САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

Базы данных

Лекция 11
NoSQL. MongoDB.
Базовые операции

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

Самара

План лекции

- NoSQL
  - Недостатки реляционных БД
  - NoSQL БД
  - Теорема CAP

- MongoDB:
  - Добавление данных
  - Выборка данных
  - Изменение данных
  - Удаление данных

Реляционная модель БД

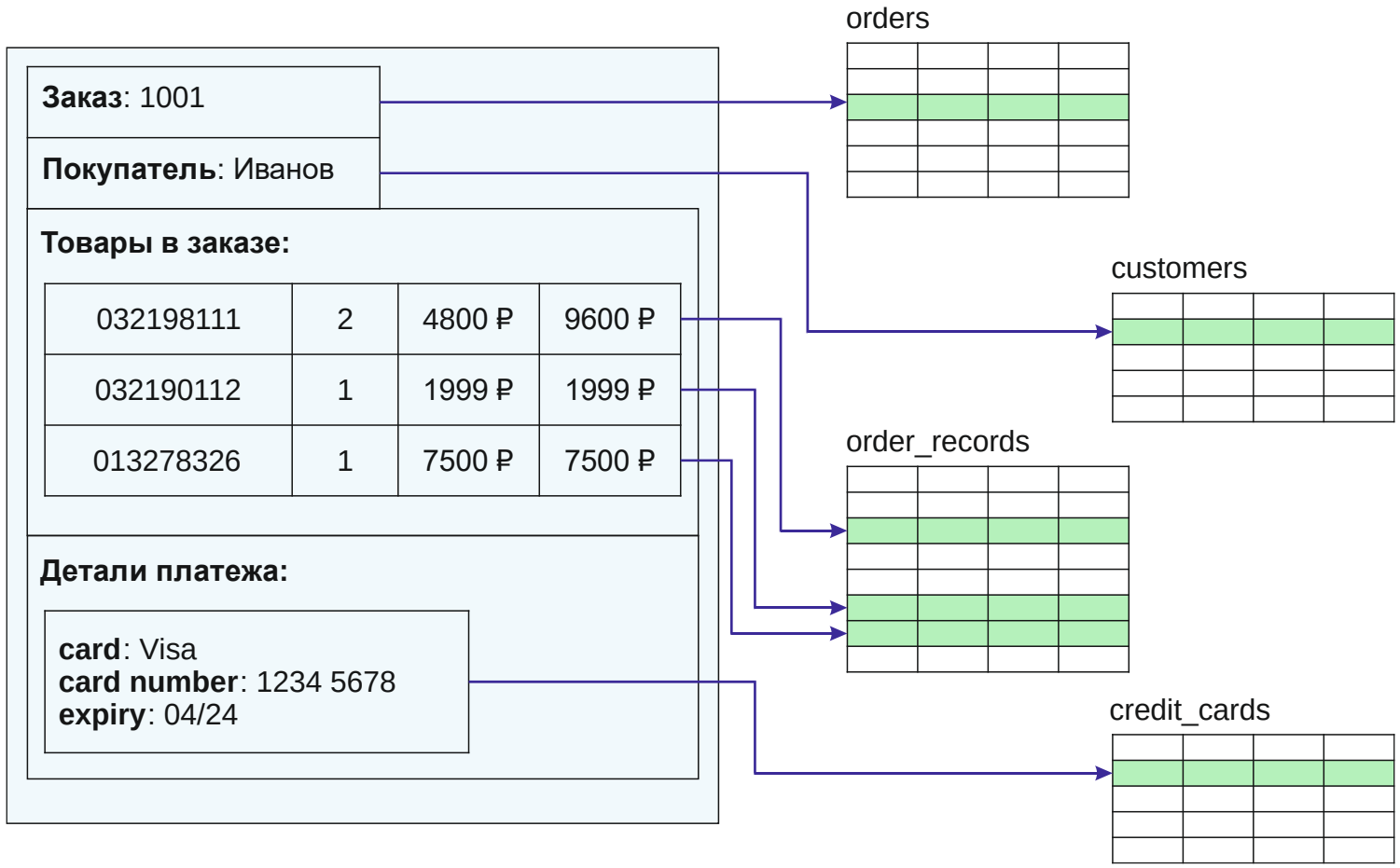
Достоинства:

- Простота представления структуры БД, нормализация данных;
- Стандарт SQL;
- Выполнение требований ACID (атомарность, согласованность, изолированность, долговечность).

