



САМАРСКИЙ УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

# Базы данных

## Лекция 12

MongoDB.

Расширенные возможности

Агафонов Антон Александрович  
д.т.н., доцент кафедры ГИИБ

Самара



- Расширенные возможности MongoDB
  - Агрегация данных
  - Представления
  - Индексирование
  - Пространственные запросы
  - Транзакции





Операции агрегации данных обрабатывают набор документов и возвращают результаты обработки. Агрегация данных может быть использована для:

- группировки значений из нескольких документов;
- выполнения операций над сгруппированными данными для получения единственного результата;
- анализа изменения данных в течение определенного промежутка времени.

Для агрегации данных используются:

- конвейеры агрегации, которые являются предпочтительным методом агрегирования данных;
- методы агрегации, которые просты, но лишены возможностей конвейера агрегации.





Метод	Описание
<code>db.collection.estimatedDocumentCount()</code>	Возвращает приблизительное количество документов в коллекции или представлении
<code>db.collection.count()</code>	Возвращает количество документов в коллекции или представлении
<code>db.collection.distinct()</code>	Возвращает массив документов, которые имеют различные значения для указанного поля





Выборка массива статусов документов без повторений

```
db.inventory.distinct( "status" )  
  
[ 'A', 'D' ]
```

```
SELECT DISTINCT status FROM inventory;
```





Конвейер агрегации состоит из одного или нескольких этапов обработки документов:

- каждый этап выполняет операцию над входными документами. Например, на этапе может происходить фильтрация документов, группировка документов и вычисление некоторых значений;
- документы, полученные как результат работы этапа, передаются на следующий этап;
- конвейер агрегации может возвращать результаты для групп документов, например, вернуть среднее, максимальное и минимальное значения;
- начиная с MongoDB 4.2, конвейер агрегации может использоваться для обновления документов.





```
db.collection.aggregate(  
    pipeline, options)
```

```
db.collection.aggregate(  
    [ { <stage> }, ... ]  
)
```

Агрегация данных из коллекции `collection`

- `pipeline` – последовательность операций или этапов агрегации данных;
- `options` – дополнительные параметры.

Оператор	Описание
<code>\$count</code>	Возвращает количество документов на данном этапе конвейера агрегации.
<code>\$group</code>	Группирует входные документы по указанному выражению-идентификатору и применяет выражения-аккумуляторы, если они указаны, к каждой группе. Выводит по одному документу для каждой отдельной группы. Выходные документы содержат только поле-идентификатор и, если указано, поля-аккумуляторы.





Оператор	Описание
<code>\$limit</code>	Передаёт первые $n$ документов без изменений в конвейер, где $n$ – указанное ограничение. Для каждого входного документа выводит либо один документ (для первых $n$ документов), либо ноль документов (после первых $n$ документов).
<code>\$lookup</code>	Выполняет левое внешнее соединение с другой коллекцией в той же базе данных, чтобы фильтровать документы из «объединенной» коллекции для обработки.
<code>\$match</code>	Фильтрует поток документов, позволяя только документам, удовлетворяющим условию, передаваться без изменений на следующий этап конвейера. Использует стандартные запросы MongoDB.
<code>\$merge</code>	Записывает результирующие документы конвейера агрегации в коллекцию. Должен быть последним этапом конвейера.
<code>\$project</code>	Изменяет структуру каждого документа в потоке, например, добавляя новые поля или удаляя существующие поля. Для каждого входного документа выводит один документ.







Оператор	Описание
<code>\$set</code>	Добавляет новые поля в документы. Как и <code>\$project</code> , <code>\$set</code> изменяет структуру каждого документа; в частности, путем добавления новых полей в выходные документы, которые содержат как существующие поля из входных документов, так и вновь добавленные поля.
<code>\$skip</code>	Пропускает первые $n$ документов, где $n$ – указанный номер пропуска, и передает остальные документы без изменений в конвейер.
<code>\$sort</code>	Переупорядочивает поток документов по указанному ключу сортировки. Меняется только порядок, документы остаются без изменений.
<code>\$unset</code>	Удаляет/исключает поля из документов.
<code>\$unwind</code>	Преобразует поле массива из входных документов для вывода документа для каждого элемента. Каждый выходной документ заменяет массив значением элемента. Для каждого входного документа выводит $n$ документов, где $n$ – количество элементов массива, которое может быть равно нулю для пустого массива.





Оператор	Описание
<code>\$accumulator</code>	Возвращает результат пользовательской функции-аккумулятора.
<code>\$avg</code>	Возвращает среднее числовых значений. Игнорирует нечисловые значения.
<code>\$first</code>	Возвращает значение из первого документа для каждой группы. Порядок определяется только в том случае, если документы отсортированы.
<code>\$last</code>	Возвращает значение из последнего документа для каждой группы. Порядок определяется только в том случае, если документы отсортированы.
<code>\$max</code>	Возвращает наибольшее значение выражения для каждой группы.
<code>\$min</code>	Возвращает наименьшее значение выражения для каждой группы.
<code>\$sum</code>	Возвращает сумму числовых значений. Игнорирует нечисловые значения.



SQL-операторы и функции	Операторы агрегации в MongoDB
WHERE	\$match
GROUP BY	\$group
HAVING	\$match
SELECT	\$project
ORDER BY	\$sort
LIMIT	\$limit
SUM()	\$sum
COUNT()	\$count \$sortByCount
JOIN	\$lookup



Коллекция zipcodes (<http://media.mongodb.org/zips.json>):

```
{  
  _id: "01012",  
  city: "CHESTERFIELD",  
  pop: 177,  
  state: "MA",  
  loc: [ -72.833309, 42.38167 ]  
}
```



Подсчет количества записей с популяцией больше 1000 для каждого штата

```
db.zips.aggregate( [ {  
  $match: { pop: { $gt: 1000 } }  
}, {  
  $group: {  
    _id: '$state',  
    cities_count: { $count: {} }  
  }  
} ] )
```

```
{ _id: "WV", cities_count: 315}  
{ _id: "NM", cities_count: 131}  
...
```

```
SELECT state AS _id, COUNT(*) AS cities_count  
FROM zips  
WHERE pop > 1000  
GROUP BY state;
```





Выборка штатов с популяцией больше 10 миллионов

```
db.zips.aggregate( [  
  { $group: { _id: "$state", totalPop: { $sum: "$pop" } } },  
  { $match: { totalPop: { $gte: 10 * 1000 * 1000 } } }  
] )
```

```
{ _id: "OH", totalPop: 10846517 }  
{ _id: "FL", totalPop: 12686644 }  
{ _id: "IL", totalPop: 11427576 }  
...
```

```
SELECT state AS _id, SUM(pop) AS totalPop  
FROM zips  
GROUP BY state  
HAVING totalPop >= 10 * 1000 * 1000;
```



Подсчет среднего населения города по штатам

```
db.zips.aggregate( [  
  { $group: { _id: { state: "$state", city: "$city" }, pop: { $sum: "$pop" } } },  
  { $group: { _id: "$_id.state", avgCityPop: { $avg: "$pop" } } },  
  { $sort: { avgCityPop: -1 } }  
] )
```

После первого этапа:

```
{ _id: { state: "CO", city: "EDGEWATER" }, pop: 13154 }
```

После сортировки:

```
{ _id: "DC", avgCityPop: 303450 }  
{ _id: "CA", avgCityPop: 27756.42723880597 }  
{ _id: "FL", avgCityPop: 27400.958963282937 }  
...
```





Поиск городов с наибольшей и наименьшей популяцией для каждого штата

```
db.zips.aggregate( [
  { $group: { _id: { state: "$state", city: "$city" }, pop: { $sum: "$pop" } } },
  { $sort: { pop: 1 } },
  { $group: {
    _id: "$_id.state",
    biggestCity: { $last: "$_id.city" }, biggestPop: { $last: "$pop" },
    smallestCity: { $first: "$_id.city" }, smallestPop: { $first: "$pop" }
  } },
  { $project: {
    _id: 0, state: "$_id",
    biggestCity: { name: "$biggestCity", pop: "$biggestPop" },
    smallestCity: { name: "$smallestCity", pop: "$smallestPop" }
  }
} ] )

{ state: "AL",
  biggestCity: { name: "BIRMINGHAM", pop: 242606 },
  smallestCity: { name: "ALLEN", pop: 0 }}
```

...







