ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Нереляционные базы данных (NoSQL)

Магистратура, прикладная математика и информатика, профиль: машинное обучение на текстах и графах
Магистратура, прикладная информатика (заочное отделение)

Пальчевский Евгений Владимирович
Старший преподаватель департамента анализа данных и машинного обучения

О курсе

	Количество лекций	Количество семинаров	Количество задач, в том числе и домашних	Количество контрольных работ	Зачет или экзамен
Модуль 6	5	14	4	2	Зачет

За всё вы получаете баллы в соответствии с балльно-рейтинговой системой (БРС)

https://palchevsky.ru/materials.php - учебные материалы и рейтинги

Балльно-рейтинговая система (БРС)

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Модуль 2	Посещение семинарских занятий	Баллы за задачи	Баллы за контрольную работу	Зачет или экзамен
	0,43	5	10,50	Зачет
Максимум	6	20	21	60
Как считалось?	6/14, где 14 – количество семинарских занятий	4 задачи * 5	21/2, где 2 – количество к/р (задач)	Набираете 40 или больше баллов за работу в семестре, получаете автомат (100 баллов). Зачтено идет от 50 и выше

В сумме за работу в семестре можно набрать 47 баллов.
Всего 6 задач: 4 – обычных, 2 – контрольных

Баллы за задачи

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Модуль	Вид задачи	Баллы
2	Сложный вариант	5

Дедлайн по задачам

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Модуль	# темы	Дедлайн	Группы
2	Все	15.01.2024	МО23-1м
2	Все	20.12.2023	ДПИ23-1м

Онлайн-консультации для сдачи долгов

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

#	Дата	Группы	Время
1	18.09.2023	Все группы	20:00-22:00
2	25.09.2023		20:00-22:00
3	02.10.2023		20:00-22:00
4	16.10.2023		20:00-22:00
5	23.10.2023		20:00-22:00
6	30.10.2023		20:00-22:00
7	15.11.2023		20:00-22:00
8	29.11.2023		20:00-22:00
9	06.12.2023		20:00-22:00
10	13.12.2023		20:00-22:00
11	20.12.2023		20:00-22:00
12	27.12.2023		20:00-22:00

Ссылки на консультации, а также актуальный рейтинг можно найти тут:
https://docs.google.com/spreadsheets/d/114tKSw7EcygY9BcS-gqnnbcC2-vTQmmAzOTzwRMKI7M/

Контакты с преподавателем

Социальная сеть / мессенджер / почта	Контакт
Электронная почта	teelxp@inbox.ru, evpalchevskij@fa.ru
VK	https://vk.com/teelxp
WhatsApp	+7-937-485-80-48

https://palchevsky.ru/materials.php - учебные материалы и рейтинги

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Материал на осенний семестр 2023/24 учебного года
Модуль 2

Темы лекционных и семинарских занятий

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

1. Нереляционные базы данных (1 лекция, 0 семинарских занятия).
2. Документно-ориентированные базы данных (1 лекция, 5 семинарских занятий).
3. Графовые базы данных (1 лекция, 5 семинарских занятия).
4. Распределенные базы данных (2 лекции, 5 семинарских занятия).

Курсы для дополнительного освоения материала

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

1. **“Введение в базы данных”**. Базовый курс по SQL и NoSQL.
Ссылка: https://stepik.org/course/125212/promo

2. **“Программирование на Python”**. Базовый курс.
Ссылка: https://stepik.org/course/67/promo

3. **“Разработка бэкенд-приложений на Django”**. Продвинутый курс.
Ссылка: https://stepik.org/course/155057/promo

4. **“Интерактивный тренажер по SQL”**. Базовый курс.
Ссылка: https://stepik.org/course/63054/promo

Рекомендации к решению задач

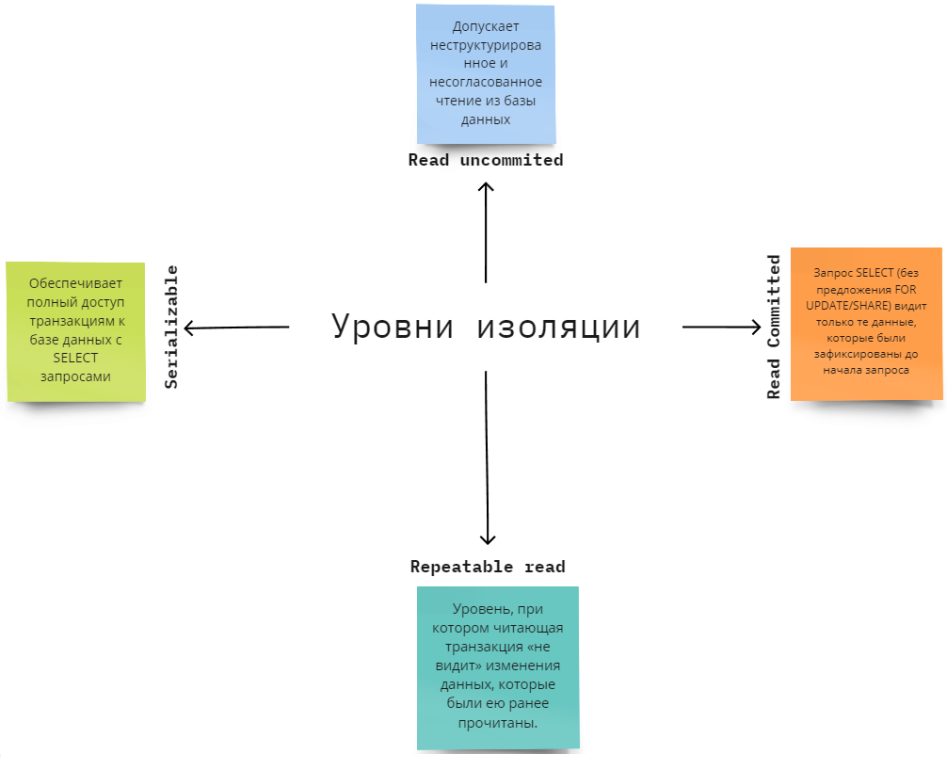
ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Видеолекции, которые помогут решить задачи и получить интересные навыки разработки находятся в плейлисте:
https://www.youtube.com/watch?v=QwraZWO62PY&list=PLNSAyqUuk6sS2Ea2UCBwpm6qWL0rdH5TS

Лекция №1: Нереляционные базы данных. SQL vs NoSQL

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Особенность (свойства, функциональность)	Реляционные базы данных (SQL)	Нереляционные базы данных (NoSQL)
Подходящие рабочие нагрузки	Предназначены для ***OLTP*** и ***OLAP***.	Такие базы данных работают по создаваемым шаблонам. Можем создавать собственные структуры данных.
Модель данных	Табличная	Ключ-значение, документы и графы.
Свойства ACID	SQL СУБД обеспечивают полный набор требований к транзакционной системе, обеспечивающий наиболее надёжную и предсказуемую её работу, иначе говоря ACID.	Базы данных NoSQL зачастую предлагают компромисс, смягчая жесткие требования свойств ACID ради более гибкой модели данных, которая допускает горизонтальное масштабирование.
Производительность	Зависит от дисковой подсистемы. Максимальная производительность требует определенной оптимизации структуры таблицы, запросов и индексов.	Зависит от: 1. Аппаратного обеспечения и пропускной способности сети сервера/кластера. 2. Задержки сети приложения, с которым работает NoSQL база данных.
Масштабирование	Масштабируются путем увеличения вычислительных возможностей аппаратного обеспечения или добавления отдельных копий для рабочих нагрузок чтения.	Базы данных NoSQL обычно поддерживают высокую разделяемость благодаря шаблонам доступа с возможностью масштабирования на основе распределенной архитектуры.
API	Запросы на запись и извлечение данных составляются на языке SQL. Эти запросы анализирует и выполняет реляционная база данных.	Объектно-ориентированные API позволяют разработчикам приложений без труда осуществлять запись и извлечение структур данных.



Лекция №1: Нереляционные базы данных. NoSQL

SQL	NoSQL
База данных	База данных
Таблица	Коллекция
Ряд	Документ
Столбец	Поле
Первичный ключ	ObjectID
Индекс	Индекс
Представление	Представление
Вложенная таблица	Встроенный документ
Массив	Массив

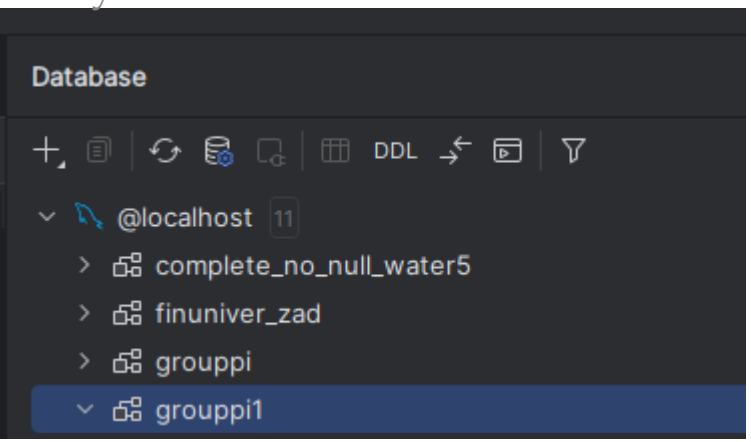
Лекция №1: Нереляционные базы данных. Синтаксис SQL (для понимания и повтора)

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

1. Создание базы данных

```
CREATE DATABASE
GroupPI1;
```

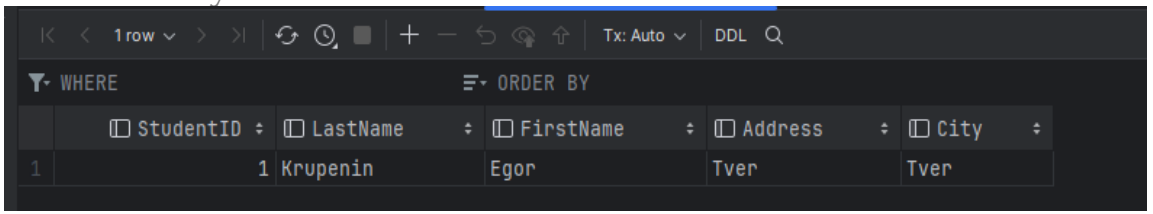
Результат:



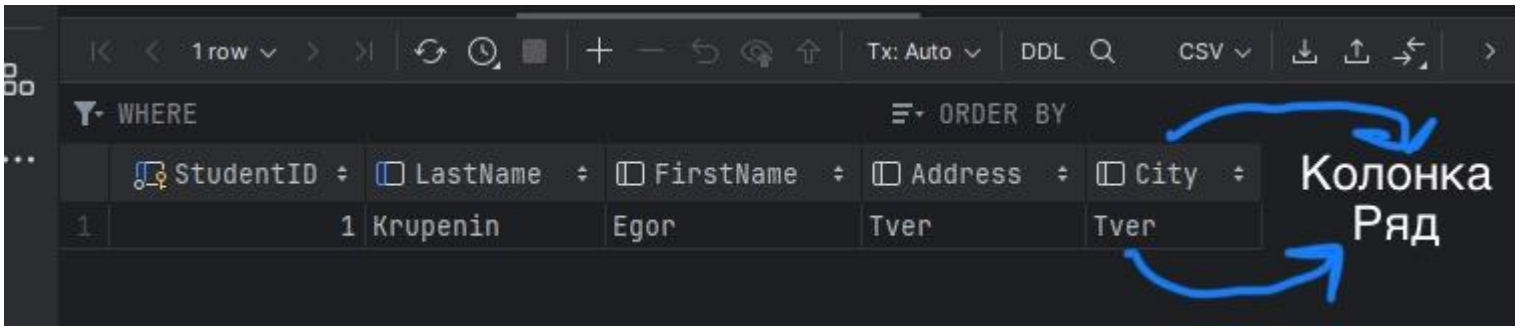
2. Создание таблицы

```
use GroupPI1;
CREATE TABLE Students (
    StudentID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
INSERT INTO Students (StudentID, LastName, FirstName, Address, City)
VALUES ('1', 'Krupenin', 'Egor', 'Tver', 'Tver')
```

Результат:



3. Ряд и столбец



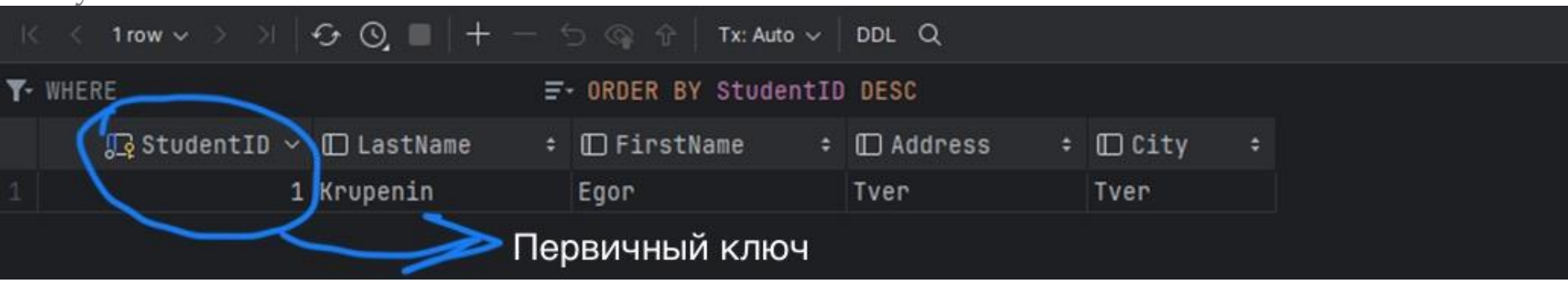
Лекция №1: Нереляционные базы данных. Синтаксис SQL (для понимания и повтора)

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

4. Первичный ключ

```
ALTER TABLE grouppi1.students
ADD PRIMARY KEY (StudentID)
```

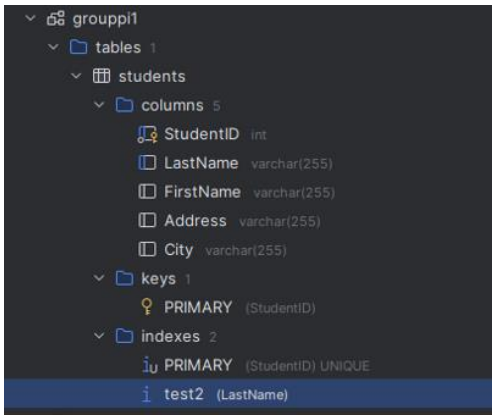
Результат:



5. Индексы

```
CREATE INDEX test2
ON grouppi1.students (LastName);
```

Результат:



Лекция №1: Нереляционные базы данных. Синтаксис SQL (для понимания и повтора)

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

6. Представление

6.1. Создаем еще одну таблицу в БД:

```
use GroupPI1;
CREATE TABLE Students1 (
    StudentID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
INSERT INTO Students1 (StudentID, LastName, FirstName, Address, City)
VALUES ('1', 'Titov', 'Semen', 'Moscow', 'Moscow')
```

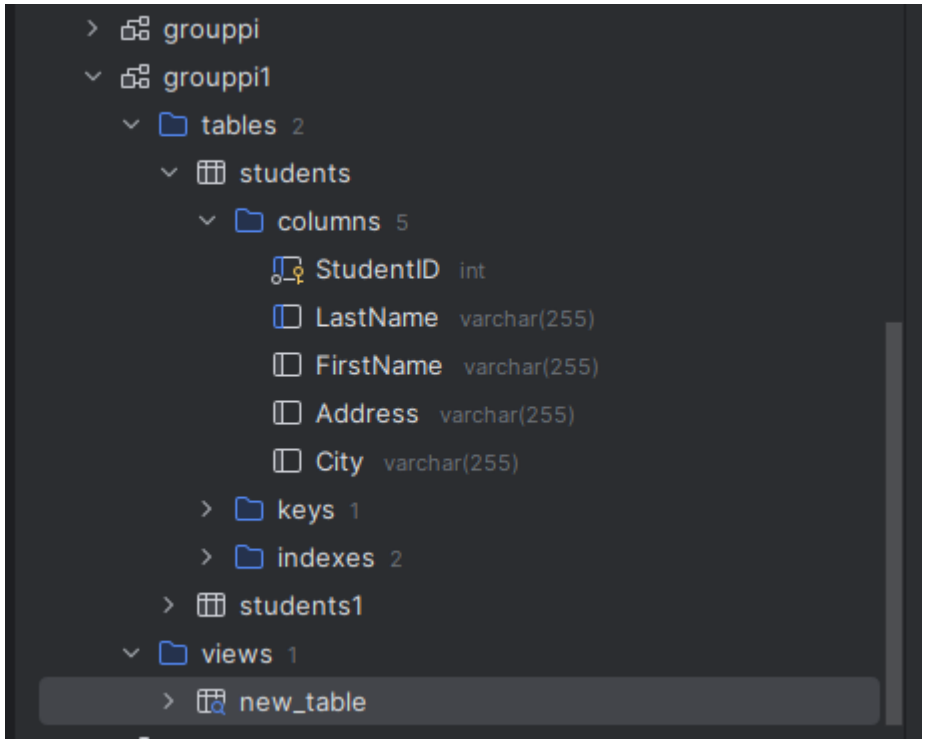
6.2. Присваиваем первичный ключ:

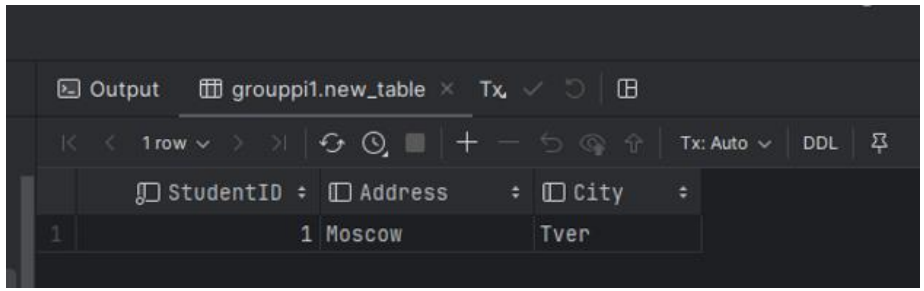
```
ALTER TABLE grouppi1.students
ADD PRIMARY KEY (StudentID)
```

6.3. Создаем представление:

```
CREATE VIEW new_table
AS SELECT grouppi1.students.StudentID, grouppi1.students1.Address,
grouppi1.students.City
FROM students, students1
WHERE students.StudentID = students1.StudentID
```

6.4. Результат:





7. Вложенные таблицы (Nested Table)

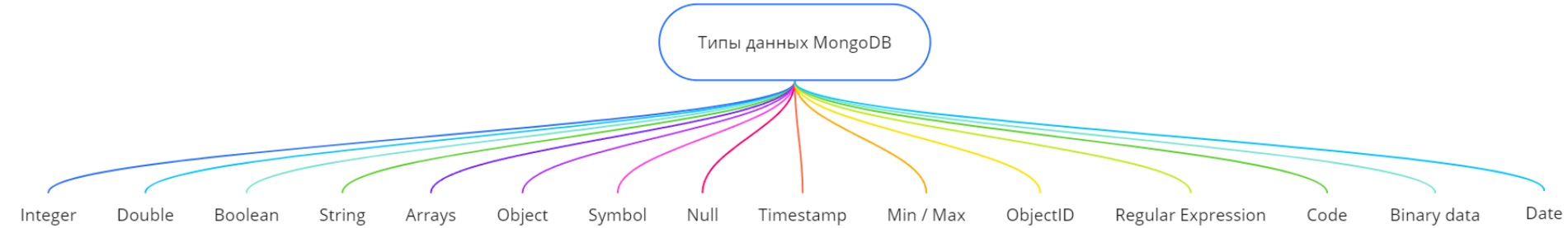
Вложенную таблицу можно рассматривать как одномерный массив, в котором индексами служат значения целочисленного типа. Грубо говоря, небольшая таблица в виде коллекции внутри таблицы. Вложенная таблица может иметь пустые элементы, которые появляются после их удаления встроенной процедурой DELETE. Количество элементов может динамически увеличиваться. Максимальный размер – 2 ГБ. Очень удобно использовать при работе с большими однородными или темпоральными однородными данными. Однородные данные – это данные одного типа, а темпоральные – данные, привязанные к дате и времени.

8. Массивы

```
CREATE TABLE group2 (
    FirstName       text,
    LastName        text[],
    GroupHistory    text[][]
);
```

Лекция №1: Нереляционные базы данных. Типы данных и синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ



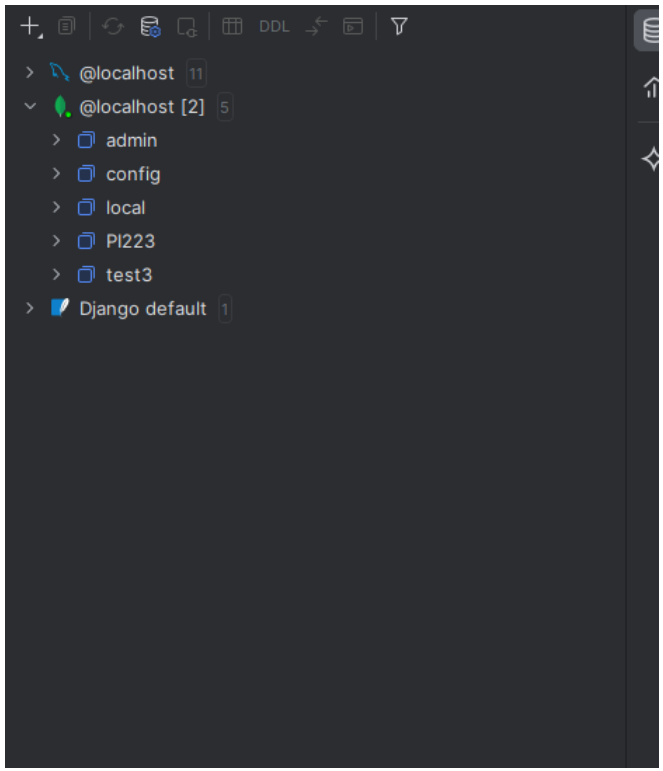
1. Создание базы данных

```
use PI223
```

1.1. Чтобы база отображалась в списке БД

```
db.PI2231.insertOne(
    {
        "StudentName" : "Egor",
        "StudentLastName" : "Krupenin"
    }
)
```

Результат:



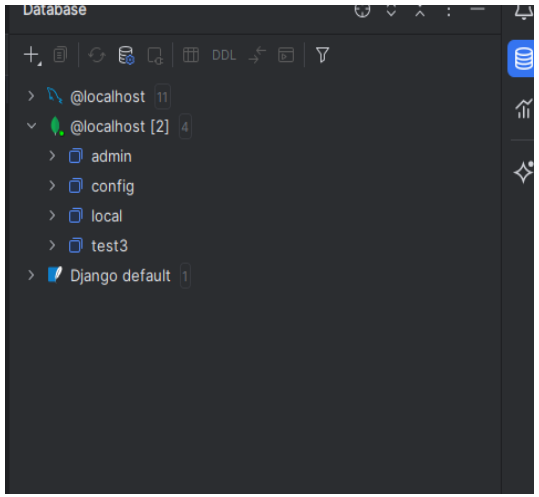
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

2. Удаление базы данных

```
use PI223
db.dropDatabase()
```

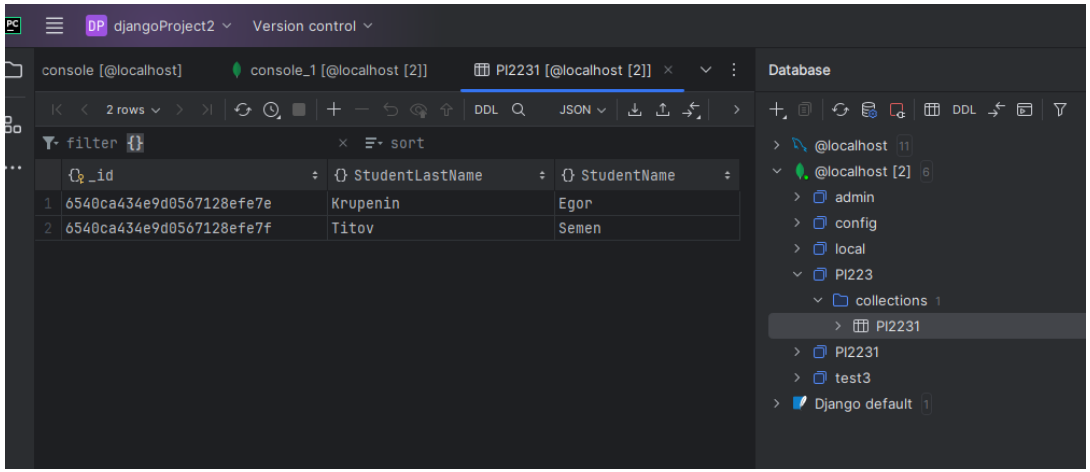
Результат:



3. Добавление нескольких элементов в коллекцию

```
use PI223
db.PI2231.insertMany([
  {

    "StudentName": "Egor",

    "StudentLastName": "Krupenin"

  },
  {

    "StudentName": "Semen",

    "StudentLastName": "Titov"

  }]
  )
```

