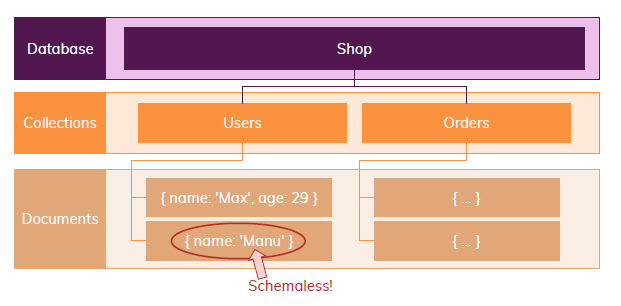
Mongodb



**JSON (BSON) Data Format**

{

"name": "Max",

"age": 29,

"address":

{

"city": "Munich"

},

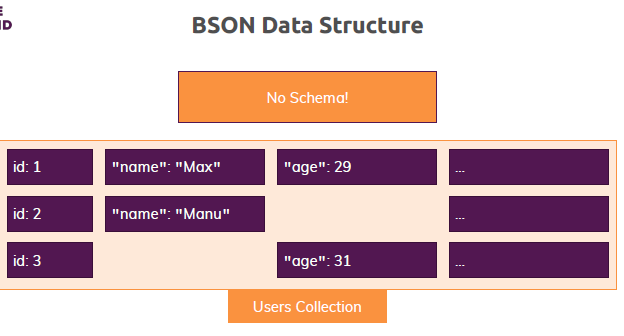
"hobbies": [

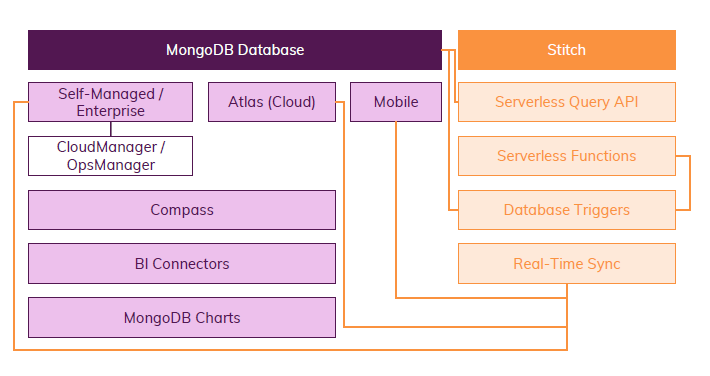
{ "name": "Cooking" },

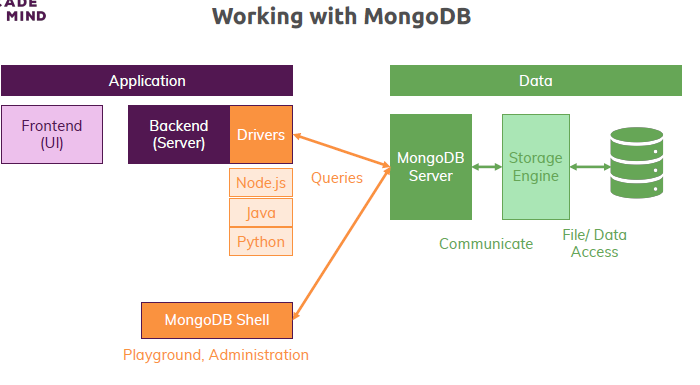
{ "name": "Sports" }

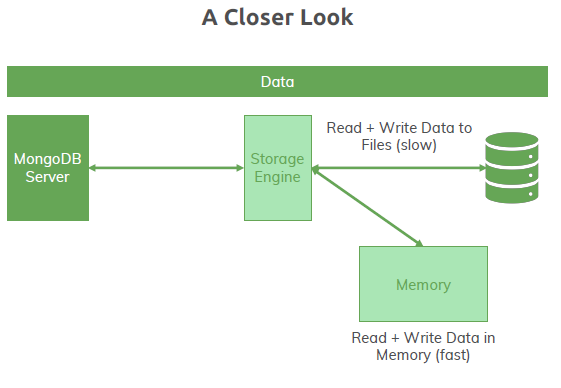
]