Запись среднего населения города по штатам в новую коллекцию

```
db.zip.aggregate( [  
  { $group: { _id: { state: "$state", city: "$city" }, pop: { $sum: "$pop" } } },  
  { $group: { _id: "$_id.state", avgCityPop: { $avg: "$pop" } } },  
  { $merge: "state_avg_data" }  
] )
```

```
db.state_avg_data.find().count()
```

51



Представление (view) – объект базы данных, предназначенный только для чтения, содержимое которого определяется запросом агрегации данных из других коллекций или представлений.

Содержимое представления не сохраняется на диске и вычисляется в момент запроса к представлению.

Преимущества использования представлений:

- ▲ исключение некоторой информации из возвращаемых приложению документов, например, персональных данных;
- ▲ добавление вычисляемых полей в представление;
- ▲ создание представления с использованием сложного контейнера агрегации и скрытие деталей реализации от приложения.





## Создание представления

```
db.createCollection(  
  "<viewName>",  
  {  
    "viewOn" : "<source>",  
    "pipeline" : [<pipeline>],  
    "collation" : { <collation> }  
  }  
)
```

Создает новую коллекцию или представление.

```
db.createView(  
  "<viewName>",  
  "<source>",  
  [<pipeline>],  
  {  
    "collation" : { <collation> }  
  }  
)
```

Создает новое представление как результат применения указанного конвейера агрегации к исходной коллекции или представлению.



Представление, возвращающее среднее население по штатам

```
db.createView(  
  "avgPopulation",  
  "zips",  
  [  
    { $group: { _id: { state: "$state", city: "$city" }, pop: { $sum: "$pop" } } },  
    { $group: { _id: "$_id.state", avgCityPop: { $avg: "$pop" } } },  
    { $sort: { avgCityPop: -1 } }  
  ]  
)
```

```
db.avgPopulation.find()
```

```
{ _id: "DC", avgCityPop: 303450 }  
{ _id: "CA", avgCityPop: 27756.42723880597 }  
{ _id: "FL", avgCityPop: 27400.958963282937 }  
...
```



## Пример создания представления с соединением коллекций

Представление, возвращающее результат соединения

```
db.createView("extZips", "zips",
[
  { $lookup: {
    from: "avgPopulation",
    localField: "state",
    foreignField: "_id",
    as: "avgPopulationDocs"
  }
},
{ $project: { city: 1, state: 1, pop: 1, avgStatePop: "$avgPopulationDocs.avgCityPop" } },
{ $unwind: "$avgStatePop" }
]
)

db.extZips.find()

{ _id: "01035", city: "HADLEY", pop: 4231, state: "MA", avgStatePop: 14855.37037037037 }
...
```





---

`db.collection.drop(<options>)`

Удаление коллекции или представления.

- 
- Удаление и создание представления заново
  - Использование команды `collMod`

