# Bitwise operator

MongoDB provides a $bit operator to perform a bitwise update of a field. This operator supports bitwise xor, bitwise or, and bitwise and operator.

**Important Points:**

- Use $bit operator only with integer fields(either 32-bit integer or 64-bit integer)
- To specify a field in embedded/nested documents or in an array use dot notation.
- All the numbers in the mongo shell are double not an integer. So, you need to use NumberInt() or the NumberLong() constructor to specify integers.
- You can use this operator in methods like update(), findAndModify(), etc., according to your requirements.

# Bitwise type:

| Name | Description |
|---|---|
| `$bitsAllClear` | Matches numeric or binary values in which a set of bit positions *all* have a value of `0`. |
| `$bitsAllSet` | Matches numeric or binary values in which a set of bit positions *all* have a value of `1`. |
| `$bitsAnyClear` | Matches numeric or binary values in which *any* bit from a set of bit positions has a value of `0`. |
| `$bitsAnySet` | Matches numeric or binary values in which *any* bit from a set of bit positions has a value of `1`. |

# Query

MongoDB Query allows retrieving data from the mongo  MongoDB database. Query provides simplicity in the process of fetching data from the database, it's similar to SQL queries in SQL Database language.

Here is a basic example of a MongoDB query:
Javascript
db.collections.find({ field: value })

To find students with both lobby and campus permission we use
db.students_permission.find({ permission
:{$bitsAllSets:[LOBBY_PERMISSION,CAMPUS_PERMISSION]} });

```
b> db.students_permission.find({permissions:{$bitsAllSet:[LOBBY_PERMISS
ON,CAMPUS_PERMISSION]}});
[
  {
    _id: ObjectId('6663ff4286ef416122dcfcd5'),
    name: 'George',
    age: 21,
    permissions: 6
  },
  {
    _id: ObjectId('6663ff4286ef416122dcfcd6'),
    name: 'Henry',
    age: 27,
    permissions: 7
  },
  {
    _id: ObjectId('6663ff4286ef416122dcfcd7'),
    name: 'Isla',
    age: 18,
    permissions: 6
  }
]
b>
```

# Projection

MongoDB provides a special feature that is known as Projection. It allows you to select only the necessary data rather than selecting whole data from the document

Because MongoDB projection is built on top of the existing **find()** method, you can use any projection query without significant modifications to the existing functions. Plus, projection is a key factor when finding user-specific data from a given data set.

```
db.collection_name.find({},{<field> : <value>})
```

## Operators in MongoDB Projection

Using the MongoDB projection method to retrieve specific data from a document will positively impact database performance because it reduces the workload of the find query, minimizing resource usage.

To reduce the workload, MongoDB offers the below operators that can be used within a projection query:

- $
- $elemMatch
- $slice
- $meta

## $ Operator

The $ operator is used to limit the contents of an array to project the first element that matches the query condition on the array field. Starting from MongoDB 4.4, if no query condition is present, the first element will be returned in the specified array.

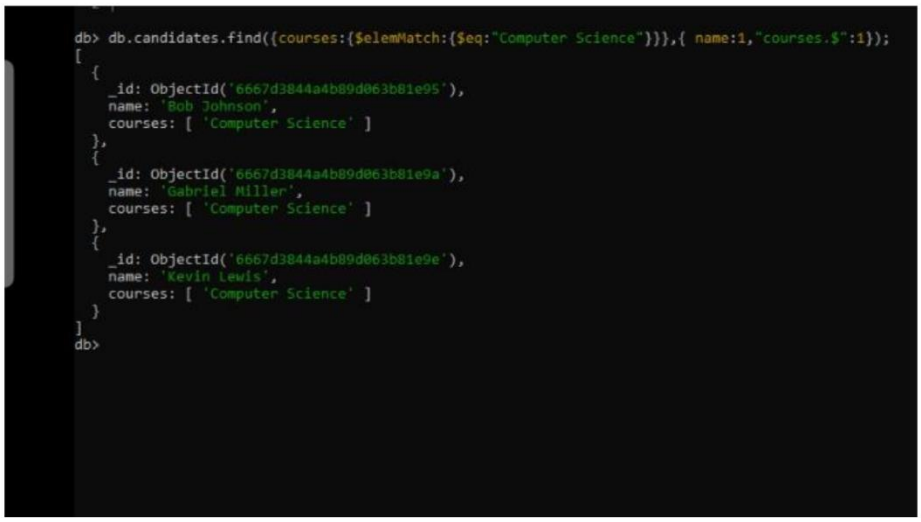
Syntax:

```
db.collection.find( { <array>: <condition> ... }, { "<array>.$": 1 } )
```

Limitations of the $ operator:

- Only a single $ operator can be used in a single query.
- The query must only consist of a single condition on the array field where it will be applied.
- The sort() function in the find() method will be applied before the $ operator. This function may cause the sort order not to be represented correctly.
- The $ operator can only be applied at the end of a field path. This restriction was introduced in MongoDB 4.4 to mitigate any formatting issues.
- The $slice projection expression cannot be used with $ operator as a part of the same expression.

We will use the “studentgrades” Collection to demonstrate the $ operator.




# $elemMatch

The $elemMatch operator matches documents that contain an array field with at least one element that matches all the specified query criteria.

Example:

{ <field>: { $elemMatch: { <query1>, <query2>, ... } } }

### Returned Element

Both the $ operator and the $elemMatch operator project the **first** matching element from an array based on a condition.

The $ operator projects the first matching array element from each document in a collection based on some condition from the query statement.

The $elemMatch  projection operator takes an explicit condition argument. This allows you to project based on a condition not in the query, or if you need to project based on multiple fields in the array's embedded documents.

```
db.players.find( {}, { games: { $elemMatch: { score: { $gt: 5 } } }, joined: 1, lastLogin: 1 } )
```

# Output: