Tecnologie e applicazioni web

MongoDB

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Academic year: 2021/2022

About MongoDB

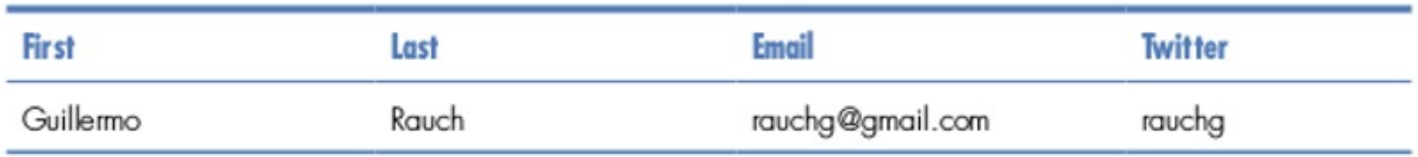
MongoDB is a DBMS:

- Non relational
- Oriented to documents (and not to data relations)
- With a dynamic schema (schema-less)
- JSON-style documents

Why is it interesting?
Nice integration with dynamic languages like JavaScript. Database structure can change on-the-fly while developing our application

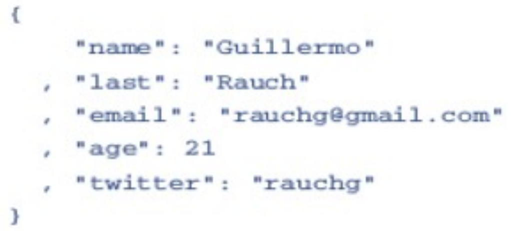
About MongoDB

Relational database

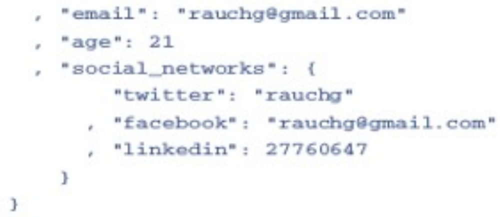


First	Last	Email	Twitter
Guillermo	Rauch	rauchg@gmail.com	rauchg

MongoDB:



```
{
    "name": "Guillermo"
  , "last": "Rauch"
  , "email": "rauchg@gmail.com"
  , "age": 21
  , "twitter": "rauchg"
}
```



```
  , "email": "rauchg@gmail.com"
  , "age": 21
  , "social_networks": {
        "twitter": "rauchg"
      , "facebook": "rauchg@gmail.com"
      , "linkedin": 27760647
    }
}
```

Features

Query ad Hoc:

Supports querying document fields, intervals and regular expressions

Indexing:

Every field can be indexed to speedup the queries

Aggregation:

Supports efficient data aggregation functions (to compute different statistics on data)

Features

File storage:

Can be used as a distributed filesystem. Files are chunked and distributed on multiple nodes (GridFS)

Sharding:

Data in a collection can be distributed to multiple MongoDB nodes in a Cloud infrastructure. Supports automatic load balancing mechanisms.

