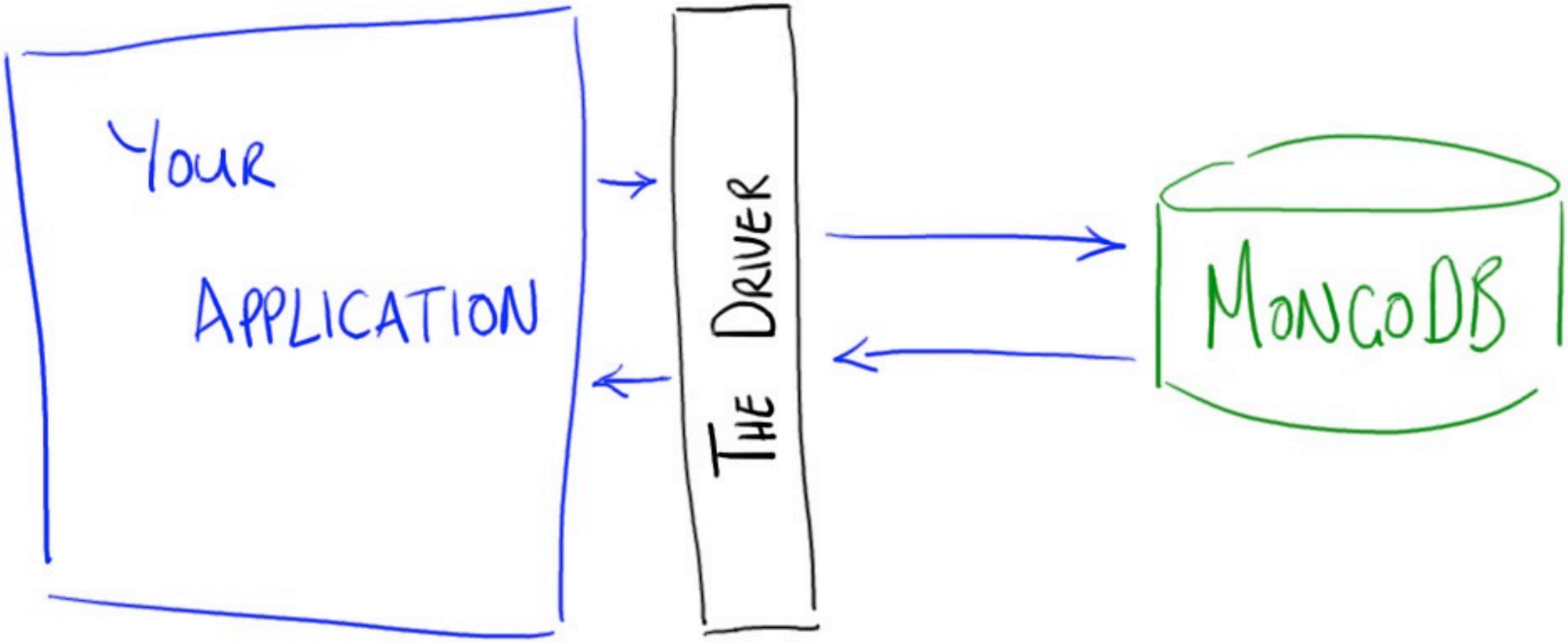


Backward Compatibility

What did we do?

mongoDB

Architecture

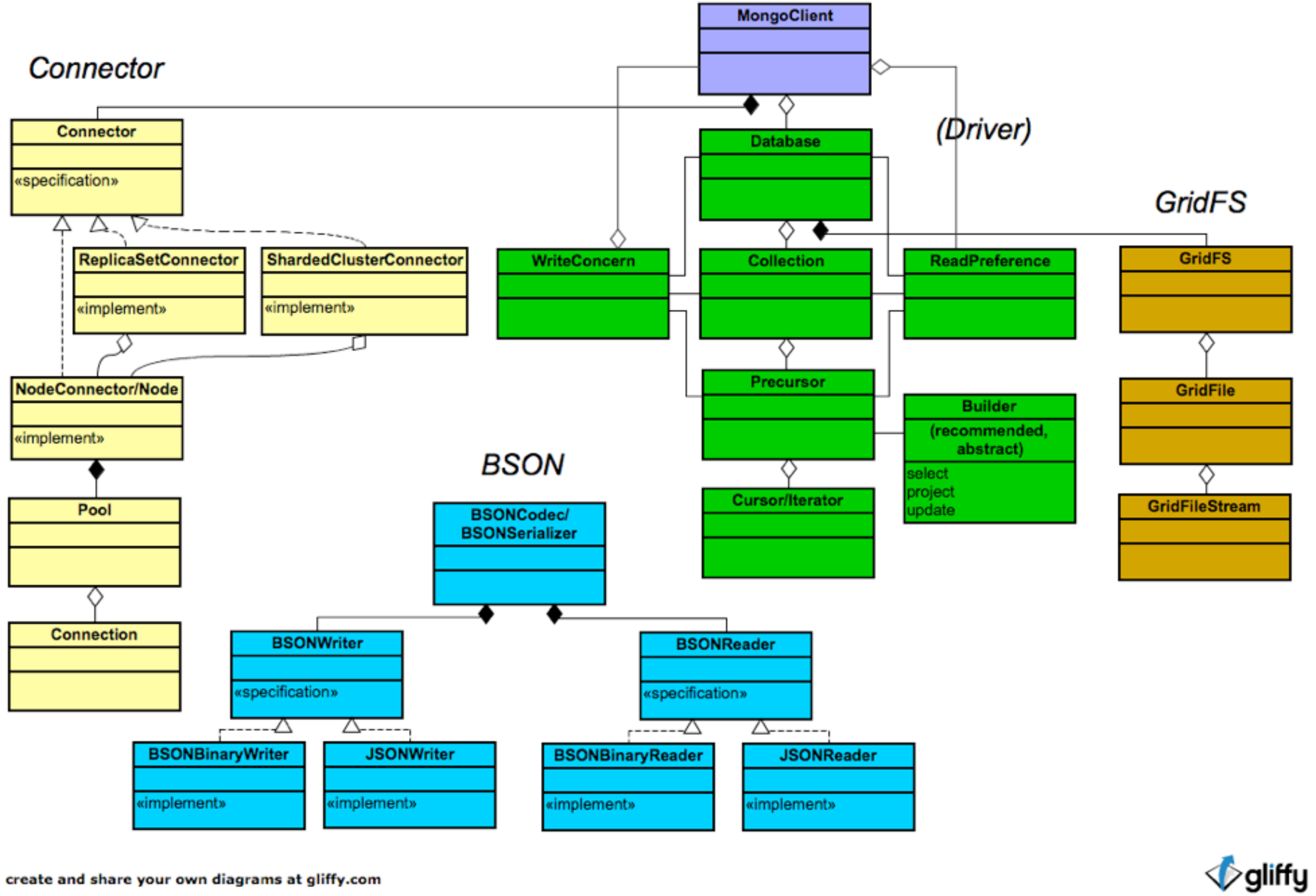


How Hard Can It Be?

Domain

MongoDB is an open-source document database, featuring:

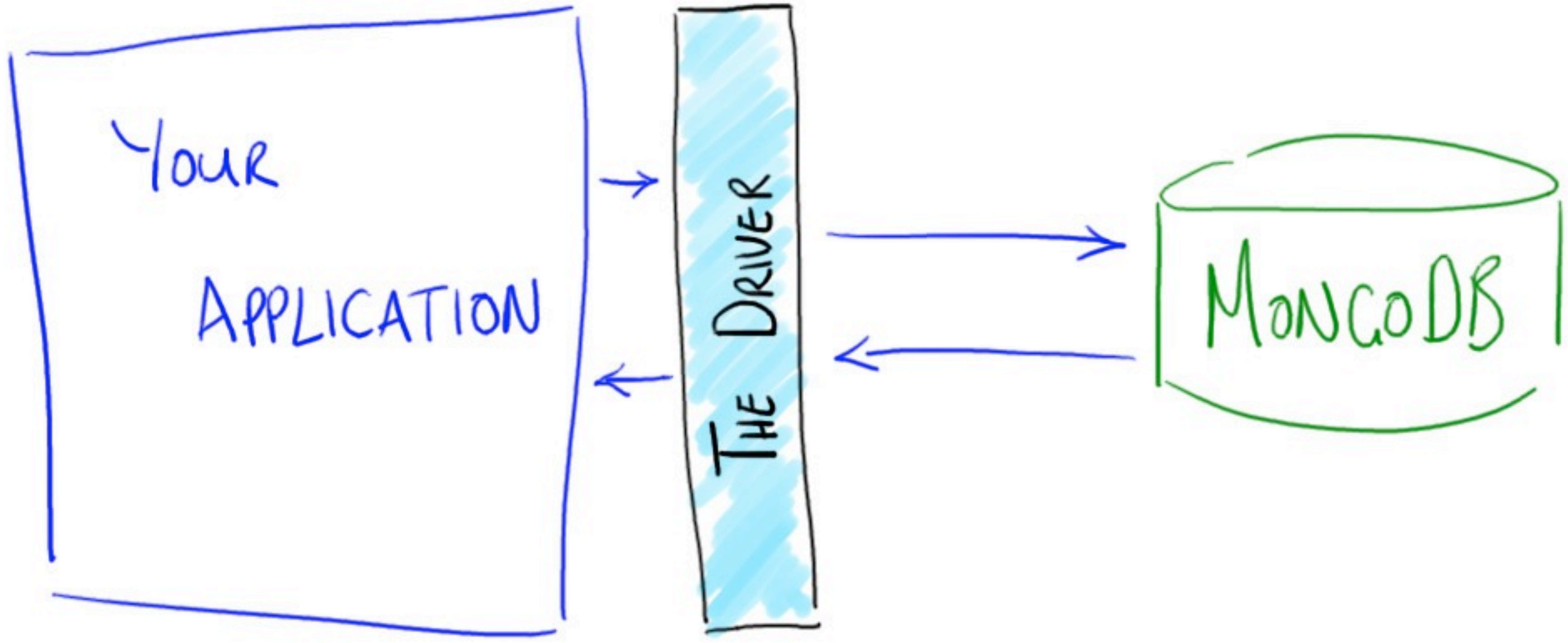
- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS



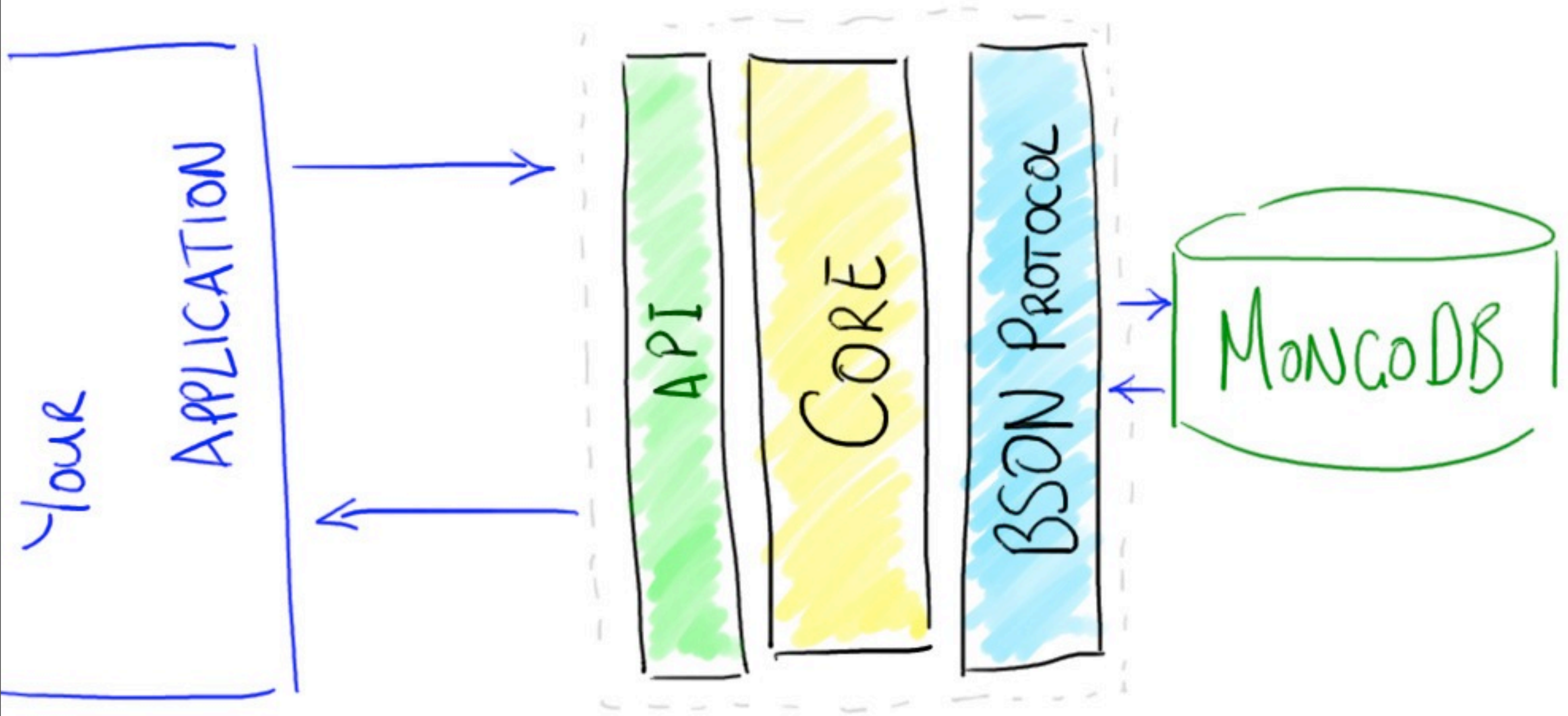
Bit more complex...

MongoDB is an open-source document database, featuring:

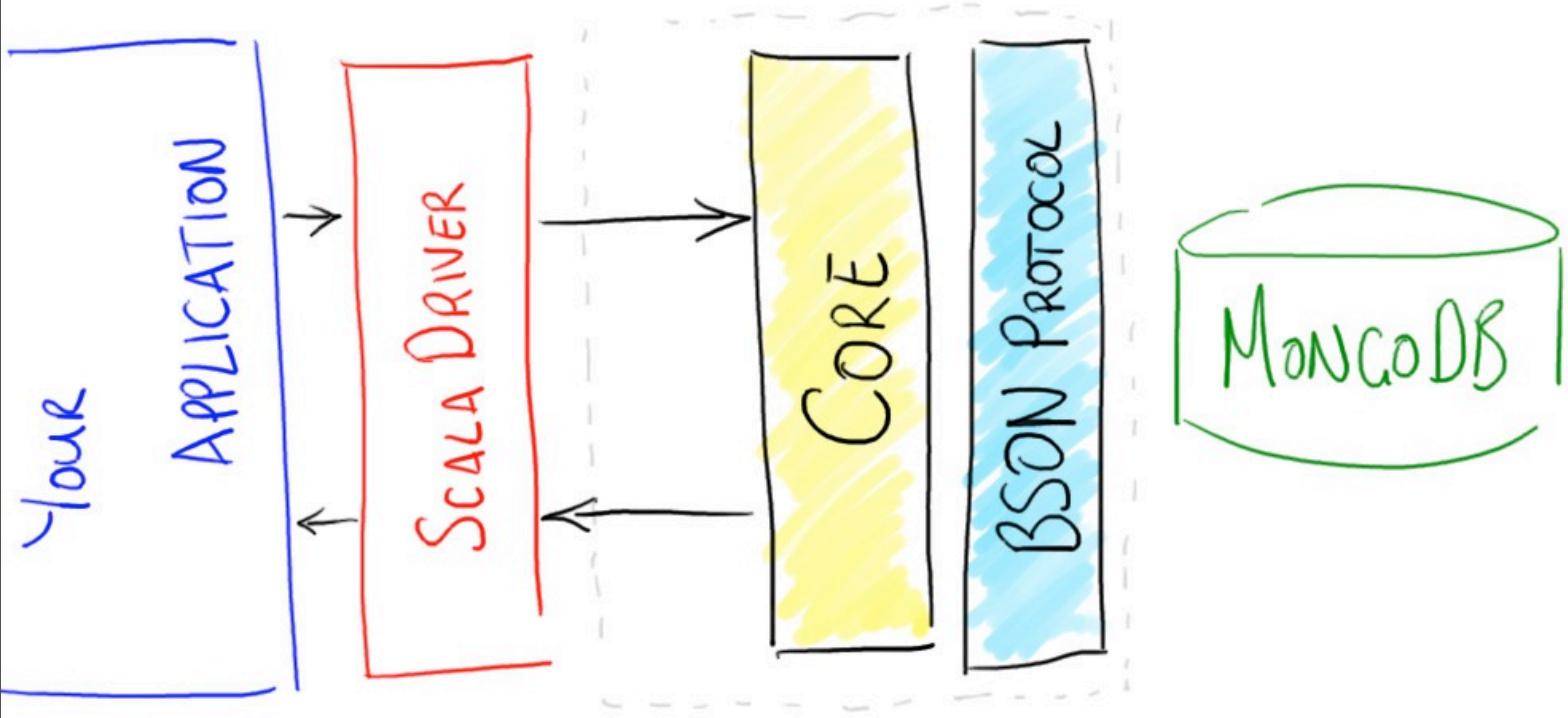
- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS



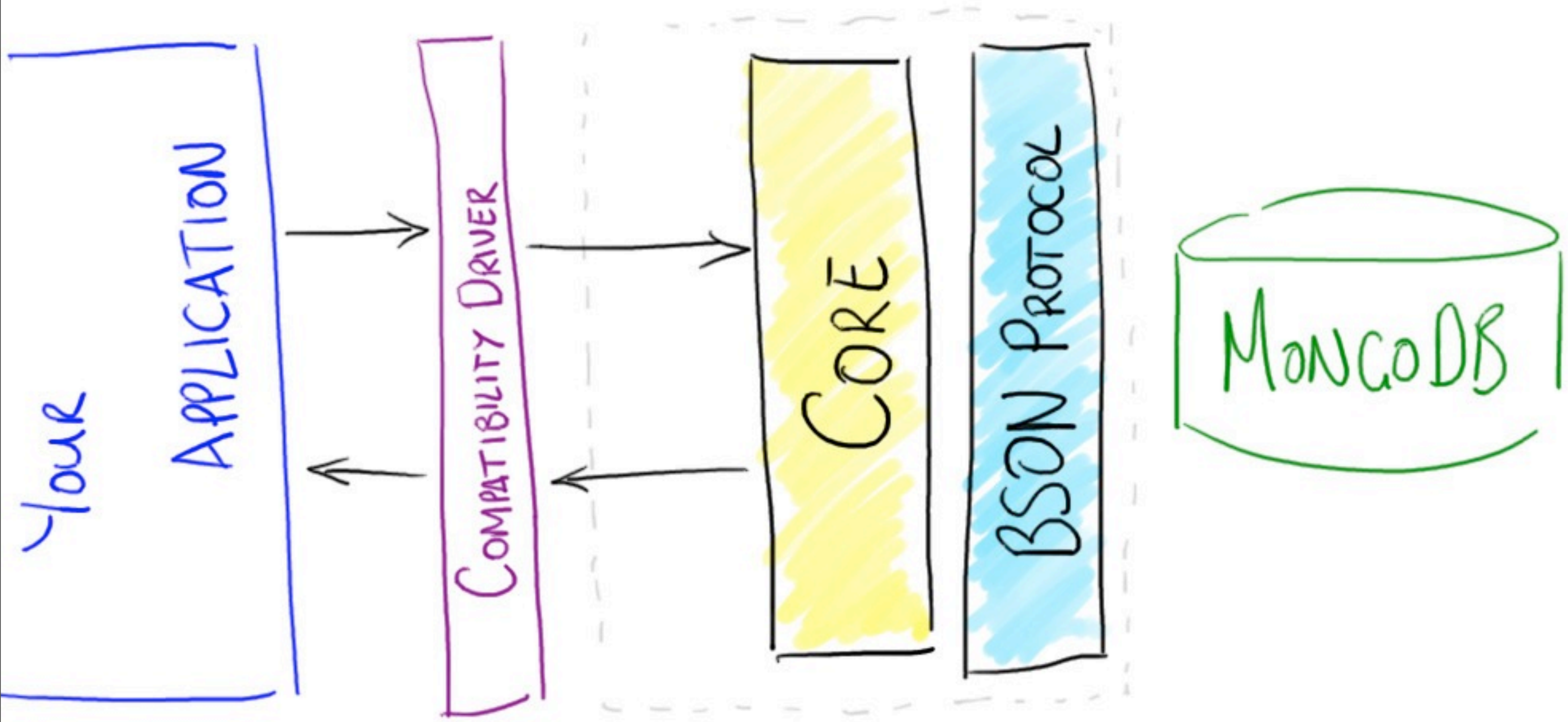
Instead of this...



Model the Domain



Scala Driver



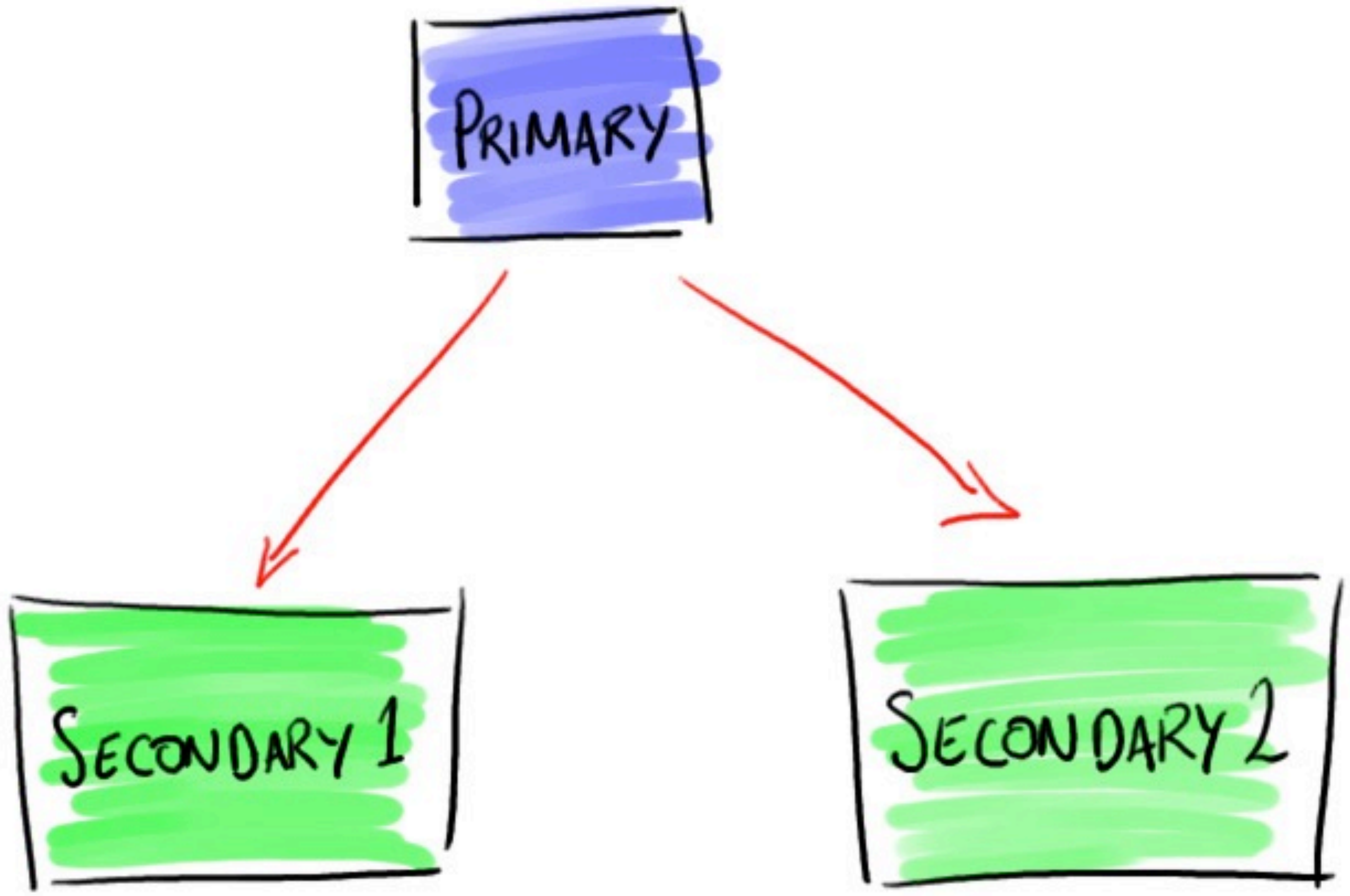
Backwards Compatible!

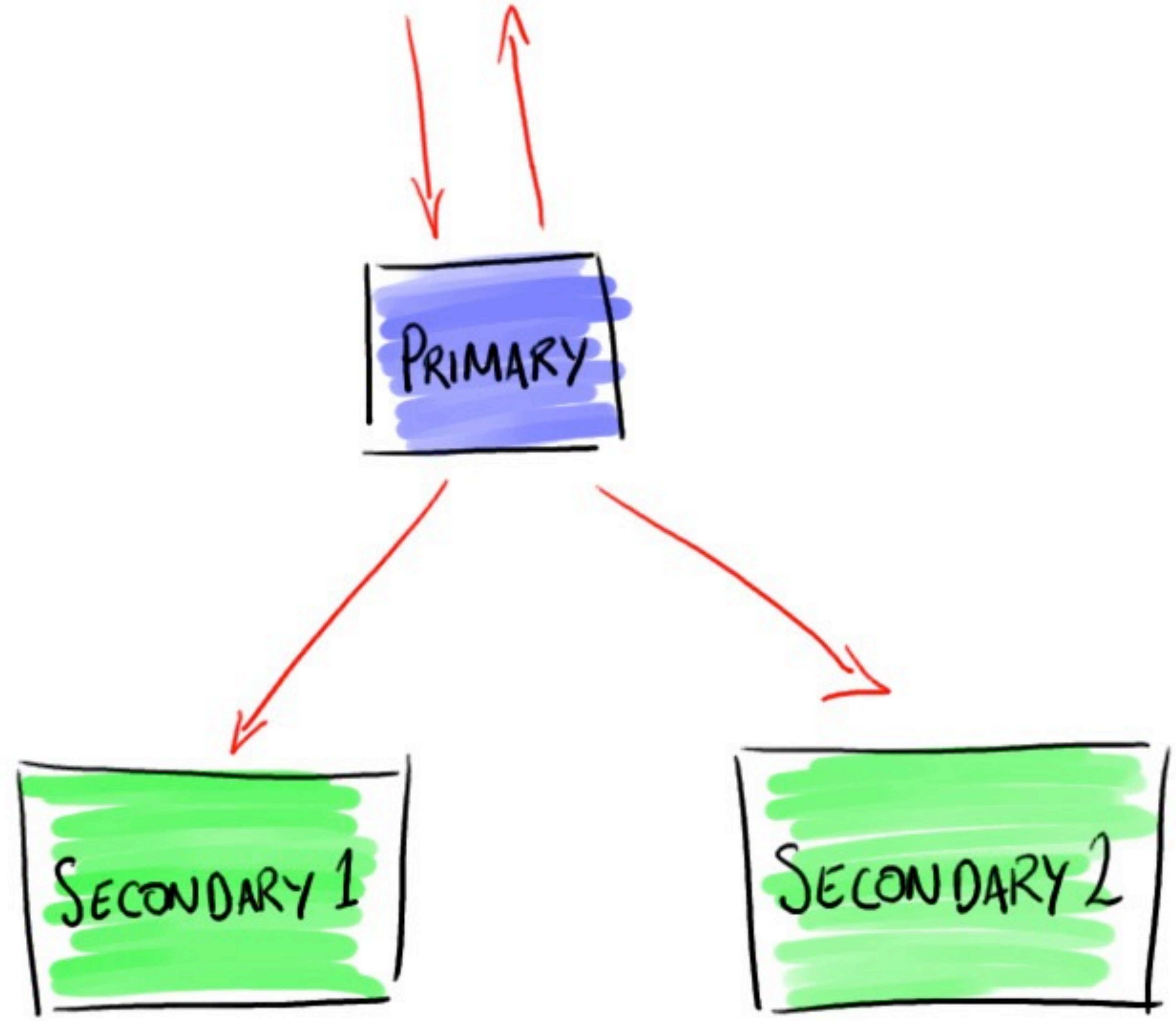
MongoDB is an open-source document database, featuring:

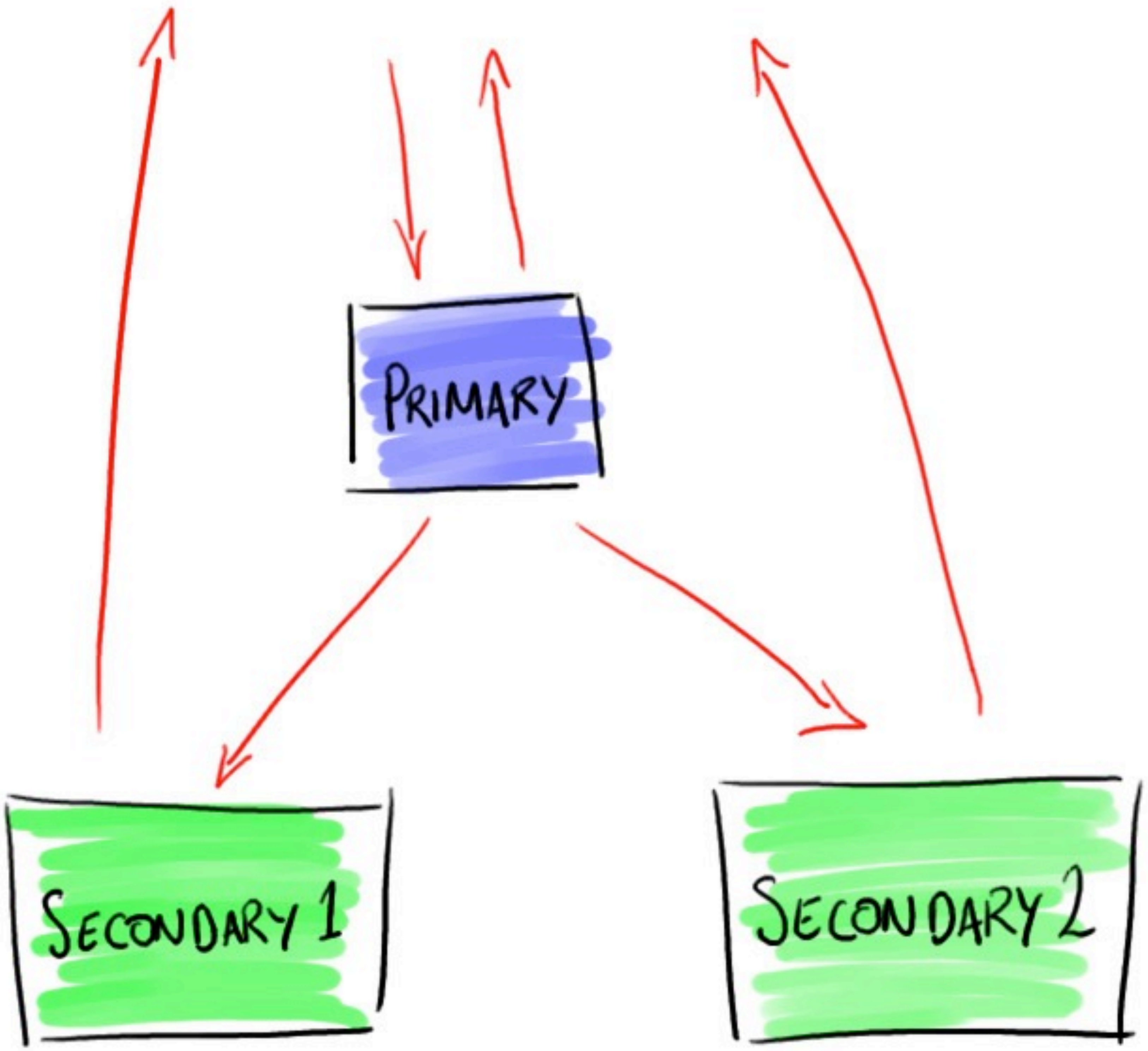
- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

