Create and Query a Time Series Collection

On this page

This page shows how to create and query a time series collection with code examples.

Create a Time Series Collection

Before you can insert data into a time series collection, you must explicitly create the collection using either the `db.createCollection()` method or the `create` command:

```
db.createCollection(    "weather",    {       timeseries: {          timeField: "timesta
mp",          metaField: "metadata",          granularity: "hours"       }    })
```

Note

Feature Compatibility Version

You can only create time series collections on a system with featureCompatibilityVersion set to 5.0 or greater.

`timeseries` Object Fields

When creating a time series collection, specify the following options:

Field	Type	Description
`timeseries.timeField`	string	Required. The name of the field which contains the date in each time series document. Documents in a time series collection must have a valid BSON date as the value for the `timeField`.

Field	Type	Description
`timeseries.metaField`	string	Optional. The name of the field which contains metadata in each time series document. The metadata in the specified field should be data that is used to label a unique series of documents. The metadata should rarely, if ever, change. The name of the specified field may not be `_id` or the same as the `timeseries.timeField`. The field can be of any type.
`timeseries.granularity`	string	Optional. Possible values are: • `"seconds"` • `"minutes"` • `"hours"` By default, MongoDB sets the `granularity` to `"seconds"` for high-frequency ingestion. Manually set the `granularity` parameter to improve performance by optimizing how data in the time series collection is stored internally. To select a value for `granularity`, choose the closest match to the time span between consecutive incoming measurements. If you specify the `timeseries.metaField`, consider the time span between consecutive incoming measurements that have the same unique value for the `metaField` field. Measurements often have the same unique value for the `metaField` field if they come from the same source. If you do not specify `timeseries.metaField`, consider the time span between all measurements that are inserted in the collection.
`expireAfterSeconds`	number	Optional. Enable the automatic deletion of documents in a time series collection by specifying the number of seconds after which documents expire. MongoDB deletes expired documents automatically. See Set up Automatic Removal for Time Series Collections (TTL) for more information.

Other options allowed with the `timeseries` option are:

- `storageEngine`
- `indexOptionDefaults`
- `collation`
- `writeConcern`
- `comment`

Tip

See:

`db.createCollection()` and `create`.

Insert Measurements into a Time Series Collection

Each document you insert should contain a single measurement. To insert multiple documents at once, issue the following command:

```
db.weather.insertMany( [   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-18T00:00:00.000Z"),      "temp": 12   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-18T04:00:00.000Z"),      "temp": 11   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-18T08:00:00.000Z"),      "temp": 11   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-18T12:00:00.000Z"),      "temp": 12   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-18T16:00:00.000Z"),      "temp": 16   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-18T20:00:00.000Z"),      "temp": 15   }, {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-19T00:00:00.000Z"),      "temp": 13   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-19T04:00:00.000Z"),      "temp": 12   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-19T08:00:00.000Z"),      "temp": 11   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-19T12:00:00.000Z"),      "temp": 12   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-19T16:00:00.000Z"),      "temp": 17   },   {      "metadata": { "sensorId": 5578, "type": "temperature" },      "timestamp": ISODate("2021-05-19T20:00:00.000Z"),      "temp": 12   }] )
```

To insert a single document, use the `db.collection.insertOne()` method.

Tip

Optimize Insert Performance

To learn how to optimize inserts for large operations, see Optimize Inserts.

Query a Time Series Collection

You can query a time series collection the same way you would query a standard MongoDB collection.

To return one document from a time series collection, run:

```
db.weather.findOne({   "timestamp": ISODate("2021-05-18T00:00:00.000Z")})
```

Example output:

```
{   timestamp: ISODate("2021-05-18T00:00:00.000Z"),   metadata: { sensorId: 5578, type:
'temperature' },   temp: 12,   _id: ObjectId("62f11bbf1e52f124b84479ad")}
```

Run Aggregations on a Time Series Collection

For additional query functionality, use an aggregation pipeline such as:

```
db.weather.aggregate( [   {      $project: {         date: {            $dateToParts: {
date: "$timestamp" }         },         temp: 1      }   },   {      $group: {         _
id: {            date: {               year: "$date.year",               month: "$date.m
onth",               day: "$date.day"            }         },         avgTmp: { $avg:
"$temp" }      }   }] )
```

The example aggregation pipeline groups all documents by the date of the measurement and then returns the average of all temperature measurements that day:

```
 { "_id" : {    "date" : {       "year" : 2021,       "month" : 5,       "day" : 18    }
},  "avgTmp" : 12.714285714285714}{  "_id" : {    "date" : {       "year" : 2021,       "m
onth" : 5,       "day" : 19    }  },  "avgTmp" : 13}
```

← Time SeriesList Time Series Collections in a Database →