MongoDB 101 Workshop

Sig Narváez

mongoDB®

Who Am I?

Sigfrido “Sig” Narváez

Principal Solution Architect
MongoDB

@SigNarvaez

Disclaimer

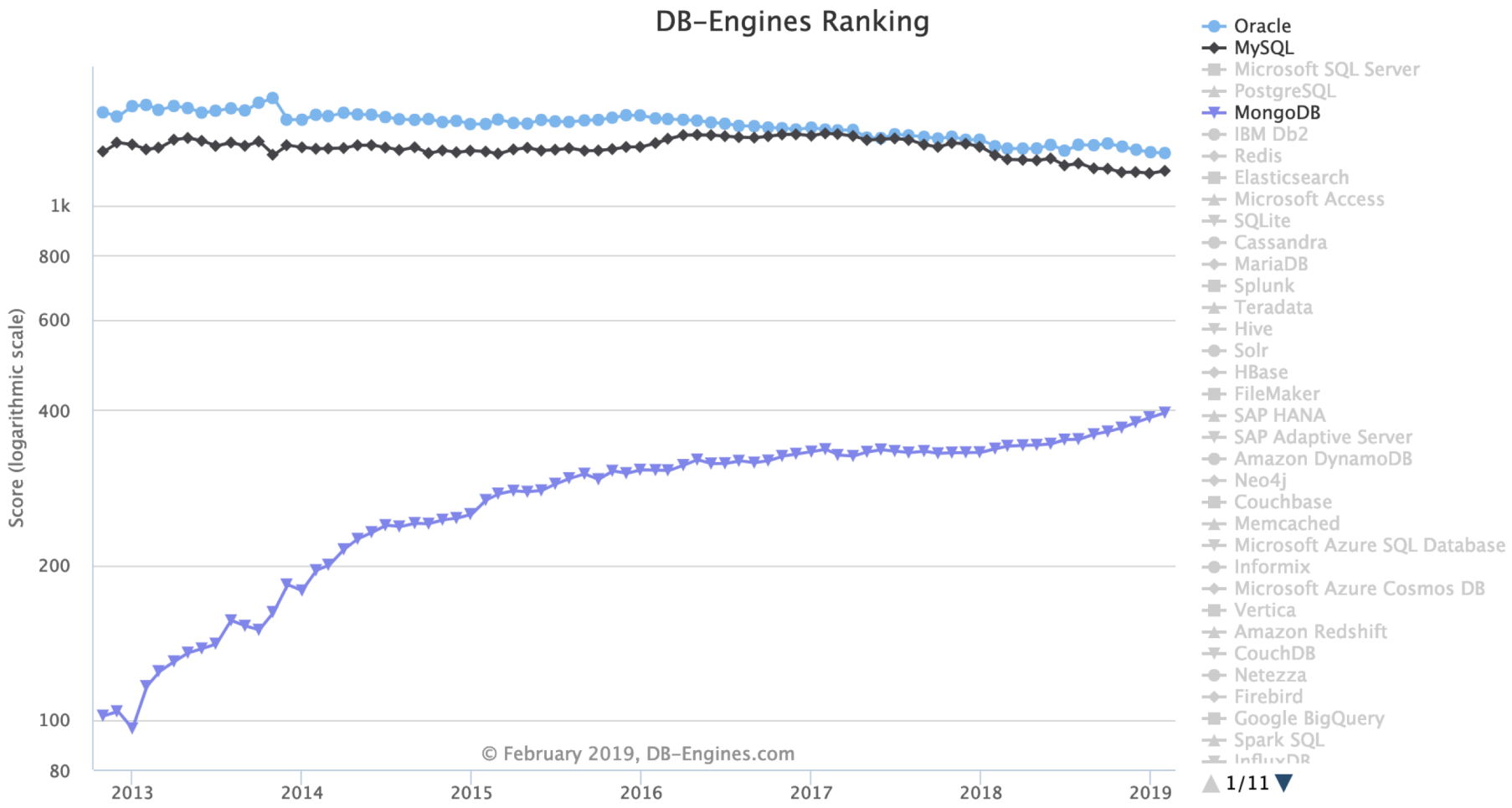
Why are you here?

mongoDB.

DB-Engines Ranking - Trend Popularity

The DB-Engines Ranking ranks database management systems according to their popularity.

Read more about the method of calculating the scores.



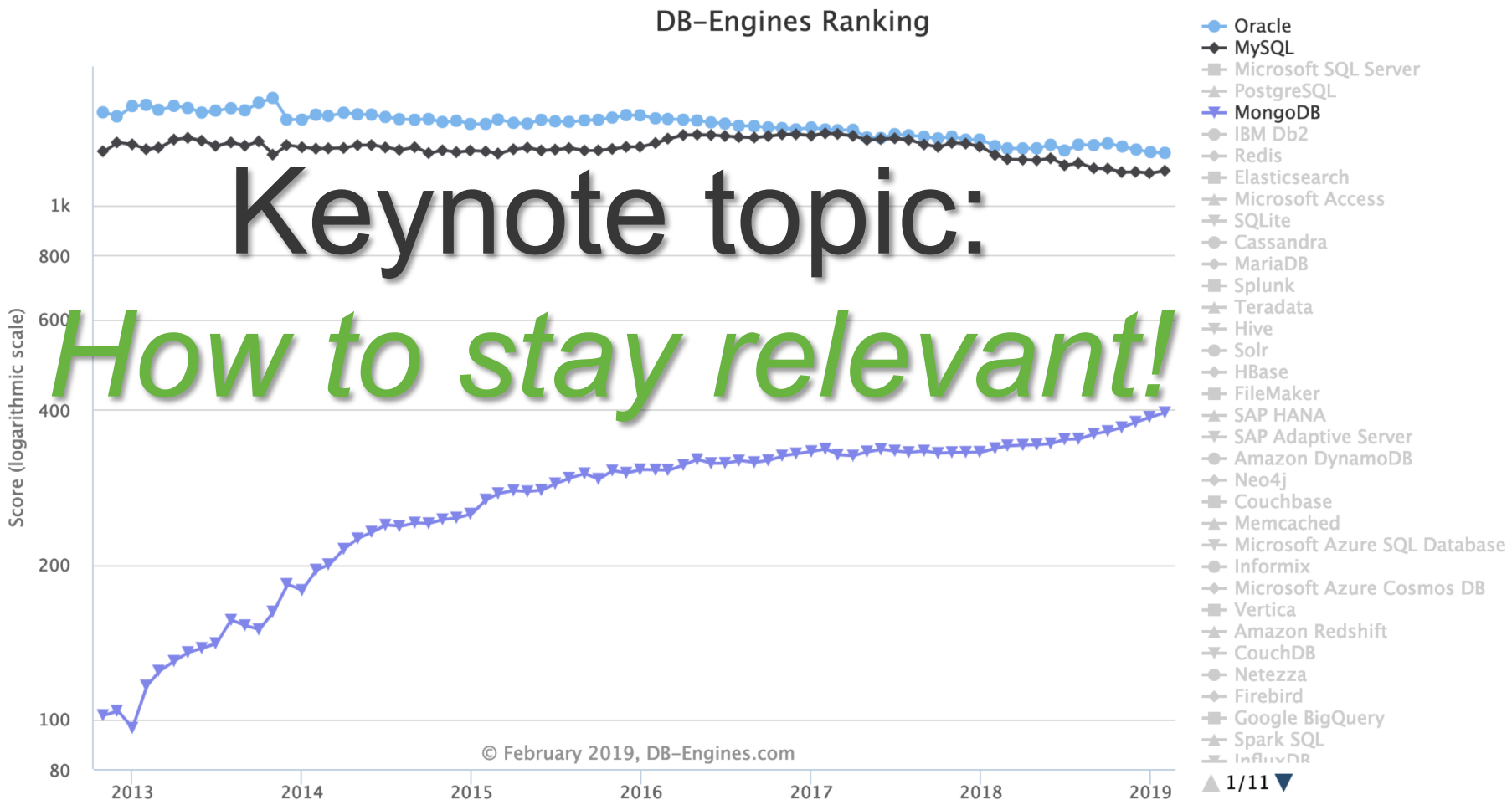
https://db-engines.com/en/ranking_trend

mongoDB.

DB-Engines Ranking - Trend Popularity

The DB-Engines Ranking ranks database management systems according to their popularity.

Read more about the method of calculating the scores.



Keynote topic:

How to stay relevant!

https://db-engines.com/en/ranking_trend

mongoDB.

MongoDB 101 Workshop

Introduction to MongoDB (30 mins)

Setting up MongoDB (30 mins)
- Laptop: Standalone Engine & Shell, Compass, MongoDB Driver (Python)
- Data Generator: MGenerateJS
- Cloud: MongoDB Atlas account

Querying, Indexing and Aggregation (30 mins)
- Using Compass, MongoDB Shell & Python

Schema Design & Use Cases (30 mins)
- Embedding & Referencing, 1-1, 1-M, M-M, Patterns
- Single View, IoT, Time Series, eCommerce/Content Management

MongoDB Atlas (60 mins)
- Management, Clusters, Security
- High Availability & Scalability (Replication & Sharding)
- Managed BI Connector & Charts
- ODBC for BI Connector
- MongoDB Stitch*

mongoDB.

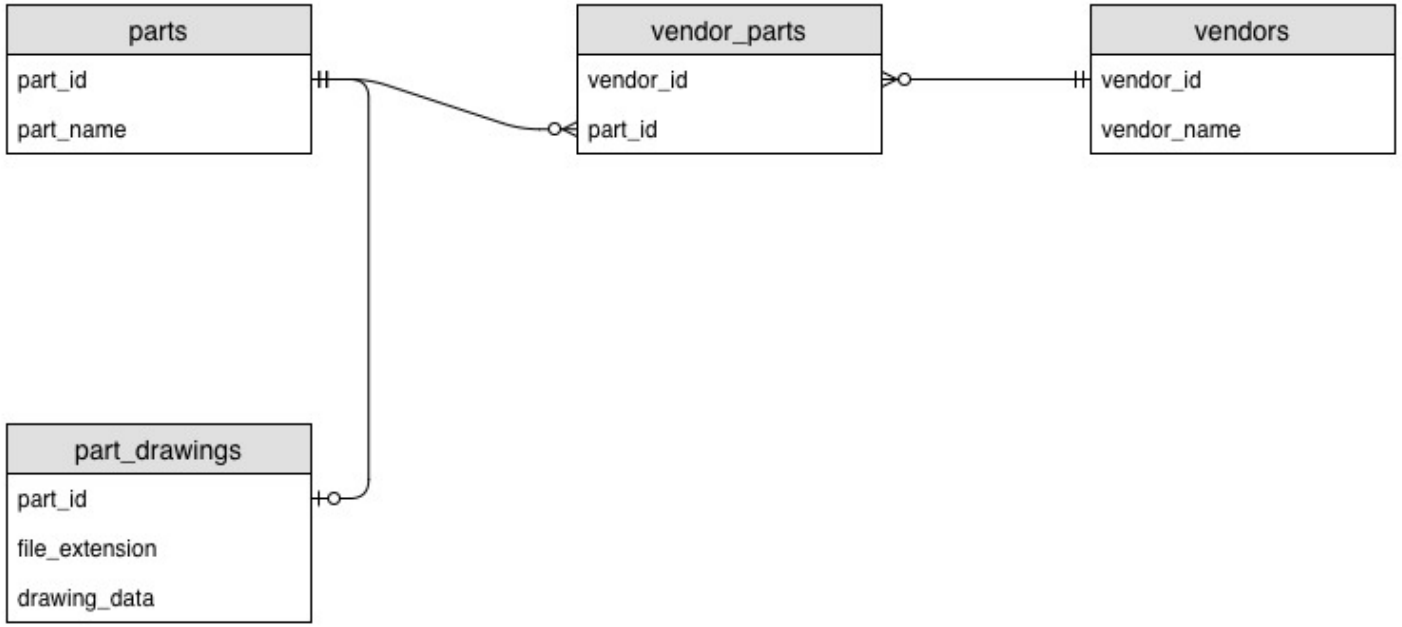
Introduction to MongoDB

9:00 – 9:30

Introduction to MongoDB

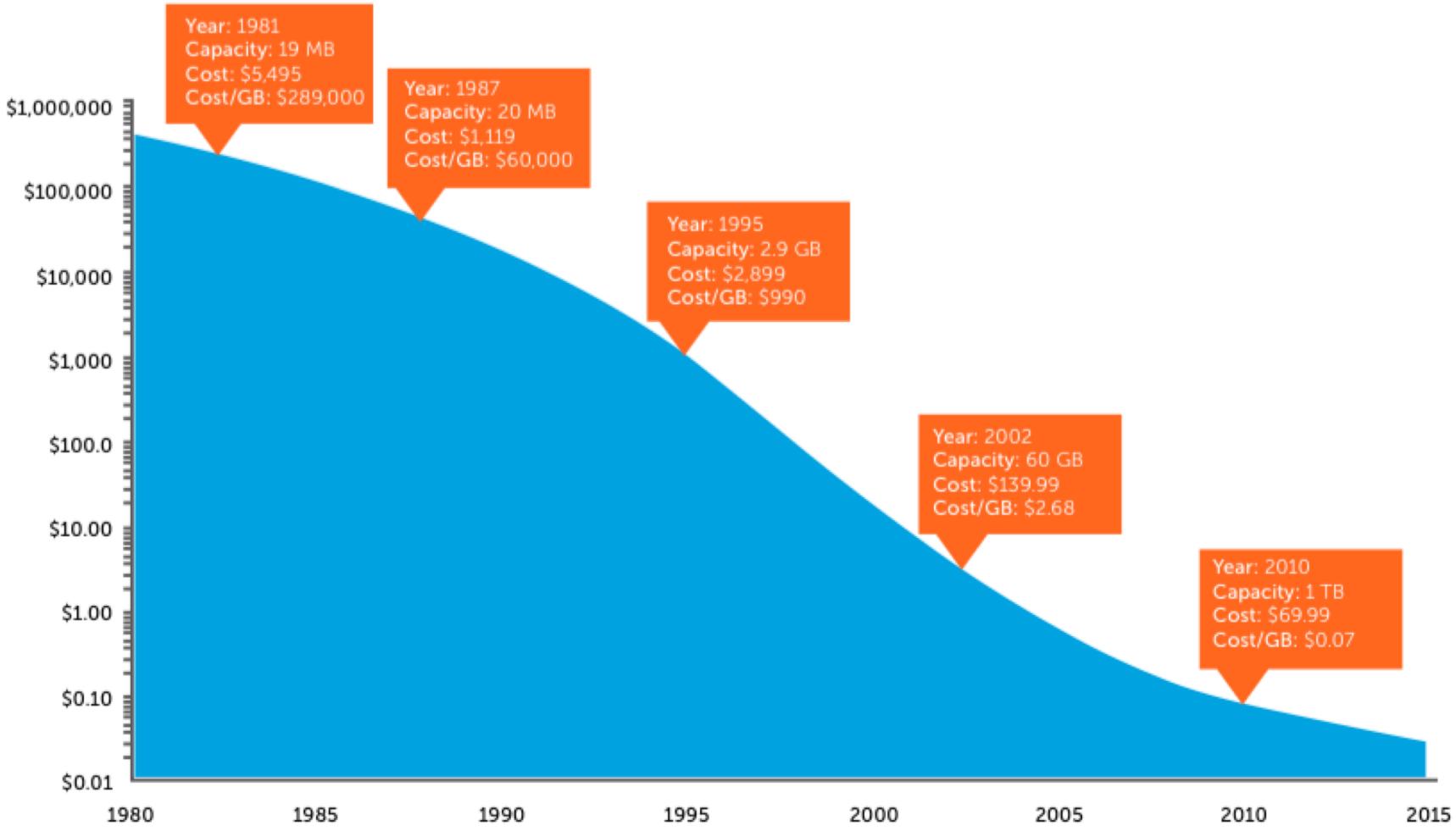
V1.0

Relational Database Design

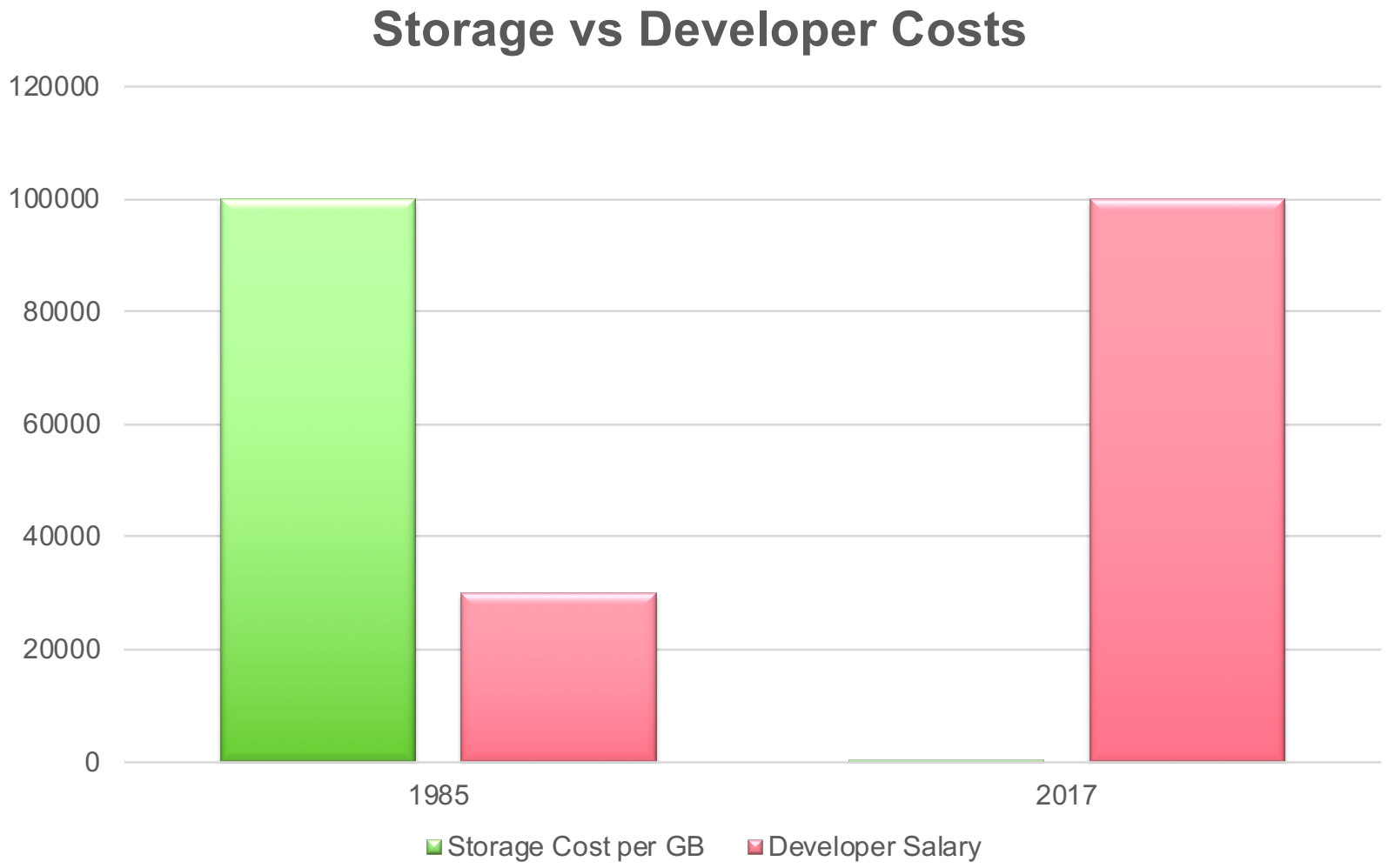


Why Did We Normalize Data?

Storage cost over time



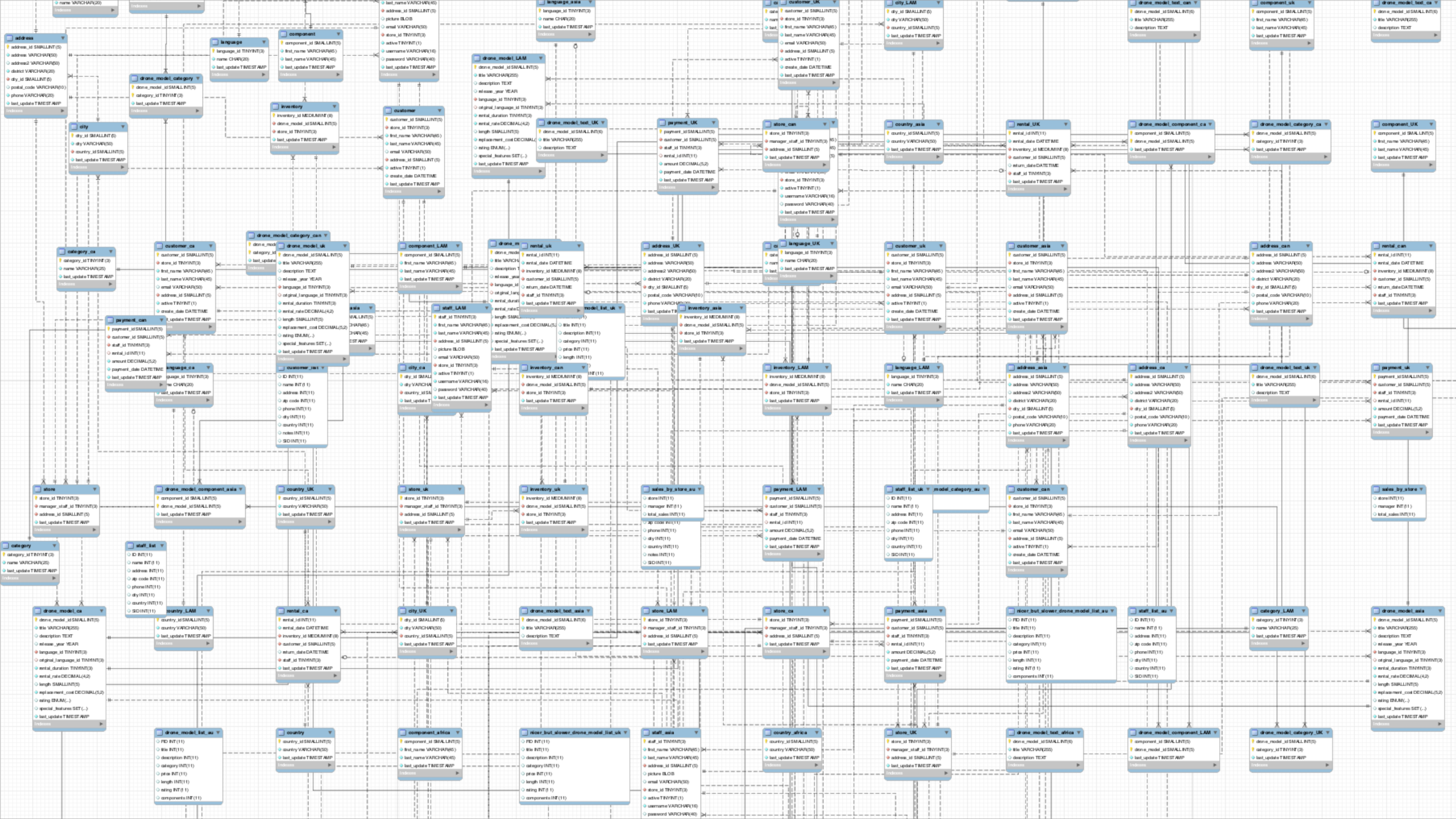
Costs have Shifted! – **Developers are the new kingmakers!**



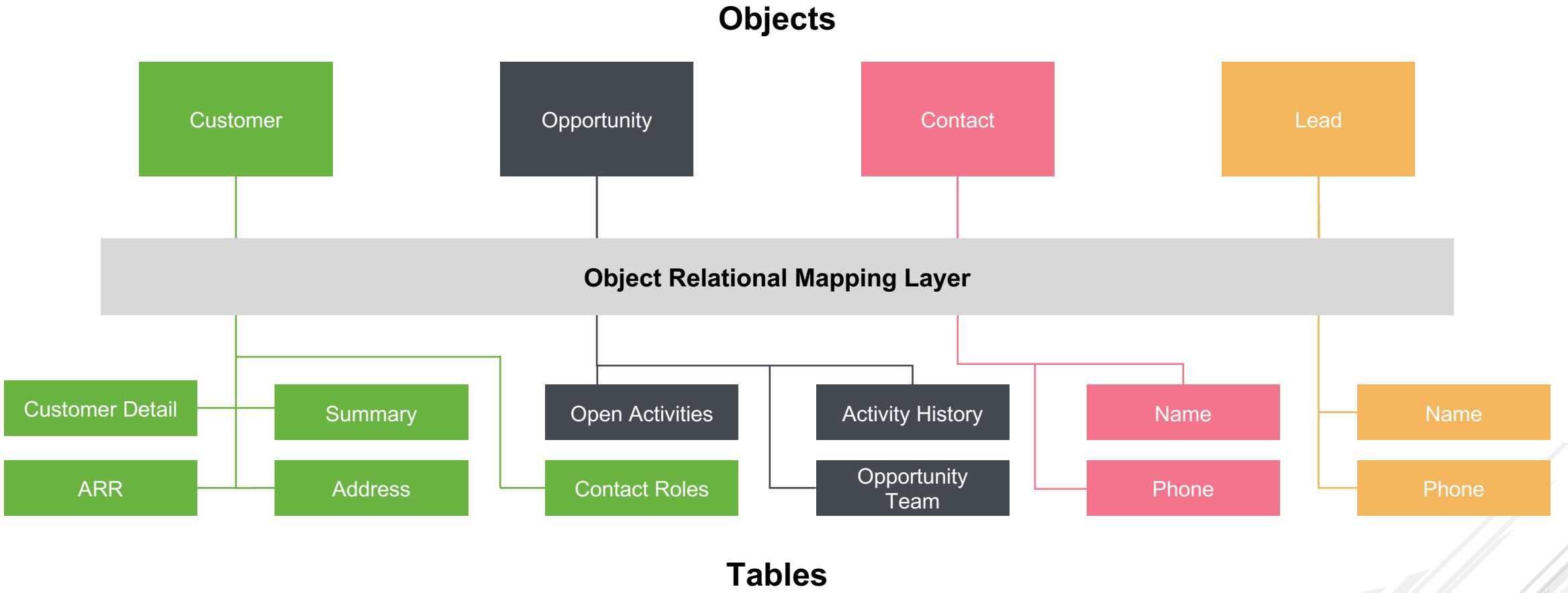
mongoDB.