Изменение представления

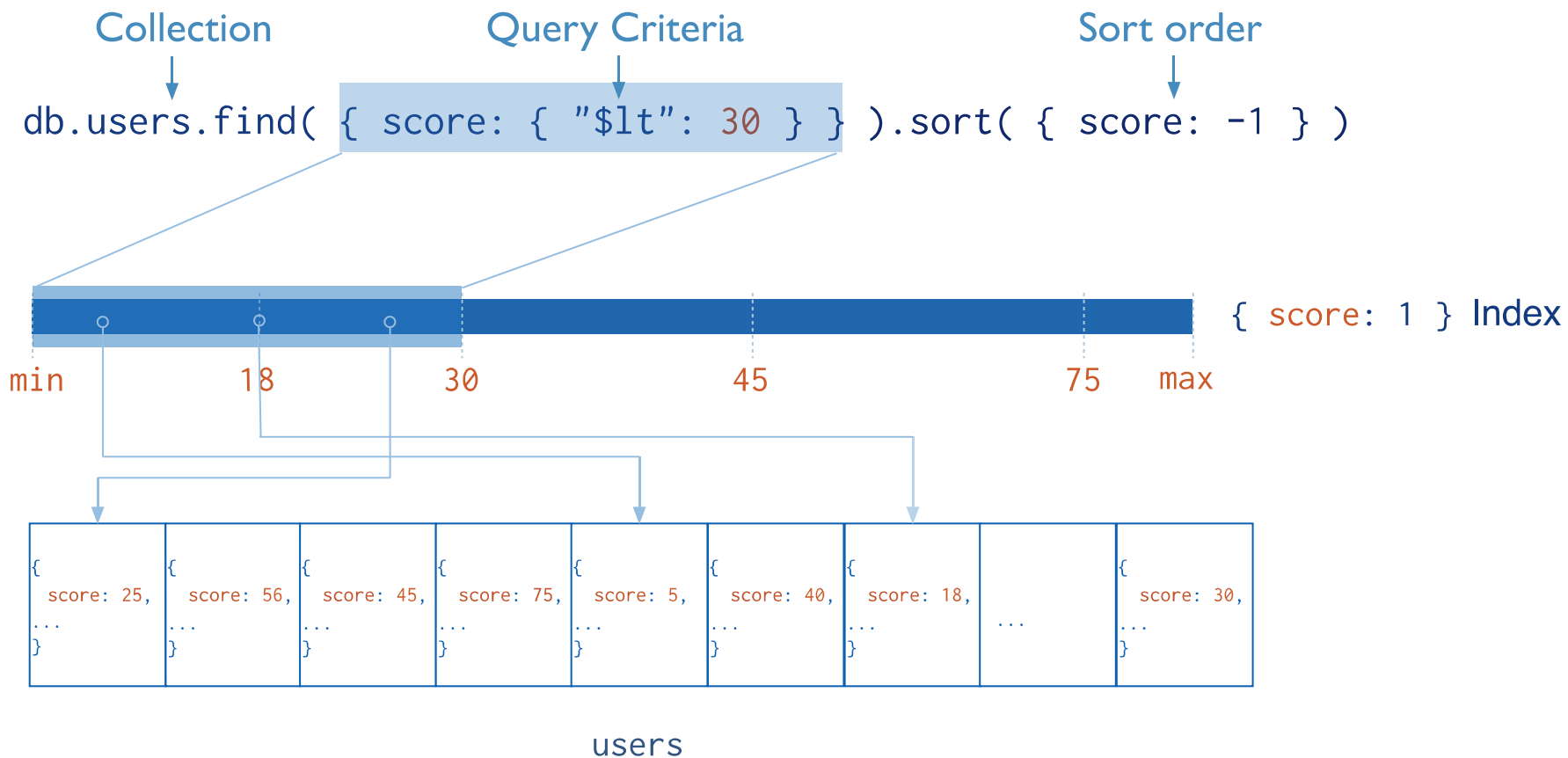
---





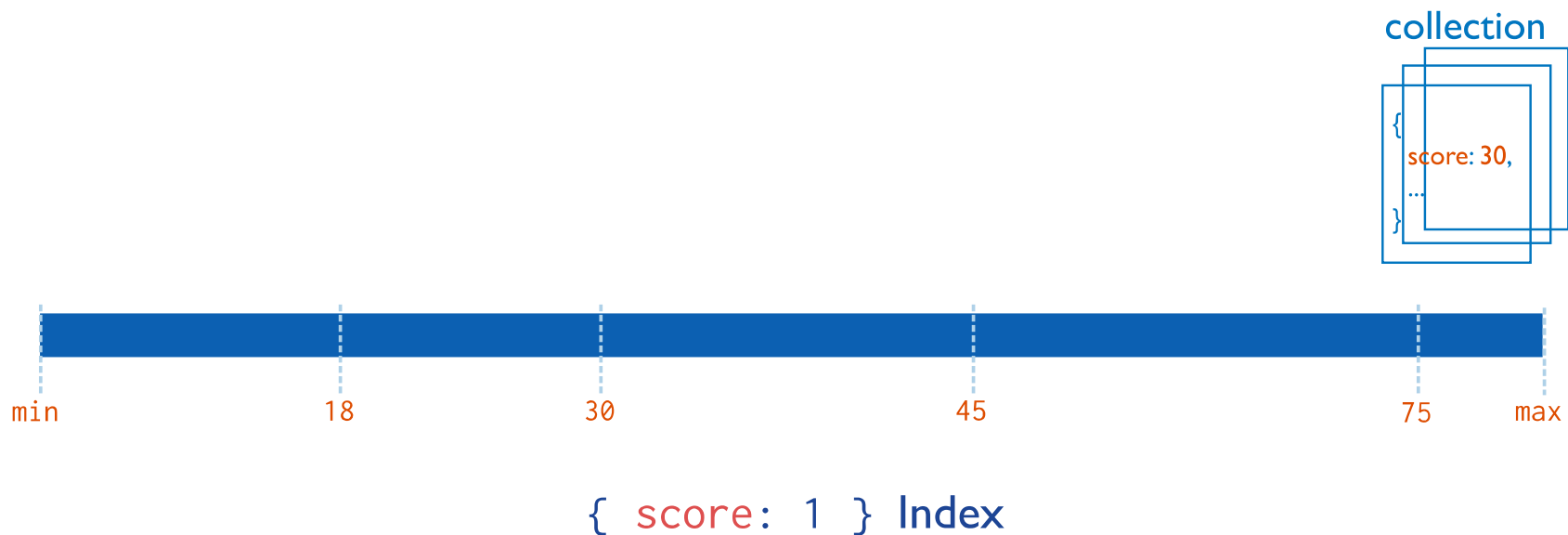
# Индексы

Индекс – объект базы данных, создаваемый с целью повышения производительности поиска данных.





