How to Insert Documents Conditionally in MongoDB: The “If Not Exists” Guide

Authored by

stats writer

November 30, 2025

MongoDB provides an "if not exists" clause to help users insert documents into a collection only if the document does not already exist. This helps to ensure data integrity by preventing duplicate data from being entered into the collection. The syntax for this clause is db.collection.insert({query}, {$set: {query}}, {upsert: true}). The upsert option is set to true, which means that if the document does not exist, it will be inserted, otherwise, the existing document will be left unchanged.

You can use the following syntax to insert a document into a collection in MongoDB only if it doesn't already exist:

```
db.teams.update(
{
team : 'Hornets'
},
{
$setOnInsert: {team: 'Hornets', points: '58', rebounds: '20'}
},
{upsert: true}
)
```

This particular code checks if the field "team" has a value of "Hornets." If this value exists, then nothing will happen.

However, if this value does not exist then it will insert a document with specific values for the "team", "points", and "rebounds" fields.

The following example shows how to use this syntax in practice.

Example: Insert if Not Exists in MongoDB

Suppose we have a collection called teams with the following documents:

```
db.teams.insertOne({team: "Mavs", points: 30, rebounds: 8})
db.teams.insertOne({team: "Spurs", points: 35, rebounds: 12})
db.teams.insertOne({team: "Rockets", points: 20, rebounds: 7})
db.teams.insertOne({team: "Warriors", points: 25, rebounds: 5})
db.teams.insertOne({team: "Cavs", points: 23, rebounds: 9})
```

Suppose we use the following code to attempt to insert a document for the team "Mavs":

```
db.teams.update(
{
team : 'Mavs'
},
{
$setOnInsert: {team: 'Mavs', points: '58', rebounds: '20'}
},
{upsert: true}
)
```

Since the field "team" already contains information for the "Mavs", none of the documents will be modified.

However, suppose we use the following code to insert a document for the team "Hornets":

```
db.teams.update(
{
team : 'Hornets'
},
{
$setOnInsert: {team: 'Hornets', points: '58', rebounds: '20'}
},
{upsert: true}
)
```

Since the field "team" does not already contain information for the "Hornets", a new document will be added to the collection with the values that we specified for each field.

Here's what the updated collection looks like:

```
{ _id: ObjectId("6203df361e95a9885e1e764a"),
team: 'Mavs',
points: 30,
rebounds: 8 }
{ _id: ObjectId("6203df361e95a9885e1e764b"),
team: 'Spurs',
points: 35,
rebounds: 12 }
{ _id: ObjectId("6203df361e95a9885e1e764c"),
team: 'Rockets',
```

points: 20,
rebounds: 7 }
{ _id: ObjectId("6203df361e95a9885e1e764d"),
team: 'Warriors',
points: 25,
rebounds: 5 }
{ _id: ObjectId("6203df361e95a9885e1e764e"),
team: 'Cavs',
points: 23,
rebounds: 9 }
{ _id: ObjectId("6203e17de42bfba74fc73325"),
team: 'Hornets',
points: '58',
rebounds: '20' }

Notice that a new document has been added for the "Hornets" team.

The following tutorials explain how to perform other common operations in MongoDB: