文書

各コレクションには複数のデータグループがあり、それぞれのエントリは「ドキュメント」と呼ばれます。ドキュメントは、従来のデータベースにおける「ROW」に相当します。

ドキュメントを追加する

使用するデータベースを指定したら、次のコマンドを使用できます。

```
db.コレクション名.insertOne(ドキュメント)
```

ドキュメントはJSON形式です。コマンドを入力すると、データベースマネージャーはオブジェクトを返します。このオブジェクトには、処理が正常に完了したかどうかを示すブール値と、ドキュメントに自動的に割り当てられたIDが含まれます。

複数のドキュメントを一度に追加することもできます。

```
db.コレクション名.insertMany([ドキュメント1,ドキュメント2...])
```

従来のデータベースとは異なり、すべてのドキュメントが同じ構造を持つ必要はありません。

あるドキュメントが特定のプロパティを持ち、同じコレクション内の別のドキュメントが異なるプロパティを持つ場合でも、問題にはなりません。ただし、論理的な理由から、同じプロパティを維持するように努めます。
構造。

ドキュメントを探す

コレクションに文書が集まったら、この情報を取得したいと考えています。
保存されます。

検索機能

このために、次の 2 つの関数にアクセスできます。

```
# すべての結果を取得します:
db.collectionName. ind(); # 最初の
結果を取得します: db.collectionName. indOne();
```

しかし、これは正確ではありません。より正確な検索を行いたい場合は、これらの関数にオブジェクトの形でパラメータを追加することができます。

```
db.collectionName.  ind({propertyToSearch:ValueToSearch}); # たとえば:

db.users.  ind({email:"maurice@gmail.com"});
```

比較演算子

これは特定の情報を見つけるのに適しています。しかし、ある値より大きい、または小さい情報を検索したい場合もあります。その場合は、以下のいずれかのキーワードが必要になります。

- $eq は等しい。
- $lt はより小さい。 $lte は以下。
- $gt はより大きい。 $gteは以
- 上。 $ ne は等しくない。 $in は配列内。
- $nin は配列内にない。
-
-
-

これらは同じように使用されます:

```
db.collectionName.  ind({propertyName:{$comparator:value}}); # 例: db.users.  ind({age:
{$gt:18}});
```

論理演算子

比較は、「AND」、「OR」演算子などと組み合わせて行うこともできます。

- $andドキュメントには次のすべての値が含まれている必要があります。
- $orドキュメントには次の値のいずれかが含まれている必要があります。
- $notドキュメントには次のすべての値が含まれていてはなりません。 $norドキュ
- メントには次の値のいずれかが含まれていてはなりません。

これらを使用するには、次の構文を使用します。

```
db.collectionName.  ind({$operator:[{property1:value},{property2:value}]}); # 例: db.users.  ind({$and:[{age:
{$gt:18}},{email:/
gmail.com$/]})
```

この最後の例からわかるように、不完全な情報を検索する場合は、REGEX を使用することもできます。

ドキュメントを更新する

ドキュメントを更新するには、次の構文を使用する必要があります。

```
# ドキュメントを編集します。
db.CollectionName.updateOne(　ilter,update); # 複数のドキュメ
ントを変更します。
db.CollectionName.updateMany(　ilter,update); # 1 つのドキュメ
ントを置き換えます。
db.CollectionName.replaceOne(　ilter,document);
```

- フィルターは、「　ind」メソッドで使用されるフィルターと同じです。updateは、更新される値プロパ
- ティを含む「更新演算子」を含むオブジェクトです。

次に例を示します。

```
db.users.updateOne({　irstname:"モーリス"},{$set:{age:42}})
```

この例では、「Maurice」という名前を持つユーザーを検索し、そのユーザーの「年齢」プロパティが42であることを示します。このプロパティが存在する場合は更新され、存在しない場合は作成されます。

使用されている更新演算子は「$set」であることに注意してください。これは、特定のフィールドに値を設定できるため、最も重要な演算子です。その他の演算子については、公式ドキュメントをご覧ください。

ドキュメントを削除する

追加、読み取り、更新はできました。あとは削除するだけです。それほど難しくはありません。主な方法は2つあります。

```
# 1 つのアイテムを削除します
db.CollectionName.deleteOne(　ilter); # 複数のアイテ
ムを削除しますdb.CollectionName.deleteMany(　ilter);
```

「検索」と同じフィルターを使用します。以下に簡単な例を示します。

```
db.users.deleteMany({age:{$lt:18}});
```

ここでは未成年のユーザーをすべて削除します。