МИНИСТЕРСТВОНАУКИИВЫСШЕГООБРАЗОВАНИЯРФ

**Московский авиационный институт**

**(национальный исследовательский университет)**

Институт№8

Компьютерные науки и прикладная математика

Кафедра 806 «Вычислительная математика и программирование»

Курсовой проект

по дисциплине «Базы Данных»

**Выполнил:** студент группы М8О-310Б-20

Кирсанов Григорий Романович

**Приняла:** Чумакова Екатерина Витальевна

**Оценка**: **Дата**:

Москва, 2022

# Содержание

[Содержание 2](#_Toc129014822)

[Введение 1](#_Toc129014823)

[1. Анализ предметной области и постановка задачи 1](#_Toc129014824)

[1.1. Описание предметной области и назначение системы 1](#_Toc129014825)

[1.2. Цели разработки и способы достижения 2](#_Toc129014826)

[1.3. Ожидаемые результаты 2](#_Toc129014827)

[1.4. Информационный образ системы 2](#_Toc129014828)

[1.5. Входные и выходные потоки 3](#_Toc129014829)

[1.5.1. Входные потоки 3](#_Toc129014830)

[1.6. Функциональный образ системы 4](#_Toc129014831)

[1.7. Требования и ограничения на использование системы 5](#_Toc129014832)

[1.8. Взаимодействие с другими программами 6](#_Toc129014833)

[2. Концептуальное и логическое проектирование базы данных 7](#_Toc129014834)

[2.1. Выделение информационных объектов и определение атрибутов объектов 7](#_Toc129014835)

[2.2. Разработка логической структуры базы данных 9](#_Toc129014836)

[2.3. Построение диаграммы сущность-связь 10](#_Toc129014837)

[2.3.1. Раскрытие связей многие ко многим 10](#_Toc129014838)

[2.3.2. Поддержка ограничений целостности, списки доменов атрибутов 11](#_Toc129014839)

[2.3.3. Нормализация объектной модели 12](#_Toc129014840)

[2.4. Выбор используемой СУБД 12](#_Toc129014841)

[3. Физическое проектирование базы данных 13](#_Toc129014842)

[3.1. Создание физической БД 13](#_Toc129014843)

[3.2. Описание запросов и процедур к БД 13](#_Toc129014844)

[3.2.2. Получение данных о студенте по части его фамилии 16](#_Toc129014845)

[3.2.3. Получение данных о расписании группы 18](#_Toc129014846)

[3.2.4. Получение данных о выплатах студентам больше заданной суммы 18](#_Toc129014847)

[3.2.5. Получение данных о всех изменениях таблицы «students» 19](#_Toc129014848)

[3.2.6. Вывод кафедры с количеством сотрудников 19](#_Toc129014849)

[3.2.7. Вывод расходов факультета на стипендии и мат. помощь 19](#_Toc129014850)

[3.3. Создания вспомогательных объектов базы данных 20](#_Toc129014851)

[3.3.1. Создание триггера на добавление оценки студенту 20](#_Toc129014852)

[3.3.2. Журналирование изменений в таблице «students» 20](#_Toc129014853)

[3.3.3. Индексы 21](#_Toc129014854)

[4. Разработка прикладной программы 22](#_Toc129014855)

[4.1. Основные аспекты приложения 22](#_Toc129014856)

[4.2. Поиск студентов 24](#_Toc129014857)

[4.3. Расписание 26](#_Toc129014858)

[4.4. Выплаты 26](#_Toc129014859)

[4.5. Журнал изменений таблицы «студенты» 27](#_Toc129014860)

[4.6. Вывод кафедр с количеством сотрудников 27](#_Toc129014861)

[4.7. Расходы факультетов 28](#_Toc129014862)

[4.8. Выставление оценок студентам 28](#_Toc129014863)

[Заключение 29](#_Toc129014864)

[Источники 30](#_Toc129014865)

[Приложение 30](#_Toc129014866)

[1. Разработка ролей базы данных 30](#_Toc129014867)

# Введение

Во всем мире организации накапливают или уже накопили в процессе своей административной и хозяйственной деятельности большие объемы данных, в том числе и в электронном виде. Эти коллекции данных хранят в себе большие потенциальные возможности по извлечению новой аналитической информации, на основе которой можно и необходимо строить стратегию организации, выявлять тенденции развития рынка, находить новые решения, улучшать внутренние процессы организации, ускоряя работу сотрудников и увеличивая их эффективность. хранения, упорядочения и анализа больших объемов информации предназначены комплексы средств, именуемых информационными системами. Одними из видов таких, сегодня уже автоматизированных информационных систем, являются базы данных, управляемые с помощью систем управления базами данных. Лидером на рынке СУБД на данный момент является PostgreSQL. PostgreSQL- свободная объектнореляционная система управления базами данных, существует в реализациях для множества UNIX-подобных систем в том числе Linux, MacOS и Windows. Сильными сторонами являются высокопроизводительные и надёжные механизмы транзакций, расширяемая система встроенных языков программирования, расширяемость, а также механизмы наследования.

# 1. Анализ предметной области и постановка задачи

## 1.1. Описание предметной области и назначение системы

Было принято решение разработать систему для «МАИ» позволяющую:

1. ускорить работу сотрудников разных отделов университета
2. автоматизировать валидацию введенных сотрудниками данных
3. обеспечить безопасность системы и контроль за изменениями данных

База данных создается для упрощения работы преподавателей и контролирующих органов (администрации).

Основными удобствами являются:

1. скорость проставления оценок студентам, и контроль за ними в случае спорных ситуаций
2. удобство доступа к расписанию занятий
3. контроль и сбор статистики для хозяйственной части университета
4. контроль за успеваемостью студентов

## 1.2. Цели разработки и способы достижения

Цели разработки системы:

1. упростить работу преподавателей
2. увеличить контроль за деятельностью университета
3. уменьшить загруженность рабочих мест и их размеры (перевод множества рабочих процессов из бумажного вида в электронный)
4. уменьшение бумажного документооборота

Способы достижения указанных целей:

1. анализ предметной области для выявления сущностей и атрибутов сущностей для хранения в базе данных
2. анализ процессов университета для определения доступа к хранимым данным и их представления
3. проанализировать связи между таблицами

## 1.3. Ожидаемые результаты

В результате работы ожидается:

1. уменьшение необходимости в бумажных документах
2. ускорение доступа к данным для всех участников программы (сотрудники, студенты, контролирующие органы)
3. экономия времени сотрудников, что ведет к увеличению финансовой эффективности университета

## 1.4. Информационный образ системы

Выделим необходимые сущности:

1. subject - предмет, преподаваемый в университете
2. faculty - факультет
3. st-group - учебная группа
4. department - кафедра
5. post - должность, занимаемая сотрудником
6. employee - сотрудник
7. schedule - учебное расписание
8. student - студент
9. payment - выплаты студентам
10. auditorium - аудитория
11. inventory - инвентарь для аудиторий (компьютеры, доски и др.)
12. statement - учебная ведомость

## 1.5. Входные и выходные потоки

### 1.5.1. Входные потоки

**Данные о студенте:**

1. Фамилия
2. Имя
3. Учебная группа

**Данные об оценках студентов:**

1. Преподаватель
2. Студент
3. Выставленная оценка (в пятибалльной шкале)
4. Дата выставления оценки

Данные об инвентаре

1. Название инвентаря
2. Закрепленная аудитория

Данные о расписании

1. Предмет
2. Учебная группа
3. Ведущий преподаватель
4. Дата проведения пары

Данные о сотруднике

1. Имя
2. Фамилия
3. Занимаемая должность
4. Кафедра, на которую устроен преподаватель

### 1.5.2. Выходные потоки

Данные о студенте

1. ФИО
2. Успеваемость студента
3. Учебная группа

Данные о расходах факультетов

1. Название факультета
2. Расходы

Данные о расходах на выплаты студентам

1. ФИО студента
2. Суммарная выплата за все время

Данные о расписании конкретной группы

1. ФИО Преподавателя
2. Дата проведения
3. Название предмета

Данные количестве сотрудников на кафедрах

1. Номер кафедры
2. Название кафедры
3. Количество сотрудников

## 1.6. Функциональный образ системы

Задачи, решаемые разрабатываемой системой:

1. Получение данных о студентах по фамилии или части фамилии и контроль за их успеваемостью
2. Получение данных о расписании группы по ее названию
3. Получение данных о выплатах студентам, с установкой минимального лимита по выплатам
4. Получение данных о изменения, вносимых в таблицу «students» для

обеспечения безопасности

1. Вывод кафедр в отсортированном порядке с указанием минимального

числа сотрудников

1. Вывод расходов всех факультетов на стипендии и мат. помощь

## 1.7. Требования и ограничения на использование системы

Выявление групп лиц, пользователей системы и назначена им прав доступа к данным. Скрипт на языке SQL приведен в приложении ниже. Сокращение прав доступа.

