




ОНЛАЙН-ОБРАЗОВАНИЕ

Онлайн-образование



Меня хорошо видно && слышно?

Ставьте  , если все хорошо
Напишите в чат, если есть проблемы
заодно проверяем, включена ли запись занятия

Включил Юджин запись ли ты



Оптимизация производительности MongoDB



Аристов Евгений

telegram @AEugene

<https://aristov.tech>

Правила вебинара



Активно участвуем



Задаем вопрос в чат



Вопросы вижу в чате, могу ответить не сразу

Маршрут вебинара

1. Профилирование
2. Виды и построение индексов в `mongodb`
3. Оптимизации CRUD.
4. Дисковые движки.
5. GridFS
6. Производительность кластеров.

Цели вебинара | После занятия вы сможете

1

Настраивать производительность кластера

2

Построить производительные индексы

3

Оптимизировать БД для CRUD

The background of the slide features an aerial view of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue gradient and a network of white lines connecting various points, creating a digital or technological aesthetic.

Профилирование

Профилирование

Профилирование работы в MongoDB

- специальная коллекция **system.profile**.
- является ограниченной коллекцией (*capped collection*) - её размер ограничен 1Мб.
- по умолчанию профилирование запросов отключено, и никакой информации о работе системы не сохраняется.

db.getProfilingStatus() - returns if profiling is on and slow threshold

db.setProfilingLevel(level, <slowms>) 0=off 1=slow 2=all

db.setProfilingLevel(level, slowms, <samplerate>) 0..1 - % медленных запросов

<https://docs.mongodb.com/manual/reference/database-profiler/>

Профилирование

Уровни профилирования:

0, профилирование отключено полностью. Это режим по умолчанию, если вы еще не успели настроить профилирование в своей БД, то вы должны увидеть следующее:

```
> db.getProfilingStatus()  
{ "was" : 0, "slowms" : 100 }
```


Профилирование

Уровни профилирования:

1, профилирование медленных запросов. Этот режим использую на продакшене, потому что он позволяет логировать только запросы, выполнявшиеся дольше определенного порога(threshold). Когда вы устанавливаете этот режим, второй параметр в `db.setProfilingLevel` становится обязательным и указывает на размер порога срабатывания в миллисекундах. Рекомендованный порог в 100 мс, но это дело вкуса:

```
> db.setProfilingLevel(1, 100)
```

```
{ "was" : 0, "slowms" : 100, "ok" : 1 }
```

```
> db.getProfilingStatus()
```

```
{ "was" : 1, "slowms" : 100 }
```


Профилирование

Уровни профилирования:

2, профилирование всех запроса. Хорошо подходит для разработки, но на продакшене использовать нецелесообразно: старые данные профайлера быстро затираются, а накладные расходы на поддержание столь подробного лога, увеличиваются.

```
> db.setProfilingLevel(2)
```

```
{ "was" : 1, "slowms" : 100, "ok" : 1 }
```

```
> db.getProfilingStatus()
```

```
{ "was" : 2, "slowms" : 100 }
```


Профилирование

Структура документов в `system.profile`

Все записи профайлера представляют собой обычные документы со следующим набором основных полей:

- **op**, тип операции(*insert, query, update, remove, getmore, command*)
- **ns**, коллекция(а точнее [namespace](#)), над которой производится операция
- **millis**, время выполнения операции в миллисекундах
- **ts**, время(*timestamp*) операции. Большого значения это не имеет, но это дата **окончания** выполнения операции.
- **client**, IP-адрес или имя хоста, с которого была отправлена команда
- **user**, авторизованный пользователь, который выполнил запрос. Если вы не используете авторизацию, то в профайлер будет записана пустая строка.

В дополнение к основным полям, есть ещё поля, специфические для каждого типа запроса. Для поиска(*find*) это будет сам запрос(*query*), информация о числе просканированных (*nscanned*) и возвращенных (*nreturned*) документов, для изменения(*update*) это будет число обновленных (*nupdated*) и перемещённых на диске (*nmoved*) элементом и т.д.

Профилирование

Запросы к профайлеру

// Вывести все данные в порядке убывания даты создания

```
> db.system.profile.find().sort({$natural:-1});
```

// Найти все операции длиннее 5 мс.

```
> db.system.profile.find( { millis : { $gt : 5 } } );
```

// Вывести все данные в порядке убывания времени выполнения

// (самые тяжелые запросы в начале)

```
> db.system.profile.find().sort({millis:-1});
```

Профилирование

Продвинутые запросы

Используем [Aggregation Framework](#) для написания запросов

- позволяет видеть общую картину
- определенный день
- средние показатели
- сгруппировать по коллекциям
- ...

