Преподаватель:

\_\_\_\_\_\_\_\_Говоров А.И.

Дата: «\_\_» \_\_\_\_\_\_\_ 20\_\_г.

Оценка \_\_\_\_\_\_\_

Выполнил:

Студент группы № Y2439

\_\_\_\_\_\_\_\_\_Рахимов А. А.

**ОТЧЕТ**

**О ЛАБОРАТОРНОЙ РАБОТЕ № 7**

По теме: MongoDB

По дисциплине: Основы проектирования баз данных

Специальность 09.02.07«Информационные системы и программирование»

Дисциплина «Технологии разработки и защиты баз данных»

**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИТМО»**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

Санкт-Петербург

2020

**Цель работы:** овладеть практическими навыками и умениями реализации баз данных в MongoDB.

**Задание:**

1. Попытаться уменьшить размер модели реляционной базы данных, с которой Вы работали.
2. Реализовать Вашу модель в MongoDB.
3. Заполнить модель данными, настолько, чтобы все запросы реализуемые Вами в пункте 4 выдавали непустые строки.
4. Реализовать минимум 40% Ваших запросов, сделанных в работе по реляционным базам данных в MongoDB.

**Ниже представлены логи процесса реализации модели в MongoDB**

> use Library;

>

>

> db.createCollection("book\_instance");

>

> db.book\_instance.insert(

... {

... "Libr\_id": 1,

... "Status\_book": true,

... "book": {

... "name": "Prestiplenie i nakazanie",

... "date\_edition": new Date(1934, 1, 4, 6, 12),

... "authors": "Dostoevskiy",

... "count\_instance": 6,

... "id\_book": 1

... },

... "Tabel\_num": 1

... }

... );

>

>

WriteResult({ "nInserted" : 1 })

>

> db.book\_instance.insert(

... {

... "Libr\_id": 2,

... "Status\_book": false,

... "book": {

... "name": "Voyna i mur",

... "date\_edition": new Date(1732, 14, 11, 6, 12),

... "authors": "Lev Tolstoy",

... "count\_instance": 3,

... "id\_book": 2

... },

... "Tabel\_num": 3

... }

... );

>

>

WriteResult({ "nInserted" : 1 })

> db.book\_instance.insert(

... {

... "Libr\_id": 3,

... "Status\_book": false,

... "book": {

... "name": "Vishneviy sad",

... "date\_edition": new Date(1912, 8, 3, 6, 12),

... "authors": "Chehov",

... "count\_instance": 8,

... "id\_book": 7

... },

... "Tabel\_num": 6

... }

... );

>

>

WriteResult({ "nInserted" : 1 })

> db.book\_instance.insert(

... {

... "Libr\_id": 4,

... "Status\_book": true,

... "book": {

... "name": "Vishneviy sad",

... "date\_edition": new Date(1912, 8, 3, 6, 12),

... "authors": "Chehov",

... "count\_instance": 8,

... "id\_book": 7

... },

... "Tabel\_num": 4

... }

... );

>

>

WriteResult({ "nInserted" : 1 })

>

>

> db.createCollection("readers");

>

>

> db.readers.insert(

... {

... "number\_passport": 400052525,

... "number\_ticket": 1,

... "fio": "Dominic Mussalyamov",

... "address": "Olgino",

... "numper\_phone": 8963143244,

... "date\_birth": new Date(2001, 8, 8, 6, 12),

... "education": "Srednee",

... }

... );

WriteResult({ "nInserted" : 1 })

> db.readers.insert(

... {

... "number\_passport": 40963666,

... "number\_ticket": 2,

... "fio": "Kirill Romanov",

... "address": "Leninsliy",

... "numper\_phone": 89535763521,

... "date\_birth": new Date(2001, 03, 8, 6, 12),

... "education": "Srednee",

... }

... );

> db.get\_book.insert(

... {

... book\_instance: "5f85b8fc86e275a58572f61b",

... readers: "5f85be4f86e275a58572f61e",

... "date\_get": new Date('2018-12-26'),

... "date\_back": new Date('2019-01-30'),

... "sum": 100

... }

... );

> db.get\_book.insert(

... {

... book\_instance: ObjectId("5f85b8fc86e275a58572f61b"),

... readers: ObjectId("5f85b8fc86e275a58572f61b"),

... "date\_get": new Date('2018-11-05'),

... "date\_back": new Date('2018-12-30'),

... "sum": 30

... }

... )

> db.get\_book.insert(

... {

... book\_instance: "5f85b8fc86e275a58572f61b",

... readers: "5f85b8fc86e275a58572f61b",

... "date\_get": new Date('2018-11-05'),

... "date\_back": new Date('2018-12-30'),

... "sum": 30

... }

... );

WriteResult({ "nInserted" : 1 })

> db.readers.insert(

... {

... "number\_passport": 4083525,

... "number\_ticket": 3,

... "fio": "Ilay Karpenko",

... "address": "Petroga",

... "numper\_phone": 89535763556,

... "date\_birth": new Date(2001, 05, 12, 6, 12),

... "education": "Srednee",

... }

... );

@(shell):3:15

> db.readHall.insert(

... {

... "datatime\_open": new Date('2019-01-12 09:00:00'),

... "datatime\_open": new Date('2019-01-12 21:00:00')

... }

... )

WriteResult({ "nInserted" : 1 })

> db.readHall.insert(

... {

... "datatime\_open": new Date('2019-01-13 09:00:00'),

... "datatime\_open": new Date('2019-01-13 21:00:00')

... }

... );

WriteResult({ "nInserted" : 1 })

> db.readHall.insert(

... {

... "datatime\_open": new Date('2019-01-14 12:00:00'),

... "datatime\_open": new Date('2019-01-14 18:00:00')

... }

... );

WriteResult({ "nInserted" : 1 })

> db.createCollection("recording");

{ "ok" : 1 }

> db.recording.insert(

... {

... readers: ObjectId("5f85be0e86e275a58572f61d"),

... readHall: ObjectId("5f85e1d486e275a58572f625"),

... "datatime\_open": new Date('2019-01-14 12:00:00'),

... "datatime\_closed": new Date('2019-01-14 18:00:00')

... }

... );

WriteResult({ "nInserted" : 1 })

> db.recording.insert(

... {

... readers: ObjectId("5f85be4f86e275a58572f61e"),

... readHall: ObjectId("5f85e21086e275a58572f627"),

... "datatime\_open": new Date('2019-01-16 09:00:00'),

... "datatime\_closed": new Date('2019-01-18 18:00:00')

... }

... )

WriteResult({ "nInserted" : 1 })

> db.Librar.insert(

... {

... "address": "Petroga",

... readHall: ObjectId("5f85e21086e275a58572f627"),

... "datatime\_open": new Date('2019-01-16 09:00:00'),

... "datatime\_closed": new Date('2019-01-18 18:00:00'),

... "Librarian": {

... \_id: 1,

... "fio": "Alexandra Dadaria",

... "graphic\_work": "5/7",

... "numper\_phone": 9893475893,

... "number\_passport": 40023414,

... "date\_birth": new Date(1994-04-15),

... "education": "Srednee"

... }

... }

... )

WriteResult({ "nInserted" : 1 })

> db.Librar.insert(

... {

... "address": "Petroga",

... readHall: ObjectId("5f85e1f886e275a58572f626"),

... "datatime\_open": new Date('2019-01-17 12:00:00'),

... "datatime\_closed": new Date('2019-01-19 21:00:00'),

... "Librarian": {

... \_id: 2,

... "fio": "Chesnokova Irina",

... "graphic\_work": "3/7",

... "numper\_phone": 98934758743,

... "number\_passport": 400123324,

... "date\_birth": new Date(1992-10-03),

... "education": "Srednee"

... }

... }

... );

WriteResult({ "nInserted" : 1 })

> db.Librar.insert(

... {

... "address": "Olgino",

... readHall: ObjectId("5f85e1d486e275a58572f625"),

... "datatime\_open": new Date('2019-12-01 9:00:00'),

... "datatime\_closed": new Date('2020-01-19 21:00:00'),

... "Librarian": {

... \_id: 3,

... "fio": "Vera Brejneva",

... "graphic\_work": "1/7",

... "numper\_phone": 98938677743,

... "number\_passport": 405784324,

... "date\_birth": new Date(1967-04-15),

... "education": "Srednee"

... }

... }

... );

WriteResult({ "nInserted" : 1 })

> db.createCollection("registration");

{ "ok" : 1 }

> db.registration.insert(

... {

... Librar: ObjectId("5f85e75a86e275a58572f62b"),

... readers: ObjectId("5f85be0e86e275a58572f61d"),

... "datatime\_open": new Date('2019-12-01 9:00:00'),

... "datatime\_closed": new Date('2020-01-19 21:00:00')

... }

... );

WriteResult({ "nInserted" : 1 })

> db.registration.insert(

... {

... Librar: ObjectId("5f85e75a86e275a58572f62b"),

... readers: ObjectId("5f85bea886e275a58572f61f"),

... "datatime\_open": new Date('2020-01-16 9:00:00'),

... "datatime\_closed": new Date('2020-03-19 21:00:00')

... }

... );

WriteResult({ "nInserted" : 1 })

> db.registration.insert(

... {

... Librar: ObjectId("5f85e75a86e275a58572f62b"),

... readers: ObjectId("5f85be4f86e275a58572f61e"),

... "datatime\_open": new Date('2020-01-09 9:00:00'),

... "datatime\_closed": new Date('2020-02-30 21:00:00')

... }

... );

WriteResult({ "nInserted" : 1 })

> db.registration.insert(

... {

... Librar: ObjectId("5f85e7f786e275a58572f62c"),

... readers: ObjectId("5f85be4f86e275a58572f61e"),

... "datatime\_open": new Date('2020-02-01 9:00:00'),

... "datatime\_closed": new Date('2020-02-30 21:00:00')

... }

... );

WriteResult({ "nInserted" : 1 })

> db.registration.insert(

... {

... Librar: ObjectId("5f85e7f786e275a58572f62c"),

... readers: ObjectId("5f85be0e86e275a58572f61d"),

... "datatime\_open": new Date('2020-04-14 9:00:00'),

... "datatime\_closed": new Date('2020-05-01 21:00:00')

... }

... );

WriteResult({ "nInserted" : 1 })

////////////////////////////////////////////////////////////////////////////////

................................................................................

................................................................................

////////////////////////////////////////////////////////////////////////////////

Запросы:

>

> db.get\_book.find({sum: {$gt: 30, $lt: 200}}).sort({$natural: -1});

> db.book\_instance.find({"book.count\_instance": {$gt: 3}}, {\_id: 0}).limit(2)

> db.Librar.find({"Librarian.education": {$exists: ["Srednee", "Vishee"]}}).sort({"Librarian.\_id": 1});

> db.registration.find({readers: ObjectId("5f85be4f86e275a58572f61e")}).limit(2);

**Запросы к модели базы данных в MongoDB:**

Вывод записей, у которых читатели выплатили сумму штрафа не более чем 200р и не менее, чем 30 упорядоченное по убыванию:

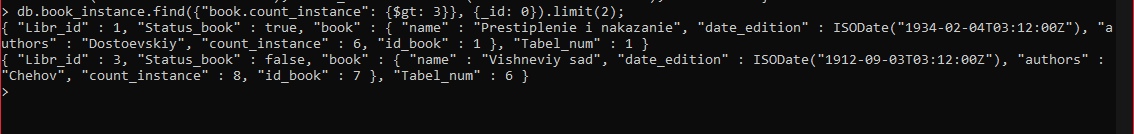
> db.get\_book.find({sum: {$gt: 30, $lt: 200}}).sort({$natural: -1});

Изображение выглядит как текст

Автоматически созданное описание

Вывод информации о книгах, у которых количество экземпляров больше 3. С исключением идентификатора и ограничением 2 записями:

> db.Librar.find({"Librarian.education": {$exists: ["Srednee", "Vishee"]}}).sort({"Librarian.\_id": 1});



Вывод информации о библиотеках, у работников, которых образование либо Среднее, либо Высшее, отсортированное по возрастанию идентификатора работников библиотеки

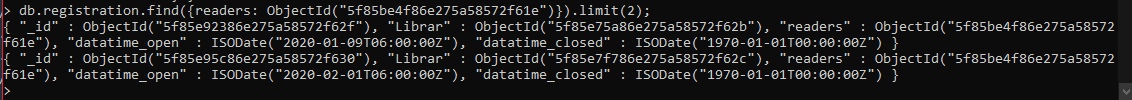
> db.Librar.find({"Librarian.education": {$exists: ["Srednee", "Vishee"]}}).sort({"Librarian.\_id": 1});

Изображение выглядит как текст

Автоматически созданное описание

Вывод информации о проделанных записях читателя с уникальным идентификатором, ограниченными 2 записями

> db.registration.find({readers: ObjectId("5f85be4f86e275a58572f61e")}).limit(2);



> db.readers.aggregate(

... [

... {$lookup:{

... from: "get\_book",

... localField: "\_id",

... foreignField: "readers",

... as:"get\_book"

... }}

... ]

... ).pretty();

Изображение выглядит как текст

Автоматически созданное описание

> db.get\_book.aggregate([

{$group: {\_id:"get\_books", total\_sum:{$sum: "$sum"}}},

{$match: {sum: {$gt: 30, $lt: 200}}}

]

);

**Вывод:** в ходе выполнения лабораторной работы были освоены практические навыки реализации баз данных в MongoDB.