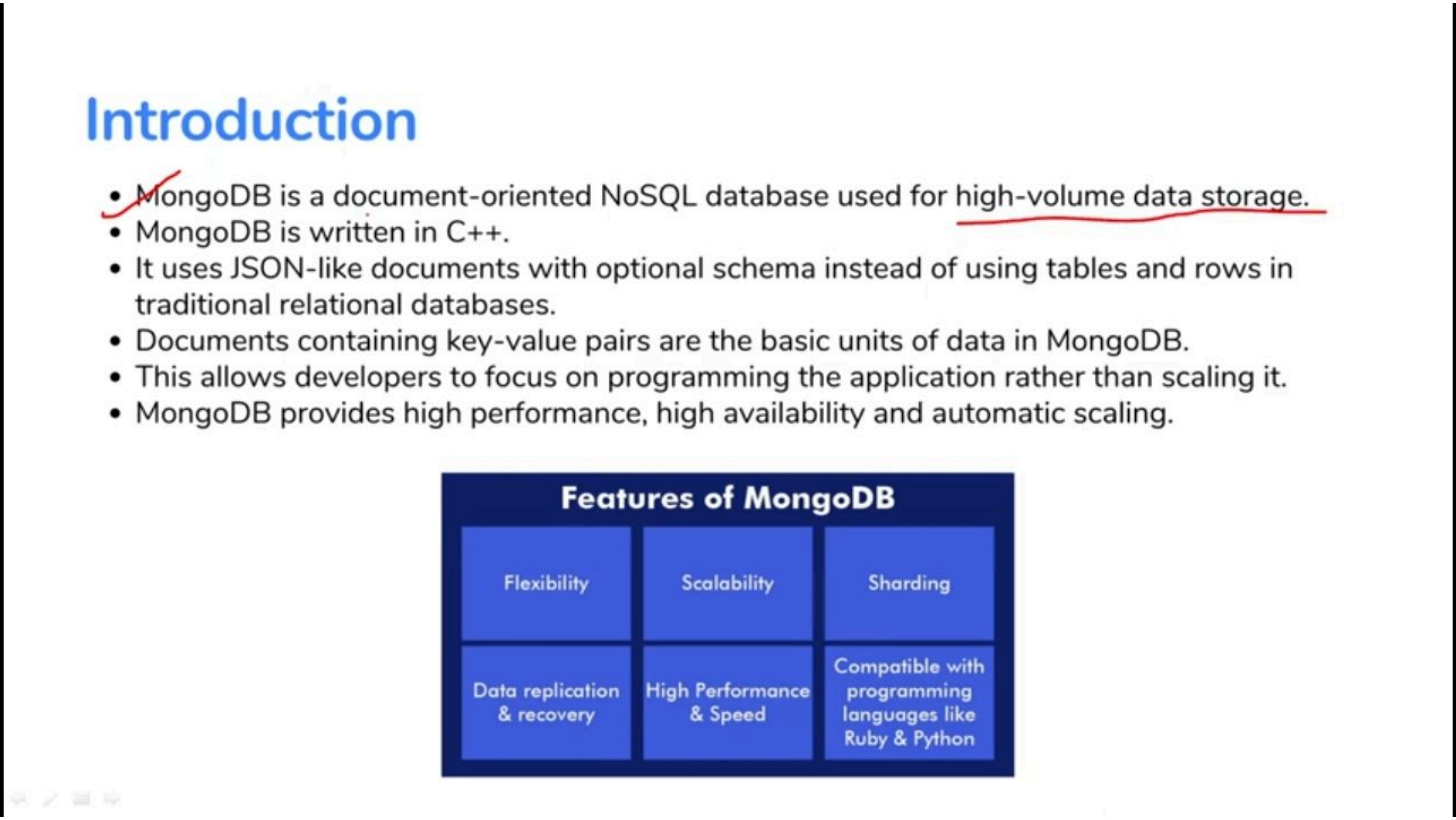
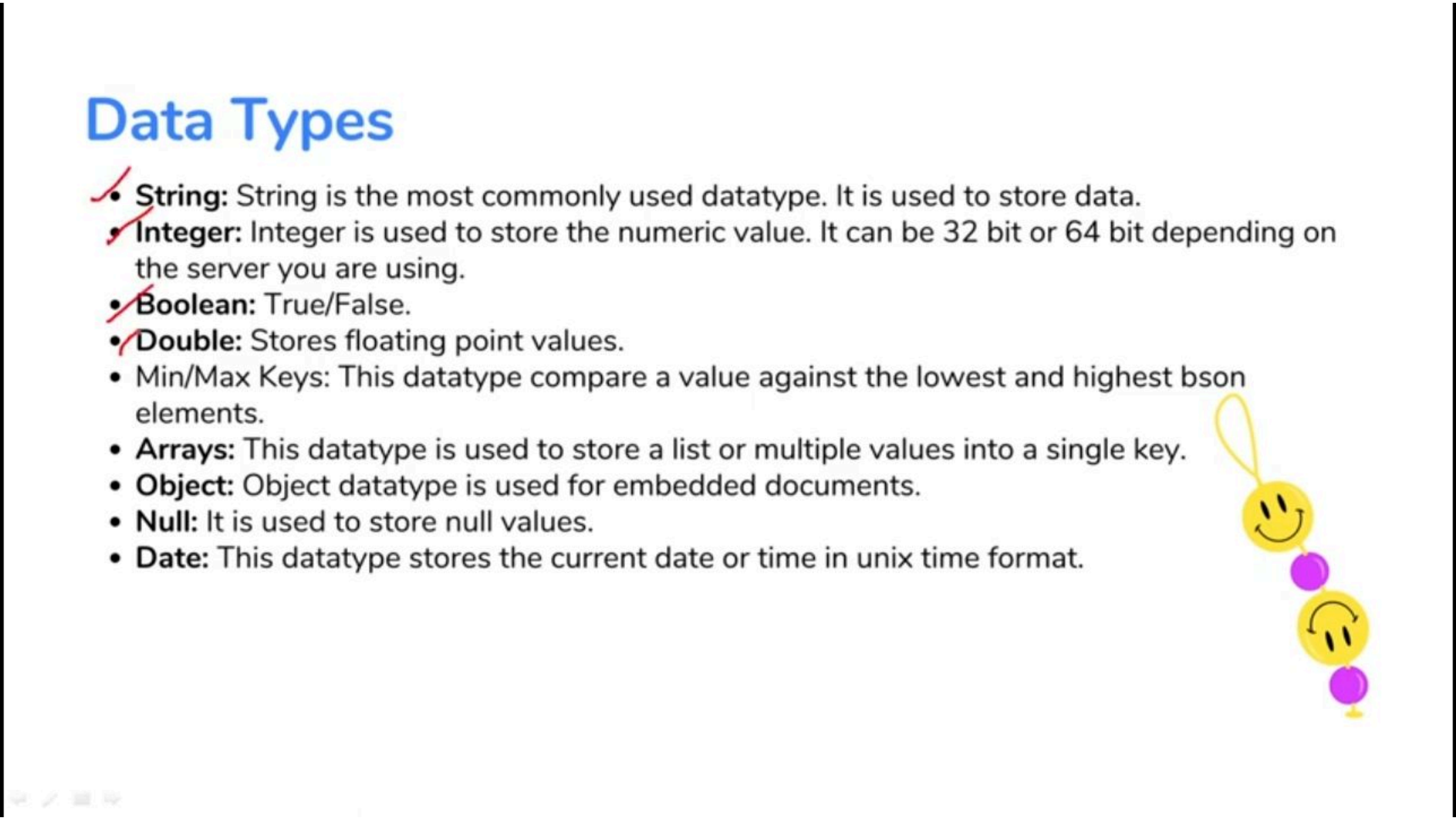


