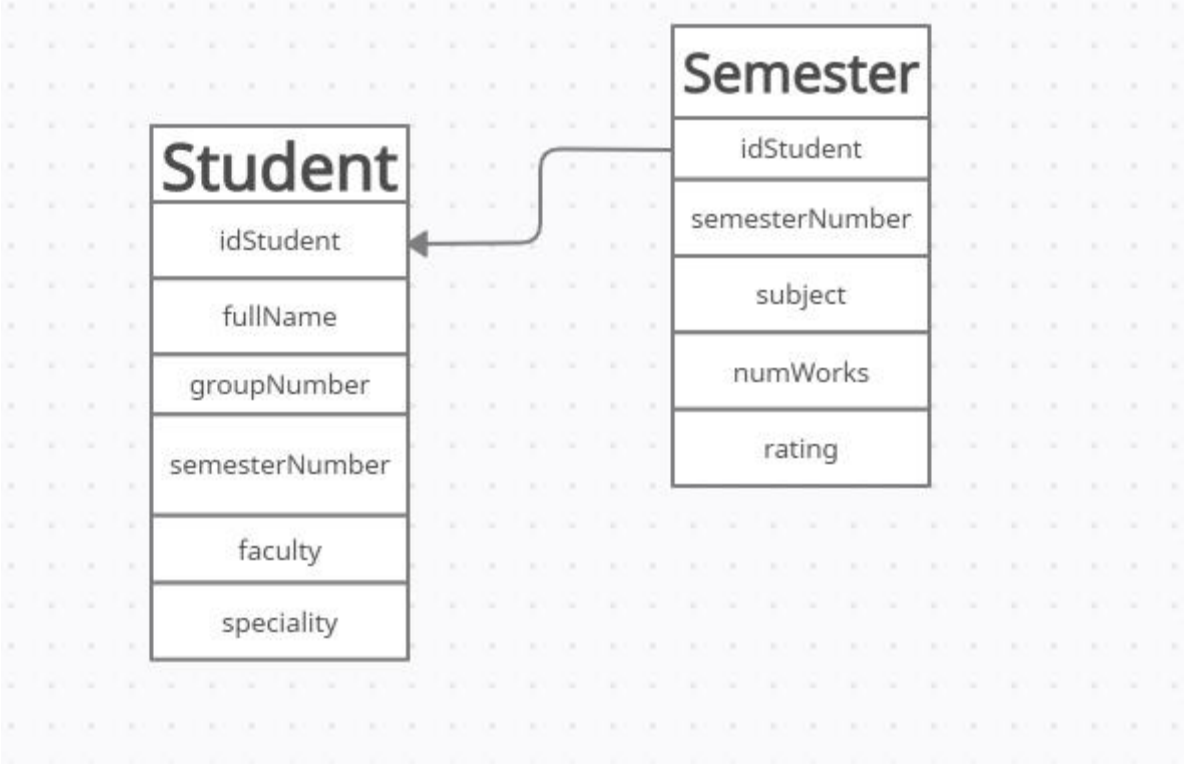
Модель данных

NoSQL модель данных

Графическое представление схемы БД:



Описание назначений коллекций, типов данных и сущностей:

Нереляционная база данных MongoDB состоит из 2 сущностей:

1. Student – данная сущность предназначается для хранения данных об аспирантах.
   1. “idStudent”, int – идентификационный номер аспиранта. 4B
   2. “fullName”, string – содержит фамилию, имя, отчество аспиранта. 50 * 2B = 100B
   3. “groupNumber”, int – номер группы. 4B
   4. “semesterNumber”, int – текущий семестр обучения. 4B
   5. “faculty”, string – факультет, к которому принадлежит аспирант. 50 * 2B = 100B
   6. “speciality”, string – специальность аспиранта. 50 * 2B = 100B

   Итого: 312B

2. Semester – данная сущность служит для описания деятельности в семестре.
    1. “idStudent”, int – идентификационный номер аспиранта. 4B
    2. “fullName”, string – содержит фамилию, имя, отчество аспиранта. 50 * 2B = 100B
    3. “semesterNumber”, int – номер текущего семестра аспиранта. 4B
    4. “subject”, string – название показателя. 50 * 2B = 100B
    5. numWorks, int – количество работ для показателя. 4B
    6. rating, int – рекомендуемая оценка за семестр. 4B

    Итого: 216B

Оценка объёма информации:

“Чистый объём”:

Пусть N – количество аспирантов.

