Data Model in MongoDB

$lookup (aggregation): it adds an array of related data from the other document. It perform an equality match between a field from the input documents with a field from the documents of the "joined" collection.

```
>db.collectionname.aggregate({
    $lookup:
      {
        from: <collection to join>,
        localField: <field from the input documents>,
        foreignField: <field from the documents of the "from" collection>,
        as: <output array field>
      }
  });
```

Data Model in MongoDB

```
MongoDB Enterprise > db.emp3.findOne();
{
        "_id" : ObjectId("60911b086c4df9fcbff4474a"),
        "EID" : 1001,
        "Lname" : "Gupta",
        "Fname" : "Ramesh",
        "ADDRESS" : "SECTOR 7,Rohini,Gurgaon",
        "City" : "Gurgaon",
        "PHONE" : NumberLong("9999002727"),
        "EMAIL" : "RK@YAHOO.CO.IN",
        "DOB" : "9/1/1990",
        "DOJ" : "3/15/2012"
}
MongoDB Enterprise > db.salary.findOne();
{
        "_id" : ObjectId("608a7474a538f278ddd80e5a"),
        "EID" : 1004,
        "DEPT" : "MIS",
        "DESI" : "Manager",
        "SALARY" : 134789
}
MongoDB Enterprise >
```

Data Model in MongoDB

```
> db.emp3.aggregate([
        {$lookup:
        {from: "salary" ,
        localField: "EID",
        foreignField: "EID",
        as: "SalDetails" }}
        ]);
```

```
{
        "_id" : ObjectId("60911b086c4df9fcbff4474a"),
        "EID" : 1001,
        "Lname" : "Gupta",
        "Fname" : "Ramesh",
        "ADDRESS" : "SECTOR 7,Rohini,Gurgaon",
        "City" : "Gurgaon",
        "PHONE" : NumberLong("9999002727"),
        "EMAIL" : "RK@YAHOO.CO.IN",
        "DOB" : "9/1/1990",
        "DOJ" : "3/15/2012",
        "SalDetails" : [
                {
                        "_id" : ObjectId("608a7474a538f278ddd80e5b"),
                        "EID" : 1001,
                        "DEPT" : "OPS",
                        "DESI" : "Director",
                        "SALARY" : 380000
                }
        ]
}
```

Relationship in MongoDB

Relationships in MongoDB represent how various documents are logically related to each other. Relationships can be modeled via Embedded and Referenced approaches. Such relationships can be either

One – to - One (1:1),

```
{
        "_id" : ObjectId("60934813c75b92ae37219fda"),
        "cid" : "c0002",
        "name" : "Gaurav",
        "phone" : "9999009890",
        "addressid" : ObjectId("609004da2beb855ec8035465")
}
```

One – to – Many(1:N),

Many – to- One (N:1)

Many – to- Many (N:N)

```
MongoDB Enterprise > db.client.find().pretty();
{
        "_id" : ObjectId("6093472ac75b92ae37219fd9"),
        "cid" : "c0001",
        "name" : "jainy",
        "phone" : "9899090987",
        "addressid" : [
                ObjectId("609004da2beb855ec8035463"),
                ObjectId("609004da2beb855ec8035464"),
                ObjectId("609004da2beb855ec8035465")
        ]
}
```

MongoDb Drivers

Drivers are the packages we install for different programming languages in which the application might be written.