Недостатки:

- Потеря соответствия (impedance mismatch) – различие между реляционной моделью и структурами данных, находящимися в памяти;
- Сложность горизонтального масштабирования базы данных.

Потеря соответствия



Сложность масштабирования

Вертикальное масштабирование — увеличение производительности каждого компонента системы с целью повышения общей производительности.

Горизонтальное масштабирование — разбиение системы на более мелкие структурные компоненты и разнесение их по отдельным физическим машинам (или их группам), и (или) увеличение количества серверов, параллельно выполняющих одну и ту же функцию.

Возможности:

- Использование общей дисковой подсистемы;
- Сегментирование базы данных (sharding).

NoSQL базы данных

- Модель «ключ — значение» является простейшим вариантом, использующим ключ для доступа к значению. Такие базы данных как правило используют хеш-таблицу, в которой находится уникальный ключ и указатель на конкретный объект данных.
  Riak, Amazon DynamoDB, Redis, Berkeley DB, MemcacheDB

- Документоориентированные СУБД служат для хранения иерархических структур данных (документов). В отличие от БД типа «ключ-значение» позволяют выполнять запросы на основе содержимого.
  CouchDB, Couchbase, MongoDB, eXist, Berkeley DB XML

- В системах типа «семейство столбцов» данные хранятся в виде разреженной матрицы, строки и столбцы которой используются как ключи. Преимущества хранения данных в столбцах заключаются в быстром поиске/доступе и агрегации данных.
  Apache HBase, Apache Cassandra, ScyllaDB, Apache Accumulo, Hypertable

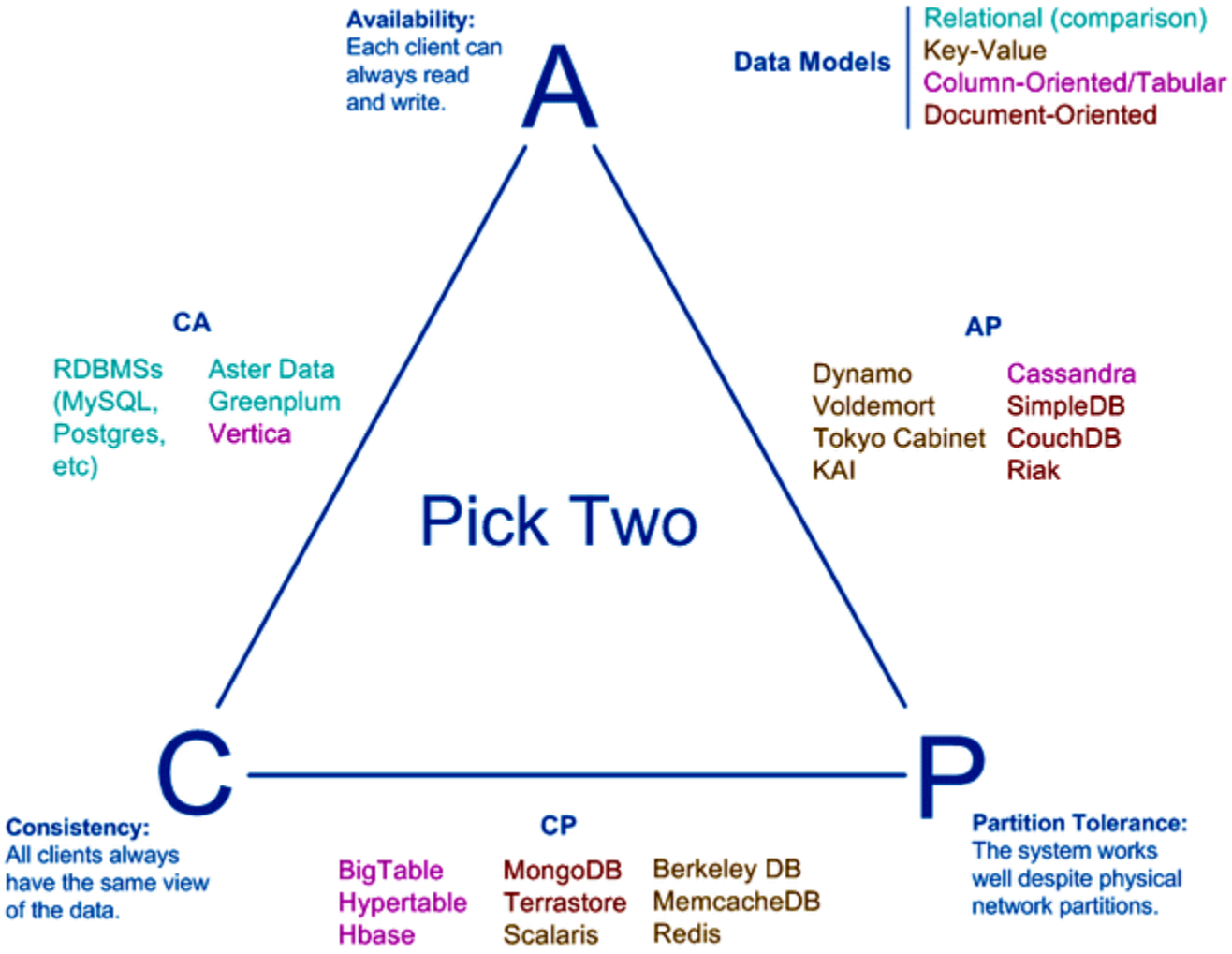
- Графовые СУБД применяются для хранения графовых структур. Такие базы данных используют рёбра и узлы для представления данных.
  Neo4j, OrientDB, AllegroGraph, Blazegraph, InfiniteGraph, FlockDB, Titan

