Нереляционные базы данных

Лабораторная работа 2

Обновление

## Обновление данных: замена и $set

В простейшей форме, `update` принимает 2 аргумента: селектор (where) для выборки и то, чем обновить соответствующее поле. Чтобы Roooooodles прибавил в весе, используем следующий запрос:

```
db.unicorns.update({name: 'Roooooodles'}, {weight: 590})
```

(Если в ходе экспериментов вы удалили данные из ранее созданной коллекции `unicorns`, сделайте всем документам `remove`, и вставьте их заново с помощью кода из лабораторной работы 1)

В реальной жизни, конечно, следует обновлять документы, выбирая их по `_id`, однако, поскольку я не знаю какой `_id` MongoDB сгенерировала для вас, будем выбирать по имени — `name`. Теперь, давайте взглянем на обновленную запись:

```
db.unicorns.find({name: 'Roooooodles'})
```

Вот и первый сюрприз, который нам преподнёс `update`. Документ не найден, поскольку второй параметр используется для **полной замены** оригинала. Иными словами, `update`нашёл документ по имени и заменил его целиком на новый документ (свой второй параметр). Вот в чём отличие от SQL-команды `UPDATE`. Иногда это идеальный вариант, который может использоваться для некоторых действительно динамических обновлений. Однако, если вам нужно всего лишь изменить пару полей, лучше всего использовать модификатор `$set`:

```
db.unicorns.update({weight: 590}, {$set: {name: 'Roooooodles', dob: new
Date(1979, 7, 18, 18, 44), loves: ['apple'], gender: 'm', vampires: 99}})
```

Это восстановит утерянные ранее поля. Поле `weight` не перезапишется, поскольку мы его не передали в запрос. Теперь, если выполнить:

```
db.unicorns.find({name: 'Roooooodles'})
```

мы получим ожидаемый результат. Таким образом, в первом примере правильно было бы обновить `weight` следующим образом:

```
db.unicorns.update({name: 'Roooooodles'}, {$set: {weight: 590}})
```

## Модификаторы обновления

Кроме `$set` можно использовать и другие модификаторы для разных изящных вещей. Все эти модификаторы обновления действуют над полями — так что ваш документ не окажется перезаписан целиком. Например, модификатор `$inc` служит для того, чтобы изменить поле

на положительную (увеличить) или отрицательную (уменьшить) величину. Например, если единорог Pilot был ошибочно награждён за убийство пары лишних вампиров, мы можем исправить эту ошибку следующим образом:

```
db.unicorns.update({name: 'Pilot'}, {$inc: {vampires: -2}})
```

Если Aurora внезапно пристрастилась к сладостям, мы можем добавить соответствующее значение к ее полю `loves` с помощью модификатора `$push`:

```
db.unicorns.update({name: 'Aurora'}, {$push: {loves: 'sugar'}})
```

Информацию об остальных модификаторах можно найти в разделе Обновление на сайте MongoDB.

## Обновление/вставка

Один из приятных сюрпризов операции обновления — это возможность обновления/вставки *(`upsert` от `update` — обновить и `insert` — вставить)* Обновление/вставка обновляет документ, если он найден, или создаёт новый — если не найден. Обновление/вставка — полезная вещь в некоторых случаях; когда столкнётесь с подобным, сразу поймёте. Чтобы разрешить вставку при обновлении, установите третий параметр в `true`.

Пример из жизни — счётчик посещений для веб-сайта. Если мы хотим в реальном времени видеть количество посещений страницы, мы должны посмотреть, существует ли запись, и — в зависимости от результата — выполнить `update` либо `insert`. Если опустить (или установить в `false`) третий параметр, следующий пример не сработает:

```
db.hits.update({page: 'unicorns'}, {$inc: {hits: 1}});
db.hits.find();
```

Однако, если разрешить вставку при обновлении, результаты будут иными:

```
db.hits.update({page: 'unicorns'}, {$inc: {hits: 1}}, true);
db.hits.find();
```

Поскольку документы с полем `page`, равным `unicorns`, не существуют, то будет создан новый документ. Если выполнить это вторично, существующий документ будет обновлён, и поле `hits` увеличится до 2.

```
db.hits.update({page: 'unicorns'}, {$inc: {hits: 1}}, true);
db.hits.find();
```

## Множественные обновления

Последний сюрприз метода `update` — это, то что он по умолчанию обновляет лишь один документ. До сих пор это было логично в случае с уже рассмотренными примерами. Однако, если выполнить что-нибудь вроде:

```
db.unicorns.update({}, {$set: {vaccinated: true }});
db.unicorns.find({vaccinated: true});
```

, то вы очевидно будете ожидать, что все единороги будут привиты *(`vaccinated`)*. Чтобы это сработало, нужно установить четвертый параметр в `true`:

```
db.unicorns.update({}, {$set: {vaccinated: true }}, false, true);
db.unicorns.find({vaccinated: true});
```

### Выводы к лабораторной работе

В этой лабораторной работе было завершено введение в основные CRUD операции над коллекциями. Мы детально рассмотрели `update` и увидели три его интересных режима работы. Во-первых, в отличие от SQL-команды `UPDATE`, в MongoDB `update` заменяет документ целиком. Из-за этого модификатор `$set` очень полезен. Во-вторых, `update` поддерживает интуитивно простое обновление/вставку, которое особенно полезно с модификатором `$inc`. И, наконец, в-третьих, по умолчанию, `update` обновляет лишь первый найденный документ.

Помните, что мы рассматриваем MongoDB с точки зрения её консоли. Используемые вами драйверы и библиотеки могут иметь иное поведение и реализовывать иной API. Например, драйвер для Ruby сливает два параметра в один хэш:`{:upsert => false, :multi => false}`.

### Требования к сдаче лабораторной работы

1. Отчет к лабораторной работе, содержащий пошаговое исполнение заданий и промежуточные результаты

### Задания к лабораторной работе

1. Проведите 3 замены с помощью команды update. Убедитесь, что документы были заменены, а не обновлены.
2. Проведите 3 обновления данных, используя модификатор $set. Убедитесь, что поля были обновлены, а документы не заменеы.
3. Проведите по 3 обновления с помощью модификаторов $inc, $push. При наличии интернета изучите один из модификаторов, не рассмотренных в лабораторной работе, покажите его работу, проведя 3 обновления.
4. Проведите 3 обновления с применением параметра создания документа, если он не был найден для обновления. Приведите пример, когда эта функция является полезной, и когда — вредной. Убедитесь, что при параметре создания, равном false, создание не производится, и поведение команды совпадает с опущенным параметром.

5. Проведите 3 массовых обновления с помощью соответствующего параметра команды update. Убедитесь, что по-умолчанию обновляется только один документ.