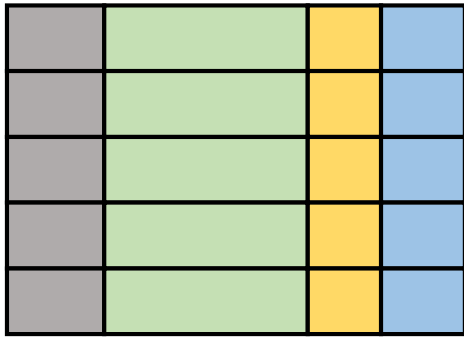


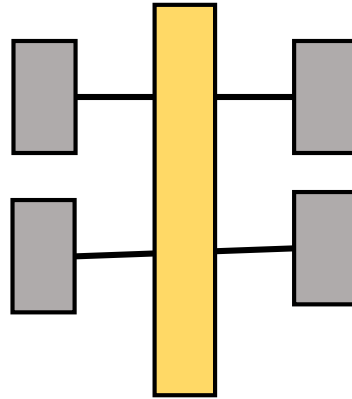


Day 26

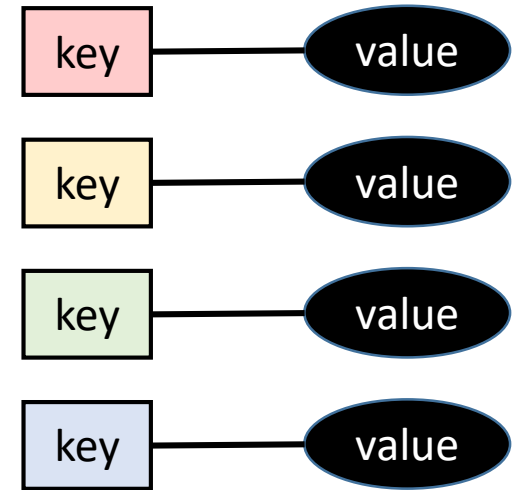
Types of Databases



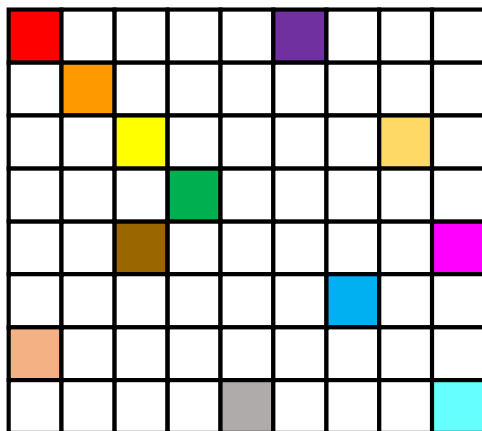
Relational



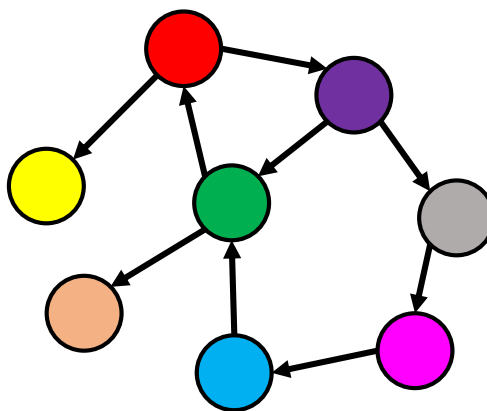
Analytical (OLAP)



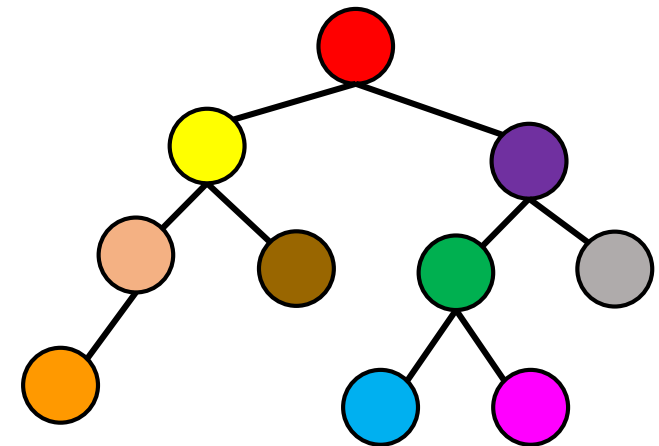
Key-Value



Column



Graph



Document



Examples of NoSQL Database

Key-Value



Document



mongoDB



CouchDB
relax



Solr

Graph



Column



Analytical

ORACLE





BASE Properties



Basically

The database appears to be available most of the time



Available



Soft state

Data in store may change over time even without input



Eventual Consistency

Data will be come consistent over time



Key-Value vs Document Database

Key-Value

- Uses keys to access the value
- Values are blobs - can store anything
 - Image, video, PDF, Word document
- Values are opaque - cannot be queried

