Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра «Проектирования информационно-компьютерных систем»

**ОТЧЕТ**

по лабораторной работе №6-7

по дисциплине «Системы и методы управления базами данных»

На тему: «ЗАПРОСЫ К БАЗЕ ДАННЫХ MONGODB.

ВЫБОРКА ДАННЫХ. ВЛОЖЕННЫЕ ОБЪЕКТЫ. ИСПОЛЬЗОВАНИЕ КУРСОРОВ. АГРЕГИРОВАННЫЕ ЗАПРОСЫ. ИЗМЕНЕНИЕ ДАННЫХ»

**Выполнил:** Студент

гр.914302 Григорян К

**Проверила:** Лукашевич А. Э.

Минск 2022

**Практическое задание 1:**

1. *Создайте коллекцию towns, включающую следующие документы:*

{name: "Punxsutawney ",

populatiuon: 6200,

last\_sensus: ISODate("2022-01-31"),

famous\_for: [""],

mayor: {

name: "Jim Wehrle"

}}

{name: "New York",

populatiuon: 22200000,

last\_sensus: ISODate("2022-07-31"),

famous\_for: ["status of liberty", "food"],

mayor: {

name: "Michael Bloomberg",

party: "I"}}

{name: "Portland",

populatiuon: 528000,

last\_sensus: ISODate("2022-07-20"),

famous\_for: ["beer", "food"],

mayor: {

name: "Sam Adams",

party: "D"}}

1. *Сформировать запрос, который возвращает список городов с независимыми мэрами (*party="I")*. Вывести только название города и информацию о мэре.*

db.towns.find({"mayor.party":"I"},{name:1,mayor:1,\_id:0})

1. *Сформировать запрос, который возвращает список беспартийных мэров (*party *отсутствует*)*. Вывести только название города и информацию о мэре.*

db.towns.find({"mayor.party":{$exists:false}},{name:1,mayor:1,\_id:0})

**Практическое задание 2:**

1. *Сформировать функцию для вывода списка самцов единорогов.*

*-*

1. *Создать курсор для этого списка из первых двух особей с сортировкой в лексикографическом порядке.*

*-*

1. *Вывести результат, используя forEach.*

*var cursor = db.unicorns.find({gender:"m"});null;cursor.sort({name:1}).limit(2);null; cursor.forEach(function(obj){ print(obj.name); })*

1. *Содержание коллекции единорогов unicorns:*

db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});

db.unicorns.insert({name: 'Roooooodles', 44), loves: ['apple'], weight: 575, gender: 'm', vampires: 99});

db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});

db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});

db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});

db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});

db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});

db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});

db.unicorns.insert ({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});

db.unicorns.insert ({name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165})

**Практическое задание 3:**

*Вывести количество самок единорогов весом от 500 до 600 кг.*

db.unicorns.find({weight:{$gte:500, $lte:600},gender : 'f'}).count()

Функция **distinct()** позволяет найти уникальные различающиеся значения для одного или нескольких полей документа.

Например, в нескольких документах определено name: "Tom". Нужно найти только уникальные различающиеся значения для одного из полей документа. Для этого можно воспользоваться функцией distinct:

|  |  |
| --- | --- |
|  | > db.users.distinct("name")  ["Tom", "Bill", "Alex"] |

**Практическое задание 4:**

*Вывести список предпочтений.*

db.unicorns.distinct("loves")

Использование метода aggregate аналогично применению выражения GROUP BY в SQL. Метод group принимает три параметра:

* $group: агрегатор, который вернет новый документ
* \_id: указывает на ключ, по которому надо проводить группировку ($+назване поля)
* $sum: оператор для вычисления.

Например:

> db.users.aggregate({"$group":{\_id:"$name",count:{$sum:1}}})

Пояснение. Параметр \_id указывает, что группировка будет проводиться по ключу name: \_id:"$name".

Значение параметра $sum инициализирует начальное значение поля count. Это поле будет представлять количество элементов для группы. На рисунке 1 приведен пример подсчет количества документов с разными именами name.

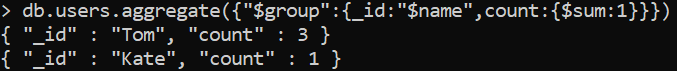


Рисунок 1 – Пример использования метода aggregate

(<https://docs.mongodb.com/manual/aggregation/>).

**Практическое задание 5:**

*Посчитать количество особей единорогов обоих полов.*

db.unicorns.aggregate({"$group":{\_id:"$gender",count:{$sum:1}}})

|  |  |
| --- | --- |
|  | WriteResult({"nInserted" : 1 })  **Практическое задание 6:**   1. *Выполнить команду:*   > db.unicorns.save({name: 'Barny', loves: ['grape'],  weight: 340, gender: 'm'})   1. *Проверить содержимое коллекции unicorns.* |

Функция **update** позволяет более **детально настроить обновление**.