S - Select (право на чтение)

I - Insert (право на вставку данных)

U - Update (право на обновление данных)

D - Delete (право на удаление)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ресурс | Студент | Преподаватель | Администрация | Администрация |
|  |  |  | факультета | университета |
| schedule | S | S | SIU | SIU |
| statement | S | SIU | SIU | SIU |
| st-group | - | S | SIU | SIU |
| auditorium | - | S | SIU | SIU |
| subject | - | S | SIU | SIU |
| employee | - | - | SIU | SIU |
| student | - | - | SIU | SIU |
| employee | - | - | SIU | SIU |
| payment | S | - | SIU | SIU |
| faculty | - | - | S | SIU |
| department | - | - | S | SIU |
| post | - | - | S | SIU |
| inventory | - | - | S | SIU |

Таблица 1.1. Права доступа к ресурсам БД

## 1.8. Взаимодействие с другими программами

База данных, разрабатываемая для университета, является составной частью информационной системы и взаимодействует как с графическим интерфейсом, так и с другими ее частями.

# 2. Концептуальное и логическое проектирование базы данных

## 2.1. Выделение информационных объектов и определение атрибутов объектов

1. payment
   * + id: int, первичный ключ (уникальный идентификатор выплаты)
     + sum: int (сумма выплаты)
     + student-id: int, внешний ключ (идентификатор студента, кому была

произведена выплата)

1. statement
   * + id: int, первичный ключ (уникальный идентификатор выставленной оценки)
     + student-id: int, внешний ключ (идентификатор студента, кому была выставлена оценка)
     + employee-id: int, внешний ключ (идентификатор преподавателя, выставившего оценку)
     + estimation: int (выставленная оценка)
2. schedule
   * + id: int, первичный ключ (уникальный идентификатор проводимой пары)
     + subject-id: int, внешний ключ (идентификатор проводимого предмета) group-id: int, внешний ключ (идентификатор учебной группы) employee-id: int, внешний ключ (идентификатор сотрудника, проводящего пару)
     + curr-date: date (дата проставления оценки)
3. schedule
   * id: int, первичный ключ (уникальный идентификатор проводимой пары)
   * subject-id: int, внешний ключ (идентификатор проводимого предмета)
   * group-id: int, внешний ключ (идентификатор учебной группы)
   * employee-id: int, внешний ключ (идентификатор сотрудника, проводящего пару)
   * event-date: date (дата проведения занятий)
4. subject
   * id: int, первичный ключ (уникальный идентификатор предмета)
   * name: varchar (Название предмета)
5. inventory
   * id: int, первичный ключ (уникальный идентификатор инвентаря)
   * name: varchar (Название инвентаря)
   * auditorium-id: int, внешний ключ (идентификатор аудитории, закрепленной за предметом инвентаря)
6. department
   * id: int, первичный ключ (уникальный идентификатор кафедры) name: varchar (название кафедры)

number: int (номер кафедры)

* + faculty-id: int, внешний ключ (название факультета на котором на-

ходится кафедра)

1. auditorium
   * id: int, первичный ключ (уникальный идентификатор аудитории)
   * department-id: int, внешний ключ (идентификатор кафедры за которой закреплена аудитория)
   * size: int (вместимость аудитории)
   * number: int (номер аудитории)
2. post
   * id: int, первичный ключ (уникальный идентификатор должности)
   * name: varchar(255) (название должности)
   * salary: int (заработная плата)
3. employee
   * id: int, первичный ключ (уникальный идентификатор сотрудника)
   * firstname: varchar(255) (имя сотрудника)
   * lastname: varchar(255) (фамилия сотрудника)
   * post-id: int, внешний ключ (идентификатор занимаемой должности)
   * department-id: int, внешний ключ (идентификатор кафедры, на ко-

торую устроен сотрудник)

1. faculty
   * id: int, первичный ключ (уникальный идентификатор факультета)
   * name: varchar(255) (название факультета)
2. st-group
   * + id: int, первичный ключ (уникальный идентификатор группы)
     + name: varchar(255) (название группы)
     + faculty-id: int, внешний ключ (идентификатор факультета на кото-

ром находится группа)

1. student
   * + id: int, первичный ключ (уникальный идентификатор студента)
     + firstname: varchar(255) (имя студента)
     + lastname: varchar(255) (фамилия студента)
     + group-id: int, внешний ключ (идентификатор учебной группы студента)

## 2.2. Разработка логической структуры базы данных

Для разработки логической структуры БД определим три типа связей

между сущностями

* One-To-One
* One-To-Many
* Many-To-Many

Отобразим разработанную структуру при помощи сервиса «drawsql»

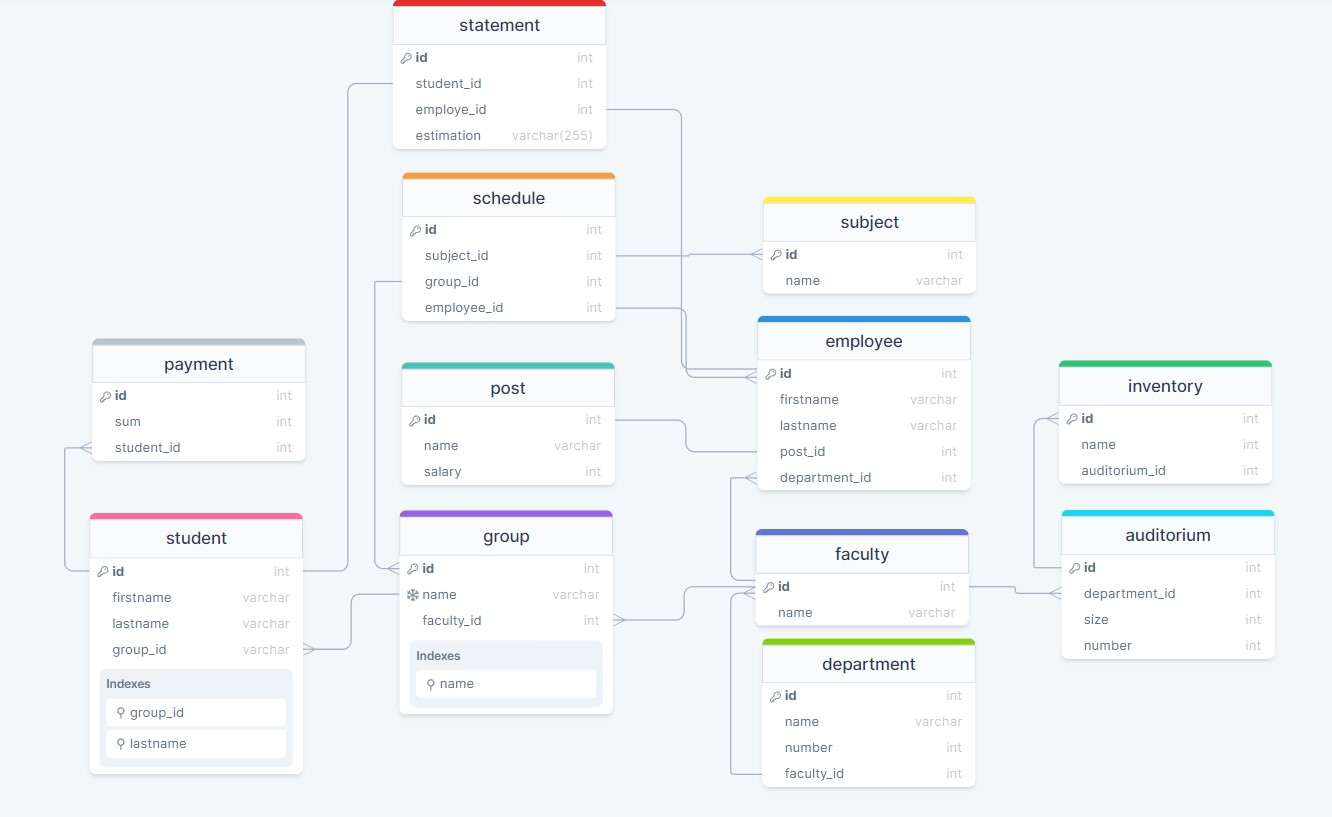


Рис. 2.1. Схема БД университета

## 2.3. Построение диаграммы сущность-связь

### 2.3.1. Раскрытие связей многие ко многим

Для раскрытия связей многие ко многим применяются связующие таблицы, в представленной ранее схеме такими таблицами являются «statement» и «schedule». Разберем их более подробно:

schedule

* id: int, первичный ключ (уникальный идентификатор проводимой пары)
* subject-id: int, внешний ключ (идентификатор проводимого предмета)
* group-id: int, внешний ключ (идентификатор учебной группы)
* employee-id: int, внешний ключ (идентификатор сотрудника, проводящего пару)

statement

* id:int, первичный ключ(уникальный идентификатор выставленной оценки)
* student-id: int, внешний ключ (идентификатор студента, кому была выставлена оценка)
* employee-id: int, внешний ключ (идентификатор преподавателя, выставившего оценку)
* estimation:int(дополнительное поле, отвечающее за выставленную оценку)

### 2.3.2. Поддержка ограничений целостности, списки доменов атрибутов

При определении атрибутов сущностей были заданы такие атрибуты как NULL, NOT NULL, UNIQUE, DEFAULT. Все первичные ключи являются уникальными, не допускающие NULL значений, и изменений, что контролируется СУБД