Теорема CAP

Теорема CAP — эвристическое утверждение о том, что в любой реализации распределённых вычислений возможно обеспечить не более двух из трёх следующих свойств:

- **согласованность** данных (consistency) — во всех вычислительных узлах в один момент времени данные не противоречат друг другу;
- **доступность** (availability) — любой запрос к распределённой системе завершается корректным откликом, однако без гарантии, что ответы всех узлов системы совпадают;
- **устойчивость к разделению** (partition tolerance) — расщепление распределённой системы на несколько изолированных секций не приводит к некорректности отклика от каждой из секций.

Теорема CAP


Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

Критика концепции

- **Согласованность (consistency)** в CAP означает линеаризуемость, а не фиксацию завершенной транзакции, как в ACID. **Линеаризуемость** – это локальное и неблокируемое свойство программы, при котором результат любого параллельного выполнения операций эквивалентен их последовательному выполнению. В реальности это значит, что информация во всех репликах, включая кэшированные данные, должна быть одна и та же. Достичь этого не очень просто.
- Определение **доступности (availability)** ничего не говорит про временную задержку обработки данных (latency), при большом значении которой систему сложно назвать доступной на практике.
- **Устойчивость к разделению** (**partition tolerance**) означает, что для связи используется асинхронная сеть, которая может терять или задерживать сообщения, что характерно для любой Интернет-системы.

Альтернативные подходы. BASE

BASE-архитектура:

- базовая доступность (basically available) – подход к проектированию приложения, чтобы сбой в некоторых узлах приводил к отказу в обслуживании только для незначительной части сессий при сохранении доступности в большинстве случаев;
- неустойчивое состояние (soft-state) – возможность жертвовать долговременным хранением состояния сессий (промежуточные результаты выборок, информация о навигации и контексте) в пользу фиксации обновлений только критичных операций;
- согласованность в конечном счёте (eventually consistent) – предполагает возможность противоречивости данных в некоторых случаях, но гарантирует итоговую целостность информации в практически обозримое время.

Альтернативные подходы. PACELC

Теорема PACELC — расширение теоремы CAP, которое гласит, что в случае разделения сети (P) в распределённой компьютерной системе необходимо выбирать между доступностью (A) и согласованностью (C) (согласно теореме CAP), но в любом случае, даже если система работает нормально в отсутствии разделения (E), нужно выбирать между задержками (L) и согласованностью (C).

В логическом выражении PACELC формулируют следующим образом:
IF P -> (C or A), ELSE (C or L)

PACELC расширяет и уточняет CAP-теорему, регламентируя необходимость поиска компромисса между временной задержкой и согласованностью данных в распределенных системах.

Документоориентированная СУБД MongoDB

MongoDB — документоориентированная система управления базами данных, не требующая описания схемы таблиц. Считается одним из классических примеров NoSQL-систем, использует JSON-подобные документы и схему базы данных.