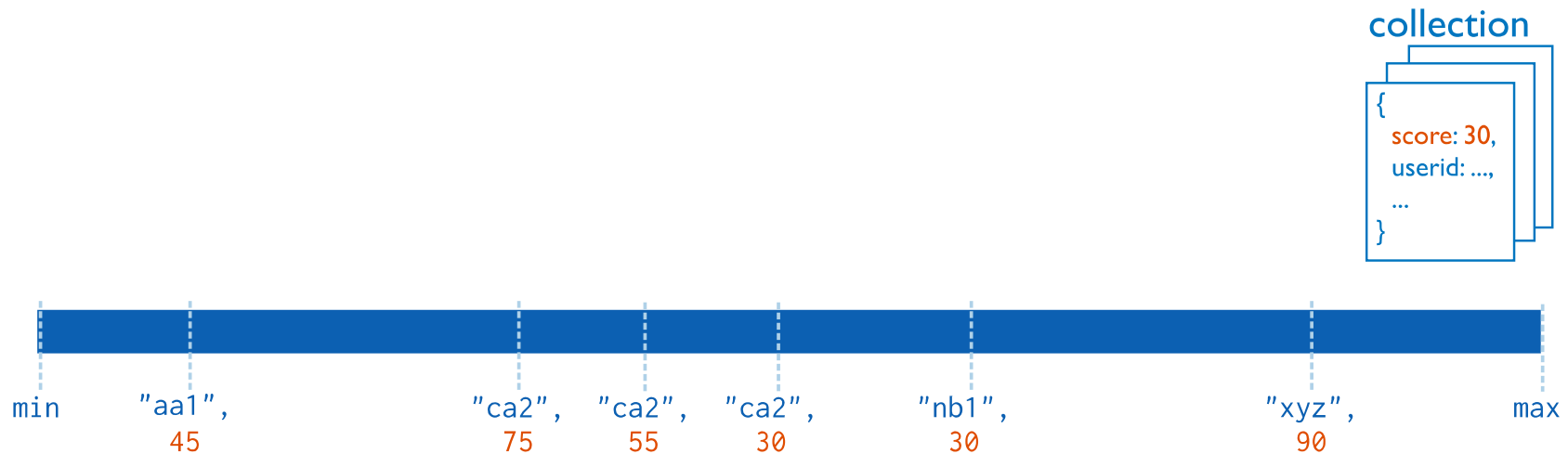
- **Индекс по одному полю.** Поддерживается создание определяемых пользователем индексов по возрастанию/убыванию для одного поля документа. По умолчанию, MongoDB создает уникальный индекс по полю `_id`.







