Day 27

Insert

- Inserting a single document

Missing _id field. MongoDB will assign an `_id` with an `ObjectId` type

```
db.tv_shows.insert({
  title: "Disenchantment", season: 1,
  episodes: [ { ... }, { ... }, { ... } ]
})
```

Document to be inserted

- Inserting multiple documents

```
db.tv_shows.insertMany([
  { _id: 'abc', ... },
  { _id: 'def', ... },
  { ... }
])
```

Inserting multiple document simultaneously. Example shown with `_id` field. MongoDB will not assign a key

Insert

Returns the document with _id property

Insert the new Document into the collection

```
@Autowired
private MongoTemplate mongoTemplate;

Document toInsert = //...
Document newDoc = mongoTemplate.insert(toInsert, "tv_shows");

ObjectId id = newDoc.getObjectId();
```

Get the new _id

```
List<Document> docsToInsert = new LinkedList<>();
// Add Document to list
...

Collection<Document> newDocs = mongoTemplate.insert(
     docsToInsert, "tv_shows");
```

Bulk insert. The returned collection will contain _id

Delete

- Deleting a single document
  - Find the first document that satisfy the requirement

```
db.tv_shows.deleteOne(
   { cancelled: { $exists: true } }
)
```

- Deleting multiple documents

```
db.tv_shows.deleteMany(
   { cancelled: { $exists: true } }
)

db.tv_shows.deleteMany({})
```

Delete

Create a `Query` (predicate) by defining a predicate

Delete all documents in the collection that matches the predicate

```
@Autowired
private MongoTemplate mongoTemplate;

Query query = Query.query(
   Criteria.where("cancelled").exists(true));
DeleteResult result = mongoTemplate.remove(query, "tv_shows");

System.out.printf("Deleted documents: %d\n", result.getDeleteCount());
```

`remove()` returns the number of deleted documents

Returns the deleted document

```
Document deleted = mongoTemplate.findAndemove(
   query, Document.class, "tv_shows");
```

Updates

- Update a single document

```
db.tv_shows.updateOne(
   { lang: "eng" },
   {
      $set: { lang: "English" }
   }
)
```

Set a predicate to filter the documents for updates

`$set` - adds `lang` if it does not exists, update `lang` if it exists

- Update all documents

```
db.tv_shows.update(
   { lang: "eng" },
   {
      $set: { locale: "GB" }
   }
)
```

Will insert the attribute `locale` to all documents with the attribute `lang` having `eng` value. Assuming `locale` does not exists

Updates

```
db.tv_shows.updateOne(
  { title: "Arrow" },
  {
    $set: { status: "Ended" },
    $inc: { "rating.average": .1 },
    $unset: { webChannel: "" },
    $set: { seasons: 6 }
  }
)
```

Modify the `status` property

Increment the `average` property by .1

Remove the `webChannel` property

Add a new property to document

Update Operators

- `$currentDate`, `$inc`, `$min`, `$max`, `$unset`
- See https://docs.mongodb.com/manual/reference/operator/update-field/

Updates

```
Query query = Query.query(
  Criteria.where("title").is("Arrow")
);

Update updateOps = new Update()
  .set("status", "Ended")
  .inc("rating.average", .1)
  .unset("webChannel")
  .set("seasons", 6);

UpdateResult updateResult = mongoTemplate.updateMulti(
  query, updateOps, Document.class, "tv_shows");

System.out.printf("Documents updated: %d\n",
    updateResult.getModifiedCount());
```

List of mutations to be applied to documents

Updates

```
db.tv_shows.updateOne(
   { title: "Arrow" },
   {
      $push: { genres: "Mystery" },
      $pop: { genres: -1 },
   }
)
```

Appends a new value to the array `genres` property

Removes an element from the front of `genres` array

- `$pop` removes an item from an array
  - 1 removes from the back, element with the largest index
  - -1 removes from the front, element with the 0 index

Updates

```
Query query = Query.query(
  Criteria.where("title").is("Arrow")
);

Update updateOps = new Update()
  .push("genres").each("Mystery")
  .pop("genres", Update.Position.FIRST);

UpdateResult updateResult = mongoTemplate.updateMulti(
  query, updateOps, Document.class, "tv_shows");

System.out.printf("Documents updated: %d\n",
    updateResult.getModifiedCount());
```

Operations on array. Other operations include `atPosition()`, `slice()`, `sort()`

Upsert

- Combination of update and insert, upsert mode is by default off
- Insert a document if it does not match the predicate
- Update the document if it matches the predicate

```
db.tv_shows.update(
  { title: "Disenchantment", season: 1 },
  {
    title: "Disenchantment", season: 1,
    lang: "eng", genres: [ 'Animation', 'Comedy' ],
    ...
  }
  { upsert: true }
)
```