1. payment
   * + sun: NOT NULL
     + student-id: NOT NULL, ON DELETE CASCADE
2. statement
   * + student-id: NOT NULL, ON DELETE CASCADE
     + employee-id: NOT NULL, ON DELETE CASCADE
     + estimation: NOT NULL
     + curr-data: DEFAULT now()
3. schedule
   * + subject-id: NOT NULL, ON DELETE CASCADE
     + group-id: NOT NULL, ON DELETE CASCADE
     + employee-id: NOT NULL, ON DELETE CASCADE
     + eventDate: DEFAULT now()
4. subject
   * name: UNIQUE, NOT NULL
5. inventory
   * auditorium-id: NOT NULL, ON DELETE CASCADE
   * name: NOT NULL
6. department
   * student-id: NOT NULL, ON DELETE CASCADE
   * employee-id: NOT NULL, ON DELETE CASCADE
   * estimation: NOT NULL
   * curr-data: DEFAULT now()
7. auditorium
   * size: NOT NULL
   * number: NOT NULL
   * faculty-id: NOT NULL, ON DELETE CASCADE
8. post
   * name: NOT NULL
9. employee
   * post-id: NOT NULL, ON DELETE CASCADE
   * department-id: NOT NULL, ON DELETE CASCADE
10. faculty
    * name: UNIQUE, NOT NULL
11. st-group
    * + name: NOT NULL
      + faculty-id: NOT NULL, ON DELETE CASCADE
12. student

- group-id: NOT NULL, ON DELETE CASCADE

### 2.3.3. Нормализация объектной модели

Хранение информации в реляционной БД, предполагает такую структуру, в которой данные хранятся в оптимальном объеме. Для этого предусмотрен механизм нормализации. Проверим все сущности бд на соответствен

3-ей нормальной форме, которая предполагает условия:

1. В таблице не должно быть дублирующих строк
2. В каждой ячейке таблицы хранится атомарное значение
3. В столбце хранятся данные одного типа
4. Все не ключевые столбцы таблицы должны зависеть от полного ключа
5. Таблица должна иметь первичный ключ
6. Все колонки в таблице зависят от первичного ключа и не зависят друг

от друга

## 2.4. Выбор используемой СУБД

Дляразработкибазыданныхбылоприняторешениеиспользоватьреляционные БД свободные в использовании. Лидером на рынке СУБД на данный момент является PostgreSQL. PostgreSQL- свободная объектно-реляционная система управления базами данных, существует в реализациях для множества UNIX-подобных систем в том числе Linux, MacOS и Windows. Ее сильными сторонами являются высокопроизводительные и надёжные механизмы транзакций, расширяемая система встроенных языков программирования, расширяемость, а также механизмы наследования. Данная СУБД признана многими разработчиками и крупными корпорациями что позволяет не сомневаться в ее надежности и удобстве использования.

# 3. Физическое проектирование базы данных

## 3.1. Создание физической БД

Для создания физической базы данных были применены SQL скрипты, описанные ниже, результатом работы которых стало создание таблиц с

определённым набором столбцов имеющих типы описанные выше.

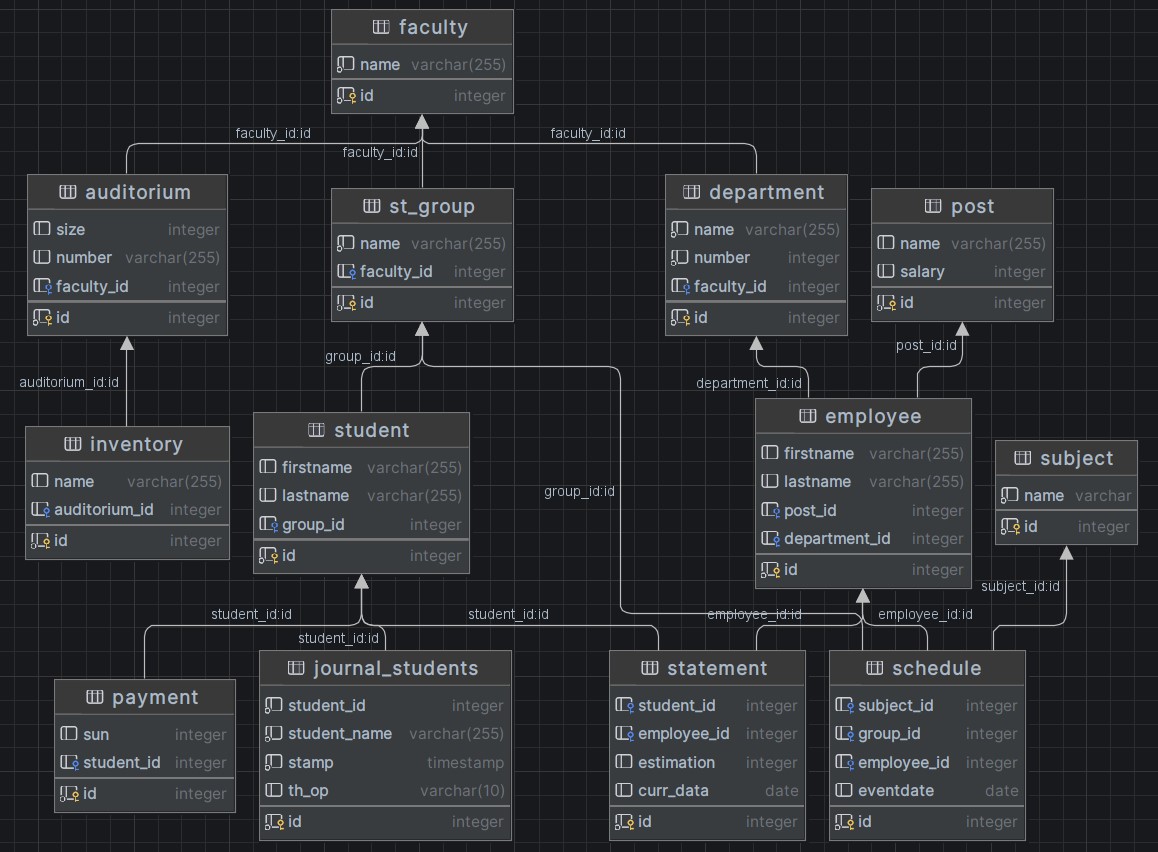


Рис. 3.1. БД

## 3.2. Описание запросов и процедур к БД

**3.2.1. Создания основных объектов базы данных на языке SQL в синтаксисе выбранной СУБД**

Скрипт создания таблиц:

subject (

name varchar unique not null

);

CREATE TABLE IF NOT EXISTS faculty ( id serial primary key, name varchar(255) unique not null );

CREATE TABLE IF NOT EXISTS st\_group ( id serial primary key, name varchar(255) unique not null , faculty\_id int references faculty(id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS department ( id serial primary key, name varchar(255) unique not null, number int unique not null, faculty\_id int references faculty(id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS post ( id serial primary key, name varchar(255) NOT NULL , salary int

);

employee (

firstname varchar(255), lastname varchar(255), post\_id int references post(id) ON DELETE CASCADE, department\_id int references department(id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS schedule ( id serial primary key, subject\_id int references subject(id) ON DELETE CASCADE, group\_id int references st\_group(id) ON DELETE CASCADE, employee\_id int references employee(id)ON DELETE CASCADE, eventDate date default now()

);

CREATE TABLE IF NOT EXISTS student ( id serial primary key, firstname varchar(255), lastname varchar(255), group\_id int references st\_group(id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS payment ( id serial primary key, sun int NOT NULL , student\_id int references student(id) ON DELETE CASCADE

);

auditorium (

size int NOT NULL , number varchar(255) NOT NULL , faculty\_id int references faculty(id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS inventory ( id serial primary key, name varchar(255) NOT NULL , auditorium\_id int references auditorium(id) ON DELETE CASCADE

);

CREATE TABLE IF NOT EXISTS statement ( id serial primary key, student\_id int references student(id) ON DELETE CASCADE, employee\_id int references employee(id) ON DELETE CASCADE, estimation integer NOT NULL , curr\_data date default now()

);

Для демонстрации работы приложения база данных была наполнена тестовыми данными, ниже приведены скрипт наполнения таблицы:

### 3.2.2. Получение данных о студенте по части его фамилии

SELECT concat(firstname,' ', lastname) as name, st\_group.name as group , avg(statement.estimation)

FROM student, st\_group, statement where student.group\_id = st\_group.id and

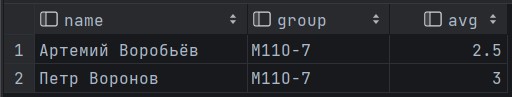


Рис. 3.2. Получение данных о студенте

statement.student\_id = student.id and student.lastname like '%В%'

GROUP BY st\_group.name, firstname, lastname;

### 3.2.3. Получение данных о расписании группы

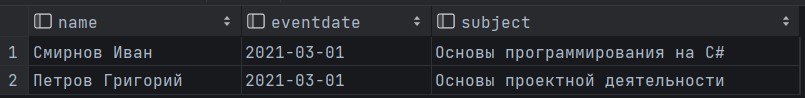


Рис. 3.3. Расписание группы

SELECT concat(employee.lastname, ' ', employee.firstname) AS name, schedule.eventdate, subject.name AS subject

FROM st\_group, schedule, employee, subject

WHERE schedule.group\_id = st\_group.id and employee\_id = employee.id and subject\_id = subject.id and st\_group.name = 'M11O-2';

### 3.2.4. Получение данных о выплатах студентам больше заданной суммы

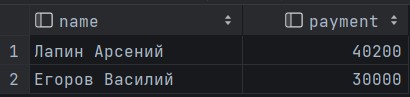


Рис. 3.4. Выплаты

SELECT name, payment

FROM (SELECT concat(lastname, ' ', firstname) AS name, sum(payment.sun) AS payment

FROM payment, student

WHERE student.id = payment.student\_id

GROUP BY lastname, firstname) AS tt

WHERE payment > 15000;

### 3.2.5. Получение данных о всех изменениях таблицы «students»

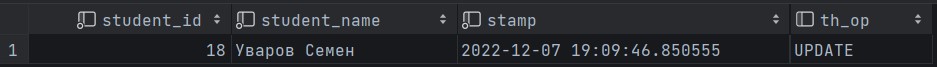


Рис. 3.5. Журналирование системы

SELECT student\_id, student\_name, stamp, th\_op from journal\_students;

### 3.2.6. Вывод кафедры с количеством сотрудников

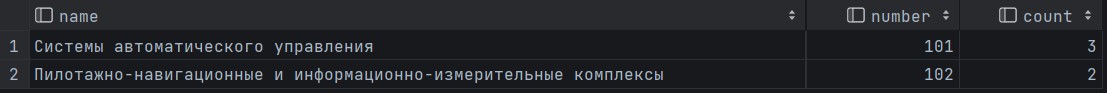


Рис. 3.6. Вывод количества сотрудников на кафедре

SELECT \* from (SELECT department.name, department.number,

count(employee.id) as count

FROM department, employee

WHERE employee.department\_id = department.id

GROUP BY department.name, department.number) as tt

ORDER BY count DESC

LIMIT 2;

### 3.2.7. Вывод расходов факультета на стипендии и мат. помощь

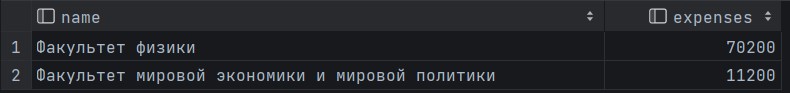


Рис. 3.7. Вывод расходов факультета

SELECT faculty.name as name, sum(sun) as expenses from (SELECT DISTINCT faculty\_id, payment.sun from st\_group, student, payment WHERE payment.student\_id = student.id and student.group\_id = st\_group.id and st\_group.faculty\_id = faculty\_id) as tt, faculty

WHERE tt.faculty\_id = faculty.id group by faculty.name;

## 3.3. Создания вспомогательных объектов базы данных

### 3.3.1. Создание триггера на добавление оценки студенту

Данный тригер контролирует оценки выставляемы студенту преподователем, в случе если оценка меньше 2 или более 5, то будет вызвана ошибка

не допускающая вставку некорректных данных.

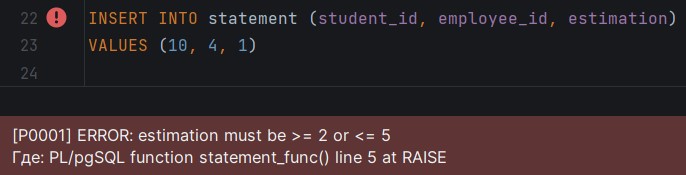


Рис. 3.8. Ошибка в случае ввода некорректных данных

CREATE FUNCTION statement\_func()

RETURNS TRIGGER AS

BEGIN

IF (new.estimation < 2 or new.estimation > 5) THEN

RAISE EXCEPTION 'estimation must be F= 2 or F= 5'; end if; return new;

END

LANGUAGE plpgsql;

CREATE TRIGGER statement\_insert\_trigger

BEFORE INSERT OR UPDATE

ON statement FOR EACH ROW

EXECUTE PROCEDURE statement\_func();

### 3.3.2. Журналирование изменений в таблице «students»

Для обеспечения безопасности данных было принято решение вести

журналирование всех изменений вносимых в таблицу «students»

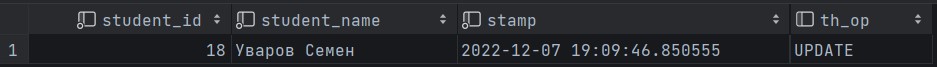


Рис. 3.9. Журнал

CREATE TABLE IF NOT EXISTS journal\_students ( id serial not null primary key, student\_id serial not null, student\_name varchar(255) not null, stamp timestamp NOT NULL, th\_op varchar(10)

);

CREATE OR REPLACE FUNCTION journal\_students\_func()

RETURNS TRIGGER AS

BEGIN

IF (TG\_OP = 'DELETE') THEN

INSERT INTO journal\_students(student\_id, student\_name,

th\_op, stamp)

VALUES (old.id, concat(old.lastname, ' ', old.firstname),

TG\_OP, now());

return new; else

INSERT INTO journal\_students(student\_id, student\_name, th\_op,

stamp) VALUES (new.id, concat(new.lastname, ' ', new.firstname),

TG\_OP, now());

return old; end if;

END

LANGUAGE plpgsql;

CREATE TRIGGER students\_trigger

AFTER INSERT OR UPDATE OR DELETE

ON student FOR EACH ROW

EXECUTE PROCEDURE journal\_students\_func();

### 3.3.3. Индексы

Для ускорения поиска по таблицам были разработаны индексы.

CREATE INDEX student\_index

ON student (firstname, lastname);

CREATE INDEX group\_index

ON st\_group (name);

CREATE INDEX employee\_index

ON employee (firstname, lastname);

# 4. Разработка прикладной программы

Для демонстрации возможностей разработанной базы данных, было написано приложение с графическим интерфейсом, реализующая заложенные ранее возможности ПО.

## 4.1. Основные аспекты приложения

Для написания ПО было принято решение использовать язык программирования JAVA. Основными факторами для выбора данного языка программирования было: его высокоуровневость (простота написания кода), наличие большого количества библиотек (при написании кода были использованы такие библиотеки как JDBC для обеспечения работы с базой данных, Lombok для ускорения написания кода и увеличения его читабельности, OpenCSV для удобного чтения и конвертации данных из csv формата и драйвер postgresql, необходимый для корректной работы JDBC). Для графического интерфейса использовался фреймворк JavaFX.

Файловое дерево проекта:

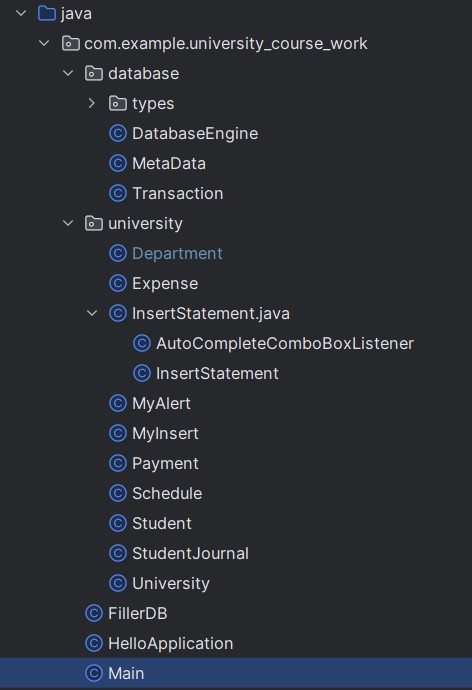


Рис. 4.1. Файловое дерево

Для декомпозиции и инкапсуляции кода отвечающего за доступ к БД

было написано несколько дополнительных классов.

1. DatabaseEngine - отвечает за подключение к БД и выполнение SQL запросов
2. MetaData - отвечает за получение информации о базе данных (созданных таблиц и др. информация)
3. Transaction - отвечает за механизм транзакций

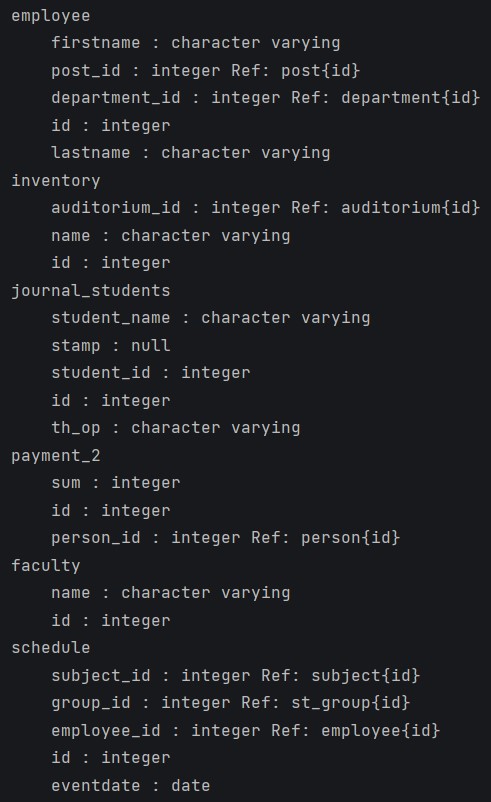


Рис. 4.2. Meta Data

## 4.2. Поиск студентов

Поиск студентов осуществляется по фамилии или части фамилии, результатом работы ПО является таблица, в которой выводите ФИО студента его текущая группа и средний балл.