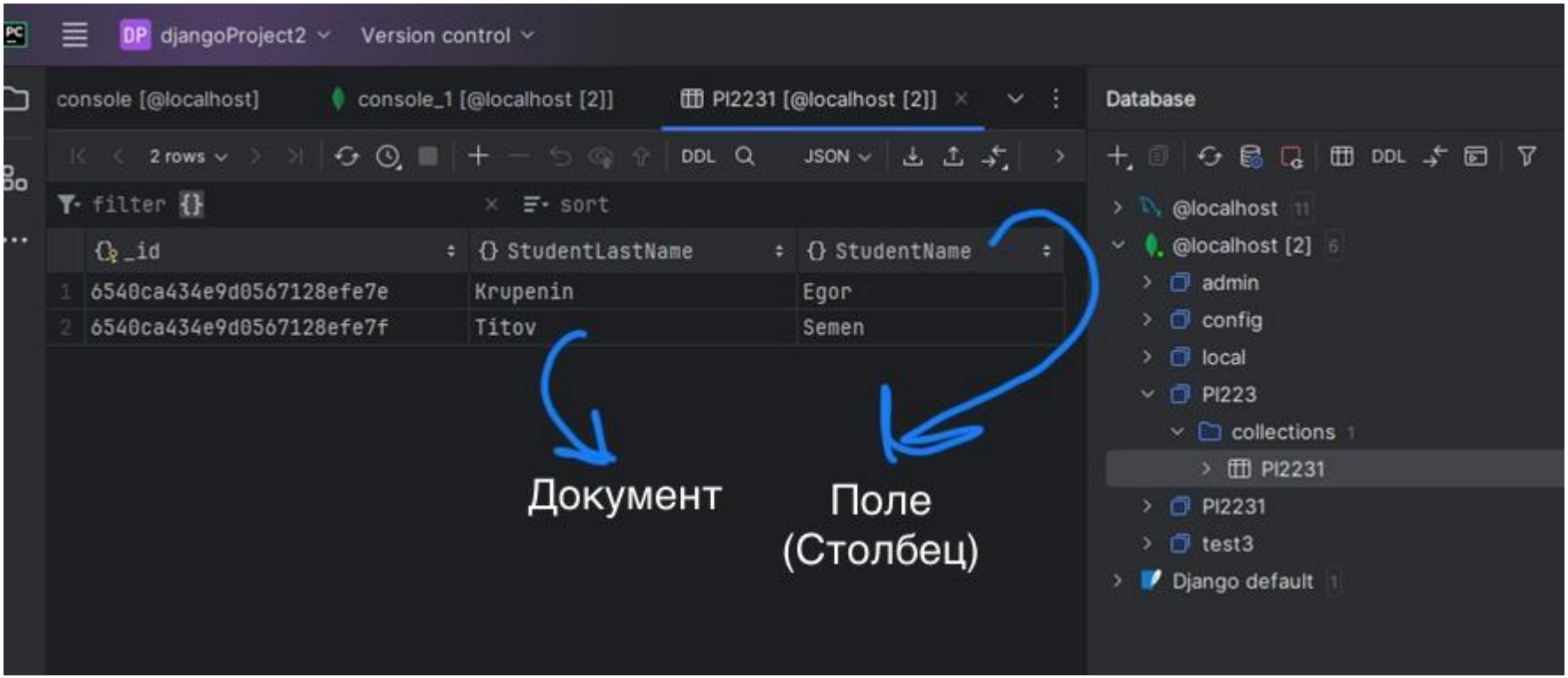
Результат:



Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

4. Поле и документ



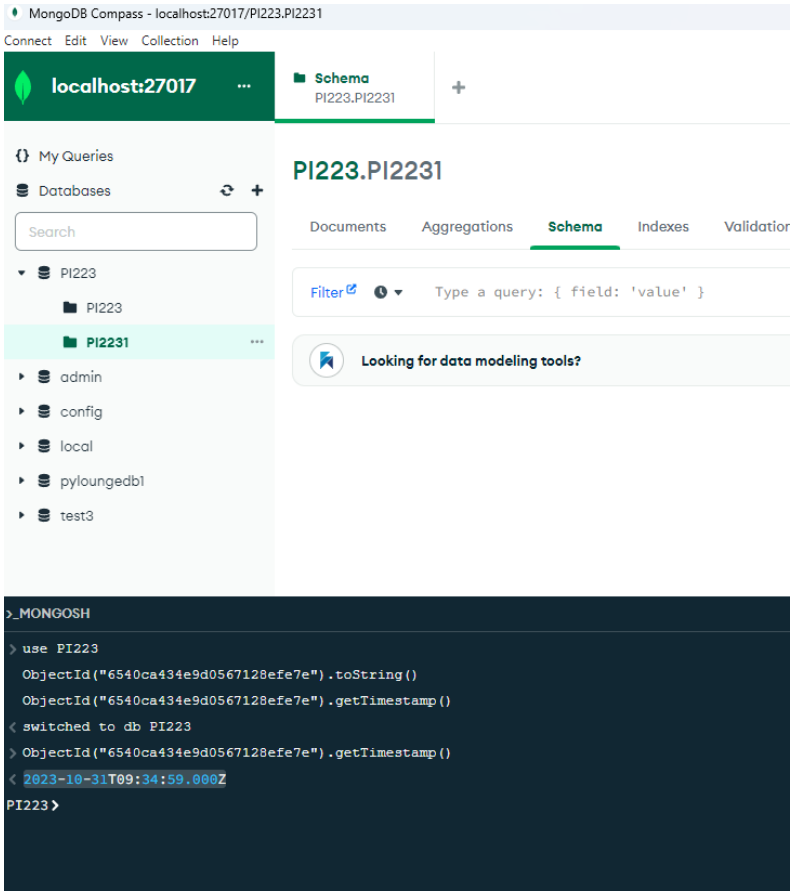
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

5. ObjectID

```
ObjectId("6540ca434e9d0567128efe7e").getTimestamp()
```

Результат (в PyCharm данная функция не работает):



Перевод ObjectID в строку:

```
use PI223
ObjectId("6540ca434e9d0567128efe7e").toString()
```

Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

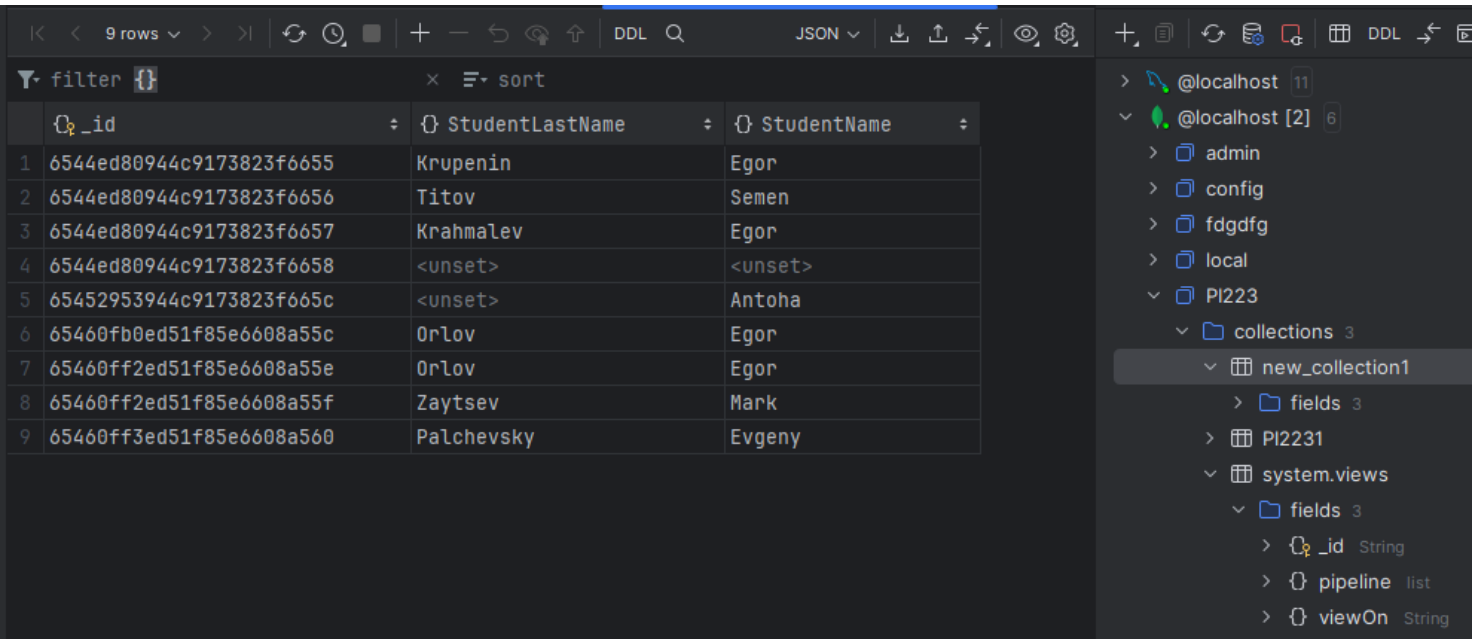
ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

6. Индексы

```
use PI223
db.PI2231.createIndex({"StudentName" : 1})
db.PI2231.getIndexes()
db.PI2231.find({StudentName: "Egor"})
```

7. Представление

```
use PI223
db.createView(
  "new_collection1",
  "PI2231",
  [ { $project: { "StudentLastName": "$StudentLastName", "StudentName": "$StudentName"} } ]
)
```



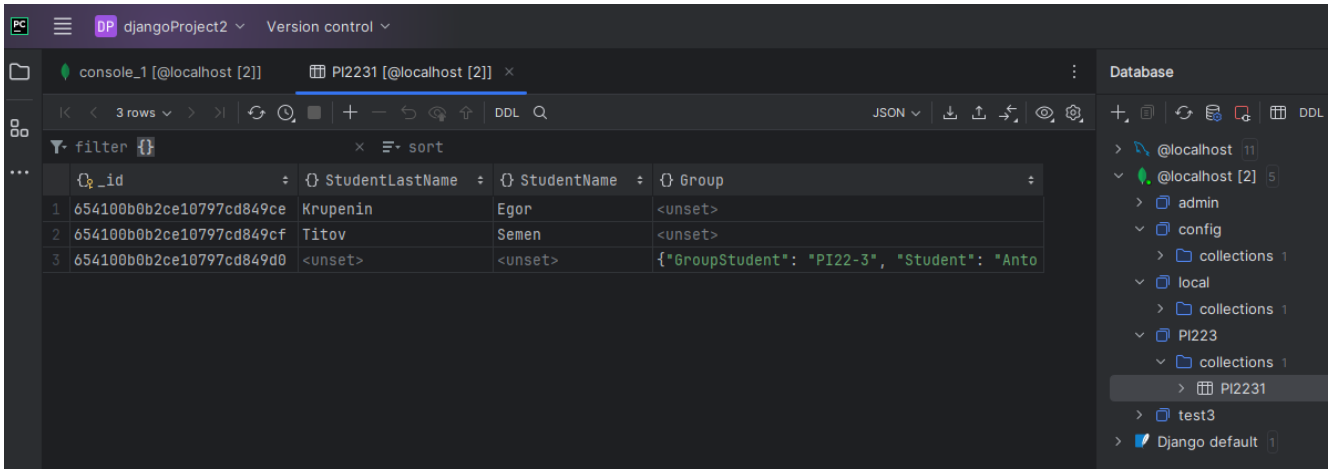
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

8. Встроенный документ

```
db.PI2231.insertMany([

  {

    "StudentName": "Egor",

    "StudentLastName": "Krupenin"

  },
  {

    "StudentName": "Semen",

    "StudentLastName": "Titov"

  },
  {
    Group: {"GroupStudent": "PI22-3", "Student": "Antonov Alexander"}
  }]
  )
```

Результат:

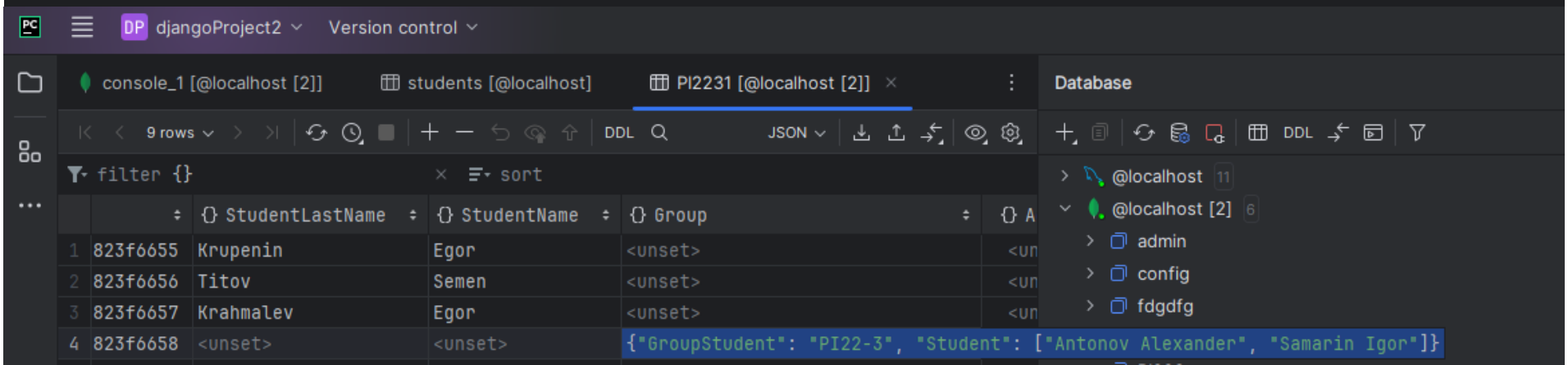


Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

9. Массивы

```
use PI223
db.PI2231.insertMany([

  {

    "StudentName": "Egor",

    "StudentLastName": "Krupenin"

  },
  {

    "StudentName": "Semen",

    "StudentLastName": "Titov"

  },
  {
    Group: {"GroupStudent": "PI22-3", "Student": ["Antonov Alexander", "Samarin Igor"]}
  }]
  )
```



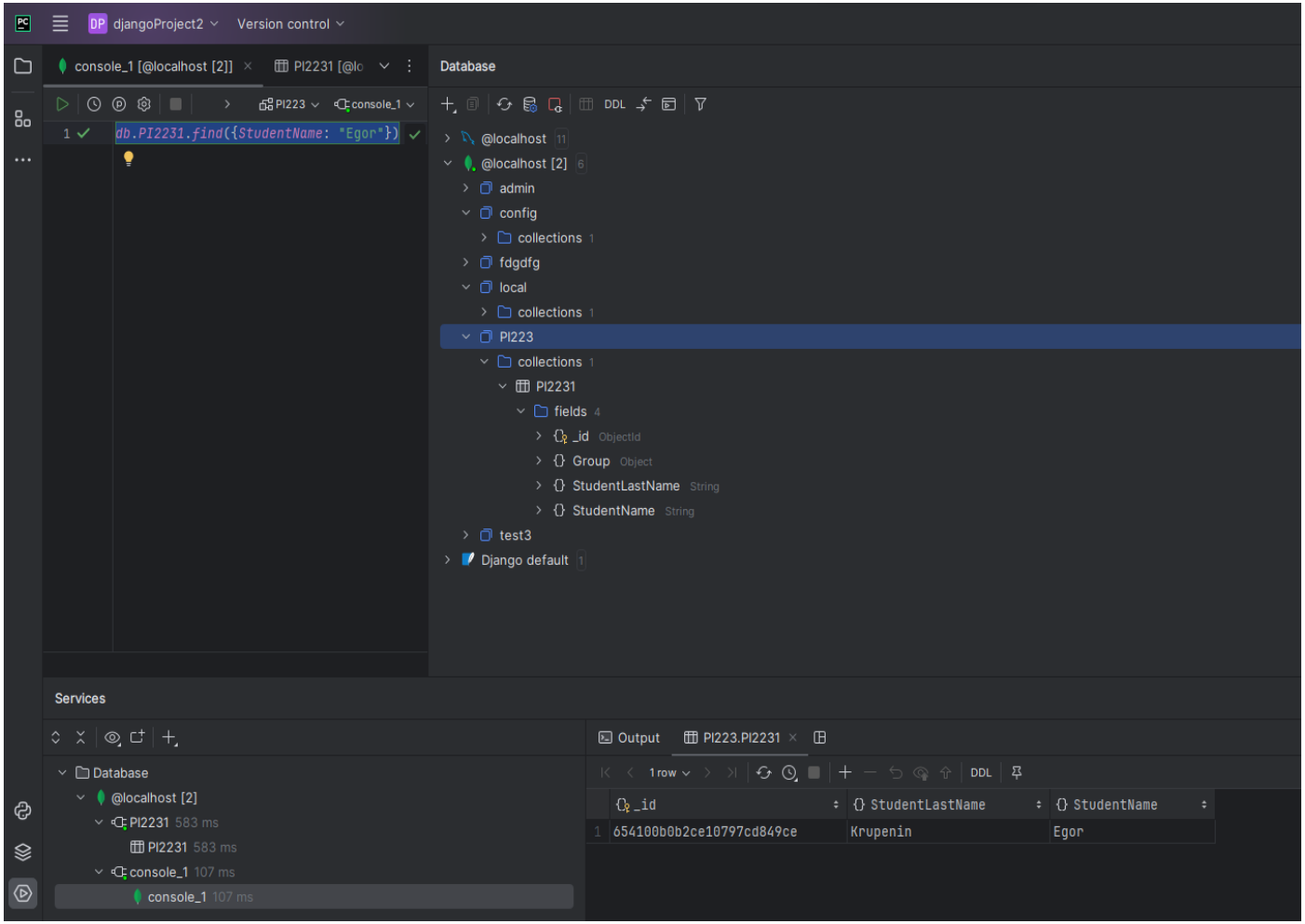
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

10. Фильтрация

```
db.PI2231.find({StudentName: "Egor"})
```

Результат:



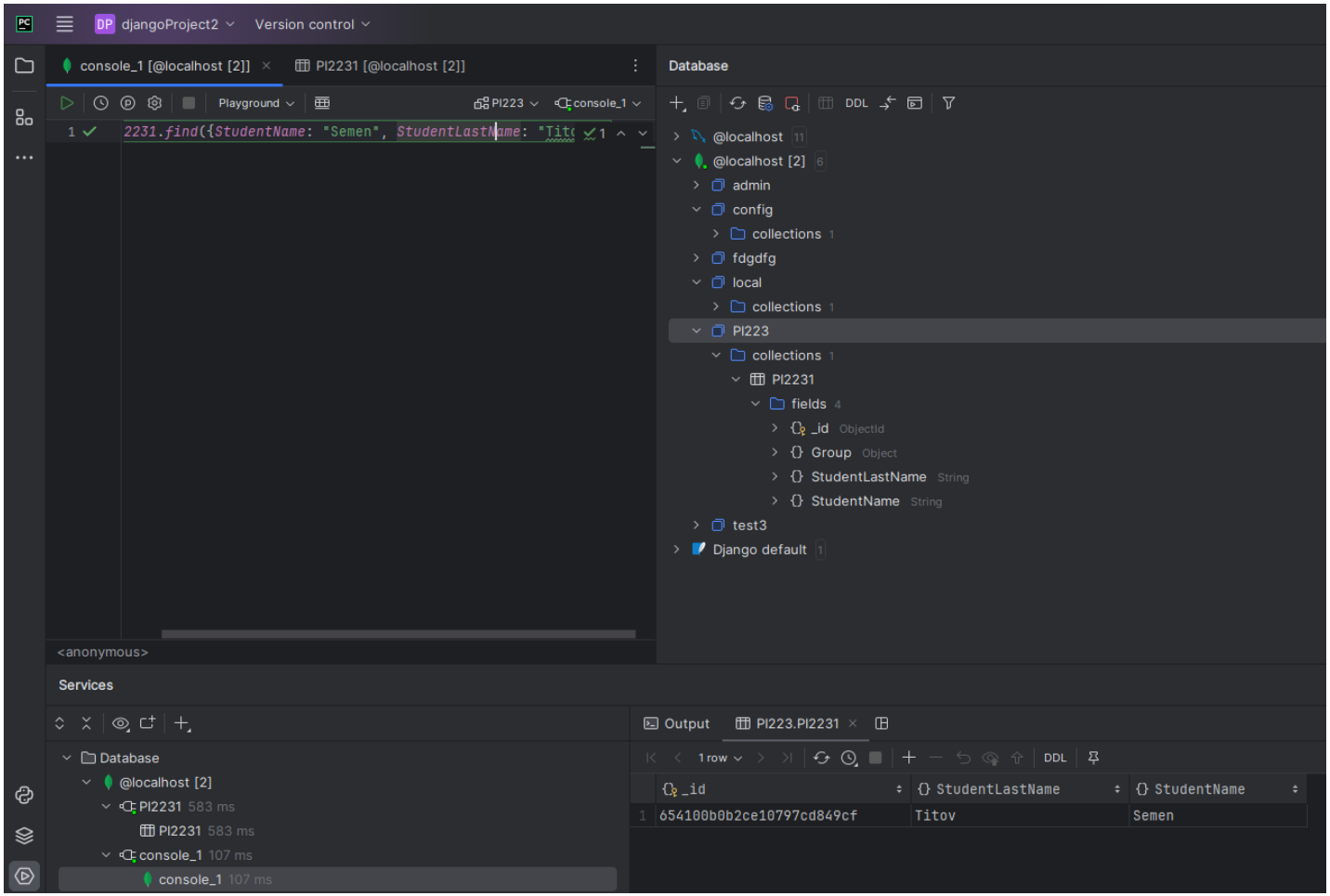
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

11. Фильтрация по нескольким значениям

```
db.PI2231.find({StudentName: "Semen", StudentLastName: "Titov"})
```

Результат:



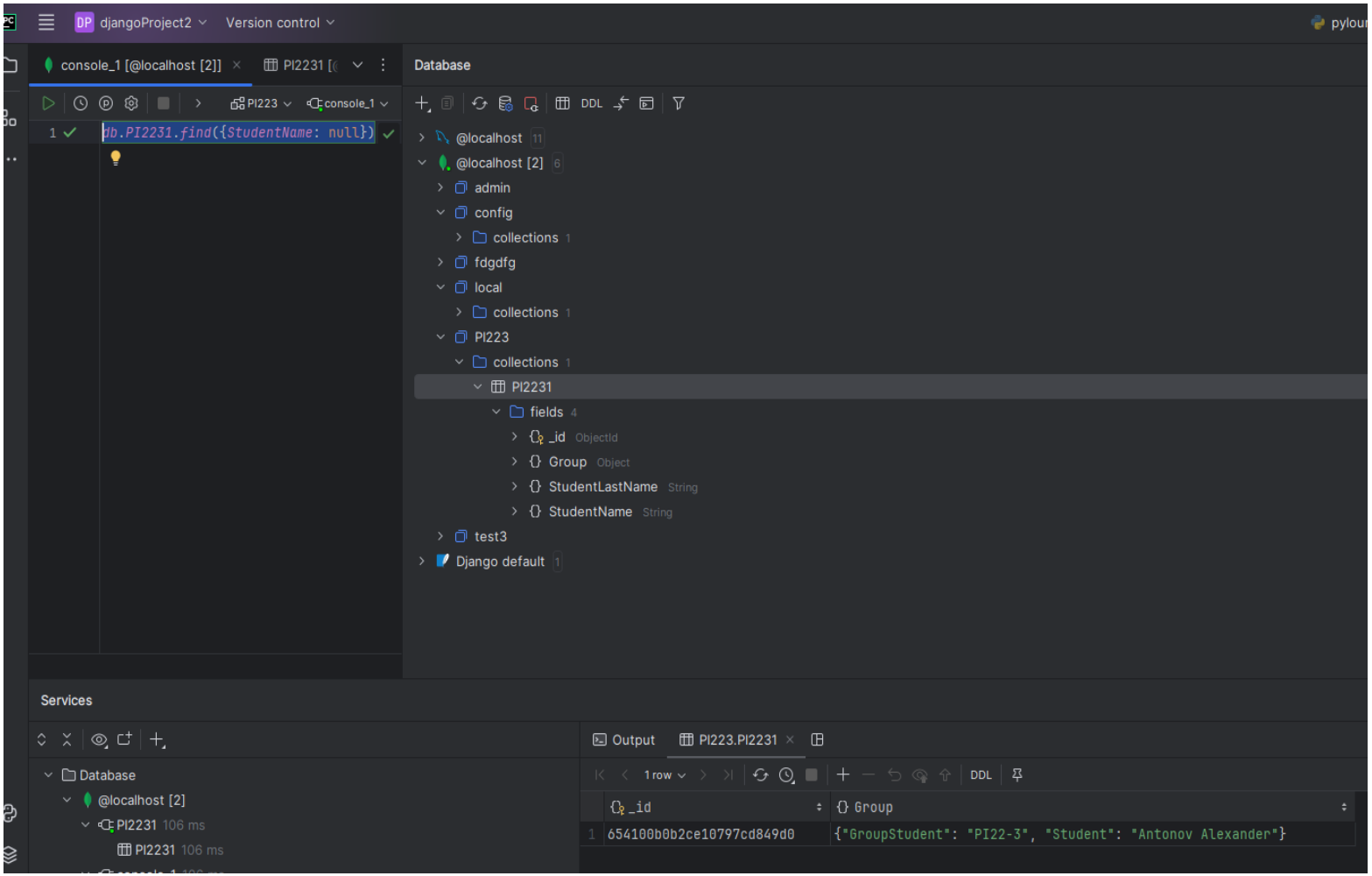
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

12. Фильтрация по отсутствующим значениям

```
db.PI2231.find({StudentName: null})
```

Результат:



