

Булевы операторы

Булевы операторы

- `$ne` не равно
 - Дает отрицание равенства с указанным значением
- `$not`
 - Дает отрицание результата другого оператора
 - Большинство операторов имеют противоположный оператор:
`$in` - `$nin`, `$gt` - `$lte`
- `$or`
 - Логическая дизъюнкция
 - Используется при отборе значений разных ключей, иначе `$in`
- `$and`
 - Логическая конъюнкция
 - Воспринимается по умолчанию при перечислении ключей в условии
- `$exists`
 - Проверка наличия ключа
 - Эквивалент `{ключ:{$exists: false}}` - `{ключ: null}`

Агрегация

Метод aggregate()

В MongoDB для агрегации используется метод aggregate(). Данный метод имеет следующий синтаксис:

```
db.ИМЯ_КОЛЛЕКЦИИ.aggregate(ОПЕРАЦИЯ_АГРЕГАЦИИ)
```

пример:

```
db.developers.aggregate  
( [ {$group :{ _id : "Developers", total_salary: { $sum : "$salary" }}} ] )
```

результат:

```
{ "_id" : "Developers", "total_salary" : 10500 }
```

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

<https://docs.mongodb.com/manual/reference/method/db.collection.aggregate/>

Метод aggregate()

Collection

```
db.orders.aggregate( [  
  $match stage → { $match: { status: "A" } },  
  $group stage → { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }  
] )
```

{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }
{ cust_id: "A123", amount: 300, status: "D" }

orders

\$match →

{ cust_id: "A123", amount: 500, status: "A" }
{ cust_id: "A123", amount: 250, status: "A" }
{ cust_id: "B212", amount: 200, status: "A" }

\$group →

Results	
{	<code>_id: "A123",</code>
	total: 750
}	
{	<code>_id: "B212",</code>
	total: 200
}	

Список операций для агрегации

Выражение	Описание	Пример
\$sum	Суммирует указанные значения всех документов в коллекции.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{\$group : {_id : "\$title", total_salary : {\$sum : "\$salary"}}}])
\$avg	Рассчитывает среднее значение указанного поля поля для всех документов коллекции.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{\$group : {_id : "\$title", avg_salary : {\$avg : "\$salary"}}}])
\$min	Получает минимальное значение указанного поля документа в коллекции	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{\$group : {_id : "\$title", min_salary : {\$min : "\$salary"}}}])
\$max	Получает максимальное значение указанного поля документа в коллекции	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{\$group : {_id : "\$title", max_salary : {\$max : "\$salary"}}}])

Выражение	Описание	Пример
\$push	Вставляет значение в массив в результирующем документе.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{\$group : {_id : "\$title", skills : {\$push: "\$skills"}}}])
\$addToSet	Вставляет значение в массив в результирующем документе, но не создаёт дубликаты.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{\$group : {_id : "\$title", skills : {\$addToSet : "\$skills"}}}])
\$first	Получает первый документ из сгруппированных. Обычно используется вместе с сортировкой.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{\$group : {_id : "\$title", first_skill : {\$first : "\$skills"}}}])
\$last	Получает крайний документ из сгруппированных. Обычно используется вместе с сортировкой.	db.ИМЯ_КОЛЛЕКЦИИ.aggregate([{\$group : {_id : "\$title", last_skill : {\$last : "\$skills"}}}])

Связанные функции

\$project	Используется для выбора некоторых специальных полей из коллекции.
\$match	Операция фильтрации, которая может уменьшить количество документов, которые передаются для ввода в следующий этап.
\$group	Агрегация
\$sort	Сортирует документы
\$skip	С помощью данной команды можно проигнорировать список документов в имеющемся множестве.
\$limit	Ограничивает количество документов для вывода на количество, переданное методу, начиная, с текущей позиции.
\$unwind	Используется для “разматывания” документов, который используют массивы.