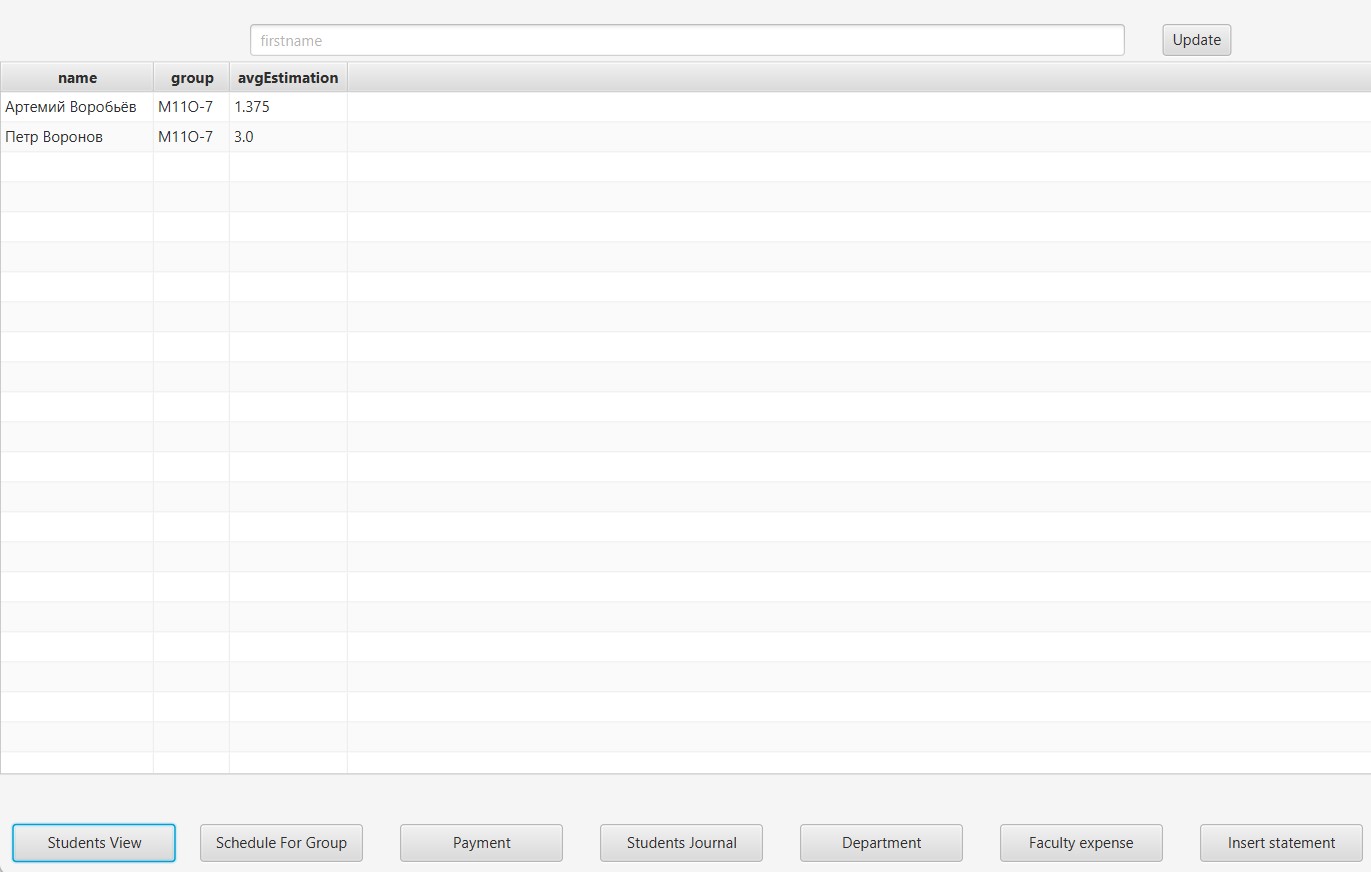


Рис. 4.3. Students 1

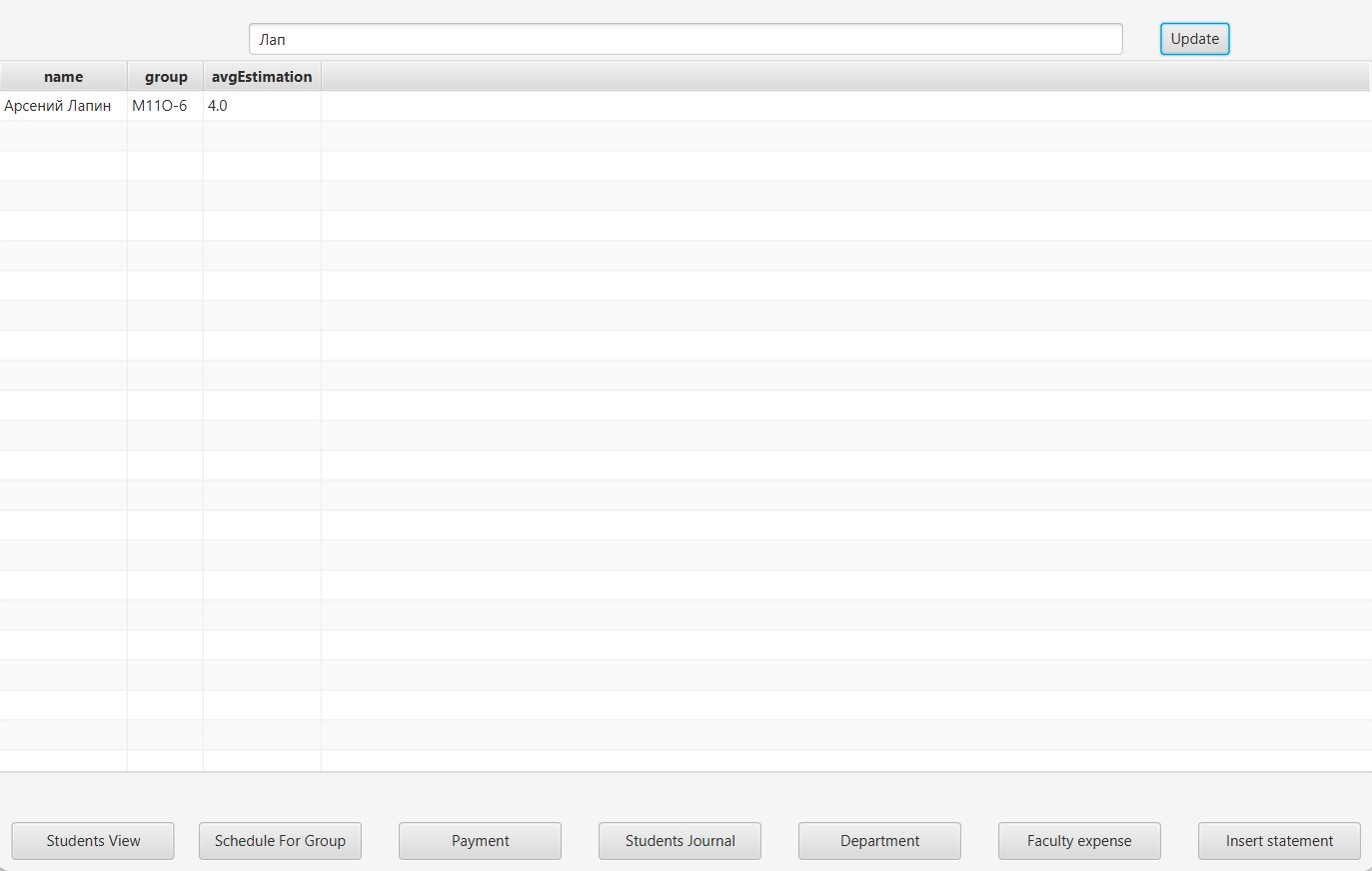


Рис. 4.4. Students 2

В случае если по результату запроса не было найдено ни одного студента выводится предупредительное сообщение

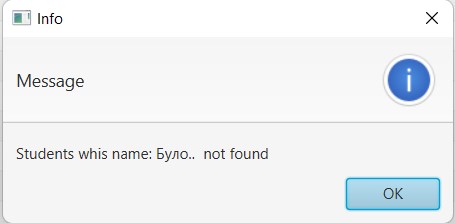


Рис. 4.5. Предупреждение

## 4.3. Расписание

Вывод расписания осуществляется путем ввода группы в поле в верх-

ней части окна и нажатия кнопки выполнить.

