

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВОЙ ПРОЕКТ
по дисциплине «Введение в нереляционные базы данных»
Тема: Создание системы учета успеваемости студентов

Студент гр. 6304

Виноградов К.А.

Преподаватель

Заславский М.М.

Санкт-Петербург

2019

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ

Студент Виноградов К.А.

Группа 6304

Тема проекта: Создание системы учета успеваемости студентов

Исходные данные:

Требуется реализовать систему учета оценок (электронный журнал) для студентов вузов

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарии использования»

«Модель данных»

«Заключение»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студент

Виноградов К.А.

Преподаватель

Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания приложения для учета оценок студентов, основанного на базе данных MongoDB. Разработка велась на стеке технологий Nodejs + Express + MongoDB. Для более удобной работы с MongoDB использовался ODM mongoose. Найти исходный код и всю дополнительную информацию можно по ссылке: <https://github.com/zoOm60rus/WebJournal>.

SUMMARY

During this course we should have developed an application. The one we chose was an electronic journal, the students statistic storage, built on a NoSQL type DB – MongoDB. We developed this application with NodeJS, Express and MongoDB technologies stack. For pleasant usage the mongoose, an ODM for MongoDB was chosen. You can find all the code and additional info following: <https://github.com/zoOm60rus/WebJournal>.

Оглавление

1. ВВЕДЕНИЕ.....	6
2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ.....	6
3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ.....	6
3.1. Макеты UI	6
3.2. Сценарии использования	8
4. МОДЕЛЬ ДАННЫХ	12
4.1. MongoDB.....	12
4.2. SQL – модель.....	16
4.3. Подсчет объемов данных моделей	17
4.4. Сравнение моделей.....	20
4.5. Недостатки и пути для улучшения полученного решения.....	21
ЗАКЛЮЧЕНИЕ	22
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	23

1. ВВЕДЕНИЕ

Цель работы – создать высокопроизводительное и удобное решение для учета успеваемости студентов.

Было решено разработать веб-приложение, которое позволит хранить в электронном виде оценки студентов по всем предметам в каждом семестре, а также по необходимости изменять или удалять их.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение, основанное на СУБД MongoDB, с возможностью хранения, удаления, редактирования и отображения хранимой в ней информации.

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

3.1. Макеты UI

1. Форма входа (Рис. 1).

Вход

Электронная почта

Пароль

Регистрация Войти

The image shows a login form with a light gray background. At the top, the word 'Вход' is displayed in a large, bold, black font. Below it, the label 'Электронная почта' is followed by a white text input field with a thin gray border. Underneath, the label 'Пароль' is followed by a white password input field with a thin gray border. At the bottom, there are two buttons: a blue text link 'Регистрация' on the left and a solid blue button 'Войти' on the right.

Рисунок 1 – Форма входа в систему

2. Страница поиска студентов (Рис. 2).

Find a student

Menu

Student list

	ФИО	Факультет	Курс	Специальность
1	test3 test3 test3	ФРТ	1	test
2	test3 test3 test3	ФРТ	1	test
3	test4 test4 test4	ФКТИ	3	
4	Михайлов Алексей Валерьевич	ФКТИ	2	
5	Кушнарв Матвей Павлович	ФКТИ	2	
6	a a a	ФКТИ	1	Программная инженерия
7	b b b	ФКТИ	1	Программная инженерия

Поиск

Введите ФИО

Выберите Факультет

Не выбрано

Выберите курс

Не выбрано

Выберите специальность

Не выбрано

Найти

Footer

Рисунок 2 – Страница поиска студентов

3. Страница профиля пользователя (Рис. 3).

