**Frequently Asked Questions**

**Question:** What all data types MongoDB support?

**Ans:**

|  |  |  |
| --- | --- | --- |
| **Datatype** | **Number** | **Description** |
| Double | 1 | Used to stored floating point values |
| String | 2 | Commonly used datatype and it is UTF-8 valid |
| Object | 3 | Used for storing embedded objects |
| Array | 4 | Used for storing embedded objects |
| Binary Data | 5 | Used to store binary data |
| Undefined | 6 | Used to store undefined value |
| Object Id | 7 | Used to store document's ID |
| Boolean | 9 | Used to store Boolean value |
| Date | 10 | Used to store current date time in UNIX format. |
| Null | 11 | Used to store null value |

**Question:** Can we have fields that we can name as "\_id" to override the default "\_id" field?

**Ans:** Yes, we can override default \_id field. .

Eg :db.names.insert({\_id:2, name:"John"})

**Question:** What is the default data type in MongoDB?

**Ans:** When inserting numeric values the default datatype supported is double.

**Question:** Can we insert data from another file say xml file into collection?

**Ans:** First we have to validate the data using json validator before importing the file in MongoDB.

**Question**: 0 means not to display that field and :1 means to display that field but what exactly does the :1 in the query signify?

**Ans**:The projection document limits the fields to return for all matching documents. The projection document can specify the inclusion of fields or the exclusion of field and has the following form:

{ field1: <value>, field2: <value> ... }

The <value> can be any of the following:

* 1 or true to include the field in the return documents.
* 0 or false to exclude the field.
* Expression using a Projection Operators.

**Question**: How to put where condition on a field in an embedded document?

**Ans: {**

**\_id: 1,**

**name: "sue",**

**age: 19,**

**type: 1,**

**status: "P",**

**favorites: { artist: "Picasso", food: "pizza" },**

**finished: [ 17, 3 ],**

**badges: [ "blue", "black" ],**

**points: [**

**{ points: 85, bonus: 20 },**

**{ points: 85, bonus: 10 }**

**]**

**},**

We use the dot notation to return specific fields in an embedded document.

The following example specifies a projection to return: the \_id field, name field, status field, and the foodfield inside  the favorites document; the food field remains embedded in the favorites document.

db.users.find(

{ “favorites.food”:”pizza”},

{ name: 1, status: 1, "favorites.food": 1 }

)

**Question:** How to fetch the records in particular order like order by clause in SQL?

**Ans:** The sort() method specifies the order in which the query returns matching documents. You must apply sort() to the cursor before retrieving any documents from the database.

Example

A collection orders contain the following documents:

{ \_id: 1, item: { category: "cake", type: "chiffon" }, amount: 10 }

{ \_id: 2, item: { category: "cookies", type: "chocolate chip" }, amount: 50 }

{ \_id: 3, item: { category: "cookies", type: "chocolate chip" }, amount: 15 }

{ \_id: 4, item: { category: "cake", type: "lemon" }, amount: 30 }

{ \_id: 5, item: { category: "cake", type: "carrot" }, amount: 20 }

{ \_id: 6, item: { category: "brownies", type: "blondie" }, amount: 10 }

The following query specifies a sort on the amount field in descending order.

db.orders.find().sort( { amount: -1 } )

The query returns the following documents, in descending order of amount:

{ "\_id" : 2, "item" : { "category" : "cookies", "type" : "chocolate chip" }, "amount" : 50 }

{ "\_id" : 4, "item" : { "category" : "cake", "type" : "lemon" }, "amount" : 30 }

{ "\_id" : 5, "item" : { "category" : "cake", "type" : "carrot" }, "amount" : 20 }

{ "\_id" : 3, "item" : { "category" : "cookies", "type" : "chocolate chip" }, "amount" : 15 }

{ "\_id" : 1, "item" : { "category" : "cake", "type" : "chiffon" }, "amount" : 10 }

{ "\_id" : 6, "item" : { "category" : "brownies", "type" : "blondie" }, "amount" : 10 }

**Question:** What is the difference between insert() and insertmany()

**Ans:** **db.collection.insert()** inserts a single document or multiple documents into a collection. To insert a single document, pass a document to the method; to insert multiple documents, pass an array of documents to the method.

db.users.insert( [

{ name: "sue", age: 26, status: "A", }

])

**db.collection.insertMany()**  inserts multiple documents into a collection.

The following example inserts three new documents into the users collection. Each document has three fields name, age, and status. Since the documents do not specify an \_id field, MongoDB adds the \_idfield with an ObjectId value to each document.  See [Insert Behavior](https://docs.mongodb.com/manual/tutorial/insert-documents/#write-op-insert-behavior).

db.users.insertMany( [

{ name: "bob", age: 42, status: "A", },

{ name: "ahn", age: 22, status: "A", },

{ name: "xi", age: 34, status: "D", }

]

)

**Question:** can we specifiy datatypes and schema design while creating table

**Ans:** If application require strict schema design we can specify datatypes while creating table with the validator option in createCollection() method

**ex** db.createCollection( "contacts",

{ validator: { $or:

[

{ phone: { $type: "string" } },

{ email: { $regex: /@mongodb\.com$/ } },

{ status: { $in: [ "Unknown", "Incomplete" ] } }

]

}

} )