**Лабораторная работа №4**

**Конвейерная обработка документов коллекции**

Cледует выполнять команды, выделенные **жирным шрифтом**

В MongoDB реализован инструмент для выполнения цепочки операторов обработки документов коллекции — фреймворк агрегирования данных . На вход агрегирования поступают исходные документы коллекции, для которых последовательно выполняется цепочка операций. Документы на выходе одной операции поступают на вход следующей. Результат последней операции является результатом всей обработки. Конвейерную обработку выполняет метод aggregate: db.<Коллекция>.aggregate ([<Операция агрегации>,..]);¶

<Операция агрегации> содержит оператор — выполняемое действие и аргументы — атрибуты обрабатываемых входных документов, записанные в форме JSON-объекта:

{<Оператор>: [<Аргумент1>, < Аргумент2> …]}

или для одиночного аргумента операция агрегации имеет вид:

{<Оператор>: <Аргумент>}

Метод aggregate возвращает единственный документ с атрибутом "result", содержащий массив документов, элементы которого являются результатом агрегатной обработки исходных документов.

В цепочке могут использоваться следующие операции агрегации.

1. {$match: {<Селектор>}} Оператор $match фильтрует входные документы, отбирая соответствующие заданному селектору. В селекторе используются те же условия отбора, что и в методе find.

2. {$project: {<Спецификация атрибута>,...}} управляет набором атрибутов в выходных документах, добавляя новые, удаляя и переименовывая атрибуты входных документов. <Спецификация атрибута> имеет следующие формы:

· <Атрибут входного документа>:1 включает атрибут входного документа в выходной документ;

· \_id: 0 исключает вывод идентификатора документа (по умолчанию идентификаторы выводятся);

· <Новый атрибут >: <Выражение для значения атрибута> используется для создания новых атрибутов, значения которых вычисляются из атрибутов входного документа.

<Выражение для значения атрибута> записывается в форме JSON-объекта, содержащего массив:

{<Операция>: [<Операнд1>, <Операнд2>…]}

или

{<Операция: <Единственный операнд>}.

**Тип операции зависит от типа операндов**.

Поддерживаются следующие типы операций:

· булевские ($and, $or, $not),

· строковые ($concat, $substr, $toUpper и т. д.),

· арифметические ($add, $multiply и т. д.),

· операции сравнения ($eg, $gt, $gte и т. д.) и другие типы.

Например, в агрегации db.authors.aggregate ({$match:{"Au\_lname":/Лес/}},

{$project:{\_id: 0,

"Автор": {$concat: ["$Au\_lname"," ","$Au\_ fname"]},

"birthday":1}});

задан конвейер из двух операций:

· $match выбирает из коллекции authors документы об авторах, в фамилии которых присутствует слог «Лес», и передает следующей операции;

· $project для выбранных авторов отменяет вывод ключа \_id, создает новый атрибут "Автор", в котором соединяются ($concat) в одной строке фамилия и имя автора, затем выводится атрибут "birthday". Операция $concat работает начиная с версии 3.0.4.

3. Операция {$unwind: <Имя атрибута-массива>} «разворачивает» массив, создавая для каждого элемента указанного массива новый документ, в котором присутствуют все атрибуты исходного документа с одним из элементов разворачиваемого массива.

4. Операция группировки входных документов

{$group: {\_id: <Группирующее выражение>,

<Агрегирующий атрибут1>:

{<Оператор агрегации1>: <Выражение агрегирования1>}, ............}}

$group действует по аналогии с group by в SQL-Select: выполняет группировку путем создания нового документа из нескольких входных документов с одинаковым значением группирующего выражения и вычисления агрегирующих атрибутов по значениям атрибутов документов в группе. Способ вычисления агрегирующих атрибутов задается оператором агрегации.

