**Обновление данных**

**Метод save**

Как и другие системы управления базами данных MongoDB предоставляет возможность обновления данных. Наиболее простым для использования является метод save. В качестве параметра этот метод принимает документ.

В этот документ в качестве поля можно передать параметр \_id. Если метод находит документ с таким значением \_id, то документ обновляется. Если же с подобным \_id нет документов, то документ вставляется.

Если параметр \_id не указан, то документ вставляется, а параметр \_id генерируется автоматически как при обычном добавлении через функцию insert:

|  |  |
| --- | --- |
| 1 | > db.users.save({name: "Eugene", age : 29, languages: ["english", "german",  "spanish"]}) |

В качестве результата функция возвращает объект WriteResult. Например, при успешном сохранении мы получим:

|  |  |
| --- | --- |
| 1 | WriteResult({"nInserted" : 1 }) |

**update**

Более детальную настройку при обновлении предлагает функция **update**. Она принимает три параметра:

* query: принимает запрос на выборку документа, который надо обновить
* objNew: представляет документ с новой информацией, который заместит старый при обновлении
* options: определяет дополнительные параметры при обновлении документов. Может принимать два аргумента: upsert и multi.

Если параметр upsert имеет значение true, то mongodb будет обновлять документ, если он найден, и создавать новый, если такого документа нет. Если же он имеет значение false, то mongodb не будет создавать новый документ, если запрос на выборку не найдет ни одного документа.

Параметр multi указывает, должен ли обновляться первый элемент в выборке (используется по умолчанию false, если данный параметр не указан) или же должны обновляться все документы в выборке (multi:true).

Например:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {name: "Tom", age : 25}, {upsert: true}) |

Теперь документ, найденный запросом {name : "Tom"}, будет перезаписан документом {"name": "Tom", "age" :"25"}.

Функция update() также возвращает объект WriteResult. Например:

|  |  |
| --- | --- |
| 1 | WriteResult({"nMatched" : 1, "nUpserted": 0, "nModified": 1}) |

В данном случае результат говорит нам о том, что найден один документ, удовлетворяющий условию, и один документ был обновлен.

**Обновление отдельного поля**

Часто не требуется обновлять весь документ, а только значение одного из его ключей. Для этого применяется оператор **$set**. Если документ не содержит обновляемое поле, то оно создается.

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom", age: 29}, {$set: {age : 30}}) |

Если обновляемого поля в документе нет, до оно добавляется:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom", age: 29}, {$set: {salary : 300}}) |

В данном случае обновлялся только один документ, первый в выборке. Указав значение multi:true, мы можем обновить все документы выборки:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$set: {name: "Tom", age : 25}},  {multi:true}) |

Для простого увеличения значения числового поля на определенное количество единиц применяется оператор **$inc**. Если документ не содержит обновляемое поле, то оно создается. Данный оператор применим только к числовым значениям.

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$inc: {age:2}}) |

**Удаление поля**

Для удаления отдельного ключа используется оператор **$unset**:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$unset: {salary: 1}}) |

Если вдруг подобного ключа в документе не существует, то оператор не оказывает никакого влияния. Также можно удалять сразу несколько полей:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$unset: {salary: 1, age: 1}}) |

**updateOne и updateMany**

Метод **updateOne** похож на метод update за тем исключением, что он обновляет только один документ.

|  |  |
| --- | --- |
| 1 | > db.users.updateOne({name : "Tom", age: 29}, {$set: {salary : 360}}) |

Если необходимо обновить все документы, соответствующие некоторому критерию, то применяется метод **updateMany()**:

|  |  |
| --- | --- |
| 1 | > db.users.updateMany({name : "Tom"}, {$set: {salary : 560}}) |

**Обновление массивов**

**Оператор $push**

Оператор $push позволяет добавить еще одно значение к уже существующему. Например, если ключ в качестве значения хранит массив:

|  |  |
| --- | --- |
| 1 | > db.users.updateOne({name : "Tom"}, {$push: {languages: "russian"}}) |

Если ключ, для которого мы хотим добавить значение, не представляет массив, то мы получим ошибку Cannot apply $push/$pushAll modifier to non-array.

Используя оператор $each, можно добавить сразу несколько значений:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$push: {languages: {$each: ["russian",  "spanish", "italian"]}}}) |

Еще пара операторов позволяет настроить вставку. Оператор **$position** задает позицию в массиве для вставки элементов, а оператор **$slice** указывает, сколько элементов оставить в массиве после вставки.

