mongoDB

{ name: mongo, type: DB }

mongoDB

MongoDB

The Complete Developer's Guide

What is MondoDB?

MongoBD is a database which is created by the company who is also called MongoDB. The name stems from the word “humongous”. This database is built to store a lot of data but also being able to work with the huge data efficiently. Ultimately, this is a database solution.

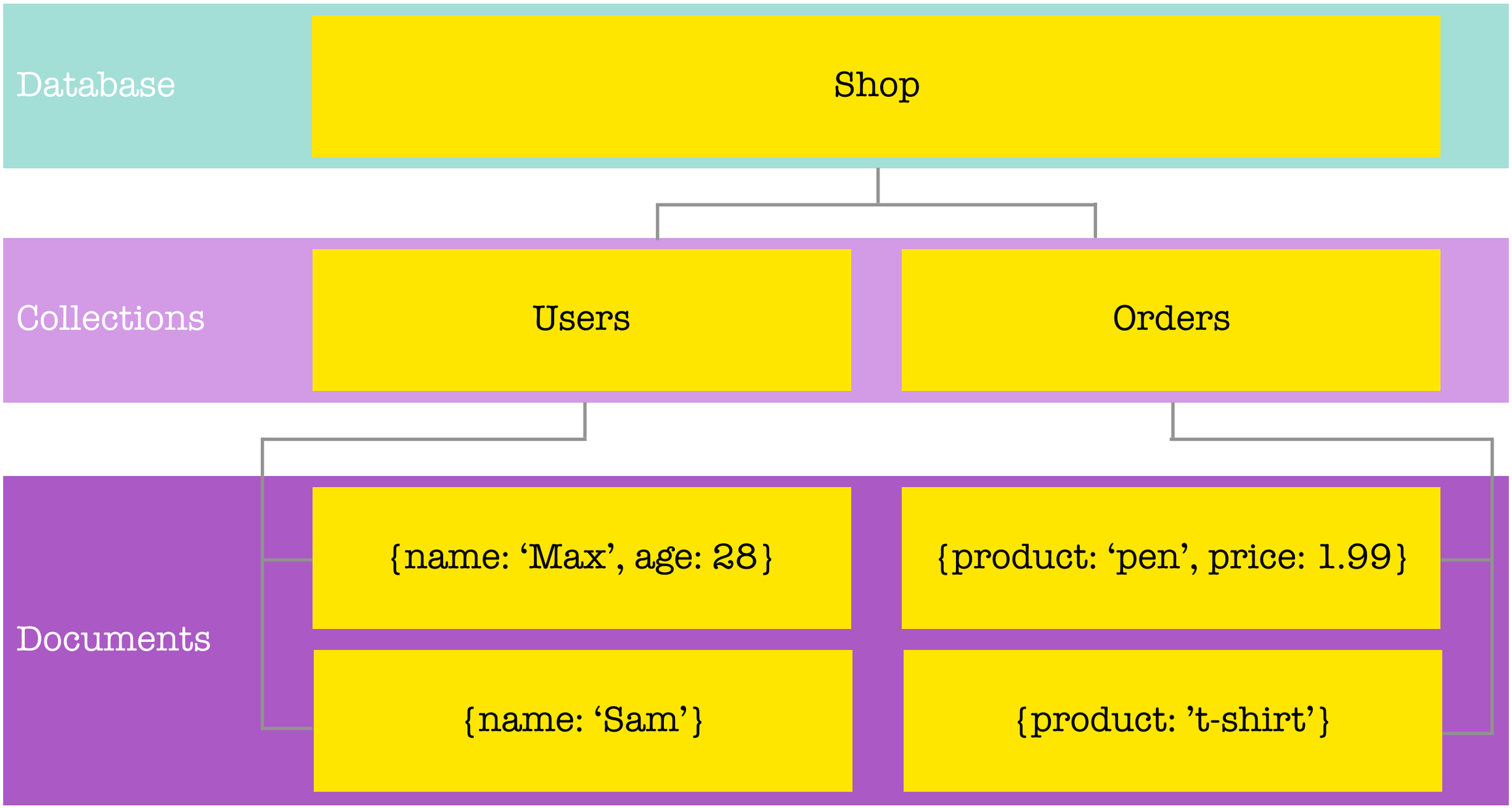
There are many database solutions such as mySQL, PostgresSQL, TSQL etc.

