

LINKS

<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-windows/> - MongoDB Community Edition

<https://www.mongodb.com/try/download/shell> - Shell

Introduction to MongoDB

* NoSQL Database or non-relational database
* Unstructured
* MongoDB != RDBMS
* Supports Horizontal Scalability
* It doesn’t use tables for storing data
* Used to store big data and real-time web application

Advantages of NoSQL

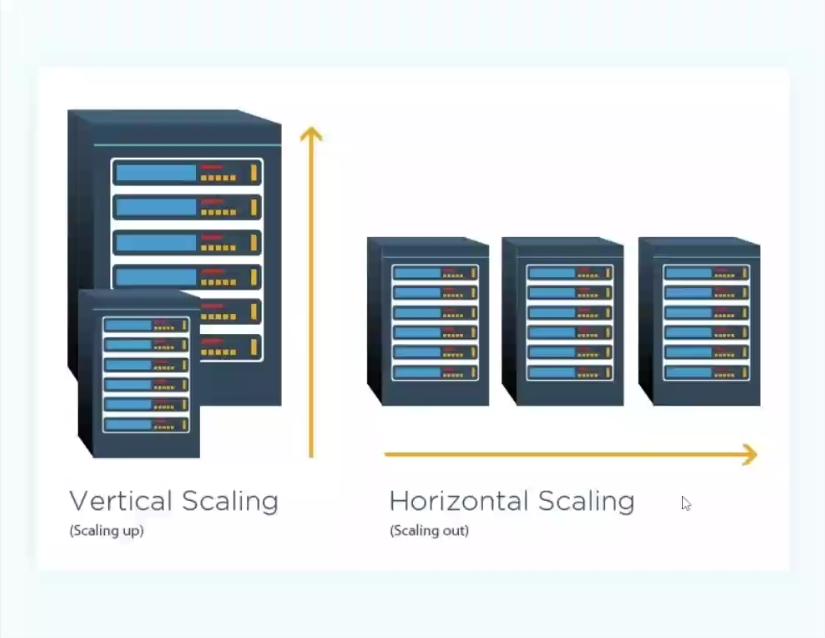
* It supports query language
* It provides fast performance
* It provides horizontal scalability

Vertical Scalability

* Vertical Scaling (Scale Up)  
  Upgrading the **existing machine’s capacity** — more CPU, RAM, or SSD.
* **Example:**  
  Upgrading MongoDB server from 8 GB RAM to 32 GB RAM.
* **Pros:**
  + Simpler to implement
  + No change in application architecture
* **Cons:**
  + Limited by hardware capacity
  + Downtime may be needed for upgrades
* **Applications**
  + Simpler applications or early stages
  + Databases with single-instance support

Horizontal Scalability

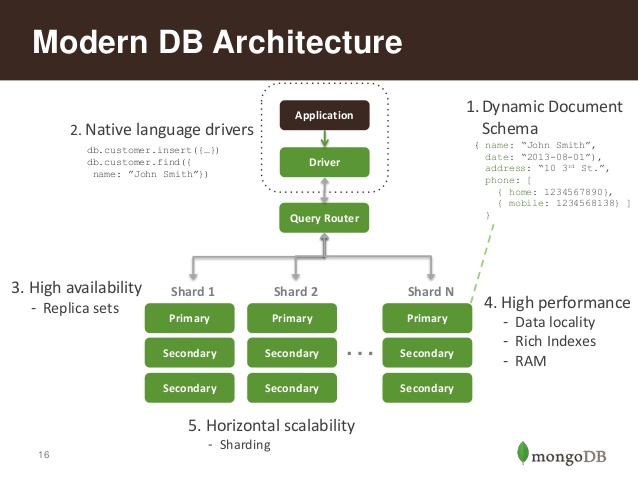
* **Horizontal Scaling (Scale Out)**  
  Adding **more machines/nodes** to your system.()
* **Example:**  
  Instead of running MongoDB on one server, run it on **3 servers**. The data and load are **distributed**.
* **Pros:**
* Fault-tolerant (if one server fails, others still run)
* Better performance for large-scale systems
* Easier to scale indefinitely
* **Cons:**
* More complex (requires load balancing, syncing)
* Higher infrastructure management
* **Applications:**
* Web applications
* Distributed databases (e.g., MongoDB sharding)
* Cloud environments



