Distributed Data Systems

MAHESH CHAUDHARI, PH.D

Announcement

Due Dates:

- Group Project Phase-1 Due (Feb 7)
- Programming Assignment 1 - Feb 12 Midnight

Quiz-1

- Quiz 1 - Feb 20 at 8:50 AM Rooms: 154-156 On Canvas (No remote exam allowed without prior authorization)
  - Multiple Choice
  - Programming
  - Study Guide:
    - Slides (Material covered till Feb 12), practice programming, Programming Assignment-1
    - In-class practice
    - Airflow, NoSQL, MongoDB

1 A4 size page cheat sheet

- 1-side only Hand-written
- Not a common one between friends and colleagues
- Submit it along with the quiz-1 paper

Contents

MongoDB
- Supported Data Types
- Operations
  - Create
  - Read
  - Update
  - Delete

Contents

MongoDB

- Supported Data Types
- Operations
  - Create
  - Read
  - Update
  - Delete

MongoDB

Terminology

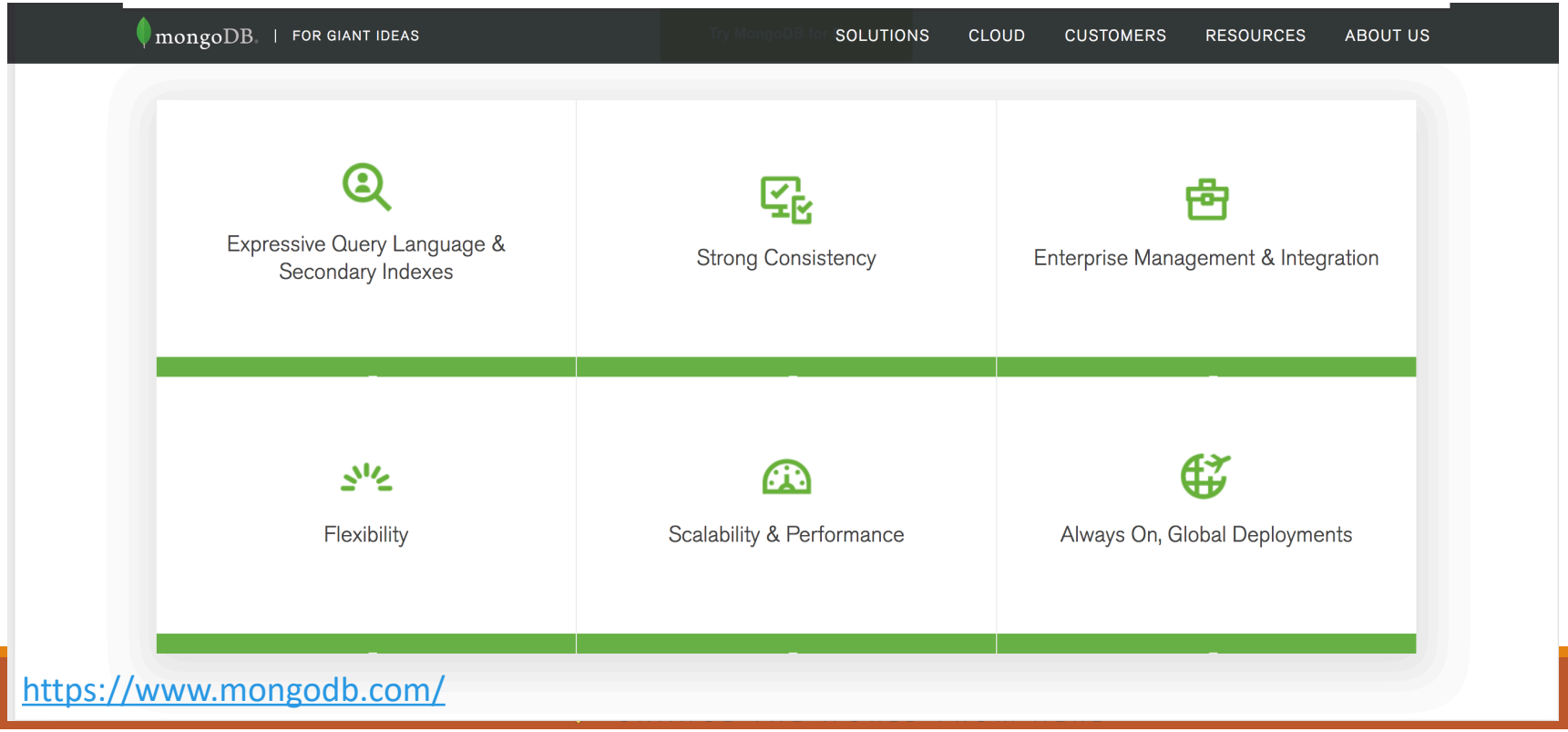
- Documents
- Collections
- Databases

SQL (RDBMS)	MongoDB
database	database
table	collection
row	document
column	field

https://docs.mongodb.com/manual/reference/sql-comparison/

MongoDB

Why MongoDB?



https://www.mongodb.com/

MongoDB 

Why MongoDB?

- Easy of use
  - No defined schema.
- Easy scaling
  - MongoDB takes care of
    - Loading data across a cluster.
    - Redistributing documents automatically.
    - Balancing data.
    - Routing user request to the correct machines.

MongoDB

Why MongoDB?

- Many features
  - Creating, reading, updating, and deleting (CRUD) data.
  - Indexing : Supports secondary indexes, allowing fast queries.
  - Aggregation pipeline : Allow you to build complex aggregations from simple pieces.
  - Special collection types : Time-to-live collections (session), fixed-size collections.
  - File storage : Stores large files and file meta data.
- Supported drivers
  - C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go, Swift, etc.

https://docs.mongodb.com/ecosystem/drivers/

Contents

MongoDB

- **Supported Data Types**
- Operations
  - Create
  - Read
  - Update
  - Delete

MongoDB

Commonly Supported Data Types

- Null- `null`
- Boolean – `true`, `false`
- Number – 64 bit float (default), `NumberInt()` (4-byte), `NumberLong()` (8 byte)
- String – UTF-8 characters
- Date – Stored as millisecond since the epoch (No time zone). - `new Date()`
- Regular expression – Use JavaScript's regular expression syntax.
- Array – `[Element1, Element2,…]`
- Embedded document – Entire documents can be embedded as value.
- Object ID – 12-Byte ID for documents. "`_id`"
- Binary data – String of arbitrary bytes.
- Code – JavaScript code.

*Examples are included in day4_examples.js.

MongoDB

Commonly Supported Data Types

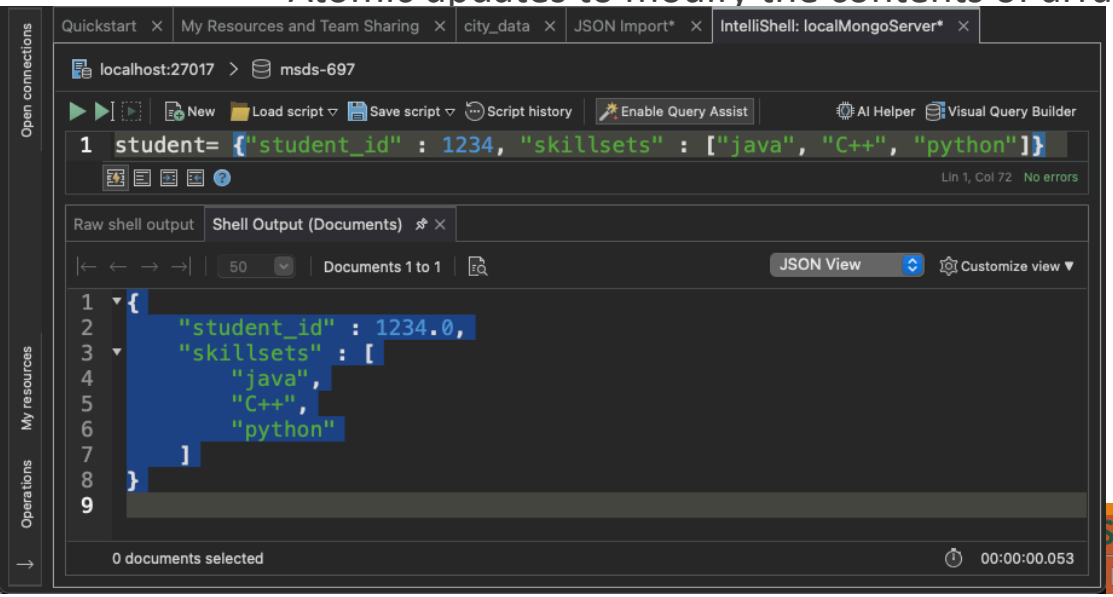
- Null- `null`
- Boolean – `true`, `false`
- Number – 64 bit float (default), `NumberInt()` (4-byte), `NumberLong()` (8 byte)
- String – UTF-8 characters
- Date – Stored as millisecond since the epoch (No time zone). `new Date()`
- Regular expression – Use JavaScript's regular expression syntax.
- Array – `[Element1, Element2,…]`
- Embedded document – Entire documents can be embedded as value.
- Object ID – 12-Byte ID for documents. "`_id`"
- Binary data – String of arbitrary bytes.
- Code – JavaScript code.

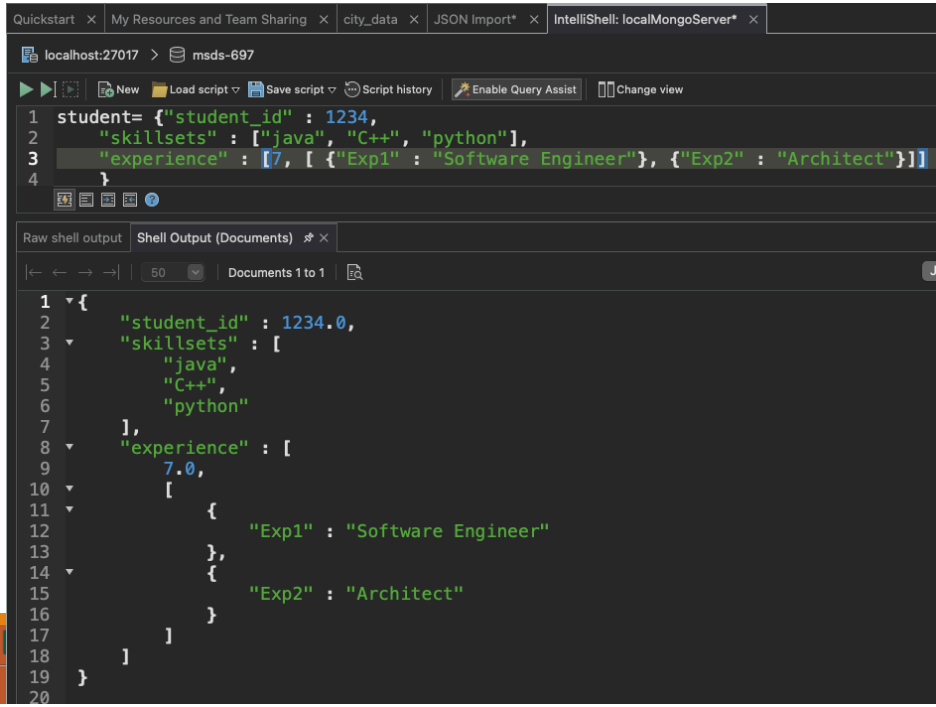
*Examples are included in day4_examples.js.

MongoDB

Commonly Supported Data Types

- Array – `[Element1, Element2,…]`
  - Used for ordered operations and unordered operations.
  - Can contain different data types.
  - Atomic updates to modify the contents of arrays.





MongoDB

Commonly Supported Data Types

- Null- `null`
- Boolean – `true`, `false`
- Number – 64 bit float (default), `NumberInt()` (4-byte), `NumberLong()` (8 byte)
- String – UTF-8 characters
- Date – Stored as millisecond since the epoch (No time zone). `new Date()`
- Regular expression – Use JavaScript's regular expression syntax.
- Array – `[Element1, Element2,…]`
- Embedded document – Entire documents can be embedded as value.
- Object ID – 12-Byte ID for documents. "`_id`"
- Binary data – String of arbitrary bytes.
- Code – JavaScript code.

*Examples are included in day4_examples.js.

MongoDB

Commonly Supported Data Types

- Embedded document – Entire documents can be embedded as value.

```
1  student= {"student_id" : 1234, "name" : "Jane Smith",
2      "address" : {"street" : "101 Howard St.", "city" : "San Francisco",
3      "state" : "CA"}
4      }
```

Raw shell output | Shell Output (Documents) | 50 | Documents 1 to 1 | JSON View

```
1  {
2      "student_id" : 1234.0,
3      "name" : "Jane Smith",
4      "address" : {
5          "street" : "101 Howard St.",
6          "city" : "San Francisco",
7          "state" : "CA"
8      }
9  }
10
```

*Examples are included in day4_examples.js.

MongoDB

Commonly Supported Data Types

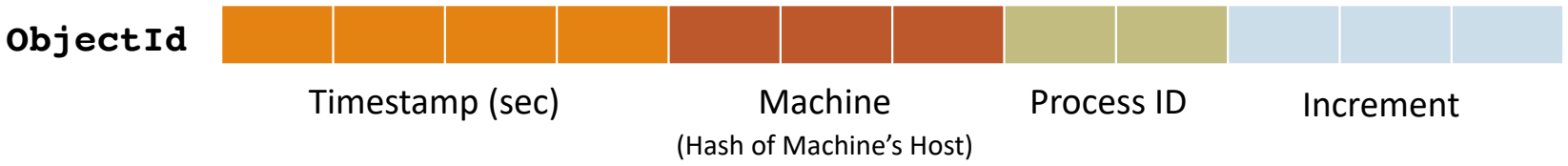
- Null- `null`
- Boolean – `true`, `false`
- Number – 64 bit float (default), `NumberInt()` (4-byte), `NumberLong()` (8 byte)
- String – UTF-8 characters
- Date – Stored as millisecond since the epoch (No time zone). `new Date()`
- Regular expression – Use JavaScript's regular expression syntax.
- Array – `[Element1, Element2,…]`
- Embedded document – Entire documents can be embedded as value.
- Object ID – 12-Byte ID for documents. "`_id`"
- Binary data – String of arbitrary bytes.
- Code – JavaScript code.

*Examples are included in day4_examples.js.

MongoDB

Commonly Supported Data Types

- Object ID – 12-byte ID for documents and a **default for "`_id`"**.
  - Generated by the driver on the client.
  - Every document in MongoDB must have an "`_id`" key.
    - Its value can be any type, but needs to be unique for a collection.
    - Default is `ObjectId`.



*Examples are included in day4_examples.js.

MongoDB

Commonly Supported Data Types

- Object ID – 12-byte ID for documents and a **default for "`_id`"**.
  - Generated by the driver on the client.
  - Every document in MongoDB must have an "`_id`" key.
    - Its value can be any type, but needs to be unique for a collection.
    - Default is `ObjectId`.

```
ObjectId()

xt
ObjectId("5c242885997ba4facbd85675")
```

*Examples are included in day4_examples.js.

Example Data

Data 1 - Friends

- We will create documents and insert to “Friends”

Data2 - Business

- Import data from business.json that includes business information in New York.

Contents

MongoDB

- Supported Data Types
- **Operations**
  - Create
  - Read
  - Update
  - Delete

MongoDB

CRUD Overview

- **Create (Insert)**

Operator	About	Example
`use databse_name`	Create/Choose a database	`use msds697`
`db.createCollection(name, options)`	Create a collection	`db.createCollection('friends')`
`db.collection_name`	Access collection from the db variable.	`db.friends`
`db.collection_name.insertOne(document)`	Add a document to a collection	`db.friends.insertOne(mahesh)`
`db.collection_name.insertMany([doc1, doc2])`	Bulk Insert : Takes an array of document	`db.friends.insertMany([diane, yannet, shan, robert])`
`$mongoimport`	Import raw data	`$mongoimport --db msds697 --collection business --file ../Data/business.json`

MongoDB

CRUD Overview

- **Read**

Operator	About	Example
`db.collection_name.find(query, projection)`	Select all documents in the collection satisfying query and projection.	`db.friends.find({"address.city":"San Francisco"})`
`db.collection_name.findOne(query, projection)`	same as find() method with a limit of 1.	`db.business.findOne()`

- **U**pdate

Operator	About	Example
`db.collection_name.updateOne(filter, update)`	Updates a single document within the collection based on the filter.	`db.friends.updateOne({"first_name":"Jane"}, {$set: {"title":"Associate Professor"}})`
`db.collection_name.updateMany(filter, update)`	Updates all documents that match the specified filter for a collection.	`db.friends.updateMany({"address.state":"CA"}, {$set: {"address.country":"USA"}})`

https://www.mongodb.com/docs/manual/crud/

MongoDB

CRUD Overview

- **U**pdate

Operator	About	Example
`db.collection_name.replaceOne (filter, update)`	Replaces a single document within the collection based on the filter.	`db.friends.replaceOne({"first_name":"Jane" }, {"title":"Associate Professor"})`

This is different from updateOne because this command will replace the entire document.

To safe-guard against accidental inserts, newer mongodb needs the flag {upsert : true}

updateOne updates the existing fields.

https://docs.mongodb.com/manual/reference/method/db.collection.update/

MongoDB

CRUD Overview

- **D**elete

Operator	About	Example
`db.collection_name.deleteMany(filter)`	Removes all documents that match the filter from a collection.	`db.friend.deleteMany({"address.city":"San Francisco"})`
`db.collection_name.deleteOne(filter)`	Removes a single document from a collection.	`db.friends.deleteOne({"address.city":"San Francisco"})`
`db.collection_name.drop()`	Delete entire collection efficiently.	`db.friend.drop()`

Contents

MongoDB
- Supported Data Types
- Operations
  - **Create**
  - Read
  - Update
  - Delete

MongoDB

CRUD Overview

- Create (Insert)
  1. Create/Choose a database
     - `use database_name`
  2. Check which database you're using.
     - `db`
  3. Create Collection
     - `db.createCollection(name, options)`
  4. Access collection from the db variable.
     - `db.collection_name`

MongoDB

CRUD Overview

- Create (Insert)

5. `insert()` : Add a document to a collection.
   - `db.collection_name.insertOne(document)`

6. `insertMany()` : Bulk Insert
   - Takes an **array** of document.
   - `db.collection_name.insertMany([document1, document2])`
   - ex. `db.friend.insertMany([diane, yannet])`

Example 1

Create a database called "msds697" and create a collection called “friends".

Create and insert document called “mahesh" and to "friends".

Create and insert document called "diane" and to "friends".

Create multiple documents called "yannet" and “shan”, and insert to "friends".

```
mahesh = {"name" : “Mahesh Chaudhari”, "title" : “Assistant Professor”,
        "address" : {"street" : "101 Howard St", "city" : "San Francisco",
        "state" : “CA”}}
diane = {"name" : "Diane MK Woodbridge",
        "address" : {"street" : "101 Howard St", "city" : "San Francisco",
        "state" : “CA”}}
yannet = {"name":"Yannet Interian",
         "address" : {"street" : "101 Howard", "city" : "San Francisco",
         "state" : "CA"}}
shan = {"name":"Shan Wang"}
```

Example 1

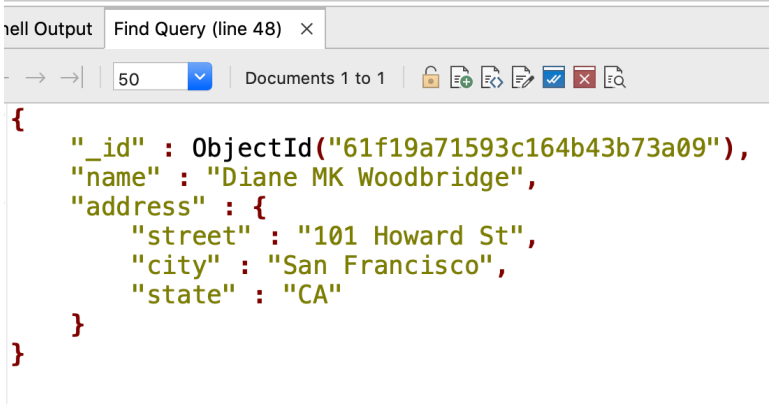
Create a database called "msds697" and create a collection called "friends".

Create and insert document called "diane" and to “friends".

Create multiple documents called "yannet" and “shan”, and insert to "friends".

```
// Create/Insert
use msds697
db.friends.drop()
db.createCollection('friends')

db.friends.insert(diane)
db.friends.find()
```



Example 1

Create a database called "msds697" and create a collection called "friends".

Create and insert document called "diane" and to “friends".

Create multiple documents called "yannet" and “shan”, and insert to "friends".

```
//Bulk Insert
db.friends.insert([yannet, shan])
db.friends.find()
```

hell Output | Find Query (line 53) ×

50 | Documents 1 to 3


```
{
    "_id" : ObjectId("61f19bd1593c164b43b73a13"),
    "name" : "Diane MK Woodbridge",
    "address" : {
        "street" : "101 Howard St",
        "city" : "San Francisco",
        "state" : "CA"
    }
}
{
    "_id" : ObjectId("61f19bea593c164b43b73a16"),
    "name" : "Yannet Interian",
    "address" : {
        "street" : "101 Howard",
        "city" : "San Francisco",
        "state" : "CA"
    }
}
{
    "_id" : ObjectId("61f19bea593c164b43b73a17"),
    "name" : "Shan Wang",
    "address" : {
```

MongoDB

CRUD Overview

- Create (Insert)
  7. Import raw data
     - **`mongoimport`** on terminal
       - Imports content from an Extended JSON, CSV, or TSV export created by mongoexport, or potentially, another third-party export tool.
       - `Refer to the slides DistributedDataSystems_MongoDB_Installation on canvas.`

https://docs.mongodb.com/manual/reference/program/mongoimport/

MongoDB

CRUD Overview

- Create (Insert)
  7. Import raw data
     - **`mongoimport`** on terminal
       - Options
         - --db : The name of the database on which to run the mongoimport.
         - --collection : The collection to import.
         - --file : The location and name of a file containing the data to import.
         - --mode : insert, upsert
           - insert (Default) : Allows to import a document that contains a duplicate value for a field with a unique index, such as _id.
           - upsert : Replace existing documents with matching documents from the import file and insert all other documents.
         - etc.

https://docs.mongodb.com/manual/reference/program/mongoimport/
https://docs.mongodb.com/manual/reference/program/mongoimport/#ex-mongoimport-merge

Example 2

Import “business.json” to the `msds697` database’s `business` collection.

Example 2

Import “business.json” to the `msds697` database’s `business` collection.

On terminal (Not mongo shell !)

```
(DistributedSystems) mchaudhari@ML-ITS-903658 data % mongoimport --db msds697 --collection business --file business.json
2024-02-01T21:15:44.849-0800    connected to: mongodb://localhost/
2024-02-01T21:15:45.426-0800    25359 document(s) imported successfully. 0 document(s) failed to import.
(DistributedSystems) mchaudhari@ML-ITS-903658 data %
```

Example 2

Import “business.json” to the `msds697` database’s `business` collection.

```
use msds697
db

db.business.findOne()
```

Text | Text | Document

50 | Documents 1 to 1 | JSON View


```
{
    "_id" : ObjectId("5c254de5691742b0f05b74ba"),
    "address" : {
        "building" : "8825",
        "coord" : [
            -73.8803827,
            40.7643124
        ],
        "street" : "Astoria Boulevard",
        "zipcode" : "11369"
    },
    "borough" : "Queens",
    "cuisine" : "American ",
    "grades" : [
        {
            "date" : ISODate("2014-11-15T00:00:00.000+0000"),
            "grade" : "Z",
            "score" : 38.0
        },
        {
            "date" : ISODate("2014-05-02T00:00:00.000+0000"),
            "grade" : "A",
            "score" : 10.0
        },
        {
```

Contents

MongoDB
- Supported Data Types
- Operations
  - Create
  - **Read**
  - Update
  - Delete

MongoDB

CRUD Overview

- Read
  1. `db.collection_name.find(query, projection)`
     - Select all documents in the collection satisfying query and projection.
  2. `db.collection_name.findOne(query, projection)`
     - Same as find() method with a limit of 1.

  - Parameters
    - **`query`** - Specifies selection filter ({key : value}) using query operators. To return all documents in a collection, omit this parameter or pass an empty document ({}).
    - **`projection`** - Specifies the fields to return in the documents that match the query filter.
      - ex. {key : true/false}
    - *Will cover more about query and projection operators in the next class.*

https://docs.mongodb.com/manual/reference/method/db.collection.find/
https://docs.mongodb.com/manual/reference/method/db.collection.findOne/

Example 3

In `msds697`,

- Find all documents where address's city is San Francisco.
- Find one document where address's city is San Francisco.

Example 3

In `msds697` and in the `friends` collection,

- Find all documents where address's city is San Francisco.
- Find one document where address's city is San Francisco.

```
25
26 db.friends.find({"address.city" : "San Francisco"})
27
```

Raw shell output | Find Query (line 14)

50 | Documents 1 to 2

```
1  {
2      "_id" : ObjectId("65bbfc721c54d6fd80aa4d63"),
3      "first_name" : "Jane",
4      "last_name" : "Smith",
5      "address" : {
6          "street" : "101 Howard St.",
7          "city" : "San Francisco",
8          "state" : "CA",
9          "country" : "USA"
10     },
11     "title" : "Associate Professor"
12 }
```

1 document selected

Example 4

Find all businesses in "Manhattan" in "business" under the “msds697” database.

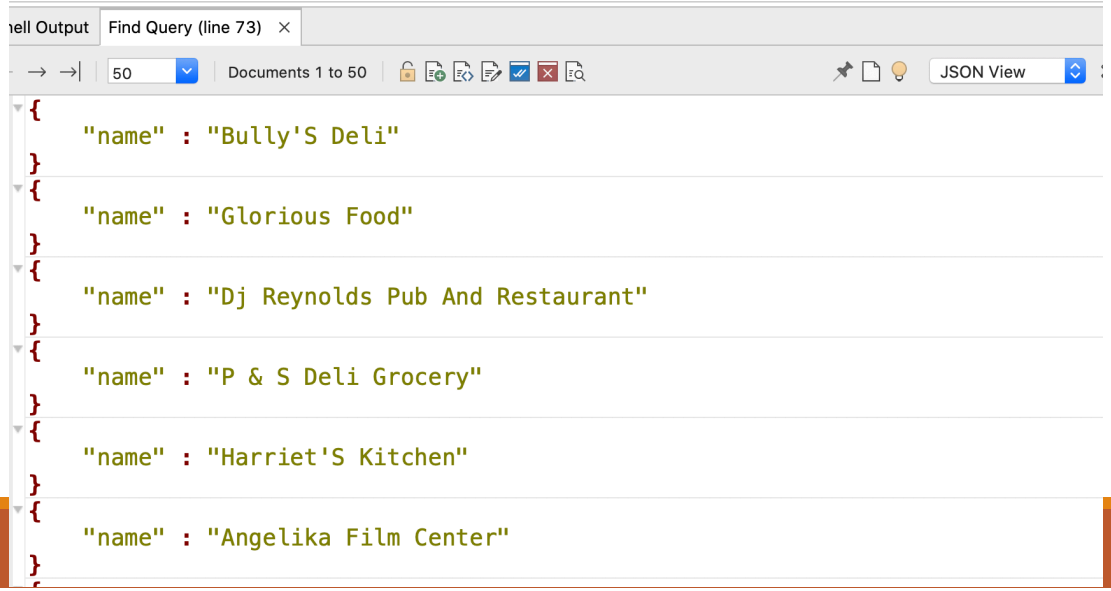
- Only business names?

Example 4

Find all businesses in "Manhattan" in "business" under the “msds697” database.

- Only business names?

```
// Example 4
//Find all businesses in "Manhattan" in "business" under the “msds697” database
db.business.find({"borough":"Manhattan"})
//Only business names?
db.business.find({"borough":"Manhattan"},{"name":true,"_id":false})
```



Contents

MongoDB
- Supported Data Types
- Operations
  - Create
  - Read
  - **Update**
  - Delete

MongoDB

CRUD Overview

- Update
  - Update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.
    1. `db.collection_name.updateOne(filter, update, options)`
       - Updates a single document within the collection based on the filter.
    2. `db.collection_name.updateMany(filter, update, options)`
       - Updates all documents that match the specified filter for a collection.

  - Arguments
    - Required
      - **`filter`**: Criteria to filter.
      - **`update`**: Change criteria including update operators.
    - Optional
      - **`upsert`** : When no document meets the Query criteria, it creates (inserts) a new document (default = false).

https://docs.mongodb.com/manual/reference/method/db.collection.update/

MongoDB

CRUD Overview

- Update
  - In order to update certain portions of a document,
    - Use update modifier on **update** field.
      - **Field modifier** - `$set, $unset, $inc, $rename`
      - Array modifier - `$push, $pop, $pull, $`

Field modifier	Description	Example
$set	Sets the value of a field in a document.	`db.friends.updateOne({"name":"Diane"}, {$set:{"title":"Dr"}})`
$unset	Removes the specified field from a document.	`db.friends.updateOne({"name":"Diane"}, {$unset:{"title":""}})`
$inc	Increments the value of the field by the specified amount.	`db.friends.updateOne("name" : "Diane"}, {$inc:{"working" : 1}})`
$rename	Renames a field.	`db.friends.updateMany({}, {$rename : {“working”:"yearsAtWork"}})`

More field modifiers: https://docs.mongodb.com/manual/reference/operator/update/

MongoDB

CRUD Overview

- Update
  - Field modifier
    - **`$set`** : replaces the value of a field with the specified value.
      - ex. { $set: { <field1>: <value1>, ... } }
    - **`$unset`** : delete a particular field.
      - ex. { $unset: { <field1>: "", ... } }

https://docs.mongodb.com/manual/reference/operator/update/set/
https://docs.mongodb.com/manual/reference/operator/update/unset/

Example 5

Set "title" as "Assistant Professor", to all the documents where "name" is set as “Jane Smith”.

Unset "title", to all the documents where "name" is set as "Diane MK Woodbridge”.

Set "title" as "Administrative Director", to all the documents where "name" is set as "Kirsten Keihl”.

- If there is no corresponding document, create one.

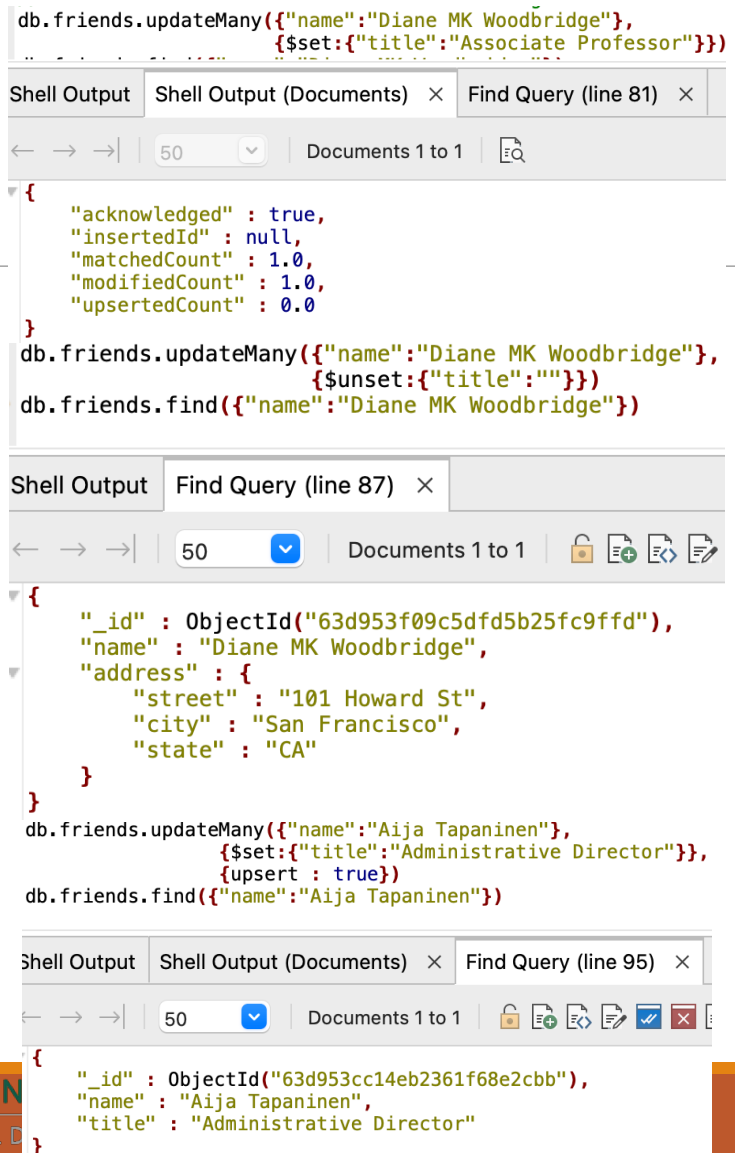
Example 5

Set "title" as "Associate Professor", to all the documents where "name" is set as "Diane MK Woodbridge".

