TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

MongoDB

EXPERIENTIAL learning factory

Introduction

- MongoDB is an open-source document database and leading NoSQL database.
- MongoDB is written in C++.
- MongoDB is a document database designed for ease of development and scaling.
- A record in MongoDB is a document, which is a data structure composed of field and value pairs.
- MongoDB documents are similar to JSON objects.
- The values of fields may include other documents, arrays, and arrays of documents.
- MongoDB stores documents in collections.
- Collections are similar to tables in relational databases.

Overview

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. Collections do not enforce a schema. Documents within a collection can have different fields.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Document

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

The advantages of using documents are:

- ❑ Documents (i.e. objects) correspond to native data types in many programming languages.
- ❑ Embedded documents and arrays reduce need for expensive joins.
- ❑ Dynamic schema supports fluent polymorphism.

- JSON stands for **J**ava**S**cript **O**bject **N**otation
- JSON is a lightweight format for storing and transporting data
- JSON is often used when data is sent from a server to a web page
- JSON is "self-describing" and easy to understand

JSON Syntax Rules

- ✓ Data is in name/value pairs
- ✓ Name should be always a string
- ✓ Data is separated by commas
- ✓ Curly braces hold objects
- ✓ Square brackets hold arrays

Advantages

- **Schema less** – MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another.
- Structure of a single object is clear.
- No complex joins.
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- **Ease of scale-out** – MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

Setting up the Environment

- Start Server

  Create a folder with name “data” and within this folder create one more folder “db”.

  Execute the command “mongod –dbpath ‘path/data/db’ ”.

- Start Client

  use the command “mongo” to run the client.

Starting Commands

- cls – clear the screen.
- show databases / show dbs – to show databases.
- To create a database and use.

  “use database_name”

  this returns a reference “db” of the database selected.
- Creating Collection.

  db.createColletion(“collection_name”)

➢ One Hard Disk can contain many Databases , one Database can contain many collections and a collection can contain many documents.

Deep into createCollection()

Syntax

db.createCollection("name", {options});

Following are the list of options you can use:

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.**
autoIndexId	Boolean	(Optional) If true, automatically create index on _id field.s Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.**
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

Inserting Documents

- insert({ }) or insert([{ } ,{ }])

  This method is used to insert one or more documents. This can take an json object or array of json objects as an argument.

  This returns WriteResult({ "nInserted" : no of documents inserted })

- insertOne({ })

  This method is used to insert one document. This takes json object as an argument.

- insertMany([{ }, { }])

  This method is used to many documents at a time. This method takes an array of json objects as an argument.

  insert() and insertMany() returns a json Object

Synatx: db.collection_name.insert()

_id field

- Every document in mongoDB will have a key “_id”.
- By default the value of “_id” key is an ObjectId(unique ID).
- As per our requirement we can over write the value of this “_id” field.
- “_id” field is having a unique constraint by default, therefore with in the collection we cannot have 2 or more documents with same value for “_id” field.

Retrieving Documents

- Syntax

  db.collection_name.find({ key: value }, { key : 0/1 })

  Selection Projection

- This find() can take two arguments, Selection and Projection.
- Both the arguments are optional.
- find() without arguments provides all the documents in the collection.
- Selection

  The process of selecting particular document from the collection.

- Projection

  The process of selecting the fields in the documents.

  By default “_id” field is selected.

Deep into find()

- find()

  Displays all the documents based on the selection and projection.

- findOne()

  Displays the very first document based on the selection and projection.

  Returns cursor object.

  We cannot use pretty() on findOne().

- limit(n)

  Displays the first “n” documents.

  Accepts integer as an argument.

- Count()

  Used to count the no of documents.

  limit() doesn’t show any impact on count().

Comparison Operators

- $eq – Matches values that are equal to a specified value. - { <field>: { $eq: <value> } }
- $ne – Matches all values that are not equal to a specified value. - {field: {$ne: value} }
- $gte - Matches values that are greater than or equal to a specified value. - {field: {$gte: value} }
- $gt – Matches values that are greater than a specified value. - {field: {$gt: value} }
- $lte – Matches values that are less than or equal to a specified value. - { field: { $lte: value} }
- $lt – Matches values that are less than a specified value. - {field: {$lt: value} }
- $in – Matches any of the values specified in an array.

  { field: { $in: [<value1>, <value2>, ... <valueN>] } }
- $nin – Matches none of the values specified in an array.

  { field: { $nin: [<value1>, <value2> ... <valueN>]} }

Logical Operators

- $and – Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.

  { $and: [{ <expression1> }, { <expression2> } , ... , { <expressionN> }] }

- $not – Inverts the effect of a query expression and returns documents that do *not* match the query expression.

  { field: { $not: { <operator-expression> } } }

- $nor – Joins query clauses with a logical NOR returns all documents that fail to match both clauses.

  { $nor: [{ <expression1> }, { <expression2> }, ... { <expressionN> }] }

- $or – Joins query clauses with a logical OR returns all documents that match the conditions of either clause

  { $or: [{ <expression1> }, { <expression2> }, ... , { <expressionN> }] }

Element Operators

- $exists – Matches documents that have the specified field.

  { field: { $exists: <boolean> } }

- $type – Selects documents if a field is of the specified type.

  { field: { $type: <data type> } }

Sorting

- To sort documents in MongoDB, you need to use **sort()** method.
- The method accepts a document containing a list of fields along with their sorting order.
- To specify sorting order 1 and -1 are used.
- 1 is used for ascending order while -1 is used for descending order.

Syntax

The basic syntax of sort() method is as follows:

```
db.COLLECTION_NAME.find().sort({KEY:1})
```

Update( )

- With the help of update() we can update the documents in a collection.
- Syntax

  db.collection_name.update({selection}, {updation}, {options})
- First argument is a query to select the document from the collection.
- Second argument is a document which replaces the selected document.
- Third argument is optional.
- By default update() will update only the first document selected.
- If we want to update multiple documents then we have to pass an option in the third argument.

  { multi: true }
- This multi option will work only if we use update operators.
- With out using the update operators this update() will replace the existing document instead of just updating it.

Update Operators

- $set - { $set: { <field1>: <value1>, ... } }
- $unset - { $unset: { <field1>: "", ... } }
- $rename - {$rename: { <field1>: <newName1>, <field2>: <newName2>, ... } }
- $inc - { $inc: { <field1>: <amount1>, <field2>: <amount2>, ... } }
- $mul - { $mul: { <field1>: <number1>, ... } }
- $min - { $min: { <field1>: <value1>, ... } }

The $min updates the value of the field to a specified value if the specified value is less than the current value of the field.

- $max - { $max: { <field1>: <value1>, ... } }

The $max operator updates the value of the field to a specified value if the specified value is greater than the current value of the field.

Delete Operation

remove()

- It is used to delete one or more documents from the collection.
- It deletes all the documents selected based on the condition.
- There is an option called as justOne. By default it is set to false. In order to delete only one document then we can set it to true.

deleteOne()

- It is used to delete only one document from the collection.

deleteMany()

- From mongoDB version 3.2 this method is introduced.
- It is used to delete multiple documents which satisfy the given condition(selection).

Aggregation

- Aggregations operations process data records and return computed results.
- Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result.
- The aggregation pipeline is a framework for data aggregation modeled on the concept of data processing pipelines.
- Documents enter a multi-stage pipeline that transforms the documents into aggregated results.
- The MongoDB aggregation pipeline consists of stages. Each stage transforms the documents as they pass through the pipeline. Pipeline stages do not need to produce one output document for every input document, some stages may generate new documents or filter out documents.

Aggregation Operators

- **$project** – Used to select some specific fields from a collection.
- **$match** – This is a filtering operation and thus this can reduce the amount of documents that are given as input to the next stage.
- **$group** – This does the actual aggregation as discussed above.
- **$sort** – Sorts the documents.
- **$skip** – With this, it is possible to skip forward in the list of documents for a given amount of documents.
- **$limit** – This limits the amount of documents to look at, by the given number starting from the current positions.
- **$unwind** – This is used to unwind document that are using arrays. When using an array, the data is kind of pre-joined and this operation will be undone with this to have individual documents again. Thus with this stage we will increase the amount of documents for the next stage.

Indexing

- Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a collection to select those documents that match the query statement. This scan is highly inefficient and require MongoDB to process a large volume of data.
- Indexes are special data structures, that store a small portion of the data set in an easy-to-traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field as specified in the index.

Syntax

The basic syntax of createIndex() method is as follows().

```
db.COLLECTION_NAME.createIndex({KEY:1})
db.COLLECTION_NAME.dropIndex({key: 1})
```

TESTYANTRA

SOFTWARE SOLUTIONS (INDIA) PVT. LTD.

Thank You !!!

No.01, 3rd Cross Basappa Layout, Gavipuram Extension,
Kempegowda Nagar, Bengaluru, Karnataka 560019

praveen.d@testyantra.com

www.testyantra.com

EXPERIENTIAL
learning factory