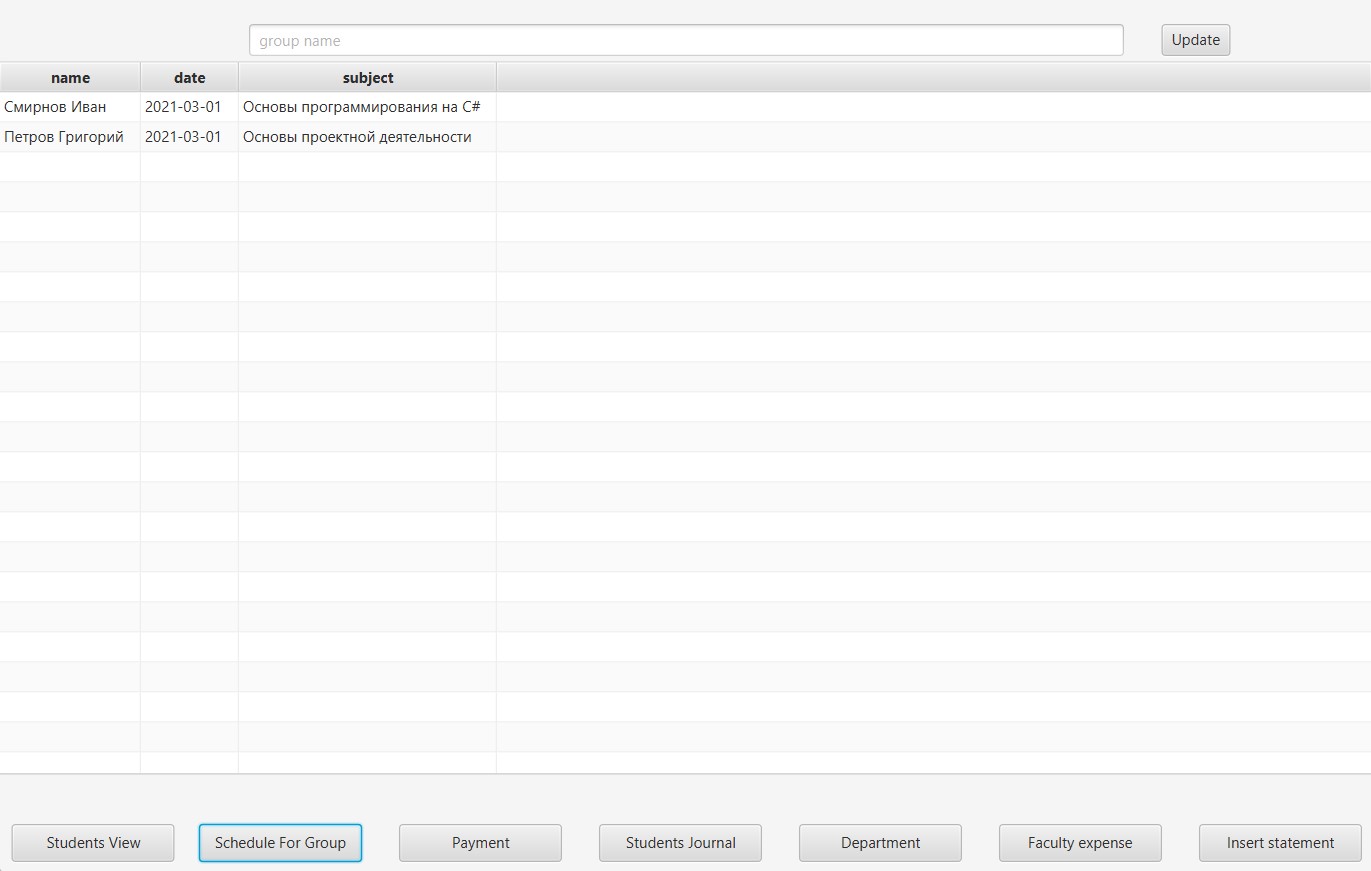


Рис. 4.6. Расписание

## 4.4. Выплаты

Для контроля за выплатами студентам была разработана система их вывода в виде таблицы с возможностью фильтрации по ее минимальному

размеру.