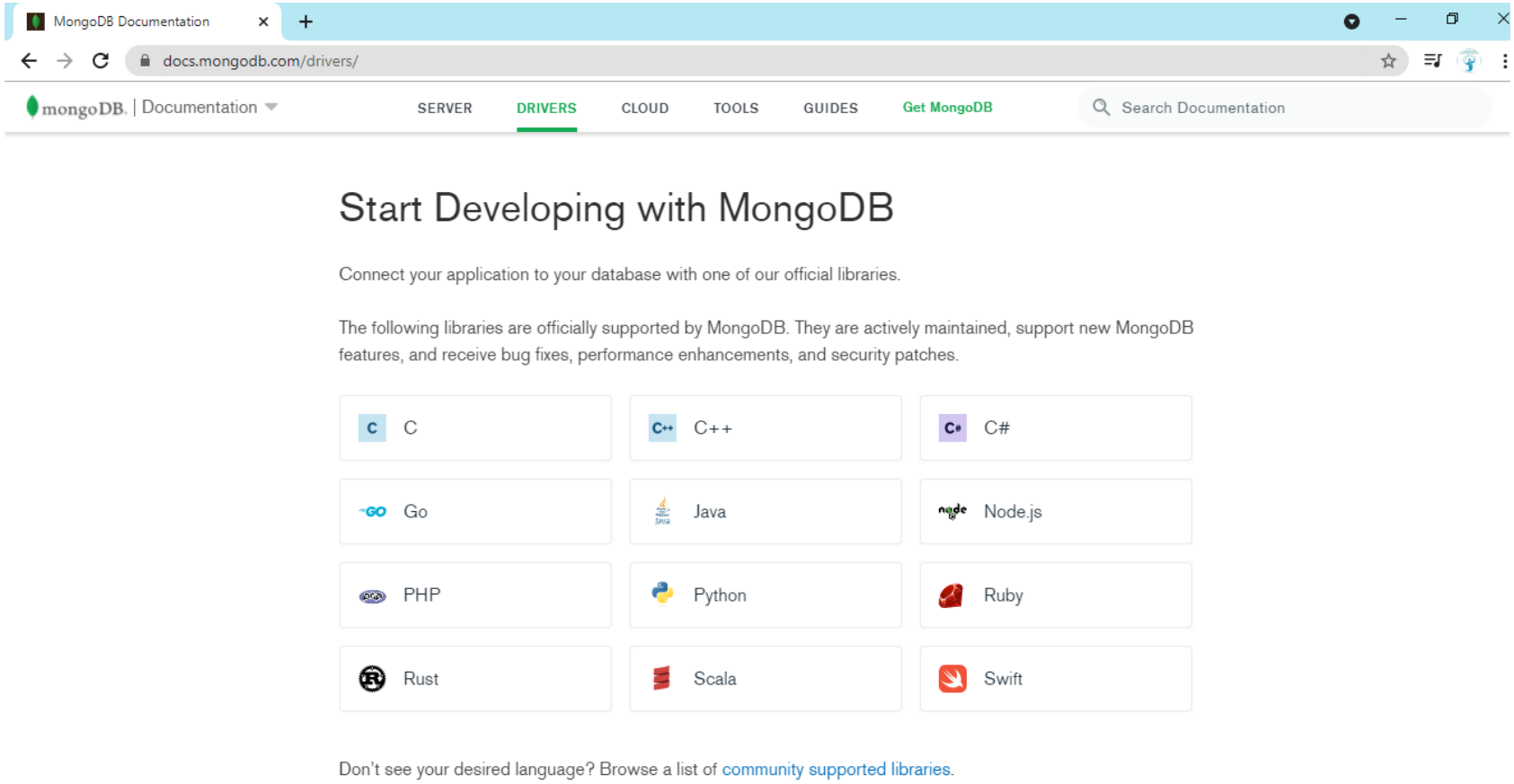
These can be downloaded for MongoDb official page under the doc tab:

https://www.mongodb.com/try/download/community

These are the bridge between the database & the application

MongoDb Drivers

Choose the appropriate driver as per your application



- Create a document to track book details and author details should be added as a reference document.
- Retrieve the book title ,publisher & and author name using $lookup aggregation.
- Display the aggregated data of employee and salary details.

Data Types in MongoDB

In MongoDB data is data representation is done in JSON (JavaScript Object Notation) document format which is binary encoded and is termed as BSON. MongoDB supports many data types. Such as :

Integer – This type is used to store a numerical value.

```
> db.testdt.insert({"integer" :125});
```

Boolean – This type is used to store a boolean (true/ false) value.

```
> db.testdt.insert({"registered" :true});
```

Double – This type is used to store floating point values.

```
> db.testdt.insert({"amount" : 3745.95});
```

String – This is the most commonly used datatype to store the data.

```
> db.testdt.insert({“greeting" : “Welcome to MongoDB”});
```

Arrays – This type is used to store arrays or list or multiple values into one key.

```
> var courses = ["SQL" ,"PBI", "MongoDB"]
> db.testdt.insert({"module" : courses});
```

Data Typesin MongoDB

Object – This datatype is used for embedded documents.

```
> var hrs = {"SQL" : 25, "PowerBi" :20 , "MongoDB" : 15};
> db.testdt.insert({"Duration" : hrs});
```

Null – This type is used to store a Null value.

```
> db.testdt.insert({"email" : null});
```

Date – This datatype is used to store the date.

```
> var d1 = Date();
> var d2 = ISODate();
> var JD = ISODate("2021-05-01");
> db.testdt.insert({"Stringdate" : d1, "ISODate" : d2 , "JoiningDate": JD});
```

Timestamp – Timestamp stores 64-bit value. This can be handy for recording when a document has been modified or added.

```
>var v1= new Timestamp();
> db.testdt.insert({"login": v1 });
```

Data Types in MongoDB

Object ID – This datatype is used to store the document's ID.

```
> var v1 = ObjectId("6093b6d573c5517d62544e4e");
> db.testdt.insert({"refdocid" : v1});
```

MongoDb also allows us to query the data on the basis of data type.

$type selects documents where the value of the field is an instance of the specified type. Querying by data type is useful when dealing with highly unstructured data where data types are not predictable.

```
{ field: { $type: <BSON type> } }
```

```
> db.testdt.find({"JoiningDate" : {$type : "date"}})
> db.testdt.find({"JoiningDate" : {$type : "string"}})
```

MongoDB GridFS

GridFS is a frame work to store & access large set of data. It divides the data into chunks and store then into different documents.