Match the following predicate

Document to insert or to be updated

Document cannot contain any `$set` operator

Set `upsert` to true

Upsert

```
Query query = Query.query(
   Criteria.where("title").is("Disenchantment")
      .and("season").is(1)
);
Update updateOps = new Update()
   .set("title", "Disenchantment")
   .set("season", 1)
   .set("lang", "eng")
   .push("genres").each(List.of("Animation", "Comedy").toArray());

UpdateResult updateResult =
   mongoTemplate.upsert(query, updateOps, "tv_shows");

System.out.printf("Documents updated: %d\n",
      updateResult.getModifiedCount());
```

Indexes

- Like RDBMS, indexes are used by MongoDB to efficiently access document when querying MongoDB
- The `_id` field is always indexed

Create a index on the attribute `title`

```
db.tv_shows.createIndex({
  title: 1
})
```

Ascending

Create a composite index on the attributes `title` and `imdb.rating`

```
db.tv_shows.createIndex({
  year: 1
  "imdb.rating": -1
})
```

Descending

Using Indexes

```
db.tv_shows.find({ "title": /bad/i })
```

```
db.tv_shows.find({ "year": { $gte: 1990 } })
```

```
db.tv_shows.find({
  "year": { $gte: 1990 } },
  "imdb.rating": { $gte: 5 } }
)
```

```
db.tv_shows.find({
  "title": /bad/i },
  "imdb.rating": { $gte: 5 } }
)
```

Covered Queries

- Result can be returned entirely from the index

```
db.tv_shows.find(
   {
      "year": { $gte: 1990 }
   },
   {
      _id: 0,
      "imdb.rating": 1
   }
)
```

Text Searches

- MongoDB supports text searches rather than just matching with regular expression
  - Eg. language, accents, etc.
- `$text` - special operator for searching text
- Need a text index but only limited to 1 text index per collection
  - Text index are expensive

Create a text index on attribute `plot`

```
db.tv_shows.createIndex({
  plot: "text"
})
```

Text Searches

```
db.tv_shows.find({
   $text: {
      $search: "fIgHt"
   }
})
```

Search the `plot` attribute using text search.
Case insensitive

```
db.tv_shows.find({
   $text: {
      $search: "fight run",
      $caseSensitivity: true
   }
})
```

Treated as individual words viz. bag of words

Text Searches

```
TextCriteria textCriterial = TextCriteria.forDefaultLanguage()
   .matchingPhrase("fIgHt");

TextQuery query = TextQuery.queryText(textCriteria);

List<Document> results = mongoTemplate.find(query,
   Document.class, "tv_shows");
```

Use `TextCriteria` and `TextQuery` for text searches

```
TextCriteria textCriterial =
TextCriteria.forDefaultLanguage()
   .matchingPhrase("fight run")
   .caseSensitive(true);
```

Match the phrase viz. words appear in the listed order not bag of words

Case sensitive search

Text Searches

- `TextCriteria` has other methods for searching text
  - `matchingAny(String... words)` - match any of the given words, bag of words
  - `matchingPhrase(String phrase)` - match a phrase
  - `notMatchingAny(String... words)` - not matching any of the given words
  - `notMatchingPhrase(String phrase)` - do not match the phrase

```
List<String> matches = new LinkedList<>();
List<String> notMatch = new LinkedList<>();
for (String word: listOfWords.split("," )) {
   String w = word.trim();
   if (w.startsWith("-"))
      notMatch.add(w.substring(1));
   else
      matches.add(w);
}
TextCriteria textCriteria = TextCriteria.forDefaultLanguage()
   .matchingAny(matches)
   .notMatchingAny(notMatch);
```

Text Searches

```
db.tv_shows.find(
   {
      $text: { $search: "fight run" }
   },
   { score: {
         $meta: "textScore"
      }
   }
)
```

Adds a new attribute `score` to the search result. The `score` attribute holds the a value telling how well the document matches the search terms. The high the better

Text Searches

```
TextCriteria textCriterial =TextCriteria.forDefaultLanguage()
  .matchingPhrase("fIgHt");

TextQuery query = TextQuery.queryText(textCriteria)
  .sortByScore();
query.setScoreFieldName("score");

List<Document> results = mongoTemplate.find(query,
  Document.class, "tv_shows");
```

Sort the result by the text score

Add a new property in the result called `score` for the text score

Text scoring algorithm
https://ananya281294.medium.com/mongo-maths-676469e55f78

Unused

Performing Inserts

Select the database

Specify the collection

```
client.db('movies')
  .collection('inventory')
  .insert({
    item: 'game console',
    instock: [
      { warehouse: 'Ang Mo Kio', qty: 10 },
      { warehouse: 'Eunos', qty: 30 }
    ]
  })
  .then(result => {
    // result
  })
  .catch(error => {
    // handle error
  })
```

Document to be inserted

```
db.tv_shows.insert({
  item: 'game console',
  instock: [
    { warehouse: 'Ang Mo Kio', qty: 10 },
    { warehouse: 'Eunos', qty: 30 }
  ]
})
```

Result from Insert

Summary

```
{
  result: { ok: 1, n: 1 },
  ops: [
    {
      item: 'game console'
      instock: [ ... ],
      _id: abc123
    }
  ],
  insertedCount: 1
  insertedIds: [ '0': abc123 ]
}
```

Arrays containing list of inserted documents

Inserted document with the `_id` if non is provided

Number of inserted documents and list of `_id` index into `ops` array

Arrays - Primitives

```
{ ... genres: [ "Science-Fiction", "Drama", "Crime" ] },
{ ... genres: [ "Drama", "Crime" ] },
{ ... genres: [ "Thriller", "Drama", "Crime" ] },
```

```
db.inventory.find({ "genres": "Crime" })
```

Find any document with `Drama`

```
db.inventory.find({ "genres":
  { $in: [ "Science-Fiction", "Crime" ] })
```

Find any document with `Science-Fiction` or `Crime`

```
db.inventory.find({ "genres":
  { $all: [ "Science-Fiction", "Crime" ] })
```

Find any document with `Science-Fiction` and `Crime`

Arrays

```
{ item: "journal", instock: [
   { warehouse: “Ang Mo Kio", qty: 5 },
   { warehouse: "Clementi", qty: 15 } ]
},
{ item: "notebook", instock: [
   { warehouse: "Clementi", qty: 5 } ]
},
{ item: "paper", instock: [
   { warehouse: "Ang Mo Kio", qty: 60 },
   { warehouse: "Bedok", qty: 15 } ]
},
{ item: "planner", instock: [
   { warehouse: "Ang Mo Kio", qty: 40 },
   { warehouse: "Bedok", qty: 5 } ]
},
{ item: "postcard", instock: [
   { warehouse: "Bedok", qty: 15 },
   { warehouse: "Clementi", qty: 35 } ]
}
```

Sample from https://docs.mongodb.com/manual/tutorial/query-array-of-documents/

Arrays - Embedded Document

Will find documents filtered by `warehouse` and `qty`
But the 2 conditions applies to any document in the array

```
db.inventory.find({
   $and: [
      { "instock.warehouse": "Ang Mo Kio" },
      { "instock.qty": { $gte: 15 } }
   ]
)
```

Addressing an embedded document

```
db.inventory.find({
   instock: {
      $elemMatch: {
         warehouse: "Ang Mo Kio",
         qty: { $gte: 5 }
      }
   }
})
```

```
{ item: "journal", instock: [
   { warehouse: "Ang Mo Kio", qty: 5 },
   { warehouse: "Clementi", qty: 15 } ]
}
```

```
{ item: "paper", instock: [
   { warehouse: "Ang Mo Kio", qty: 60 },
   { warehouse: "Bedok", qty: 15 } ]
}
```

Updates

```
db.inventory.update(
   { "instock.warehouse": "Ang Mo Kio" },
   {
      $inc: {
         "instock.$[].qty": 10
      }
   }
)
```

Every embedded document in the array increment the `qty` field by 10

```
db.inventory.update(
   { "_id": ObjectId("abc123") },
   {
      $set: {
         "instock.$[1].qty": 10
      }
   }
)
```

Set the second document's `qty` attribute in the instock array to 10

Updates

```
db.inventory.update(
   { "_id": ObjectId("abc123") },
   {
      $push: {
         instock: { warehouse: "Eunos", qty: 30 }
      }
   }
)
```

Append the new document to the array `instock`

```
db.inventory.update(
   { "_id": ObjectId("abc123") },
   {
      $pop: { instock: -1 }
   }
)
```

Remove an element from the front of an array. 1 is from the back

Updates

```
db.inventory.update(
  { "instock.warehouse": "Ang Mo Kio" },
  {
    $set: {
      "instock.$[elem].qty": 0}
    }
  },
  {
    arrayFilters: [
      { "elem.qty": { $lte: -1 }
    ]
  }
)
```

`elem` acts like a control variable in a for loop

For every matched document, set the `instock.qty` to zero if the `instock.qty` is <= -1