Document

- Data are typically stored as JSON
- Data can be nested and hierarchical
- Any item in the document can be queried

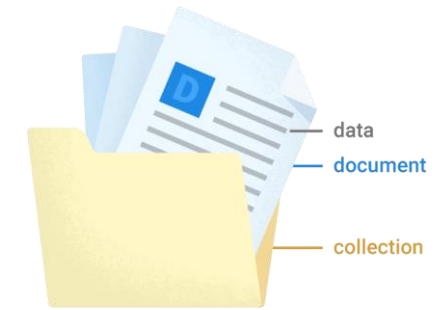


What is MongoDB?

- Document database
 - A document is a JSON object
- A record is a JSON object
 - Most RDBMS allows JSON data type as part of a record - a column
- Use cases
 - Big data
 - Content management
 - Complex data
 - Mobile apps



Terminology



RDBMS (MySQL)	Document (MongoDB)
Database	Database
Table	Collection
Record / Row	Document
Column	Field / Attribute
Index	Index
Joins	Linking and Embedding



An Example MongoDB Document

```
{
  Primary key → _id: ObjectId("54759eb3c090d83494e2d804",
    title: "MongoDB: The Definitive Guide",
    authors: [
      { _id: "kchodorow", name: "Kristina Chodorow" },
      { _id: "mdirold", name: "Mike Dirolf" }
    ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    thumbnail: BinData(0, "AREhMQ==")
    publisher: {
      name: "O'Reilly Media",
      founded: 1980,
      locations: ["CA", "NY" ]
    }
  }
}
```

MongoDB data type - BSON



Create a Cluster

mongoDB Atlas All Clusters

Singapore Usage This Month: \$0.00 details Chuk

CONTEXT

CHUK'S ORG - 2019-06-08 > PROJECT 0

Project 0

Clusters

ATLAS

Clusters

Data Lake BETA

SECURITY

Database Access

Network Access

Advanced

PROJECT

Access Management

Activity Feed

Alerts 0

Settings

SERVICES

Charts

Find a cluster...



Create a cluster

Choose your cloud provider, region, and specs.

Build a Cluster

Once your cluster is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#).



System Status: All Good Last Login: 137.132.218.222

©2019 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales



Cluster Configurations

[CLUSTERS](#) > CREATE NEW CLUSTER

Create New 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](#).

Global Cluster Configuration >

Cloud Provider & Region

GCP, Iowa (us-central1) >

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage)
Encrypted >

Additional Settings

MongoDB 4.0, No Backup >

Cluster Name

Cluster0 >

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Cancel

Create Cluster



Cluster Configuration - Provider and Region

Cloud Provider & Region

GCP, Singapore (asia-southeast1) ▾



Create a **free tier cluster** by selecting a region with **FREE TIER AVAILABLE** and choosing the **M0** cluster tier below.

★ Recommended region ⓘ

NORTH AMERICA / SOUTH AMERICA		EUROPE / MIDDLE EAST / AFRICA	ASIA PACIFIC
South Carolina (us-east1) ★		Belgium (europe-west1) ★ FREE TIER AVAILABLE	Taiwan (asia-east1) ★ FREE TIER AVAILABLE
N. Virginia (us-east4) ★		Finland (europe-north1) ★	Tokyo (asia-northeast1) ★
Los Angeles (us-west2) ★		London (europe-west2) ★	Osaka (asia-northeast2) ★
Iowa (us-central1) ★ FREE TIER AVAILABLE		Frankfurt (europe-west3) ★	Singapore (asia-southeast1) ★ FREE TIER AVAILABLE
Oregon (us-west1) ★		Netherlands (europe-west4) ★	Hong Kong (asia-east2) ★
Montreal (northamerica-northeast1) ★		Zurich (europe-west6) ★	Mumbai (asia-south1) ★
Sao Paulo (southamerica-east1) ★		AUSTRALIA	
		Sydney (australia-southeast1) ★	



Cloud Configuration - Cluster Tier

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage) Encrypted ▼

Base hourly rate is for a MongoDB replica set with **3 data bearing servers**.

Shared Clusters for development environments and low-traffic applications

Tier	RAM	Storage	vCPU	Base Price
✓ M0 Sandbox	Shared	512 MB	Shared	Free forever
M0 clusters are best for getting started, and are not suitable for production environments.				
100 max connections Low network performance 100 max databases 500 max collections				
M2	Shared	2 GB	Shared	\$9 / MONTH
M5	Shared	5 GB	Shared	\$25 / MONTH

Dedicated Clusters for development environments and low-traffic applications

Tier	RAM	Storage	vCPU	Base Price
M10	1.7 GB	10 GB	0.5 vCPUs	from \$0.10/hr
M20	3.75 GB	20 GB	1 vCPU	from \$0.23/hr

Dedicated Clusters for high-traffic applications and large datasets

Tier	RAM	Storage	vCPU	Base Price
------	-----	---------	------	------------



Create Cluster



Chuk ▾

[CLUSTERS](#) > [CREATE NEW CLUSTER](#)

Create New 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](#).

Global Cluster Configuration >

Cloud Provider & Region

GCP, Singapore (asia-southeast1) >

Cluster Tier

M0 Sandbox (Shared RAM, 512 MB Storage) >
Encrypted

Additional Settings

MongoDB 4.0, No Backup >



Cluster Name

Cluster0 >

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Cancel

Create Cluster





Creating Cluster

mongoDB Atlas All Clusters

Singapore Usage This Month: \$0.00 details Chuk

CONTEXT

Project 0

We are deploying your changes: 3 of 3 servers complete (current action: configuring MongoDB)

CHUK'S ORG - 2019-06-08 > PROJECT 0

ATLAS

Clusters

Data Lake BETA

SECURITY

Database Access

Network Access

Advanced

PROJECT

Access Management

Activity Feed

Alerts 0

Settings

SERVICES

Charts

Clusters

Find a cluster...

SANDBOX

Cluster0

Version 4.0.12

CONNECT

METRICS

COLLECTIONS

...

CLUSTER TIER

M0 Sandbox (General)

REGION

GCP / Singapore (asia-southeast1)

TYPE

Replica Set - 3 nodes

LINKED STITCH APP

None Linked

Your cluster is being created.

New clusters take between 7-10 minutes to provision.



System Status: All Good Last Login: 137.132.218.222

©2019 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales



Cluster Ready

mongoDB Atlas

All Clusters

Singapore Usage This Month: \$0.00 [details](#) Chuk

CONTEXT

CHUK'S ORG - 2019-06-08 > PROJECT 0

Project 0

Build a New Cluster

ATLAS

Clusters

Data Lake BETA

SECURITY

Database Access

Network Access

Advanced

PROJECT

Access Management

Activity Feed

Alerts 0

Settings

SERVICES

Charts

Find a cluster...

SANDBOX

Cluster0

Version 4.0.12

CONNECT METRICS COLLECTIONS ...

CLUSTER TIER

M0 Sandbox (General)

REGION

GCP / Singapore (asia-southeast1)

TYPE

Replica Set - 3 nodes

LINKED STITCH APP

None Linked

Operations R: W: 100.0/s

0

Last 6 Hours

Logical Size 0.0 B

512.0 MB max

0.0 B

Last 6 Hours

Connections 0

100 max

0

Last 6 Hours

Enhance Your Experience

For dedicated throughput, richer metrics and enterprise security options, upgrade your cluster now!

Upgrade

System Status: All Good Last Login: 137.132.218.222

©2019 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales



Create Users

mongoDB Atlas

All Clusters

Singapore Usage This Month:\$0.00 details Chuk

CONTEXT

CHUK'S ORG - 2019-06-08 > PROJECT 0

Project 0

Database Access

ATLAS

Clusters

Data Lake BETA

SECURITY

Database Access

Network Access

Advanced

PROJECT

Access Management

Activity Feed

Alerts 0

Settings

SERVICES

Charts

MongoDB Users MongoDB Roles

+ ADD NEW USER

User Name	Authentication Method	MongoDB Roles	Actions
bob	SCRAM	readWriteAnyDatabase@admin	EDIT DELETE
fred	SCRAM	atlasAdmin@admin	EDIT DELETE
wilma	SCRAM	readWriteAnyDatabase@admin	EDIT DELETE

System Status: All Good Last Login: 137.132.218.222

©2019 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales



Create User

×

Add New User

SCRAM Authentication

SCRAM is MongoDB's default authentication method.

bwayne

e.g. new-user_31

●●●●●●●●●●

SHOW

🔑 Autogenerate Secure Password

User Privileges

Atlas admin

Read and write to any database

Only read any database

Select Custom Role ▾

[Add Default Privileges](#)

☐ Save as temporary user

Cancel

Add User



Network Access

The screenshot shows the MongoDB Atlas 'Network Access' configuration page. A modal window titled 'Add Whitelist Entry' is open in the center. The modal contains the following elements:

- Title:** Add Whitelist Entry
- Instruction:** Add a whitelist entry using either CIDR notation or a single IP address. [Learn more.](#)
- Action Button:** ADD CURRENT IP ADDRESS
- Form Fields:**
 - Whitelist Entry:** A text input field containing '0.0.0.0/0'.
 - Comment:** A text input field containing 'All host'.
- Checkbox:** ☐ Save as temporary whitelist
- Buttons:** Cancel and Confirm

The background interface shows the 'Network Access' section of the MongoDB Atlas console, with a table of IP addresses and a sidebar with navigation options like 'Project 0', 'Clusters', 'Data Lake', 'Database Access', 'Network Access', 'Advanced', 'Access Management', 'Activity Feed', 'Alerts', 'Settings', 'Charts', and 'System Status'.



