Introduction to MongoDB

MongoDB CRUD Operations

MongoDB CRUD Operations

MongoDB CRUD Introduction

MongoDB stores data in the form of **documents**, which are JSON-like field and value pairs. **Documents** are analogous to structures in programming languages that associate keys with values (e.g. dictionaries, hashes, maps, and associative arrays).

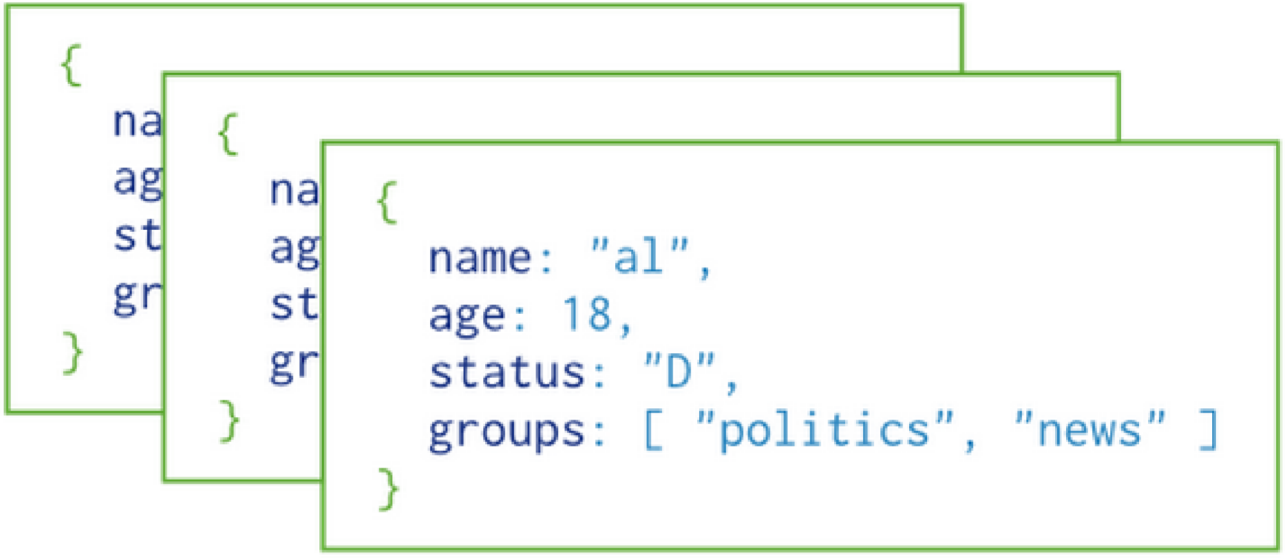
Formally, MongoDB documents are **BSON documents**. BSON is a binary representation of JSON with additional type information.

A MongoDB document.

MongoDB CRUD Operations

MongoDB CRUD Introduction

MongoDB stores all **documents** in **collections**. A collection is a group of related documents that have a set of shared common indexes. Collections are analogous to a table in relational databases.



A collection of MongoDB documents.