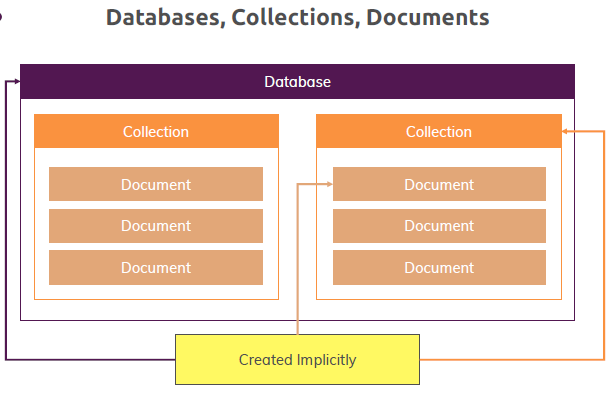
}

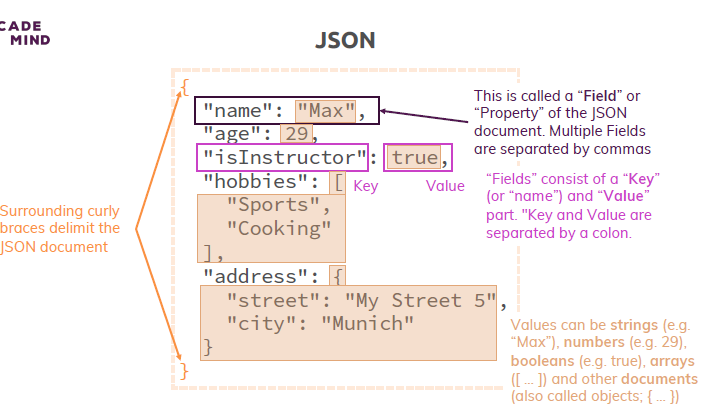


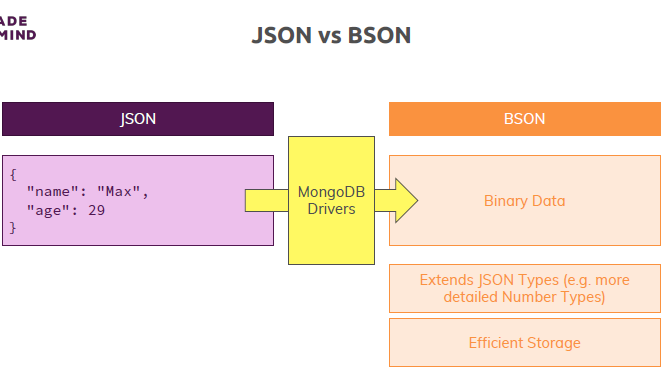


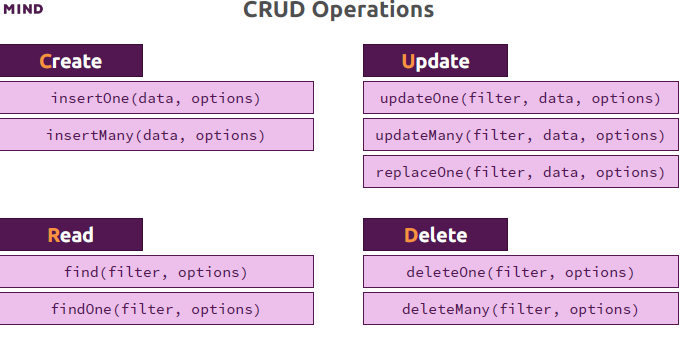