Возможности:

- поддержка запросов по значениям полей документа;
- поддержка индексов;
- поддержка репликации, т.е. работа с двумя или более копиями данных на различных узлах;
- **поддержка горизонтального масштабирования;**
- поддержка агрегации данных.

Недостатки:

- отсутствие схемы усложняет разработку типовых приложений. Кроме того, это увеличивает объёмы базы, т.к. каждый документ, кроме своих данных, хранит так же имена полей;
- денормализация данных и отсутствие JOIN. Возможности денормализации ограничиваются размером документа (16 Мб по умолчанию). Существующий аналог `JOIN` – оператор `$lookup` – не работает для сегментированных коллекций;
- заявленная поддержка транзакций ограничена, выполнение транзакций занимает относительно много времени на сегментированных коллекциях.

Основные понятия MongoDB

Документ — упорядоченный набор ключей со связанными значениям.

Коллекция — группа документов.

База данных — группа коллекций.

Реляционная СУБД	**MongoDB**
База данных	База данных
Таблица	Коллекция
Кортеж / строка	Документ
Столбец	Поле
Первичный ключ	Первичный ключ (ключ по умолчанию `_id`)

Типы данных

Спецификация	Описание
`Null`	Null можно использовать для обозначения как нулевого значения, так и несуществующего поля: `{"x" : null}`
Логический тип	Тип данных, который можно использовать для значений `true` и `false`: `{"x" : true}`

Числовой тип	По умолчанию используются 64-битные числа с плавающей точкой:
	`{"x" : 3.14}`
	`{"x" : 3}`
	Для хранения целых чисел используются классы `NumberInt` или `NumberLong`, которые обозначают 4-байтовые или 8-байтовые целые числа со знаком соответственно:
	`{"x" : NumberInt ("3")}`
	`{"x" : NumberLong ("3")}`

Строковый тип	Строка символов в кодировке UTF-8 может быть представлена с использованием строкового типа: `{"x" : "foobar"}`

Типы данных

Спецификация	Описание
Дата	Дата хранится в виде 64-битных целых чисел, обозначающих миллисекунды с момента эпохи Unix (1 января 1970 г.). Часовой пояс не сохраняется: `{"x" : new Date()}`
Массив	Наборы или списки значений могут быть представлены в виде массивов: `{"x" : ["a", "b", "c"]}`
Вложенный документ	Документы могут содержать вложенные документы: `{"x" : {"foo": "bar"} }`
Идентификатор объекта	Идентификатор объекта – это 12-байтовый идентификатор для документов: `{"x" : ObjectId ()}`
Двоичные данные	Двоичные данные – это строка из произвольных байтов.
Код	MongoDB также позволяет хранить произвольный код JavaScript в запросах и документах: `{"x" : function() { /* ... */ }}`

Создание документов

`db.collection.insertOne(`
`  <document>,`
`  {`
`  writeConcern: <document>`
`  }`
`)`
	Добавление одного документа в коллекцию `collection`

`db.collection.insertMany(`
`  [ <document 1> , <document 2>, ... ],`
`  {`
`  writeConcern: <document>,`
`  ordered: <boolean>`
`  }`
`)`
	Добавление нескольких документов в коллекцию `collection`

Создание коллекции

Если коллекция не существует, операции вставки создадут коллекцию.

Создание документов. Примеры

Создание одного документа с помощью `insertOne()`

```
use db_lectures
db.products.insertOne( { item: "card", qty: 15 } );
```

Результат добавления:

```
{
    "acknowledged" : true,
    "insertedId" : ObjectId("62df9bbf5dc132d500314e6b")
}
```

Создание документов. Примеры

Создание документа с явным указанием идентификатора

```
db.products.insertOne( { _id: 10, item: "box", qty: 20 } );
{ "acknowledged" : true, "insertedId" : 10 }
```

Создание документа с тем же идентификатором

```
db.products.insertOne( { _id: 10, "item" : "package", "qty" : 200 } );
WriteError({
        "index" : 0,
        "code" : 11000,
        "errmsg" : "E11000 duplicate key error collection:
                    db_lectures.products index: _id_ dup key: { _id: 10.0 }",
        "op" : {
                "_id" : 10,
                "item" : "package",
                "qty" : 200
        }
})
```


