Know NoSQL

Cheap jokes and pointless blathering on about MongoDB and other NoSQL engines

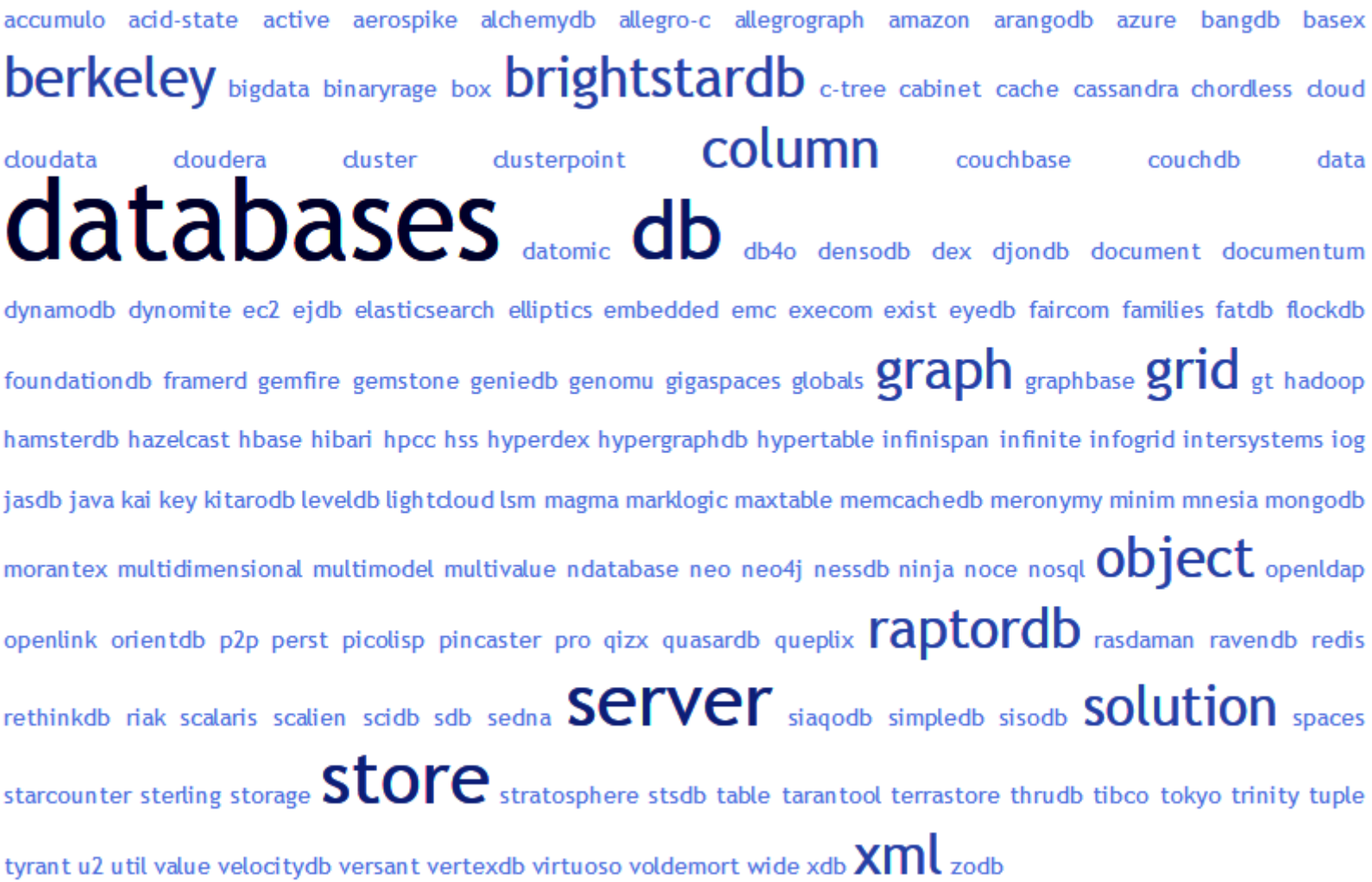
Who is Dan Wilson

- Blog @ nodans.com, dzone.com
- Managing Director of Model Glue
- Manager of TACFUG, NCDevCon
- Principal Partner of DataCurl LLC: Home of ChallengeWave.com

NoSQL Engines are fast and loose ways of storing and retrieving data

"A DBA walks into a NOSQL bar, but turns and leaves because he couldn't find a table" (webtonull)

Landscape


accumulo acid-state active aerospike alchemydb allegro-c allegrograph amazon arangodb azure bangdb basex
berkeley bigdata binaryrage box brightstardb c-tree cabinet cache cassandra chordless cloud
cloudata cloudera cluster clusterpoint column couchbase couchdb data
databases datomic db db4o densodb dex djondb document documentum
dynamodb dynomite ec2 ejdb elasticsearch elliptics embedded emc execom exist eyedb faircom families fatdb flockdb
foundationdb framerd gemfire gemstone geniedb genomu gigaspaces globals graph graphbase grid gt hadoop
hamsterdb hazelcast hbase hibari hpcc hss hyperdex hypergraphdb hypertable infinispan infinite infogrid intersystems iog
jasdb java kai key kitarodb leveldb lightcloud lsm magma marklogic maxtable memcachedb meronymy minim mnesia mongodb
morantex multidimensional multimodel multivalue ndatabase neo neo4j nessdb ninja noce nosql object openldap
openlink orientdb p2p perst picolisp pincaster pro qizx quasardb queplix raptordb rasdaman ravendb redis
rethinkdb riak scalaris scalien scidb sdb sedna server siaqodb simpledb sisodb solution spaces
starcounter sterling storage store stratosphere stsdb table tarantool terrastore thrudb tibco tokyo trinity tuple
tyrant u2 util value velocitydb versant vertexdb virtuoso voldemort wide xdb xml zodb

Examples of NoSQL You Already Use

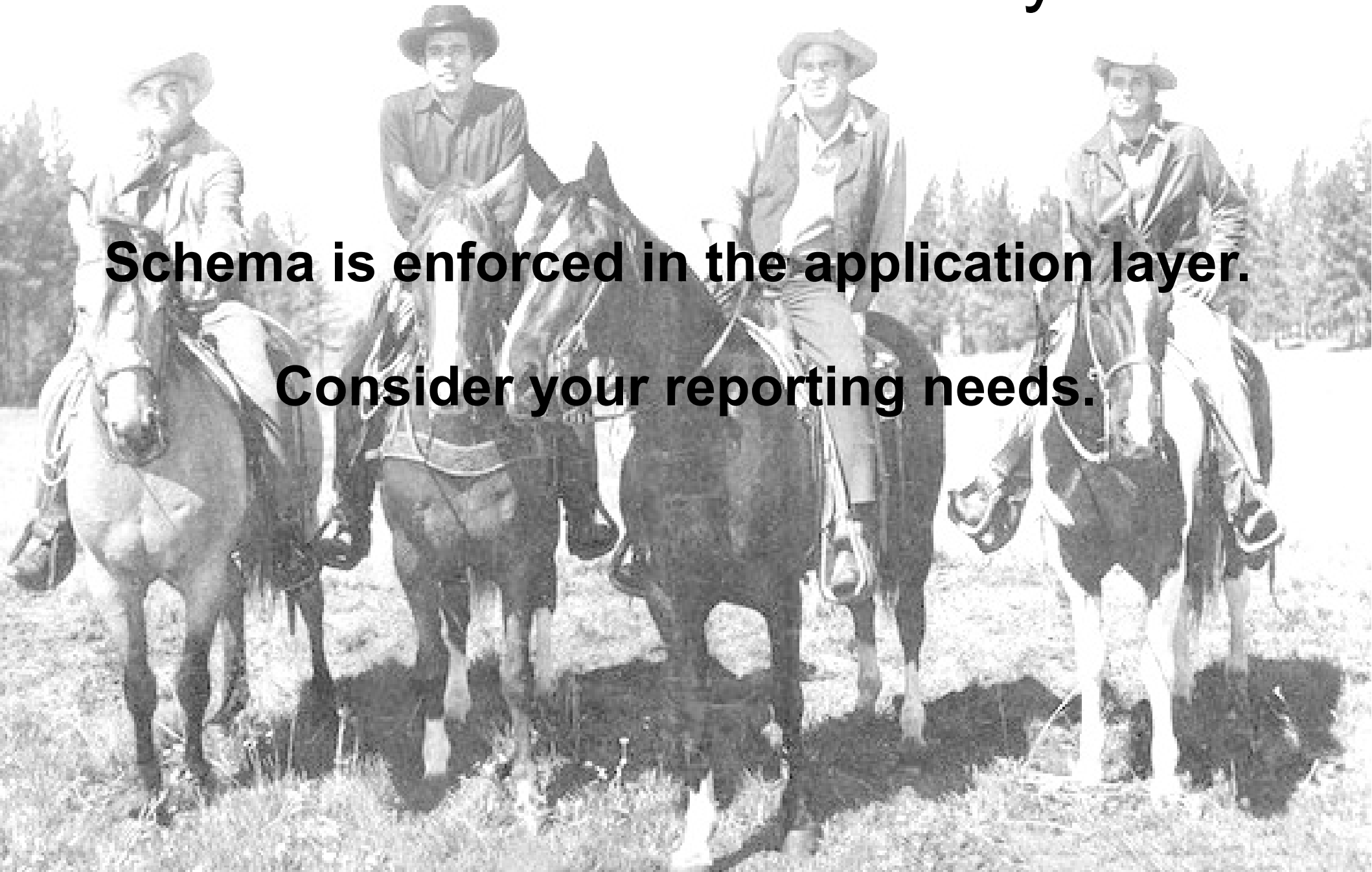
BerkleyDB: Subversion

OpenLDAP: Single Sign-On

MemcachedD and ehCache: Application level caching

IBM Lotus/Domino: Email (actually, I hope you aren't using this)

Schemaless? Not Really....



Schema is enforced in the application layer.

Consider your reporting needs.

Document Databases

```
{
        "_id" : ObjectId("51fff472e09a41ac19dd1876"),
        "NAME" : "Marc",
        "LOVESMONGO" : true,
        "COUNTER" : 1,
        "LOVESSQL" : true,
        "KIDS" : [
                {
                        "NAME" : "Alexis",
                        "AGE" : 7,
                        "DESCRIPTION" : "crazy",
                        "HAIR" : "blonde"
                },

                {
                        "NAME" : "Pauly",
                        "AGE" : 0
                }
        ]
}
```

So, How Does It Store Data?

Database = Database

Table = Collection

Row = Document

Join = Ouch

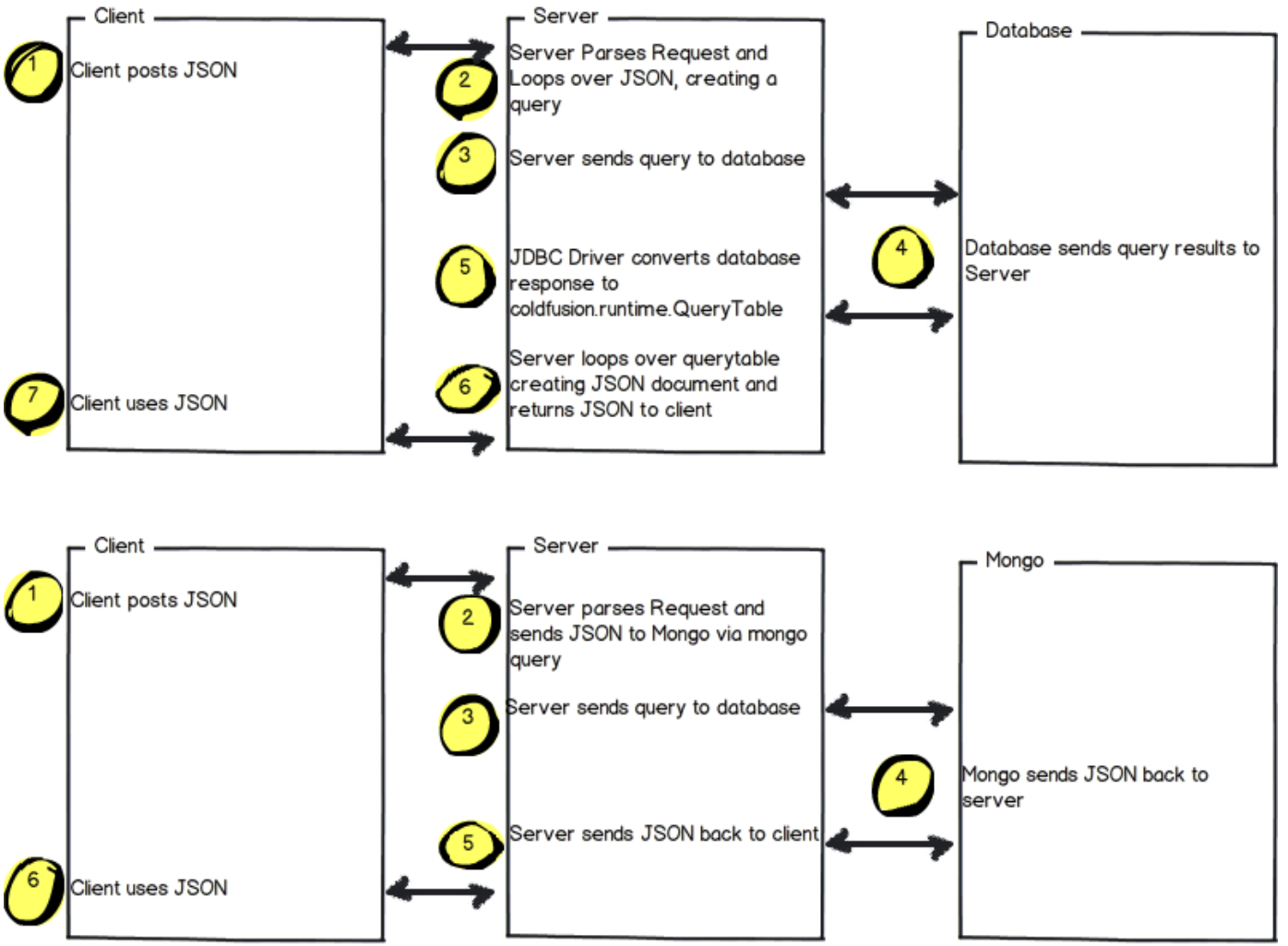
Ouch = Scalable

Ouch = Shardable

Document Collections

```
> db.celebs.find().pretty()
{
        "_id" : ObjectId("51fff479e09a41ac19dd188a"),
        "ZIP" : "90210",
        "NAME" : "Warren Beatty"
},
{
        "_id" : ObjectId("51fff479e09a41ac19dd188d"),
        "ZIP" : "90265",
        "NAME" : "Bill Murray"
},
{
        "_id" : ObjectId("51fff479e09a41ac19dd188e"),
        "ZIP" : "90272",
        "NAME" : "Bill Cosby"
}
```

Getting JSON: Relational vs Document



NoSQL vs SQL Examples

SQL

```
CREATE TABLE users (
    id MEDIUMINT NOT NULL AUTO_INCREMENT,
    user_id Varchar(30), age Number, status char(1),
    PRIMARY KEY (id)
)
```

MongoDB

```
db.users.insert( {
    user_id: "abc123",
    age: 55,
    status: "A"
 } )
```

NoSQL vs SQL Examples

SQL

```
CREATE INDEX idx_userid_asc_age_desc
ON users(
  user_id,
  age DESC
)
```

MongoDB

```
db.users.ensureIndex( {
  user_id: 1,
  age: -1
} )
```

NoSQL vs SQL Examples

SQL

```
INSERT INTO users( user_id, age, Status)
VALUES ( "bcd001", 45, "A")
```

MongoDB

```
db.users.insert( {
  user_id: "bcd001", age: 45, status: "A"
} )
```

NoSQL vs SQL Examples

SQL

```
SELECT *
FROM user
```

MongoDB

```
db.users.find()
```

NoSQL vs SQL Examples

```
SELECT DISTINCT(status)
FROM users
```

SQL

```
db.users.distinct( "status" )
```

MongoDB

NoSQL vs SQL Examples

SQL

```
SELECT *
FROM users
WHERE status = "A"
```

MongoDB

```
db.users.find(
    { status: "A" }
)
```

NoSQL vs SQL Examples

SQL

```
SELECT user_id, status
FROM users
WHERE status = "A"
```

MongoDB

```
db.users.find(
    { status: "A" },
    { user_id: 1, status: 1, _id: 0 }
)
```

NoSQL vs SQL Examples

SQL

```
SELECT *
FROM users
WHERE status = "A" OR age = 50
```

MongoDB

```
db.users.find(
    {
       $or: [  { status: "A" } , { age: 50 } ]
    }
)
```

NoSQL vs SQL Examples

SQL

```
SELECT *
FROM users
WHERE age > 25 AND   age <= 5
```

MongoDB

```
db.users.find(
   { age: { $gt: 25, $lte: 50 } }
)
```

NoSQL vs SQL Examples

SQL

```
SELECT *
FROM users
WHERE age > 25 AND   age <= 5
```

MongoDB

```
db.users.find(
   { age: { $gt: 25, $lte: 50 } }
)
```

NoSQL vs SQL Examples

SQL

```
SELECT *
FROM users
WHERE user_id like "bc%"
```

MongoDB

```
db.users.find(
   { user_id: /^bc/ }
)
```

NoSQL vs SQL Examples

SQL

```
SELECT *
FROM users
LIMIT 5
SKIP 10
```

MongoDB

```
db.users.find().limit(5).skip(10)
```

NoSQL vs SQL Examples

SQL

```
UPDATE users
SET status = "C"
WHERE age > 25
```

MongoDB

```
db.users.update(
   { age: { $gt: 25 } },
   { $set: { status: "C" } },
   { multi: true }
)
```

NoSQL vs SQL Examples

SQL

```
DELETE FROM users
WHERE status = "D"
```

MongoDB

```
db.users.remove(
   { status: "D" }
)
```

NoSQL vs SQL Examples

SQL

```
SELECT COUNT(*) AS count
FROM orders
```

MongoDB

```
db.orders.aggregate( [
  { $group: { _id: null, count: { $sum: 1 } } }
] )
```

NoSQL vs SQL Examples

SQL

```
SELECT cust_id, SUM(price) AS total
FROM orders
GROUP BY cust_id
ORDER BY total
```

MongoDB

```
db.orders.aggregate( [
  { $group: { _id: "$cust_id", total: { $sum: "$price" } } },
  { $sort: { total: 1 } }
] )
```

NoSQL vs SQL Examples

SQL

```
SELECT cust_id, SUM(price) AS total
FROM orders
GROUP BY cust_id
ORDER BY total
```

MongoDB

```
db.orders.aggregate( [
  { $group: { _id: "$cust_id", total: { $sum: "$price" } } },
  { $sort: { total: 1 } }
] )
```

NoSQL vs SQL Examples

SQL

```
SELECT COUNT(*)
FROM (SELECT cust_id, ord_date
      FROM orders
      GROUP BY cust_id, ord_date)
as DerivedTable
```

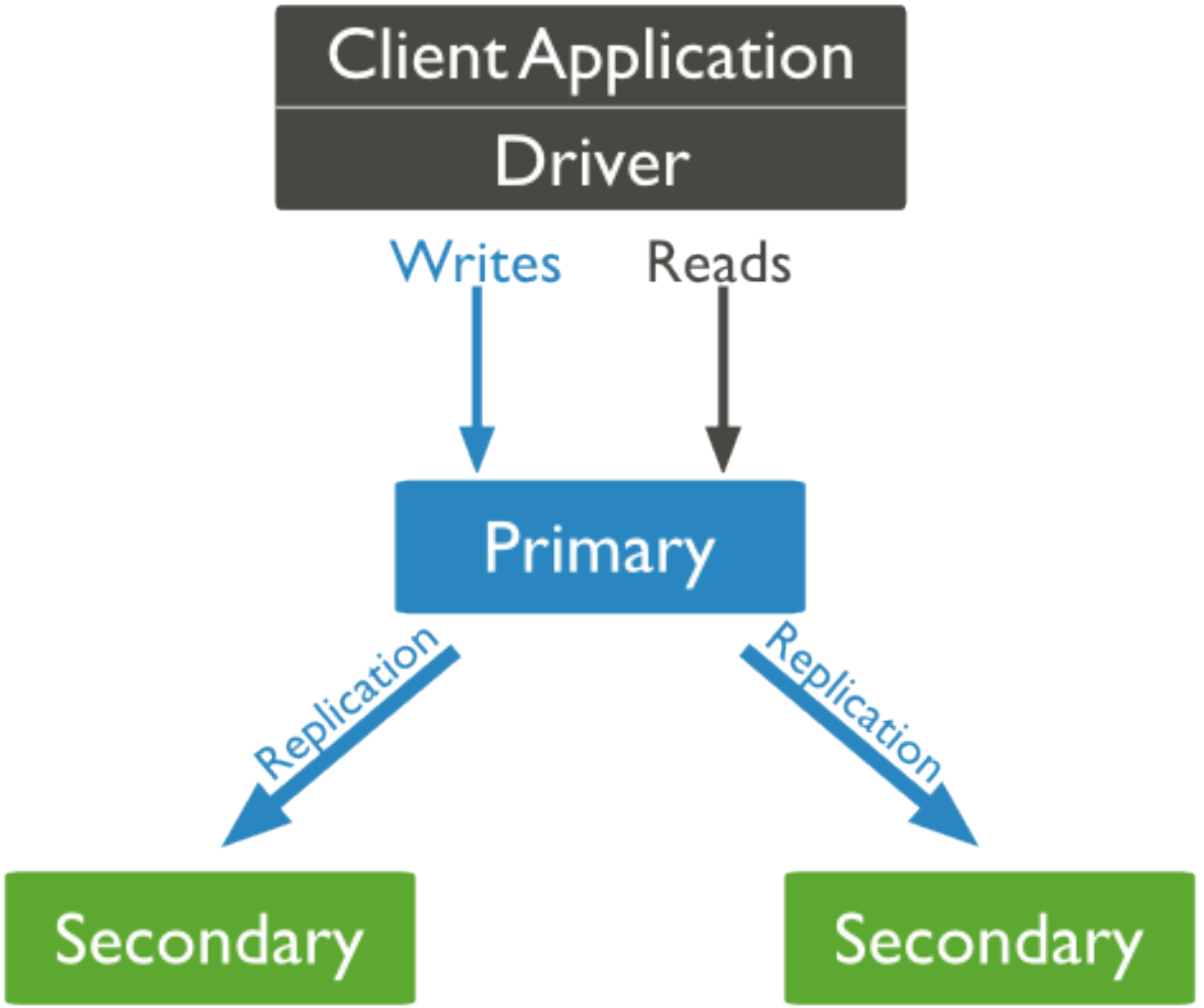
MongoDB

```
db.orders.aggregate( [
   { $group: { _id: { cust_id: "$cust_id",  ord_date: "$ord_date" } } },
   { $group: { _id: null, count: { $sum: 1 } } }
] )
```

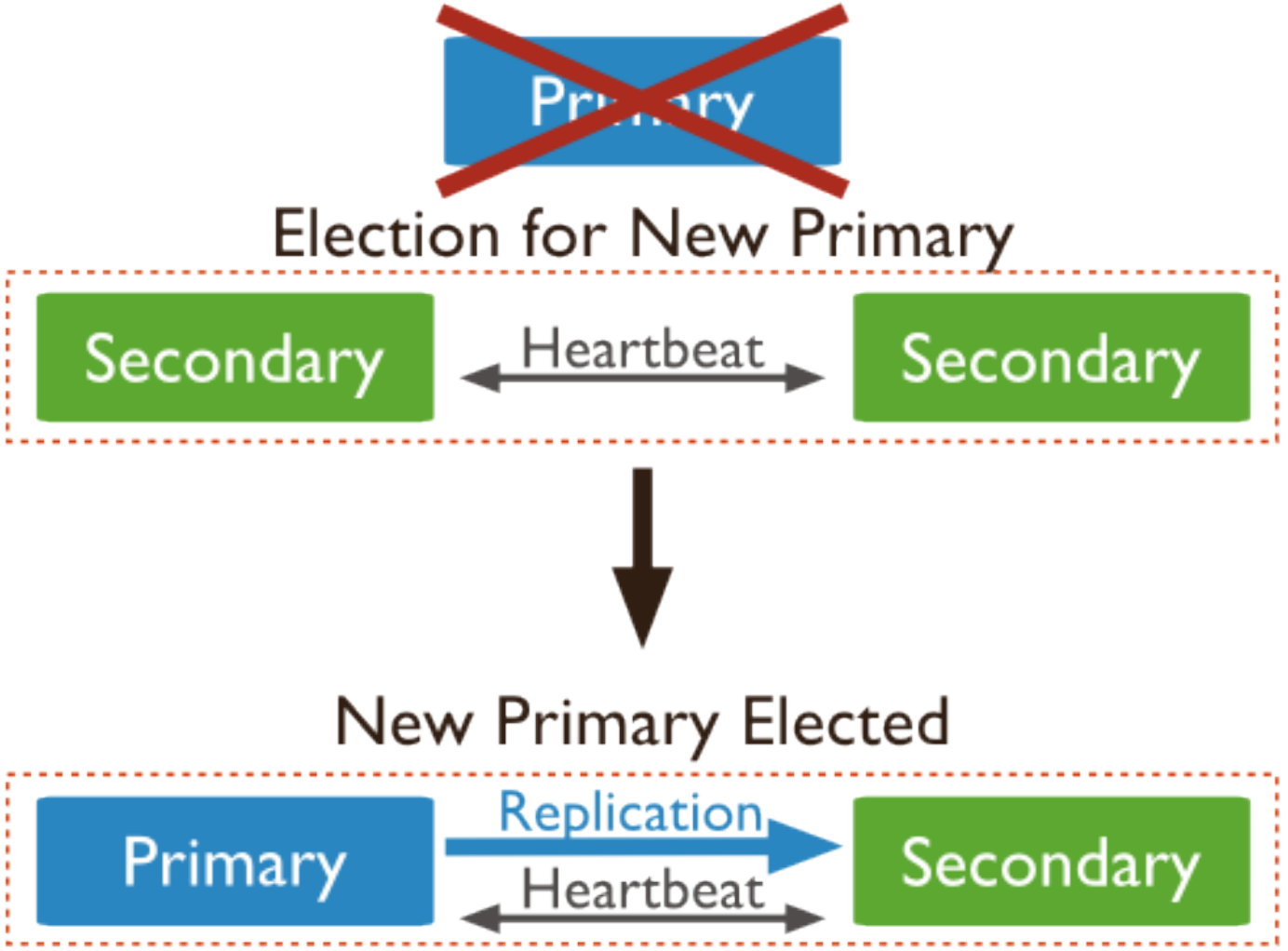
Divergence from Relational Database Models