Introduction

- MongoDB is a document-oriented NoSQL database used for high-volume data storage.
- MongoDB is written in C++.
- It uses JSON-like documents with optional schema instead of using tables and rows in traditional relational databases.
- Documents containing key-value pairs are the basic units of data in MongoDB.
- This allows developers to focus on programming the application rather than scaling it.
- MongoDB provides high performance, high availability and automatic scaling.

Features of MongoDB

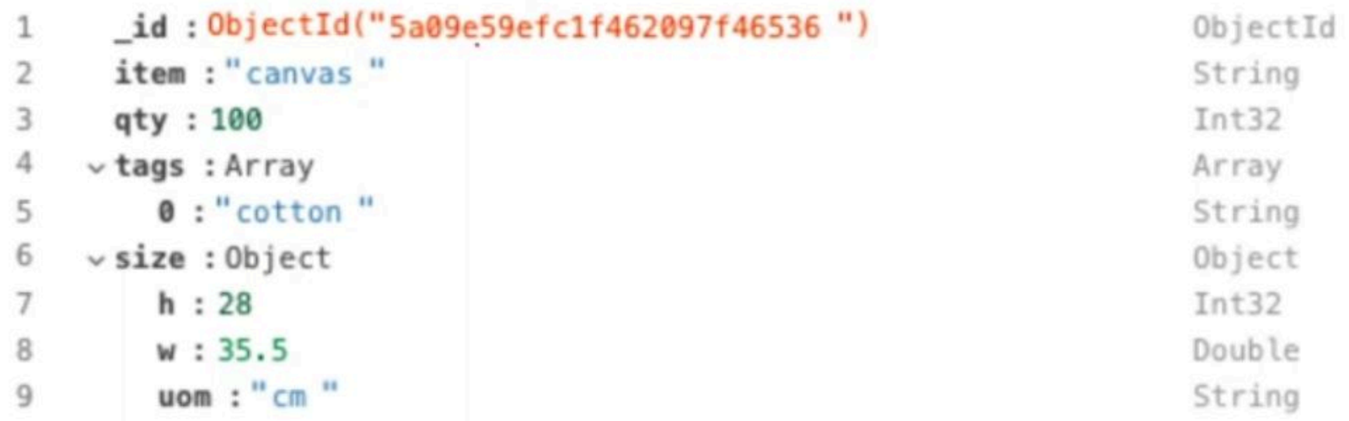
Flexibility	Scalability	Sharding
Data replication & recovery	High Performance & Speed	Compatible with programming languages like Ruby & Python



Data Types

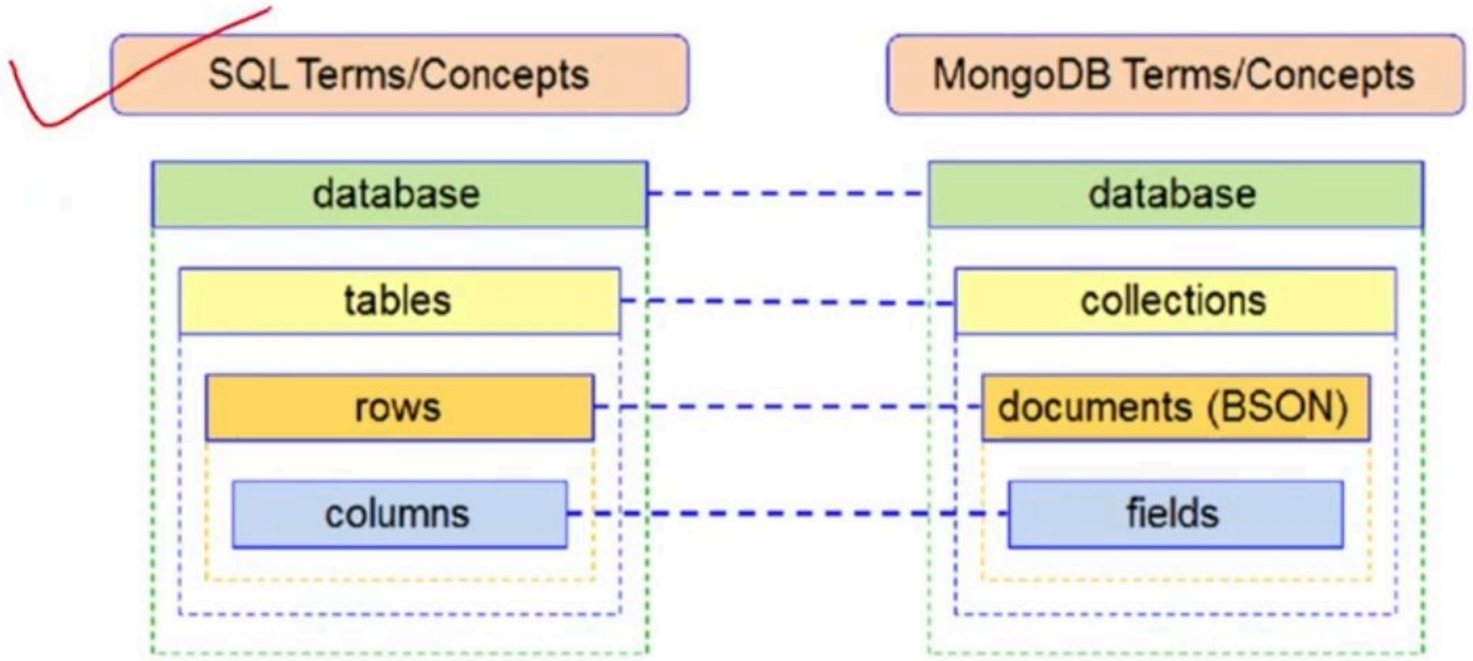
- **String:** String is the most commonly used datatype. It is used to store data.
- **Integer:** Integer is used to store the numeric value. It can be 32 bit or 64 bit depending on the server you are using.
- **Boolean:** True/False.
- **Double:** Stores floating point values.
- Min/Max Keys: This datatype compare a value against the lowest and highest bson elements.
- **Arrays:** This datatype is used to store a list or multiple values into a single key.
- **Object:** Object datatype is used for embedded documents.
- **Null:** It is used to store null values.
- **Date:** This datatype stores the current date or time in unix time format.

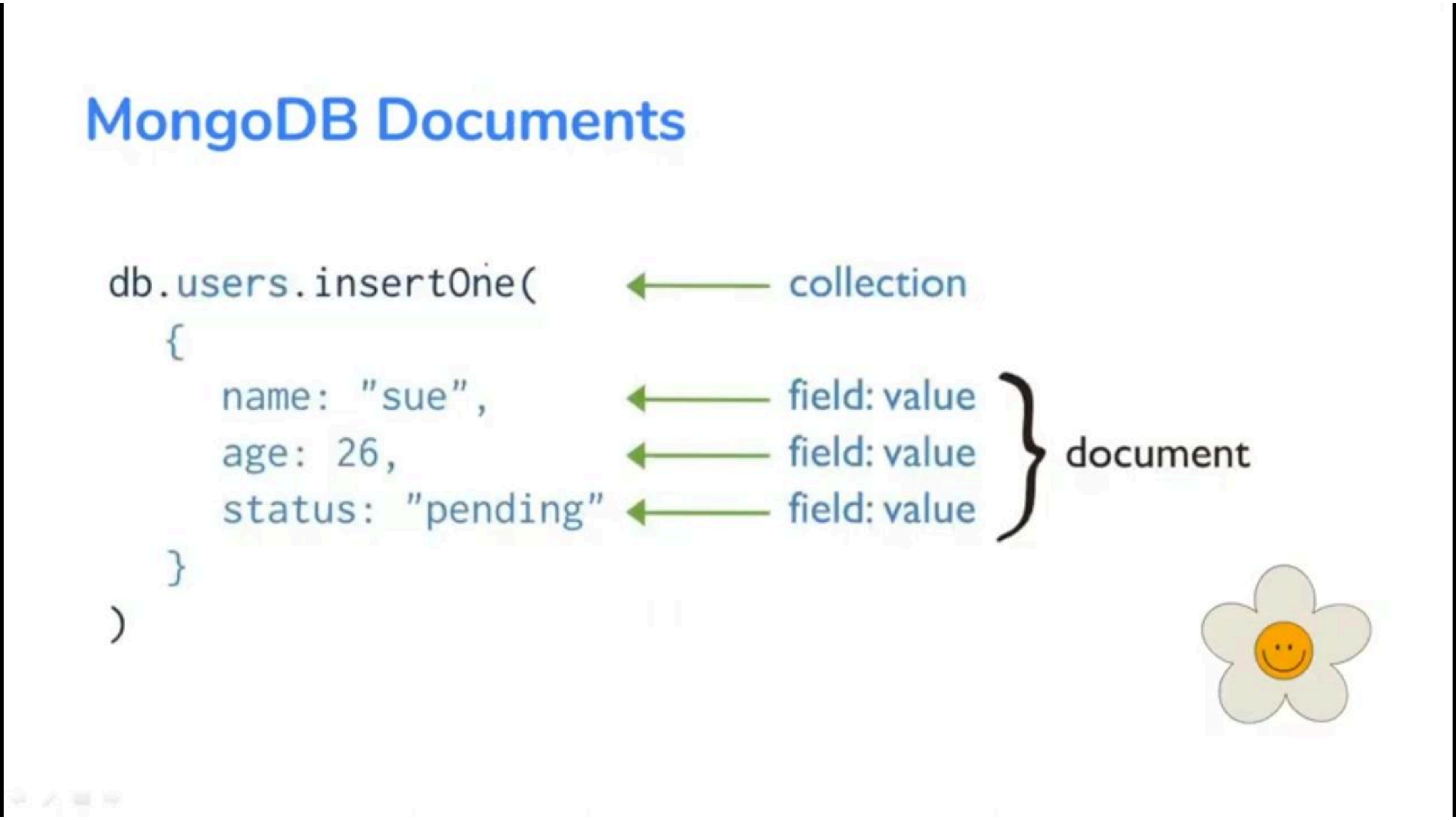
Data Types



```
  _id : ObjectId("5a09e59efc1f462097f46536 ")              ObjectId
  item : "canvas "                                          String
  qty : 100                                                 Int32
v tags : Array                                              Array
    0 : "cotton "                                           String
v size : Object                                             Object
    h : 28                                                  Int32
    w : 35.5                                                Double
    uom : "cm "                                             String
```