Multi Server

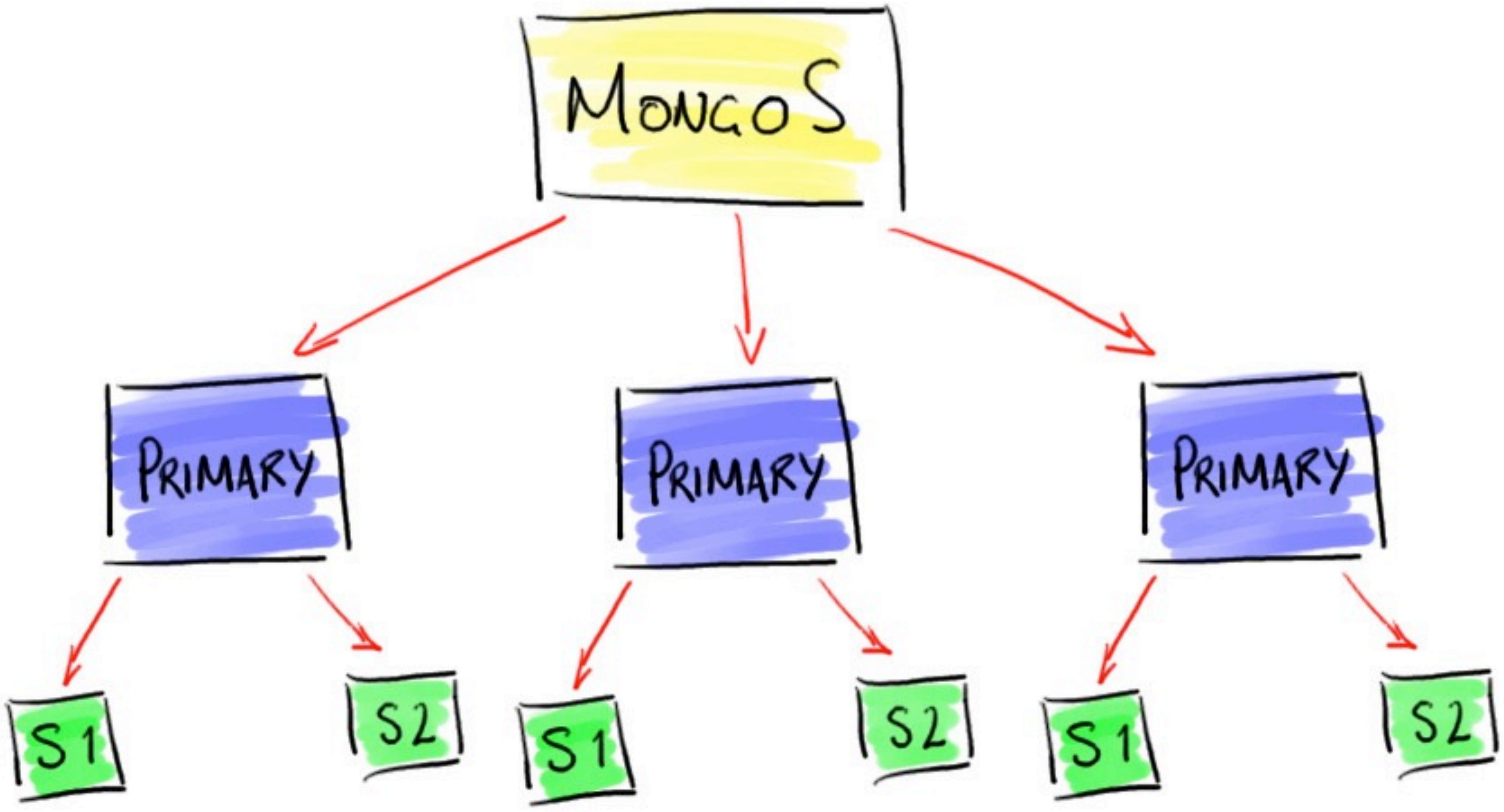
Replica Set

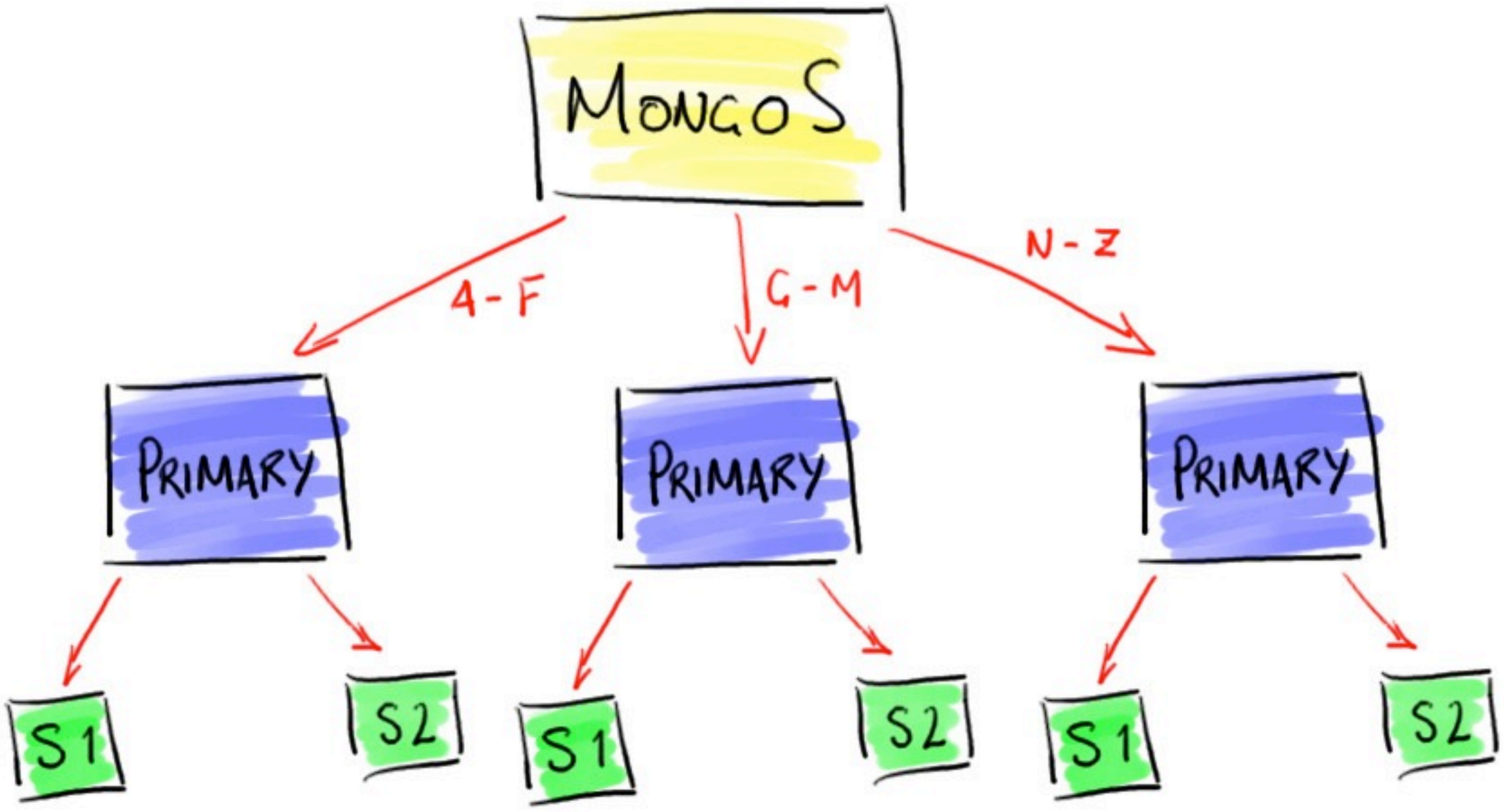



PRIMARY
SECONDARY 1
SECONDARY 2

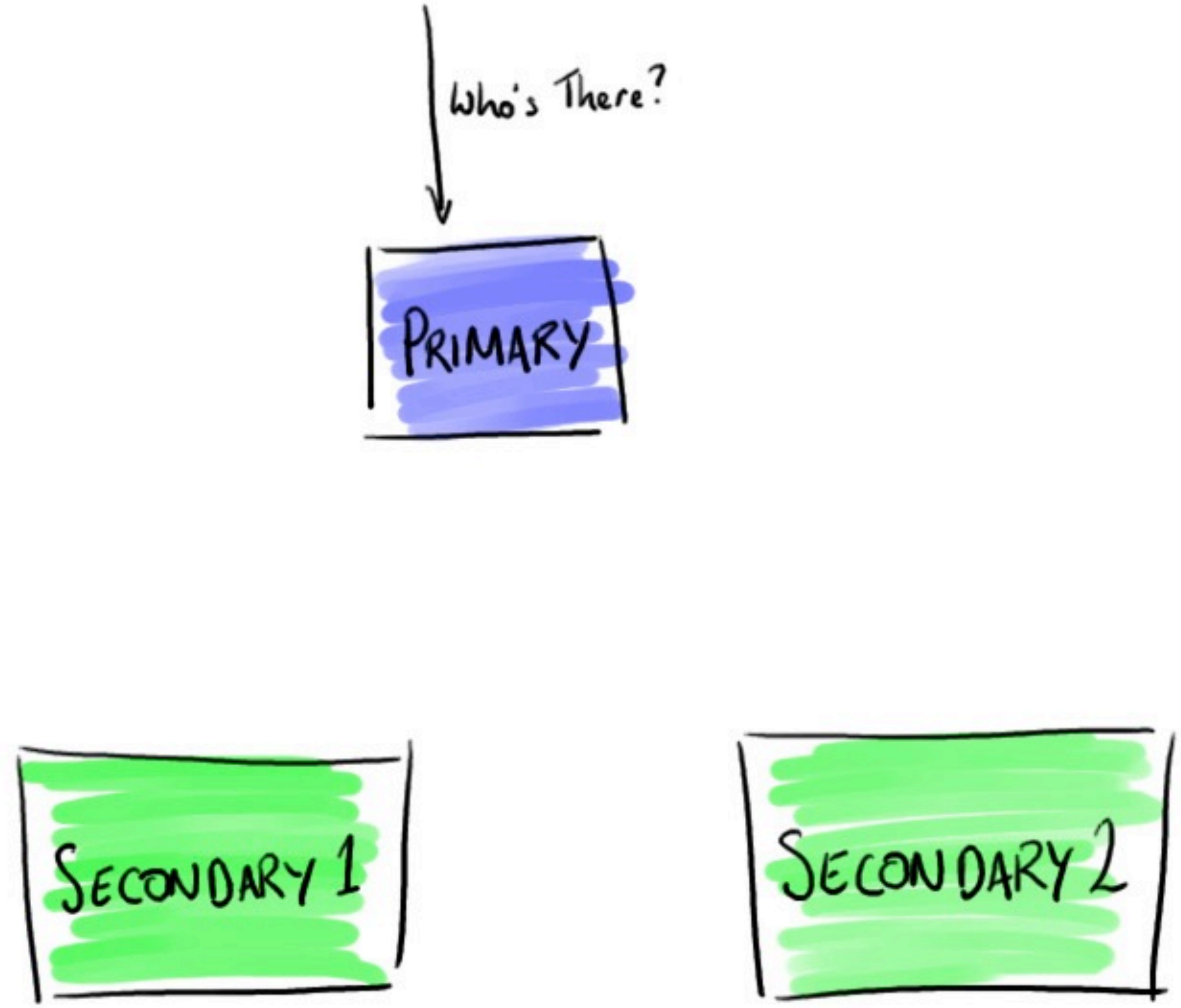

PRIMARY
SECONDARY 1
SECONDARY 2

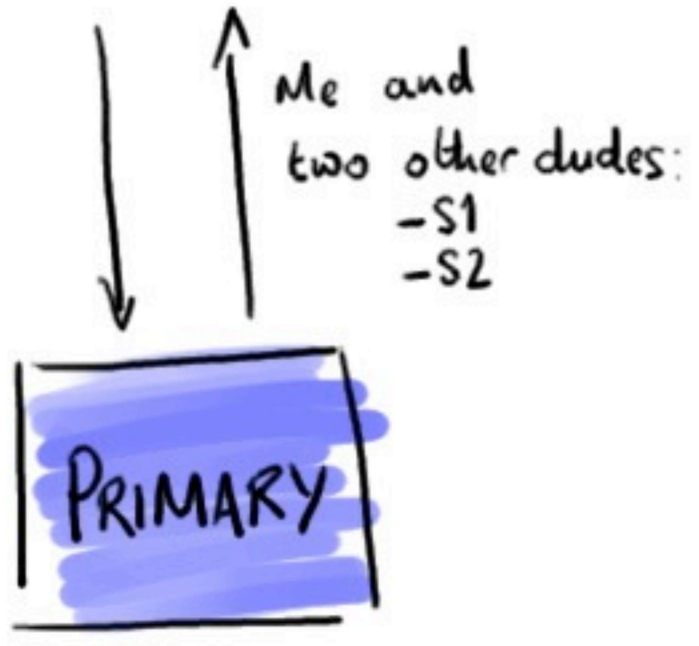
Sharding

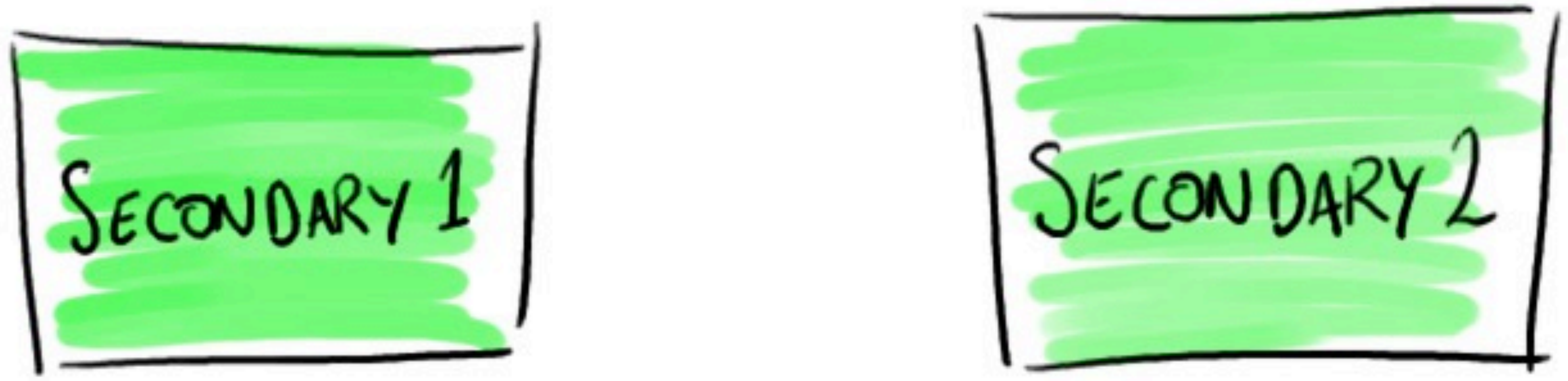


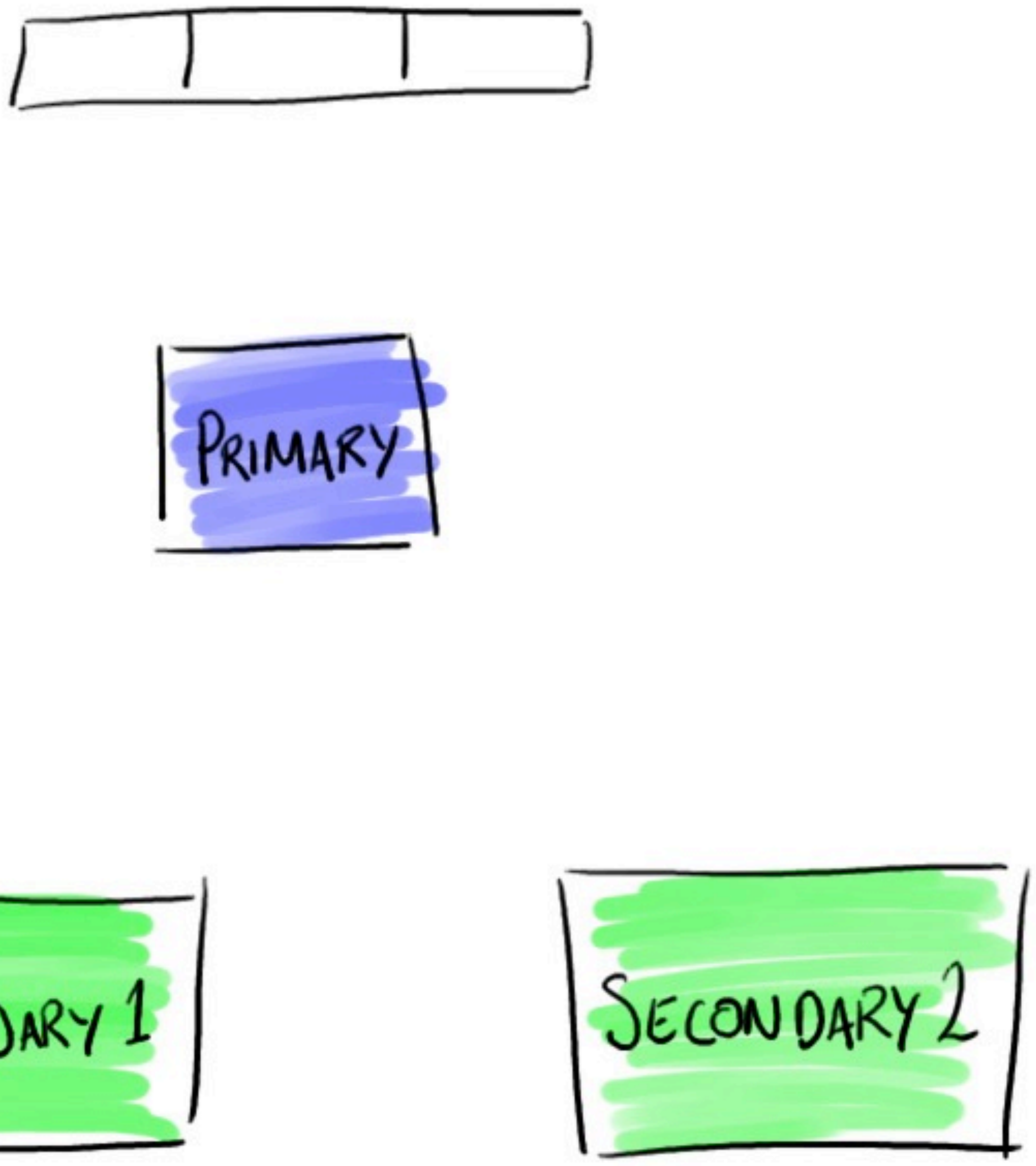

MongoS
A - F
G - M
N - Z
PRIMARY
PRIMARY
PRIMARY
S1
S2
S1
S2
S1
S2

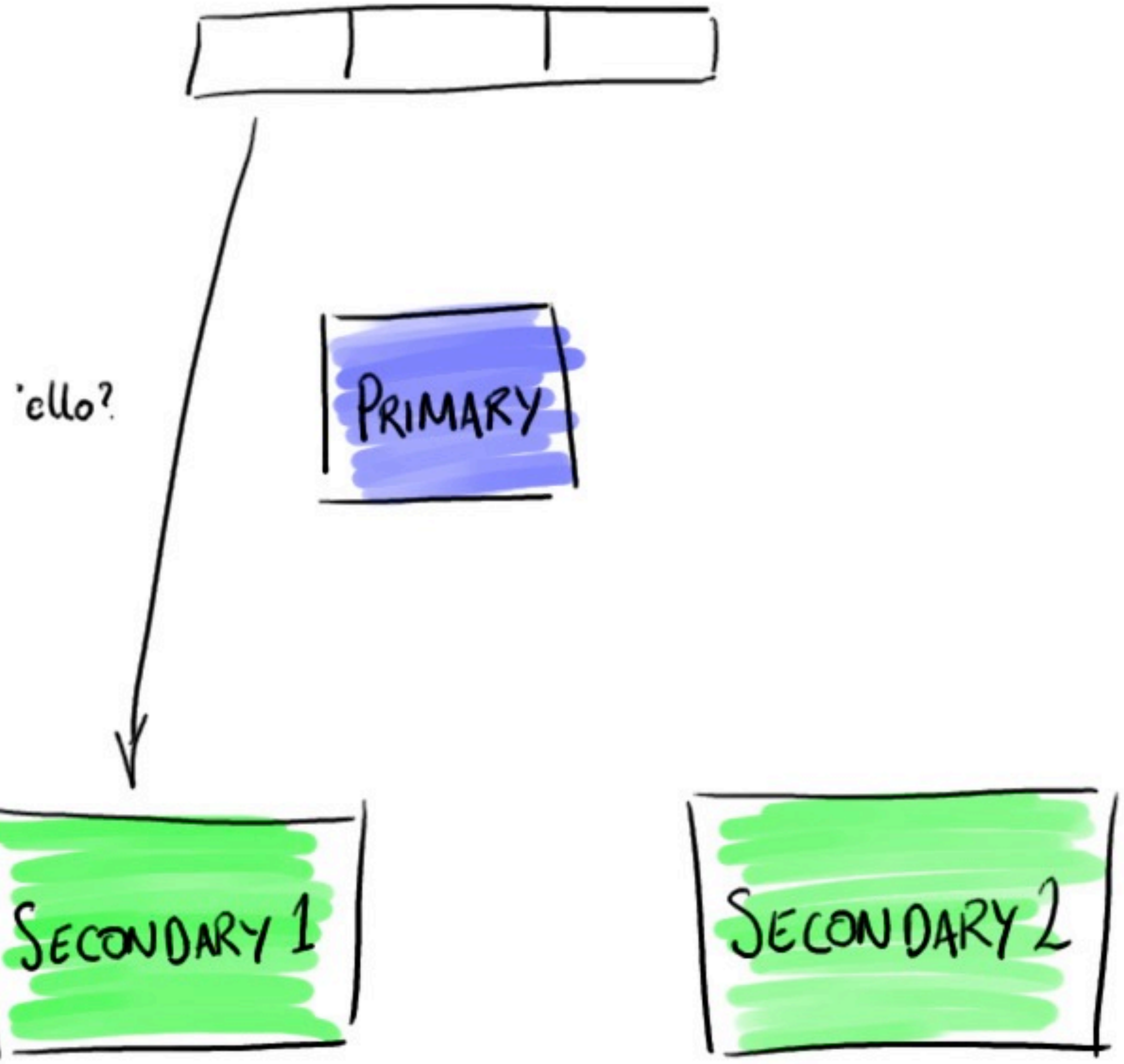
Finding Servers the Old Way

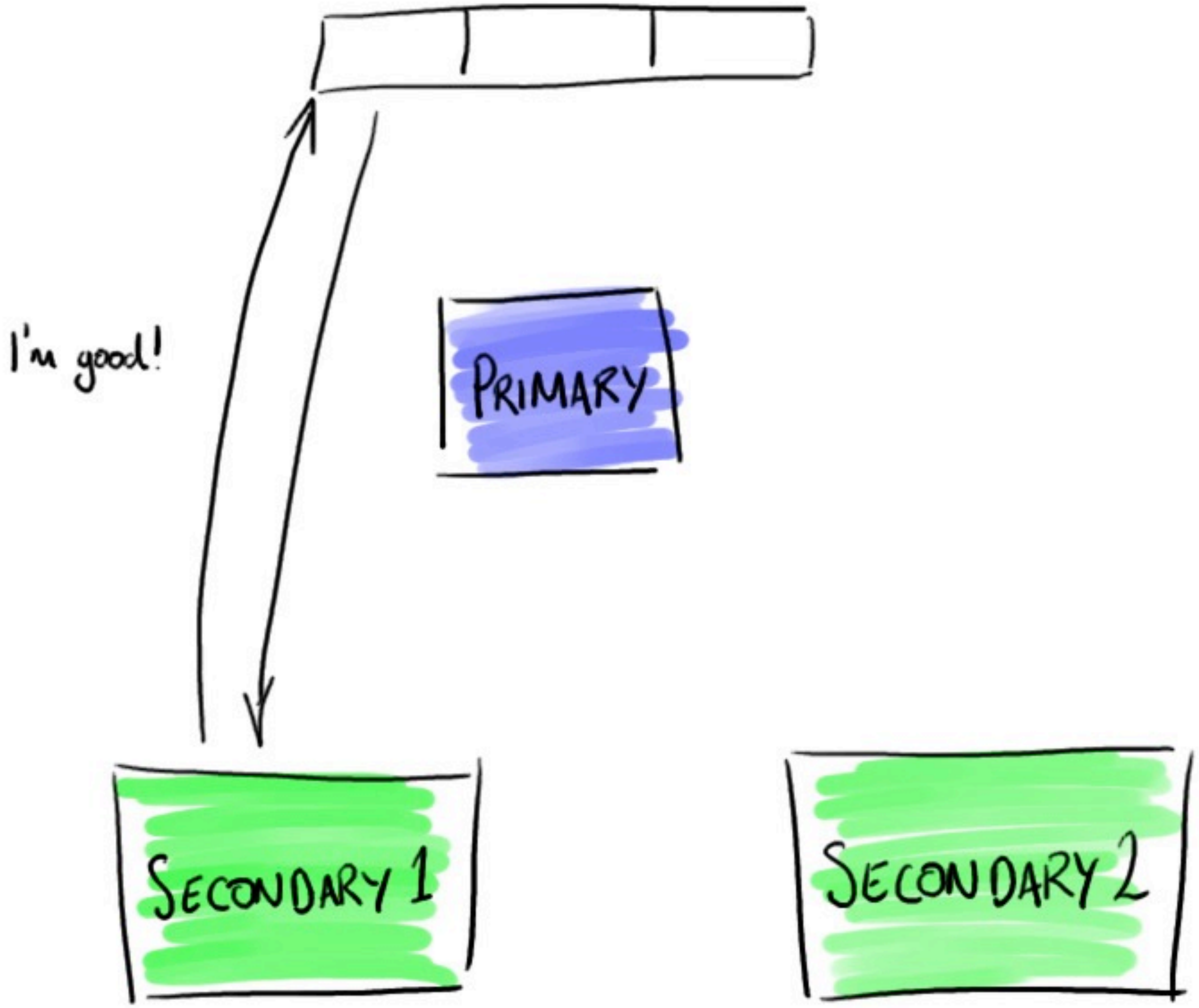

Who's There?
PRIMARY
SECONDARY 1
SECONDARY 2

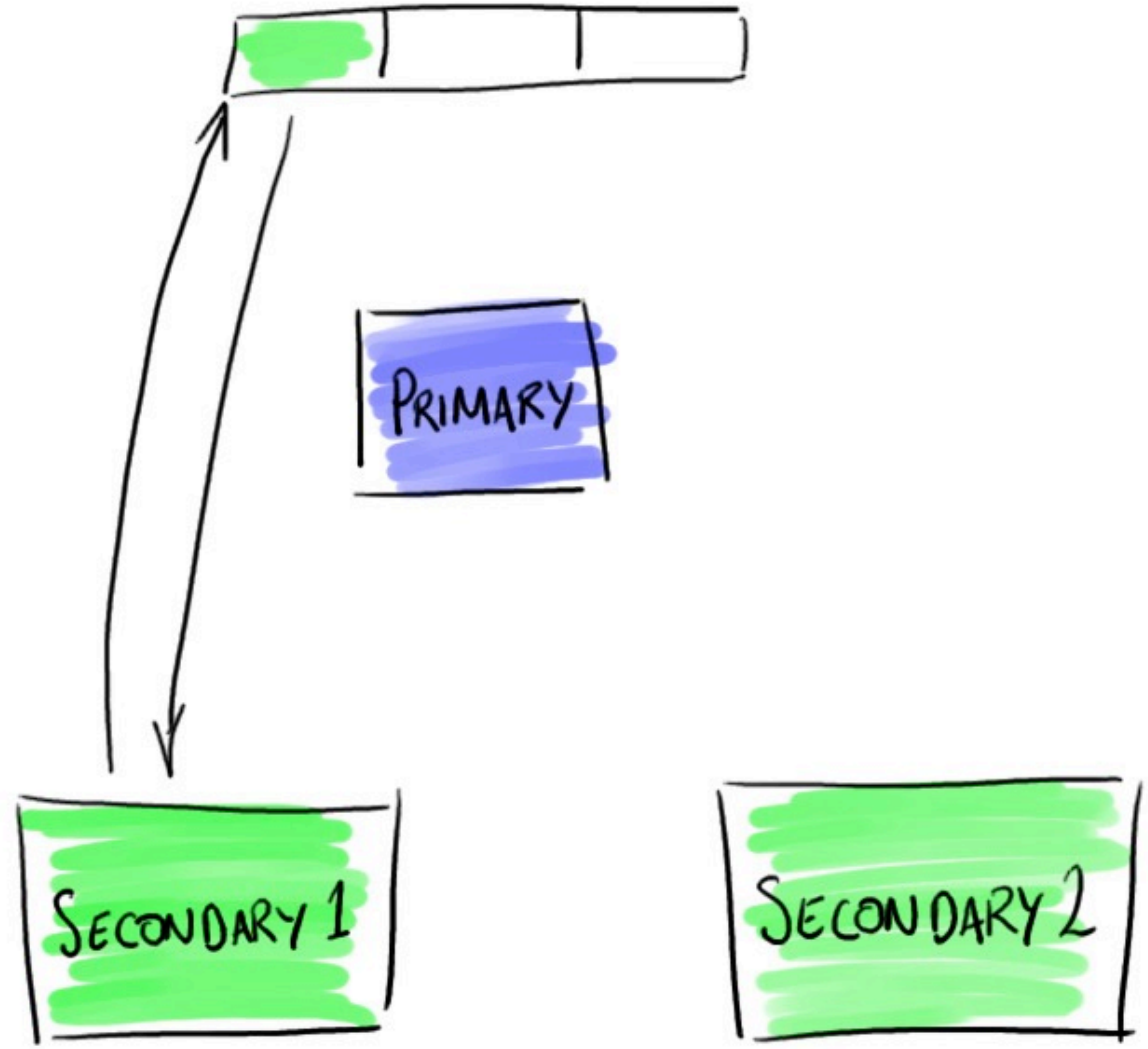

Me and
two other dudes:
-S1
-S2
PRIMARY

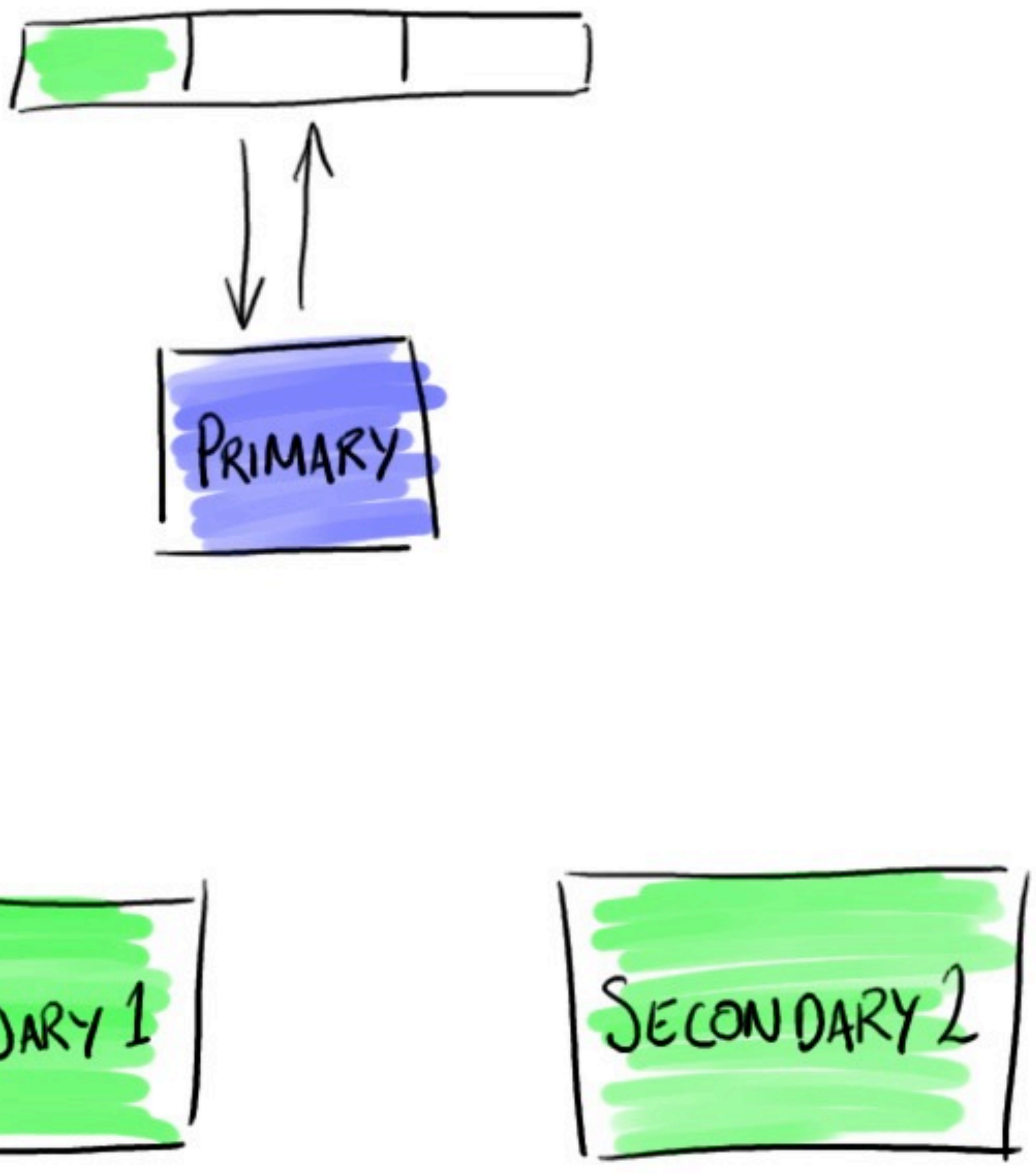

SECONDARY 1
SECONDARY 2

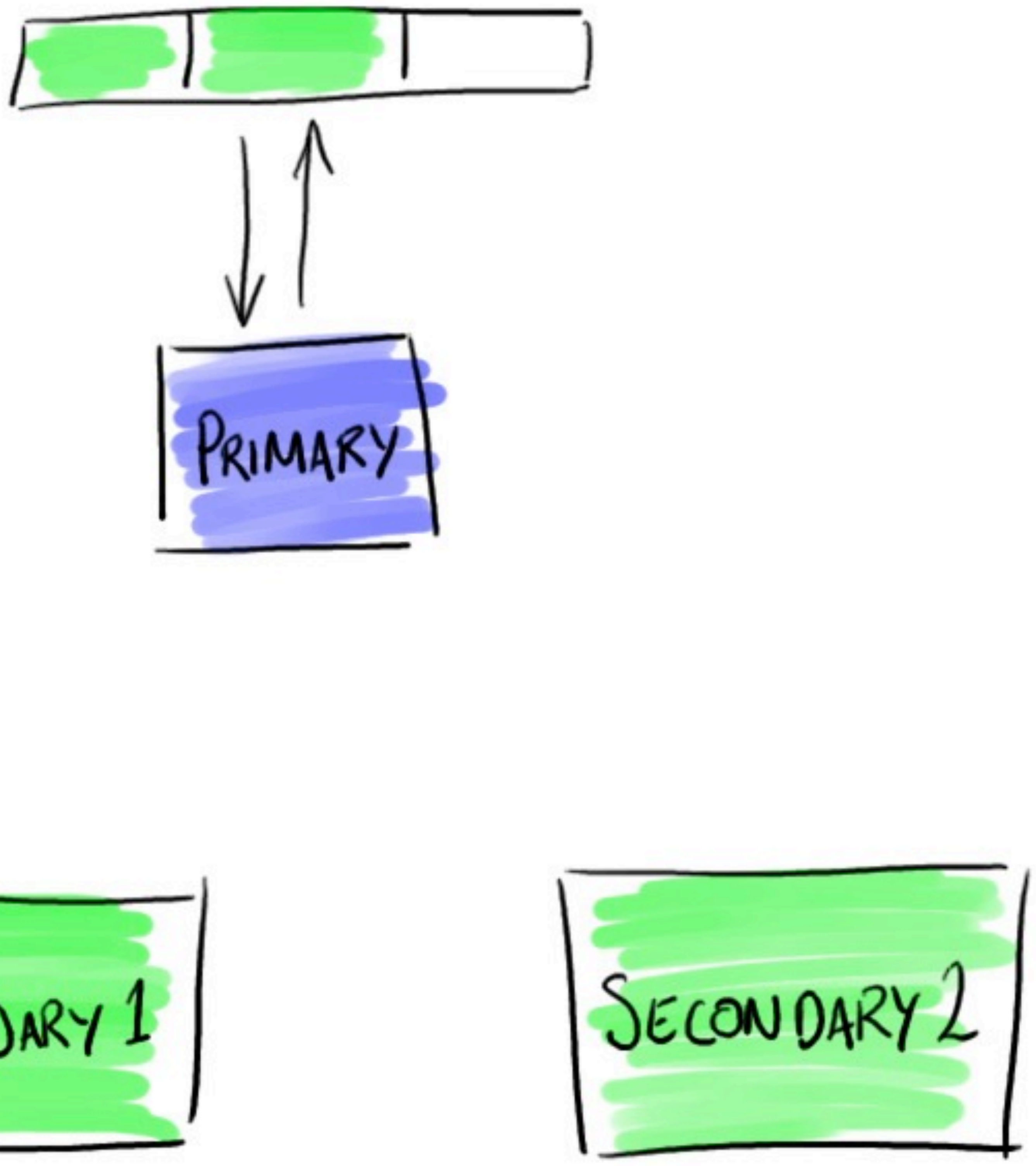

PRIMARY
SECONDARY 1
SECONDARY 2

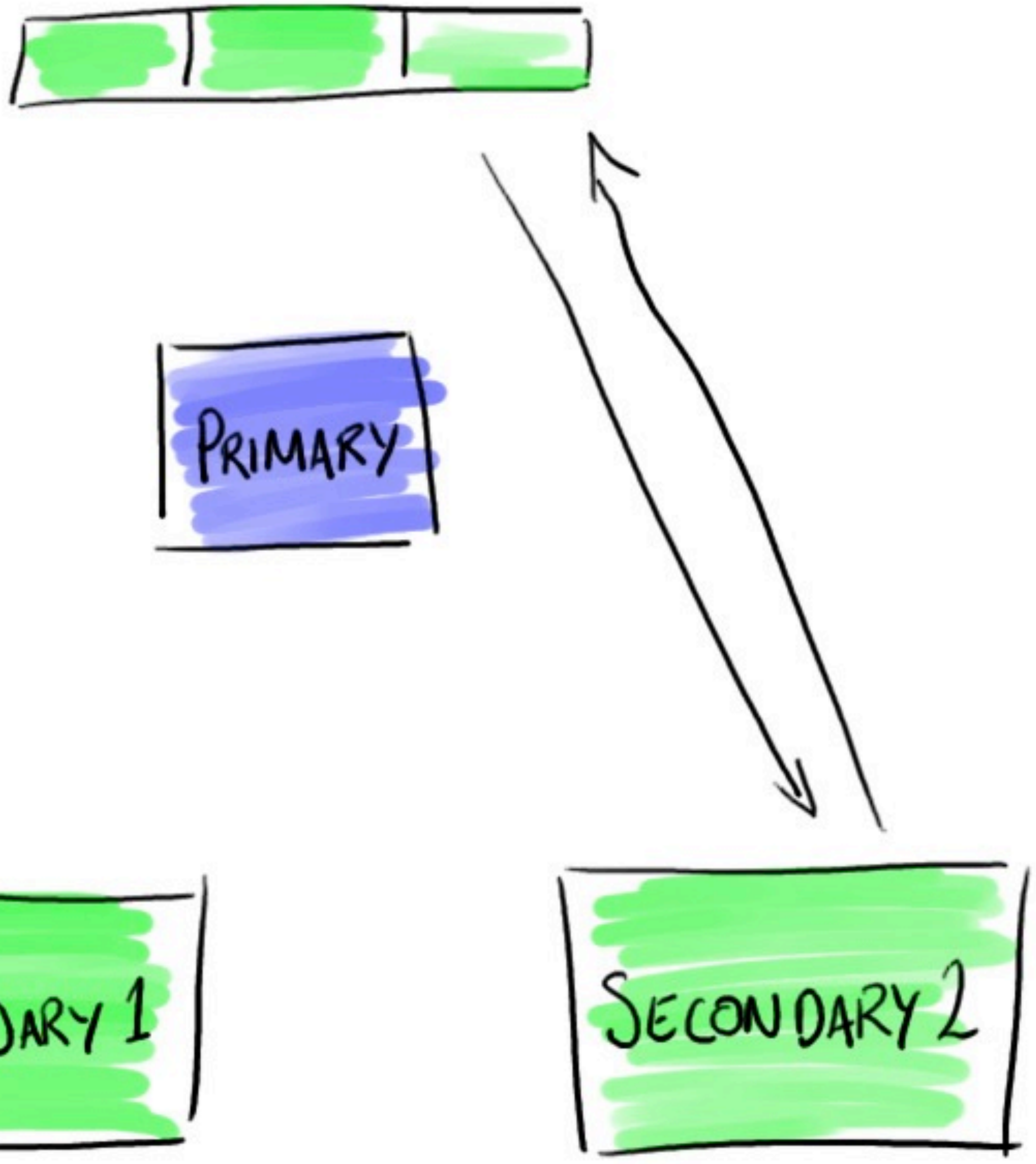

'ello?
PRIMARY
SECONDARY 1
SECONDARY 2


I'm good!
PRIMARY
SECONDARY 1
SECONDARY 2

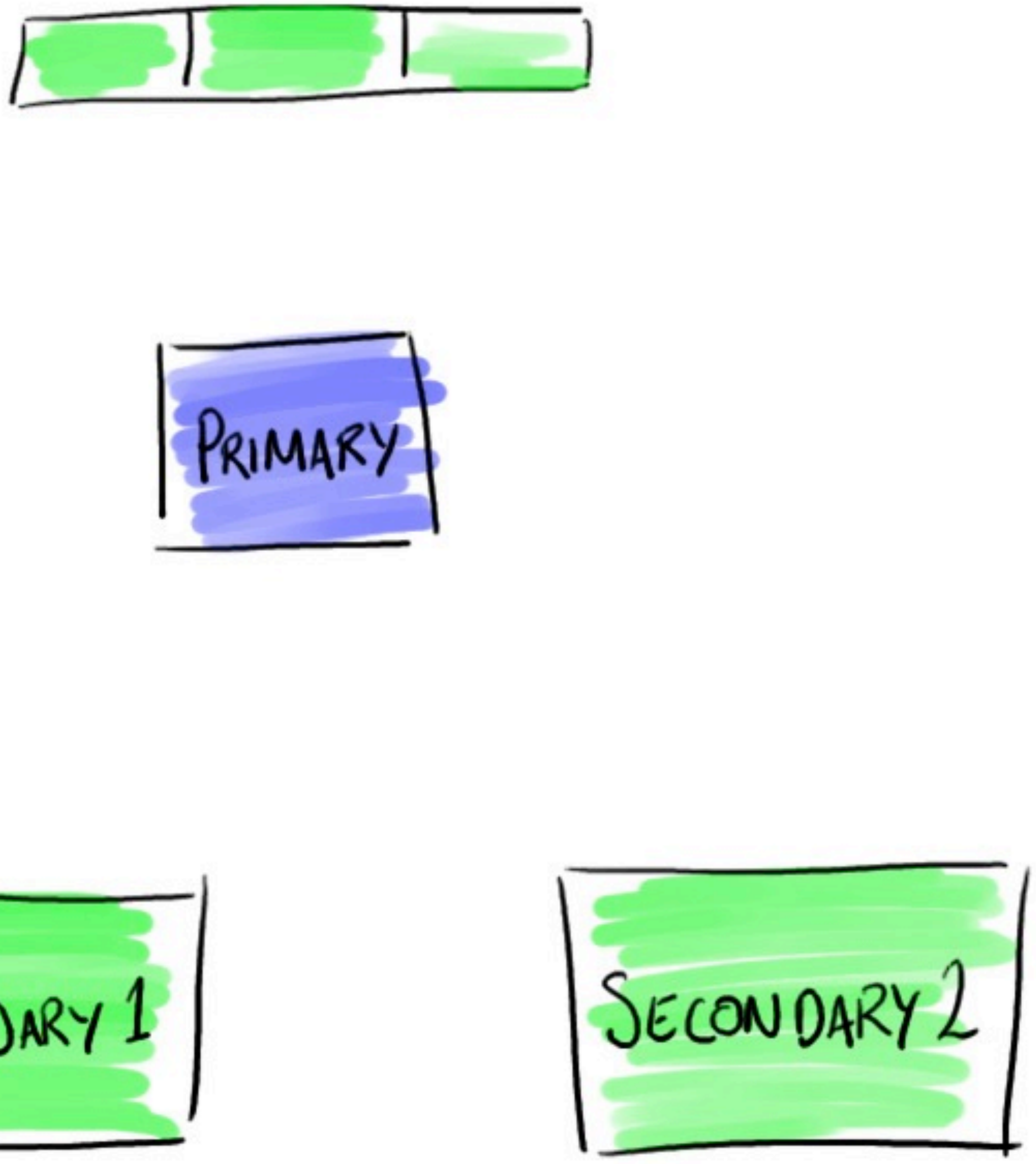

PRIMARY
SECONDARY 1
SECONDARY 2


PRIMARY
SECONDARY 1
SECONDARY 2

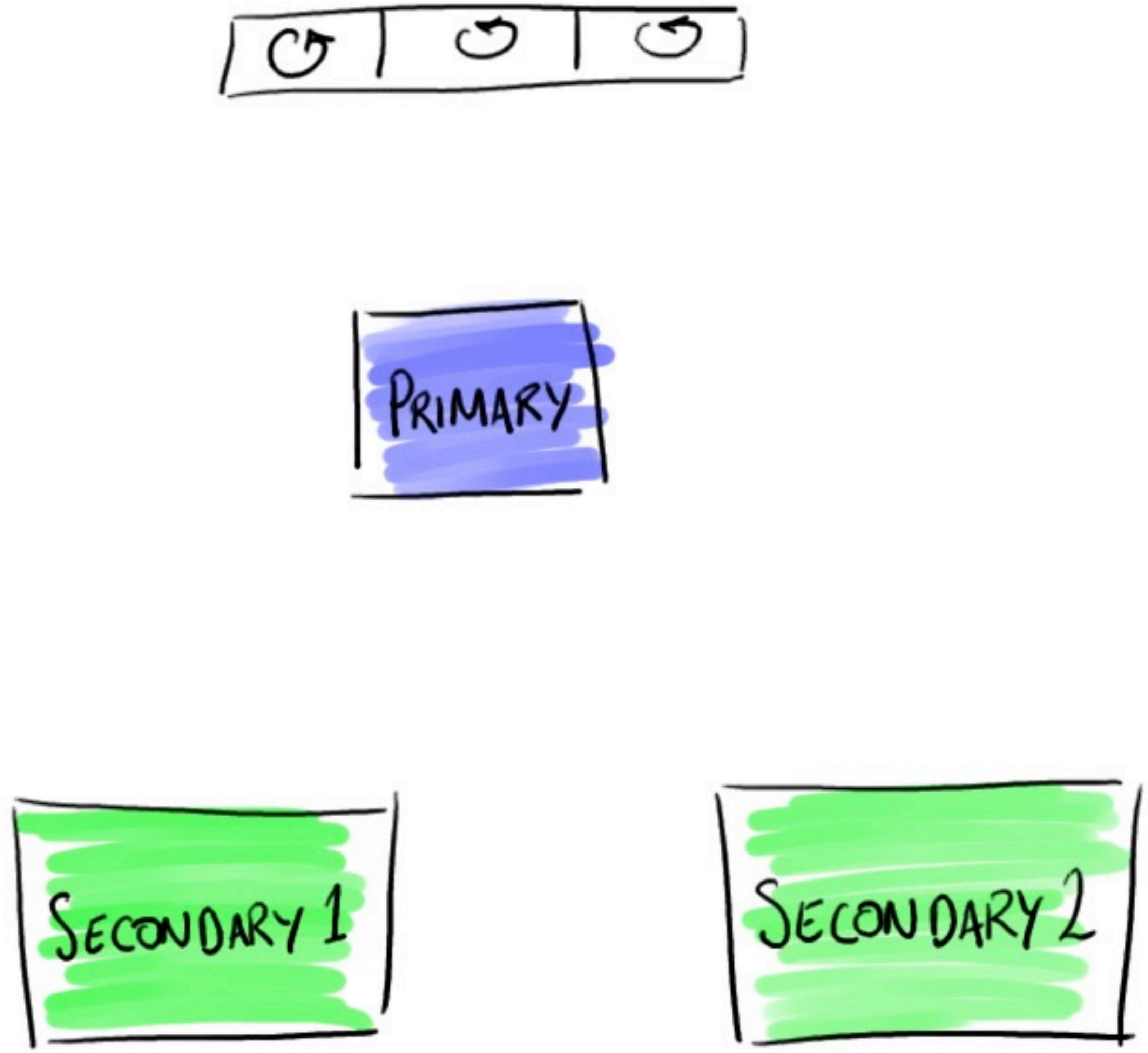

PRIMARY
SECONDARY 1
SECONDARY 2

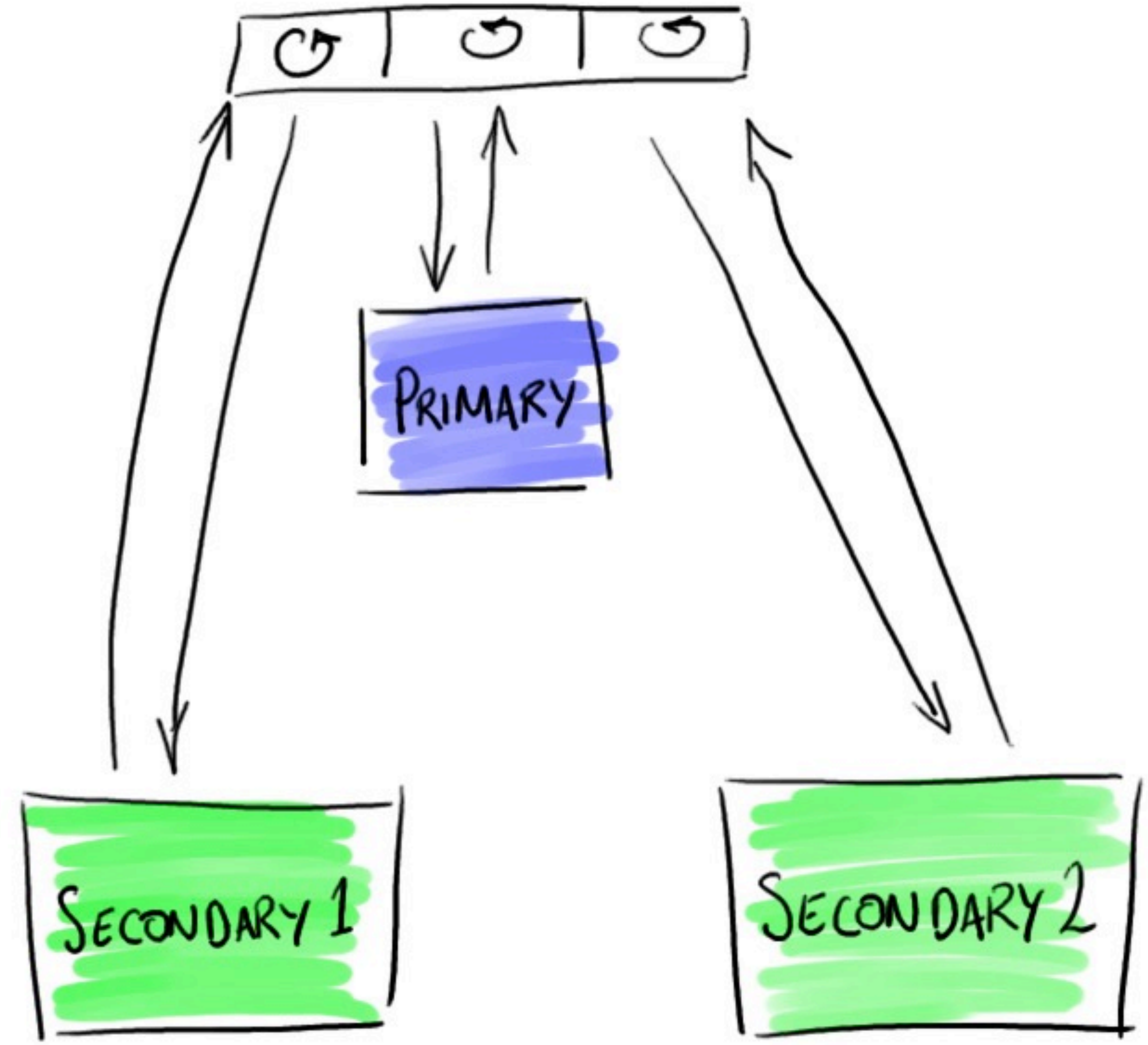

PRIMARY
SECONDARY 1
SECONDARY 2

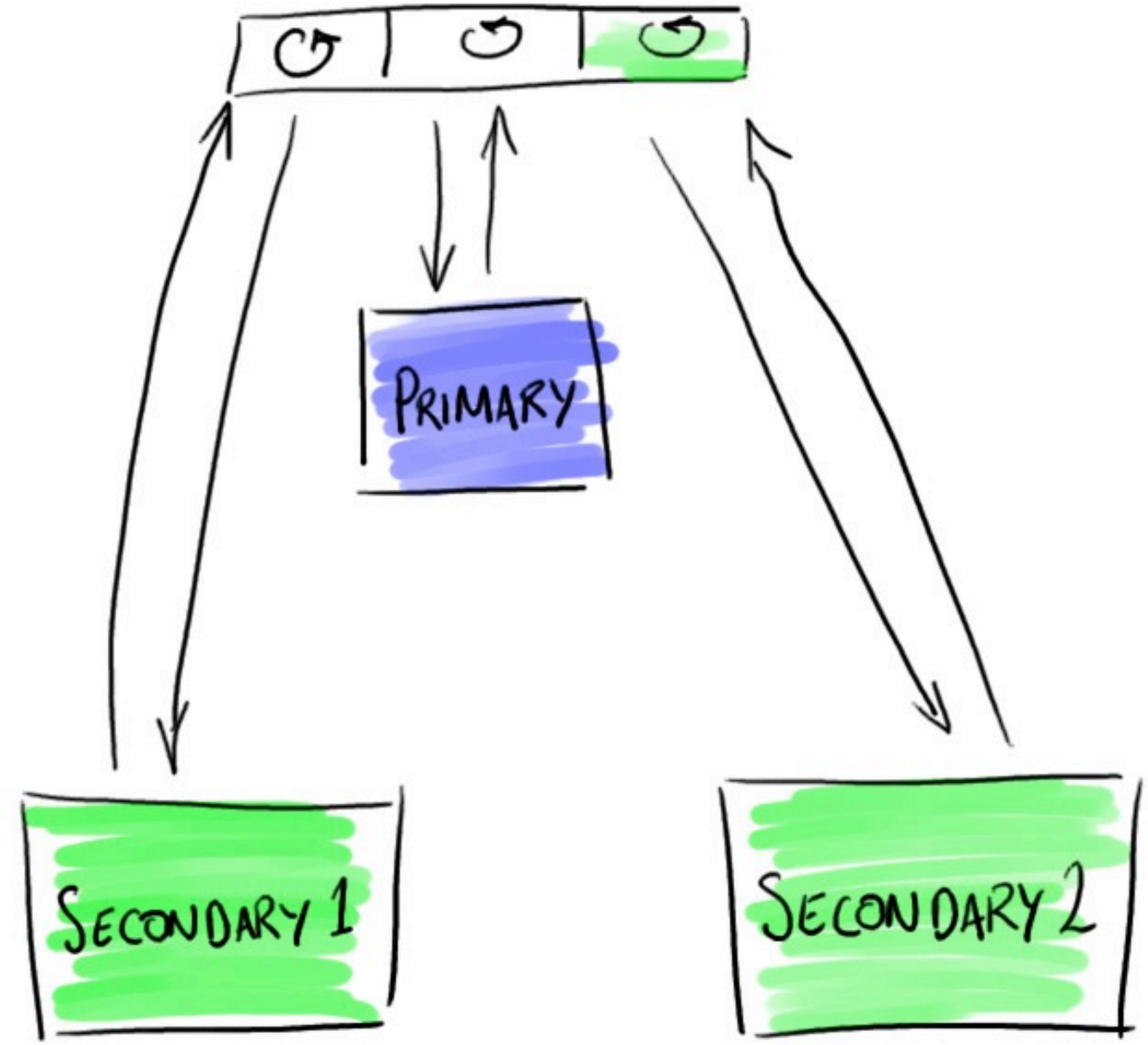
All is Well!

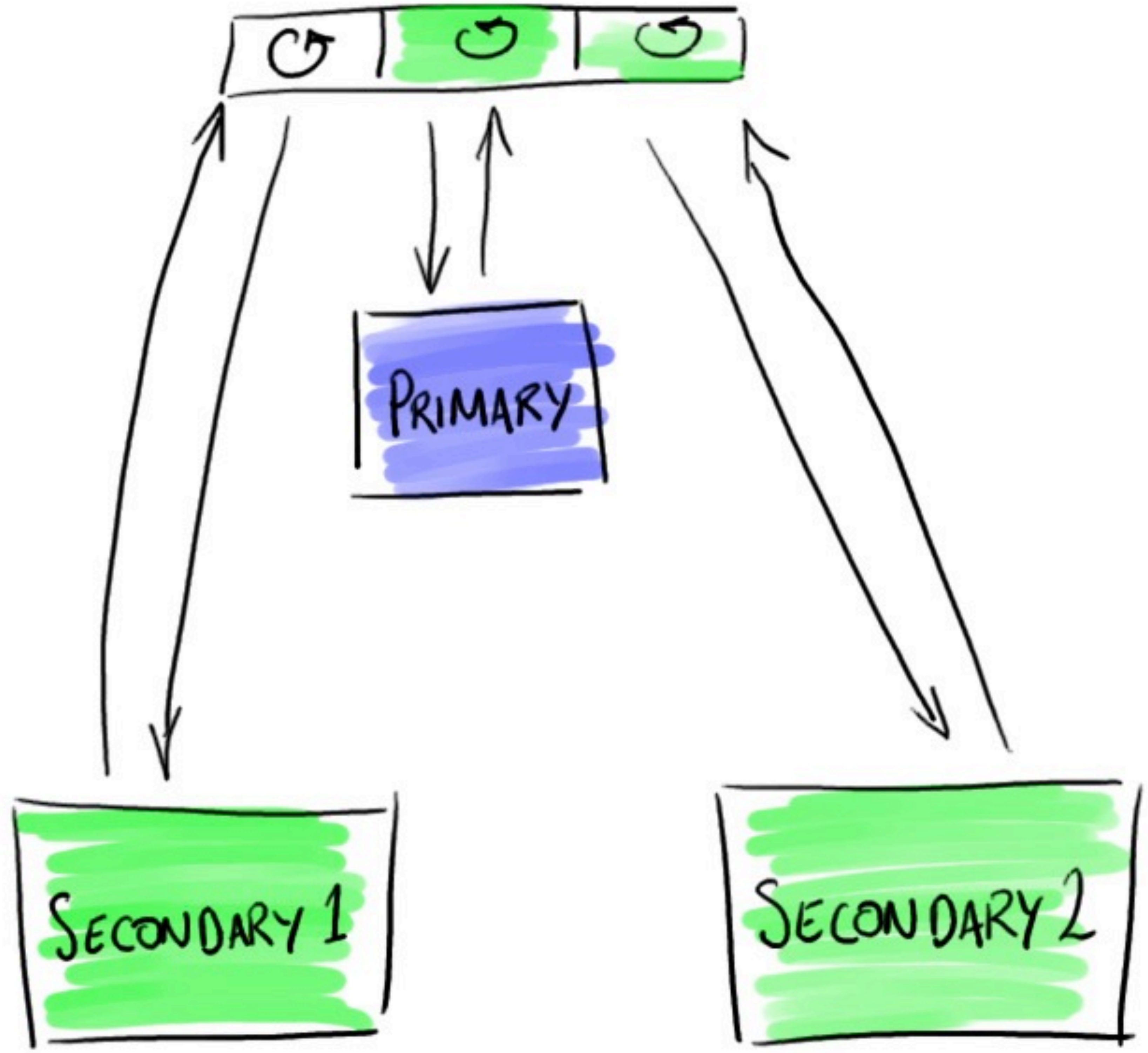


But now…


PRIMARY
SECONDARY 1
SECONDARY 2


PRIMARY
SECONDARY 1
SECONDARY 2


PRIMARY