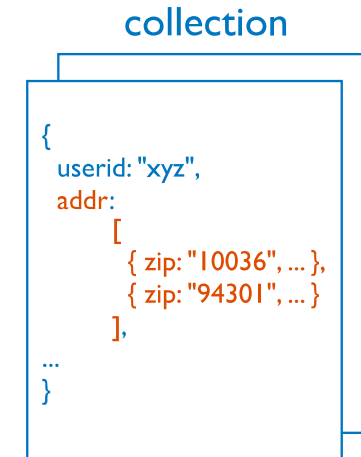
- **Составной индекс.** Поддерживается создание индексов по нескольким полям. Порядок полей, перечисленных в составном индексе, имеет значение.



{ userid: 1, score: -1 } Index



- **Многоключевой (мультиключевой) индекс.** Используется для индексации содержимого, хранящегося в массивах. Если индексируется поле, содержащее значение массива, MongoDB создает отдельные записи индекса для каждого элемента массива.



{ "addr.zip": 1 } Index





- **Пространственный индекс.** Используется для поддержки эффективных геопространственных запросов. MongoDB предоставляет два специальных индекса: двумерные индексы, которые используют планарную геометрию при возврате результатов, и двумерные индексы, которые используют сферическую геометрию для возврата результатов.
- **Текстовый индекс.** Используется для устаревшего полнотекстового поиска.
- **Хэш-индекс.** Используется для поддержки сегментирования на основе хэша.





```
db.collection.createIndex(keys, options, commitQuorum)
```

keys	Документ, содержащий пары поля и значения, где поле является ключом индекса, а значение описывает тип индекса для этого поля. Для индекса поля с сортировкой поля по возрастанию указывается значение 1; для индекса по убыванию указывается -1.
options	Опциональный параметр. Документ, содержащий набор параметров, управляющих созданием индекса. Поддерживаются параметры name – название индекса, unique – для создания уникального индекса и т.д.
commitQuorum	Опциональный параметр. Число реплик, которые должны сообщить об успешном создании индекса.





### wikipedia

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
30.96 GB	15 M	4.23 kB	1	532.68 MB

Выборка количества документов в БД

```
db.wikipedia.find( ).count()
```

```
15138138
```





Выборка документов в БД

```
db.wikipedia.find( { _id: "Linkin Park" } )  
  
{  
  _id: 'Linkin Park',  
  type: 'page',  
  sections: [  
    {  
      title: '',  
      depth: 0,  
      sentences: [ { text: 'Linkin Park is an American rock band from Agoura Hills, California.' }, ... ]  
    }, ...  
  ],  
  categories: [  
    'Linkin Park',  
    'Alternative rock groups from California', ...  
  ],  
  images: [  
    { url: 'https://upload.wikimedia.org/wikipedia/commons/4/42/LinkinParkBerlin2010.jpg', file:  
'File:LinkinParkBerlin2010.jpg', ... }, ...  
  ],  
  coordinates: [ ], ...  
  title: 'Linkin Park'  
}
```



