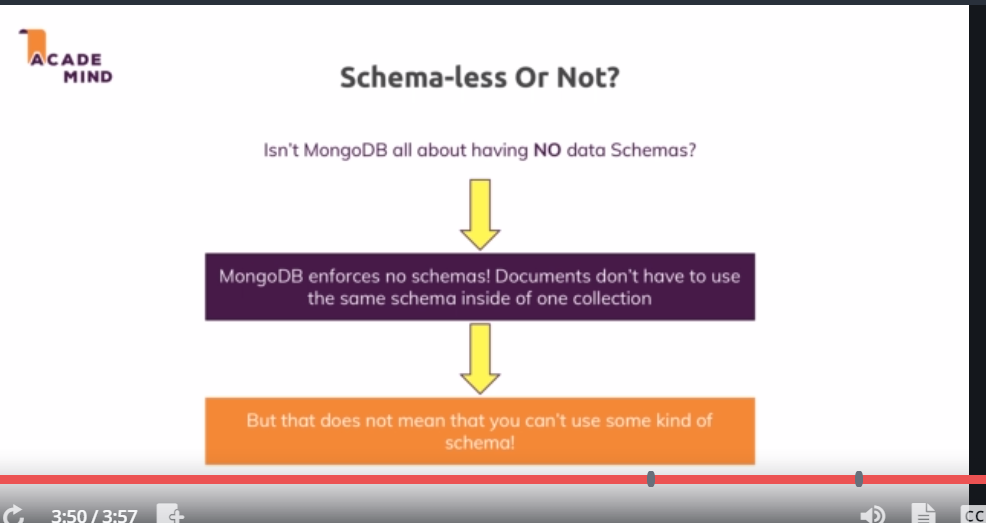
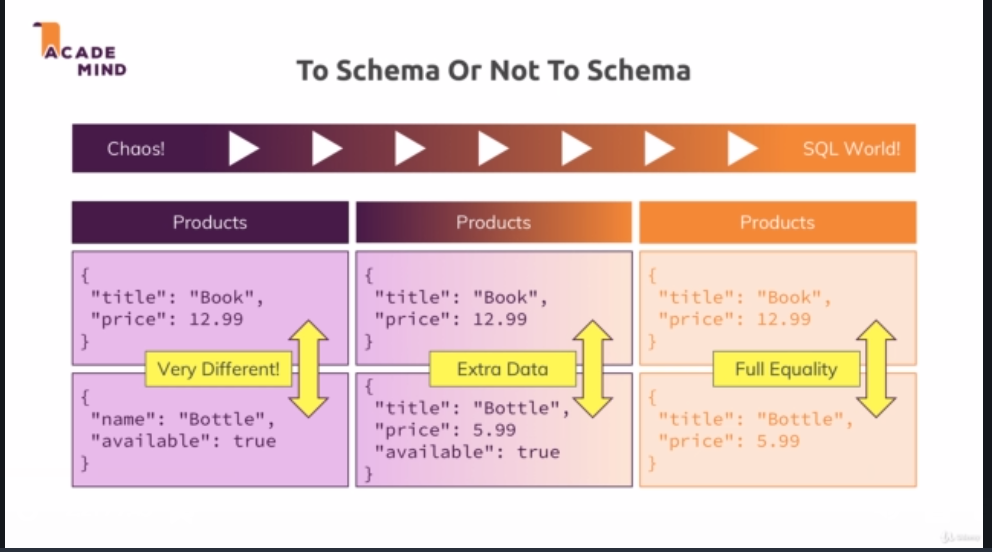
33)why do we need schemas



34)Structuring documents



In real word, extreme right and middle approach is used. Here we got the best of 2 words.