В результате чистый объём будет равен N * (312 + 216) = N * 528B.

“Фактический объём”:

Пусть N – количество аспирантов.

В результате фактический объём будет равен N * 540B.

Избыточность модели: (A * 540B) / (A * 520B)

Примеры запросов:

Добавление нового аспиранта:

```
db.students.insertOne(
        ‘idStudent’: 124
        ‘fulllName’: ‘Иванов Иван Иванович’
        ‘groupNumber’: 7392
        ‘semesterNumber’: 6
        ‘faculty’: ‘ФКТИ’
        ‘speciality’: ‘Примат’
)
```

Общее число аспирантов:

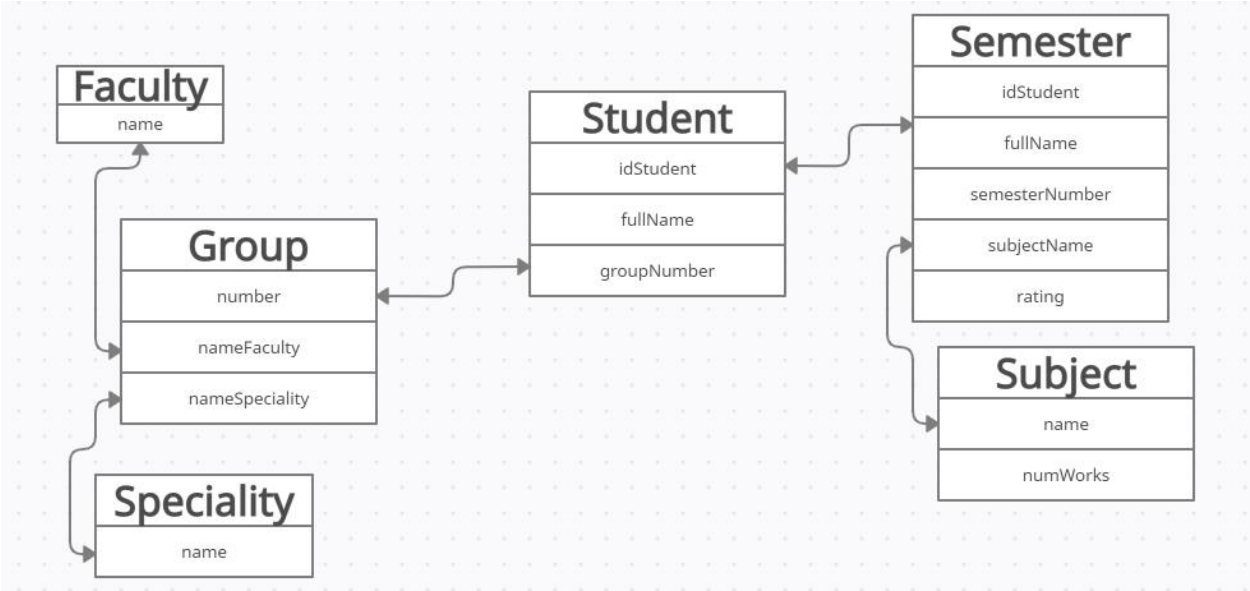
db.students.count()

Вывод всех данных аспирантов:

db.students.find({idStudent : 124})

SQL модель данных

Графическое представление



Описание назначений коллекций, типов данных и сущностей:

1. Student – данная сущность предназначается для хранения данных об аспирантах.
    1. “idStudent”, int – идентификационный номер аспиранта. 4B
    2. “fullName”, string – содержит фамилию, имя, отчество аспиранта. 50 * 2B = 100B
    3. “groupNumber”, int – номер группы. 4B

    Итого: 108B
2. Group – данная сущность предназначается для хранения данных о группе аспиранта.
    1. “number”, int – номер группы. 4B

2. “nameFaculty”, string – факультет, к которому принадлежит аспирант. 50 * 2B = 100B
3. “nameSpeciality”, string – специальность аспиранта. 50 * 2B = 100B

Итого: 204B

3. Faculty – сущность для хранения данных о факультете.
    1. “name”, string – факультет, к которому принадлежит аспирант. 50 * 2B = 100B

    Итого: 100B

4. Speciality – сущность для хранения данных о специальности.
    1. “speciality”, string – специальность аспиранта. 50 * 2B = 100B

    Итого: 100B

5. Semester – данная сущность служит для описания деятельности в семестре.
        1. “idStudent”, int – идентификационный номер аспиранта. 4B
        2. “fullName”, string – содержит фамилию, имя, отчество аспиранта. 50 * 2B = 100B
        3. “semesterNumber”, int – текущий семестр обучения. 4B
        4. “subjectName”, string – название показателя. 50 * 2B = 100B
        5. rating, int – рекомендуемая оценка за семестр. 4B

        Итого: 212B

6. Subject – сущность для хранения данных о предмете и количества работ за семестр

        1. “name”, string – название показателя. 50 * 2B = 100B
        2. numWorks, int – количество работ для показателя. 4B

        Итого: 104B

Оценка объёма информации:

Пусть N – количество аспирантов.

Тогда чистый объём будет равен N * (104 + 212 + 100 + 100 + 204 + 108) = N * 828B

Пусть у каждого аспиранта 3 предмета, тогда фактический объём данных будет равен N * (104 + 104 + 220 + 118 + 220 + 122) = N * 888

Избыточность модели: (N * 888)/ (N * 828B)

Примеры запросов:

Добавление нового аспиранта в таблицу:

INSERT INTO Student (idStudent, fullName, groupNumber) VALUES (185, ‘Петров Петр Петрович’, ‘3221’)

Найти данные о семестре студента по id.

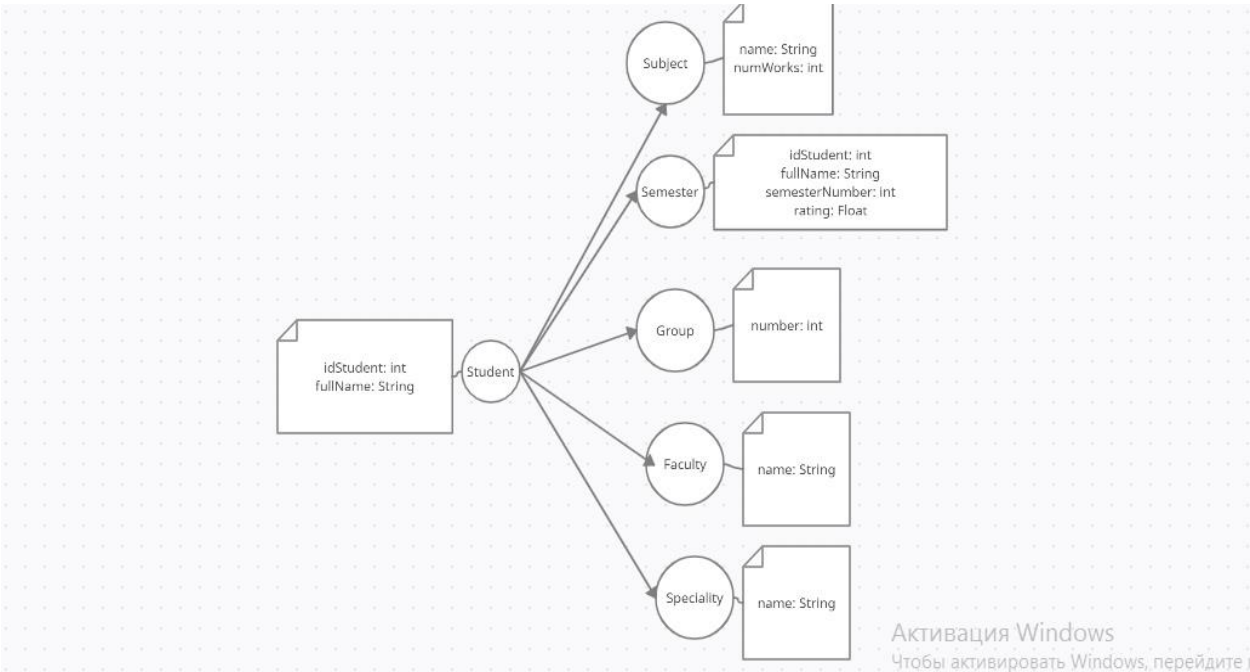
SELECT * FROM Semester INNER JOIN Student ON idStudent=164