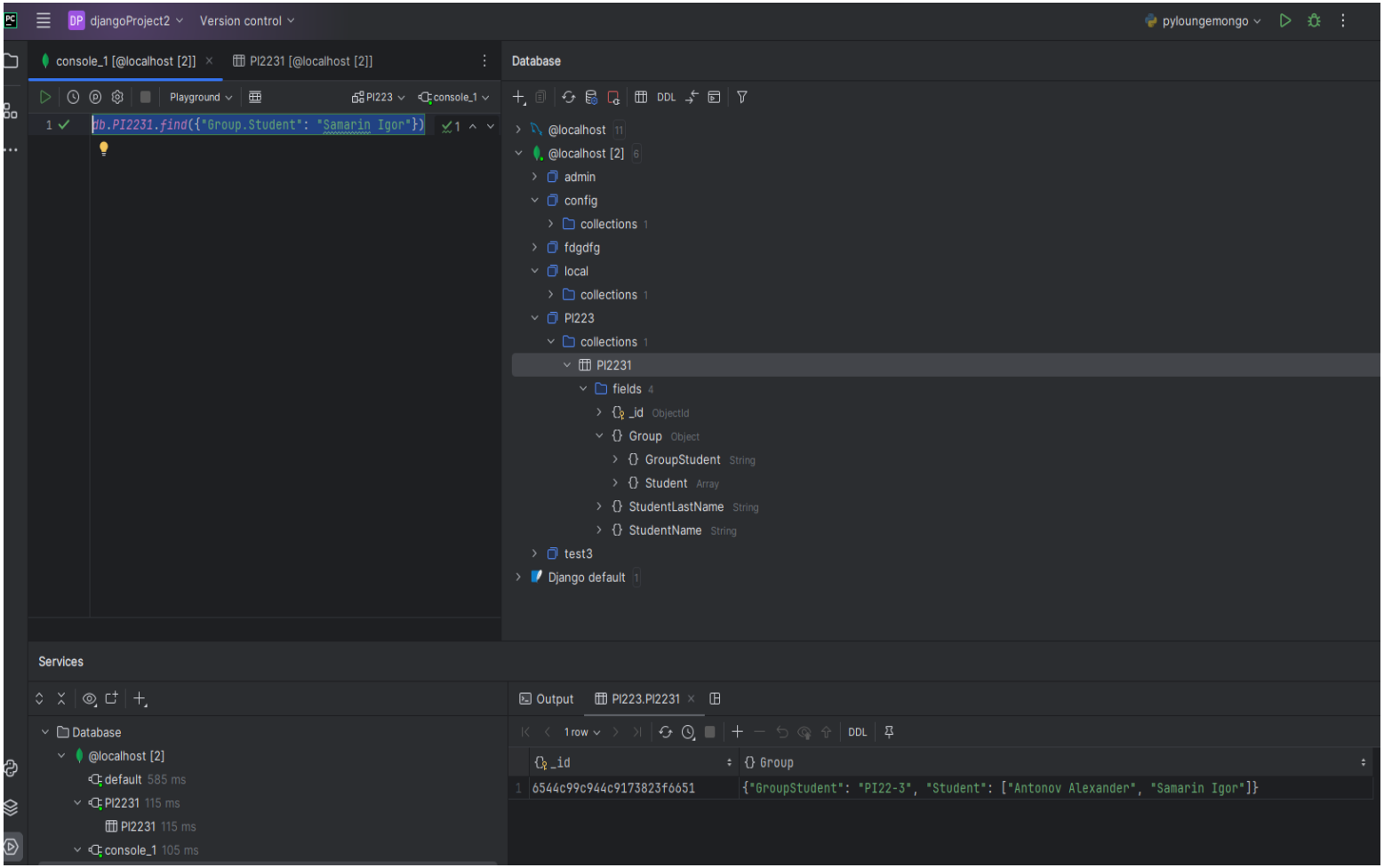
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

13. Фильтрация по элементам массива или вложенного объекта

```
db.PI2231.find({"Group.Student": "Samarin Igor"})
```

Результат:



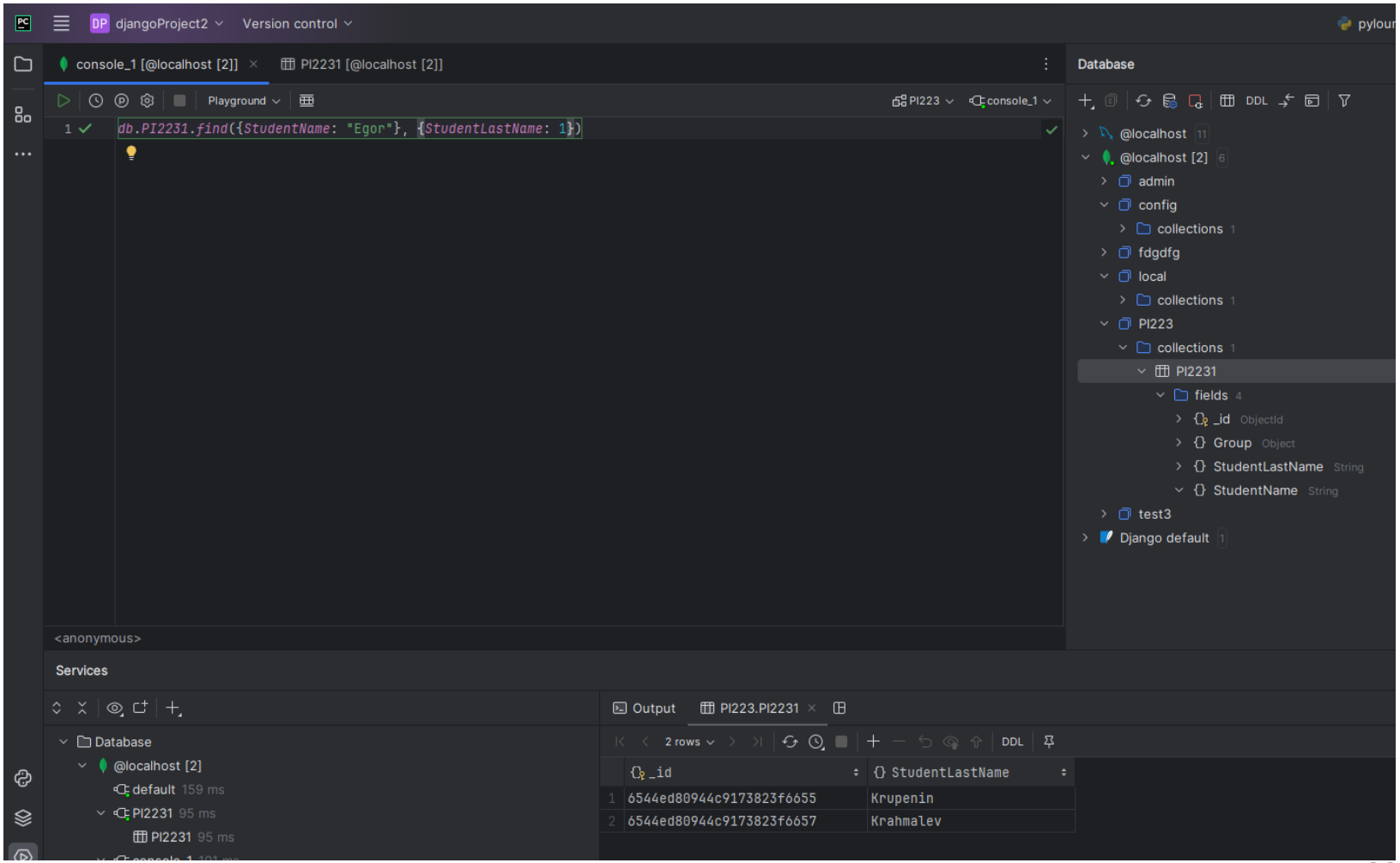
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

14. Проекция

```
db.PI2231.find({StudentName: "Egor"}, {StudentLastName: 1})
```

Результат:



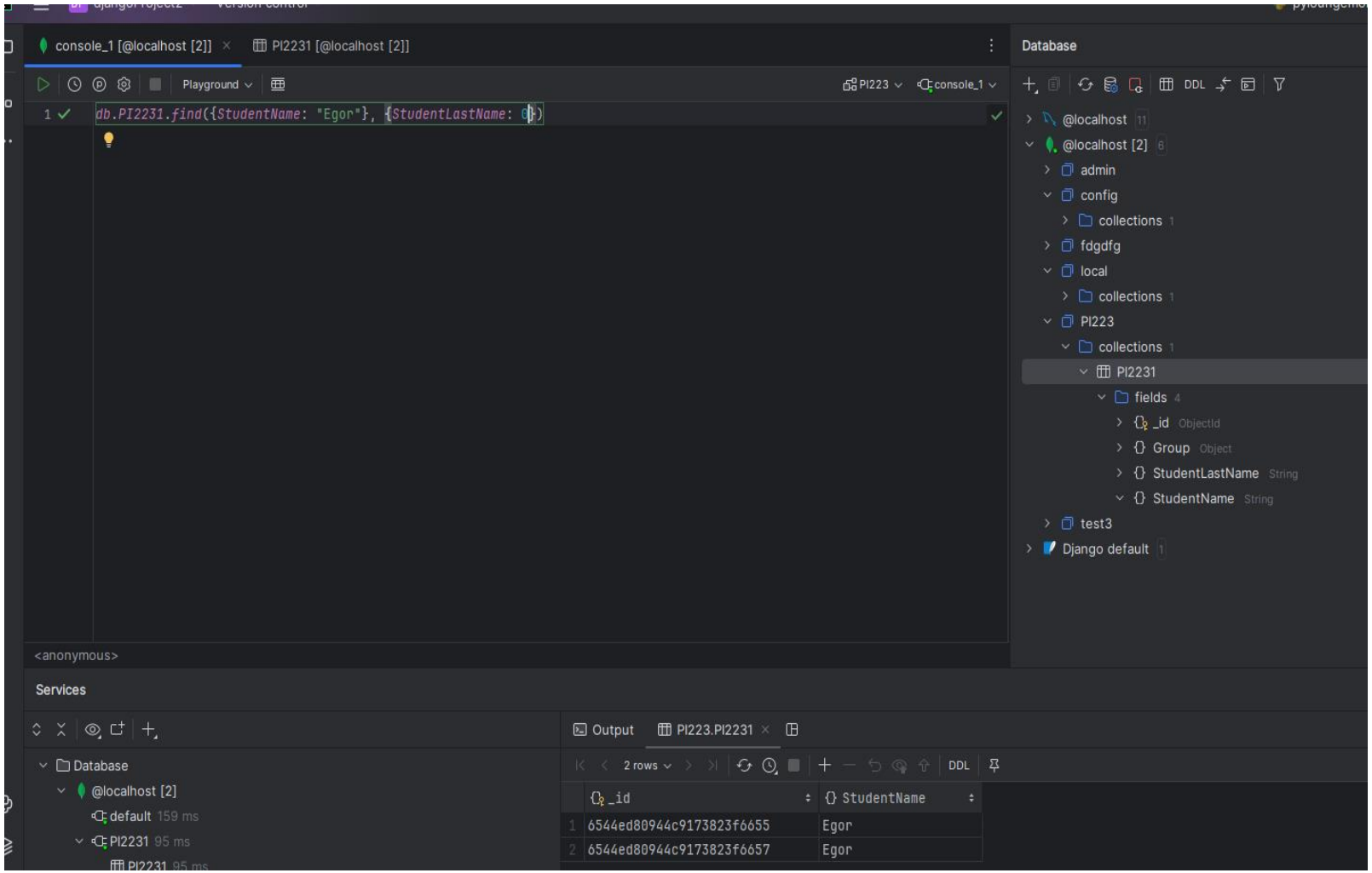
Лекция №1: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

14.1. Проекция с заменой на 0

```
db.PI2231.find({StudentName: "Egor"}, {StudentLastName: 0})
```

Результат:



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

15. Поиск одиночного документа

```
db.PI2231.findOne({StudentName: "Semen"})
```

Результат:

```
> db.PI2231.findOne({StudentName: "Semen"})
< {
    _id: ObjectId("6544ed80944c9173823f6656"),
    StudentName: 'Semen',
    StudentLastName: 'Titov'
  }
Enterprise PI223>
```

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

16. Курсоры через while

```
var cursor = db.PI2231.find()

while(cursor.hasNext()){
  obj = cursor.next();
  print(obj["StudentName"]);
}
```

Результат:

```
> var cursor = db.PI2231.find()

  while(cursor.hasNext()){
    obj = cursor.next();
    print(obj["StudentName"]);
  }
< Egor
< Semen
< Egor
```

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

16.1. Курсоры через for

```
var cursor = db.PI2231.find()
cursor.forEach(function(obj){
  print(obj.StudentName);
})
```

Результат:

```
> var cursor = db.PI2231.find()
  cursor.forEach(function(obj){
    print(obj.StudentName);
  })
Egor
Semen
Egor
```

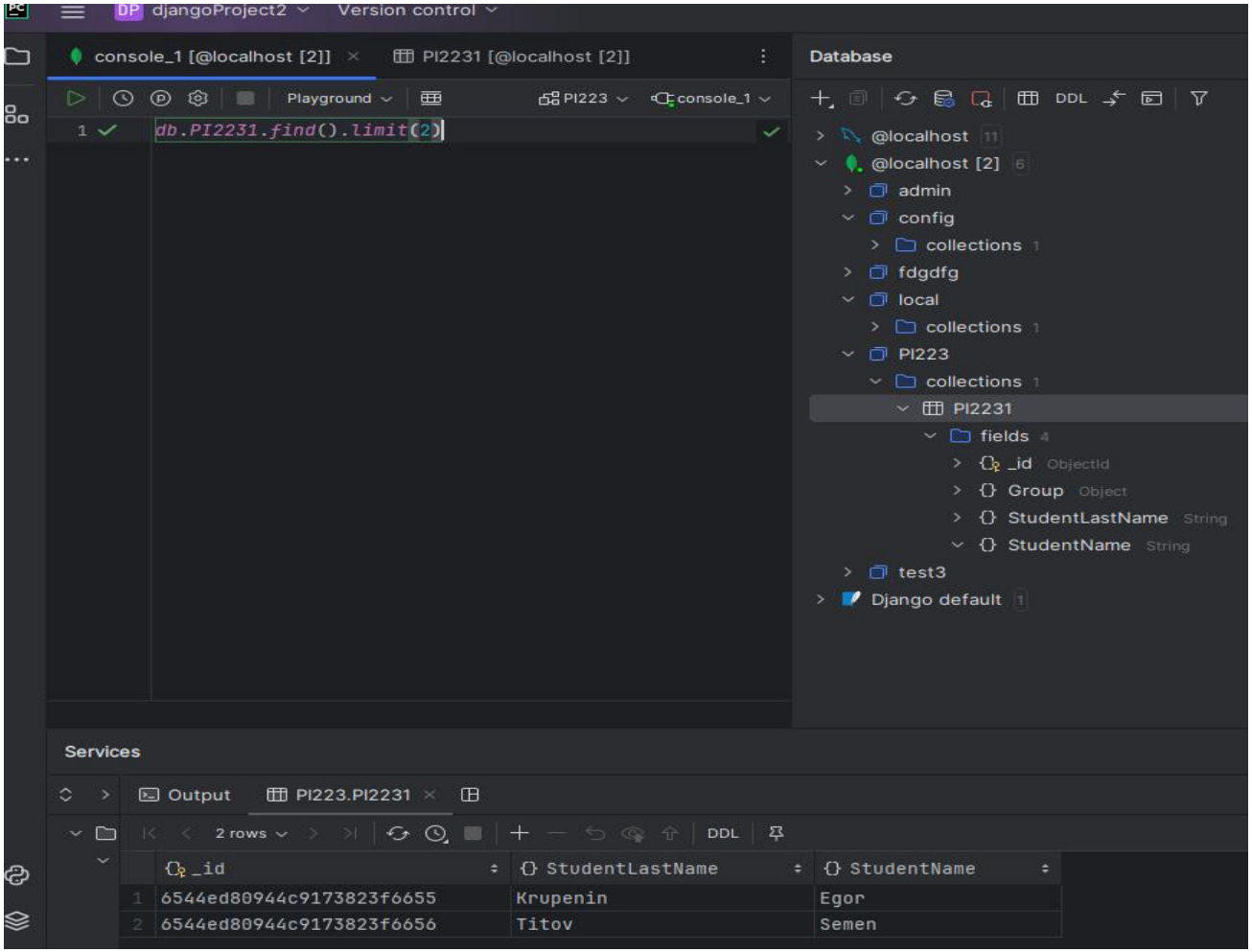
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

17. Пагинация и сортировка. Ограничение вывода

```
db.PI2231.find().limit(2)
```

Результат:



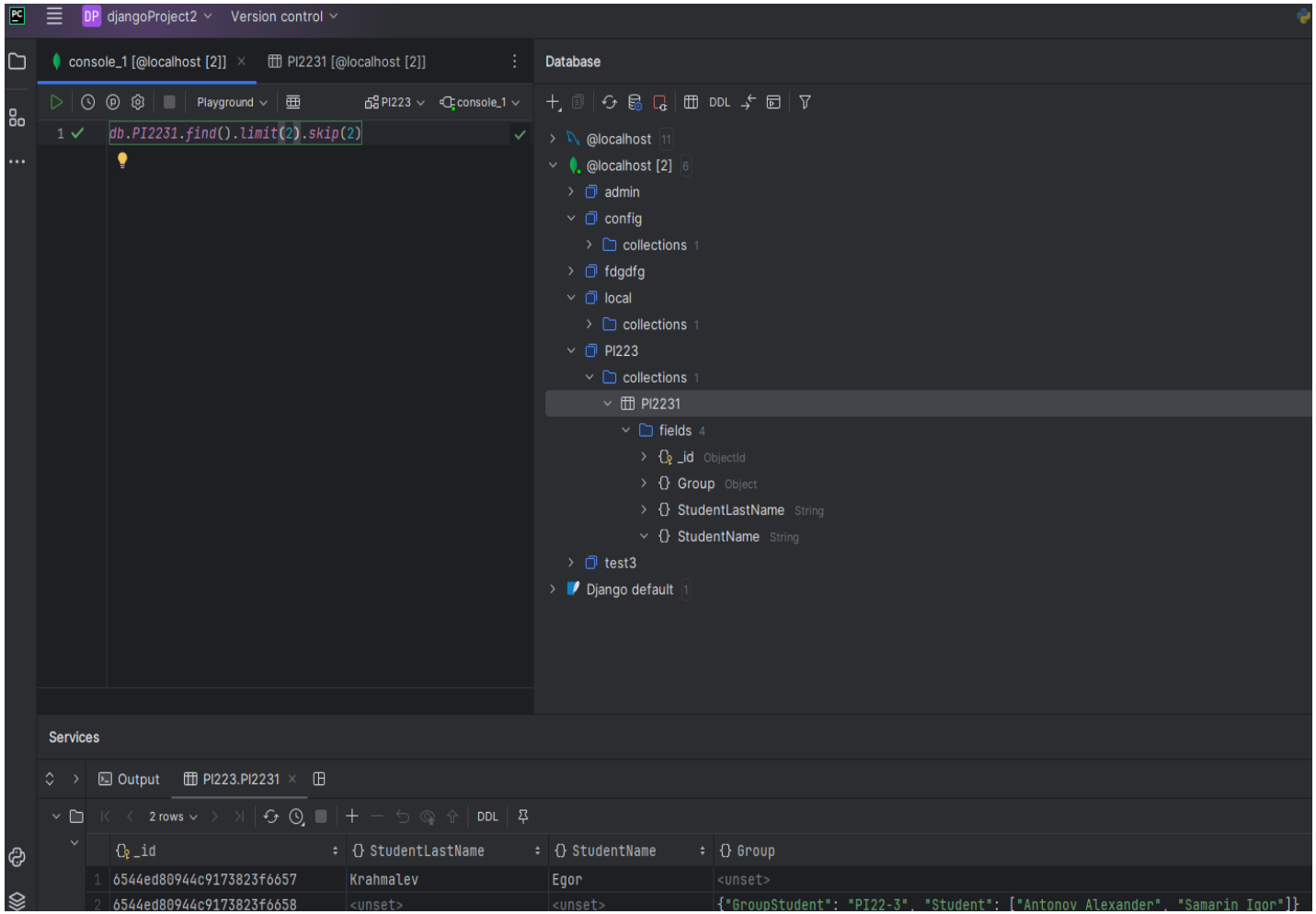
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

17.1. Пагинация и сортировка. Ограничение вывода и удаление

```
db.PI2231.find().limit(2).skip(2)
```

Результат:



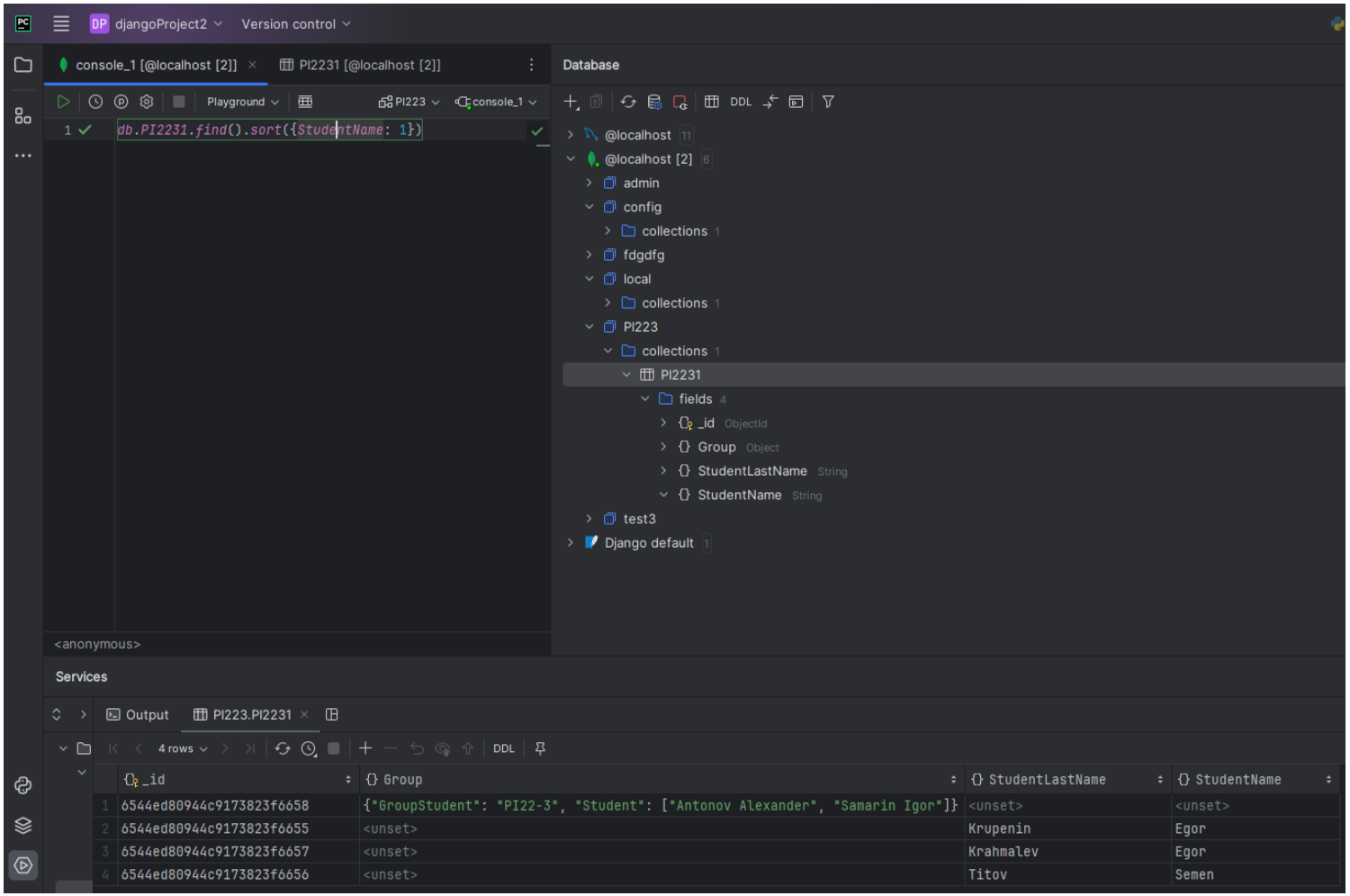
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

17.2. Пагинация и сортировка. Классическая сортировка в алфавитном порядке

```
db.PI2231.find().sort({StudentName: 1})
```

Результат:



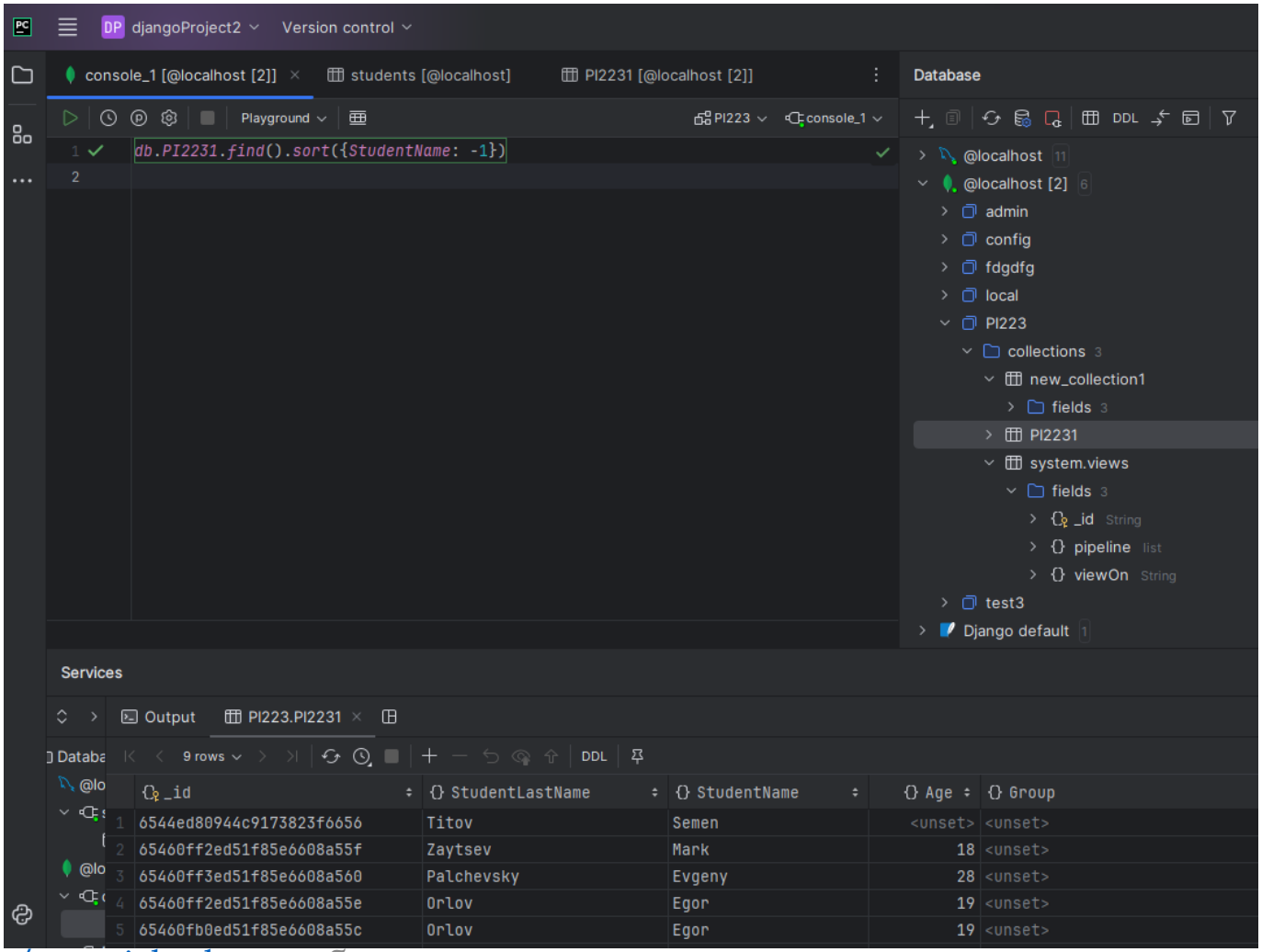
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