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$push: {languages: {$each: ["german",  "spanish", "italian"], $position:1, $slice:5}}}) |

В данном случае элементы ["german", "spanish", "italian"] будут вставляться в массив languages с 1-го индекса, и после вставки, в массиве останутся только 5 первых элементов.

**Оператор $addToSet**

Оператор $addToSet подобно оператору $push добавляет объекты в массив. Отличие состоит в том, что $addToSet добавляет данные, если их еще нет в массиве:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$addToSet: {languages: "russian"}}) |

**Удаление элемента из массива**

Оператор $pop позволяет удалять элемент из массива:

> db.users.update({name : "Tom"}, {$pop: {languages: 1}})

Указывая для ключа languages значение 1, мы удаляем первый элемент с конца. Чтобы удалить первый элемент сначала массива, надо передать отрицательное значение:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$pop: {languages: -1}}) |

Несколько иное действие предполагает оператор $pull. Он удаляет каждое вхождение элемента в массив. Например, через оператор $push мы можем добавить одно и то же значение в массив несколько раз. И теперь с помощью $pull удалим его:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$pull: {languages: "english"}}) |

А если мы хотим удалить не одно значение, а сразу несколько, тогда мы можем применить оператор $pullAll:

|  |  |
| --- | --- |
| 1 | > db.users.update({name : "Tom"}, {$pullAll: {languages: ["english", "german",  "french"]}}) |

**Удаление данных**

**Последнее обновление: 30.10.2015**

Для удаления документов в MongoDB предусмотрен метод remove:

|  |  |
| --- | --- |
| 1 | > db.users.remove({name : "Tom"}) |

Метод remove() возвращает объект WriteResult. При успешном удалении одного документа результат будет следующим:

|  |  |
| --- | --- |
| 1 | WriteResult({"nRemoved" : 1}) |

В итоге все найденные документы с name=Tom будут удалены. Причем, как и в случае с find, мы можем задавать условия выборки для удаления различными способами (в виде регулярных выражений, в виде условных конструкций и т.д.):

|  |  |
| --- | --- |
| 1  2 | > db.users.remove({name : /T\w+/i})  > db.users.remove({age: {$lt : 30}}) |

Метод remove также может принимать второй необязательный параметр булевого типа, который указывает, надо удалять один элемент или все элементы, соответствующие условию. Если этот параметр равен true, то удаляется только один элемент. По умолчанию он равен false:

|  |  |
| --- | --- |
| 1 | > db.users.remove({name : "Tom"}, true) |

Чтобы удалить разом все документы из коллекции, надо оставить пустым параметр запроса:

|  |  |
| --- | --- |
| 1 | > db.users.remove({}) |

**Удаление коллекций и баз данных**

Мы можем удалять не только документы, но и коллекции и базы данных. Для удаления коллекций используется функция **drop**:

|  |  |
| --- | --- |
| 1 | > db.users.drop() |

И если удаление коллекции пройдет успешно, то консоль выведет:

true

Чтобы удалить всю базу данных, надо воспользоваться функцией **dropDatabase()**:

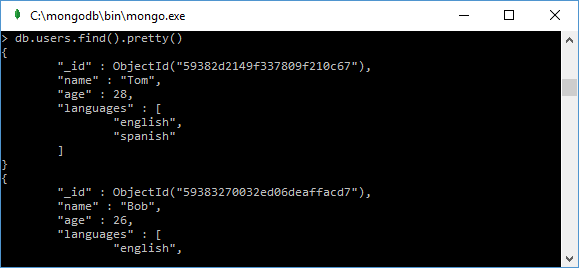
|  |  |
| --- | --- |
| 1 | > db.dropDatabase() |

**Выборка из БД**

Наиболее простой способом получения содержимого БД представляет использование функции find. Действие этой функции во многом аналогично обычному запросу SELECT \* FROM Table, который извлекает все строки. Например, чтобы извлечь все документы из коллекции users, созданной в прошлой теме, мы можем использовать команду db.users.find().

Для вывода документов в более удобном наглядном представлении мы можем добавить вызов метода **pretty()**:

|  |  |
| --- | --- |
| 1 | > db.users.find().pretty() |



Однако, если нам надо получить не все документы, а только те, которые удовлетворяют определенному требованию. Например, мы ранее в базу добавили следующие документы:

|  |  |
| --- | --- |
| 1  2  3 | > db.users.insertOne({"name": "Tom", "age": 28, languages: ["english",  "spanish"]})  > db.users.insertOne({"name": "Bill", "age": 32, languages: ["english",  "french"]})  > db.users.insertOne({"name": "Tom", "age": 32, languages: ["english",  "german"]}) |

Выведем все документы, в которых name=Tom:

|  |  |
| --- | --- |
| 1 | > db.users.find({name: "Tom"}) |

Такой запрос выведет нам два документа, в которых name=Tom.

