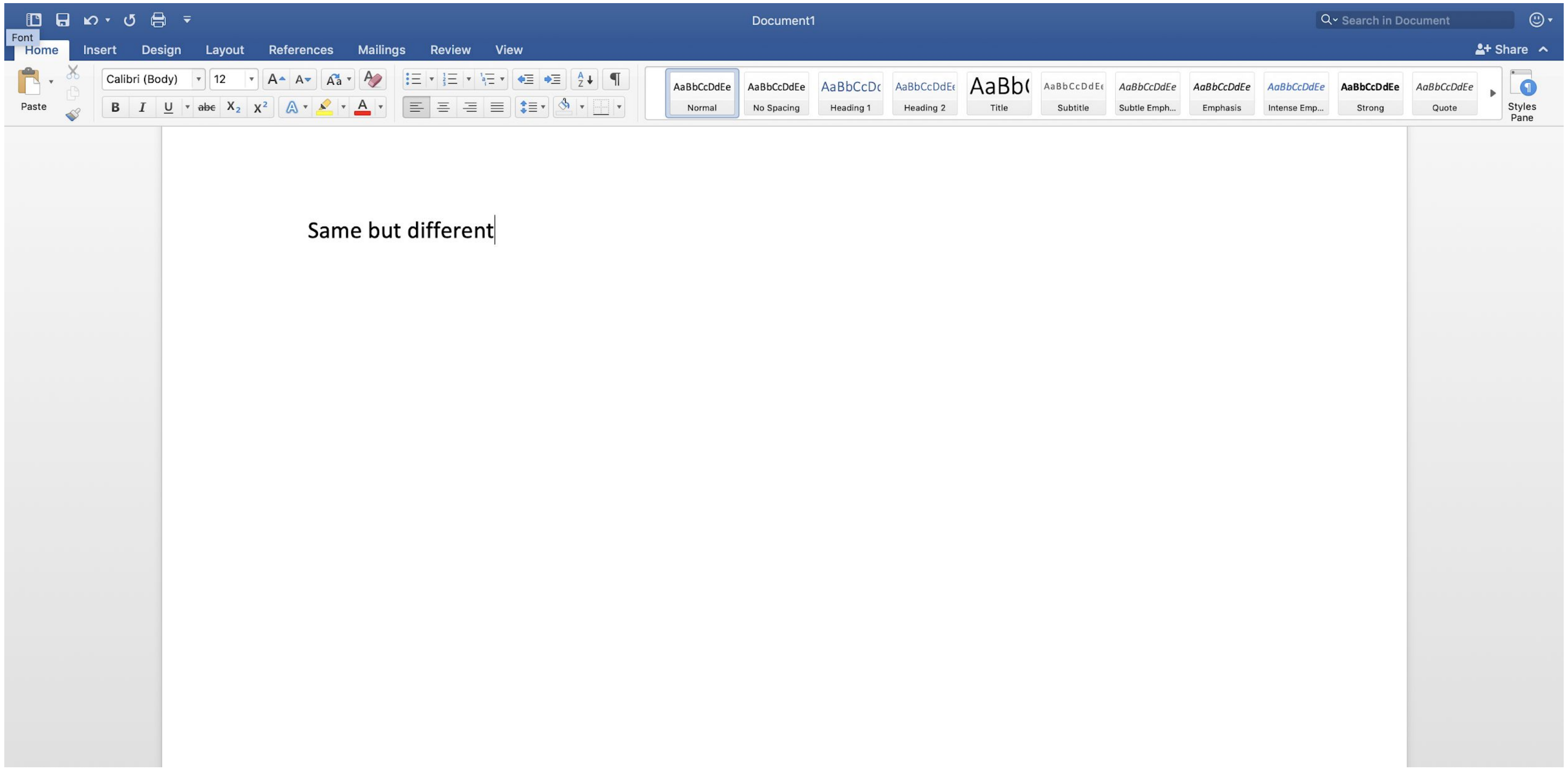
Getting Started with MongoDB

Nooruddin Abbas Ali
Principal Solutions Architect

What are we talking about ?

1. Some terminology and concepts
2. How can I run MongoDB ?
3. MongoDB Architecture
4. Tips to get more out of your MongoDB
5. Useful links to follow up
6. Q&A

MongoDB stores data in documents



Same but different

MongoDB stores data in documents

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

MongoDB stores data in documents

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

Fields

MongoDB stores data in documents

```
first_name: "Paul",
surname: "Miller",
cell: "447557505611",
city: "London",
location: [45.123,47.232],
profession: ["banking", "finance", "trader"],
cars: [
    {
        model: "Bentley",
        year: 1973
    },
    {
        model: "Rolls Royce",
        year: 1965
    }
]
}
```

Values

mongoDB.

MongoDB stores data in documents

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

Strings

MongoDB stores data in documents

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

int32

long

double

decimal

MongoDB stores data in documents

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

MongoDB stores data in documents

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

Arrays

MongoDB stores data in documents

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

Array

of sub-documents

MongoDB stores data in documents

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

mongoDB.

Modelling data in MongoDB

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232

Modelling data in MongoDB

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232

Professions

ID	user_id	profession
10	1	banking
11	1	finance
12	1	trader

Modelling data in MongoDB

```
{
   first_name: "Paul",
   surname: "Miller",
   cell: "447557505611",
   city: "London",
   location: [45.123,47.232],
   profession: ["banking", "finance", "trader"],
   cars: [
      {
         model: "Bentley",
         year: 1973
      },
      {
         model: "Rolls Royce",
         year: 1965
      }
   ]
```

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232

Professions

ID	user_id	profession
10	1	banking
11	1	finance
12	1	trader

Cars

ID	user_id	model	year
20	1	Bentley	1973
21	1	Rolls Royce	1965

Modelling data in MongoDB

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232

Professions

ID	user_id	profession
10	1	banking
11	1	finance
12	1	trader

Cars

ID	user_id	model	year
20	1	Bentley	1973
21	1	Rolls Royce	1965

Collection vs Tables

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

```
{
    first_name: ”Lauren",
    surname: ”Schaefer",
    cell: ”1235552222",
    city: ”Lancaster",
    profession: [”software engineer", ”developer advocate"],
}
```

```
{
    first_name: ”Sydney",
    surname: ”Schaefer",
    city: ”Lancaster",
    school: ”Daisy’s Daycare”
}
```

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232
2	Lauren	Schaefer	1235552222	Lancaster	NULL	NULL
3	Sydney	Schaefer	NULL	Lancaster	NULL	NULL

Collection vs Tables

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232
2	Lauren	Schaefer	1235552222	Lancaster	NULL	NULL
3	Sydney	Schaefer	NULL	Lancaster	NULL	NULL

```
{
    first_name: ”Lauren",
    surname: ”Schaefer",
    cell: ”1235552222",
    city: ”Lancaster",
    profession: [”software engineer", ”developer advocate"],
}
```

```
    rst_name: ”Sydney",
    urname: ”Schaefer",
    ity: ”Lancaster",
    chool: ”Daisy’s Daycare”
}
```

Collection vs Tables

```
{
    first_name: "Paul",
    surname: "Miller",
    cell: "447557505611",
    city: "London",
    location: [45.123,47.232],
    profession: ["banking", "finance", "trader"],
    cars: [
        {
            model: "Bentley",
            year: 1973
        },
        {
            model: "Rolls Royce",
            year: 1965
        }
    ]
}
```

```
{
    first_name: "Lauren",
    surname: "Schaefer",
    cell: "1235552222",
    city: "Lancaster",
    profession: ["software engineer", "developer advocate"],
}
```

Users

ID	first_name	surname	cell	city	location_x	location_y
1	Paul	Miller	447557505611	London	45.123	47.232
2	Lauren	Schaefer	1235552222	Lancaster	NULL	NULL
3	Sydney	Schaefer	NULL	Lancaster	NULL	NULL

