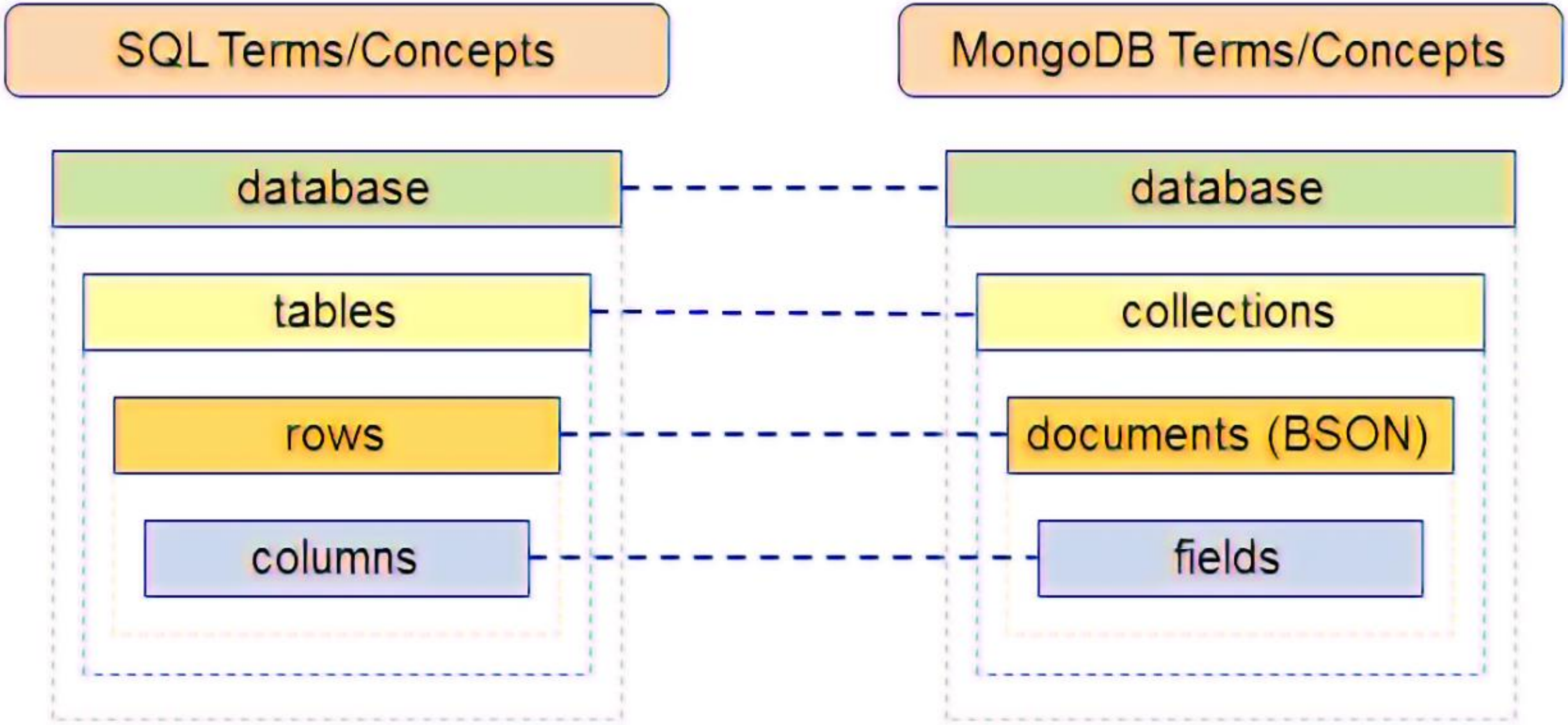
MongoDB

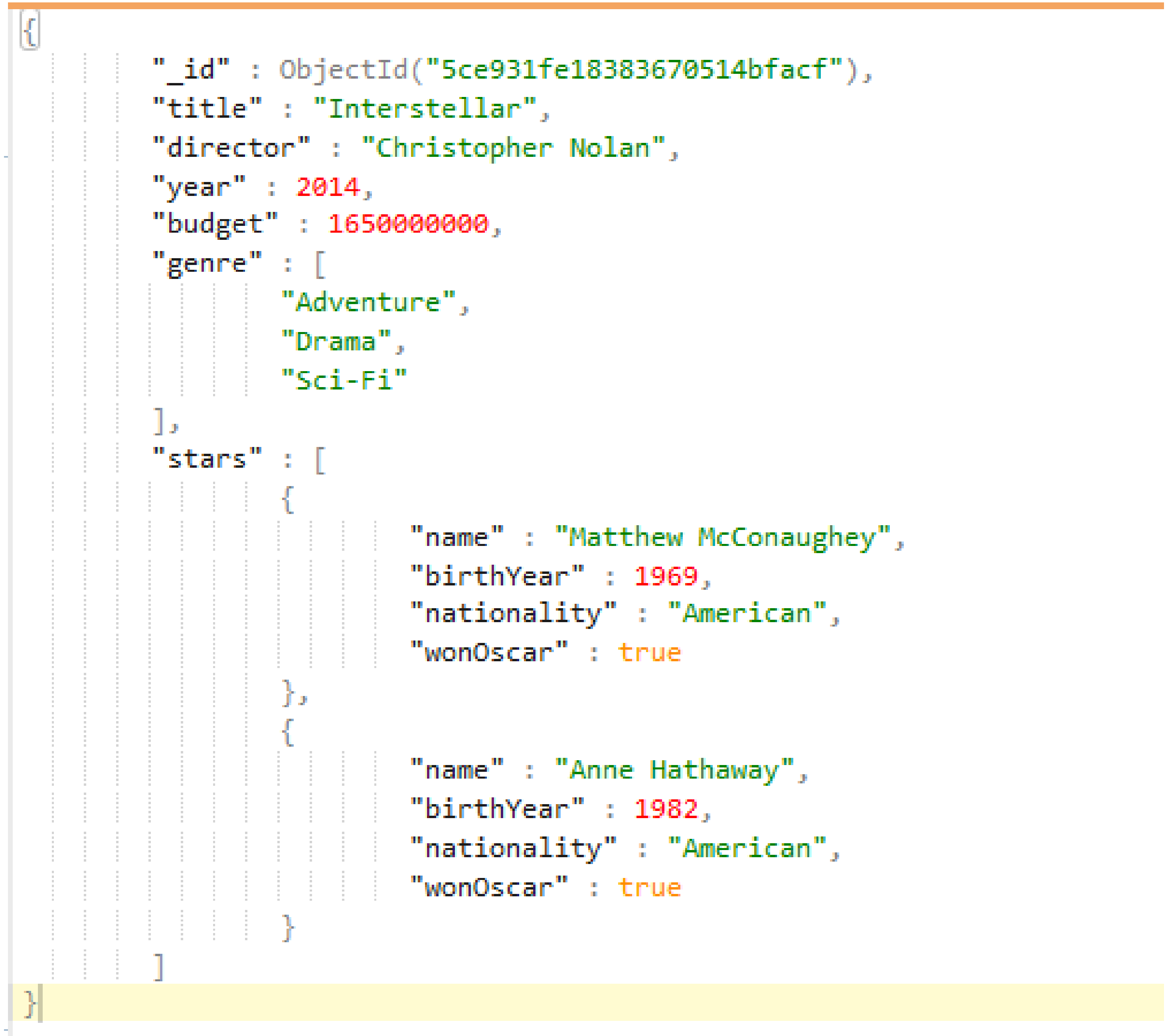
Intro

- MongoDB is based on a NoSQL database that is used for storing data in a key-value pair. Its working is based on the concept of document and collection.

- Mongo DB can be defined as a document-oriented database system that uses the concept of NoSQL. It also provides high availability, high performance, along with automatic scaling. This open-source product was developed by the company - 10gen in October 2007



SQL Terms/Concepts	MongoDB Terms/Concepts
database	database
tables	collections
rows	documents (BSON)
columns	fields



```
{
        "_id" : ObjectId("5ce931fe18383670514bfacf"),
        "title" : "Interstellar",
        "director" : "Christopher Nolan",
        "year" : 2014,
        "budget" : 1650000000,
        "genre" : [
                "Adventure",
                "Drama",
                "Sci-Fi"
        ],
        "stars" : [
                {
                        "name" : "Matthew McConaughey",
                        "birthYear" : 1969,
                        "nationality" : "American",
                        "wonOscar" : true
                },
                {
                        "name" : "Anne Hathaway",
                        "birthYear" : 1982,
                        "nationality" : "American",
                        "wonOscar" : true
                }
        ]
}
```

What is database?

- In MongoDB, a database can be defined as a physical container for collections of data. Here, on the file system, every database has its collection of files residing. Usually, a MongoDB server contains numerous databases.

- mongod is the server, to which requests to connect database are passed

What are collections

- Collections can be defined as a cluster of MongoDB documents that exist within a single database. You can relate this to that of a table in a relational database management system. MongoDB collections do not implement the concept of schema. Documents that have collection usually contain different fields. Typically, all the documents residing within a collection are meant for a comparable or related purpose.

What are documents?

- A document can be defined as a collection of key-value pairs that contain dynamic schema. Dynamic schema is something that documents of the equal collection do not require for having the same collection of fields or construction, and a common field is capable of holding various types of data.

Here is a table showing the relation between the terminologies used in RDBMS and MongoDB:

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple or Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary key / Default key
Mysqld / Oracle	mongod

Advantages of using MongoDB

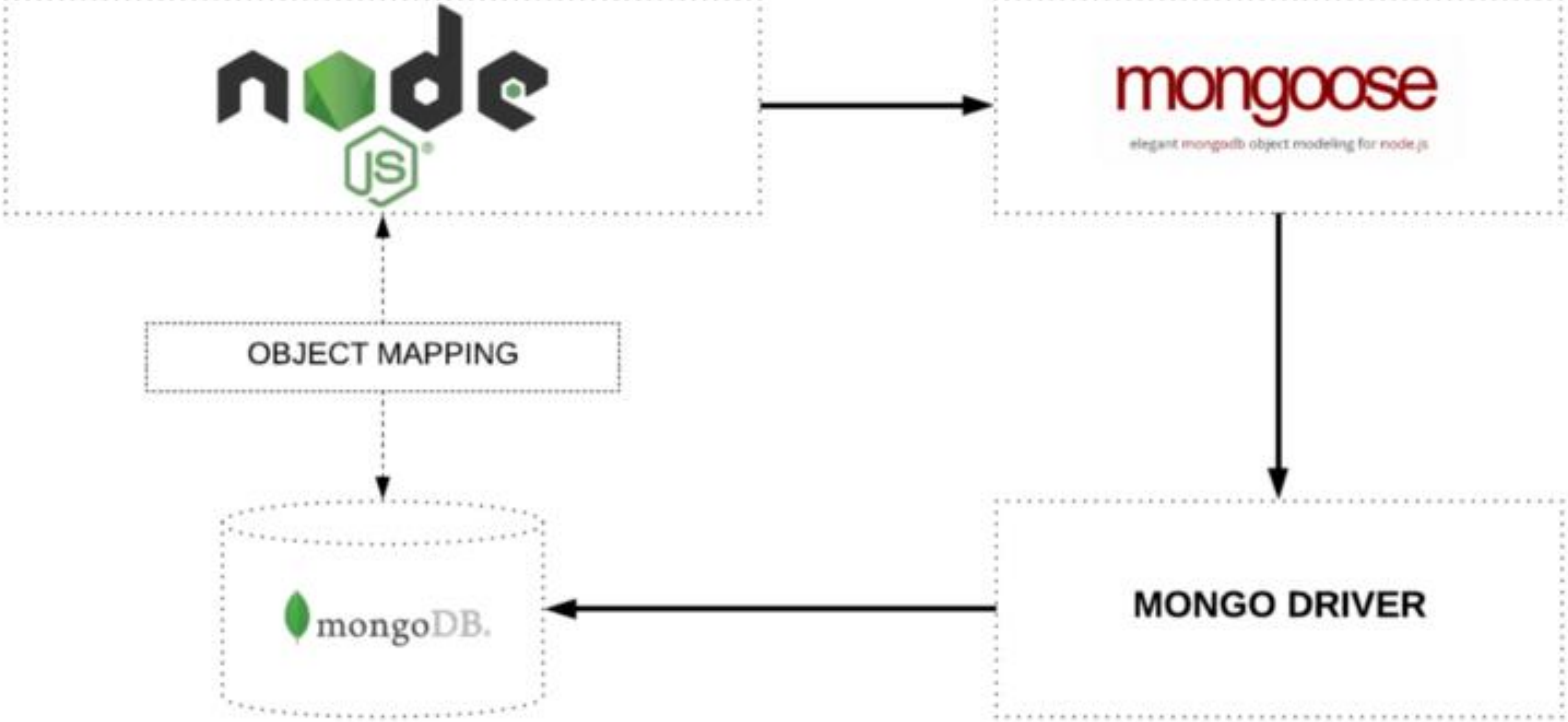
- It is easy to set up, i.e., install the MongoDB.
- Since MongoDB is a schema-less database, so there is no hassle of schema migration.
- Since it is a document-oriented language, document queries are used, which plays a vital role in supporting dynamic queries.
- Easily scalable.
- It is easy to have a performance tuning as compared to other relational databases.
- It helps in providing fast accessing of data because of its nature of implementing the internal memory to store the data.
- MongoDB is also used as a file system that can help in easy management of load balancing.
- MongoDB also supports the searching using the concept of regex (regular expression) as well as fields.
- Users can run MongoDB as a windows service also.
- It does not require any VM to run on different platforms

What is Mongoose?

- Mongoose is an Object Document Mapper (ODM). This means that Mongoose allows you to define objects with a strongly-typed schema that is mapped to a MongoDB document.
- Mongoose provides an incredible amount of functionality around creating and working with schemas. Mongoose currently contains eight SchemaTypes that a property is saved as when it is persisted to MongoDB. They are:
  - String
  - Number
  - Date
  - Buffer
  - Boolean
  - Mixed
  - ObjectId
  - Array
- Each data type allows you to specify:
  - a default value
  - a custom validation function
  - indicate a field is required
  - a get function that allows you to manipulate the data before it is returned as an object
  - a set function that allows you to manipulate the data before it is saved to the database
  - create indexes to allow data to be fetched faster

Data type

- The **Number** and **Date** properties both support specifying a minimum and maximum value that is allowed for that field.
- Most of the eight allowed data types should be quite familiar to you. However, there are several exceptions that may jump out to you, such as **Buffer, Mixed, ObjectId, and Array.**
- The **Buffer** data type allows you to save binary data. A common example of binary data would be an image or an encoded file, such as a PDF document.
- The **Mixed data** type turns the property into an "anything goes" field. This field resembles how many developers may use MongoDB because there is no defined structure. Be wary of using this data type as it loses many of the great features that Mongoose provides, such as data validation and detecting entity changes to automatically know to update the property when saving.
- The **ObjectId** data type commonly specifies a link to another document in your database. For example, if you had a collection of books and authors, the book document might contain an ObjectId property that refers to the specific author of the document.
- The **Array data** type allows you to store JavaScript-like arrays. With an Array data type, you can perform common JavaScript array operations on them, such as push, pop, shift, slice, etc.


node JS
mongoose
elegant mongodb object modeling for node.js
OBJECT MAPPING
mongoDB.
MONGO DRIVER