Профилирование

- количество “проблемных” запросов(*count*) в определенный день и средним временем выполнения (*avg_ms*)

```
db.system.profile.aggregate([{$match: {ts:{$gte:ISODate("2020-09-30T00:00:00.000Z"),  
$lt:ISODate("2020-10-20T00:00:00.000Z")}}}, {$group:{_id:null, count:{$sum:1}, avg_ms:{$avg:'$millis'}}}])
```

- количество “проблемных” запросов(*count*) в определенный день и средним временем выполнения (*avg_ms*) + метрики

```
db.system.profile.aggregate([{$match: {ts:{$gte:ISODate("2020-05-11T00:00:00.000Z"),  
$lt:ISODate("2020-05-12T00:00:00.000Z")}}}, {$group:{_id:'$op', count:{$sum:1}, avg_ms:{$avg:'$millis'},  
min_ms:{$min:'$millis'}, max_ms:{$max:'$millis'}}}])
```

- сгруппируем данные по коллекции

```
db.system.profile.aggregate([{$match: {ts:{$gte:ISODate("2020-05-11T00:00:00.000Z"),  
$lt:ISODate("2020-05-12T00:00:00.000Z")}}}, {$group:{_id:'$ns', count:{$sum:1}, avg_ms:{$avg:'$millis'},  
min_ms:{$min:'$millis'}, max_ms:{$max:'$millis'}}}])
```

- гистограмма с разбиением по 5мс

```
db.system.profile.aggregate([{$project: {'ms':{'$subtract':['$millis',{ '$mod':['$millis', 5]}}}}, {$group:{_id:'$ms',  
sum:{$sum:1}}}, {$sort:{_id:1}}])
```

Профилирование

Автоматизируем процесс:

создадим файл profile.js

```
function profile_hist(){  
    res = db.system.profile.aggregate([{$project: {'ms':{'$subtract':['$millis',{  
5}]}}}}, {$group:{_id:'$ms', sum:{$sum:1}}}, {$sort:{_id:1}}]);  
    res.forEach(function(i) { print(i['_id'], '\t',i['sum']); });  
}
```

```
$ mongo [db_name] --shell profile.js
```

```
> profile_hist()
```

<https://gist.github.com/amezhenin/8132687> //скрипты

Профилирование

Увеличиваем размер `system.profile`

Данная коллекция имеет ограничение в 1Мб. Это значение можно изменить, если вам нужен больший объем лога. Так как **`system.profile`** является ограниченной коллекцией, мы не можем изменить размер зарезервированного под неё места, но мы можем пересоздать её с другими опциями:

```
db.setProfilingLevel(0) // останавливаем профилирование
db.system.profile.drop() // удаляем коллекцию
// создаем ограниченную коллекцию с нужными параметрами
db.createCollection( "system.profile", { capped: true, size:4000000 } )
db.setProfilingLevel(1) // включаем профилирование назад
```

Профилирование

Логирование

logpath=/var/log/mongodb/mongodb.log

..

Wed Jun 26 22:02:06.197 [conn1599] insert test_db.events ninserted:1 keyUpdates:0 locks(micros) w:31 **152ms**

..

Wed Jun 26 22:05:01.022 [conn1588] command test_db.\$cmd command: { aggregate: "events", pipeline: [...] }
ntoreturn:1 keyUpdates:0 numYields: 109 locks(micros) r:1788726 reslen:762 **921ms**

..

Thu Jun 26 22:46:51 [initandlisten] connection accepted from xxx.xxx.xxx.xxx:56918 #80045 (22 connections now open)

..

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network diagram. This diagram consists of numerous small dots connected by thin white lines, creating a complex web-like structure that spans the width of the slide. Centered within this band is the main title text.

Планы запросов

Планы запросов

<https://docs.mongodb.com/manual/core/query-plans/>

The image features a background of a dense city skyline, likely New York City, viewed from an elevated perspective. The entire image is overlaid with a semi-transparent blue and green gradient. A network of thin, light blue lines connects various points across the image, creating a digital or technological feel. The text "Вопросы?" is centered in the middle of the image in a large, white, sans-serif font.

Вопросы?

The background of the entire image is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue overlay covers the entire image. In the center, there is a network of thin, light blue lines connecting various points, creating a web-like pattern. The word "Индексы" is written in white, bold, sans-serif font, centered within this network pattern.

Индексы

Индексы

- Аналогичны индексам в других БД
- default по полю _id
- B-tree структура

```
db.collection.createIndex(  
  <key and index type specification>,  
  <options>  
)
```

```
db.collection.getIndexes();
```

Индексы

Свойства:

уникальность

```
db.users.createIndex(  
  { email: 1 },  
  { unique: true }  
)
```

разряженные индексы (по умолчанию плотные)

```
db.users.createIndex(  
  { email: 1 },  
  { sparse: true }  
)
```

TTL Indexes

// Удалить документ через 2 минуты

```
db.users.createIndex(  
  { email: 1 },  
  { expireAfterSeconds: 120 }  
)
```


Индексы

Виды индексов:

- **одно поле**

```
db.users.createIndex(  
  { email: 1 }  
)
```

- **составной индекс**

```
db.products.createIndex(  
  { item: 1, quantity: -1 },  
  { name: "query for inventory" }  
)
```

- **multikey**

```
{  
  name: 'Merrick'  
  addr: [ {zip: '147000',... },  
           {zip: '147000',... } ]  
}  
db.users.createIndex(  
  { "addr.zip": 1 }  
)
```

Индексы

- **гео индексы**

- 2d <https://docs.mongodb.com/manual/core/2d/> uses planar geometry
- 2dsphere <https://docs.mongodb.com/manual/core/2dsphere/> use spherical geometry