Создание документов. Примеры

Создание нескольких документов с помощью `insertMany()`

```
db.products.insertMany( [
    { item: "card", qty: 15 },
    { item: "envelope", qty: 20 },
    { item: "stamps" , qty: 30 }
]);
```

Результат добавления:

```
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("62dfa8f75dc132d500314e6d"),
        ObjectId("62dfa8f75dc132d500314e6e"),
        ObjectId("62dfa8f75dc132d500314e6f")
    ]
}
```

Создание документов. Примеры

Создание нескольких документов с помощью `insertMany()`

```
db.products.insertMany( [
    { _id: 13, item: "envelopes", qty: 60 },
    { _id: 13, item: "stamps", qty: 110 },
    { _id: 14, item: "packing tape", qty: 38 }
]);
```

```
E11000 duplicate key error collection:
db_lectures.products index: _id_ dup key: { _id: 13 }
```


Выборка документов

`db.collection.find(` `query, projection)`	Выборка документов из коллекции `collection` • `query` – определяет условие фильтрации документов; • `projection` – определяет поля, которые должны возвращаться в документах, соответствующих фильтру запроса.

Выборка документов. Примеры

Заполнение коллекции

```
db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Выборка все документов

```
db.inventory.find( {} )
```

```
SELECT * FROM inventory;
```

Фильтрация документов. Операторы сравнения

```
{
  <field1>: <value1>,
  <field2>: { <operator>: <value> },
  ...
}
```

Оператор	Описание
$eq	Равно (=)
$ne	Не равно (!=)
$gt	Больше (>)
$gte	Больше или равно (>=)
$lt	Меньше (<)
$lte	Меньше или равно (<=)
$in	Соответствие хотя бы одному элементу массива
$nin	Несоответствие ни одному элементу массива

Фильтрация документов. Логические операторы

Оператор	Описание
`$and`	Логическое И
`$or`	Логическое ИЛИ
`$not`	Логическое отрицание
`$nor`	Логическое НЕ-ИЛИ. Возвращает все документы, которые не соответствуют обоим предложениям

Фильтрация документов. Проверка элементов и массивов

Оператор	Описание
`$exists`	Возвращает документы, которые содержат указанное поле
`$type`	Возвращает документы, если поле имеет указанный тип.
`$regex`	Возвращает документы, значения полей которых соответствуют указанному регулярному выражению

Оператор	Описание
`$all`	Возвращает документы, массивы которых содержат все элементы, указанные в запросе
`$elemMatch`	Возвращает документы, если элемент поля-массива соответствует всем указанным условиям
`$size`	Возвращает документы, если поле-массив имеет указанный размер

Выборка документов по условию равенства

```
{ <field1>: <value1>, ... }
```

Выборка документов указанного статуса

```
db.inventory.find( { status: "D" } )

{ _id: "62dfce485dc132d500314e72", item: "paper", qty: 100,
       size: { h: 8.5, w: 11, uom: "in" }, status: "D" }
{ _id: "62dfce485dc132d500314e73", item: "planner", qty: 75,
       size: { h: 22.85, w: 30, uom: "cm" }, status: "D" }
```

```
SELECT * FROM inventory WHERE status = 'D';
```

Выборка документов с использованием оператора IN

```
{ <field1>: { <operator1>: <value1> }, ... }
```

Выборка документов указанного статуса (из множества)

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

```
{ _id: "62dfce485dc132d500314e70", item: "journal", qty: 25,
        size: { h: 14, w: 21, uom: "cm" }, status: "A" }
{ _id: "62dfce485dc132d500314e71", item: "notebook", qty: 50,
        size: { h: 8.5, w: 11, uom: "in" }, status: "A" }
{ _id: "62dfce485dc132d500314e72", item: "paper", qty: 100,
        size: { h: 8.5, w: 11, uom: "in" }, status: "D" }
{ _id: "62dfce485dc132d500314e73", item: "planner", qty: 75,
        size: { h: 22.85, w: 30, uom: "cm" }, status: "D" }
{ _id: "62dfce485dc132d500314e74", item: "postcard", qty: 45,
        size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
```

```
SELECT * FROM inventory WHERE status IN ('A', 'D');
```


Выборка документов с использованием оператора AND

Выборка документов по составному условию

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