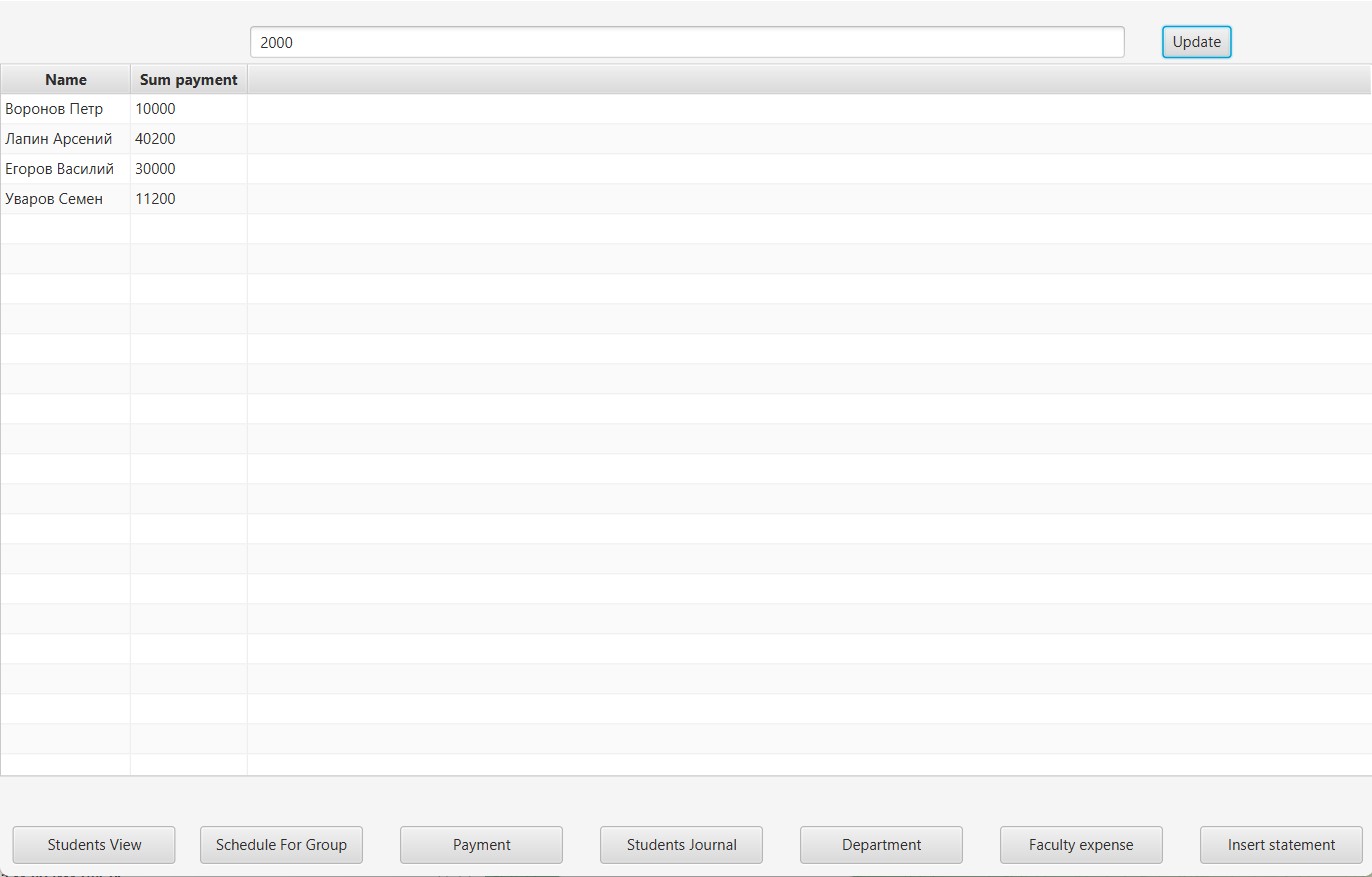


Рис. 4.7. Выплаты

## 4.5. Журнал изменений таблицы «студенты»

Для мониторинга вносимых изменений в таблицу «students» имеется

возможность вывода в виде таблицы

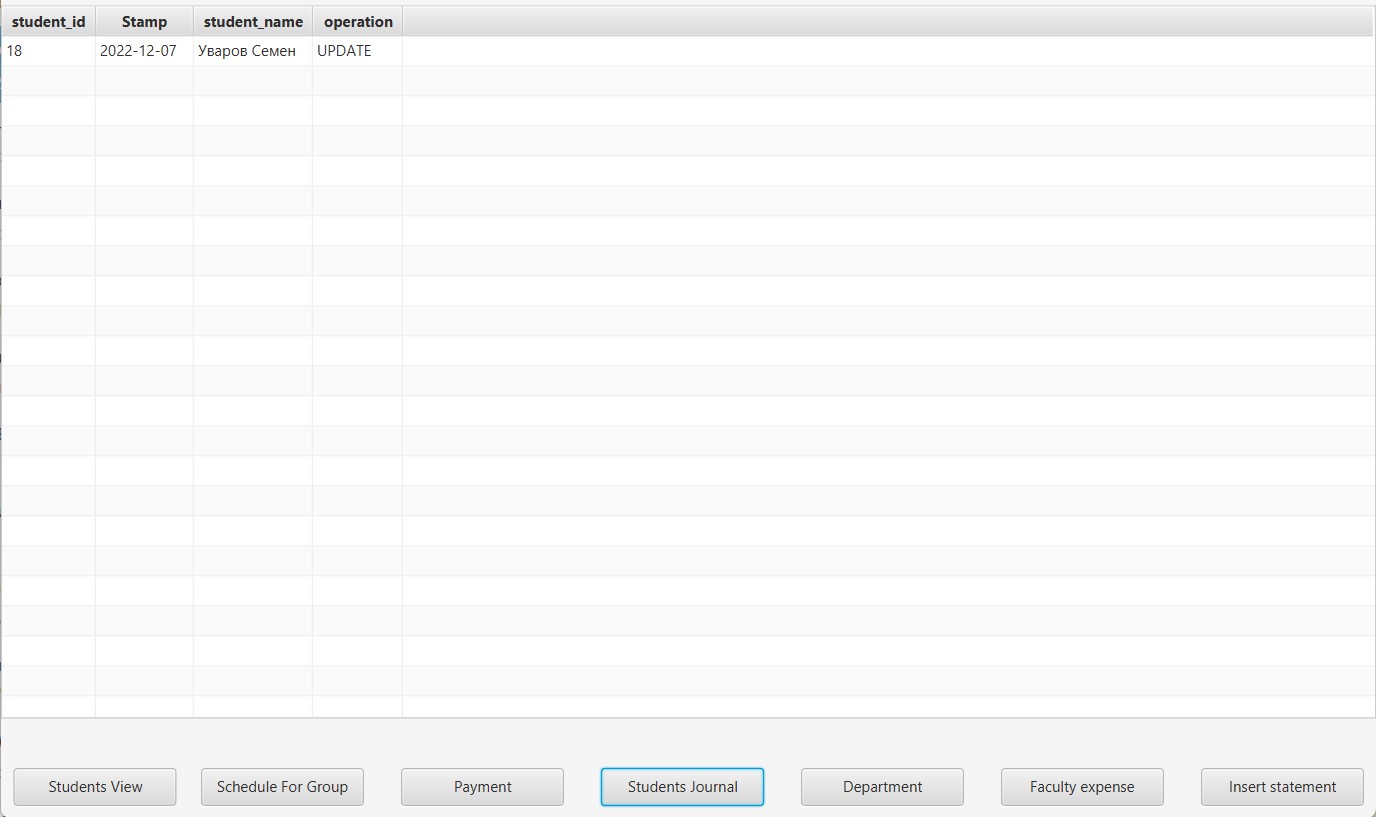


Рис. 4.8. Журнал

## 4.6. Вывод кафедр с количеством сотрудников

Для мониторинга количества сотрудников для возможной оптимизации внутренних процессов имеется возможность вывода кафедр с количеством сотрудников в виде таблицы с фильтрацией по количеству выводимых кафедр.

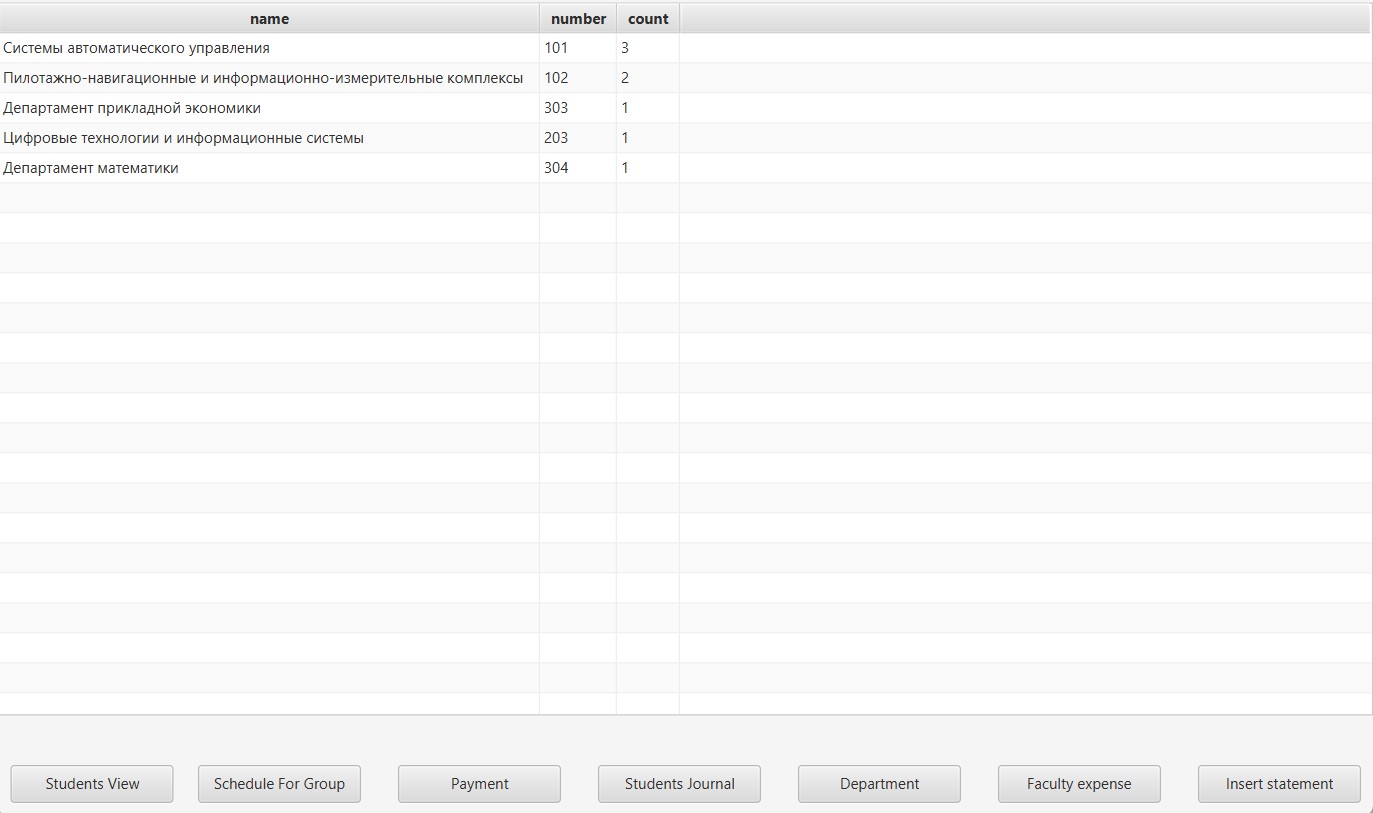


Рис. 4.9. Кафедры

## 4.7. Расходы факультетов

Для оценки эффективности работы кафедр была добавлена возможность просмотра их расходов.