Tabular/Relational vs. Object Oriented

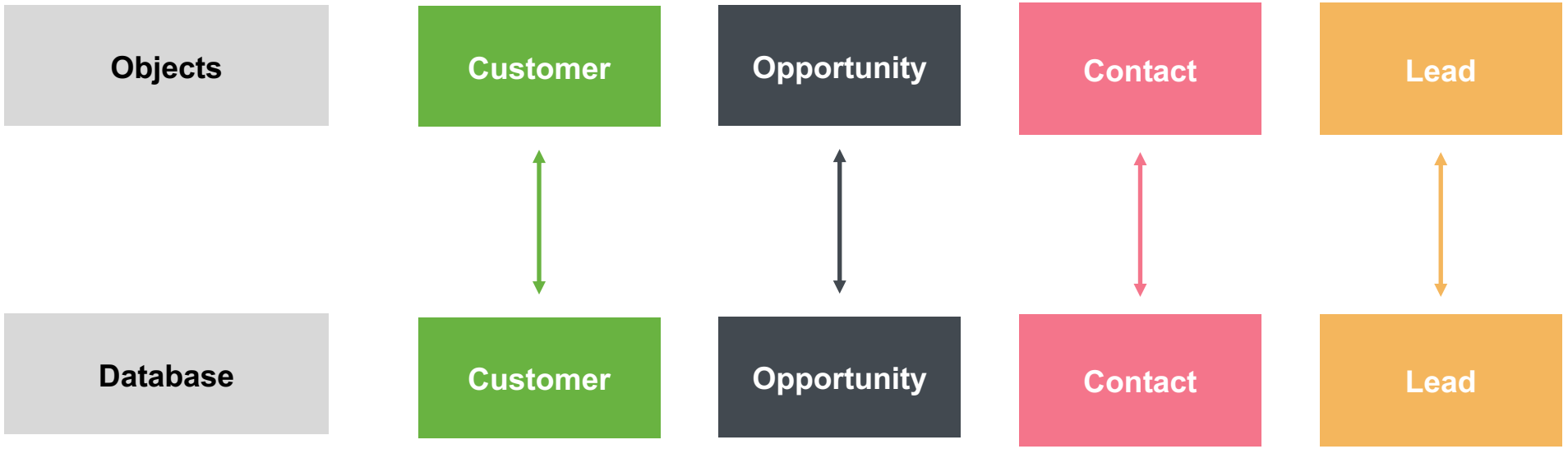
mongoDB.


address
address_id SMALLINT(5)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(5)
postal_code VARCHAR(10)
phone VARCHAR(20)
last_update TIMESTAMP
Indexes
city
city_id SMALLINT(5)
city VARCHAR(50)
country_id SMALLINT(5)
last_update TIMESTAMP
Indexes
language
language_id TINYINT(3)
name CHAR(20)
last_update TIMESTAMP
Indexes
component
component_id SMALLINT(5)
first_name VARCHAR(45)
last_name VARCHAR(45)
last_update TIMESTAMP
Indexes
drone_model_category
drone_model_id SMALLINT(5)
category_id TINYINT(3)
last_update TIMESTAMP
Indexes
inventory
inventory_id MEDIUMINT(8)
drone_model_id SMALLINT(5)
store_id TINYINT(3)
last_update TIMESTAMP
Indexes
customer
customer_id SMALLINT(5)
store_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(50)
address_id SMALLINT(5)
active TINYINT(1)
create_date DATETIME
last_update TIMESTAMP
Indexes
drone_model_LAM
drone_model_id SMALLINT(5)
title VARCHAR(255)
description TEXT
release_year YEAR
language_id TINYINT(3)
original_language_id TINYINT(3)
rental_duration TINYINT(3)
rental_rate DECIMAL(4,2)
length SMALLINT(5)
replacement_cost DECIMAL(5,2)
rating ENUM(...)
special_features SET(...)
last_update TIMESTAMP
Indexes
drone_model_text_UK
drone_model_id SMALLINT(6)
title VARCHAR(255)
description TEXT
Indexes
payment_UK
payment_id SMALLINT(5)
customer_id SMALLINT(5)
staff_id TINYINT(3)
rental_id INT(11)
amount DECIMAL(5,2)
payment_date DATETIME
last_update TIMESTAMP
Indexes
store_can
store_id TINYINT(3)
manager_staff_id TINYINT(3)
address_id SMALLINT(5)
last_update TIMESTAMP
Indexes
store_id TINYINT(3)
store_id TINYINT(3)
active TINYINT(1)
username VARCHAR(16)
password VARCHAR(40)
last_update TIMESTAMP
Indexes
country_asia
country_id SMALLINT(5)
country VARCHAR(50)
last_update TIMESTAMP
Indexes
rental_UK
rental_id INT(11)
rental_date DATETIME
inventory_id MEDIUMINT(8)
customer_id SMALLINT(5)
return_date DATETIME
staff_id TINYINT(3)
last_update TIMESTAMP
Indexes
drone_model_text_can
drone_model_id SMALLINT(6)
title VARCHAR(255)
description TEXT
Indexes
drone_model_component_ca
component_id SMALLINT(5)
drone_model_id SMALLINT(5)
last_update TIMESTAMP
Indexes
drone_model_category_ca
drone_model_id SMALLINT(5)
category_id TINYINT(3)
last_update TIMESTAMP
Indexes
component_UK
component_id SMALLINT(5)
first_name VARCHAR(45)
last_name VARCHAR(45)
last_update TIMESTAMP
Indexes
drone_model_text_ca
drone_model_id SMALLINT(6)
title VARCHAR(255)
description TEXT
Indexes
category_ca
category_id TINYINT(3)
name VARCHAR(25)
last_update TIMESTAMP
Indexes
customer_ca
customer_id SMALLINT(5)
store_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(50)
address_id SMALLINT(5)
active TINYINT(1)
create_date DATETIME
last_update TIMESTAMP
Indexes
drone_model_category_can
drone_model_UK
drone_model_id SMALLINT(5)
title VARCHAR(255)
description TEXT
release_year YEAR
language_id TINYINT(3)
original_language_id TINYINT(3)
rental_duration TINYINT(3)
rental_rate DECIMAL(4,2)
length SMALLINT(5)
replacement_cost DECIMAL(5,2)
rating ENUM(...)
special_features SET(...)
last_update TIMESTAMP
Indexes
component_LAM
component_id SMALLINT(5)
first_name VARCHAR(45)
last_name VARCHAR(45)
last_update TIMESTAMP
Indexes
rental_uk
rental_id INT(11)
rental_date DATETIME
inventory_id MEDIUMINT(8)
customer_id SMALLINT(5)
return_date DATETIME
staff_id TINYINT(3)
last_update TIMESTAMP
Indexes
address_UK
address_id SMALLINT(5)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(5)
postal_code VARCHAR(10)
phone VARCHAR(20)
last_update TIMESTAMP
Indexes
language_UK
language_id TINYINT(3)
name CHAR(20)
last_update TIMESTAMP
Indexes
customer_uk
customer_id SMALLINT(5)
store_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(50)
address_id SMALLINT(5)
active TINYINT(1)
create_date DATETIME
last_update TIMESTAMP
Indexes
customer_asia
customer_id SMALLINT(5)
store_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(50)
address_id SMALLINT(5)
active TINYINT(1)
create_date DATETIME
last_update TIMESTAMP
Indexes
address_can
address_id SMALLINT(5)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(5)
postal_code VARCHAR(10)
phone VARCHAR(20)
last_update TIMESTAMP
Indexes
rental_can
rental_id INT(11)
rental_date DATETIME
inventory_id MEDIUMINT(8)
customer_id SMALLINT(5)
return_date DATETIME
staff_id TINYINT(3)
last_update TIMESTAMP
Indexes
payment_can
payment_id SMALLINT(5)
customer_id SMALLINT(5)
staff_id TINYINT(3)
rental_id INT(11)
amount DECIMAL(5,2)
payment_date DATETIME
last_update TIMESTAMP
Indexes
language_ca
language_id TINYINT(3)
name CHAR(20)
last_update TIMESTAMP
Indexes
customer_ast
ID INT(11)
name INT(1 1)
address INT(11)
zip code INT(11)
phone INT(11)
city INT(11)
country INT(11)
notes INT(11)
SID INT(11)
staff_LAM
staff_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
address_id SMALLINT(5)
picture BLOB
email VARCHAR(50)
store_id TINYINT(3)
active TINYINT(1)
username VARCHAR(16)
password VARCHAR(40)
last_update TIMESTAMP
Indexes
drone_model_list_uk
title INT(11)
description INT(11)
category INT(11)
price INT(11)
length INT(11)
inventory_asia
inventory_id MEDIUMINT(8)
drone_model_id SMALLINT(5)
store_id TINYINT(3)
last_update TIMESTAMP
Indexes
city_ca
city_id SMALLINT(5)
city VARCHAR(50)
country_id SMALLINT(5)
last_update TIMESTAMP
Indexes
inventory_can
inventory_id MEDIUMINT(8)
drone_model_id SMALLINT(5)
store_id TINYINT(3)
last_update TIMESTAMP
Indexes
inventory_LAM
inventory_id MEDIUMINT(8)
drone_model_id SMALLINT(5)
store_id TINYINT(3)
last_update TIMESTAMP
Indexes
language_LAM
language_id TINYINT(3)
name CHAR(20)
last_update TIMESTAMP
Indexes
address_asia
address_id SMALLINT(5)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(5)
postal_code VARCHAR(10)
phone VARCHAR(20)
last_update TIMESTAMP
Indexes
address_ca
address_id SMALLINT(5)
address VARCHAR(50)
address2 VARCHAR(50)
district VARCHAR(20)
city_id SMALLINT(5)
postal_code VARCHAR(10)
phone VARCHAR(20)
last_update TIMESTAMP
Indexes
drone_model_text_uk
drone_model_id SMALLINT(6)
title VARCHAR(255)
description TEXT
Indexes
payment_uk
payment_id SMALLINT(5)
customer_id SMALLINT(5)
staff_id TINYINT(3)
rental_id INT(11)
amount DECIMAL(5,2)
payment_date DATETIME
last_update TIMESTAMP
Indexes
store
store_id TINYINT(3)
manager_staff_id TINYINT(3)
address_id SMALLINT(5)
last_update TIMESTAMP
Indexes
drone_model_component_asia
component_id SMALLINT(5)
drone_model_id SMALLINT(5)
last_update TIMESTAMP
Indexes
country_UK
country_id SMALLINT(5)
country VARCHAR(50)
last_update TIMESTAMP
Indexes
store_uk
store_id TINYINT(3)
manager_staff_id TINYINT(3)
address_id SMALLINT(5)
last_update TIMESTAMP
Indexes
inventory_uk
inventory_id MEDIUMINT(8)
drone_model_id SMALLINT(5)
store_id TINYINT(3)
last_update TIMESTAMP
Indexes
sales_by_store_au
store INT(11)
manager INT(11)
total_sales INT(11)
zip code INT(11)
phone INT(11)
city INT(11)
country INT(11)
notes INT(11)
SID INT(11)
payment_LAM
payment_id SMALLINT(5)
customer_id SMALLINT(5)
staff_id TINYINT(3)
rental_id INT(11)
amount DECIMAL(5,2)
payment_date DATETIME
last_update TIMESTAMP
Indexes
staff_list_uk
ID INT(11)
name INT(1 1)
address INT(11)
zip code INT(11)
phone INT(11)
city INT(11)
country INT(11)
SID INT(11)
model_category_au
customer_can
store_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
email VARCHAR(50)
address_id SMALLINT(5)
active TINYINT(1)
create_date DATETIME
last_update TIMESTAMP
Indexes
sales_by_store
store INT(11)
manager INT(11)
total_sales INT(11)
category
category_id TINYINT(3)
name VARCHAR(25)
last_update TIMESTAMP
Indexes
staff_list
ID INT(11)
name INT(1 1)
address INT(11)
zip code INT(11)
phone INT(11)
city INT(11)
country INT(11)
SID INT(11)
drone_model_ca
drone_model_id SMALLINT(5)
title VARCHAR(255)
description TEXT
release_year YEAR
language_id TINYINT(3)
original_language_id TINYINT(3)
rental_duration TINYINT(3)
rental_rate DECIMAL(4,2)
length SMALLINT(5)
replacement_cost DECIMAL(5,2)
rating ENUM(...)
special_features SET(...)
last_update TIMESTAMP
Indexes
country_LAM
country_id SMALLINT(5)
country VARCHAR(50)
last_update TIMESTAMP
Indexes
rental_ca
rental_id INT(11)
rental_date DATETIME
inventory_id MEDIUMINT(8)
customer_id SMALLINT(5)
return_date DATETIME
staff_id TINYINT(3)
last_update TIMESTAMP
Indexes
city_UK
city_id SMALLINT(5)
city VARCHAR(50)
country_id SMALLINT(5)
last_update TIMESTAMP
Indexes
drone_model_text_asia
drone_model_id SMALLINT(6)
title VARCHAR(255)
description TEXT
Indexes
store_LAM
store_id TINYINT(3)
manager_staff_id TINYINT(3)
address_id SMALLINT(5)
last_update TIMESTAMP
Indexes
store_ca
store_id TINYINT(3)
manager_staff_id TINYINT(3)
address_id SMALLINT(5)
last_update TIMESTAMP
Indexes
payment_asia
payment_id SMALLINT(5)
customer_id SMALLINT(5)
staff_id TINYINT(3)
rental_id INT(11)
amount DECIMAL(5,2)
payment_date DATETIME
last_update TIMESTAMP
Indexes
nicer_but_slower_drone_model_list_au
PID INT(11)
title INT(11)
description INT(11)
category INT(11)
price INT(11)
length INT(11)
rating INT(1 1)
components INT(11)
staff_list_au
ID INT(11)
name INT(1 1)
address INT(11)
zip code INT(11)
phone INT(11)
city INT(11)
country INT(11)
SID INT(11)
category_LAM
category_id TINYINT(3)
name VARCHAR(25)
last_update TIMESTAMP
Indexes
drone_model_asia
drone_model_id SMALLINT(5)
title VARCHAR(255)
description TEXT
release_year YEAR
language_id TINYINT(3)
original_language_id TINYINT(3)
rental_duration TINYINT(3)
rental_rate DECIMAL(4,2)
length SMALLINT(5)
replacement_cost DECIMAL(5,2)
rating ENUM(...)
special_features SET(...)
last_update TIMESTAMP
Indexes
drone_model_list_au
PID INT(11)
title INT(11)
description INT(11)
category INT(11)
price INT(11)
length INT(11)
rating INT(1 1)
components INT(11)
country
country_id SMALLINT(5)
country VARCHAR(50)
last_update TIMESTAMP
Indexes
component_africa
component_id SMALLINT(5)
first_name VARCHAR(45)
last_name VARCHAR(45)
last_update TIMESTAMP
Indexes
nicer_but_slower_drone_model_list_uk
PID INT(11)
title INT(11)
description INT(11)
category INT(11)
price INT(11)
length INT(11)
rating INT(1 1)
components INT(11)
staff_asia
staff_id TINYINT(3)
first_name VARCHAR(45)
last_name VARCHAR(45)
address_id SMALLINT(5)
picture BLOB
email VARCHAR(50)
store_id TINYINT(3)
active TINYINT(1)
username VARCHAR(16)
password VARCHAR(40)
country_africa
country_id SMALLINT(5)
country VARCHAR(50)
last_update TIMESTAMP
Indexes
store_UK
store_id TINYINT(3)
manager_staff_id TINYINT(3)
address_id SMALLINT(5)
last_update TIMESTAMP
Indexes
drone_model_text_africa
drone_model_id SMALLINT(6)
title VARCHAR(255)
description TEXT
Indexes
drone_model_component_LAM
component_id SMALLINT(5)
drone_model_id SMALLINT(5)
last_update TIMESTAMP
Indexes
drone_model_category_UK
drone_model_id SMALLINT(5)
category_id TINYINT(3)
last_update TIMESTAMP
Indexes

Go from this….


Objects
Customer
Opportunity
Contact
Lead
Object Relational Mapping Layer
Customer Detail
Summary
Open Activities
Activity History
Name
Name
ARR
Address
Contact Roles
Opportunity Team
Phone
Phone
Tables

To this: store objects directly…



Our founders believe that coding should be natural, and so should using a database.

They want the experience of using MongoDB to be simple and natural. Thus, the leaf.

Why MongoDB?

Intelligent Operational Data Platform

Best way to work with data

Intelligently put data where you need it

Freedom to run anywhere

mongoDB.

The evolution of MongoDB

3.0	3.2	3.4	3.6	4.0
Doc-Level Concurrency Compression Storage Engine API ≤50 replicas Auditing ++ Ops Manager	Document Validation $lookup Fast Failover Simpler Scalability Aggregation ++ Encryption At Rest In-Memory Storage Engine BI Connector MongoDB Compass APM Integration Profiler Visualization Auto Index Builds Backups to File System	Linearizable reads Intra-cluster compression Views Log Redaction Graph Processing Decimal Collations Faceted Navigation Zones ++ Aggregation ++ Auto-balancing ++ ARM, Power, zSeries BI & Spark Connectors ++ Compass ++ Hardware Monitoring Server Pool LDAP Authorization Encrypted Backups Cloud Foundry Integration	Change Streams Retryable Writes Expressive Array Updates Query Expressivity Causal Consistency Consistent Sharded Sec. Reads Compass Community Ops Manager ++ Query Advisor Schema Validation End to End Compression IP Whitelisting Default Bind to Localhost Sessions WiredTiger 1m+ Collections MongoDB BI Connector ++ Expressive $lookUp R Driver Atlas Cross Region Replication Atlas Auto Storage Scaling	Multi-Document ACID Transactions Atlas Global Clusters Atlas HIPAA Atlas LDAP Atlas Audit Atlas Encrypted Storage Engine Atlas AWS Backup Snapshots Atlas Full CRUD Agg Pipeline Type Conversions 40% Faster Shard Migrations Snapshot Reads Non-Blocking Secondary Reads SHA-2 TLS 1.1+ Compass Agg Pipeline Builder Compass Export to Code Charts Beta Free Monitoring Cloud Service Ops Manager K8s & OpenShift MongoDB Stitch GA MongoDB Mobile GA

mongoDB

MongoDB Use Cases

Single View

- Real-time views of your business that integrate all of your siloed data.
- Flexible Data Model

Internet of Things

- Variety of data – multiple versions and schemas over time – older & newer models
- Horizontal Scale

Mobile

- JSON Payloads – single fetch
- Sync from Data Center to Device

Real Time Analytics

- Analyze on the spot – no ETL / DW
- MQL, SQL or ML/Data Science

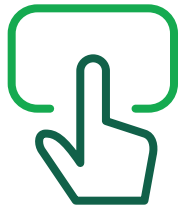
Personalization

Catalog

Content Management

https://www.mongodb.com/use-cases

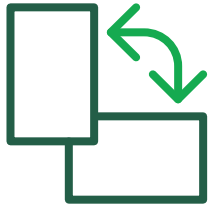
Best Way to Work With Data



Easy:
Work with data in a natural, intuitive way

Flexible:
Adapt and make changes quickly

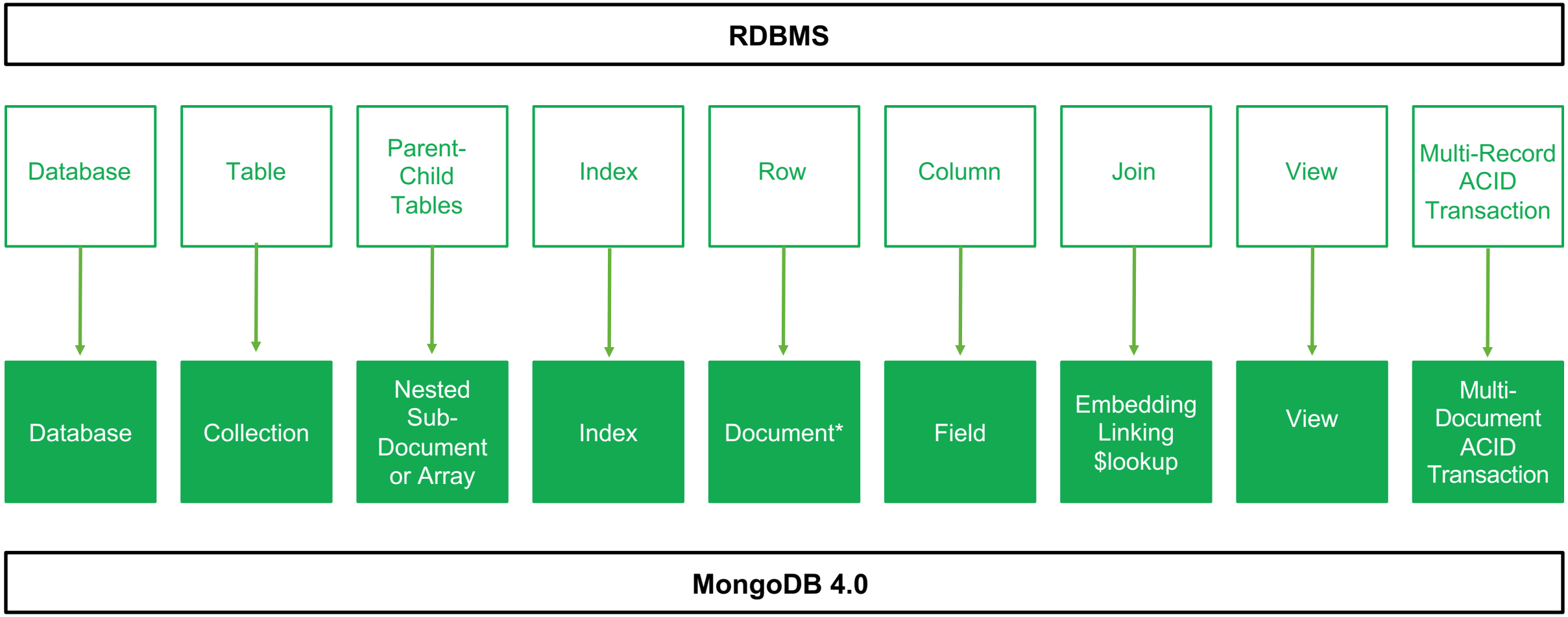
Fast:
Get great performance with less code



Versatile:
Supports a wide variety of data models and queries

The Document Model

Terminology



* Proper document schema design yields more entity data per document than found in a relational database row

mongoDB.

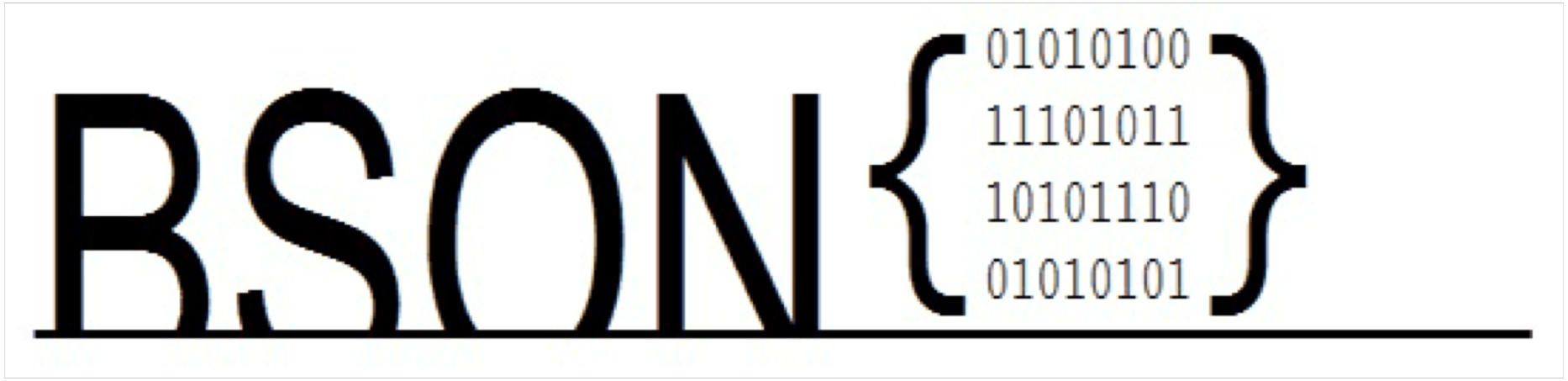
JSON is Widely Used

JSON is fast, lean, flexible

JSON is supported by almost all languages and tools

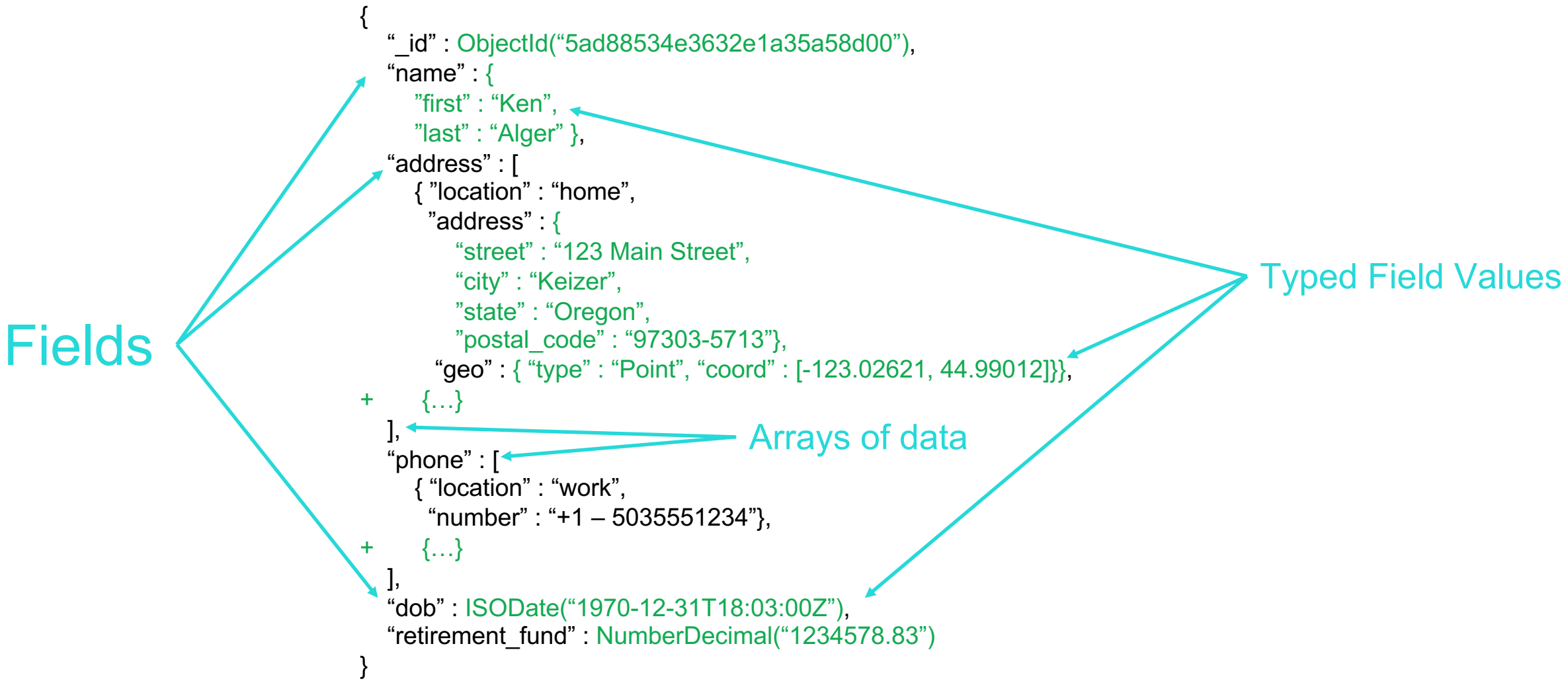
JSON is used for modern communication and data exchange

Rich Data Similar to JSON



Binary serialization of JSON data representation.
Easier to transfer than text files.
Lightweight, transversable, efficient.

Document Data Model



mongoDB.

Document Data Model

- Naturally maps objects to code – developers think in terms of objects, why shouldn't your data be modeled that way?
- Represent data of any structure
- Strongly typed for ease of processing – Over 20 binary encoded JSON data types
- Access by idiomatic drivers in all major programming languages

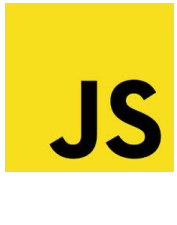
Drivers & Frameworks

Drivers







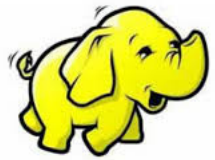





(Beta)

Frameworks




mongoDB.

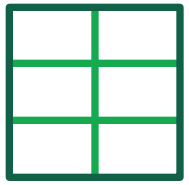
I’ve heard MongoDB is NoSQL – Does that mean I can’t query?

MongoDB Query Language

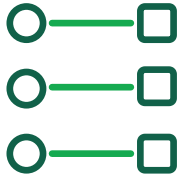
mongoDB.

Multiple data models, rich query functionality

JSON Documents



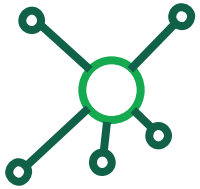
Tabular



Key-Value

Text

Geospatial



Graph

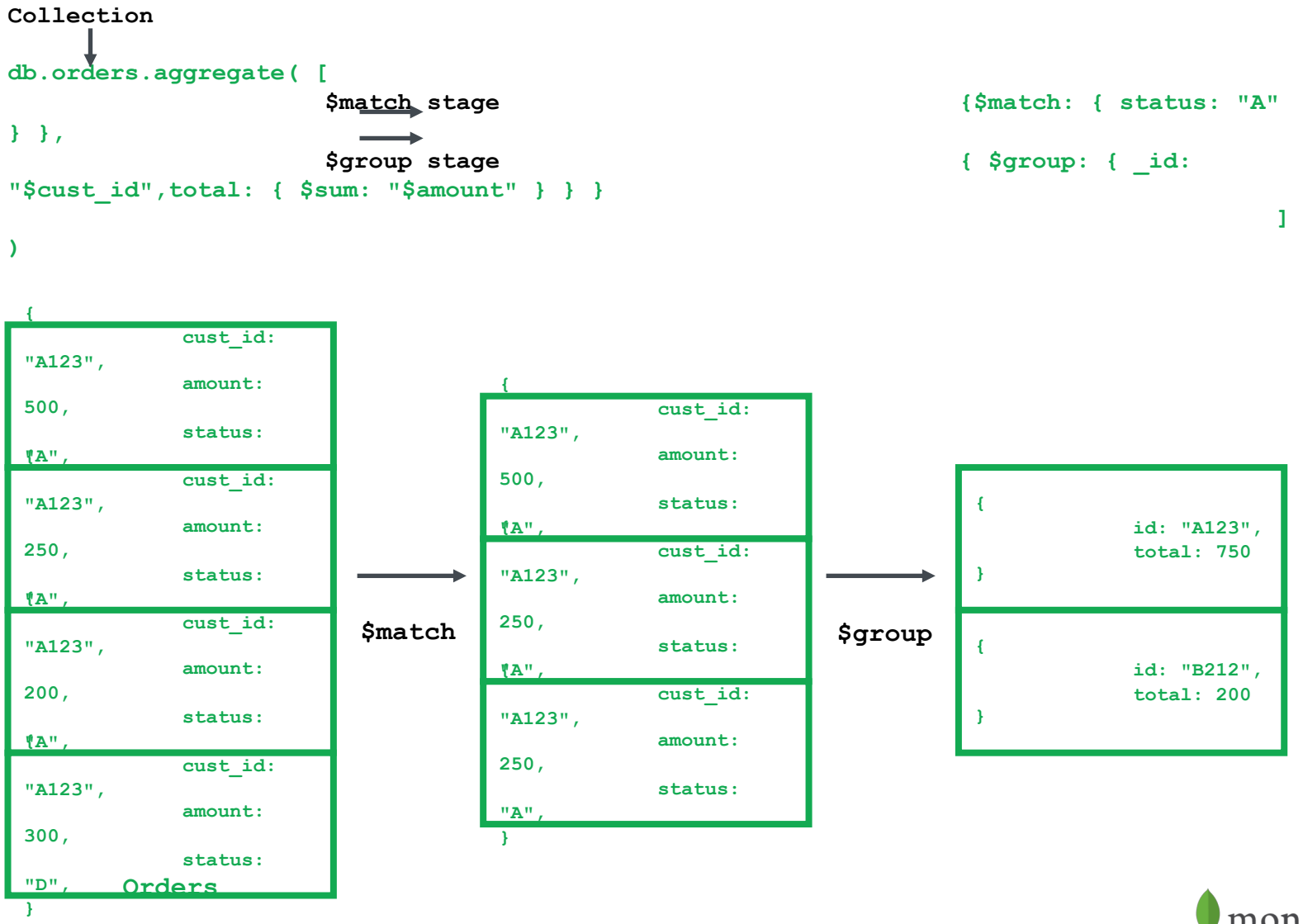
Rich Queries

Point | Range | Geospatial | Faceted Search | Aggregations | JOINs | Graph Traversals

Aggregation Framework

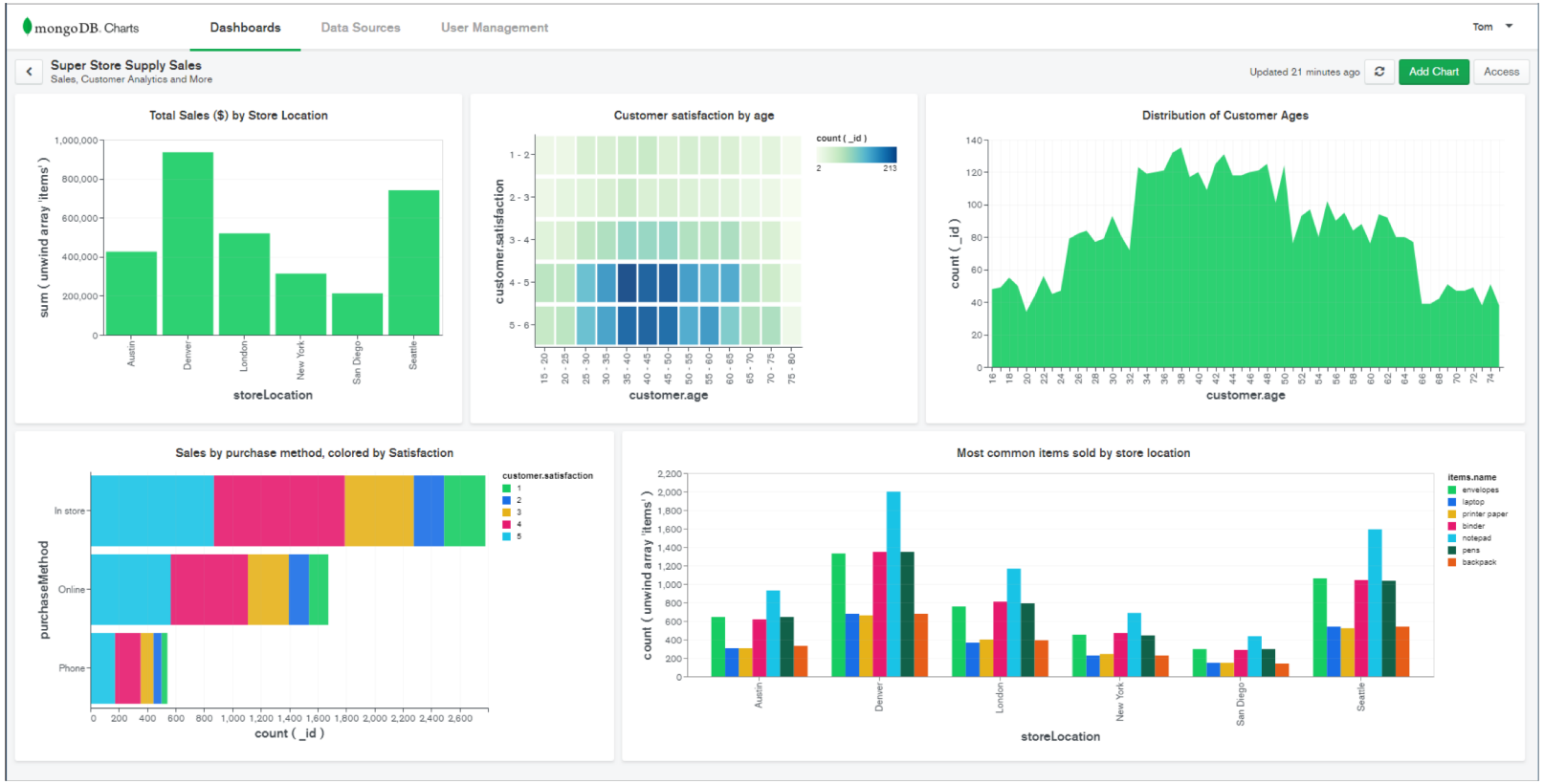
Advanced data processing pipeline for transformations and analytics

- Multiple stages
- Similar to a unix pipe
  - Build complex pipeline by chaining commands together
- Rich Expressions



mongoDB.

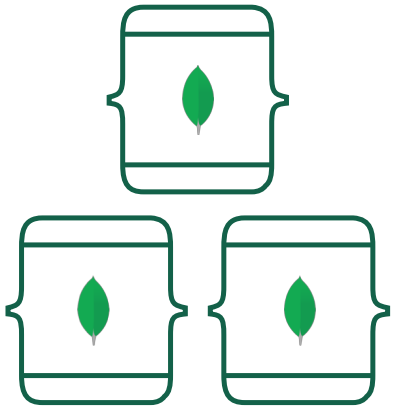
Sophisticated Analytics & Visualizations Of Data In Place



- Rich MongoDB query language & idiomatic drivers
- Connector for BI
- Connector for Spark
- Charts (beta)

mongoDB.

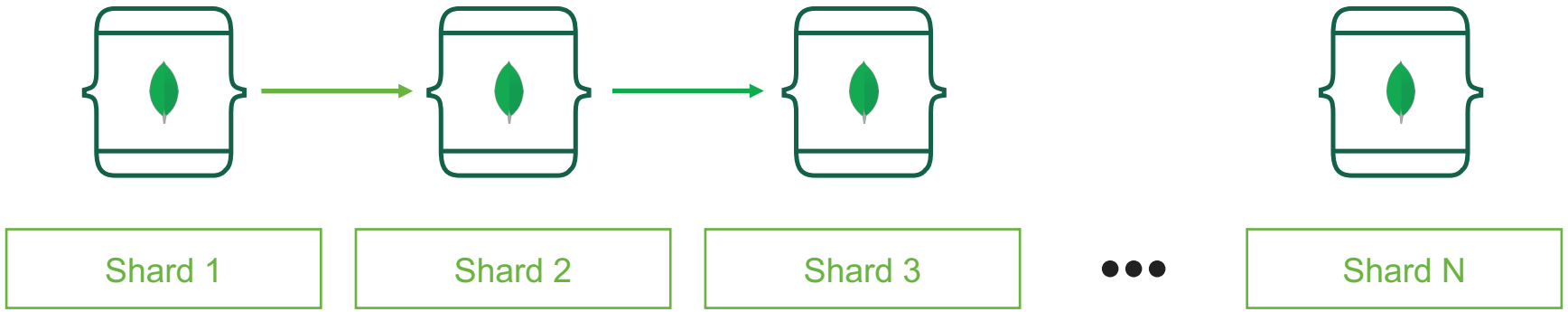
Data Availability



Replica Sets

- Up to 50 replicas, distributed across racks, data centers, and regions
- Self-healing
- Tunable durability and consistency controls
- Always-on write availability with retryable writes

Data Scalability



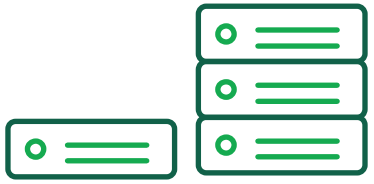
Sharding

- Automatically scale beyond the constraints of a single node
- Application transparent
- Scale and rebalance incrementally, in real time
- Unlike NoSQL systems that randomly spray data across a cluster, MongoDB exposes multiple data distribution policies to optimize for query patterns and locality

Run Anywhere

Local

On-premises
Server & Mainframe



Private cloud

Hybrid cloud

Public cloud

Fully managed cloud service

- Database that runs the same everywhere
- Leverage the benefits of a multi-cloud strategy
- Global coverage
- Avoid lock-in

Convenience: same codebase, same APIs, same tools, wherever you run

mongoDB.

MongoDB Server

- The fastest, easiest way to get started with MongoDB.
- https://www.mongodb.com/cloud/atlas

Downloadable solutions are available too, but must be self managed.

- Community Edition
- Enterprise Edition

mongoDB® Atlas: The only *true* multi-cloud database as a service

Database that runs the same everywhere

Consistent experience across AWS, Azure, and GCP

Coverage in any geography

Deploy in over 50 regions worldwide

Create globally distributed databases with a few clicks

Leverage the benefits of a multi-cloud strategy

Exploit the benefits of AWS, Azure, or GCP services on your data

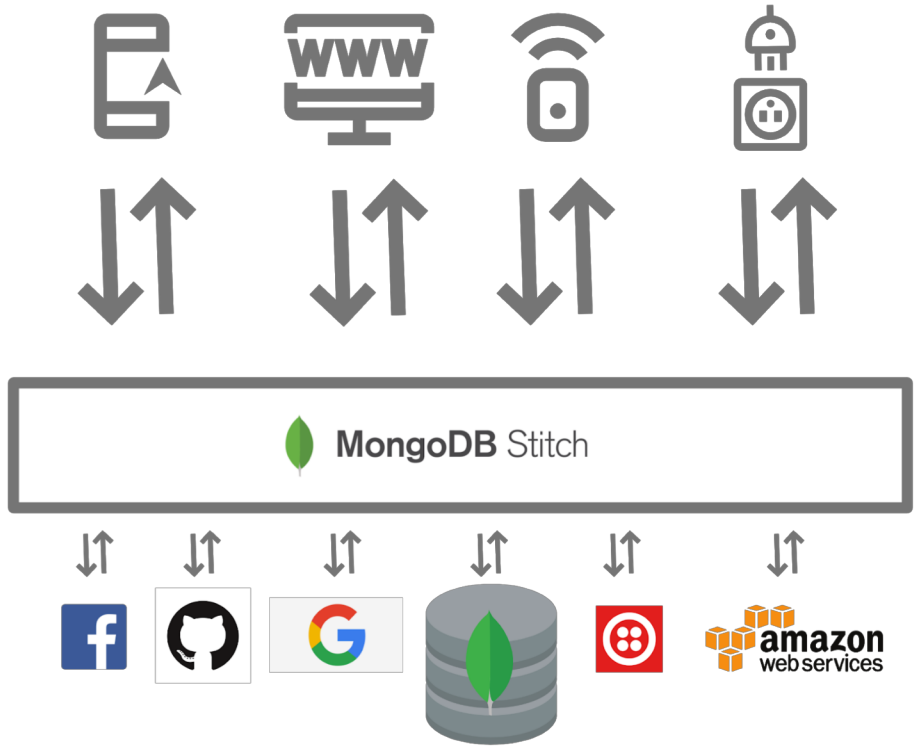
Avoid lock-in

Easily migrate data between cloud providers

MongoDB Serverless

mongoDB. Stitch

- The serverless platform from MongoDB.
- https://www.mongodb.com/cloud/stitch



MongoDB GUI

- The easiest way to explore and manage your MongoDB Data
- https://www.mongodb.com/products/compass

MongoDB for Analytics

Setting up MongoDB

9:30 – 10:00

MongoDB 101 Workshop

Introduction to MongoDB (30 mins)

Setting up MongoDB (30 mins)
- Laptop: Standalone Engine & Shell, Compass, MongoDB Driver (Python)
- Data Generator: MGenerateJS
- Cloud: MongoDB Atlas account

Querying, Indexing and Aggregation (30 mins)
- Using Compass, MongoDB Shell & Python

Schema Design & Use Cases (30 mins)
- Embedding & Referencing, 1-1, 1-M, M-M, Patterns
- Single View, IoT, Time Series, eCommerce/Content Management

MongoDB Atlas (60 mins)
- Management, Clusters, Security
- High Availability & Scalability (Replication & Sharding)
- Managed BI Connector & Charts
- ODBC for BI Connector
- MongoDB Stitch*

mongoDB.

Downloads & Setup

MongoDB Server
https://www.mongodb.com/download-center/community

MongoDB Compass - 1.16.4 (stable version)
https://www.mongodb.com/download-center/compass

Node.JS and NPM (Node Package Manager)
https://nodejs.org/en/download/

https://www.npmjs.com/get-npm

MGenerateJS (random data generator)
https://www.npmjs.com/package/mgeneratejs

Python / Anaconda
https://www.python.org/downloads/

https://www.anaconda.com/distribution/

PIP (Package Installer for Python)
https://pip.pypa.io/en/stable/installing/

Python MongoDB Driver
http://api.mongodb.com/python/current/installation.html

GitHub
https://desktop.github.com/

Clone https://github.com/snarvaez/mdb-101

Run MongoDB Server (local)

https://docs.mongodb.com/manual/installation/

Terminal 1 (Server):
./mongod --dbpath ../data

Terminal 2 (Shell):
./mongo

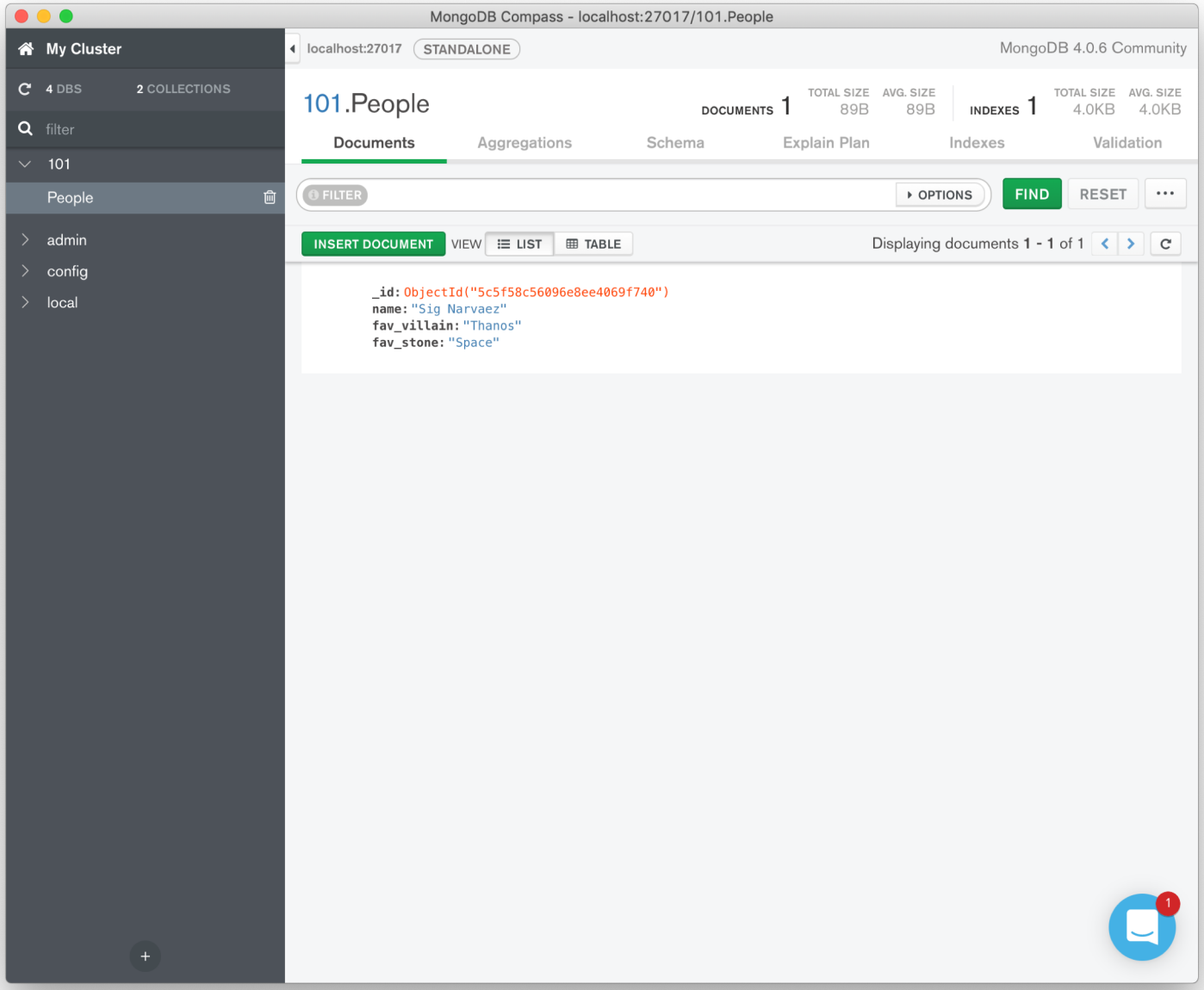
```
> use 101
switched to db 101

> db.People.save({"name":"Sig Narvaez", "fav_villain":"Thanos"});
WriteResult({ "nInserted" : 1 })

> db.People.count();
1

> db.People.find({"fav_villain":”Thanos"});
{ "_id" : ObjectId("5c5f53430d71dc70544ef715"), "name" : "Sig Narvaez", "fav_villain" : "Thanos” }
```

Run MongoDB Compass



Setup MongoDB Driver

https://docs.mongodb.com/ecosystem/drivers/

https://docs.mongodb.com/ecosystem/drivers/python/

http://api.mongodb.com/python/current/installation.html

http://api.mongodb.com/python/current/tutorial.html

python -m pip install pymongo

git clone https://github.com/snarvaez/mdb-101

python mdb-101.py

Setup MongoDB Data Generator

Node Package Manager (NPM) and Node.js

- https://docs.npmjs.com/downloading-and-installing-node-js-and-npm

MGenerateJS

- https://github.com/rueckstiess/mgeneratejs/projects

```
mgeneratejs -n 1 people_template.json

mgeneratejs -n 1 people_template.json | ./mongoimport -d 101 -c People

mgeneratejs -n 10000 people_template.json | ./mongoimport -d 101 -c
People
```

Extra time?

- Create variance in your collection(s)
- Change the template (e.g. add/remove fields)

MongoDB Atlas Account

cloud.mongodb.com

New Account

Use Credit Code & Setup Credit Card (optional)

Create M0 Free Cluster

Whitelist current IP

Setup DB User

Insert data with generator

Connect/Browse with UI & Compass

$75 credit:
MDBATRMOUG75

mongoDB.

Querying, Indexing & Aggregation

10:00 – 10:30

MongoDB 101 Workshop

Introduction to MongoDB (30 mins)

Setting up MongoDB (30 mins)
- Laptop: Standalone Engine & Shell, Compass, MongoDB Driver (Python)
- Data Generator: MGenerateJS
- Cloud: MongoDB Atlas account

Querying, Indexing and Aggregation (30 mins)
- Using Compass, MongoDB Shell & Python

Schema Design & Use Cases (30 mins)
- Embedding & Referencing, 1-1, 1-M, M-M, Patterns
- Single View, IoT, Time Series, eCommerce/Content Management

MongoDB Atlas (60 mins)
- Management, Clusters, Security
- High Availability & Scalability (Replication & Sharding)
- Managed BI Connector & Charts
- ODBC for BI Connector
- MongoDB Stitch*

mongoDB.

Use Your Favorite Programming Language

Drivers

Drivers for most popular programming languages and frameworks


Java


Ruby

JavaScript


Perl


Python

Shell

Command-line shell for interacting directly with database

```
> db.contact.insert({name: "John Smith", address: "10
3rd St.", phone: {home: 1234567890, mobile:
1234568138}})
>
> db.contact.findOne()
{
        "_id": ObjectId("5106c1c2fc629bfe52792e86"),
        "name": "John Smith",
        "address": "10 3rd St.",
        "phone": {
                "home": 1234567890,
                "mobile": 1234568138 }
}
```

Query Language and Operators

```
// find customers with any claims
> db.customers.find( {claims: {$exists: true }} )

// find customers matching a regular expression
> db.customers.find( {last: /^rog*/i } )

// count customers by city
> db.customers.find( {city: 'Los Angeles'} ).count()
```

Queries and What They Match

`{a: 10}`	Docs where `a` is `10`, or an array containing the value `10`.
`{a: 10, b: "hello"}`	Docs where `a` is `10` and `b` is `"hello"`.
`{a: {$gt: 10}}`	Docs where `a` is greater than `10`. *Also available:* `$lt` *(<),* `$gte` *(>=),* `$lte` *(<=), and* `$ne` *(!=).*
`{a: {$in: [10, "hello"]}}`	Docs where `a` is either `10` or `"hello"`.
`{a: {$all: [10, "hello"]}}`	Docs where `a` is an array containing both `10` and `"hello"`.
`{"a.b": 10}`	Docs where `a` is an embedded document with `b` equal to `10`.
`{a: {$elemMatch: {b: 1, c: 2}}}`	Docs where `a` is an array that contains an element with both `b` equal to `1` and `c` equal to `2`.
`{$or: [{a: 1}, {b: 2}]}`	Docs where `a` is `1` or `b` is `2`.
`{a: /^m/}`	Docs where `a` begins with the letter `m`. *One can also use the regex operator:* `{a: {$regex: "^m"}}`.
`{a: {$mod: [10, 1]}}`	Docs where `a` mod `10` is `1`.
`{a: {$type: 2}}`	Docs where `a` is a string. *See bsonspec.org for more.*
`{ $text: { $search: "hello" } }`	Docs that contain `"hello"` on a text search. Requires a text index.

Secondary Indexes

```
// Create index on policyId (ascending)
> db.policies.createIndex( { policyId : 1 } )

// Create index on policyId (descending)
> db.policies.createIndex( { policyId : -1 } )

// Create index on arrays for values on the "types" field – multi key index
> db.policies.createIndex( { types : 1 } )   // { types: ["home", "auto", "life"] }

// Compound index on the policyId and type fields
> db.policies.createIndex( { policyId : 1, type : 1 } )

// Compound index supports both the below queries
> db.policies.find( { policyId : 123, type : 'life'} )
> db.policies.find( { policyId : 123 } )

// Text index on the comments field
> db.policies.createIndex( {comments: "text"} )

// Search comments and also score text search results
> db.policies.find ({$text: { $search: "payment" }}, { score: { $meta: "textScore" })
```

mongoDB

Java - Basic Query API Example

Objective	Code	CLI

Find all contacts with at least one mobile phone	`Map expr = new HashMap();`
	`expr.put("phones.type",`
	`"mobile");`
		`db.contact.find({"phones.type":"mobile"});`

Find contacts with NO phones	`Map expr = new HashMap();`
	`Map q1 = new HashMap();`
	`q1.put("$exists", false);`
	`expr.put("phones", q1);`
		`db.contact.find({"phones":{"$exists":false}});`

```
List fetchGeneral(Map expr)
{
    List l = new ArrayList();
    DBObject dbo = new BasicDBObject(expr);
    Cursor c = collection.find(dbo);
    while (c.hasNext()) }
     Map m = c.next();
     showMap(m);
          l.add(m);
    }
    return l;
}

// collection.insert(Map m)
// collection.update(Map filter, Map repl)
// collection.remove(Map filter)
```

```
{    customer_id : 1,
     first_name : "Mark",
     last_name : "Smith",
     city : "San Francisco",
     phones: [ {
           number : "1-212-777-1212", dnc : true,
           type : "home"
     },{
           number : "1-212-777-1213",
           type : "cell"
     }] }
```

mongoDB

C# Examples

```
// To directly connect to a single MongoDB server
// (this will not auto-discover the primary even if it's a member of a replica set)
var client = new MongoClient();

// or use a connection string
var client = new MongoClient("mongodb://localhost:27017");

// or, to connect to a replica set, with auto-discovery of the primary, supply a seed list of
var client = new MongoClient("mongodb://localhost:27017,localhost:27018,localhost:27019");
```

```
var database = client.GetDatabase("foo");
```

```
var collection = database.GetCollection<BsonDocument>("bar");
```

```
var count = await collection.CountAsync(new BsonDocument());

Console.WriteLine(count);
```

```
var document = await collection.Find(new BsonDocument()).FirstOrDefaultAsync();
Console.WriteLine(document.ToString());
```

Python Example

```
import pymongo

myData = {
  "name": "jane",
  "id": "K2",
 # no title? No problem
  "hireDate": datetime.date(2011, 11, 1),
  "phones": [
  { "type": "work",
    "number": "1-800-555-1212"
  },
  { "type": "home",
    "number": "1-866-444-3131"
  }
  ]
}
coll.insert(myData)
print coll.find_one({"bizKey": "K2" })

expr = { "$or": [ {"phones": { "$exists": false }}, {"name": "jane"}]}
for c in coll.find(expr):
        print [ k.upper() for k in sorted(c.keys()) ]
```

- **Easy and fast to create scripts due to "fidelity-parity" of MongoDB map data and python (and perl, ruby, and Javascript) structures**
- Data types and structure in scripts are exactly the same as that read and written in Java and C++

Rich query functionality

Expressive Queries	• Find anyone with phone # “1-212…” • Check if the person with number “555…” is on the “do not call” list
Geospatial	• Find the best offer for the customer at geo coordinates of 42nd St. and 6th Ave
Text Search	• Find all tweets that mention the firm within the last 2 days
Aggregation	• Count and sort number of customers by city, compute min, max, and average spend
Native Binary JSON Support	• Add an additional phone number to Mark Smith’s record without rewriting the document • Update just 2 phone numbers out of 10 • Sort on the modified date
JOIN ($lookup)	• Query for all San Francisco residences, lookup their transactions, and sum the amount by person
Graph Queries ($graphLookup)	• Query for all people within 3 degrees of separation from Mark

MongoDB

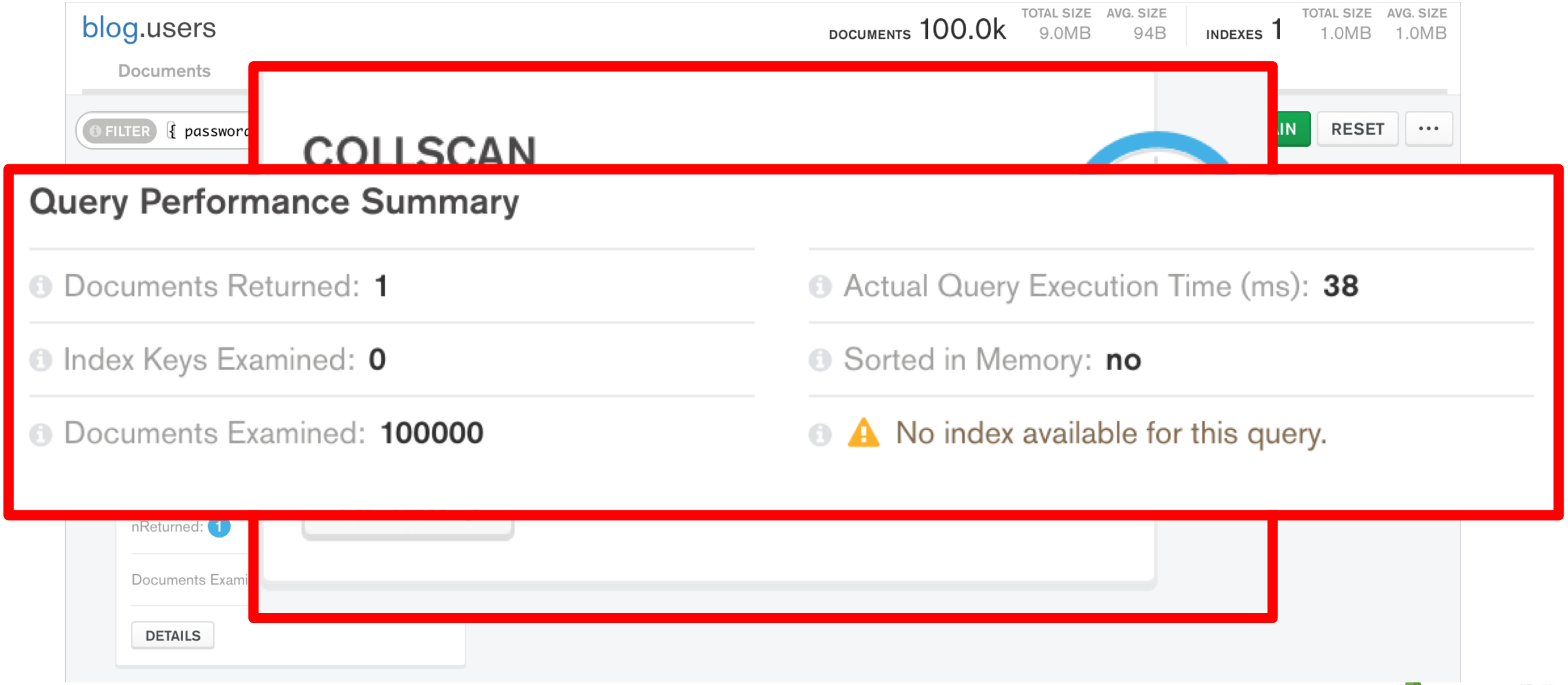
```
{       customer_id : 1,
        first_name : "Mark",
        last_name : "Smith",
        city : "San Francisco",
        phones: [       {
                        number : "1-212-777-1212",
                        type : "work"
        },
        {
                        number : "1-212-777-1213",
                        type : "cell"
        }]
.....
```

mongoDB.

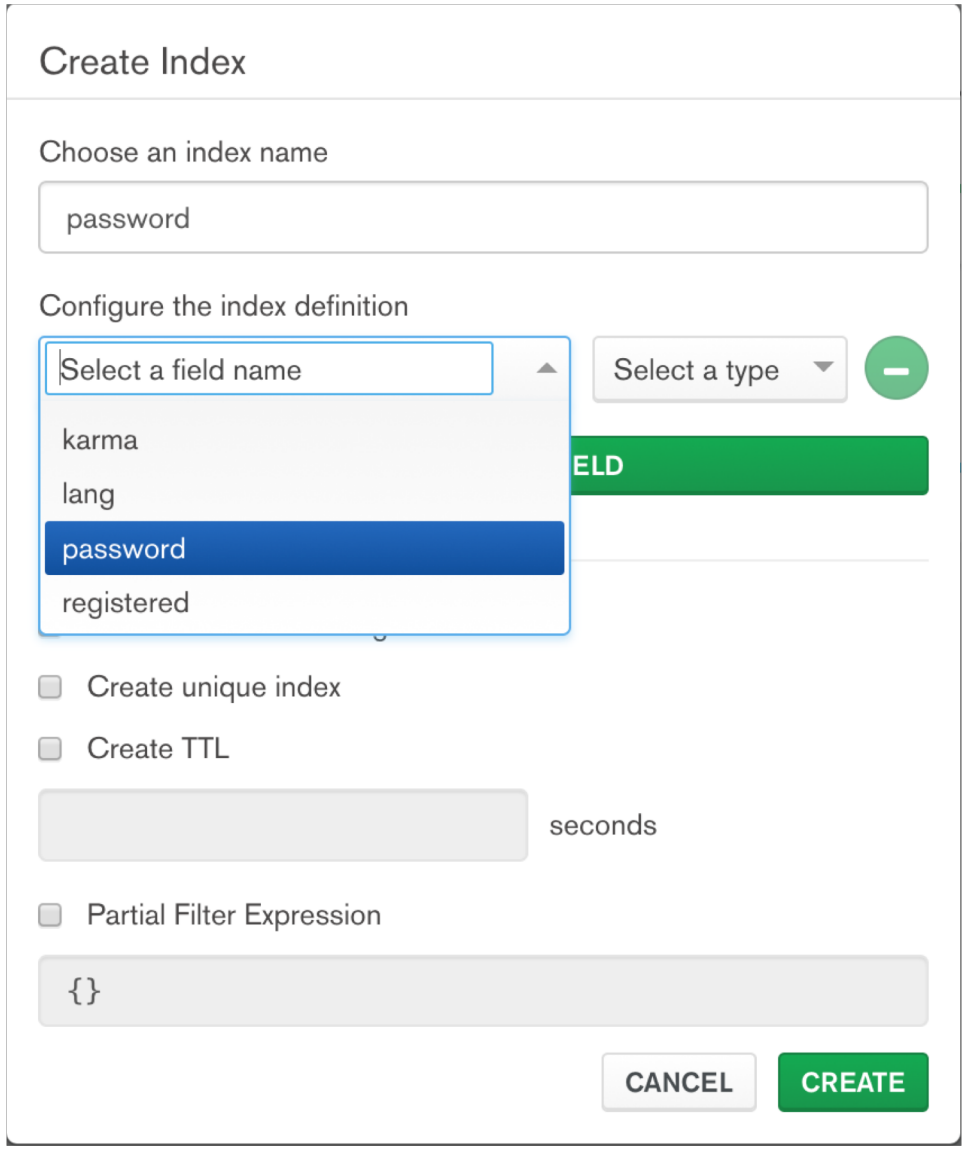
Indexes in MongoDB

Indexes support the efficient execution of queries in MongoDB.

Can still query w/o index → collscan

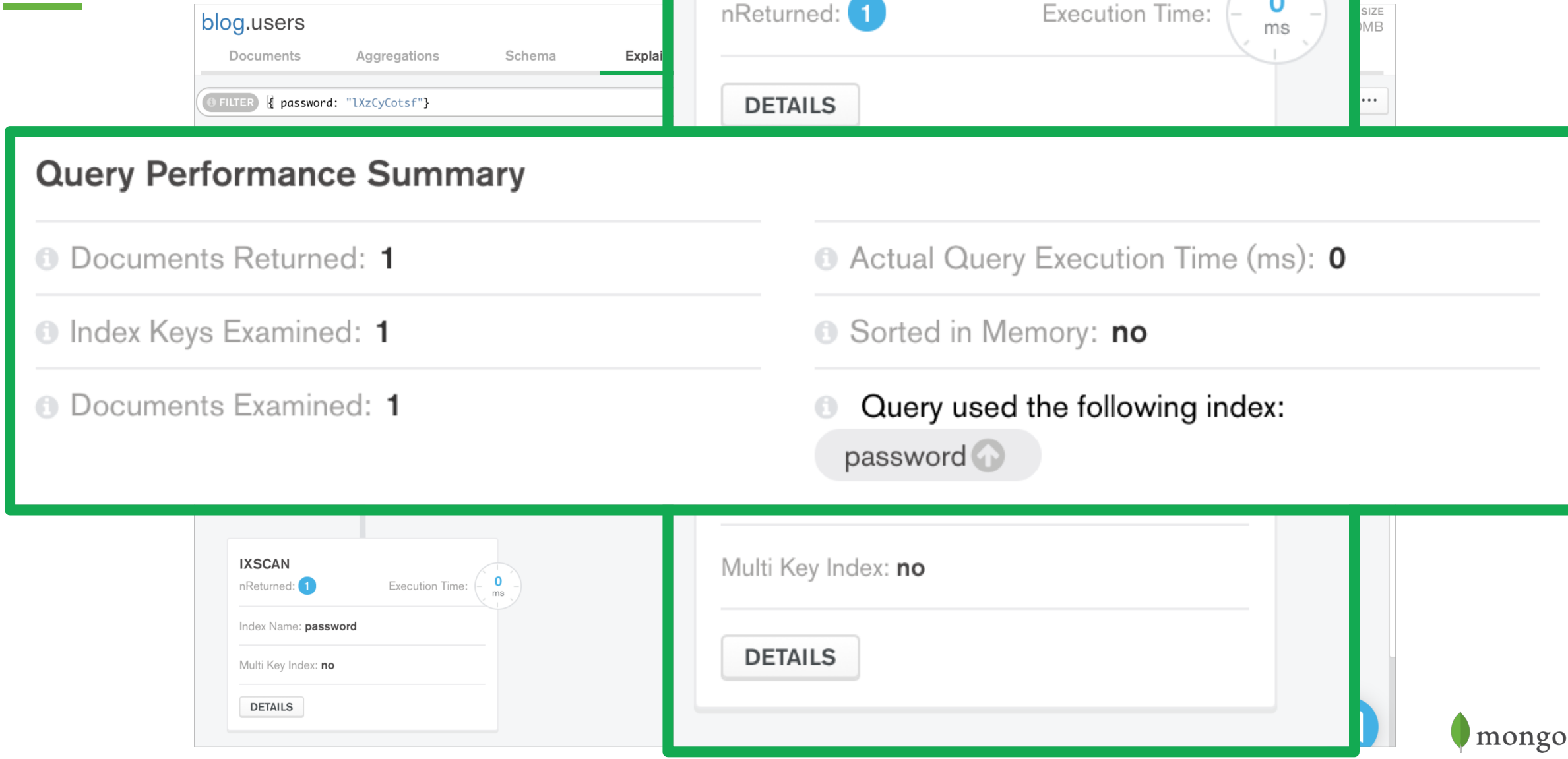


Index Creation in Compass



mongoDB.

Search with Index



mongoDB.

Index Types

Single Field

Compound Field

Multikey

Geospatial

Text

Hashed

Index Properties

Unique

Partial

Sparse

TTL

https://docs.mongodb.com/manual/indexes

Aggregation Pipeline

PIPELINE

*nix command line pipe

```
ps ax|grep mongod|head 1
```

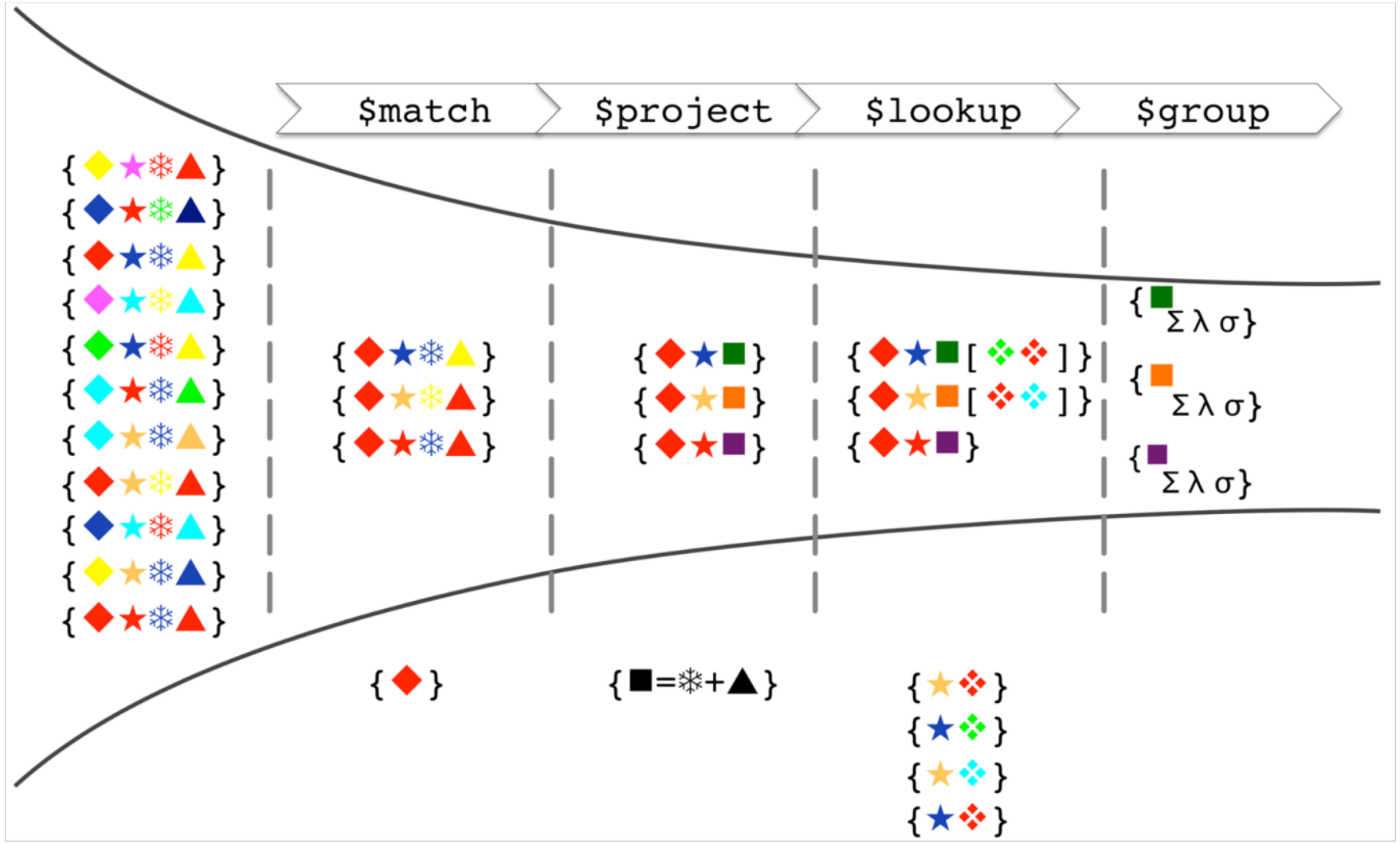
PIPELINE

MongoDB document pipeline

```
$match |$group | $sort
```

mongoDB.

