Replacing Document in collection

db.collection.replaceOne() method is used to replace the entire content of a document except the _id field. Based on the condition, whole document is replaced for the 1st matching record if there are multiple records matching the given criteria. The _id field remains unchanged after the replace. This method requires only field/value pairs.

Syntax

db.COLLECTION_NAME.replaceOne(SELECTION_CRITERIA, Replaced_DATA)

where

SELECTION_CRITERIA -- The selection criteria for the replacing the document
Replaced_DATA -- The replacement document on the matching criteria.

The replacement document may also have different fields from the original document. _id field may be given in new document to be replaced but it has no significance.

In the following example, we are replacing document matching with criteria “class”:”10th” with a new document having only 2 fields class and student count in school collection:

```
> db.collection.find();
{
   "_id" : ObjectId("5eb24d83db4e42dfda38301f"),  "class" : "11th",  "section" : "A section",  "student count" : "40"
}
{
   "_id" : ObjectId("5eb24d83db4e42dfda383020"),  "class" : "12th",  "section" : "C section",  "student count" : "30"
}
{
   "_id" : ObjectId("5eb24d83db4e42dfda383021"),  "class" : "12th",  "section" : "B section",  "student count" : "35"
}
{
   "_id" : ObjectId("5eb24f5db4e42dfda383022"),  "class" : "10th",  "section" : "A section",  "student count" : "40"
}
{
   "_id" : ObjectId("5eb24f1dddb4e42dfda383023"),  "class" : "12th",  "section" : "B section",  "student count" : "39"
}

> db.school.replaceOne({"class":"10th"},{"class":"10th","student count":"NIL"})
{ "acknowledged" : true,  "matchedCount" : 1,  "modifiedCount" : 1 }
> db.collection.find();
{
   "_id" : ObjectId("5eb24d83db4e42dfda38301f"),  "class" : "11th",  "section" : "A section",  "student count" : "40"
}
{
   "_id" : ObjectId("5eb24d83db4e42dfda383020"),  "class" : "12th",  "section" : "C section",  "student count" : "30"
}
{
   "_id" : ObjectId("5eb24d83db4e42dfda383021"),  "class" : "12th",  "section" : "B section",  "student count" : "35"
}
{
   "_id" : ObjectId("5eb24f5db4e42dfda383022"),  "class" : "10th",  "student count" : "NIL"
}
{
   "_id" : ObjectId("5eb24f1dddb4e42dfda383023"),  "class" : "12th",  "section" : "B section",  "student count" : "39"
}
```
**Update Operators**

Update methods support various field and array operators. It has the following syntax:

**Syntax**

```
{  
  <operator1>: { <field1>: <value1>, ... },  
  <operator2>: { <field2>: <value2>, ... },  
  ...  
}
```

Where operator can be field or array operator.

1. **Field operators**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$set</td>
<td>Sets the value of a field in a document.</td>
</tr>
<tr>
<td>$currentDate</td>
<td>Sets the value of the specified field to current date</td>
</tr>
<tr>
<td>$inc</td>
<td>Increments the value of the specified field by the specified amount.</td>
</tr>
<tr>
<td>$min</td>
<td>if the given value is less than the existing field value, then only updates the field</td>
</tr>
<tr>
<td>$max</td>
<td>if the given value is greater than the existing field value, then only updates the field</td>
</tr>
<tr>
<td>$mul</td>
<td>Used to multiply the value of the specified field by the given amount.</td>
</tr>
<tr>
<td>$rename</td>
<td>Renames the specified field.</td>
</tr>
<tr>
<td>$setOnInsert</td>
<td>Sets the value of a field if an update results in an insert of a document. Has no effect on update operations that modify existing documents.</td>
</tr>
<tr>
<td>$unset</td>
<td>Removes/deletes the specified field from a document.</td>
</tr>
</tbody>
</table>

2. **Array Operators**
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Used as a placeholder to update the first element of the array in the document that matches the criteria.</td>
</tr>
<tr>
<td>$[]</td>
<td>Used as a placeholder to update all elements of the array in the documents that matches the criteria.</td>
</tr>
<tr>
<td>$addToSet</td>
<td>Adds given elements to the array, but the existing elements will not be added.</td>
</tr>
<tr>
<td>$pop</td>
<td>Removes the first or last item of the array.</td>
</tr>
<tr>
<td>$pull</td>
<td>Removes all array elements that match specified criteria.</td>
</tr>
<tr>
<td>$push</td>
<td>Adds an item to the array.</td>
</tr>
<tr>
<td>$pullAll</td>
<td>Removes all matching values from an array.</td>
</tr>
</tbody>
</table>

**Modifiers with update method**

Update methods supports following 4 modifiers to update documents based on the criteria.

<table>
<thead>
<tr>
<th>Modifier name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$each</td>
<td>If we wish to add multiple items to array in update, $each is used with $push and $addToSet operators</td>
</tr>
<tr>
<td>$position</td>
<td>Used to add element at a particular position in array. It is used with $push operator. Without $position, element is added at the end of the array.</td>
</tr>
<tr>
<td>$slice</td>
<td>Used to limit the size of updated arrays when used with $push, i.e. limit the number of elements.</td>
</tr>
<tr>
<td>$sort</td>
<td>Modifies the order of the elements of an array when used with $push operation.</td>
</tr>
</tbody>
</table>

**Syntax of $each modifier**

```
{ $addToSet: { <field>: { $each: [ <value1>, <value2> ... ] } } }
```
• $each modifier can be Used alongwith the $addToArray set array operator to add multiple values to an array <field>. If the values do not exist in the <field> of array it adds to it, if exist do nothing.

```
{ $push: { <field>: { $each: [ <value1>, <value2> ... ] } } }
```

• $each modifier can be Used with the $push operator to append multiple values to an array <field>.

**Syntax of $position modifier**

```
{ $push: {
  <field>: {
    $each: [ <value1>, <value2>, ... ],
    $position: <num>
  }
}
```

Where <num> is the position for the element in the array, and array index starts with 0.

**Syntax of $slice modifier**

```
{ $push: {
  <field>: {
    $each: [ <value1>, <value2>, ... ],
    $slice: <num>
  }
}
```

The <num> can be:
- **Zero**  - To update the array <field> to an empty array [].
- **Negative**  - To update the array <field> to contain only the last <num> elements.
- **Positive**  - To update the array <field> contain only the first <num> elements.

**Syntax of $sort modifier**

```
{ $push: {
  <field>: {
    $each: [ <value1>, <value2>, ... ],
    $sort: <sort specification>
  }
}
```

where <sort specification> is

  1 for ascending
-1 for descending.

It may also be used with field i.e. { field: 1 } or { field: -1 }.

**Assignment**

1. how to replacement document in MongoDB?

2. define various operators used in update() command?