WELCOME

MongoDB Atlas

Developer Data Platform Workshop

David Hiltenbrand - Solutions Architect
Memphis, TN

Soheyl Rafi - Solutions Architect
Atlanta, GA

Ian Paeth - Enterprise Account Executive
Atlanta, GA

Agenda

Setting up Atlas

Downloading Compass

CRUD Operations

Aggregations

Charts/ Search / Vector Search

Repo

https://github.com/srafimdb/AtlasWorkshop/blob/main/README.md

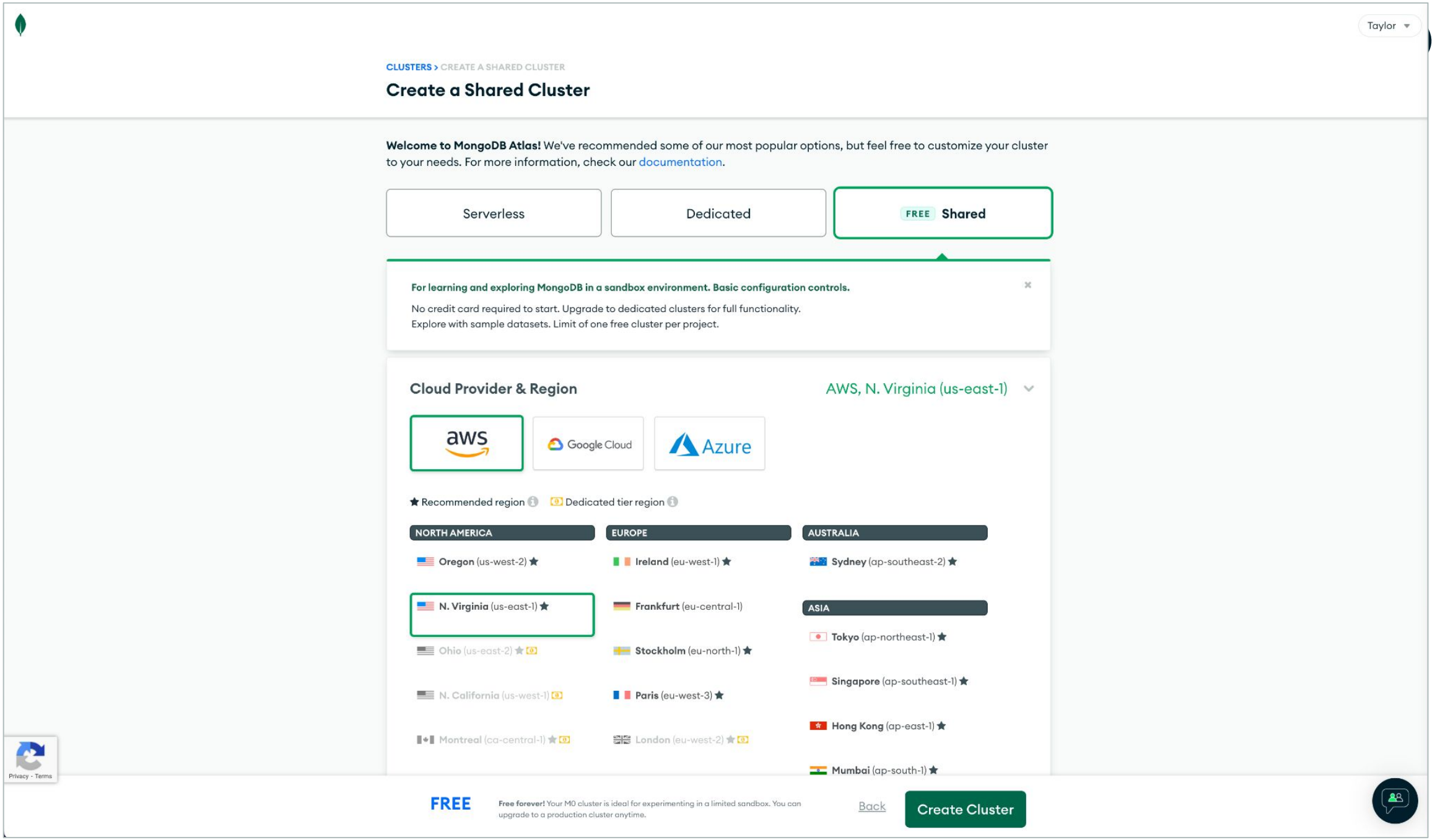
or

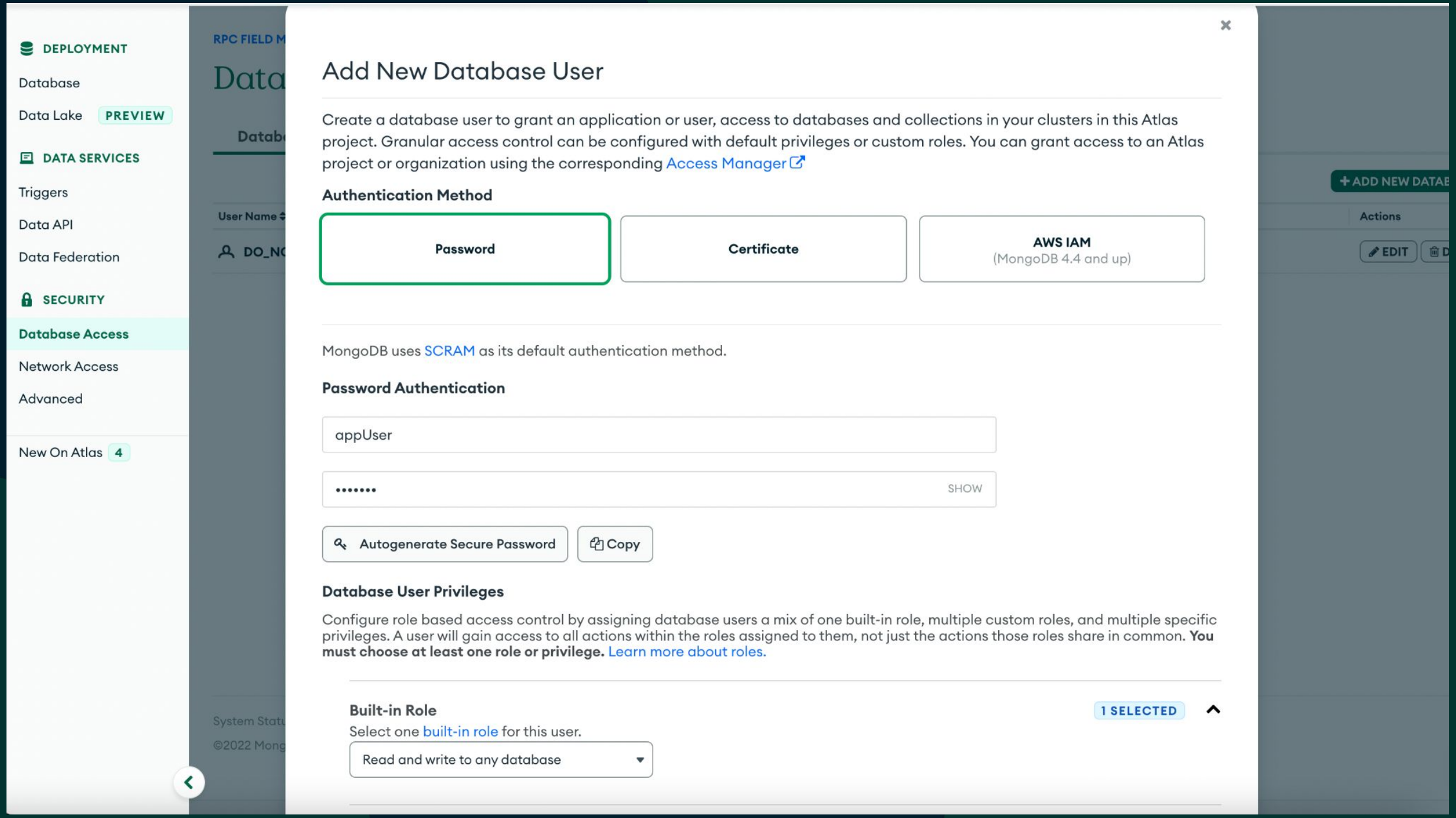
https://shorturl.at/lxFU1

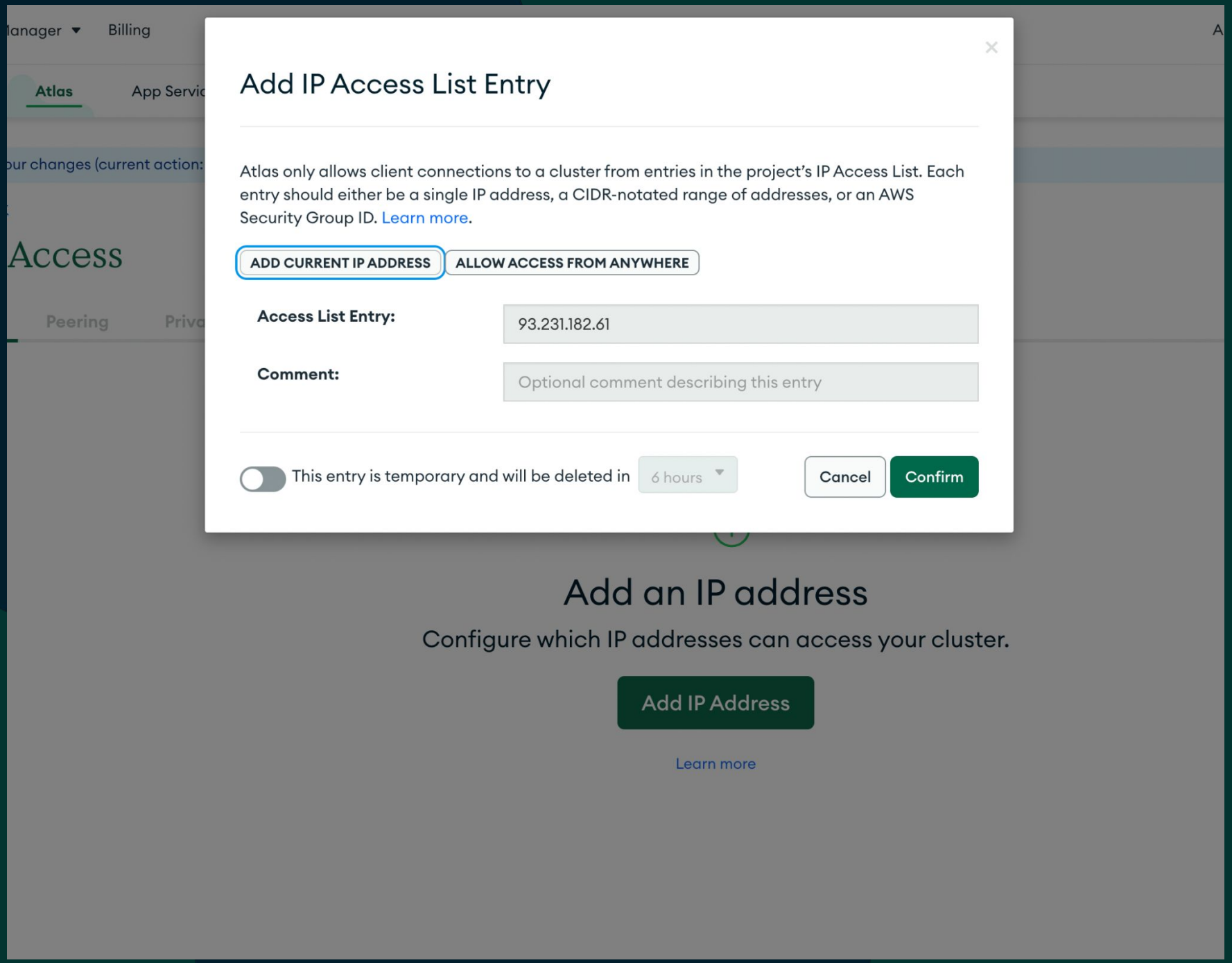
Now it's your turn to try MongoDB Atlas

Exercise One

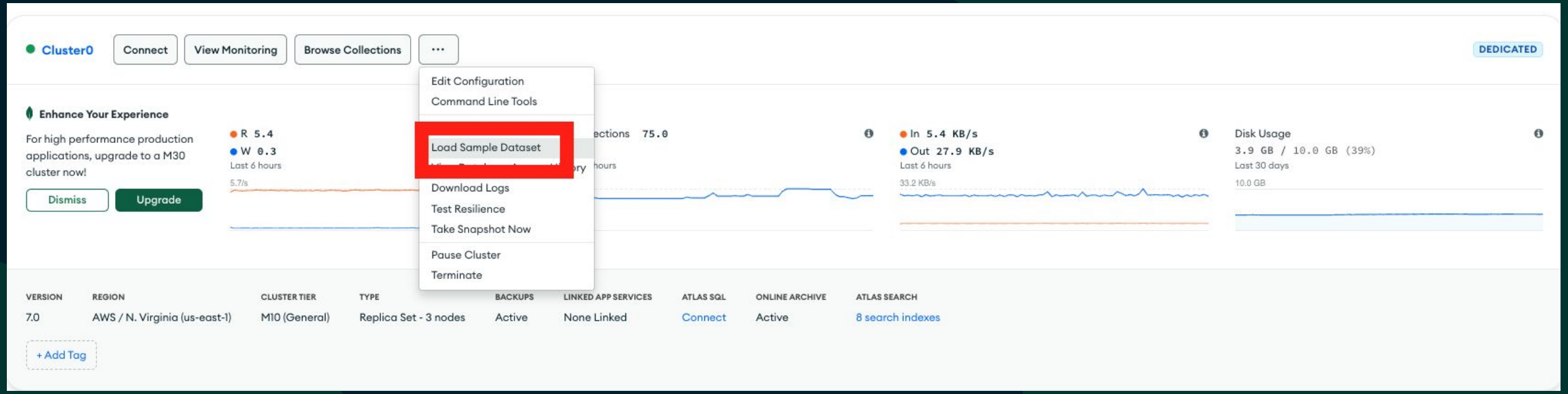
MongoDB Atlas Demo


Taylor
CLUSTERS > CREATE A SHARED CLUSTER
Create a Shared Cluster
Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our documentation.
Serverless
Dedicated
FREE Shared
For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.
No credit card required to start. Upgrade to dedicated clusters for full functionality.
Explore with sample datasets. Limit of one free cluster per project.
Cloud Provider & Region
AWS, N. Virginia (us-east-1)
aws
Google Cloud
Azure
★ Recommended region
Dedicated tier region
NORTH AMERICA
Oregon (us-west-2) ★
N. Virginia (us-east-1) ★
Ohio (us-east-2) ★
N. California (us-west-1)
Montreal (ca-central-1) ★
EUROPE
Ireland (eu-west-1) ★
Frankfurt (eu-central-1)
Stockholm (eu-north-1) ★
Paris (eu-west-3) ★
London (eu-west-2) ★
AUSTRALIA
Sydney (ap-southeast-2) ★
ASIA
Tokyo (ap-northeast-1) ★
Singapore (ap-southeast-1) ★
Hong Kong (ap-east-1) ★
Mumbai (ap-south-1) ★
Privacy - Terms
FREE
Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.
Back
Create Cluster


DEPLOYMENT
Database
Data Lake PREVIEW
DATA SERVICES
Triggers
Data API
Data Federation
SECURITY
Database Access
Network Access
Advanced
New On Atlas 4
Add New Database User
Create a database user to grant an application or user, access to databases and collections in your clusters in this Atlas project. Granular access control can be configured with default privileges or custom roles. You can grant access to an Atlas project or organization using the corresponding Access Manager
Authentication Method
Password
Certificate
AWS IAM
(MongoDB 4.4 and up)
MongoDB uses SCRAM as its default authentication method.
Password Authentication
appUser
SHOW
Autogenerate Secure Password
Copy
Database User Privileges
Configure role based access control by assigning database users a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. You must choose at least one role or privilege. Learn more about roles.
Built-in Role
1 SELECTED
Select one built-in role for this user.
Read and write to any database


Billing
Atlas
Access
Peering
Add IP Access List Entry
Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address, a CIDR-notated range of addresses, or an AWS Security Group ID. Learn more.
ADD CURRENT IP ADDRESS
ALLOW ACCESS FROM ANYWHERE
Access List Entry:
93.231.182.61
Comment:
Optional comment describing this entry
This entry is temporary and will be deleted in
6 hours
Cancel
Confirm
Add an IP address
Configure which IP addresses can access your cluster.
Add IP Address
Learn more

Load Sample Dataset



Download Compass

https://www.mongodb.com/try/download/compass

MongoDB Compass

Easily explore and manipulate your database with Compass, the GUI for MongoDB. Intuitive and flexible, Compass provides detailed schema visualizations, real-time performance metrics, sophisticated querying abilities, and much more.

Please note that MongoDB Compass comes in three versions: **a full version** with all features, **a read-only version** without write or delete capabilities, and **an isolated edition**, whose sole network connection is to the MongoDB instance.

For more information, see our documentation pages.

- **Compass**

  The full version of MongoDB Compass, with all features and capabilities.

- **Readonly Edition**

  This version is limited strictly to read operations, with all write and delete capabilities removed.

- **Isolated Edition**

  This version disables all network connections except the connection to the MongoDB instance.

Available Downloads

Version: 1.33.1 (Stable)

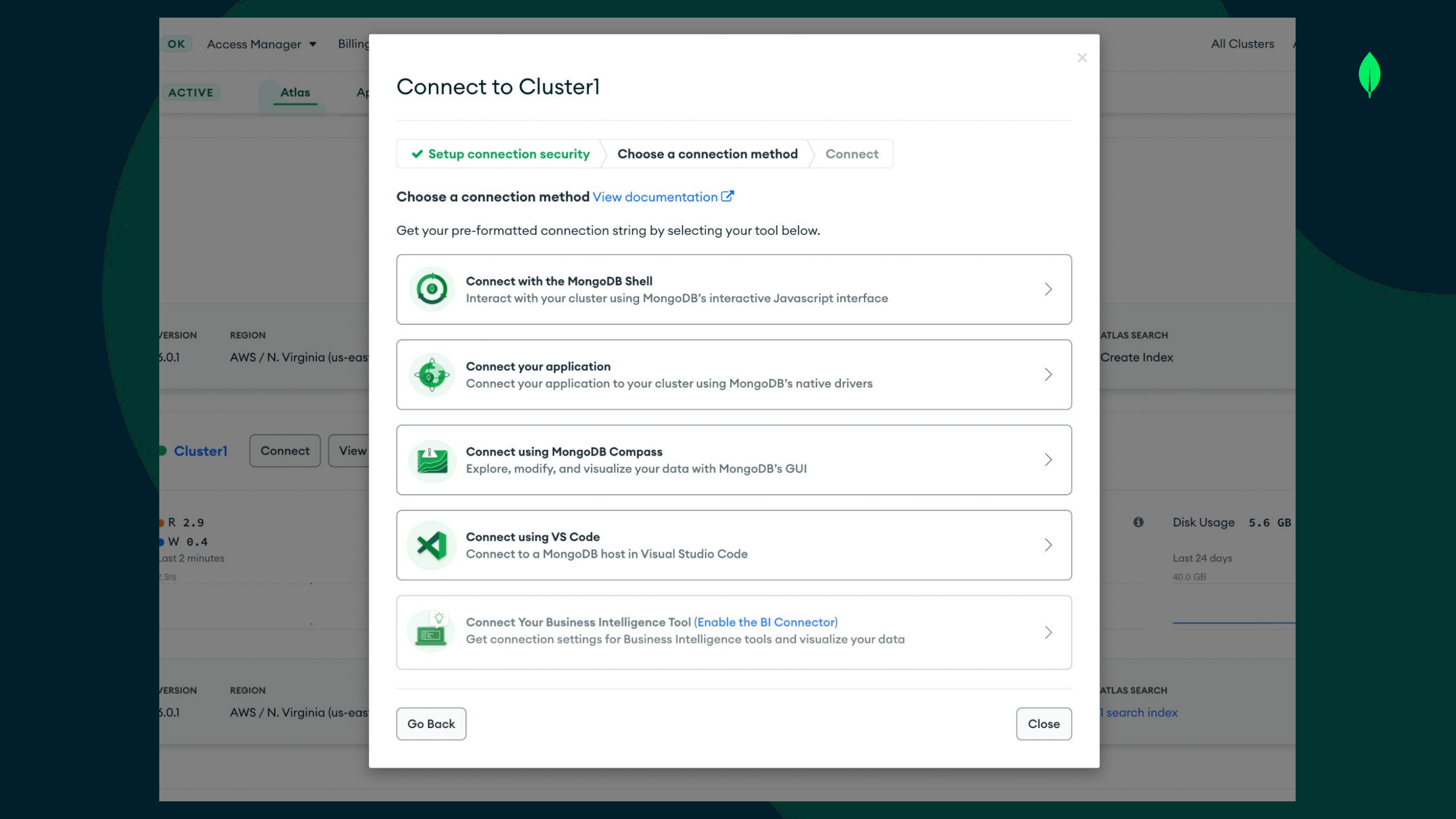
Platform: macOS 64-bit (10.14+)

Package: dmg

Download | Copy Link

Documentation

Archived releases


OK
Access Manager
Billing
All Clusters
ACTIVE
Atlas
Connect to Cluster1
Setup connection security
Choose a connection method
Connect
Choose a connection method View documentation
Get your pre-formatted connection string by selecting your tool below.
Connect with the MongoDB Shell
Interact with your cluster using MongoDB's interactive Javascript interface
Connect your application
Connect your application to your cluster using MongoDB's native drivers
Connect using MongoDB Compass
Explore, modify, and visualize your data with MongoDB's GUI
Connect using VS Code
Connect to a MongoDB host in Visual Studio Code
Connect Your Business Intelligence Tool (Enable the BI Connector)
Get connection settings for Business Intelligence tools and visualize your data
Go Back
Close
VERSION
REGION
AWS / N. Virginia (us-east
ATLAS SEARCH
Create Index
Cluster1
Connect
View
R 2.9
W 0.4
Last 2 minutes
Disk Usage 5.6 GB
Last 24 days
40.0 GB
search index