MongoDB is most importantly a database server that allows us to run different databases on it for example a Shop database. Within the database we would have different collections such as a Users collection or a Orders collection. We can have multiple databases and multiple collections per database.

Inside of the collection we have something called documents. Documents look like JavaScript JSON objects. Inside of a collection the documents are schema-less and can contain different data. This is the flexibility that MongoDB provides us with whereas SQL based database are very strict about the data stored within the database tables. Therefore, the MongoDB database can grow with the application needs. MongoDB is a No-SQL database.

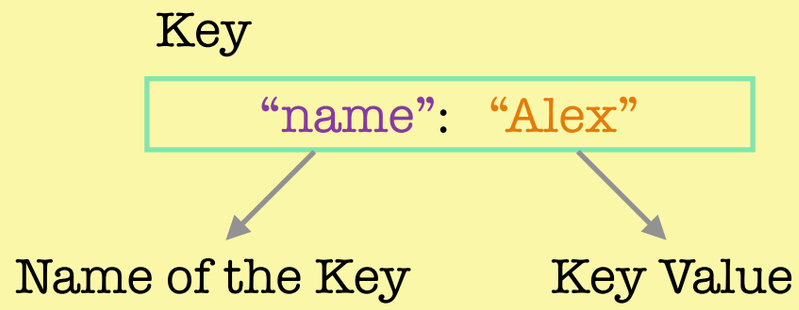
Typically we will need some kind of structure in a collection because applications typically requires some type of structure to work with the data.

Diagram 1.1:



JSON (BSON) Data Format:

```
{
    “name”: “Alex”,
    “age”: 29,
    “address”: {
        “city”: “Munich”
    },
    “hobbies”: [
        { “name”: “Cooking” },
        { “name”: “Football” }
    ]
}
```



The above is an example of the JSON data format. A single document is surrounded by curly brackets. The data is normally structured with a Keys. Keys consist of a Name of the Key and a Key value. The Name of the Key (*which will be referred to as Key from now on*) and the Key Value must be wrapped around quotation marks (unless if the data is a type of number).

There are different types of values we can store such as: string, number, booleans and arrays.

We can also nest documents within documents. This allows us to create complex relations between

data and store them within one document, which makes working with the data and fetching data more efficient because it is contained in one document in a logical way. SQL in contrast requires more complex method of fetching data which require joins to find data in table A and data in table B to retrieve the relevant data.

Behind the scenes on the server, MongoDB converts the JSON data to a binary version of the data which can be stored and queried more efficiently. We do not need to concern ourselves with BSON as we would tend to work with JSON data.

The whole theme of MongoDB is flexibility, optimisation and usability and it is what really sets MongoDB apart from other database solutions because it is so efficient from a performance perspective as we can query data in the format we need it instead of running complex restructuring on the server.

The Key MongoDB Characteristics.

MongoDB is a no SQL solution because it is following an opposite concept/philosophy to SQL based databases. Instead of normalising the data i.e. storing data distributed across multiple tables where every table has a clear schema and then using relations, MongoDB goes for storing data together in a document. It does not force a schema hence schema-less/No-SQL.

We can have multiple documents in a single collection and they can have different structures as we have seen in Diagram 1.1. This is important, it can lead to messy data but it still our responsibility as developers to work with clean data and to implement a solution that works. On the other hand this provides us with a lot of flexibility. We could use mongoDB for applications that might still evolve, where the exact data requirements are not set yet. MongoDB allows us to started and we could always add data with more information in the same collection at a later point in time.

We also work with less relations. There are some relations, but with these embedded (nested) documents, we have less collections (tables) which we connect but instead we store data together. This is where the efficiencies is derived from, since data is stored together and when we fetch data from our application it does not require to reach out to multiple tables and merge the data because all the data is already within the single collection. This is where the speed, performance and flexibility comes from and can be seen beneficial for when building applications. This is the main reason why No-SQL solutions are so popular for read and write heavy applications.