Accessing the Cluster Programmatically

The screenshot displays the MongoDB Atlas user interface with a modal dialog titled "Connect to Cluster0" open. The dialog has a close button (X) in the top right corner. It features a progress bar with three steps: "Setup connection security" (completed with a green checkmark), "Choose a connection method" (current step), and "Connect". Below the progress bar, there is a link to "View documentation" and a paragraph stating: "See methods to add data and diagnostics in the [Command Line Tools](#) shortcut from within your cluster."

The dialog lists three connection methods, each with an icon, a title, a description, and a right-pointing arrow:

- Connect with the Mongo Shell**: Mongo Shell with TLS/SSL support is required.
- Connect Your Application**: Get a connection string and view driver connection examples.
- Connect with MongoDB Compass**: Download Compass to explore, visualize, and manipulate your data.

At the bottom of the dialog are "Go Back" and "Close" buttons. The background interface shows the "mongoDB Atlas" header, "All Clusters" tab, and a sidebar menu with sections like "CONTEXT" (Project 0), "ATLAS" (Clusters, Data Lake BETA, SECURITY, PROJECT), and "SERVICES" (Charts). A cluster overview card for "Cluster0" is visible, showing "0.0 B" usage and a "512.0 MB max" limit. A "Build a New Cluster" button is also present.



Connection String

✕

Connect to Cluster0

✓ Setup connection security

Choose a connection method

Connect

Choose a connection method [View documentation](#)

See methods to add data and diagnostics in the [Command Line Tools](#) shortcut from within your cluster.

Connect with the Mongo Shell
Mongo Shell with TLS/SSL support is required

>

Connect Your Application
Get a connection string and view driver connection examples

>

Connect with MongoDB Compass
Download Compass to explore, visualize, and manipulate your data

>



✓ Setup connection security

✓ Choose a connection method

Connect

1

Choose your driver version

DRIVER

Node.js

⌵

VERSION

3.0 or later

⌵

2

Add your connection string into your application code

Connection String

Connection String Only

Full Driver Example

mongodb+srv://<username>:<password>@cluster0-ycqap.gcp.mongodb.net/te

<

>

⌵

⌶

Copy

Replace <password> with the password for the <username> user.
When entering your password, make sure that any special characters are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back

Close



Accessing the Cluster via Command Line Tools

CHUK'S ORG - 2019-06-08 > PROJECT 0

Clusters

Build a New Cluster

Find a cluster...

SANDBOX

Cluster0

Version 4.0.12

CONNECT

METRICS

COLLECTIONS

...

CLUSTER TIER

M0 Sandbox (General)

REGION

GCP / Singapore (asia-southeast1)

TYPE

Replica Set - 3 nodes

LINKED STITCH APP

None Linked

Edit Configuration

Command Line Tools

Load Sample Dataset

Terminate

Operations R: 0 W: 0

100.0/s

Logical Size 0.0 B

512.0 MB
max

Last 6 Hours

Enhance Your Experience

For dedicated throughput, richer metrics and enterprise security options, upgrade your cluster now!

Upgrade





MongoDB GUI Client

Robo 3T - 1.3

File View Options Window Help

Cluster0 (3)

- Replica Set (3 nodes)
 - cluster0-shard-00-00-ycqap....
 - cluster0-shard-00-01-ycqap....
 - cluster0-shard-00-02-ycqap....
- System
 - mflix
 - Collections (6)
 - comments
 - movies**
 - sessions
 - theaters
 - users
 - watching_pings
 - Functions
 - Users

Welcome x * db.getCollection('movi... x

Cluster0 cluster0-shard-00-00-ycqap.gcp.mongodb.net:27017 mflix

```
db.getCollection('movies').findOne({})
```

0.461 sec.

Key	Value	Type
(1) ObjectId("573a1390f29313caabcd4132")	{ 15 fields }	Object
_id	ObjectId("573a1390f29313caabcd4132")	ObjectId
title	Carmencita	String
year	1894	Int32
runtime	1	Int32
cast	[1 element]	Array
[0]	Carmencita	String
poster	http://ia.media-imdb.com/images/M/MV5BMjAzNDEwMzk3O...	String
plot	Performing on what looks like a small wooden stage, wear...	String
fullplot	Performing on what looks like a small wooden stage, wear...	String
lastupdated	2015-08-26 00:03:45.040000000	String
type	movie	String
directors	[1 element]	Array
[0]	William K.L. Dickson	String
imdb	{ 3 fields }	Object
rating	5.9	Double
votes	1032	Int32
id	1	Int32
countries	[1 element]	Array
[0]	USA	String
rated	NOT RATED	String
genres	[2 elements]	Array
[0]	Documentary	String
[1]	Short	String

Logs



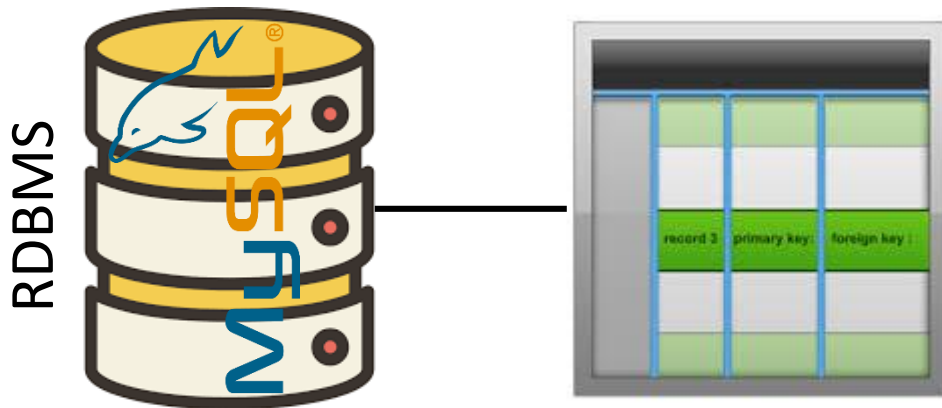
MongoDB - How its Different from RDBMS

- All documents has a primary key called `_id`
 - Either assigned by the user
 - Or assigned by MongoDB when document is added to collection
- Collections are created on demand
 - When the first document is added to a collection
- Documents in a collection do not have to be identical
 - Same schema
- Mongo allows you to query over non existence collections and attributes
 - Will not flag as error

Schemas

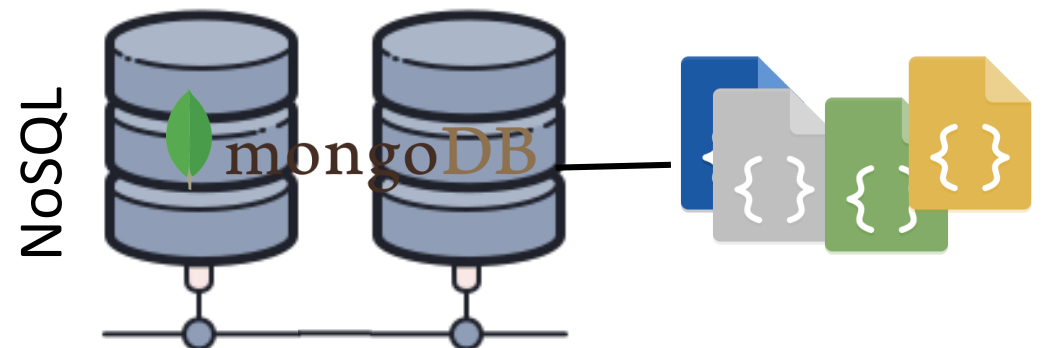
- Explicit schema
- Schema on write
 - Inserts must conform to the predefined schema

```
create table customer (  
  cust_id int primary key,  
  name varchar(64),  
  ...  
)
```



- Implicit schema
- Schema on read
 - Find out about the schema when a document is read

```
db.customer.insert({  
  _id: ObjectId("abc123"),  
  name: "Fred",  
  ...  
})
```





MongoDB Shell Basics

- Connecting to cluster

```
mongo "mongodb+srv://<cluster_name>.gcp.mongodb.net" --username <user_name>
```

- List databases

```
show databases
```

- Select a database

- The selected database is held in a variable call db

```
use <database_name>
```

- List collections

```
show collections
```



Connecting to MongoDB

Dependencies

ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools

DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Web

WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data MongoDB

NOSQL

Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Add Spring Data
MongoDB dependency





MongoTemplate

- SpringBoot's API to query and mutate Mongo database
 - Like JdbcTemplate and RedisTemplate, wrapper over Mongo library
 - Will use it for CRUD operations, will use R with the native Mongo library

@Bean

```
public MongoTemplate mongoTemplate() {  
    return new MongoTemplate(mongoClient(), "movies");  
}
```

Database name



Create the template with the
client and the database name

An instance of MongoClient



Configuring MongoDB Connection

Get an existing database or
create a new database

@Configuration

```
public class AppConfig {  
    @Value("${MONGO_URL}")  
    private String connectionString;  
  
    private MongoClient client = null;
```

