МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО
ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное Государственное Автономное
Образовательное Учреждение Высшего Образования
«Санкт-Петербургский Политехнический
Университет Петра Великого»
Институт Компьютерных Наук и
Кибербезопасности
Высшая Школа Технологий Искусственного
Интеллекта
Направление: 02.03.01 Математика и
Компьютерные Науки

Теоретические основы баз данных
Курсовая работа на тему «Электронная подача Заявления на поступление в ВУЗ»

Студент,
Группы 5130201/20002
_____ Салимли А.

Преподаватель,
к.т.н., доц.
_____ Попов С.Г.

«_____»_____ 2024 г.

Санкт-Петербург, 2024 г.

Содержание:

1. Анализ предметной области

1.1 Предметная область

Высшее учебное заведение (далее ВУЗ), имеет несколько самостоятельных учебных подразделений - институт.

Институт в свою очередь, имеет направления подготовки, которые по желанию должен выбрать абитуриент.

Абитуриент - претендент который подает заявление, на поступление в учебное заведение, например в ВУЗ, для получения высшего образования. В данном проекте рассматривается поступление абитуриента в один ВУЗ, по направлению подготовки, в котором заинтересован абитуриент.

После успешной сдачи, единственного государственного экзамена (далее ЕГЭ), и сдачи вступительных испытаний, абитуриент подает заявку, которая содержит в себе - документы (например - паспорт РФ, аттестат, СНИЛС), результаты испытания и заявление, которое в свою очередь содержит в себе желаемую форму обучения, выбранное направление подготовки и основу обучения.

После подачи всех документов и заявления, заявка абитуриента рассматривается, после чего, абитуриента уведомляют, о статусе зачисления (зачислен/ отклонен).

Если абитуриент прошел все испытания и подал необходимые атрибуты заявления, его зачисляют в список студентов группы по направлению.

1.2 Определения:

Абитуриент - желающий, который подает заявление на поступление в учебное заведение (например, вуз или институт) для получения высшего образования.
Претендент - лицо желающее сдать документы
Аттестат - официальный документ, выдаваемый после окончания среднего полного либо специального образовательного учреждения, подтверждающий получение базового образования или квалификации.
Форма обучения - способ организации учебного процесса, определяющий время и место обучения для студента. Может включать очные (полный дневной режим), заочные (дистанционное обучение без регулярного посещения учебных занятий) и вечерние (занятия проводятся после рабочего дня) формы обучения.
Институт - самостоятельное образовательное подразделение вуза, специализирующееся на определенной области знаний.
ВУЗ - высшее учебное заведение, предлагающее программы образования и научных исследований на более высоком уровне, чем среднее образование. Направления - сферы знаний или профессиональных областей обучения, в которых студент может выбрать специализацию для своего будущего образования и карьеры.
Студент - лицо, которое учится в высшем учебном заведении и получает образование в рамках программы обучения.
Вступительные экзамены - испытания или тесты, составленные учебным заведением, чтобы оценить знания и навыки абитуриента для определения его способности и пригодности к поступлению на обучение.
Общее количество баллов - сумма набранных очков или оценок, назначенных за выполнение заданий или успешное прохождение экзаменов. Оно может использоваться для определения успеха абитуриента при поступлении или для измерения его академической успеваемости в течение обучения.
Университет - высшее учебное заведение, где готовят специалистов по фундаментальным и прикладным наукам и как правило осуществляют научно-исследовательские работы.
Паспорт - основной документ удостоверяющий личность гражданина на территории страны его рождения. (В нашем случае РФ)
ФИО - Фамилия Имя Отчество (последнее если есть).
Единый государственный экзамен (ЕГЭ) — это стандартизированный экзамен, который проводят в конце обучения в школе, лицее, гимназии и других учреждениях общего среднего образования. В ходе этого экзамена оцениваются знания учащихся по основным школьным дисциплинам. Результаты ЕГЭ используются для приёма в вузы и другие образовательные учреждения.

1.3 Цели создания:

1. Построение списков абитуриентов в реальном масштабе времени (1 час по мин.образования).
2. Автоматизированная система рассмотрения заявлений на поступления в ВУЗ.
3. Онлайн информирование (Информирование абитуриента о состоянии документов и статуса зачисления).
4. Круглосуточный прием заявлений на поступления.

1.4 ER-диаграмма:

1. Абитуриент выбирает направление

2. Конкретный институт ведет подготовку по выбранному абитуриентом направлению

3. Абитуриент сдает ЕГЭ и вступительные испытания на основе дисциплин, которые необходимы для выбранного направления

4. Абитуриент подает заявку

5. Заявка в свою очередь содержит заявление

6. Заявка так же имеет статус (зачисления)