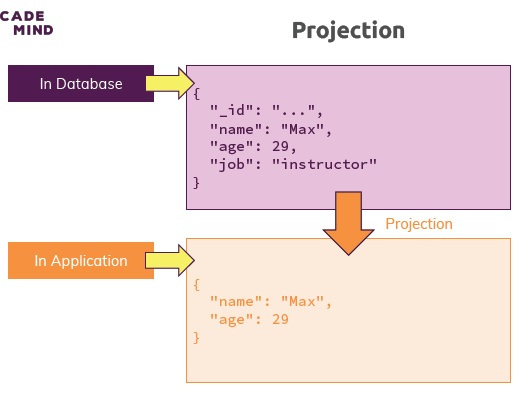
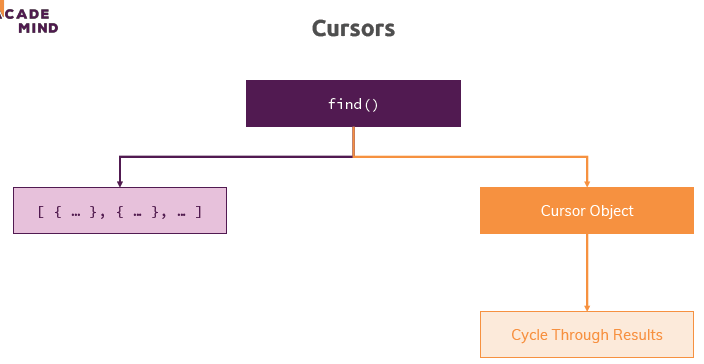


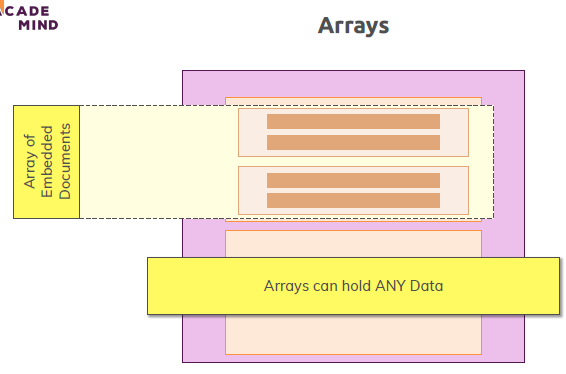
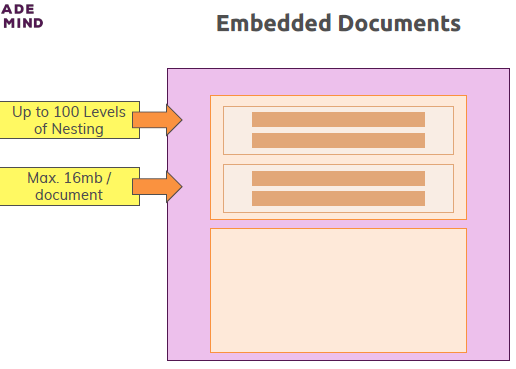


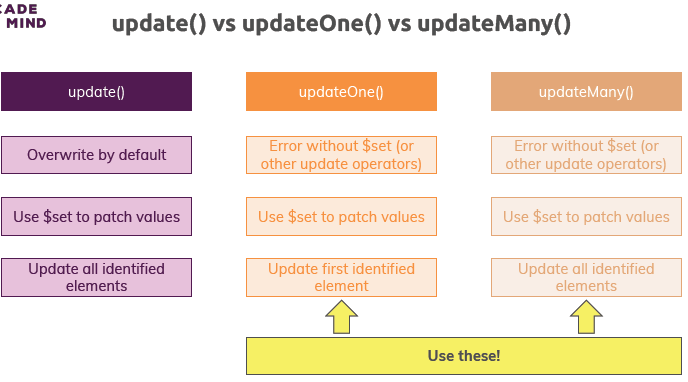


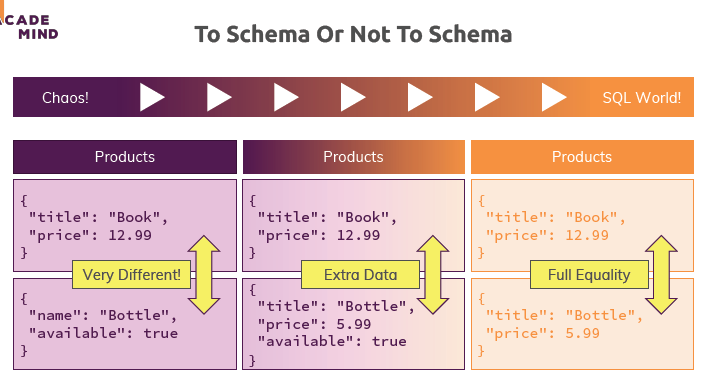


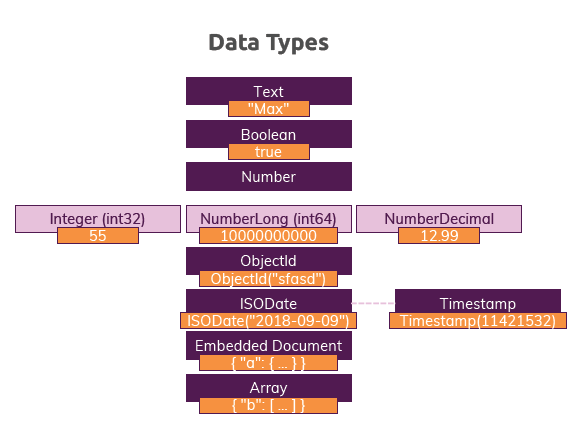












**Important data type limits are:**

* Normal integers (int32) can hold a maximum value of +-2,147,483,647
* Long integers (int64) can hold a maximum value of +-9,223,372,036,854,775,807
* Text can be as long as you want - the limit is the 16mb restriction for the overall document

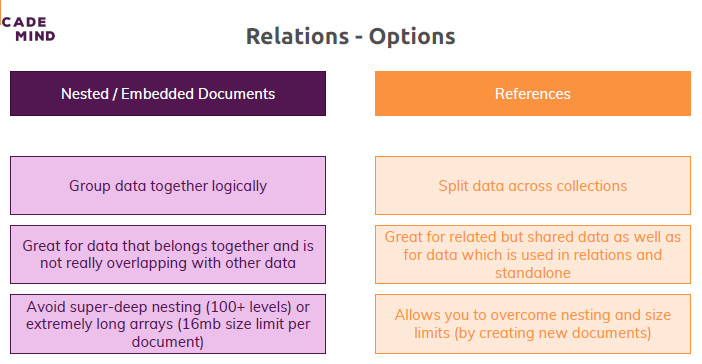
It's also important to understand the difference between int32 (NumberInt), int64 (NumberLong) and a normal number as you can enter it in the shell. The same goes for a normal double and NumberDecimal.

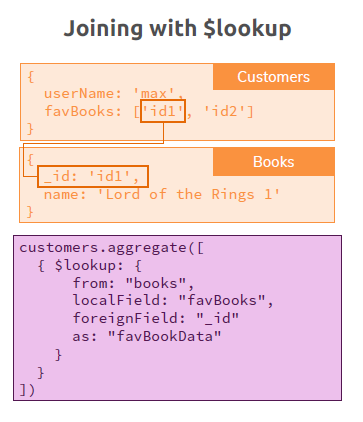
**NumberInt** creates a **int32** value => NumberInt(55)

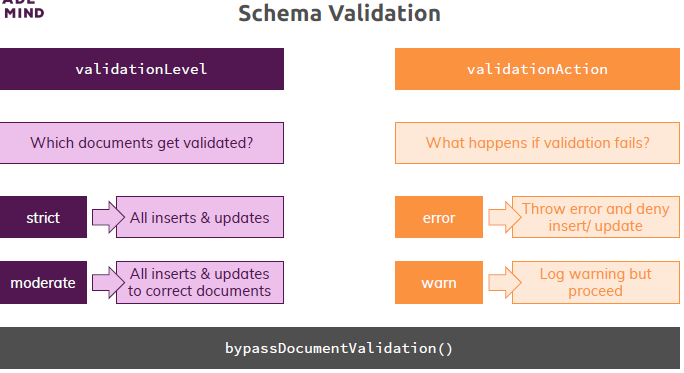
**NumberLong** creates a **int64** value => NumberLong(7489729384792)

If you just use a number (e.g. insertOne({a: 1}), this will get added as a **normal double** into the database. The reason for this is that the shell is based on JS which only knows float/ double values and doesn't differ between integers and floats.

