Создание индексов и использование агрегационных функций.

Создание индексов

Создание индекса на поле author для ускорения поиска по авторам:

```
libraryDB> db.books.createIndex({ author: 1 })
author_1
libraryDB>
```

db.books.createIndex() – метод для создания индекса на коллекции books.
{ author: 1 } – это определение индекса:
author – поле, по которому будет создан индекс.
1 указывает на сортировку по возрастанию. Если бы мы хотели создать индекс по убыванию, использовали бы -1.

Создание составного индекса на поля year и price:

```
libraryDB> db.books.createIndex({ year: 1, price: 1 })
year_1_price_1
libraryDB>
```

db.books.createIndex() – метод для создания индекса на коллекции books.
{ year: 1, price: 1 } – это определение составного индекса:
year: 1 – сортировка по возрастанию для поля year.
price: 1 – сортировка по возрастанию для поля price.

Выполнение агрегационных запросов

Подсчет количества книг:

```
libraryDB> db.books.aggregate([
...   { $count: "totalBooks" }
... ])
[ { totalBooks: 7 } ]
libraryDB>
```

db.books.aggregate() – метод для выполнения агрегации на коллекции books.
{ $count: "totalBooks" } – оператор $count, который считает количество документов в коллекции и возвращает результат с именем поля "totalBooks".

Альтернативный способ подсчета с использованием метода countDocuments():

```
libraryDB> db.books.countDocuments()
7
libraryDB>
```

Агрегационный запрос, чтобы найти сумму цен всех книг:

```
libraryDB> db.books.aggregate([
...   {
...     $group: {
...       _id: null,
...       totalPrice: { $sum: "$price" }
...     }
...   }
... ])
[ { _id: null, totalPrice: 50.63 } ]
libraryDB>
```

db.books.aggregate() – метод для выполнения агрегации на коллекции books.
$group – оператор, который позволяет группировать документы.
_id: null – указывает, что все документы будут сгруппированы в один результат.
totalPrice: { $sum: "$price" } – создаем новое поле totalPrice, которое будет содержать сумму всех значений поля price.

Агрегационный запрос, чтобы вычислить среднюю цену всех книг:

```
libraryDB> db.books.aggregate([
...   {
...     $group: {
...       _id: null,
...       averagePrice: { $avg: "$price" }
...     }
...   }
... ])
[ { _id: null, averagePrice: 7.232857142857143 } ]
libraryDB>
```

db.books.aggregate() – метод для выполнения агрегации на коллекции books.
$group – оператор, который позволяет группировать документы.
_id: null – указывает, что все документы будут сгруппированы в один результат.
averagePrice: { $avg: "$price" } – создаем новое поле averagePrice, которое будет содержать среднее значение всех цен из поля price.

Если нужно вычислить среднюю цену только для определенных книг (например, только доступных), можно добавить стадию `$match` перед `$group`:

```
libraryDB> db.books.aggregate([
...   { $match: { available: true } },
...   {
...     $group: {
...       _id: null,
...       averagePrice: { $avg: "$price" }
...     }
...   }
... ])
[ { _id: null, averagePrice: 6.67 } ]
libraryDB>
```

{ $match: { available: true } }, // Фильтруем только доступные книги

Агрегационный запрос, который группирует книги по жанрам и подсчитывает количество книг в каждом жанре:

```
libraryDB> db.books.aggregate([
...   { $unwind: "$genres" },
...   {
...     $group: {
...       _id: "$genres",
...       count: { $sum: 1 }
...     }
...   },
...   { $sort: { count: -1 } }
... ])
[
  { _id: 'Classic', count: 4 },
  { _id: 'Dystopian', count: 2 },
  { _id: 'Tragedy', count: 1 },
  { _id: 'Historical', count: 1 },
  { _id: 'Political Fiction', count: 1 },
  { _id: 'Coming of Age', count: 1 },
  { _id: 'Science Fiction', count: 1 },
  { _id: 'Historical Fiction', count: 1 },
  { _id: 'Fantasy', count: 1 }
]
libraryDB>
```

$unwind: Этот оператор "разворачивает" массив genres, создавая отдельный документ для каждого элемента в массиве. Это позволяет нам группировать книги по каждому жанру отдельно.
$group: Оператор, который группирует документы по значению genres и создает поле count, которое подсчитывает количество документов в каждой группе с помощью $sum: 1.
_id: "$genres": Указывает, что группировка происходит по полю genres.
$sort: Оператор для сортировки результатов по количеству книг в порядке убывания.

Сортировка и лимитирование

Агрегационный запрос, который сортирует книги по году выпуска в порядке убывания и возвращает только 3 самых новые книги:

```
libraryDB> db.books.aggregate([ { $sort: { year: -1 } }, { $limit: 3 }] )
[
  {
    _id: ObjectId('670a2f218aa6c98a9cc73c02'),
    title: 'To Kill a Mockingbird',
    author: 'Harper Lee',
    year: 1960,
    genres: [ 'Classic', 'Historical' ],
    price: 5.49,
    available: true
  },
  {
    _id: ObjectId('670a2f2d8aa6c98a9cc73c05'),
    title: 'The Catcher in the Rye',
    author: 'J.D. Salinger',
    year: 1951,
    genres: [ 'Classic', 'Coming of Age' ],
    price: 8.29,
    available: false
  },
  {
    _id: ObjectId('670a2f218aa6c98a9cc73c01'),
    title: '1984',
    author: 'George Orwell',
    year: 1949,
    genres: [ 'Dystopian', 'Political Fiction' ],
    price: 8.99,
    available: false
  }
]
libraryDB>
```

$sort: Оператор для сортировки документов. { year: -1 } указывает, что книги должны быть отсортированы по полю year в порядке убывания (самые новые книги первыми).
$limit: Оператор, который ограничивает количество возвращаемых документов. В данном случае мы устанавливаем его на 5, чтобы получить только 5 самых новых книг.