- No Natural Cross Collection Joining
  - But you can use Map/Reduce
- No Need to create new tables/columns
- No Referential Integrity
  - But this can be a good thing

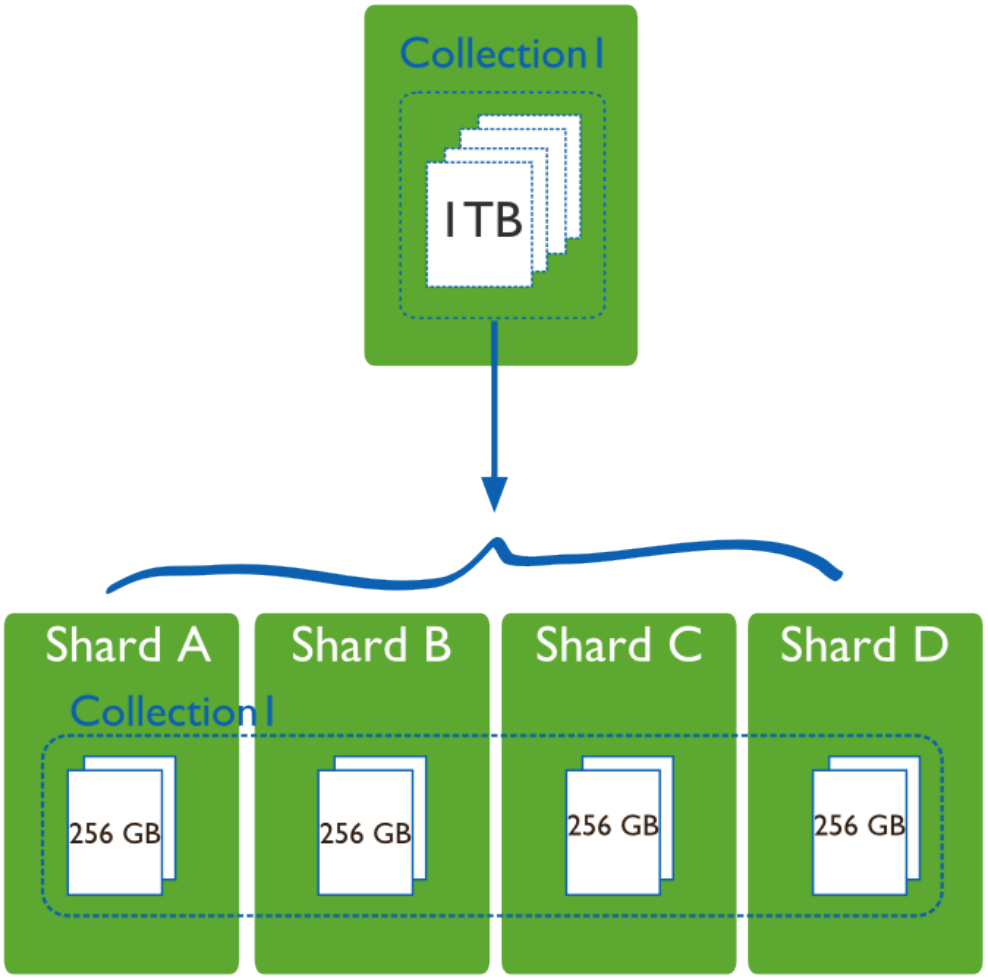
Advantages - Replication



Advantages - Failover



Advantages - Sharding



Storage formats

{"BSON": ["awesome", 5.05, 1986]}

"\x31\x00\x00\x00\x04BSON\x00\x26\x00\x00\x00\x020\x00\x08\x00\x00\x00awesome\x00\x011\x00\x33\x33\x33\x33\x33\x33\x14\x40\x102\x00\xc2\x07\x00\x00\x00\x00"

Show some code already

Thanks

Dan Wilson

twitter.com/DanWilson

nodans.com

datacurl.com

challengewave.com

model-glue.com