**NumberDecimal** creates a high-precision double value => NumberDecimal("12.99") => This can be helpful for cases where you need (many) exact decimal places for calculations.

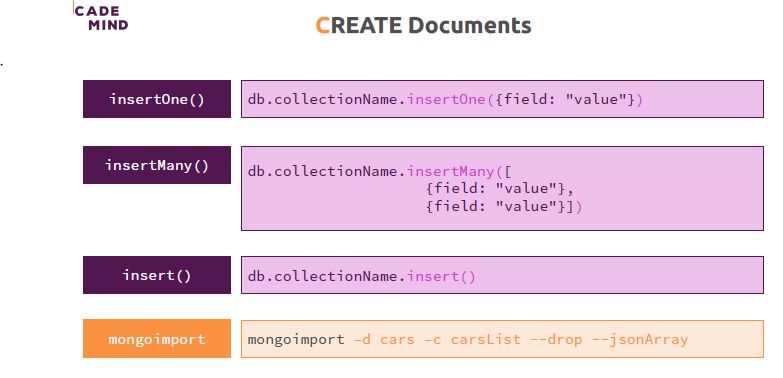


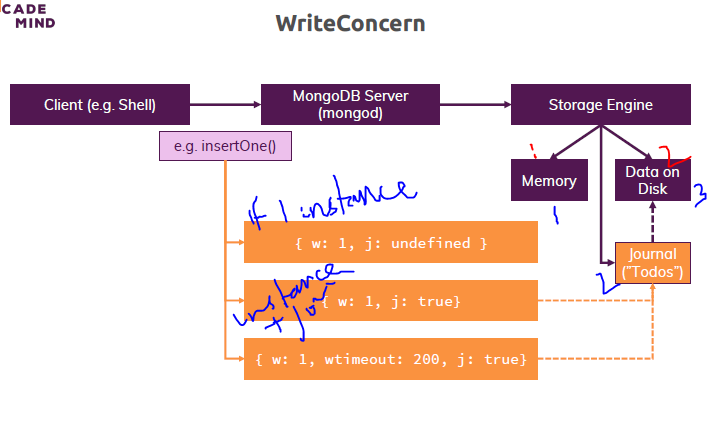




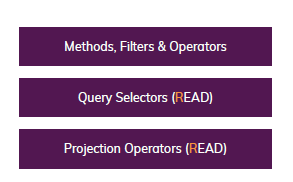