```
{ _id: "62dfce485dc132d500314e70", item: "journal", qty: 25,
        size: { h: 14, w: 21, uom: "cm" }, status: "A" }
```

```
SELECT * FROM inventory WHERE status = 'A' AND qty < 30;
```

Выборка документов с использованием оператора OR

Выборка документов по составному условию

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $gte: 100 } } ] } )
```

```
{ _id: "62dfce485dc132d500314e70", item: "journal", qty: 25,
        size: { h: 14, w: 21, uom: "cm" }, status: "A" }
{ _id: "62dfce485dc132d500314e71", item: "notebook", qty: 50,
        size: { h: 8.5, w: 11, uom: "in" }, status: "A" }
{ _id: "62dfce485dc132d500314e72", item: "paper", qty: 100,
        size: { h: 8.5, w: 11, uom: "in" }, status: "D" }
{ _id: "62dfce485dc132d500314e74", item: "postcard", qty: 45,
        size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
```

```
SELECT * FROM inventory WHERE status = 'A' OR qty >= 100;
```


Выборка документов с использованием операторов AND и OR

Выборка документов по составному условию

```
db.inventory.find( {
    status: "A",
    $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
} )
```

```
{ _id: "62dfce485dc132d500314e70", item: "journal", qty: 25,
        size: { h: 14, w: 21, uom: "cm" }, status: "A" }
{ _id: "62dfce485dc132d500314e74", item: "postcard", qty: 45,
        size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
```

```
SELECT * FROM inventory
WHERE status = 'A' AND (qty < 30 OR item LIKE 'p%');
```

Фильтрация по вложенным полям

Выборка документов по условию на поле вложенного документа

```
db.inventory.find( { "size.uom": "in" } )
```

```
{ _id: "62dfce485dc132d500314e71", item: "notebook", qty: 50,
        size: { h: 8.5, w: 11, uom: "in" }, status: "A" }
{ _id: "62dfce485dc132d500314e72", item: "paper", qty: 100,
        size: { h: 8.5, w: 11, uom: "in" }, status: "D" }
```

Выборка документов по составному условию

```
db.inventory.find( { "size.h": { $lt: 15 }, "size.uom": "in", status: "D" } )
```

```
{ _id: "62dfce485dc132d500314e72", item: "paper", qty: 100,
        size: { h: 8.5, w: 11, uom: "in" }, status: "D" }
```


Фильтрация по значениям в массиве

```
db.inventory.insertMany([
    { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
    { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
    { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] },
    { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
    { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }
]);
```

Выборка документов по точному совпадению элементов массива

```
db.inventory.find( { tags: ["red", "blank"] } )
```

```
{ _id: "62e021fd5dc132d500314e76", item: "notebook", qty: 50,
        tags: ["red", "blank"], dim_cm: [ 14, 21 ] }
```


Фильтрация по значениям в массиве

Выборка документов по совпадению элементов массива

```
db.inventory.find( { tags: { $all: ["red", "blank"] } } )
```

```
{ _id: "62e021fd5dc132d500314e75", item: "journal", qty: 25,
        tags: ["blank", "red"], dim_cm: [ 14, 21 ] }
{ _id: "62e021fd5dc132d500314e76", item: "notebook", qty: 50,
        tags: ["red", "blank"], dim_cm: [ 14, 21 ] }
{ _id: "62e021fd5dc132d500314e77", item: "paper", qty: 100,
        tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] }
{ _id: "62e021fd5dc132d500314e78", item: "planner", qty: 75,
        tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] }
```

Фильтрация по значениям в массиве

Выборка документов по составному условию на элементы массива

```
db.inventory.find( { dim_cm: { $gt: 15, $lt: 20 } } )
```

```
{ _id: "62e021fd5dc132d500314e75", item: "journal", qty: 25,
        tags: ["blank", "red"], dim_cm: [ 14, 21 ] }
{ _id: "62e021fd5dc132d500314e76", item: "notebook", qty: 50,
        tags: ["red", "blank"], dim_cm: [ 14, 21 ] }
{ _id: "62e021fd5dc132d500314e77", item: "paper", qty: 100,
        tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] }
{ _id: "62e021fd5dc132d500314e79", item: "postcard", qty: 45,
        tags: ["blue"], dim_cm: [ 10, 15.25 ] }
```