Функция принимает три параметра:

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

Если параметр upsert имеет значение true, что 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, если данный параметр не указан) или же должны обновляться все документы в выборке.

Например:

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

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

Функция update() также возвращает объект WriteResult. Например:

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

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

**Практическое задание 7:**

1. *Для самки единорога* Ayna *внести изменения в БД: теперь ее вес 800, она убила 51 вампира.*
2. db.unicorns.update({name: "Ayna"},{$set: {weight: 800, vampires: 51}})
3. *Проверить содержимое коллекции unicorns.*

**Для обновления значения только одного из ключей** используется оператор **$set.**

Если документ не содержит обновляемое поле, то оно создается.

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

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

|  |  |
| --- | --- |
|  | > db.users.update({name : "Tom"},  {$set: {name: "Tom", age : 25, married : false}}, {multi:true})  **Практическое задание 8:**   1. *Для самца единорога* Raleigh *внести изменения в БД: теперь он любит рэдбул.*   *Для пары lowers:*  db.unicorns.update({name: "Raleigh"},{$set: {loves: ["apple", "lemon"]}})  *Для изменения одной:*  db.unicorns.update({name: "Raleigh"},{$set: {loves: "redbul"}})   1. *Проверить содержимое коллекции unicorns.* |
|  |  |

**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.

|  |  |
| --- | --- |
|  | > db.users.update({name : "Tom"}, {$inc: {salary:100}}) |

**Практическое задание 9:**

1. *Всем самцам единорогов увеличить количество убитых вапмиров на 5.*
2. db.unicorns.update({gender:'m'},{$inc:{ vampires:5}},{multi:true} )

*updateMany*

1. *Проверить содержимое коллекции unicorns.*

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

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

Если подобного ключа в документе не существует, то оператор не оказывает никакого влияния.

Можно удалять сразу несколько ключей:

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

**Практическое задание 10:**

1. *Изменить информацию о городе Портланд: мэр этого города теперь беспартийный.*
2. db.towns.update({name:"Portland"},{$unset:{"mayor.party":1}})

*(если надо обратно вернуть, то вот запрос:* db.towns.update({name:"Portland"},{$set:{"mayor.party":"D"}})

1. *Проверить содержимое коллекции towns.*

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

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

Если ключ, для которого нужно добавить значение, не представляет массив, то получится ошибка Cannot apply $push/$pushAll modifier to non-array.

**Практическое задание 11:**

1. *Изменить информацию о самце единорога Pilot: теперь он любит и шоколад.*
2. db.unicorns.update({name:"Pilot"},{$push:{loves:"chocolate"}})
3. *Проверить содержимое коллекции unicorns.*

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

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

Используя оператор **$each**, можно **добавить сразу несколько значений**:

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

**Практическое задание 12:**

1. *Изменить информацию о самке единорога* Aurora: *теперь она любит еще и сахар, и лимоны.*
2. db.unicorns.update({name:"Aurora"},{$addToSet:{loves:{$each:["lemon","sugar"]}}})
3. *Проверить содержимое коллекции unicorns.*

**Практическое задание 13:**

1. *Создайте коллекцию towns, включающую следующие документы:*

{name: "Punxsutawney ",

popujatiuon: 6200,

last\_sensus: ISODate("2022-01-31"),

famous\_for: ["phil the groundhog"],

mayor: {

name: "Jim Wehrle"

}}

{name: "New York",

popujatiuon: 22200000,

last\_sensus: ISODate("2022-07-31"),

famous\_for: ["status of liberty", "food"],

mayor: {

name: "Michael Bloomberg",

party: "I"}}

{name: "Portland",

popujatiuon: 528000,

last\_sensus: ISODate("2022-07-20"),

famous\_for: ["beer", "food"],

mayor: {

name: "Sam Adams",

party: "D"}}

1. *Удалите документы с беспартийными мэрами.*

db.towns.remove({"mayor.party":{$exists:false}})

1. *Проверьте содержание коллекции.*
2. *Очистите коллекцию.*

db.towns.remove({})

1. *Просмотрите список доступных коллекций.*

db.getCollectionNames()

**Практическое задание 1:**

1. *Создайте коллекцию зон обитания единорогов, указав в качестве идентификатора кратко название зоны, далее включив полное название и описание.*

db.place.insert({\_id:"tw", name:"town", description:"Cold and dirty"})

db.place.insert({\_id:"se",name:"sea",description:"Blue and warm"})

1. *Включите для нескольких единорогов в документы ссылку на зону обитания, использую второй способ автоматического связывания.*

db.unicorns.update({\_id:ObjectId("636e3bdc3c6dae7bb1e6b5ee")},{$set:

