Documents

On this page

On this page

Document Structure

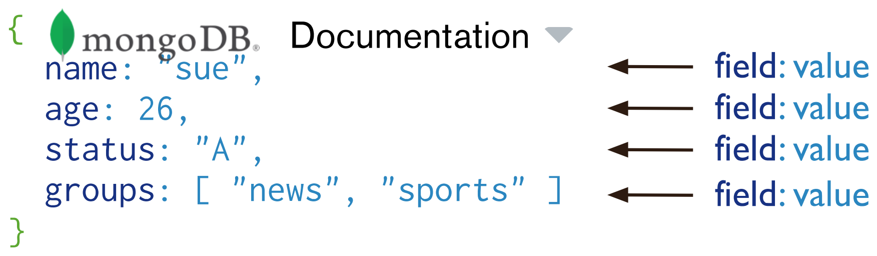
Dot Notation

Document Limitations

Other Uses of the Document Structure

Further Reading

MongoDB stores data records as BSON documents. BSON is a binary representation of JSON documents, though it contains more data types than JSON. For the BSON spec, see bsonspec.org. See also BSON Types.



```
{
  name: "sue",
  age: 26,
  status: "A",
  groups: [ "news", "sports" ]
}
```

Document Structure

MongoDB documents are composed of field-and-value pairs and have the following structure:

```
{
```

Give Feedback

```
    field1: value1,
    field2: value2,
    field3: value3,
    ...
    fieldN: valueN
}
```

The value of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents. For example, the following document contains values of varying types:

```
var mydoc = {
               _id: ObjectId("5099803df3f
               name: { first: "Alan", las
               birth: new Date('Jun 23, 1
               death: new Date('Jun 07, 1
               contribs: [ "Turing machin
               views : NumberLong(1250000
            }
```

The above fields have the following data types:

- `_id` holds an ObjectId.
- `name` holds an *embedded document* that contains the fields `first` and `last`.
- `birth` and `death` hold values of the *Date* type.
- `contribs` holds an *array of strings*.
- `views` holds a value of the *NumberLong* type.

Field Names

Give Feedback

Field names are strings.

Documents have the following restrictions on field names:

- The field name `_id` is reserved for use as a primary key; its value must be unique in the collection, is immutable, and may be of any type other than an array. If the `_id` contains subfields, the subfield names cannot begin with a (`$`) symbol.
- Field names **cannot** contain the `null` character.
- The server permits storage of field names that contain dots (`.`) and dollar signs (`$`).
- MongodB 5.0 adds improved support for the use of (`$`) and (`.`) in field names. There are some restrictions. See Field Name Considerations for more details.

BSON documents may have more than one field with the same name. Most MongoDB interfaces, however, represent MongoDB with a structure (e.g. a hash table) that does not support duplicate field names. If you need to manipulate documents that have more than one field with the same name, see the driver documentation for your driver.

Some documents created by internal MongoDB processes may have duplicate fields, but *no* MongoDB process will *ever* add duplicate fields to an existing user document.

Field Value Limit

Give Feedback

MongoDB 2.6 through MongoDB versions with featureCompatibilityVersion (fCV) set to `"4.0"` or earlier

For indexed collections, the values for the indexed fields have a Maximum Index Key Length. See Maximum Index Key Length for details.

Dot Notation

MongoDB uses the *dot notation* to access the elements of an array and to access the fields of an embedded document.

Arrays

To specify or access an element of an array by the zero-based index position, concatenate the array name with the dot (`.`) and zero-based index position, and enclose in quotes:

```
"<array>.<index>"
```

For example, given the following field in a document:

```
{
   ...
   contribs: [ "Turing machine", "Turing 
   ...
}
```

To specify the third element in the `contribs` array, use the dot notation `"contribs.2"`.

Give Feedback

For examples querying arrays, see:

- Query an Array
- Query an Array of Embedded Documents

TIP

See also:

- `$[]` all positional operator for update operations,
- `$[<identifier>]` filtered positional operator for update operations,
- `$` positional operator for update operations,
- `$` projection operator when array index position is unknown
- Query an Array for dot notation examples with arrays.

Embedded Documents

To specify or access a field of an embedded document with dot notation, concatenate the embedded document name with the dot (`.`) and the field name, and enclose in quotes:

```
"<embedded document>.<field>"
```

For example, given the following field in a document:

Give Feedback

```
{
   ...
   name: { first: "Alan", last: "Turing" }
   contact: { phone: { type: "cell", numb
   ...
}
```

- To specify the field named `last` in the `name` field, use the dot notation `"name.last"`.
- To specify the `number` in the `phone` document in the `contact` field, use the dot notation `"contact.phone.number"`.

For examples querying embedded documents, see:

- Query on Embedded/Nested Documents
- Query an Array of Embedded Documents

Document Limitations

Documents have the following attributes:

Document Size Limit

The maximum BSON document size is 16 megabytes.

The maximum document size helps ensure that a single document cannot use excessive amount of RAM or, during transmission, excessive amount of bandwidth. To store documents larger than the maximum size, MongoDB provides the GridFS API. See `mongofiles` and the documentation for your driver for more information about GridFS.

On this page

Give Feedback

Document Field Order

MongoDB preserves the order of the document fields following write operations *except* for the following cases:

- The `_id` field is always the first field in the document.
- Updates that include `renaming` of field names may result in the reordering of fields in the document.

The `_id` Field

In MongoDB, each document stored in a collection requires a unique _id field that acts as a primary key. If an inserted document omits the `_id` field, the MongoDB driver automatically generates an ObjectId for the `_id` field.

This also applies to documents inserted through update operations with upsert: true.

The `_id` field has the following behavior and constraints:

- By default, MongoDB creates a unique index on the `_id` field during the creation of a collection.
- The `_id` field is always the first field in the documents. If the server receives a document that does not have the `_id` field first, then the server will move the field to the beginning.

Give Feedback

_ If the `_id` contains subfields, the subfield names cannot begin

with a (`$`) symbol.

- The `_id` field may contain values of any BSON data type, other than an array, regex, or undefined.

> **WARNING**
>
> To ensure functioning replication, do not store values that are of the BSON regular expression type in the `_id` field.

The following are common options for storing values for `_id`:

- Use an ObjectId.
- Use a natural unique identifier, if available. This saves space and avoids an additional index.
- Generate an auto-incrementing number.
- Generate a UUID in your application code. For a more efficient storage of the UUID values in the collection and in the `_id` index, store the UUID as a value of the BSON `BinData` type.
  Index keys that are of the `BinData` type are more efficiently stored in the index if:
  - the binary subtype value is in the range of 0-7 or 128-135, and
  - the length of the byte array is: 0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16, 20, 24, or 32.

On this page

Give Feedback

- Use your driver's BSON UUID facility to generate UUIDs. Be aware that driver implementations may implement UUID serialization and deserialization logic differently, which may not be fully compatible with other drivers. See your driver documentation for information concerning UUID interoperability.

NOTE

Most MongoDB driver clients will include the `_id` field and generate an `ObjectId` before sending the insert operation to MongoDB; however, if the client sends a document without an `_id` field, the `mongod` will add the `_id` field and generate the `ObjectId`.

Other Uses of the Document Structure

In addition to defining data records, MongoDB uses the document structure throughout, including but not limited to: query filters, update specifications documents, and index specification documents

Query Filter Documents

Query filter documents specify the conditions that determine which records to select for read, update, and delete operations.

On this page

Give Feedback

You can use `<field>:<value>` expressions to specify the equality condition and query operator expressions.

```
{
  <field1>: <value1>,
  <field2>: { <operator>: <value> },
  ...
}
```

For examples, see:

- Query Documents
- Query on Embedded/Nested Documents
- Query an Array
- Query an Array of Embedded Documents

Update Specification Documents

Update specification documents use update operators to specify the data modifications to perform on specific fields during an `db.collection.update()` operation.

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

For examples, see Update specifications.

Give Feedback

Index Specification Documents

Index specification documents define the field to index and the index type:

```
{ <field1>: <type1>, <field2>: <type2>, .
```

Further Reading

For more information on the MongoDB document model, download the MongoDB Application Modernization Guide.

The download includes the following resources:

- Presentation on the methodology of data modeling with MongoDB
- White paper covering best practices and considerations for migrating to MongoDB from an RDBMS data model
- Reference MongoDB schema with its RDBMS equivalent
- Application Modernization scorecard

On this page

Give Feedback