Выборка документов в БД по идентификатору

```
db.wikipedia.find( { _id: "Linkin Park" } ).explain("executionStats" )
```

```
{ explainVersion: '1',
  queryPlanner:
    { namespace: 'wikipedia.wikipedia',
      indexFilterSet: false,
      parsedQuery: { _id: { '$eq': 'Linkin Park' } },
      winningPlan: { stage: 'IDHACK' },
      rejectedPlans: [] },
  executionStats:
    { executionSuccess: true,
      nReturned: 1,
      executionTimeMillis: 3,
      totalKeysExamined: 1,
      totalDocsExamined: 1,
      executionStages: { stage: 'IDHACK', nReturned: 1, ... } },
  command: { find: 'wikipedia', filter: { _id: 'Linkin Park' }, '$db': 'wikipedia' },
}
```





Выборка документов в БД по названию

```
db.wikipedia.find( { title: "Linkin Park" } ).explain("executionStats" )
```

```
{ explainVersion: '1',
  queryPlanner:
    { namespace: 'wikipedia.wikipedia',
      indexFilterSet: false,
      parsedQuery: { title: { '$eq': 'Linkin Park' } },
      winningPlan:
        { stage: 'COLLSCAN',
          filter: { title: { '$eq': 'Linkin Park' } },
          direction: 'forward' },
      rejectedPlans: [] },
  executionStats:
    { executionSuccess: true,
      nReturned: 1,
      executionTimeMillis: 58712,
      totalKeysExamined: 0,
      totalDocsExamined: 15138138,
      executionStages: { stage: 'COLLSCAN', filter: { title: { '$eq': 'Linkin Park' } }, nReturned: 1, ... }
    },
  command: { find: 'wikipedia', filter: { title: 'Linkin Park' }, '$db': 'wikipedia' },
}
```





Выборка документов в БД по названию

```
db.wikipedia.createIndex( { title: 1 } )
```

```
db.wikipedia.find( { title: "Linkin Park" } ).explain("executionStats" )
```

```
{ explainVersion: '1',
  queryPlanner:
    { namespace: 'wikipedia.wikipedia',
      indexFilterSet: false,
      parsedQuery: { title: { '$eq': 'Linkin Park' } },
      winningPlan:
        { stage: 'FETCH',
          inputStage:
            { stage: 'IXSCAN', keyPattern: { title: 1 }, indexName: 'title_1', isMultiKey: false, ... },
          rejectedPlans: [] },
      executionStats:
        { executionSuccess: true,
          nReturned: 1,
          executionTimeMillis: 13,
          totalKeysExamined: 1,
          totalDocsExamined: 1,
          executionStages: { stage: 'FETCH', nReturned: 1, ... } },
      command: { find: 'wikipedia', filter: { title: 'Linkin Park' }, '$db': 'wikipedia' },
  }
```





## Выполнение запроса с фильтрацией по значениям в массиве

Выборка документов по категории

```
db.wikipedia.find( { categories: { $all: ["Grammy Award winners"] } }, { _id :1 } )
```

...

```
{ _id: 'Elvis Aaron Presley' }
```

```
{ _id: 'Enya' }
```

```
{ _id: 'Eric Patrick Clapton' }
```

```
{ _id: 'Ennio Morricone' }
```

```
{ _id: 'Elvis Costello' }
```

...