Exercise Two

MongoDB Data GUI & CRUD

Developer tools ecosystem

Build Applications
IDE Integrations
VS Code for MongoDB

Navigate & explore your data
MongoDB Compass

Interface with MongoDB
MongoDB Shell

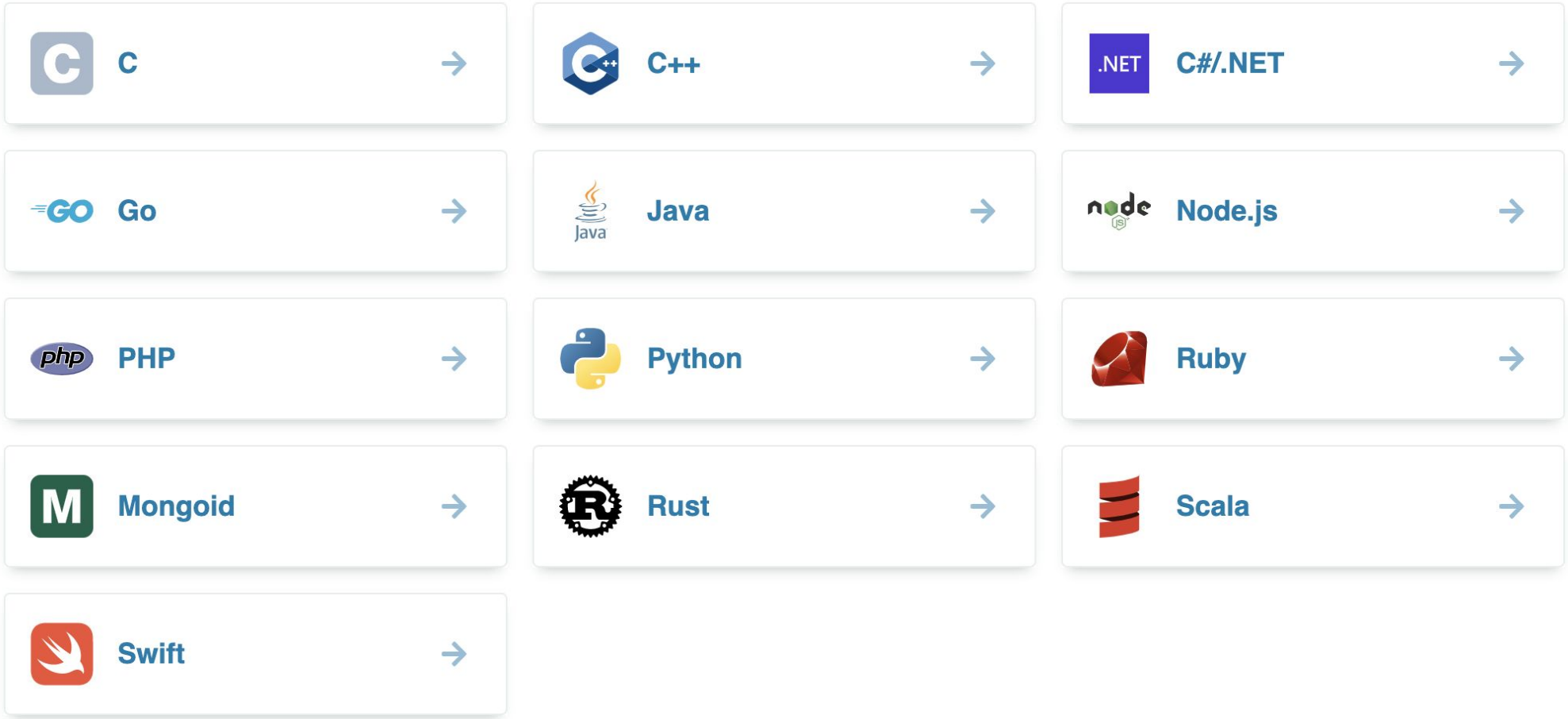
MongoDB Query API: rich and expressive

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"
         }
     ]
}
```

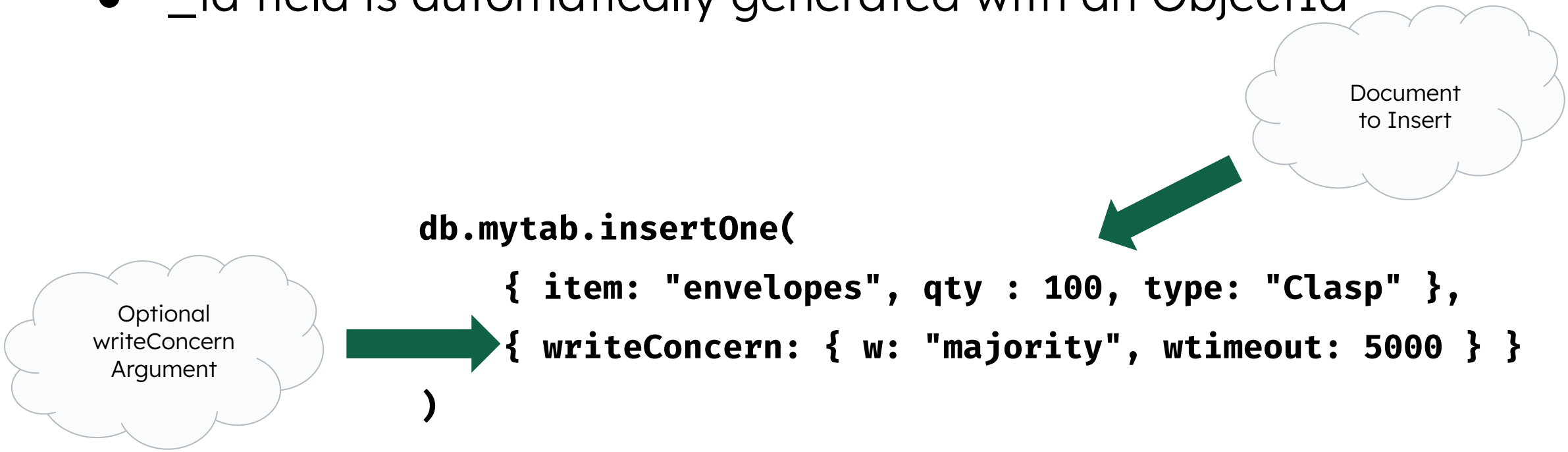
Intuitive: client drivers



- Common CRUD capabilities but idiomatic to each language
- Uniform HA & Failover capabilities across all

insertOne()

- Inserts document(s) into a collection
- Can optionally define the writeConcern of the operation
- _id field is automatically generated with an ObjectId



find() and findOne()

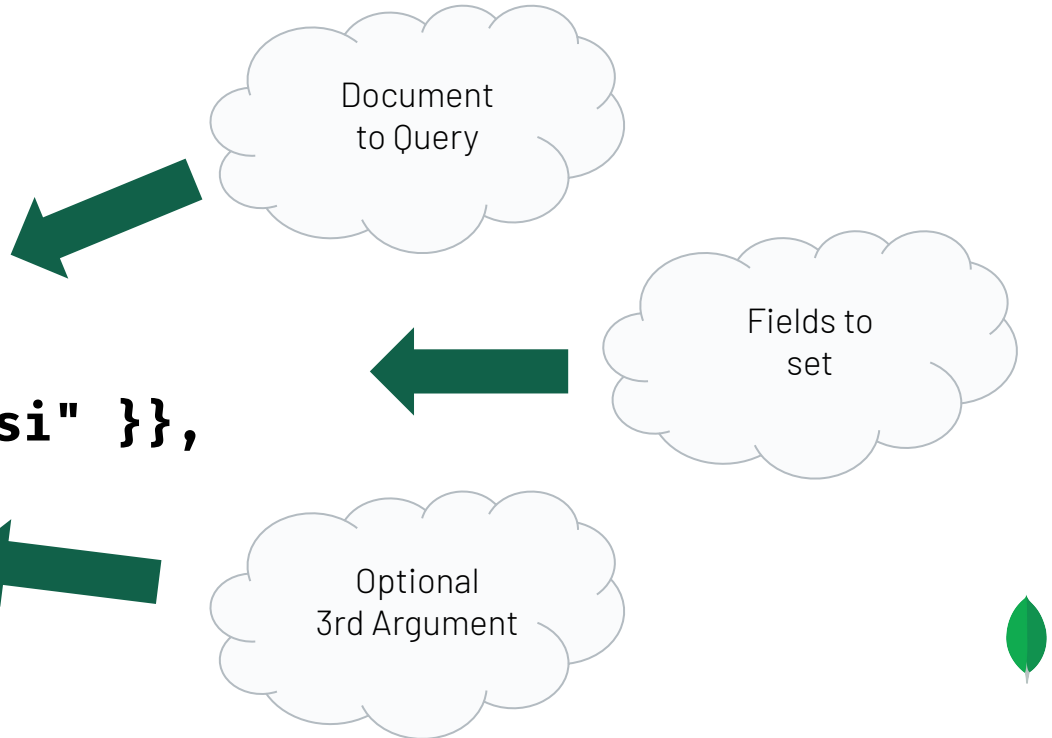
- Get documents from the collection (table)
- Can filter by content (query) and by what is returned (project)
- `findOne()` returns just the first document

```
db.mytab.find({x:1, y:2}, {a:1, b:1, c:1})
```

updateMany() and updateOne()

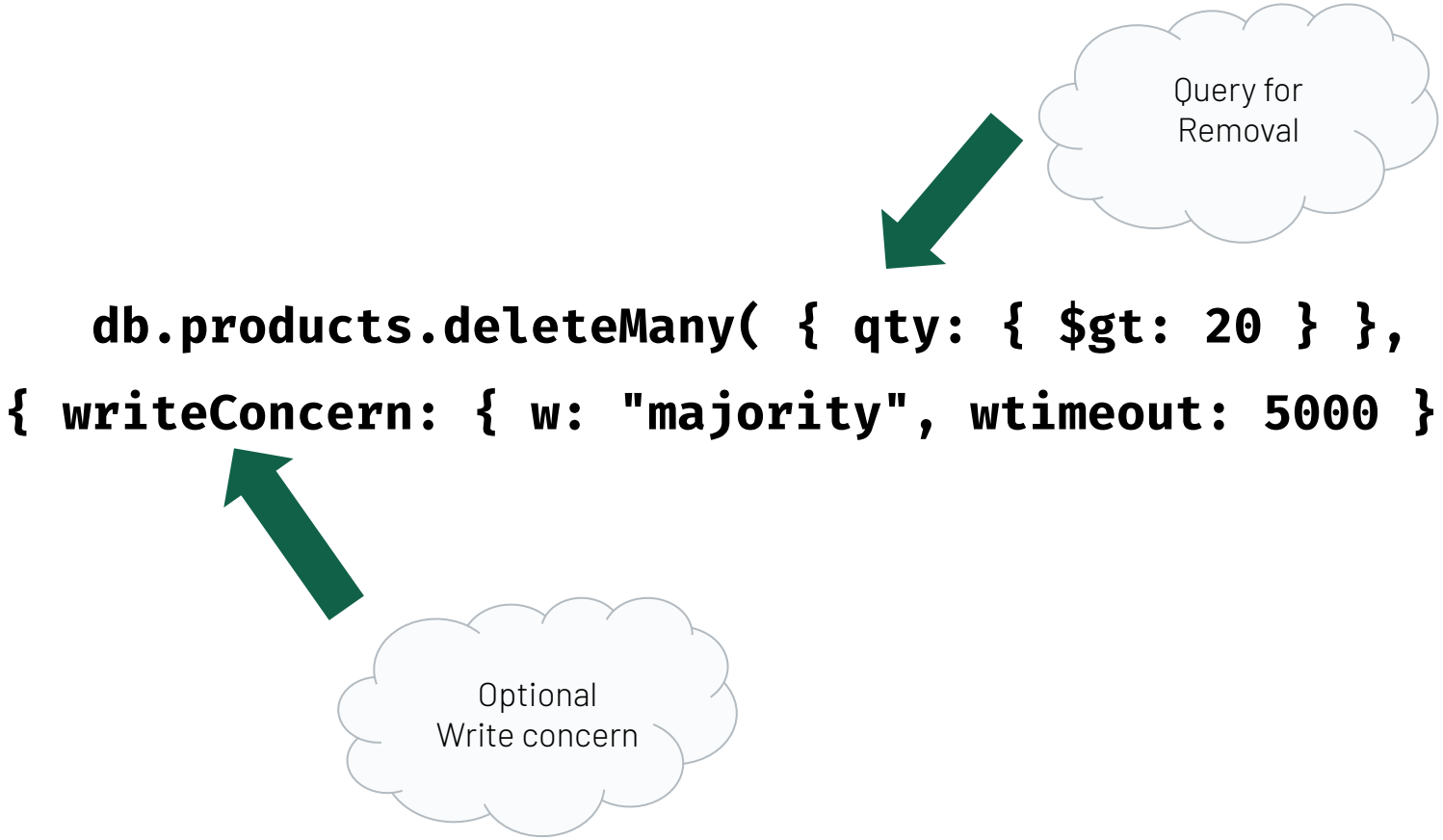
- Modifies existing document(s) in a collection
- Use with a field update operator (MQL)
- Query for the documents you want to update
- Can optionally upsert, multi-update, define writeConcern, etc.

```
db.books.updateOne(
    { author: “Roger Federer” },
    { $set: { author: "Lionel Messi" }},
 { multi: true }
)
```



deleteMany()

- Deletes a document from the collection
- Takes an optional writeConcern argument



Comparison Query Operators

$lt	:	Exists and less than
$lte	:	Exists and less than or equal to
$gt	:	Exists and greater than
$gte	:	Exists and greater than or equal to
$ne	:	Does not exist or does but not equal to
$in	:	Exists and in a set
$nin	:	Does not exist or not in a set

```
{ age : { $gte : 21 }}
{ temp : { $gt: 10, $lt: 30 }}
```

Field Update Operators (well some…)

$currentDate	:	Sets the value of a field to current date
$inc	:	Increments value
$min	:	Update if field is less than $min value
$max	:	Update if field is greater than $max value
$mul	:	Multiply field value
$set	:	Set value of field
$unset	:	Removes field from document

```
{ $set : { bestDB : “MongoDB”}}
{ $mul: { amount : 5, ... } }
```

We will now be using

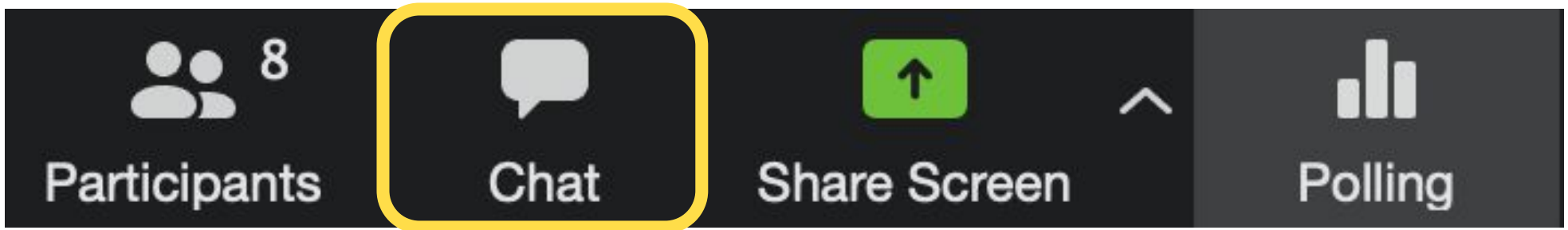
sample_airbnb.listingsAndReviews

& sample_mflix.movies

database/collection from now on

JEOPARDY!

Please answer via the **Chat**



We will do the first one together now.

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	... from 1987
`db.movies.find({genres: "Comedy"})`	

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre
`db.movies.find({genres: {$in:["Comedy", "Drama"]}})`	

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre
`db.movies.find({genres: {$in:["Comedy", "Drama"]}})`	"Comedy" or "Drama" as the genre

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre
`db.movies.find({genres: {$in:["Comedy", "Drama"]}})`	"Comedy" or "Drama" as the genre
`db.movies.find({revenues: {$exists: true}})`	

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre
`db.movies.find({genres: {$in:["Comedy", "Drama"]}})`	"Comedy" or "Drama" as the genre
`db.movies.find({revenues: {$exists: true}})`	Documents that have a revenue field

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre
`db.movies.find({genres: {$in:["Comedy", "Drama"]}})`	"Comedy" or "Drama" as the genre
`db.movies.find({revenues: {$exists: true}})`	Documents that have a revenue field
`db.movies.find({imdb.rating: {$gt: 8.0}, rated: "PG"})`	

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre
`db.movies.find({genres: {$in:["Comedy", "Drama"]}})`	"Comedy" or "Drama" as the genre
`db.movies.find({revenues: {$exists: true}})`	Documents that have a revenue field
`db.movies.find({imdb.rating: {$gt: 8.0}, rated: "PG"})`	IMDB Rating > 8.0 and PG Rating

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre
`db.movies.find({genres: {$in:["Comedy", "Drama"]}})`	"Comedy" or "Drama" as the genre
`db.movies.find({revenues: {$exists: true}})`	Documents that have a revenue field
`db.movies.find({imdb.rating: {$gt: 8.0}, rated: "PG"})`	IMDB Rating > 8.0 and PG Rating
`db.movies.find({title: {$regex: '^Dr. Strangelove'}})`	

CRUD: Jeopardy

Query	What's the Question?
`db.movies.find({year: 1987})`	… from 1987
`db.movies.find({genres: "Comedy"})`	"Comedy" as one of their genres
`db.movies.find({genres: ["Comedy"]})`	"Comedy" as the *only* genre
`db.movies.find({genres: {$in:["Comedy", "Drama"]}})`	"Comedy" or "Drama" as the genre
`db.movies.find({revenues: {$exists: true}})`	Documents that have a revenue field
`db.movies.find({imdb.rating: {$gt: 8.0}, rated: "PG"})`	IMDB Rating > 8.0 and PG Rating
`db.movies.find({title: {$regex: '^Dr. Strangelove'}})`	Title starting with "Dr. Strangelove"

https://shorturl.at/lxFU1

Hands-On: Your turn

sample_airbnb.listingsAndReviews

5.6k DOCUMENTS 4 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

Filter {"address.market":"Hong Kong", "property_type" : "Apartment"} Reset Find </> More Options

ADD DATA EXPORT COLLECTION 1 – 20 of 444

```
_id: "10893326"
listing_url: "https://www.airbnb.com/rooms/10893326"
name: "Designer Apartment in Sheung Wan"
summary: "We live in a beautiful and spacious one bedroom apartment in the heart…"
space: ""
description: "We live in a beautiful and spacious one bedroom apartment in the heart…"
neighborhood_overview: "Often described as the new SOHO, there are chic coffee shops and inter…"
notes: ""
transit: "The area is well served by minibuses, City Bus, taxis, and the old sty…"
access: ""
interaction: ""
house_rules: ""
property_type: "Apartment"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "1"
maximum_nights: "1125"
cancellation_policy: "strict_14_with_grace_period"
last_scraped: 2019-03-11T04:00:00.000+00:00
calendar_last_scraped: 2019-03-11T04:00:00.000+00:00
first_review: 2016-03-06T05:00:00.000+00:00
last_review: 2019-02-11T05:00:00.000+00:00
accommodates: 2
```

```
use sample_airbnb
db.listingsAndReviews.find({"address.mar
ket":"Hong Kong","property_type" :
"Apartment"})
```

sample_airbnb.listingsAndReviews

sample_airbnb.listingsAndReviews

5.6k DOCUMENTS 4 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

Filter { "bedrooms": {"$gte": 3}, "price": {"$gte": 1400, "$lte": 1500}} Reset Find </> More Options

ADD DATA EXPORT COLLECTION 1 - 7 of 7


```
_id: "1176693"
listing_url: "https://www.airbnb.com/rooms/1176693"
name: "BEST REVIEWS*BEST MALLS*SAFE STAY*DIMSUM*CWB*MTR"
summary: "Location is OUTSTANDING as MTR Causeway BAY is within 1 min walk. You …"
space: "FACTS - located in the heart of one of Hong Kong's most bustling place…"
description: "Location is OUTSTANDING as MTR Causeway BAY is within 1 min walk. You …"
neighborhood_overview: "This enigmatic city of skyscrapers, ancient traditions and heavenly fo…"
notes: "Please note that not every neighbor is happy about the New Sharing Eco…"
transit: "HOW TO GET TO CAUSEWAY BAY: By Public transport: MTR: Island Line: Tin…"
access: "The entire apartment with all amenities is accessible for my guests. A…"
interaction: "I will respond to your questions within very short notice. My reservat…"
house_rules: "Early check-in and late check-out is possible for an additional fee. P…"
property_type: "Apartment"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "2"
maximum_nights: "1124"
cancellation_policy: "strict_14_with_grace_period"
last_scraped: 2019-03-11T04:00:00.000+00:00
```


```
use sample_airbnb
db.listingsAndReviews.find({ "bedrooms":
{"$gte": 3}, "price": {"$gte": 1400,
"$lte": 1500}})
```

sample_airbnb.listingsAndReviews

5.6k DOCUMENTS 4 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

Filter { "amenities": { "$all": ["Wifi", "Kitchen"]}} Reset Find </> More Options

ADD DATA EXPORT COLLECTION 1 – 20 of 4767


```
_id: "10047964"
listing_url: "https://www.airbnb.com/rooms/10047964"
name: "Charming Flat in Downtown Moda"
summary: "Fully furnished 3+1 flat decorated with vintage style.  Located at the…"
space: "The apartment is composed of 1 big bedroom with double sized bed, a gu…"
description: "Fully furnished 3+1 flat decorated with vintage style.  Located at the…"
neighborhood_overview: "With its diversity Moda- Kadikoy is one of the most colorfull neighbou…"
notes: ""
transit: ""
access: ""
interaction: ""
house_rules: "Be and feel like your own home, with total respect and love..this woul…"
property_type: "House"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "2"
maximum_nights: "1125"
cancellation_policy: "flexible"
last_scraped: 2019-02-18T05:00:00.000+00:00
calendar_last_scraped: 2019-02-18T05:00:00.000+00:00
first_review: 2016-04-02T04:00:00.000+00:00
last_review: 2016-04-02T04:00:00.000+00:00
accommodates: 6
```


```
use sample_airbnb
db.listingsAndReviews.find({
"amenities": { "$all": ["Wifi",
"Kitchen"]}})
```

sample_airbnb.listingsAndReviews

5.6k DOCUMENTS 4 INDEXES

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

Filter {":": 3}, "price": {"$gte": 1400, "$lte": 1500}, "amenities": { "$all": ["Wifi", "Kitchen"]}} Reset Find </> More Options

ADD DATA | EXPORT COLLECTION 1 – 4 of 4