Операторы агрегации:

· {$sum: <Выражение>} возвращает сумму числовых значений заданного выражения, вычисленного на документах в группе; нечисловые значения пропускаются;

· {$avg: <Выражение>} вычисляет арифметическое среднее числовых значений заданного выражения на документах в группе;

· {$first: <Выражение>} ({$last: < выражение >}) возвращает значение выражения, вычисленного по первому (последнему) документу в группе; используется, если определен порядок документов в группе;

· {$max: <Выражение>} ({$min: < выражение >}) возвращает максимальное (минимальное) значение выражения по документам в группе;

· {$push: <Выражение>} создает массив из результатов вычисления выражения для каждого документа в группе; порядок элементов в массиве не определен; · {$addToSet: <Выражение>} создает массив из уникальных значений результатов вычисления выражения для каждого документа в группе.

5. Операция сортировки документов {$sort: {<Атрибут1>: <Порядок>,, …}} упорядочивает на выходе поток входных документов, выполняя ступенчатую сортировку по перечисленным атрибутам. Порядок сортировки задается значениями: 1 — сортировка по возрастанию атрибута, 0 — по убыванию.

6. Операция {$skip: <Число документов>} задает количество первых пропускаемых документов. Передает оставшиеся документы следующей операции в конвейере.

7. Операция {$limit: <Число документов>} ограничивает число документов, передаваемых следующей операции в конвейере.

8. Операция {$out: "<Выходная коллекция>"} записывает итоговые документы конвейера агрегации в заданную коллекцию. Оператор $out должен быть последним этапом конвейера.

Рассмотрим пример конвейерной обработки коллекции документов. Пусть коллекция Writer содержит сведения о четырех писателях:

{Автор: "Кант И.", Страна: "Германия", Название: ["Критика чистого разума"], Тип:"Философия"},

{Автор: "Пелевин В. О.", Страна: "Россия", Название: ["Чапаев и Пустота"], Тип:"Художественная литература"},

{Автор: "Пелевин В. О.", Страна: "Россия", Название: ["Бетман Аполло"], Тип:"Фантастика"},

{Автор: "Лукьяненко C. В.", Страна: "Россия", Название: ["Ночной дозор", "Дневной дозор"], Тип:"Фантастика"}.

Построим конвейер для подсчета числа книг каждого типа для русских писателей. Процесс обработки требует фильтрации документов по селектору {Страна: "Россия"} и последующей группировки по атрибуту "Тип". В результате должны получиться документы следующего вида:

{"Тип": "Фантастика", "Количество": 3}.

Создадим новую базу данных

**> use База4**

и в ней новую коллекцию

Например, пусть в коллекции writer находятся 4 документа, содержащих данные о писателях и их книгах:

**> db.createCollection("writer")**

**Добавьте указанные выше документы в коллекцию самостоятельно**

**Создание конвейера обработки по шагам**

1. Выбор российских авторов.

**> db.Writer.aggregate ({$match:{Страна:/Рос/}});**

Получим массив из трех отфильтрованных документов:

{ "\_id" : ObjectId("5872a038bcae9444dc182648"), "Автор" : "Пелевин В. О.", "Страна" : "Россия", "Название" : [ "Чапаев и Пустота" ], "Тип" : "Художественная литература" }

{ "\_id" : ObjectId("5872a067bcae9444dc182649"), "Автор" : "Пелевин В. О.", "Страна" : "Россия", "Название" : [ "Бетман Аполло" ], "Тип" : "Фантастика" }

{ "\_id" : ObjectId("5872a08bbcae9444dc18264a"), "Автор" : "Лукьяненко C. В.", "Страна" : "Россия", "Название" : [ "Ночной дозор", "Дневной дозор" ], "Тип" : "Фантастика" }

1. В коллекции, полученной фильтрацией, разворачиваем массив названий книг.

**> db.Writer.aggregate ({$match:{Страна:/Рос/}}, {$unwind: "$Название"});**

В результате для каждой книги формируется отдельный документ. Последний документ для автора Лукьяненко C. В., имеющего две книги, разворачивается в два документа этого автора:

{ "\_id" : ObjectId("5872a038bcae9444dc182648"), "Автор" : "Пелевин В. О.", "Страна" : "Россия", "Название" : "Чапаев и Пустота", "Тип" : "Художественная литература" }

{ "\_id" : ObjectId("5872a067bcae9444dc182649"), "Автор" : "Пелевин В. О.", "Страна" : "Россия", "Название" : "Бетман Аполло", "Тип" : "Фантастика" }

{ "\_id" : ObjectId("5872a08bbcae9444dc18264a"), "Автор" : "Лукьяненко C. В.", "Страна" : "Россия", "Название" : "Ночной дозор", "Тип" : "Фантастика" }

{ "\_id" : ObjectId("5872a08bbcae9444dc18264a"), "Автор" : "Лукьяненко C. В.", "Страна" : "Россия", "Название" : "Дневной дозор", "Тип" : "Фантастика" }

Документы, содержащие несколько массивов, могут быть развернуты отдельно по каждому массиву или по нескольким любым массивам.