-API Provided by MongoDb for storing large files such as audio, video and images.

-Package that can be plucked into any application to make storing large files easier

Provides a way for storing large files in database instead of in the file system.

MongoDB GridFS

Problem: In MongoDB document size is limited to 16 MB.

Gridfs Solves the size limitation problem

1. Breaks the files to smaller managable chunks
2. Stores these chunks of data in one collection called **fs.chunks**
3. Stores the information about the whole file itself in another collection called **fs.files**
4. Connects these documents by properties that are references to each other

MongoDB GridFS

fs.chunks collection

1. The size of each chunk is 255KB
2. No. of chunks created depends on the file size
3. Chunks stores the actual data.
4. Each chunk is linked to the fille information by “files_id” property.
5. The “files_id” points to a document that is stored in fs.files collection.

```
MongoDB Enterprise > db.fs.chunks.find({},{data:0}).pretty();
{
        "_id" : ObjectId("6093e3130ee4d7e7115c5ea2"),
        "files_id" : ObjectId("6093e3120ee4d7e7115c5ea1"),
        "n" : 0
}
{
        "_id" : ObjectId("6093e70fe8a48c9a7a484db0"),
        "files_id" : ObjectId("6093e70fe8a48c9a7a484daf"),
        "n" : 0
}
{
        "_id" : ObjectId("6093e70fe8a48c9a7a484db1"),
        "files_id" : ObjectId("6093e70fe8a48c9a7a484daf"),
        "n" : 1
}
{
        "_id" : ObjectId("6093e70fe8a48c9a7a484db2"),
        "files_id" : ObjectId("6093e70fe8a48c9a7a484daf"),
        "n" : 2
}
```

MongoDB GridFS

fs.files collection contains the information about the file

1. File name
2. Average size of each chunk
3. Upload date
4. Size of file (in bytes)
5. File metadata

```
MongoDB Enterprise > db.fs.files.find().pretty();
{
        "_id" : ObjectId("6093e3120ee4d7e7115c5ea1"),
        "length" : NumberLong(130797),
        "chunkSize" : 261120,
        "uploadDate" : ISODate("2021-05-06T12:37:39.852Z"),
        "filename" : "boy.jpg",
        "metadata" : {

        }
}
{
        "_id" : ObjectId("6093e70fe8a48c9a7a484daf"),
        "length" : NumberLong(1679701),
        "chunkSize" : 261120,
        "uploadDate" : ISODate("2021-05-06T12:54:39.149Z"),
        "filename" : "me.jpg",
        "metadata" : {

        }
}
```

MongoDB GridFS

The **mongofiles** utility makes it possible to manipulate files stored in your MongoDB instance in GridFS objects from the command line.

The mongofiles tool is part of the MongoDB Database Tools package.

https://www.mongodb.com/try/download/database-tools

Run mongofiles from the system command line, not the mongo shell.

MongoDB GridFS

mongofiles <options> <connection-string> <command> <filename or _id>

Options. You may use one or more of these options to control the behaviour of mongofiles.

Connection String. The connection string of the mongod to connect to with mongofiles.

Command - Use one of these commands to determine the action of mongofiles.

Filename - name of the file to be saved in the data base

C:\....\bin>mongofiles put me.jpg --db=img

C:\....\bin>mongofiles --help

C:\....\bin>mongofiles get me.jpg --db=img

db.fs.files.find().pretty();

MongoDB GridFS

Once the file is stored in the database fs.files & fs.chunks collection can be used to get the information about the file

>db.fs.files.find().pretty();

```
MongoDB Enterprise > db.fs.files.find().pretty();
{
        "_id" : ObjectId("6093e3120ee4d7e7115c5ea1"),
        "length" : NumberLong(130797),
        "chunkSize" : 261120,
        "uploadDate" : ISODate("2021-05-06T12:37:39.852Z"),
        "filename" : "boy.jpg",
        "metadata" : {

        }
}
```

>db.fs.chunks.find().pretty();

```
i87q0+Ko16/Jqbz38dvHoOv8AX6rfp/4j2ptfjHDj8+mH8Twzr4U6CmPx6JfFp
enSuGmyW0a9Da9Vr6Lc2/wCI9lcOrH+o/n0/c6s14/8AFdKvC+TVF9tfR4+baN
G2hfH+jQmrVr16tHp/V+Le/d9e7rT+HpGjjX/L/AMXx6f0t4V8enTpN9F/9jo/
69Po0+PVza99er/affu2vdx/ydVTRQ+FWmf9nrkL6PVf8ARzb9V+PvdWNWNW
/ACbp/wB6/wB99PbbvbTpNjNPl/sdf/2Q==")
}
MongoDB Enterprise > db.fs.chunks.find({},{data:0}).pretty();
{
        "_id" : ObjectId("6093e3130ee4d7e7115c5ea2"),
        "files_id" : ObjectId("6093e3120ee4d7e7115c5ea1"),
        "n" : 0
}
```