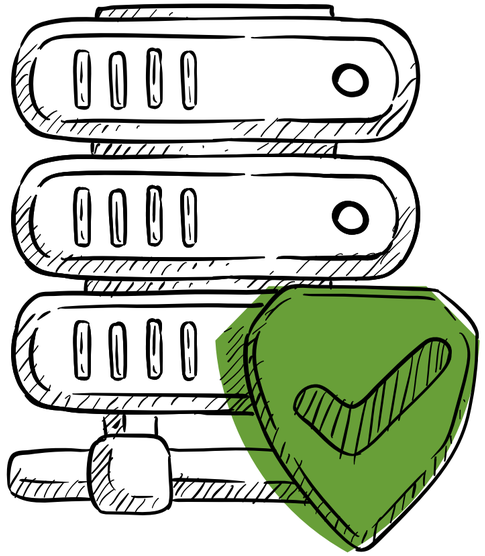
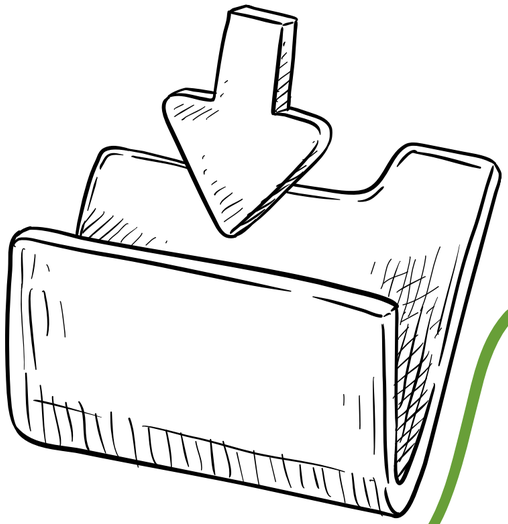
MONGO DATABASE