**CRUD**

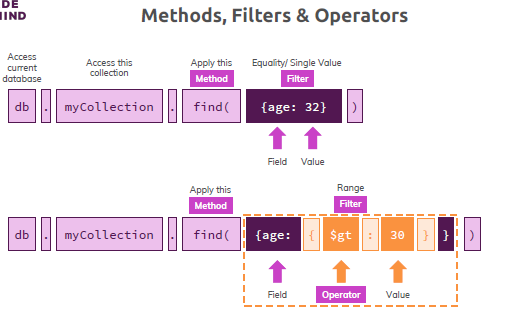
**Insert**

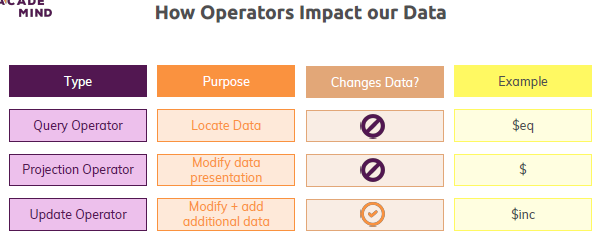


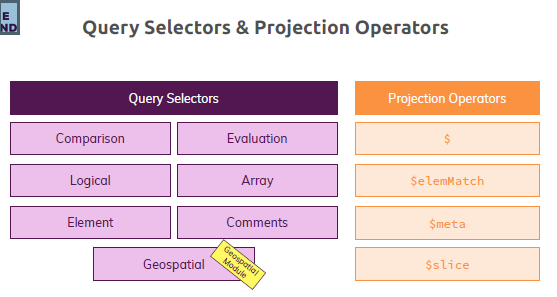


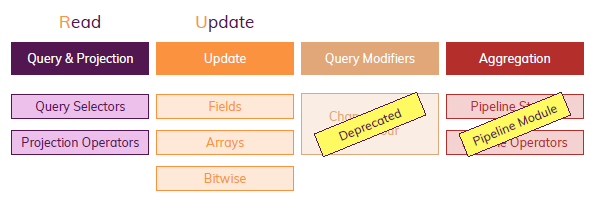
**Select Data**

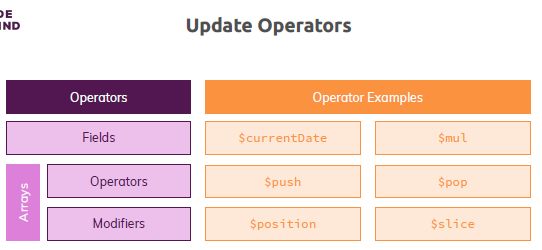




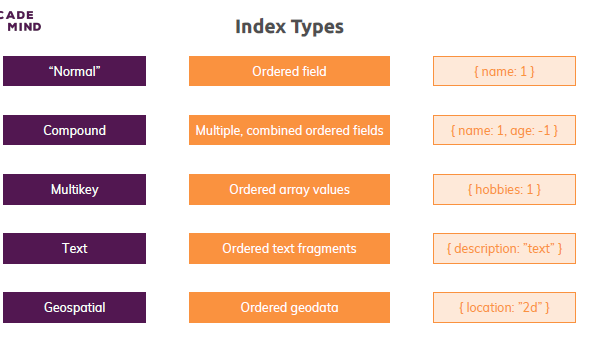
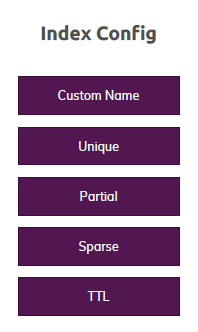