MongoDB CRUD Operations

MongoDB CRUD Introduction

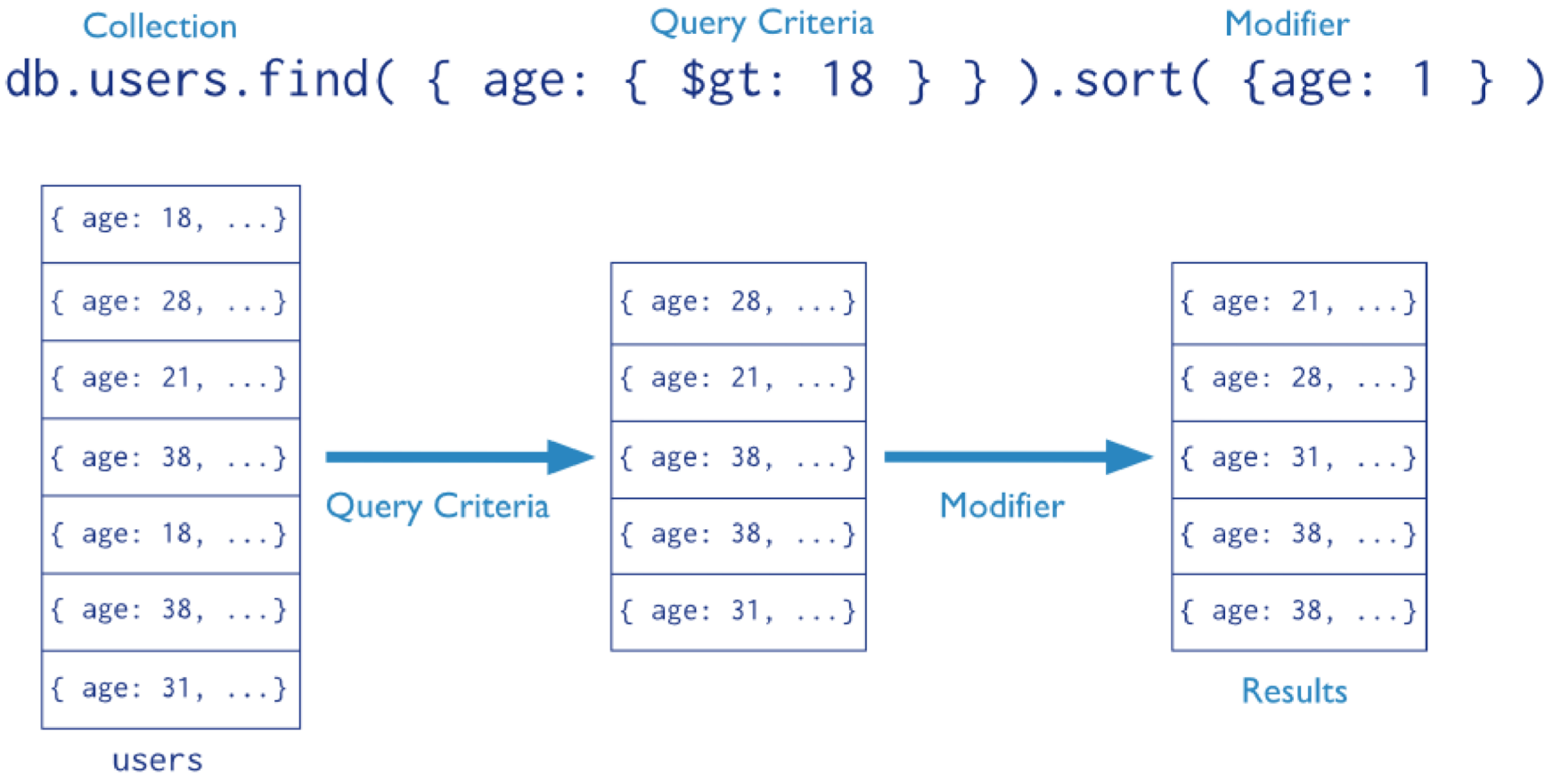
Queries on MongoDB

In MongoDB **a query targets a specific collection of documents**. Queries specify **criteria**, or **conditions**, that identify the documents that MongoDB returns to the clients.

A query may include a **projection** that specifies the fields from the matching documents to return. You can optionally **modify queries** to impose **limits**, **skips**, and **sort** orders.

MongoDB CRUD Operations

MongoDB CRUD Introduction

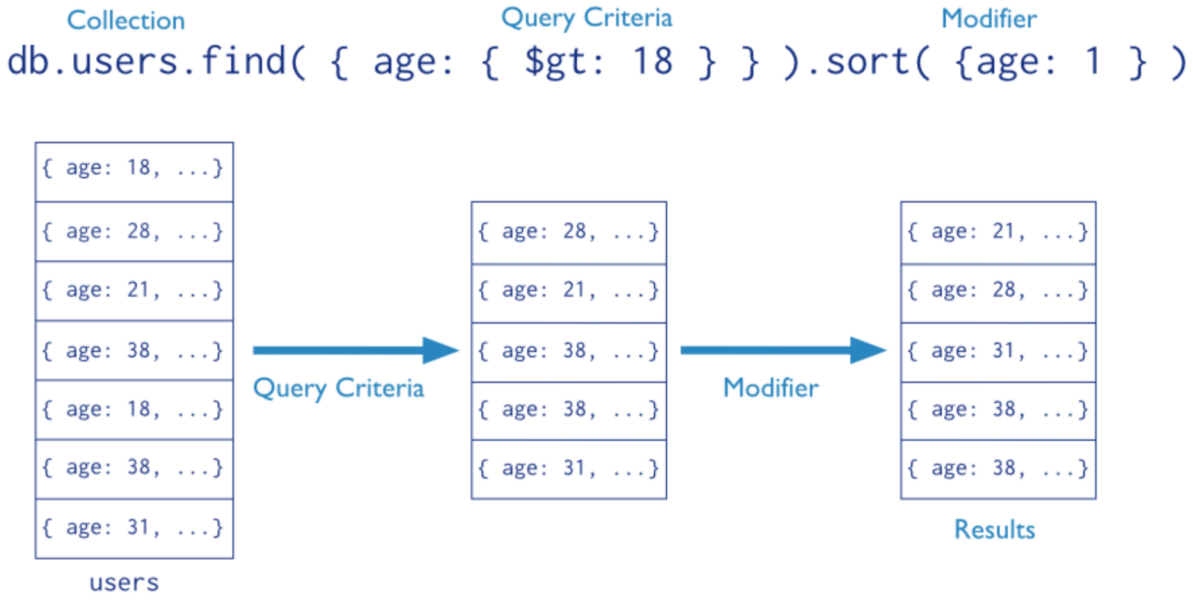


mongoDB

MongoDB CRUD Operations

MongoDB Queries

Query Statements



In the diagram, the query selects documents from the users collection. Using a query selection operator to define the conditions for matching documents, the query selects documents that have age greater than (i.e. $gt) 18. Then the sort() modifier sorts the results by age in ascending order.

MongoDB CRUD Operations

MongoDB Queries

Read operations, or queries, retrieve data stored in the database. In MongoDB, **queries select documents from a single collection**.

Queries specify **criteria**, or **conditions**, that *identify the documents that MongoDB returns to the clients*. A query may include a **projection** that *specifies the fields from the matching documents to return*. The projection limits the amount of data that MongoDB returns to the client over the network.

The following diagram highlights the components of a MongoDB query operation:

mongoDB

MongoDB CRUD Operations

MongoDB Queries

Query Interface

For query operations, MongoDB provides a **db.collection.find()** method. The method accepts both the **query criteria** and **projections** and returns a **cursor** to the matching documents. You can optionally modify the query to **impose limits, skips, and sort orders**.

The following diagram highlights the components of a MongoDB query operation:

mongoDB

MongoDB CRUD Operations

MongoDB Queries

Query Interface

The next diagram shows the same query in SQL:

The following diagram highlights the components of a MongoDB query operation:

MongoDB CRUD Operations

MongoDB Queries

db.collection.find()

```
db.collection.find(<criteria>, <projection>)
```

Selects documents in a collection and returns a cursor to the selected documents. [1]

Parameter	Type	Description
`criteria`	document	Optional. Specifies selection criteria using query operators. To return all documents in a collection, omit this parameter or pass an empty document (`{}`).
`projection`	document	Optional. Specifies the fields to return using projection operators. To return all fields in the matching document, omit this parameter.

MongoDB CRUD Operations

MongoDB Queries

Query Interface

```
db.users.find( { age: { $gt: 18 } },
               { name: 1, address: 1 } ).limit(5)
```

This query selects the documents in the users collection that match the condition age is greater than 18. To specify the greater than condition, query criteria uses the greater than (i.e. $gt) query selection operator.

The query returns at most 5 matching documents (or more precisely, a cursor to those documents).

The matching documents will return with only the _id, name and address fields.

MongoDB CRUD Operations

MongoDB Queries

Query Behavior

MongoDB queries exhibit the following **behavior**:

- All queries in MongoDB address a *single collection*.
- You can modify the query to impose limits, skips, and sort orders.
- The order of documents returned by a query is not defined unless you specify a sort().

MongoDB provides a **db.collection.findOne()** method as a special case of find() that returns a single document.

MongoDB CRUD Operations

MongoDB Queries

Projection

Queries in MongoDB return all fields in all matching documents by default. To limit the amount of data that MongoDB sends to applications, include a projection in the queries. By projecting results with a subset of fields, applications reduce their network overhead and processing requirements.

Projections, which are the *second* argument to the *find*() method, may either specify a list of fields to return *or* list fields to exclude in the result documents.

By default, the _id field is included in the results. To suppress the _id field from the result set, specify _id: 0 in the projection document.

> IMPORTANT:
>
> Except for excluding the `_id` field in inclusive projections, you cannot mix exclusive and inclusive projections.

MongoDB CRUD Operations

MongoDB Queries

db.collection.find()

The `projection` parameter takes a document of the following form:

```
{ field1: <boolean>, field2: <boolean> ... }
```

The `<boolean>` value can be any of the following:

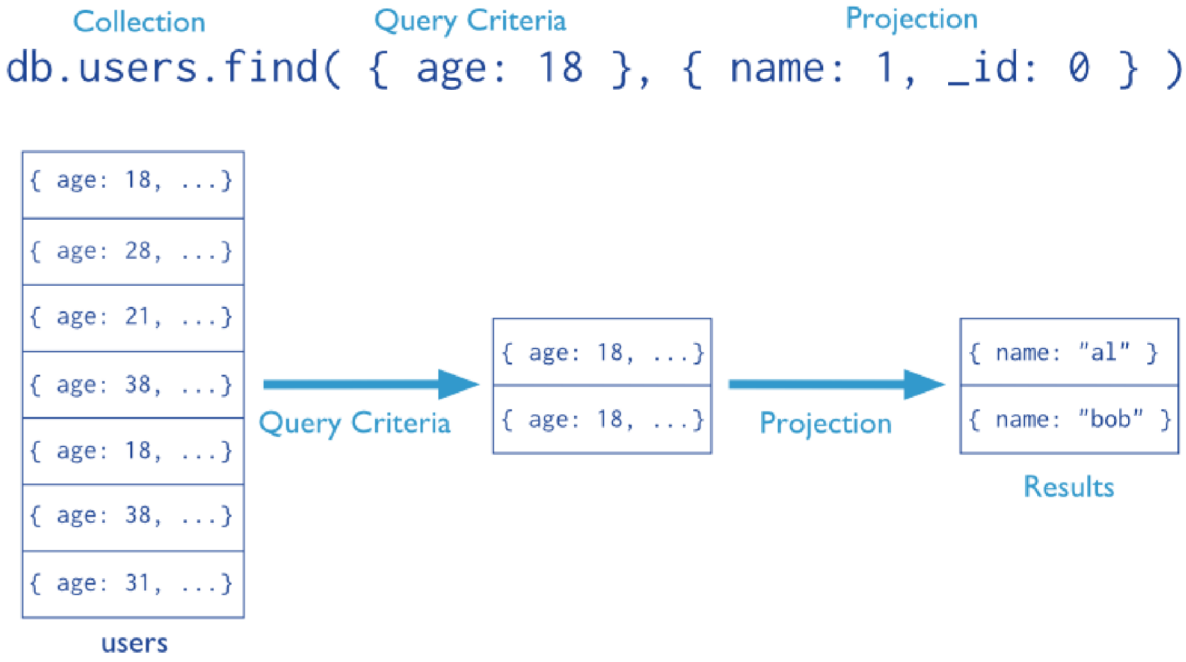
- `1` or `true` to include the field. The `find()` method always includes the _id field even if the field is not explicitly stated to return in the projection parameter.
- `0` or `false` to exclude the field.

A `projection` *cannot* contain *both* include and exclude specifications, except for the exclusion of the `_id` field. In projections that *explicitly include* fields, the `_id` field is the only field that you can *explicitly exclude*.

MongoDB CRUD Operations

MongoDB Queries

Projection



In the diagram, the query selects from the users collection. The criteria matches the documents that have age equal to 18. Then the projection specifies that only the name field should return in the matching documents.