17.3. Пагинация и сортировка. Классическая сортировка в обратном порядке

```
db.PI2231.find().sort({StudentName: -1})
```

Результат:



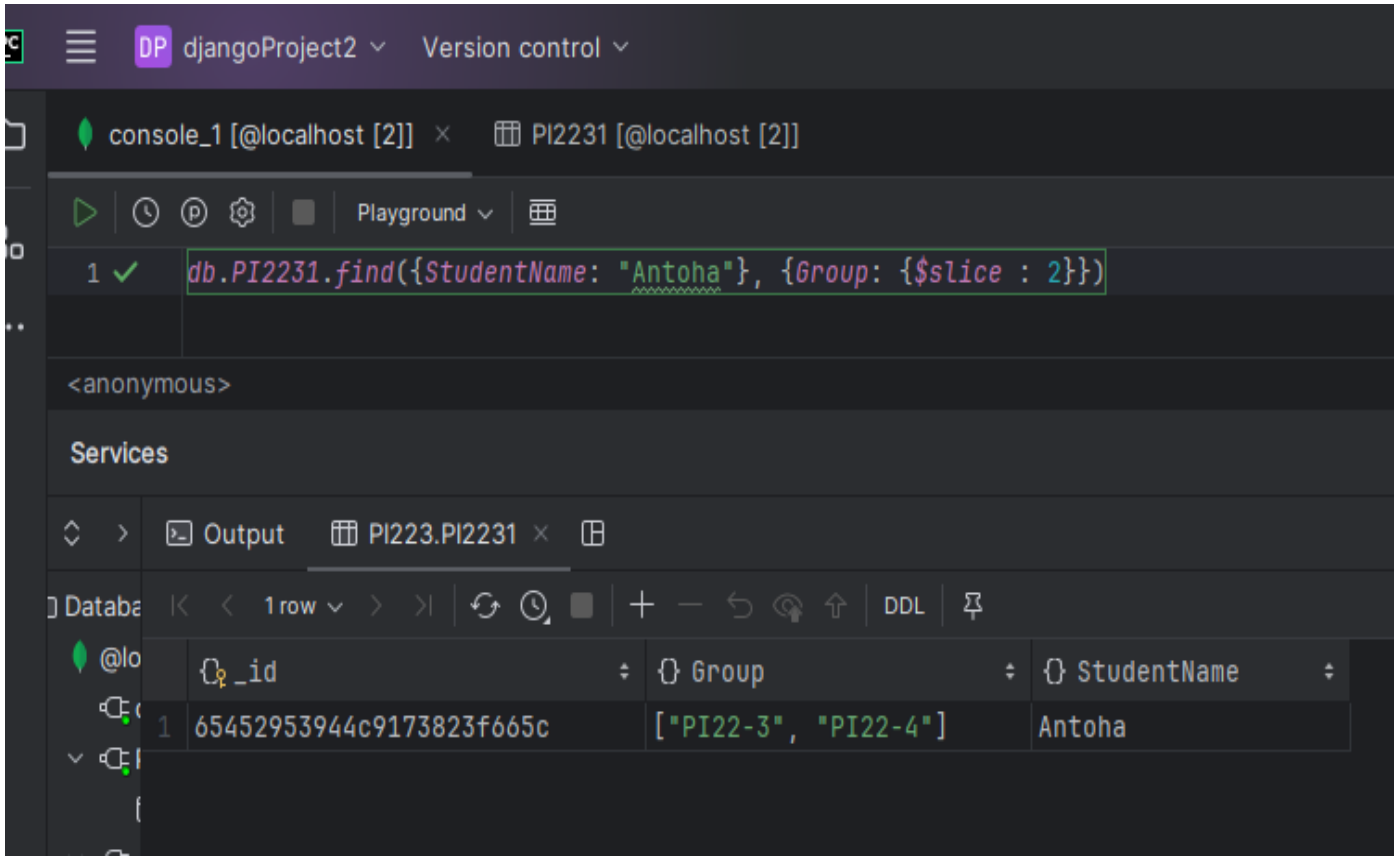
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

18. Slice (срез)

```
db.PI2231.find({StudentName: "Antoha"}, {Group: {$slice : 2}})
```

Результат:



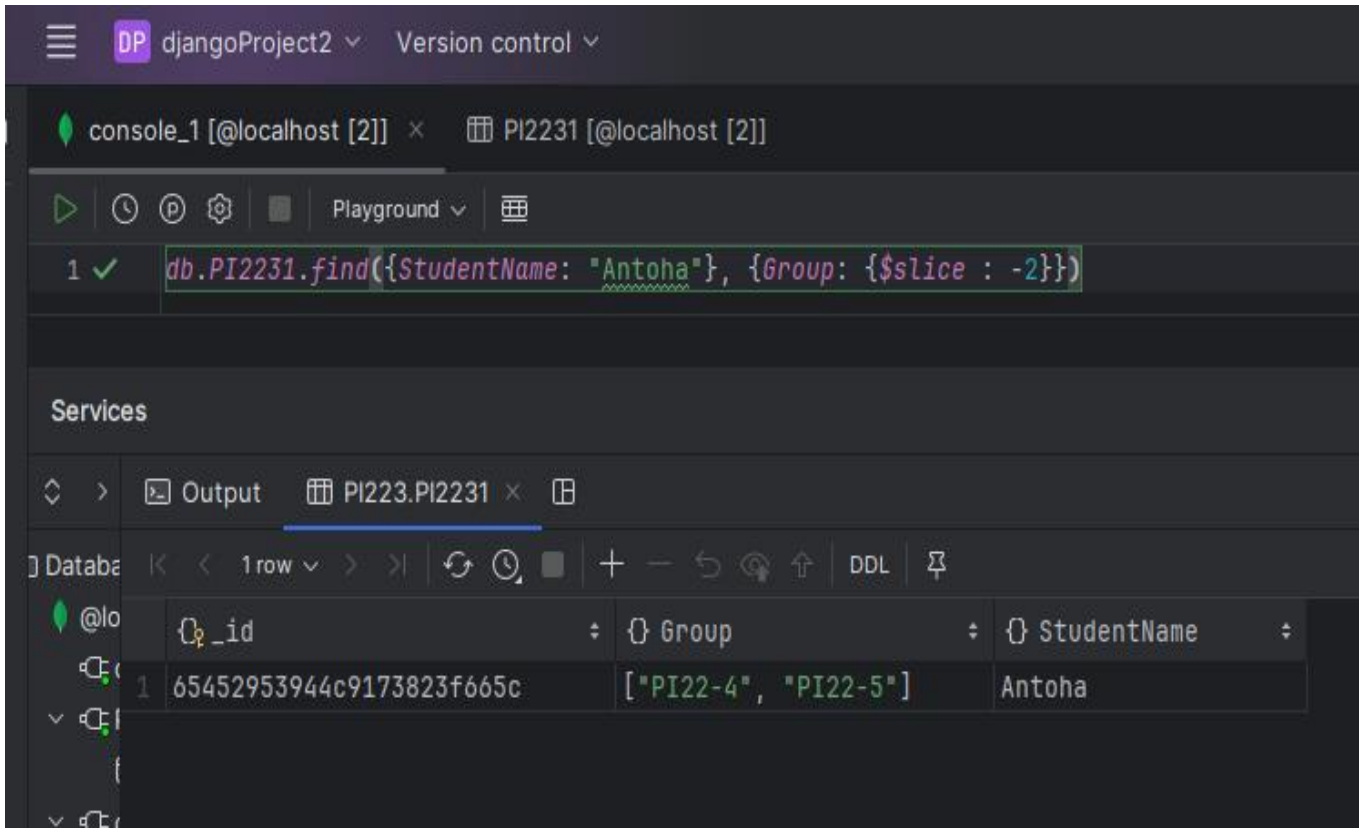
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

18.1. Slice (срез) – обратный срез

```
db.PI2231.find({StudentName: "Antoha"}, {Group: {$slice : -2}})
```

Результат:



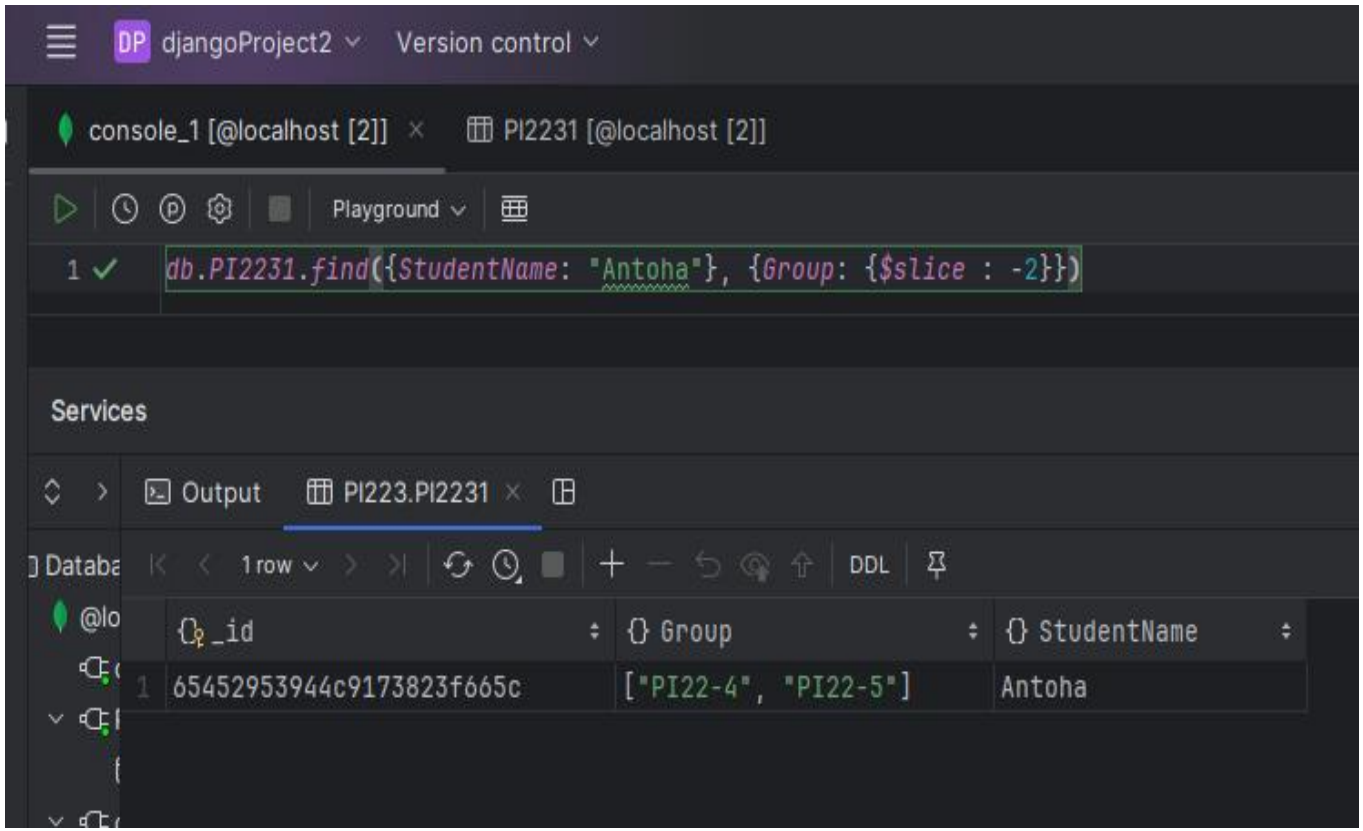
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

18.1. Slice (срез) – обратный срез

```
db.PI2231.find({StudentName: "Antoha"}, {Group: {$slice : -2}})
```

Результат:



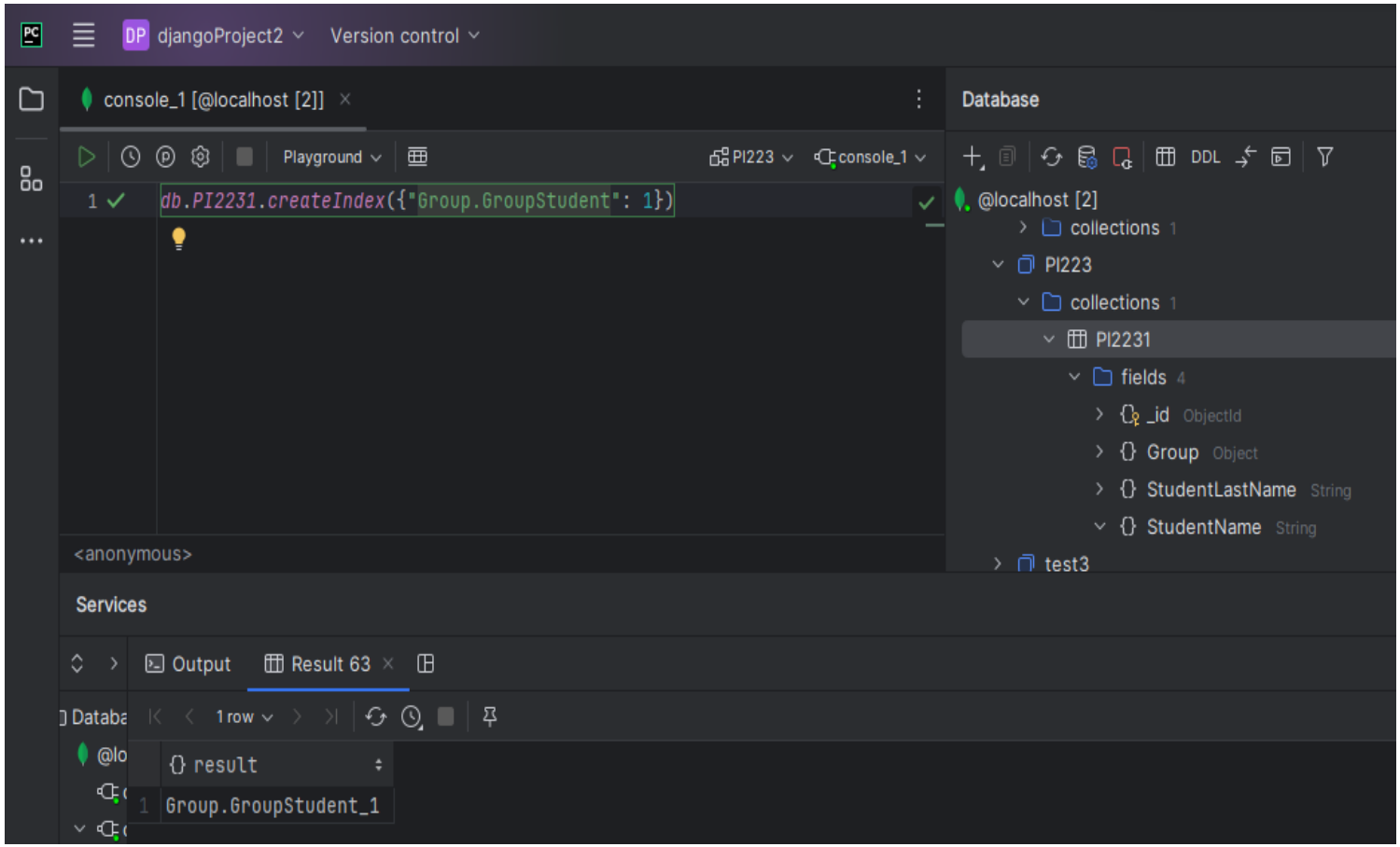
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

19. Индексы. Создание одиночного индекса

```
db.PI2231.createIndex({"Group.GroupStudent": 1})
```

Результат:



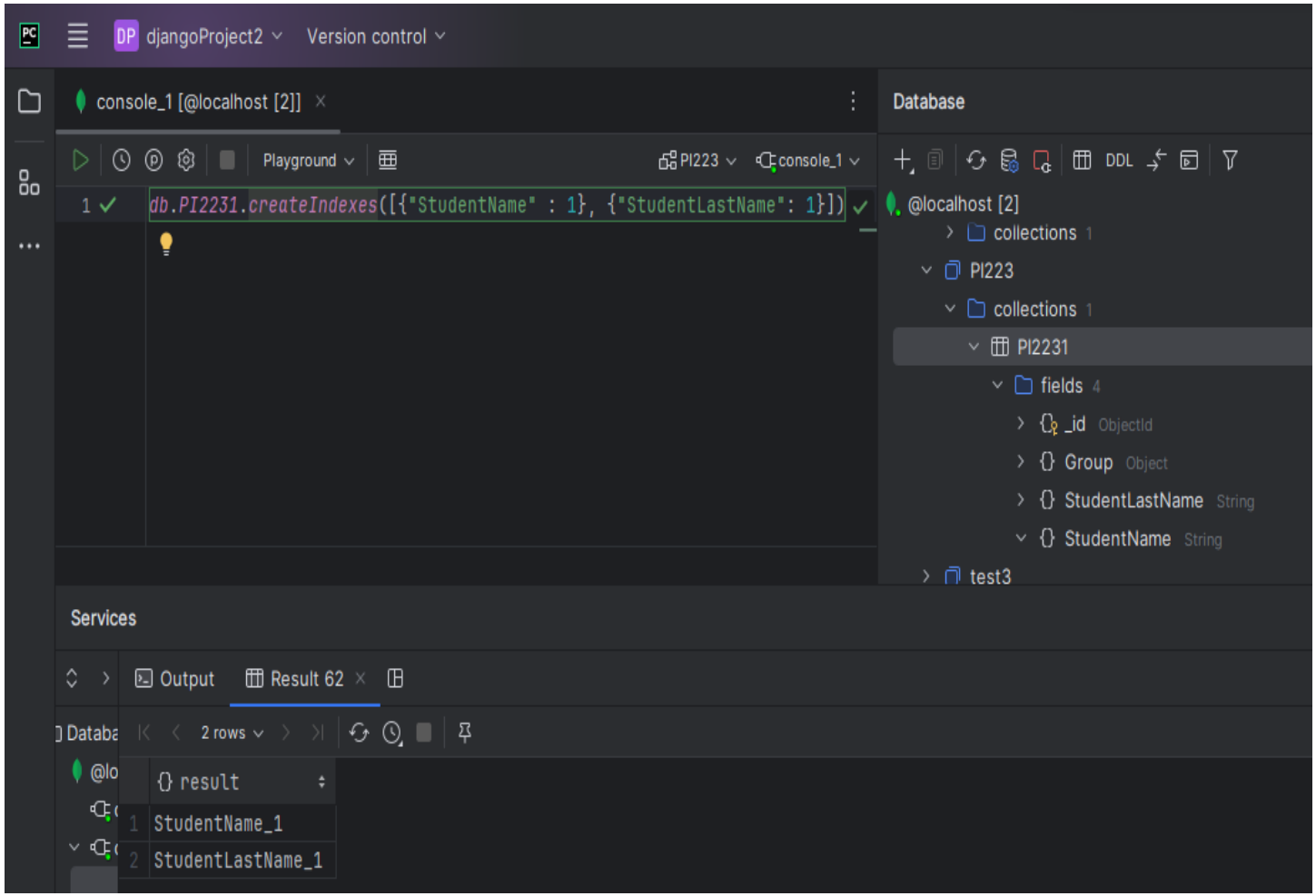
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

19.1. Индексы. Одновременное создание нескольких индексов

```
db.PI2231.createIndexes([{"StudentName" : 1}, {"StudentLastName": 1}])
```

Результат:



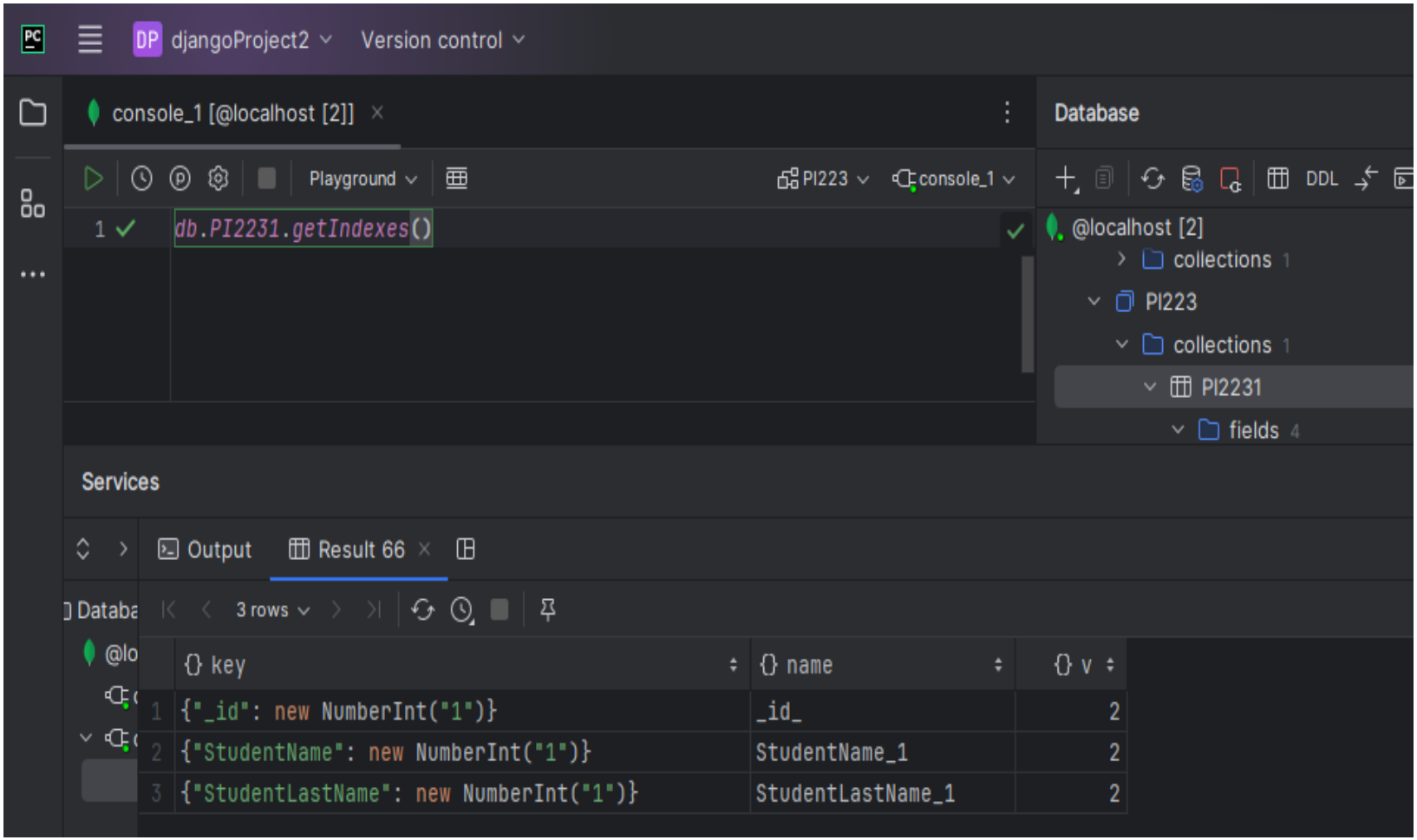
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

19.2. Индексы. Удаление

```
db.PI2231.dropIndex("Group.GroupStudent_1")
db.PI2231.getIndexes()
```

Результат:



