

mongoDB®

An introduction to MongoDB

MongoDB CRUD Operations

- ▶ CRUD operations *create, read, update, and delete* documents.
 - The Create operation is used to insert new documents in the MongoDB database.
 - The Read operation is used to query a document in the database.
 - The Update operation is used to modify existing documents in the database.
 - The Delete operation is used to remove documents in the database.

Create Operations

- ▶ Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.
- ▶ MongoDB provides the following methods to insert documents into a collection:
 - ▶ `db.collection.insertOne()`
 - ▶ New in version 3.2
 - ▶ `db.collection.insertMany()`
 - ▶ New in version 3.2
- ▶ In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

```
db.users.insertOne(  ← collection
{
  name: "sue",        ← field: value
  age: 26,             ← field: value
  status: "pending"   ← field: value
}                    } document
)
```

Read Operations

- ▶ Read operations retrieve documents from a collection; i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:
- ▶ `db.collection.find()`

Update Operations

- ▶ Update operations modify existing documents in a collection. MongoDB provides the following methods to update documents of a collection:
 - ▶ `db.collection.updateOne()`
 - ▶ New in version 3.2
 - ▶ `db.collection.updateMany()`
 - ▶ New in version 3.2
 - ▶ `db.collection.replaceOne()`
 - ▶ New in version 3.2
- ▶ In MongoDB, update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.
- ▶ You can specify criteria, or filters, that identify the documents to update. These filters use the same syntax as read operations.

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

- ← collection
- ← query criteria
- ← projection
- ← cursor modifier

P E T E R

```
db.users.updateMany(  
  { age: { $lt: 18 } },  
  { $set: { status: "reject" } }  
)
```




← collection
← update filter
← update action

Delete Operations

- ▶ Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:
 - ▶ `db.collection.deleteOne()`
 - ▶ New in version 3.2
 - ▶ `db.collection.deleteMany()`
 - ▶ New in version 3.2
- ▶ In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.
- ▶ You can specify criteria, or filters, that identify the documents to remove. These filters use the same syntax as read operations.

```
db.users.deleteMany(  
  { status: "reject" }  
)
```



collection

delete filter

Datatypes in MongoDB

Data Types	Description
String	String is the most commonly used datatype. It is used to store data. A string must be UTF 8 valid in mongodb.
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	This datatype is used to store boolean values. It just shows YES/NO values.
Double	Double datatype 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.
Symbol	It is generally used for languages that use a specific type.
Date	This datatype stores the current date or time in unix time format. It makes you possible to specify your own date time by creating object of date and pass the value of date, month, year into it.

MongoDB Comparison Operations

Name	Description
<code>\$eq</code>	Matches values that are equal to a specified value.
<code>\$gt</code>	Matches values that are greater than a specified value.
<code>\$gte</code>	Matches values that are greater than or equal to a specified value.
<code>\$in</code>	Matches any of the values specified in an array.
<code>\$lt</code>	Matches values that are less than a specified value.
<code>\$lte</code>	Matches values that are less than or equal to a specified value.
<code>\$ne</code>	Matches all values that are not equal to a specified value.
<code>\$nin</code>	Matches none of the values specified in an array.

- `db.inventory.find({"quantity": { $gte: 12000}}).pretty()`
- `db.inventory.find({"quantity": { $lt: 5000}}).pretty()`
- `db.inventory.find({"quantity": { $gt: 5000}}).pretty()`

- ▶ \$in and \$nin Operators

- ▶ `db.inventory.find({"price": { $in: [3, 6]}}).pretty()`

Logical Operations

Name	Description
<code>\$and</code>	Joins query clauses with a logical <code>AND</code> returns all documents that match the conditions of both clauses.
<code>\$not</code>	Inverts the effect of a query expression and returns documents that do <i>not</i> match the query expression.
<code>\$nor</code>	Joins query clauses with a logical <code>NOR</code> returns all documents that fail to match both clauses.
<code>\$or</code>	Joins query clauses with a logical <code>OR</code> returns all documents that match the conditions of either clause.

- `db.employees.find({ $and: [{"job_role": "Store Associate"}, {"emp_age": {$gte: 20, $lte: 30}}]}).pretty()`
- `db.employees.find({ $or: [{"job_role": "Senior Cashier"}, {"job_role": "Store Manager"}]}).pretty()`

Element

The element query operators are used to identify documents using the fields of the document. The table given below lists the current element operators.

Name	Description
<code>\$exists</code>	Matches documents that have the specified field.
<code>\$type</code>	Selects documents if a field is of the specified type.