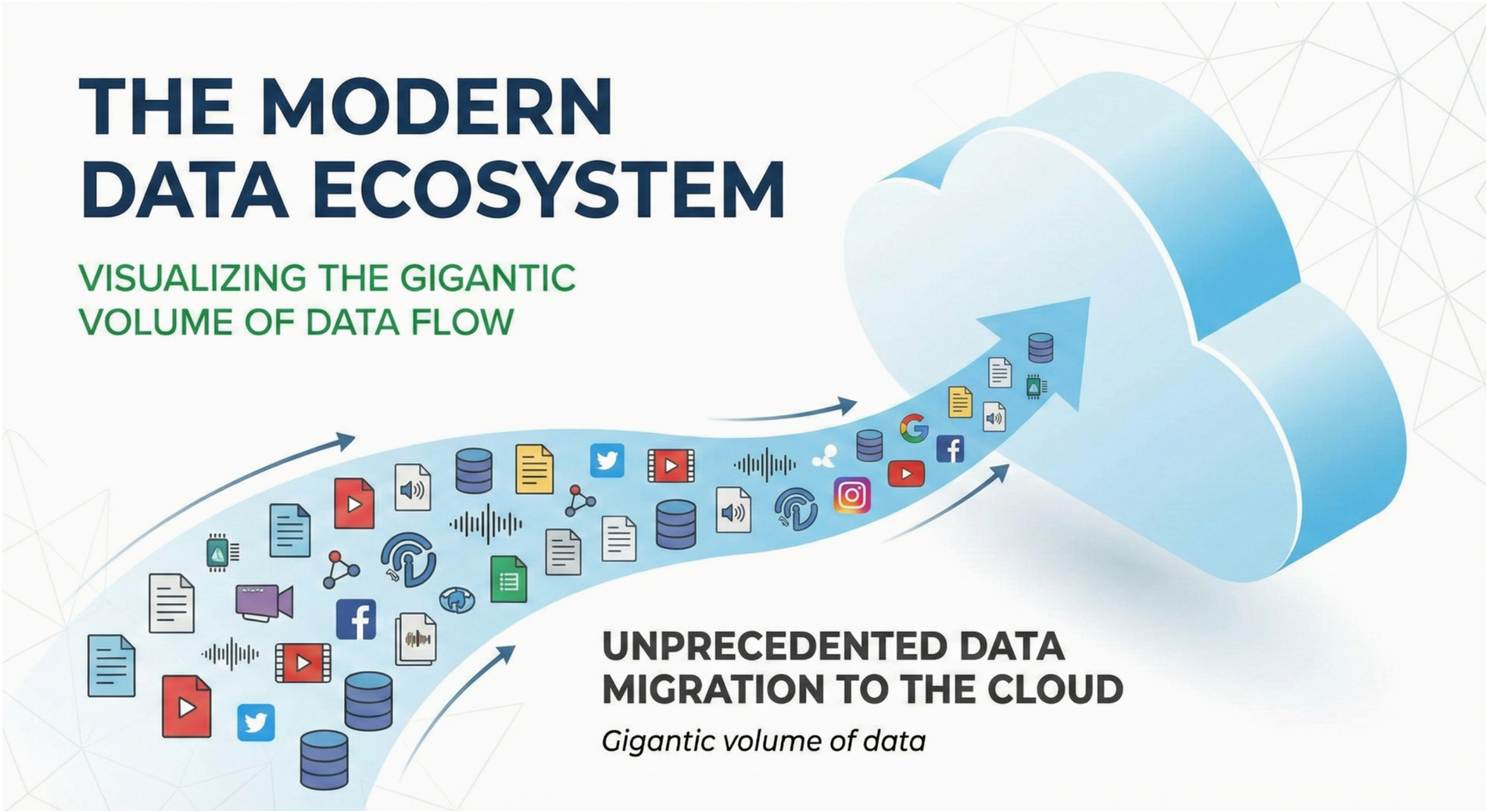
Prepared by
Noha Shehab





THE MODERN DATA ECOSYSTEM

VISUALIZING THE GIGANTIC VOLUME OF DATA FLOW

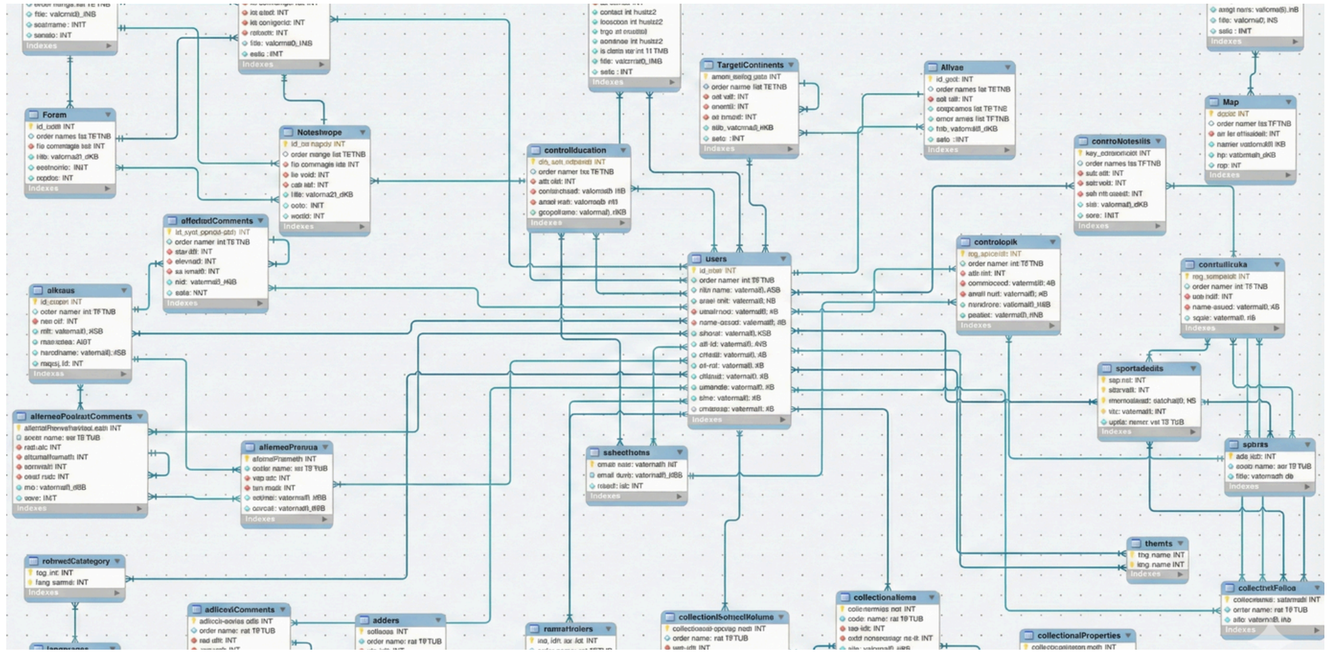


UNPRECEDENTED DATA MIGRATION TO THE CLOUD

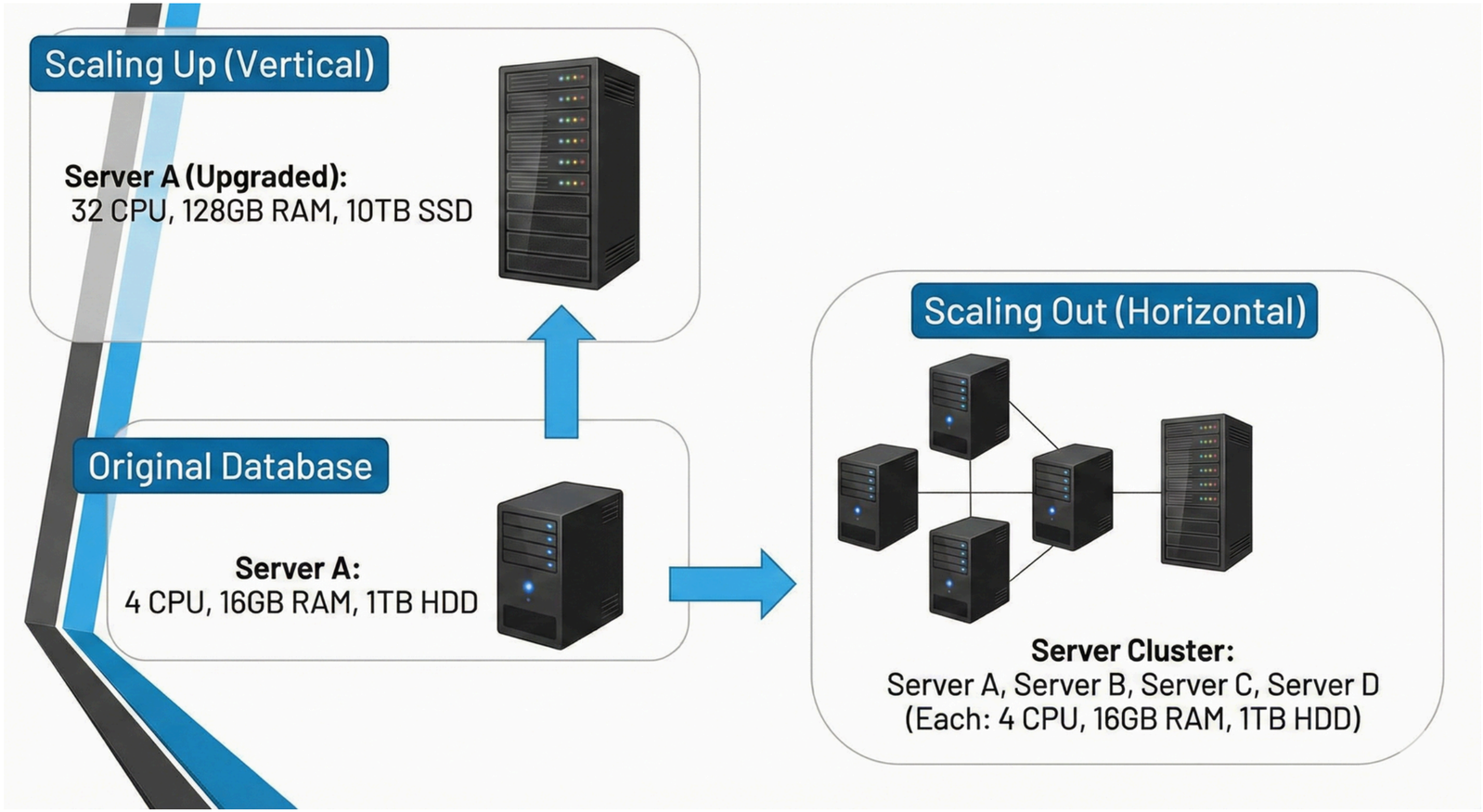
Gigantic volume of data

CONCEPT OF NOSQL DATABASES GREW WITH INTERNET GIANTS

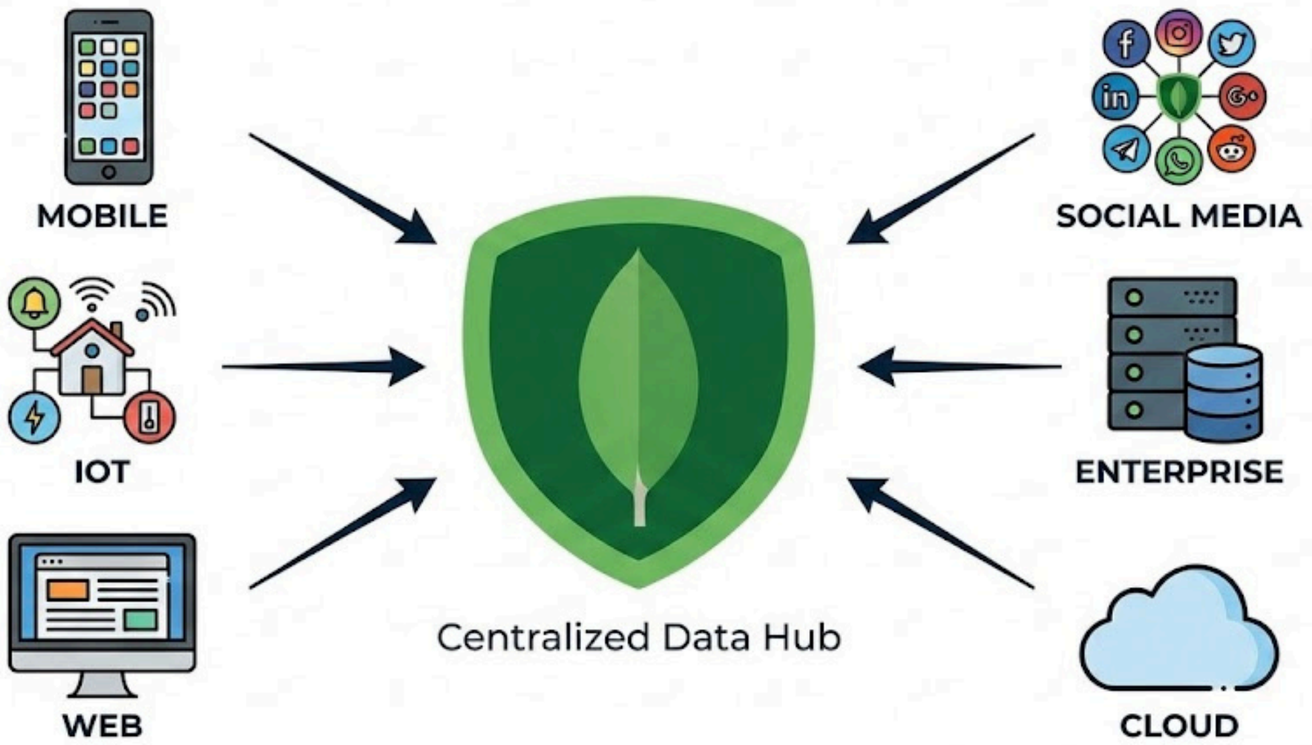
Gigantic volume of data

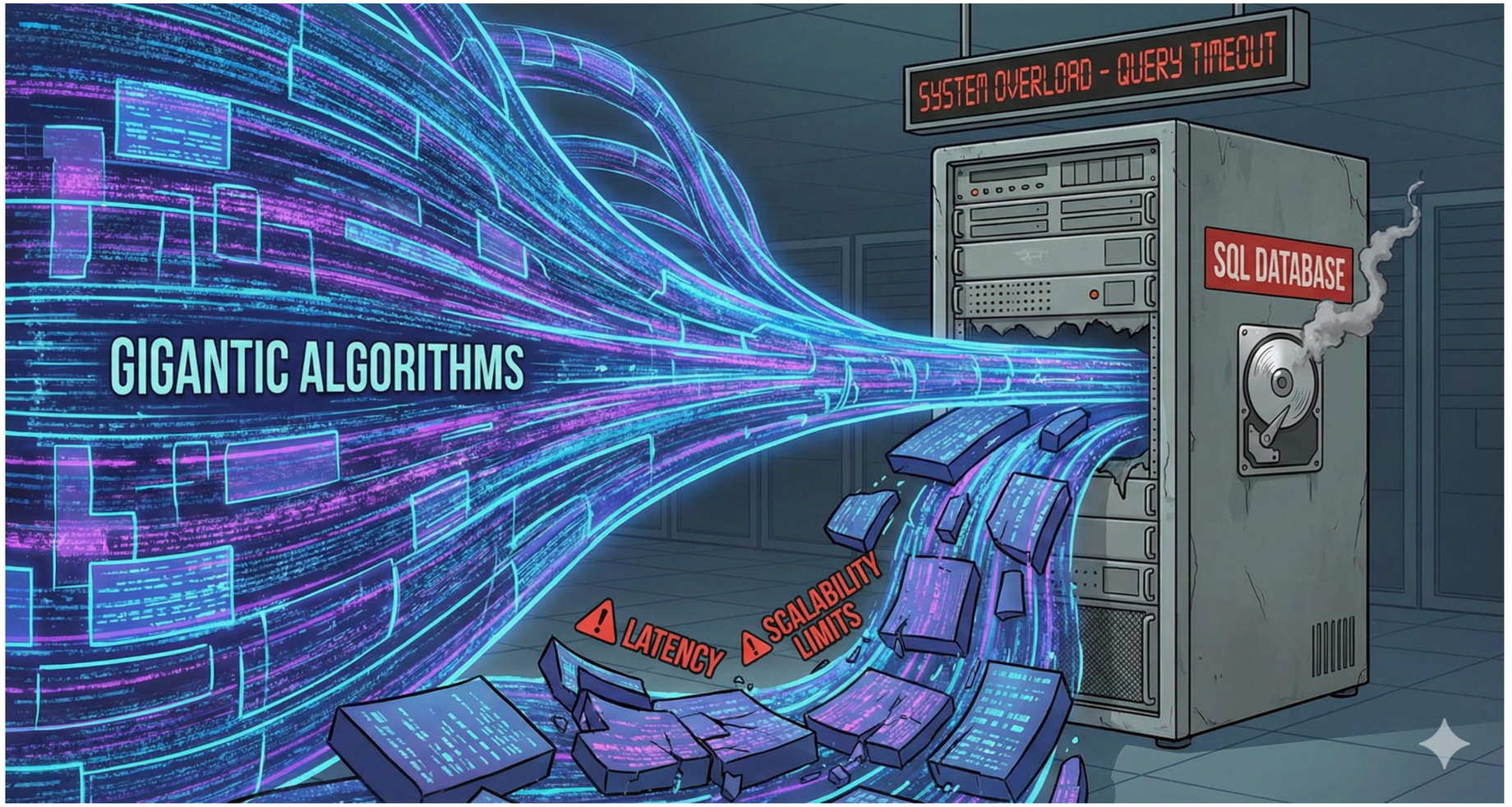


SCALING DATABASES



SOURCES OF DATA EVERYWHERE

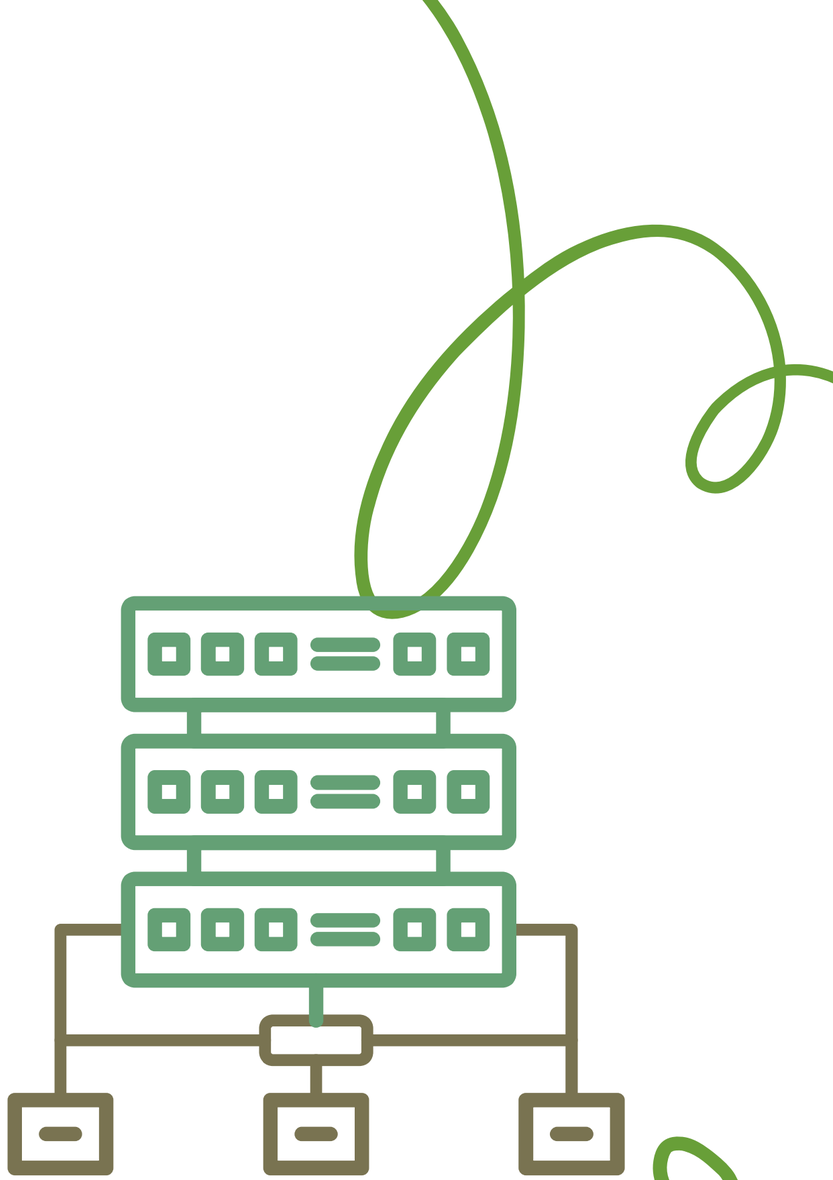



GIGANTIC ALGORITHMS
SYSTEM OVERLOAD - QUERY TIMEOUT
SQL DATABASE
LATENCY
SCALABILITY LIMITS


APACHE HBASE
OrientDB
cassandra
riak
amazon DynamoDB
Neo4j
Bigdata
No SQL
mongoDB
OrientDB
Elasticsearch
CouchDB
ArangoDB
redis

WHAT IS NOSQL

- NoSQL stands for "Not Only SQL." It is a category of database management systems designed to handle vast amounts of unstructured or changing data.
- Unlike traditional databases that require a rigid "schema" (a blueprint of what data must look like), NoSQL is flexible.



Key Characteristics of NoSQL

- **Flexible Schemas:** You don't need to define your data structure upfront. You can add new types of information on the fly.
- **Horizontal Scaling**: Instead of buying a bigger, more expensive server (Vertical Scaling), you can just add more small, cheap servers to handle more traffic.
- **High Performance**: Many NoSQL databases are optimized for specific tasks, like lightning-fast lookups or real-time web apps.

NoSQL: Not Only SQL

Key Value

- **Usage:** Briskly changing data and high availability
- **Popular DBs:** Riak, Redis

Column Based

- **Usage:** Read/write extensions
- **Popular DBs:** HBase, Cassandra

Document Database

- **Usage:** Working with occasionally changing consistent data
- **Popular DBs:** Couchbase, MongoDB

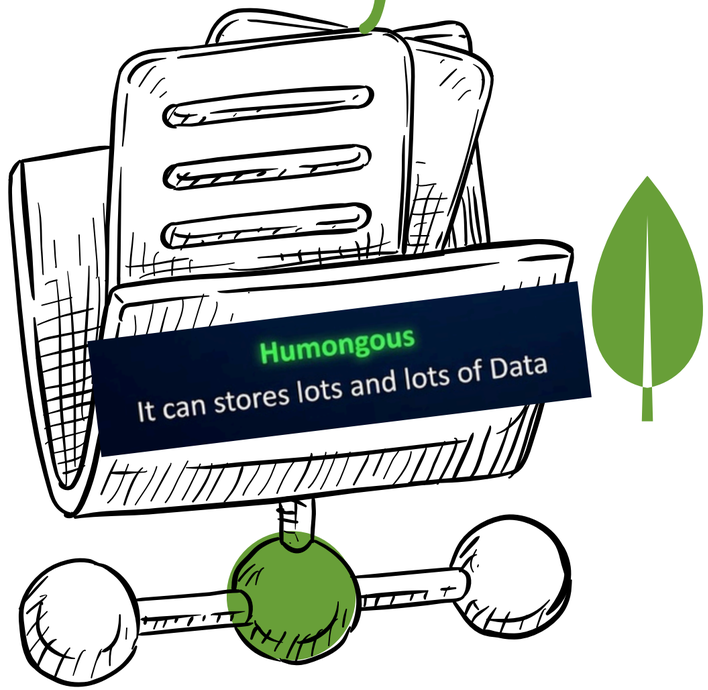
Graph Database

- **Usage:** Spatial data storage
- **Popular DBs:** Neo4J, Big data

RELATIONAL VS NONRELATIONAL

Feature	SQL (Relational)	NoSQL (Non-Relational)
Data Model	Tables with fixed rows/columns	Documents, Key-Value, Graphs
Schema	Predefined and rigid	Dynamic and flexible
Scaling	Vertical (bigger server)	Horizontal (more servers)
Best For	Complex queries and transactions	Rapid growth and varied data
Examples	MySQL, PostgreSQL, Oracle	MongoDB, Cassandra, Redis

WHAT IS MONGODB



MongoDB is currently the most popular NoSQL database. It is specifically a document-oriented database. Instead of using tables and rows, it stores data in a format called BSON (which is essentially a fast, binary version of JSON)

STORING-DATA ?

- Document: The basic unit of data (like a single record or row). It's a set of key-value pairs.
- Collection: A group of documents (like a table).
- Database: A container that holds multiple collections.

new.json

```
{
  "user_id": 101,
  "name": "Noha",
  "interests": ["coding", "hiking"],
  "status": "active"
}
```

MONGODB

Origins: From 10gen to "Humongous"

- 2007–2008: Founded by the DoubleClick team as 10gen. They pivoted from building a cloud platform to focusing specifically on the database layer because existing options weren't flexible or scalable enough.
- The Name: They renamed the product MongoDB, derived from the word "humongous," to signify its ability to handle massive data sets.

Scaling and Cloud Maturity

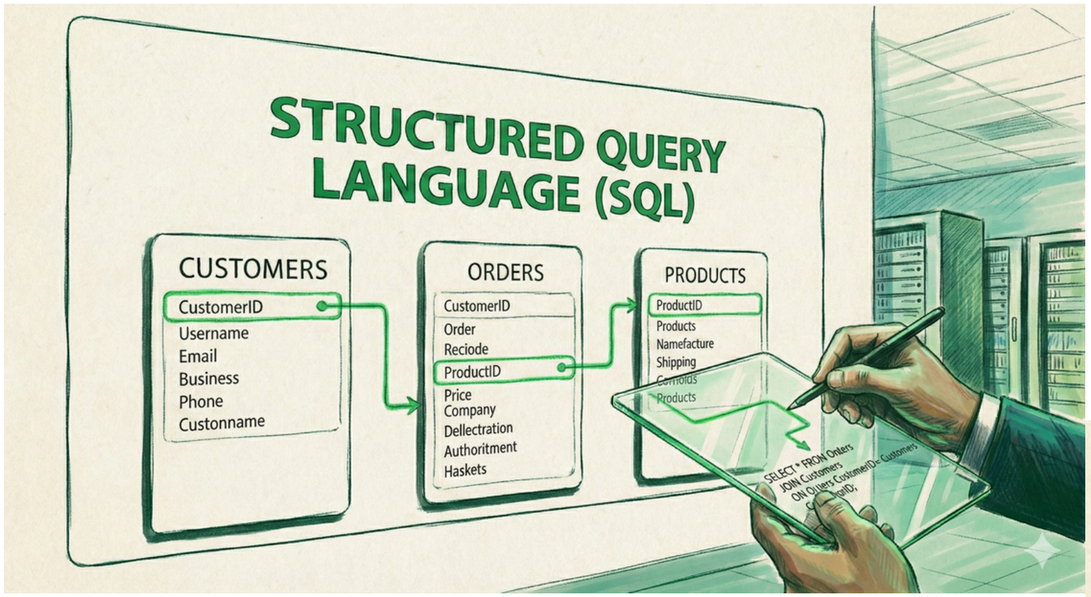
- 2009–2010: Released as open-source and introduced sharding, which allowed the database to scale horizontally across multiple servers.
- 2016: Launched MongoDB Atlas, a fully managed cloud service (DBaaS) that revolutionized how companies deploy and manage the software.

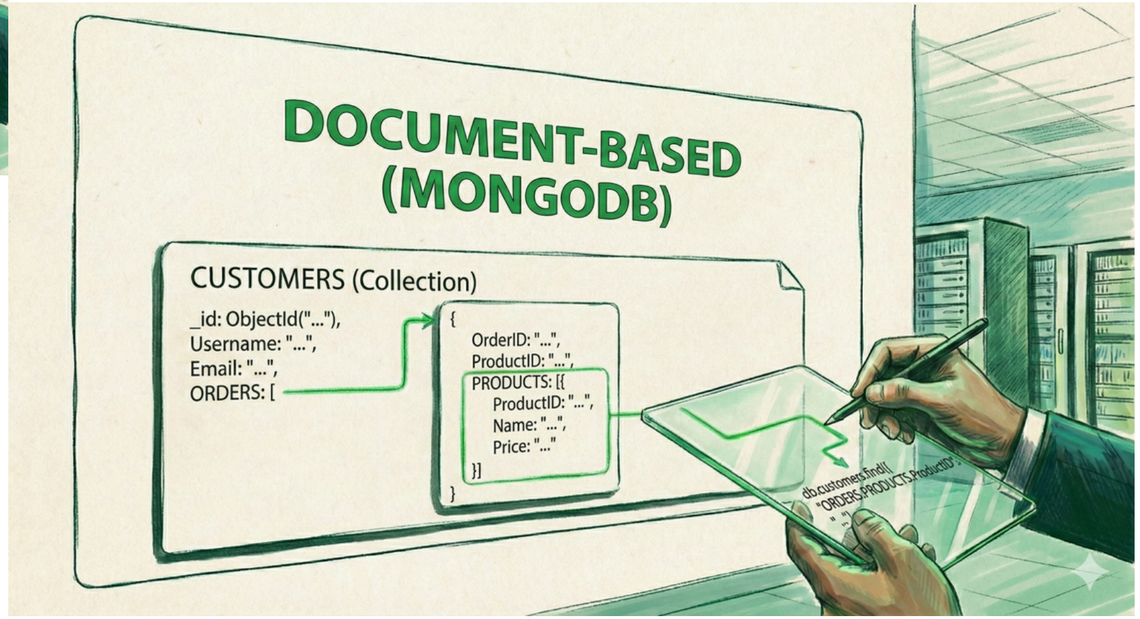
IPO and the AI Era

- 2017–2018: Went public on the NASDAQ and added ACID transactions, making the database suitable for complex financial data traditionally reserved for SQL.
- 2023–Present: Shifted focus to Generative AI, integrating Vector Search and AI-native features into the Atlas platform to support modern application development.

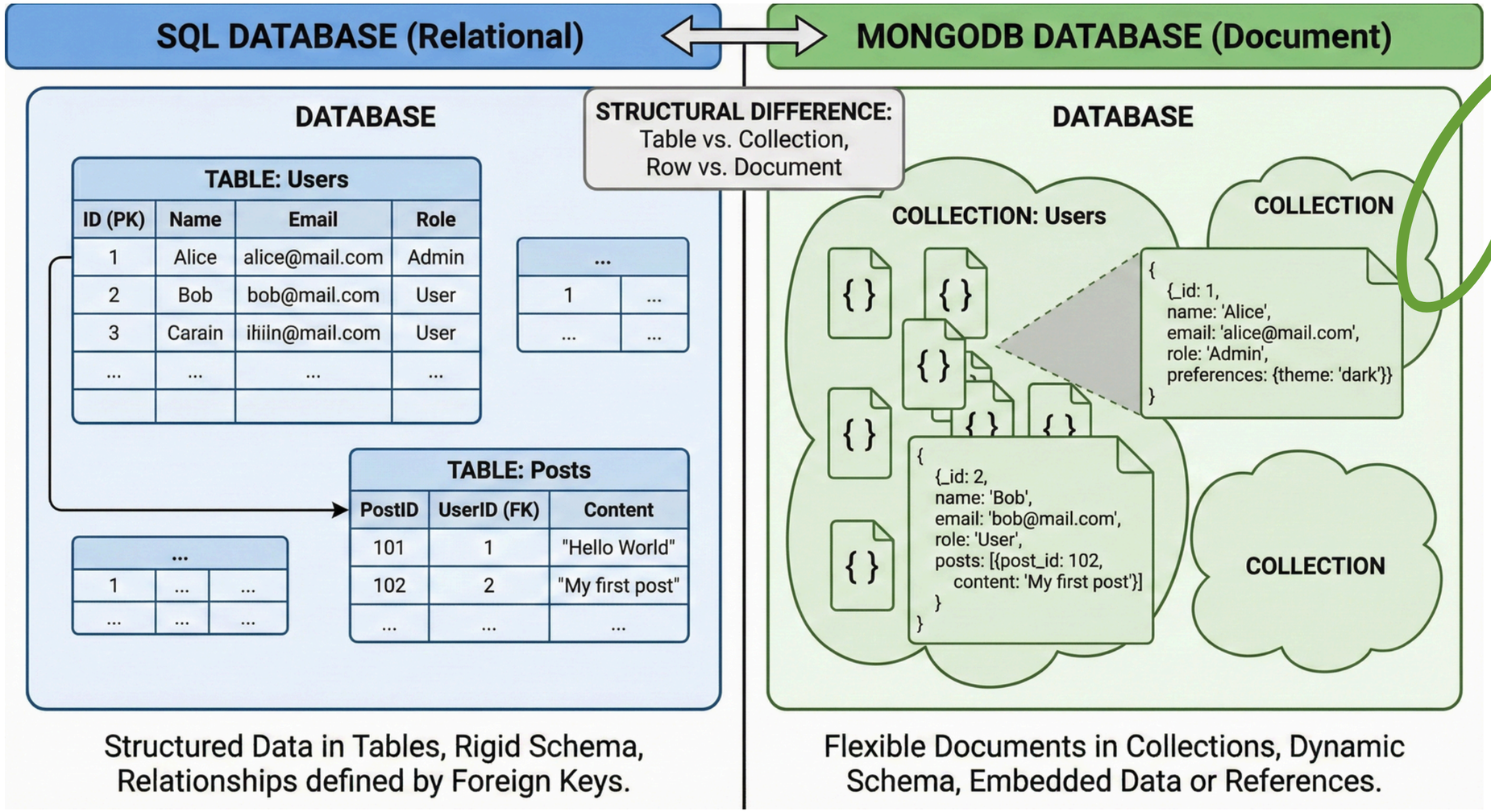
Times New Roman

SQL - NO SQL


STRUCTURED QUERY LANGUAGE (SQL)
CUSTOMERS
CustomerID
Username
Email
Business
Phone
Custonname
ORDERS
CustomerID
Order
Reciode
ProductID
Price
Company
Dellectration
Authoritment
Haskets
PRODUCTS
ProductID
Products
Namefacture
Shipping
Products


DOCUMENT-BASED (MONGODB)
CUSTOMERS (Collection)
_id: ObjectId("..."),
Username: "...",
Email: "...",
ORDERS: [
{
OrderID: "...",
ProductID: "...",
PRODUCTS: [{
ProductID: "...",
Name: "...",
Price: "..."
}]
}

SQL - NO SQL

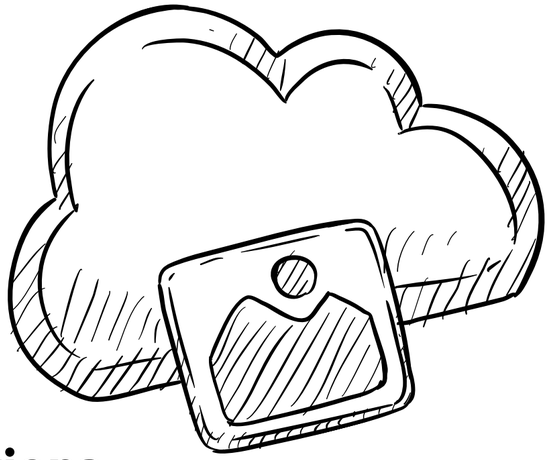

SQL DATABASE (Relational)
MONGODB DATABASE (Document)
STRUCTURAL DIFFERENCE:
Table vs. Collection,
Row vs. Document
DATABASE
TABLE: Users
ID (PK) | Name | Email | Role
1 | Alice | alice@mail.com | Admin
2 | Bob | bob@mail.com | User
3 | Carain | ihiin@mail.com | User
TABLE: Posts
PostID | UserID (FK) | Content
101 | 1 | "Hello World"
102 | 2 | "My first post"
DATABASE
COLLECTION: Users
COLLECTION
COLLECTION
{_id: 1,
name: 'Alice',
email: 'alice@mail.com',
role: 'Admin',
preferences: {theme: 'dark'}}
{_id: 2,
name: 'Bob',
email: 'bob@mail.com',
role: 'User',
posts: [{post_id: 102,
content: 'My first post'}]
}
Structured Data in Tables, Rigid Schema, Relationships defined by Foreign Keys.
Flexible Documents in Collections, Dynamic Schema, Embedded Data or References.