```
_id: "1176693"
listing_url: "https://www.airbnb.com/rooms/1176693"
name: "BEST REVIEWS*BEST MALLS*SAFE STAY*DIMSUM*CWB*MTR"
summary: "Location is OUTSTANDING as MTR Causeway BAY is within 1 min walk. You …"
space: "FACTS - located in the heart of one of Hong Kong's most bustling place…"
description: "Location is OUTSTANDING as MTR Causeway BAY is within 1 min walk. You …"
neighborhood_overview: "This enigmatic city of skyscrapers, ancient traditions and heavenly fo…"
notes: "Please note that not every neighbor is happy about the New Sharing Eco…"
transit: "HOW TO GET TO CAUSEWAY BAY: By Public transport: MTR: Island Line: Tin…"
access: "The entire apartment with all amenities is accessible for my guests. A…"
interaction: "I will respond to your questions within very short notice. My reservat…"
house_rules: "Early check-in and late check-out is possible for an additional fee. P…"
property_type: "Apartment"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "2"
maximum_nights: "1124"
cancellation_policy: "strict_14_with_grace_period"
last_scraped: 2019-03-11T04:00:00.000+00:00
calendar_last_scraped: 2019-03-11T04:00:00.000+00:00
first_review: 2013-07-04T04:00:00.000+00:00
last_review: 2018-07-29T04:00:00.000+00:00
accommodates: 8
```


```
use sample_airbnb
db.listingsAndReviews.find({"address.mar
ket":"Hong Kong", "property_type" :
"Apartment", "bedrooms": {"$gte": 3},
"price": {"$gte": 1400, "$lte": 1500},
"amenities": { "$all": ["Wifi",
"Kitchen"]}})
```

sample_airbnb.listingsAndReviews

5.6k DOCUMENTS 4 INDEXES

Documents | Aggregations | Schema | **Explain Plan** | Indexes | Validation

Filter {"address.market":"Hong Kong", "property_type" : "Apartment", "bedrooms": {"$gte": 3}, ' Reset Explain </> More Options

VIEW VISUAL TREE RAW JSON

Query Performance Summary Learn more

Documents Returned: **4**

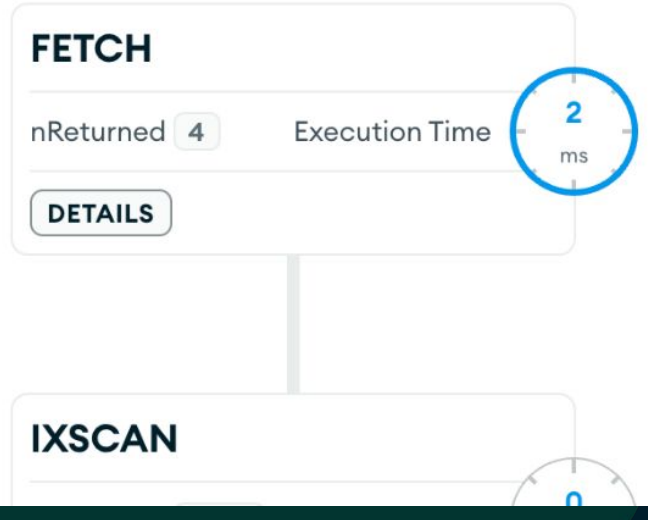
Index Keys Examined: **3626**

Documents Examined: **3626**

Actual Query Execution Time (ms): **14**

Sorted in Memory: **no**

Query used the following index: PROPERTY_TYPE ↑ ROOM_TYPE ↑ BEDS ↑




```
use sample_airbnb
db.listingsAndReviews.find({"address.market":"Hong Kong", "property_type" : "Apartment", "bedrooms": {"$gte": 3}, "price": {"$gte": 1400, "$lte": 1500}, "amenities": { "$all": ["Wifi", "Kitchen"]}}).explain()
```

Exercise Four

Indexes and Explain Plan

At the end of this, you should be able to…

Identify how a query was run using the Explain Plan

Create an index on a collection

See how an index affects query plans

View Winning Plan

```
> db.movies.
find({'scores.score':{$lt:60}}).explain()
```

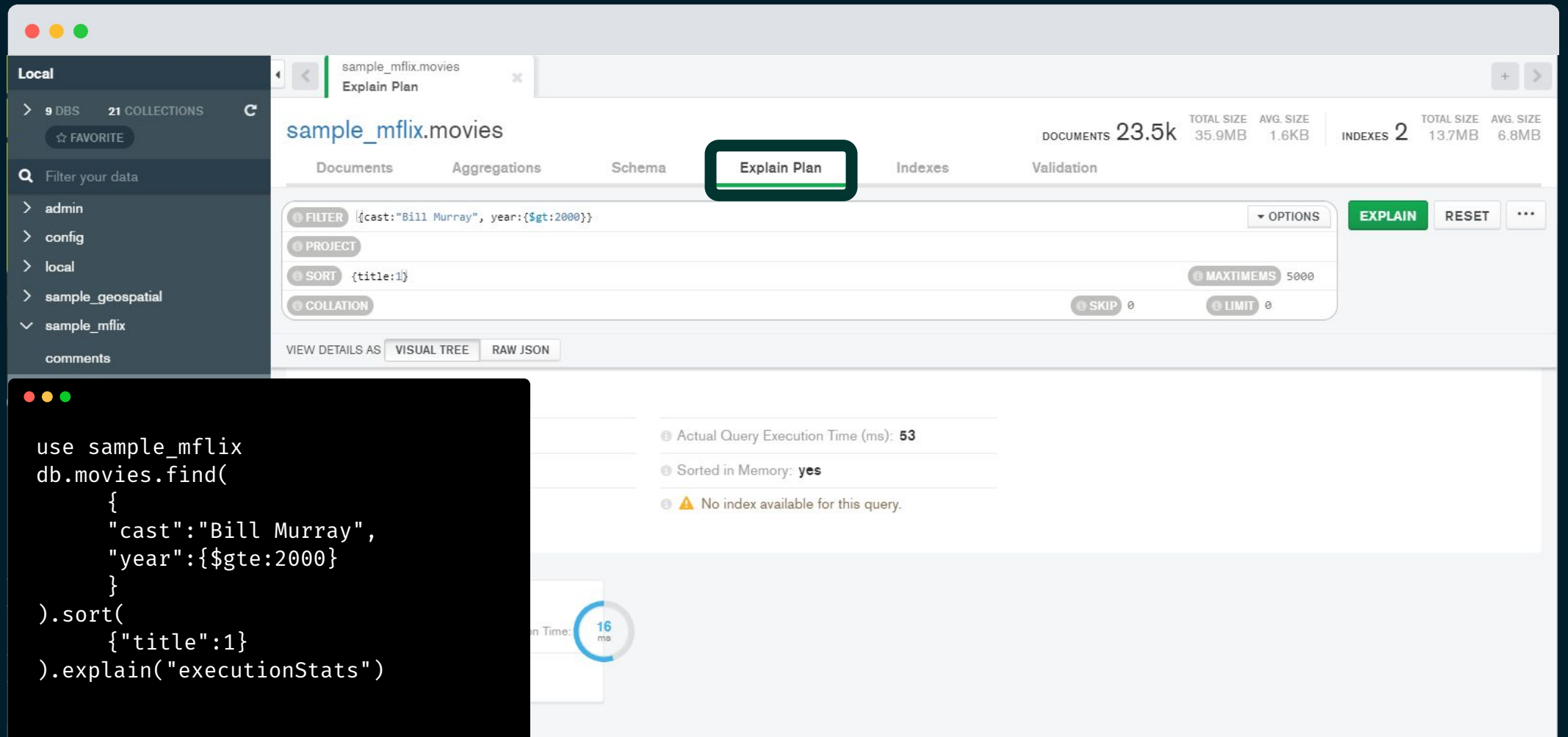
Execution Info

```
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: 'sample_training.grades',
    indexFilterSet: false,
    parsedQuery: { 'scores.score': { '$lt': 60 } },
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { 'scores.type': { '$lt': 60 } },
      direction: 'forward'
    },
    rejectedPlans: []
  }
```

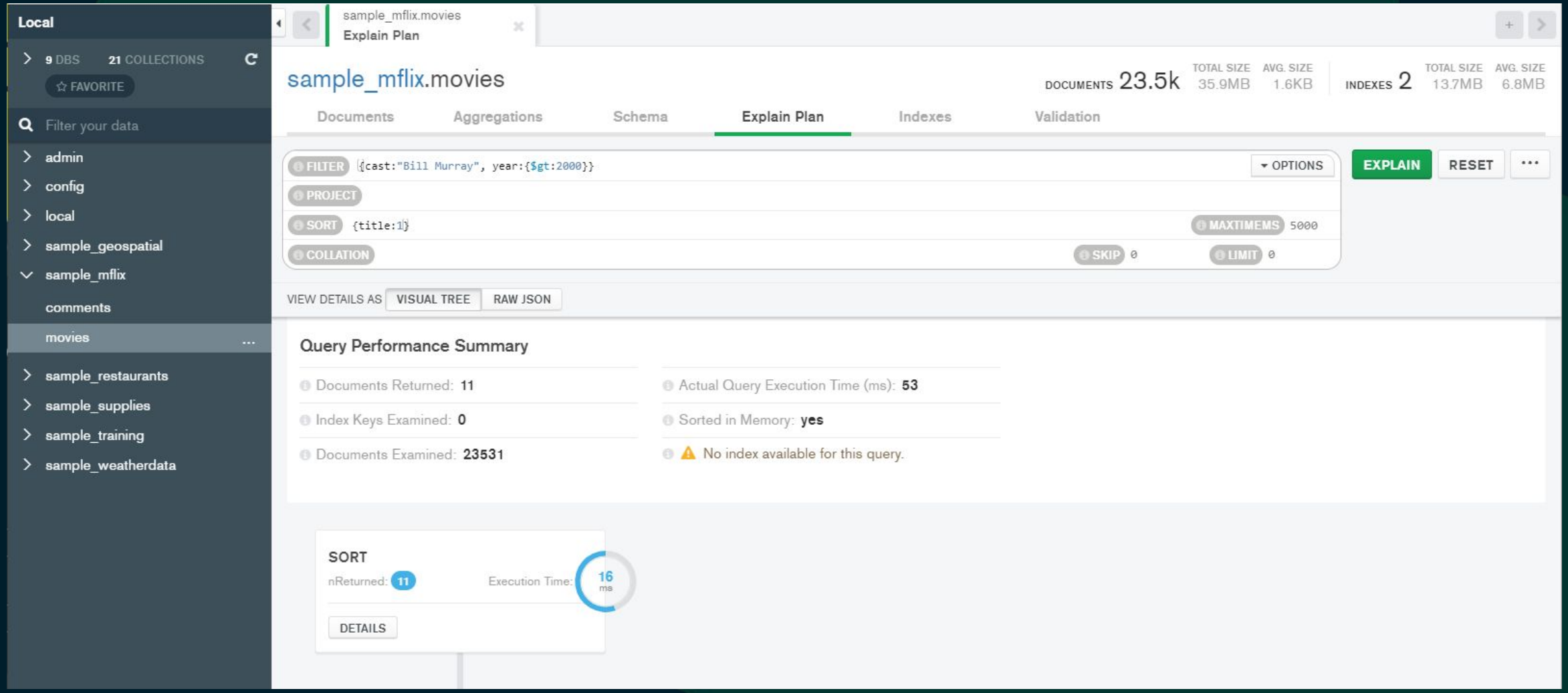
Let’s run a simple Equality, Sort, Range query

```
use sample_mflix
db.movies.find(
    {
        "cast":"Bill Murray",
        "year":{$gte:2000}
    }
).sort(
    {"title":1}
)
```

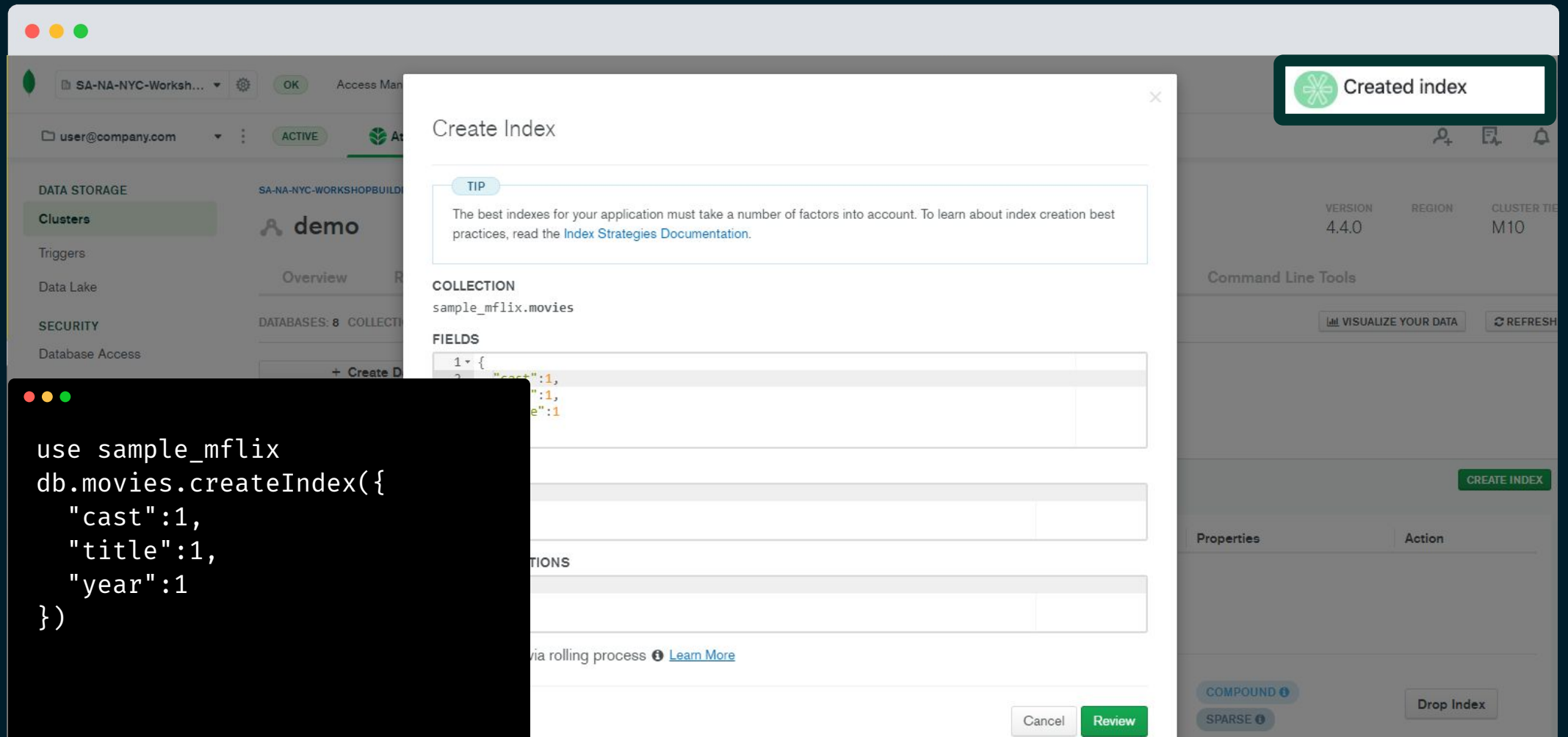
What results are you seeing from the Explain Plan?



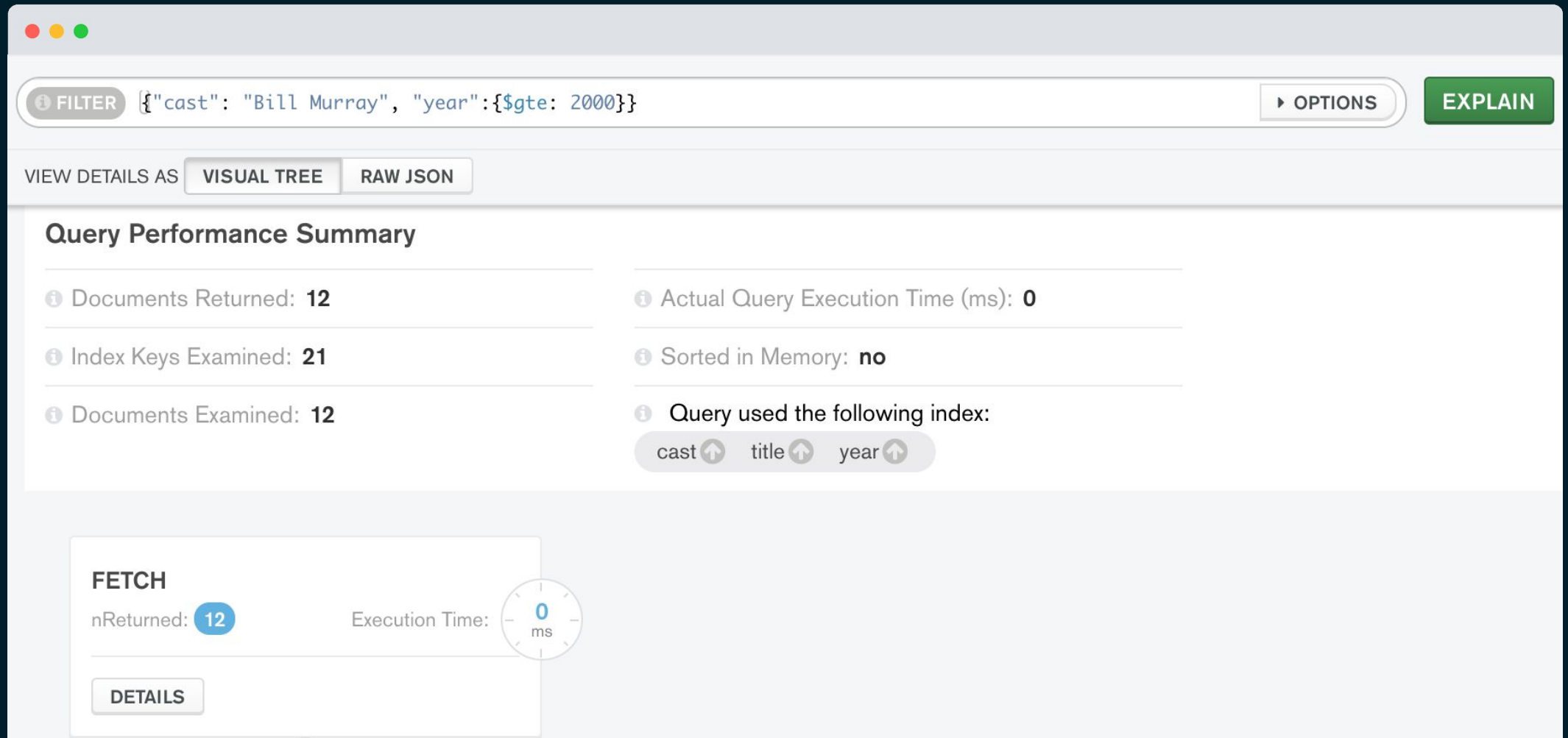

```
use sample_mflix
db.movies.find(
      {
      "cast":"Bill Murray",
      "year":{$gte:2000}
      }
).sort(
      {"title":1}
).explain("executionStats")
```


Local
9 DBS 21 COLLECTIONS
FAVORITE
Filter your data
admin
config
local
sample_geospatial
sample_mflix
comments
movies
sample_restaurants
sample_supplies
sample_training
sample_weatherdata
sample_mflix.movies
Explain Plan
sample_mflix.movies
DOCUMENTS 23.5k
TOTAL SIZE 35.9MB
AVG. SIZE 1.6KB
INDEXES 2
TOTAL SIZE 13.7MB
AVG. SIZE 6.8MB
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
FILTER {cast:"Bill Murray", year:{$gt:2000}}
OPTIONS
EXPLAIN
RESET
PROJECT
SORT {title:1}
MAXTIMEMS 5000
COLLATION
SKIP 0
LIMIT 0
VIEW DETAILS AS
VISUAL TREE
RAW JSON
Query Performance Summary
Documents Returned: 11
Index Keys Examined: 0
Documents Examined: 23531
Actual Query Execution Time (ms): 53
Sorted in Memory: yes
No index available for this query.
SORT
nReturned: 11
Execution Time: 16 ms
DETAILS

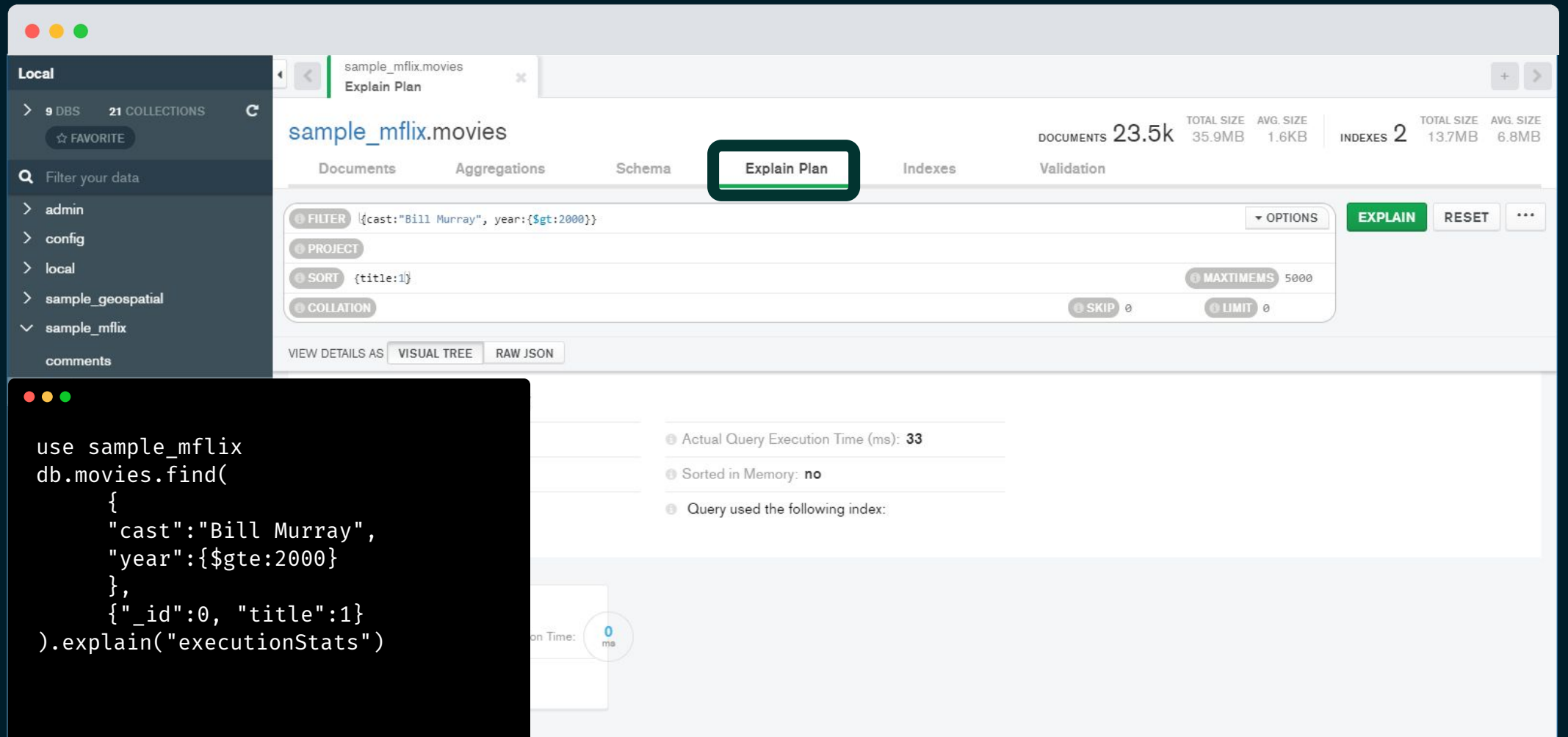
Create an index for cast, title, year



What's good and bad about the Equality, Range, Sort index we just created?



What do you think will happen if we return only the title?



Exercise Five

Aggregation Framework

*nix Pipes

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

MQL Pipelines

```
match
group
sort
```

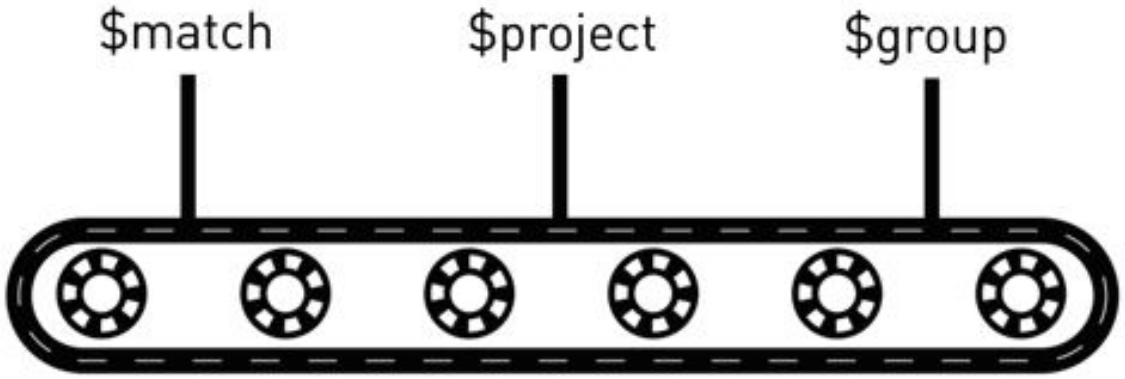
At the end of this, you should be able to…

Have a basic understanding of what the aggregation framework is

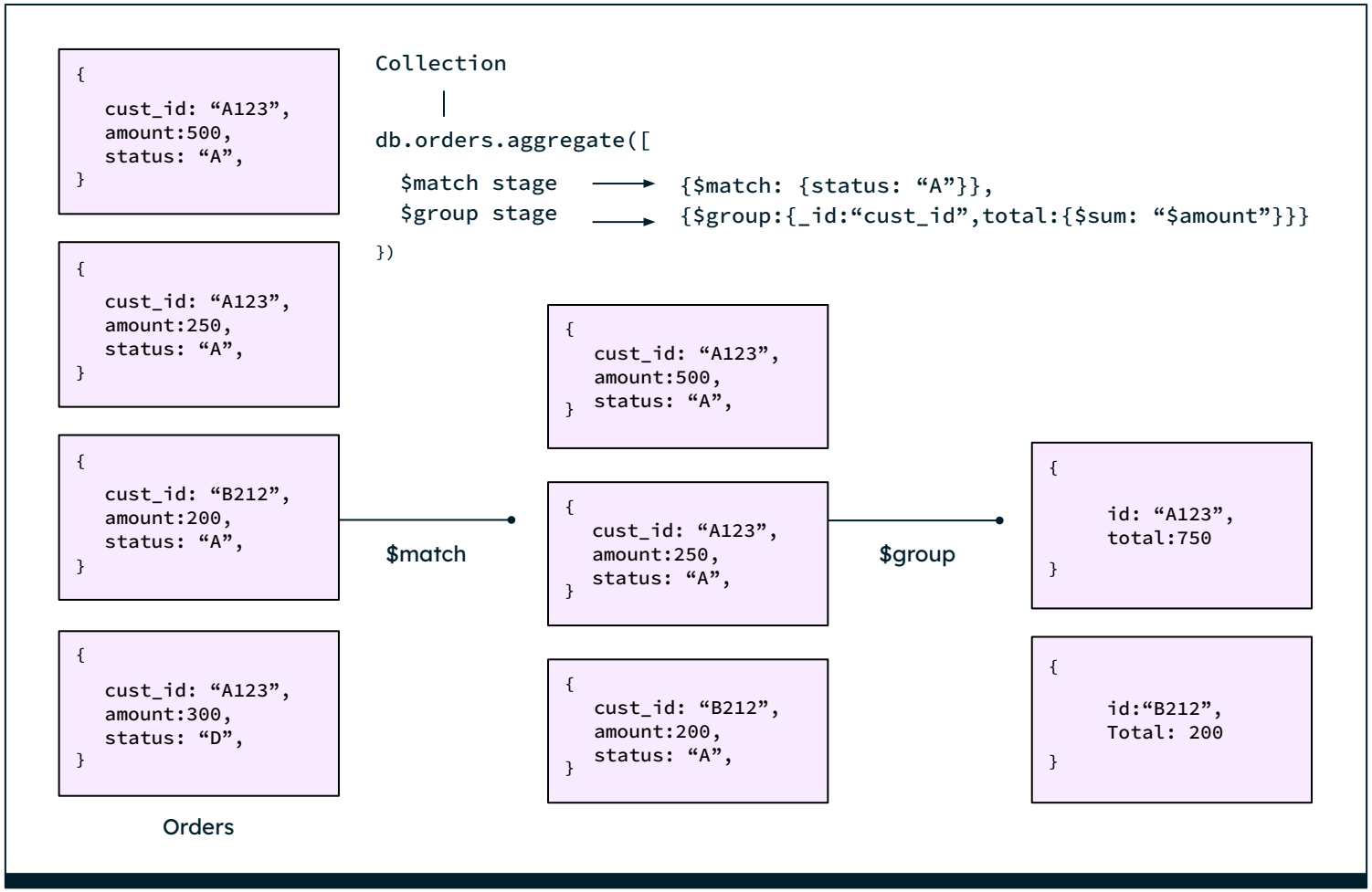
Build an aggregation pipeline to perform analytics on your documents

Aggregation -> Pipeline

- Each transformation is a single step known as a stage.
- Compared to one huge SQL style statement this is
  - Easier to understand
  - Easier to debug
  - Easier for MongoDB to rewrite and optimize



Aggregation pipelines



Aggregation features

A feature rich framework for data transformation and Analytics

Pipeline Stages

- $match
- $group
- $facet
- $geoNear
- $graphLookup
- $lookup
- $merge
- $project
- $search
- $sort
- $setWindowFields
- $unionWith
- $unwind
- ...and more

Operators

- Mathematical

  $add, $abs, $subtract, $multiply, $divide, $log, $log10, $stdDevPop, $stdDevSam, $avg, $sqrt, $pow, $sum, $zip, $convert, $round, etc.

- Array

  $push, $reduce, $reverseArray, $addToSet, $arrayElemAt, $slice, etc.

- Conditionals

  $and, $or, $eq, $lt, $lte, $gt, $gte, $cmp, $cond, $switch, $in, etc.

- Temporal

  Window Functions

  $dateAdd, $dateDiff, $dateSubstract, $dateTrunc

  $dateFromParts, $dateToParts, $dateFromString, $dateToString, $dayOfMonth, $isoWeek, $minute, $month, $year, etc.

- String

  $toUpper, $toLower, $substr, $strcasecmp, $concat, $split, etc.

- Literals

  $exp, $let, $literal, $map, $type, etc.

- Regex

  $regexFnd, $regexMatch, etc

- Trigonometry

  $sin, $cos, $degreesToRadians, etc.

- Custom Aggregation Expressions

PRACTICAL MONGODB AGGREGATIONS EBOOK

Compared to SQL JOINs and aggregation

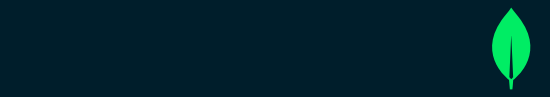
```
SELECT
    city,
    SUM(annual_spend) Total_Spend,
    AVG(annual_spend) Average_Spend,
    MAX(annual_spend) Max_Spend,
    COUNT(annual_spend) customers
FROM (
    SELECT t1.city, customer.annual_spend
    FROM customer
    LEFT JOIN (
        SELECT address.address_id, city.city,
               address.customer_id, address.location
        FROM address LEFT JOIN city
        ON address.city_id = city.city_id
        ) AS t1
    ON
    (customer.customer_id = t1.customer_id AND
              t1.location = "home")
) AS t2
GROUP BY city;
```

SQL queries have a nested structure

Understanding the outer layers requires understanding the inner ones

So SQL has to be read “inside-out”

Complex queries fast to create, optimize, and maintain

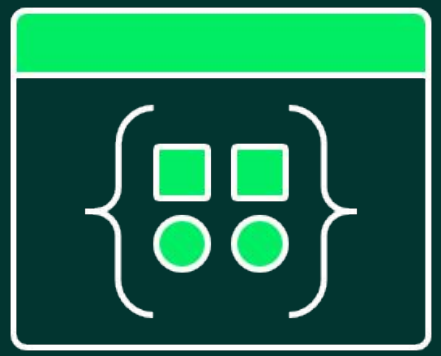


```
db.customers.aggregate([
    {
        $unwind: "$address",
    },
    {
        $match: {"address.location": "home"}
    },
    {
         $group: {
            _id: "$address.city",
            totalSpend: {$sum: "$annualSpend"},
            averageSpend: {$avg: "$annualSpend"},
            maximumSpend: {$max: "$annualSpend"},
            customers: {$sum: 1}
        }
    }
])
```

These “phases” are distinct and easy to understand

They can be thought about in order… no nesting

MongoDB’s aggregation framework has the flexibility you need to get value from your data, but without the complexity and fragility of SQL



Materialized Views

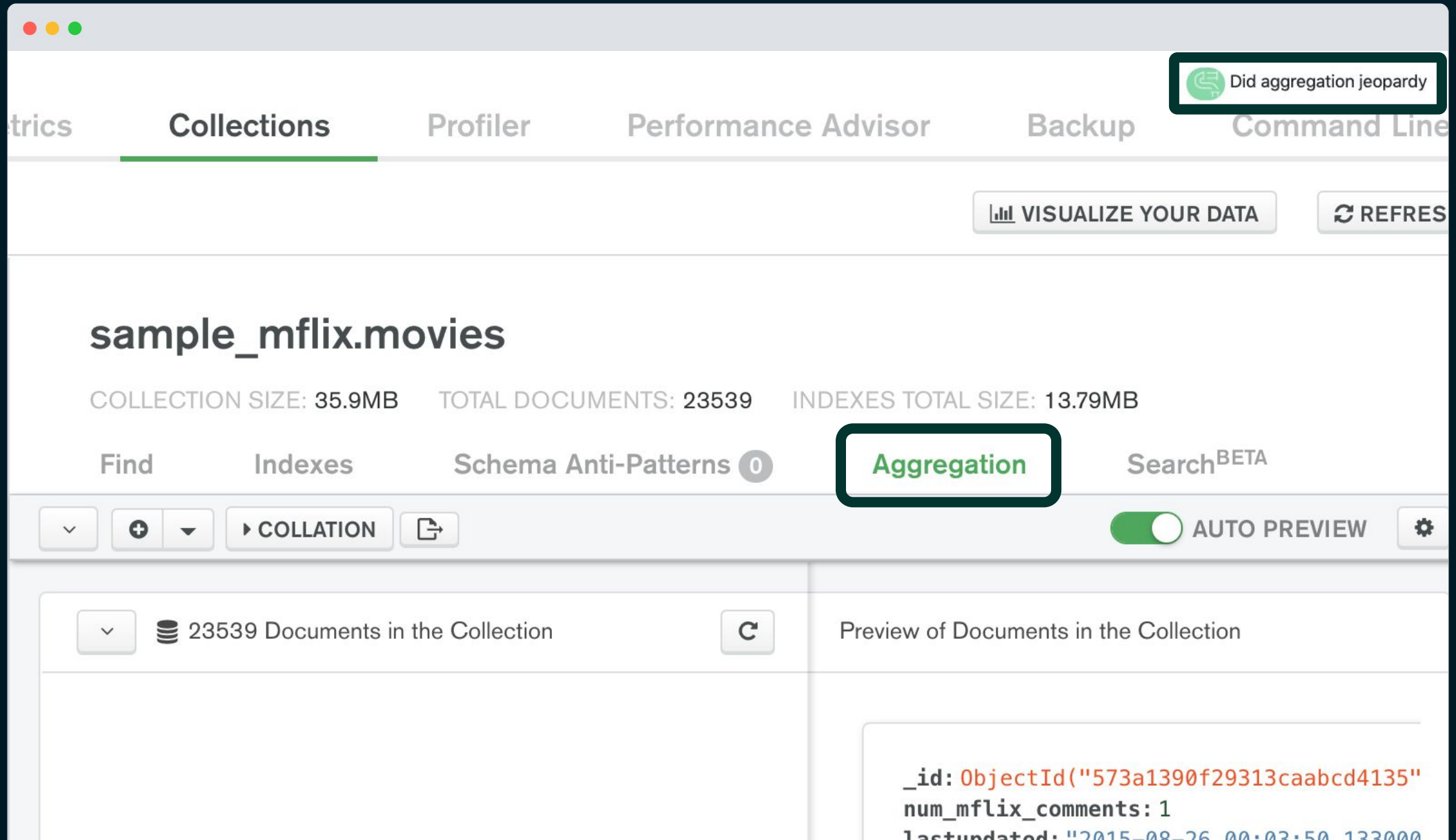
Faster insights on your data: pre-compute and store results of common analytics queries

With $merge stage aggregation pipeline outputs with existing result sets to increment and enrich views

- Updated each time the pipeline is run
- Output to sharded and unsharded collections
- Define indexes on each view

With uniqueKey, control how documents are added to the view: Insert, Replace, Merge

JEOPARDY!


Did aggregation jeopardy
trics
Collections
Profiler
Performance Advisor
Backup
Command Line
VISUALIZE YOUR DATA
REFRES
sample_mflix.movies
COLLECTION SIZE: 35.9MB
TOTAL DOCUMENTS: 23539
INDEXES TOTAL SIZE: 13.79MB
Find
Indexes
Schema Anti-Patterns 0
Aggregation
Search BETA
COLLATION
AUTO PREVIEW
23539 Documents in the Collection
Preview of Documents in the Collection
_id: ObjectId("573a1390f29313caabcd4135"
num_mflix_comments: 1

Aggregation: Jeopardy

Aggregation	What's the question?
`{$match: {genres: "Comedy"}}`	

Aggregation: Jeopardy

Aggregation	What's the question?
`{$match: {genres: "Comedy"}}`	How can you find all movies with Comedy as a Genre?

Aggregation: Jeopardy

Aggregation	What's the question?
`{$match: {genres: "Comedy"}}`	How can you find all movies with Comedy as a Genre?
`{$unwind: {path: "$countries"}}`	

Aggregation: Jeopardy

Aggregation	What's the question?
`{$match: {genres: "Comedy"}}`	How can you find all movies with Comedy as a Genre?
`{$unwind: {path: "$countries"}}`	How can you create an individual document for each country the movie was in?

Aggregation: Jeopardy

Aggregation	What's the question?
`{$match: {genres: "Comedy"}}`	How can you find all movies with Comedy as a Genre?
`{$unwind: {path: "$countries"}}`	How can you create an individual document for each country the movie was in?
`{ $group: { _id: "$countries", count: {$sum: 1}} }`	

Aggregation: Jeopardy

Aggregation	What's the question?
`{$match: {genres: "Comedy"}}`	How can you find all movies with Comedy as a Genre?
`{$unwind: {path: "$countries"}}`	How can you create an individual document for each country the movie was in?
`{ $group: { _id: "$countries", count: {$sum: 1}} }`	How can you count all the comedies grouped by country?

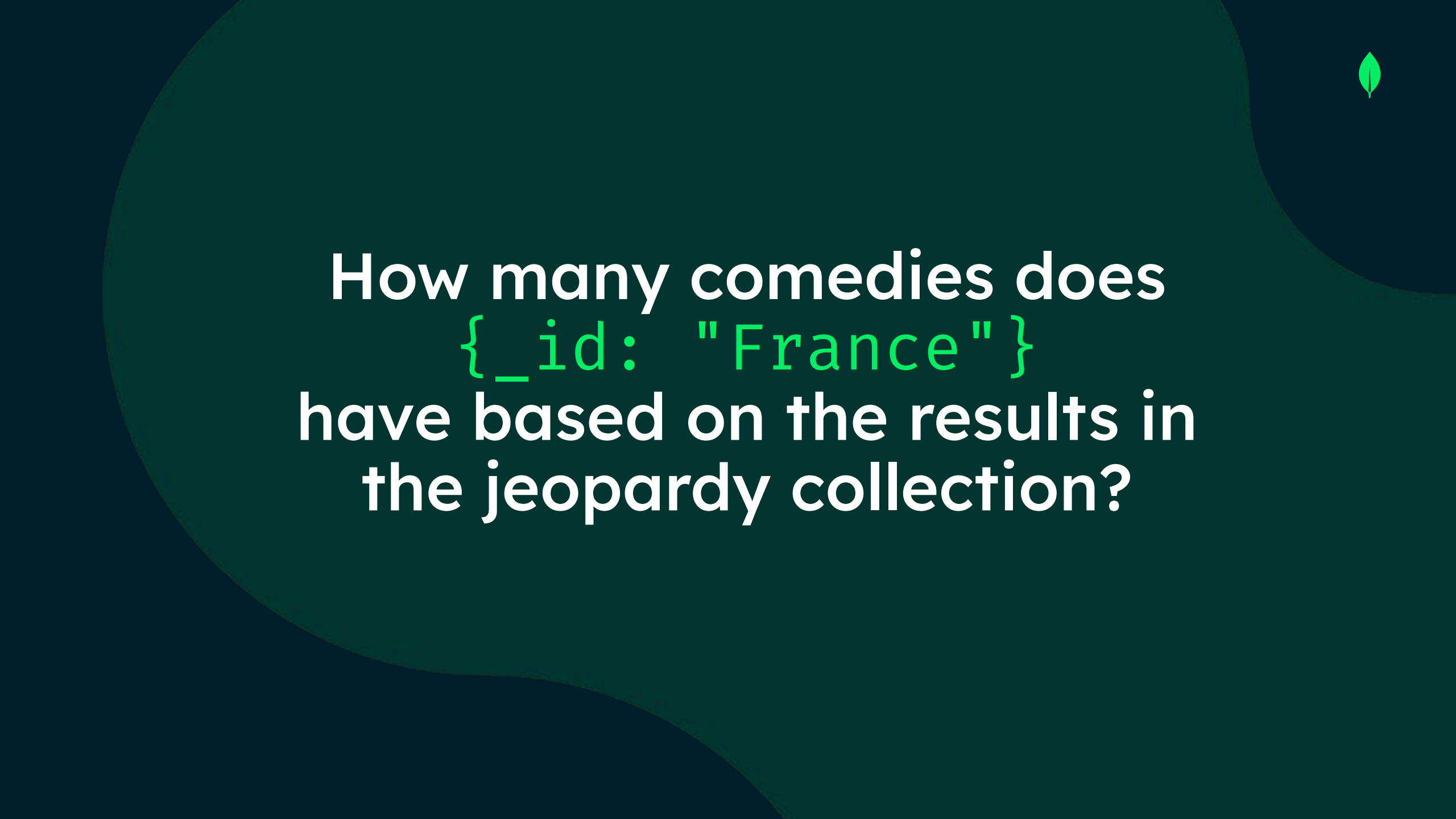
Aggregation: Jeopardy

Aggregation	What's the question?
`{$match: {genres: "Comedy"}}`	How can you find all movies with Comedy as a Genre?
`{$unwind: {path: "$countries"}}`	How can you create an individual document for each country the movie was in?
`{ $group: { _id: "$countries", count: {$sum: 1}} }`	How can you count all the comedies grouped by country?
`{$out: "jeopardy"}`	

Aggregation: Jeopardy

Aggregation	What's the question?
`{$match: {genres: "Comedy"}}`	How can you find all movies with Comedy as a Genre?
`{$unwind: {path: "$countries"}}`	How can you create an individual document for each country the movie was in?
`{ $group: { _id: "$countries", count: {$sum: 1}} }`	How can you count all the comedies grouped by country?
`{$out: "jeopardy"}`	How can you output the results into a new collection, jeopardy?

DOUBLE JEOPARDY!


How many comedies does
{_id: "France"}
have based on the results in
the jeopardy collection?

TRIPLE JEOPARDY!

Exercise 2

Calculate how many Airbnb properties there are in each country, ordered by count, list the one with the largest count first. You should continue working with namespace **`sample_airbnb.listingsAndReviews`**.

- **Hint:** to count things, you add the explicit value 1 to an accumulator using $sum

ANSWER: Exercise 2

Calculate how many Airbnb properties there are in each country, ordered by count, list the one with the largest count first.

```
groupfield = "$address.country"
groupstage = { $group: { _id: groupfield, count:{$sum:1}}}
sortstage = {$sort:{count:-1}}
pipe = [groupstage,sortstage]
db.listingsAndReviews.aggregate(pipe).pretty()
```

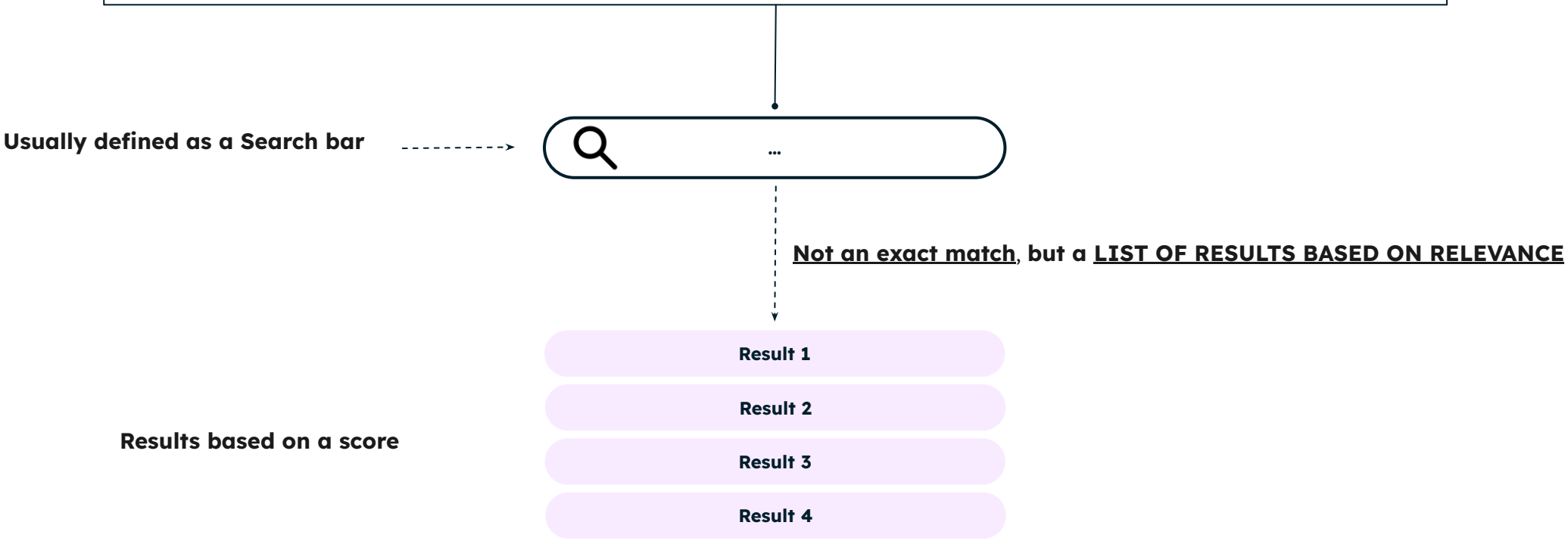
or

```
db.listingsAndReviews.aggregate([{$sortByCount:"$address.country"}]).pretty()
```

Search

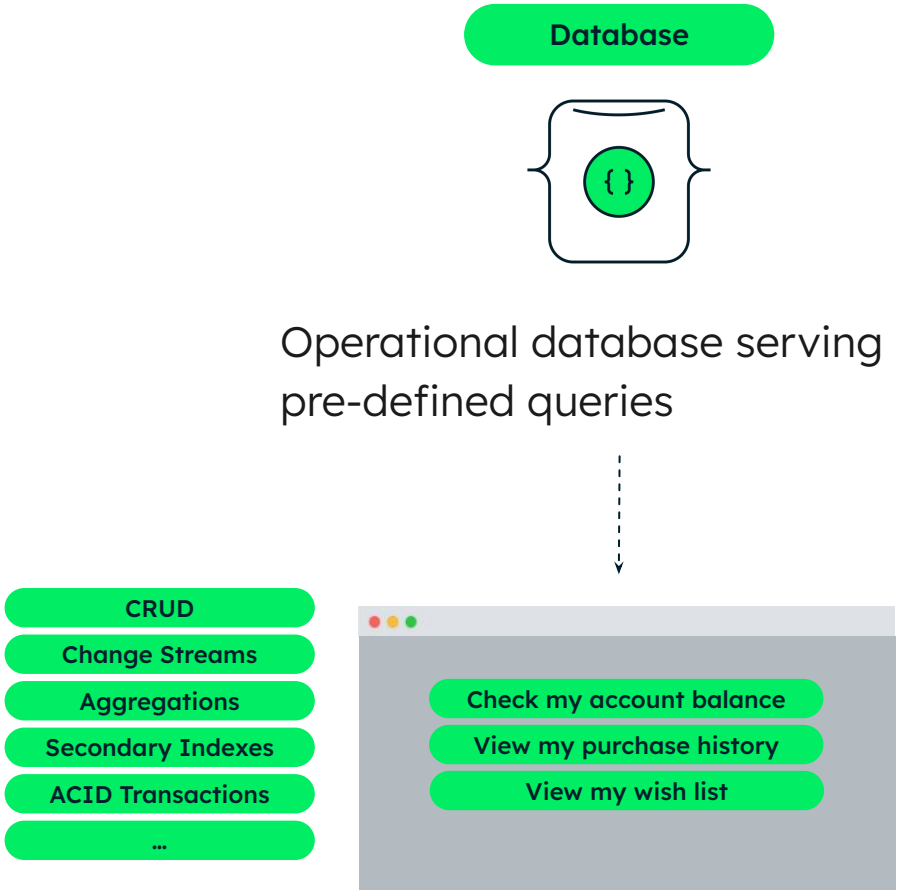
What is Search?

Search, or “Full-Text Search” is the ability to search across all of your data and efficiently return a list of results, ranked based on how well they matched to the search term



How developers build app search today

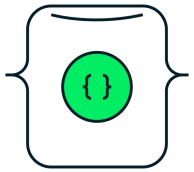
Stand up a database to provide the application's persistence layer



How developers build app search today

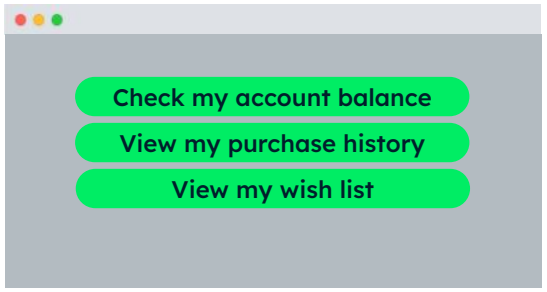
Bolt-on a search engine to power search across application data

Database

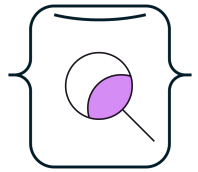


Operational database serving pre-defined queries

- CRUD
- Change Streams
- Aggregations
- Secondary Indexes
- ACID Transactions
- ...



Search Engine

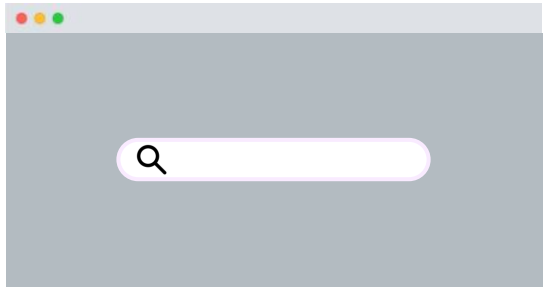


Lucene

elasticsearch

Solr

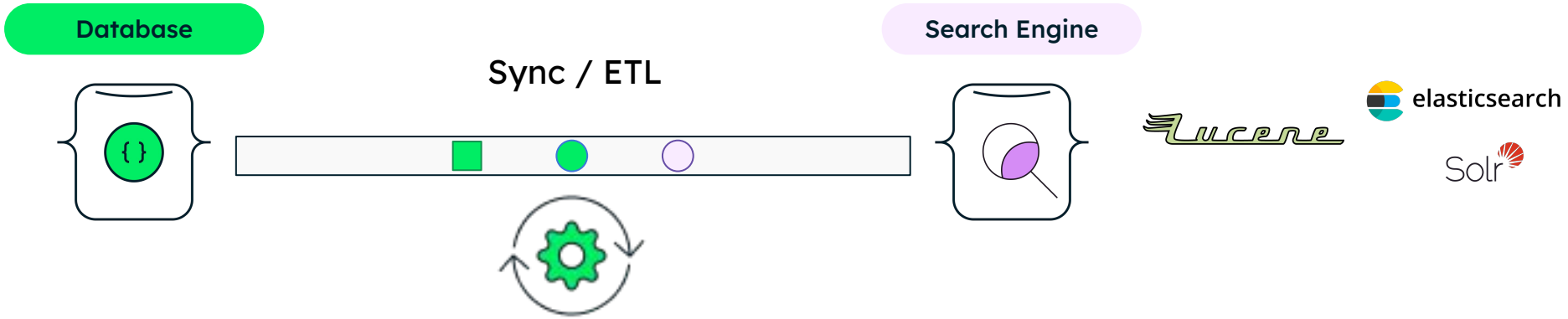
Search engine inferring intent from free form, natural language queries



- Fuzzy matching
- Autocomplete
- Facets and counts
- Synonyms
- Scoring / Weighting
- moreLikeThis

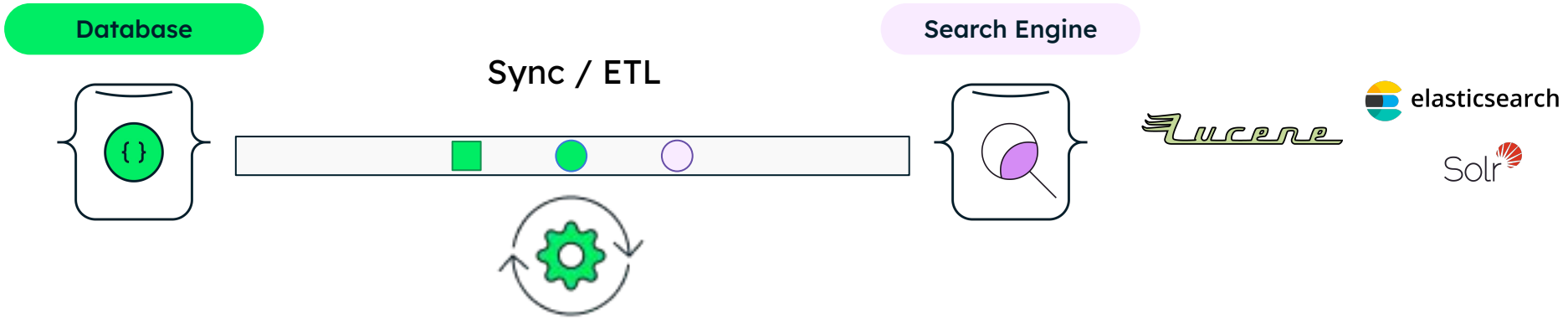
How developers build app search today

Stand up a replication mechanism to keep the systems in sync



How developers build app search today

Architectural complexity in our application stack



Developer friction

Different query APIs and drivers for database and search, coordinate schema changes

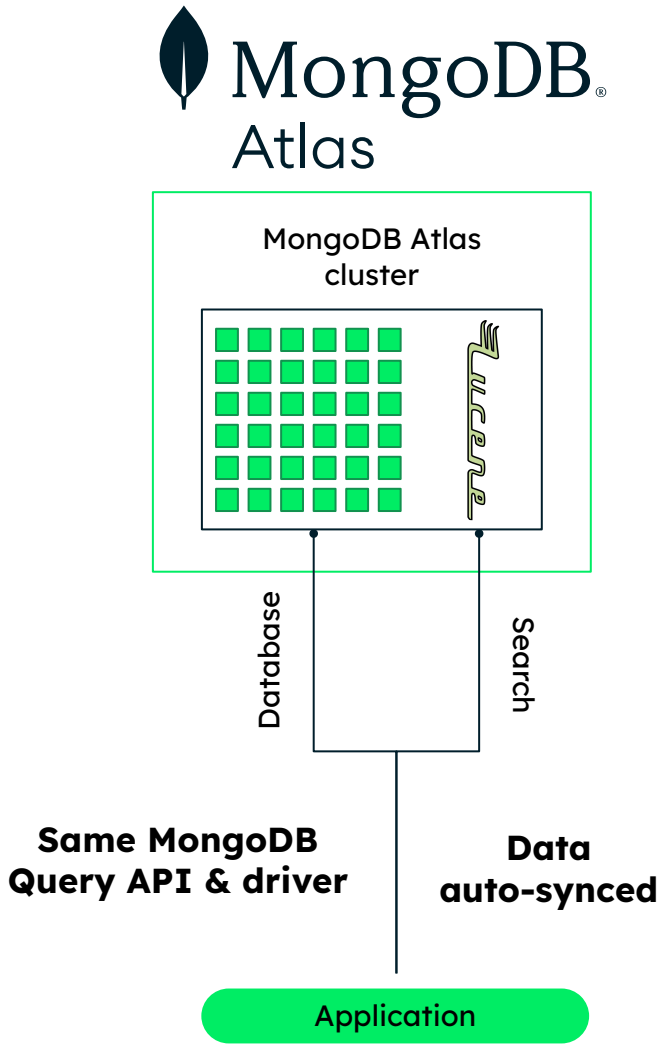
Pay the sync tax

Requires its own systems and skills. Recovering sync errors can consume 10% of a developer's time

Operational overhead

More to provision, secure, upgrade, patch, back up, monitor, scale, etc.

How does Atlas Search address that pain?



Embeds a fully managed Apache Lucene index right alongside the database

Improved developer productivity

Build database and search features using the same query API and driver

Simplified data architecture

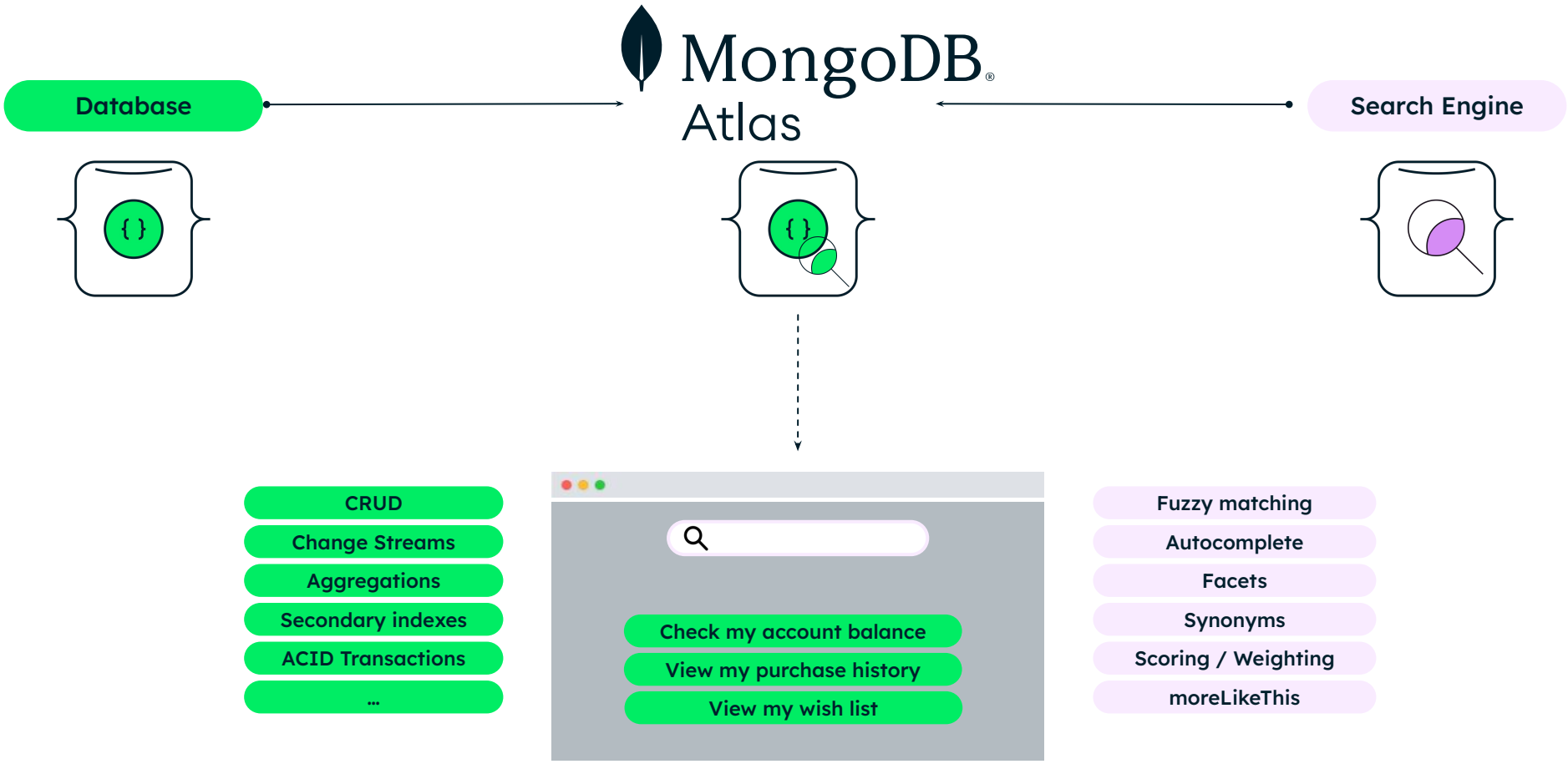
Automatic data synchronization, even as your data and schema changes

Fully managed platform

Get the security, performance, and reliability of Atlas

Unifying database and search

Compress 3 systems into 1, ship search features **30%-50% faster**

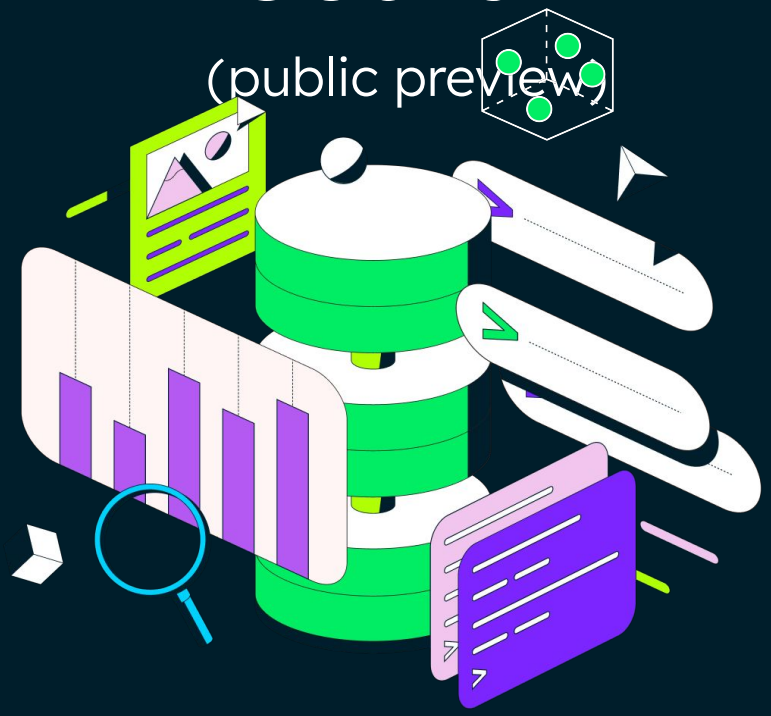


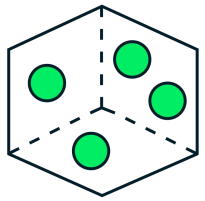
Vector Search

SPOTLIGHT

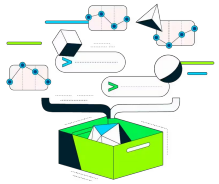
Atlas Vector Search

(public preview)

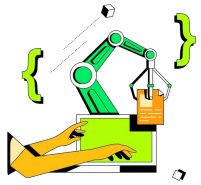




Build intelligent applications powered by semantic search and generative AI over any type of data



Store vector embeddings right next to your source data and metadata with the power of the document model. Vectors inserted or updated in the database are automatically synchronized to the vector index



Remove operational heavy lifting with the battle tested, fully managed Atlas platform

Why is Vector Search needed ?

- Syntactically similar sentences can have very different meanings.
- Syntactic search = matching of search terms
- Semantic search = meaning and context

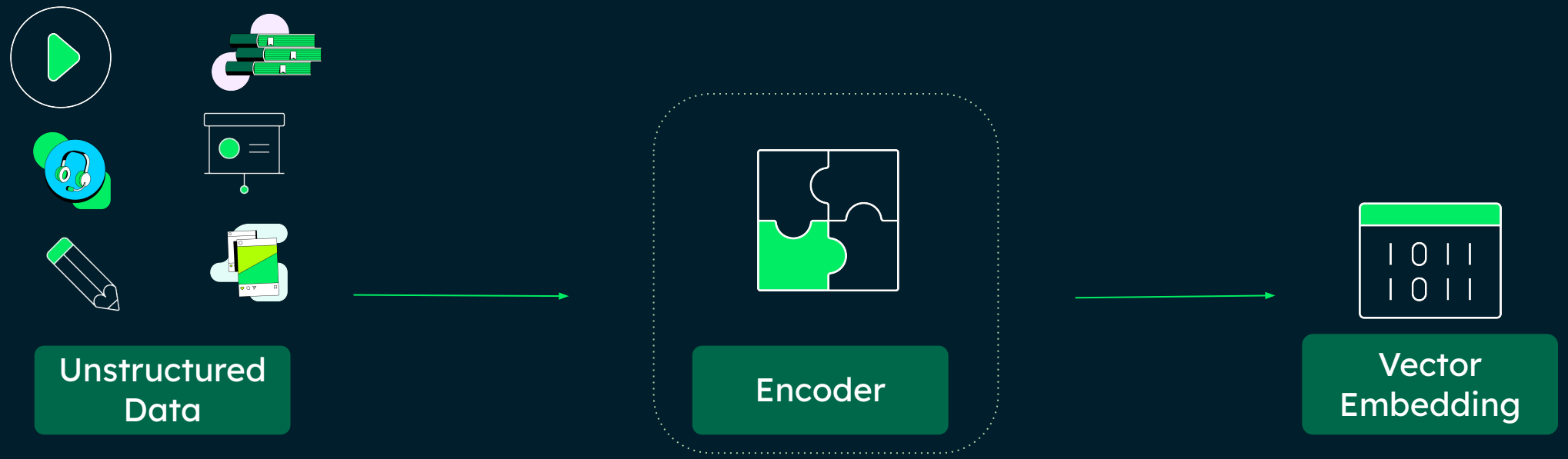
Sentence#1: **“Mike Tyson knocked out Michael Spinks.”**

Sentence#2: **“Michael Spinks knocked out Mike Tyson.”** *(not true btw)*

Vectors are a numeric representation of data and related context

"a quick brown fox" = [0.743, 0.720, -0.325, 0.195, 0.835, -0.945, ...]

Vector embeddings are produced by sending data through an encoding model



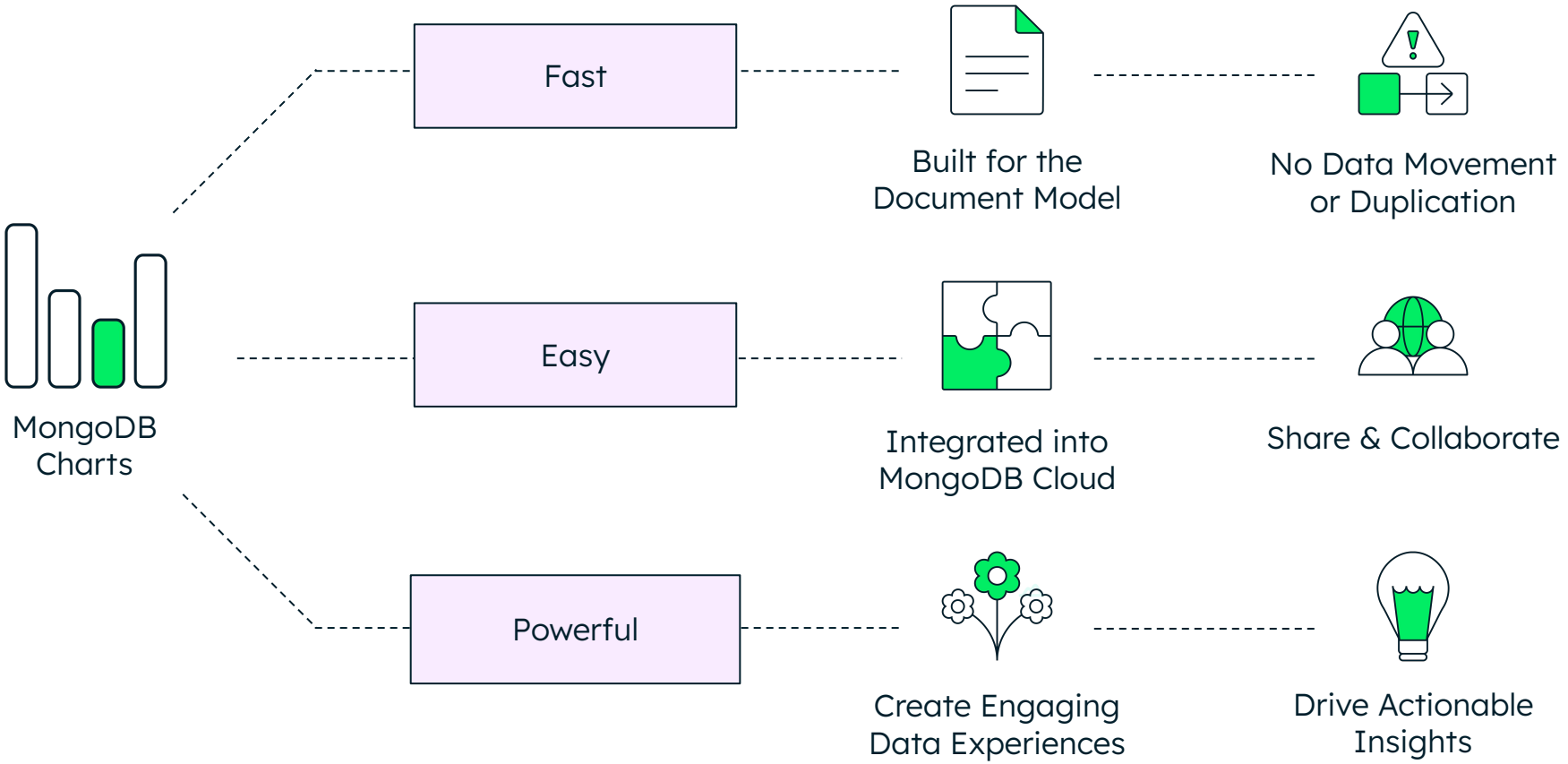
E.g. OpenAI, Cohere, Anthropic,
HuggingFace, Vertex, etc.

Vector Search powers a number of key use cases

- Semantic Search
- Similarity Search
- Recommendation Engines
- Long-term memory for LLMs
- Q & A Systems
- Image, Audio, Multimedia Search

MongoDB Charts

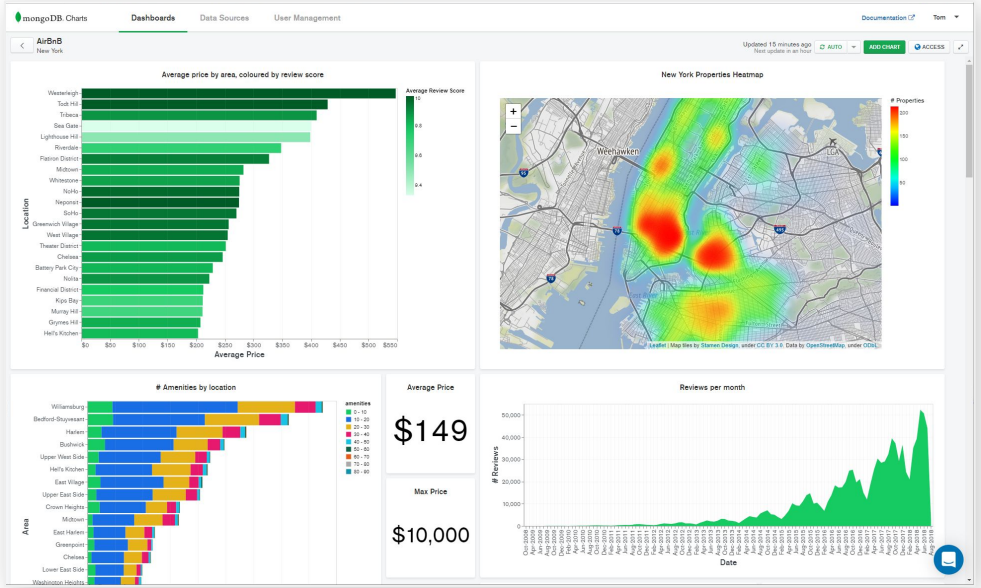
The best way to visualize MongoDB Data



Easy to get started and share

- Integrated into the MongoDB Data Application Platform, there's no setup required and total cost of ownership is reduced

- Share and collaborate with dashboards so your teams can work together or self-serve to make decisions in real-time



Atlas SQL Interface, Connectors, and Drivers

Leverage SQL tools to easily query and generate rapid insights from MongoDB Atlas

Atlas SQL Interface

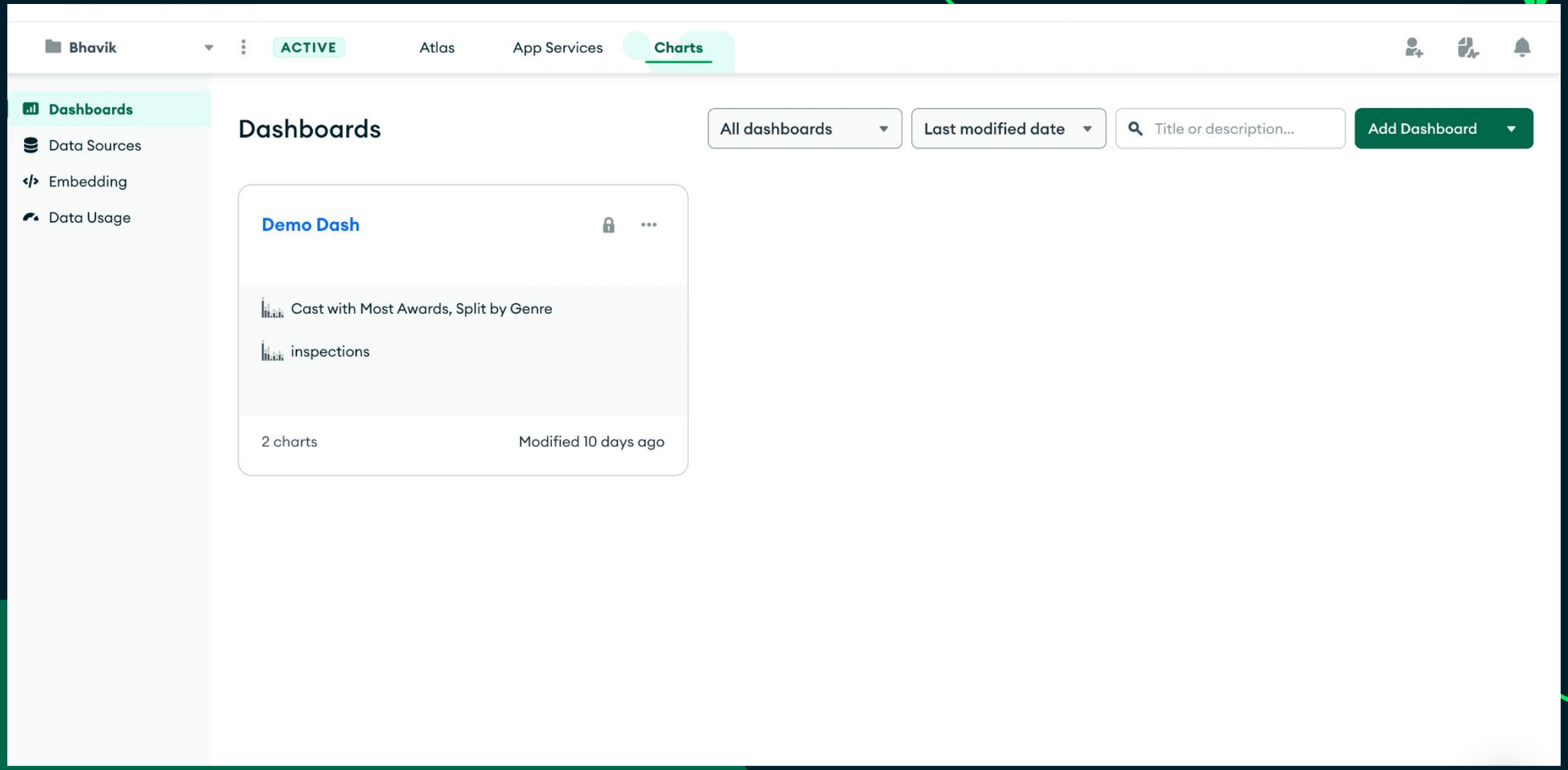
Visualize data using **Custom Connectors** for BI tools

Use **JDBC/ODBC Drivers** for other SQL connections

tableau

Power BI

Looker


Bhavik
ACTIVE
Atlas
App Services
Charts
Dashboards
Data Sources
Embedding
Data Usage
Dashboards
All dashboards
Last modified date
Title or description...
Add Dashboard
Demo Dash
Cast with Most Awards, Split by Genre
inspections
2 charts
Modified 10 days ago

Charts workshop: Step 1

Create a New Dashboard

1. Click the Dashboards tab
2. Click the New/Add Dashboard button
3. Enter the Title: Movie Details
4. Click Create

Charts workshop: Step 2

Create Chart and Select Data Source

1. Click Add Chart button
2. Select the sample_mflix.movies data source under Project and demo cluster
3. Enter the Title: Movie Details

Charts workshop: Step 3

Create Chart Showing Directors with the Most Awards

1. Select Chart Type: Column / Stacked
2. X Axis: **director**
    a. Sort By: Aggregated Value, Descending
    b. Limit: 10

Charts workshop: Step 4

Create Chart Showing Directors with the Most Awards

5. Y Axis: **awards.wins**
    a. Aggregate: sum
6. Series: **genres**
    a. Array Reduction: Unwind Array
7. Title: **Directors with Most Awards, Split by Genre**
8. Click Save and Close

Thank you

MongoDB Technical Overview

The evolution of MongoDB

2015

3.0 & 3.2

Doc-Level Concurrency
RAFT / Fast Failover
$lookup
Ops Manager
Compression
≤50 replicas
Aggregation ++
Encrypted and In-Memory storage engines
BI Connector
Compass

2016

3.4

Views Graph Processing
Zones ++Aggregation ++
Auto-balancing ++
Linearizable Reads Decimal
Intra-cluster Compression Log Redaction Spark Connector
++ BI Connector ++

2017

3.6

Change Streams
Retryable Writes
Schema Validation Expressive $lookUp
Query Expressivity
Causal Consistency Consistent Sharded Secondary Reads
Query Advisor
End to End Compression
WiredTiger 1m+ Collections
MongoDB BI Connector ++
R Driver
Charts (post GA)
Atlas X-Region Replication Atlas Auto Storage Scaling

2018

4.0

Replica set transactions

Atlas global clusters

40% Faster Shard Migrations
Atlas HIPAA
Atlas LDAP
Atlas Audit
Atlas Enc. Storage Engine
Atlas Backup Snapshots
Type Conversions
Snapshot Reads
Non-Blocking Sec. Reads
SHA-2 & TLS 1.1+
Compass Agg Pipeline Builder
Compass Export to Code
Free Monitoring Cloud Service
Ops Manager K8s Beta

2019

4.2

Distributed Transactions
Client-Side Field Level Encryption
Materialized Views
Wildcard Indexes
Global PIT Reads
Large Transactions
Mutable Shard Key Values
Atlas Data Lake (Beta)
Atlas Auto Scaling (Beta)
Atlas Search (Beta)
Multi-CAs
Expressive Updates
Apache Kafka Connector
MongoDB Charts GA
Retryable Reads & Writes
New Index Builds
10x Faster stepDown
Storage Node Watchdog
Zstandard Compression

2020

4.4

Union
Custom Agg Expressions
Refinable Shard Keys
Compound Hashed Shard Keys
Mirrored Reads
Hedged Reads
Resumable Initial Sync
Time-Based Oplog Retention
Simultaneous Indexing
Hidden Indexes
Streaming Replication
Global Read/Write Concerns
Rust & Swift Drivers GA
TLS 1.3 & Faster Client Auth
OCSP Stapling
Kerberos Utility
Atlas Online Archive
Auto-Scaling
Schema Recommendations
AWS IAM Auth & Atlas x509
Federated Queries
Multi-cloud clusters

2021–2022

5.x

Time Series collections
Clustered indexes
Window functions
Live resharding
Client-Side FLE KMIP & cloud KMS
Atlas Serverless (preview)
Atlas Search fast facets, function scores, synonyms
Long running snapshot reads
Sharded $lookup & $graphlookup
Majority write concern default
4x faster initial sync
Schema validation diagnostics
New MongoDB Shell GA
Resumable index builds
Rewritten Swift driver
Rewritten C# LINQ provider and .NET Analyzer
PyMongoArrow API
New accumulator operators
Charts on Data Lake
x509 certificate rotation
Auditing ++
Atlas K8S controller
Ops Manager migration wizard

2022–2023

6.x

Queryable encryption (preview)
Atlas Serverless GA
Column store indexes
Cluster-to-cluster sync
Independent analytic nodes scaling
Flexible sync GA
Asymmetric sync preview
Kotlin and Flutter SDKs for Realm
$lookup perf improvements
Atlas CLI GA
Atlas Data API GA
Secondary index on measurements for time series
Time series read perf +
Atlas Data Lake with fully managed storage
Atlas Data Federation
Atlas Charts dashboard embedding
Atlas SQL interface
Encrypted audit log
Change streams improvements
Announcement of relational to MongoDB migrator

WiredTiger (Acquisition + Integration)

MongoDB Atlas

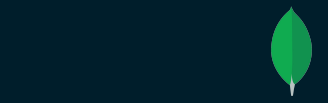
ACID (Transactions)

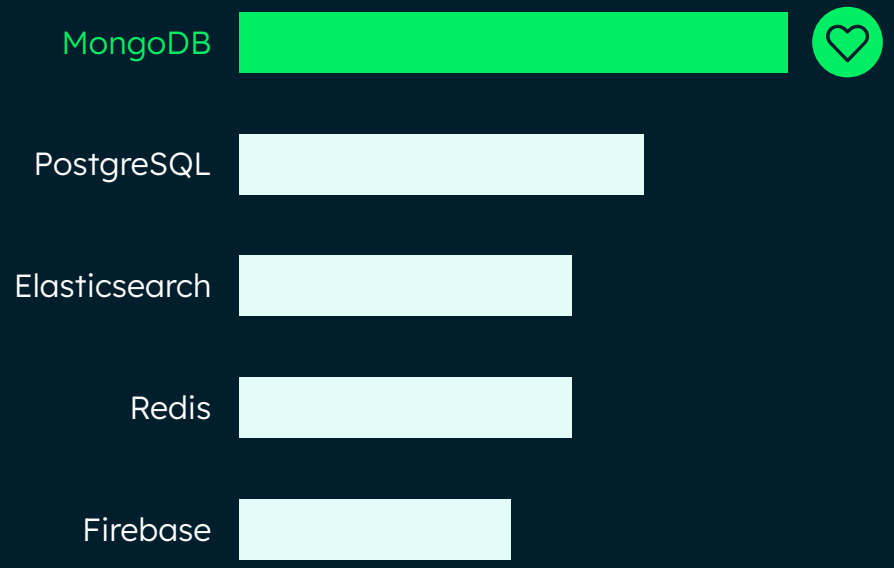
Multi-Cloud

Stable API & Rapid Releases

Developer Data Platform

Why MongoDB?





Consistently voted
“most wanted” database

STACK OVERFLOW | DEVELOPER SURVEY

Developers love using MongoDB

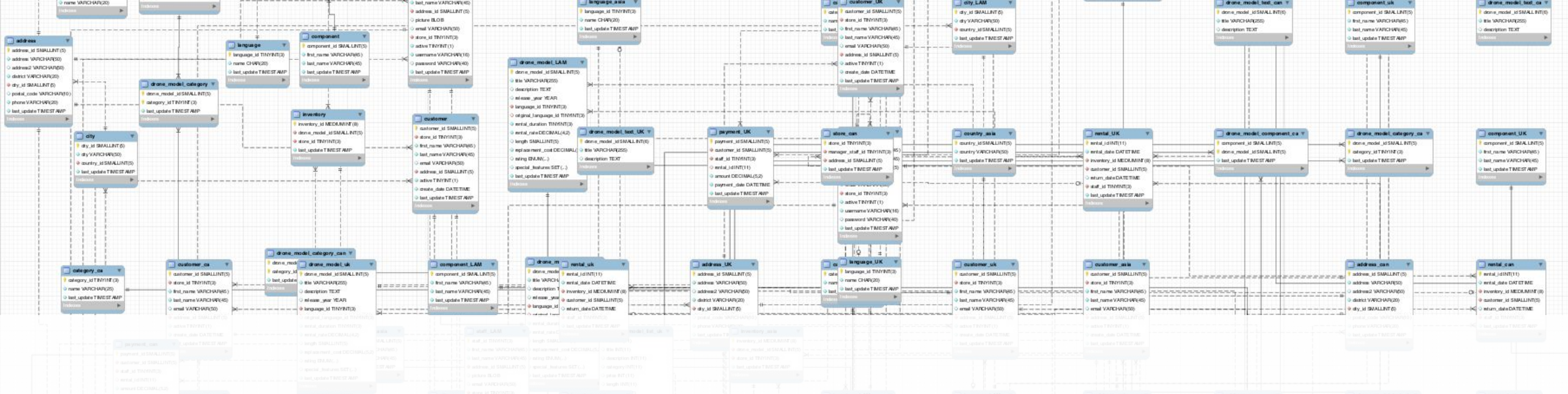
Intuitive and flexible
document data model

Powerful query engine and
idiomatic API to work with data

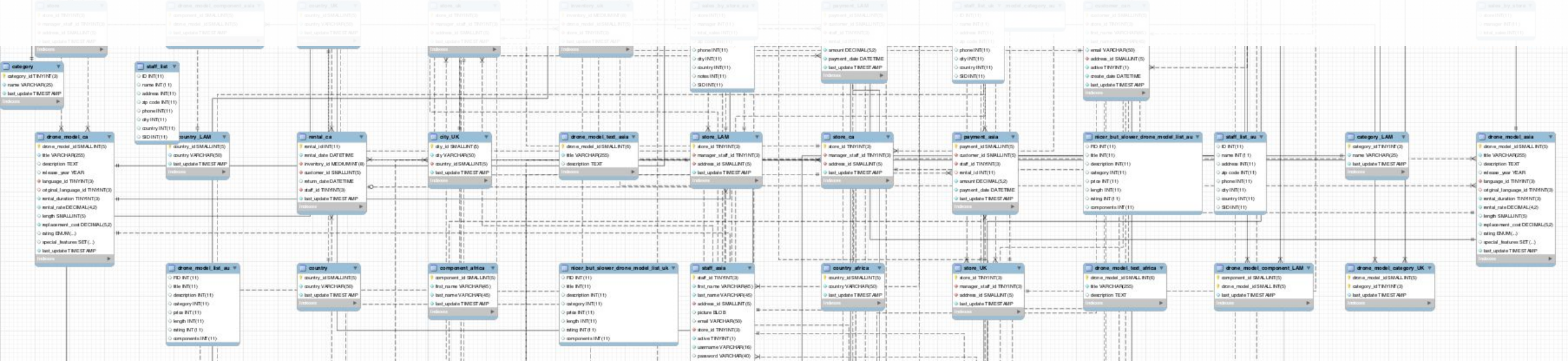
Distributed systems architecture
for resilience & scale-out

The Document Model and MongoDB Query API

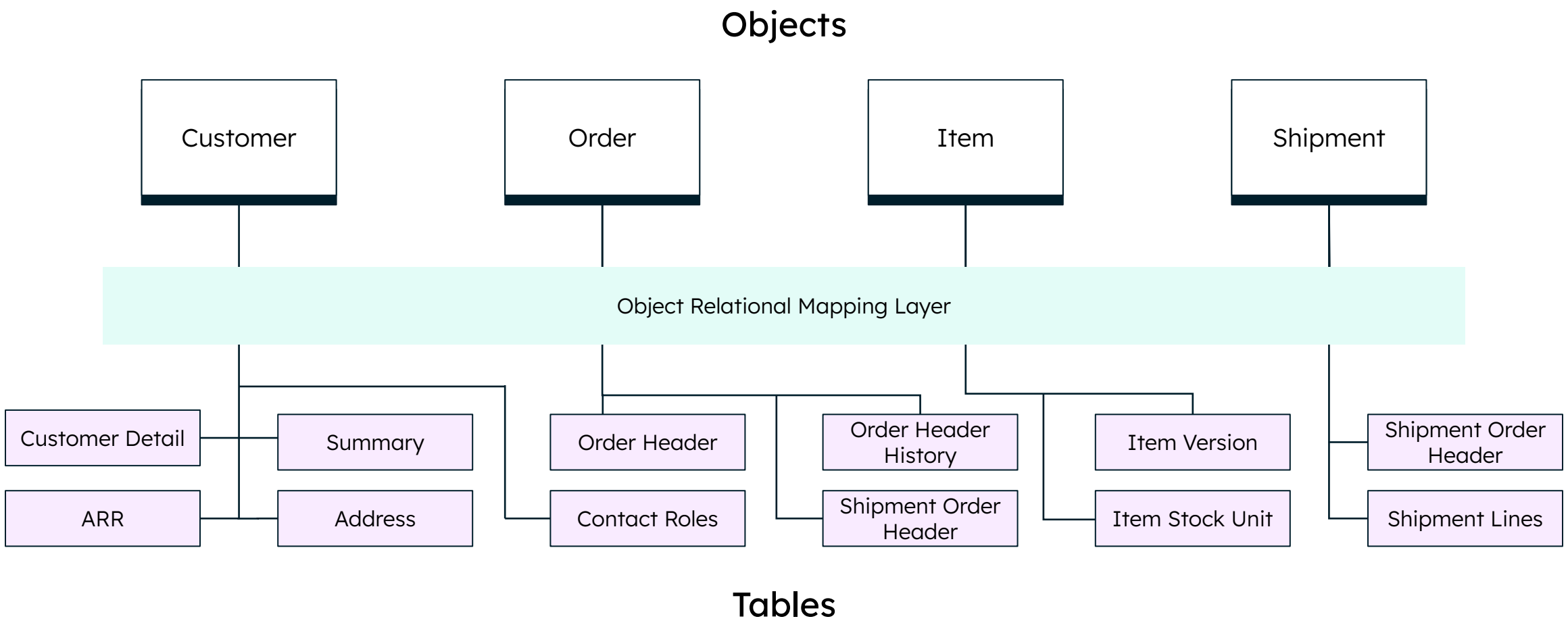
The fastest way to innovate



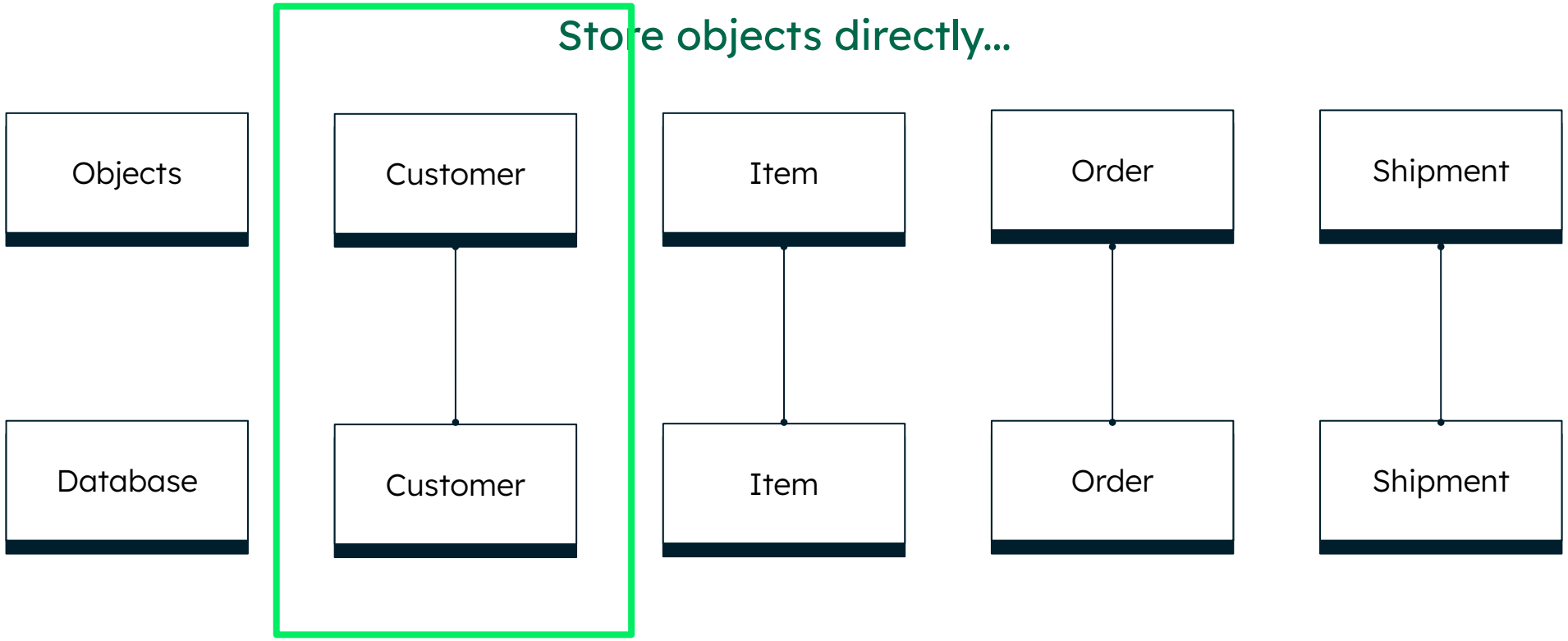
You probably have **thousands** of tables



Go from this...

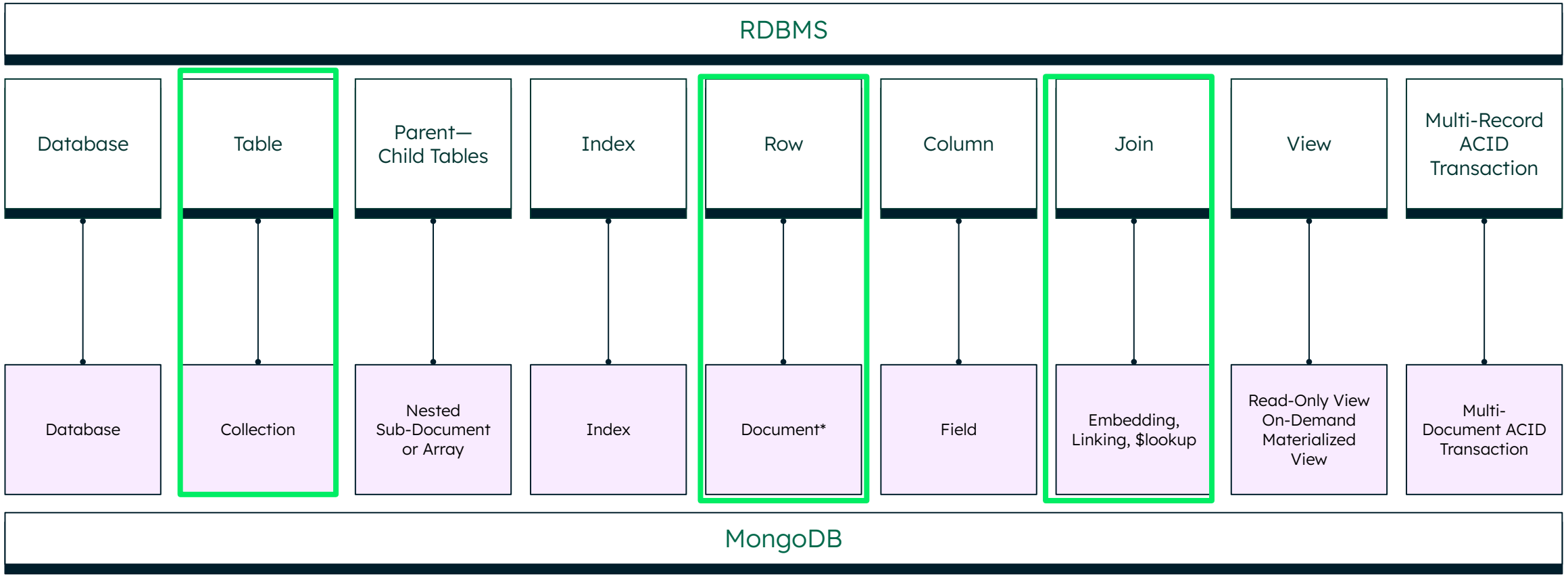

Objects
Customer
Order
Item
Shipment
Object Relational Mapping Layer
Customer Detail
ARR
Summary
Address
Order Header
Contact Roles
Order Header History
Shipment Order Header
Item Version
Item Stock Unit
Shipment Order Header
Shipment Lines
Tables

...to this


Store objects directly...
Objects
Customer
Item
Order
Shipment
Database
Customer
Item
Order
Shipment

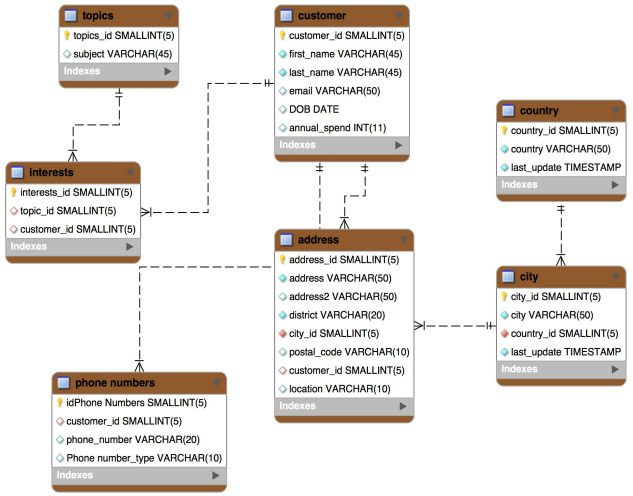
Some terminology

A comparison



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

INTUITIVE



Tabular (Relational) Data Model

Related data split across multiple records and tables

Contrasting data models

Document Data Model

Related data contained in a single, rich document

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

INTUITIVE

Naturally maps to objects in code

- Eliminates requirements to use ORMs
- Breaks down complex interdependencies between developer and DBAs teams

Represent data of any structure

- Polymorphic: each document can contain different fields
- Modify the schema at any time

Strongly typed for ease of processing

- Over 20 binary encoded JSON data types

Access by idiomatic drivers in all major programming language

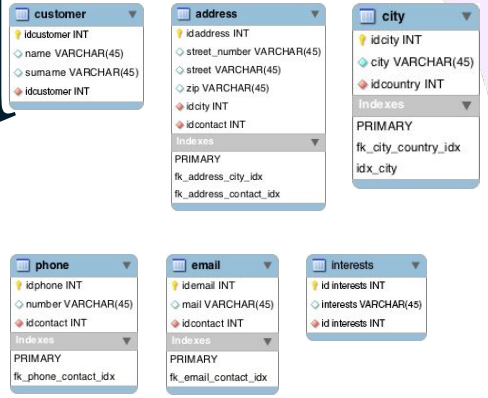
Document data model

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

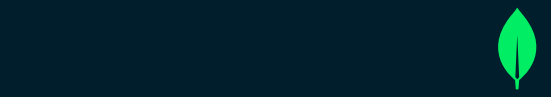
Intuitive and fast

Compared to storing data across multiple tables, a single document data structure:

- Presents a single place for the database to read and write data
- Denormalized data eliminates JOINs for most operational queries
- Simplifies query development and optimization