- `db.employees.find({ "emp_age": { $exists: true, $gte: 30}}).pretty()`
- `db.employees.find({ "emp_age": { $type: "double"}})`

Array Operators

Name	Description
<code>\$all</code>	Matches arrays that contain all elements specified in the query.
<code>\$elemMatch</code>	Selects documents if element in the array field matches all the specified <code>\$elemMatch</code> conditions.
<code>\$size</code>	Selects documents if the array field is a specified size.

- `db.inventory.find({ "category": { $all: ["healthy", "organic"] }}).pretty()`
- `db.inventory.find({ "category": { $size: 2 }}).pretty()`

Evaluation Operator

The MongoDB evaluation operators can evaluate the overall data structure or individual field in a document.

Name	Description
<code>\$expr</code>	Allows use of aggregation expressions within the query language.
<code>\$jsonSchema</code>	Validate documents against the given JSON Schema.
<code>\$mod</code>	Performs a modulo operation on the value of a field and selects documents with a specified result.
<code>\$regex</code>	Selects documents where values match a specified regular expression.
<code>\$text</code>	Performs text search.
<code>\$where</code>	Matches documents that satisfy a JavaScript expression.

```
db.inventory.find({"quantity": {$mod: [3000, 1000]}}).pretty()
```

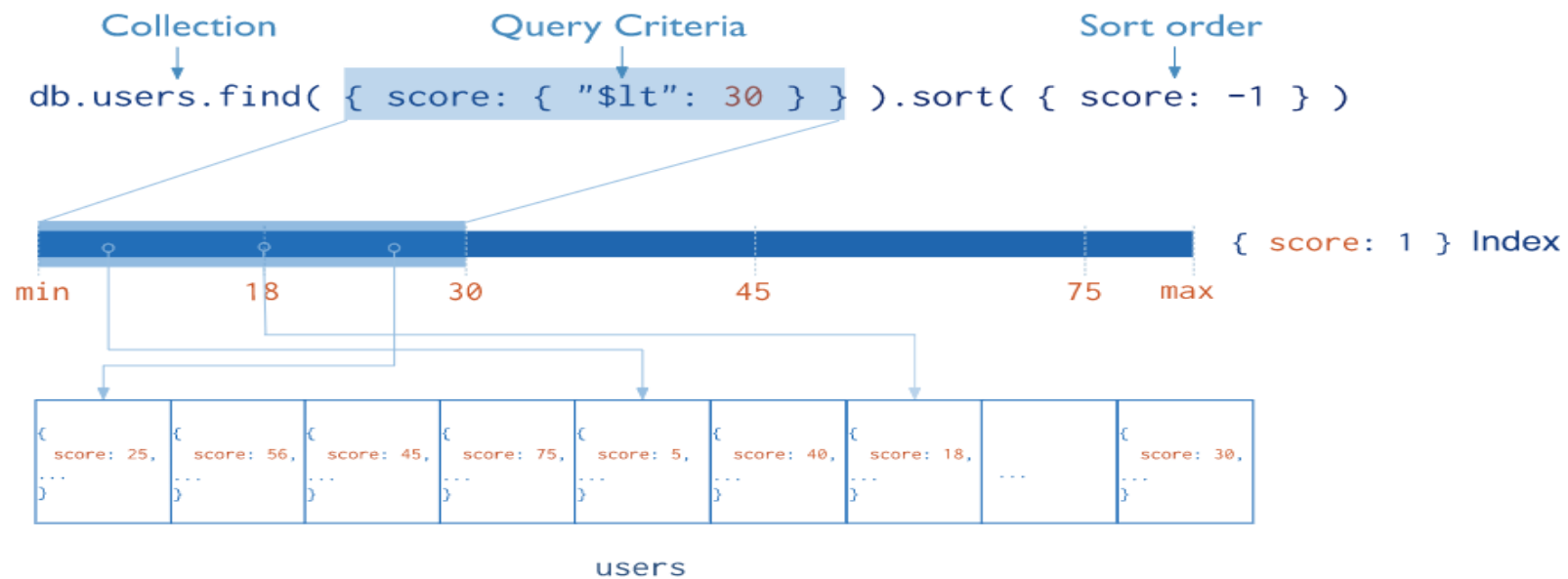
Project operation

Name	Description
<code>\$</code>	Projects the first element in an array that matches the query condition.
<code>\$elemMatch</code>	Projects the first element in an array that matches the specified <code>\$elemMatch</code> condition.
<code>\$meta</code>	Projects the document's score assigned during <code>\$text</code> operation.
<code>\$slice</code>	Limits the number of elements projected from an array. Supports skip and limit slices.

Indexes

- ▶ Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a *collection scan*, i.e. scan every document in a collection, to select those documents that match the query statement.
- ▶ If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.
- ▶ Indexes are special data structures that store a small portion of the collection's data set in an easy to traverse form.
- ▶ The index stores the value of a specific field or set of fields, ordered by the value of the field. The ordering of the index entries supports efficient equality matches and range-based query operations. In addition, MongoDB can return sorted results by using the ordering in the index.

- The following diagram illustrates a query that selects and orders the matching documents using an index:





- ▶ To create an index in the Mongo Shell, use
`db.collection.createIndex()`

MongoDB sort() method

- ▶ In MongoDB, sort() method is used to sort the documents in the collection. This method accepts a document containing list of fields along with their sorting order.
- ▶ The sorting order is specified as 1 or -1.
 - 1 is used for ascending order sorting.
 - -1 is used for descending order sorting.
- ▶ `db.COLLECTION_NAME.find().sort({KEY:1})`
 - ▶ `db.student.find().sort({"Course":-1})`

MongoDB limit() Method

- ▶ In MongoDB, limit() method is used to limit the fields of document that you want to show. Sometimes, you have a lot of fields in collection of your database and have to retrieve only 1 or 2. In such case, limit() method is used.
- ▶ The MongoDB limit() method is used with find() method.
 - ▶ `db.COLLECTION_NAME.find().limit(NUMBER)`
 - ▶ `db.student.find().limit(1)`

MongoDB skip() method

- ▶ In MongoDB, skip() method is used to skip the document. It is used with find() and limit() methods.

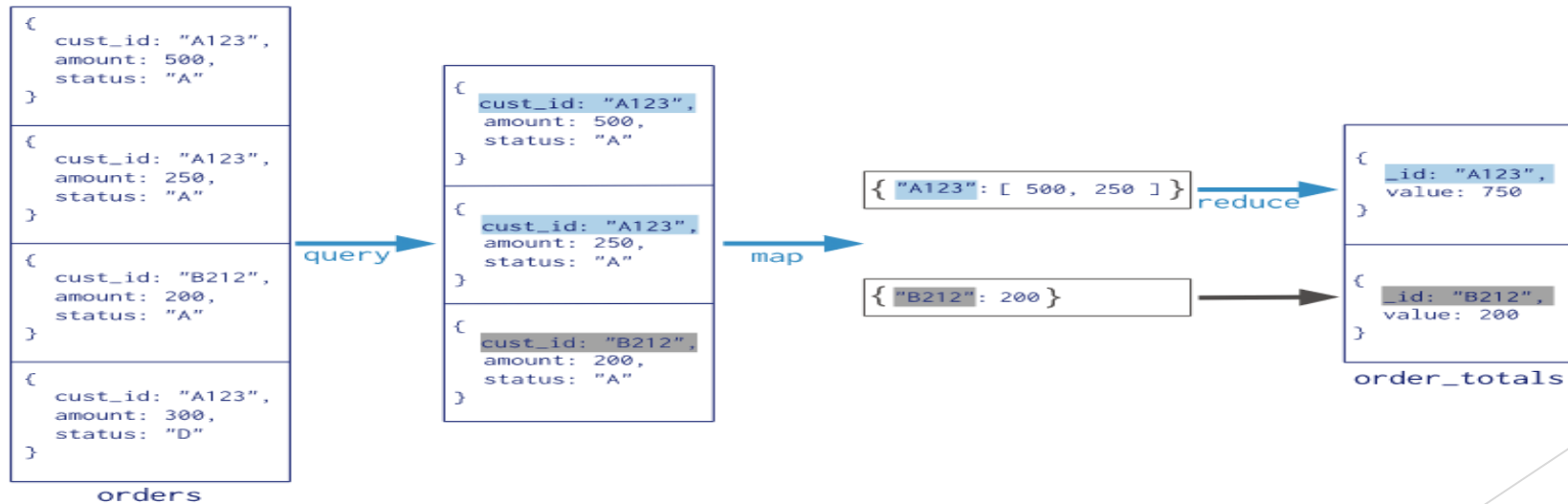
```
db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)
```

- ▶ `db.student.find().limit(1).skip(2)`

Map Reduce

- ▶ Map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results.
- ▶ To perform map-reduce operations, MongoDB provides the mapReduce database command.

Collection
↓
db.orders.mapReduce(
 map → function() { emit(this.cust_id, this.amount); },
 reduce → function(key, values) { return Array.sum(values) },
 query → { query: { status: "A" },
 output → out: "order_totals"
 }
)



mongoimport

- ▶ The `mongoimport` tool imports content from an Extended JSON, CSV, or TSV export created by `mongoexport` , or potentially, another third-party export tool.