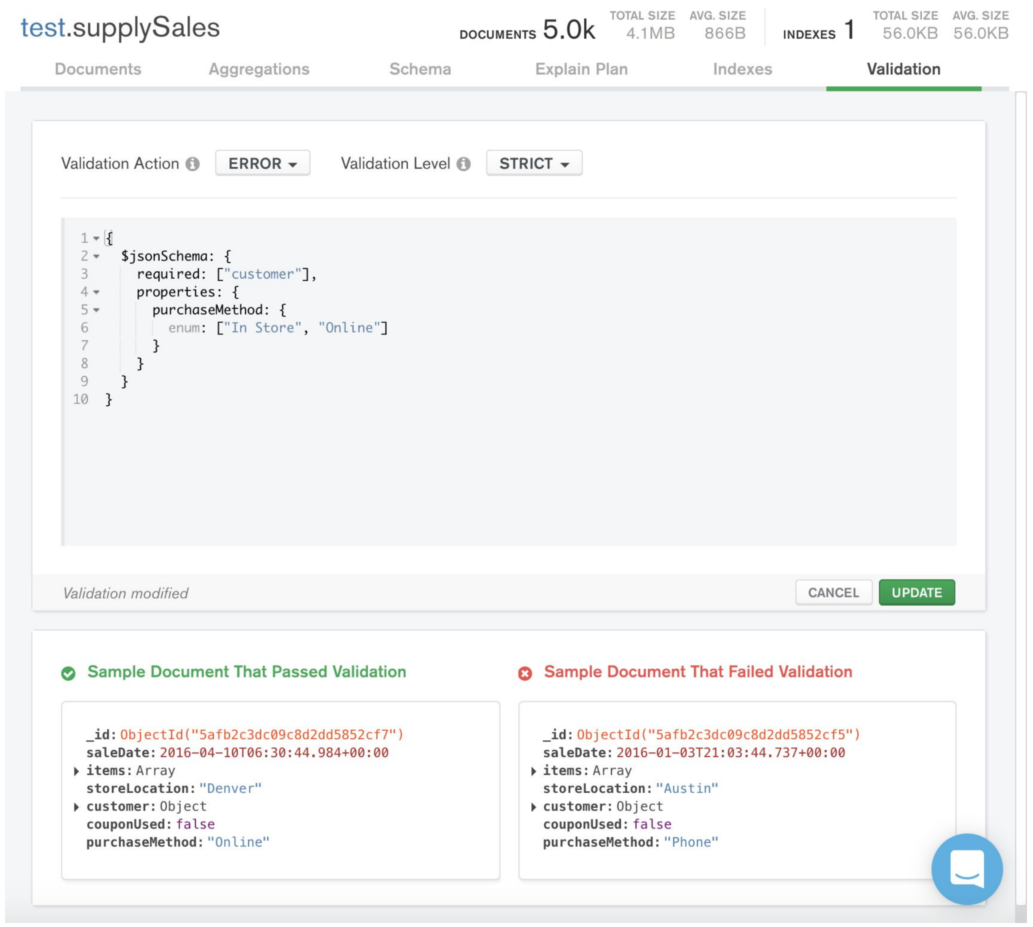
```
{
    first_name: "Sydney",
    surname: "Schaefer",
    city: "Lancaster",
    school: "Daisy's Daycare"
}
```

Flexible

Schema?

mongoDB.

Flexible Schema Validation



Validation Action ERROR
Validation Level STRICT

```
{
  $jsonSchema: {
    required: ["customer"],
    properties: {
      purchaseMethod: {
        enum: ["In Store", "Online"]
      }
    }
  }
}
```

Validation modified

Sample Document That Passed Validation

```
_id: ObjectId("5afb2c3dc09c8d2dd5852cf7")
saleDate: 2016-04-10T06:30:44.984+00:00
items: Array
storeLocation: "Denver"
customer: Object
couponUsed: false
purchaseMethod: "Online"
```

Sample Document That Failed Validation

```
_id: ObjectId("5afb2c3dc09c8d2dd5852cf5")
saleDate: 2016-01-03T21:03:44.737+00:00
items: Array
storeLocation: "Austin"
customer: Object
couponUsed: false
purchaseMethod: "Phone"
```

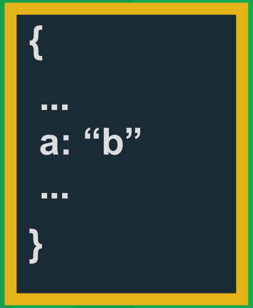
Document

```
{
 ...
 a: “b”
 ...
}
```

Row

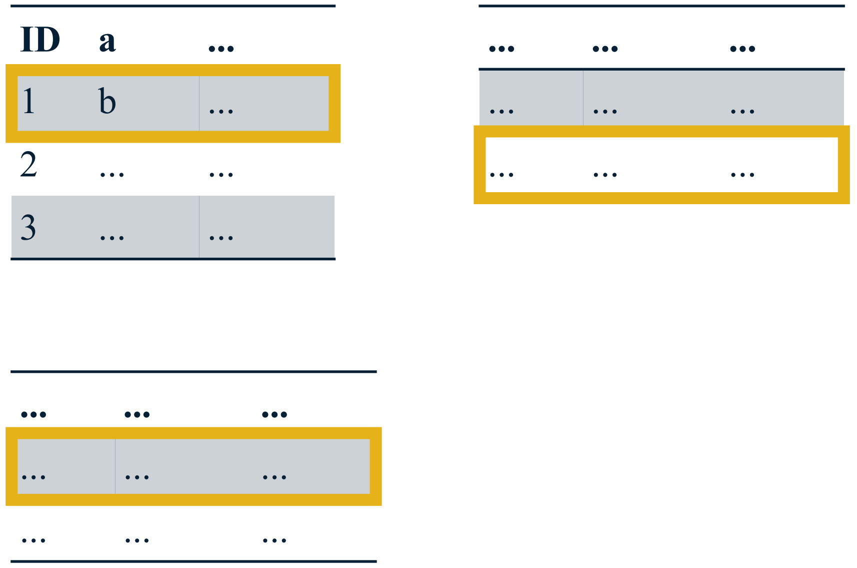
ID	a	...
1	b	...
2	...	...
3	...	...

Document



```
{
 …
 a: "b"
 …
}
```

Row(s)



ID	a	…
1	b	…
2	…	…
3	…	…

…	…	…
…	…	…
…	…	…

…	…	…
…	…	…
…	…	…

Field

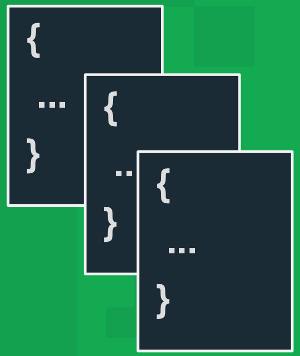
```
{
…
a: "b"
…
}
```

```
{
…
a: "c"
…
}
```

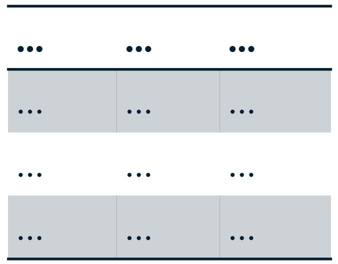
Column

ID	a	…
1	b	…
2	c	…
3	…	…

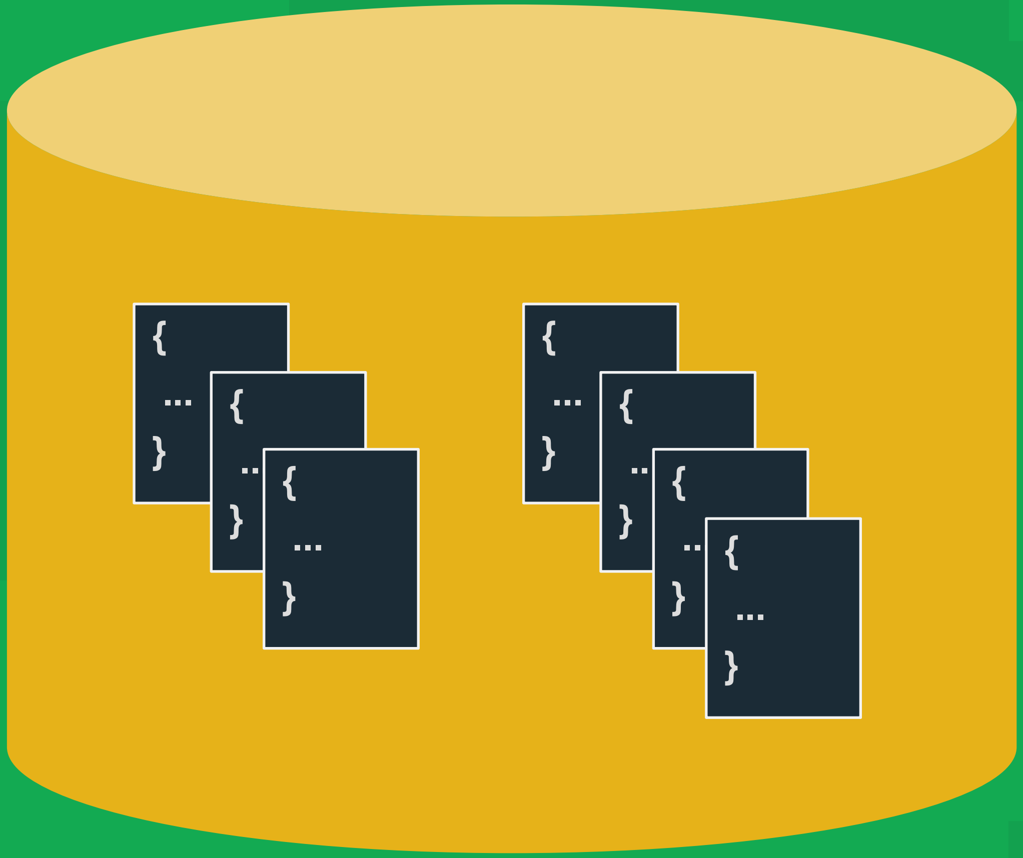
Collection



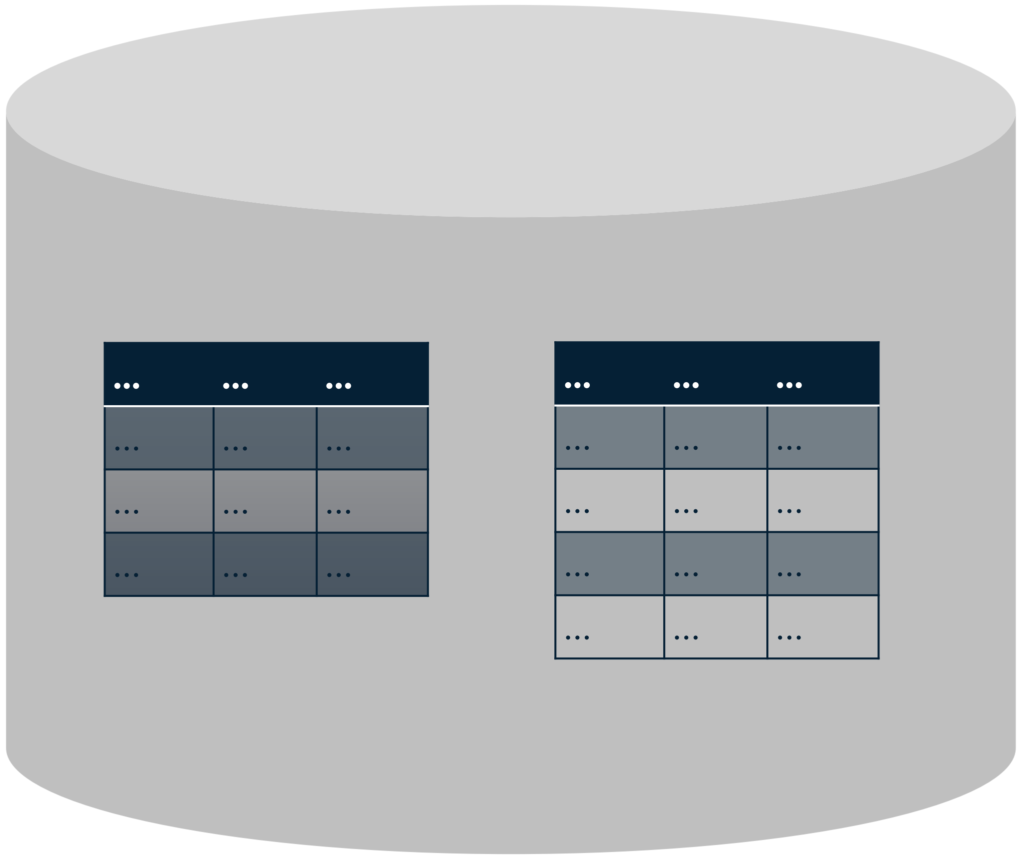
Table



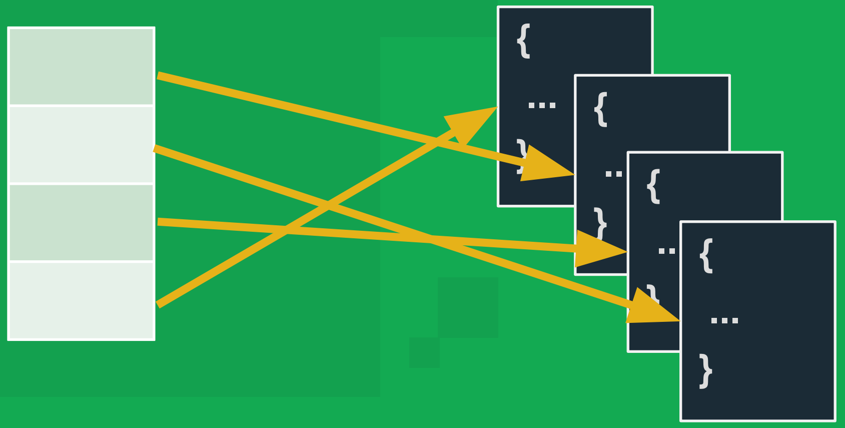
Database



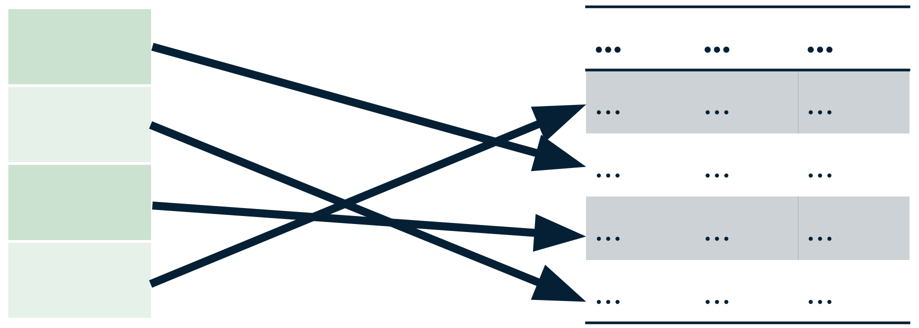
Database



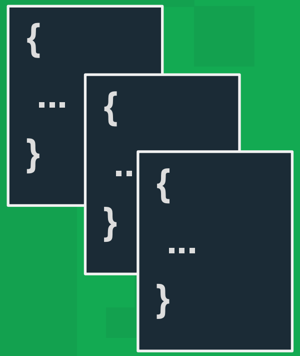
Index



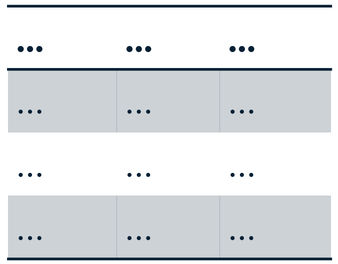
Index



View



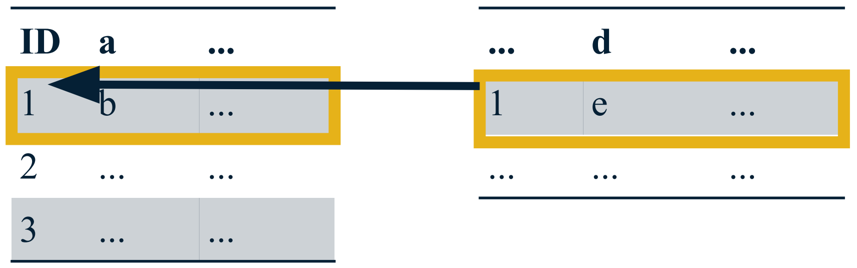
View



Embedding

```
{
 ...
 a: "b",
 ...
 c: {
     d: "e"
     ...
   },
 ...
}
```

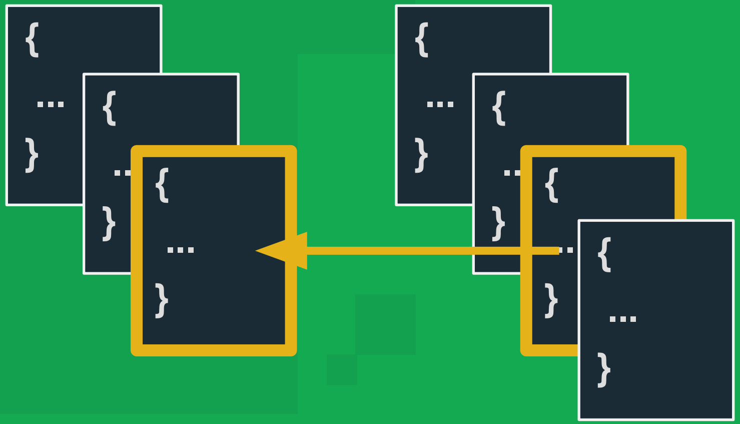
Join



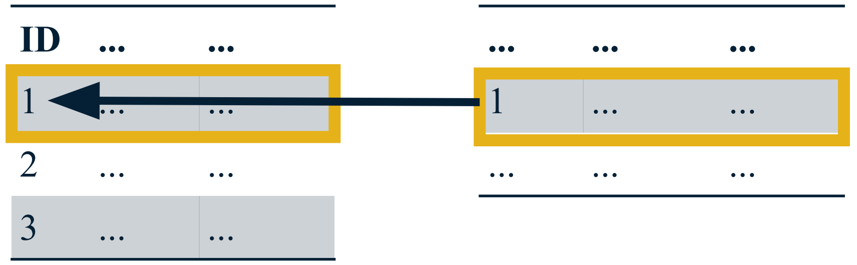
ID	a	...
1	b	...
2	...	...
3	...	...