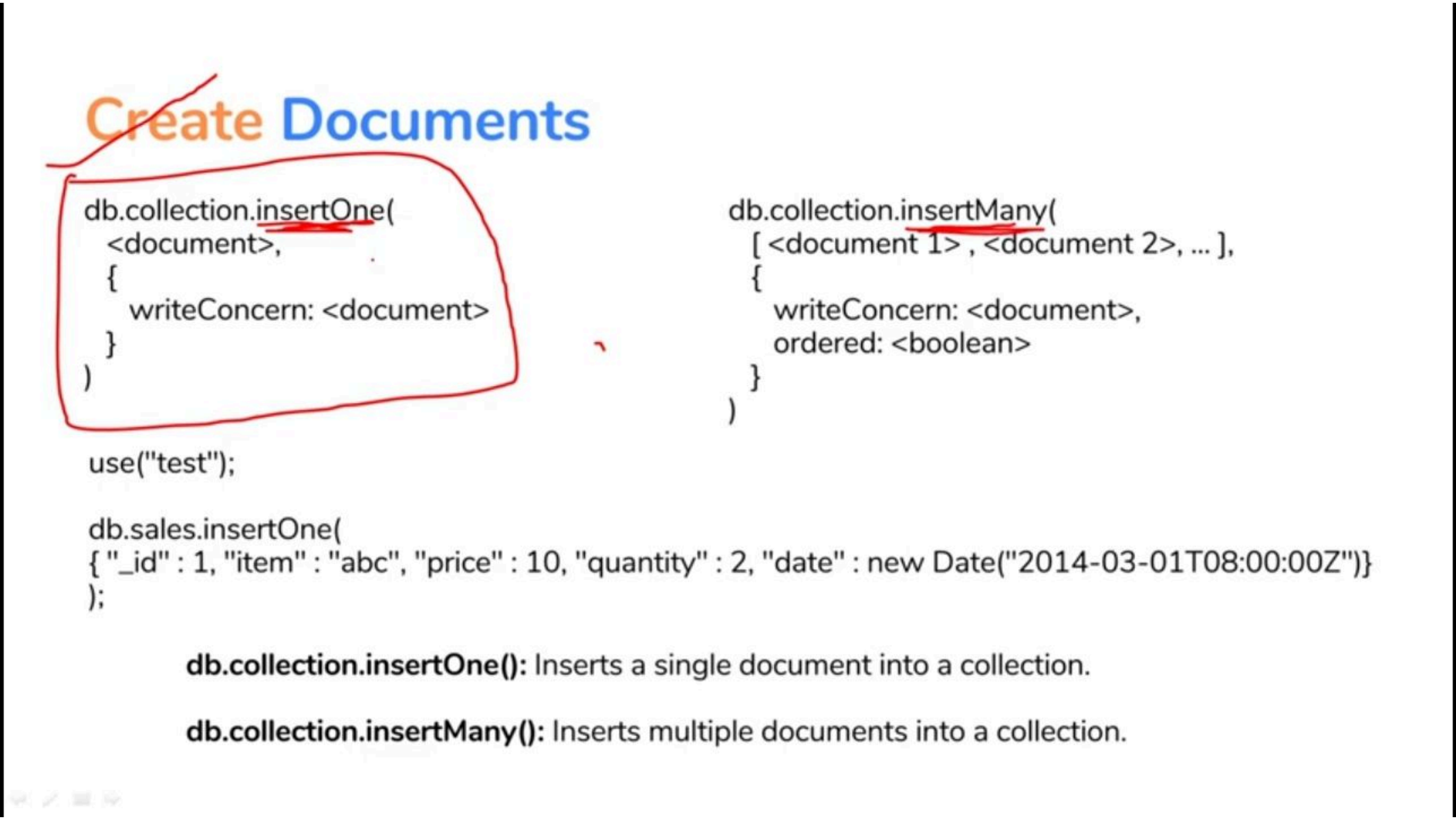
SQL vs MongoDB





MongoDB Documents

```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```



Create Documents

```
db.collection.insertOne(
  <document>,
  {
    writeConcern: <document>
  }
)
```

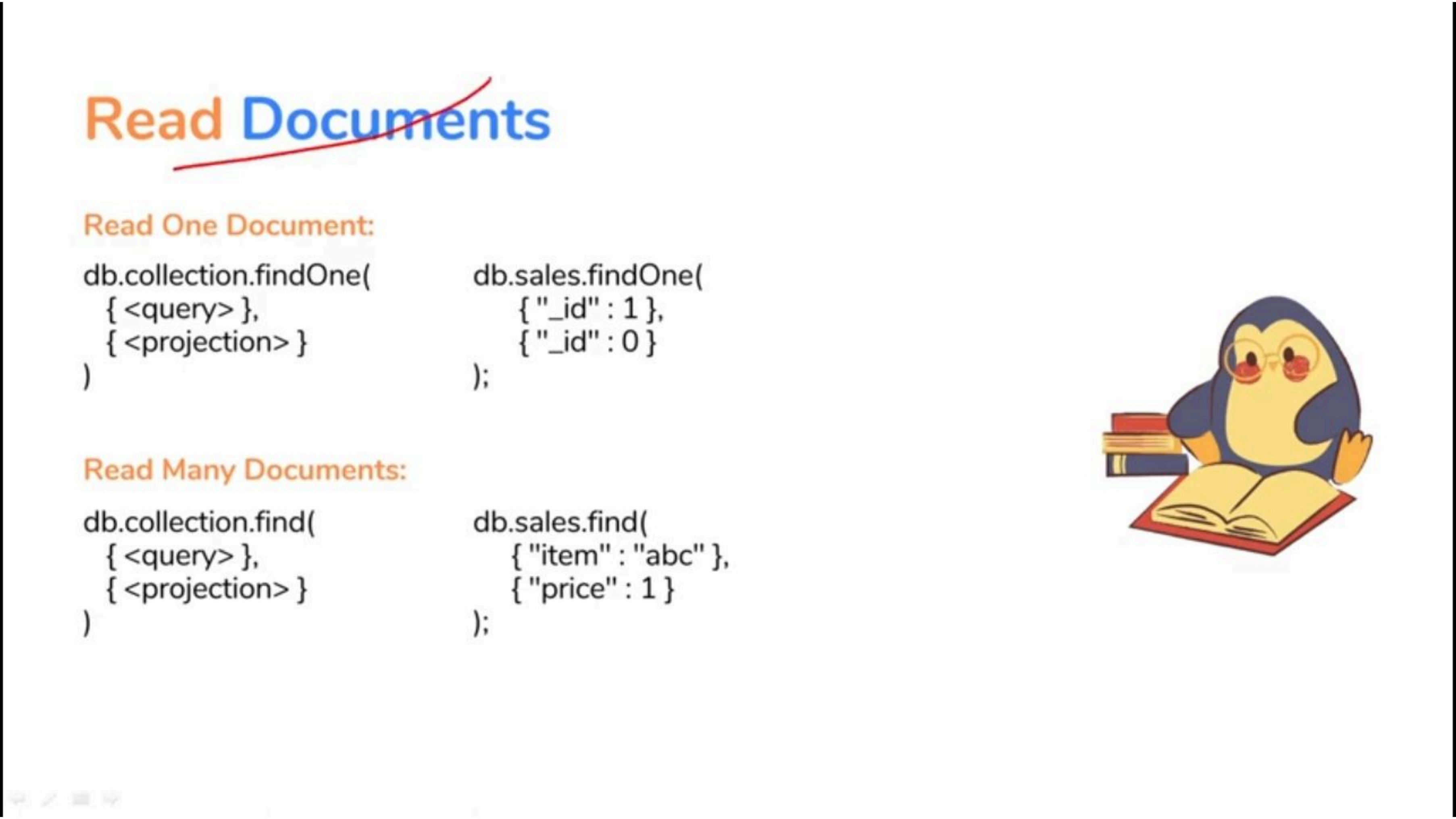
```
db.collection.insertMany(
  [ <document 1> , <document 2>, ... ],
  {
    writeConcern: <document>,
    ordered: <boolean>
  }
)
```

```
use("test");

db.sales.insertOne(
{ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : new Date("2014-03-01T08:00:00Z")}
);
```

db.collection.insertOne(): Inserts a single document into a collection.

db.collection.insertMany(): Inserts multiple documents into a collection.



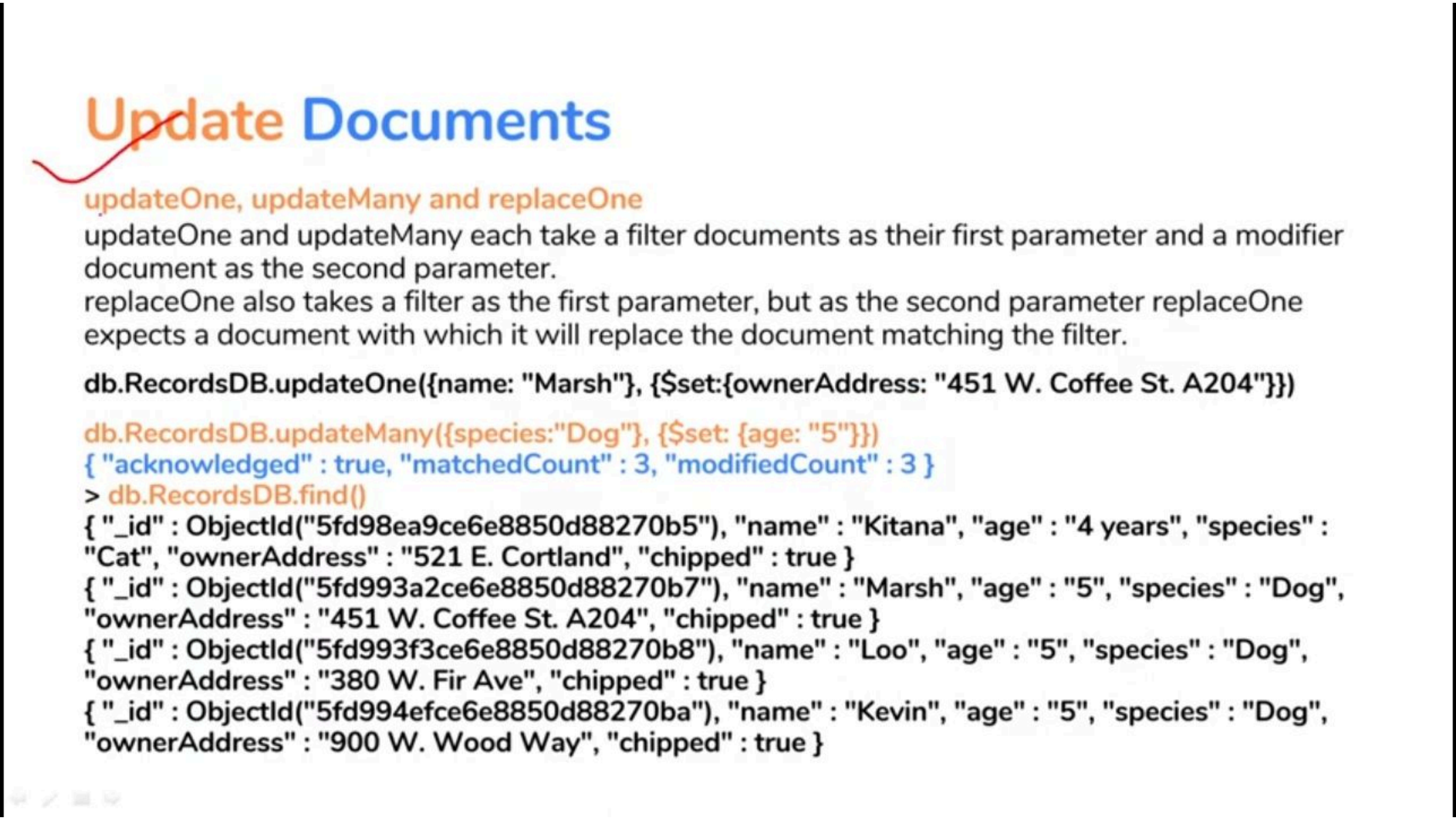
Read Documents

Read One Document:

```
db.collection.findOne(              db.sales.findOne(
  { <query> },                          { "_id" : 1 },
  { <projection> }                      { "_id" : 0 }
)                                   );
```

Read Many Documents:

```
db.collection.find(                 db.sales.find(
  { <query> },                          { "item" : "abc" },
  { <projection> }                      { "price" : 1 }
)                                   );
```



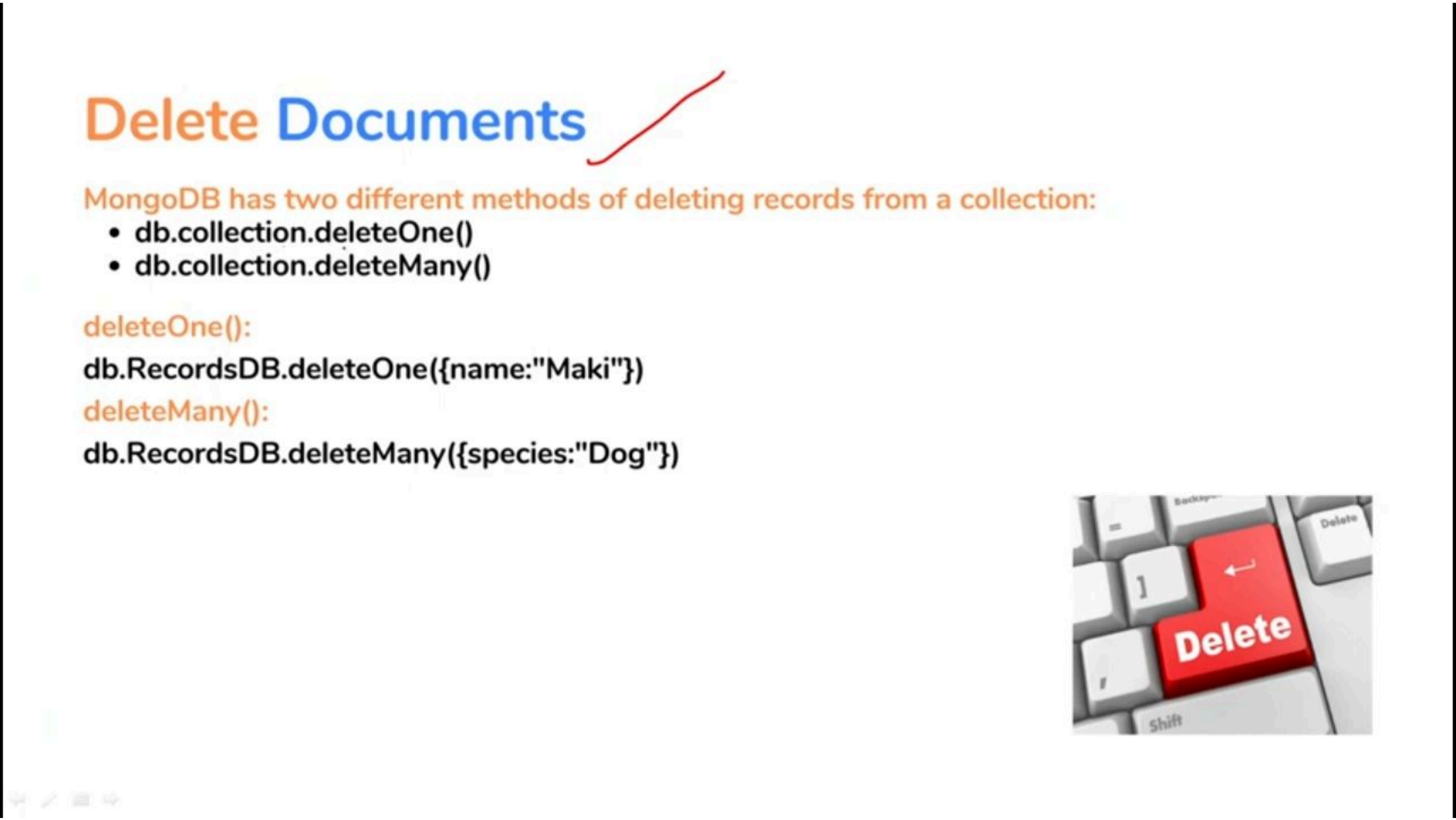
Update Documents

updateOne, updateMany and replaceOne

updateOne and updateMany each take a filter documents as their first parameter and a modifier document as the second parameter.
replaceOne also takes a filter as the first parameter, but as the second parameter replaceOne expects a document with which it will replace the document matching the filter.