[GeoJSON Point:](#)

```
location: {  
  type: "Point",  
  coordinates: [-73.856077, 40.848447]  
}  
db.places.insert(  
  {  
    loc : { type: "Point", coordinates: [ -73.97, 40.77 ] },  
    name: "Central Park",  
    category : "Parks"  
  }  
)
```

https://www.researchgate.net/publication/339292186_ANALIZ_PROIZVODITELNOSTI_BAZ_DANNYH_POSTGRESOLPOSTGIS_IMONGODB_DLA_GEOPROSTRANSTVENNYH_ZAPROSOV_PERFORMANCE_ANALYSIS_FOR_GEOSPATIAL_QUERIES_POSTGRESOLPOSTGIS_DATABASE_VS_MONGODB

Индексы

- **ТЕКСТОВЫЕ ИНДЕКСЫ** <https://docs.mongodb.com/manual/core/index-text/>

```
db.reviews.createIndex( { comments: "text" } )  
db.reviews.createIndex(  
  {  
    subject: "text",  
    comments: "text"  
  }  
)
```

только 1 на коллекцию

- **Wildcard Text Indexes** <https://docs.mongodb.com/manual/core/index-wildcard/>

```
db.collection.createIndex( { "$**": "text" } )
```

- **can be part of a compound indexes**

```
db.collection.createIndex( { a: 1, "$**": "text" } )
```

- **Case Insensitivity**

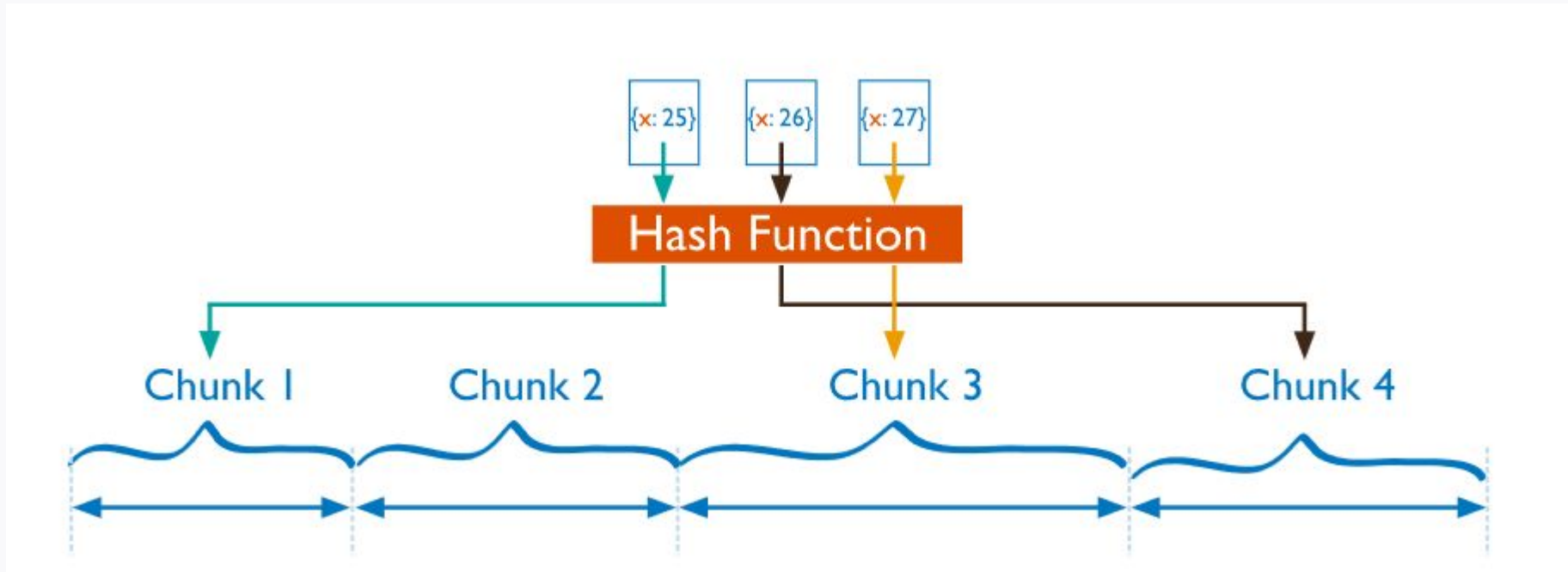
```
db.reviews.createIndex( { comments: "text" }, { name: "date_category_fr", collation: { locale: "fr", strength: 2 } } )
```