```
  _id: 12345678
> name: Object
> address: Array
> phone: Array
  email: "john.doe.@mongodb.com"
  dob: 1966-07-30 01:00:00:000
˅ interests:Array
      0: "Cycling"
      1: "IoT"
```

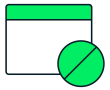


A row in a RDBMS table is most similar to a _____ in MongoDB?

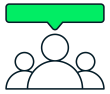
a) Object
b) Document
c) Collection
d) Record

Flexible Schema: unlocking developer velocity

Avoids need to update ORM class mappings and recompile programming language classes



Schema changes don’t lock the database, or cause performance degradation while tables are altered



Breaks down complex inter-group dependencies and expensive coordination before new code is released

Flexible: adapt to change

Add new fields dynamically at runtime

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

}
```

```
{
  "_id" : ObjectId("5ad88534e3632e1a35a58d00"),
  "name" : {
    "first" : "John",
    "last" : "Doe" },
  "address" : [
    { "location" : "work",
      "address" : {
        "street" : "16 Hatfields",
        "city" : "London",
        "postal_code" : "SE1 8DJ"},
      "geo" : { "type" : "Point", "coord" : [
         51.5065752,-0.109081]}},
+   {...}

  ],
   "phone" : [
     { "location" : "work",
       "number" : "+44-1234567890"},

+   {...}
  ],
  "dob" : ISODate("1977-04-01T05:00:00Z"),
  "retirement_fund" : NumberDecimal("1292815.75")
}
```

Data governance

JSON Schema

Enforces strict schema structure over a complete collection for data governance & quality

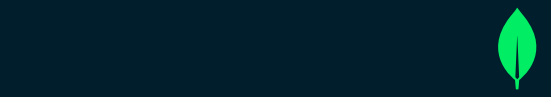
- Builds on document validation introduced by restricting new content that can be added to a document
- Enforces presence, type, and values for document content, including nested array
- Simplifies application logic

Tunable

Enforce document structure, log warnings, or allow complete schema flexibility

Queryable

Identify all existing documents that do not comply



What is the statement equivalent to ALTER TABLE in MongoDB?

a) ALTER COLLECTION

b) ALTER DOCUMENT

c) ALTER TABLE

d) None of the above

Fully featured secondary indexes—document optimized—extended beyond RDBMS experiences

Index types

Primary index

Every collection has a primary key index

Compound index

Index against multiple keys in the document

Multikey index

Index into arrays

Wildcard index

Auto-index all matching fields, sub-documents & arrays

Text indexes

Support for text searches

Geospatial indexes

2d & 2d sphere indexes for spatial geometries

Hashed indexes

Hashed based values for sharding

Index features

TTL indexes

Single field indexes, when expired delete the document

Unique indexes

Ensures value is not duplicated

Partial indexes

Expression based indexes, allowing indexes on subsets of data

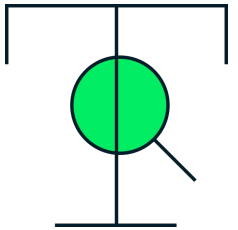
Case insensitive indexes

Supports text search using case insensitive search

Sparse indexes

Only index documents which have the given field

Powerful:
ATLAS Search $SEARCH single command integrating Lucene engine

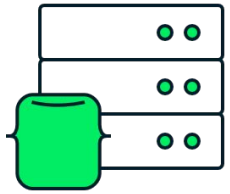


Multi-cloud global database

Freedom and flexibility

Broadest reach: private, hybrid, public

Laptop



Mainframe

Server

Self-managed
in the cloud

Database
as a Service

Consistent developer experience

Same codebase, same APIs, same tools, wherever you run

Don’t I get this freedom with relational databases?

Fragmentation across different cloud platforms

- Different versions
- Different database features and options
- Different scaling characteristics
- Different cross-region options
- Different monitoring, alerting, backup, and recovery
- Different security controls

Architectural mis-alignment

High availability: replica sets

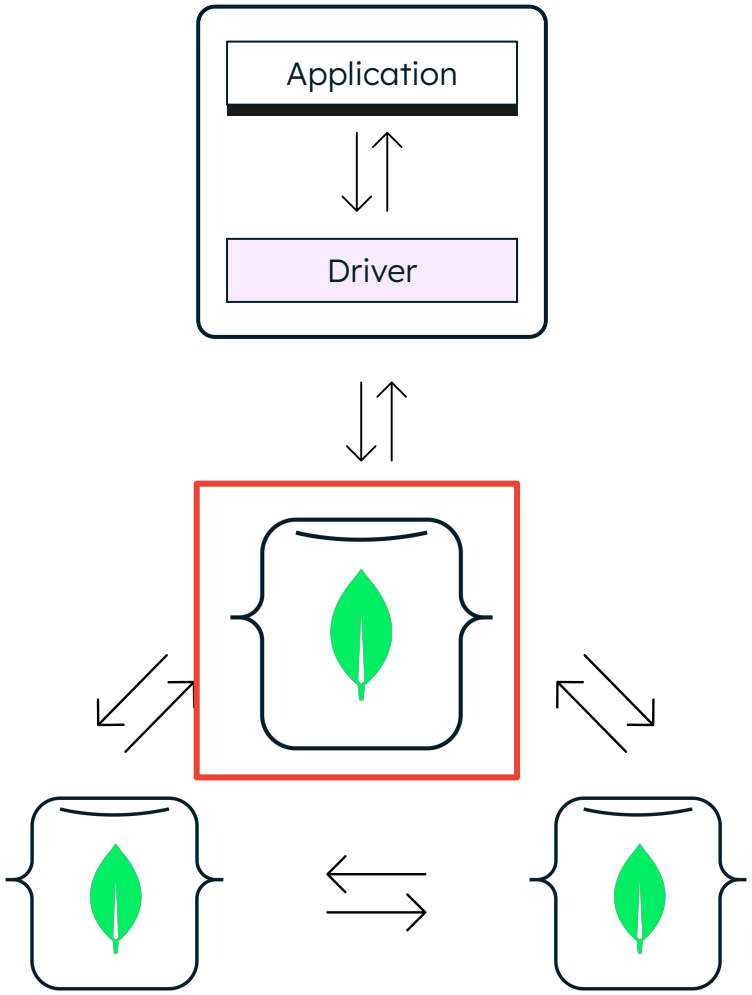
Automatic failover

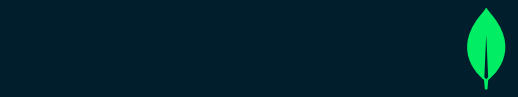
- Typical failover in 5 seconds or less
- Retriable reads and writes to catch temporary exceptions

Data center aware, tunable durability, and consistency

Addresses availability considerations:

- High availability
- Disaster recovery
- Maintenance





How many hours of downtime should I plan on for maintenance?

a) 0

b) 2

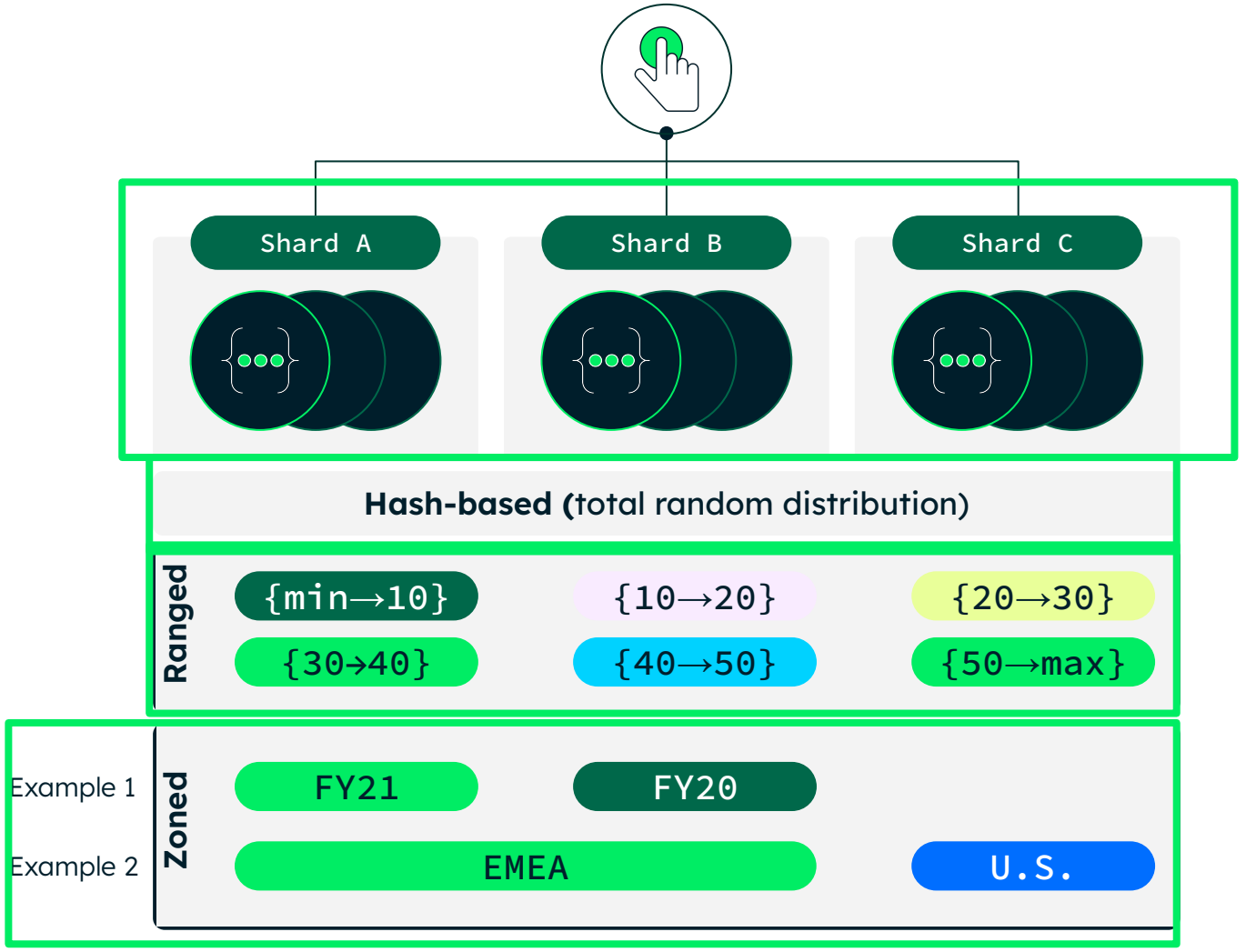
c) 1

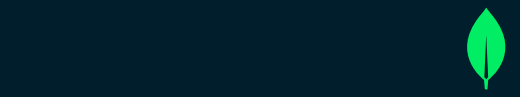
d) 3

Scalability: sharding

Native-sharding for horizontal scale-out

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





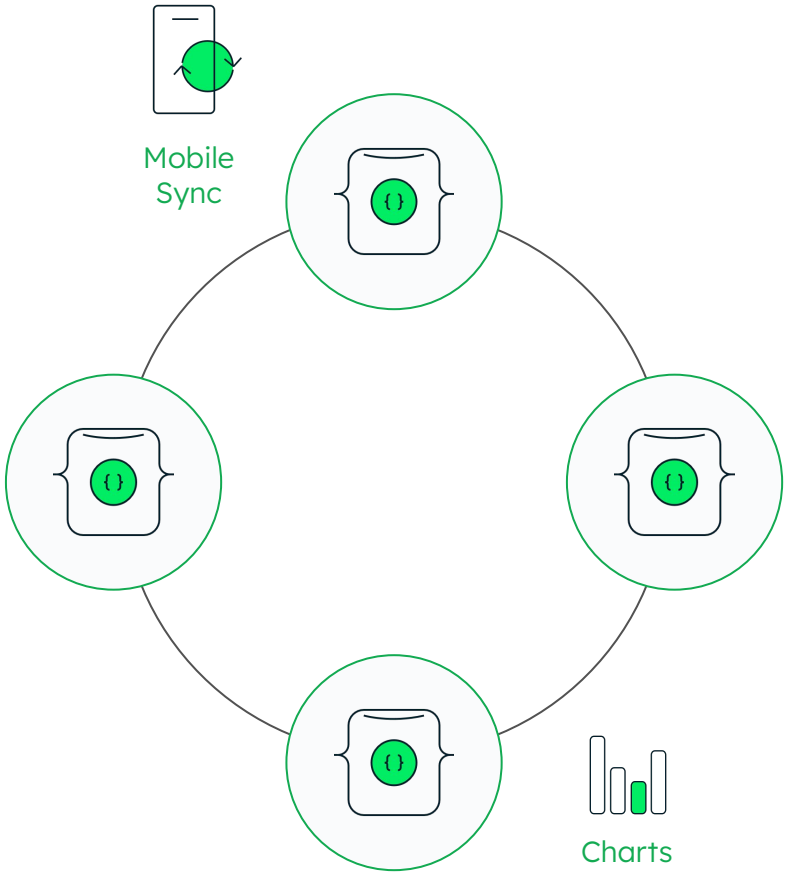
Which sharding policy would get data closest to the end user?