```
db.RecordsDB.updateOne({name: "Marsh"}, {$set:{ownerAddress: "451 W. Coffee St. A204"}})
```

```
db.RecordsDB.updateMany({species:"Dog"}, {$set: {age: "5"}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
> db.RecordsDB.find()
{ "_id" : ObjectId("5fd98ea9ce6e8850d88270b5"), "name" : "Kitana", "age" : "4 years", "species" :
"Cat", "ownerAddress" : "521 E. Cortland", "chipped" : true }
{ "_id" : ObjectId("5fd993a2ce6e8850d88270b7"), "name" : "Marsh", "age" : "5", "species" : "Dog",
"ownerAddress" : "451 W. Coffee St. A204", "chipped" : true }
{ "_id" : ObjectId("5fd993f3ce6e8850d88270b8"), "name" : "Loo", "age" : "5", "species" : "Dog",
"ownerAddress" : "380 W. Fir Ave", "chipped" : true }
{ "_id" : ObjectId("5fd994efce6e8850d88270ba"), "name" : "Kevin", "age" : "5", "species" : "Dog",
"ownerAddress" : "900 W. Wood Way", "chipped" : true }
```



Delete Documents

MongoDB has two different methods of deleting records from a collection:

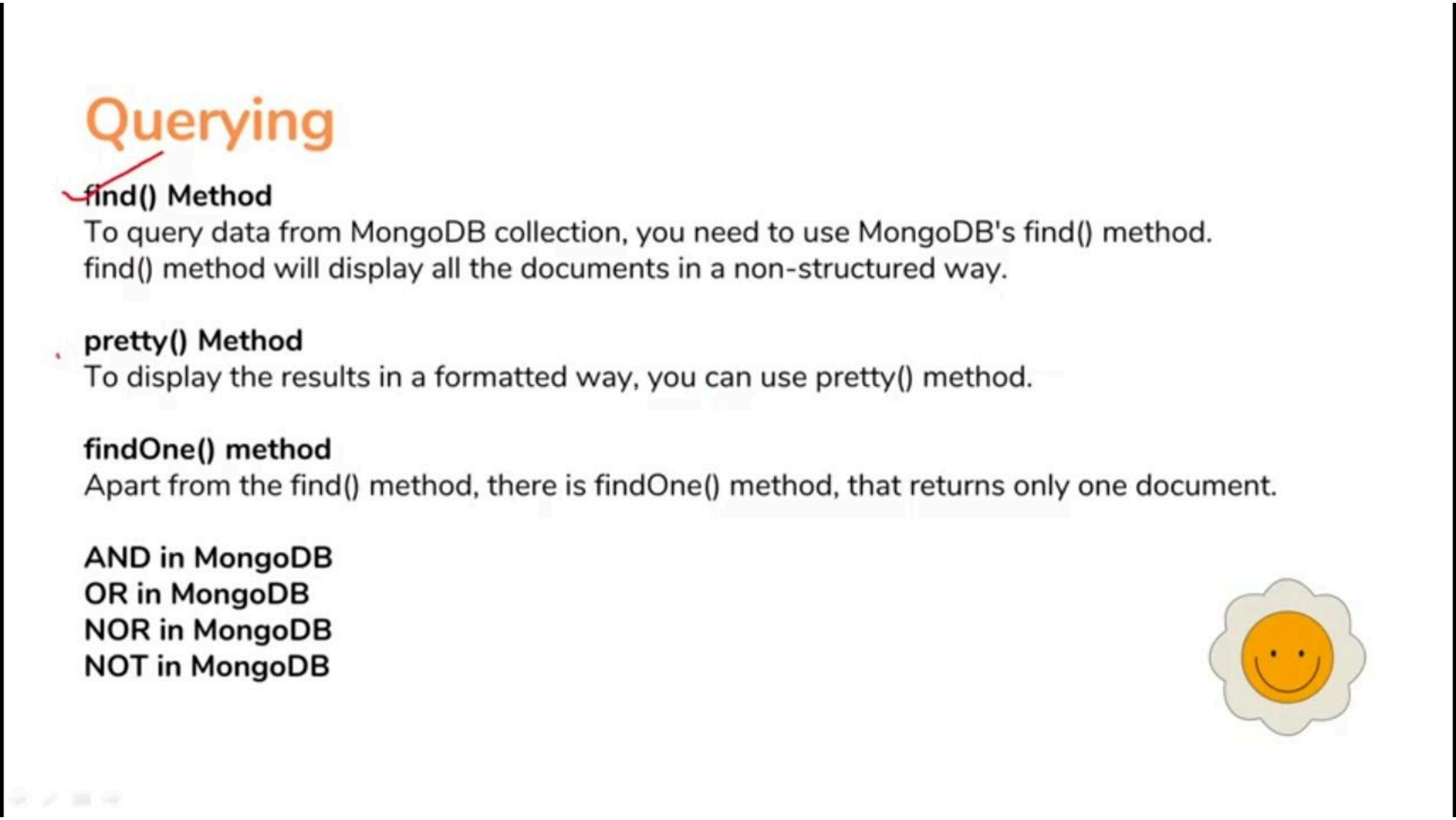
- db.collection.deleteOne()
- db.collection.deleteMany()

deleteOne():

```
db.RecordsDB.deleteOne({name:"Maki"})
```

deleteMany():