Например, пусть в коллекцию добавляется документ с двумя массивами, "Название" и "Формы":

**<db.Writer.insert ({Автор:"Пушкин А. С.", Страна: "Россия", Название: ["Руслан и Людмила", "Капитанская дочка"], Формы: ["Стихи", "Проза"]});**

Тогда его последовательное разворачивание по обоим массивам

**> db.Writer.aggregate ({$match:{Страна:/Рос/}}, {$unwind: "$Название"}, {$unwind: "$Формы"});**

приводит к созданию документов, в которых представлены все пары элементов из этих массивов:

{ "\_id" : ObjectId("5872a2febcae9444dc18264b"), "Автор" : "Пушкин А. С.", "Страна" : "Россия", "Название" : "Руслан и Людмила", "Формы" : "Стихи" }

{ "\_id" : ObjectId("5872a2febcae9444dc18264b"), "Автор" : "Пушкин А. С.", "Страна" : "Россия", "Название" : "Руслан и Людмила", "Формы" : "Проза" }

{ "\_id" : ObjectId("5872a2febcae9444dc18264b"), "Автор" : "Пушкин А. С.", "Страна" : "Россия", "Название" : "Капитанская дочка", "Формы" : "Стихи" }

{ "\_id" : ObjectId("5872a2febcae9444dc18264b"), "Автор" : "Пушкин А. С.", "Страна" : "Россия", "Название" : "Капитанская дочка", "Формы" : "Проза" }

Обратите внимание, в результат попадают только те документы, которые содержат разворачиваемые массивы. В приведенном примере разворачивание продемонстрировало формирование документов с семантическим несоответствием названия и формы произведения ("Название": "Капитанская дочка",

"Формы": "Стихи"). Причина несоответствия в отрыве связанных по смыслу данных при размещении названия и формы произведения в разных массивах. Для исключения подобных ошибок связанные данные должны размещаться в одном массиве. Например, можно представить сведения о названии и форме произведения в виде вложенного массива:

**<db.Writer.insert ({Автор:"Пушкин А. С.", Страна:"Россия", Произведение: [["Руслан и Людмила", "Стихи"], ["Капитанская дочка", "Проза"]]});**

Теперь разворачивание внешнего массива "Произведение" дает верный результат**:**

**db.Writer.find()**

{ "\_id" : ObjectId("58729ffebcae9444dc182647"), "Автор" : "Кант И.", "Страна" : "Германия", "Название" : [ "Критика чистого разума" ], "Тип" : "Философия" }

{ "\_id" : ObjectId("5872a038bcae9444dc182648"), "Автор" : "Пелевин В. О.", "Страна" : "Россия", "Название" : [ "Чапаев и Пустота" ], "Тип" : "Художественная литература" }

{ "\_id" : ObjectId("5872a067bcae9444dc182649"), "Автор" : "Пелевин В. О.", "Страна" : "Россия", "Название" : [ "Бетман Аполло" ], "Тип" : "Фантастика" }

{ "\_id" : ObjectId("5872a08bbcae9444dc18264a"), "Автор" : "Лукьяненко C. В.", "Страна" : "Россия", "Название" : [ "Ночной дозор", "Дневной дозор" ], "Тип" : "Фантастика" }

{ "\_id" : ObjectId("5872a2febcae9444dc18264b"), "Автор" : "Пушкин А. С.", "Страна" : "Россия", "Название" : [ "Руслан и Людмила", "Капитанская дочка" ], "Формы" : [ "Стихи", "Проза" ] }

{ "\_id" : ObjectId("5872a4b7bcae9444dc18264c"), "Автор" : "Пушкин А. С.", "Страна" : "Россия", "Произведение" : [ [ "Руслан и Людмила", "Стихи" ], [ "Капитанская дочка", "Проза" ] ] }

Так как новый документ о Пушкине А. С. не содержит массив "Название", он не повлияет на результаты следующих шагов.