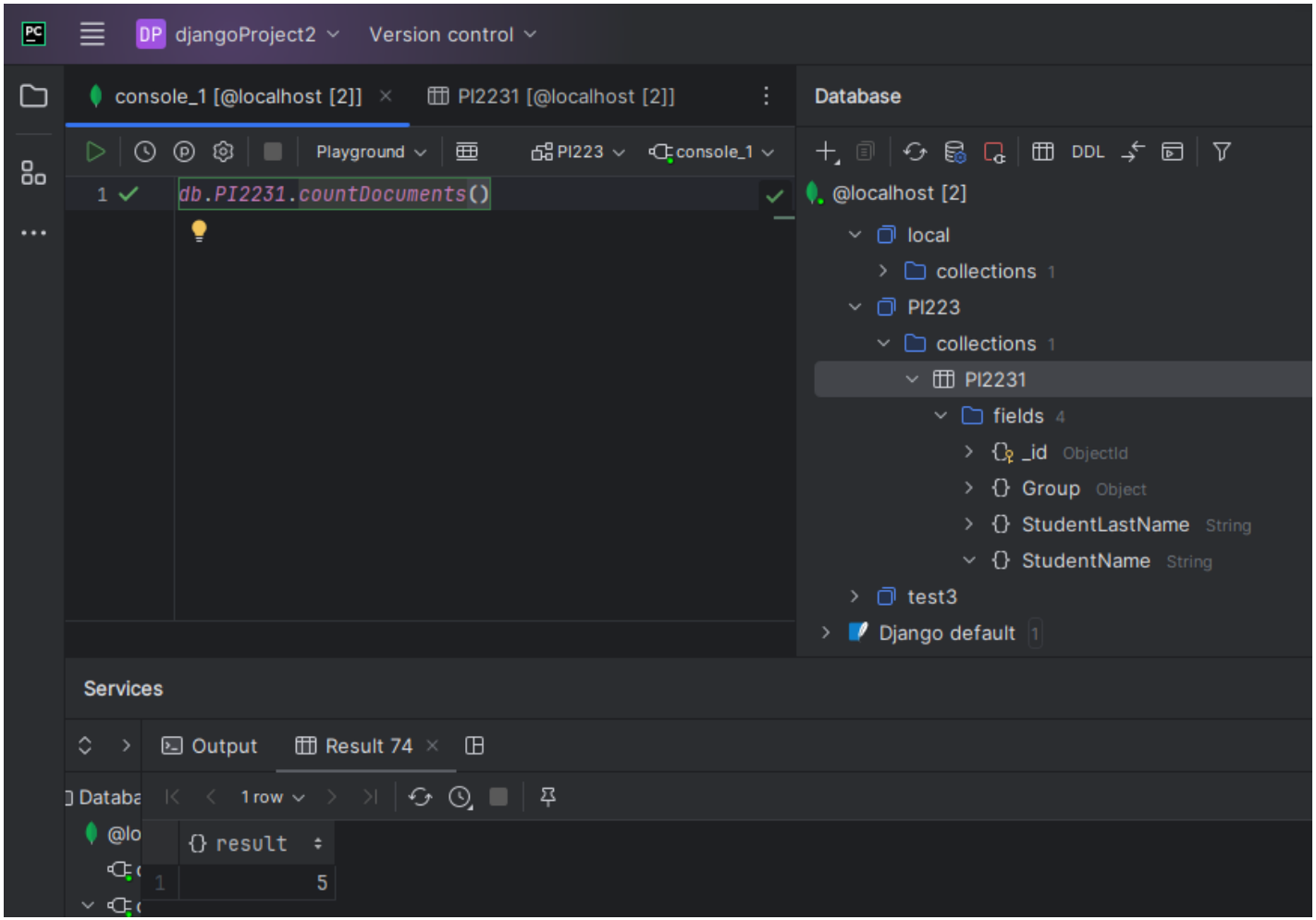
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

20. Агрегатные функции. Счетчик. Общее число документов коллекции

```
db.PI2231.countDocuments()
```

Результат:



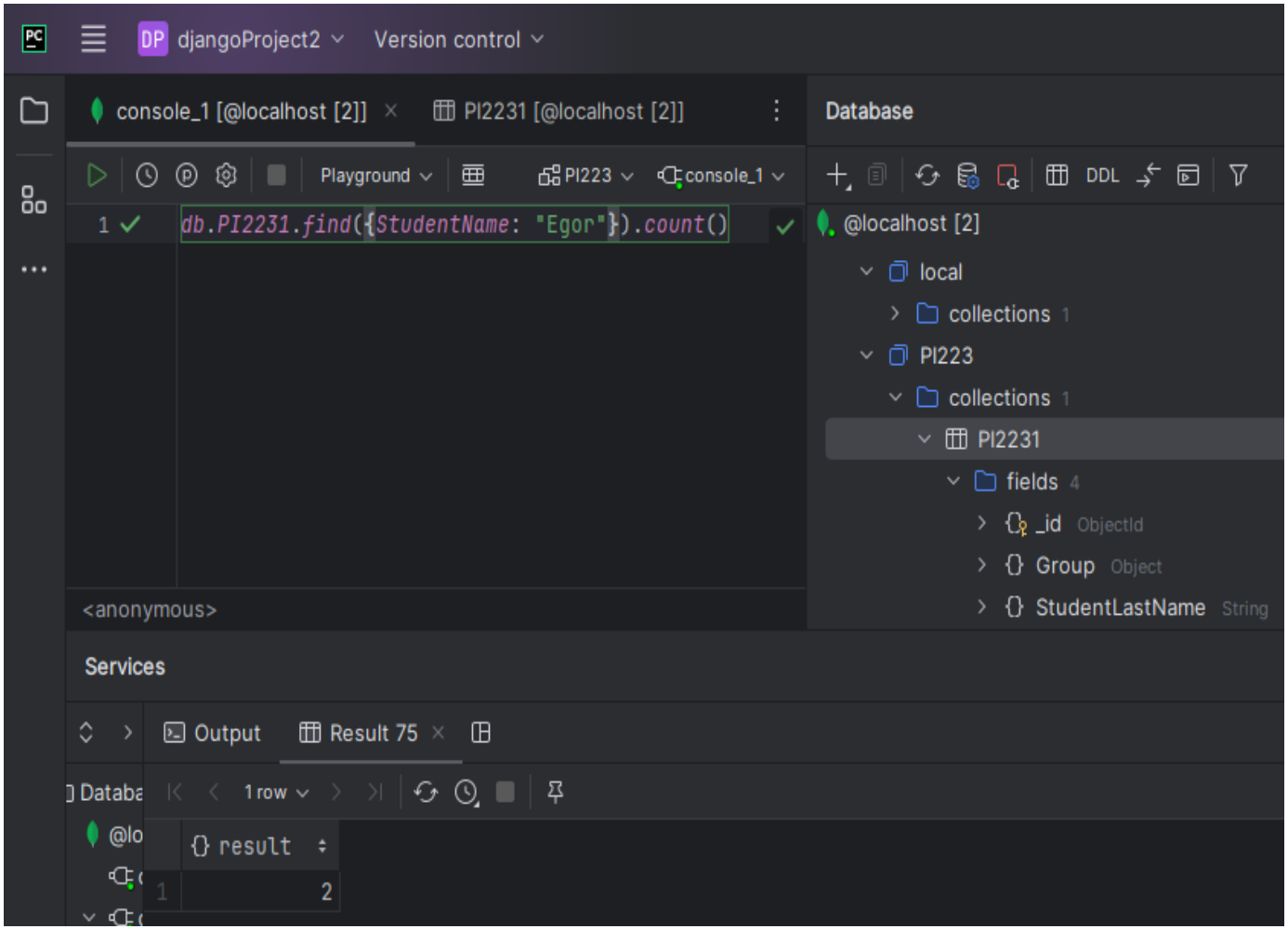
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

20.1. Агрегатные функции. Сортировка и счетчик

```
db.PI2231.find({StudentName: "Egor"}).count()
```

Результат:



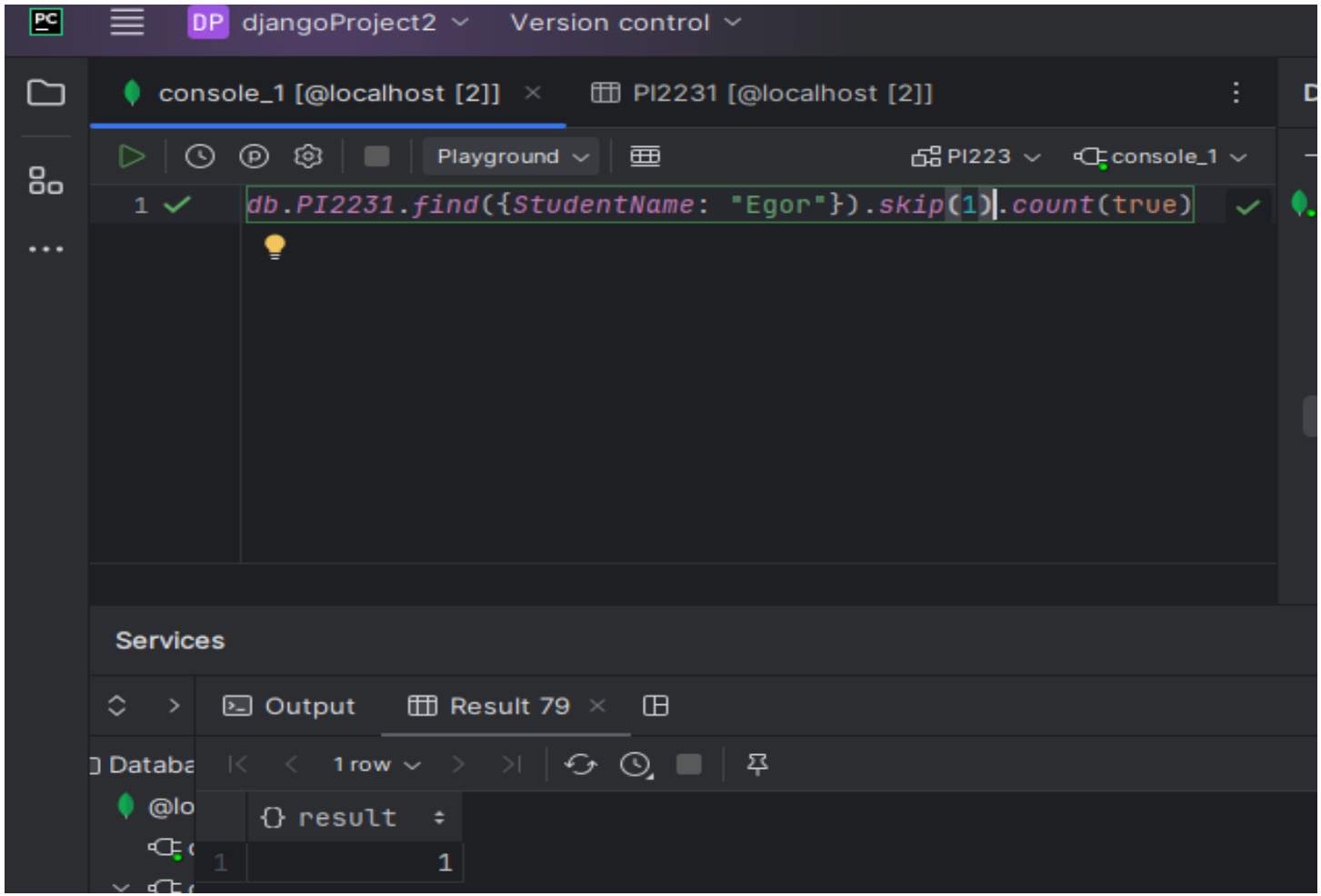
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

20.2. Агрегатные функции. Сортировка, счетчик и удаление

```
db.PI2231.find({StudentName: "Egor"}).skip(1).count(true)
```

Результат:



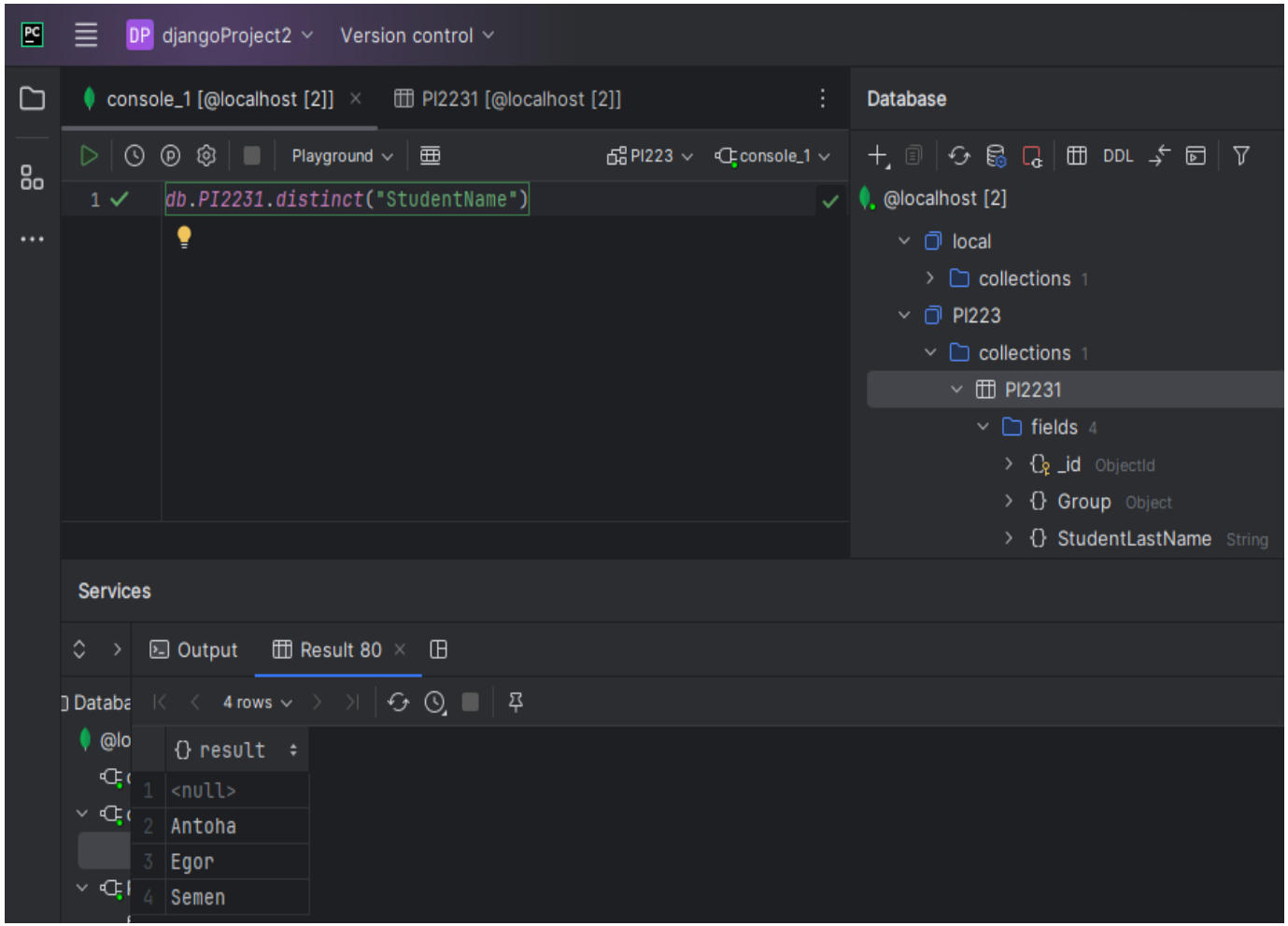
Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

20.3. Агрегатные функции. Вывод уникальных значений по полям

```
db.PI2231.distinct("StudentName")
```

Результат:



Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

20.4. Агрегатные функции. Min и Max

Добавим некоторые документы:

```
db.PI2231.insertOne({StudentName: "Egor", StudentLastName: "Orlov", "Age": 19})
db.PI2231.insertOne({StudentName: "Mark", StudentLastName: "Zaytsev", "Age": 18})
db.PI2231.insertOne({StudentName: "Evgeny", StudentLastName: "Palchevsky", "Age": 28})
```

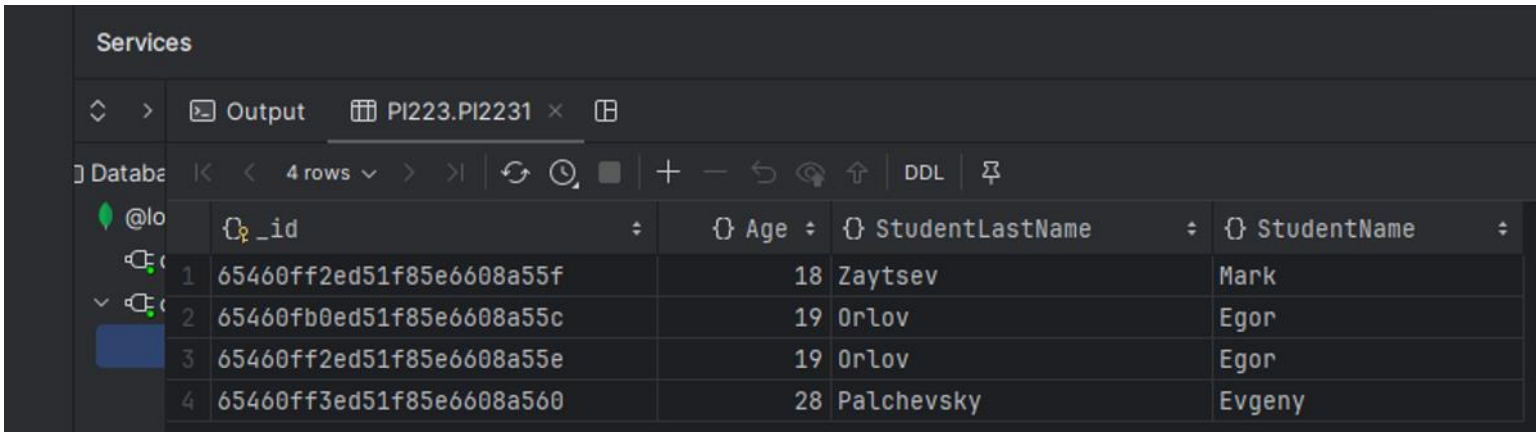
Присвоим полю «Age» индекс:

```
db.PI2231.createIndex({"Age": 1})
```

Теперь выведем студентов, у кого возраст 18 и выше:

```
db.PI2231.find().min({Age:18}).hint({Age:1})
```

Результат:



	_id	Age	StudentLastName	StudentName
1	65460ff2ed51f85e6608a55f	18	Zaytsev	Mark
2	65460fb0ed51f85e6608a55c	19	Orlov	Egor
3	65460ff2ed51f85e6608a55e	19	Orlov	Egor
4	65460ff3ed51f85e6608a560	28	Palchevsky	Evgeny

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

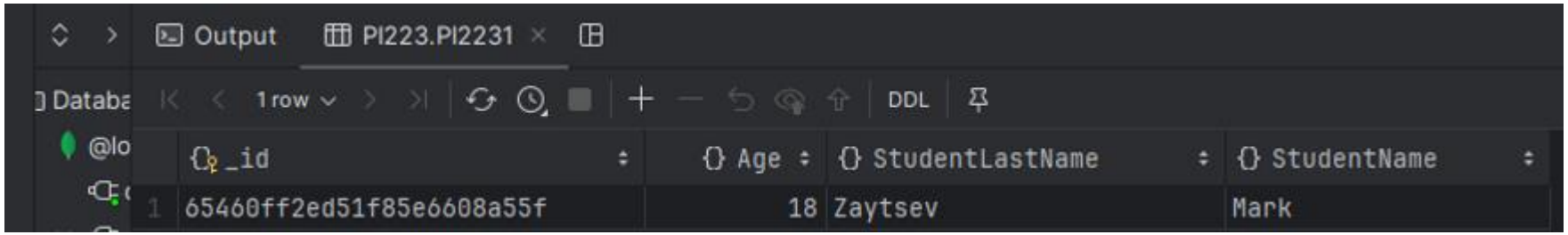
21. Операторы выборки. Условные операторы

- **$eq** (равно)
- **$ne** (не равно)
- **$gt** (больше чем)
- **$lt** (меньше чем)
- **$gte** (больше или равно)
- **$lte** (меньше или равно)
- **$in** определяет массив значений, одно из которых должно иметь поле документа
- **$nin** определяет массив значений, которые не должно иметь поле документа

Выведем студентов, у которых возраст меньше 19:

```
db.PI2231.find({Age: {$lt: 19}})
```

Результат:



Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

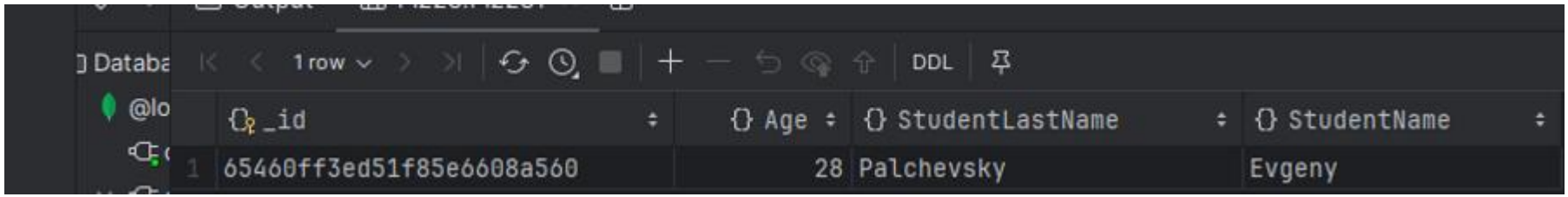
21.1. Операторы выборки. Условные операторы

- **$eq** (равно)
- **$ne** (не равно)
- **$gt** (больше чем)
- **$lt** (меньше чем)
- **$gte** (больше или равно)
- **$lte** (меньше или равно)
- **$in** определяет массив значений, одно из которых должно иметь поле документа
- **$nin** определяет массив значений, которые не должно иметь поле документа

Выведем студентов, у которых возраст больше 19:

```
db.PI2231.find({Age: {$gt: 19}})
```

Результат:



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

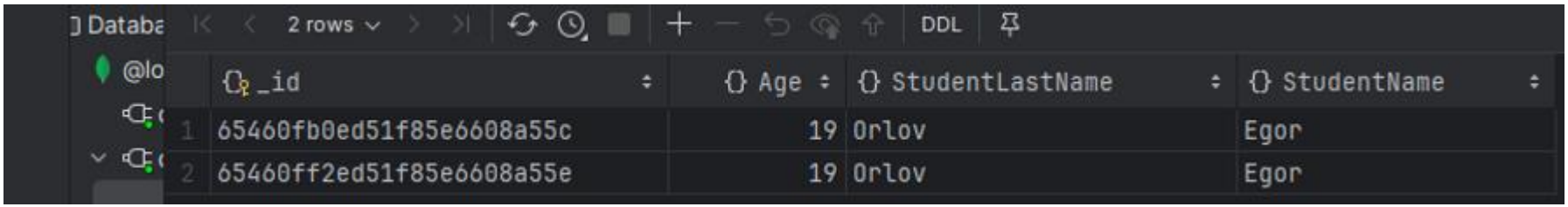
21.2. Операторы выборки. Условные операторы

- **$eq** (равно)
- **$ne** (не равно)
- **$gt** (больше чем)
- **$lt** (меньше чем)
- **$gte** (больше или равно)
- **$lte** (меньше или равно)
- **$in** определяет массив значений, одно из которых должно иметь поле документа
- **$nin** определяет массив значений, которые не должно иметь поле документа

Выведем студентов, возраст которых варьируется в промежутке от 18 до 28 не включительно:

```
db.PI2231.find({Age: {$gt : 18, $lt: 28}})
```

Результат:



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

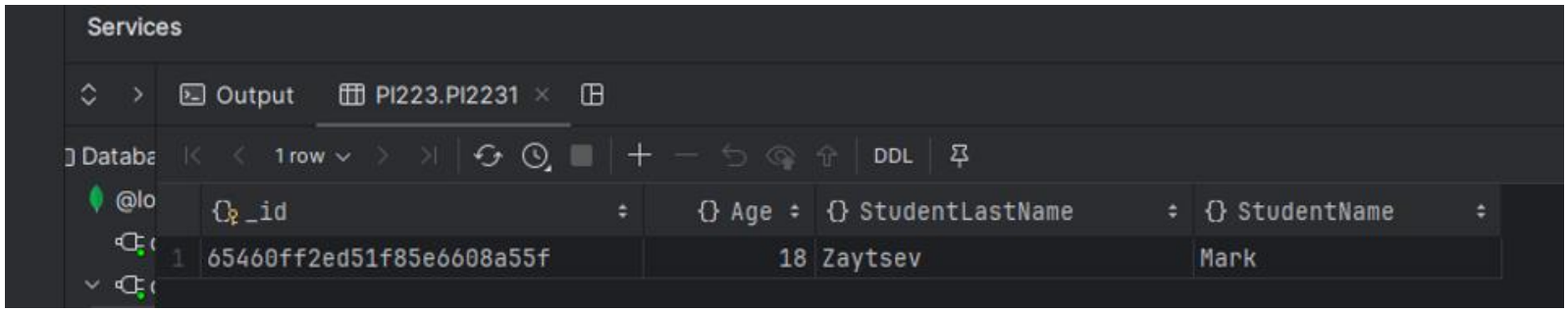
21.3. Операторы выборки. Условные операторы

- **$eq** (равно)
- **$ne** (не равно)
- **$gt** (больше чем)
- **$lt** (меньше чем)
- **$gte** (больше или равно)
- **$lte** (меньше или равно)
- **$in** определяет массив значений, одно из которых должно иметь поле документа
- **$nin** определяет массив значений, которые не должно иметь поле документа

Выведем студентов, возраст которых равен 18:

```
db.PI2231.find({Age: {$eq : 18}})
```

Результат:



Лекция №2: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

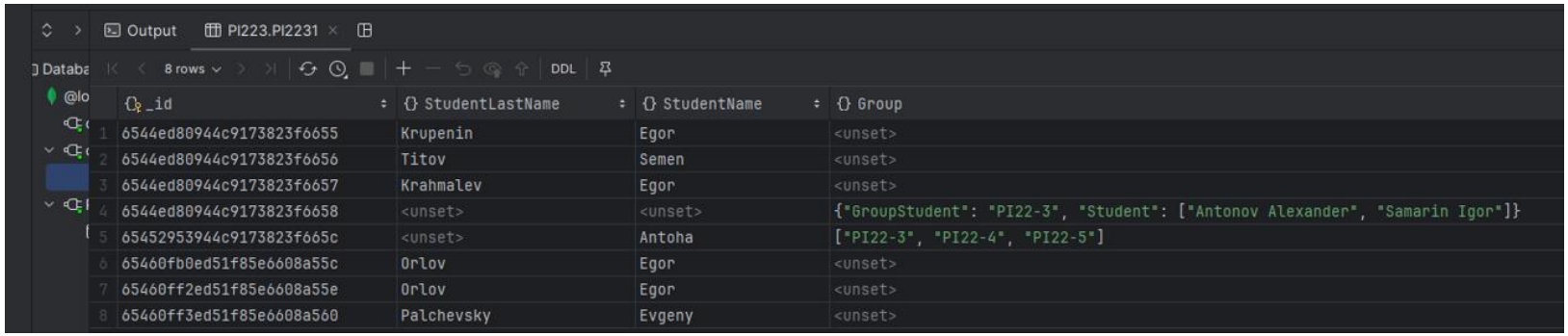
21.4. Операторы выборки. Условные операторы

- **$eq** (равно)
- **$ne** (не равно)
- **$gt** (больше чем)
- **$lt** (меньше чем)
- **$gte** (больше или равно)
- **$lte** (меньше или равно)
- **$in** определяет массив значений, одно из которых должно иметь поле документа
- **$nin** определяет массив значений, которые не должно иметь поле документа

Выведем студентов, возраст которых НЕ равен 18:

```
db.PI2231.find({Age: {$ne : 18}})
```

Результат:



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

22. Логические операторы

- **$or** (ИЛИ)
- **$and** (И)

Выведем всех студентов, имя которых Егор или студентам 18 лет:

```
use PI223
db.PI2231.find({$or : [{StudentName: "Egor"}, {Age: 18}]})
```

Результат:

Output | Result 5 | PI223.PI2231 | 5 rows | DDL

	_id	StudentLastName	StudentName	Age
1	6544ed80944c9173823f6655	Krupenin	Egor	<unset>
2	6544ed80944c9173823f6657	Krahmalev	Egor	<unset>
3	65460fb0ed51f85e6608a55c	Orlov	Egor	19
4	65460ff2ed51f85e6608a55e	Orlov	Egor	19
5	65460ff2ed51f85e6608a55f	Zaytsev	Mark	18

Лекция №3: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

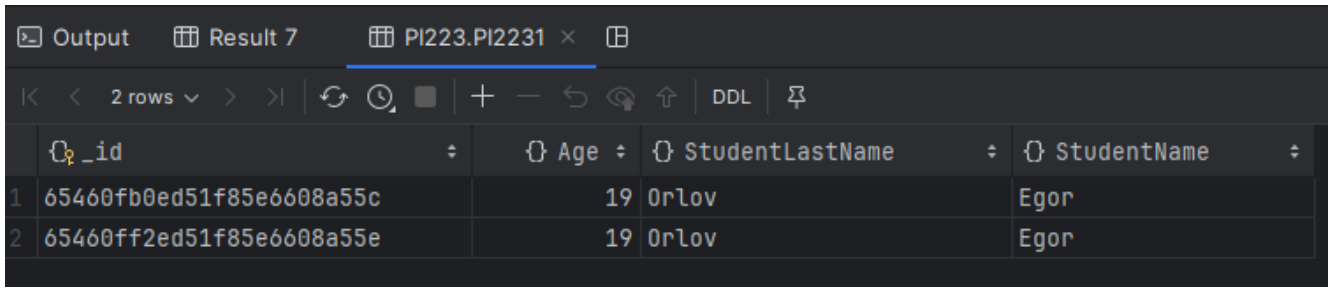
22.1. Логические операторы

- **$or** (ИЛИ)
- **$and** (И)

Выведем всех студентов, имя которых Егор и им 19 лет:

```
use PI223
db.PI2231.find({$and : [{StudentName: "Egor"}, {Age: 18}]})
```

Результат:



	_id	Age	StudentLastName	StudentName
1	65460fb0ed51f85e6608a55c	19	Orlov	Egor
2	65460ff2ed51f85e6608a55e	19	Orlov	Egor

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

23. Логические операторы. Поиск по массивам. Оператор **$all**

- **$all**: определяет набор значений, которые должны иметься в массиве
- **$size**: определяет количество элементов, которые должны быть в массиве
- **$elemMatch**: определяет условие, которому должны соответствовать элементы в массиве

Ищем студентов в массиве Group, группа которых равна PI22-3.

```
use PI223
db.PI2231.find({Group: {$all: ["PI22-3"]}})
```

Результат:

Output | Result 9 | PI223.PI2231

1 row | DDL

	_id	Group	StudentName
1	65452953944c9173823f665c	["PI22-3", "PI22-4", "PI22-5"]	Antoha

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

23.1. Логические операторы. Поиск по массивам. Оператор **$elemMatch**

- **$all**: определяет набор значений, которые должны иметься в массиве
- **$size**: определяет количество элементов, которые должны быть в массиве
- **$elemMatch**: определяет условие, которому должны соответствовать элементы в массиве

Найдем студентов, у которых за курс по MongoDB оценочка выше 2.

```
use PI223
db.PI2231.find({courses: {$elemMatch: {name: "MongoDB", grade: {$gt: 2}}}})
```

Результат:

Output | PI223.PI2231 | Result 13

2 rows | DDL

	_id	Group	StudentLastName	StudentName	courses
1	654f676767db131e90a1c29e	PI22-3	Bespalaya	Angelina	[{"name": "Python", "grade": new NumberInt("3")}, {"name": "MongoDB", "grade": new NumberInt("3")}]
2	654f676767db131e90a1c29f	PI22-3	Guseva	Anzhelika	[{"name": "Java", "grade": new NumberInt("5")}, {"name": "MongoDB", "grade": new NumberInt("5")}]

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

23.2. Логические операторы. Поиск по массивам. Оператор **$size**

- **$all**: определяет набор значений, которые должны иметься в массиве
- **$size**: определяет количество элементов, которые должны быть в массиве
- **$elemMatch**: определяет условие, которому должны соответствовать элементы в массиве

Извлечем все документы колонки «Курсы», в которых в массиве содержатся два элемента.

```
use PI223
db.PI2231.find({courses: {$size: 2}})
```

Результат:

Output | PI223.PI2231 | Result 22

2 rows | DDL | JSON

_id	Group	StudentLastName	StudentName	courses
654f676767db131e90a1c29e	PI22-3	Bespalaya	Angelina	[{"name": "Python", "grade": new NumberInt("3")}, {"name": "MongoDB", "grade": new NumberInt("3")}]
654f676767db131e90a1c29f	PI22-3	Guseva	Anzhelika	[{"name": "Java", "grade": new NumberInt("5")}, {"name": "MongoDB", "grade": new NumberInt("5")}]

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

23.3. Логические операторы. Остальное. Оператор **$exists**

$exists: позволяет извлечь только те документы, в которых определенный ключ присутствует или отсутствует

Вернем все документы, в которых заполнено поле Group (т.е. ключ – это, например, поле)

```
use PI223
db.PI2231.find({Group: {$exists: true}})
```

Результат:

	_id	Group	StudentName	StudentLastName	courses
1	6544ed80944c9173823f6658	{"GroupStudent": "PI22-3", "Student": ["Antonov Alexander", "Samarin Igor"]}	<unset>	<unset>	<unset>
2	65452953944c9173823f665c	["PI22-3", "PI22-4", "PI22-5"]	Antoha	<unset>	<unset>
3	654f676767db131e90a1c29e	PI22-3	Angelina	Bespalaya	[{"name": "Python", "grade": new NumberInt("3")}, {"name": "M
4	654f676767db131e90a1c29f	PI22-3	Anzhelika	Guseva	[{"name": "Java", "grade": new NumberInt("5")}, {"name": "Mon

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

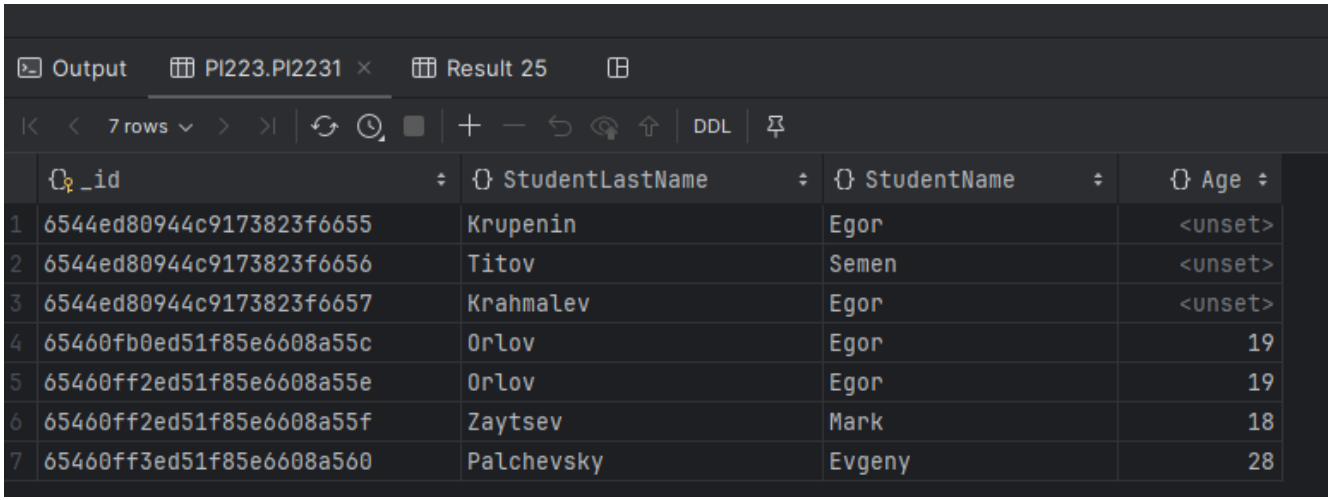
23.4. Логические операторы. Остальное. Оператор **$exists**

$exists: позволяет извлечь только те документы, в которых определенный ключ присутствует или отсутствует

Вернем все документы, в которых НЕ заполнено поле Group (т.е. ключ – это, например, поле)

```
use PI223
db.PI2231.find({Group: {$exists: false}})
```

Результат:



	_id	StudentLastName	StudentName	Age
1	6544ed80944c9173823f6655	Krupenin	Egor	<unset>
2	6544ed80944c9173823f6656	Titov	Semen	<unset>
3	6544ed80944c9173823f6657	Krahmalev	Egor	<unset>
4	65460fb0ed51f85e6608a55c	Orlov	Egor	19
5	65460ff2ed51f85e6608a55e	Orlov	Egor	19
6	65460ff2ed51f85e6608a55f	Zaytsev	Mark	18
7	65460ff3ed51f85e6608a560	Palchevsky	Evgeny	28

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

23.5. Логические операторы. Остальное. Оператор **$type**

$type: извлекает только те документы, в которых определенный ключ имеет значение определенного типа, например, строку или число:

```
use PI223
db.PI2231.find({Group: {$type:"string"}})
db.PI2231.find({Age: {$type:"number"}})
```

Результат (строки):

Output | PI223.PI2231

3 rows | DDL | JSON

	_id	Group	StudentName	StudentLastName	courses
1	65452953944c9173823f665c	["PI22-3", "PI22-4", "PI22-5"]	Antoha	<unset>	<unset>
2	654f676767db131e90a1c29e	PI22-3	Angelina	Bespalaya	[{"name": "Python", "grade": new NumberInt("3")}, {"name": "MongoDB", "grade": new NumberInt(
3	654f676767db131e90a1c29f	PI22-3	Anzhelika	Guseva	[{"name": "Java", "grade": new NumberInt("5")}, {"name": "MongoDB", "grade": new NumberInt("5

Результат (числа):

Output | PI223.PI2231

4 rows | DDL

	_id	Age	StudentLastName	StudentName
1	65460ff2ed51f85e6608a55f	18	Zaytsev	Mark
2	65460fb0ed51f85e6608a55c	19	Orlov	Egor
3	65460ff2ed51f85e6608a55e	19	Orlov	Egor
4	65460ff3ed51f85e6608a560	28	Palchevsky	Evgeny

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

23.6. Логические операторы. Остальное. Оператор **$regex**

$regex: задает регулярное выражение, которому должно соответствовать значение поля.

Например, пусть поле StudentName обязательно имеет букву "A":

```
use PI223
db.PI2231.find({StudentName: {$regex:"A"}})
```

Результат:

Output | Result 34 | PI223.PI2231

3 rows | DDL | JSON

	_id	Group	StudentLastName	StudentName	courses
1	654f676767db131e90a1c29e	PI22-3	Bespalaya	Angelina	[{"name": "Python", "grade": new NumberInt("3")}, {"name": "MongoDB", "grade": new NumberInt(
2	65452953944c9173823f665c	["PI22-3", "PI22-4", "PI22-5"]	<unset>	Antoha	<unset>
3	654f676767db131e90a1c29f	PI22-3	Guseva	Anzhelika	[{"name": "Java", "grade": new NumberInt("5")}, {"name": "MongoDB", "grade": new NumberInt("5

Можно использовать и регулярные выражения: например, значение поля StudentName должно оканчиваться на "na".

```
use PI223
db.PI2231.find({StudentName: {$regex:"na$"}})
```

Результат:

1 row | DDL | JSON

	_id	Group	StudentLastName	StudentName	courses
1	654f676767db131e90a1c29e	PI22-3	Bespalaya	Angelina	[{"name": "Python", "grade": new NumberInt("3")}, {"name": "MongoDB", "grade": new NumberInt("3")}]

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

24. Обновление и замена данных. Оператор **$replaceOne**

$replaceOne: заменяет действующий документ на новый.

У данной функции есть несколько параметров:

- **filter:** принимает запрос на выборку документа, который надо обновить
- **update:** представляет новый документ, который заместит старый при обновлении
- **options:** определяет дополнительные параметры при обновлении документов, основным из которых является параметр **upsert**.

```
use PI223
db.PI2231.replaceOne({StudentName: "Semen", StudentLastName: "Titov"}, {StudentName: "Semen", StudentLastName: "Titov", Group: "PI22-2"})
```

Результат:

filter {} × sort

	_id	Group	StudentLastName	StudentName
1	6544ed80944c9173823f6655	PI22-2	<unset>	<unset>
2	6544ed80944c9173823f6656	PI22-2	Titov	Semen

Лекция №3: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

24.1. Обновление и замена данных. Операторы **$updateOne** и **$updateMany**

$updateOne: обновляет один документ.

$updateMany: обновляет несколько документов.

Обновим документ «Семен Титов», присвоив значение полю «Age»:

```
use PI223
db.PI2231.updateOne({StudentName: "Semen", StudentLastName: "Titov"}, {$set: {Age: 28}})
```

Результат:

	_id	Group	Age	StudentLastName	StudentName
1	6544ed80944c9173823f6655	PI22-2	<unset>	<unset>	<unset>
2	6544ed80944c9173823f6656	PI22-2	28	Titov	Semen
3	6544ed80944c9173823f6657	<unset>	23	Krahmalev	Egor

Если необходимо обновить все поля, имя которых «Егор», то делаем следующее:

```
use PI223
db.PI2231.updateMany({StudentName: "Egor"}, {$set: {Age: 23}})
```

Результат:

11 rows — DDL — JSON — filter {} — sort

	_id	Group	Age	StudentLastName	StudentName
1	6544ed80944c9173823f6655	PI22-2	<unset>	<unset>	<unset>
2	6544ed80944c9173823f6656	PI22-2	28	Titov	Semen
3	6544ed80944c9173823f6657	<unset>	23	Krahmalev	Egor
4	6544ed80944c9173823f6658	{"GroupStudent": "PI22-3", "Student": ["Antonov Alexan	<unset>	<unset>	<unset>
5	65452953944c9173823f665c	["PI22-3", "PI22-4", "PI22-5"]	<unset>	<unset>	Antoha
6	65460fb0ed51f85e6608a55c	<unset>	23	Orlov	Egor
7	65460ff2ed51f85e6608a55e	<unset>	23	Orlov	Egor
8	65460ff2ed51f85e6608a55f	<unset>	18	Zaytsev	Mark
9	65460ff3ed51f85e6608a560	<unset>	28	Palchevsky	Evgeny
10	654f676767db131e90a1c29e	PI22-3	<unset>	Bespalaya	Angelina
11	654f676767db131e90a1c29f	PI22-3	<unset>	Guseva	Anzhelika

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

24.2. Обновление и замена данных. Оператор **$inc**

$inc: увеличение значения числового поля на определенное количество единиц .

Добавим студенту Егору Крахмалеву 10 лет к его текущему возрасту:

```
use PI223
db.PI2231.updateOne({StudentName: "Egor", StudentLastName: "Krahmalev"}, {$inc: {Age: 10}})
```

Результат (было 23, а стало 33):

filter {} × sort

	_id	Group	Age	StudentLastName	StudentName
1	6544ed80944c9173823f6655	PI22-2	<unset>	<unset>	<unset>
2	6544ed80944c9173823f6656	PI22-2	28	Titov	Semen
3	6544ed80944c9173823f6657	<unset>	33	Krahmalev	Egor

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

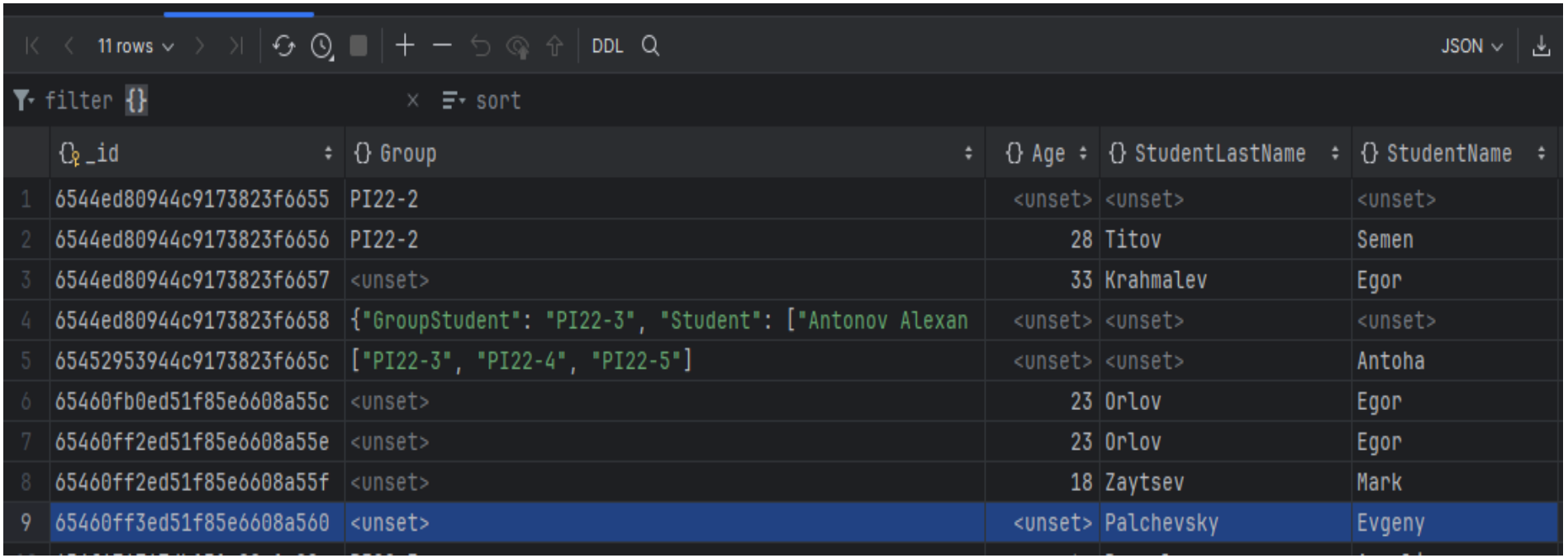
24.3. Обновление и замена данных. Удаление поля. Оператор **$unset**

$unset: удаление определенного поля.

Удалим у Евгения Пальчевского возраст:

```
db.PI2231.updateOne({StudentName: "Evgeny", StudentLastName: "Palchevsky"}, {$unset: {Age: 28}})
```

Результат (было 28, а стало ничего):



	_id	Group	Age	StudentLastName	StudentName
1	6544ed80944c9173823f6655	PI22-2	<unset>	<unset>	<unset>
2	6544ed80944c9173823f6656	PI22-2	28	Titov	Semen
3	6544ed80944c9173823f6657	<unset>	33	Krahmalev	Egor
4	6544ed80944c9173823f6658	{"GroupStudent": "PI22-3", "Student": ["Antonov Alexan	<unset>	<unset>	<unset>
5	65452953944c9173823f665c	["PI22-3", "PI22-4", "PI22-5"]	<unset>	<unset>	Antoha
6	65460fb0ed51f85e6608a55c	<unset>	23	Orlov	Egor
7	65460ff2ed51f85e6608a55e	<unset>	23	Orlov	Egor
8	65460ff2ed51f85e6608a55f	<unset>	18	Zaytsev	Mark
9	65460ff3ed51f85e6608a560	<unset>	<unset>	Palchevsky	Evgeny

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

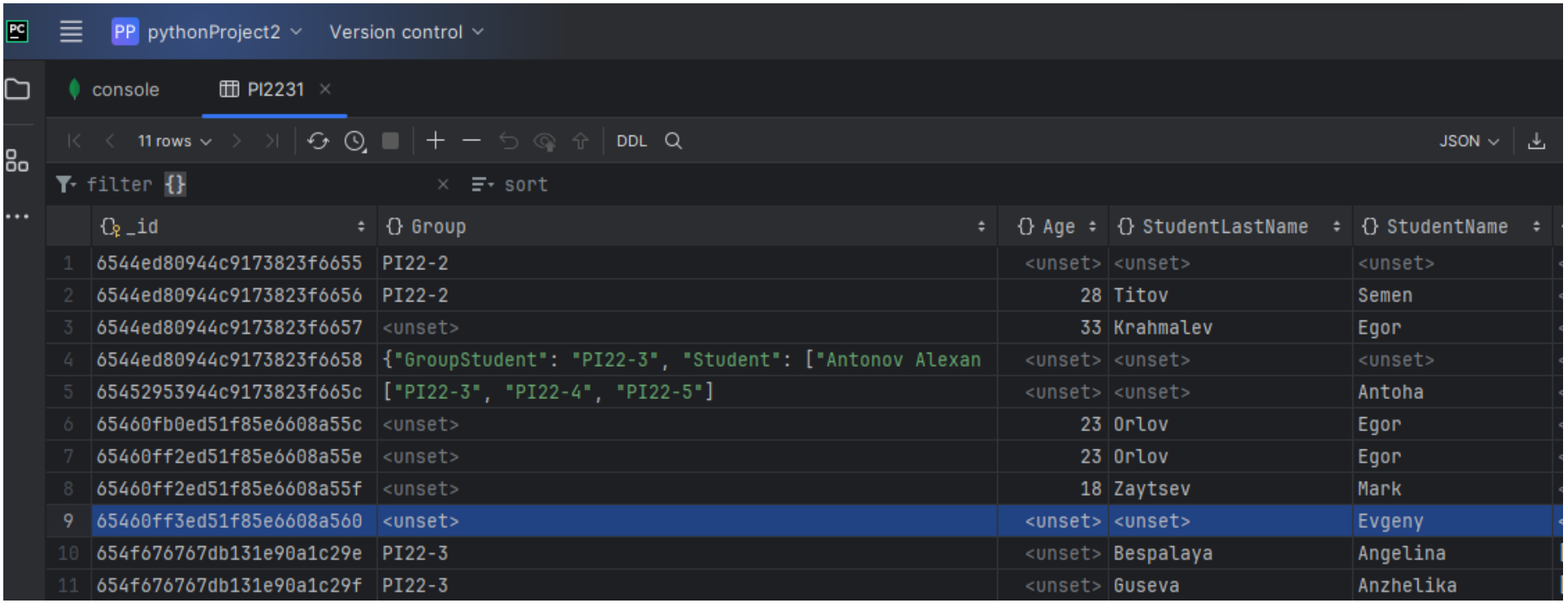
24.3. Обновление и замена данных. Удаление поля. Оператор **$unset**

$unset: удаление определенного поля.

Удалим у Евгения Пальчевского возраст и фамилию:

```
use PI223
db.PI2231.updateOne({StudentName: "Evgeny", StudentLastName: "Palchevsky"}, {$unset: {Age: 28, StudentLastName: "Palchevsky"}})
```

Результат (теперь еще и без фамилии):



	_id	Group	Age	StudentLastName	StudentName
1	6544ed80944c9173823f6655	PI22-2	<unset>	<unset>	<unset>
2	6544ed80944c9173823f6656	PI22-2	28	Titov	Semen
3	6544ed80944c9173823f6657	<unset>	33	Krahmalev	Egor
4	6544ed80944c9173823f6658	{"GroupStudent": "PI22-3", "Student": ["Antonov Alexan	<unset>	<unset>	<unset>
5	65452953944c9173823f665c	["PI22-3", "PI22-4", "PI22-5"]	<unset>	<unset>	Antoha
6	65460fb0ed51f85e6608a55c	<unset>	23	Orlov	Egor
7	65460ff2ed51f85e6608a55e	<unset>	23	Orlov	Egor
8	65460ff2ed51f85e6608a55f	<unset>	18	Zaytsev	Mark
9	65460ff3ed51f85e6608a560	<unset>	<unset>	<unset>	Evgeny
10	654f676767db131e90a1c29e	PI22-3	<unset>	Bespalaya	Angelina
11	654f676767db131e90a1c29f	PI22-3	<unset>	Guseva	Anzhelika

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

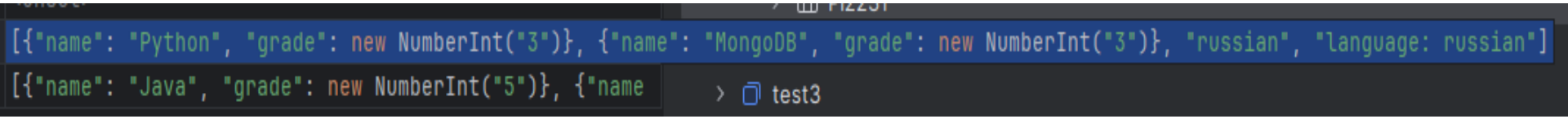
24.4. Обновление и замена данных. Обновление массивов. Оператор **$push**

$push: добавление еще одного значения к существующему.

Добавим к существующему массиву данные:

```
use PI223
db.PI2231.updateOne({StudentName: "Angelina"}, {$push: {courses: "language: russian"}})
```

Результат:



Через **updateMany** можем обновить несколько значений.

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

24.5. Обновление и замена данных. Обновление массивов. Оператор **$each**

$each: добавление нескольких значений к существующему.

Добавим к существующему массиву данные:

```
use PI223
db.PI2231.updateOne({StudentName: "Anzhelika"}, {$push: {courses: {$each: ["NoSQL", "JavaTop"]}}})
```

Результат:

```
Angelina    [{"name": "Python", "grade": new NumberInt("3")}, {"na
Anzhelika   [{"name": "Java", "grade": new NumberInt("5")}, {"name": "MongoDB", "grade": new NumberInt("5")}, "NoSQL", "JavaTop"]
```

system.views

Через **$position** и **$slice** можем задать позицию в массиве для вставки элементов и ограничить вывод элементов соответственно.

```
use PI223
db.PI2231.updateOne({StudentName: "Anzhelika"}, {$push: {courses: {$each: ["NoSQL", "JavaTop"], $position:1, $slice:2}}})
```

В данном случае элементы массива будут вставляться в массив «курсы» с 1-го индекса, и после вставки в массиве останутся только 2 первых элемента.