```
db.RecordsDB.deleteMany({species:"Dog"})
```



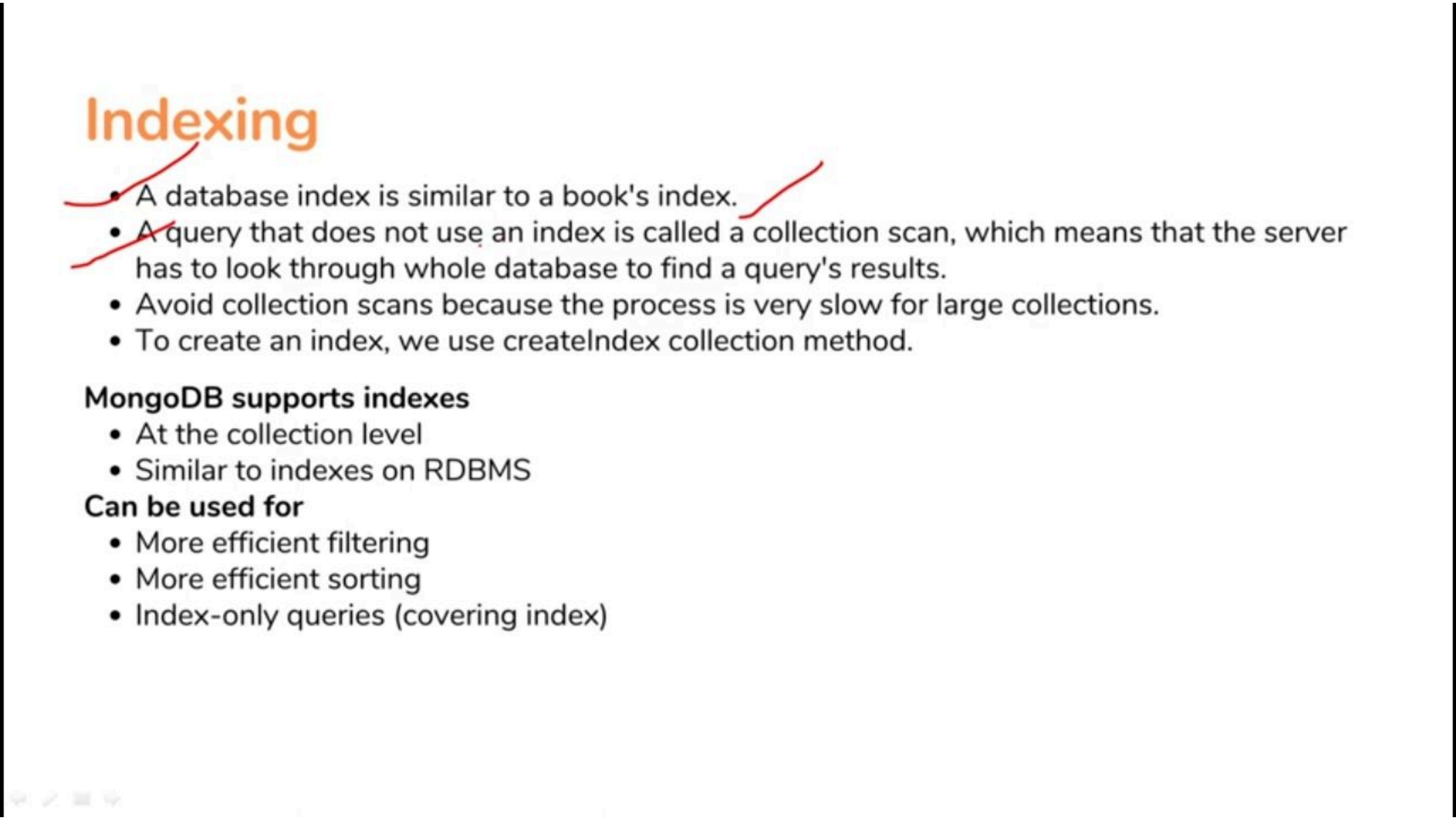
Querying

find() Method
To query data from MongoDB collection, you need to use MongoDB's find() method.
find() method will display all the documents in a non-structured way.

pretty() Method
To display the results in a formatted way, you can use pretty() method.

findOne() method
Apart from the find() method, there is findOne() method, that returns only one document.

AND in MongoDB
OR in MongoDB
NOR in MongoDB
NOT in MongoDB



Indexing

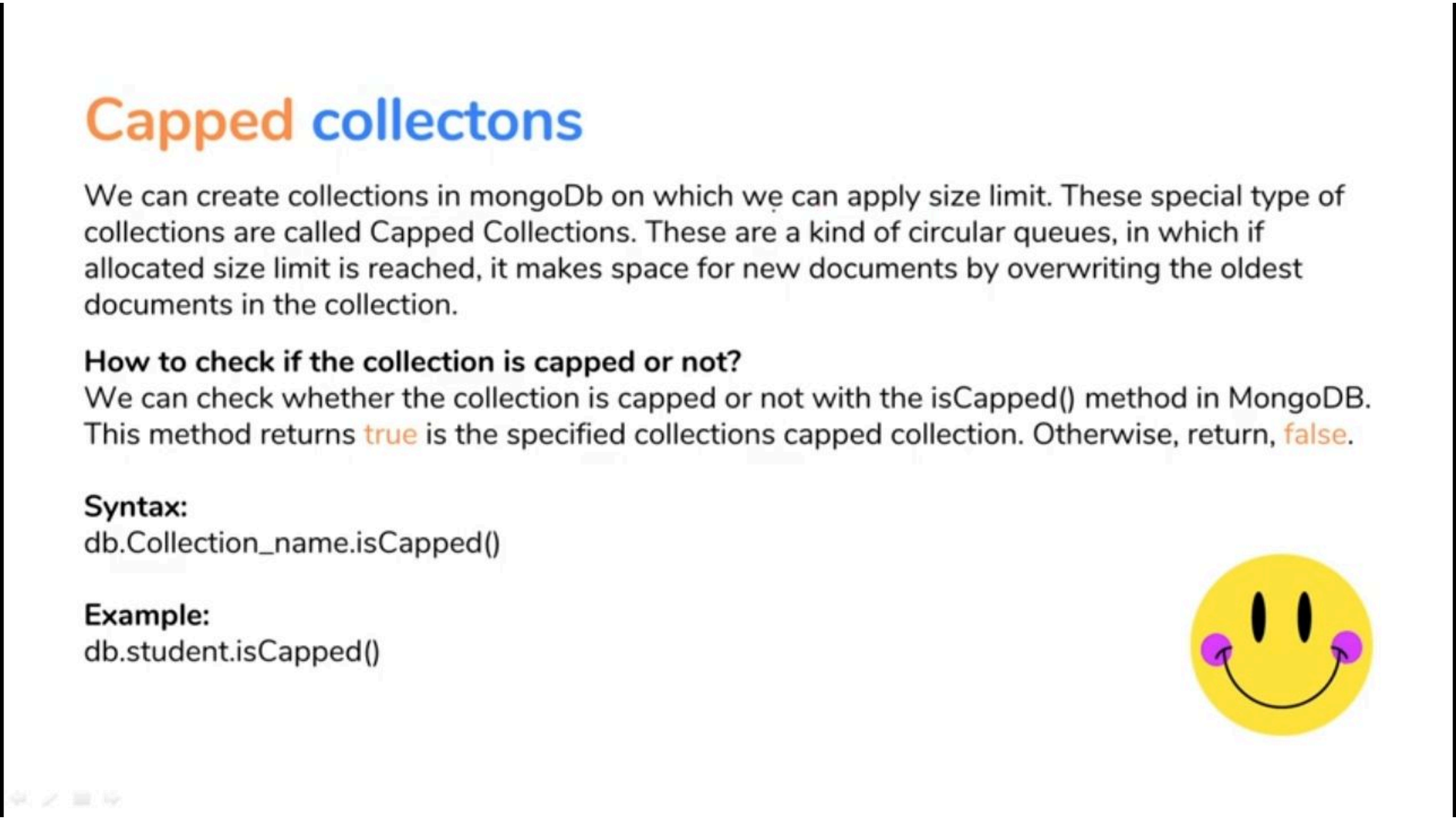
- A database index is similar to a book's index.
- A query that does not use an index is called a collection scan, which means that the server has to look through whole database to find a query's results.
- Avoid collection scans because the process is very slow for large collections.
- To create an index, we use createIndex collection method.

MongoDB supports indexes

- At the collection level
- Similar to indexes on RDBMS

Can be used for

- More efficient filtering
- More efficient sorting