### Выборка документов по категории

```
db.wikipedia.find( { categories: { $all: ["Grammy Award winners"] } }, { _id :1 } )
```

```
{ ...
  executionStats:
    { executionSuccess: true,
      nReturned: 2653,
      executionTimeMillis: 65667,
      totalKeysExamined: 0,
      totalDocsExamined: 15138138,
      executionStages: {
        stage: 'PROJECTION_SIMPLE',
        inputStage:
          { stage: 'COLLSCAN',
            filter: { categories: { '$eq': 'Grammy Award winners' } }, ...
          }, ...
      }
    }
  }
}
```



```
db.wikipedia.createIndex( { categories: 1 } )
```

Выборка документов по категории

```
db.wikipedia.find( { categories: { $all: ["Grammy Award winners"] } }, { _id :1 } )
```

```
{ ...
  executionStats:
    { executionSuccess: true,
      nReturned: 2653,
      executionTimeMillis: 638,
      totalKeysExamined: 2653,
      totalDocsExamined: 2653,
      executionStages: {
        stage: 'PROJECTION_SIMPLE',
        inputStage:
          { stage: 'IXSCAN',
            filter: { categories: { '$eq': 'Grammy Award winners' } }, ...
          }, ...
      }
    }
}
```





Оператор	Описание
<code>\$geoIntersects</code>	Возвращает геометрию, которая пересекается с указанной геометрией, заданной в формате GeoJSON
<code>\$geoWithin</code>	Возвращает геометрию, которая находится в границах указанной геометрии
<code>\$near</code>	Возвращает геопространственные объекты вблизи указанной точки
<code>\$nearSphere</code>	Возвращает геопространственные объекты вблизи указанной точки на сфере



```
db.wikipedia_geo.createIndex( { spatial: "2dsphere" } )
```

Поиск документов вблизи указанного положения

```
db.wikipedia_geo.find( {  
  spatial: {  
    $near: {  
      $geometry: { type: "Point", coordinates: [ 50.14083, 53.20278 ] },  
      $minDistance: 1000,  
      $maxDistance: 5000  
    }  
  }, { _id :1 }  
} )  
  
{ _id: 'Soyuz carrier rocket monument' }  
{ _id: 'Samara National Research University' }  
{ _id: 'Kuybyshev Square' }  
{ _id: 'Stalin\'s bunker' }  
{ _id: 'Monument to Vasily Chapaev' }  
...
```



MongoDB – распределенная СУБД с ACID-совместимыми транзакциями между наборами реплик и/или сегментов (шардов).

Базовый API	API обратного вызова
Требует явного вызова для запуска транзакции и ее фиксации	Запускает транзакцию, выполняет указанные операции и фиксацию (или прерывает работу при возникновении ошибки)
Не включает логику обработки ошибок <code>TransientTransactionError</code> и <code>UnknownTransactionCommitResult</code> , вместо этого обеспечивая гибкость, позволяющую включить настраиваемую обработку этих ошибок	Автоматически включает логику обработки ошибок <code>TransientTransactionError</code> и <code>UnknownTransactionCommitResult</code>
Требует явного логического сеанса для передачи его в API для конкретной транзакции	Требует явного логического сеанса для передачи его в API для конкретной транзакции





<code>Session.startTransaction(&lt;options&gt;)</code>	Запускает мультидокументную транзакцию, связанную с сеансом. В любой момент времени для сеанса может быть открыта только одна транзакция.
<code>Session.commitTransaction()</code>	Сохраняет изменения, внесенные операциями в многодокументной транзакции, и завершает транзакцию.
<code>Session.abortTransaction()</code>	Завершает транзакцию с несколькими документами и откатывает любые изменения данных, сделанные операциями внутри транзакции. То есть транзакция завершается без сохранения каких-либо изменений, сделанных операциями в транзакции.







```
session = db.getMongo().startSession( );

zips_collection = session.getDatabase("db_lectures").zips;
inv_collection = session.getDatabase("db_lectures").inventory;

session.startTransaction( );

zips_collection.updateOne( { _id: '03101' }, { $set: { pop: 10000 } } );
inv_collection.insertOne( { item: 'card', status: 'C', qty: 10 } );

session.commitTransaction();
session.endSession();
```





**САМАРСКИЙ** УНИВЕРСИТЕТ  
SAMARA UNIVERSITY

**БЛАГОДАРЮ  
ЗА ВНИМАНИЕ**

Агафонов А.А.  
д.т.н., доцент кафедры ГИИБ