{place:{$ref:"place", $id: "sa"}}})

db.unicorns.update({\_id:ObjectId("6371ff1702a5e0966061956a")},{$set:

{place:{$ref:"place", $id: "sa"}}})

db.unicorns.update({\_id: ObjectId(

"6371fab202a5e0966061955f")},{$set:{place:{$ref:"place",$id: "tw"}}})

db.unicorns.update({\_id: ObjectId("6371fafc02a5e09660619560")},{$set:{place:{$ref:"place",$id:"fr"}}})

1. *Проверьте содержание коллекции единорогов.*
2. *Содержание коллекции единорогов unicorns:*

db.unicorns.insert({name: 'Horny', loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});

db.unicorns.insert({name: 'Roooooodles', 44), loves: ['apple'], weight: 575, gender: 'm', vampires: 99});

db.unicorns.insert({name: 'Solnara', loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});

db.unicorns.insert({name:'Ayna', loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});

db.unicorns.insert({name:'Kenny', loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});

db.unicorns.insert({name: 'Raleigh', loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});

db.unicorns.insert({name: 'Leia', loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});

db.unicorns.insert({name: 'Pilot', loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});

db.unicorns.insert ({name: 'Nimue', loves: ['grape', 'carrot'], weight: 540, gender: 'f'});

db.unicorns.insert {name: 'Dunx', loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165}

**Практическое задание 2:**

1. *Проверьте, можно ли задать для коллекции unicorns индекс для ключа name с флагом* unique*.*

db.unicorns.ensureIndex({"name" : 1}, {"unique" : true})

1. *Содержание коллекции единорогов unicorns:*

db.unicorns.insert({name: 'Horny', dob: new Date(1992,2,13,7,47), loves: ['carrot','papaya'], weight: 600, gender: 'm', vampires: 63});

db.unicorns.insert({name: 'Aurora', dob: new Date(1991, 0, 24, 13, 0), loves: ['carrot', 'grape'], weight: 450, gender: 'f', vampires: 43});

db.unicorns.insert({name: 'Unicrom', dob: new Date(1973, 1, 9, 22, 10), loves: ['energon', 'redbull'], weight: 984, gender: 'm', vampires: 182});

db.unicorns.insert({name: 'Roooooodles', dob: new Date(1979, 7, 18, 18, 44), loves: ['apple'], weight: 575, gender: 'm', vampires: 99});

db.unicorns.insert({name: 'Solnara', dob: new Date(1985, 6, 4, 2, 1), loves:['apple', 'carrot', 'chocolate'], weight:550, gender:'f', vampires:80});

db.unicorns.insert({name:'Ayna', dob: new Date(1998, 2, 7, 8, 30), loves: ['strawberry', 'lemon'], weight: 733, gender: 'f', vampires: 40});

db.unicorns.insert({name:'Kenny', dob: new Date(1997, 6, 1, 10, 42), loves: ['grape', 'lemon'], weight: 690, gender: 'm', vampires: 39});

db.unicorns.insert({name: 'Raleigh', dob: new Date(2005, 4, 3, 0, 57), loves: ['apple', 'sugar'], weight: 421, gender: 'm', vampires: 2});

db.unicorns.insert({name: 'Leia', dob: new Date(2001, 9, 8, 14, 53), loves: ['apple', 'watermelon'], weight: 601, gender: 'f', vampires: 33});

db.unicorns.insert({name: 'Pilot', dob: new Date(1997, 2, 1, 5, 3), loves: ['apple', 'watermelon'], weight: 650, gender: 'm', vampires: 54});

db.unicorns.insert ({name: 'Nimue', dob: new Date(1999, 11, 20, 16, 15), loves: ['grape', 'carrot'], weight: 540, gender: 'f'});

db.unicorns.insert {name: 'Dunx', dob: new Date(1976, 6, 18, 18, 18), loves: ['grape', 'watermelon'], weight: 704, gender: 'm', vampires: 165

**Практическое задание 3:**

1. *Получите информацию о всех индексах коллекции unicorns .*

db.unicorns.getIndexes()

1. *Удалите все индексы, кроме индекса для идентификатора.*

*db.unicorns.dropIndex("name\_1")*

1. *Попытайтесь удалить индекс для идентификатора.*

*-*

**Практическое задание 4:**

1. *Создайте объемную коллекцию numbers, задействовав курсор:*

for(i = 0; i < 100000; i++){db.numbers.insert({value: i})}

var cursor = db.numbers.find();null;

for(i = 0; i < 100000; i++){db.numbers.insert({value: i})};

1. *Выберите последних четыре документа.*

db.numbers.find().limit(4).sort({value: -1})