Пример aggregate

Подсчет количества по факультетам

```
db.Students.aggregate([{"$group":{"_id":"$Факультет",count:{$sum:1}}}]])
```

```
> db.Students.aggregate([{"$group":{"_id":"$Факультет",count:{$sum:1}}}]])
{ "_id" : "Курчатовский Институт", "count" : 12 }
{ "_id" : "Учебно-научный институт гравитации и космологии", "count" : 53 }
{ "_id" : "Институт иностранных языков", "count" : 2469 }
{ "_id" : "Институт прикладных технико-экономических исследований и экспертиз", "count" : 289 }
{ "_id" : "Кафедра сравнительной образовательной политики", "count" : 135 }
{ "_id" : "Филологический", "count" : 9450 }
{ "_id" : "Институт международных программ", "count" : 286 }
{ "_id" : "Экологический", "count" : 1900 }
{ "_id" : "Инженерный", "count" : 8727 }
{ "_id" : "Институт мировой экономики и бизнеса", "count" : 2253 }
{ "_id" : "Гуманитарных и социальных наук", "count" : 8581 }
{ "_id" : "Институт гостиничного бизнеса и туризма", "count" : 1812 }
{ "_id" : "Экономический", "count" : 7587 }
{ "_id" : "Физико-математических и естественных наук", "count" : 4263 }
{ "_id" : "Медицинский", "count" : 11681 }
{ "_id" : "Юридический", "count" : 6062 }
{ "_id" : "Научно-образовательный центр Нанотехнологии", "count" : 20 }
{ "_id" : "Аграрный", "count" : 3749 }
```

Команда group

```
db.collection.group({ key, reduce, initial [, keyf] [, cond] [, finalize] })
```

<https://docs.mongodb.com/manual/reference/method/db.collection.group/>

Команда group. Аргументы

- `key` - документ, описывающий, по каким полям группировать. Например, чтобы группировать по полю `category_id`, нужно в качестве ключа задать `{category id: true}`. Ключ может быть составным. Так, чтобы сгруппировать множество отзывов по `user_id` и `rating`, нужно задать ключ `{user_id: true, rating: true}`. Параметр `key` обязателен, если только не используется `keyf`.
- `keyf` - JavaScript-функция, которая, будучи применена к документу, генерирует ключ для этого документа. Это полезно, когда ключ группировки вычисляется динамически. Например, если требуется сгруппировать результирующий набор по дню недели, в который создавался документ, но это значение не хранится, то для генерации ключа можно воспользоваться функцией:

```
function ( doc) {return {day: doc.created_at.getDay()};
```

Эта функция генерирует ключи вида `{ day: 1}`. Отметим, что параметр `keyf` обязателен, если только не используется `key` для задания стандартного ключа.

Команда group. Аргументы

- `initial` - документ, используемый в качестве основы для порождения результатов агрегирования. При первом обращении к функции `reduce` этот начальный документ используется в качестве первого значения агрегатора и обычно содержит все агрегируемые ключи. Например, если для каждой группы вычисляется сумма поданных голосов и общее число документов в группе, то начальный документ будет выглядеть так:

`{vote_surn: 0.0, doc count: 0}`.

Этот параметр обязателен.

Команда group. Аргументы

- `reduce` - JavaScript-функция, выполняющая агрегирование. Она принимает два аргумента: текущий документ и документ-агрегатор, в котором хранятся результаты агрегирования. Начальным значением агрегатора будет документ `initial`. Вот пример функции `reduce` для агрегирования голосов и количества документов:

```
function ( doc, aggregator ) {  
    aggregator.doc_count += 1;  
    aggregator.vote_sum += doc.vote_count;  
}
```

Отметим, что функция `reduce` не обязана что-то возвращать; она просто должна модифицировать объект-агрегатор. Параметр `reduce` обязателен.