Подсчёт аспирантов на 7 семестре.

SELECT COUNT(*) as count FROM db.Semester WHERE semesterNumber = 7

Neo4j модель данных

Графическое представление



Разработанная модель включает в себя следующие сущности: Student, Subject, Semester, Group, Faculty, Speciality.

Описание назначений коллекций, типов данных и сущностей**:**

1. Сущность Student содержит следующие свойства:

- idStudent – идентификационный номер аспиранта. Тип данных – int(4B);
- fullName - содержит фамилию, имя, отчество аспиранта. Тип данных – String(50 * 2B = 100B);

Итого: 104B

2. Сущность Subject содержит следующие свостйа:

- name - название показателя. Тип данных - String(50 * 2B = 100B);
- numWorks - количество работ для показателя. Тип данных – int(4B);

Итого: 104B

3. Сущность Semester содержит следующие свойства:

- idStudent – идентификационный номер аспиранта. Тип данных – int(4B);
- fullName - содержит фамилию, имя, отчество аспиранта. Тип данных – String(50 * 2B = 100B);
- semesterNumber - текущий семестр обучения. Тип данных – int(4B);
- rating - рекомендуемая оценка за семестр. Тип данных - int(4B);

Итого: 112B

4. Сущность Group содержит следующие свойства:

- number - номер группы. Тип данных – int(4B);

Итого: 4B

5. Сущность Faculty содержит следующие свойства:

- name - факультет, к которому принадлежит аспирант. Тип данных - String(50 * 2B = 100B);

Итого: 100B

6. Сущность Speciality содержит следующие свойства:

- name - специальность аспиранта. Тип данных - String(50 * 2B = 100B);

Итого: 100B

Оценка объёма информации:

“Чистый объём”:

Пусть N – количество аспирантов. Тогда чистый объём будет равен N * 524.

В свою очередь фактический объём будет равен N * 5 * 34 * 524 = N * 89080.

Избыточность модели: (N * 89080) / (N * 524)

Примеры запросов:

Вывод аспирантов из одной группы:

```
MATCH (id:Group)-[:Is]->(Kind {number: “7382”})
RETURN id
```

Выбор конкретного студента:

```
MATCH (a:Student {idStudent: 325})
RETURN a
```

Сравнение MongoDB и SQL моделей данных

Запросы на языке SQL являются более громоздкими и сложными, чем на языке MongoDB и Neo4j. В реализации нереляционной модели данных MongoDB мы создали в разы меньше сущностей и связей между ними, чем при использовании SQL и Neo4j модели.

“Чистый” объём памяти необходимый для нереляционной модели Neo4j оказался меньше, чем для MongoDB и SQL, однако данная модель будет отличаться наибольшей избыточностью.

Исходя из вышеперечисленного, можем сделать вывод, что для поставленной для проекта задачи, MongoDB подходит больше.