­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.**](#AnalisPO)3

1.1 [Предметная область](#PredmetnayaOblast) 3

1.2 [Определения](#Opredelenie)  4

1.3 [Цели создания](#CeliSozdaniya) 5

1.4 [ER-диаграмма](#ER) 5

1.5 [Описание сущностей и атрибутов](#OpisanieSushnostey) 7

1.6 [Схема объектов](#ShemaObjectov) 8

**2.** [**Проектирование базы данных**](#ProectirovanieBD)9

2.1 [Атрибуты таблиц базы данных](#AtributiTablicBD)  9

2.2 [Размеры полей](#RazmeriPoley)  11

2.3 [Заполнение базы данных](#ZapolnenieTablic) 14

**3.** [**Запросы**](#Zaprosi)15

3.1 [Дисциплина которую сдавал абитуриент А который поступал по направлению В](#Z31) 15

3.2 [Число абитуриентов из института А](#Z32) 16

3.2.2 [Число абитуриентов по каждому направлению института А](#Z322)  16

3.3 [Число подавших абитуриентов с каждого института](#Z35) 16

3.4 [Абитуриент с максимальным числом результата](#Z34) 17

3.5 [Число направлений с одинаковым числом абитуриентов](#Z35) 18

3.6 [Дисциплина по которой подано больше результатов чем у дисциплины А](#Z36) 18

3.7 [Абитуриенты которые не подавались в институт А](#Z37) 19

3.8 [Для всех институтов и всех дисциплин посчитать число абитуриентов](#Z38) 19

3.9 [Explain запросов](#Explain) 21

**4.** [**Заключение**](#Zakluchenie)24

**5.** [**Источники**](#Istochniki) 25

**6.** [**Приложение А**](#PrilojenieA)26

**7.** [**Приложение В**](#PrilojenieB)28

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

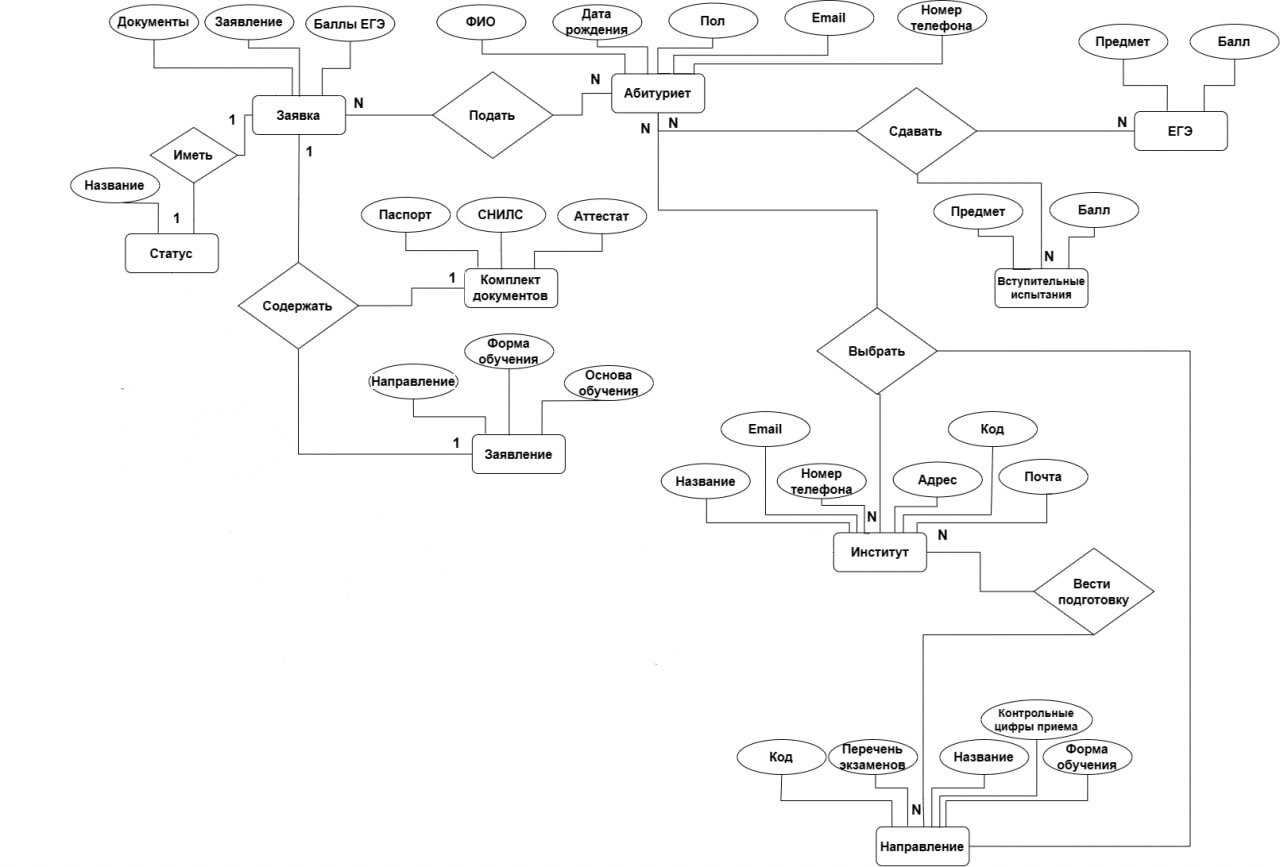
2. Конкретный институт ведет подготовку по выбранному абитуриентом направлению

