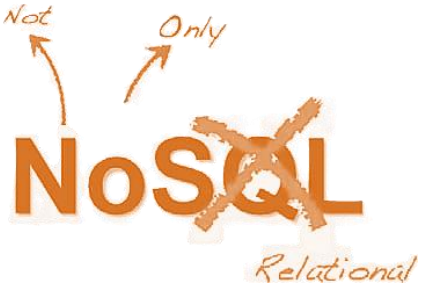


Document DB Intro

sohailimran2@yahoo.com

Ref: Web

Intro

A *document store database* (also known as a *document-oriented database*, *aggregate database*, or *simply document store* or *document database*) is a database that uses a document-oriented model to store data.

Built around **JSON-like** documents, document databases are both natural and flexible for developers to work with.

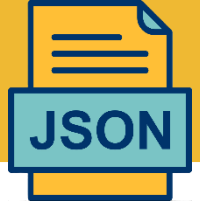
Document store databases store each record and its associated data within a single *document*.

Each document contains semi-structured data that can be queried against using various query and analytics tools of the DBMS.

With relational databases, you must create a schema before you load any data.
With document store databases (and most other NoSQL databases), you have no such requirement.
You can just go ahead and load the data without any predefined schema.

mongoDB

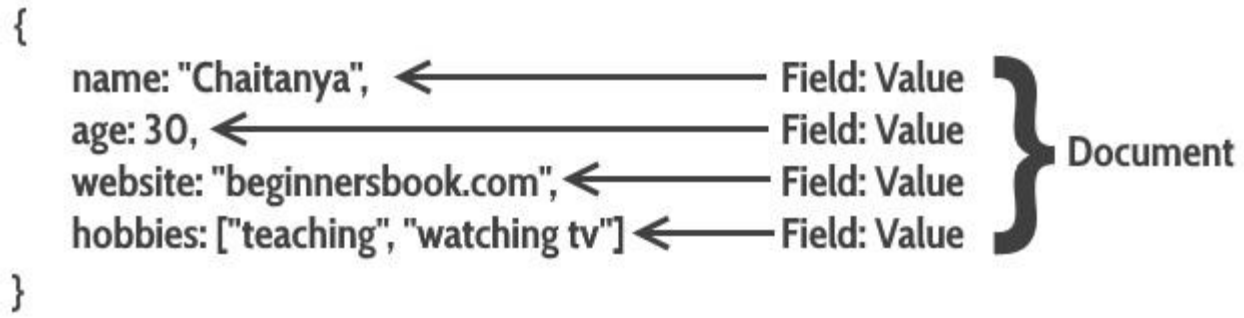
NoSQL



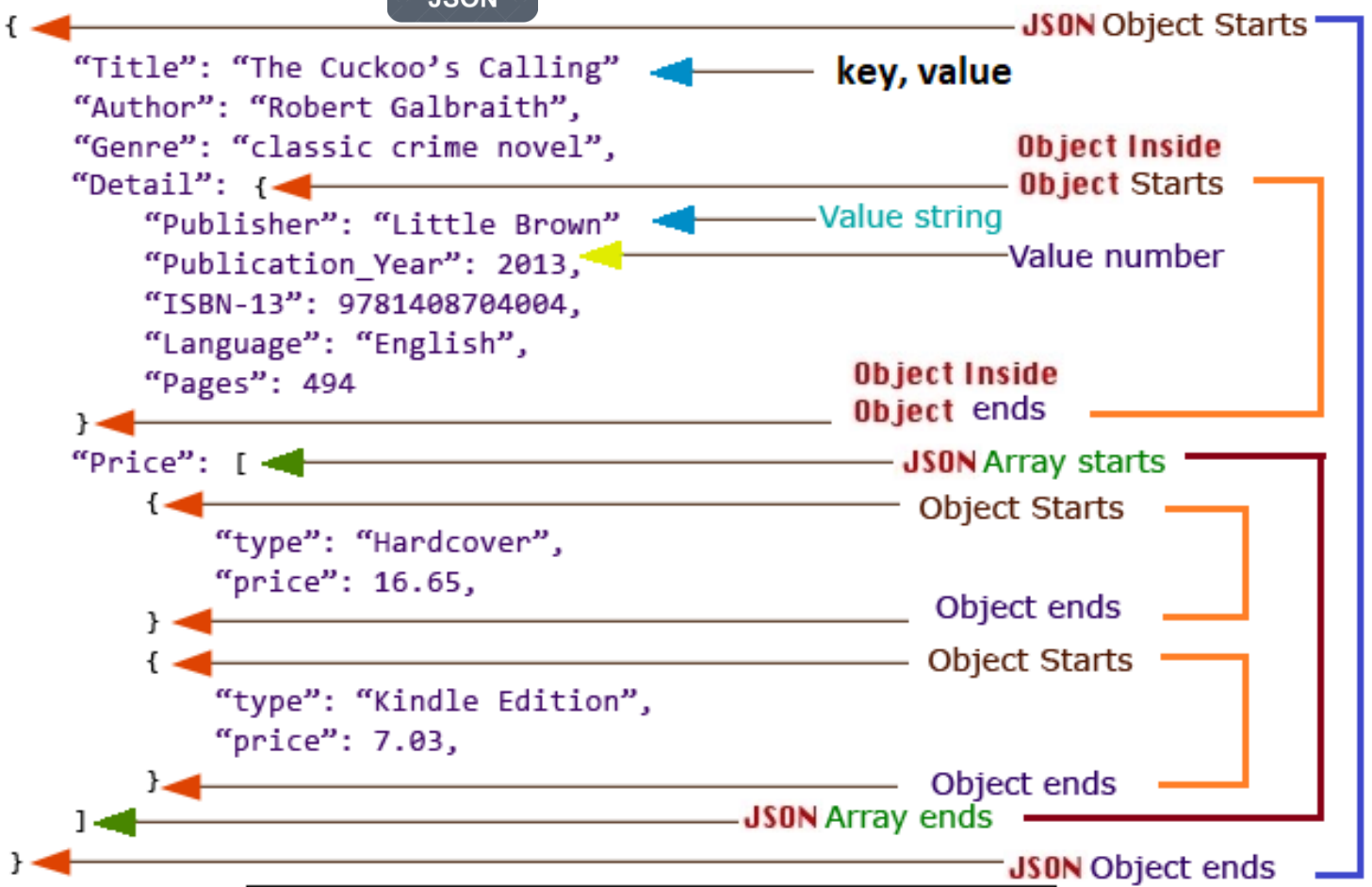
{JSON} JavaScript Object Notation

It is used to store information in an organized, and easy-to-access manner.
It offers a human-readable collection of data which can be accessed logically.
It is lightweight data-interchange format.
It is easy to read and write than XML.
It is language independent.
It supports array, object, string, number and values.