SECONDARY 1
SECONDARY 2


PRIMARY
SECONDARY 1
SECONDARY 2

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

The New API

Caveats

- It won’t look like this
- Haven’t decided consistent names yet
- Need something that suits all drivers

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

```
patron = {
  _id: "joe",
  name: "Joe Bookreader",
  address: {
    street: "123 Fake St",
    city: "Faketon",
    state: "MA",
    zip: 12345
  }
  books: [ 27464, 747854, ...]
}
```

```java
List books = new BasicDBList();
books.add(27464, 747854);
new BasicDBObject("_id", "joe")
          .append("name", "Joe Bookreader")
          .append("address", new BasicDBObject("street", "123 Fake St")
                                            .append("city", "Faketon")
                                            .append("state", "MA")
                                            .append("zip", 12345))
          .append("books", books);

collection.insert(books);
```

Building a Document

```
DBCollection collection =
    database.getCollection("coll");

ArrayList<Patron> resultsToReturn = new ArrayList<Patron>();

DBObject query = new BasicDBObject("name", theNameToFind);
DBCursor results = collection.find(query);
for (DBObject dbObject : results) {
    Patron patron = new Patron((String) dbObject.get("name"),
                               new Address((String)dbObject.get("street"),
                                           (String)dbObject.get("city"),
                                           (String)dbObject.get("state"),
                                           (Integer)dbObject.get("zip")),
                               (BasicDBList)dbObject.get("books"));
    resultsToReturn.add(patron);
}
return resultsToReturn;
```

Getting it back

```java
DBCollection collection =
    database.getCollection("coll");

ArrayList<Patron> resultsToReturn = new ArrayList<Patron>();

DBObject query = new BasicDBObject("name", theNameToFind);
DBCursor results = collection.find(query);
for (DBObject dbObject : results) {
    Patron patron = new Patron((String) dbObject.get("name"),
                               new Address((String)dbObject.get("street"),
                                           (String)dbObject.get("city"),
                                           (String)dbObject.get("state"),
                                           (Integer)dbObject.get("zip")),
                               (BasicDBList)dbObject.get("books"));
    resultsToReturn.add(patron);
}
return resultsToReturn;
```

Casting is fun

```
MongoCollection<Patron> collection =
    database.getCollection("coll", new PatronCodec());



Document query = new Document("name", theNameToFind);









return collection.find(query).into(new ArrayList<Patron>());
```

New API

```
MongoCollection<Patron> collection =
    database.getCollection("coll", new PatronCodec());
```

```
Document query = new Document("name", theNameToFind);
```

```
return collection.find(query).into(new ArrayList<Patron>());
```

Separation of concerns

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

Find

```
collection.find(query).skip(1000).limit(100);
```

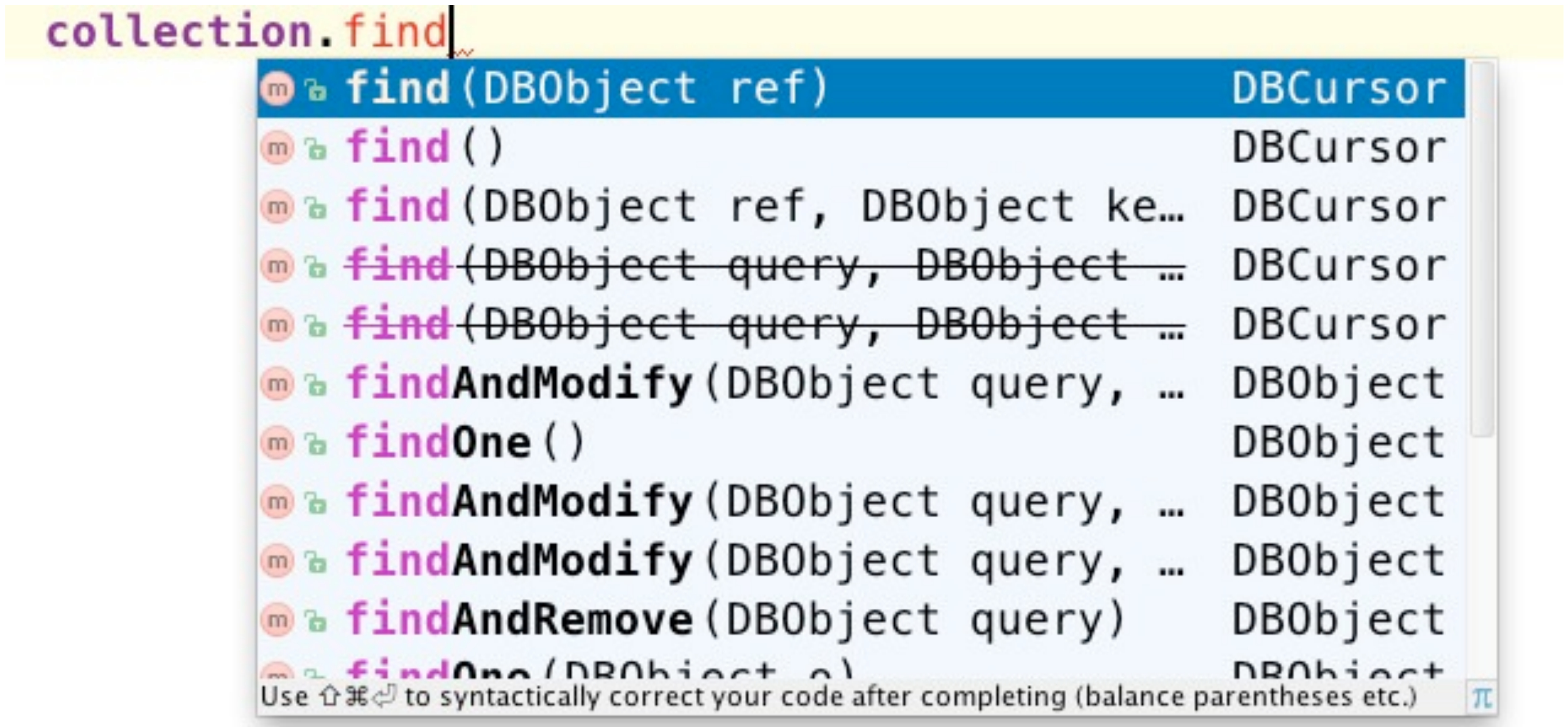
Find

```
collection.find(query).skip(1000).limit(100);

collection.find(query).skip(1000).limit(100);
```

Find

```
collection.find(query).skip(1000).limit(100);

collection.find(query).skip(1000).limit(100);


collection.find(query, fields);
```

Find



```
collection.find
find(DBObject ref)                          DBCursor
find()                                      DBCursor
find(DBObject ref, DBObject ke…             DBCursor
~~find(DBObject query, DBObject …~~         DBCursor
~~find(DBObject query, DBObject …~~         DBCursor
findAndModify(DBObject query, …             DBObject
findOne()                                   DBObject
findAndModify(DBObject query, …             DBObject
findAndModify(DBObject query, …             DBObject
findAndRemove(DBObject query)               DBObject
findOne(DBObject o)                         DBObject
Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.)
```

Which One?

