НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

(повна назва інституту/факультету)

КАФЕДРА інформатики та програмної інженерії

(повна назва кафедри)

**КУРСОВА РОБОТА**

з дисципліни «Бази даних»

(назва дисципліни)

на тему:\_\_База даних приймальної комісії університету\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студента (ки) \_\_2\_\_ курсу \_ІП-13\_\_\_\_\_ групи

спеціальності 121 «Інженерія програмного забезпечення»

\_Замкового Дмитра Володимировича\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_Ліщук Олександр Васильович\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2022 рік

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет Інформатики та обчислювальної техніки

(повна назва)

Кафедра Інформатики та програмної інженерії

(повна назва)

Дисципліна Бази даних

Курс \_\_2\_\_\_ Група \_\_ІП-13\_\_ Семестр \_\_\_3\_\_

**З А В Д А Н Н Я**

**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

\_\_\_\_\_\_Замковому Дмитру Володимировичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові)

1. Тема роботи \_\_\_\_\_База даних приймальної комісії університету\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

керівник роботи Ліщук Олександр Васильович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

2. Строк подання студентом роботи \_\_\_\_\_08.07.2023\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Вихідні дані до роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_завдання на розробку бази даних для зарахування абітурієнтів приймальною комісією університету\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

4) Створення бази даних, у форматі обраної системи управління базою даних

5) Створення користувачів бази даних

6) Імпорт даних з використанням засобів СУБД в створену базу даних

7) Створення мовою SQL запитів

8) Оптимізація роботи запитів

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Дата видачі завдання\_\_\_\_08.11.2022**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| №  з/п | Назва етапів виконання курсового проекту | Строк виконання етапів проекту | Примітка |
| 11 | Аналіз предметного середовища | 12.11.2022 |  |
| 22 | Побудова ER-моделі | 21.11.2023 |  |
| 33 | Побудова реляційної схеми з ER-моделі | 01.12.2022 |  |
| 44 | Створення бази даних, у форматі обраної системи управління базою даних | 01.12.2022 |  |
| 55 | Створення користувачів бази даних | 08.01.2023 |  |
| 66 | Імпорт даних з використанням засобів СУБД в створену базу даних | 20.12.2022 |  |
| 77 | Створення мовою SQL запитів | 07.01.2023 |  |
| 88 | Оптимізація роботи запитів | 08.01.2023 |  |
| 99 | Оформлення пояснювальної записки | 08.01.2023 |  |
| 110 | Захист курсової роботи | 11.01.2022 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Студент \_\_\_\_\_\_\_\_\_\_\_\_ \_\_**Замковий Д.В.\_\_\_\_\_\_\_

(підпис ) (прізвище та ініціали)

**Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Ліщук О.В.**\_\_\_\_\_\_\_\_**

(підпис ) (прізвище та ініціали)

**ЗМІСТ**

[Вступ 6](#_Toc124041478)

[1. Опис предметного середовиша 7](#_Toc124041479)

[2. Ппостановка завдання 8](#_Toc124041480)

[3. Проектування бази даних 9](#_Toc124041481)

[3.1. Побудова ER-моделі 9](#_Toc124041482)

[3.1.1. Бізнес-правила 9](#_Toc124041483)

[3.1.2. Вибір сутностей 9](#_Toc124041484)

[3.1.3. Набори атрибутів сутностей 10](#_Toc124041485)

[3.2. Даталогічна модель бази даних 14](#_Toc124041486)

[4. Реалізація бази даних 16](#_Toc124041487)

[4.1. Створення бази даних у форматі СУБД PostgreSQL 16](#_Toc124041488)

[4.1.1. Створення таблиць 16](#_Toc124041489)

[4.1.2. Створення зав’язків 17](#_Toc124041490)

[4.1.3. Створення обмежень 19](#_Toc124041491)

[4.2. Імпортування даних у таблиці 19](#_Toc124041492)

[5. Робота з базою даних 20](#_Toc124041493)