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](#Image3) и [4](#Image4), 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.

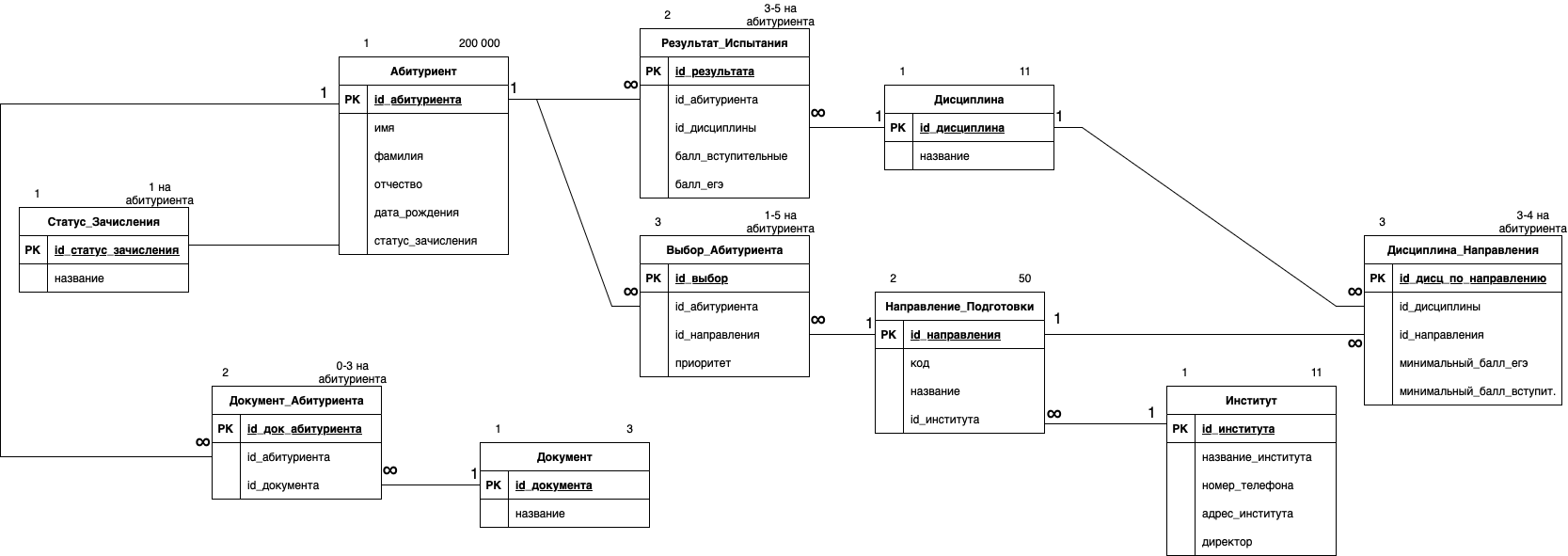
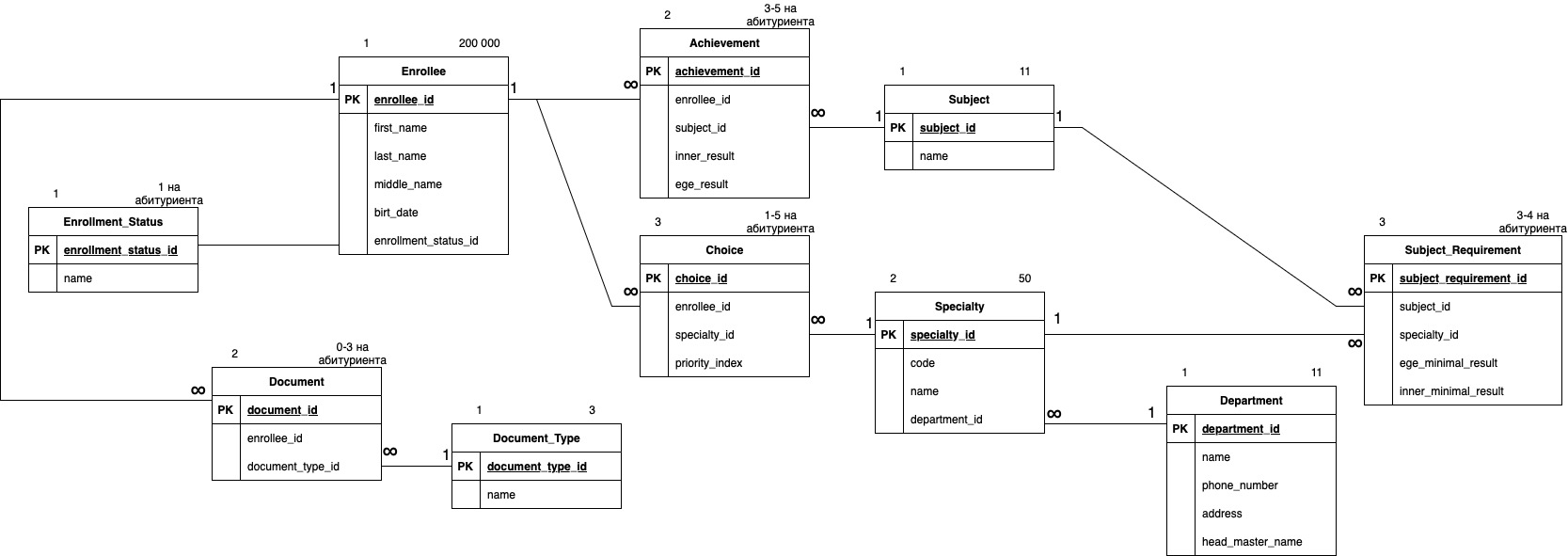
****

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 А который поступал по направлению В

2. Из института А посчитать число абитуриентов

2.2 Из института А 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 А

7. Найти абитуриентов которые не подавались в институт А

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 А 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

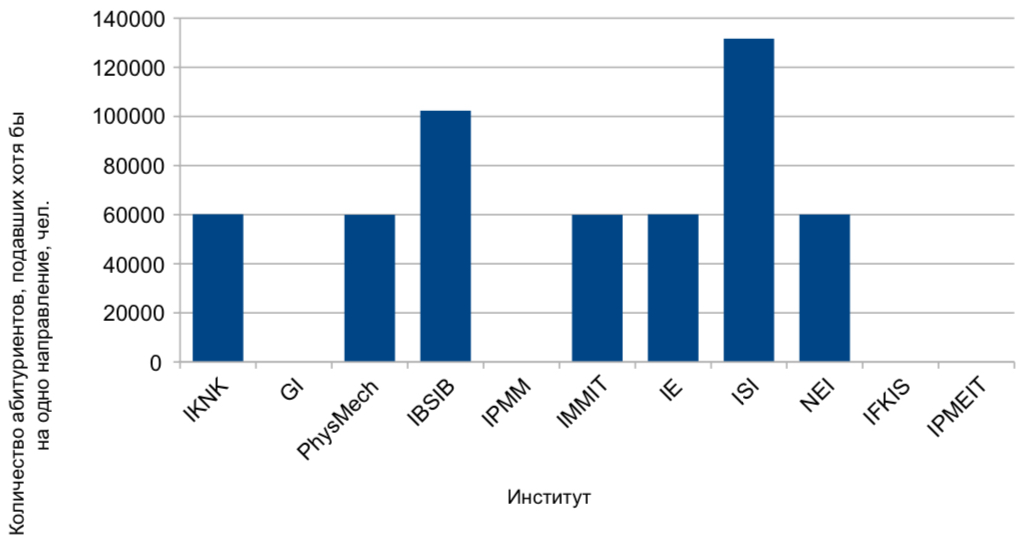
from Department D

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 А**

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 |

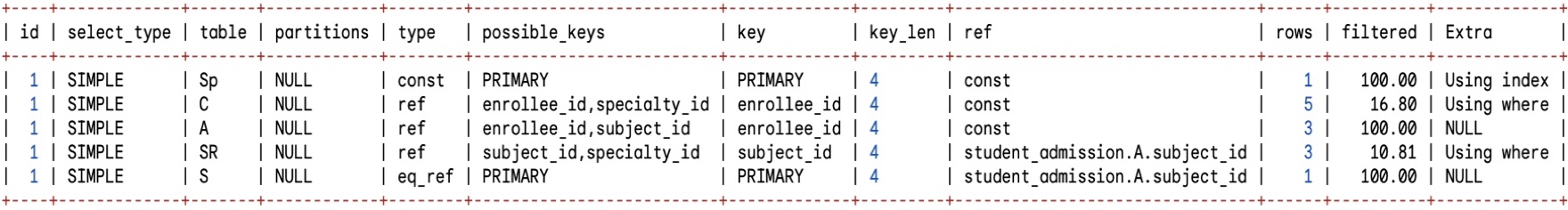
Запрос 3.8(в) (Время запроса - 0.073 сек.)

Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

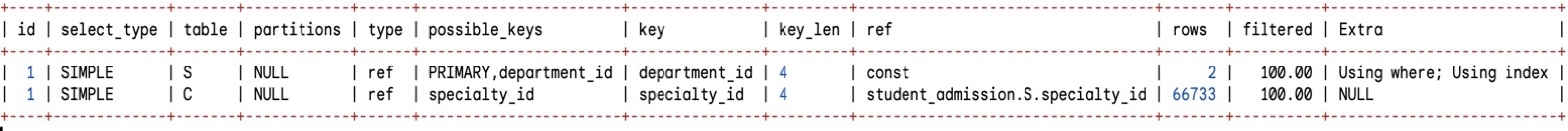
Автоматически созданное описание

Диаграмма к запросу 8(а).

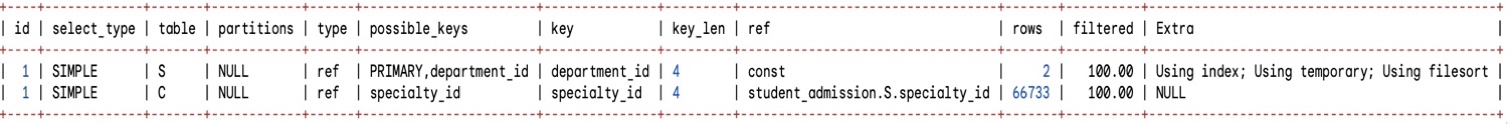
3.4 Explain



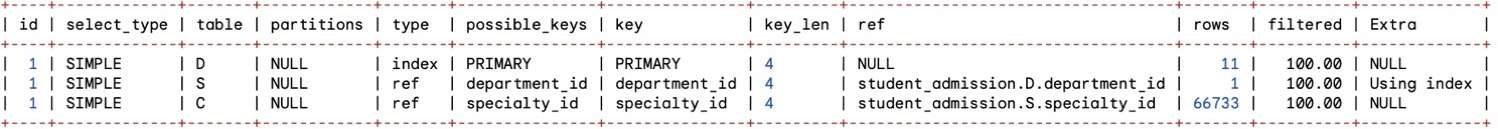
explain запроса 1.



explain запроса 2.



explain запроса 2.1.

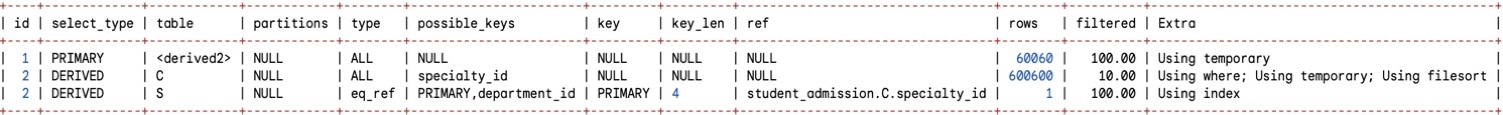


explain запроса 3.

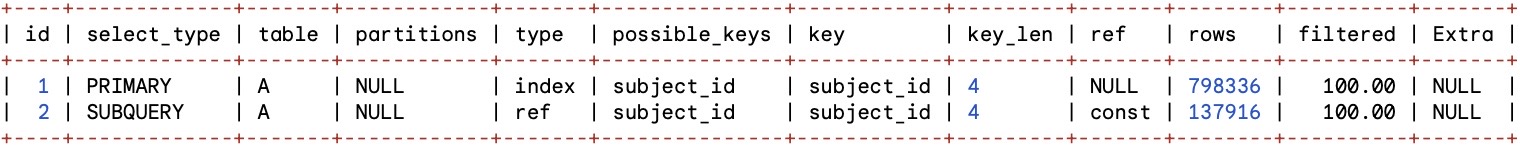
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

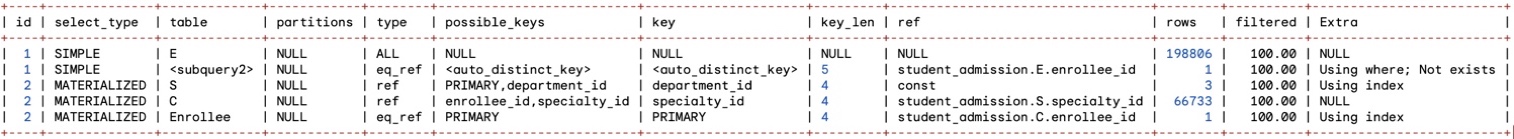
explain запроса 4.



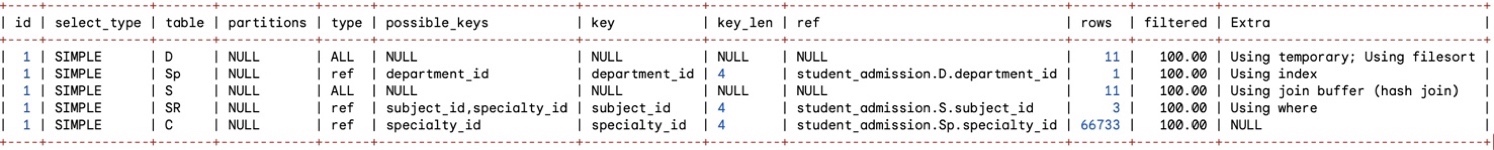
explain запроса 5.



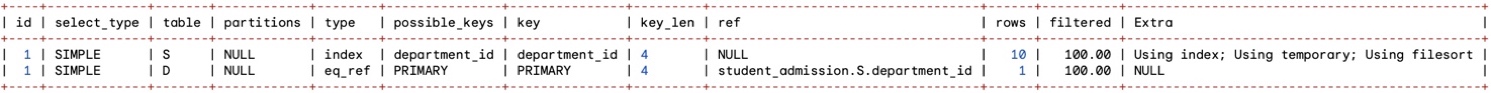
explain запроса 6.



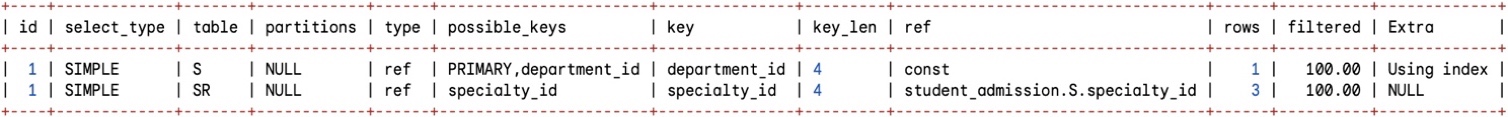
explain запроса 7.



explain запроса 8 (а).



explain запроса 8 (б).



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](#KodPB)). 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 ре- сурс]. – 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.")