Find a student

Browsersync: connected

Михайлов Алексей Валерьевич

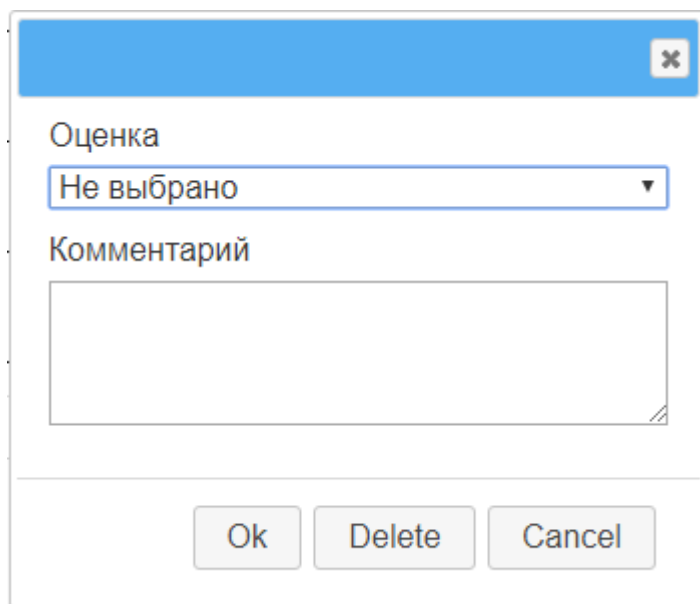
+ Семестр 4

Матанализ	1 2 3 1 1 3 3 3 +
Алгебра	1 2 3 2 3 1 2 +
Программирование	1 2 3 4 4 3 2 +
Экономика	1 2 3 4 н 3 2 +
Право	1 2 3 3 3 3 +

Footer

Рисунок 3 – Страница профиля пользователя

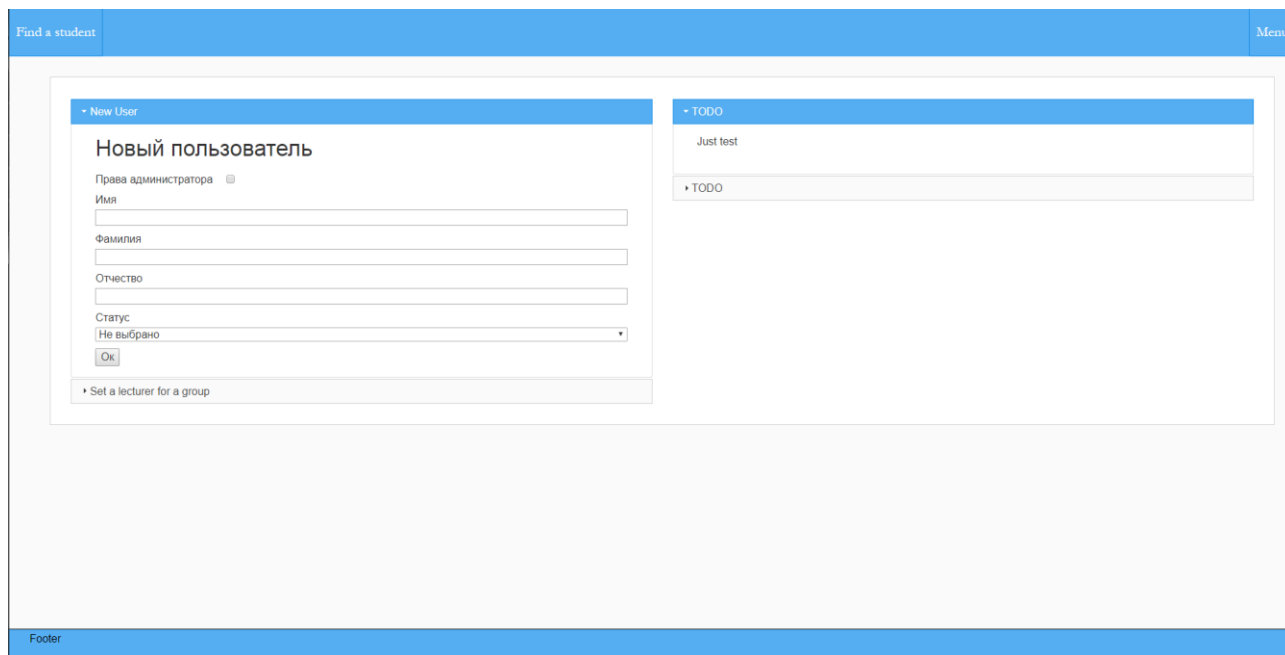
4. Форма добавления/изменения оценки (Рис. 4).