[5.1. Створення користувачів 20](#_Toc124041494)

[5.1.1. Абітурієнт 20](#_Toc124041495)

[5.1.2. Член приймальної комісії 20](#_Toc124041496)

[5.1.3. Декан 21](#_Toc124041497)

[5.2. SQL запити 22](#_Toc124041498)

[5.2.1. Тригери, що будуть змінюватись користувачами 22](#_Toc124041499)

[5.2.2. Процедури та функцій для взаємодії користувачів з базою даних 24](#_Toc124041500)

[5.2.3. Представлення 30](#_Toc124041501)

[5.2.4. DML-запити 31](#_Toc124041502)

[5.2.6. Індекси 44](#_Toc124041503)

[Висновки 46](#_Toc124041504)

[Список використаної літератури 47](#_Toc124041505)

[Додаток А. Тексти INSERT запитів 48](#_Toc124041506)

# ВСТУП

Створення власної бази даних є основою будь-якої компанії. Ці бази даних були дуже громіздкими архівами, тому доводилося переглядати гори різних паперів, щоб знайти інформацію, що цікавить. Також такий великий обсяг погано структуризованої та неавтоматизованої . Для їх заміни було створено цифрові бази даних, у яких дані були організовані як таблиць. Це значно прискорило пошук та обробку потрібної нам інформації.

Обрана мною тема є актуальною, оскільки це дасть можливість прискорити та оптимізувати роботу приймальної комісії Університету.

Метою цієї роботи є створення бази даних для приймальної комісії Університету, яка дасть можливість спростити роботу працівників та надати можливість абітурієнтам подавати всі дані дистанційно.

1. ОПИС ПРЕДМЕТНОГО СЕРЕДОВИЩА

Структура університету передбачує в собі наявність декілька різних факультетів, а кожен факультет в свою чергу має декілька різних кафедр.

Кожен факультет характеризується повною та скороченою назвою, кількістю бюджетних та контрактних місць та номером телефону. Кафедра в свою чергу - Факультетом, до якого вона прив'язана, спеціальністю назвою та номером.

Також абітурієнт при подачі заяви вибирає одну з наведених спеціальностей по номеру та назві. Подані заяви абітурієнтом характеризуються пріоритетом, вступним балом та курсом вступу.

При подачі документів абітурієнт також подає результати свого ЗНО, які розрізняються предметом з вибраного списку та балом, який він за них отримав. Самого абітурієнта можна охарактеризувати за допомогою таких параметрів: ПІБ, дата народження, номер телефону, емейлом (електронною адресою), середнім балом атестату, конкурсним балом, наявністю чи відсутністю квоти, чи виконав абітурієнт умови вступу та чи потребує він місце в гуртожитку.

При зарахуванні приймальною комісією створюється запис в таблиці "Зараховані абітурієнти", а саме: ідентифікатором студента (згенерованим при записі), заявою, по якій він вступив та інформацією про те, чи має отримувати він стипендію.

Після закінчення вступної кампанії декан має змогу переглянути список зарахованих та сформувати наказ про зарахування.

1. ПОСТАНОВКА ЗАВДАННЯ

Метою даної лабораторної роботи є розробка бази даних для приймальної комісії Університету. Тобто організація даних таким чином, щоб робота приймальної комісії виконувалась максимально швидко та ефективно. Отже основні задачі:

Абітурієнт повинен мати змогу:

* Подавати документи та інформацію потрібну для вступу
* Вибирати пріоритет для поданих заяв
* Переглядати список всіх поданих заяв
* Переглядати чи було його прийнято за будь-якою заявою

Приймальна комісія перед тим як прийняти студента має мати змогу:

* Переглянути документи та інформацію подану абітурієнтом
* Обраховувати вступний бал абітурієнта
* Переглядати список всіх заяв та пріоритетів, які подавав абітурієнт
* Позначати заяву, як зараховану
* Формувати текст листа, для повідомлення студента про зарахування

Декан, для підписання наказу про зарахування, повинен мати змогу переглядати список усіх абітурієнтів, яких відібрала приймальна комісія

* Переглядати список всіх зарахованих абітурієнтів
* Сформувати текст наказу про вступ

1. ПРОЕКТУВАННЯ БАЗИ ДАНИХ
   1. Побудова ER-моделі

Після аналізу було виділено такі сутності та зв’язки між ними:

* + 1. Бізнес-правила
* Номери телефоні мають відповідати міжнародному формату, а саме лежати в межах від 100000000000 до 999999999999;
* Результат ЗНО та вступний бал може бути в межах від 100 до 200;
* Вік абітурієнта має бути не меншим як 16 років;
  + 1. Вибір сутностей
* Зарахований абітурієнт
* Спеціальність
* Заява
* Пріоритет по кафедрам
* Кафедра
* Факультет
* Документ
* Абітурієнт
* Результат ЗНО
* Предмет ЗНО
  + 1. Набори атрибутів сутностей

Таблиця 3.1 – Сутності та їхні атрибути

|  |  |
| --- | --- |
| Сутність | Атрибут |
| enrolled\_abiturient | id  id\_statement  scholarship |
| specialty | id  specialty\_name |
| statements | id  id\_abiturient  id\_specialty  priorities  score  course |
| department\_priority | id  id\_department  id\_statement  priorities |
| department | id  id\_faculty  department\_name  phone\_number |
| faculty | id  full\_name  short\_name  phone\_number |

Продовження таблиці 3.1

|  |  |
| --- | --- |
| document | id  id\_abiturient  short\_description  document\_url |
| abiturient | id  first\_name  last\_name  fathers\_name  birthday  phone\_number  email  certificate\_score  quota  is\_conditions  is\_dormitory |
| zno\_result | id  id\_abiturient  id\_subject  result |
| zno\_list | id  subject\_name  min\_score |

Сутність enrolled\_abiturient буде пов’язана один до багатьох з сутністю statements, бо одна заява може бути прийнята тільки один раз.

Сутність specialty буде пов’язана один до багатьох з сутністю statements, бо одна заява може тільки одну спеціальність.

Сутність statements буде пов’язана один до багатьох з сутністю department\_priority, бо одна заява може мати декілька пріоритетів на зарахування по кафедрам.

Сутність department буде пов’язана один до багатьох з сутністю department\_priority, бо по одній кафедрі може бути декілька пріоритетів.

Сутність faculty буде пов’язана один до багатьох з сутністю department, бо один факультет може містити в собі багато кафедр.

Сутність abiturient буде пов’язана один до багатьох з сутністю statements, бо одна у одного абітурієнта може бути декілька заяв.

Сутність abiturient буде пов’язана один до багатьох з сутністю document, бо один абітурієнт може подати декілька заяв.

Сутність abiturient буде пов’язана один до багатьох з сутністю zno\_result, бо один абітурієнт має декілька результатів ЗНО.

Сутність zno\_list буде пов’язана один до багатьох з сутністю zno\_result, бо по одному предмету може бути багато результатів у різних абітурієнтів.

