The CAP Theorem

Theorem: You can have at most two of these properties for any database system.

MongoDB Overview

+ Distributed database system
+ Open-source software (sponsored by MongoDB Inc.)
+ Designed for storing and processing web data
+ Document-oriented data model
+ "Schemaless" (schema-on-reads)
+ Rich query language
+ Secondary indexes
+ Horizontal scaling through replication and sharding
+ Runs on-premise and in cloud (Atlas offering)
+ Primary datastore for many web applications
+ Multi-document transaction support

- Sharding is not automatic
Replication in MongoDB

- High-availability
- Redundancy
- Automatic failovers
- Load balancing reads
Sharding in MongoDB

- **shard key** = one or more fields of a document which determine how documents get sliced
- Documents with the same shard key are assigned to the same chunk
- Chunks are assigned to a shard

<table>
<thead>
<tr>
<th>Key Range</th>
<th>Chunk</th>
<th>Shard</th>
</tr>
</thead>
<tbody>
<tr>
<td>0...20</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>21...40</td>
<td>2</td>
<td>B</td>
</tr>
<tr>
<td>41...60</td>
<td>3</td>
<td>A</td>
</tr>
<tr>
<td>61...80</td>
<td>4</td>
<td>B</td>
</tr>
<tr>
<td>81...100</td>
<td>5</td>
<td>A</td>
</tr>
</tbody>
</table>
Sharding + Replication

- Each shard is deployed as a replica set
- Scales both reads and writes
- Widely used in prod environments
Data Model

- MongoDB Document = BSON object
- Unordered key/value pairs with nesting
- Documents have unique identifiers (_id)
- Data types: String, Int, Double, Boolean, Date, Timestamp, Array, Object, ObjectId
- Documents are nested via Object type
- Max document size: 16 MB (including nested objects)
- Documents grouped into collections
- Collections grouped into databases
Writing to Mongo

db.coll.insertOne(document)
db.coll.insert([document1, document2, documentn])

> doc = {
"company name": "Google Inc.",
"exchange": "NASDAQ",
"symbol": "GOOG"
} {
"company name": "Google Inc.",
"exchange": "NASDAQ",
"symbol": "GOOG"
}
>
> db.market.insertOne(doc)
{
"acknowledged": true,
"insertedId": ObjectId("5f7e2215801f0b72e50f3fd8")
}
> _
Writing to Mongo

```javascript
> doc = {
    "company name": "Google Inc.",
    "exchange": "NASDAQ",
    "symbol": "GOOG",
    "summary": {
        "date": 20211022,
        "open": 2807.02,
        "high": 2831.17,
        "low": 2743.41
    }
}

> db.market.insert(doc)
WriteResult({ "nInserted" : 1 })
```
Writing to Mongo

```javascript
> doc = {
"company name": "Google Inc.",
"symbol": "GOOG",
"exchange": "NASDAQ",
"summary": [{
="date": 20201007,
"open": 1464.29,
"high": 1480.93,
"low": 1461.47},
{"date": 20201006,
"open": 1476.89,
"high": 1480.93,
"low": 1453.44}]

"company name": "Google Inc.",
"symbol": "GOOG",
"exchange": "NASDAQ",
"summary": []

"date": 20201007,
"open": 1464.29,
"high": 1480.93,
"low": 1461.47
},

"date": 20201006,
"open": 1476.89,
"high": 1480.93,
"low": 1453.44
}

> db.market.insert(doc)
WriteResult({ "nInserted" : 1 })
```
Reading from Mongo

db.coll.findOne(selection, projection)
db.coll.find(selection, projection)

> selection = {
  "company name": "Google Inc.",
  "symbol": "GOOG"
}
> projection = {
  "company name": 1,
  "exchange": 1,
  "symbol": 1,
  "_id": 0
}

> db.market.find(selection, projection).pretty()

```json
{
  "company name": "Google Inc.",
  "exchange": "NASDAQ",
  "symbol": "GOOG"
}
{
  "company name": "Google Inc.",
  "exchange": "NASDAQ",
  "symbol": "GOOG"
}
{
  "company name": "Google Inc.",
  "symbol": "GOOG",
  "exchange": "NASDAQ"
}
```
Reading from Mongo

```bash
> selection = {"summary.date": 20211022}
{ "summary.date" : 20211022 }
> projection = {"summary.date": 1, "summary.open": 1, "summary.high": 1, "_id":0}
{ "summary.date" : 1, "summary.open" : 1, "summary.high" : 1, "_id" : 0 }
>
> db.market.find(selection, projection)
{ "summary" : { "date" : 20211022, "open" : 2807.02, "high" : 2831.17 } }

> selection = {"summary.date": 20211022, "symbol": "GOOG"}
{ "summary.date" : 20211022, "symbol" : "GOOG" }
> projection = {"summary.date": 1, "summary.open": 1, "summary.high": 1, "_id":0}
{ "summary.date" : 1, "summary.open" : 1, "summary.high" : 1, "_id" : 0 }
>
> db.market.find(selection, projection).pretty()
{
    "summary" : {
        "date" : 20211022,
        "open" : 2807.02,
        "high" : 2831.17
    }
}
```
Reading from Mongo

```javascript
> selection = { "$or": [{ "summary.date": 20211022 }, { "summary.date": 20201007 }] }
{ "or": [
  { "summary.date": 20211022 },
  { "summary.date": 20201007 }
]
}
> projection = { "summary.date": 1, "summary.open": 1, "summary.high": 1, _id:0 }
{ "summary.date": 1, "summary.open": 1, "summary.high": 1, "_id": 0 }
>
> db.market.find(selection, projection)
{ "summary": { "date": 20211022, "open": 2807.02, "high": 2831.17 } }
{ "summary": [ { "date": 20201007, "open": 1464.29, "high": 1468.96 }, { "date": 20201006, "open": 1476.89, "high": 1480.93 } ]
```
Reading from Mongo

```javascript
> selection = { "summary.low": { "$gte": 1450, "$lte": 1455 } 
  { "summary.low" : { "$gte" : 1450, "$lte" : 1455 } } 
> projection = { "summary.date": 1, "summary.open": 1, "summary.high": 1, _id:0} 
  { "summary.date" : 1, "summary.open" : 1, "summary.high" : 1, "_id" : 0 } 

> db.market.find(selection, projection).pretty()

{  
   "summary" : [  
     {  
       "date" : 20201007,  
       "open" : 1464.29,  
       "high" : 1468.96  
     },  
     {  
       "date" : 20201006,  
       "open" : 1476.89,  
       "high" : 1480.93  
     }  
   ]
}

Range operators:
$lt
$gt
$lte
$gte
Updates in Mongo

db.coll.update(selection, update)
db.coll.updateMany(selection, update)

> doc = {"company name": "Alphabet, Inc."} { "company name": "Alphabet, Inc." }
> db.market.updateMany({}, {"$set": doc}) { "acknowledged": true, "matchedCount": 3, "modifiedCount": 3 }
>
> doc = {"summary": {"date": 20201008, "open": 1465.09, "high": 1485.45, "low": 1465.09}}
{ "summary": {
   "date": 20201008,
   "open": 1465.09,
   "high": 1485.45,
   "low": 1465.09
}
> db.market.update("_id": ObjectId("5f7f8a1ad400cb46a62c861a"), {"$addToSet": doc}) WriteResult({ "nMatched": 1, "nUpserted": 0, "nModified": 1 })
>
Deletes in Mongo

db.coll.remove(selection)

> doc = {"_id": ObjectId("5f7f8a1ad400cb46a62c861a")}
{ "_id" : ObjectId("5f7f8a1ad400cb46a62c861a") } 
> 
> db.market.remove(doc)
WriteResult({ "nRemoved" : 1 })
>

> doc = {"company name": "Alphabet, Inc."}
{ "company name" : "Alphabet, Inc." } 
> 
> db.market.remove(doc)
WriteResult({ "nRemoved" : 2 })
>
MongoDB code lab

- Clone [snippets](#) repo
- Open [mongodb notebook](#)
- Create database and import collections
- Practice CRUD operations
Practice Problem

Translate the following SQL query into MongoDB’s query language:

```sql
SELECT Title, Artist, Date, 'Height (cm)', 'Width (cm)
FROM Artworks
WHERE Nationality = 'Swedish'
AND Classification = 'Sculpture'
ORDER BY 'Height (cm)' DESC, 'Width (cm)' DESC
LIMIT 1;
```
Project 6