MongoDB CRUD Operations

MongoDB Queries

Projection Examples

Exclude One Field From a Result Set

```
db.records.find( { "user_id": { $lt: 42 } },
                 { "history": 0 } )
```

This query selects documents in the records collection that match the condition { "user_id": {$lt: 42 } }, and uses the projection { "history": 0 } to exclude the history field from the documents in the result set.

MongoDB CRUD Operations

MongoDB Queries

Projection Examples

Return Two fields and the _id Field

```
db.records.find( { "user_id": { $lt: 42 } },
                 { "name": 1, "email": 1 } )
```

This query selects documents in the records collection that match the query { "user_id": { $lt: 42 } } and uses the projection { "name": 1, "email": 1 } to return just the _id field (implicitly included), name field, and the email field in the documents in the result set.

MongoDB CRUD Operations

MongoDB Queries

Projection Examples

Return Two Fields and Exclude _id

```
db.records.find( { "user_id": { $lt: 42} },
                 { "_id": 0, "name": 1 , "email": 1 } )
```

This query selects documents in the records collection that match the query { "user_id": { $lt:42} }, and only returns the name and email fields in the documents in the result set.

MongoDB CRUD Operations

MongoDB Queries

Find All Documents in a Collection

The find() method with no parameters returns all documents from a collection and returns all fields for the documents.

For example, the following operation returns all documents in the respective collections:

```
db.students.find()
db.students.find({})
```

```
db.books.find()
db.books.find({})
```

```
db.bios.find()
db.bios.find({})
```

MongoDB CRUD Operations

MongoDB Queries

Find Documents that Match Query Criteria

To find documents that match a set of selection criteria, call find() with the <criteria> parameter.

The following operation returns all the documents from the students collection where *birth_year* is equal to 1990:

```
db.students.find({ "birth_year": 1990 })
```

The following operation returns all the documents from the books collection where *publicationYear* is equal to 2011:

```
db.books.find({ "publicationYear": 2011 })
```

MongoDB CRUD Operations

MongoDB Queries

Find Documents that Match Query Criteria

The following operation returns all the documents from the books collection where a book contains the tag ‘html’:

```
db.books.find({“tags”: “html”})
```

The following operation returns all the documents from the students collection where *birth_year* is equal to 1990 and gender is “H”:

```
db.students.find({ "birth_year": 1990, gender: “H” })
```

MongoDB CRUD Operations

MongoDB Queries

Query Using Operators

The following operation returns all the documents from the students collection where *birth_year* is greater or equal to 1990:

```
db.students.find({ "birth_year": {$gte: 1990} })
```

The following operation returns all the documents from the books collection where *publicationYear* is lower or equal to 2009:

```
db.books.find({ "publicationYear": {$lte: 2009} })
```

MongoDB CRUD Operations

MongoDB Queries

Query Using Operators

Name	Description
`$gt`	Matches values that are greater than the value specified in the query.
`$gte`	Matches values that are greater than or equal to the value specified in the query.
`$in`	Matches any of the values that exist in an array specified in the query.
`$lt`	Matches values that are less than the value specified in the query.
`$lte`	Matches values that are less than or equal to the value specified in the query.
`$ne`	Matches all values that are not equal to the value specified in the query.
`$nin`	Matches values that **do not** exist in an array specified to the query.

http://docs.mongodb.org/manual/reference/operator/query/

MongoDB CRUD Operations

MongoDB Queries

Query Using Operators

The following operation returns all the documents from the students collection where *birth_year* is equal to 1980 or 1985:

```
db.students.find({ "birth_year": {$in:[1980, 1985]} })
```

```
db.students.find({ $or: [{“birth_year”: 1980},
                         {“birth_year”: 1985}]})
```

MongoDB CRUD Operations

MongoDB Queries

Query Using Operators

The following operation returns all the documents from the books collection where *tags* contains “html” or “css”:

```
db.books.find({"tags": {$in: ["html", "css"]}})
```

```
db.books.find({$or: [{"tags": "html"},
                     {"tags": "css"} ]})
```

```
db.books.find({$and: [{"tags": "html"},
                      {"tags": "css"}]})
```

MongoDB CRUD Operations

MongoDB Queries

Query Using Operators

The following operation returns all the documents from the students collection where *birth_year* is between 1990 and 1994:

```
db.students.find({ "birth_year": {$gte: 1990, $lte: 1994}})
```

The following operation returns all the documents from the books collection where *publicationYear* is between 1990 and 1994:

```
db.books.find({"publicationYear": {$gte: 1990, $lte: 1994}})
```

MongoDB CRUD Operations

MongoDB Queries

Query Using Operators

The following operation returns all the documents from the students collection where *birth_year* is between 1990 and 1994:

```
db.students.find({$and: [{"birth_year": {$gte: 1990}},
                         {"birth_year":  {$lte: 1994}}]})
```

The following operation returns all the documents from the books collection where *publicationYear* is between 1990 and 1994:

```
db.books.find({$and: [{"publicationYear": {$gte: 1990}},
                      {"publicationYear": {$lte: 1994}}]})
```

MongoDB CRUD Operations

MongoDB Queries

Query an Array of Documents

The following operation returns all the documents from the books collection where a book has exactly 2 authors:

```
db.books.find({"author": {$size: 2}})
```

The following operation returns all the documents from the books collection where a book has exactly 1 author:

```
db.books.find({"author": {$size: 1}})
```