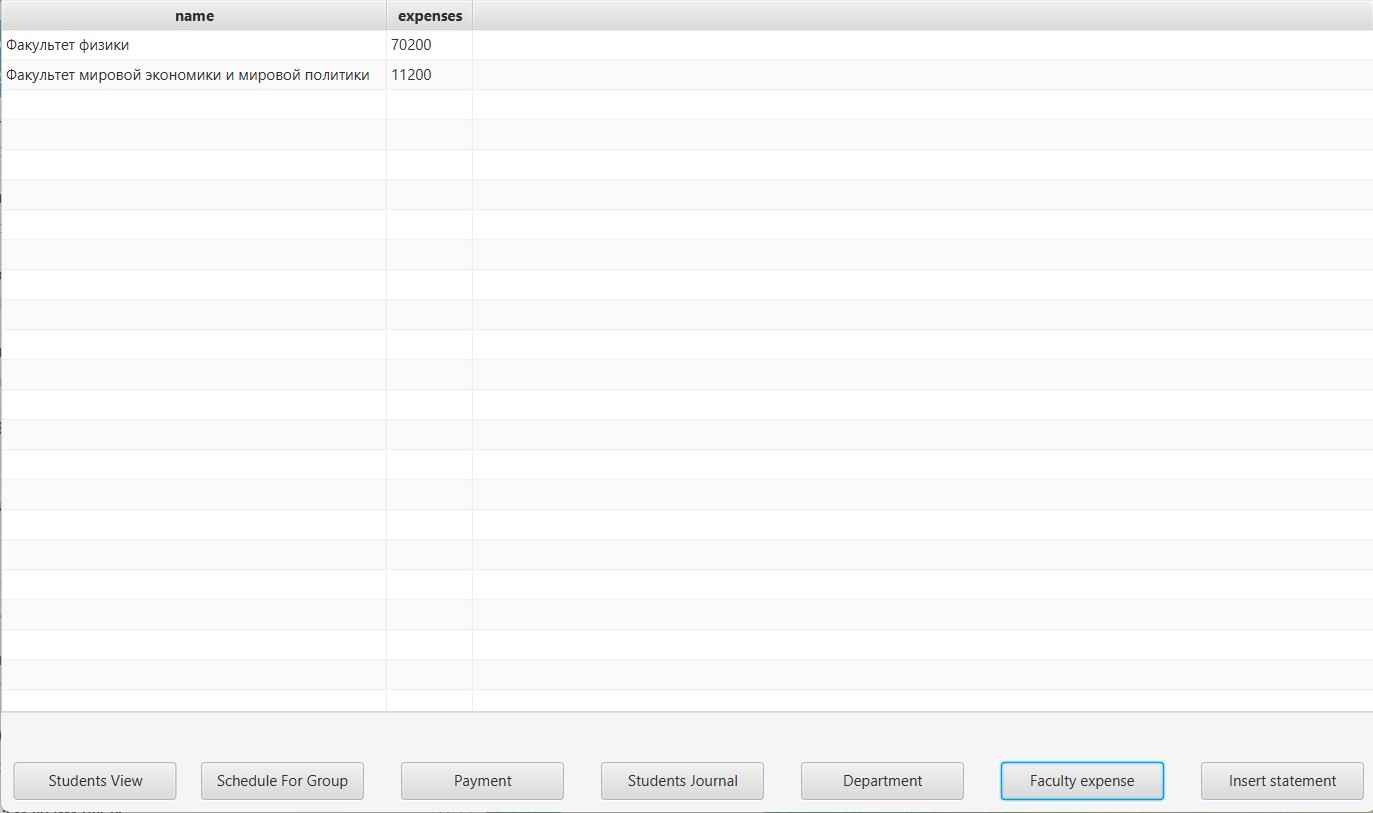


Рис. 4.10. Факультеты

## 4.8. Выставление оценок студентам

Для удобства преподавателей была разработана возможность в электронном виде проставлять оценки студентам. В связи с возможностью обучения большого количества студентов была реализована комбинация выпадающего списка и текстового поля (при вводе в текстовое поле части фамилии студента происходит фильтрация и в выпадающий список попадает необходимая их часть).

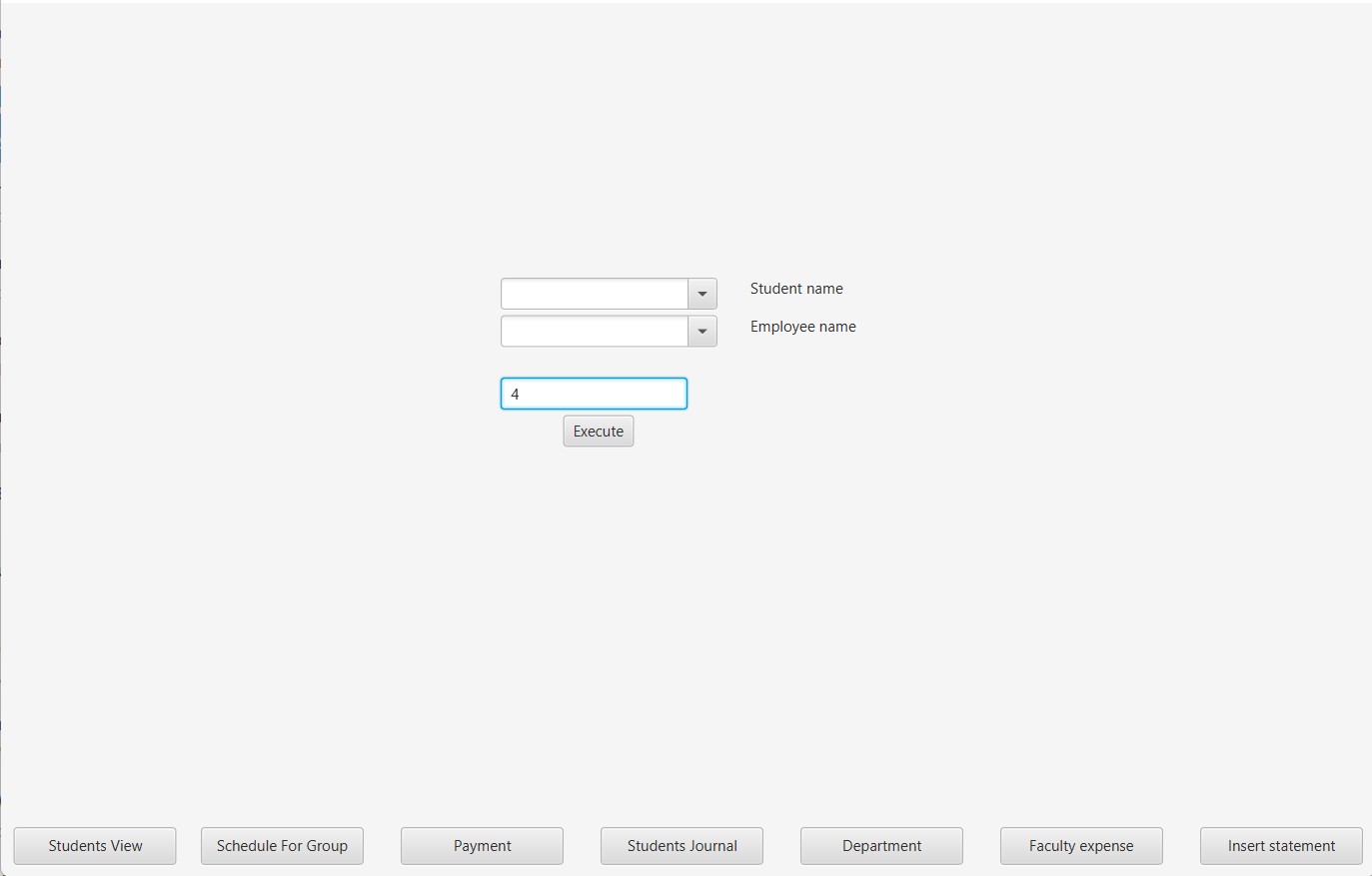


Рис. 4.11. Студенты

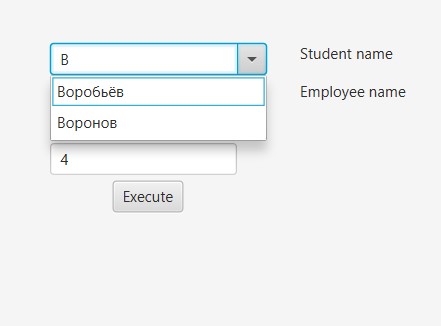


Рис. 4.12. Студенты

# Заключение

В ходе выполнения курсового проекта была реализована база данных для университета, позволяющая оптимизировать внутренние процессы. Был проведен предметный анализ изучаемой области, построен функциональный образ системы, разработы права доступа к конкретным частям базы данных, продумано взаимодействие с другими программами, разработана структура базы данных, выполнен анализ представленных на рынке СУБД для выбора наиболее подходящей. Также разработано ПО предоставляющее графический интерфейс для доступа к БД с возможностью просмотра списка студентов, расписания для группы, выплат стипендий и мат. помощи, просмотра расходов факультета, а также выставление оценок студентам с контролем правильности введенных данных.

# Источники

1. Документация PostgreSQl [Электронный ресурс],

URL:https://www.postgresql.org/ (дата обращения 10.12.2022)

1. Википедия[Электронныйресурс],URL:https://ru.wikipedia.org/wiki/(да-

та обращения 20.12.2022)

1. Habr PostgreSQl [Электронный ресурс], Изучаем PostgreSQL, автор АндррейБириуков,URL:https://habr.com/ru/company/otus/blog/706346/(дата обращения 15.12.2022)

# Приложение

## 1. Разработка ролей базы данных

create role teacher; create role student; create role faculty\_member; create role university\_employee;

GRANT SELECT ON payment, schedule, statement to student; GRANT SELECT ON schedule, statement, st\_group, student,

auditorium to teacher;

GRANT INSERT ON statement to teacher;

GRANT SELECT ON schedule, statement, st\_group, auditorium,

subject, employee, student, payment, faculty, department TO faculty\_member; GRANT UPDATE ON schedule, statement, st\_group, auditorium, subject, employee, student, payment TO faculty\_member; GRANT INSERT ON schedule, statement, st\_group, auditorium, subject, employee, student, payment TO faculty\_member; GRANT SELECT ON schedule, statement, st\_group, auditorium, subject, employee,student, payment, faculty, department, post, inventory TO university\_employee;

GRANT INSERT ON schedule, statement, st\_group, auditorium, subject, employee, student, payment, faculty, department, post, inventory

TO university\_employee;

GRANT UPDATE ON schedule, statement, st\_group, auditorium, subject, employee, student, payment, faculty, department, post, inventory

TO university\_employee;

Электродинамические свойства специальных блоков перехода описываются матрицей рассеяния вида:

с элементами

где W1 и W2 – волновые сопротивления граничащих сред, связанные с материальными параметрами сред, заполняющих блоки, соотношениями:

Создать новый документ, скопировать в него текст записки курсового проекта по Базам данных. Провести форматирование в соответствии с заданием.

фыаыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыывававававаававававававаавававаавававававаавыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыыы

Создать новый документ, скопировать в него текст записки курсового проекта по Базам данных. Провести форматирование в соответствии с заданиемыв

вввЫваываввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввЫвввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввввЫва