**TÀI LIỆU MONGODB**

1. **Kiểu dữ liệu**

* **Chuỗi:** Đây là kiểu dữ liệu được sử dụng phổ biến nhất để lưu giữ dữ liệu. Chuỗi trong MongoDB phải là UTF-8 hợp lệ.
* **Số nguyên:** Kiểu dữ liệu này được sử dụng để lưu một giá trị số. Số nguyên có thể là 32 bit hoặc 64 bit phụ thuộc vào Server của bạn.
* **Boolean**: Kiểu dữ liệu này được sử dụng để lưu giữ một giá trị Boolean (true/false).
* **Double**: Kiểu dữ liệu này được sử dụng để lưu các giá trị số thực dấu chấm động.
* **Min/ Max keys:** Kiểu dữ liệu này được sử dụng để so sánh một giá trị với các phần tử BSON thấp nhất và cao nhất.
* **Mảng:** Kiểu dữ liệu này được sử dụng để lưu giữ các mảng hoặc danh sách hoặc nhiều giá trị vào trong một key.
* **Timestamp:** Đánh dấu thời điểm một Document được sửa đổi hoặc được thêm vào.
* **Object:** Kiểu dữ liệu này được sử dụng cho các Document được nhúng vào.
* **Null:** Kiểu dữ liệu này được sử dụng để lưu một giá trị Null.
* **Symbol:** Kiểu dữ liệu này được sử dụng giống như một chuỗi
* **Date :** Kiểu dữ liệu này được sử dụng để lưu giữ date và time hiện tại trong định dạng UNIX time.
* **Object ID:** Kiểu dữ liệu này được sử dụng để lưu giữ ID của Document.
* **Binary data:** Kiểu dữ liệu này được sử dụng để lưu giữ dữ liệu nhị phân.
* **Code:** Kiểu dữ liệu này được sử dụng để lưu giữ JavaScrip code vào trong Document.
* **Regular expression:** Kiểu dữ liệu này được sử dụng để lưu giữ Regular Expresion.

| **Type** | **Number** | **Alias** |
| --- | --- | --- |
| Double | 1 | “double” |
| String | 2 | “string” |
| Object | 3 | “object” |
| Array | 4 | “array” |
| Binary data | 5 | “binData” |
| Undefined | 6 | “undefined” |
| ObjectId | 7 | “objectId” |
| Boolean | 8 | “bool” |
| Date | 9 | “date” |
| Null | 10 | “null” |
| Regular Expression | 11 | “regex” |
| DBPointer | 12 | “dbPointer” |
| JavaScript | 13 | “javascript” |
| Symbol | 14 | “symbol” |
| JavaScript (with scope) | 15 | “javascriptWithScope” |
| 32-bit integer | 16 | “int” |
| Timestamp | 17 | “timestamp” |
| 64-bit integer | 18 | “long” |
| Decimal128 | 19 | “decimal” |
| Min key | -1 | “minKey” |
| Max key | 127 | “maxKey” |

1. **Tạo và Xóa CSDL :**

**-** Cú pháp TẠO: **use** <DBNAME>

**-** Cú pháp Xóa: **db.dropDatabase()**

1. **Tạo Collection**

* **Tạo Collection không ràng buộc**
* Cú pháp tạo Collection:  **db.createCollection(name, options)**
* **VD: >db.createCollection("mycollection"){ "ok" : 1 }**
* Cú pháp xóa Collection: **db.nameCollection.drop()**
* **VD: >db.myCollection.drop();**
* Tạo giới hạn cho Collection:
  + Options:

|  |  |  |
| --- | --- | --- |
| **Trường** | **Kiểu** | **Miêu tả** |
| capped | Boolean | (Tùy ý) Nếu true, kích hoạt một Capped Collection. Đây là một Collection có kích cỡ cố định mà tự động **ghi đè các entry cũ nhất khi nó tiếp cận kích cỡ tối đa**. **Nếu bạn xác định là true, thì bạn cũng cần xác định tham số size** |
| autoIndexID | Boolean | (Tùy ý) Nếu true, tự động tạo chỉ mục trên các trường \_id. Giá trị mặc định là false |
| size | Số | (Tùy ý) Xác định kích cỡ tối đa (giá trị byte) cho một Capped Collection. **Nếu tham số capped là true, thì bạn cũng cần xác định trường này** |
| max | Số | (Tùy ý) Xác định số Document tối đa được cho phép trong một Capped Colleciton |

VD:

>db.createCollection("mycol", { capped : true, autoIndexID : true, size : 6142800, max : 10000 } )

{ "ok" : 1 }

* **Tạo Collection có ràng buộc: Sử dụng Data Modeling**
* **Schema Validation:**
  + **Cú Pháp :**

**db.createCollection("collection\_name", {**

**validator: {**

**$jsonSchema: {**

**{JSON\_schema}**

**}**

**}**

**})**

* **Các tùy chọn cơ bản nhất trong đối tượng JSON của Schema Validation bao gồm:** 
  + **"required": xác định các trường bắt buộc phải có trong các tài liệu trong collection.**
  + **"properties": xác định các thuộc tính của mỗi trường và kiểu dữ liệu của chúng.**
  + **"additionalProperties": xác định liệu có cho phép các trường không được định nghĩa trong "properties" hay không.**
  + **"minimum", "maximum", "exclusiveMinimum", "exclusiveMaximum": xác định các giới hạn giá trị tối thiểu và tối đa cho một trường.**
  + **“bsonType”: định nghĩa 1 kiểu dữ liệu**

**VD:**  **Ví dụ, đoạn mã sau đây định nghĩa một Schema Validation cho collection "students", trong đó yêu cầu các tài liệu phải có trường "name", "age" và "address", và trường "age" phải có giá trị tối thiểu là 18:**

**db.createCollection("students", {**

**validator: {**

**$jsonSchema: {**

**bsonType: "object",**

**required: [ "name", "age" ],**

**properties: {**

**name: {**

**bsonType: "string",**

**description: "must be a string and is required"**

**},**

**age: {**

**bsonType: "int",**

**minimum: 18,**

**description: "must be an integer and is required"**

**},**

**}**

**}**

**}**

**})**

1. **Document**

* **Dạng 1 document:**

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

* **Truy xuất Array:**
* **Cú pháp : "<array>.<index>"**

**VD: { ... contribs: [ "Turing machine" , "Turing test" , "Turingery" ], ... }**

**Để chỉ định phần tử thứ ba mảng acontribs, hãy sử dụng ký hiệu dấu chấm "contribs.2"**

* **Truy xuất Object:**
* **Cú pháp : "<embedded document>.<field>"**

**VD: { ... contact: { phone: { type: "cell" , number: "111-222-3333" } }, ... }**

**Để chỉ định number trong phone trong contact field, hãy sử dụng ký hiệu dấu chấm "contact.phone.number" .**

* **Chèn 1 document**
* Cú Pháp: **>db.COLLECTION\_NAME.insertOne(Many)({document})**

VD: >db.mycol.insert({

\_id: ObjectId(7df78ad8902c),

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

by: 'tutorials point',

url: 'http://www.tutorialspoint.com',

tags: ['mongodb', 'database', 'NoSQL'],

likes: 100

})

* **Chèn nhiều document**
* Cú Pháp: **>db.COLLECTION\_NAME.insertMany([{document\_1},{document\_2},.....])**

VD: db.inventory.insertMany([

{ item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },

{ item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },

{ item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }

])

* **Truy Vấn document**
* **TRUY VẤN VỚI VALUE LÀ 1 GIÁ TRỊ KHÁC MẢNG VÀ OBJECT**
* Cú pháp: **db.CollectionName.find({Điều Kiện},{Projection})** trả về các document thỏa mãn
* **Cú pháp của điều kiện {field : value}**

**+ VD: Điều kiện tìm trong Collection students có Name là Nam**

**Db.students.find({“Name”: Nam})**

* **Projection: Trả về các cột muốn hiển thị ra với: 1 là hiển thị và 0 là không hiển thị**

**+VD: với VD như trên chỉ hiện tên và tuổi của bạn Nam, không hiện \_id của Nam**

**Db.students.find({“Name”: Nam}, {Name:1, Age: 1,\_id:0 })**

* Sau khi Find() xong có thể sử dụng một số hàm khác để in ra dữ liệu theo ý muốn:
* **db.CollectionName.find({Điều Kiện},{Projection}).Options()**
* **các Options:**
* **limit(number): Giới hạn số lượng document trả về.**
* **skip(number): Bỏ qua một số document trong kết quả truy vấn.**
* **sort({field:value}): Sắp xếp kết quả theo thứ tự tăng hoặc giảm dần của một hoặc nhiều trường (value = 1: tăng, -1: giảm)**
* **count({điều kiện}): Đếm số lượng document trong kết quả truy vấn.**
* **aggregate(): Tính toán các giá trị tổng hợp dựa trên các phép toán như $group, $project, $match,...**

**VD:** Câu truy vấn dưới sẽ lấy ra tất cả các document trong collection **users** có trường **age** có giá trị lớn hơn 18, và chỉ trả về các trường **name** và **email**. Sau đó, nó sẽ sắp xếp kết quả theo thứ tự tăng dần của trường **name** và giới hạn số lượng document trả về là 10

* **db.users.find({"age": {"$gt": 18}}, {"name": 1, "email": 1}).sort({"name": 1}).limit(10)**
* **Điều kiện >, <, <=,>=, = và !=**

|  |  |  |  |
| --- | --- | --- | --- |
| **Phép toán** | **Cú pháp** | **Ví dụ** | **Mệnh đề WHERE tương đương** |
| Equality | {<field>:<value>} | db.mycol.find({"by":"tutorials point"}).pretty() | where by = 'tutorials point' |
| Less Than | {< field >:{$lt:<value>}} | db.mycol.find({"likes":{$lt:50}}).pretty() | where likes < 50 |
| Less Than Equals | {< field >:{$lte:<value>}} | db.mycol.find({"likes":{$lte:50}}).pretty() | where likes <= 50 |
| Greater Than | {< field >:{$gt:<value>}} | db.mycol.find({"likes":{$gt:50}}).pretty() | where likes > 50 |
| Greater Than Equals | {< field >:{$gte:<value>}} | db.mycol.find({"likes":{$gte:50}}).pretty() | where likes >= 50 |
| Not Equals | {< field >:{$ne:<value>}} | db.mycol.find({"likes":{$ne:50}}).pretty() | where likes != 50 |

* **Toán tử logic**
* **AND: dấu ‘,’ hoặc $and:[]**
* Cú pháp1**:** **>db.mycol.find({key1:value1, key2:value2});**
* Cú pháp 2**: >db.mycol.find( { $and: [{key1: value1},key2:value2}])**

**VD1: >db.students.find({"Name" :"Nam" ,"Age" : 18 }**

**VD2: >db. students.find({$and:[{ "Name" :"Nam" },{"Age" : 18 }]})**

* **OR: $or:[]**
* Cú pháp **: >db.mycol.find( { $or: [ {key1: value1}, {key2:value2}])**

**VD: >db. students.find({$or:[{ "Name" :"Nam" },{"Age" : 18 }]})**

* **Update document**
* Cú Pháp**: db.collection.updateMany(One)({Điều kiện truy vấn}, {update}, {options})**
* **Điều kiện truy vấn: biểu thức điều kiện**
* **update: $set{{“key\_1”: valueNew\_1, “key\_2”: valueNew\_2,...}}**
* **options: các tùy chọn cập nhật. Các tùy chọn này có thể bao gồm upsert, multi và writeConcern.**
  + **Multi : true : cập nhật nhiều document**

**VD: db.products.updateMany(**

**{ category: "electronics" }, // Tìm tất cả các sản phẩm có category là "electronics"**

**{ $set: { quantity: 50 } }, // Cập nhật trường quantity thành 50**

**{ multi: true } // Áp dụng cập nhật cho nhiều tài liệu (Không nhất thiết phải có)**

**)**

* **Delete document**
* Cú pháp: **db.collection.deleteOne(Many)({filter}, {options});**
* Trong đó:
  + collection: tên của bộ sưu tập chứa document cần xóa.
  + filter: biểu thức truy vấn để xác định document cần xóa.
  + options (tùy chọn): một đối tượng bao gồm các tùy chọn cho việc xóa document.
* Các tùy chọn trong options bao gồm:
  + writeConcern: cấu hình cho việc ghi dữ liệu và đảm bảo tính nhất quán trong cơ sở dữ liệu.
  + collation: cấu hình cho việc so sánh các chuỗi với một tập hợp các quy tắc nhất định.
  + hint: cho phép bạn chỉ định chỉ mục (index) sẽ được sử dụng để tìm kiếm document.

VD: **db.students.deleteOne({ “Name”: "Nam" })**

* **Hàm aggregate(): dùng để thực hiện các phép tính trên dữ liệu, chẳng hạn như phép nhóm, sắp xếp, lọc, tính toán, ghép bảng và tạo ra các báo cáo phức tạp.**
* Cú pháp: **db.collection.aggregate(pipeline)**
* **$ dùng để tham chiếu đến 1 field được chỉ định**
* Trong đó:
  + pipeline: là một mảng các giai đoạn của pipeline, mỗi giai đoạn được mô tả bởi một document chứa một hoặc nhiều trường, ví dụ { $match: {...} }, { $group: {...} }, { $project: {...} } và { $sort: {...} }. Các giai đoạn này được thực hiện theo thứ tự được liệt kê trong mảng pipeline.
* Các giai đoạn của pipeline bao gồm:
  + $match: lọc các document dựa trên một số điều kiện truy vấn.
  + $group: nhóm các document dựa trên các trường cụ thể và thực hiện các phép tính trên các nhóm này.
  + $project: chọn các trường cần hiển thị và định dạng lại kết quả đầu ra.
  + $sort: sắp xếp các document dựa trên các trường cụ thể.
* Trong $group thường sử dụng:

|  |  |  |
| --- | --- | --- |
| **Biểu thức** | **Miêu tả** | **Ví dụ** |
| $sum | Tổng giá trị được xác định từ tất cả Document trong Collection đó | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : "$likes"}}}]) |
| $avg | Tính trung bình của tất cả giá trị đã cho từ tất cả Document trong Collection đó | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$avg : "$likes"}}}]) |
| $min | Lấy giá trị nhỏ nhất của các giá trị từ tất cả Document trong Collection đó | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$min : "$likes"}}}]) |
| $max | Lấy giá trị lớn nhất của các giá trị từ tất cả Document trong Collection đó | db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$max : "$likes"}}}]) |
| $push | Chèn giá trị vào trong một mảng trong Document kết quả | db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$push: "$url"}}}]) |
| $addToSet | Chèn giá trị tới một mảng trong Document kết quả, nhưng không tạo các bản sao | db.mycol.aggregate([{$group : {\_id : "$by\_user", url : {$addToSet : "$url"}}}]) |
| $first | Lấy Document đầu tiên từ Source Document theo nhóm | db.mycol.aggregate([{$group : {\_id : "$by\_user", first\_url : {$first : "$url"}}}]) |
| $last | Lấy Document cuối cùng từ Source Document theo nhóm | db.mycol.aggregate([{$group : {\_id : "$by\_user", last\_url : {$last : "$url"}}}]) |