Unset "title", to all the documents where "name" is set as "Diane MK Woodbridge".

Set "title" as "Administrative Director", to all the documents where "name" is set as "Aija Tapaninen".

- If there is no corresponding document, create one.



MongoDB

CRUD Overview

- Update
  - Field modifier
    - **`$inc`** : increments a field by a specified value.
      - If the field does not exist, $inc creates the field and sets the field to the specified value.
      - Useful for updating votes, scores, etc.
      - ex. { $inc: { <field1>: <amount1>, <field2>: <amount2>, ... } }
    - **`$rename`**: updates the name of a field
      - ex. {$rename: { <field1>: <newName1>, <field2>: <newName2>, ... } }

https://docs.mongodb.com/manual/reference/operator/update/inc/
https://docs.mongodb.com/manual/reference/operator/update/rename/

Example 6

Increase "kidsCount" by 1 for all documents, where "name" is "Shan Wang".

Rename "address" field to "officeAddress" for all the documents.

Example 6

Increase "kidsCount" by 1 for all documents, where "name" is "Shan Wang".

```
db.friends.updateMany({"name" : "Shan Wang"}, {$inc:{"kidsCount" : 1}})
```

```
ell Output | Shell Output (Documents) ×
50 | Documents 1 to 1 | Pin Result | JSON Vi
{
    "acknowledged" : true,
    "matchedCount" : 1.0,
    "modifiedCount" : 1.0
}
```

Rename "address" field to "officeAddress" for all the documents.

```
db.friends.updateMany({},{$rename : {"address":"officeAddress"}})
db.friends.find()
```

```
ell Output | Shell Output (Documents) ×
50 | Documents 1 to 1 | Pin Result
{
    "acknowledged" : true,
    "matchedCount" : 4.0,
    "modifiedCount" : 2.0
}
```

MongoDB

CRUD Overview

- Update
  - Field modifier
    - **`$min`**: Updates the value of the field to a specified value *if* the specified value is **less than** the current value of the field.
      - If the field does not exist, the $min operator sets the field to the specified value.
      - ex. { $min: { <field1>: <value1>, ... } }
    - **`$max`**: Updates the field to a specified field if the specified value is **greater than** the existing current field value.
      - If the field does not exists, the $max operator sets the field to the specified value.
      - ex. { $max: { <field1>: <value1>, ... } }

https://docs.mongodb.com/manual/reference/operator/update/min/
https://docs.mongodb.com/manual/reference/operator/update/max/

Example 7

For documents where "name" is "Diane MK Woodbridge", set "numCats" to 1 if it is either not set or the existing value larger than 1.

For documents where "name" is "Diane MK Woodbridge", set "numDogs" to 1 if it is either not set or the existing value is smaller than 1.

- What happens if you try to update it to {$max : {“numDogs”:0}}, after the query above?

Example 7

For documents where "name" is "Diane MK Woodbridge", set "numCats" to 1 if it is either not set or the existing value larger than 1.

For documents where "name" is "Diane MK Woodbridge", set "numDogs" to 1 if it is either not set or the existing value is smaller than 1.

- What happens if you try to update to {$max : {“numDogs”:0}}, after the query above?

```
db.friends.updateMany({"name":"Diane MK Woodbridge"},{$min : {"numCats":1}})
db.friends.updateMany({"name":"Diane MK Woodbridge"},{$max : {"numDogs":1}})
db.friends.find({"name" : "Diane MK Woodbridge"})
```

ell Output | Find Query (line 119) ×

50 | Documents 1 to 1 | JSON View


```
{
    "_id" : ObjectId("61f1f0e3593c164b43b73a2a"),
    "name" : "Diane MK Woodbridge",
    "officeAddress" : {
        "street" : "101 Howard St",
        "city" : "San Francisco",
        "state" : "CA"
    },
    "numCats" : 1.0,
    "numDogs" : 1.0
}
```

MongoDB

CRUD Overview

- Update
  - In order to update certain portions of a document,
    - Use update modifier on **update** field.
      - Field modifier - `$set, $unset, $inc, $rename`
      - **Array modifier** - `$push, $pop, $pull, $`

Array modifier	Description	Example
$push	Adds an item to an array.	`db.business.updateMany({"name":"White Castle"}, {$push:{"grades":{"date":new Date(),"grade":"A"}}})`
$pop	Removes the first or last item of an array.	`db.business.updateMany({"name":"White Castle"}, {$pop:{"grades":1})`
$pull	Removes all array elements that match a specified query.	`db.business.updateMany({"name":"White Castle"}, {$pull:{"grades":{"grade":"C"}}})`
$	Acts as a placeholder to update the first element that matches the query condition.	More details and examples in page 56.

More array modifiers: https://docs.mongodb.com/manual/reference/operator/update-array/

MongoDB

CRUD Overview

- Update
  - Array modifier
    - Operators
      - **`$push`** : Add elements to the end of an array.
        - `{$push : {<field1> : <value1>, ...}}`
      - **`$pop`** : Removes the first or last element of an array.
        - `{$pop : {<field> : <–1|1>, ...}}`
          - -1 - remove the first element of an array.
          - 1 - remove the last element in an array.
      - **`$pull`** : removes from an existing array **all** instances of a value or values that match a specified condition.
        - `{$pull : {criteria}}`

https://docs.mongodb.com/manual/reference/operator/update/

Example 7

In the "business" collection

- Insert a new grades with "date" : today, "grade" : "A", and “score”: 9, for "White Castle" on "Pennsylvania Avenue"
  - **`new Date()`** returns the current date as a Date object.
- Remove the last grades that we just entered, for "White Castle" on "Pennsylvania Avenue".
- Remove all reviews with Cs for restaurant_id, 40364467 .

