Mongo Crud operations

By default mongo creates a collection in the db for us if collection doesn't exist on InsertOne()

insertMany([d1, d2, d3]);

Find() → $in → allows us to select all docs that have a field value equal to any of the values specified in the array

db.zips.find({city: {$in: ["PH", "CH"]}});

$eq → to find documents with a field & a value.

$gt, $gte, $lt, $lte → less than, greater than equal to

ex → db.sales.find({"items.price": {$gt: 50}});

$elemMatch → to get values out of an array.

db.accounts.find({products: {$elemMatch: {$eq: "Invest"}}});

We can also use this for instances where a single array match multiple query criteria

{field: {$elemMatch: {query1, query2 ...}}}

like an and operation.

logical operators, && || or operators, if we just use multiple query with comma seperator it is considered as and operator.

```
ex) db.collections.find({
      $or : [
        { expression 1 },
        { expression 2 },
      ]
    });
```

But when we are using same operator more than once in our query, we need explicit $and operator. because json can't store two variables in the same or operator.

# Replacing a document -> for incorrect insertion.
replaceOne()

db.collection.replaceOne(filter, replacement, options)

-id is a good field comes this category

UpdateOne().

db.collection.updateOne(<filter>, <update>, {options})

↓

In update we can use $set & $push.

$set -> Adds a new field & value to doc, replaces the value of a field with a specified value.

$push → Appends a value to an array if absent
$push adds the array field with the value as its element.

Upsert → Insert a document with provided inputs if matching documents don't exist

The update operations will be carried out

```
db.podcasts.updateOne({title: "The xyz"},
                      {$set: {topics: ["database", "mongo"]}},
                      {upsert: true})
```

```
db.podcasts.updateOne({_id: "_____"}, {$push: {
  hosts: "Nic"}})
```

FindAndModify() → Returns a document that has been updated.

If we want to get the no of users who downloaded a podcast we will first do ~~find~~ .updateOne & then findOne.

Here we are doing 2 round trips to the server & Another user can modify the doc before the findOne() returning a diff version of the doc

To prevent the last use case we can use

FindAndModify()

ex→

```
db.podcasts.findAndModify({
  query: {_id: ObjectId("62...")},
  update: {$inc: {downloads: 1}},
  new: true
});
```

We set new → true so that the new version of updated document is returned.

updateMany() → filter doc, update doc, options object

↓

It is not an all-or-nothing operation

will not roll back updates

Updates will be visible as soon as they're performed

Not good for financial transactions

deleteOne → takes filter & options.

deleteMany() → same just add filter.

Cursor → pointer to the result set of query.

Cursor methods which are chained to queries performs actions on the result set

ex→

PTO

db.companies.find({category_code: "music"}).sort({name: 1}) , 1 represents ascending & -1 " descending

we can sort on multiple properties at the same time by passing additional fields

db.companies.find({category_code: "music"}, {name: 1}).sort({name: 1})

we can also limit this by just adding .limit(N) → no. to limit.

Projection → measure we only want to see name field

Projection → Specific data from a query in MongoDB

db.collection.find({query}, {projection});

ex → db.Sales.find({sector: "Rest"}, {name: 1, Order: 1});

this will result with 3 fields: id, name & Order, if we want to exclude the _id field we will set _id: 0} → inside projection to not get in result set

Inclusion & exclusion statements can't be combined in projections

_id field is an exception.

Exclude fields →

db.collection.find({ query }, { date: 0, 'address.zip': 0 })

projection to remove

Count documents in Mongo

db.collection.countDocuments(<query>, <options>)