MongoDB

**MongoDB,**

* Open-source NoSQL document-oriented database
* Stores data in JSON-like BSON documents



**Key Features**

* Schema-less (dynamic schema)
* Horizontal scaling
* Integrated aggregation
* Powerful querying
* High availability via replication

NoSQL

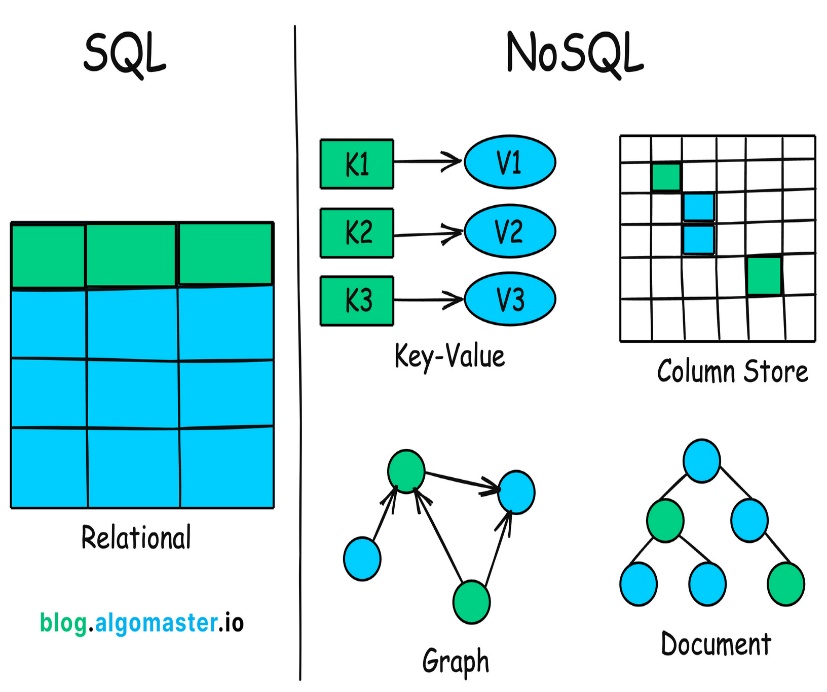
* Not Only SQL
* Designed to handle unstructured/semi-structured data
* Stores data in key-value, document, column, or graph formats

Limitations of Traditional (SQL) Databases

* Fixed schema
* Complex joins
* Not great with large-scale distributed systems

Why NoSQL?

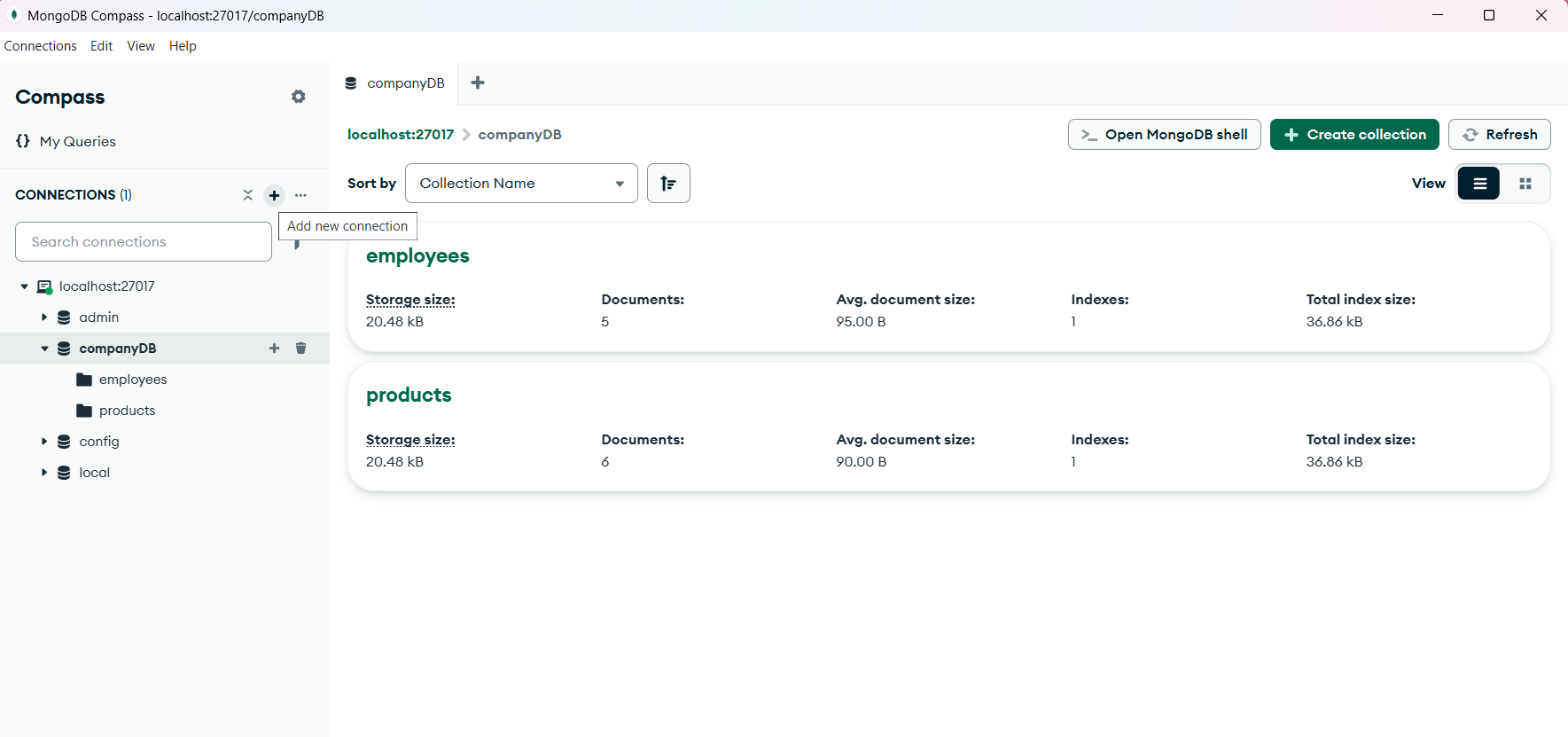
* Flexible schema
* High scalability (horizontal)
* Better performance for Big Data & real-time applications





MongoDB - Exercises

1. In MongoDB Compass, click on Add New Connection and type your Database Name and Collection Name.



1. Next open the Open MongoDB Shell on the top right corner and use the database, insert the values and start coding.

--Use the created DB

db["employees"].find() #press enter

use companyDB

switched to db companyDB

--Insert Values into the DB

db.employees.insertMany([

{name: "Neha Reddy",department: "Marketing",salary: 45000,age: 2 },

{name: "Faizan Ali",department: "Engineering",salary: 58000,age: 32 },

{name: "Divya Mehta",department: "HR",salary: 40000,age: 29 },

{name: "Ravi Verma",department: "Sales",salary: 35000, age: 26}

]);

--To fetch all the existing records  
db.employees.find()

--To fetch 1st Record alone

db.employees.findOne()

--To fetch name and salary alone for all the Employees

db.employees.find({},{name:1,salary:1})

--To get details of a particular employee by name

db.employees.findOne({name: "Neha Reddy"})

--To get employee details whose salary is greater than 50,000

db.employees.find({salary:{$gt:50000}})

--Display the employees whose age is between 28 and 32

db.employees.find({age: {$gte:28,$lte:32}})

--To display the employees whose department is either in sales or in HR

db.employees.find({department: {$in: ["HR", "Sales"]}})

--Employees other than Marketing department

db.employees.find({department: {$ne: "Marketing"}})

--Employee names staring with A

db.employees.find({name: {$regex: "^A"}})

--Sort employees by salary in descending order

db.employees.find().sort({salary:-1})

--Sort employees by salary in descending order and show the top 3(Highest Salary)

db.employees.find().sort({salary:-1}).limit(3)

--Update the salary of Ravi Verma to 37,000

db.employees.updateOne({name:"Ravi Verma"}, {$set: {salary:37000}})

--Will only update the top one in case of more than one Ravi Verma

--If there are more than one Ravi Verma and we need to update the salary of all then updateMany is used

--Incrementing all the Engineers department employees by 5000

db.employees.updateMany({department:"Engineering"},{$inc:{salary:5000}})

--Deleting by id

db.employees.deleteOne({\_id: ObjectId('687a0d6d3bd99838fb289d22')})

--Deleting many records using column

db.employees.deleteMany({department:"Sales"})