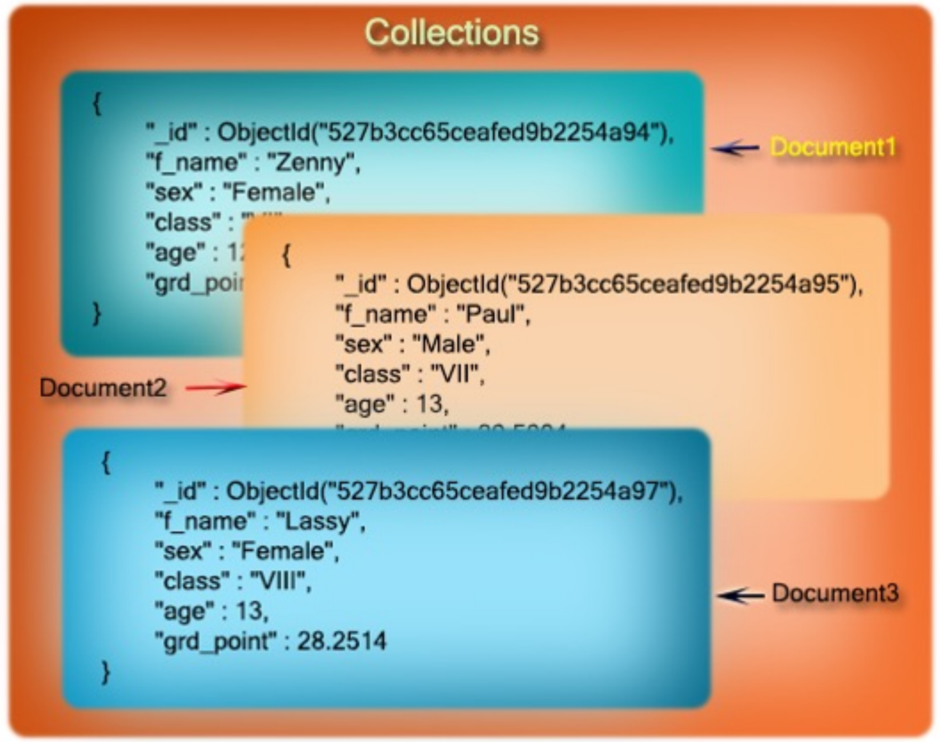
MongoDB

https://docs.mongodb.com/manual/#

The DBMS allows the creation of multiple **databases**.

Each **database** is composed by **collections**.
Each **collection** is a set of **documents**
Each **document** is composed by one or more **fields**

MongoDB

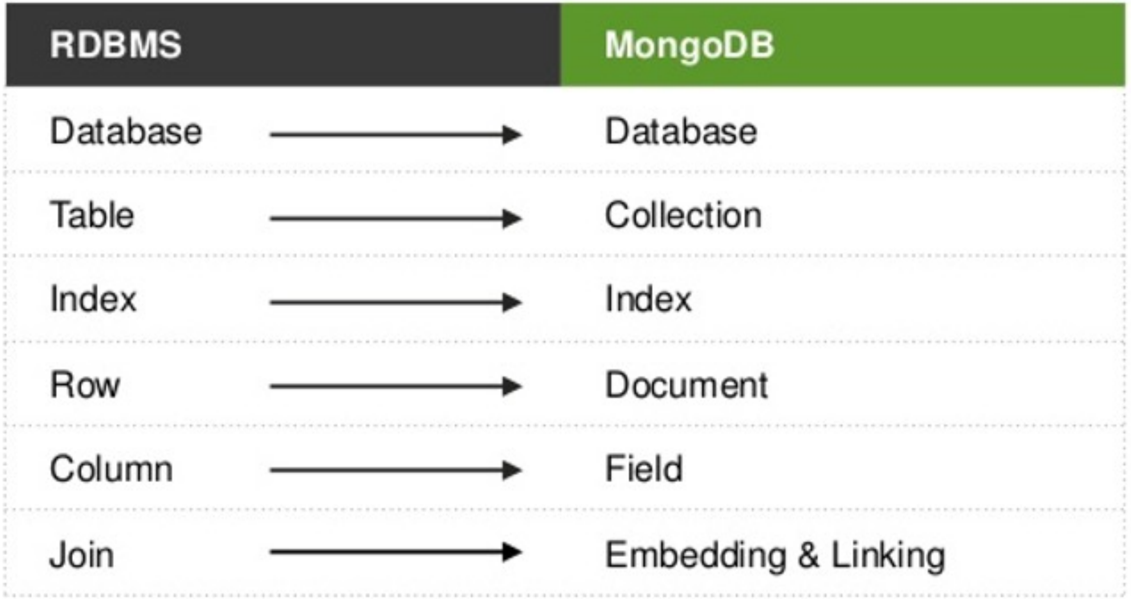


```
{
        "_id" : ObjectId("527b3cc65ceafed9b2254a94"),
        "f_name" : "Zenny",
        "sex" : "Female",
        "class" : "
        "age" : 1
        "grd_poi
}
```

```
{
        "_id" : ObjectId("527b3cc65ceafed9b2254a95"),
        "f_name" : "Paul",
        "sex" : "Male",
        "class" : "VII",
        "age" : 13,
```

```
{
        "_id" : ObjectId("527b3cc65ceafed9b2254a97"),
        "f_name" : "Lassy",
        "sex" : "Female",
        "class" : "VIII",
        "age" : 13,
        "grd_point" : 28.2514
}
```

Terminology



RDBMS		MongoDB
Database	→	Database
Table	→	Collection
Index	→	Index
Row	→	Document
Column	→	Field
Join	→	Embedding & Linking

MongoDB vs. relational

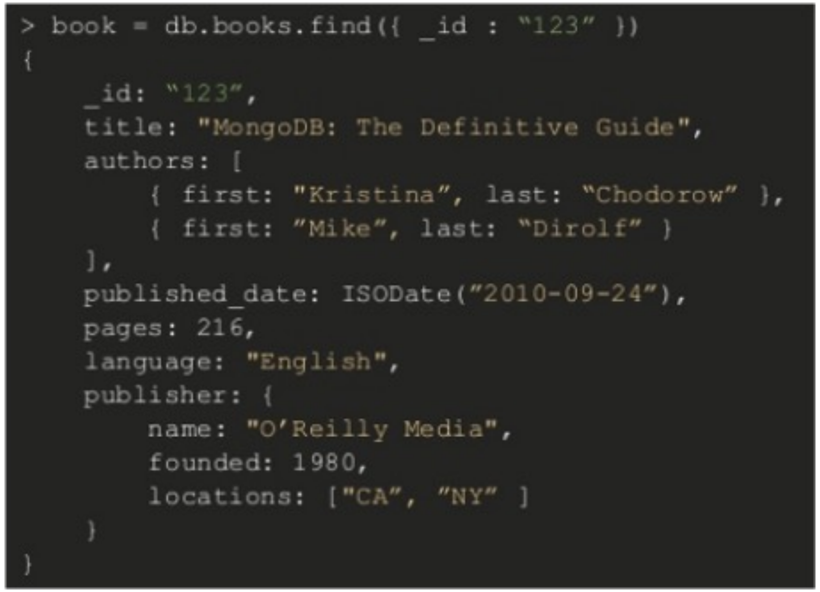
- Each document in a collection can be composed by different fields
  - Increased flexibility since data can be stored and loaded without a predefined schema
- Each document can contain other documents (Embedding)
  - This mechanism can replace the usual join operation in relational databases.

One-to-many relations

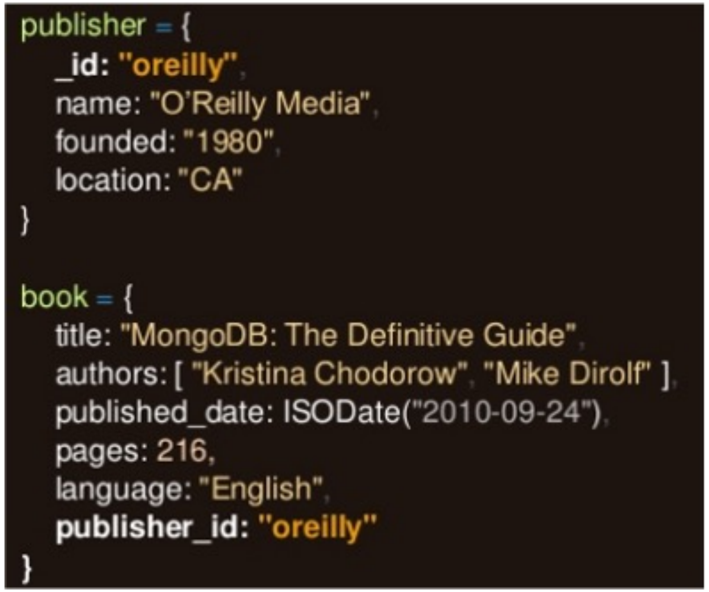
Two alternatives:

1. By embedding documents inside the same parent document (fast readings but might be more complex to keep data consistency)
2. By referencing document ids like in relational databases (slower readings but data is not replicated)

One-to-many relations



```
> book = db.books.find({ _id : "123" })
{
    _id: "123",
    title: "MongoDB: The Definitive Guide",
    authors: [
        { first: "Kristina", last: "Chodorow" },
        { first: "Mike", last: "Dirolf" }
    ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher: {
        name: "O'Reilly Media",
        founded: 1980,
        locations: ["CA", "NY" ]
    }
}
```