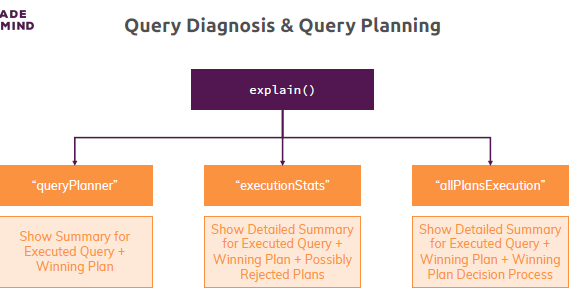


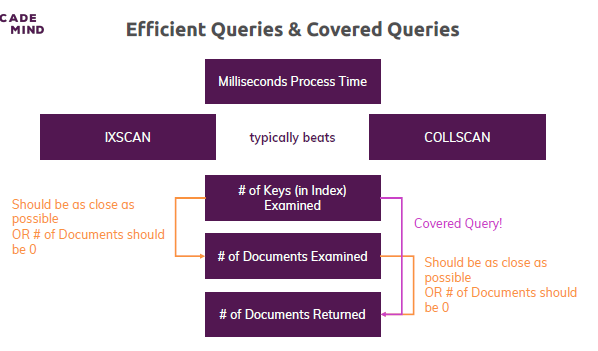


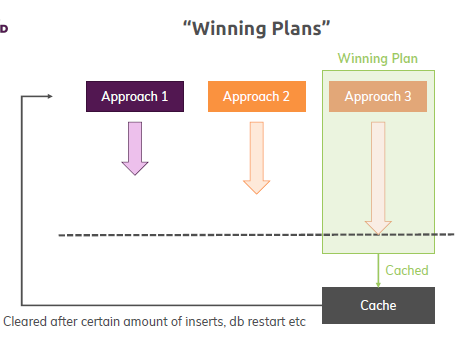


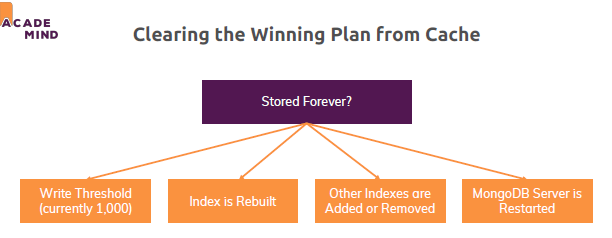
Index

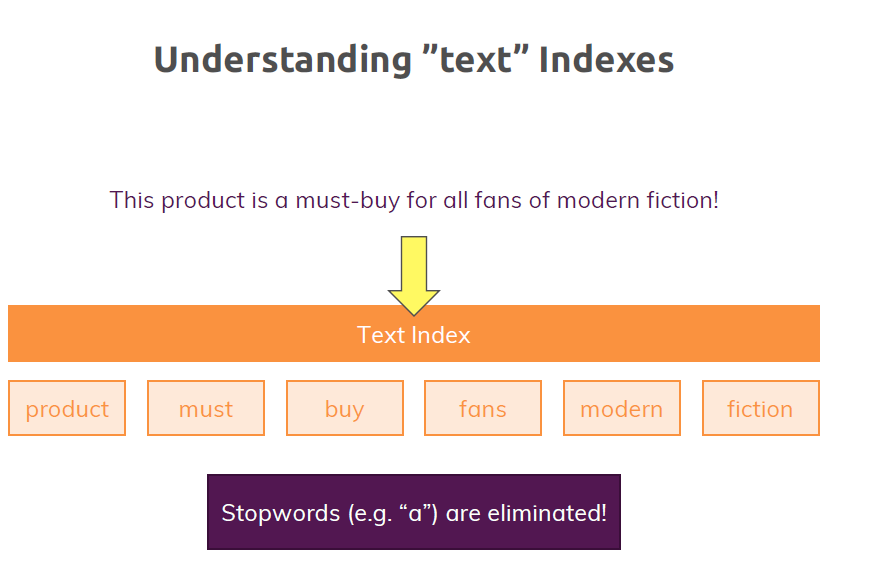
 

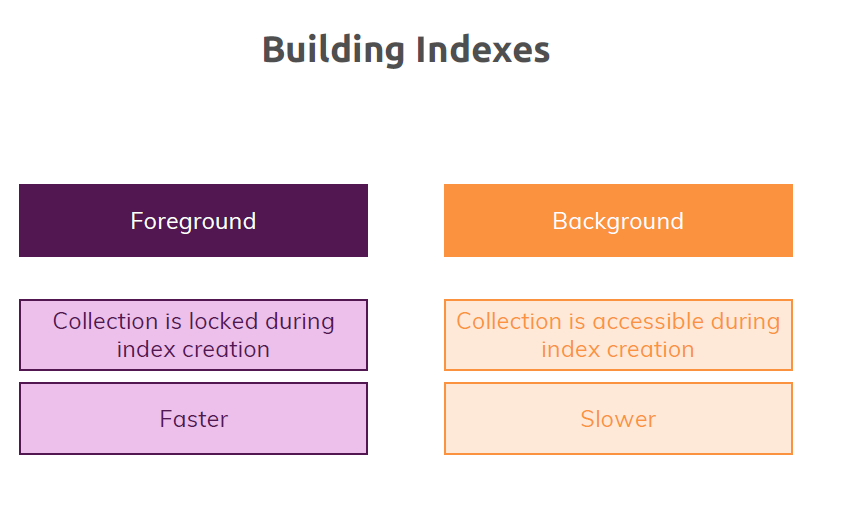












Aggregation framework