- Index-only queries (covering index)



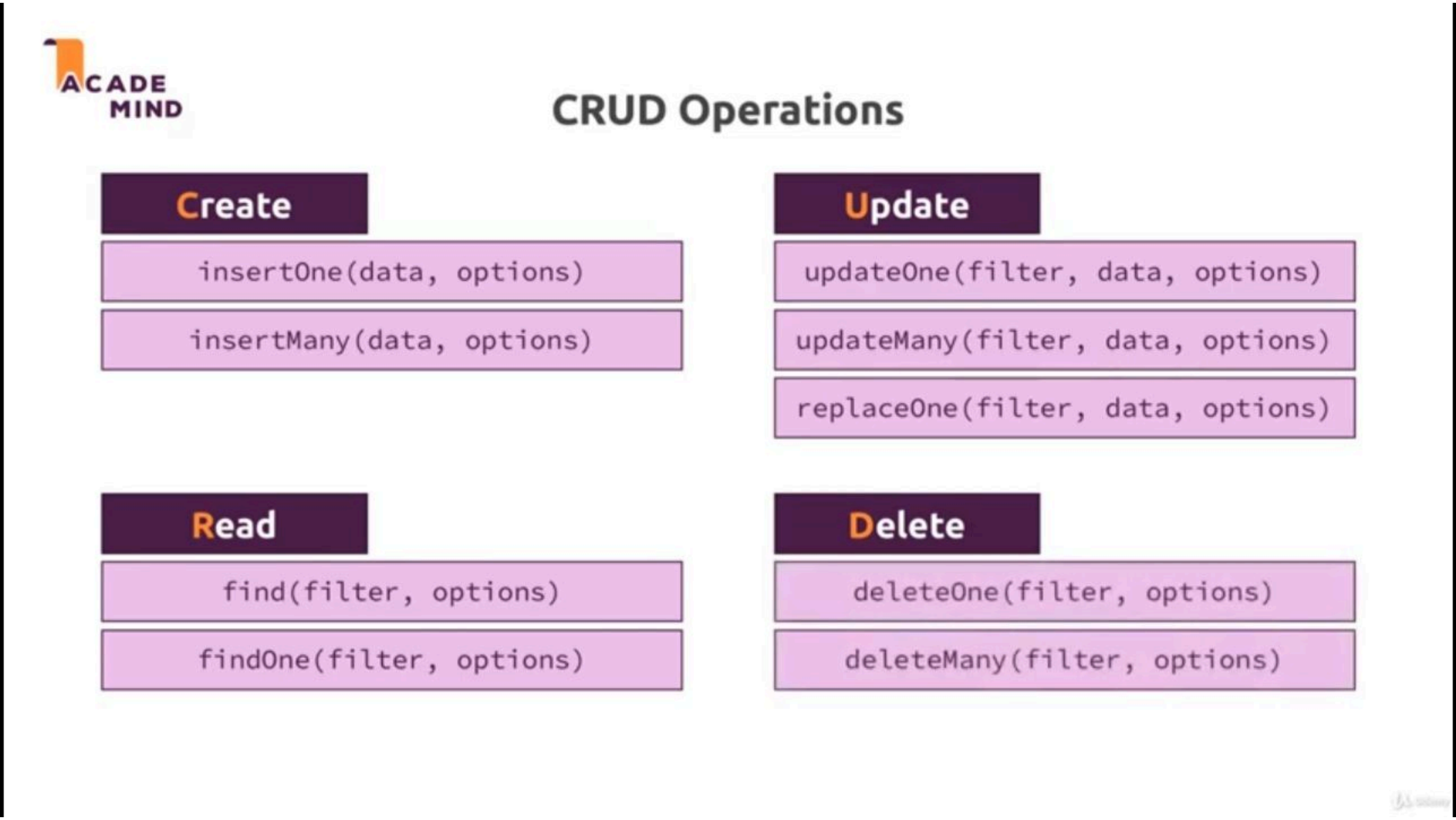
Capped collectons

We can create collections in mongoDb on which we can apply size limit. These special type of collections are called Capped Collections. These are a kind of circular queues, in which if allocated size limit is reached, it makes space for new documents by overwriting the oldest documents in the collection.

How to check if the collection is capped or not?
We can check whether the collection is capped or not with the isCapped() method in MongoDB. This method returns true is the specified collections capped collection. Otherwise, return, false.

Syntax:
```
db.Collection_name.isCapped()
```

Example:
```
db.student.isCapped()
```



ACADE
MIND

CRUD Operations

Create

```
insertOne(data, options)
insertMany(data, options)
```

Update

```
updateOne(filter, data, options)
updateMany(filter, data, options)
replaceOne(filter, data, options)
```

Read

```
find(filter, options)
findOne(filter, options)
```

Delete

```
deleteOne(filter, options)
deleteMany(filter, options)
```