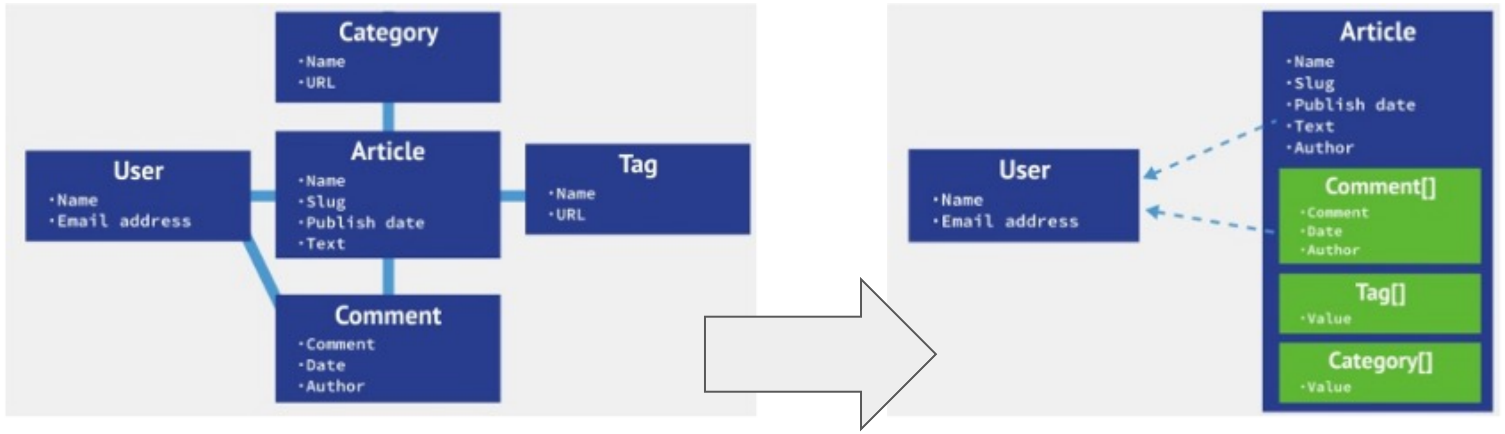
```
publisher = {
    _id: "oreilly",
    name: "O'Reilly Media",
    founded: "1980",
    location: "CA"
}

book = {
    title: "MongoDB: The Definitive Guide",
    authors: [ "Kristina Chodorow", "Mike Dirolf" ],
    published_date: ISODate("2010-09-24"),
    pages: 216,
    language: "English",
    publisher_id: "oreilly"
}
```

One-to-many relations

Which solution to prefer? Depends on the specific scenario… but MongoDB offers greater flexibility than a relational database

Example: blogging platform (embedding)



MongoDB Shell

A simple command line tool to execute CRUD (Create, Read, Update, Delete) operations is included.

Syntax JavaScript-like with APIs similar to the ones usable with Node.js.

```
$ mongo
> show log global;
```

MongoDB Shell

Command	Description
> show dbs	Visualize the database list
> use <db>	Change the currently active database
> show collections	Visualize all the collections of the currently active database
> db.<collection>.find()	Shows all the documents in a collection (this is actually a query…)

Query

Reading operations on documents are realized by providing special documents named **Query Filter Documents**

https://docs.mongodb.com/manual/tutorial/query-documents/

```
{
  <field1>: <value1>,
  <field2>: { <operator>: <value> },
  ...
}
```

Query

Examples:

```
db.inventory.find({ status: "A", qty: { $lt: 30 }})

SELECT * FROM inventory WHERE status = "A" AND qty < 30


db.inventory.find({ status: "A" },{ item: 1,status: 1 })

SELECT _id, item, status from inventory WHERE status = "A"
```

Atomic operations

MongoDB operations are atomic at a document level (including all the embedded documents)

- Transactions involving the modification of multiple documents should be manually implemented with the two-phase-commits pattern

Moreover, a client can observe document modifications before they are made persistent (**read uncommitted** behaviour)

Using MongoDB in Node.js

We can use MongoDB with the ufficial Node.js driver: https://www.npmjs.com/package/mongodb

- Insert “mongodb” as a package.json dependence
- Get the `MongoClient` object to establish a connection to the database and interact with the collections

Mongoose

Mongoose is a popular Object Document Mapping **ODM** library to map JavaScript objects in MongoDB

Allow us to define a document schema through JavaScript objects and to perform an automatic mapping from/to the database

http://mongoosejs.com/docs/guide.html

Mongoose

Mongoose most important concepts are:

Schemas:

To describe the document structure of a certain collection (together with their methods!)

Models:

Are functions (constructors) to instantiate objects given a certain schema and store them automatically in the database

Mongoose models

Once a model is defined, it can be used to:

- Query the database
    - Ex: <model>.find({ })
- Create and store a new object
    - Ex: <model>.create({obj})
- Remove existing objects
    - Ex: <model>.remove({}) or <model>.deleteOne({})