BITWISE TYPES

Bitwise operators return data based on bit position conditions.

Syntax

{ $bit: { <field1>: { <or|and|xor>: <int> } } }

Bitwise

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`.

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

Example

```
db> const l_p=1;

db> const c_p=2;

db> db.sp.find({
... permissions:{$bitsAllSet:[l_p,c_p]}});
[
  {
    _id: ObjectId('666852859d450f507f54827e'),
    name: 'George',
    age: 21,
    permissions: 6
  },
  {
    _id: ObjectId('666852859d450f507f54827f'),
    name: 'Henry',
    age: 27,
    permissions: 7
  },
  {
    _id: ObjectId('666852859d450f507f548280'),
    name: 'Isla',
    age: 18,
    permissions: 6
  }
]
```

GEOSPATIAL

Geospatial queries in MongoDB allow you to perform queries on geospatial data, such as locations and shapes. There are two types of geospatial indexes in MongoDB: 2d and 2d sphere.

2d Indexes 2d indexes support queries that calculate geometries on a two-dimensional plane. To create a 2d index, use the db.collection.createIndex() method, specifying the location field as the key and the string literal "2d" as the index type:

```
db.collection.createIndex( { <location field> : "2d" } )
```

2dsphere Indexes 2dsphere indexes support queries that calculate geometries on a sphere, such as the surface of the Earth. To create a 2dsphere index, use the db.collection.createIndex() method, specifying the location field as the key and the string literal **"2dsphere"** as the index type:

```
db.collection.createIndex( { <location field> : "2dsphere" } )
```

Name	Description
`$geoIntersects`	Selects geometries that intersect with a GeoJSON geometry. The `2dsphere` index supports `$geoIntersects`.
`$geoWithin`	Selects geometries within a bounding GeoJSON geometry. The `2dsphere` and `2d` indexes support `$geoWithin`.
`$near`	Returns geospatial objects in proximity to a point. Requires a geospatial index. The `2dsphere` and `2d` indexes support `$near`.
`$nearSphere`	Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The `2dsphere` and `2d` indexes support `$nearSphere`.

Example

```
Current Mongosh Log ID: 66640228cfc60363b8cdcdf5
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&s
erverSelectionTimeoutMS=2000&appName=mongosh+2.2.6
Using MongoDB:          7.0.11
Using Mongosh:          2.2.6

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-06-08T11:39:34.025+05:30: Access control is not enabled for the da
tabase. Read and write access to data and configuration is unrestricted
------

test> use db
switched to db db
db> show dbs
admin    40.00 KiB
config  108.00 KiB
db      192.00 KiB
local    72.00 KiB
db> show collections
locations
students
students_permission
Please enter a MongoDB connection string (Default: mongodb://localhost/):
db>
```

To Find Location:

```
db> db.locations.find({
... location:{
... $geoWithin:{
... $centerSphere:[[-74.005,40.712],0.00621376]}}});
[
  {
    _id: 1,
    name: 'Coffee Shop A',
    location: { type: 'Point', coordinates: [ -73.985, 40.748 ] }
  },
  {
    _id: 2,
    name: 'Restaurant B',
    location: { type: 'Point', coordinates: [ -74.009, 40.712 ] }
  },
  {
    _id: 5,
    name: 'Park E',
    location: { type: 'Point', coordinates: [ -74.006, 40.705 ] }
  }
]
db>
```