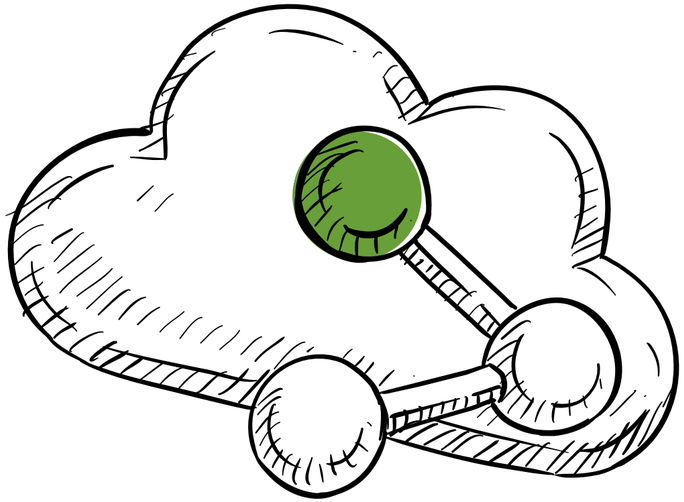
NOT-ONLY SQL

NO SCHEMA

Documents

{id:1 name:'Ali',age:30}
{id:2 name:'Ahmed',age:35}
{id:4 name:'Khaled',email:Khaled@gmail.com}





No/Few Relations

COLLECTION: Customers

```
{ id: 1, name: 'Ali', age: 30 }

{ id: 2, name: 'Khaled', age: 35 }

{ id: 3, name: 'Lara', age: 30,
  email: 'lara@gmail.com' }
```

Flexible Schema:
Lara has an extra 'email' field not present in other docs.

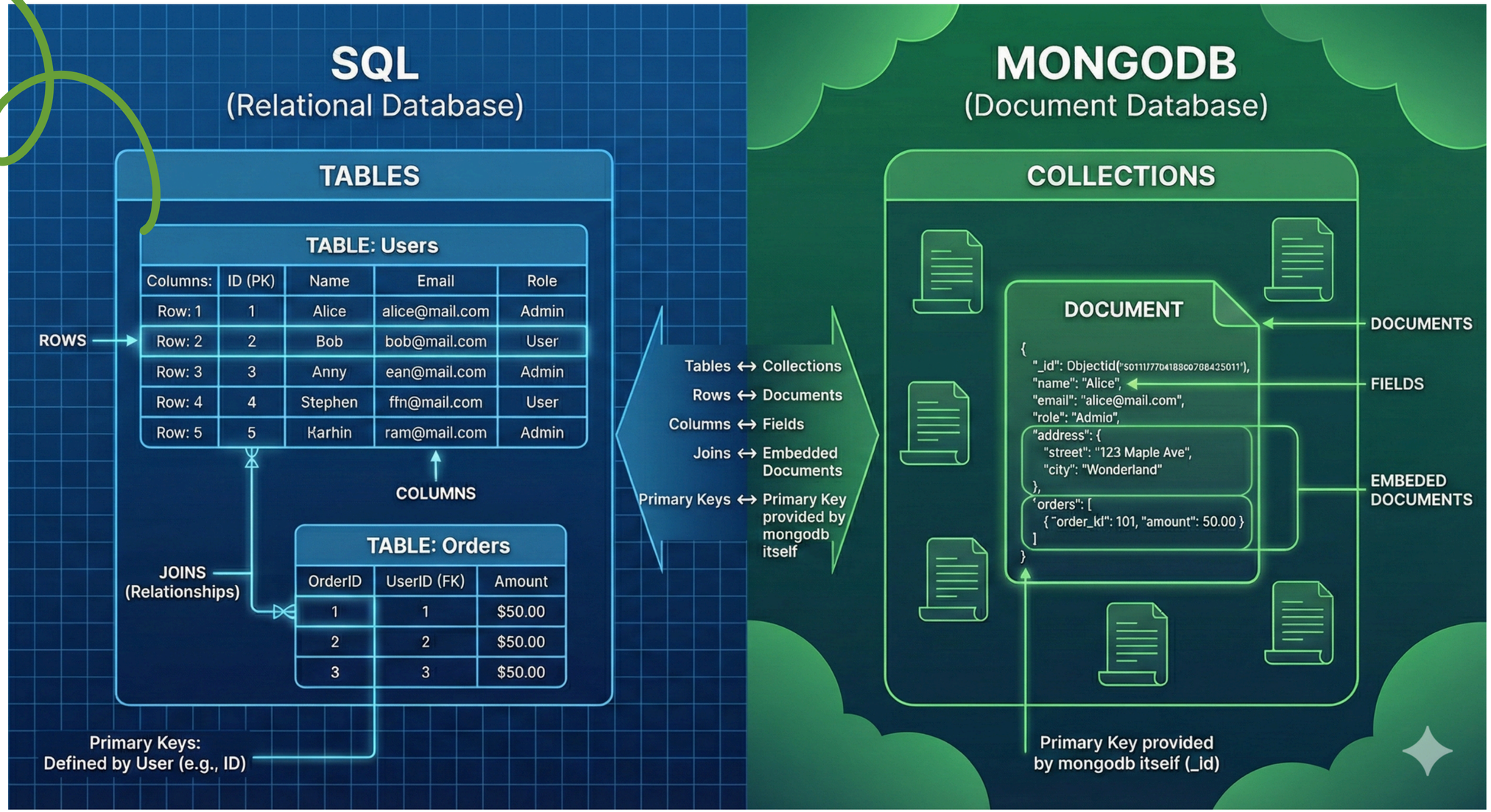
COLLECTION: Products

```
{ id: 1, title: 'hat', price: 120 }

{ id: 2, title: 'shoe', price: 320 }
```

COLLECTION: Orders

```
{ id: 14,
  customer: { id: 3, email: 'lara@gmail.com' },
  product: 3 }

{ id: 15,
  customer: { id: 2, name: 'Khaled' },
  product: 3 }
```

Embedded Document:
Customer data is nested inside the Order record (No JOINs).


SQL
(Relational Database)
TABLES
TABLE: Users
Columns:
ID (PK)
Name
Email
Role
Row: 1
1
Alice
alice@mail.com
Admin
Row: 2
2
Bob
bob@mail.com
User
Row: 3
3
Anny
ean@mail.com
Admin
Row: 4
4
Stephen
ffn@mail.com
User
Row: 5
5
Karhin
ram@mail.com
Admin
ROWS
COLUMNS
TABLE: Orders
OrderID
UserID (FK)
Amount
1
1
$50.00
2
2
$50.00
3
3
$50.00
JOINS
(Relationships)
Primary Keys:
Defined by User (e.g., ID)
Tables ↔ Collections
Rows ↔ Documents
Columns ↔ Fields
Joins ↔ Embedded Documents
Primary Keys ↔ Primary Key provided by mongodb itself
MONGODB
(Document Database)
COLLECTIONS
DOCUMENT
{
"_id": Objectid("s0111f77b4188c07d84250111"),
"name": "Alice",
"email": "alice@mail.com",
"role": "Admin",
"address": {
"street": "123 Maple Ave",
"city": "Wonderland"
},
"orders": [
{ "order_id": 101, "amount": 50.00 }
]
}
DOCUMENTS
FIELDS
EMBEDED DOCUMENTS
Primary Key provided
by mongodb itself (_id)

SQL (RELATIONAL)

Data Schemas

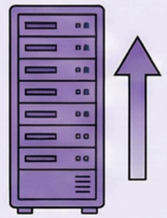
TABLE A	
Column	Type
name 1	data(10)
name 2	data(12)
name 3	Text
name 4	Text

TABLE B	
Column	Type
name 1	data(10)
name 2	Text
name 3	Text
name 4	data(10)
name 5	data(12)

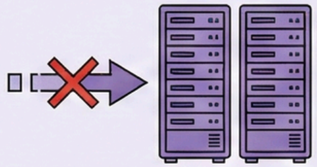
TABLE C	
Column	Type
name 1	data(10)
name 2	data(10)
name 3	Text
name 4	Text

Relations

Horizontal scaling is difficult, vertical is possible



VERTICAL SCALING
(UPGRADE HARDWARE)



HORIZONTAL SCALING
(DIFFICULT)

Limitations for huge numbers of read/write queries /second

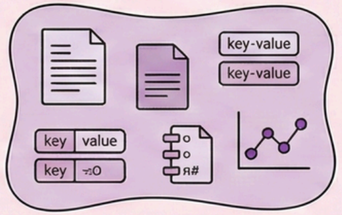
READ/WRITE

SQL vs. NoSQL DATABASE COMPARISON

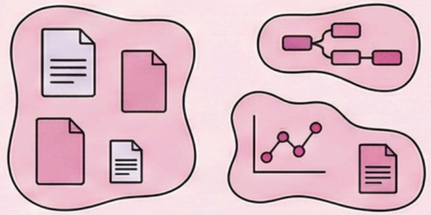
NoSQL (NON-RELATIONAL)

Schemas less

COLLECTION 1

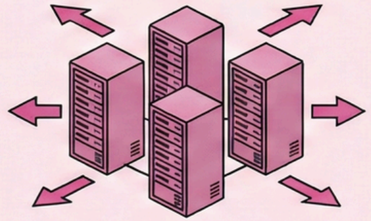


COLLECTION 2

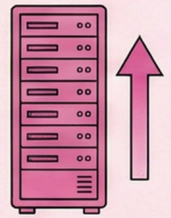


No/few Relations

Both horizontal & vertical scaling are possible

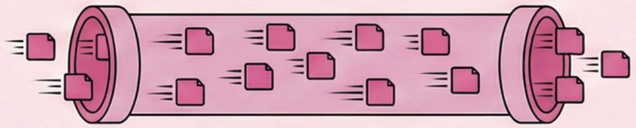


HORIZONTAL SCALING
(ADD MORE SERVERS)



VERTICAL SCALING

Great performance for huge read/write requests

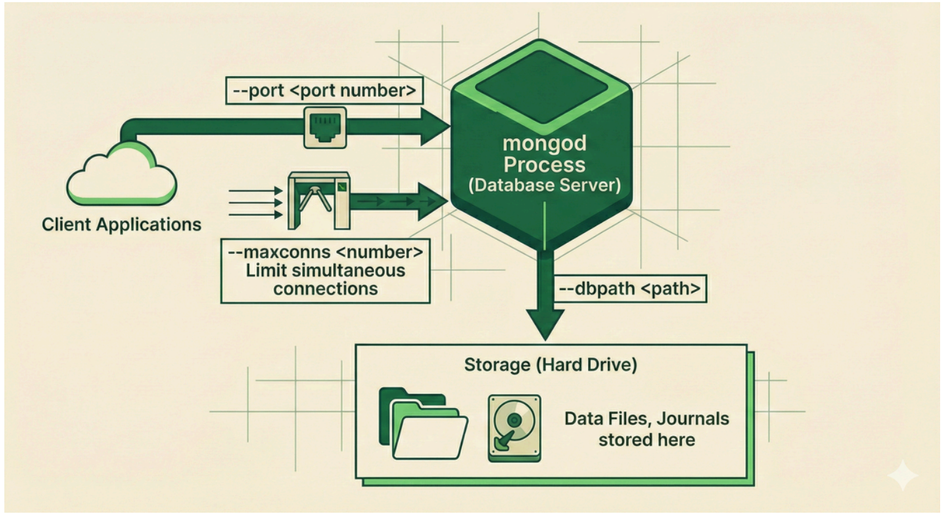


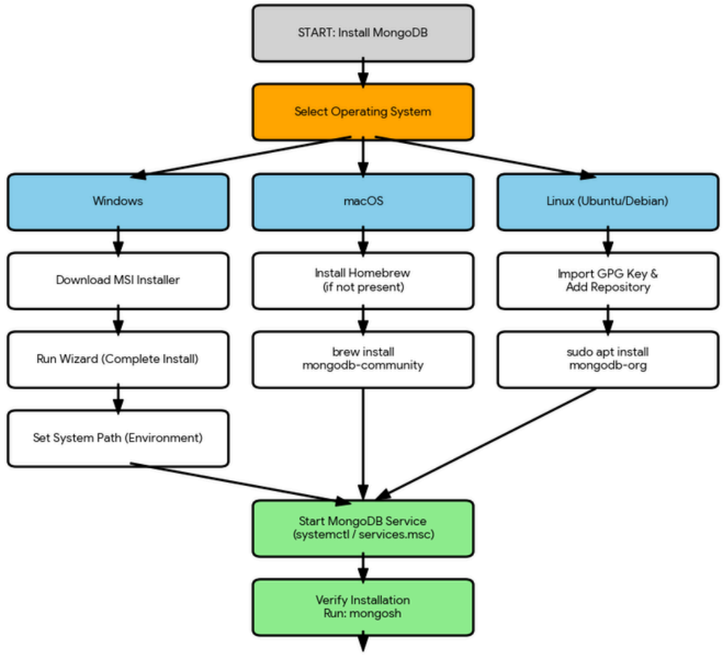
HUGE READ/WRITE

INSTALLATION

Installation

- Install mongodb service
- Install client shell
  - All instances of MongoDB come with a command line program we can use to
  - interact with our database using Javascript





commands

```
show dbs;
show collections;
db.getUsers;
```

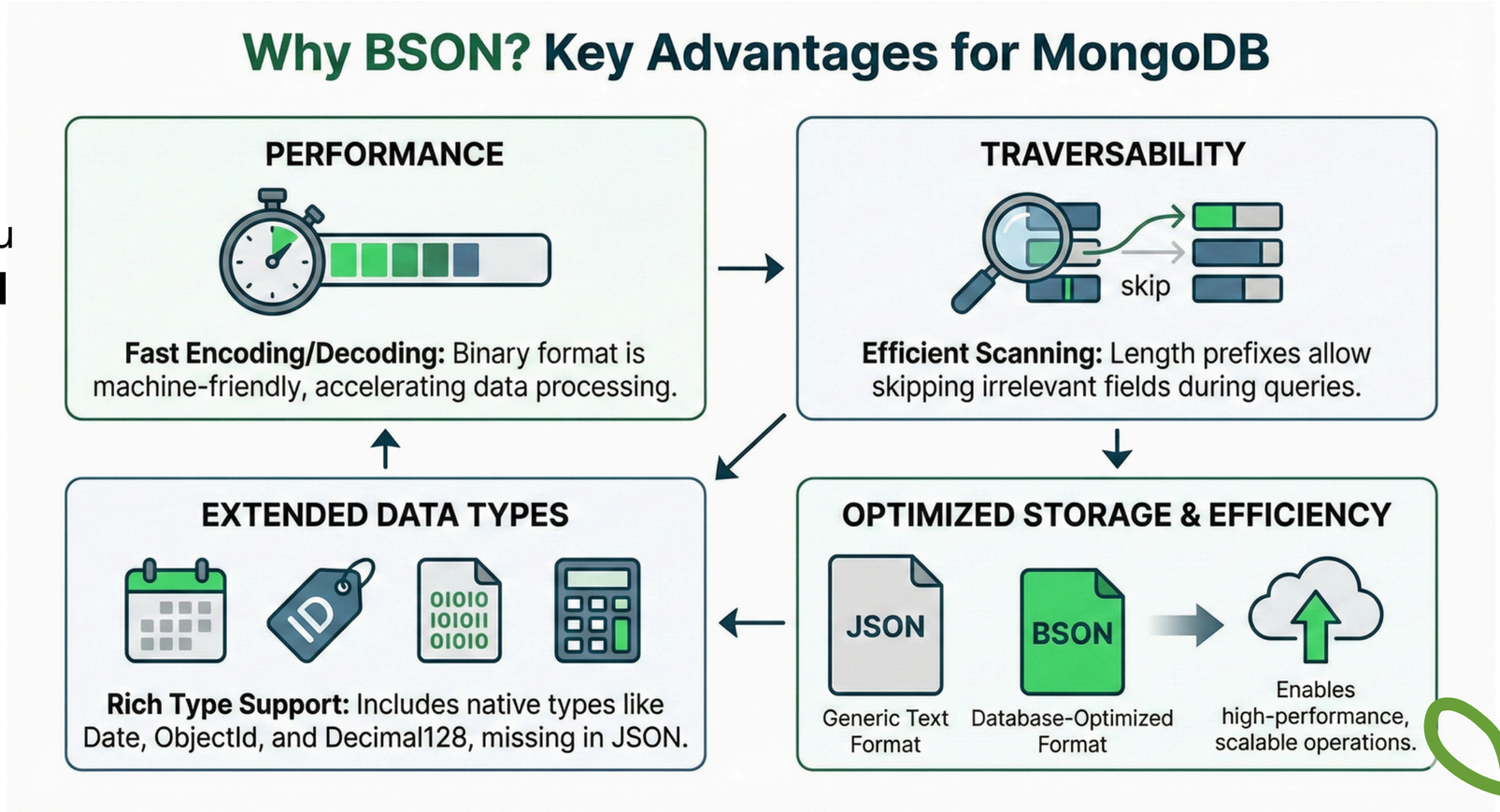
INSTALLATION

- **Mongod**
  - **Default Port: 27017**
  - The mongod (MongoDB Daemon) is the primary background process for the MongoDB system.
  - It is the heart of your database operations, acting as the bridge between your data and your applications.
- **Mongod Responsibilities**
  - Data Request Handling:
    - Processes all incoming queries and commands from the client (e.g., mongosh or application drivers).
  - Data Access Management:
    - Controls how data is read from and written to the disk.
  - Background Operations:
    - Performs essential "housekeeping" tasks such as:
    - Index building and maintenance.
    - Memory management.
    - Data compaction and cleanup.

BSON

JSON and BSON

- Think of JSON as the language you speak to the database, and **BSON** as the optimized language the database uses to store and organize that information efficiently.
- BSON is a **binary-encoded** serialization of JSON-like documents.

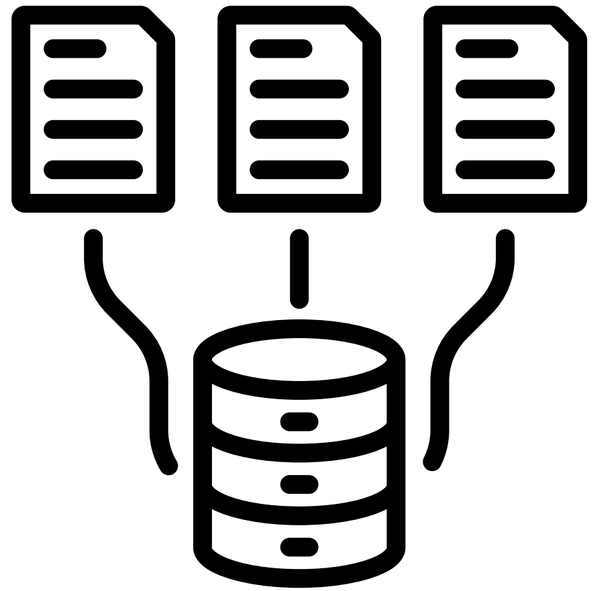


COLLECTIONS

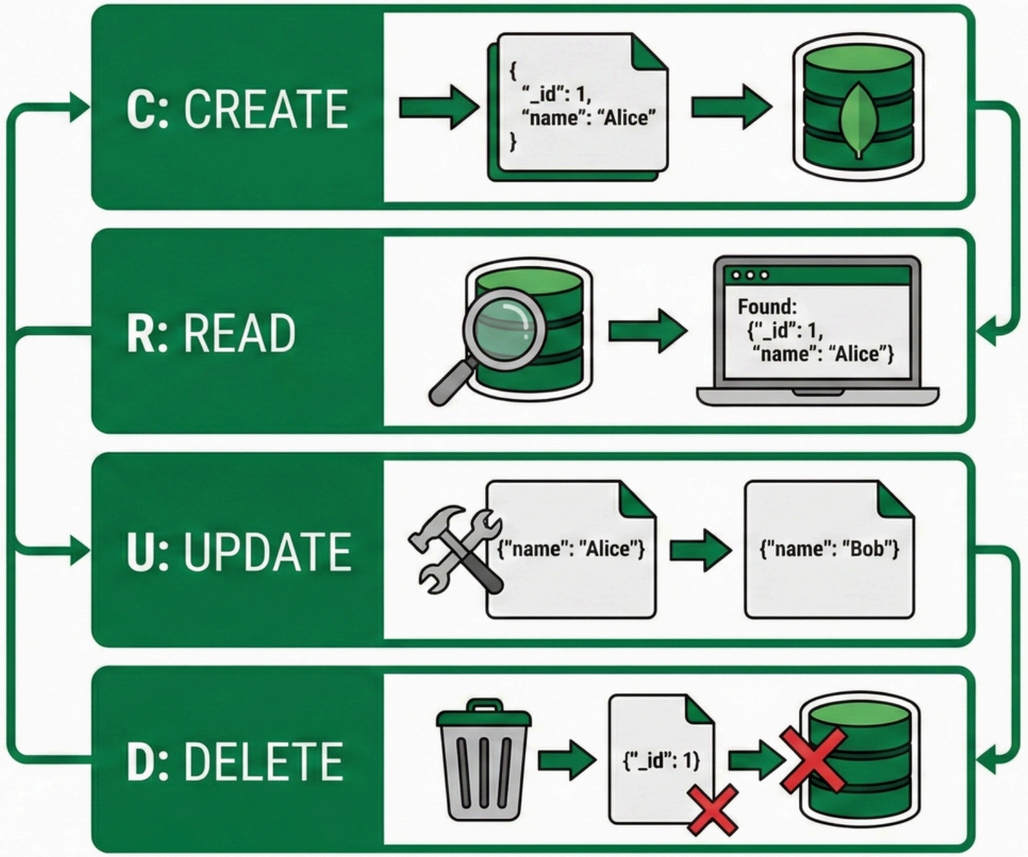
- MongoDB Collection
  - A MongoDB collection is a grouping of documents stored within a MongoDB database, acting as the NoSQL equivalent of a table in a relational database. **Since collections are schema-less, documents within the same collection can have different fields and structures.**

commands

```
db.createCollection("customers");

show collections

db.getCollectionNames()

db.getCollectionInfos({name:'employees'})

db.customers.drop()
```



CRUD Operations in MongoDB



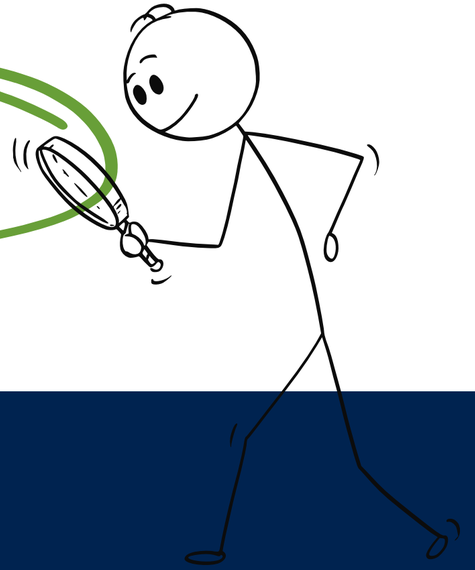
INSERT

- insertOne(): insert only one document into a collection
- insertMany(): insert multiple documents into a collection

commands

```
db.users.insertOne({
  name: "Noha Shehab",
  email: "nshehab@iti.gov.eg",
  age: 33,
  status: "active"
});

db.products.insertMany([
  { item: "Laptop", qty: 25, tags: ["electronics", "office"] },
  { item: "Coffee Mug", qty: 100, tags: ["kitchen", "home"] },
  { item: "Desk Chair", qty: 15, tags: ["furniture"] }
]);
```

FIND..



commands

```
Find All     ⟶ db.users.find({})
Exact Match  ⟶ db.users.find({ "city": "London" })
Comparison   ⟶ db.users.find({ "age": { "$gte": 21 } }) (Age ≥ 21)
Sort & Limit⟶ db.users.find().sort({ "joinDate": -1 }).limit(10)
```

commands

```
Projection

db.instructors.find(
    {}, // condition
    {firstName:1 , lastName:1} // projection
)
```