MongoDB CRUD Operations

MongoDB Queries

Query an Array of Documents

The following operation returns all the documents from the books collection where a book has exactly 2, 3 or 4 authors:

```
db.books.find({$or: [{"author": {$size: 2}},
                     {"author": {$size: 3}},
                     {"author": {$size: 4}}]})
```

MongoDB CRUD Operations

MongoDB Queries

Query with regular expressions

The following operation returns all the documents from the students collection where student name starts with vocal:

```
db.students.find({"firstname": {$regex: /^[aeiou]/, $options: 'i'}})
```

```
db.students.find({"firstname": {$regex: /^[aeiou]/i}})
```

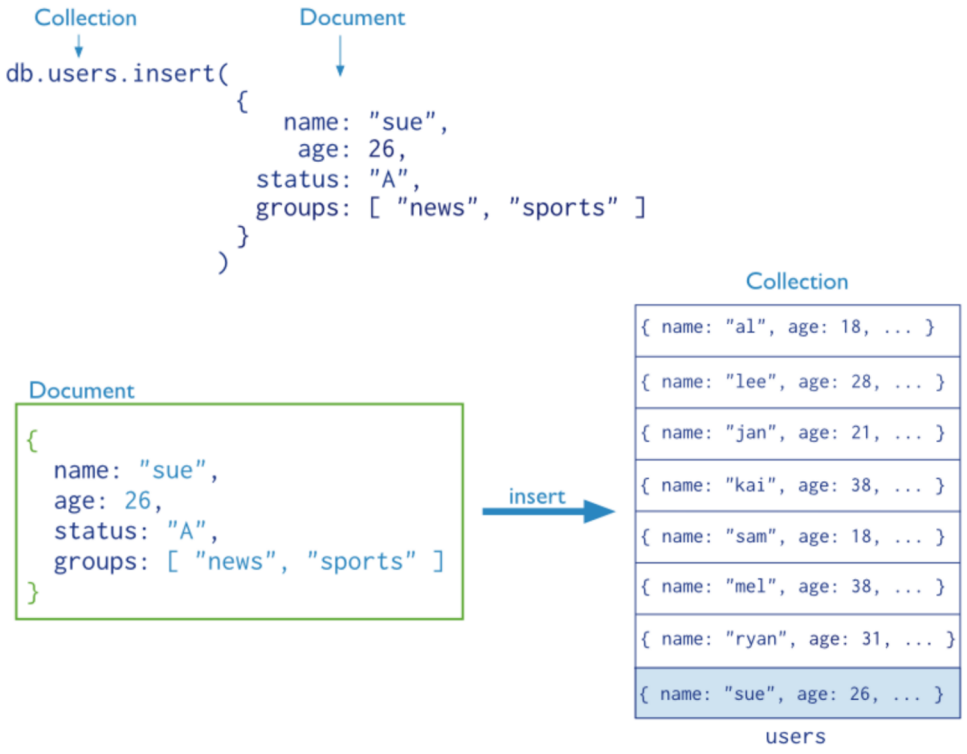
```
db.students.find({"firstname": /^[aeiou]/i})
```

MongoDB CRUD Operations

MongoDB CRUD Introduction

Data Modification on MongoDB

Data modification refers to operations that create, update, or delete data. In MongoDB, these operations modify the data of a single collection. For the update and delete operations, you can specify the criteria to select the documents to update or remove.



mongoDB

MongoDB CRUD Operations

MongoDB – Update a collection

db.collection.update()

```
db.collection.update(query, update, options)
```

Modifies an existing document or documents in a collection. The method can modify specific fields of an existing document or documents or replace an existing document entirely, depending on the update parameter.

By default, the update() method updates a **single** document. Set the Multi Parameter to update all documents that match the query criteria.

MongoDB CRUD Operations

MongoDB – Update a collection

db.collection.update()

The update() method has the following form:

```
db.collection.update(
    <query>,
    <update>,
    {
      upsert: <boolean>,
      multi: <boolean>,
      writeConcern: <document>
    }
))
```

MongoDB CRUD Operations

MongoDB – Update a collection

db.collection.update()

Parameter	Type	Description
`query`	document	The selection criteria for the update. Use the same query selectors as used in the `find()` method.
`update`	document	The modifications to apply. For details see Update Parameter.
`upsert`	boolean	Optional. If set to `true`, creates a new document when no document matches the query criteria. The default value is `false`, which does *not* insert a new document when no match is found.
`multi`	boolean	Optional. If set to `true`, updates multiple documents that meet the `query` criteria. If set to `false`, updates one document. The default value is `false`. For additional information, see Multi Parameter.

MongoDB CRUD Operations

MongoDB – Update a collection

db.collection.update()

```
db.books.update(
   { _id: 1 },
   {
     $inc: { stock: 5 },
     $set: {
       item: "ABC123",
       "info.publisher": "2222",
       tags: [ "software" ],
       "ratings.1": { by: "xyz", rating: 3 }
     }
   }
)
```

MongoDB CRUD Operations

MongoDB findAndModify

findAndModify

The findAndModify command modifies and returns a single document. By default, the returned document does not include the modifications made on the update. To return the document with the modifications made on the update, use the new option.