A modal dialog box with a blue header bar containing a close button (X). The form contains two main sections: 'Оценка' (Grade) with a dropdown menu currently showing 'Не выбрано' (Not selected), and 'Комментарий' (Comment) with a large text area. At the bottom, there are three buttons: 'Ok', 'Delete', and 'Cancel'.

Рисунок 4 – Форма добавления/изменения оценки

5. Страница администрирования (Рис. 5).



The administration page features a blue header with 'Find a student' on the left and 'Menu' on the right. The main content area is divided into two panels. The left panel, titled 'New User', contains a form for adding a new user with fields for 'Имя' (Name), 'Фамилия' (Surname), 'Отчество' (Patronymic), and 'Статус' (Status) with a dropdown menu. Below the form is an 'Ok' button and a link 'Set a lecturer for a group'. The right panel, titled 'TODO', contains a 'Just test' button and a 'TODO' link. A blue footer bar at the bottom contains the word 'Footer'.

Рисунок 5 – Страница администрирования

3.2. Сценарии использования

В системе предполагается существование трех ролей: студент, преподаватель и администратор. При этом один пользователь может совмещать несколько ролей (например студент-преподаватель или преподаватель-администратор). Для каждой из ролей описываются свои сценарии использования.

Данная версия проекта предполагает, что:

- Студент может:
 - посмотреть свои оценки и информацию о них

Для этого студент должен:

1. Войти на сайт через форму входа, используя правильную комбинацию email-пароль.
2. Перейти на страницу своего профиля одним из способов:
 - a. Выбрав пункт «Мой профиль» в меню в правом верхнем углу экрана.
 - b. Перейти на страницу поиска студентов, найти себя и кликнуть по своему ФИО.
 - c. Вбить в поисковой строке браузера
“http://_sitename_/students/_свой_ID_”
3. Развернуть нужный семестр, и, если необходимо, навести указатель на нужную оценку.

Примечание: студент может просматривать только свою страницу, на страницах остальных студентов отображается ошибка «Недостаточно прав»

- Преподаватель может:
 - посмотреть оценки студентов, в группах которых он установлен как преподаватель по предмету, по этому предмету и информацию о них

- изменить/удалить оценки студентов, в группах которых он установлен как преподаватель по предмету, по этому предмету и информацию о них
- добавить оценки студентам, в группах которых он установлен как преподаватель по предмету, по этому предмету и информацию о них

Для того, чтобы посмотреть оценки своего студента, преподаватель должен:

1. Войти на сайт через форму входа, используя правильную комбинацию email-пароль.
2. Перейти на страницу профиля студента одним из способов:
 - a. Перейти на страницу поиска студентов, найти необходимого и кликнуть по его ФИО.
 - b. Вбить в поисковой строке браузера “http://_sitename_/students/_ID_студента_”
3. Развернуть нужный семестр, и, если необходимо, навести указатель на нужную оценку.

Для того, чтобы изменить/удалить оценки своего студента, преподаватель должен:

1. Все действия сценария «Посмотреть оценку» роли Преподаватель.
2. Выбрать нужную оценку и кликнуть по ней.
3. Изменить (если нужно) данные в форме и кликнуть по кнопке подтверждения, если необходимо изменить, или кнопке удаления, если необходимо удалить.

Для того, чтобы добавить оценки своему студенту, преподаватель должен:

1. Все действия сценария «Посмотреть оценку» роли Преподаватель.
2. Выбрать нужный предмет и кликнуть по кнопке добавления оценки в строке предмета.
3. Заполнить форму и кликнуть по кнопке подтверждения.