Индексы

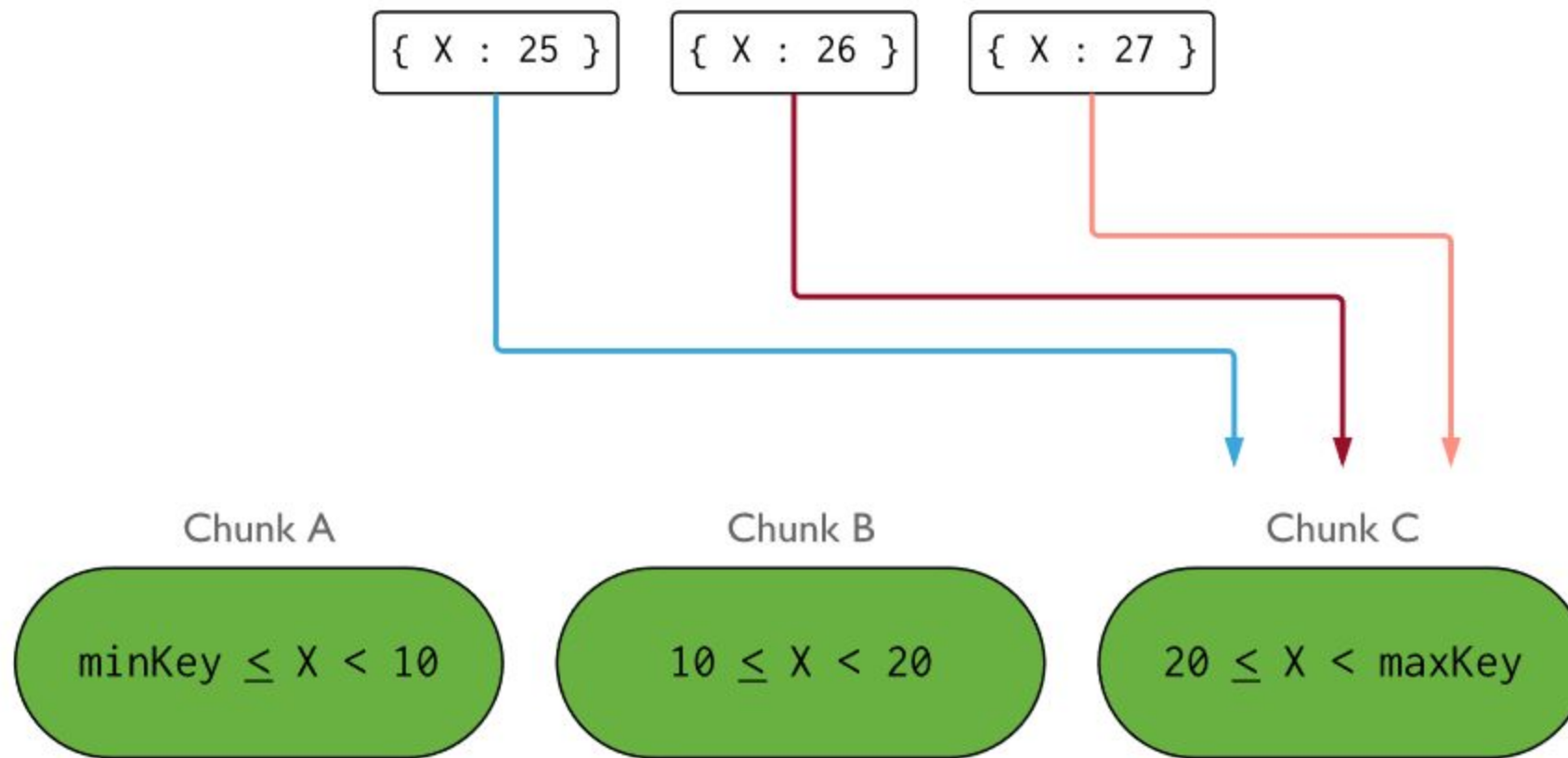
- **хэш индексы**

```
db.collection.createIndex( { _id: "hashed" } )
```

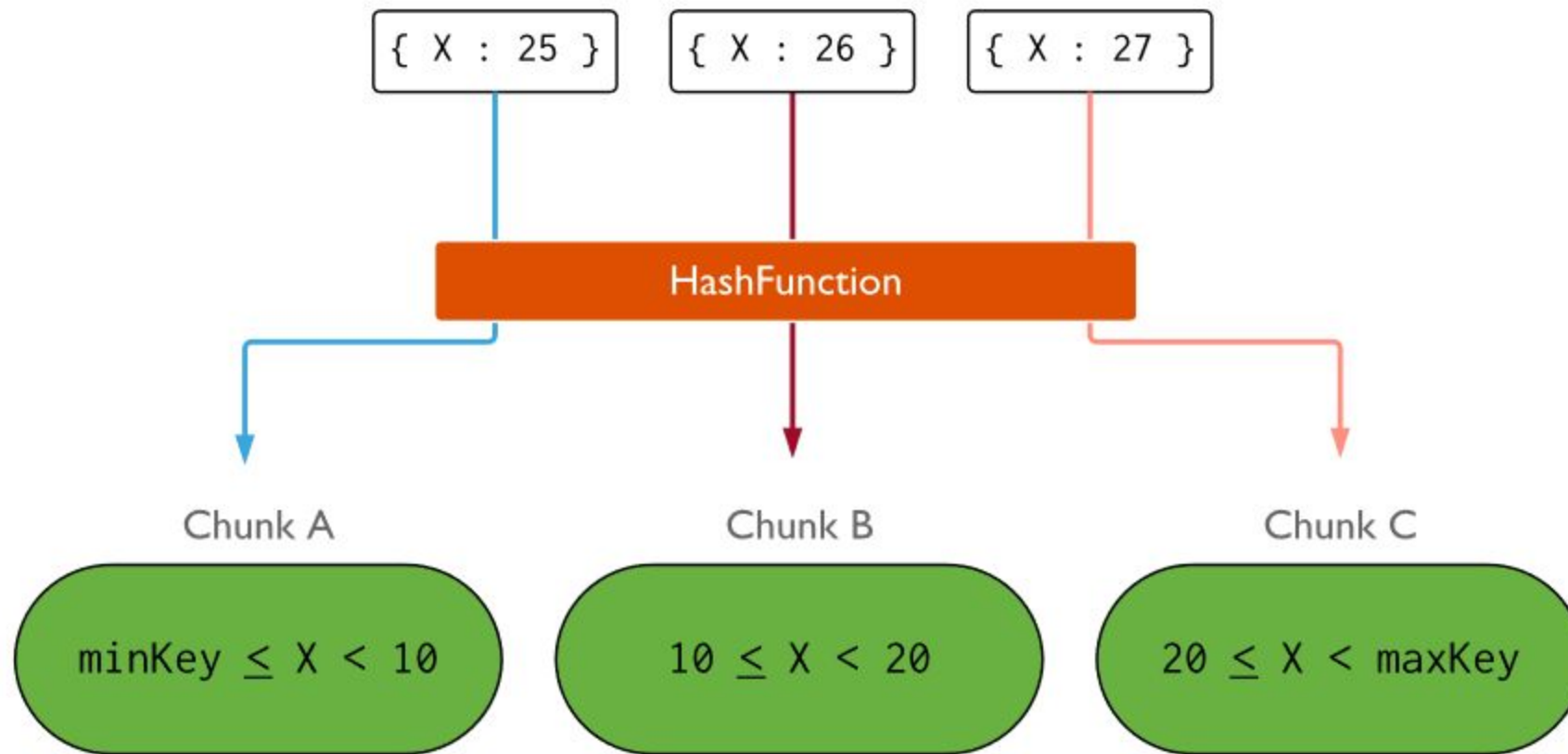
```
sh.shardCollection( "database.collection", { <field> : "hashed" } )
```



Индексы



Индексы



Индексы

new in 4.4

Hidden indexes

<https://docs.mongodb.com/manual/core/index-hidden/>

Индексы

Оптимизация. План выполнения

```
db.users.find({}).explain('allPlansExecution')
```

```
db.users.find({}).hint({_id: 1}) //to use index
```

```
db.users.find({}).hint({$natural: 1}) //NOT to use index
```


Индексы

Ограничения:

- максимум 64 индекса на коллекцию
- ключи не могут быть больше 1023 символов
- имя индекса включая неймспейс < 127
- снижают производительность добавления документов
- без индексная сортировка в памяти ограничена 32мб



Вопросы?



МИНИТЕСТ

<https://forms.gle/akESFyTj1YPiFbZ98>

3-5 минут

The background of the slide is a blue-tinted aerial photograph of a dense city skyline, likely New York City. Overlaid on this is a semi-transparent network pattern consisting of white dots connected by thin white lines, creating a web-like effect across the center of the image.

Оптимизации CRUD

CRUD

CRUD - IFUR <https://docs.mongodb.com/manual/crud/>

find()

```
db.users.find({ age: 27 })
```

```
db.users.findOne({ age: 27 })
```

```
db.users.find({ nick: 'John', age: 27 })
```

```
db.users.find({}, { nick: true, age: true }) // проекция
```

```
db.users.find({}, { age: false })
```

```
db.users.find({ age: 27 }).limit(1)
```

```
db.users.find({ age: 27 }).skip(50)
```

```
db.users.find({ age: 27 }).count()
```

CRUD

null

```
{ "_id" : ObjectId("4ba0f0dfd22aa494fd523621"), "y" : null }
```

```
{ "_id" : ObjectId("4ba0f0dfd22aa494fd523622"), "y" : 1 }
```

```
{ "_id" : ObjectId("4ba0f148d22aa494fd523623"), "y" : 2 }
```

```
db.users.find({ y: null }) //1
```

```
db.users.find({ z: null }) //3
```

```
db.users.find({ z: { $eq: null, $exists: true } })
```

Perl Compatible Regular Expression

```
db.users.find({ name: /joe/i })
```

```
db.users.find({ name: { $regex: /joe/i } })
```

<https://habr.com/ru/post/343968/> - batch read

CRUD

insert

```
db.inventory.insertOne(
  { item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm" } }
)
db.inventory.insertMany([
  { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },
  { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
  { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }
])
```

Write Concern Specification

Write concern can include the following fields: { w: <value>, j: <boolean>, wtimeout: <number> }

```
db.inventory.insert(
  { sku: "abcdxyz", qty : 100, category: "Clothing" },
  { writeConcern: { w: "majority", j: true, wtimeout: 5000 } }
)
```

CRUD

update

whole object update

```
db.people.update({name: 'Tom'}, {age: 29, name: 'Tom'})
```

```
db.people.updateMany({name: 'Tom'}, {age: 29, name: 'Tom'}) //Will replace all matching documents.
```

```
db.people.updateOne({name: 'Tom'}, {age: 29, name: 'Tom'}) //Will replace only first matching document.
```

only 1 field

```
db.people.update({name: 'Tom'}, {$set: {age: 29}})
```

updateOne vs replaceOne

<https://docs.mongodb.com/manual/reference/method/db.collection.replaceOne/index.html>

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

CRUD

update

\$push

```
db.people.update({name: 'Tom'}, {$push: {nicknames: 'Tommy'}})
```

// This adds the string 'Tommy' into the nicknames array in Tom's document.

\$pull

```
db.people.update({name: 'Tom'}, {$pull: {nicknames: 'Tommy'}})
```

// This removes the string 'Tommy' from the nicknames array in Tom's document.

\$pop

```
db.people.update({name: 'Tom'}, {$pop: {siblings: -1}})
```

// This will remove the first value from the siblings array, which is 'Marie' in this case.

CRUD

update

обновление встроенных документов - \$ до первого вхождения

```
{name: 'Tom', age: 28, marks: [50, 60, 70]}
```

```
db.people.update({name: "Tom", marks: 50}, {"$set": {"marks.$": 55}})
```

```
{name: 'Tom', age: 28, marks: [{subject: "English", marks: 90}, {subject: "Maths", marks: 100}, {subject: "Computes", marks: 20}]}
```

```
db.people.update({name: "Tom", "marks.subject": "English"}, {"$set": {"????????": 85}})
```

CRUD

update

обновление встроенных документов - \$ до первого вхождения

```
{name: 'Tom', age: 28, marks: [50, 60, 70]}
```

```
db.people.update({name: "Tom", marks: 50}, {"$set": {"marks.$": 55}})
```

```
{name: 'Tom', age: 28, marks: [{subject: "English", marks: 90},{subject: "Maths", marks: 100}, {subject: "Computes", marks: 20}]}
```

```
db.people.update({name: "Tom", "marks.subject": "English"}, {"$set": {"marks.$.marks": 85}})
```

CRUD

delete

Удаляет все документы, соответствующие параметру запроса:

```
db.people.deleteMany({name: 'Tom'})
```

```
db.people.remove({name: 'Tom'})
```

Или просто один

```
db.people.deleteOne({name: 'Tom'})
```

```
db.people.remove({name: 'Tom'}, true)
```


CRUD

bulk write

<https://docs.mongodb.com/manual/core/bulk-write-operations/>

The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band that contains a white network or mesh pattern of interconnected lines and dots. Centered within this band is the word "Вопросы?" in a large, white, sans-serif font.

Вопросы?

The background of the image is an aerial photograph of a dense urban area, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer that features a white network pattern of interconnected dots and lines, resembling a data or communication network. The text is centered within this blue layer.

Загрузка Open Street Map 2 geo

OSM 2 geo

<https://github.com/autumnus/pbf2mongo>

Скачаем карты с Open Street Map

<https://www.openstreetmap.org/search?query=%D0%BC%D0%BE%D1%81%D0%BA%D0%B2%D0%B0#map=10/55.7252/37.6290>

можно самим представить на карте используя

<https://www.qgis.org/ru/site/>

The background of the image is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. The image is divided into three horizontal sections. The top and bottom sections show the city buildings. The middle section is a solid blue band with a white, glowing network pattern of lines and dots. The title text is centered within this blue band.

Дисковые движки

Дисковые движки

- **MMAPv1** removed in 4.2
- **In-Memory** is available in MongoDB Enterprise
- **WiredTiger** is the default storage engine starting in MongoDB 3.2

<https://www.percona.com/blog/2019/01/03/mongodb-engines-mmapv1-vs-wiredtiger/>

ДИСКОВЫЕ ДВИЖКИ

- **MMAPv1** removed in 4.2

МИНУСЫ:

- Doesn't support compression
- Operation on a single document transaction
- very poor locking
- low CPU performance, adding additional CPU not improve performance
- Encryption is not possible
- small chance to tune it

ДИСКОВЫЕ ДВИЖКИ

- **In-Memory** is available in MongoDB Enterprise

mongod --storageEngine inMemory --dbpath <path>

- --dbpath, or [storage.dbPath](#) if using a configuration file. Although the in-memory storage engine does not write data to the filesystem, it maintains in the --dbpath small metadata files and diagnostic data as well temporary files for building large indexes.
- By default, the in-memory storage engine uses 50% of physical RAM minus 1 GB.
- "WT_CACHE_FULL: operation would overflow cache" when operation exceeds memory size

ДИСКОВЫЕ ДВИЖКИ

- **In-Memory**

Replica Set

You can deploy [mongod](#) instances that use in-memory storage engine as part of a replica set. For example, as part of a three-member replica set, you could have:

- two [mongod](#) instances run with in-memory storage engine.
- one [mongod](#) instance run with [WiredTiger](#) storage engine. Configure the WiredTiger member as a hidden member (i.e. [hidden: true](#) and [priority: 0](#)).

With this deployment model, only the [mongod](#) instances running with the in-memory storage engine can become the primary. Clients connect only to the in-memory storage engine [mongod](#) instances. Even if both [mongod](#) instances running in-memory storage engine crash and restart, they can sync from the member running WiredTiger. The hidden [mongod](#) instance running with WiredTiger persists the data to disk, including the user data, indexes, and replication configuration information.

-

ДИСКОВЫЕ ДВИЖКИ

- **In-Memory**

Sharding

```
sh.addShardTag("shardC", "inmem")
```

To the other shards, add a separate tag persisted .

```
sh.addShardTag("shardA", "persisted")
```

```
sh.addShardTag("shardB", "persisted")
```

```
sh.addTagRange("test.analytics", { shardKey: MinKey }, { shardKey: MaxKey }, "inmem")
```

```
sh.addTagRange("salesdb.orders", { shardKey: MinKey }, { shardKey: MaxKey }, "persisted")
```

ДИСКОВЫЕ ДВИЖКИ

- **WiredTiger** is the default storage engine starting in MongoDB 3.2

СВОЙСТВА

- Document Level Concurrency
- uses MultiVersion Concurrency Control (MVCC)
- Snapshots and Checkpoints
- Journal (WAL)
- Compression (zlib, zstd since 4.2)
- Memory Use (50% of (RAM - 1 GB))

<https://docs.mongodb.com/manual/reference/configuration-options/#storage-options>

Starting in version 3.6, MongoDB configures WiredTiger to create checkpoints (i.e. write the snapshot data to disk) at intervals of 60 seconds. In earlier versions, MongoDB sets checkpoints to occur in WiredTiger on user data at an interval of 60 seconds or when 2 GB of journal data has been written, whichever occurs first.