...	d	...
1	e	...
...	...	...

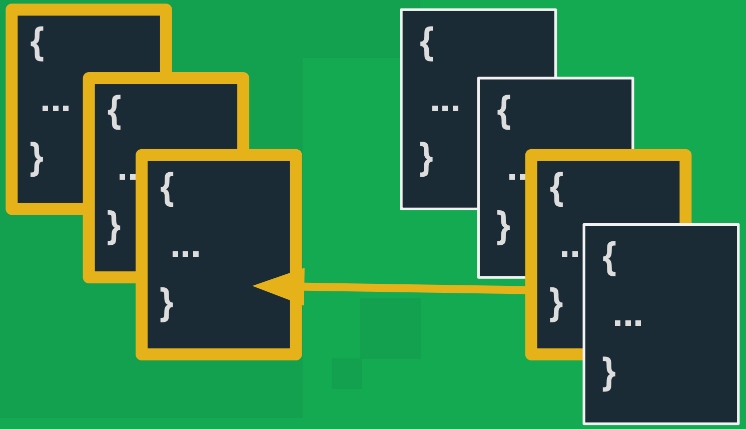
Database References



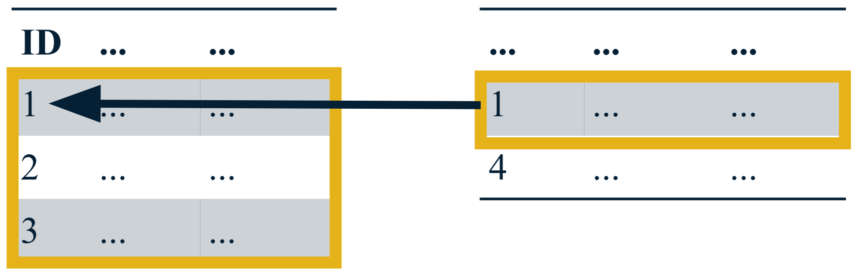
Join



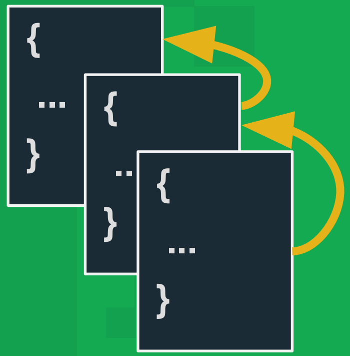
$lookup
(Aggregation Pipeline)



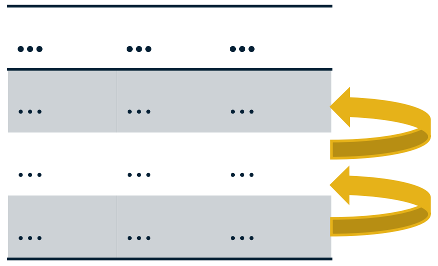
Left Outer Join



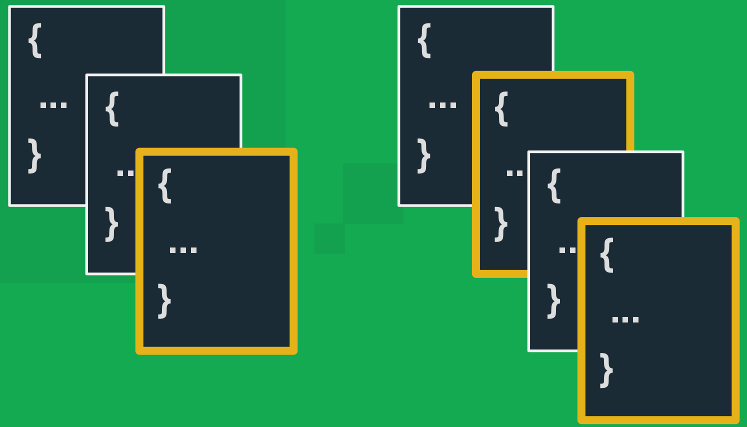
$graphLookup (Aggregation Pipeline)



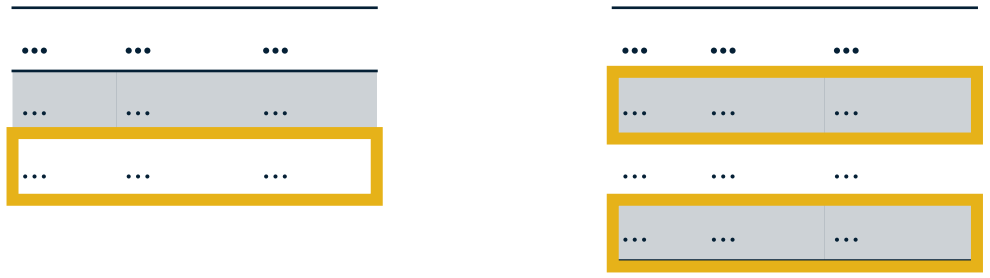
Recursive Common Table Expressions



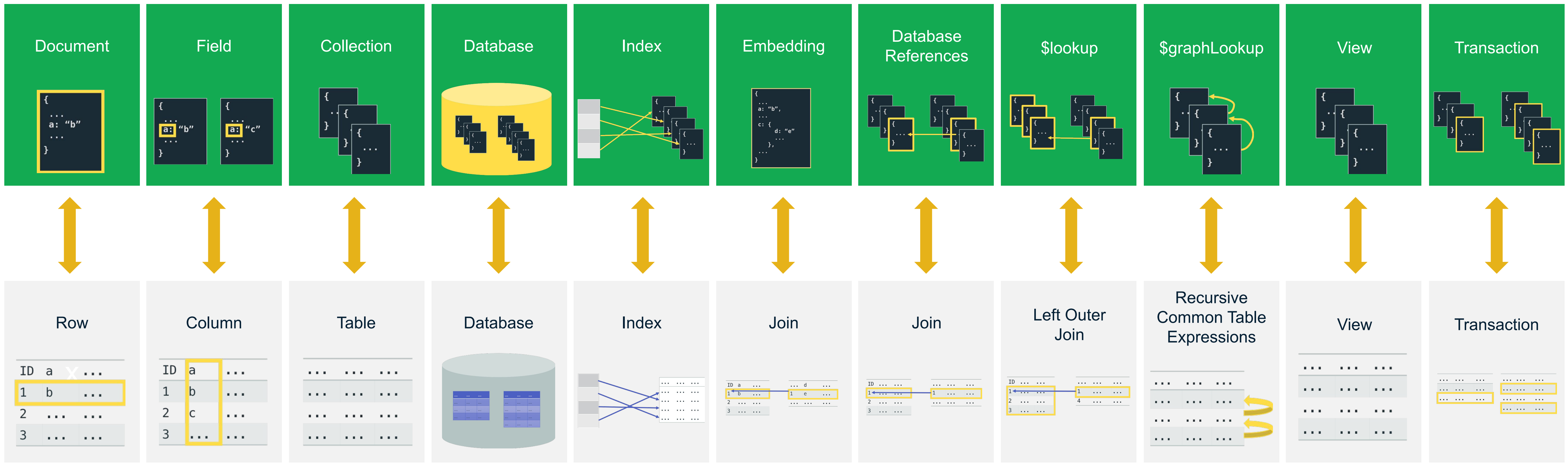
Multi-Document ACID Transaction



Multi-Record ACID Transaction

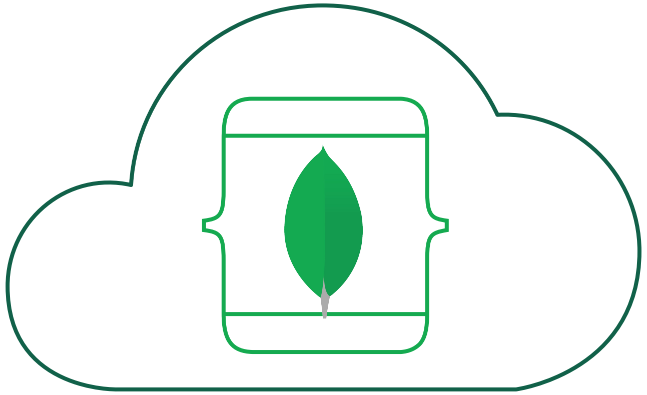


MongoDB Terminology



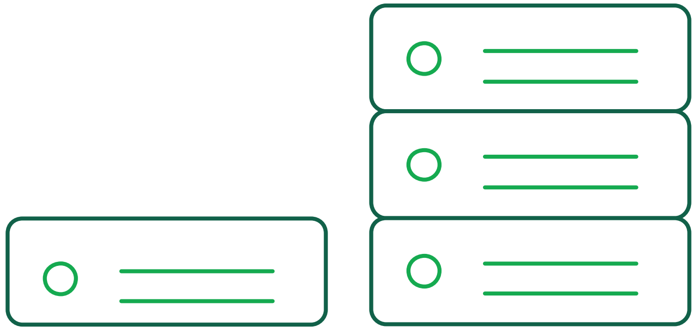
1. Some terminology and concepts
2. **How can I run MongoDB ?**
3. MongoDB Architecture
4. Tips to get more out of your MongoDB
5. Useful links to follow up
6. Q&A

MongoDB Atlas



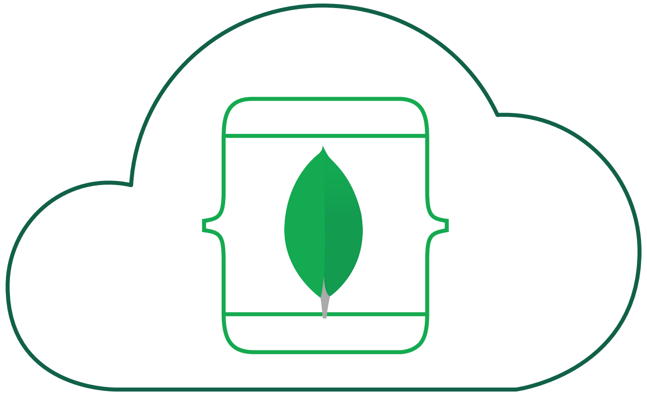
Fully Managed Cloud Service

MongoDB Server



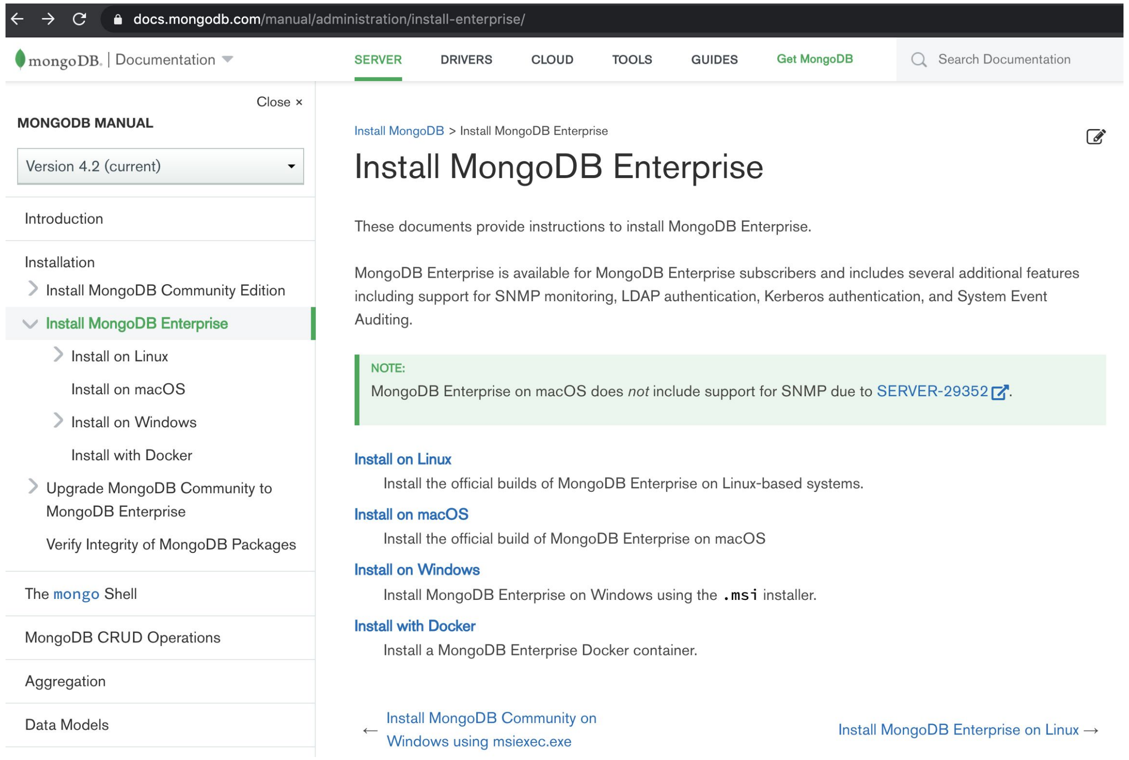
Self Managed

MongoDB Atlas
Demo



Fully Managed Cloud Service

MongoDB Server



docs.mongodb.com/manual/administration/install-enterprise/

mongoDB. | Documentation

SERVER DRIVERS CLOUD TOOLS GUIDES Get MongoDB Search Documentation

Close ×

MONGODB MANUAL

Version 4.2 (current)

- Introduction
- Installation
  - Install MongoDB Community Edition
  - Install MongoDB Enterprise
    - Install on Linux
    - Install on macOS
    - Install on Windows
    - Install with Docker
  - Upgrade MongoDB Community to MongoDB Enterprise
  - Verify Integrity of MongoDB Packages
- The mongo Shell
- MongoDB CRUD Operations
- Aggregation
- Data Models

Install MongoDB > Install MongoDB Enterprise

Install MongoDB Enterprise

These documents provide instructions to install MongoDB Enterprise.

MongoDB Enterprise is available for MongoDB Enterprise subscribers and includes several additional features including support for SNMP monitoring, LDAP authentication, Kerberos authentication, and System Event Auditing.

NOTE:
MongoDB Enterprise on macOS does *not* include support for SNMP due to SERVER-29352.

Install on Linux
Install the official builds of MongoDB Enterprise on Linux-based systems.

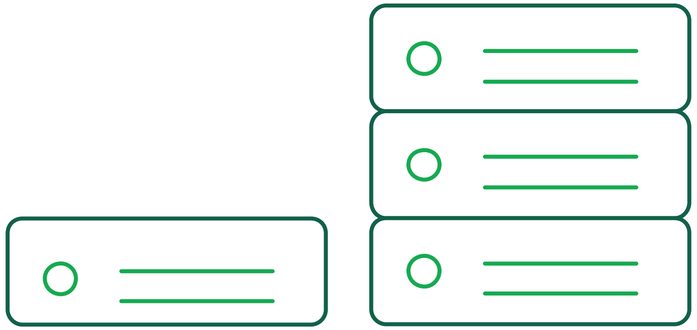
Install on macOS
Install the official build of MongoDB Enterprise on macOS

Install on Windows
Install MongoDB Enterprise on Windows using the `.msi` installer.

Install with Docker
Install a MongoDB Enterprise Docker container.

← Install MongoDB Community on Windows using msiexec.exe

Install MongoDB Enterprise on Linux →



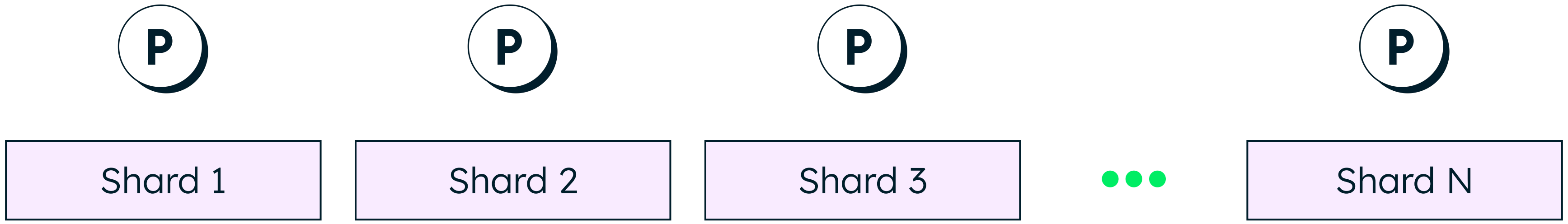
Self Managed

https://docs.mongodb.com/manual/administration/install-enterprise/

1. Some terminology and concepts
2. How can I run MongoDB ?
3. **MongoDB Architecture**
4. Tips to get more out of your MongoDB
5. Useful links to follow up
6. Q&A

Whiteboard

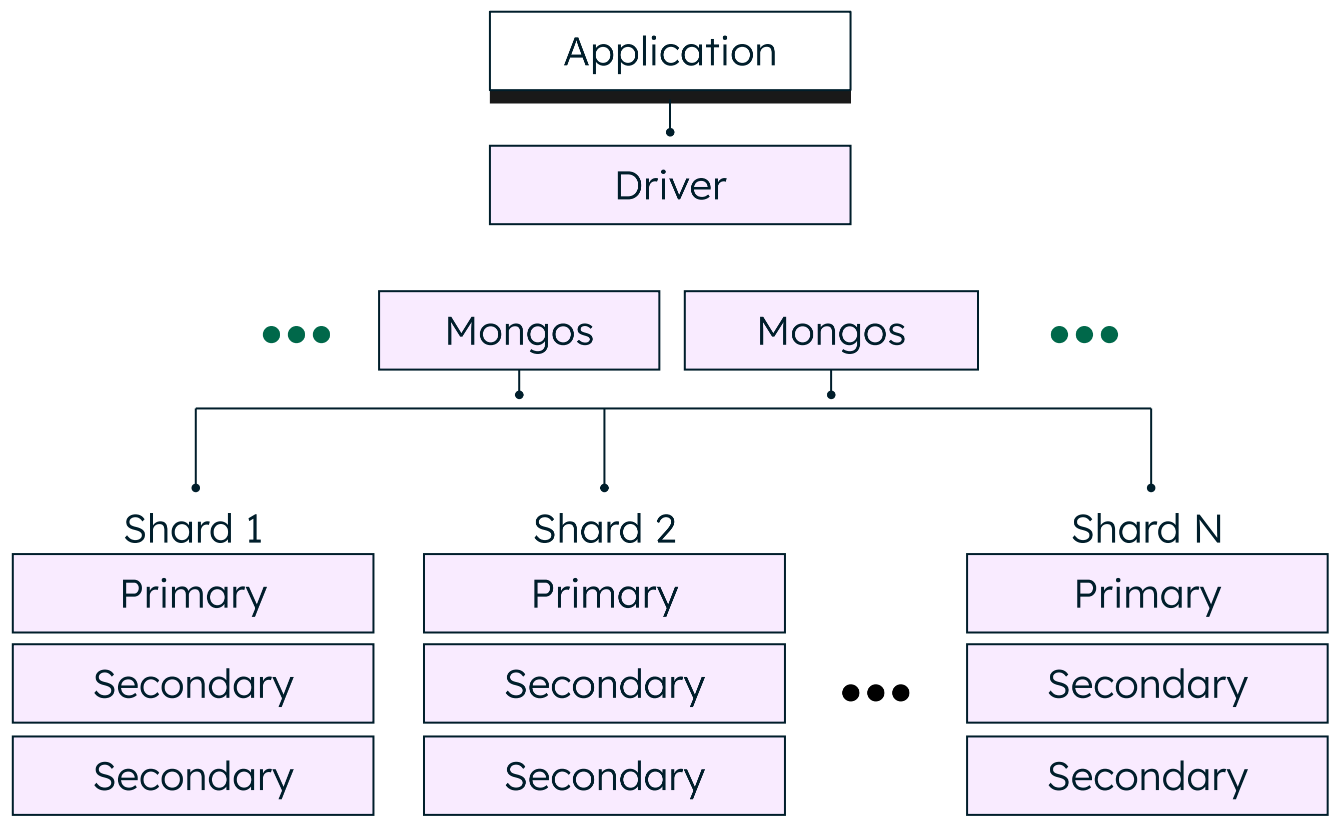
Cost-effective at any scale



Native-Sharding for horizontal scale-out

- Automatically scale beyond the constraints of a single node
- Application transparent
- Scale, refine, rebalance, and reshard data at any time
- Unlike NoSQL systems that randomly spray data across a cluster, MongoDB exposes multiple data distribution policies (hashed, ranged, zoned) to optimize for query patterns and locality

Sharding architecture



Horizontal scalability

Sharding

High availability

Replica sets

C reate

R ead

U pdate

D elete

C onnect

1. Some terminology and concepts
2. How can I run MongoDB ?
3. MongoDB Architecture
4. **Tips to get more out of your MongoDB**
5. Useful links to follow up
6. Q&A

mongoDB.

Use Indexes for Read Speed

- Very important for **reads**
- However, be aware of the **overhead**.
- New in MongoDB 6.x, **Clustered Indexes**

Use Indexes for Read Speed

***Indexes** support the **efficient** execution of queries in MongoDB.*

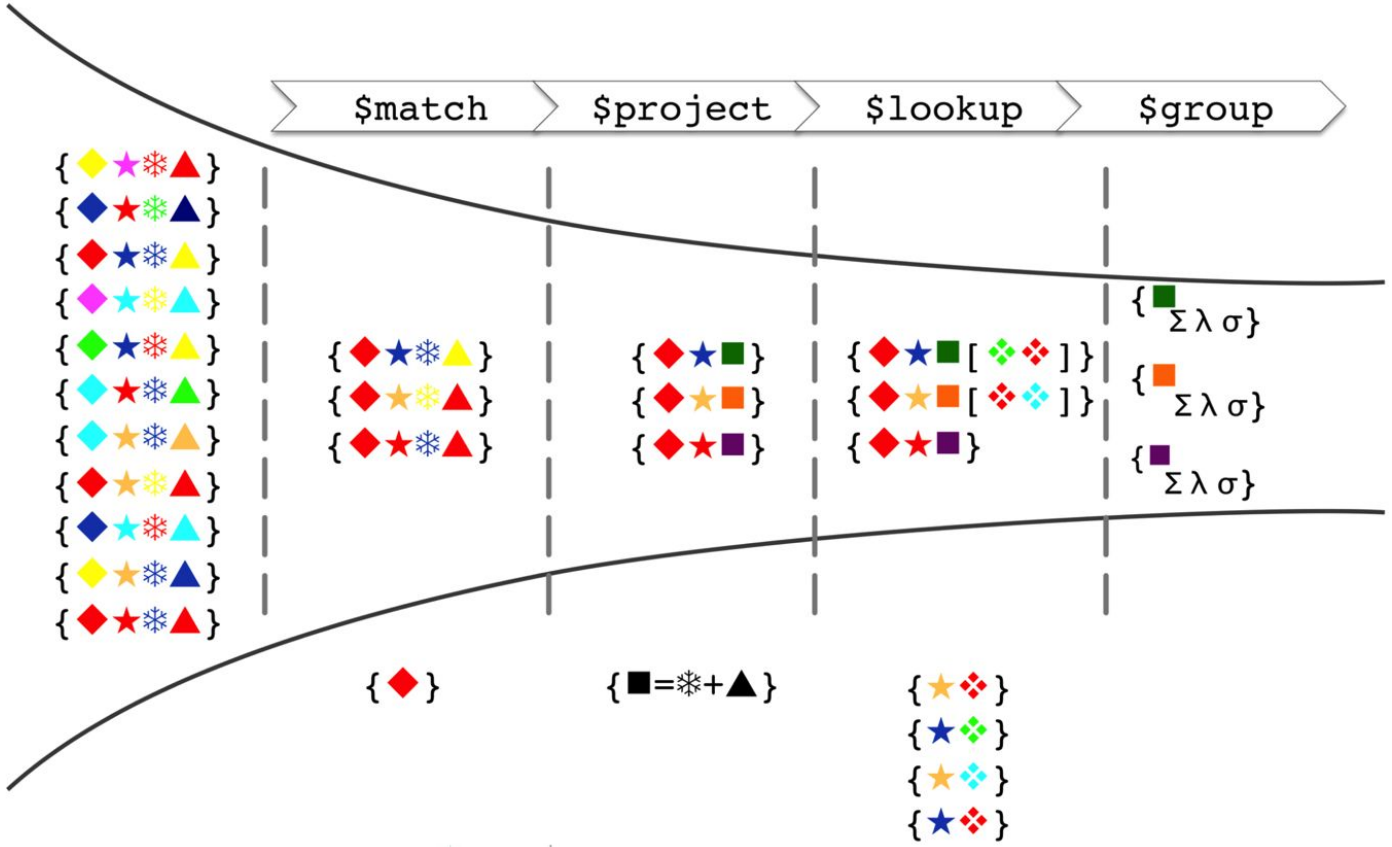
Index Types

Single Field `{ karma: 1}`
Compound Field `{ karma: 1, user_id: -1 }`
Multikey `{ “address.postal_code”: 1 }`
Geospatial
Text
Hashed
Wildcard

Reduce Aggravations with the Aggregation Framework

- Use whenever possible
- Operations are done server-side
- Order of stages matters

Aggregation



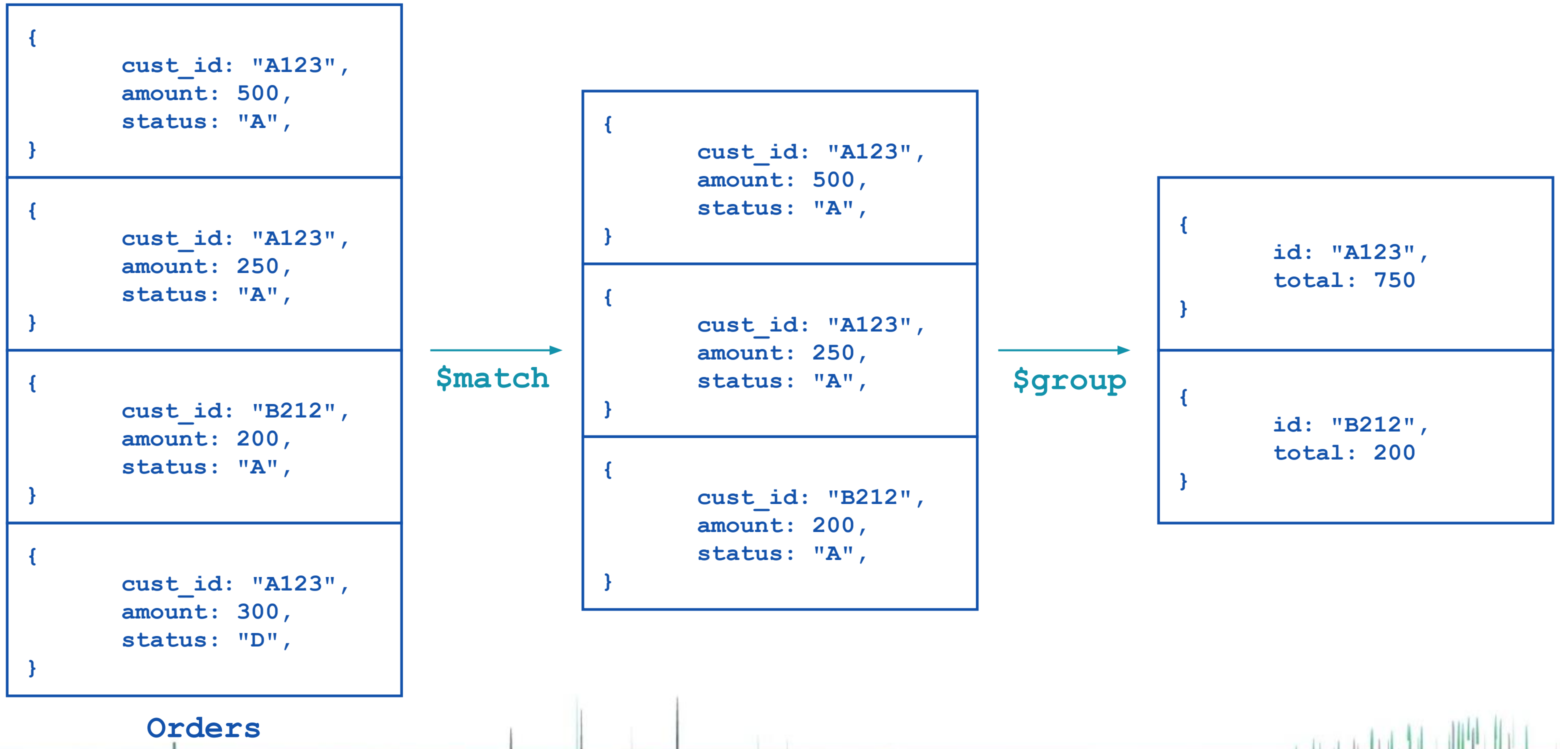
Pipeline

*nix command line pipe

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

Aggregation

```
Collection
    ↓
db.orders.aggregate( [
          $match stage →              {$match: { status: "A" } },
          $group stage                { $group: { _id: "$cust_id",total: { $sum:"$amount" } } }
                     ] )
```



```
{
      cust_id: "A123",
      amount: 500,
      status: "A",
}
{
      cust_id: "A123",
      amount: 250,
      status: "A",
}
{
      cust_id: "B212",
      amount: 200,
      status: "A",
}
{
      cust_id: "A123",
      amount: 300,
      status: "D",
}
```

```
{
      cust_id: "A123",
      amount: 500,
      status: "A",
}
{
      cust_id: "A123",
      amount: 250,
      status: "A",
}
{
      cust_id: "B212",
      amount: 200,
      status: "A",
}
```

```
{
      id: "A123",
      total: 750
}
{
      id: "B212",
      total: 200
}
```

Model Data Using Schema Design Patterns

- Different way of modeling from the legacy database paradigm.
- Schema Design is important.

Why Do We Create Models?

Ensure:

- Good performance
- Scalability

despite constraints

Hardware

- RAM faster than Disk
- Disk cheaper than RAM
- Network latency
- Reduce costs $$$

Database Server

- Maximum size for a document

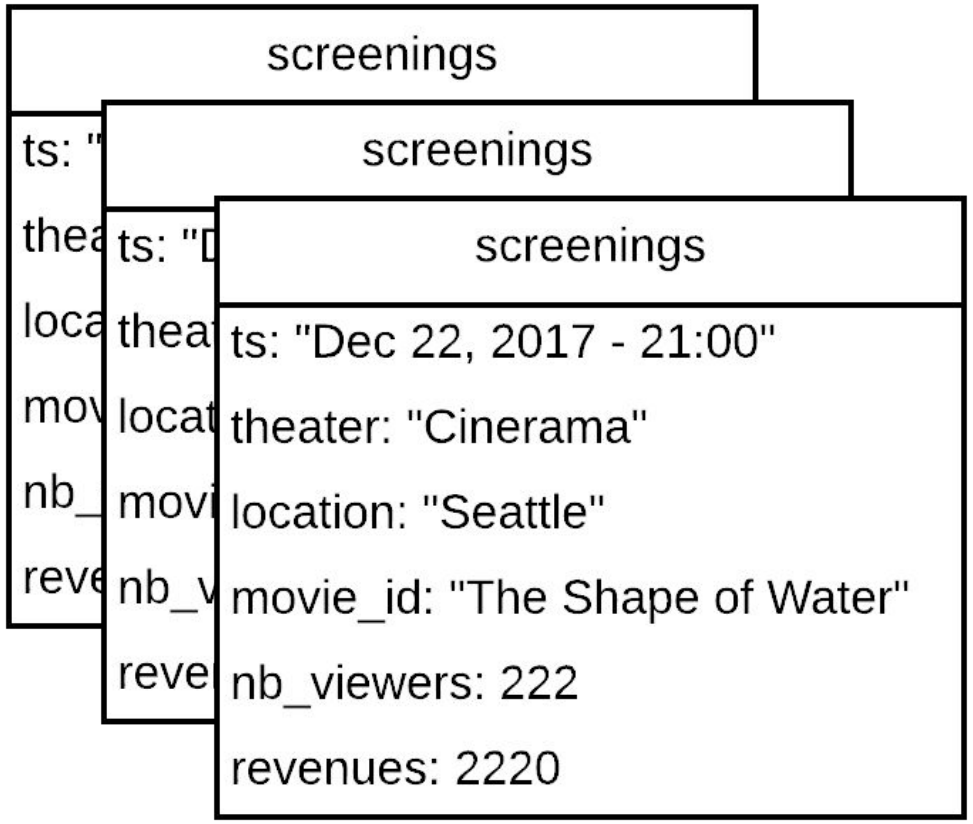
Data set

- Size of data

Patterns by Category

- Representation
  - Attribute
  - Schema Versioning
  - Document Versioning
  - Tree
  - Polymorphism
  - Pre-Allocation
- Frequency of Access
  - Subset
  - Approximation
  - Extended Reference
- Grouping
  - Computed
  - Bucket
  - Outlier

Processing overhead ... repeated calculations

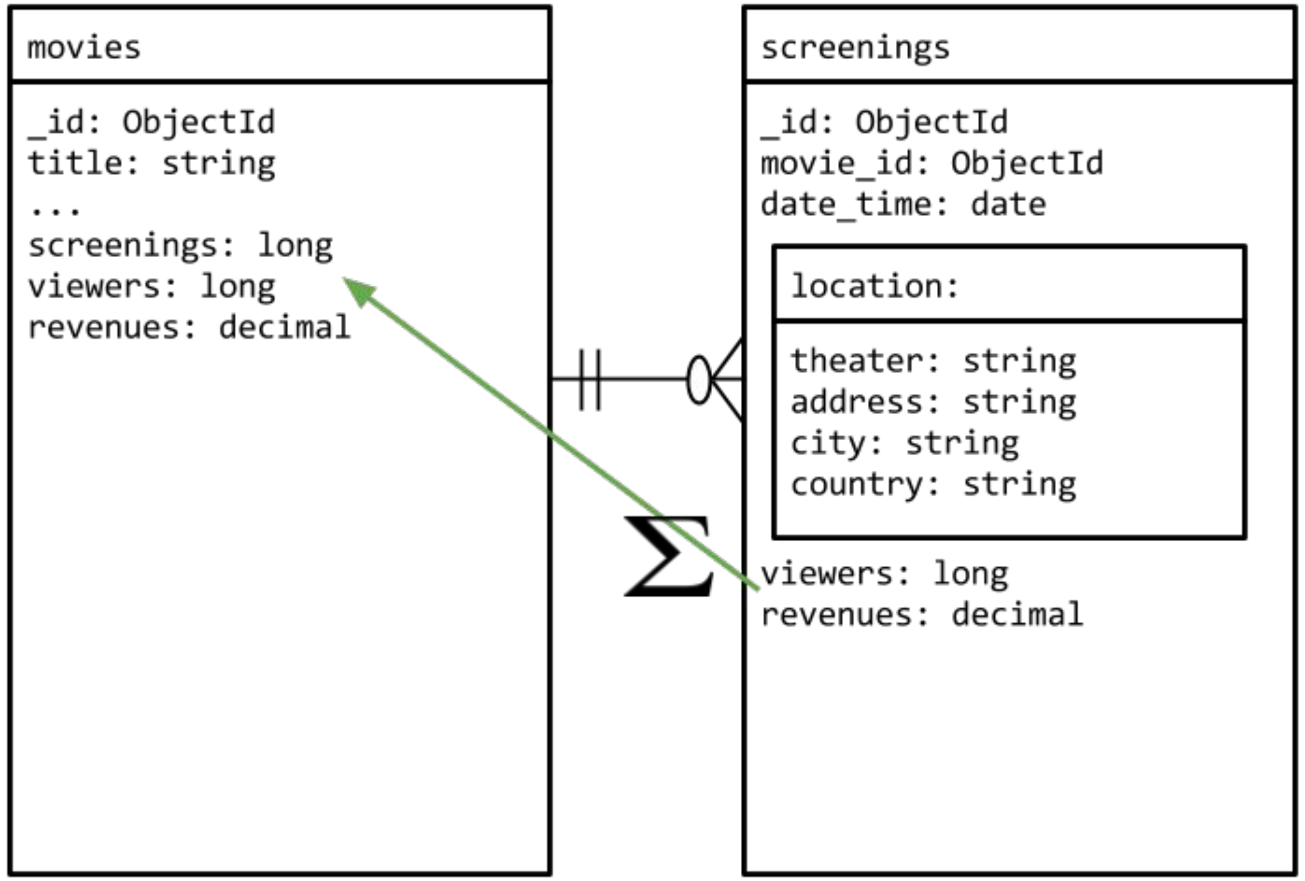


```
{
   title: "The Shape of Water",
   ...
   viewings: 5,000
   viewers: 385,000
   revenues: 5,074,800
}
```

Computed

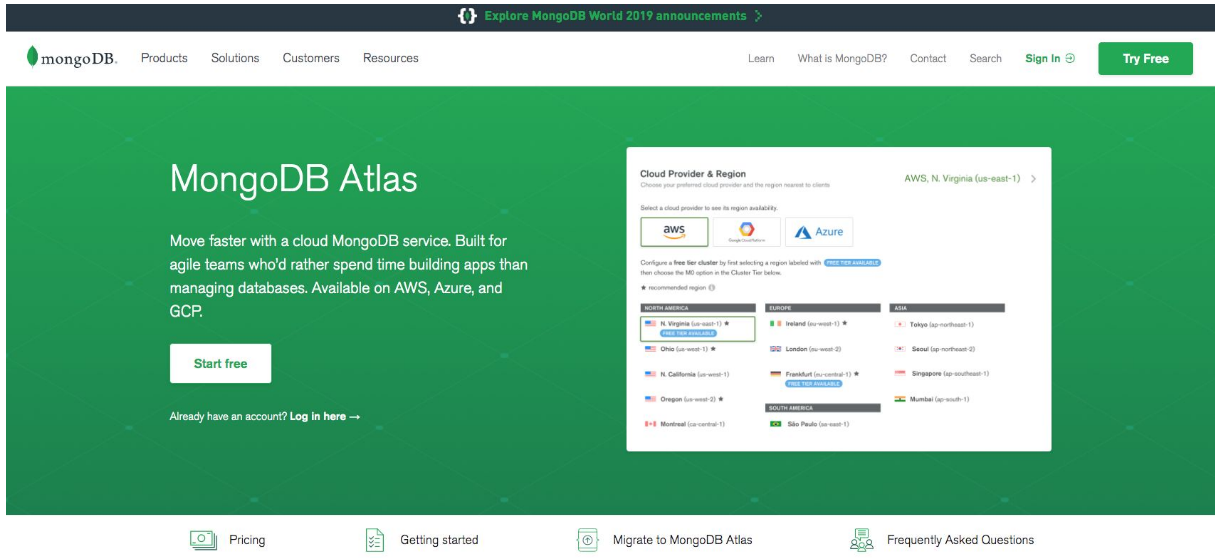
For example:

- Apply a sum, count, ...
- *rollup* data by minute, hour, day
- As long as you don't mess with your source, you can recreate the *rollups*



mongoDB.

Sign up for MongoDB Atlas



{} Explore MongoDB World 2019 announcements >

mongoDB. Products Solutions Customers Resources Learn What is MongoDB? Contact Search Sign In Try Free

MongoDB Atlas

Move faster with a cloud MongoDB service. Built for agile teams who'd rather spend time building apps than managing databases. Available on AWS, Azure, and GCP.

Start free

Already have an account? **Log in here →**

Cloud Provider & Region
Choose your preferred cloud provider and the region nearest to clients

AWS, N. Virginia (us-east-1) >

Select a cloud provider to see its region availability.

aws | Google Cloud Platform | Azure

Configure a **free tier cluster** by first selecting a region labeled with FREE TIER AVAILABLE then choose the M0 option in the Cluster Tier below.

★ recommended region

NORTH AMERICA	EUROPE	ASIA
N. Virginia (us-east-1) ★ FREE TIER AVAILABLE	**Ireland** (eu-west-1) ★	**Tokyo** (ap-northeast-1)
Ohio (us-west-1) ★	**London** (eu-west-2)	**Seoul** (ap-northeast-2)
N. California (us-west-1)	**Frankfurt** (eu-central-1) ★ FREE TIER AVAILABLE	**Singapore** (ap-southeast-1)
Oregon (us-west-2) ★	SOUTH AMERICA	**Mumbai** (ap-south-1)
Montreal (ca-central-1)	**São Paulo** (sa-east-1)	

Pricing Getting started Migrate to MongoDB Atlas Frequently Asked Questions

mongodb.com/cloud/atlas

Additional resources

- The MongoDB Docs: https://docs.mongodb.com/
- JSON Schema Validation – Locking down your model the smart way: https://www.mongodb.com/blog/post/json-schema-validation--locking-down-your-model-the-smart-way
- JSON Schema Validation - Checking Your Arrays: https://www.mongodb.com/blog/post/json-schema-validation--checking-your-arrays
- Quick Start blog series in a variety of programming languages: https://www.mongodb.com/blog/channel/quickstart
- Understanding MongoDB indexes: https://docs.mongodb.com/manual/indexes/
- M121: The MongoDB Aggregation Framework: https://university.mongodb.com/courses/M121/about
-

Additional resources contd.

- Advanced Schema Design Patterns (webinar): https://www.mongodb.com/presentations/advanced-schema-design-patterns
- Building with Patterns: A Summary (blog series): https://www.mongodb.com/blog/post/building-with-patterns-a-summary
- M320: Data Modeling (MongoDB University Course – brand new!): https://university.mongodb.com/courses/M320/about

1. Some terminology and concepts
2. How can I run MongoDB ?
3. MongoDB Architecture
4. Tips to get more out of your MongoDB
5. Useful links to follow up
6. **Q&A**

Back to Basics

Questions & Answers

Nooruddin Abbas Ali
Principal Solutions Architect
nooruddin.abbasali@mongodb.com