Примечание: на страницах своих студентов преподаватель видит только предметы, которые ведет у этого студента, на страницах остальных студентов отображается ошибка «Недостаточно прав»

- Администратор может:
 - посмотреть оценки студентов
 - изменить/удалить оценки студентов
 - добавить оценки студентам
 - добавить пользователя в систему
 - добавить преподавателя группе

Для того, чтобы посмотреть оценки студентов администратор должен:

1. Все действия сценария «Посмотреть оценки» роли Преподаватель

Для того, чтобы изменить/удалить оценки студентов администратор должен:

1. Все действия сценария «Изменить/удалить оценки» роли Преподаватель

Для того, чтобы добавить оценки студентам администратор должен:

1. Все действия сценария «Добавить оценки» роли Преподаватель

Для того, чтобы добавить пользователя администратор должен:

1. Войти на сайт через форму входа, используя правильную комбинацию email-пароль.
2. Перейти на страницу администрирования одним из способов:
 - а. Кликнуть по пункту «Администрирование» в меню в правом верхнем углу экрана
 - б. Вбить адрес “http://_sitename/administration” в поисковой строке браузера
3. Развернуть форму добавления пользователя, заполнить ее и кликнуть по кнопке подтверждения.

Для того, чтобы добавить преподавателя группе администратор должен:

1. Выполнить пункты 1 и 2 сценария «Добавить пользователя» роли Администратор.
2. Развернуть форму добавления преподавателя группе, заполнить ее и кликнуть по кнопке подтверждения.

Примечание: администратор может выполнять все действия и просматривать все страницы без ограничений.

4. МОДЕЛЬ ДАННЫХ

4.1. MongoDB

Так как разработка велась с использованием ODM Mongoose, все документы, а также запросы будут предъявлены в его нотации.

Отдельными документами являются: пользователь, факультет, специализация, группа, кафедра. Документы студент и преподаватель являются поддокументами документа пользователь. Документ предмет является поддокументом документа студент. Документ оценка является поддокументом документа предмет.

Модель документа пользователь:

```
{
  name: String,
  surname: String,
  patronymic: String,

  email: {
    type: String,
    required: true
  },
  password: {
    type: String,
    required: true
  },
  isAdmin: {
    type: Boolean,
    required: true
  },
  studentProfile: {
    semesters:
      [
        {
          number: String,
          subjects:
            [
              {
                name: String,
                marks:
                  [
                    {
                      value: String,
                      date: {
                        type: Date,
                        default: Date.now
                      },
                      lastUpdated: {
                        type: Date,
                        default: Date.now
                      },
                      comments: String
                    }
                  ]
                }
            ]
          }
        ]
      },
    fakult: String,
    spec: String,
    group: String,
```

```

    course: String
  },
  lecturerProfile: {
    department: String,
    subjects: [String],
    studInfo: [
      {
        group: String,
        subject: String
      }
    ]
  }
}

```

Модель документа факультет:

```

{
  name: String,
  specs: [String]
}

```

Модель документа специализация:

```

{
  name: String,
  groups: [String],
  eduInfo: [
    {
      semester: String,
      subjects: [String]
    }
  ]
}

```

Модель документа группа:

```

{
  name: String,
  spec: String,
  course: String,
  students: [{ type: Schema.Types.ObjectId, ref: 'Student' }],
  subjects: [
    {
      name: String,
      lecturers: [
        {
          type: Schema.Types.ObjectId,
          ref: 'Lecturer'
        }
      ]
    }
  ]
}

```

```
}
]
}
```

Модель документа кафедры:

```
{
  name: String,
  lecturers: [{ type: Schema.Types.ObjectId, ref: 'Lecturer' }]
}
```

Визуальное представление моделей и их связей (Рис. 6):