a) Hashed

b) Range

c) Zoned

Workload isolation



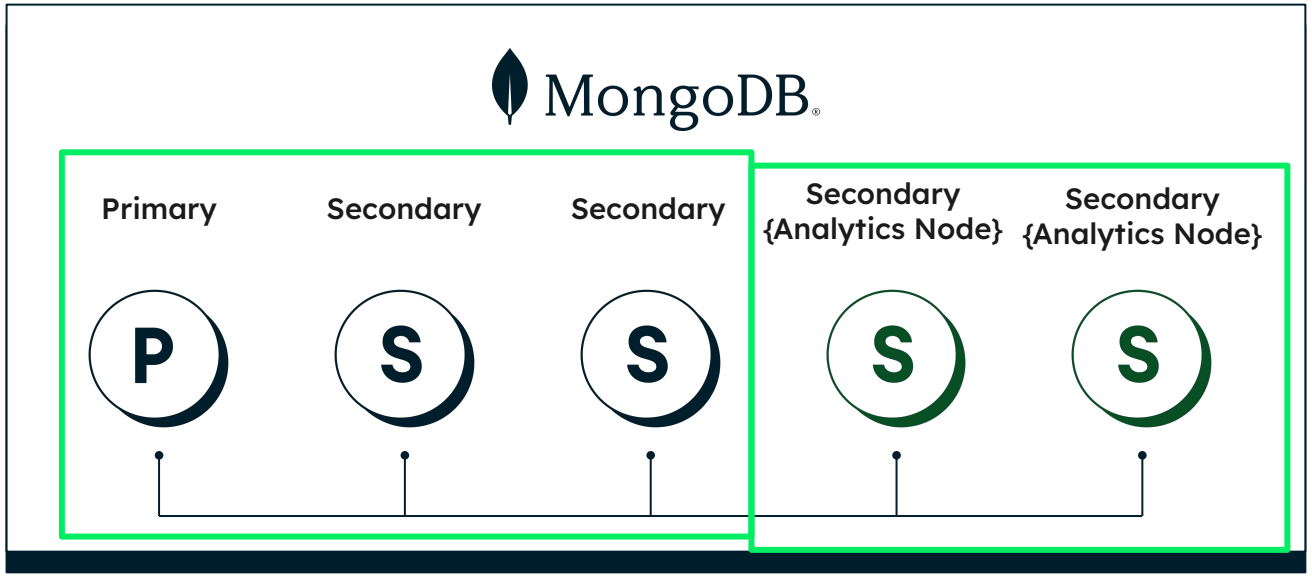
Enable different workloads on the same data

- 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

MongoDB workload isolation

Transactional Applications

Rich MongoDB Query API and distributed architecture allows you to run both Transactions and Analytics on the same cluster with no resource contention



Analytics Consumers

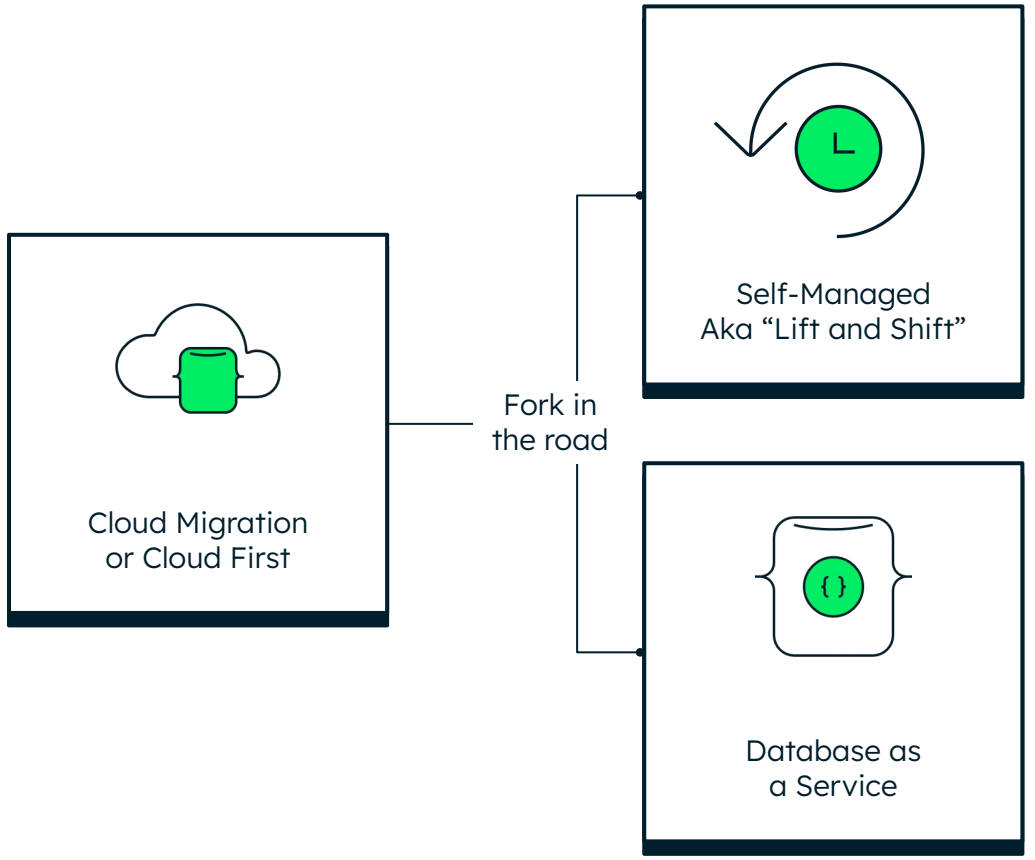
2 choices as you move to the cloud: Self-Managed or DBaaS

Self-managed

1. Provision instances and storage
2. Configure HA
3. Configure security
4. Configure backup/restore
5. Monitoring & alerting
6. Ongoing upgrades & maintenance

Fully-managed DBaaS

Choose instance, hit deploy, available in a couple of minutes



mongoDB® Atlas

MongoDB Atlas

Self-service and elastic

Global and highly available

Secure by default

Comprehensive monitoring

Managed backup

Cloud agnostic

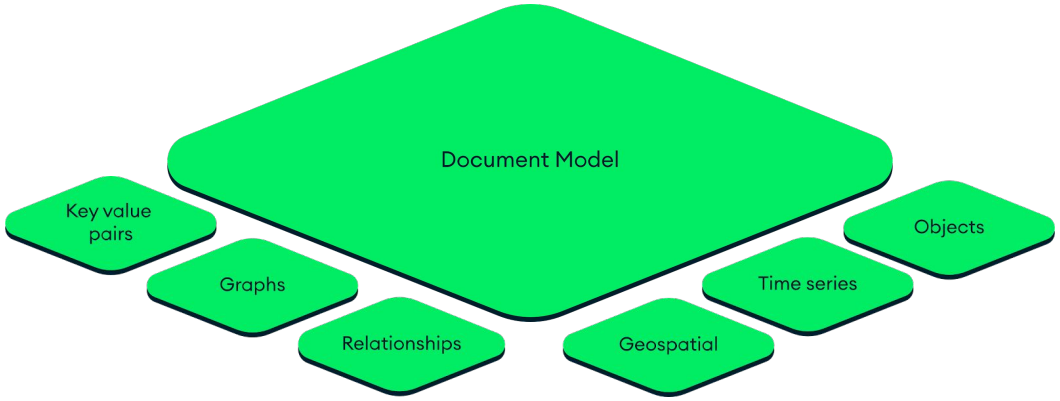
amazon web services™

Microsoft Azure

Google Cloud Platform

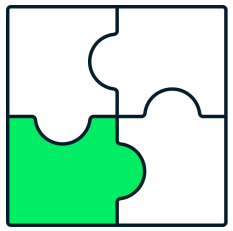
Application Data Platform

Built around a modern, distributed databases
built for developers



Data model that maps to how developers think/code; flexible while allowing data governance when needed

Strongly consistent, support for ACID transactions

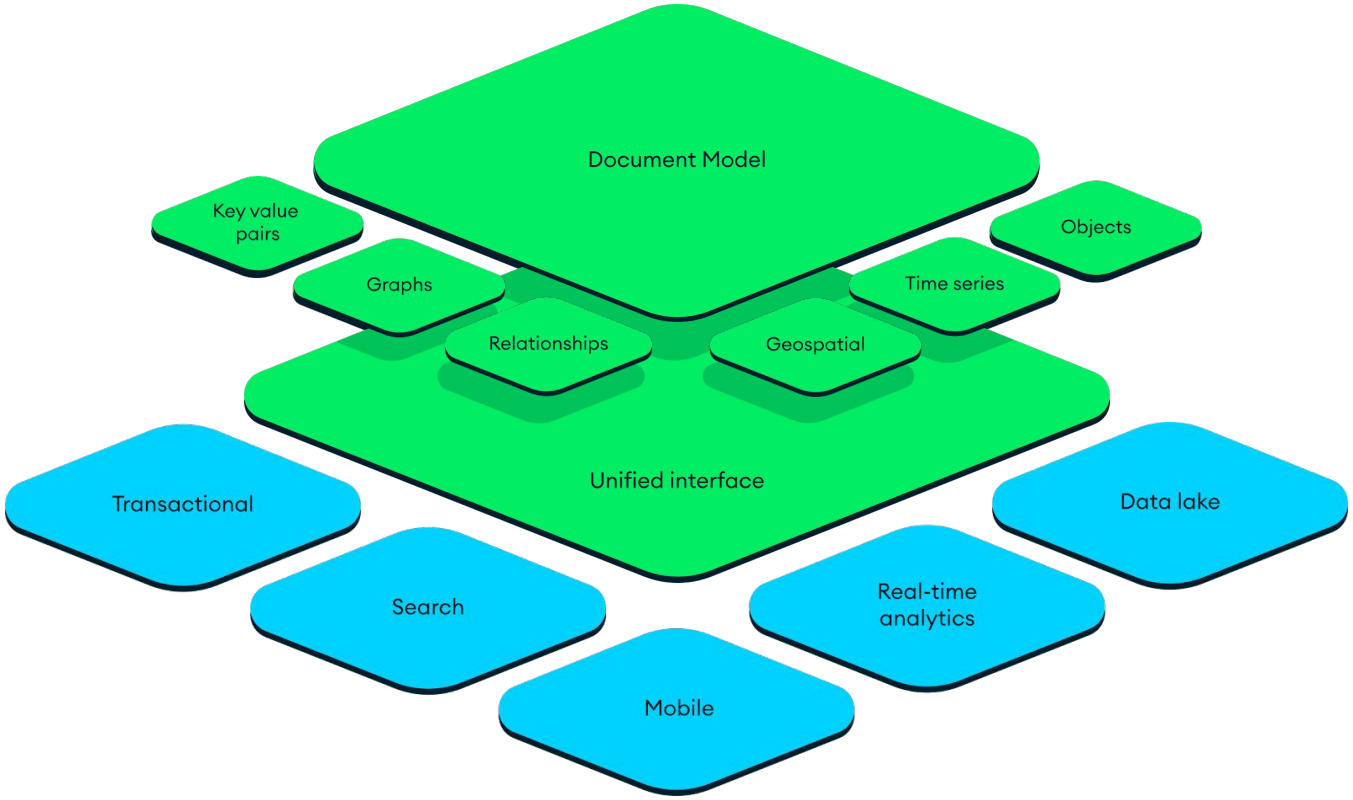


Address a range of application use cases w/o adding complexity

Able to support full-text search functionality for delivering a great user experience

Able to support data on mobile devices at the edge w/o having to manually sync data

Able to deliver real-time analytics on live data w/o having to move data back & forth



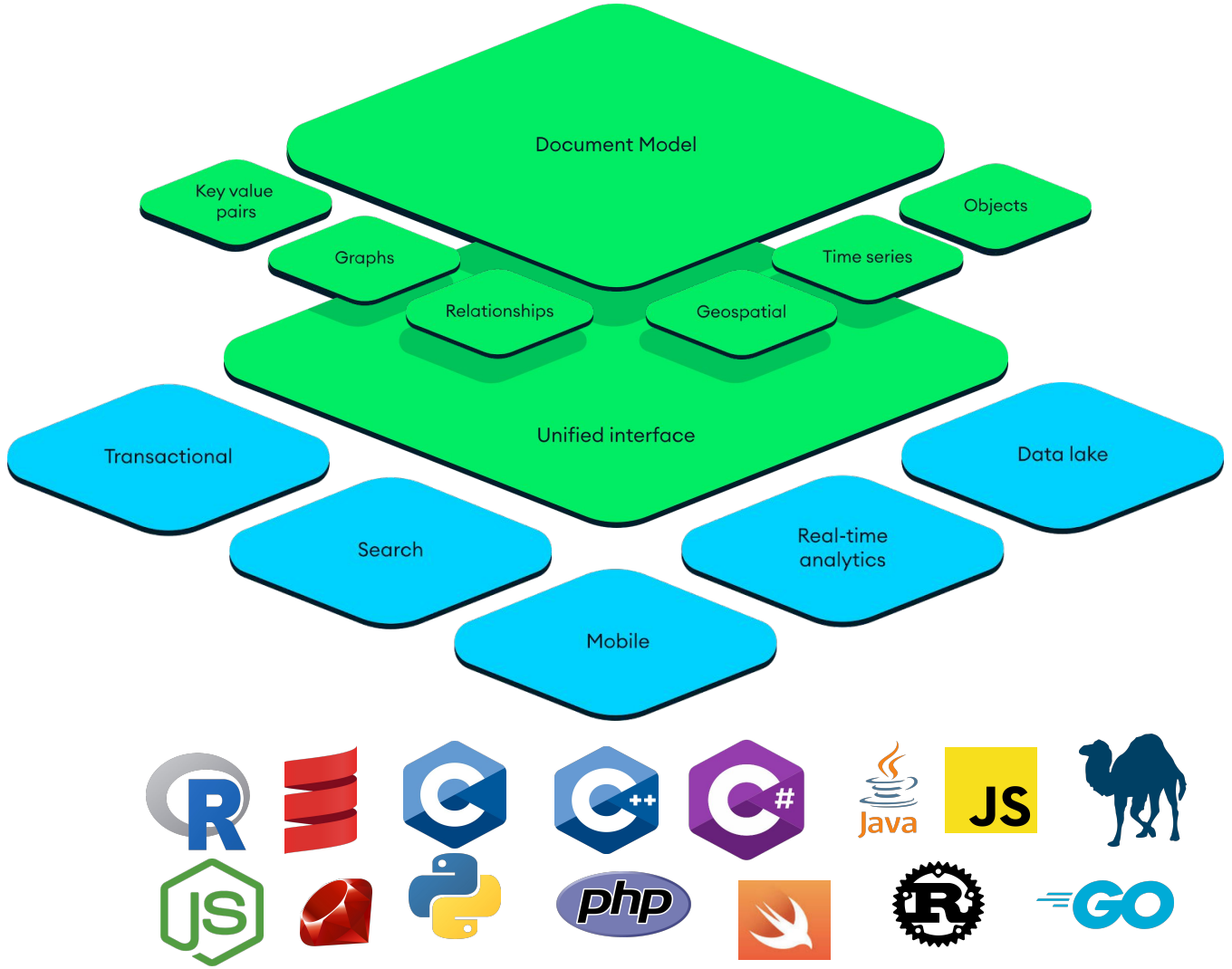
Address a range of application use cases w/o adding complexity

Able to support full-text search functionality for delivering a great user experience

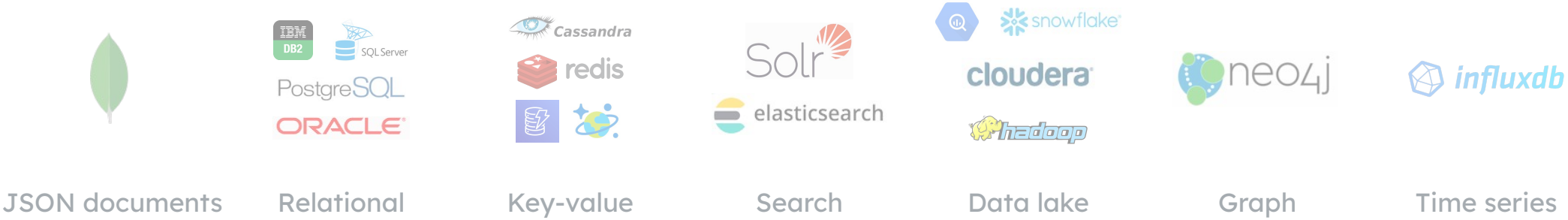
Able to support data on mobile devices at the edge w/o having to manually sync data

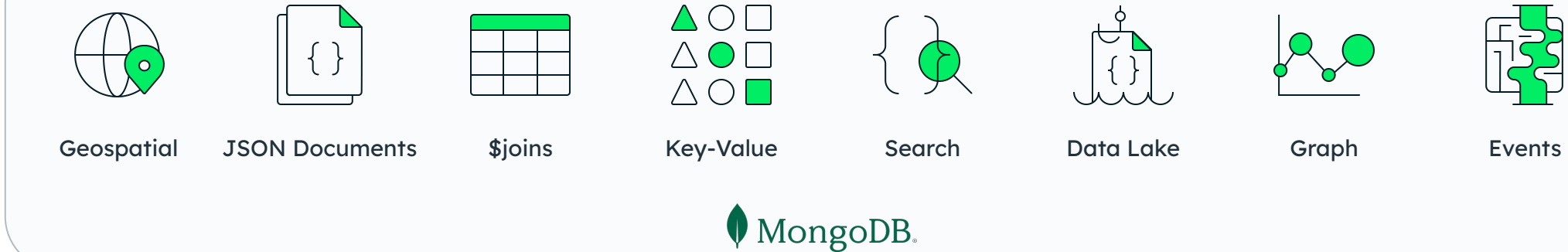
Able to deliver real-time analytics on live data w/o having to move data back & forth

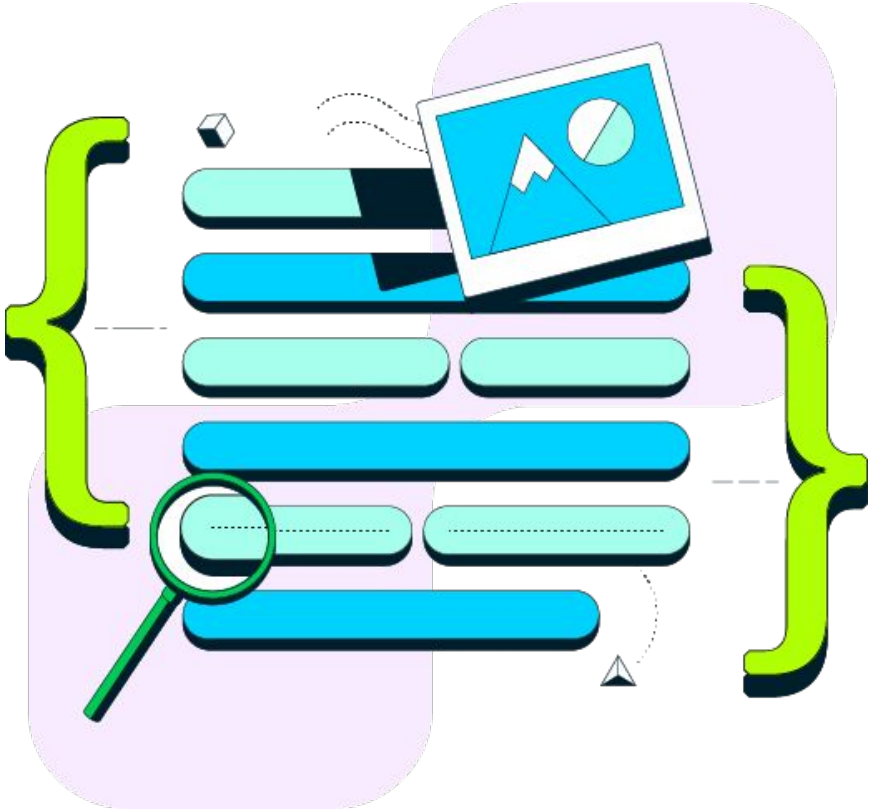
One easy query interface for a variety of workloads



Reduce complexity







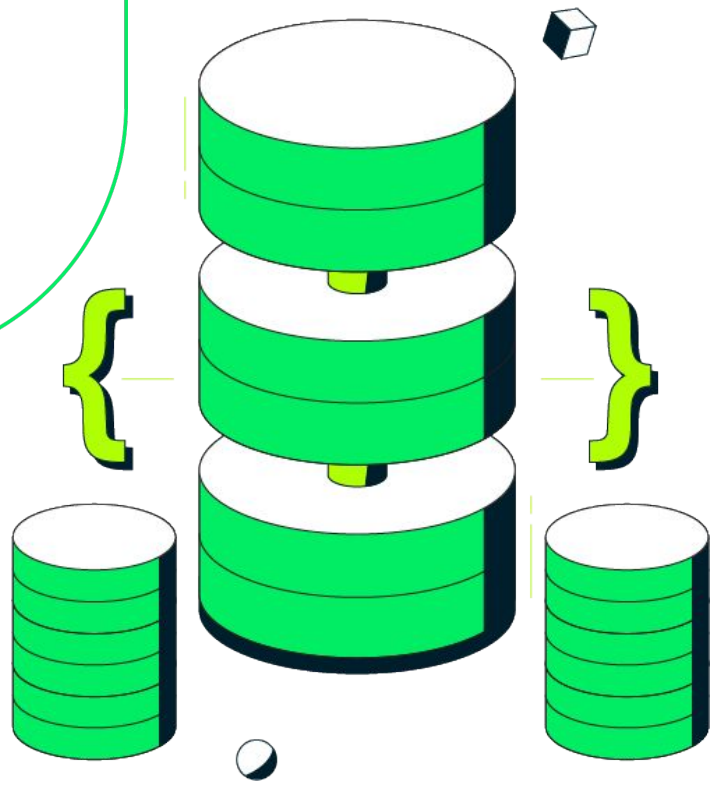
Atlas Search

Embedded full-text search in MongoDB Atlas that gives developers a seamless and scalable experience for building relevance-based app features

- Leverage the power of **Apache Lucene**, the world's leading search engine technology
- Eliminate the the need to run a separate search system alongside your database

Customers are delivering search features 30%-50% FASTER

Atlas Data Federation



Analyze cloud data

On-demand query service using the MongoDB API

Perform federated queries

Combine real-time app data with cloud data using a single connection string

Seamlessly work with data

Bring your own AWS S3 buckets so data stays in-place and in its native format

Atlas Charts

Fast to visualize

Built for the document model

No ETL, data movement or duplication required

Easy to start and share

Integrated with MongoDB Atlas

Interactive dashboards with secure sharing

Powerful insights + experiences

No-code aggregation, runs on secondary nodes

Embed charts in apps via IFrame or SDK