The command has the following syntax:

```
{
  findAndModify: <string>,
  query: <document>,
  sort: <document>,
  remove: <boolean>,
  update: <document>,
  new: <boolean>,
  fields: <document>,
  upsert: <boolean>
}
```

MongoDB CRUD Operations

MongoDB findAndModify

findAndModify

The findAndModify command takes the following fields:

`findAndModify`	string	The collection against which to run the command.
`query`	document	Optional. The selection criteria for the modification. The `query` field employs the same query selectors as used in the `db.collection.find()` method. Although the query may match multiple documents, `findAndModify` will select only one document to modify.
`sort`	document	Optional. Determines which document the operation modifies if the query selects multiple documents. `findAndModify` modifies the first document in the sort order specified by this argument.

MongoDB CRUD Operations

MongoDB findAndModify

findAndModify

The findAndModify command takes the following fields:

`remove`	Boolean	Must specify either the `remove` or the `update` field. Removes the document specified in the `query` field. Set this to `true` to remove the selected document . The default is `false`.
`update`	document	Must specify either the `remove` or the `update` field. Performs an update of the selected document. The `update` field employs the same update operators or `field: value` specifications to modify the selected document.
`new`	Boolean	Optional. When `true`, returns the modified document rather than the original. The `findAndModify` method ignores the `new` option for `remove` operations. The default is `false`.

MongoDB CRUD Operations

MongoDB findAndModify

findAndModify

The findAndModify command takes the following fields:

`fields`	document	Optional. A subset of fields to return. The `fields` document specifies an inclusion of a field with `1`, as in: `fields: { <field1>: 1, <field2>: 1, ... }`. See projection.
`upsert`	Boolean	Optional. Used in conjunction with the `update` field. When `true`, `findAndModify` creates a new document if no document matches the `query`, or if documents match the `query`, `findAndModify` performs an update. The default is `false`.

MongoDB CRUD Operations

MongoDB Remove Documents

db.collection.remove()

Removes documents from a collection.

The db.collection.remove() method can have one of two syntaxes.
The remove() method can take a query document and an optional justOne boolean:

```
db.collection.remove(
   <query>,
   <justOne>
)
```

MongoDB CRUD Operations

MongoDB Remove Documents

Remove All Documents from a Collection

To remove all documents in a collection, call the remove method with an empty query document {}. The following operation deletes all documents from the students collection:

```
db.students.remove( { } )
```

This operation is not equivalent to the drop() method.

To remove all documents from a collection, it may be more efficient to use the drop() method to drop the entire collection, including the indexes, and then recreate the collection and rebuild the indexes.

MongoDB CRUD Operations

MongoDB Remove Documents

Remove All Documents that Match a Condition

To remove the documents that match a deletion criteria, call the remove() method with the <query>parameter.

The following operation removes all the documents from the collection products where qty is greater than 20:

```
db.products.remove( { qty: { $gt: 20 } } )
```

Exercises

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes de género masculino (2895)

```

```

//Buscar los estudiantes de género femenino (348)

```

```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes de género masculino (2895)

```
> db.students.find({"gender": "H"})
> db.students.find({"gender": "H"}).count()
> db.students.count({"gender": "H"})
```

//Buscar los estudiantes de género femenino (348)

```
> db.students.find({"gender": "M"})
> db.students.find({"gender": "M"}).count()
> db.students.count({"gender": "M"})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes nacidos en el año 1993 (97)

```

```

//Buscar los estudiantes de género masculino y nacidos en el año 1993 (81)

```

```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes nacidos en el año 1993 (97)

```
> db.students.find({"birth_year": 1993})
> db.students.find({"birth_year": 1993}).count()
> db.students.count({"birth_year": 1993})
```

//Buscar los estudiantes de género masculino y nacidos en el año 1993 (81)

```
> db.students.find({gender: "H", "birth_year": 1993})
> db.students.find({gender: "H", "birth_year": 1993}).count()
> db.students.count({gender: "H", "birth_year": 1993})
> db.students.find({$and: [{gender: "H"}, {birth_year: 1993}]})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes nacidos después del año 1990 (289)

```
```

//Buscar los estudiantes nacidos antes o en el año 1990 (2954)

```
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes nacidos después del año 1990 (289)

```
> db.students.find({birth_year: {$gt: 1990}})
> db.students.find({birth_year: {$gt: 1990}}).count()
> db.students.count({birth_year: {$gt: 1990}})
```

//Buscar los estudiantes nacidos antes o en el año 1990 (2954)

```
> db.students.find({birth_year: {$lte: 1990}})
> db.students.find({birth_year: {$lte: 1990}}).count()
> db.students.count({birth_year: {$lte: 1990}})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes nacidos en la década de los 90 (387)

```
```

//Buscar los estudiantes nacidos en la década de los 80 (936)

```
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes nacidos en la década de los 90 (387)

```
> db.students.find({$and: [{birth_year: {$gte: 1990}}, {birth_year: {$lt: 2000}}]})
> db.students.find({$and: [{birth_year: {$gte: 1990}}, {birth_year: {$lt: 2000}}]}).count()
> db.students.find({"birth_year": {"$gte": 1990, "$lt": 2000}})
```

//Buscar los estudiantes nacidos en la década de los 80 (936)

```
> db.students.find({$and: [{birth_year: {$gte: 1980}}, {birth_year: {$lt: 1990}}]})
> db.students.find({$and: [{birth_year: {$gte: 1980}}, {birth_year: {$lt: 1990}}]}).count()
> db.students.find({"birth_year": {"$gte": 1980, "$lt": 1990}})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes de género femenino nacidos en la década de los 90 (48)

```

```

//Buscar los estudiantes de género masculino nacidos en la década de los 80 (851)

```