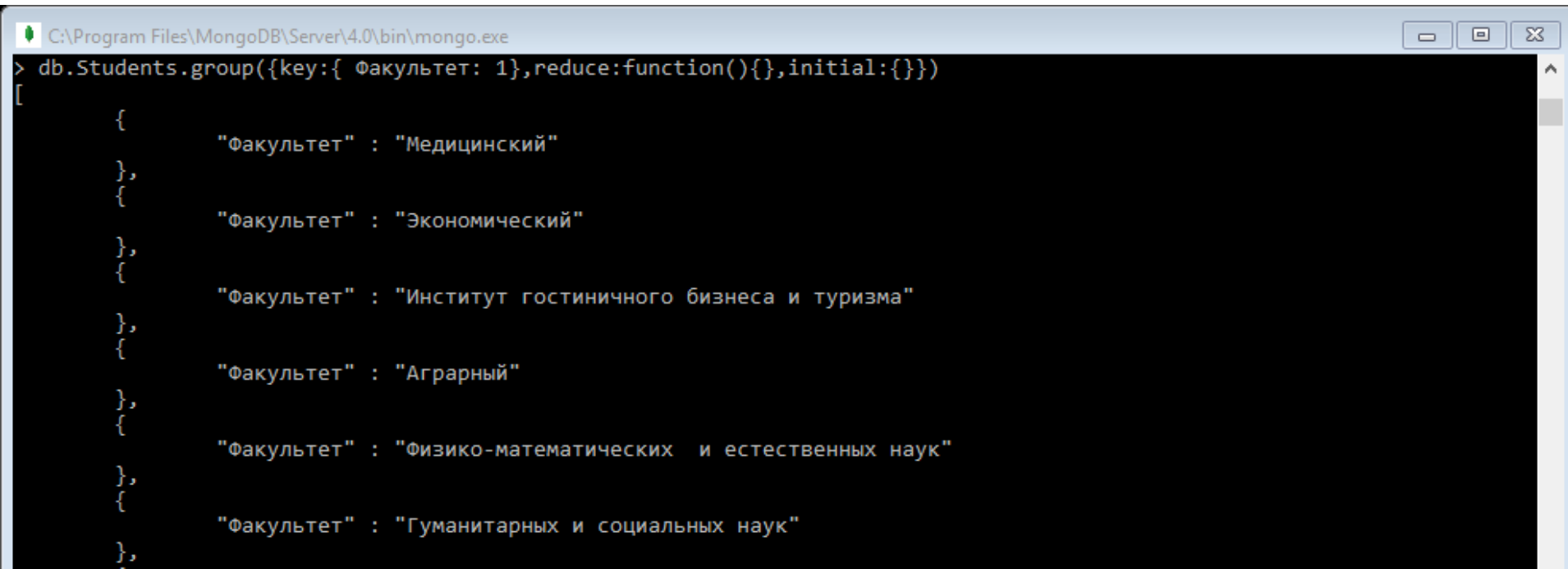
Команда group. Аргументы

- `cond` - селектор запроса, который отбирает подлежащие агрегированию документы. Если вы не хотите агрегировать всю коллекцию, то должны передать селектор. Например, чтобы агрегировать только отзывы, за которые подано более пять голосов, нужно задать такой селектор запроса: `{ vote count:{$gt: 5}}`.
- `finalize` - JavaScript-функция, которая применяется ко всем результирующим документам перед возвратом результирующего набора. Эта функция позволяет выполнить постобработку результатов операции группировки. Типичный пример - усреднение. Можно использовать уже существующие в результате группировки значения для порождения нового содержащего среднее:

```
function(doc) {  
    doc.average = doc.vote_count / doc.doc_count;  
}
```

Пример group

```
db.Students.group(  
  {  
    key: { Факультет: 1},  
    reduce: function () { },  
    initial: { }  
  }  
)
```



The screenshot shows a MongoDB shell window with the following content:

```
C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe  
> db.Students.group({key:{ Факультет: 1},reduce:function(){},initial:{}})  
[  
  {  
    "Факультет" : "Медицинский"  
  },  
  {  
    "Факультет" : "Экономический"  
  },  
  {  
    "Факультет" : "Институт гостиничного бизнеса и туризма"  
  },  
  {  
    "Факультет" : "Аграрный"  
  },  
  {  
    "Факультет" : "Физико-математических и естественных наук"  
  },  
  {  
    "Факультет" : "Гуманитарных и социальных наук"  
  },  
]
```

Пример group

```
db.Students.group(  
  {  
    key: { Факультет: 1 },  
    reduce: function( curr, result ) {  
      result.count++;  
    },  
    initial: { count: 0 }  
  }  
)
```

```
> db.Students.group({key: { Факультет: 1 },reduce: function( curr, result ) {result.count++;},initial: { count: 0 }})  
[  
  {  
    "Факультет" : "Медицинский",  
    "count" : 11681  
  },  
  {  
    "Факультет" : "Экономический",  
    "count" : 7587  
  },  
  {  
    "Факультет" : "Институт гостиничного бизнеса и туризма",  
    "count" : 1812  
  },  
  {  
    "Факультет" : "Аграрный",  
    "count" : 3749  
  },  
]
```


Пример group с ограничением

```
db.orders.group(  
  {  
    key: { ord_dt: 1, 'item.sku': 1 },  
    cond: { ord_dt: { $gt: new Date( '01/01/2012' ) } },  
    reduce: function( curr, result ) {  
      result.total += curr.item.qty;  
    },  
    initial: { total : 0 }  
  }  
)
```

```
SELECT ord_dt, item_sku,  
SUM(item_qty) as total  
FROM orders  
WHERE ord_dt > '01/01/2012'  
GROUP BY ord_dt, item_sku
```