ДИСКОВЫЕ ДВИЖКИ

Via the filesystem cache, MongoDB automatically uses all free memory that is not used by the WiredTiger cache or by other processes.

To adjust the size of the WiredTiger internal cache, see `storage.wiredTiger.engineConfig.cacheSizeGB` and `--wiredTigerCacheSizeGB`. Avoid increasing the WiredTiger internal cache size above its default value.

The background of the image is an aerial photograph of a city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue gradient and a network of white lines connecting various points, creating a digital or data network aesthetic. The text "GridFS" is centered in the middle of the image.

GridFS

GridFS

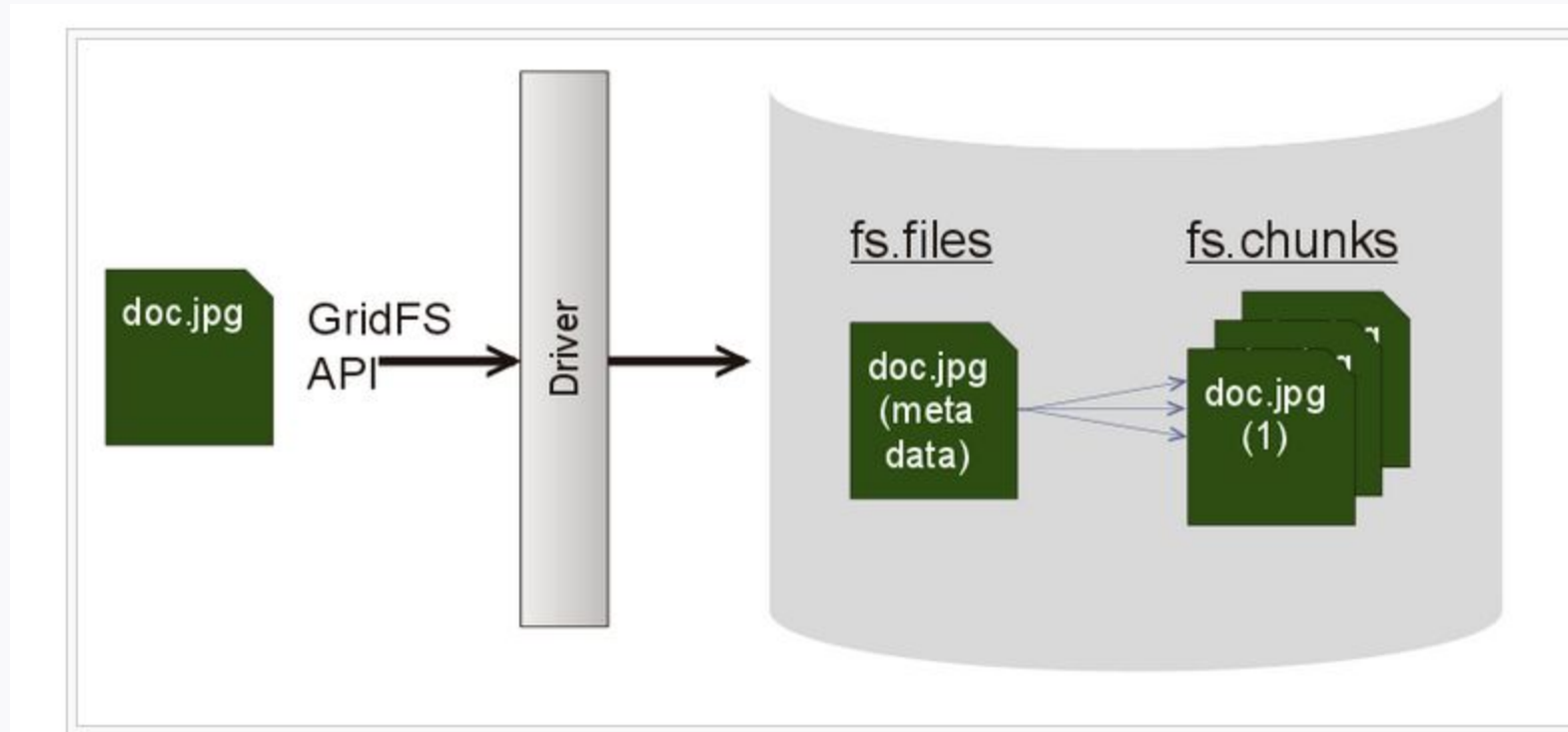
GridFS

- правила хранения больших по объему файлов в базе данных [MongoDB](#).
- GridFS позволяет хранить файлы больше 16Мб (предельный размер документа в [MongoDB](#)) за счет использования коллекций.
- GridFS разбивает большие файлы на небольшие части (255кб). Эти части сохраняются в одну коллекцию (fs.chunks), а метаданные о файле в другую коллекцию (fs.files).
- Когда происходит запрос к файлу, GridFS делает запрос в коллекцию с частями файла и затем возвращает файл целиком.