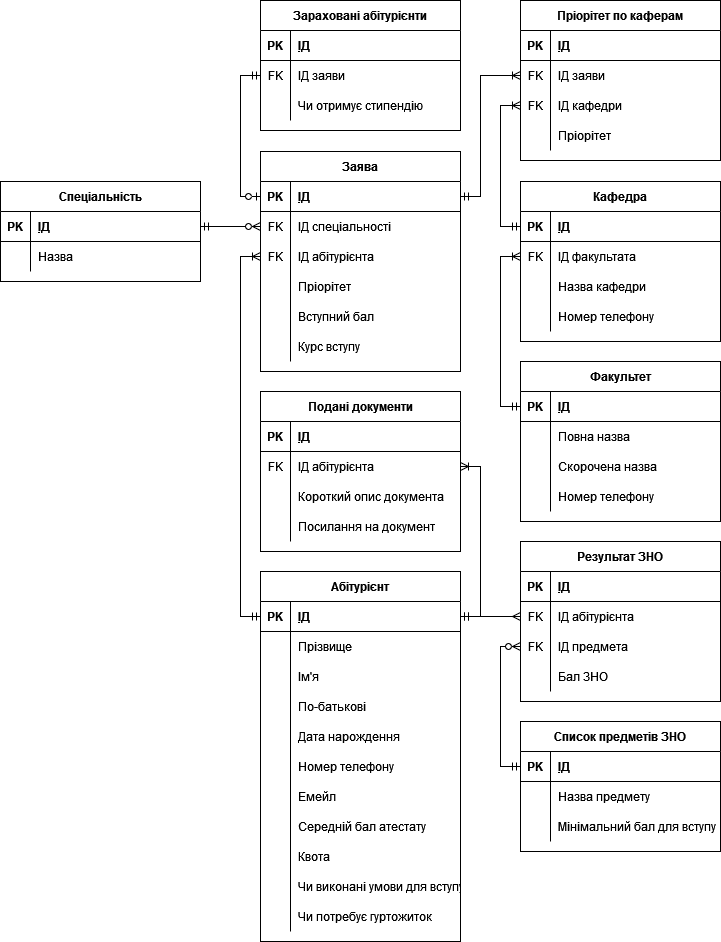


Рисунок 3.1 – ER-діаграма

* 1. Даталогічна модель бази даних
     1. Побудова необхідних відношень та визначення первинних і зовнішніх ключів

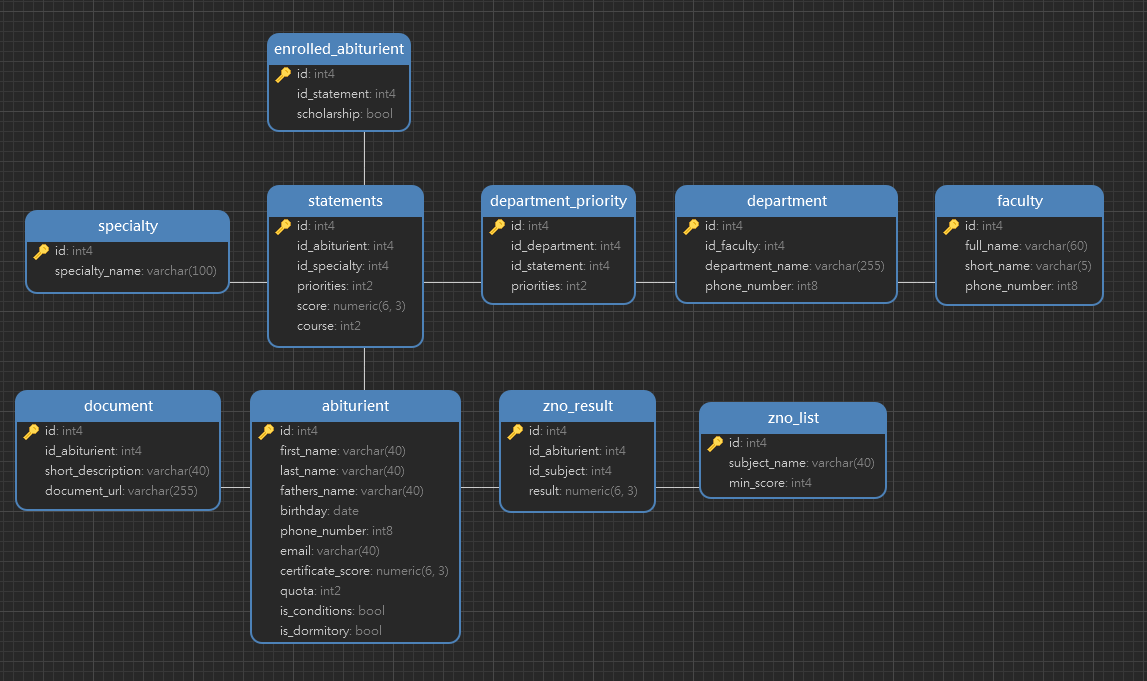


Рисунок 3.2 – Даталогічна модель бази даних

На даній схемі видно, що база даних знаходиться у третій нормальній формі, адже всі поля таблиць декомпозовані та всі атрибути таблиць функціонально повно залежать від первинного ключа. Кожен неключовий атрибут не є транзитивно залежним від первинного ключа.