@Bean

```
public MongoClient mongoClient() {  
    if (null == client)  
        client = MongoClient.create(connectionString);  
    return client;  
}
```

Create an instance of MongoClient
Communicates with MongoDB
Singleton

@Bean

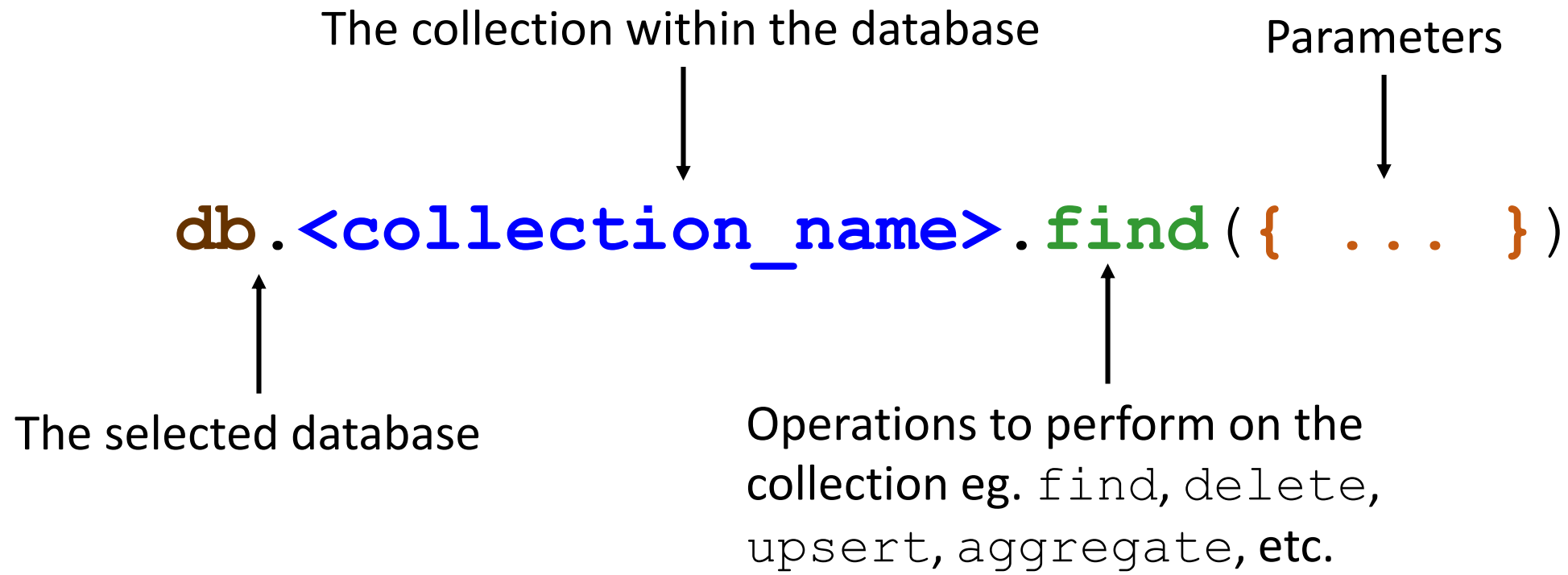
```
public MongoTemplate mongoTemplate() {  
    return new MongoTemplate(mongoClient(), "movies");  
}
```

Database name

Create the template with the
client and the database name



Database Operations





Sample Data - tv_shows

```
{
  _id: ObjectId("abc123"),
  title: "Dark Crystal",
  year: 1982,
  plot: "On another planet in the distant past, a Gelfling embarks on a
quest to find the missing shard of a magical crystal, and so restore
order to his world.",
  directors: [ "Jim Henson", "Frank Oz" ],
  imdb: {
    rating: 7.2,
    votes: 56021,
    id: "tt0083791"
  },
  gross: 41373966
  image: "https://....jpg",
  genres: [ "Adventure", "Family", "Fantasy" ],
  rating: "PG"
}
```

Read - find

```
db.tv_shows.find({ title: 'Dark Crystal' })
```

Find the document with the value
'Dark Crystal' in `title` attribute

```
db.tv_shows.find({  
  language: 'English',  
  type: 'Scripted'  
})
```

Find the English scripted TV shows



Read - find

```
@Autowired  
private MongoTemplate mongoTemplate;
```

```
Criteria criterial = Criteria.where("title").is("Dark Crystal");  
Query query = Query.query(criterial);
```

Create a predicate

```
List<Document> result = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```

Use the predicate to create a filter

```
Criteria criterial = Criteria  
    .where("language").is("English")  
    .and("type").is("Scripted");  
Query query = Query.query(criterial);
```

Search documents from the
tv_shows collection with the filter.
Returns a list of
org.bson.Document objects