Фильтрация по значениям в массиве

Выборка документов по составному условию на элемент массива

```
db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } )
```

```
{ _id: "62e021fd5dc132d500314e78", item: "planner", qty: 75,
        tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] }
```

Возврат определенных полей

Выборка документов указанного статуса

```
db.inventory.find( { status: "D" } )
```

```
{ _id: "62dfce485dc132d500314e72", item: "paper", qty: 100,
       size: { h: 8.5, w: 11, uom: "in" }, status: "D" }
{ _id: "62dfce485dc132d500314e73", item: "planner", qty: 75,
       size: { h: 22.85, w: 30, uom: "cm" }, status: "D" }
```

```
SELECT * FROM inventory WHERE status = 'D';
```

```
db.inventory.find( { status: "D" } , { item: 1, status: 1 } )
```

```
{ _id: "62dfce485dc132d500314e72", item: "paper", status: "D" }
{ _id: "62dfce485dc132d500314e73", item: "planner", status: "D" }
```

```
SELECT _id, item, status FROM inventory WHERE status = 'D';
```

Возврат определенных полей

Выборка документов указанного статуса

```
db.inventory.find( { status: "D" } , { item: 1, status: 1, _id: 0 } )
```

```
{ item: "paper", status: "D" }
{ item: "planner", status: "D" }
```

```
SELECT item, status FROM inventory WHERE status = 'D';
```

```
db.inventory.find(
    { status: "D" } ,
    { item: 1, status: 1 , _id: 0, "size.uom": 1 }
)
```

```
{ item: "paper", size: { uom: "in" }, status: "D" }
{ item: "planner", size: { uom: "cm" }, status: "D" }
```

Методы курсора

Оператор	Описание
`cursor.count()`	Изменяет курсор, чтобы он возвращал количество документов в результирующем наборе, а не сами документы
`cursor.limit()`	Ограничивает размер результирующего набора
`cursor.map()`	Применяет функцию к каждому документу в курсоре и возвращает полученные значения в виде массива
`cursor.forEach()`	Применяет JavaScript-функцию к каждому документу в курсоре
`cursor.skip()`	Возвращает курсор, который начинает возвращать результаты только после пропуска набора документов
`cursor.sort()`	Возвращает документы, упорядоченные в соответствии со спецификацией сортировки

Использование методов курсора

Подсчет количества документов по условию

```
db.inventory.find( { dim_cm: { $gt: 15, $lt: 20 } } ).count()
```

```
4
```

Выборка двух документов, начиная с четвертого

```
db.inventory.find( {} ).limit( 2 ).skip( 3 ).sort( { item: 1, qty: -1 } )
```

```
{ _id: "62dfce485dc132d500314e71", item: "notebook", qty: 50,
        size: { h: 8.5, w: 11, uom: "in" }, status: "A" }
{ _id: "62dfce485dc132d500314e72", item: "paper", qty: 100,
        size: { h: 8.5, w: 11, uom: "in" }, status: "D" }
```


Использование методов курсора

Преобразование документов

```
db.products.find( ).limit( 5 ).map( function(p) {
        return p.item + '/' + p.qty }
)
```

```
[
    "journal / 25",
    "notebook / 50",
    "paper / 100",
    "planner / 75",
    "postcard / 45"
]
```

Обновление документов

`db.collection.updateOne(`
`  <filter>, <update>, <options>`
`)`
	Обновление одного документа

`db.collection.updateMany(`
`  <filter>, <update>, <options>`
`)`
	Обновление нескольких документов

`db.collection.replaceOne(`
`  <filter>, <update>, <options>`
`)`
	Замена документа

Обновление документов

```
db.collection.updateOne(
    <filter>,
    <update>,
    {
        upsert: <boolean>,
        writeConcern: <document>,
        collation: <document>,
        arrayFilters: [ <filterdocument1>, ... ],
        hint: <document|string>
    }
)

<update> =
{
  <update operator>: { <field1>: <value1>, ... },
  <update operator>: { <field2>: <value2>, ... },
  ...
}
```

Обновление документов. Операторы работы с полями

Оператор	Описание
`$currentDate`	Устанавливает значение поля на текущую дату
`$inc`	Увеличивает значение поля на указанную величину
`$min`	Обновляет поле только в том случае, если указанное значение меньше существующего значения поля
`$max`	Обновляет поле только в том случае, если указанное значение больше, чем существующее значение поля
`$mul`	Умножает значение поля на указанную величину
`$rename`	Переименовывает поле
`$set`	Устанавливает значение поля в документе
`$setOnInsert`	Задает значение поля, если обновление приводит к вставке документа
`$unset`	Удаляет указанное поле из документа