* Визначення обмежень цілісності для спроектованих відношень. Обмеження цілісності: Рядок батьківської таблиці може бути видалений лише у тому випадку, якщо немає зовнішніх ключів, що посилаються на значення преференційного ключа цього рядка. Реалізовано відсутністю параметра ON DELETE {CASCADE|SET NULL} при створенні таблиці, що за умовчанням значитиме ON DELETE RESTRICT;
* Обов’язкові атрибути таблиць мають обмеження NOT NULL, для запобіганню помилок при роботі з даними.

1. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ
   1. Створення бази даних у форматі СУБД PostgreSQL
      1. Створення таблиць

CREATE TABLE ZNO\_list (

    id SERIAL,

    subject\_name VARCHAR(40) NOT NULL,

    min\_score INTEGER NOT NULL

);

CREATE TABLE abiturient (

    id SERIAL,

    first\_name VARCHAR(40) NOT NULL,

    last\_name VARCHAR(40) NOT NULL,

    fathers\_name VARCHAR(40),

    birthday DATE NOT NULL,

    phone\_number BIGINT NOT NULL UNIQUE,

    email VARCHAR(40) NOT NULL UNIQUE,

    certificate\_score NUMERIC(6, 3) NOT NULL,

    quota SMALLINT,

    is\_conditions BOOLEAN NOT NULL,

    is\_dormitory BOOLEAN NOT NULL

);

CREATE TABLE specialty (

    id SERIAL,

    specialty\_name VARCHAR(100) NOT NULL

);

CREATE TABLE faculty (

    id SERIAL,

    full\_name VARCHAR(60) NOT NULL,

    short\_name VARCHAR(5) NOT NULL,

    phone\_number BIGINT NOT NULL UNIQUE

);

CREATE TABLE ZNO\_result (

    id SERIAL,

    id\_abiturient INTEGER,

    id\_subject INTEGER,

    result NUMERIC(6, 3) NOT NULL

);

CREATE TABLE document (

    id SERIAL,

    id\_abiturient INTEGER,

    short\_description VARCHAR(40) NOT NULL,

    document\_url VARCHAR(255) NOT NULL

);

CREATE TABLE statements (

    id SERIAL,

    id\_abiturient INTEGER,

    id\_specialty INTEGER,

    priorities SMALLINT,

    score NUMERIC(6, 3) NOT NULL,

    course SMALLINT NOT NULL

);

CREATE TABLE department (

    id SERIAL,

    id\_faculty INTEGER,

    department\_name VARCHAR(255) NOT NULL,

    phone\_number BIGINT NOT NULL UNIQUE

);

CREATE TABLE enrolled\_abiturient (

    id SERIAL,

    id\_statement INTEGER,

    scholarship BOOLEAN NOT NULL

);

CREATE TABLE department\_priority (

    id SERIAL,

    id\_department INTEGER,

    id\_statement INTEGER,

    priorities SMALLINT

);

* + 1. Створення зав’язків

ALTER TABLE ZNO\_result

ADD PRIMARY KEY (id);

ALTER TABLE ZNO\_list

ADD PRIMARY KEY (id);

ALTER TABLE enrolled\_abiturient

ADD PRIMARY KEY (id);

ALTER TABLE abiturient

ADD PRIMARY KEY (id);

ALTER TABLE ZNO\_result

ADD PRIMARY KEY (id);

ALTER TABLE ZNO\_list

ADD PRIMARY KEY (id);

ALTER TABLE enrolled\_abiturient

ADD PRIMARY KEY (id);