Если нужны только определенные поля (например, title и year), можно добавить стадию $project перед $limit:

```
libraryDB> db.books.aggregate([
...   { $sort: { year: -1 } },
...   { $project: { title: 1, year: 1, _id: 0 } },
...   { $limit: 5 }
... ])
[
  { title: 'To Kill a Mockingbird', year: 1960 },
  { title: 'The Catcher in the Rye', year: 1951 },
  { title: '1984', year: 1949 },
  { title: 'The Hobbit', year: 1937 },
  { title: 'Brave New World', year: 1932 }
]
libraryDB>
```

{ $project: { title: 1, year: 1, _id: 0 } }, // Отображаем только title и year.

Фильтрация и группировка

Агрегационный запрос, который фильтрует книги, доступные для аренды, группирует их по авторам и считает количество книг для каждого автора:

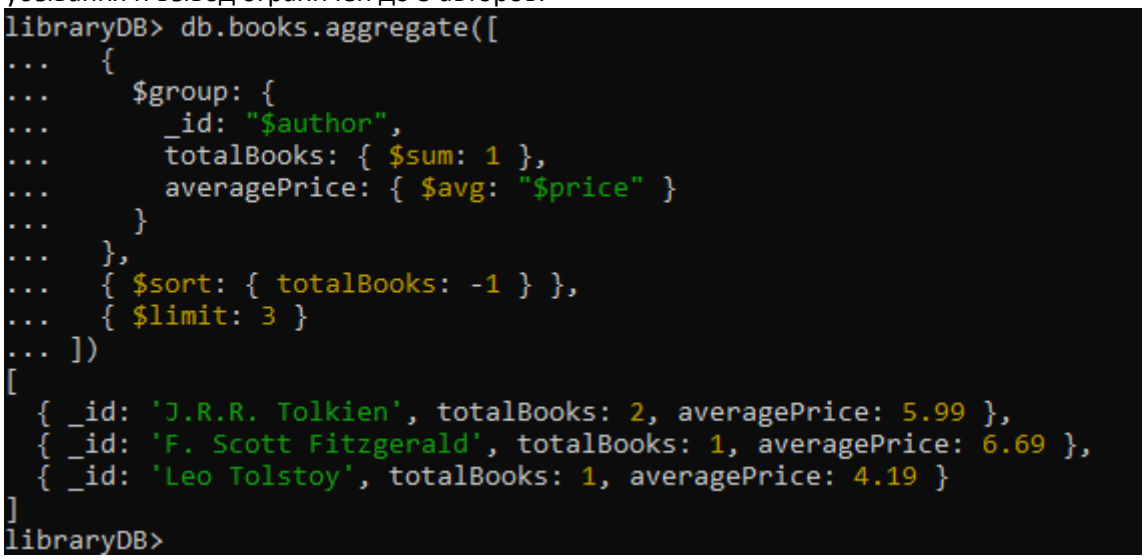
```
libraryDB> db.books.aggregate([
...   { $match: { available: true } },
...   {
...     $group: {
...       _id: "$author",
...       count: { $sum: 1 }
...     }
...   }
... ])
[
  { _id: 'Harper Lee', count: 1 },
  { _id: 'F. Scott Fitzgerald', count: 1 },
  { _id: 'Leo Tolstoy', count: 1 },
  { _id: 'J.R.R. Tolkien', count: 2 },
  { _id: 'Aldous Huxley', count: 1 }
]
libraryDB>
```

$match: Оператор, который фильтрует документы и оставляет только те, где available равно true (т.е. книги, доступные для аренды).
$group: Оператор, который группирует документы по значению поля author.
_id: "$author": Указывает, что группировка происходит по полю author.
count: { $sum: 1 }: Создает поле count, которое подсчитывает количество документов в каждой группе (количество книг для каждого автора).

Если нужно отсортировать результаты по количеству книг, можно добавить стадию $sort после $group:

{ $sort: { count: -1 } } // Сортируем по количеству книг в порядке убывания

Агрегационный запрос, который группирует книги по авторам, считает общее количество книг и среднюю цену книг для каждого автора. Результаты сортируются по количеству книг в порядке убывания и вывод ограничен до 3 авторов.



$group: Оператор, который группирует документы по значению поля author.
_id: "$author": Указывает, что группировка происходит по полю author.
totalBooks: { $sum: 1 }: Создает поле totalBooks, которое подсчитывает общее количество книг для каждого автора.
averagePrice: { $avg: "$price" }: Создает поле averagePrice, которое вычисляет среднюю цену книг для каждого автора.
$sort: Оператор, который сортирует результаты по полю totalBooks в порядке убывания (-1).
$limit: Оператор, который ограничивает количество возвращаемых документов до 3.