Its filename extension for written programming code is .json. The Internet Media type for JSON is application/json and the public.json is its Uniform Type Identifier. The file name extension is .json.



JSON structure



It can contain following:

* Array Inside Array
* "contacts": null ← Null Value

Vs


```xml
<artist>
  <artistname>Iron Maiden</artistname>
  <albums>
    <album>
      <albumname>The Book of Souls</albumname>
      <datereleased>2015</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Killers</albumname>
      <datereleased>1981</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Powerslave</albumname>
      <datereleased>1984</datereleased>
      <genre>Hard Rock</genre>
    </album>
    <album>
      <albumname>Somewhere in Time</albumname>
      <datereleased>1986</datereleased>
      <genre>Hard Rock</genre>
    </album>
  </albums>
</artist>
```

```json
{
    '_id' : 1,
    'artistName' : { 'Iron Maiden' },
    'albums' : [
        {
            'albumname' : 'The Book of Souls',
            'datereleased' : 2015,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Killers',
            'datereleased' : 1981,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Powerslave',
            'datereleased' : 1984,
            'genre' : 'Hard Rock'
        }, {
            'albumname' : 'Somewhere in Time',
            'datereleased' : 1986,
            'genre' : 'Hard Rock'
        }
    ]
}
```

Note: _id field, a unique ID, is added in the json example.