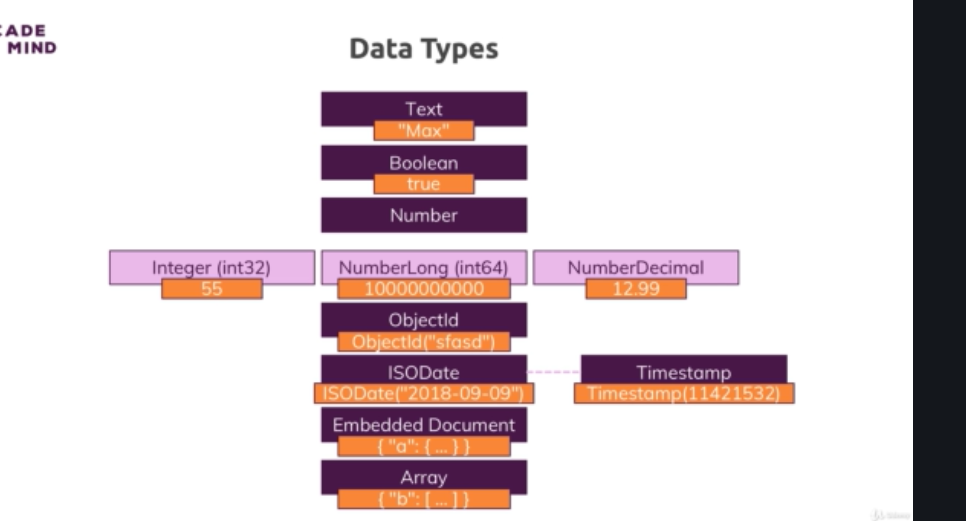
Now lets say we are going with second approach. there also we are using second approaches, so we can have extra fields is some documents. In docs in which we don’t have that extra fields, we can either will not have that property or we can have that property but value will be null. First approach is mongoish, second one is more sqlish.

36)data Types in action

Test size – 16 mega bytes for whole document.

Integars are 32 bits long, and therefore if you try to store values that are longer than that, they will overflow that range and end up with different numbers.

For larger number we have different solutions.

Any number we enter in console is treated as float and stored as 64 bits. This is bcoz console is js based and it does not differentiate between integars and floating point values. So every number in console will be stored as 64 bit float(or double) type in shell not as numberDecimal high precision type.

36)Data types in action

Lets srn this query-

**db.companies.insertOne({name: "Fresh Apples Inc", isStartup: true, employees: 33, funding: 12345678901234567890, details: {ceo: "Mark Super"}, tags: [{title: "super"},{title: "perfect"}],foundingDate: new Date(), insertedAt: new Timestamp**

Here we used functions provided by mongo db shell to inserted date and time stamp. You can find equivalents of these in documentation of drivers for different programming languages. Go to driver docs, then refrence,then api documentation, there you can find timestamp (in case of nodejs driver). So this is how you can use the types that you learned in shell in your driver too.

Lets search our document, we get this-

**> db.companies.find().pretty()**

**{**

**"\_id" : ObjectId("5cd3c0921d423d9b1a93e5a0"),**

**"name" : "Fresh Apples Inc",**

**"isStartup" : true,**

**"employees" : 33,**

**"funding" : 12345678901234567000,**

**"details" : {**

**"ceo" : "Mark Super"**

**},**

**"tags" : [**

**{**

**"title" : "super"**

**},**

**{**

**"title" : "perfect"**

**}**

**],**

**"foundingDate" : ISODate("2019-05-09T05:54:26.382Z"),**

**"insertedAt" : Timestamp(1557381266, 1)**

**}**

You can see date data type, it is date(yyyyMMDD) then timestamp(in 24 hour format) at last we have timezone, date and time are separated by T. then we have timestamp(inserAt property). It is based on current time in milliseconds, there is also has an ordinal component, so that if we insert 2 documents at same time, it can differentiate between them.

But funding value is changed. After some digits , the number is nullify or cutoff , you could say. The reason for that is I tried to store a number that was too big, bcoz a normal number that js accepts is 64 bit floating point value and that could not be stored here. that is restriction with numbers that you can face. To store large numbers use strings.

Lets store numbers in 2 ways-

**db.numbers.insertOne({a:1})**

**db.numbers.insertOne({a: NumberInt(1)})**

so in second case, size of data will be less. We checked it with **db.stats()** function.

this is one of other reasons you should use these types, to save memory.

this is because in second case, type is numberInt(size 32), while in first case it is float or double. Shell is based on js and js cnt differentiate between int and float, that is why 7 is stored as float. In other languages this is not the case, if you access db through driver and added a number there.