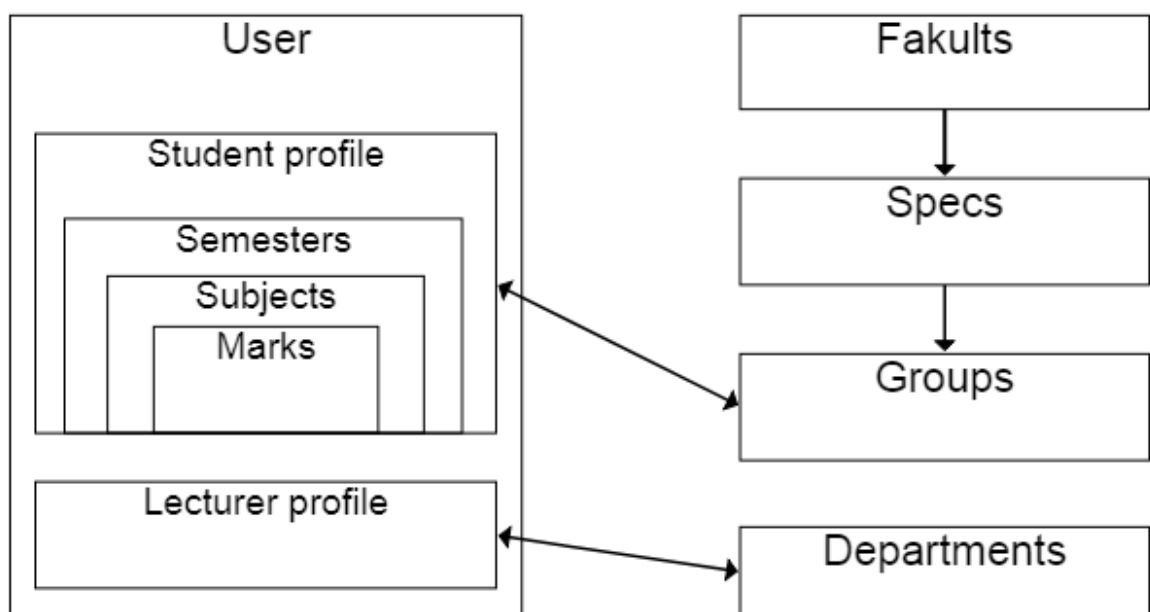


Рисунок 6 – Визуальное представление моделей и их связей

Примеры запросов:

Запрос создания нового пользователя с именем А, электронной почтой В, паролем С и статусом администратора:

```
models.User.create({
  name: 'A',
  email: 'B',
  password: 'C',
  isAdmin: true
});
```

Запрос поиска пользователя с именем А:

```
models.User.findOne({ name: 'A' }, (err, data) => {  
  });
```

Запрос поиска пользователя с идентификатором А:

```
models.User.findById('A', (err, user) => {  
  });
```

Запрос удаления пользователя с идентификатором А:

```
models.User.findByIdAndDelete('A', (err, user) => {  
  });
```

4.2. SQL – модель

Вышеописанную модель данных также можно представить в виде реляционной модели (Рис. 7):