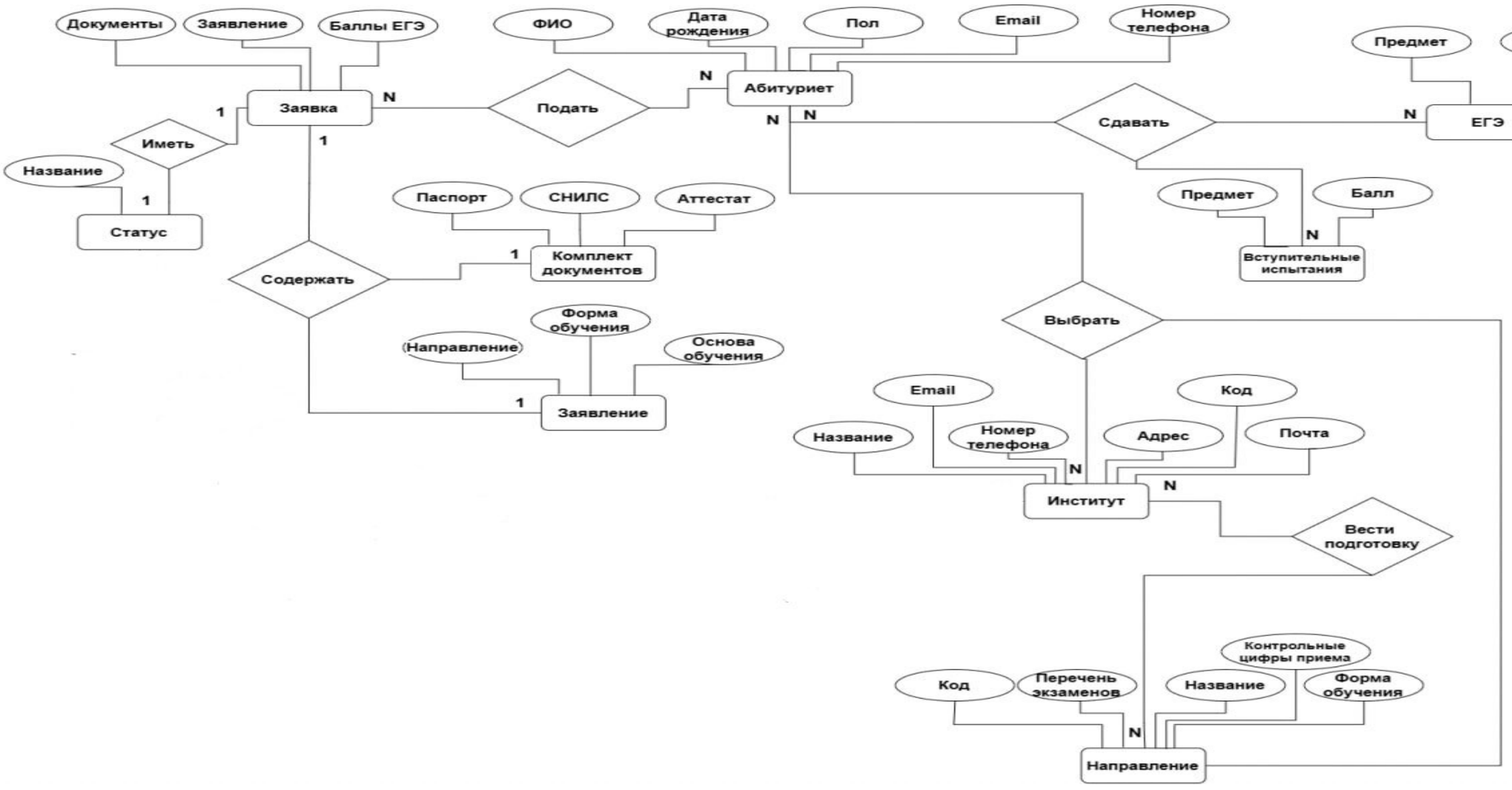


Рис. 1 ER-Диаграмм

1.5 Описание сущностей и атрибутов

1. **Абитуриент** - претендент , который подает заявление на поступление в учебное заведение (например, вуз или институт) для получения высшего образования.

   Атрибуты:

   - *Имя*
   - *Фамилия*
   - *Отчество*
   - *Дата рождения*
   - *Статус зачисления*

2. **Дисциплина** - часть учебного плана, самостоятельный предмет с аттестацией.

   Атрибуты:

   - *Название*

3. **Документ** - материальный объект с зафиксированной на нем информацией в виде текста, звукозаписи или изображения.

   Атрибуты:

   - *Название*
   - *Принадлежность*

4. **Результат испытания** - оценка знаний абитуриента по конкретной дисциплине.

   Принадлежность абитуриенту

   Атрибуты:

   - *Дисциплина*
   - *Тип*
   - *Баллы*

5. **Направление подготовки** - образовательная программа, ориентированная на приобретение определенного набора знаний и навыков в определенной сфере.

   Атрибуты:

   - *Название*
   - *Код*
   - *Институт*

6. **Институт** - подразделение ВУЗа, образовательное учреждение, где проводится подготовка специалистов в определенных областях знаний и профессиональных навыков.

   Атрибуты:

   - *Название института, директор*
   - *Адрес института, номер телефона*

1.6 Схема объектов:

Рис.2 Схема объектов

Описание схемы объектов:

У абитуриентов несколько результатов испытаний по дисциплинам, результат всегда по одной дисциплине.

Направления подготовки требуют набор из нескольких дисциплин.

В одном институте несколько направлений.

Каждый абитуриент выбирает несколько направлений.

И каждый абитуриент предоставляет несколько документов.

2. Проектирование базы данных

На рисунке 3 и 4, изображены схемы баз данных на русском и на английском языке соответственно, число сверху слева таблицы обозначаент приоритет заполнения таблицы, чем меньше число, тем выше приоритет заполнения. Сверху справа таблицы, число обозначающее количество записей.

2.1 Атрибуты таблиц базы данных:

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_дисциплины	subject_id	INT	PK	-
название	name	VARCHAR	-	-

Таблица 1: Subject

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_статус_зачисления	enrollment_status_id	INT	PK	-
название	name	VARCHAR	-	-

Таблица 2: Enrollment_Status

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_документа	document_type_id	INT	PK	-
название	Name	VARCHAR	-	-

Таблица 3: Document_Type

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_института	department_id	INT	PK	-
название	name	VARCHAR	-	-
номер телефона	phone_number	VARCHAR	-	-
адрес	address	TEXT	-	-
директор	head_master_name	VARCHAR	-	-

Таблица 4: Department

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_абитуриента	enrollee_id	INT	PK	-
имя	first_name	VARCHAR	-	-
фамилия	last_name	VARCHAR	-	-
отчество	middle_name	VARCHAR	-	-
дата_рождения	birth_date	DATE	-	-
статус_зачисления	enrollment_status_id	INT	FK	Enrollment_Status. enrollment_status_id

Таблица 5: Enrollee

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_направление	specialty_id	INT	PK	-
название	name	VARCHAR	-	-
код	code	VARCHAR	-	-
id_институт	department_id	INT	FK	Department.department_id

Таблица 6: Specialty

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_документ_абитуриента	document_id	INT	PK	-
id_абитуриент	enrollee_id	INT	FK	Enrollee.enrollee_id
id_документ	document_type_id	INT	FK	Document_Type. document_type_id

Таблица 7: Document

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_результата	achievment_id	INT	PK	-
id_абитуриент	enrollee_id	INT	FK	Enrollee.enrollee_id
id_дисциплина	subject_id	INT	FK	Subject.subject_id
балл_вступительные	inner_result	INT	-	-
балл_егэ	ege_result	INT	-	-

Таблица 8: Achievment

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
id_выбор	choice_id	INT	PK	-
id_абитуриент	enrollee_id	INT	FK	Enrollee.enrollee_id
Id_направление	specialty_id	INT	FK	Specialty.specialty_id
приоритет	priority_index	INT	-	-

Таблица 9: Choice

Имя	Имя на англ.	Тип	Тип ключа	Ссылка
дисциплина_направление_id	subject_requirement_id	INT	PK	-
id_дисциплина	subject_id	INT	FK	Subject.subject_id
id_направление	specialty_id	INT	FK	Specialty.specialty_id
минимальный_балл_егэ	ege_minimal_result	INT	-	-
минимальный_балл_вступительные	inner_minimal_result	INT	-	-

Таблица 10: Subject_Requirement

2.2 Размеры полей

Так как все названия института, направлений и дисциплин не превышают 50 символов, был использован тип VARCHAR(50).

Для фамилии имени отчества абитуриента, директора был использован так же тип VARCHAR но диапазон символов ФИО по отдельности был увеличен до VARCHAR(80).

Номера телефонов не могут превышать 20 символов, поэтому был использован тип VARCHAR(20).

Статус зачисления хранится в формате чисел от 1 до 2, где 1 - отклонено, 2 - зачислен. Следовательно был взят тип INT UNSIGNED NOT NULL.

Все личные документы имеют свой уникальный номер и не могут быть меньше или равны нулю, поэтому для документов был взят тип INT UNSIGNED NOT NULL.

PK и FK, инициалы ключей, сокращено от primary key и foreign key. Foreign key нужн для ссылок на другие таблицы и их поля, а primary key используется для уникальной идентификации записей в таблице. Он гарантирует, что каждая запись имеет уникальное значение primary key, что обеспечивает целостность данных и упрощает процессы поиска, сортировки и связывания данных. PK так же указан в таблицах в столбце *«тип ключа»* и в схеме базы данных.

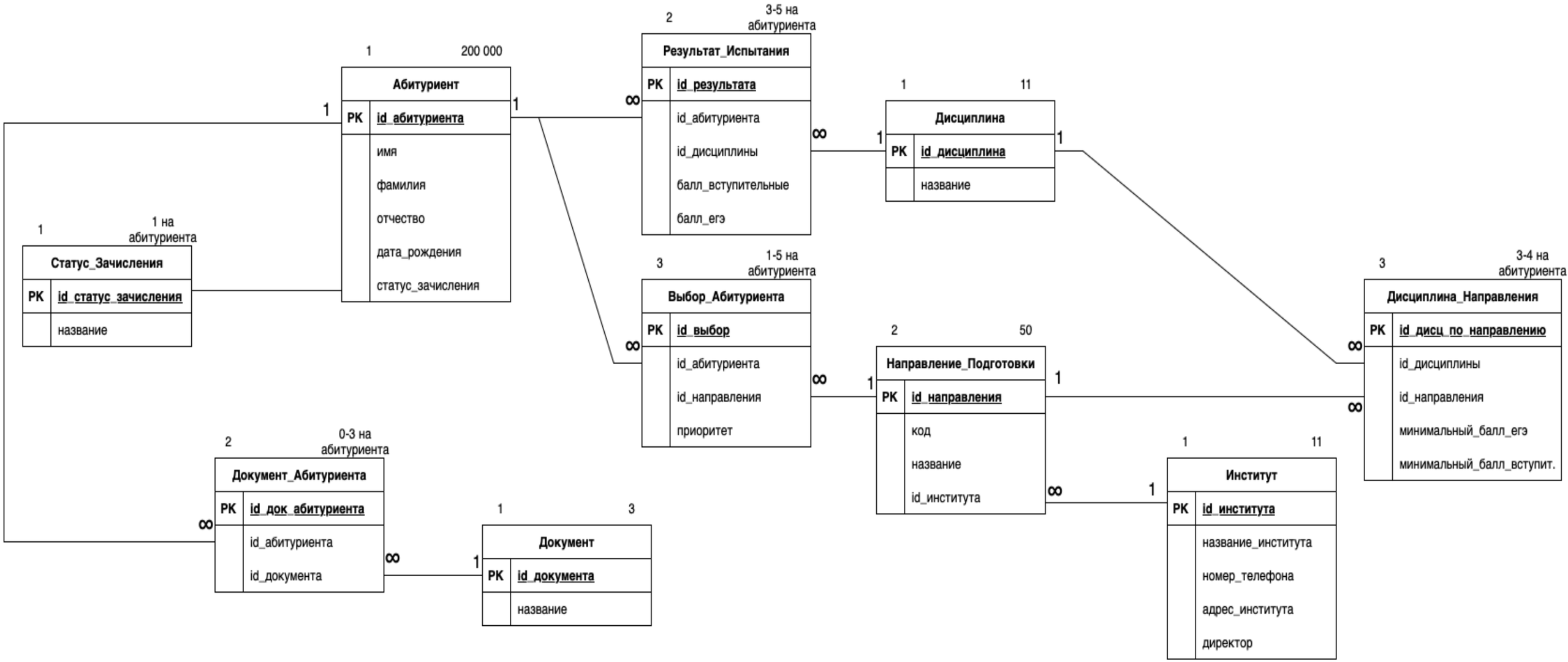


Рис. 3 Схема базы данных на русском

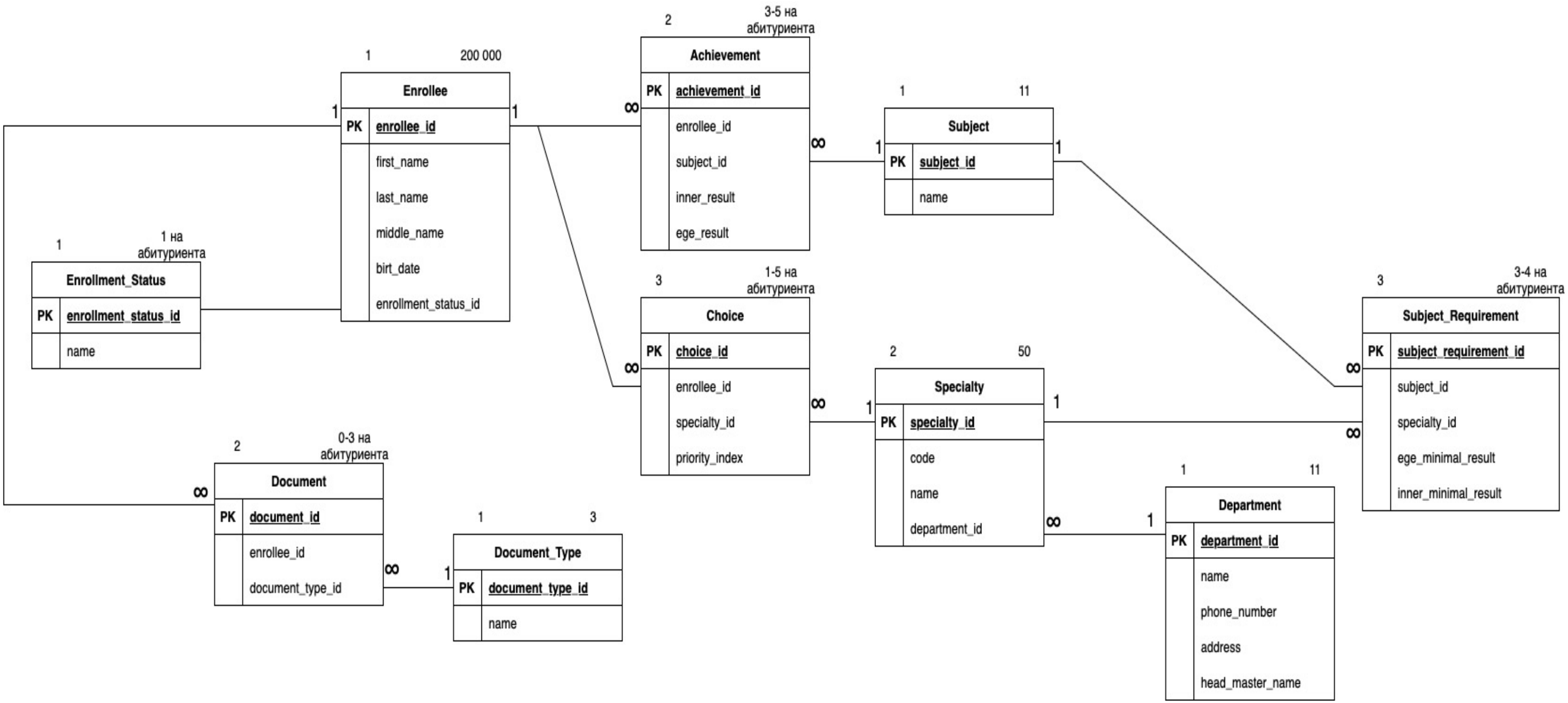


Рис. 4 Схема базы данных на английско

2.3 Заполнение таблиц базы данных:

Для заполнения таблиц базы данных была реализована программа в ИСР Visual Studio Code на языке Python (v. 3.9.4), код представлен в приложении В. С помощью библиотеки mysql.connector, VS Code подключается к базе данных, через имя пользователя, пароль, хост и порт. (По стандарту хост = «127.0.0.1», а порт = «3306». Были созданы массивы заполненные именами, фамилиями и отчествами, так же список содержащий статус зачисления, список содержащий названия институтов и список содержащий названия направлений и дисциплин, даты рождения генирировались «случайно» с помощью библиотеки random и datetime и метода timedelta .

Заполнения осуществлялись с помощью команд - INSERT INTO «название таблицы».

После успешного заполнения, выводится строка об успешном заполнении.

В случае неудачного заполнения и/ или неудачного подключения, выводится сообщения благодаря методу try.

В таблице 1 указан результат заполнения таблиц:

Таблица	**Записи**
Achievement	799680
Choice	600124
Department	11
Document	400622
Document_Type	3
Enrollee	200000
Enrollment_Status	2
Specialty	10
Subject	11
Subject_Requirement	34

Таб. 1

3. Запросы

1. Найти дисциплину который сдавал абитуриент A который поступал по направлению B