Плюсы технологии

- не обязательно хранить в памяти весь файл
- доступ к рандомной части файла, например фильм с середины

GridFS



GridFS

Chunks Collection

Каждый документ в коллекции `chunks` представляет собой отдельный фрагмент файла и имеет следующую структуру:

```
{  
  "_id" : <ObjectId>,  
  "files_id" : <ObjectId>, //Идентификатор _id "родительского" документа, как указано в коллекции файлов -  
    fs.files.  
  "n" : <num>, //Номер фрагмента в последовательности частей chunks. GridFS показывает все куски,  
    начиная с 0  
  "data" : <binary> //Полезная нагрузка блока (бинарный тип BSON Binary type)  
}
```

GridFS

Files Collection

Каждый документ в коллекции файлов представляет собой файл в GridFS.

```
{  
  "_id" : <ObjectId>,  
  "length" : <num>,  
  "chunkSize" : <num>,  
  "uploadDate" : <timestamp>,  
  "md5" : <hash>,  
  "filename" : <string>,  
  "contentType" : <string>,  
  "aliases" : <string array>,  
  "metadata" : <any>,  
}
```


GridFS

To store and retrieve files using [GridFS](#), use either of the following:

- A MongoDB driver. See the [drivers](#) documentation for information on using GridFS with your driver.
- The [mongofiles](#) command-line tool. See the [mongofiles](#) reference for documentation.

```
mongofiles -d=records put 32-corinth.lp
```

```
mongofiles -d=records get 32-corinth.lp
```

```
mongofiles -d=records delete 32-corinth.lp
```

```
mongofiles -d=records search corinth
```

```
db.fs.files.find( { filename: myFileName } ).sort( { uploadDate: 1 } )
```

```
mongofiles -d=records get_id 'ObjectId("56feac751f417d0357e7140f")'
```


The background of the slide is a high-angle, blue-tinted aerial photograph of a dense urban skyline, likely New York City. Overlaid on this image is a semi-transparent blue band across the middle, which contains a white network diagram of interconnected nodes and lines. The main title is centered within this band.

Производительность кластеров

Производительность кластера

Чек-лист при планировании схемы данных и индексов коллекции:

1. Экономим трафик репликации — минимизируем размер обновления документа.
2. Помним о rollback — бизнес-логика должна поддерживать обрыв операций записи, так же как и внезапное выключение сервера.
3. Не препятствуем шардингу — на больших коллекциях не заводим больше одного индекса с уникальностью, но лучше вообще не включать уникальность, если это не критично для бизнес-логики.
4. Изолируем поисковые запросы одним шардом — большинство поисковых запросов должно включать в себя значения ключа шардинга.
5. Балансировка шардов — тщательно выбираем тип идентификатора, самый лучший из них — GUID.

Производительность кластера

Чек-лист при планировании схемы данных и индексов коллекции:

6. Используем проекции - возвращаем только нужные поля
7. Профилируем тяжелые запросы.
8. Не забываем смотреть за индексами.
9. Мониторим память/проц/іо.
10. Баланс между синхронным/асинхронным/геораспределенным
11. Серебряной пули не существует %)

The background of the slide features an aerial view of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer that contains a network of white lines and dots, resembling a data or communication network. The text "Тюнинг линукса" is centered in the middle of the slide in a white, bold, sans-serif font.

Тюнинг линукса

Тюнинг памяти

Transparent huge pages

```
cat /proc/meminfo
```

```
cat /sys/kernel/mm/transparent_hugepage/enabled
```

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

<https://docs.mongodb.com/manual/tutorial/transparent-huge-pages/>

<https://habr.com/ru/company/tinkoff/blog/446342/>

Тюнинг памяти

Swap

`sysctl vm.swappiness`

по умолчанию 60, рекомендация 1

https://habr.com/ru/post/344836/#comment_10569644

другие настройки


<https://habr.com/ru/company/otus/blog/340870/>

The image features a blue-tinted aerial view of a dense city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue band with a white network pattern of dots and lines runs horizontally across the middle of the image. The Russian word "Порефлексируем" is written in white, bold, sans-serif font across this band.

Порефлексируем

Вопросы?


- Кто что запомнил за сегодня?
- Что больше всего понравилась на вебинаре?
- Что будете использовать в своей практике?

The image features a high-angle, aerial view of a dense urban skyline, likely New York City, with numerous skyscrapers and buildings. The entire image is tinted with a blue color scheme. A semi-transparent network of white lines and dots is overlaid on the image, particularly prominent in the central horizontal band. In the center of this band, the Cyrillic letters "ДЗ" are displayed in a large, white, sans-serif font.


ДЗ

ДЗ

Добавить индексы в свой проект, сравнить производительность запросов

The background of the image is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. The entire image is overlaid with a semi-transparent blue filter. A network of white lines and dots, resembling a molecular or digital structure, is superimposed on the blue background, particularly concentrated around the text area.

Заполните, пожалуйста,
опрос о занятии по ссылке в чате
<https://otus.ru/polls/34474/>

The background of the entire slide is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. A semi-transparent blue overlay covers the image, featuring a subtle network pattern of white lines and dots. The text is centered in the middle of the slide.

Спасибо за внимание!
Приходите на следующие вебинары

Аристов Евгений