```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes de género femenino nacidos en la década de los 90 (48)

```
> db.students.find({$and: [{gender: "M"},{birth_date: {$gte: 1990}},
                           {birth_date: {$lt: 2000}}]})
> db.students.find({"gender": "M", "birth_year": {"$gte": 1990, "$lt": 2000}})
> db.students.find({"$and": [{"gender": "M"},
                             {"birth_year": {"$gte": 1990, "$lt": 2000}}]})
```

//Buscar los estudiantes de género masculino nacidos en la década de los 80 (851)

```
> db.students.find({$and: [{gender: "H"},{birth_date: {$gte: 1980}},
                           {birth_date: {$lt: 1990}}]})
> db.students.find({"gender": "H", "birth_year": {"$gte": 1980, "$lt": 1990}})
> db.students.find({"$and": [{"gender": "H"},
                             {"birth_year": {"$gte": 1980, "$lt": 1990}}]})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes que no han nacido en el año 1985 (3147)

```

```

//Buscar aquellos estudiantes que hayan nacido en el año 1970, 1980 o 1990 (293)

```

```

//Buscar aquellos estudiantes que no hayan nacido en el año 1970, 1980 o 1990 (2950)

```

```

//Buscar los estudiantes nacidos en año par (1684)

```

```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes que no han nacido en el año 1985 (3147)

```
> db.students.find({"birth_year": {"$ne": 1985}})
```

//Buscar aquellos estudiantes que hayan nacido en el año 1970, 1980 o 1990 (293)

```
> db.students.find({"birth_year": {"$in": [1970, 1980, 1990]}})
```

//Buscar aquellos estudiantes que no hayan nacido en el año 1970, 1980 o 1990 (2950)

```
> db.students.find({"birth_year": {"$nin": [1970, 1980, 1990]}})
```

//Buscar los estudiantes nacidos en año par (1684)

```
> db.students.find({"birth_year": {"$mod": [2,0]}})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes nacidos en año impar (1559)

```
>
```

//Buscar estudiantes nacidos en año par de la década de los 70 que sean hombres (403)

```

```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes nacidos en año impar (1559)

```
> db.students.find({"birth_year": {"$mod": [2,1]}})
```

//Buscar estudiantes nacidos en año par de la década de los 70 que sean hombres (403)

```
> db.students.find({gender: "H", birth_year: {$mod:[2,0], $gte: 1970, $lt: 1980}})

> db.students.find({"$and": [{"gender": "H"},
                      {"birth_year": {"$mod": [2, 0], "$gte": 1970, "$lt":  1980} }]})

> db.students.find({$and: [{gender:"H"},{birth_year: {$mod:[2,0]}},
                           {birth_year: {$gte:1970}},{birth_year: {$lt:1980}}]})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes que tengan teléfono auxiliar (679)

```

```

//Buscar los estudiantes que no tengan teléfono auxiliar (2564)

```

```

//Buscar los estudiantes que no tengan segundo apellido (421)

```

```

//Buscar los estudiantes que tengan teléfono auxiliar y solo un apellido (71)

```

```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes que tengan teléfono auxiliar (679)

```
> db.students.find({“phone_aux": {"$exists": true}})
> db.students.find({“phone_aux": {"$exists": 1}})
```

//Buscar los estudiantes que no tengan teléfono auxiliar (2564)

```
> db.students.find({“phone_aux": {"$exists": false}})
> db.students.find({“phone_aux": {"$exists": 0}})
```

//Buscar los estudiantes que no tengan segundo apellido (421)

```
> db.students.find({“lastname2": {"$exists": 0}})
> db.students.find({“lastname2": {"$exists": false}})
```

//Buscar los estudiantes que tengan teléfono auxiliar y solo un apellido (71)

```
> db.students.find({“phone_aux”: {$exists: true}, “lastname2": {"$exists": false}})

> db.students.find({"$and": [{"phone_aux": {"$exists": true}},
                             {"lastname2": {"$exists": false}}]})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes cuyo email termine en .net (47)

```
```

//Buscar los estudiantes cuyo email termine en .org (16)

```
```

// Buscar los estudiantes cuyo teléfono empiece por 622 (201)

```
```

//Buscar los estudiantes cuyo dni empiece y termine por letra (244)

```
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes cuyo email termine en .net (47)

```
> db.students.find({"email": /\.net$/i})
> db.students.find({"email": /\.net$/i}).count()
```

//Buscar los estudiantes cuyo email termine en .org (16)

```
> db.students.find({"email": /\.org$/i})
> db.students.find({"email": /\.org$/i}).count()
```

// Buscar los estudiantes cuyo teléfono empiece por 622 (201)

```
> db.students.find({$or: [{"phone": /^622/i}, {"phone_aux": /^622/i}]})
```

//Buscar los estudiantes cuyo dni empiece y termine por letra (244)

```
> db.students.find({"dni": /^[A-Z].*[A-Z]$/i})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes cuyo nombre empiece por vocal (760)

```
```

//Buscar estudiantes cuyo nombre sea compuesto (470)

```
```

//Buscar los estudiantes con nombre más largo de 13 caracteres (138)

```
```

//Buscar los estudiantes con 3 o más vocales en su nombre (705)

```
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: students

//Buscar los estudiantes cuyo nombre empiece por vocal (760)

```
> db.students.find({"name": /^[aeiouàáèéìíòóùú]{1}/i})
```

//Buscar estudiantes cuyo nombre sea compuesto (470)

```
> db.students.find({"name": /.+\s.+/})
> db.students.find({"name": /\s/}, {"name": true})
```

//Buscar los estudiantes con nombre más largo de 13 caracteres (138)