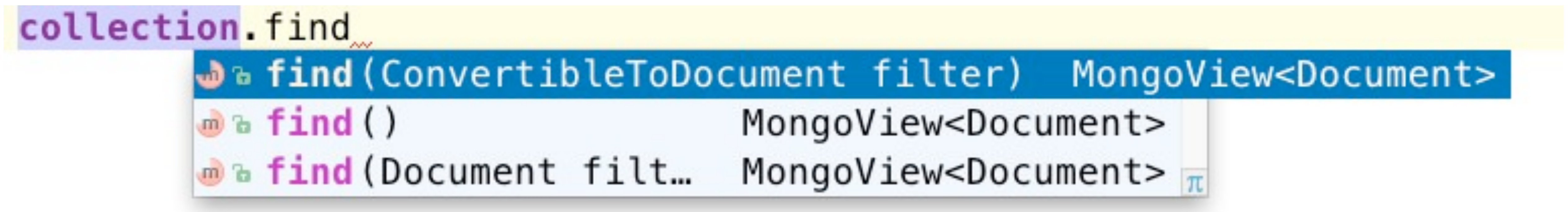
Find

```
collection.find(query).skip(1000).limit(100);

collection.find(query).skip(1000).limit(100);


collection.find(query, fields);
```

Find

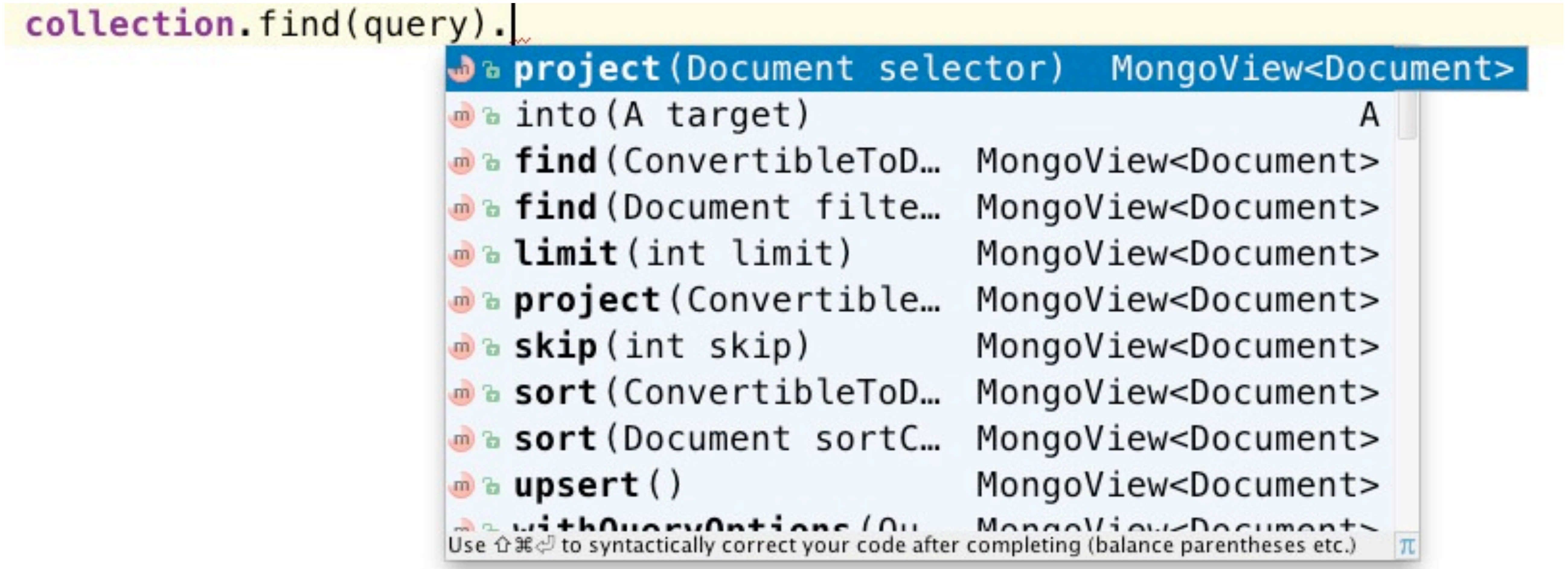
```
collection.find(query).skip(1000).limit(100);

collection.find(query).skip(1000).limit(100);


collection.find(query, fields);

collection.find(query).project(fields);
```

Find



```
collection.find
find(ConvertibleToDocument filter)  MongoView<Document>
find()                              MongoView<Document>
find(Document filt…                 MongoView<Document>
```

Fewer Decisions



```
collection.find(query).
project(Document selector)   MongoView<Document>
into(A target)                                 A
find(ConvertibleToD…     MongoView<Document>
find(Document filte…     MongoView<Document>
limit(int limit)         MongoView<Document>
project(Convertible…     MongoView<Document>
skip(int skip)           MongoView<Document>
sort(ConvertibleToD…     MongoView<Document>
sort(Document sortC…     MongoView<Document>
upsert()                 MongoView<Document>
withQueryOptions(Qu…     MongoView<Document>
```

Use ⇧⌘⏎ to syntactically correct your code after completing (balance parentheses etc.)

“Ctrl + space” friendly

Remove

```
collection.remove(query);
```

Remove

```
collection.remove(query);

collection.find(query).remove();
```

Remove

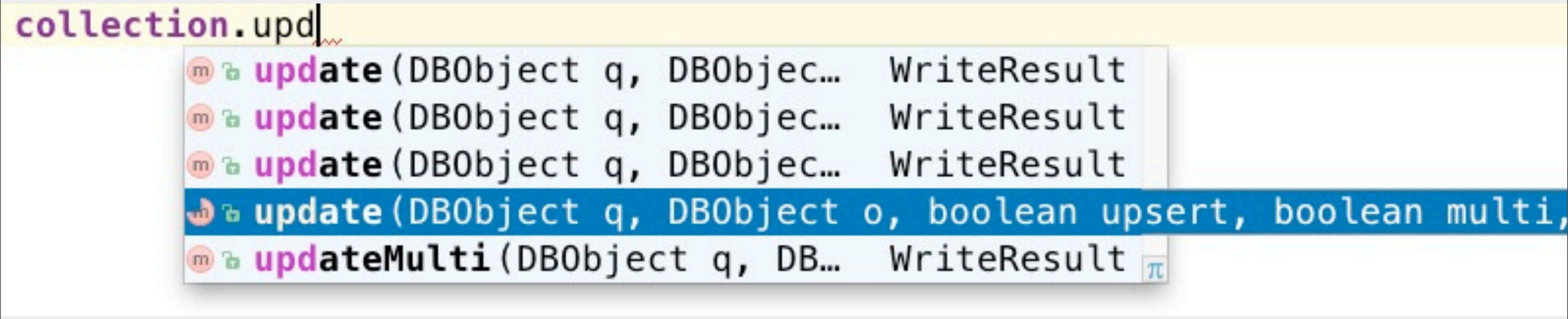
Update

```
collection.update(query, newValues)
```

Update

```
collection.update(query, newValues)

collection.find(query).updateOne(newValues);
```

Update



```
collection.upd
  update(DBObject q, DBObjec…   WriteResult
  update(DBObject q, DBObjec…   WriteResult
  update(DBObject q, DBObjec…   WriteResult
  update(DBObject q, DBObject o, boolean upsert, boolean multi,
  updateMulti(DBObject q, DB…   WriteResult
```

Overloaded Methods

Update

```
collection.update(query, newValues);

collection.find(query).updateOne(newValues);


collection.update(query, newValues, false, false, JOURNALED);
```

Update

```
collection.update(query, newValues);

collection.find(query).updateOne(newValues);


collection.update(query, newValues, false, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .updateOne(newValues);
```

Update

Update

```
collection.update(query, newValues);

collection.find(query).updateOne(newValues);


collection.update(query, newValues, false, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .updateOne(newValues);


collection.update(query, newValues, true, false, JOURNALED);
```

Update

```
collection.update(query, newValues);

collection.find(query).updateOne(newValues);


collection.update(query, newValues, false, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .updateOne(newValues);


collection.update(query, newValues, true, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .upsert()
          .updateOne(newValues);
```

Update

Update

```
collection.update(query, newValues);

collection.find(query).updateOne(newValues);


collection.update(query, newValues, false, false, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .updateOne(newValues);


collection.update(query, newValues, true, true, JOURNALED);

collection.find(query)
          .withWriteConcern(JOURNALED)
          .upsert()
          .update(newValues);
```

Update

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

MongoDB is an open-source document database, featuring:

- Document-Oriented Storage
- Full Index Support
- Replication & High Availability
- Auto-Sharding
- Querying
- Fast In-Place Updates
- Map/Reduce
- GridFS

Atomic Operations

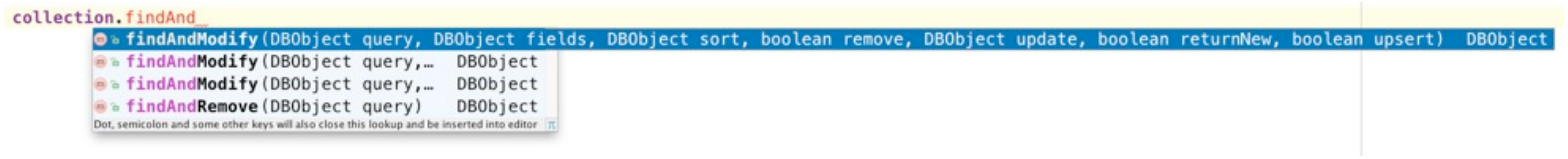
Find and Modify

```
collection.findAndModify(query, newValues);
```

Find and Modify

```
collection.findAndModify(query, newValues);

collection.find(query)
          .getOneAndUpdate(newValues);
```

Find and Modify



```
collection.findAnd
findAndModify(DBObject query, DBObject fields, DBObject sort, boolean remove, DBObject update, boolean returnNew, boolean upsert)  DBObject
findAndModify(DBObject query,…   DBObject
findAndModify(DBObject query,…   DBObject
findAndRemove(DBObject query)    DBObject
Dot, semicolon and some other keys will also close this lookup and be inserted into editor
```

They hate me!

Find and Modify

```
collection.findAndModify(query, update);

collection.find(query)
          .getOneAndUpdate(update);


collection.findAndModify(query,
                         fields,
                         criteria,
                         false,
                         newValues,
                         false,
                         false);
```

Find and Modify

```
collection.findAndModify(query, update);

collection.find(query)
          .getOneAndUpdate(update);


collection.findAndModify(query,
                         fields,
                         criteria,
                         false,
                         newValues,
                         false,
                         false);

collection.find(query)
          .project(fields)
          .sort(criteria)
          .getOneAndUpdate(newValues);
```

Find and Modify

```
collection.findAndModify(query, update);

collection.find(query)
          .getOneAndUpdate(update);


collection.findAndModify(query,
                         fields,
                         criteria,
                         false,
                         newValues,
                         true,
                         false);

collection.find(query)
          .project(fields)
          .sort(criteria)
          .updateOneAndGet(newValues);
```

Find and Modify

Consistency

```
collection.find(query).count();

collection.find(query).remove();

collection.find(query).update(newValues);

collection.find(query).updateOneAndGet(newValues);

collection.find(query).getOneAndUpdate(newValues);


collection.find(query).sort(sortCriteria).skip(9).limit(10).get();

collection.find(query).sort(sortCriteria).skip(9).limit(10).getOne();


collection.find(query).sort(ascending("name")).getOne();
```

Consistency at last

How do we know we did it?

Tests Pass...

…and more tests pass…

...even more tests pass...

It’s being used in anger

Conclusion

Conclusion

Conclusion

- Model your domain

Conclusion

- Model your domain
- Know your users

Conclusion

- Model your domain
- Know your users
- API design is hard

#gotober

@trisha_gee

Questions

http://is.gd/java3mongodb

mongoDB

```
MongoFuture<Document> future =
    collection.find(query).sort(criteria).skip(9).limit(10).asyncOne();

MongoFuture<Long> count = collection.find(query).asyncCount();

MongoFuture<WriteResult> replaceResult =
    collection.find(query).asyncReplace(replacement);
```

What about async?

Understandable Exceptions

- Client Exceptions
- Server Exceptions
- No more parsing error Strings

Documentation

- Self documenting code
- JavaDoc
- Unit, Functional and Acceptance Tests
- Blogs
- Tutorials

My Questions

mongoDB

1. Are you using the Java driver?

2. What do you like about it?

3. What are your pain points?