2. Из института A посчитать число абитуриентов
2.2 Из института A посчитать число по каждому направлению
3. С каждого института найти число подавших абитуриентов
4. Найти абитуриента с максимальным числом результата поступления
5. Число направлений с одинаковым числом абитуриентов
6. Найти дисциплину по которой подано больше результатов чем у дисциплины A
7. Найти абитуриентов которые не подавались в институт A
8. Для всех институтов и всех дисциплин посчитать число абитуриентов

Время работы запросов определялось по результатам работы команды explain для каждого запроса и выводом времени к консоль.

3.1 Дисциплина которую сдавал абитуриент A который поступал по направлению B.

```
select S.subject_id, S.name
from Choice C
join Achievment A on C.enrollee_id = A.enrollee_id
join Specialty Sp on C.specialty_id = Sp.specialty_id
join Subject_Requirement SR on Sp.specialty_id = SR.specialty_id and SR.subject_id =
A.subject_id
join Subject S on SR.subject_id = S.subject_id
where C.enrollee_id = 3 and C.specialty_id = 10;
```

Результат запроса:

subject_id	Name
32	Литература
33	Музыка

Запрос 3.1 (Время запроса - 0.0024 сек)

3.2 Количество абитуриентов из института А

```
select S.department_id, count(DISTINCT C.enrollee_id)
from Choice C
join Specialty S on C.specialty_id = S.specialty_id
where S.department_id = 1;
```

Результат запроса:

department_id	count(DISTINCT C.enrollee_id)
1	101609

Запрос 3.2 (Время запроса - 0.20 сек.)

3.2.2 Для института А посчитать число абитуриентов по каждому направлению

```
select C.specialty_id, count(DISTINCT C.enrollee_id)
from Choice C
join Specialty S on C.specialty_id = S.specialty_id
where S.department_id = 3
group by C.specialty_id;
```

Результат запроса:

specialty_id	count(DISTINCT C.enrollee_id)
2	59792
10	59545

Запрос 3.2.2 (Время запроса - 0.18 сек.)

3.3 Число подавших абитуриентов с каждого института

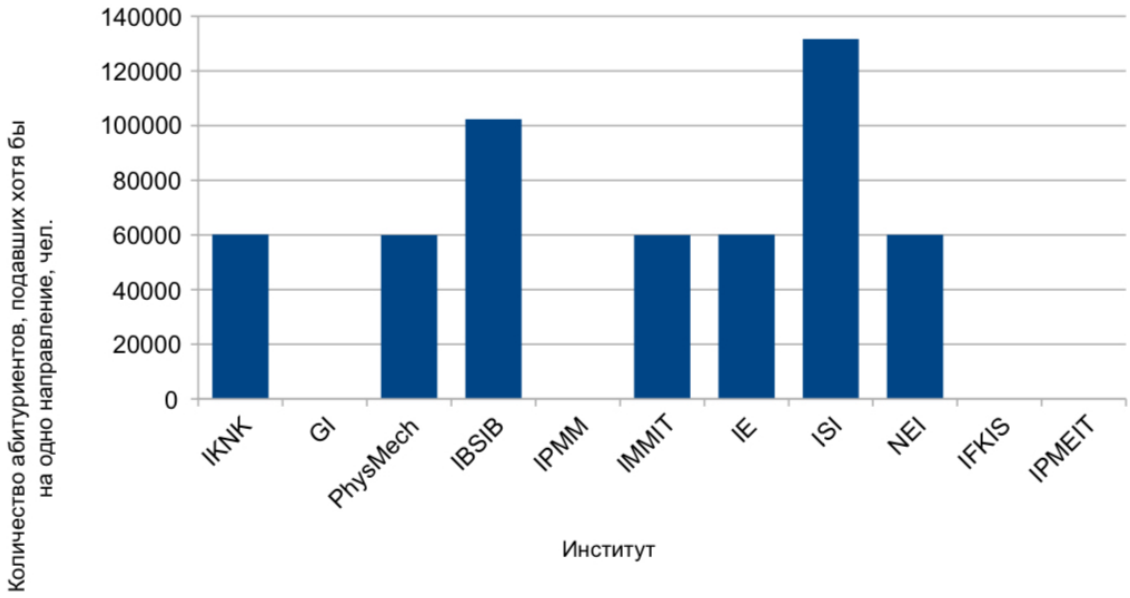
```
select D.name as department_name, count(DISTINCT C.enrollee_id) as enrollee_count
from Department D
left join Specialty S on S.department_id = D.department_id
left join Choice C on C.specialty_id = S.specialty_id
```

Результат запроса:

department_name	enrollee_count
ИКНК	101609

ГИ	132109
ФизМех	0
ИБСиБ	60380
ИПММ	0
ИММиТ	60264
ИЭ	60179
ИСИ	0
ЯИ	60166
ИФКиС	0
ИПМЭиТ	60622

Запрос 3.3 (Время запроса - 0.62 сек.)



Гистограмма к запросу 3.3

3.4 Абитуриент с максимальным числом результата поступления

```
select C.enrollee_id, C.specialty_id, sum(greatest(A.ege_result, A.inner_result)) as result
from Choice C
join Subject_Requirement SR on C.specialty_id = SR.specialty_id
join Achievment A on A.enrollee_id = C.enrollee_id and A.subject_id = SR.subject_id
group by C.specialty_id, C.enrollee_id
having result = (select max(result) from (select sum(greatest(A.ege_result, A.inner_result)) as result
    from Choice C
    join Subject_Requirement SR on C.specialty_id = SR.specialty_id
```

join Achievment A on A.enrollee_id = C.enrollee_id and A.subject_id = SR.subject_id
group by C.enrollee_id, C.specialty_id) results);

Результат запроса:

enrollee_id	specialty_id	result
406697	10	384

Запрос 3.4 (Время запроса - 0.52 сек.)

3.5 Направления с одинаковым числом абитуриентов

```
select enrollees_count, count(enrollees_count) as specialty_count
from (select count(DISTINCT C.enrollee_id) enrollees_count
    from Choice C
    join Specialty S on C.specialty_id = S.specialty_id
    where C.priority_index = 2
    group by S.specialty_id) as counts
group by enrollees_count;
```

Результат запроса:

enrollee_count	specialty_count
16102	1
16053	1
15991	1
16076	1
16055	1
15918	1
15821	1
16286	1
16088	1
15760	1

Запрос 3.5 (Время запроса 0.45 сек.)

3.6 Дисциплина по которой подано больше результатов чем у дисциплины A

```
select A.subject_id, count(A.enrollee_id) as count
```

```
from Achievment A
group by A.subject_id
having count > (select count(A.enrollee_id)
from Achievment A
where A.subject_id = 10
```

Результат запроса:

subject_id	count
29	72839
30	73179
33	72973

Запрос 3.6 (Время запроса 0.52 сек.)

3.7 Абитуриенты не подавшие в институт А

```
select E.last_name, E.first_name
from Enrollee E
where E.enrollee_id not in (select distinct C.enrollee_id
from Enrollee
join Choice C on C.enrollee_id = Enrollee.enrollee_id
left join Specialty S on C.specialty_id = S.specialty_id
where S.department_id = 2);
```

Результат запроса:

first_name	last_name
Салимли	Айзек
Кузнецов	Игорь
….	…..
Ваганова	Ольга

Запрос 3.7 (67891 строк) (Время запроса 0.64 сек.)

3.8 Число абитуриентов для института и дисциплин.

```
select
   S.name as subject_name,
   D.name as department_name,
   (select count(distinct C.enrollee_id)
```

```
    from Specialty Sp
    left join Choice C on Sp.specialty_id = C.specialty_id
    join Subject_Requirement SR on SR.specialty_id = Sp.specialty_id
    where Sp.department_id = D.department_id
      and SR.subject_id = S.subject_id
  ) as enrollee_count
from
  Subject S
  cross join Department D
group by
  S.subject_id,
  D.department_id,
  S.name,
  D.name;
```

Результат запроса:

subject_name	department_name	enrollee_count
Математика	ИКНК	101609
…..	….	….
Литература	ГИ	132109
Экономика	ГИ	60622

Запрос 3.8(а) (121 строка) (Время запроса 0.51 сек)

department_name	count(distinct specialty_id)
ИКНК	2
ГИ	3
ИБСиБ	1
ИММИТ	1
ИЭ	1
ЯИ	1
ИПМЭиТ	1

Запрос 3.8(б) (Время запроса - 0.081 сек.)

specialty_id	subject_id
3	28

3	24
3	33
3	26

Запрос 3.8(в) (Время запроса - 0.073 сек.)

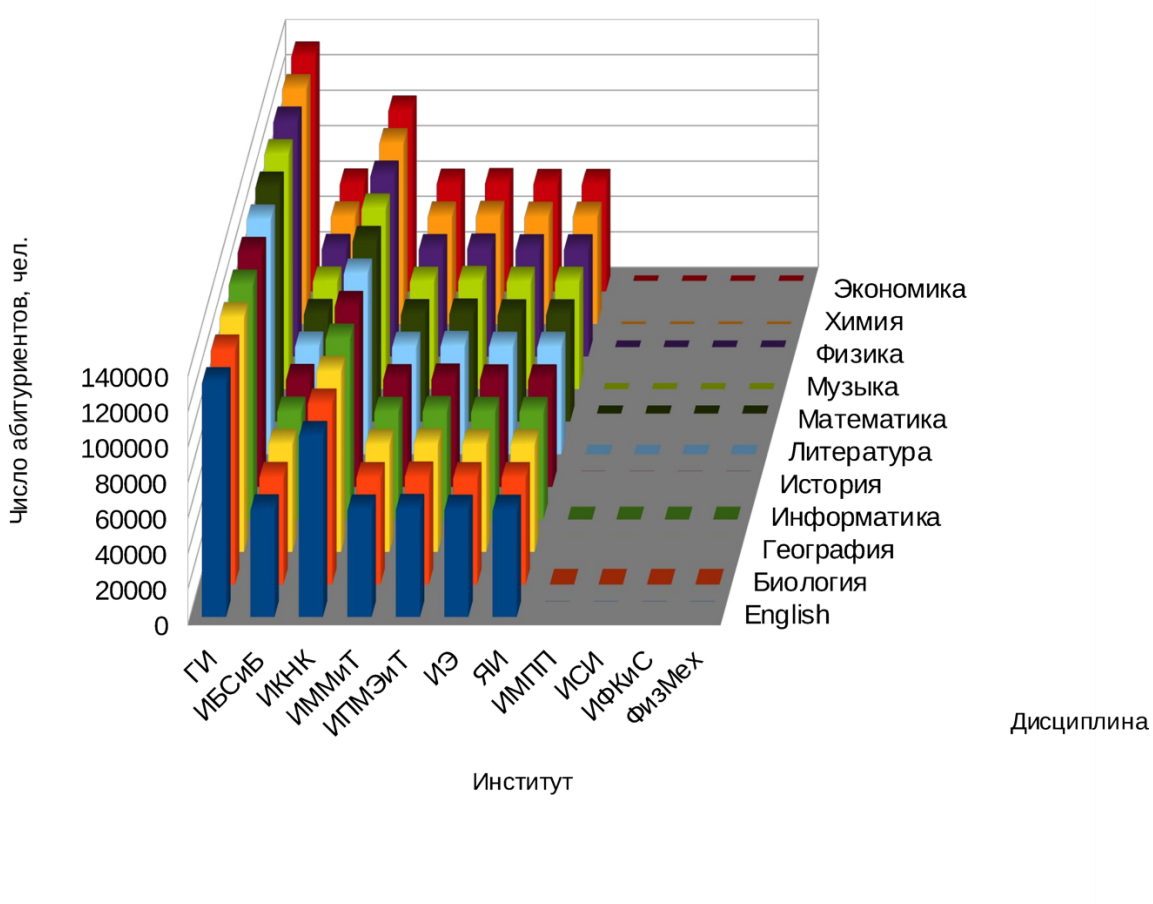


Диаграмма к запросу 8(а).

3.4 Explain

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Sp	NULL	const	PRIMARY	PRIMARY	4	const	1	100.00	Using index
1	SIMPLE	C	NULL	ref	enrollee_id,specialty_id	enrollee_id	4	const	5	16.80	Using where
1	SIMPLE	A	NULL	ref	enrollee_id,subject_id	enrollee_id	4	const	3	100.00	NULL
1	SIMPLE	SR	NULL	ref	subject_id,specialty_id	subject_id	4	student_admission.A.subject_id	3	10.81	Using where
1	SIMPLE	S	NULL	eq_ref	PRIMARY	PRIMARY	4	student_admission.A.subject_id	1	100.00	NULL

explain запроса 1.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	S	NULL	ref	PRIMARY,department_id	department_id	4	const	2	100.00	Using where; Using index
1	SIMPLE	C	NULL	ref	specialty_id	specialty_id	4	student_admission.S.specialty_id	66733	100.00	NULL

explain запроса 2.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	S	NULL	ref	PRIMARY,department_id	department_id	4	const	2	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	C	NULL	ref	specialty_id	specialty_id	4	student_admission.S.specialty_id	66733	100.00	NULL

explain запроса 2.1.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	D	NULL	index	PRIMARY	PRIMARY	4	NULL	11	100.00	NULL
1	SIMPLE	S	NULL	ref	department_id	department_id	4	student_admission.D.department_id	1	100.00	Using index
1	SIMPLE	C	NULL	ref	specialty_id	specialty_id	4	student_admission.S.specialty_id	66733	100.00	NULL

explain запроса 3.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	A	NULL	ALL	enrollee_id,subject_id	NULL	NULL	NULL	798336	100.00	Using temporary
1	PRIMARY	C	NULL	ref	enrollee_id,specialty_id	enrollee_id	4	student_admission.A.enrollee_id	2	100.00	NULL
1	PRIMARY	SR	NULL	ref	subject_id,specialty_id	subject_id	4	student_admission.A.subject_id	3	10.00	Using where
2	SUBQUERY	<derived3>	NULL	ALL	NULL	NULL	NULL	NULL	747635	100.00	NULL
3	DERIVED	A	NULL	ALL	enrollee_id,subject_id	NULL	NULL	NULL	798336	100.00	Using temporary
3	DERIVED	C	NULL	ref	enrollee_id,specialty_id	enrollee_id	4	student_admission.A.enrollee_id	2	100.00	NULL
3	DERIVED	SR	NULL	ref	subject_id,specialty_id	subject_id	4	student_admission.A.subject_id	3	10.00	Using where

explain запроса 4.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	60060	100.00	Using temporary
2	DERIVED	C	NULL	ALL	specialty_id	NULL	NULL	NULL	600600	10.00	Using where; Using temporary; Using filesort
2	DERIVED	S	NULL	eq_ref	PRIMARY,department_id	PRIMARY	4	student_admission.C.specialty_id	1	100.00	Using index

explain запроса 5.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	A	NULL	index	subject_id	subject_id	4	NULL	798336	100.00	NULL
2	SUBQUERY	A	NULL	ref	subject_id	subject_id	4	const	137916	100.00	NULL

explain запроса 6.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	E	NULL	ALL	NULL	NULL	NULL	NULL	198806	100.00	NULL
1	SIMPLE	<subquery2>	NULL	eq_ref	<auto_distinct_key>	<auto_distinct_key>	5	student_admission.E.enrollee_id	1	100.00	Using where; Not exists
2	MATERIALIZED	S	NULL	ref	PRIMARY,department_id	department_id	4	const	3	100.00	Using index
2	MATERIALIZED	C	NULL	ref	enrollee_id,specialty_id	specialty_id	4	student_admission.S.specialty_id	66733	100.00	NULL
2	MATERIALIZED	Enrollee	NULL	eq_ref	PRIMARY	PRIMARY	4	student_admission.C.enrollee_id	1	100.00	Using index

explain запроса 7.

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	D	NULL	ALL	NULL	NULL	NULL	NULL	11	100.00	Using temporary; Using filesort
1	SIMPLE	Sp	NULL	ref	department_id	department_id	4	student_admission.D.department_id	1	100.00	Using index
1	SIMPLE	S	NULL	ALL	NULL	NULL	NULL	NULL	11	100.00	Using join buffer (hash join)
1	SIMPLE	SR	NULL	ref	subject_id,specialty_id	subject_id	4	student_admission.S.subject_id	3	100.00	Using where
1	SIMPLE	C	NULL	ref	specialty_id	specialty_id	4	student_admission.Sp.specialty_id	66733	100.00	NULL

explain запроса 8 (a).

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	S	NULL	index	department_id	department_id	4	NULL	10	100.00	Using index; Using temporary; Using filesort
1	SIMPLE	D	NULL	eq_ref	PRIMARY	PRIMARY	4	student_admission.S.department_id	1	100.00	NULL

explain запроса 8 (б).

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	S	NULL	ref	PRIMARY,department_id	department_id	4	const	1	100.00	Using index
1	SIMPLE	SR	NULL	ref	specialty_id	specialty_id	4	student_admission.S.specialty_id	3	100.00	NULL

explain запроса 8 (в).

4. Заключение

В рамках курсовой работы, были получены навыки работы с MySQL 3.8. изучен синтаксис и особенности языков определения и манипулирования данными. Также были получены навыки построения ER-диаграммы, диаграммы объектов и навыки проектирования баз данных, анализа и вывода команды explain.

В рамках разработки базы данных “Электронная подача заявления на поступление в ВУЗ”:

- Разработана ER-диаграмма, которая описывает процессы, протекающие в предметной области, а также схема объектов.
- На основе построенных диаграмм была спроектирована и реализована схема базы данных.
- База данных, была заполнена случайными данными, сгенерированными программой на языке Python 3.9. (Приложение В). Заполненная база данных содержит порядка 200000 записей.
- Реализованы 9 запросов к базе данных, так же к каждому запросу приложен вывод explain метода.

Сложным запросом оказался запрос 4. Запрос работает больше 1000мс. Остальные запросы выполняются с приемлемым временем работы.

5. Источники

1. Документация pgMustard — визуализатора результатов EXPLAIN [Электронный ресурс]. – URL: https://www.pgmustard.com/docs/explain (дата обращения: 3.05.2024).
2. https://andpop.ru/courses/db_books/Dubua.pdf (дата обращения: 15.05.2024).

6. Приложение А

```
# 1) найти дисциплину который сдавал абитуриент А который поступал по
направлению В
select S.subject_id, S.name
from Choice C
join Achievment A on C.enrollee_id = A.enrollee_id
join Specialty Sp on C.specialty_id = Sp.specialty_id
join Subject_Requirement SR on Sp.specialty_id = SR.specialty_id and
SR.subject_id = A.subject_id
join Subject S on SR.subject_id = S.subject_id
where C.enrollee_id = 457322 and C.specialty_id = 3;


# 2) из института А посчитать число абитуриентов
select S.department_id, count(DISTINCT C.enrollee_id)
from Choice C
join Specialty S on C.specialty_id = S.specialty_id
where S.department_id = 1;

# 2.2) для института А посчитать число абитуриентов по каждому направлению
select C.specialty_id, count(DISTINCT C.enrollee_id)
from Choice C
join Specialty S on C.specialty_id = S.specialty_id
where S.department_id = 1
group by C.specialty_id;

# 3) с каждого института число подавших абитуриентов + к этому гистограмму
select D.name as department_name, count(DISTINCT C.enrollee_id) as
enrollee_count
from Department D
left join Specialty S on S.department_id = D.department_id
left join Choice C on C.specialty_id = S.specialty_id
group by D.department_id;

# 4) найти абитуриента с максимальным числом результата поступления
select C.enrollee_id, C.specialty_id, sum(greatest(A.ege_result,
A.inner_result)) as result
from Choice C
join Subject_Requirement SR on C.specialty_id = SR.specialty_id
join Achievment A on A.enrollee_id = C.enrollee_id and A.subject_id =
SR.subject_id
group by C.specialty_id, C.enrollee_id
having result = (select max(result) from (select sum(greatest(A.ege_result,
A.inner_result)) as result
    from Choice C
    join Subject_Requirement SR on C.specialty_id = SR.specialty_id
    join Achievment A on A.enrollee_id = C.enrollee_id and A.subject_id =
SR.subject_id
    group by C.enrollee_id, C.specialty_id) results);

# 5) число направлений с одинаковым числом абитуриентов
select enrollees_count, count(enrollees_count) as specialty_count
from (select count(DISTINCT C.enrollee_id) enrollees_count
    from Choice C
    join Specialty S on C.specialty_id = S.specialty_id
    where C.priority_index = 2
    group by S.specialty_id) as counts
group by enrollees_count;
```

```
# 6) найти дисциплину по которой подано больше результатов чем у дисциплины A
select A.subject_id, count(A.enrollee_id) as count
from Achievment A
group by A.subject_id
having count > (select count(A.enrollee_id)
from Achievment A
where A.subject_id = 23
group by A.subject_id);

# 7) найти абитуриентов которые не подавались в институт A
select E.last_name, E.first_name
from Enrollee E
where E.enrollee_id not in (select distinct C.enrollee_id
from Enrollee
join Choice C on C.enrollee_id = Enrollee.enrollee_id
left join Specialty S on C.specialty_id = S.specialty_id
where S.department_id = 2);

# 8) для всех институтов и всех дисциплин посчитать число абитуриентов
select S.name as subject_name, D.name as department_name, count(distinct
C.enrollee_id) as enrollee_count
from Subject S
cross join Department D
left join Specialty Sp on D.department_id = Sp.department_id
left join Subject_Requirement SR on S.subject_id = SR.subject_id and
Sp.specialty_id = SR.specialty_id
left join Choice C on Sp.specialty_id = C.specialty_id
group by S.subject_id, D.department_id;

select D.name as department_name, count(distinct specialty_id)
from Specialty S
join Department D on S.department_id = D.department_id
group by D.department_id;

select S.specialty_id, SR.subject_id
from Subject_Requirement SR
join Specialty S on SR.specialty_id = S.specialty_id
where department_id = 4
```

7. Приложение В

```
import random
from datetime import datetime, timedelta

import mysql.connector

def generate_random_date(start_year, end_year):
    start_date = datetime(year=start_year, month=1, day=1)
    end_date = datetime(year=end_year, month=12, day=31)
    delta = end_date - start_date
    random_date = start_date + timedelta(days=random.randint(0, delta.days))
    return random_date.strftime('%Y-%m-%d')

def generate_random_name(names):
    return random.choice(names)

def generate_random_phone_number():
    return random.randint(1000000000, 9999999999)

def generate_random_result():
    return random.randint(1, 100)

def get_existing_enrollee_ids(cursor):
    """Получает существующие enrollee_id из таблицы Enrollee."""
    cursor.execute("SELECT enrollee_id FROM Enrollee")
    return [row[0] for row in cursor.fetchall()]

def get_existing_department_ids(cursor):
    """Получает существующие department_id из таблицы Department."""
    cursor.execute("SELECT department_id FROM Department")
    return [row[0] for row in cursor.fetchall()]

ENROLLEE_COUNT = 200_000

with mysql.connector.connect(
        host='127.0.0.1',
        user='root',
        password='Ayzek123321',
        database='student_admission',
        port='3306') as connection:
    if not connection.is_connected():
        print("Not connected. Bye")
        exit(1)

    cursor = connection.cursor()
    connection.autocommit = False

    first_names = ['Айзек', 'Петя', 'Виктор', 'Михаил', 'Ланита', 'Игорь',
'Светлана', 'Ирина', 'Ольга', 'Владимир']
    last_names = ['Салимли', 'Григорьев', 'Кузнецов', 'Капустин', 'Сильванович',
'Маликовн', 'Бакинец', 'Долгов', 'Непомнящих', 'Истов']
    middle_names = ['Петрович', 'Владимирович', 'Александрович', 'Геннадьевич',
'Михайлович', 'Павлович', 'Алексеевич', 'Викторович', 'Иванович', 'Рустамович']

    adrespoly = ['Политехническая 29', 'Гидротехников 2', 'Политехническая 21',
'Обручевых 1', 'Гжатская 4','Политехническая 34','Новороссийская 50','Верности
5','Гражданский пр. 7','Верности 4','Непокоренных 6']
    document_types = ['Паспорт', 'СНИЛС', 'Аттестат']
```

```
    department_names = ['ИКНК', 'ГИ', 'ФизМех', 'ИБСиБ', 'ИМПП', 'ИММиТ', 'ИЭ',
'ИСИ', 'ЯИ', 'ИФКиС', 'ИПМЭиТ']
    subject_names = ['Математика', 'Физика', 'Химия', 'Биология', 'English',
'История', 'География',
                     'Информатика', 'Экономика', 'Литература', 'Музыка']
    specialty_names = ['МКН', 'ПИ', 'ПМиФ', 'БФ', 'БИ', 'РиОС',
                       'ЗР', 'МС', 'ЯЭ', 'Эко']
    specialty_ids = range(1, len(specialty_names) + 1)

    # Вставка данных в таблицу Subject
    query = "INSERT INTO Subject (name) VALUES (%s)"
    for subject_name in subject_names:
        cursor.execute(query, (subject_name,))

    query = "INSERT INTO Enrollment_Status (name) VALUES (%s)"
    for status in ["added", "cancelled"]:
        cursor.execute(query, (status,))

    cursor.execute("SELECT enrollment_status_id FROM Enrollment_Status")
    existing_status_ids = [row[0] for row in cursor.fetchall()]
    cursor.execute("SELECT subject_id FROM Subject")
    existing_subject_ids = [row[0] for row in cursor.fetchall()]

    # Вставка данных в таблицу Enrollee
    query = ("INSERT INTO Enrollee (first_name, last_name, middle_name,
birth_date, enrollment_status_id)"
             "VALUES (%s, %s, %s, %s, %s)")
    cursor.executemany(query, tuple(
        (first_name := generate_random_name(first_names),
         last_name := generate_random_name(last_names),
         middle_name := generate_random_name(middle_names),
         birth_date := generate_random_date(1980, 2005),
         status := random.choice(existing_status_ids)) for i in
range(ENROLLEE_COUNT)
    ))

    # Вставка данных в таблицу Department
    query = "INSERT INTO Department (name, phone_number, address,
headmaster_name) VALUES (%s, %s, %s, %s)"
    for i, dept_name in enumerate(department_names):
        phone_number = generate_random_phone_number()
        address = random.choice(adrespoly)
        headmaster_name = generate_random_name(last_names)

        cursor.execute(query, (dept_name, phone_number, address,
headmaster_name))

    # Вставка данных в таблицу Document_Type
    query = "INSERT INTO Document_Type (name) VALUES (%s)"
    for document_type_name in document_types:
        cursor.execute(query, (document_type_name,))

    # Вставка данных в таблицу Specialty
    existing_department_ids = get_existing_department_ids(cursor)
    query = "INSERT INTO Specialty (department_id, code, name) VALUES (%s, %s,
%s)"
    for i in range(len(specialty_names)):  # Используем len(specialty_names),
чтобы цикл не выходил за пределы списка
        # Выбор случайного существующего department_id
```

```
        department_id = random.choice(existing_department_ids)
        specialty_code = f"Code_{i + 1}"
        specialty_name = specialty_names[i]

        cursor.execute(query, (department_id, specialty_code, specialty_name))

    # Получение существующих enrollee_id из таблицы Enrollee
    existing_enrollee_ids = get_existing_enrollee_ids(cursor)
    if not existing_enrollee_ids:
        raise ValueError("No existing enrollee_id found in the Enrollee table")

    cursor.execute("SELECT document_type_id FROM Document_Type")
    existing_document_type_ids = [row[0] for row in cursor.fetchall()]

    # Вставка данных в таблицу Document
    query = "INSERT INTO Document (enrollee_id, document_type_id) VALUES (%s,
%s)"
    def docs():
        for enrollee_id in existing_enrollee_ids:
            # Вставка для каждого абитуриента по 1-3 документа
            document_count = random.randint(1, 3)
            for document_type_id in random.sample(existing_document_type_ids,
document_count):
                yield enrollee_id, document_type_id
    cursor.executemany(query, tuple(docs()))

    # Вставка данных в таблицу Achievment
    query = ("INSERT INTO Achievment (enrollee_id, subject_id, ege_result,
inner_result)"
             "VALUES (%s, %s, %s, %s)")
    def achievments():
        for enrollee_id in existing_enrollee_ids:
            # Вставка для каждого абитуриента по 3-5 результатов испытаний
            for subject_id in random.sample(existing_subject_ids,
random.randint(3, 5)):
                ege_result = generate_random_result()
                inner_result = generate_random_result()
                yield enrollee_id, subject_id, ege_result, inner_result
    cursor.executemany(query, tuple(achievments()))

    # Вставка данных в таблицу Choice
    query = "INSERT INTO Choice (enrollee_id, specialty_id, priority_index)
VALUES (%s, %s, %s)"
    def choices():
        for enrollee_id in existing_enrollee_ids:
            # Вставка для каждого абитуриента по 1-5 выборов
            for j, specialty_id in enumerate(random.sample(specialty_ids,
random.randint(1, 5))):
                priority_index = j + 1
                yield enrollee_id, specialty_id, priority_index
    cursor.executemany(query, tuple(choices()))

    # Вставка данных в таблицу Subject_Requirement
    query = ("INSERT INTO Subject_Requirement (subject_id, specialty_id,
ege_minimal_result, inner_minimal_result)"
             "VALUES (%s, %s, %s, %s)")
    for specialty_id in specialty_ids:
        for subject_id in random.sample(existing_subject_ids, random.randint(3,
4)):
```

```
            ege_minimal_result = generate_random_result()
            inner_minimal_result = generate_random_result()
            cursor.execute(query, (subject_id, specialty_id, ege_minimal_result,
inner_minimal_result))

    connection.commit()
    print("Данные успешно заполнены.")

print("Соединение с базой данных закрыто.")
```