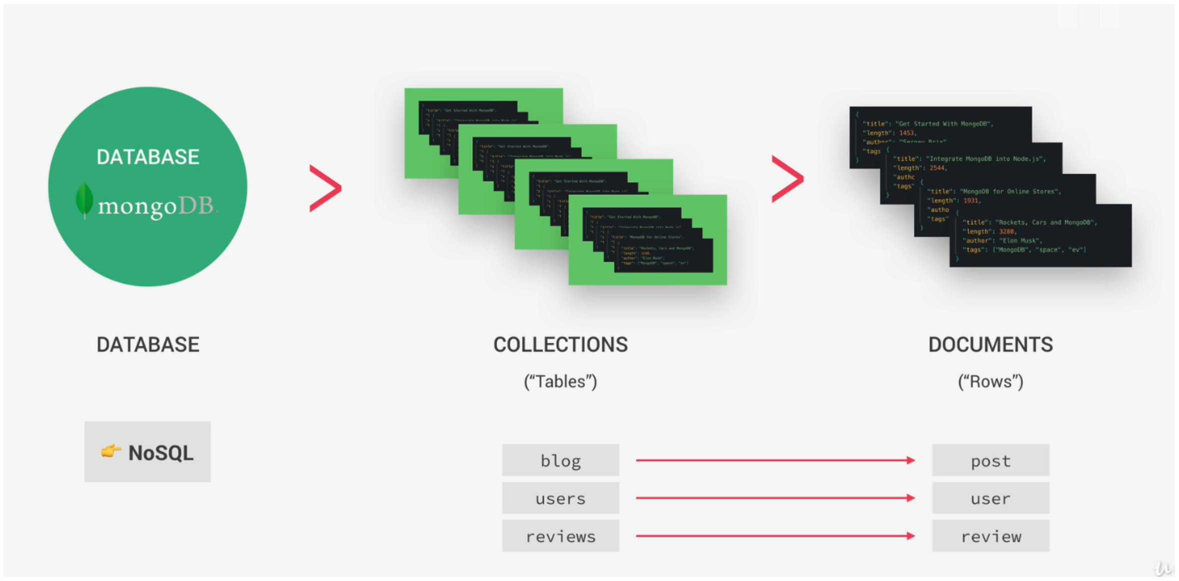
MongoDB Basics

▼ **What is MongoDB**

- It is a NoSQL database
- Each database can contain one or more **collections** (like tables in SQL databases).
- Each collection can contain one or more data structure called **document** (like rows in SQL databases)

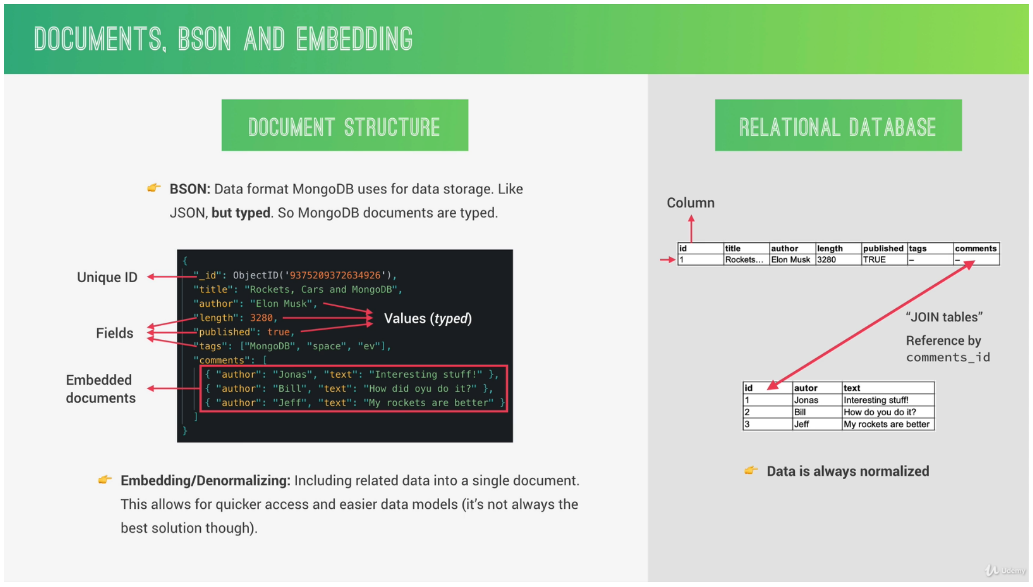
▼ Image



▼ **Features of MongoDB**

- **Document based:** MongoDB stores data in documents (field-value pair data structures, NoSQL)
- **Scalable:** Very easy to distribute data across multiple machines as your users and amount of data grows
- **Flexible:** No document data schema required, so each document can have different number and type of fields
- **Performant:** Embedded data models, indexing, sharding, flexible documents, native duplication, etc
- Free and open-source, published under SSPL License

▼ **Documents, BSON, and Embedding**



DOCUMENTS, BSON AND EMBEDDING

DOCUMENT STRUCTURE

BSON: Data format MongoDB uses for data storage. Like JSON, **but typed**. So MongoDB documents are typed.

```
{
  "_id": ObjectID('9375209372634926'),
  "title": "Rockets, Cars and MongoDB",
  "author": "Elon Musk",
  "length": 3280,
  "published": true,
  "tags": ["MongoDB", "space", "ev"],
  "comments": [
    { "author": "Jonas", "text": "Interesting stuff!" },
    { "author": "Bill", "text": "How did oyu do it?" },
    { "author": "Jeff", "text": "My rockets are better" }
  ]
}
```

Embedding/Denormalizing: Including related data into a single document. This allows for quicker access and easier data models (it's not always the best solution though).

RELATIONAL DATABASE

id	title	author	length	published	tags	comments
1	Rockets...	Elon Musk	3280	TRUE	–	–

"JOIN tables"

Reference by `comments_id`

id	autor	text
1	Jonas	Interesting stuff!
2	Bill	How do you do it?
3	Jeff	My rockets are better

Data is always normalized

▼ Creating a local database

```
use natours-test
```

- use command is used to switch to or create a new database

```
db.tours.insertOne()
```

- In the current db add tours collection (creating a tours collection in natours-test database)

```
db.tours.insertOne({ name: "The Forest Hiker", price: 297, ratings: 4.7})
```

- Inserting document inside tours collection
- Document is created after this command

```
db.tours.find()
```

- See the document created
- ID is also added automatically

```
show dbs
```

- show all the databases

```
show collections
```

- show all the collections in a database

▼ CRUD - Creating Documents

```
db.tours.insertMany()
```

- insertMany() accepts array of documents/objects to be inserted in a collection

```
db.tours.insertMany([{ name: "The Sea Explorer", price: 497, rating: 4.8}, { name: "The Snow Adventurer", price:997, rating: 4.9, diffi
```

- inserted array of documents/objects with insertMany()

▼ CRUD - Querying (Reading) Documents, (Operators)

```
db.tours.find({ name: "The Forest Hiker" })
db.tours.find({difficulty: "easy"})
```

- Finds the specific document with name “The Forest Hiker”

```
db.tours.find({ price: {$lte: 500} })
```

- lte stands for less than or equal. Find tour with price less than 500.
- $ sign is used for operator

```
db.tours.find({ price: {$lt: 500}, rating: {$gte: 4.8} })
```

- lt is less than. gte is greater than or equal to
- chaining multiple operators (two and operations)

```
db.tours.find({ $or: [ {price: {$lt: 500}}, {rating: {$gte: 4.8}} ] })
```

- using or operator (either of one equation is true)

```
db.tours.find({ $or: [ {price: {$lt: 500}}, {rating: {$gte: 4.8}} ] }, {name: 1})
```

- only show name of the matched query, not other properties

▼ CRUD - Updating Documents

```
db.tours.updateOne({ name: "The Snow Adventurer" }, { $set: { price: 597 } })
```

- updateOne() have 2 parameters, the query for the object and the set condition to update to value
- If multiple objects/documents are present, then the first document is changed
- If you want to change the value of all the documents, use updateMany()

▼ CRUD - Deleting Documents

```
db.tours.deleteMany({ ratings: {$lt: 4.8} })
```

- deleteOne() for deleting single document and deleteMany() to delete multiple documents

```
db.tours.deleteMany({})
```

- delete all the documents (be careful with this, no undo)