3. Для подсчета количества книг каждого типа необходима группировка документов по типу книги. Для этого на третьем шаге в каждом документе, полученном разворачиванием массива "Название", оставляем необходимый для группировки атрибут "Тип" и добавляем новый атрибут "Количество" равным 1 во всех документах. На следующем шаге при группировке атрибут "Количество" будет суммироваться отдельно для каждого типа книг.

**> db.Writer.aggregate ({$match:{Страна:/Рос/}}, {$unwind: "$Название"},{$project:{\_id:0, "Тип":1, "Количество":{$add: [1]}}});**

Получим

{ "Тип" : "Художественная литература", "Количество" : 1 }

{ "Тип" : "Фантастика", "Количество" : 1 }

{ "Тип" : "Фантастика", "Количество" : 1 }

{ "Тип" : "Фантастика", "Количество" : 1 }

{ "Количество" : 1 }

{ "Количество" : 1 }

4. Следующий шаг — группировка документов по значениям атрибута "Тип" с использованием оператора $sum для суммирования атрибута "Количество" для документов в группе:

**> db.Writer.aggregate ({$match:{Страна:/Рос/}}, {$unwind: "$Название"}, {$project: {\_id:0, "Тип":1, "Количество": {$add: [1]}}}, {$group: {\_id: "$Тип", "Число книг": {$sum: "$Количество"}}});**

{ "\_id" : null, "Число книг" : 2 }

{ "\_id" : "Фантастика", "Число книг" : 3 }

{ "\_id" : "Художественная литература", "Число книг" : 1 }

**Задание для самостоятельной работы**

Используя приведенную ниже таблицу создать базу данных с коллекцией, содержащей данные таблицы. Документы коллекции должны иметь атрибуты, соответствующие заголовкам столбцов таблицы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Фамилия и инициалы | Страна | Год  рождения | Картины | Техника рисования |
| Перов В Г | Россия | 1834 | Рыболов | Холст, масло |
| Саврасов А К | Россия | 1830 | Грачи прилетели, Поселок | Холст, масло |
| Шишкин И И | Россия | 1832 | Рожь, Осенний лес | Холст, масло |
| Репин И Е | Россия | 1844 | Диоген и мальчик, | Картон, масло |
| Поленов В Д | Россия | 1844 | Деревня Окулова гора | Картон, масло |
| Васнецов В М | Россия | 1848 | Жница, Автопортрет | Холст, масло |
| Суриков В И | Россия | 1848 | Боярыня Морозова | Холст, масло |
| Ван Гог | Голландия | 1853 | Звездная ночь | Холст, масло |
| Мане Эдуард | Франция | 1832 | Флейтист | Холст, масло |
| Сезан Поль | Франция | 1839 | Мальчик в красном жилете | Холст, масло |

Создать конвейер обработки документов коллекции, выполнив следующие операции

1. Произвести выбор (операцию фильтрации) российских авторов.
2. В коллекции, полученной фильтрацией, разворачиваем массив названий книг.
3. Произвести группировку документов по технике рисования
4. Произвести группировку документов по значениям атрибута "Техника рисования" с использованием оператора $sum для суммирования атрибута "Количество" для документов в группе:

**Вопросы для защиты работы**

1. Синтаксис метода aggregate
2. Какие операции могут использоваться в цепочке в процессе агрегации данных?
3. Какие поддерживаются типы операций в процессе агрегации данных?
4. Перечислите операторы агрегации
5. Какие формы имеет<Спецификация атрибута> ?

**Содержание отчета**

1.Номер и название лабораторной работы

2. Экранные формы, показывающие порядок выполнения лабораторной

работы, и результаты, полученные в ходе её выполнения.

3.Постановка задачи для самостоятельной работы

4. Экранные формы, показывающие порядок выполнения задания для

Самостоятельной работы с соответствующими пояснениями, и результаты, полученные в ходе её выполнения.

5.Ответы на вопросы для защиты