Aggregation



mongoDB.

```
{ title: "The Great
Gatsby",
  language: "English",
  subjects: [
    "Long Island",
    "New York",
    "1920s"] },
{ title: "War and
Peace",
  language: "Russian",
  subjects: [
    "Russia",
    "War of 1812",
    "Napoleon"] },
{ title: "Open City",
  language: "English",
  subjects: [
    "New York",
    "Harlem" ] },
```

```
{title: "The Great
Gatsby",
 language: "English",
 subjects: [
    "Long Island",
    "New York",
    "1920s"] },
{title: "Open City",
 language:
"English",
 subjects: [
    "New York"
    "Harlem" ] }
```

```
{title: "The Great
Gatsby",
 language: "English",
{title: "The Great
Gatsby",
{title: "The Great
Gatsby",
 language: "English",
{title: "Open City",
 language: "English",
{title: "Open City",
 language: "English",
  subjects: "Harlem"}
```

```
{ _id: "Long
Island",
{ _id: "New
York",
{ _id: "1920s",
  count: 1 },
{ _id:
"Harlem",
  count: 1 },
```

```
{ _id: "New
York",
  count: 2},
{ _id: "Long
Island",
{ _id: "1920s",
  count: 1 },
{ _id: "Harlem",
  count: 1 },
```

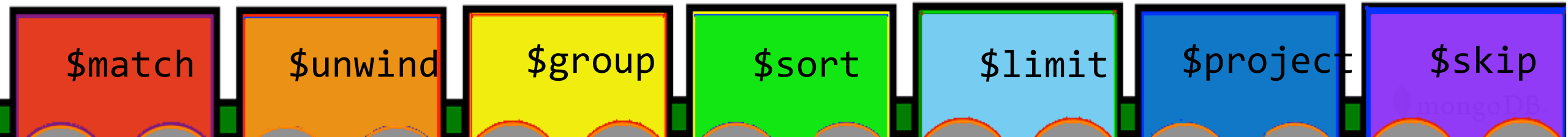
```
{"$match":{"language":"English"}}
```

```
{"$unwind":"$subjects"}
```

```
{"$group":{"_id":"$subjects",
"count":{"$sum:1}}
```

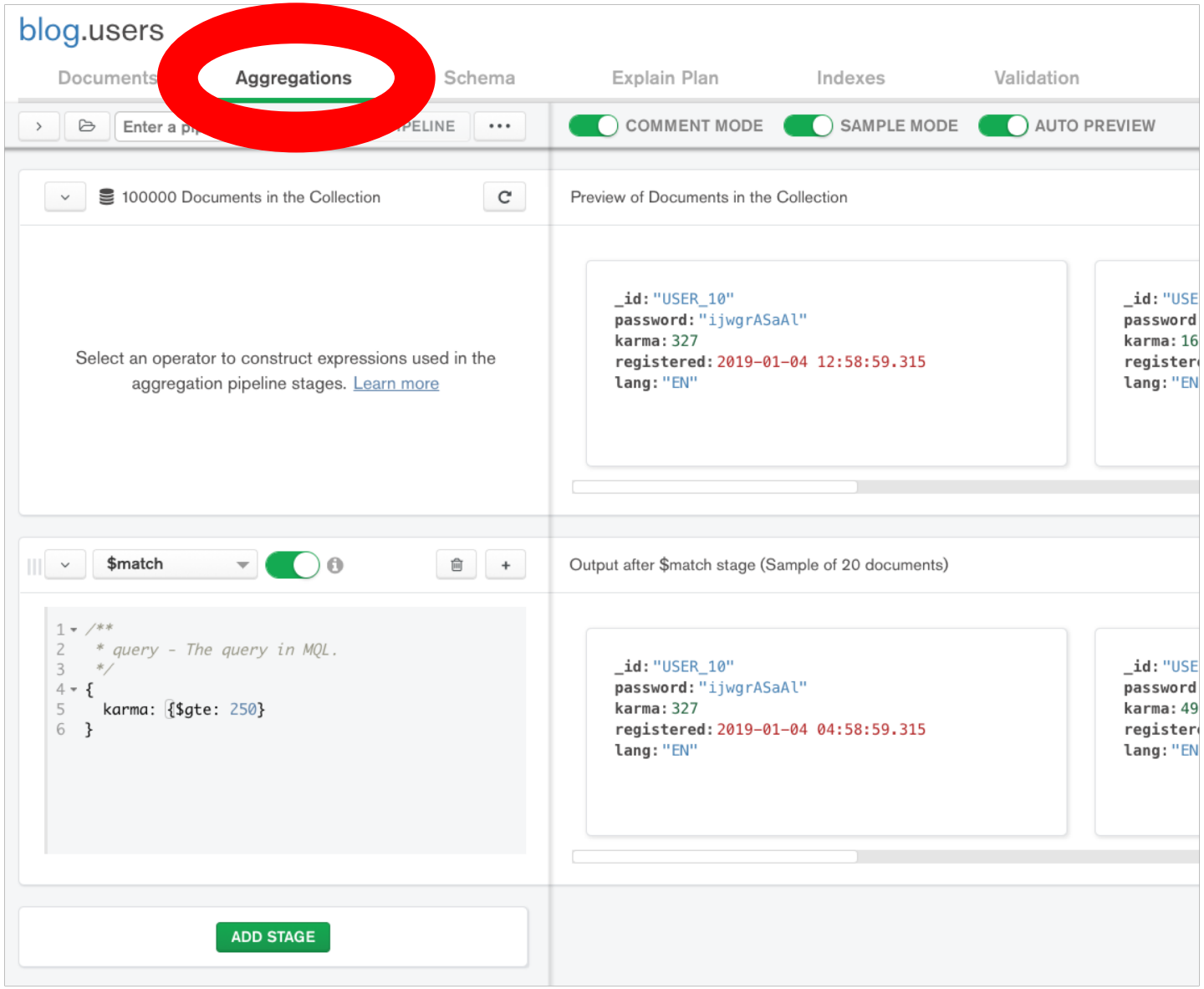
```
{"$sort:{"count":-1}
```

```
{"$limit":3}
  {"$project":...}
```



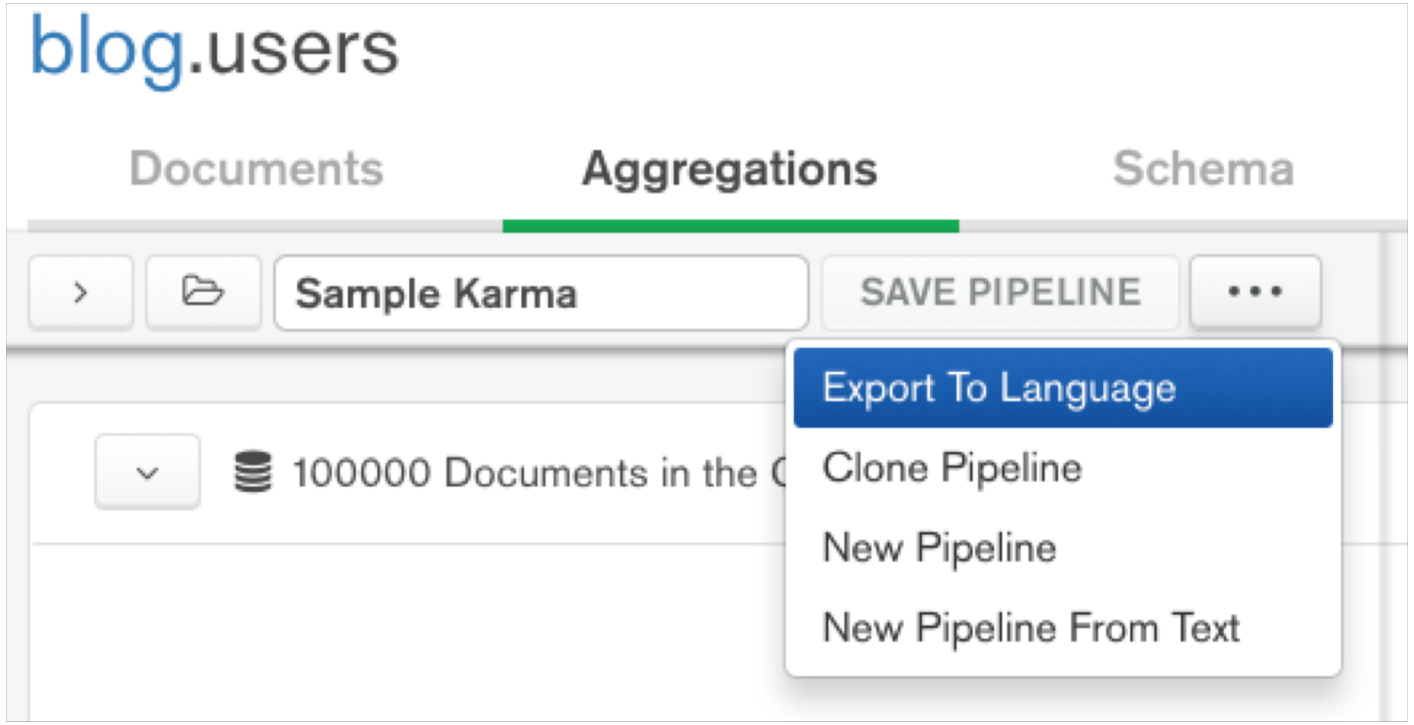
Aggregation Pipeline Builder in Compass

Pipeline Builder

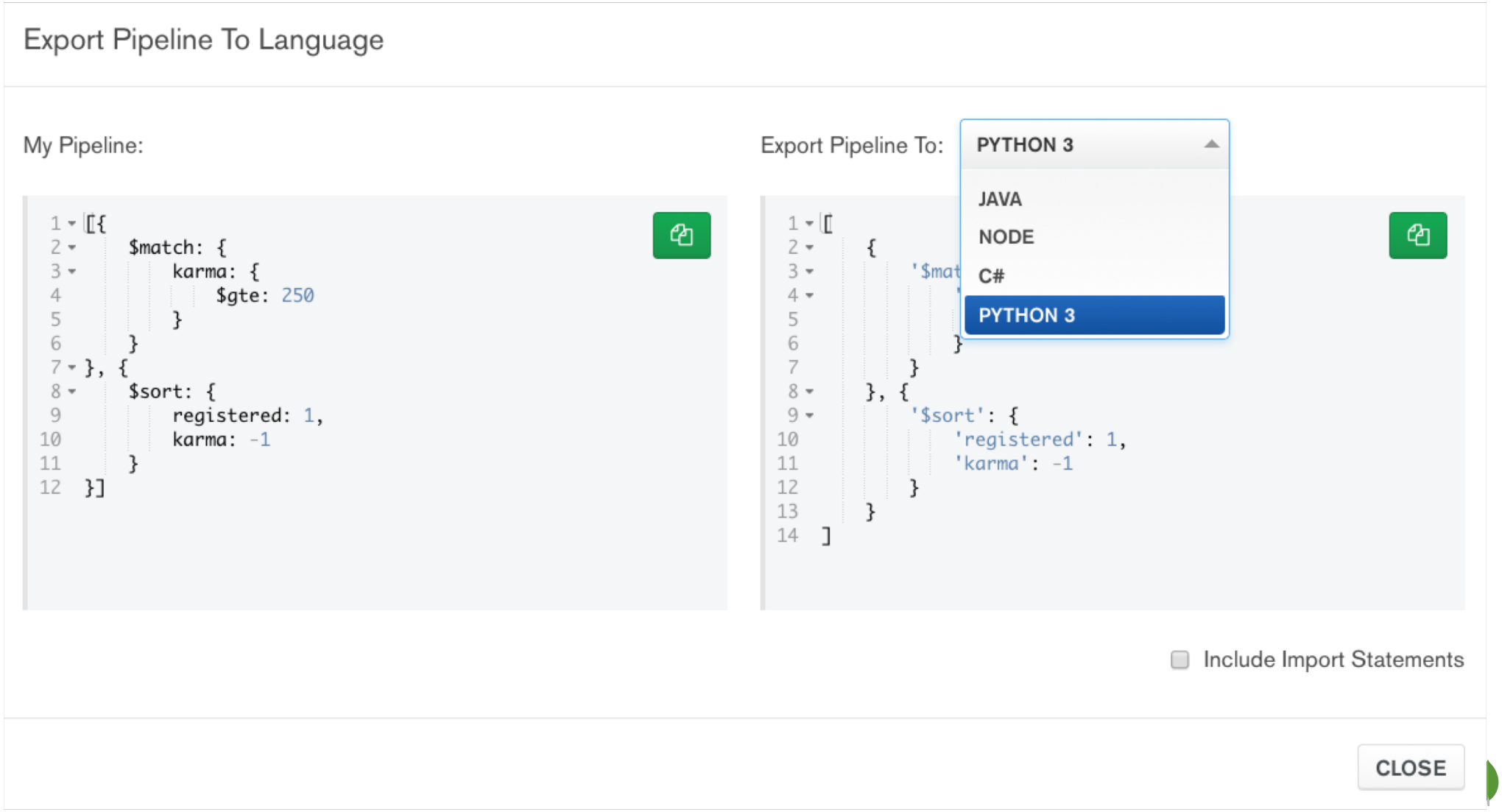


mongoDB.

Pipeline Builder



Pipeline Builder Language Exporter



Hands on!

Customer Single View Template

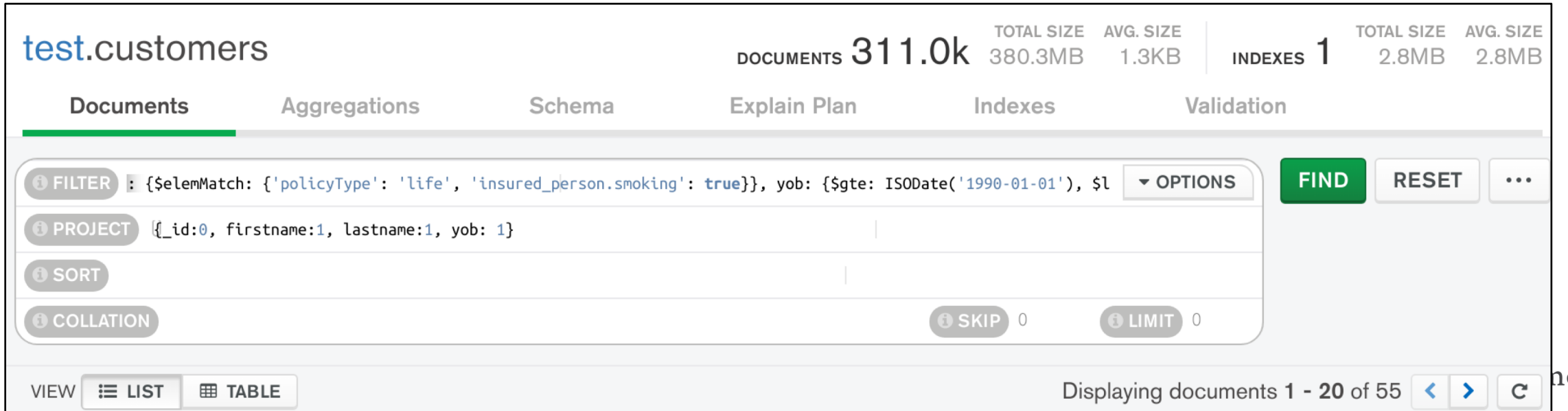
```
mgeneratejs -n 100000 CustomerSingleView.json | ./mongoimport -d 101 -c
Customers
```

Complex Query

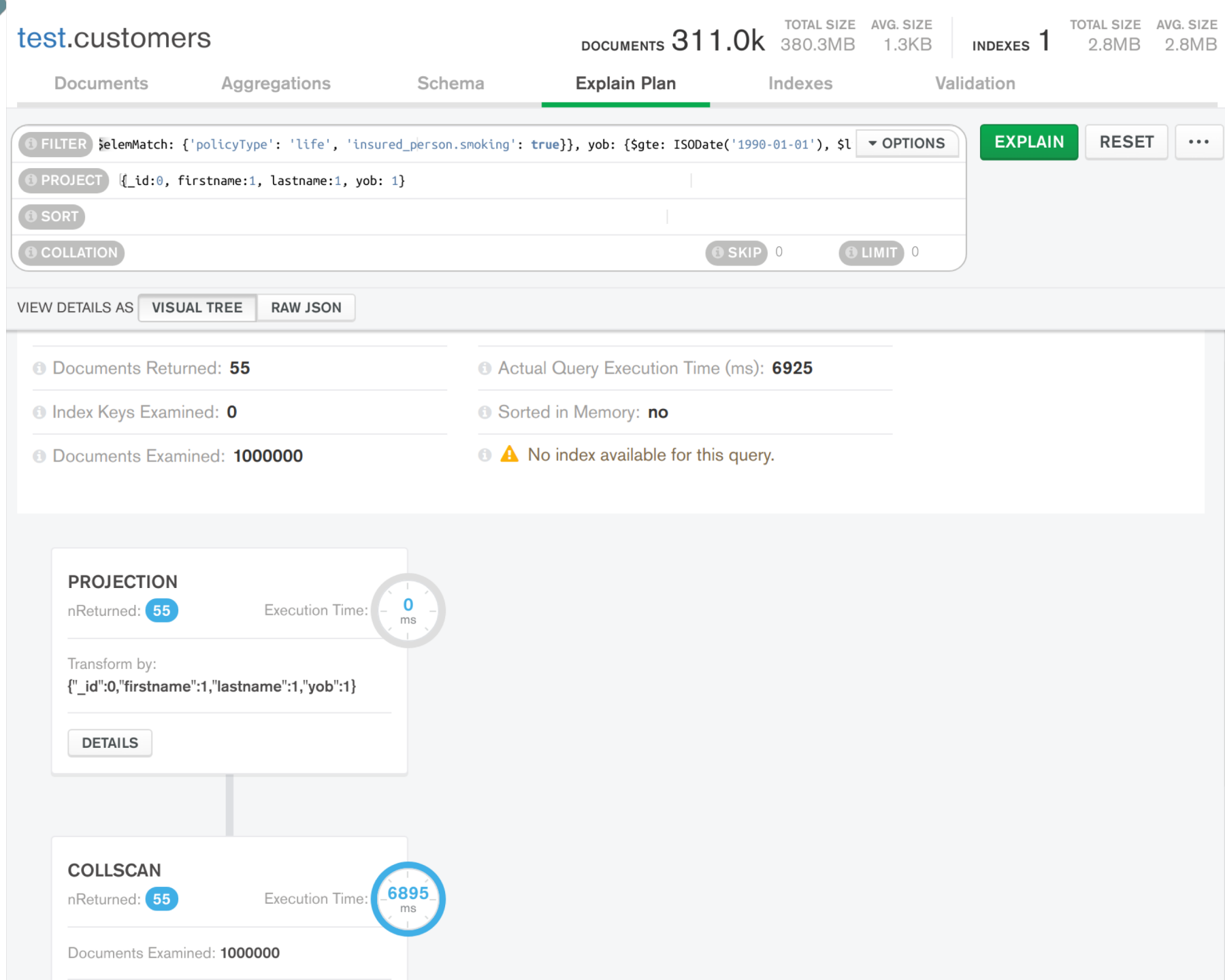
Show all insurance customers who are Female, born in 1990, live in Utah and have at least one Life insurance policy for which they are registered as a Smoker

FILTER: {'address.state': 'UT', policies: {$elemMatch: {'policyType': 'life', 'insured_person.smoking': true}}, yob: {$gte: ISODate('1990-01-01'), $lte : ISODate('1990-12-12')}, 'gender': 'Female'}

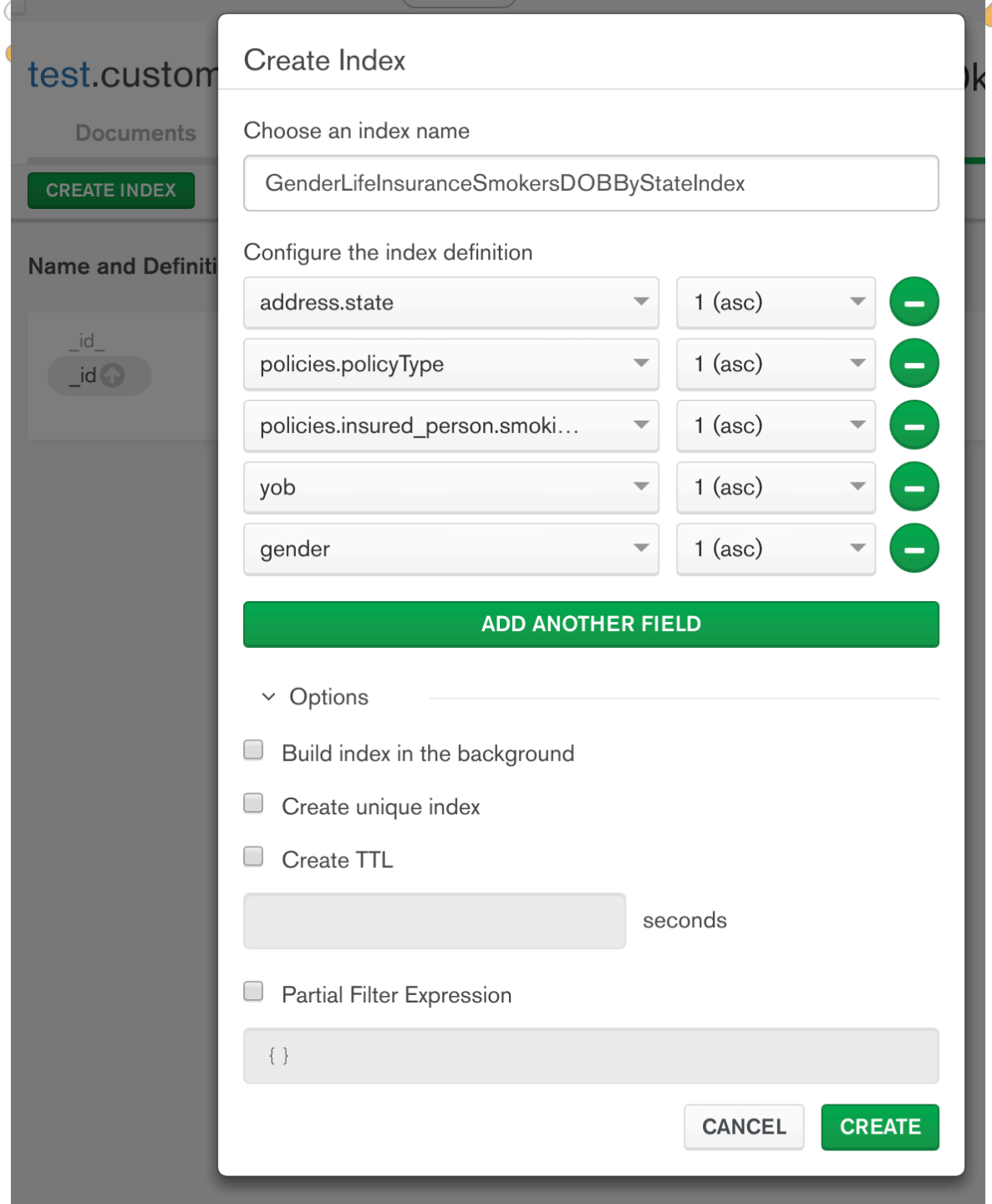
PROJECT: {_id:0, firstname:1, lastname:1, yob: 1}



Explain Plan



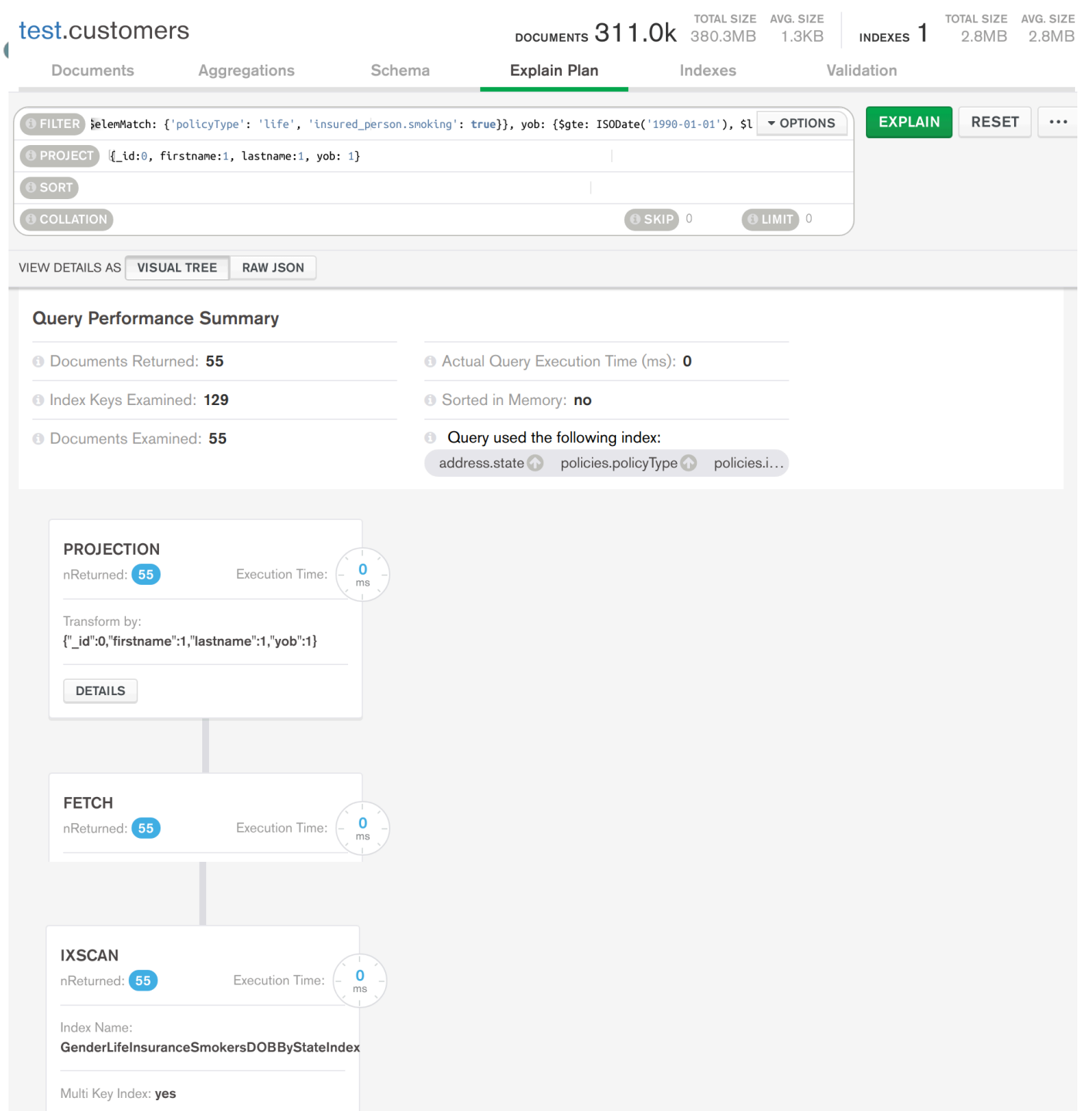
Indexes



mongoDB.

