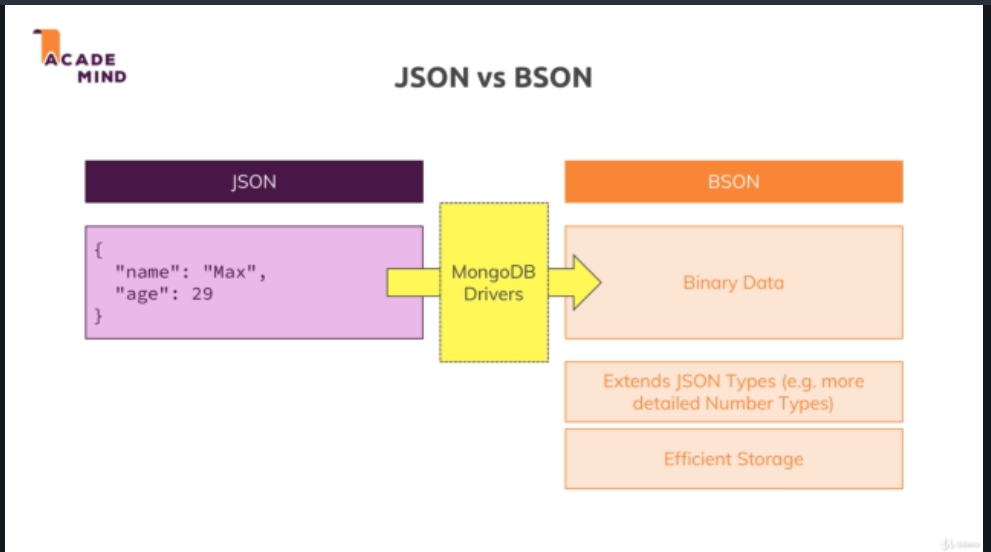
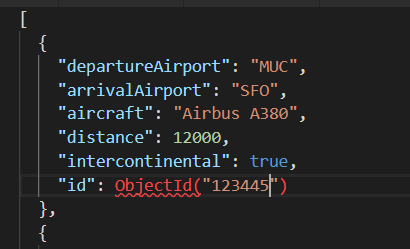
17)Comparing bson and json



This conversion is done by mongoDb drivers.

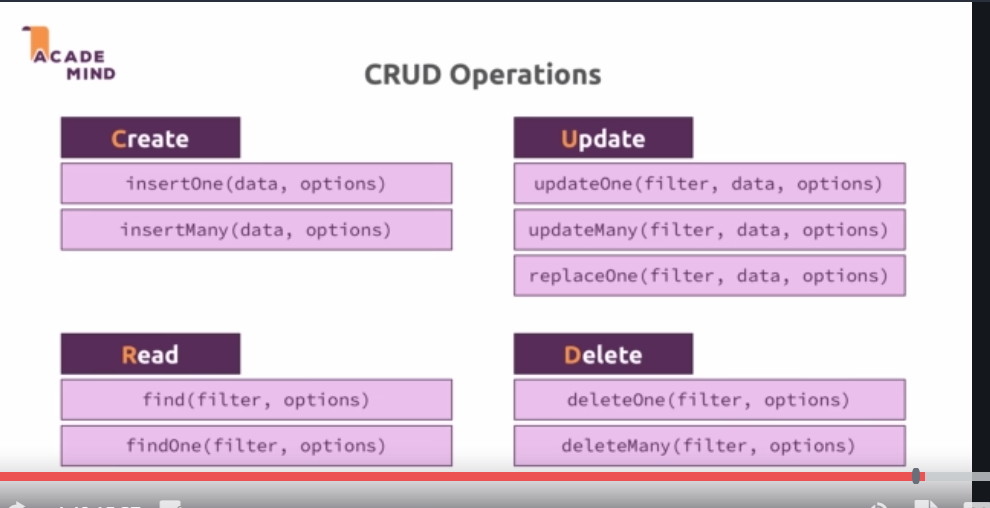
This is done because binay data is more efficient to store than json data. so it is faster and more efficient from space, size perspective and additionally it supports additional types. It slao supports additional types, the ObjectId thatw e saw in last lecture is not valid json type.

This shows error in json file-

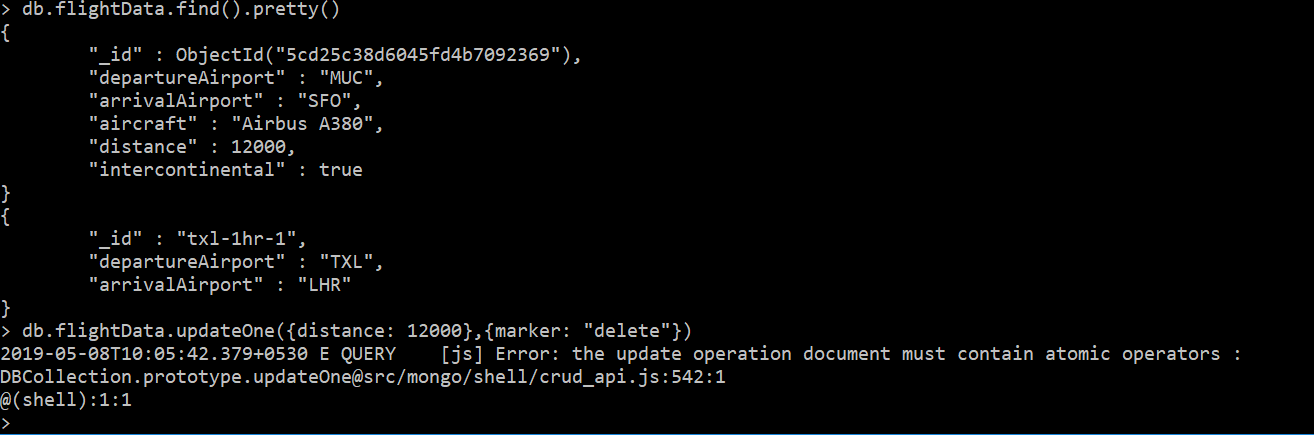


It is not valid json josn but mongo db cn understand it and store it in its binary data. you got other types too like different types of numbers with decimals and very big numbers, these are stored in different ways behind the scenes, you dnt need to care too much about that.

18)Create,Read.Update,Delete(CRUD) & MongoDb



this is our data, we tried to run a update query. We got this error-



**It says update operation must contain atomic operators.**

The thing is it simply is an incorrect type of data for updating. You dnt pass a document describing your change like this, bcoz mongo d does’nt know how to interpret this. Instead what you pass here is a document with curly braces with a special keyword, $set and this is something you can remember, whenever you see something with $ in mongodb, it’s reserve operator or word. $set is simply identified by mongodb when used in the updateOne operation to describe the change that you want to make. The value of @set is another object. it says mongo db set this value of doc you found. If this property does not exist create new.

So correct query is-

**> db.flightData.updateOne({distance: 12000},{$set: {marker: "delete"}})**

22)update vs updateMany

Both update and updateMany are used to update multiple documents. But if we run this-

**db.flightData.update({ "\_id" : ObjectId("5cd2644bd6045fd4b709236a")},{$set: {delayed: true}})**

then no matter what our document is, final document will be-

**{ "\_id" : ObjectId("5cd2644bd6045fd4b709236a"), "delayed" : false }**

So instead of updating a field, it update whole object. but if we use $set, then bith update and updateAll work in same way.

With **updateOne** and **updateAll** we have to use atomic operator, as we sae before.

To avoid confusion use updateOne and updateMany. If you want to replace something, then use **replaceOne({}, {})**

First object is filter, second one is replaced object.

23)Unerstanding the “find()” & the Cursor Object

Find methods returns us the cursor object, not the original data. this is because if it sent data, it may have to send millions of object, then it would take super long. The cursor object has lot of metadata befind it, which allow us to cycle through the results, that is what “it” command did.

There are some methods on cursor, like toArray(), it will show you all documents. Then we saw that in js we how we can get all elemenst.

**db.passengers.find().forEach(passengerData => printjson(passengerData));**

forEach metod allows you to do write some code to do something on every element that is in your database, on every document you got in there.

Exact syntax, is something you can find in driver docs.

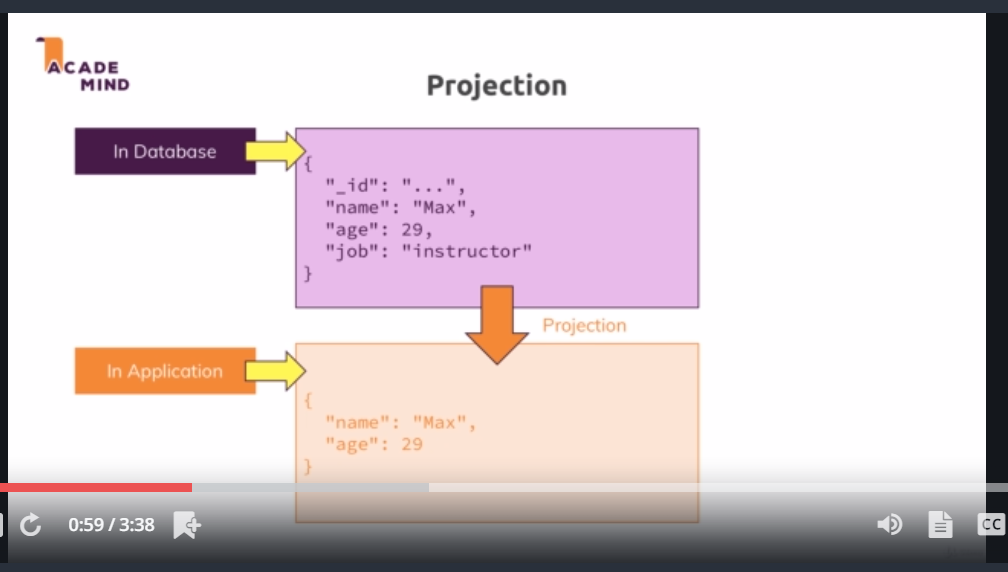
since shell is js based, this command worked in shell.

There are more functions that you can find in docs. In forEach it fetches document for only that loop cycle, so it is very efficient. It loads data on demand.

So find() returns cursor. That is reason you cannot use pretty method on findOne. Becase pretty method exist on cursor.

Shell takes the cursor and gives us first 20 documents by default. For other methods like insert,update and delete, cursor do not exist because these methids don’t fetch the data.

24)understanding projection



25)Embedded Documents &Arrays- The theory