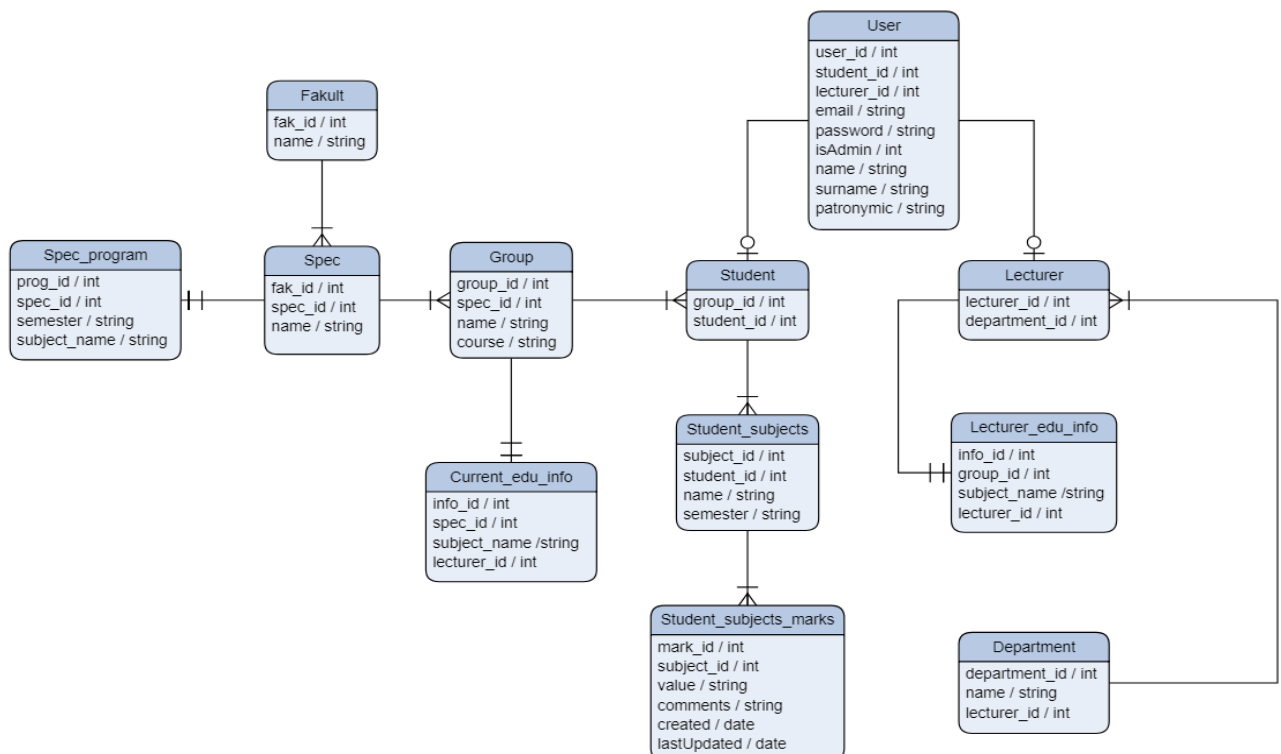


Рисунок 7 – Реляционное представление модели
Примеры запросов:

Запрос на создание нового пользователя:

```
INSERT INTO Users VALUES (email, password, name, isAdmin)
```


Запрос на удаление пользователя по id:

```
DELETE * FROM Users WHERE Users.id = id;
```

Запрос на поиск пользователя по id:

```
SELECT * FROM Users WHERE Users.id = id;
```

4.3. Подсчет объемов данных моделей

- MongoDB

Общий объем модели в MongoDB вычисляется по формуле: $U+F+S+G+D$, где U – суммарный объем документов пользователей, F – суммарный объем документов факультетов, S – суммарный объем документов специализаций, G – суммарный объем документов групп, D – суммарный объем документов кафедр.

U вычисляется по формуле: $St * St_av + Le * Le_av + Ad * Ad_av$, где St – средний объем документа студента, Le – средний объем документа преподавателя, Ad – средний объем документа администратора, а переменные с приставкой $_av$ – среднее количество пользователей данной роли.

Принимая во внимание то, что количество студентов во много раз больше количества преподавателей и администраторов, а также что профиль студента за счет информации об успеваемости так же больше, мы можем упростить эту формулу до $St * St_av$.

St вычисляется по формуле: $user_size_av + stud_size_av + marks_av * mark_size_av + sbjs_av * sems_av + sbjs_av * sbj_size_av * sems_av + sems_av * sem_size_av$, где переменная с приставкой $_av$ – это среднее количество документов с именем, а переменная с приставкой $_size_av$ – средний размер суммы полей документа без полей поддокументов.

F вычисляется по формуле: $fk_av * fk_size_av$, где переменная с приставкой $_av$ – это среднее количество документов с именем, а переменная с

приставкой `_size_av` – средний размер суммы полей документа без полей поддокументов.

S вычисляется по формуле: $sp_av * sp_size_av$, где переменная с приставкой `_av` – это среднее количество документов с именем, а переменная с приставкой `_size_av` – средний размер суммы полей документа без полей поддокументов.

G вычисляется по формуле: $gr_av * gr_size_av$, где переменная с приставкой `_av` – это среднее количество документов с именем, а переменная с приставкой `_size_av` – средний размер суммы полей документа без полей поддокументов.

D вычисляется по формуле: $dp_av * dp_size_av$, где переменная с приставкой `_av` – это среднее количество документов с именем, а переменная с приставкой `_size_av` – средний размер суммы полей документа без полей поддокументов.

Для удобства сведем все размеры к количеству полей, тогда:

$user_size_av = 5 \text{ string} + \text{bool} + \text{id}$,

$stud_size_av = 4 \text{ string} + \text{id}$,

$mark_size_av = 2 \text{ string} + 2 \text{ datetime} + \text{id}$

$sbj_size_av = \text{string} + \text{id}$

$sem_size_av = \text{string} + \text{id}$

$fk_size_av = \text{string} + sp_av * \text{string} + \text{id}$

$sp_size_av = \text{string} + gr_av * \text{string} + 12 * \text{string} + 12 * sbjs_av * \text{string} + \text{id}$

$gr_size_av = 3 \text{ string} + studs_per_gr_av * \text{id} + lects_per_sbj_av * \text{id} + sbj_av * \text{string} + \text{id}$

$dp_size_av = \text{string} + lects_per_dp_av * \text{id} + \text{id}$

Тогда итоговая формула:

$St_av * (5 \text{ string} + \text{bool} + \text{id} + 4 \text{ string} + \text{id} + (\text{marks_av} * sbjs_av * sems_av * (2 \text{ string} + 2 \text{ datetime} + \text{id})) + (sbjs_av * sems_av * (\text{string} + \text{id})) + sems_av * (\text{string} + \text{id})) + (fk_av * (\text{string} + sp_av * \text{string} + \text{id})) + (sp_av * (\text{string} + gr_av *$

$\text{string} + 12 * \text{string} + 12 * \text{sbjs_av} * \text{string} + \text{id})) + (\text{gr_av} * (3 * \text{string} + \text{studs_per_gr_av} * \text{id} + \text{lects_per_sbj_av} * \text{id} + \text{sbj_av} * \text{string} + \text{id})) + (\text{dp_av} * (\text{string} + \text{lects_per_dp_av} * \text{id} + \text{id})).$

Введем необходимые средние значения:

$\text{St_av} = 4000, \text{marks_av} = 7, \text{sbjs_av} = 6, \text{sems_av} = 4, \text{fk_av} = 5, \text{sp_av} = 6 * 5, \text{gr_av} = 6 * 5 * 2, \text{studs_per_gr_av} = 25, \text{lects_per_sbj_av} = 2, \text{dp_av} = 6, \text{lects_per_dp_av} = 20.$

При этом размер при средней длине строки в 30 символов типы будут иметь размер:

$\text{string} = 60 \text{ байт}, \text{id} = 12 \text{ байт}, \text{datetime} = 8 \text{ байт}, \text{bool} = 2 \text{ байта}.$

После подстановки:

$4000 * (5 * 60 + 2 + 12 + 4 * 60 + 12 + (7 * 6 * 4 * 2 * 60 + (2 * 60 + 2 * 8)) + (6 * 4 * (60 + 12)) + 4 * (60 + 12)) + (5 * (60 + 6 * 5 * 60 + 12)) + (6 * 5 * (60 + 6 * 5 * 2 * 60 + 12 * 60 + 12 * 6 * 60 + 12)) + (6 * 5 * 2 * (3 * 60 + 25 * 12 + 2 * 12 + 6 * 60 + 12)) + (6 * (60 + 20 * 12 + 12)) = 91837152 \text{ байта} \approx 87.6 \text{ Мбайта}$

- Реляционная БД

Общий объем в реляционной БД вычисляется как сумма средних размеров значений всех полей всех таблиц.