```
> db.students.find({"name": /.{13,}/})
> db.students.find({$where: "this.name.length >= 13"}).count()
```

//Buscar los estudiantes con 3 o más vocales en su nombre (705)

```
> db.students.find({"firstname": /.*[aeiouàáèéìíòóùú].*[aeiouàáèéìíòóùú]
                                 .*[aeiouàáèéìíòóùú].*[aeiouàáèéìíòóùú].*/i})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: bios

//Buscar aquellos desarrolladores que hayan realizado contribuciones en OOP (2)

```
```

//Buscar aquellos desarrolladores que hayan realizado contribuciones en OOP o Java (3)

```
```

//Buscar aquellos desarrolladores que hayan realizado contribuciones en OOP y Simula (2)

```
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: bios

//Buscar aquellos desarrolladores que hayan realizado contribuciones en OOP (2)

```
> db.bios.find({"contribs": "OOP"})
```

//Buscar aquellos desarrolladores que hayan realizado contribuciones en OOP o Java (3)

```
> db.bios.find({"contribs": {"$in": ["OOP", "Java"]}})
> db.bios.find({"$or": [{"contribs": "OOP"}, {"contribs": "Java"}]})
```

//Buscar aquellos desarrolladores que hayan realizado contribuciones en OOP y Simula (2)

```
> db.bios.find({"contribs": {"$all": ["OOP", "Simula"]}})
> db.bios.find({"$and": [{"contribs": "OOP"}, {"contribs": "Simula"}]})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: bios

//Buscar aquellos desarrolladores que sigan vivos (4)

```
```

//Buscar aquellos desarrolladores que hayan muerto (6)

```
```

//Buscar aquellos desarrolladores que hayan obtenido un premio en el año 2002 (1)

```
```

//Buscar aquellos desarrolladores que hayan obtenido exactamente 3 premios

```
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: bios

//Buscar aquellos desarrolladores que sigan vivos (4)

```
> db.bios.find({"deathYear": {$exists: false}})
```

//Buscar aquellos desarrolladores que hayan muerto (6)

```
> db.bios.find({"deathYear": {$exists: true}})
```

//Buscar aquellos desarrolladores que hayan obtenido un premio en el año 2002 (1)

```
> db.bios.find({"awards.year": 2002})
> db.bios.find({"awards": {"$elemMatch": {"year": 2002}}})
```

//Buscar aquellos desarrolladores que hayan obtenido exactamente 3 premios

```
> db.bios.find({"awards": {"$size": 3}})
```

MongoDB CRUD Operations

Exercises

Database: imdb – Collection: people

//Buscar las personas que sólo han actuado (no dirigido) (1909)

```
```

//Buscar las personas que sólo han dirigido (no actuado) (341)

```
```

//Buscar las personas que han actuado y dirigido (20)

```
```

//Buscar las personas que ni han actuado ni dirigido (29)

```
```

MongoDB CRUD Operations

Exercises

Database: imdb – Collection: people

//Buscar las personas que sólo han actuado (no dirigido) (1909)

```
> db.people.find({"hasActed": {$exists: true}, "hasDirected": {$exists: false}})
```

//Buscar las personas que sólo han dirigido (no actuado) (341)

```
> db.people.find({"hasActed": {$exists: false}, "hasDirected": {$exists: true}})
```

//Buscar las personas que han actuado y dirigido (20)

```
> db.people.find({"hasActed": {$exists: true}, "hasDirected": {$exists: true}})
```

//Buscar las personas que ni han actuado ni dirigido (29)

```
> db.people.find({"hasActed": {$exists: false},"hasDirected": {$exists: false}})
```

MongoDB CRUD Operations

Exercises

Database: imdb – Collection: people

//Buscar las películas protagonizadas por Penelope Cruz (2)

MongoDB CRUD Operations

Exercises

Database: imdb – Collection: people

//Buscar las películas protagonizadas por Penelope Cruz (2)

```
> db.movies.find({"actors": {$elemMatch: {"name": "Penelope Cruz"}}})
> db.movies.find({"actors.name": "Penelope Cruz"})
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: books

//Buscar aquellos libros que han sido escritos por Martin Fowler y Kent Beck (2)

```
```

//Buscar los libros que tengan el tag 'programming' y 'agile‘ (16)

```
```

//Buscar los libros con el tag html, html5, css o css3 (14)

```
```

//Buscar los libros que no tengan el tag html, html5, css o css3 (319)

```
```

MongoDB CRUD Operations

Exercises

Database: edx – Collection: books

//Buscar aquellos libros que han sido escritos por Martin Fowler y Kent Beck (2)

```
> db.books.find({"author": {"$all": ["Martin Fowler", "Kent Beck"]}})
> db.books.find({“$and”: [{“author”: “Martin Fowler”}, {“author”: “Kent Beck”}]})
```

//Buscar los libros que tengan el tag 'programming' y 'agile‘ (16)

```
> db.books.find({"tags": {"$all": ["programming","agile"]}})
> db.books.find({"$and": [ {"tags": "programming"},{"tags": "agile"}]})
```

//Buscar los libros con el tag html, html5, css o css3 (14)

```
> db.books.find({"tags": {"$in": ["html","html5","css","ccs3"]}})
```

//Buscar los libros que no tengan el tag html, html5, css o css3 (319)

```
> db.books.find({"tags": {"$nin": ["html","html5","css","ccs3"]}})
```


mongoDB