mongoDB

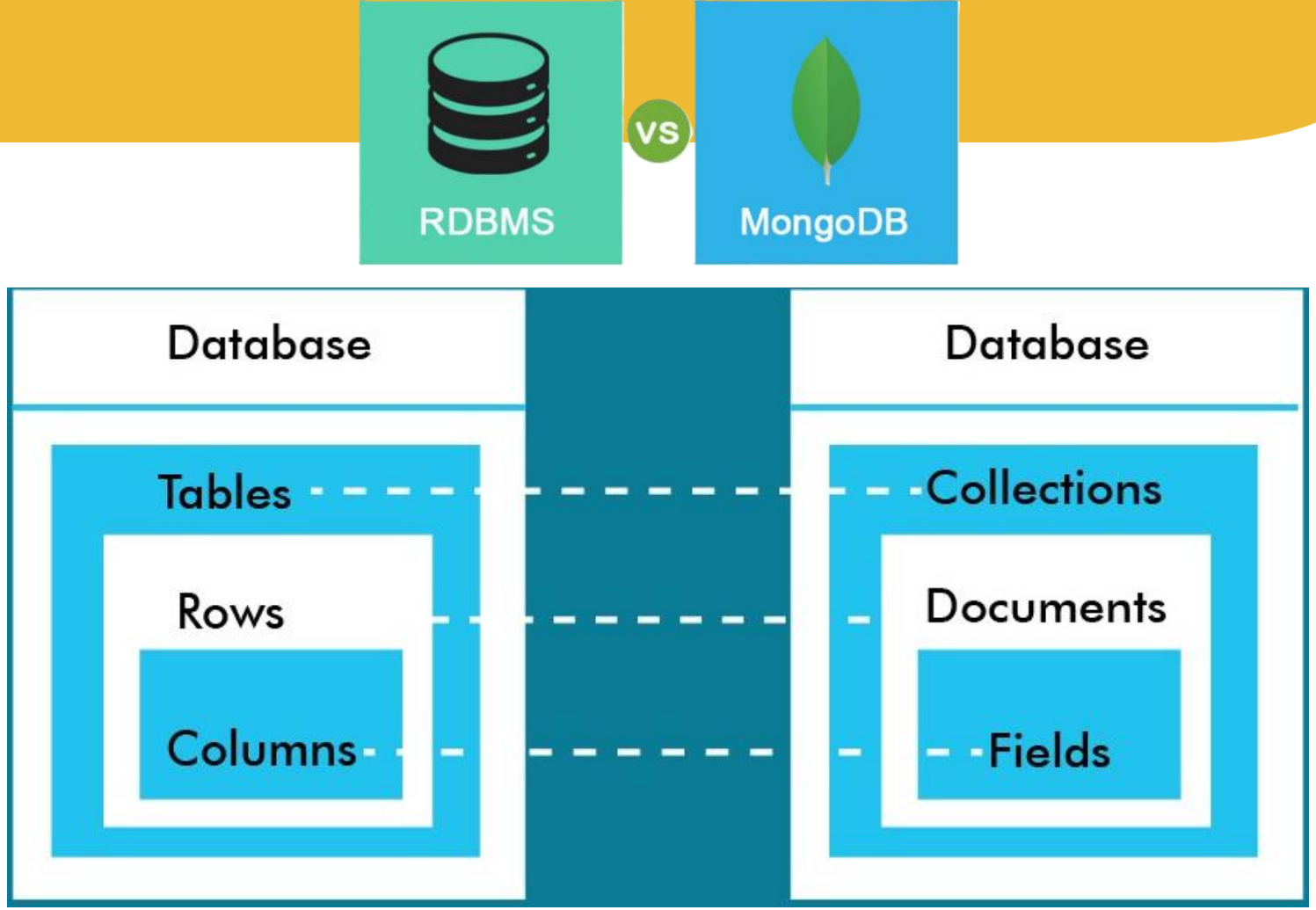
NoSQL

MongoDB is a document-oriented NoSQL database used for high volume data storage.