С помощью формулы это можно выразить как:

$\text{Fk_T_size} + \text{Sp_T_size} + \text{Prg_T_size} + \text{Gr_T_size} + \text{Cur_gr_inf_T_size} + \text{User_T_size} + \text{St_T_size} + \text{St_sbjs_T_size} + \text{St_marks_T_size} + \text{Lect_T_size} + \text{Lect_gr_inf_T_size} + \text{Dp_T_size}.$

Выразим эти переменные через поля и средние переменные

$\text{Fk_T_size} = \text{fk_av} * (\text{string} + \text{int}),$

$\text{Sp_T_size} = \text{sp_av} * (2 * \text{int} + \text{string}),$

$\text{Prg_T_size} = \text{sp_av} * (2 * \text{int} + 2 * \text{string}),$

$\text{Gr_T_size} = \text{gr_av} * (2 * \text{int} + 2 * \text{string}),$

$\text{Cur_gr_inf_T_size} = \text{gr_av} * (3 * \text{int} + \text{string}),$

$\text{User_T_size} = \text{St_av} * (4 * \text{int} + 5 * \text{string}),$

$$\begin{aligned} \text{St_T_size} &= \text{St_av} * 2 \text{ int}, \\ \text{St_sbjs_T_size} &= \text{St_av} * \text{sbjs_av} * (2 \text{ int} + 2 \text{ string}), \\ \text{St_marks_T_size} &= \text{St_av} * \text{sbjs_av} * \text{marks_av} * (2 \text{ int} + 2 \text{ string} + 2 \\ &\text{datetime}), \end{aligned}$$

Lect_T_size, Lect_gr_inf_T_size – не учитываются, аналогично нереляционной модели из-за малого влияния на результат.

$$\text{Dp_T_size} = \text{dp_av} * \text{lects_per_dp_av} * (2 \text{ int} + \text{string}).$$

При этом размер при средней длине строки в 30 символов типы будут иметь размер:

string = 60 байт, int = 4 байта, datetime = 8 байт.

После подстановки:

$$\begin{aligned} &5 * (60 + 4) + 6 * 5 * (2 * 4 + 60) + 6 * 5 * (2 * 4 + 2 * 60) + 6 * 5 * 2 * (2 * 4 \\ &+ 2 * 60) + 6 * 5 * 2 * (3 * 4 + 60) + 4000 * (4 * 4 + 5 * 60) + 4000 * 2 * 4 + 4000 * \\ &6 * (2 * 4 + 2 * 60) + 4000 * 6 * 7 * (2 * 4 + 2 * 60 + 2 * 8) + 6 * 20 * (2 * 4 + 60) = \\ &28586360 \text{ байт} \approx 27.3 \text{ Мбайта} \end{aligned}$$

4.4. Сравнение моделей

Исходя из полученных данных мы можем сделать вывод, что реляционная модель получилась примерно в 3(!) раза эффективнее по занимаемому объему, что можно объяснить более эффективной структурой, меньшим количеством и объемом полей, и, возможно, допущенными ошибками при подсчете.

Запросы примерно одинаковы по сложности – простые и однострочные при добавлении в один документ (таблицу) и многострочные при добавлении в несколько.

Однако плюсом использования mongoose является то, что он позволяет работать с документами как с объектами, что привычнее и удобнее тем программистам, которые привыкли к объектно-ориентированному подходу. К

тому же при использовании в данном проекте хоть и выявило неэффективность использования памяти, конечные величины настолько малы, что даже такая неэффективность не наложит больших затрат на оборудование.

4.5. Недостатки и пути для улучшения полученного решения

Оптимизация модели данных для более эффективного использования памяти.

В данный момент выбор студентов не совсем удобен. Для более удобного поиска студентов необходимо создать страницу групп.

Также необходимо доработать регистрацию и реализовать страницу настроек аккаунта для удобства пользователей.

ЗАКЛЮЧЕНИЕ

В ходе работы над проектом было разработано приложение учета успеваемости студентов, позволяющее хранить, редактировать и отображать информацию об их оценках.

Были изучены особенности и подходы при работе с нереляционными БД, в частности с MongoDB и его ODM mongoose.

Был проведен анализ и сравнение реализации данного проекта как на нереляционной модели данных, так и на реляционной. Были выделены достоинства и недостатки, пути и способы улучшения проекта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB: <https://docs.mongodb.com/manual/>
2. Документация NodeJS: <https://nodejs.org/ru/docs/>
3. Документация Express: <https://expressjs.com/ru/>
4. Документация mongoose: <https://mongoosejs.com/docs/>