```
List<Document> result = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```

Predicate with multiple and
conditions



Working with Document

- List of `getXXX (String fieldName)` method to return attributes from JSON document

```
getString("title"), getDouble("rating"),  
getBoolean("married"), getDate("dob")
```

- To JSON string

```
return ResponseEntity.ok(doc.toJson());
```

- From JSON string

```
JsonObject json = ...
```

```
Document doc = Document.parse(json.toString());
```

Read - find

```
db.tv_shows.find({  
  title: {  
    $regex: "crystal",  
    $options: "i"  
  }  
})
```

Find the document with the pattern 'crystal' in `title` attribute. The `i` denotes case insensitive search

```
db.tv_shows.find({  
  title: { $regex: "crystal", $options: "i" },  
  year: 1982  
})
```

Find the document with the pattern 'crystal' in `title` attribute and with the value 1982 in `year` attribute



Read - find

```
@Autowired
private MongoTemplate mongoTemplate;

Criteria criterial = Criteria.where("title")
    .regex("crystal", "i");
Query query = Query.query(criterial);

List<Document> result = mongoTemplate.find(
    query, Document.class, "tv_shows");

Criteria criterial = Criteria.andOperator(
    Criteria.where("title").regex("crystal", "i"),
    Criteria.where("year").is(1982)
);
Query query = Query.query(criterial);

List<Document> result = mongoTemplate.find(
    query, Document.class, "tv_shows");
```



An alternative for
constructing an and
predicate



Read - find

Default type for document id

```
{  
  _id: ObjectId("abc123"),  
  title: "Dark Crystal",  
  ...  
}
```

```
db.tv_shows.find({ _id: 'abc123' })
```



Will not return any result

```
db.tv_shows.find({ _id: ObjectId('abc123') })
```





Read - find

Create an instance of ObjectId

```
public Optional<Document> findTVShowById(String id) {  
    ObjectId docId = new ObjectId(id);  
    return Optional.ofNullable(  
        mongoTemplate.findById(docId, Document.class, "tv_shows");  
    );  
}
```

Either finds the document or returns null



Addressing Attributes

- Single attribute
 - `title`
- Attribute in an embedded document
 - `imdb.votes`
- Projected field
 - Value of a particular field in the current document
 - Eg. `$gross`
 - Used in aggregation

```
{
  _id: ObjectId("abc123"),
  title: "Dark Crystal",
  year: 1982,
  plot: "On another ..",
  directors: [ "Jim Henson", "Frank Oz" ],
  imdb: {
    rating: 7.2,
    votes: 56021,
    id: "tt0083791"
  },
  gross: 41373966
  image: {
    medium: "https://....jpg",
    original: "https://....jpg"
  }
  genres: [ "Adventure", "Family", "Fantasy" ],
  rating: "PG"
}
```



List of Operators

- Logical
 - `$and`, `$or`, `$not`, `$nor`
- Comparison
 - `$eq`, `$neq`, `$gt`, `$gte`, `$lt`, `$lte`, `$in`, `$nin`
- Element query
 - `$exists`, `$type`
- See <https://docs.mongodb.com/manual/reference/operator/>

Read - find

```
db.tv_shows.find({ year: { $gte: 1984 } })
```

Find the document where the `year` attribute is greater than or equal to 1984

```
db.tv_shows.find({
  $and: [
    { year: { $gte: 1984 } },
    { "imdb.rating": { $gt: 5.5 } }
  ]
})
```

Find all documents where the IMBD rating is greater than 5.5 starting from the year 1984



Read - find


```
Query query = Query.query(  
    Criteria.where("year").gte(1984)  
);  
List<Document> results = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```

```
Query query = Query.query(  
    Criteria.andOperator(  
        Criteria.where("year").gte(1984),  
        Criteria.where("imdb.rating").gt(5.5)  
    )  
);  
List<Document> results = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```

Read - find

```
db.tv_shows.find({  
  genre: {  
    $in: [ "Drama", "Horror", "Adventure" ]  
  }  
})
```

Find the documents where the genre is one of the value in the array



```
db.tv_shows.find({  
  awards: { $exists: true }  
})
```

Find the document which has the awards attribute





Read - find

```
Query query = Query.query(  
    Criteria.where("genre")  
        .in("Drama", "Horror", "Adventure")  
);  
List<Document> results = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```

```
Query query = Query.query(  
    Criteria.where("awards")  
        .exists(true)  
);  
List<Document> results = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```



Read - find

```
{ ... genres: [ "Science-Fiction", "Drama", "Crime" ] },  
{ ... genres: [ "Drama", "Crime" ] },  
{ ... genres: [ "Thriller", "Drama", "Crime" ] },
```

```
db.inventory.find({ "genres": "Drama" })
```

Find any document with Drama

```
db.inventory.find({ "genres":  
  { $in: [ "Science-Fiction", "Crime" ] } })
```

Find any document with Science-Fiction or Crime

```
db.inventory.find({ "genres":  
  { $all: [ "Science-Fiction", "Crime" ] } })
```

Find any document with Science-Fiction and Crime



Read - find

```
Query query = Query.query(  
    Criteria.where("genre").in("Drama")  
);  
List<Document> results = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```

```
Query query = Query.query(  
    Criteria.where("genre").in(List.of("Science-Fiction", "Crime"))  
);  
List<Document> results = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```

```
Query query = Query.query(  
    Criteria.where("genre").all(List.of("Science-Fiction", "Crime"))  
);  
List<Document> results = mongoTemplate.find(  
    query, Document.class, "tv_shows");
```



Read - Miscellaneous

```
db.tv_shows.distinct('rated')
```

Returns an array of unique values for that field

```
db.tv_shows.find({ year: { $gte: 2000 } }).count()
```

Count the number for documents returned by the query



Read - Miscellaneous

```
List<String> ratings = mongoTemplate.findDistinct(  
    new Query(), "rated", "tv_shows", String.class);
```

The type of the `rated` property

```
Query query = Query.query(  
    Criteria.where("year").gte(2000)  
);  
long count = mongoTemplate.count(query, "tv_shows");
```

Paginating Results

- Limiting the number of result

```
db.tv_shows.find({  
  awards: { $exists: true }  
}) .limit(10)
```

Display only 10 documents

- Return the result starting from a specified document

```
db.tv_shows.find({  
  awards: { $exists: true }  
}) .skip(10)
```

Only return the results starting from the 11th document

```
db.tv_shows.find({  
  awards: { $exists: true }  
}) .skip(10)  
  .limit(5)
```

Return documents from 11 to 15
Note order of `skip()` and `limit()` is unimportant. MongoDB will always perform `skip()` first before `limit()`



Paginating Results

```
Query query = Query.query(  
    Criteria.where("awards").exists(true)  
) .limit(10);
```

```
Query query = Query.query(  
    Criteria.where("awards").exists(true)  
) .skip(10);
```

```
Query query = Query.query(  
    Criteria.where("awards").exists(true)  
) .limit(10).skip(10);
```

Sorting Results

- Sorting the result

- -1 ascending, 1 - descending

```
db.tv_shows.find({
  awards: { $exists: true }
}).sort({ title: -1 })
```

```
db.tv_shows.find({
  awards: { $exists: true }
}).skip(10)
.limit(5)
.sort({ title: -1, gross: 1 })
```



Sorting on some attributes may require an index
`db.tv_shows.createIndex({ title: 1 })`
 Indices will be covered in the next section

Sort the result based on the following
 2 attributes

MongoDB perform the `skip()`, `limit()`
`sort()` in the order show regardless of
 the order in which they are called



Sorting Results

```
Query query = Query.query(  
    Criteria.where("awards").exists(true)  
) .with(  
    Sort.by(Sort.Direction.ASC, "title")  
);
```

```
Query query = Query.query(  
    Criteria.where("awards").exists(true)  
) .with(  
    Sort  
        .by(Sort.Direction.ASC, "title")  
        .by(Sort.Direction.DESC, "gross")  
) .skip(10) .limit(5);
```

Projections

- By default queries returns all fields
- Projection allows you to specify to MongoDB to only return a subset of fields

```
db.tv_shows.find(
  { title: /. *kill.* /, year: 1982, },
  { _id: 0, title: 1, plot: 1, gross: 1, image: 1 }
)
```

Will always return `_id` by default. Need to explicitly disable it if you're not interested in the `_id`

Will only return these attributes



Projections

```
Query query = Query.query(
    Criteria.where("title").regex(".*kill.*")
        .and("year").is(1982)
);
query.fields()
    .exclude("_id")
    .include("title", "plot", "gross", "image");

List<Document> results = mongoTemplate.query(
    query, Document.class, "tv_shows");
```



Predicate Summary

Property to apply the operation to

Criteria.where("property_name").<operation>()

```
Criteria.where("title").is("Titanic")           { title: "Titanic" }
Criteria.where("year").gt(2000)                  { year: { $gt: 2000 } }
Criteria.where("genre")                          { genre: { $in: [ "Horror", "Family" ] } }
  .in("Horror", "Family")                       { awards: { $exists: true } }
Criteria.where("awards").exists(true)            { $and: [
Criteria.andOperator(List.of(                    { title: { $regex: ".*land.*" } },
  Criteria.where("title").regex(".*land.*"),    { rated: "PG-13" }
  Criteria.where("rated").is("PG-13"))          ]}
)
```



Unused



Architecture