Теперь более сложный запрос: нам надо вывести те объекты, у которых name=Tom и одновременно age=32. То есть на языке SQL это могло бы выглядеть так: SELECT \* FROM Table WHERE Name='Tom' AND Age=32. Данному критерию у нас соответствует последний добавленный объект. Тогда мы можем написать следующий запрос:

|  |  |
| --- | --- |
| 1 | > db.users.find({name: "Tom", age: 32}) |

Также несложно отыскать по элементу в массиве. Например, следующий запрос выводит все документы, у которых в массиве languages есть english:

|  |  |
| --- | --- |
| 1 | > db.users.find({languages: "english"}) |

Усложним запрос и получим те документы, у которых в массиве languages одновременно два языка: "english" и "german":

|  |  |
| --- | --- |
| 1 | > db.users.find({languages: ["english", "german"]}) |

Теперь выведем все документы, в которых "english" в массиве languages находится на первом месте:

|  |  |
| --- | --- |
| 1 | > db.users.find({"languages.0": "english"}) |

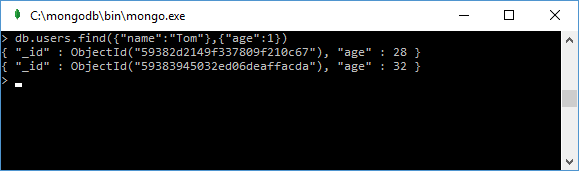
Соответственно если нам надо вывести документы, где english на втором месте (например, ["german", "english"]), то вместо нуля ставим единицу: languages.1.

**Проекция**

Документ может иметь множество полей, но не все эти поля нам могут быть нужны и важны при запросе. И в этом случае мы можем включить в выборку только нужные поля, использовав проекцию. Например, выведем только порцию информации, например, значения полей "age" у все документов, в которых name=Tom:

|  |  |
| --- | --- |
| 1 | > db.users.find({name: "Tom"}, {age: 1}) |

Использование единицы в качестве параметра {age: 1} указывает, что запрос должен вернуть только содержание свойства age.



И обратная ситуация: мы хотим найти все поля документа кроме свойства age. В этом случае в качестве параметра указываем 0:

|  |  |
| --- | --- |
| 1 | > db.persons.find({name: "Tom"}, {age: 0}) |

При этом надо учитывать, что даже если мы отметим, что мы хотим получить только поле name, поле \_id также будет включено в результирующую выборку. Поэтому, если мы не хотим видеть данное поле в выборке, то надо явным образом указать: {"\_id":0}

Альтернативно вместо 1 и 0 можно использовать true и false:

|  |  |
| --- | --- |
| 1 | > db.users.find({name: "Tom"}, {age: true, \_id: false}) |

Если мы не хотим при этом конкретизировать выборку, а хотим вывести все документы, то можно оставить первые фигурные скобки пустыми:

|  |  |
| --- | --- |
| 1 | > db.users.find({}, {age: 1, \_id: 0}) |

**Запрос к вложенным объектам**

Предыдущие запросы применялись к простым объектам. Но документы могут быть очень сложными по структуре. Например, добавим в коллекцию persons следующий документ:

|  |  |
| --- | --- |
| 1 | > db.users.insert({"name": "Alex", "age": 28, company: {"name":"microsoft",  "country":"USA"}}) |

Здесь определяется вложенный объект с ключом company. И чтобы найти все документы, у которых в ключе company вложенное свойство name=microsoft, нам надо использовать оператор точку:

|  |  |
| --- | --- |
| 1 | > db.users.find({"company.name": "microsoft"}) |

**Использование JavaScript**

MongoDB предоставляет замечательную возможность, создавать запросы, используя язык JavaScript. Например, создадим запрос, возвращающий те документы, в которых name=Tom. Для этого сначала объявляется функция:

|  |  |
| --- | --- |
| 1  2 | > fn = function() { return this.name=="Tom"; }  > db.users.find(fn) |



Этот запрос эквивалентен следующему:

|  |  |
| --- | --- |
| 1 | > db.users.find("this.name=='Tom'") |

Собственно только запросами область применения JavaScript в консоли mongo не ограничена. Например, мы можем создать какую-нибудь функцию и применять ее:

|  |  |
| --- | --- |
| 1  2  3 | > function sqrt(n) { return n\*n; }  > sqrt(5)  25 |