Efficient Query

Test	# Docs Examined	# Docs Returned	Exec Time (ms)
Rich query without index	1000000	~50	6000
Rich query with index	~50	~50	1



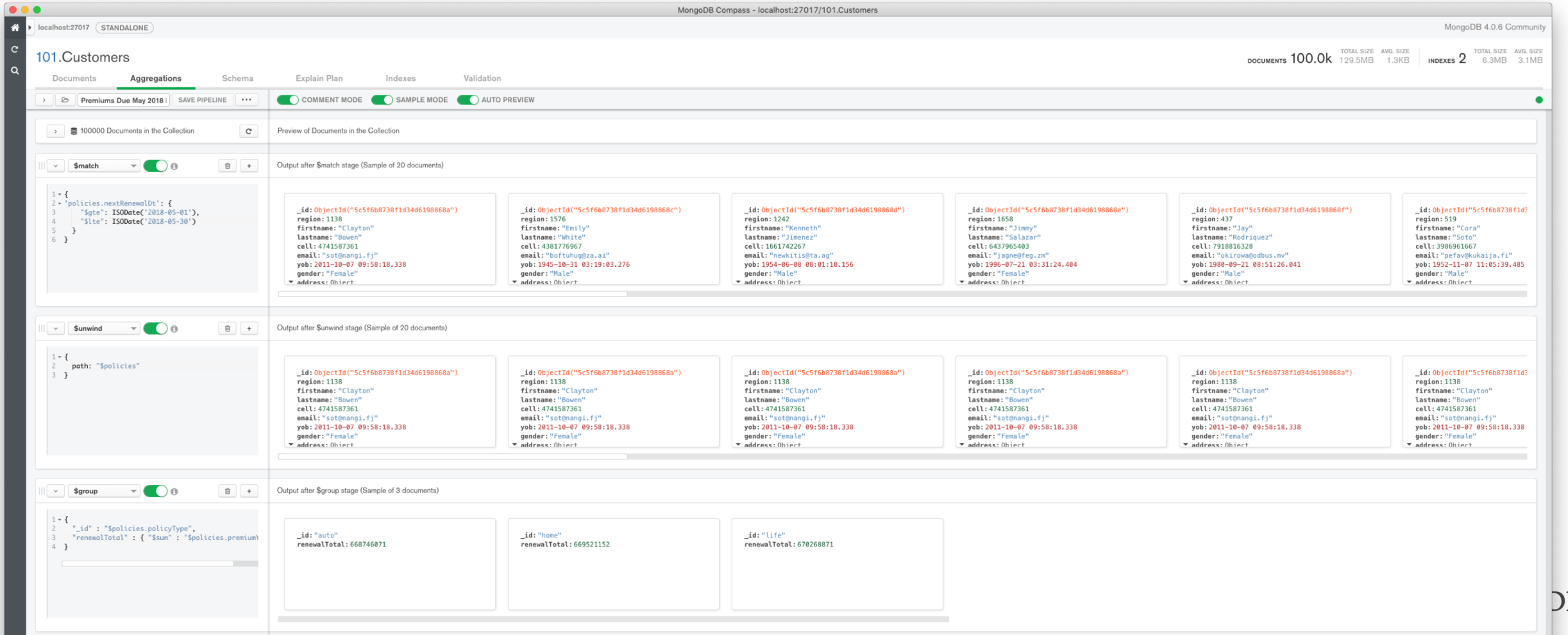
Aggregation

Show all Sum of premiums for policies due in May 2018 by Policy Type

```
1  // Sum of premiums for policies due in May 2018 by Policy Type
2  db.Customers.aggregate([
3    { "$match"  : {'policies.nextRenewalDt': {
4                        "$gte": new Date('2018-05-01'),
5                        "$lte": new Date('2018-05-30') }}},
6
7    { "$unwind" : "$policies"},
8
9    { "$group"  : { "_id" : "$policies.policyType",
10                   "renewalTotal" : { "$sum" : "$policies.premiumYear" }}}
11 ]);
```

Aggregation

Show all Sum of premiums for policies due in May 2018 by Policy Type



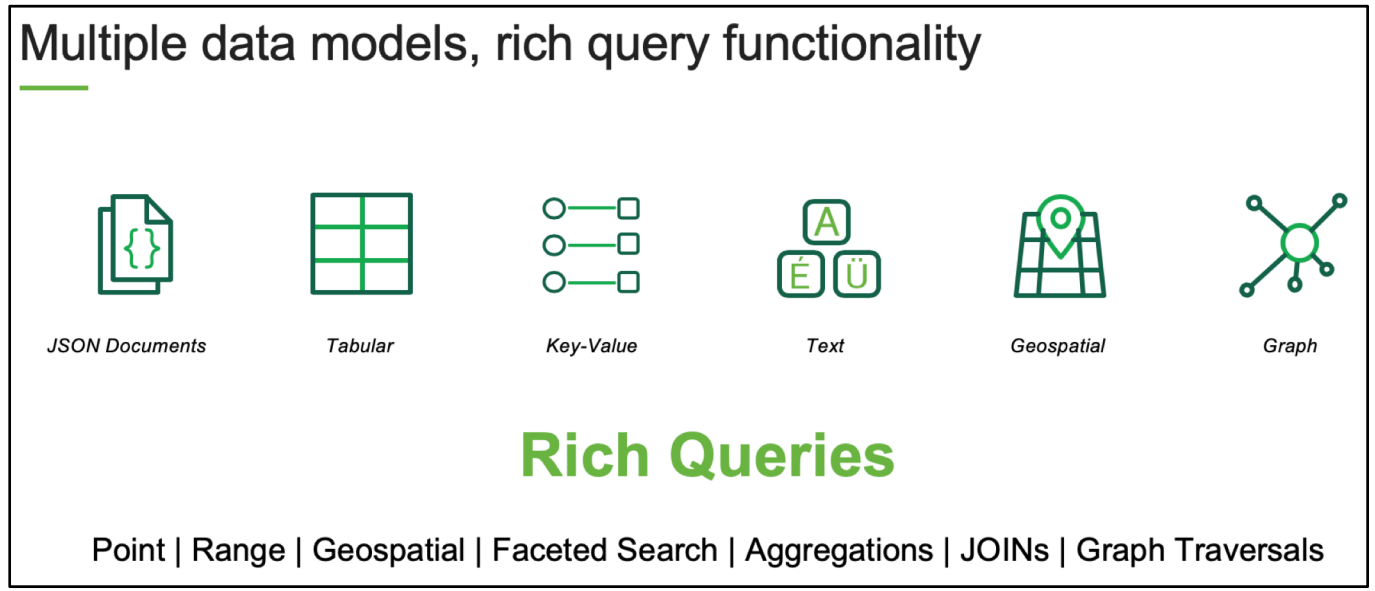
Repeat via MongoDB shell & Python

Time Check ??

QueryIndexAgg.js

QueryIndexAgg.py

Multi-Model.js



Schema Design & Use Cases

10:30 – 11:00

MongoDB 101 Workshop

Introduction to MongoDB (30 mins)

Setting up MongoDB (30 mins)
- Laptop: Standalone Engine & Shell, Compass, MongoDB Driver (Python)
- Data Generator: MGenerateJS
- Cloud: MongoDB Atlas account

Querying, Indexing and Aggregation (30 mins)
- Using Compass, MongoDB Shell & Python

Schema Design & Use Cases (30 mins)
- Embedding & Referencing, 1-1, 1-M, M-M, Patterns
- Single View, IoT, Time Series, eCommerce/Content Management

MongoDB Atlas (60 mins)
- Management, Clusters, Security
- High Availability & Scalability (Replication & Sharding)
- Managed BI Connector & Charts
- ODBC for BI Connector
- MongoDB Stitch*

mongoDB.

Tabular/Relational vs. Object Oriented

mongoDB.

Documents are FLEXIBLE

Different Documents in the same ProductsCatalog collection in MongoDB

Polymorphic Schema – Aligns with OOP principles

Car Paint products

```
{
  sku: 'PAINTZXC123',
  product_name: 'Metallic Paint',
  colors: ['Red', 'Green'],
  size_gallons: [5, 10]
}
```

Car T-Shirt products

```
{
  sku: 'TSHRTASD43546',
  product_name: 'T-shirt',
  size: ['S', 'M', 'L', 'XL'],
  colors: ['Heather Gray' … ],
  material: '100% cotton',
  wash: 'cold',
  dry: 'tumble dry low'
}
```

Car Wheels products

```
{
  sku: 'WHEELBVCX6543',
  product_name: '19" 5-spoke',
  material: 'aluminum alloy',
  color: 'silver',
  frame_material: 'aluminum',
  package_height: '20.5x32.9x55',
  weight_lbs: 5.15,
  wheel_size_in: 19
}
```

Flexible Data Governance

```
db.runCommand({collMod: "ProductCatalog",
    validator: {
        "$and": [
                {"sku":          {"$type": "string"}},
                {"product_name": {"$type": "string"}}
            ]}
        ]},
     "validationLevel": "off | moderate | strict",
     "validationAction": "error | warn"
});
```

All products must have an SKU and Name

mongoDB.

Relational – schema validation

```
db.runCommand({collMod: "orders",
    validator : {
        "$and"  : [
            {"orderID" : {"$type" : "string"}},
            {"customer.customerID" : {"$type": "string"}},
            {"items.0" : {"$exists" : true}}
    ]},
        "validationLevel": "strict",
        "validationAction": "error"
});
```

Rule

"An order must have at least one item"

Not possible with SQL DDL

Level: off | moderate | strict

Action: error | warn

```
db.orders.insert({
    "orderID": "order1",
    "customer": {"customerID": "customer1"},
    "items": []
});
WriteResult({
    "nInserted" : 0,
    "writeError" : {
        "code" : 121,
        "errmsg" : "Document failed validation"
    }
})
```

Schema on-write

Schema Design

It’s all about the data

Questions to keep in mind

Data access patterns? Number of Reads vs. Updates?

Expected size of a Document?

Data that works together, lives together

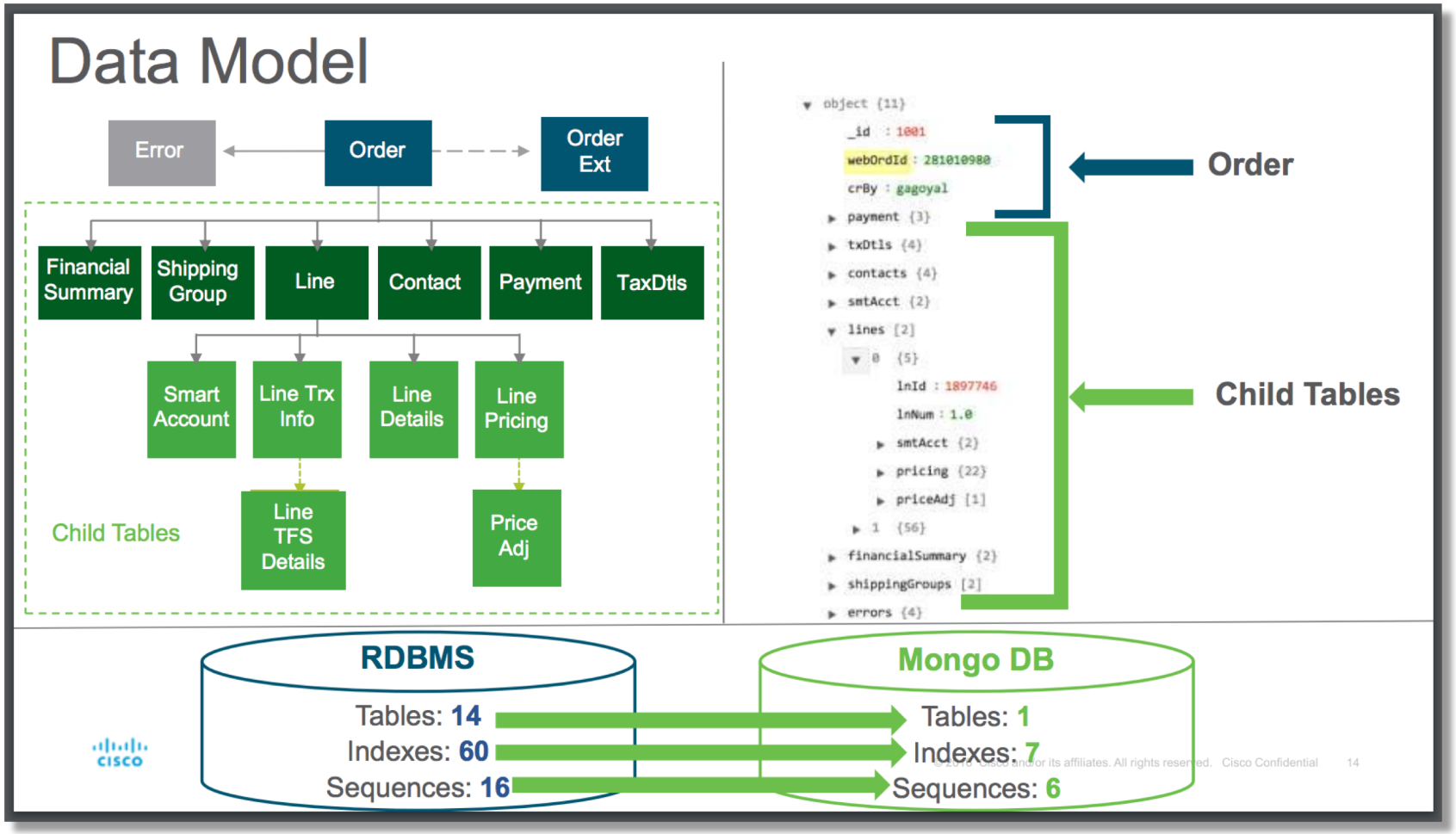
Why Schema Design is important

Almost all performance issues are related to Schema Design

CISCO's $40B eCommerce Platform

14 SQL tables ➔ 1 MongoDB Collection

60 Indexes ➔ 7 Indexes

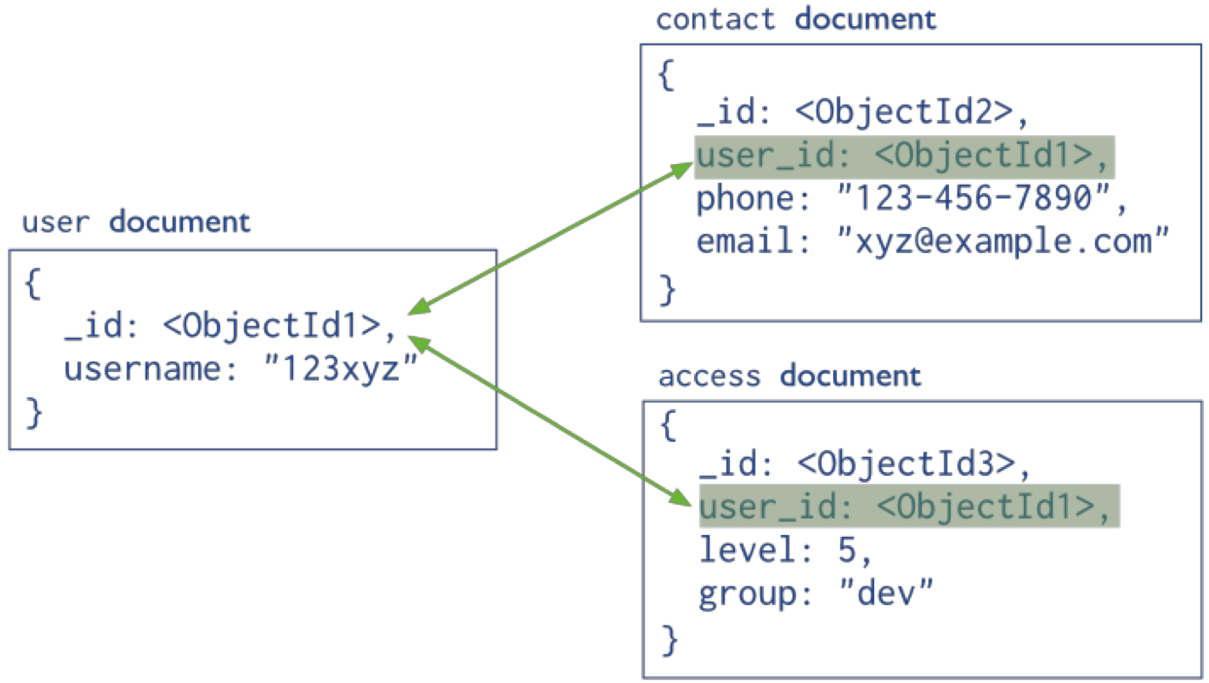


© 2015 Cisco and/or its affiliates. All rights reserved. Cisco Confidential 14

mongoDB.

What about what I know from RDBMS's?

EMBEDDING & REFERENCING



https://docs.mongodb.com/manual/core/data-modeling-introduction/

Healthcare use case

1-1

Medical Procedure collection

```
{
    "_id": 333,
    "date": "2003-02-09T05:00:00",
    "hospital": "County Hills",
    "patient": "John Doe",
    "physician": "Stephen Smith",
    "procedure": "Glucose",
    "result": {
        "value": 97,
        "measurement": "mg/dl"
    }
}
```

Embed – *weak entity*

1-M

Modeled in 2 possible ways

Embed

Patients

```
{
    _id: 2,
    first: “Joe”,
    last: “Patient”,
    addr: { …},
    procedures: [
        {
            id: 12345,
            date: 2015-02-15,
            type: “Cat scan”,
                …},
        {
            id: 12346,
            date: 2015-02-15,
            type: “blood test”,
                …}]
}
```

Reference

Patients

```
{
    _id: 2,
    first: “Joe”,
    last: “Patient”,
    addr: { …},
    procedures: [12345, 12346]}
```

Procedures

```
{
    _id: 12345,
    date: 2015-02-15,
    type: “Cat scan”,
     …}
 {
    _id: 12346,
    date: 2015-02-15,
    type: “blood test”,
    …}
```

M-M

No Join Tables

Use **arrays** instead

mongoDB.

M-M

Embedding **Physicians** in **Hospitals** collection

```
{
    _id: 1,
    name: “Oak Valley Hospital”,
    city: “New York”,
    beds: 131,
    physicians: [
        {
            id: 12345,
            name: “Joe Doctor”,
            address: {…},
                …},
        {
            id: 12346,
            name: “Mary Well”,
            address: {…},
                …}]
}
```

Data Duplication

…

is ok!

```
{
    _id: 2,
    name: “Plainmont Hospital”,
    city: “Omaha”,
    beds: 85,
    physicians: [
        {
            id: 63633,
            name: “Harold Green”,
            address: {…},
                …},
        {
            id: 12345,
            name: “Joe Doctor”,
            address: {…},
                …}]
}
```

M-M

Referencing

Hospitals

```
{
    _id: 1,
    name: “Oak Valley Hospital”,
    city: “New York”,
    beds: 131,
    physicians: [12345, 12346]
}
```

```
{
    _id: 2,
    name: “Plainmont Hospital”,
    city: “Omaha”,
    beds: 85,
    physicians: [63633, 12345]
}
```

Physicians

```
{
    id: 63633,
    name: “Harold Green”,
    hospitals: [1,2],
    …}
```

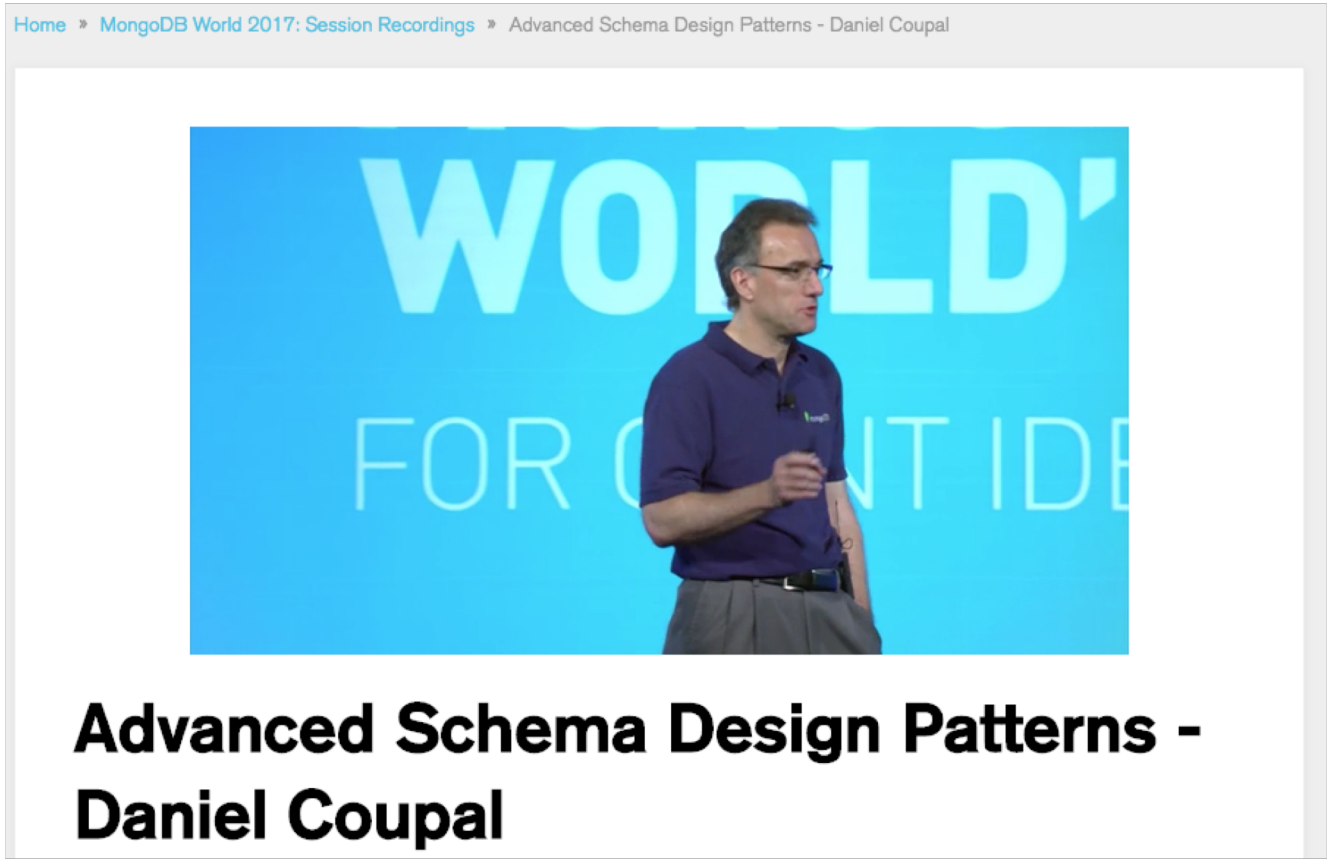
```
{
    id: 12345,
    name: “Joe Doctor”,
    hospitals: [1],
    …}
```

```
{
    id: 12346,
    name: “Mary Well”,
    hospitals: [1,2],
    …}
```

MongoDB

Schema Design Patterns

Full talk from MDBW17

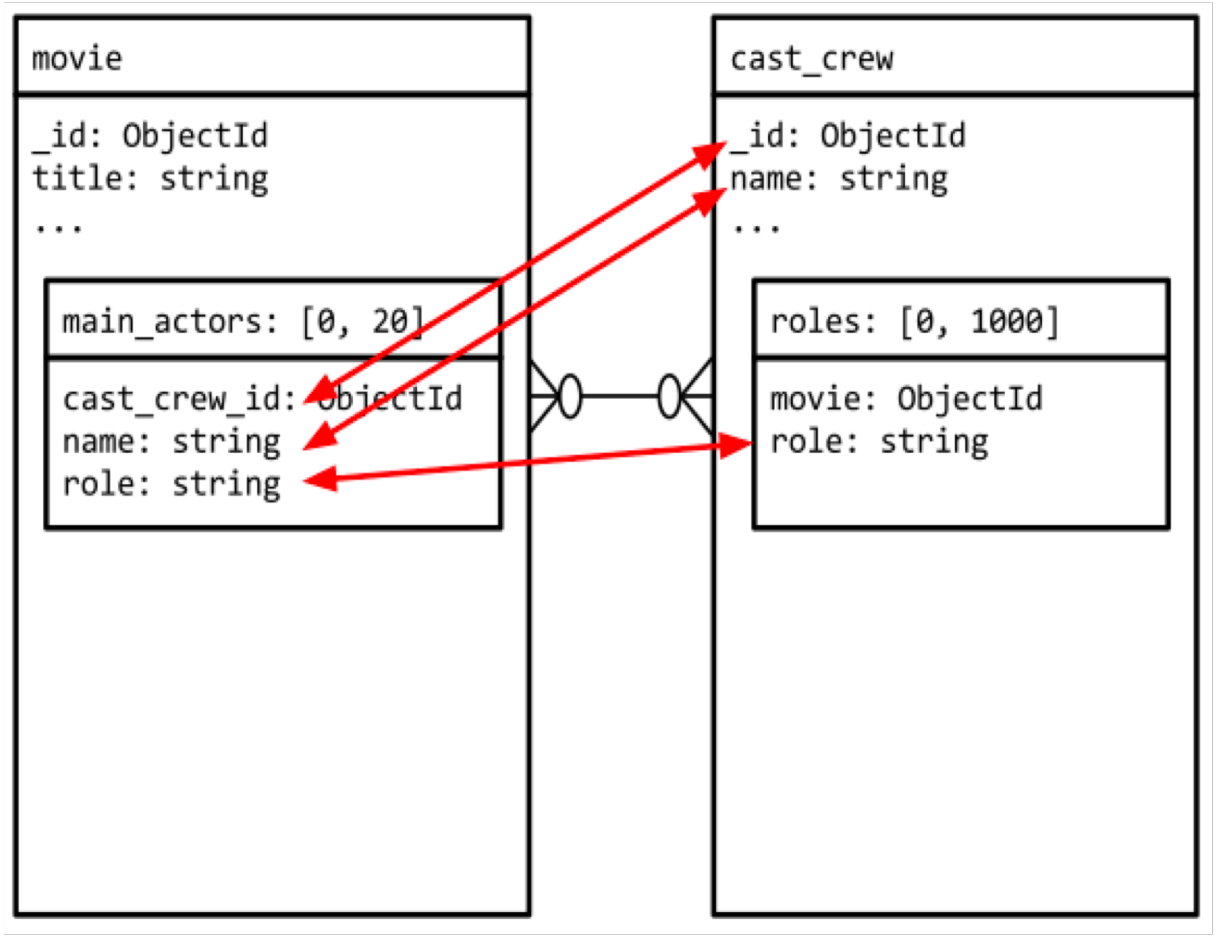


https://explore.mongodb.com

mongoDB.

SUBSET Pattern – Entertainment Use Case

- You want to display dependent information, *however only part of it*
- The rest of the data is fetched only if needed
- Examples:
  - The Cast of a Movie
  - Last 10 Movies an Actor has starred in



mongoDB.

Easy: MongoDB Multi-Document ACID Transactions

Just like relational transactions

- Multi-statement, familiar relational syntax
- Easy to add to any application
- Multiple documents in 1 or many collections and databases

ACID guarantees

- Snapshot isolation, all or nothing execution
- No performance impact for non-transactional operations

Schedule

- MongoDB 4.0: replica set
- MongoDB 4.2: extended to sharded clusters

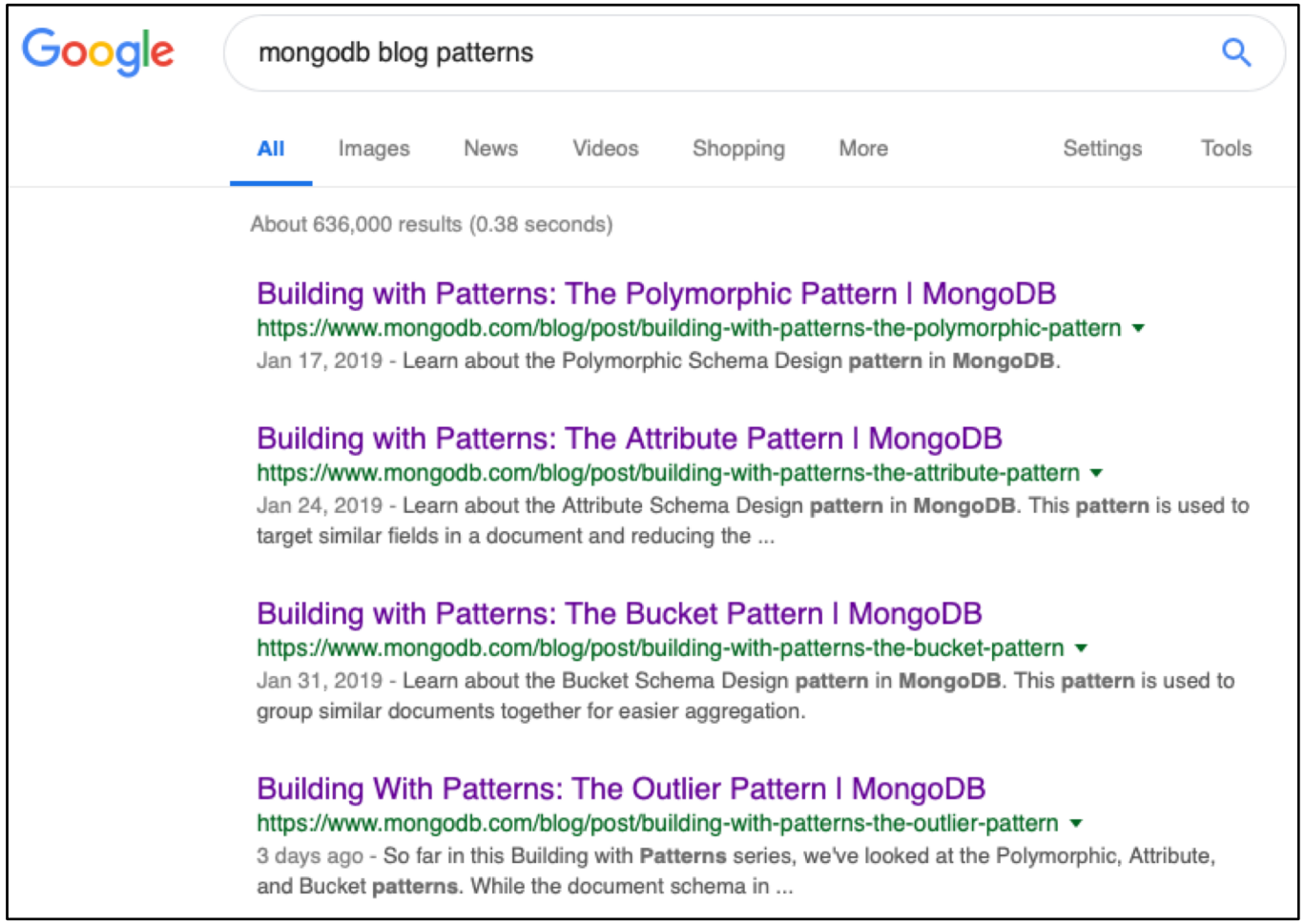
Syntax

```
with client.start_session() as s:
    s.start_transaction()
    collection_one.insert_one(doc_one, session=s)
    collection_two.insert_one(doc_two, session=s)
    s.commit_transaction()
```

Natural for developers

- Idiomatic to the programming language
- Familiar to relational developers
- Simple

Patterns Blog Posts



mongoDB.

MongoDB Atlas

11:00 – 12:00

MongoDB 101 Workshop

Introduction to MongoDB (30 mins)

Setting up MongoDB (30 mins)
- Laptop: Standalone Engine & Shell, Compass, MongoDB Driver (Python)
- Data Generator: MGenerateJS
- Cloud: MongoDB Atlas account

Querying, Indexing and Aggregation (30 mins)
- Using Compass, MongoDB Shell & Python

Schema Design & Use Cases (30 mins)
- Embedding & Referencing, 1-1, 1-M, M-M, Patterns
- Single View, IoT, Time Series, eCommerce/Content Management

MongoDB Atlas (60 mins)
- Management, Clusters, Security
- High Availability & Scalability (Replication & Sharding)
- Managed BI Connector & Charts
- ODBC for BI Connector
- MongoDB Stitch*

mongoDB.

Atlas

unlocks **agility** and **reduces cost**

Self-service and elastic

Global and highly available

Secure by default

Comprehensive monitoring

Managed backup

Cloud agnostic

MongoDB Atlas: Database as a Service

Self-service and elastic

- Deploy in minutes
- Scale up/down without downtime
- Automated upgrades

Global and highly available

- 59 Regions worldwide
- Replica sets optimized for availability
- Global clusters for read/write everywhere deployments

Secure by default

- Network isolation and Peering
- Encryption in flight and at rest with key management
- Role-based access control; auditing
- SOC 2 / Privacy Shield / HIPAA

Comprehensive Monitoring

- Performance Advisor
- Dashboards w/ 100+ metrics
- Real Time Performance
- Customizable alerting

Managed Backup

- Point in Time Restore
- Queryable backups
- Consistent snapshots

Cloud Agnostic

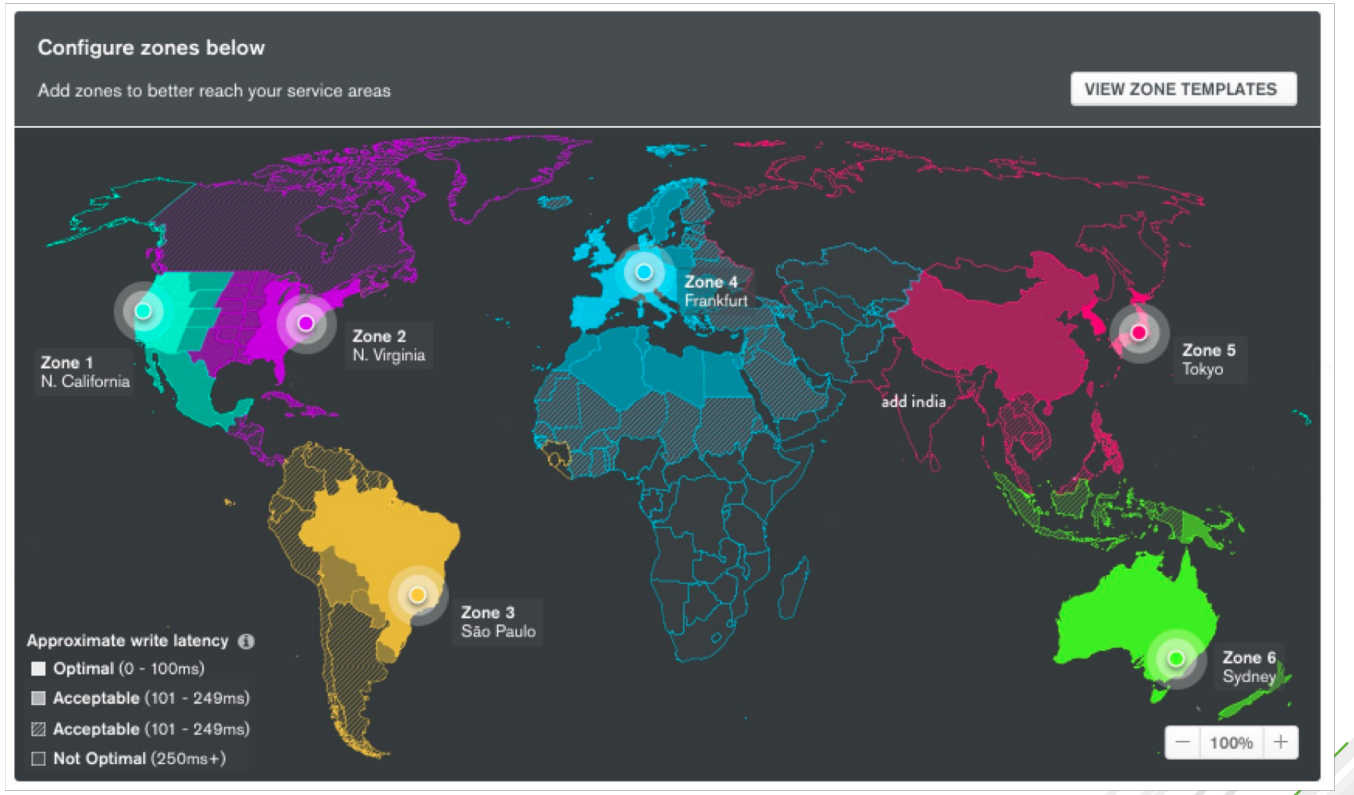
- AWS, Azure, and GCP
- Easy migrations
- Consistent experience

mongoDB

What's New: Atlas Global Clusters

Distribute your fully automated database across multiple geographically distributed *zones* made up of one or more cloud regions

- Read and write locally to provide single-digit millisecond latency for your distributed applications
- Ensure that data from certain geographies lives in predefined zones
- Easily deploy using prebuilt zone templates or build your own zones by choosing cloud regions in an easy-to-use, visual interface

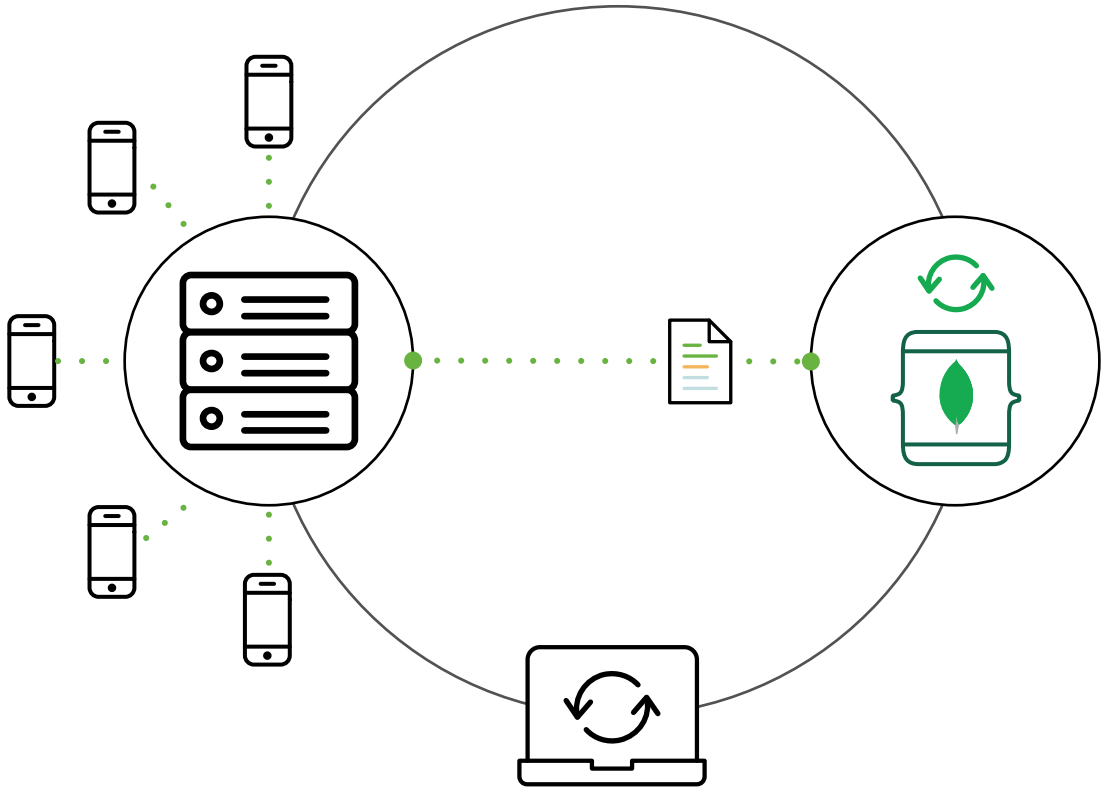


mongoDB

What's New: MongoDB Atlas Enterprise Security

- LDAP integration
- Database level auditing
- Encrypted storage engine: BYO Key Management Service (AWS initially)
- HIPAA Eligible

Freedom to run Anywhere: Cloud Migration



Migrate existing deployments running anywhere into MongoDB Atlas with minimal impact to your application.
Live migration works by:

- Performing a sync between your source database and a target database hosted in MongoDB Atlas
- Syncing live data between your source database and the target database by tailing the oplog
- Notifying you when it's time to cut over to the MongoDB Atlas cluster

mongoDB. Atlas: The only *true* multi-cloud database as a service

Database that runs the same everywhere

Consistent experience across AWS, Azure, and GCP

Coverage in any geography

Deploy in 59 regions worldwide

Create globally distributed databases with a few clicks

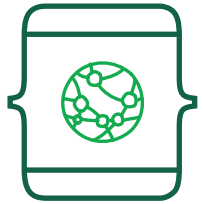
Leverage the benefits of a multi-cloud strategy

Exploit the benefits of AWS, Azure, or GCP services on your data

Avoid lock-in

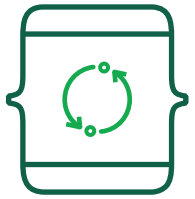
Easily migrate data between cloud providers

Intelligently put data where you need it



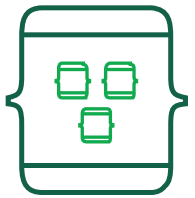
Highly Available

Built-in **multi-region** high availability, replication & automated failover



Workload Isolation

Ability to run both **operational & analytics workloads** on same cluster, for timely insight and lower cost



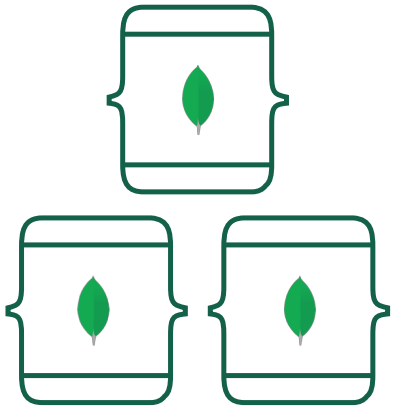
Scalability

Elastic horizontal scalability – add/remove capacity dynamically **without downtime**

Locality

Declare **data locality rules** for governance (e.g. data sovereignty), class of service & local low latency access

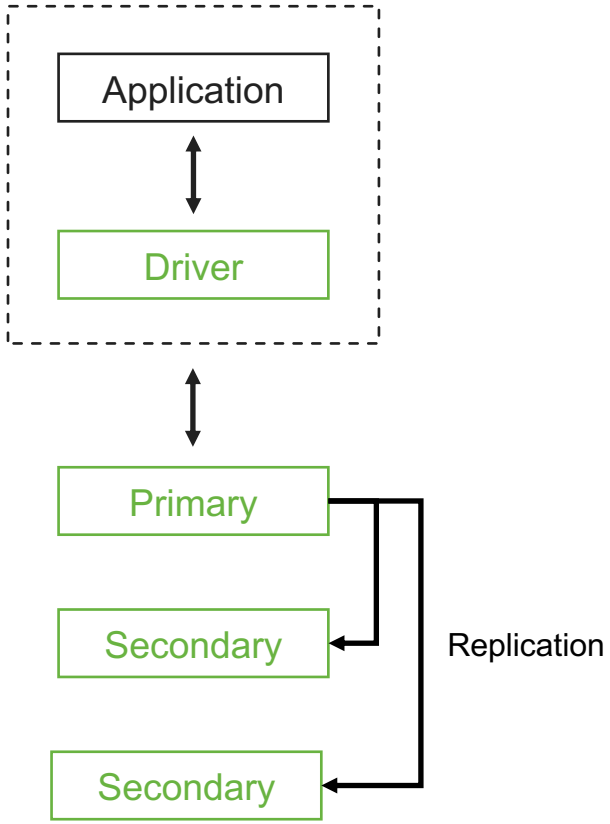
Put data where you need it: Availability



Replica Sets

- Up to 50 replicas, distributed across racks, data centers, and regions
- Self-healing using RAFT-based consensus protocol for automated failover & recovery
- Tunable durability and consistency controls
- Always-on write availability with retryable writes

MongoDB Replica Set Architecture



Replica Set – 2 to 50 copies

Self-healing

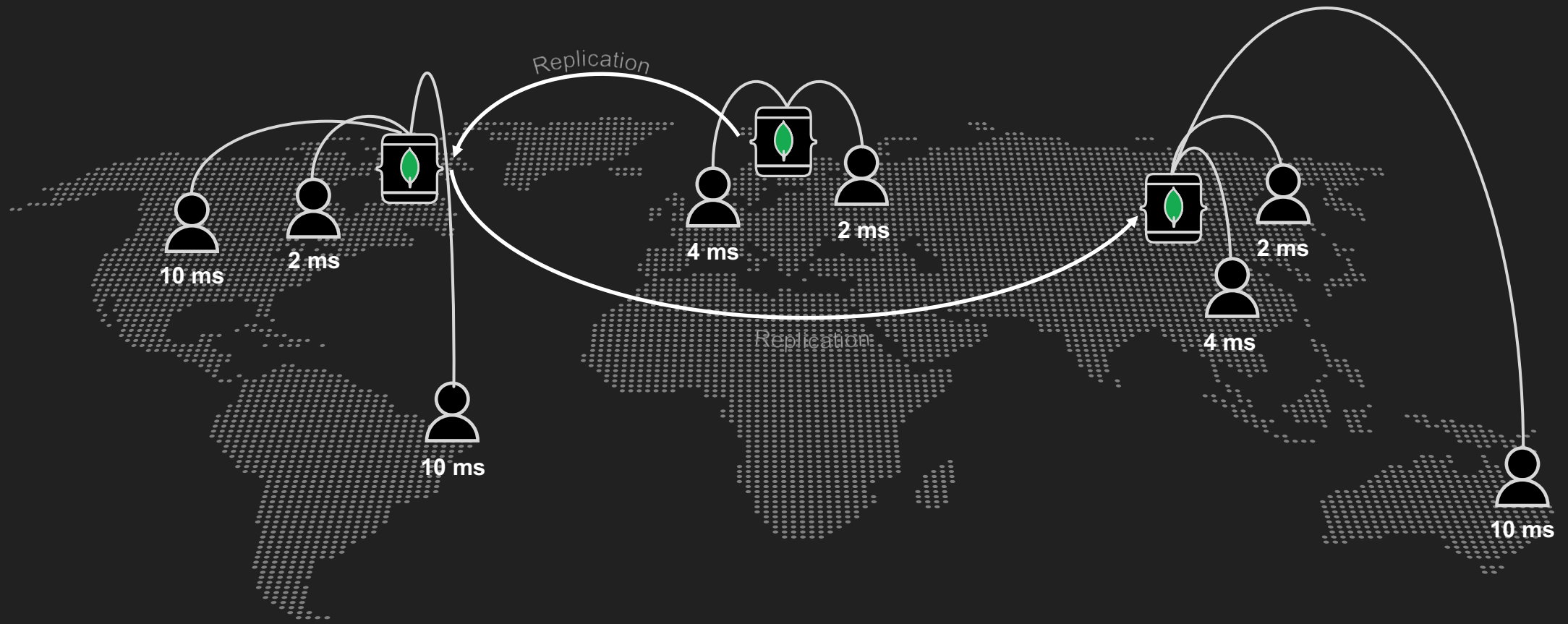
Data Center Aware

Addresses availability considerations:

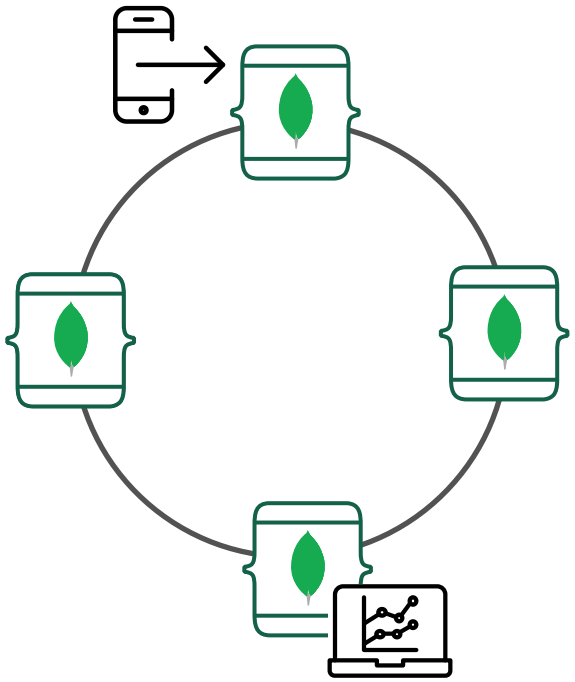
- High Availability
- Disaster Recovery
- Maintenance