Features

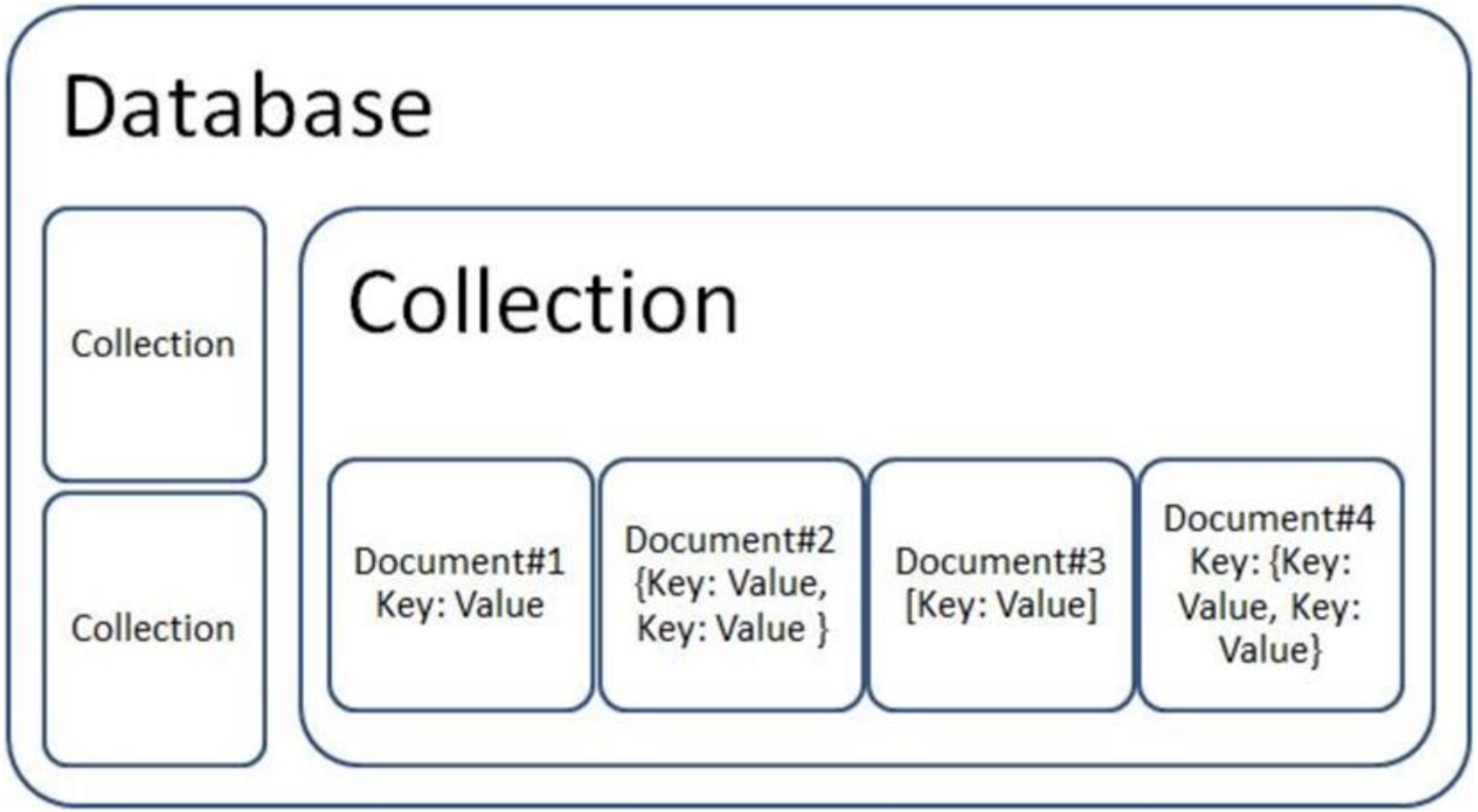
- Each database contains **collections** which in turn contains **documents**. Each document can be different with a varying number of **fields**. The size and content of each document can be different from each other.
- The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with **key-value pairs**.
- The rows (or documents as called in MongoDB) **doesn't need** to have a **schema** defined beforehand. Instead, the fields can be created on the fly.
- The data model available within MongoDB allows one to represent **hierarchical** relationships, to store arrays, and other more complex structures more easily.

Scalability – The MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database



Fields (**key: value** pairs) are stored in a **Document**, documents are stored in a **Collection,** and collections are stored in a **Database**.

collection


Database
Collection
Collection
Collection
Document#1
Key: Value
Document#2
{Key: Value,
Key: Value }
Document#3
[Key: Value]
Document#4
Key: {Key:
Value, Key:
Value}

key components

_id – This is a (12-byte) field required in every MongoDB document. The _id field represents a **unique** value in the MongoDB document. The _id field is like the document's primary key. If you create a new document without an _id field, MongoDB will automatically create the field. So for example, if we see the example of the above customer table, Mongo DB will add a 24 digit unique identifier to each document in the collection.

_Id	CustomerID	CustomerName	OrderID
563479cc8a8a4246bd27d784	11	Guru99	111
563479cc7a8a4246bd47d784	22	Trevor Smith	222
563479cc9a8a4246bd57d784	33	Nicole	333

Collection – This is a grouping of MongoDB documents. A collection is the equivalent of a table which is created in any other RDMS such as Oracle or MS SQL. A collection exists within a single database. As seen from the introduction collections don't enforce any sort of structure.

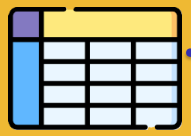
Cursor – This is a pointer to the **result set of a query**. Clients can iterate through a cursor to retrieve results.

Database – This is a **container** for collections like in RDMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.

Document - A **record** in a MongoDB collection is basically called a document. The document, in turn, will consist of field name and values.

Field - A **Name:Value pair** in a document. A document has zero or more fields.
Fields are analogous to columns in relational databases.
The diagram shows an example of Fields with Key:Value pairs. So, in the example **CustomerID:11** is one of the Key:Value pair's defined in the document.

table Vs collection



Relational Database

Student_Id	Student_Name	Age	College
1001	Chaitanya	30	Beginnersbook
1002	Steve	29	Beginnersbook
1003	Negan	28	Beginnersbook

MongoDB

```
{
 "_id": ObjectId("....."),
 "Student_Id": 1001,
 "Student_Name": "Chaitanya",
 "Age": 30,
 "College": "Beginnersbook"
}
{
 "_id": ObjectId("....."),
 "Student_Id": 1002,
 "Student_Name": "Steve",
 "Age": 29,
 "College": "Beginnersbook"
}
{
 "_id": ObjectId("....."),
 "Student_Id": 1003,
 "Student_Name": "Negan",
 "Age": 28,
 "College": "Beginnersbook"
}
```

NoSQL