**UPDATE OPERATIONS**

In Mongodb we have mainly we have three types of update operations:

* **updateOne(filters, values, options):**  Used to update a single document.
* **updateMany(filters, values, options):**  Used to update multiple documents at once.
* **replaceOne(filters, newDoc):** Used to replace an existing documents with new document. Where the objectId (\_id) remains the same.

**Update Operators:**

* **$set:**  This operators is used to update the field(s) of an existing document if exists, otherwise create a new field.
* db.<collection>.updateOne({name: “harshit”}, {$set: { age: 24 }})

[Set/Update the age field of the document where name = ‘harshit’]

* **$unset:**  this operator is used to remove/unset the field from the document.
* db.<collection>.updateOne({name: “harshit”}, {$unset: { age: 24 }})

[delete/remove the age field of the document where name = ‘harshit’]

* **$push:**  this operator is used to insert new value(s)/element(s) in the array field of the document(s).
  + If we want to push only a single element.
* db.<collection>.updateOne({name: “harshit”}, {$push: {bike: ‘Bullet 350’}})

[push the value ‘Bullet 350’ to the last of the bike array of the document where name = ‘harshit’]

* + If we want to push multiple element use **$each** operator.
* db.<collection>.updateOne({name: ‘harshit’}, {$push: {bike: {$each: [‘Bullet 350’, ‘Honda Activa’]}}})

[push the values ‘Bullet 350’ and ‘Honda Activa’ to the bike array of the document where name = ‘harshit’]

* + If we want to push multiple element with sorting use **$each**  with  **$sort** operator.
* db.<collection>.updateOne({name: ‘harshit’}, {$push: {bike: {$each: [‘Bullet 350’, ‘Honda Activa’]}}})

[push the values ‘Bullet 350’ and ‘Honda Activa’ to the bike array of the document where name = ‘harshit’]

* **$addToSet:**  This operator is used to add only the unique value/element to the end of the array. It can add only one element/value at a time.
* db.<collection>.updateOne({}, {$addToSet: {bike: ‘Bullet 350’ }})

[add the value ‘Bullet 350’ to the bike array if not already present.]

* **$pull:**  This operator is used to remove value(s)/element(s) from an array based on condition.
* db.<collection>.updateOne({}, {$pull: { cars:{average:{$lt: 15}}}})

[remove all the elements from the array cars whose average is less than 15]

* **$pop:**  this operator is used to remove a value/element in the array field of the document(s) from start (-1) or end (1).
* db.<collection>.updateOne({name: “harshit”}, {$pop: {bike: 1}})

[remove a single element from the last of the bike array of the document where name = ‘harshit’]

* db.<collection>.updateOne({name: “harshit”}, {$pop: {bike: -1}})

[remove a single element from the start of the bike array of the document where name = ‘harshit’]

* **$inc:**  This operator is used to make increment and decrement in the documents fields.

**Note: There is no other operator in mongodb to perform the decrement in the fields, $inc is used to perform both the operations.**

* db.<collection>.updateOne({ name: “harshit” }, { $inc: { weight: 4 } })

[increment the weight by 4 in the document where name = ‘harshit’]

* db.<collection>.updateOne({ name: “harshit” }, { $inc: { weight: -2 } })

[decrement the weight by 2 in the document where name = ‘harshit’]

* **$min:**  This operator updates a field only if the new value is less than the existing value. i.e only update if (new value < existing value).
* db.<collection>.updateOne({ name: “harshit” }, { $min: { age: 23 } })

[Only updates the age field with 23, if the existing age is greater than 23]

* **$max:**  This operator updates a field only if the new value is greater than the existing value. i.e only update if (new value > existing value).
* db.<collection>.updateOne({ name: “harshit” }, { $min: { age: 25 } })

[Only updates the age field with 25, if the existing age is less than 25]

* **$mul:**  This operator is used to multiply the value with the specified number.
* db.<collection>.updateOne({ name: “harshit” }, { $mul : { salary: 1.2 } })

[multiply the salary field by 1.2 and updates the new value to the field in the document where name = ‘harshit’]

* **$rename:**  This operator is used to rename a field.
* db.<collection>.updateMany({}, { $rename: { nationality: “nation” } })

[Update the field name of all the documents from nationality to nation]

* **$ :**  This is a array specific operator used to update a specific element of the array, which is matched by the query. It updates only the first matching document.
* db.<collection>.updateOne({‘hobbies’: {$elemMatch: {‘title’: ‘kabbadi’, frequency: ‘5’}}}, {$set: {‘hobbies.$.highFrequency’: true}})

[Add/Update the highFrequency field to matched embedded document]

* db.<collection>.updateOne({‘hobbies’: {$elemMatch: {‘title’: ‘kabbadi’, frequency: ‘5’}}}, {$set: {‘hobbies.$’:{title : “Kabbadi”, frequency: 6 }}})

[Update the entire matched embedded document]

* **$[ ]:**  This is a array specific operator used to update all the elements of the array. This has no query dependency.
* db.<collection>.updateOne({‘name’: ‘harshit’}, {$set: { ‘bikes.$[].isActive’: true }})

[add/update the isActive field in all the embedded documents in the array.]

* **$[ele]:**  This is a array specific operator used to update all the matching elements of the array. This has no query dependency. But a specified query is defined in the arrayFilters option.
* db.Person.updateMany({}, {$set: {'bikes.$[ele].isFuelEfficent': false}}, {arrayFilters: [{'ele.average': {$lt: 50}}]})

[add/update the isFuelEfficent field to all the embedded documents of the array where average < 50]

**Update Options:**

1. **upsert:**  This operator is used to insert a new document if no document is matched by the query.

* db.<collection>.updateOne({ name: “harshit” }, { $set: { age: 24 } }, { upsert: true })

[If no document is matched by the query, insert a new document by combining the query and the update values. {name: “harshit”, age: 24}]

1. **arrayFilters:**  This option is used to define the conditions for matching the elements inside the array for update operations.

* db.Person.updateMany({}, {$set: {'bikes.$[ele].isFuelEfficent': false}}, {arrayFilters: [{'ele.average': {$lt: 50}}]})

[add/update the isFuelEfficent field to all the embedded documents of the array where average < 50]