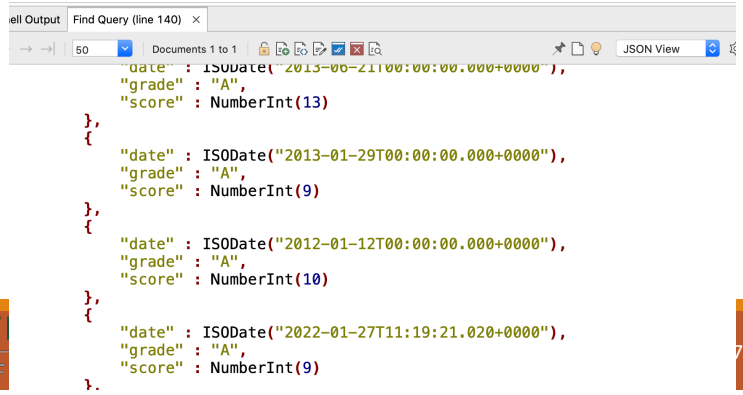
https://docs.mongodb.com/manual/reference/method/Date/

Example 7

In the "business" collection

- Insert a new grades with "date" : today, "grade" : "A", and “score”:9, for "White Castle" on "Pennsylvania Avenue"
  - **`new Date()`** returns the current date as a Date object.

```
db.business.find({"name":"White Castle", "address.street":"Pennsylvania Avenue"}
db.business.updateOne({"name":"White Castle" ,"address.street":"Pennsylvania Ave
                  {$push : {"grades": {
                                        "date" : new Date() ,
                                        "grade" : "A",
                                        "score" : NumberInt(9)
                                       }
                            }
                  })
db.business.find({"name":"White Castle", "address.street":"Pennsylvania Avenue"}
```



https://docs.mongodb.com/manual/reference/method/Date/

Example 7

In the "business" collection

- Remove the last grades that we just entered, for "White Castle" on "Pennsylvania Avenue".
- Remove all reviews with Cs for restaurant_id, 40364467 .

```
db.business.updateMany({"name":"White Castle" ,
                        "address.street":"Pennsylvania Avenue"},
                       {$pop:{"grades":1}})

db.business.update({"restaurant_id" : "40364467"},
                   {$pull:{"grades":{"grade":"C"}}})
db.business.find({"restaurant_id" : "40364467"}) // Zero Cs
```

https://docs.mongodb.com/manual/reference/method/Date/

MongoDB

CRUD Overview

- Update
  - Array modifier
    - $: Positional $ operator identifies an element in an array to update without explicitly specifying the position of the element in the array.
      - It figures out which element of the array the query matched and can be used to update that element.

Positional operator	Description	Example
$	Acts as a placeholder to update the **first element** that matches the query condition.	`db.business.updateMany({"grades.score" : 10}, {$set:{"grades.$.score": 9}})`
$[]	Acts as a placeholder to update **all elements** in an array for the documents that match the query condition.	`db.business.updateMany({"grades.score" : 10}, {$set:{“grades.$[].score”: 9}})`
$[<identifier>]	Acts as a placeholder to update **all elements that match the arrayFilters condition** for the documents that match the query condition.	`db.business.updateMany({“restaurant_id":"4035" },{$set:{"grades.$[element].score":11}}, {arrayFilters:[{"element.score":10}]})`

https://docs.mongodb.com/manual/reference/operator/update-array/

MongoDB

Syntax for $[<identifier>]:

```
db.collection.updateMany({ <query> },
                         { <update operator>: { "<array>.$[<identifier>]" : value } },
                         { arrayFilters: [ { <identifier>: <condition> } ] })
```

CRUD Overview

- Update
  - Array modifier
    - $: Positional $ operator identifies an element in an array to update without explicitly specifying the position of the element in the array.
      - It figures out which element of the array the query matched and can be used to update that element.

Positional operator	Description	Example
$	Acts as a placeholder to update the **first element** that matches the query condition.	`db.business.updateMany({"grades.score" : 10}, {$set:{"grades.$.score": 9}})`
$[]	Acts as a placeholder to update **all elements** in an array for the documents that match the query condition.	`db.business.updateMany({"grades.score" : 10}, {$set:{“grades.$[].score”: 9}})`
$[<identifier>]	Acts as a placeholder to update **all elements that match the arrayFilters condition** for the documents that match the query condition.	`db.business.updateMany({“restaurant_id":"4035" },{$set:{"grades.$[element].score":11}}, {arrayFilters:[{"element.score":10}]})`

https://docs.mongodb.com/manual/reference/operator/update-array/

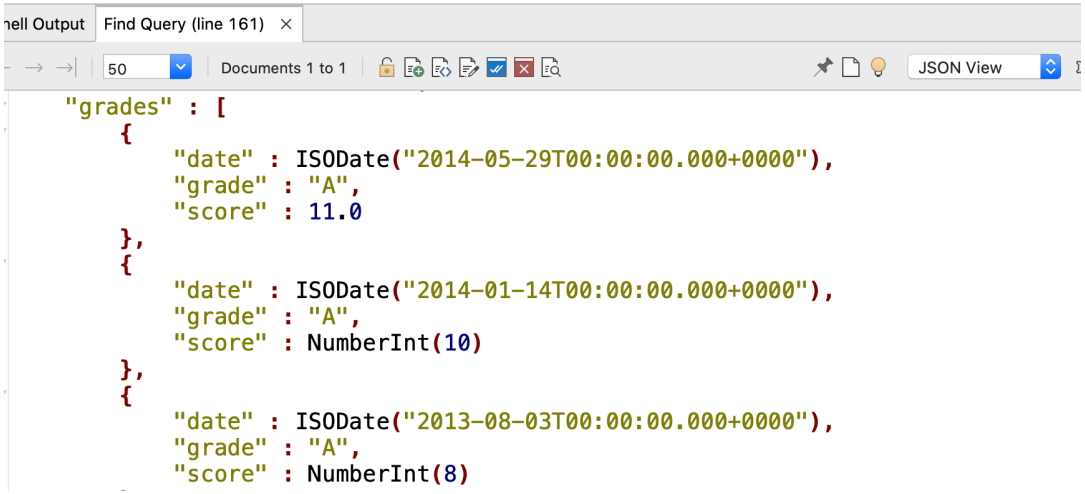
MongoDB

CRUD Overview

- Update
  - Array modifier $: Positional $ operator identifies an element in an array to update without explicitly specifying the position of the element in the array.
    - It figures out which element of the array the query matched and can be used to update that element.
    - $ updates the first element.
    - $[] updates all elements.
    - $[<identifier>] update all elements that match the arrayFilters condition.

$

```
db.business.find({"restaurant_id": "40356483"}) // has total 6 grades and 3 gra
db.business.updateMany({"restaurant_id": "40356483", "grades.score" : 10},
                       {$set:{"grades.$.score": 11}}) //One element is changed
db.business.find({"restaurant_id": "40356483"})
```



See Example 8 .

https://docs.mongodb.com/manual/reference/operator/update-array/

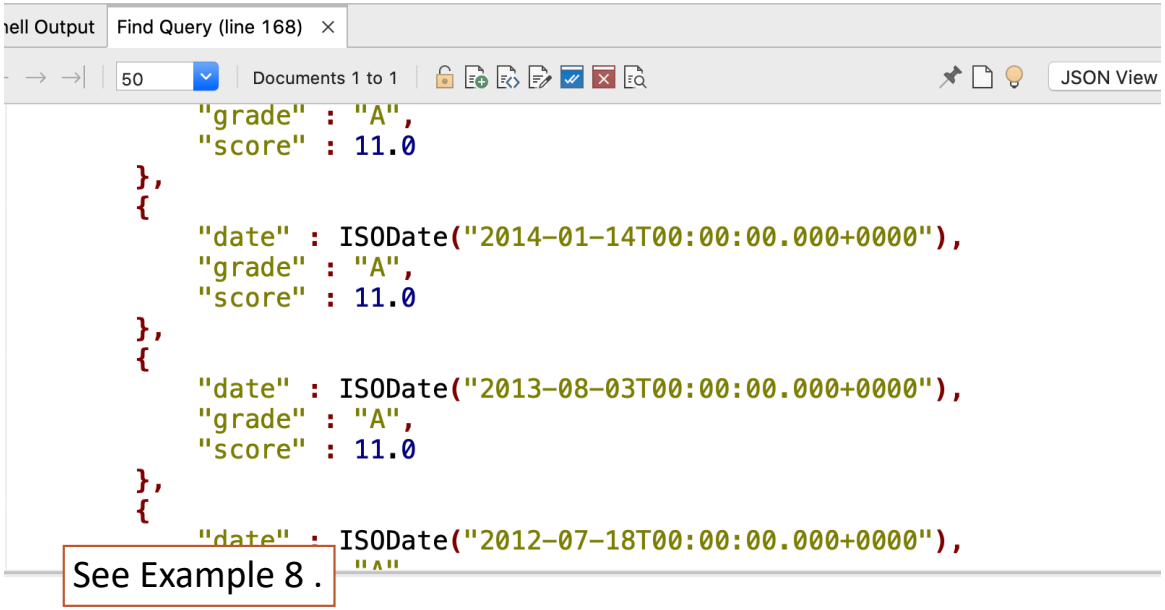
MongoDB

CRUD Overview

- Update
  - Array modifier $: Positional $ operator identifies an element in an array to update without explicitly specifying the position of the element in the array.
    - It figures out which element of the array the query matched and can be used to update that element.
    - $ updates the first element.
    - $[] updates all elements.
    - $[<identifier>] update all elements that match the arrayFilters condition.

$[ ]

```
db.business.updateMany({"restaurant_id": "40356483", "grades.score" : 10},
          {$set:{"grades.$[].score": 11}})
db.business.find({"restaurant_id": "40356483"}) // Change all 6
```



See Example 8 .

https://docs.mongodb.com/manual/reference/operator/update-array/

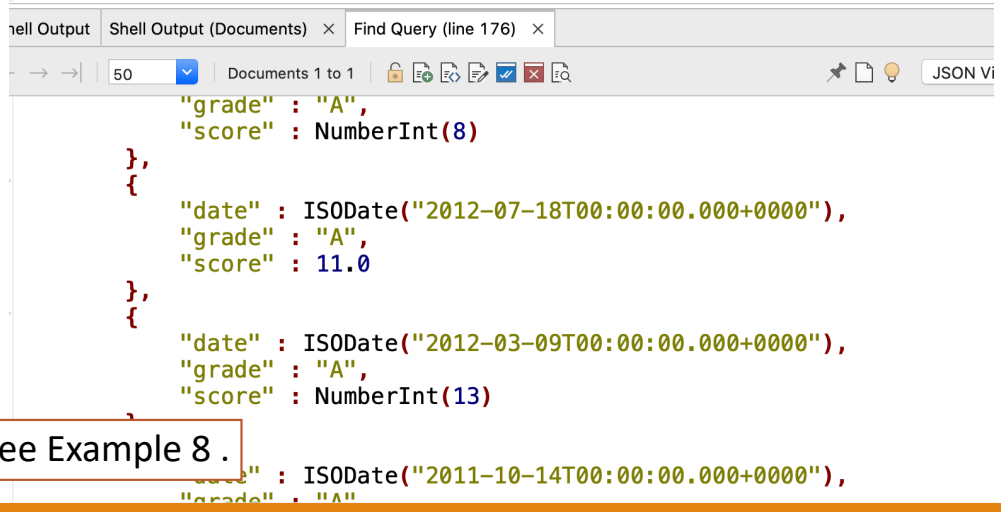
MongoDB

CRUD Overview

- Update
  - Array modifier $: Positional $ operator identifies an element in an array to update without explicitly specifying the position of the element in the array.
    - It figures out which element of the array the query matched and can be used to update that element.
    - $ updates the first element.
    - $[] updates all elements.
    - $[<identifier>] update all elements that match the arrayFilters condition.

$[<identifier>]

```
//$[<identifier>]
db.business.drop()
//mongoimport --db msds697 --collection business --file business.json
db.business.updateMany({"restaurant_id":"40356483"},
          {$set:{"grades.$[element].score":11}},
          {arrayFilters:[{"element.score":10}]})
db.business.find({"restaurant_id": "40356483"})  // Change 3
```



See Example 8 .

https://docs.mongodb.com/manual/reference/operator/update-array/

Contents

MongoDB
- Supported Data Types
- Operations
  - Create
  - Read
  - Update
  - **Delete**

MongoDB

CRUD Overview

- Delete
  - Permanently deletes documents from the database.
    1. `db.collection_name.deleteOne(filter)` : Removes a single document from a collection.
    2. `db.collection_name.deleteMany(filter)`: Removes all documents that match the filter from a collection.
    3. `db.collection_name.drop()` : Delete entire collection efficiently.

Example 9

In "friends" collection,

- Delete one item which officeAddress' city is “San Francisco”
- Delete all items which officeAddress' city is “San Francisco”
- What is the best way to drop all?
  - `.deleteMany({})` vs `.drop()`

Example 9

In "friends" collection,

- Delete one item which officeAddress' city is “San Francisco”
- Delete all items which officeAddress' city is “San Francisco”
- What is the best way to drop all?
  - `.deleteMany({})` vs `.drop()`

```
//Remove One
db.friends.deleteOne({"officeAddress.city":"San Francisco"})
db.friends.count()

//Remove Many
db.friend.deleteMany({"officeAddress.city":"San Francisco"})
db.friend.count()

//Drop the entire collection.
db.friend.drop()
db.friend.count()
```

Extra Question

For documents in business where zipcode is 10462, and street is "Castle Hill Avenue”, update its grade to "A+" if score is greater than 10.

Contents

MongoDB

- Supported Data Types
- Operations
  - Create
  - Read
  - Update
  - Delete

References

Sadalage, Pramod J., and Martin Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education, 2012.

MongoDB. Online Documentation, https://docs.mongodb.com/