Обновление документов. Операторы работы с массивами

Оператор	Описание
`$`	Модификатор, указывающий, что необходимо обновить первый элемент, соответствующий условию запроса
`$[]`	Модификатор, указывающий, что необходимо обновить все элементы, соответствующие условию запроса
`$pop`	Удаляет первый или последний элемент массива
`$pull`	Удаляет все элементы массива, соответствующие указанному запросу
`$push`	Добавляет элемент в массив
`$pullAll`	Удаляет все элементы массива, соответствующие указанным значениям

Обновление документов

Обновление одного документа указанного статуса

```
db.inventory.find( { status: "D" } )

{ _id: "62dfce485dc132d500314e72", item: "paper", qty: 100,
        size: { h: 8.5, w: 11, uom: "in" }, status: "D" }
{ _id: "62dfce485dc132d500314e73", item: "planner", qty: 75,
        size: { h: 22.85, w: 30, uom: "cm" }, status: "D" }


db.inventory.updateOne(
    { status: "D" },
    {
        $inc: { qty : -10 },
        $set: { "size.uom": "cm", item: "paper_updated" },
        $currentDate: { lastModified: true }
    }
)

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

Обновление документов

Проверка обновления документа

```
db.inventory.find( { status: "D" } )

{ _id: "62dfce485dc132d500314e72", item: "paper_updated", qty: 90,
       size: { h: 8.5, w: 11, uom: "cm" }, status: "D",
       lastModified: ISODate("2022-07-28T10:18:52.901Z") }
{ _id: "62dfce485dc132d500314e73", item: "planner", qty: 75,
       size: { h: 22.85, w: 30, uom: "cm" }, status: "D" }
```

Обновление документов

Обновление всех документов указанного статуса

```
db.inventory.updateMany(
    { status: "D" },
    {
        $unset: { "size.uom": "", "lastModified": ""},
        $set: { price: 99.99 }
    }
)
```

```
{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }
```

```
db.inventory.find( { status: "D" } )
```

```
{ _id: "62dfce485dc132d500314e72", item: "paper_updated", qty: 90,
        size: { h: 8.5, w: 11 }, status: "D", price: 99.99 }
{ _id: "62dfce485dc132d500314e73", item: "planner", qty: 75,
        size: { h: 22.85, w: 30 }, status: "D", price: 99.99 }
```

Обновление документов

Замена документа по указанному идентификатору

```
db.inventory.replaceOne (
    { _id: ObjectId("62dfce485dc132d500314e72") },
    { item: "paper", instock:
       [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }
)
```

```
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
db.inventory.find( {item: "paper" } )
```

```
{ _id: "62dfce485dc132d500314e72", item: "paper", instock:
       [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 40 } ] }
{ _id: "62e021fd5dc132d500314e77", item: "paper", qty: 100,
       tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] }
```

Удаление документов

`db.collection.deleteOne(`
`  <filter>, <options>`
`)`
	Удаление одного документа

`db.collection.deleteMany(`
`  <filter>, <options>`
`)`
	Удаление нескольких документов

Удаление документов

Удаление одного документа указанного статуса

```
db.inventory.find( { status: "D" } )
```

```
{ _id: "62dfce485dc132d500314e73", item: "planner", qty: 75,
        size: { h: 22.85, w: 30, uom: "cm" }, status: "D" }
```

```
db.inventory.deleteOne(
    { status: "D" },
)
```

```
{ "acknowledged" : true, "deletedCount" : 1}
```

```
db.inventory.find( { status: "D" } )
```

empty

Удаление документов

Удаление всех документов указанного статуса

```
db.inventory.deleteMany (
    { status: "A" },
)
```

```
{ "acknowledged" : true, "deletedCount" : 3}
```

Удаление всех документов

```
db.inventory.deleteMany ( {} )
```

```
{ "acknowledged" : true, "deletedCount" : 6}
```


САМАРСКИЙ УНИВЕРСИТЕТ
SAMARA UNIVERSITY

БЛАГОДАРЮ ЗА ВНИМАНИЕ

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