Результат:

```
Angelina    [{"name": "Python", "grade": new NumberInt("3")}, {"na
Anzhelika   [{"name": "Java", "grade": new NumberInt("5")}, "NoSQL"]
```

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

24.6. Обновление и замена данных. Обновление массивов. Оператор **$addToSet**

$addToSet: добавляет объекты в массив. Отличие состоит в том, что $addToSet добавляет данные, если их еще нет в массиве.

Добавим к существующему массиву данные:

```
use PI223
db.PI2231.updateOne({StudentName: "Anzhelika"}, {$addToSet: {courses: {lang: "russian"}}})
```

Результат:

```
Angelina    [{"name": "Python", "grade": new NumberInt("3")}, {"name": "
Anzhelika   [{"name": "Java", "grade": new NumberInt("5")}, "NoSQL", {"lang": "russian"}]
```

Лекция №3: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

24.7. Обновление и замена данных. Удаление массивов. Операторы $pop, $pull и $pullAll

$pop: удаляет элемент из массива.
$pull: удаляет каждое вхождение элемента в массив.
$pullAll: удаление нескольких значений.

Добавим данные еще раз:

```
use PI223
db.PI2231.updateOne({StudentName: "Anzhelika"}, {$push: {courses: {$each: ["NoSQL", "JavaTop", "test", "test"]}}})
```

Удалим данные:

```
use PI223
db.PI2231.updateOne({StudentName: "Anzhelika"}, {$pull: {courses: "test"}})
```

Результат:

<unset>	Bespalaya	Angelina	[{"name": "Python", "grade": new NumberInt("
<unset>	Guseva	Anzhelika	["NoSQL", "NoSQL", "NoSQL", "JavaTop"]

Удалим данные:

```
use PI223
db.PI2231.updateOne({StudentName: "Anzhelika"}, {$pullAll: {courses: ["NoSQL", "JavaTop"]}})
```

Результат:

Bespalaya	Angelina	[{"name": "Python", "grade": new NumberInt("3")}, {"name": "
Guseva	Anzhelika	["test", "test"]

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

25. Удаление данных. Операторы **$deleteOne() и $deleteMany()**

$deleteOne(): удаляет один документ.
$deleteMany(): удаляет несколько документов.

Удаляем первый найденный документ с именем «Евгений»:

```
use PI223
db.PI2231.deleteOne({StudentName: "Evgeny"})
```

Удаляем всех Егоров из нашей коллекции:

```
use PI223
db.PI2231.deleteMany({StudentName: "Egor"})
```

Результат:

		{} Age	{} StudentLastName
1		<unset>	<unset>
2		28	Titov
3	dent": "PI22-3", "Student": ["Antonov Alexander", "Samarin Igor"]}	:t>	
4	"PI22-4", "PI22-5"]	<unset>	<unset>
5		<unset>	Bespalaya
6		<unset>	Guseva

Лекция №3: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

25. Удаление данных. Операторы $deleteOne() и $deleteMany()

$deleteOne(): удаляет один документ.
$deleteMany(): удаляет несколько документов.

Удаляем всех студентов, возраст которых меньше 19:

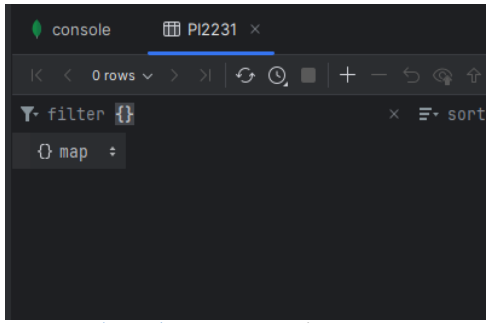
```
use PI223
db.PI2231.deleteMany({Age: {$lt: 19}})
```

Результат:

	Age	StudentLastName
1	<unset>	<unset>
2	28	Titov
3	dent": "PI22-3", "Student": ["Antonov Alexander", "Samarin Igor"]}	t>
4	"PI22-4", "PI22-5"]	<unset>	<unset>
5		<unset>	Bespalaya
6		<unset>	Guseva

Удаляем все документы из коллекции:

```
use PI223
db.PI2231.deleteMany({})
```



Результат:

https://palchevsky.ru/materials.php - учебные материалы и рейтинги

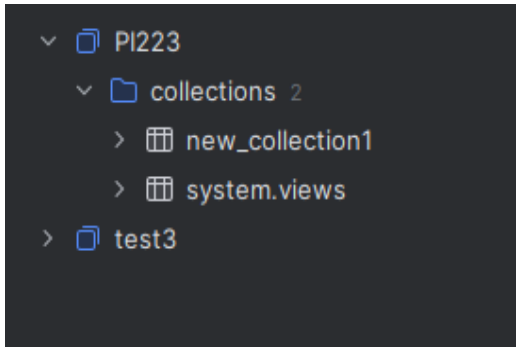
Лекция №3: Нереляционные базы данных. Синтаксис MongoDB

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

26. Манипуляции с коллекциями и базами данных

Удаление коллекции:

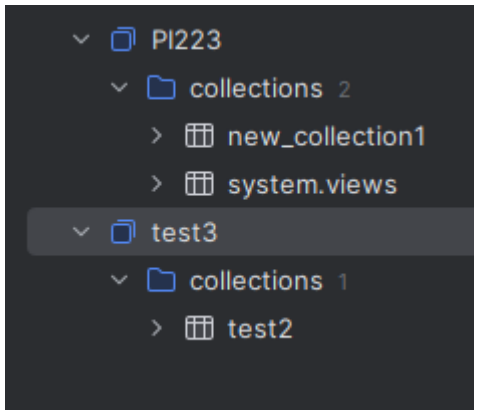
```
use PI223
db.PI2231.drop()
```



Создание и переименование коллекции:

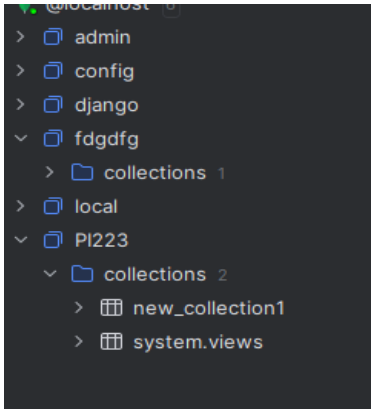
```
use test3
db.createCollection("test")
```

```
use test3
db.test.renameCollection("test2")
```



Удаление базы данных:

```
use test3
db.dropDatabase()
```



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Django и архитектура MVC

1 **Model**
Модель предметной области: *сущность; логика и процессы.*

2 **View**
Шаблоны: *html-файлы, в которых прописываем графический веб-интерфейс*

3 **Controller**
Контроллер: адреса, при открытии которых будут вызываться различные функции

В чем суть?

? Основные преимущества

Позволяет создавать динамичные веб-сайты

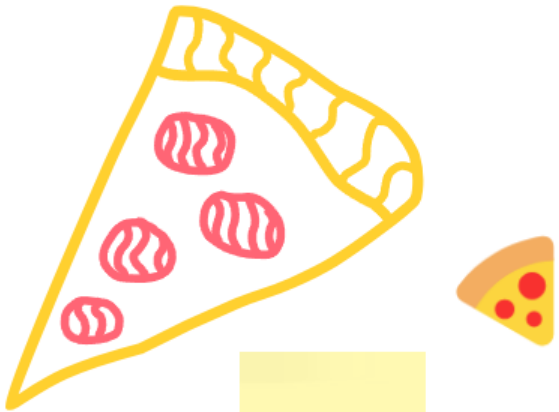
Обеспечивает чистое разделение интересов

Ускоряет разработку, благодаря использованию шаблонов проектирования

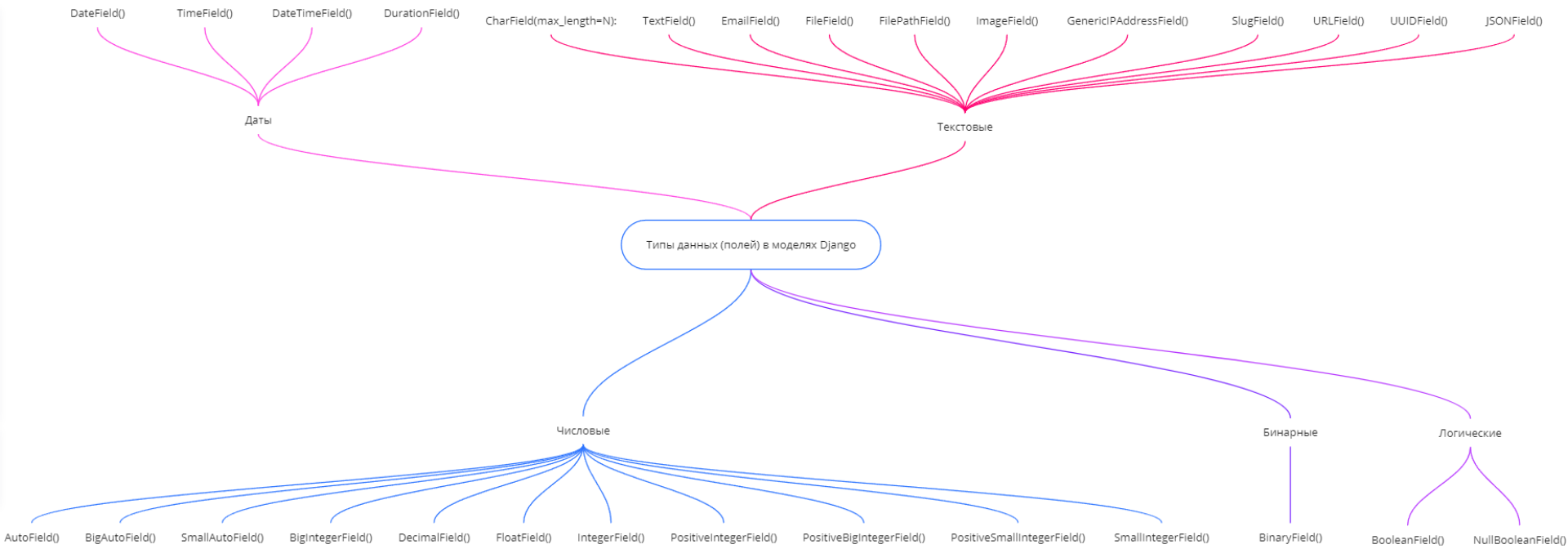
Не использует состояние представления или серверные формы

Поддерживает тестирование на основе разработки (TDD)

MVC - это архитектурный паттерн, позволяющий независимо друг от друга менять frontend (т.е. дизайн приложения) и backend (логику приложения)



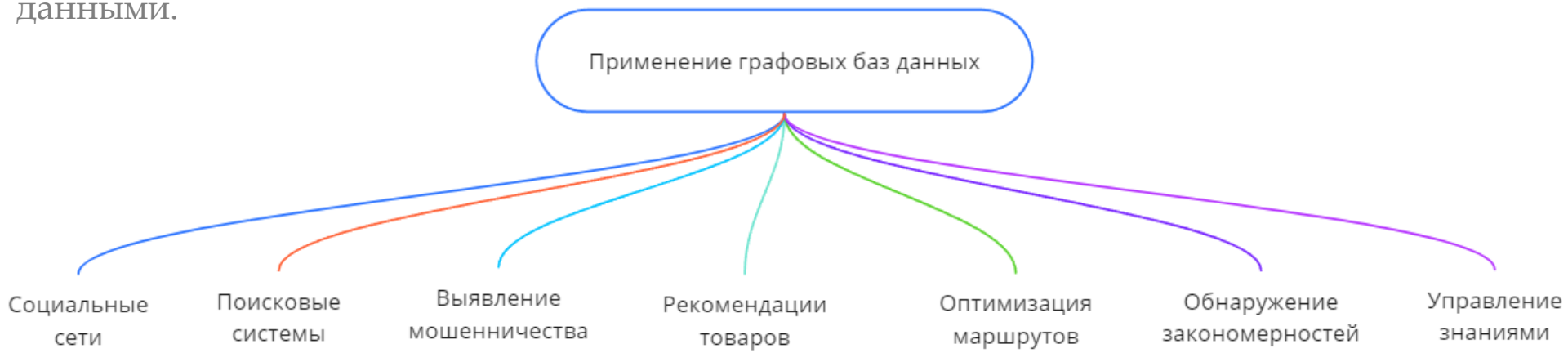
Типы данных (полей) в Django

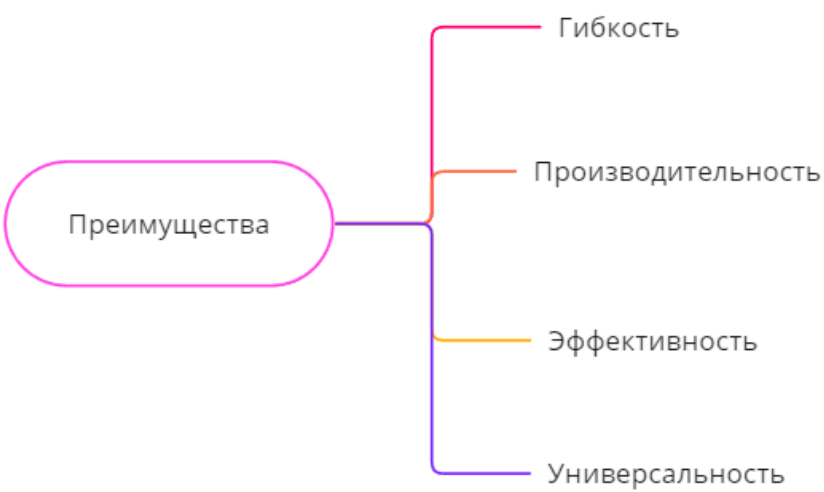


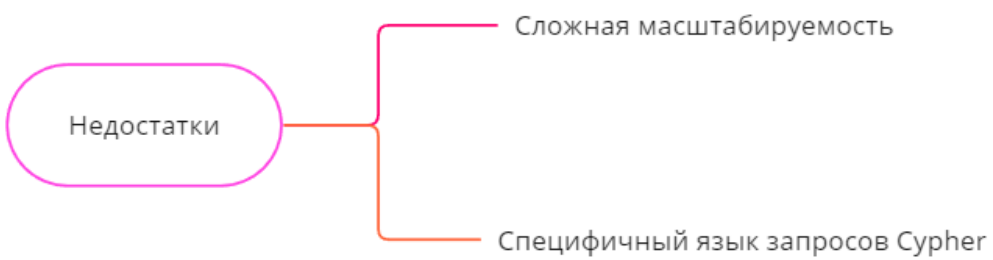
ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Что это такое, сферы применения, преимущества и недостатки

Графовые базы данных – это нереляционные системы (NoSQL), которые определяют корреляции между сложно взаимосвязанными сущностями. Такая структура позволяет обойти ограничения реляционных БД и уделяет больше внимания отношениям между данными.

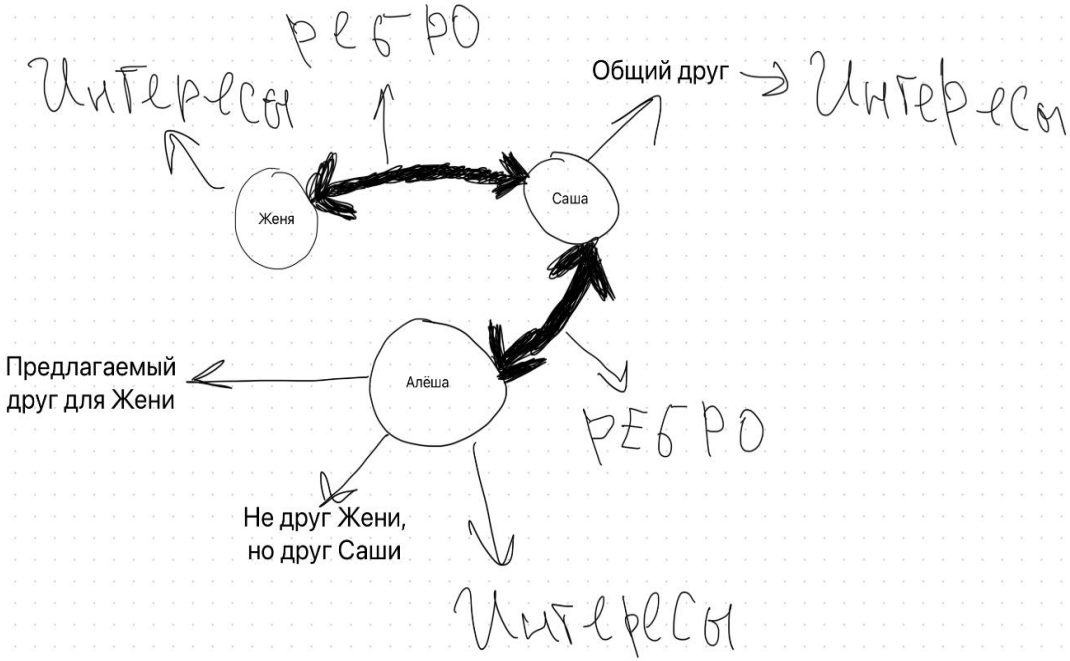






ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Определения, теоретический и практический пример



Вершина – точка в графе, отдельный объект. При этом для топологической модели графа не имеет значения координата вершины, её расположение, цвет, вкус, размер; однако при решении некоторых задачах вершины могут раскрашиваться в разные цвета или сохранять числовые значения.

Ребро – неупорядоченная пара связанных друг с другом двух вершин (в нашем случае – это пары «Женя – Саша» и «Алёша и Саша». Эти вершины называются концевыми точками или концами ребра. При этом важен сам факт наличия связи. Соответственно, каким именно образом осуществляется эта связь и по какому пути – не имеет значения; однако рёбрам может быть присвоен “вес”, что позволит говорить о “нагруженном графе”. Когда есть у ребра есть вес, мы можем решать задачи оптимизации, используемые как раз-таки в ГИСах.

Инцидентность – вершина и ребро называются инцидентными, если вершина является для этого ребра концевой. Обратите внимание, что термин “инцидентность” применим только к вершине и ребру.

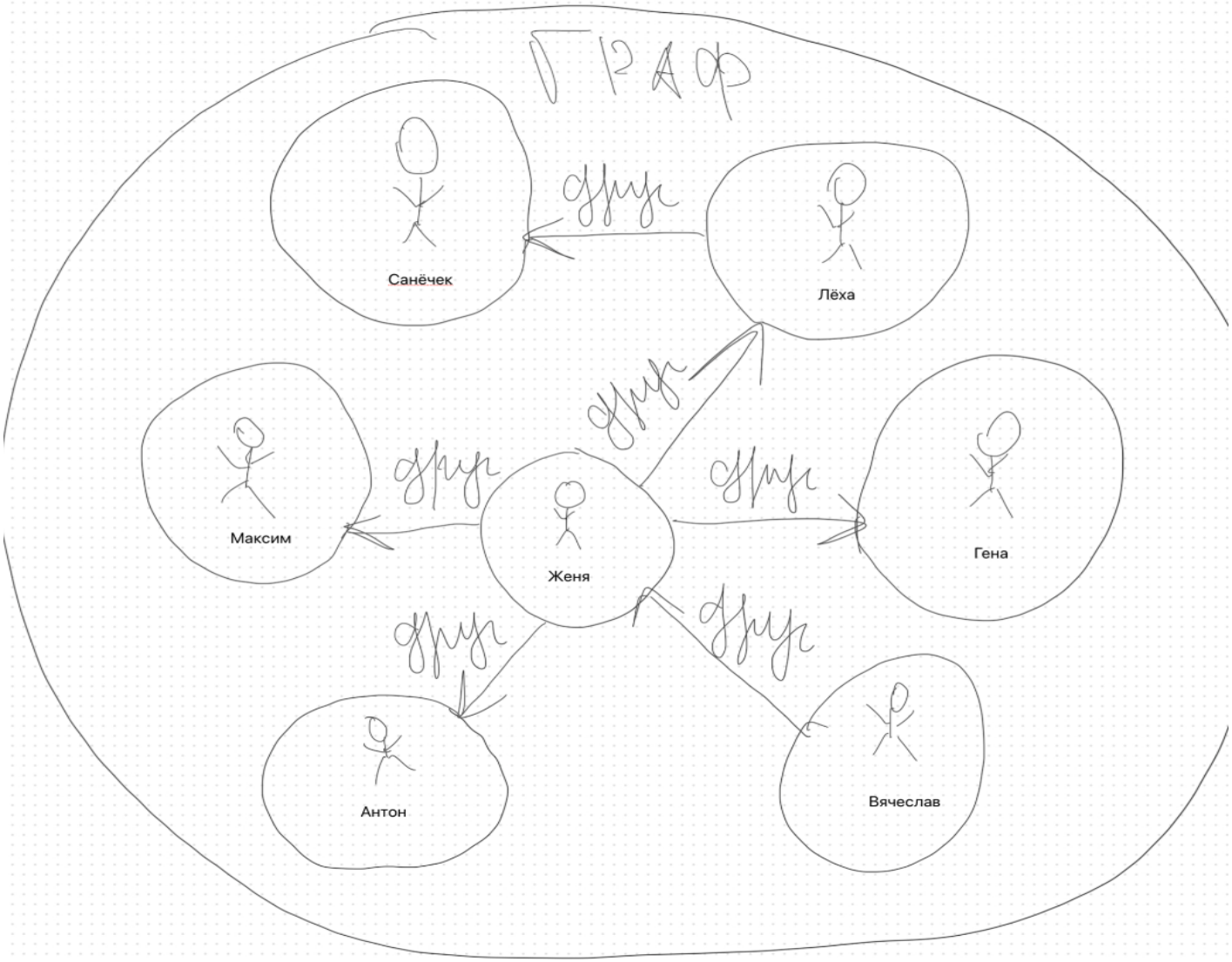
Смежность вершин – две вершины называются смежными, если они инцидентны одному ребру.

Смежность рёбер – два ребра называются смежными, если они инцидентны одной вершине.

Говоря проще: две вершины смежные, если они соединены ребром, два ребра смежные – если они соединены вершиной.

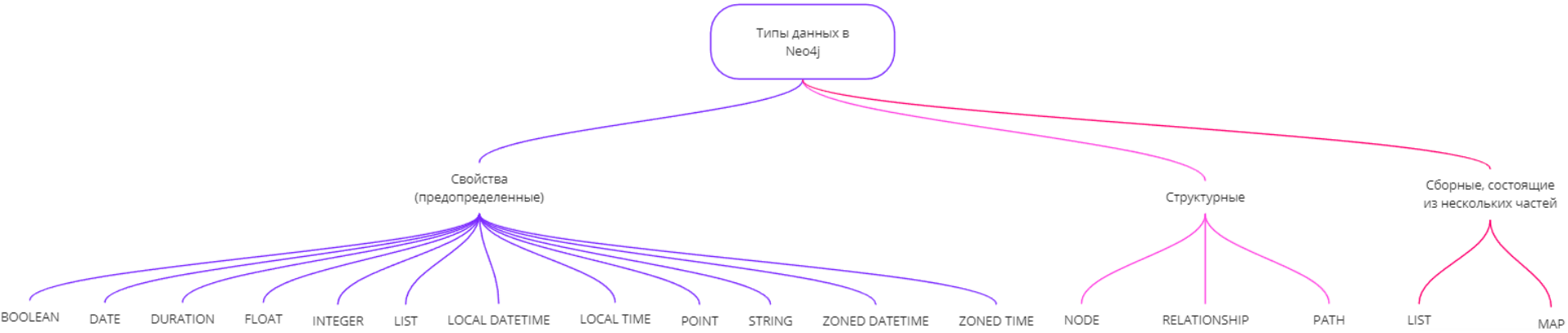
ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Теоретический и практический пример



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Основные типы данных в Neo4j



DURATION

```
UNWIND [
duration({days: 14, hours:16, minutes: 12}),
duration({months: 5, days: 1.5}),
duration({months: 0.75}),
duration({weeks: 2.5}),
duration({minutes: 1.5, seconds: 1, milliseconds: 123,
microseconds: 456, nanoseconds: 789}),
duration({minutes: 1.5, seconds: 1, nanoseconds:
123456789})
] AS aDuration
RETURN aDuration
```

Результат

aDuration
P14DT16H12M
P5M1DT12H
P22DT19H51M49.5S
P17DT12H
PT1M31.123456789S
PT1M31.123456789S

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Синтаксис в Neo4j. Язык запросов Cypher

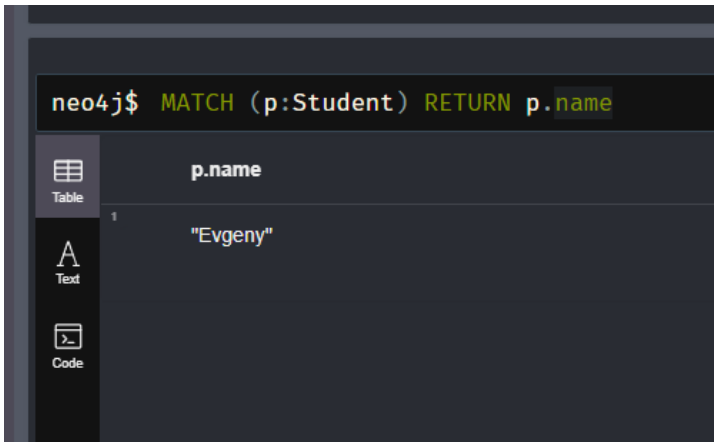
1. Создание узла

Запрос

CREATE (u1:Student {name: "Evgeny", group: "MO23-1m", from: "Moscow"})

Результат

MATCH (p:Student) RETURN p.name



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

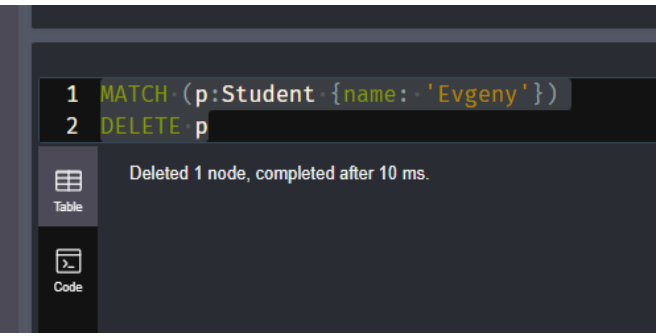
Синтаксис в Neo4j. Язык запросов Cypher

2. Удаление узла

Запрос

```
MATCH (p:Student {name: 'Evgeny'})
DELETE p
```

Результат



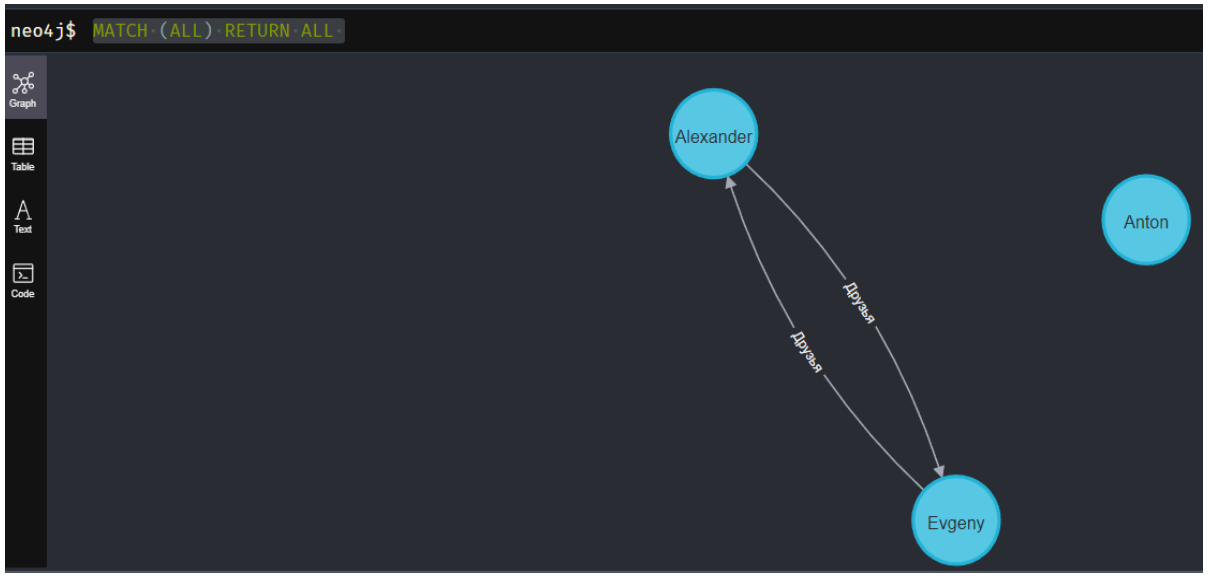
3. Создание нескольких узлов

```
CREATE (u1:Student {name: "Evgeny", group: "MO23-1m", from: "Moscow"}),(u2:Student {name: "Anton", group: "MO23-1m", from: "Moscow"}),(u3:Student {name: "Alexander", group: "MO23-1m", from: "Moscow"})
```

4. Создание связности узлов

```
MATCH (p:Student) WHERE p.name = "Evgeny"
MATCH (f:Student) WHERE f.name = "Alexander"
CREATE (p)-[:Друзья]->(f),
  (f)-[:Друзья]->(p)
```

Результат: MATCH (ALL) RETURN ALL



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

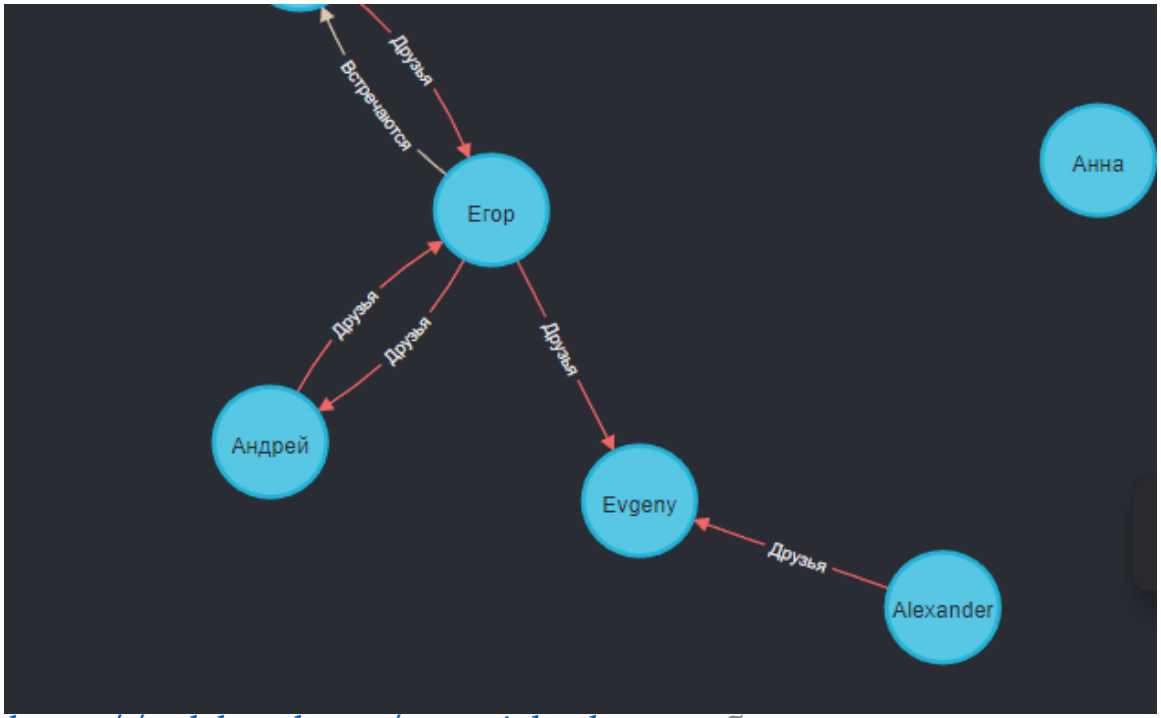
Синтаксис в Neo4j. Язык запросов Cypher

5. Удаление связей

Удаление связей «Друзья» у узла «Евгений»

```
MATCH (n:Student {name: 'Evgeny'})-
[r:Друзья]->()
DELETE r
```

Результат: MATCH (ALL) RETURN ALL



По итогу у Евгения пропали все исходящие связи.

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

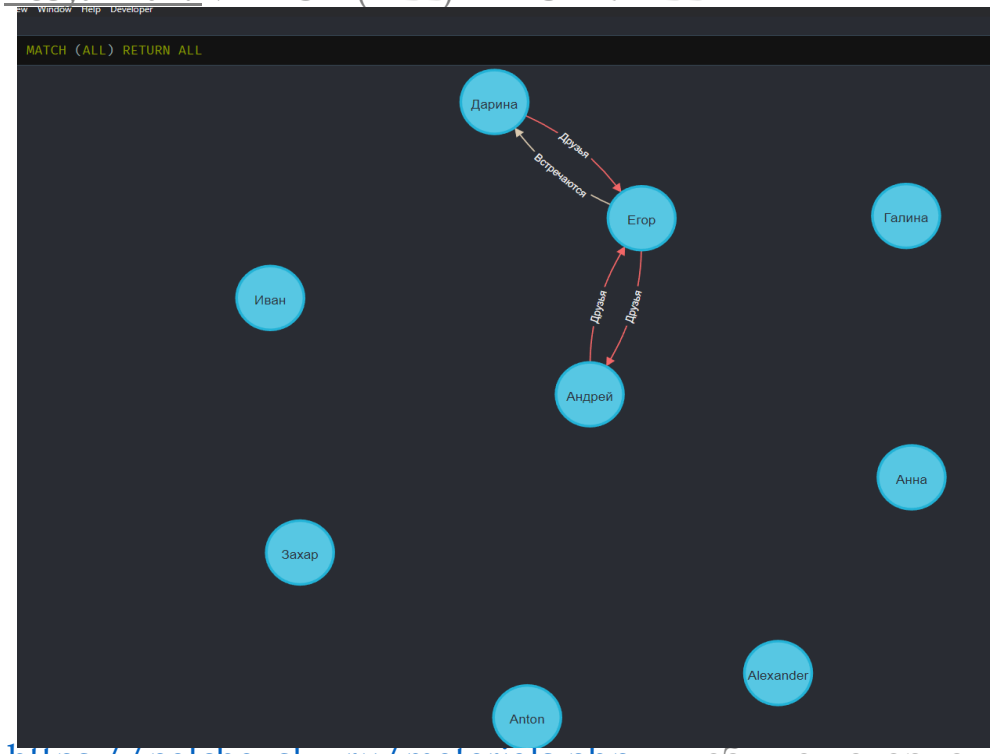
Синтаксис в Neo4j. Язык запросов Cypher

6. Удаление узла со связями

Удаление узла «Евгений» со всеми его связями

```
MATCH (p:Student {name: 'Evgeny'})
DETACH DELETE p
```

Результат: MATCH (ALL) RETURN ALL



По итогу узел «Евгений» был удален.

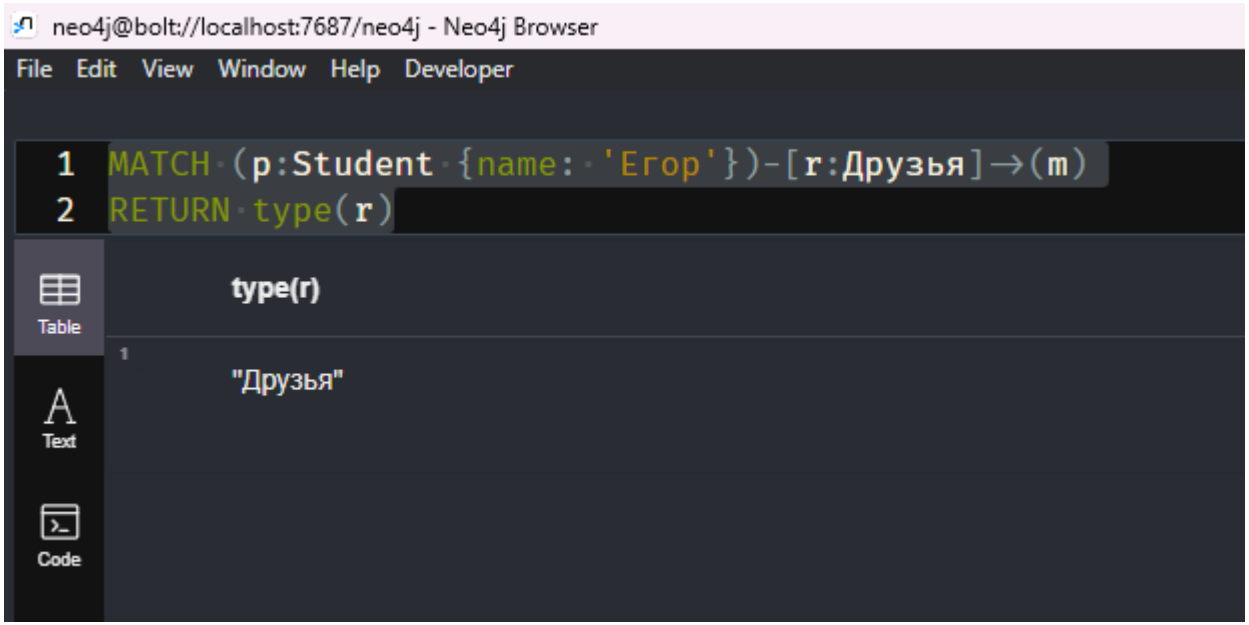
Синтаксис в Neo4j. Язык запросов Cypher

7. Возвращение типа связи

Удаление узла «Евгений» со всеми его связями

```
MATCH (p:Student {name: 'Егор'})-[r:Друзья]->(m)
RETURN type(r)
```

Результат:



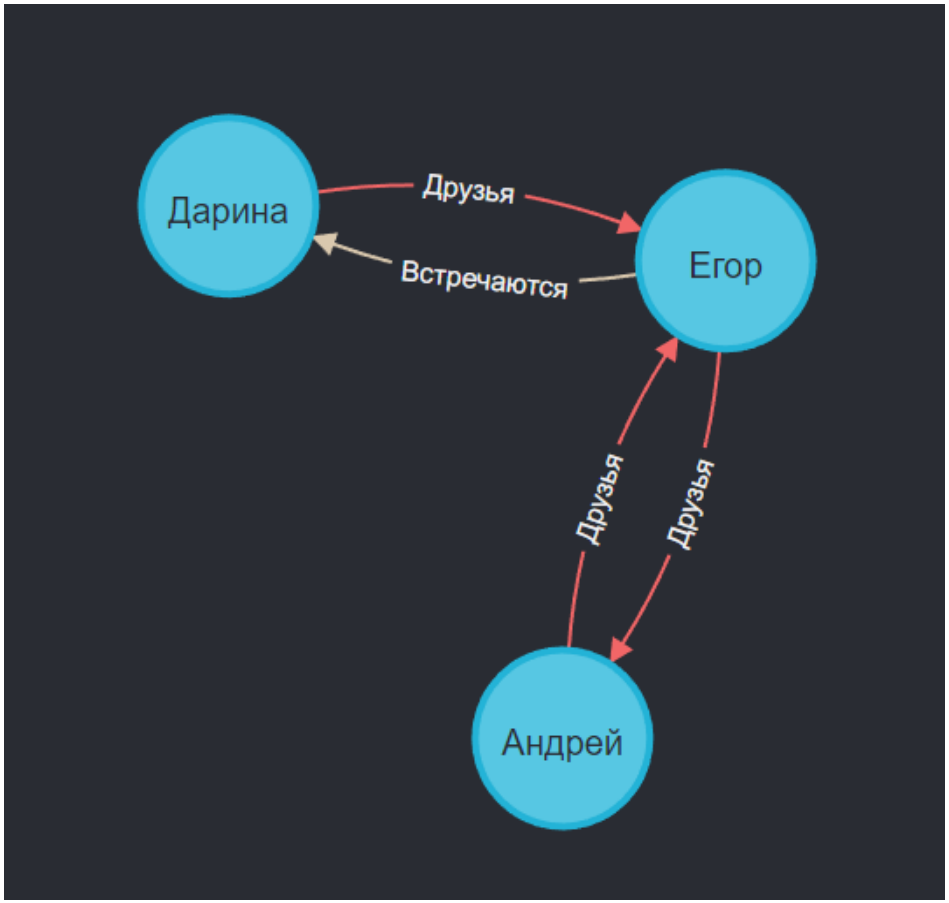
ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Синтаксис в Neo4j. Язык запросов Cypher

7. Возвращение узлов, связей и возможных путей между ними

```
MATCH (p:Student {name: 'Егор'})-[r]->(m)
RETURN *
```

Результат:

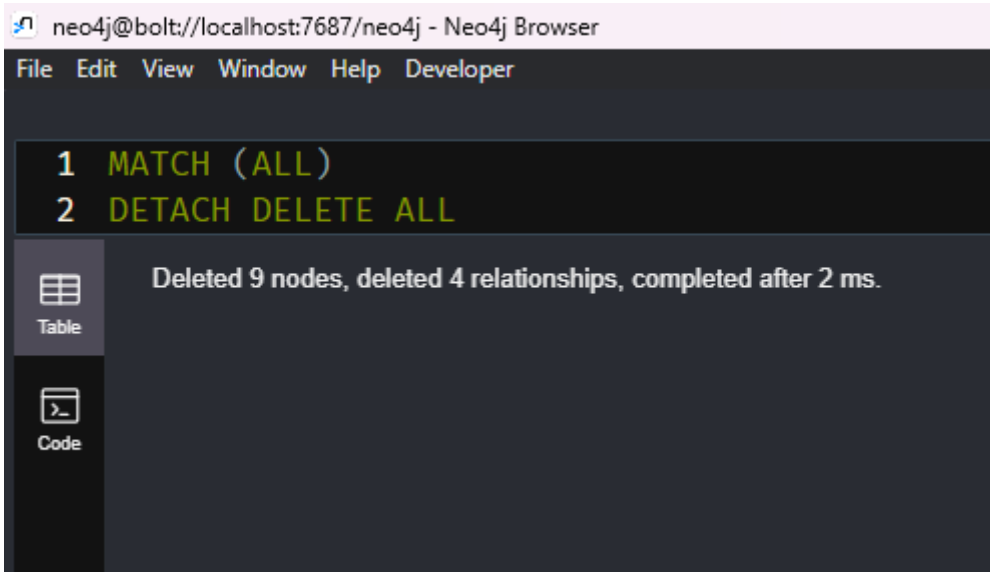


Синтаксис в Neo4j. Язык запросов Cypher

7. Возвращение узлов, связей и возможных путей между ними

MATCH (ALL)
DETACH DELETE ALL

Результат:



ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Опросник

1. Отличие SQL и NoSQL баз данных?
2. Что такое MongoDB?
3. Основные преимущества MongoDB?
4. К какому типу NoSQL-баз данных относится MongoDB?
5. Чем MongoDB отличается от MySQL?
6. Синтаксис поиска и сортировки в MongoDB?
7. Создание узла в Neo4j?
8. Создание связей между узлами в Neo4j?

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

ЗАДАЧИ ТОЛЬКО ДЛЯ ГРУПП **МО23-1М** И **ДПИ23-1М**

Семинарские занятия №1-2

Задача №1

1. При реализации проекта в обязательном порядке применяем диаграмму Ганта (можно использовать любой софт для ее реализации) с каскадной моделью жизненного цикла ПО. Должны быть задействованы как минимум следующие поля: «Задача», «Дата начала», «Дата завершения», «Исполнитель», «Статус». Пример представлен на рисунке:

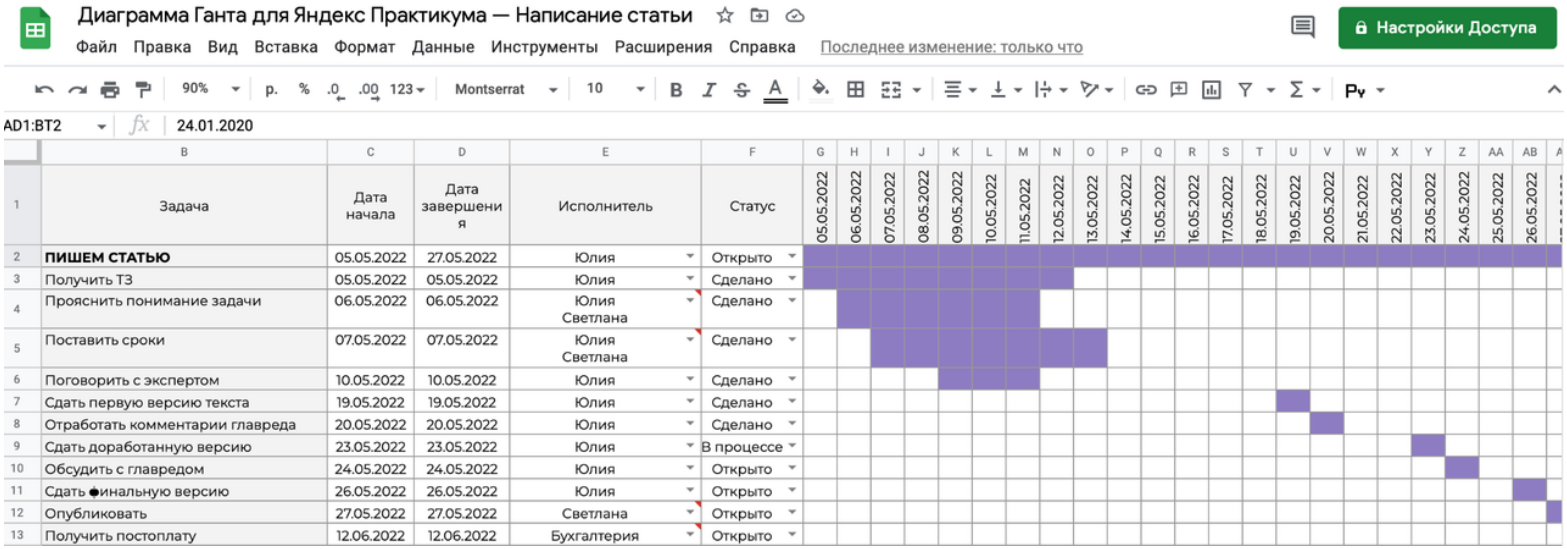


Диаграмма Ганта для Яндекс Практикума — Написание статьи

Файл Правка Вид Вставка Формат Данные Инструменты Расширения Справка Последнее изменение: только что

Настройки Доступа

AD1:BT2 | 24.01.2020

	Задача	Дата начала	Дата завершения	Исполнитель	Статус
2	**ПИШЕМ СТАТЬЮ**	05.05.2022	27.05.2022	Юлия	Открыто
3	Получить ТЗ	05.05.2022	05.05.2022	Юлия	Сделано
4	Прояснить понимание задачи	06.05.2022	06.05.2022	Юлия Светлана	Сделано
5	Поставить сроки	07.05.2022	07.05.2022	Юлия Светлана	Сделано
6	Поговорить с экспертом	10.05.2022	10.05.2022	Юлия	Сделано
7	Сдать первую версию текста	19.05.2022	19.05.2022	Юлия	Сделано
8	Отработать комментарии главреда	20.05.2022	20.05.2022	Юлия	Сделано
9	Сдать доработанную версию	23.05.2022	23.05.2022	Юлия	В процессе
10	Обсудить с главредом	24.05.2022	24.05.2022	Юлия	Открыто
11	Сдать финальную версию	26.05.2022	26.05.2022	Юлия	Открыто
12	Опубликовать	27.05.2022	27.05.2022	Светлана	Открыто
13	Получить постоплату	12.06.2022	12.06.2022	Бухгалтерия	Открыто

2. Разработать и реализовать тематический блог на фреймворке Django согласно видеолекции №1 (часть 2). Блог должен соответствовать тематике одной из магистерских диссертаций вашей команды. Т.е. берем тему магистерской диссертации любого члена команды и на эту тематику реализовываем блог с функционалом, представленным в лекции №1 (часть 2). Весь проект должен быть запущен либо в одном докере, как это показано в видеолекции №4 (https://youtu.be/VnouL-SQIjU?si=8GSaNwQ5lXA95_C8), либо в Docker Compose, что предпочтительнее, безопаснее, удобнее и быстрее. Обязательно должен присутствовать контроль версий Git.

https://palchevsky.ru/materials.php - учебные материалы и рейтинги

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Семинарские занятия №3-4. Контрольная работа №1

Задача №2

1. Добавить в блог следующие функции: пагинацию страниц, регистрацию и авторизацию, а также автоматический переводчик (с английского на русский и с русского на английский).
2. Добавить текущие функции и их реализацию в диаграмму Ганта в виде задач.

Семинарские занятия №5-6

Задача №3

1. Необходимо модернизировать авторизацию и регистрацию: должны открываться не по ссылке, а в модальном окне.
2. Добавить текущие функции и их реализацию в диаграмму Ганта в виде задач.

Семинарские занятия №7-8. Контрольная работа №2

Задача №4

1. На основе любой библиотеки машинного и глубокого обучения реализовать один из нижепредставленных методов прогнозирования:
а) полиномиальная регрессия;
б) градиентный бустинг;
в) рекуррентная нейронная сеть.
Можно использовать любой подходящий датасет:
https://www.kaggle.com/datasets
Исходные данные для обучения, а также прогнозные значения должны храниться в любой реляционной/нереляционной СУБД (на выбор разработчика).

2. Создать отдельную статичную страницу, на которой должны быть следующие функции: кнопка обучения метода прогнозирования, кнопка запуска модели прогнозирования, вывод результатов из БД.

3. Добавить текущие функции и их реализацию в диаграмму Ганта в виде задач.

https://palchevsky.ru/materials.php - учебные материалы и рейтинги

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Семинарские занятия №9-10.

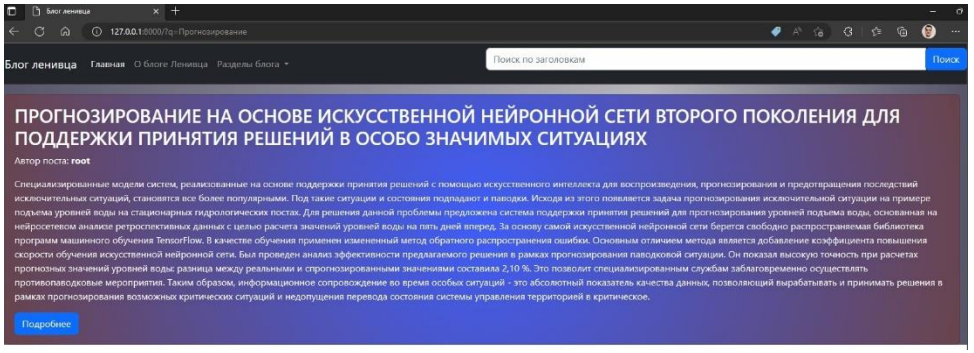
Задача №5

1. Реализовать поиск в Django как **в верхнем меню справа**, так и с расширенным поиском **на отдельной страничке**. Ссылка на страницу с поиском должна быть в верхнем меню.
2. Добавить ссылку на профиль. Внутри профиля необходимо добавить:

2.1. Поля: фамилия, имя и отчество.

2.2. Аватар с возможностью его обновления. При этом должна быть возможность загрузки аватара с компьютера.

3. Добавить текущие функции и их реализацию в диаграмму Ганта в виде задач.



Пример формы поиска в правом верхнем углу

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Семинарские занятия №11-14

Задача №6

Пройти бесплатный курс по Hadoop.
https://stepik.org/course/150/promo