Workload Isolation: operational & analytics

MongoDB's native replica sets puts your entire database right next to users



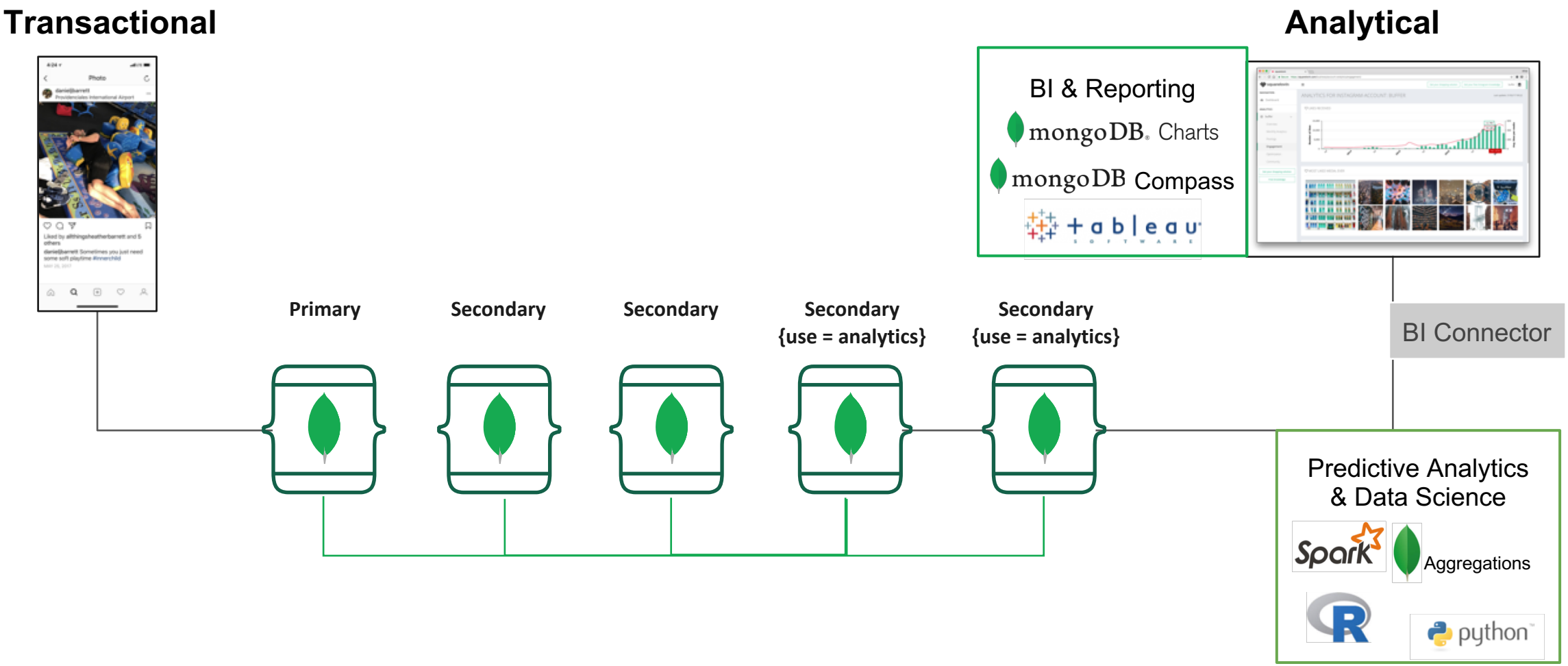
Put data where you need it: Workload Isolation



Enable different workloads on the same data

- Combine operational and analytical workloads on a single data platform
- Extract live insights from real-time data to enrich applications
- One set of nodes serving operational apps, replicating to dedicated nodes serving analytics: up to 50 nodes in a single replica set
- ETL-free

Co-locating operational and analytical workloads



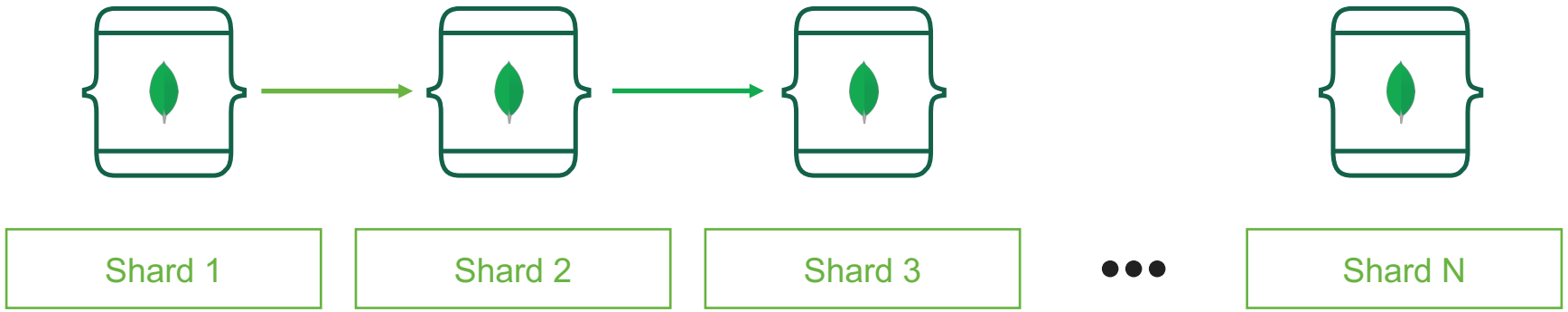
Technical Implementation

```
// Set up replica set
rs.initiate(
{
 "_id": "rs1",
 "version": 1,
 "members":
  [
    {"priority": 1, "host": "localhost:27001", "_id": 0, "tags": {"use": "op"}},
    {"priority": 1, "host": "localhost:27002", "_id": 1, "tags": {"use": "op"}},
    {"priority": 1, "host": "localhost:27003", "_id": 2, "tags": {"use": "op"}},
    {"priority": 0, "host": "localhost:27004", "_id": 3, "tags": {"use": "analytics"}},
    {"priority": 0, "host": "localhost:27005", "_id": 4, "tags": {"use": "analytics"}}]
})

// Force reads to analytics servers
read_client = MongoClient(host = 'localhost:27001', replicaset = 'rs1',
        readPreference='secondary', readPreferenceTags = ["use:analytics"])
```


mongoDB.

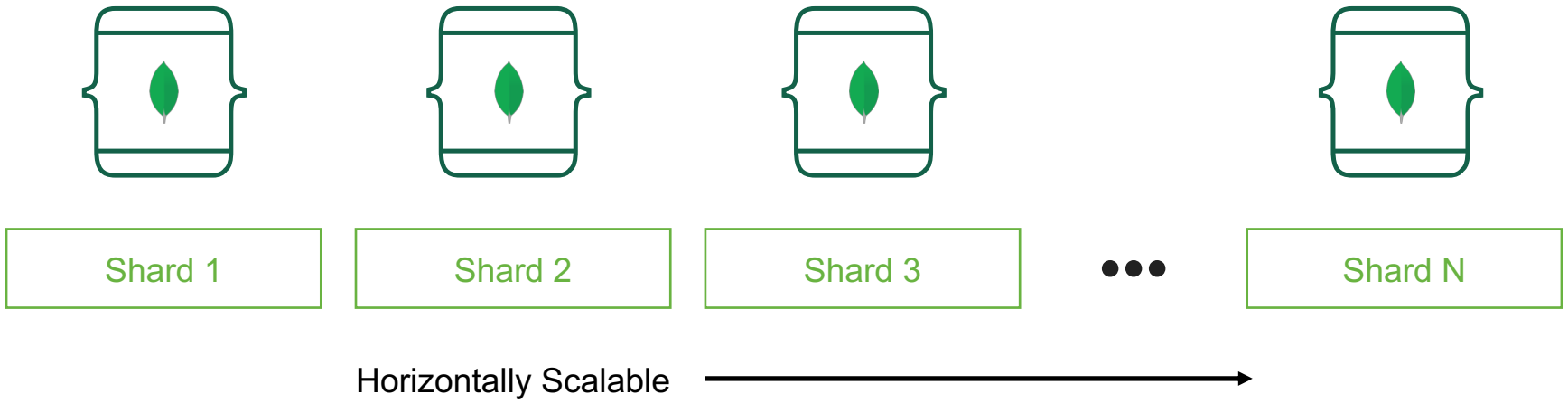
Put data where you need it: Scalability



Auto-Sharding

- Automatically scale beyond the constraints of a single node
- Application transparent
- Scale and rebalance incrementally, in real time
- Unlike NoSQL systems that randomly spray data across a cluster, MongoDB exposes multiple data distribution policies to optimize for query patterns and locality

Scaling MongoDB: Automatic Sharding

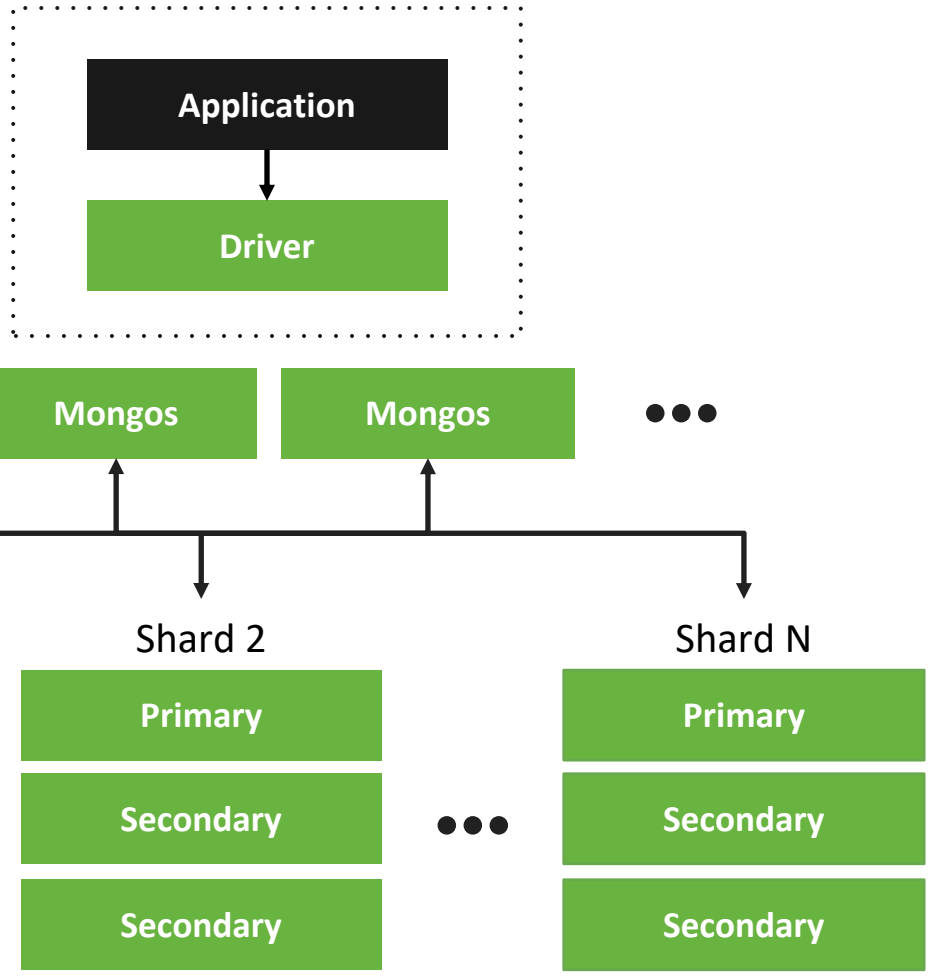


Multiple sharding policies: hashed, ranged, zoned

Increase or decrease capacity as you go

Automatic balancing for elasticity

Sharding Architecture



High availability

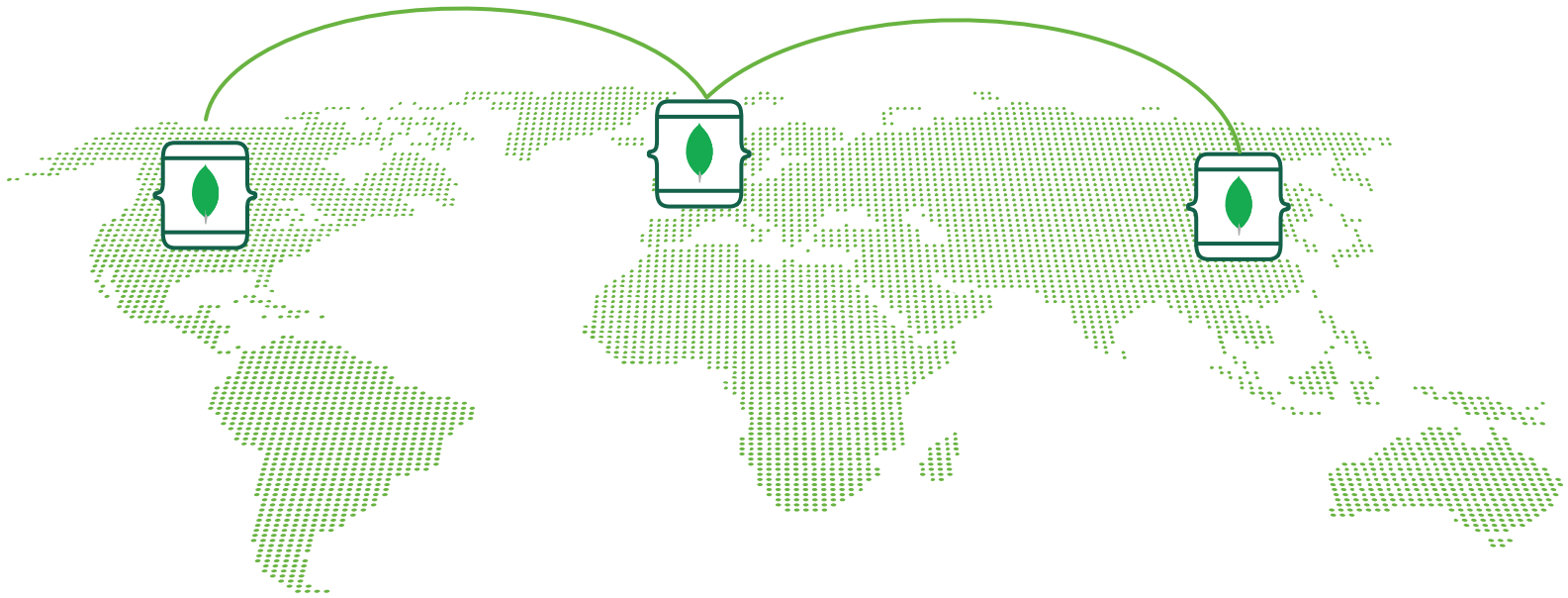
- Replica sets

Horizontal scalability

- Sharding

mongoDB.

Put data where you need it: Locality



Governance

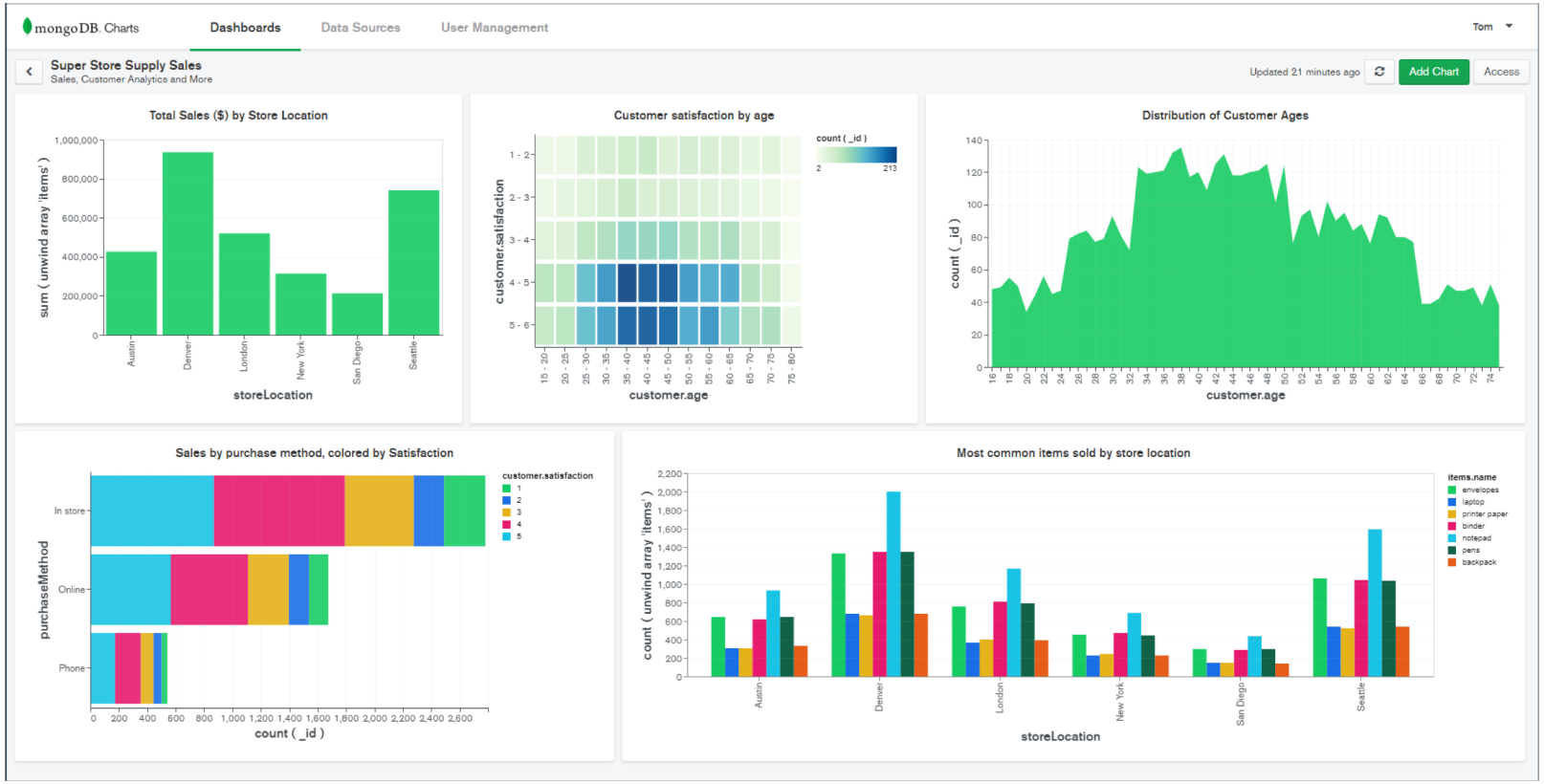
Class of Service

Latency

Intelligent Distribution via Zoned Sharding

- Policies to define data placement
- Name a server by region, tag your documents by region and MongoDB does the rest
- Documents automatically migrated as shard key ranges are modified
- Global queries across all data in all zones

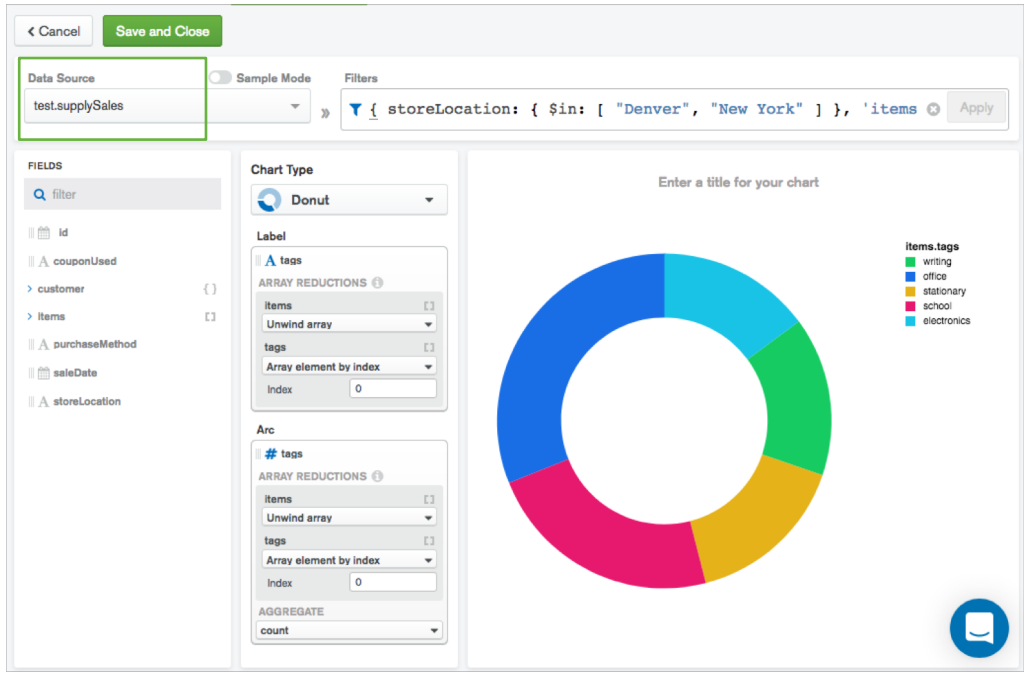
Sophisticated Analytics & Visualizations Of Data In Place



- Rich MongoDB query language & idiomatic drivers
- Connector for BI
- Connector for Spark
- Charts (beta)

mongoDB.

MongoDB Charts: Create, Visualize, Share



Work with complex data

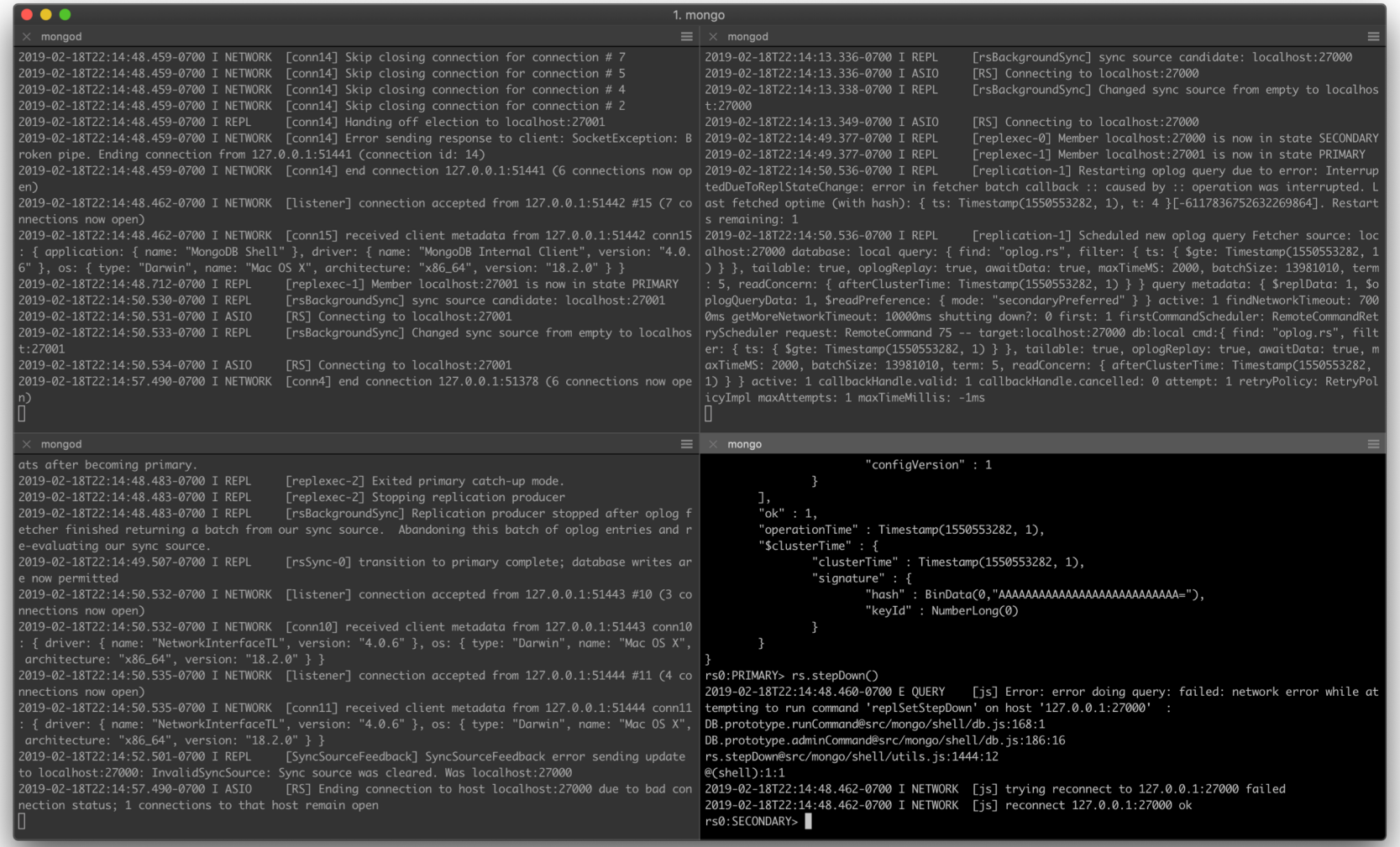
Connect to data sources securely.
Filter. Sample. Visualize.

Share dashboards and collaborate

mongoDB

MongoDB Local Replica Set Workshop

- **Replica-set.txt**



MongoDB Atlas Workshop

- **Setup M0 Cluster**
- Insert data: Customer Single View Data Generator / MongoImport / IP Whitelist
- Enable Charts
- Build a dashboard
- Repeat previous exercises against Atlas (shell or python)

Optional (credit card required)

- **Scale-up to M10**
- Enable BI Connector
- Setup ODBC Connector
- Create dashboard using Tableau, QlikView, etc.

mongoDB®

FOR GIANT IDEAS