ALTER TABLE abiturient

ADD PRIMARY KEY (id);

ALTER TABLE document

ADD PRIMARY KEY (id);

ALTER TABLE department\_priority

ADD PRIMARY KEY (id);

ALTER TABLE statements

ADD PRIMARY KEY (id);

ALTER TABLE department

ADD PRIMARY KEY (id);

ALTER TABLE specialty

ADD PRIMARY KEY (id);

ALTER TABLE faculty

ADD PRIMARY KEY (id);

ALTER TABLE ZNO\_result

ADD FOREIGN KEY (id\_abiturient) REFERENCES abiturient(id),

ADD FOREIGN KEY (id\_subject) REFERENCES ZNO\_list(id);

ALTER TABLE enrolled\_abiturient

ADD FOREIGN KEY (id\_statement) REFERENCES statements(id);

ALTER TABLE document

ADD FOREIGN KEY (id\_abiturient) REFERENCES abiturient(id);

ALTER TABLE department\_priority

ADD FOREIGN KEY (id\_department) REFERENCES department(id),

ADD FOREIGN KEY (id\_statement) REFERENCES statements(id);

ALTER TABLE statements

ADD FOREIGN KEY (id\_abiturient) REFERENCES abiturient(id),

ADD FOREIGN KEY (id\_specialty) REFERENCES specialty(id);

ALTER TABLE department

ADD FOREIGN KEY (id\_faculty) REFERENCES faculty(id);

* + 1. Створення обмежень

ALTER TABLE zno\_list

ADD CONSTRAINT zno\_list\_min\_score

CHECK(zno\_list.min\_score BETWEEN 100 AND 200);

ALTER TABLE abiturient

ADD CONSTRAINT abiturient\_phone\_number

CHECK(abiturient.phone\_number BETWEEN 100000000000 AND 999999999999),

ADD CONSTRAINT abiturient\_score

CHECK(abiturient.certificate\_score BETWEEN 100 AND 200),

ADD CONSTRAINT abiturient\_age

CHECK(EXTRACT(YEAR FROM AGE(CURRENT\_DATE, DATE '2005-11-22')) > 15);

ALTER TABLE faculty

ADD CONSTRAINT faculty\_phone\_number

CHECK(faculty.phone\_number BETWEEN 100000000000 AND 999999999999);

ALTER TABLE zno\_result

ADD CONSTRAINT zno\_result\_result

CHECK(zno\_result.result BETWEEN 100 AND 200);

ALTER TABLE statements

ADD CONSTRAINT statements\_score

CHECK(statements.score BETWEEN 100 AND 200);

ALTER TABLE department

ADD CONSTRAINT department\_phone\_number

CHECK(department.phone\_number BETWEEN 100000000000 AND 999999999999);

* 1. Імпортування даних у таблиці

Імпортувати дані у вибрану мною СУБД можна двома способами:

* Імпортувати дані з файлу csv формату;
* Імпортувати дані із файлу-скрипта sql формату.

З огляду на ці способи було створено файл-скрипт sql формату. Його зміст має однотипний текст, а саме INSERT запити наступного типу:

INSERT INTO zno\_list VALUES

(1, 'Історія України', 100),

(2, 'Математика', 125),

(3, 'Хімія', 100);

Повний текст файлу наведений у Додатку А.

1. РОБОТА З БАЗОЮ ДАНИХ
   1. Створення користувачів
      1. Абітурієнт

Текст скрипта:

CREATE ROLE abiturient NOLOGIN;

GRANT INSERT ON abiturient, "document", department\_priority, statements, zno\_result TO abiturient;

GRANT SELECT ON statements, department\_priority, department, faculty, enrolled\_abiturient TO abiturient;

GRANT EXECUTE ON PROCEDURE is\_enroled(INTEGER) TO abiturient;

GRANT EXECUTE ON PROCEDURE is\_scholarship(INTEGER) TO abiturient;

GRANT EXECUTE ON FUNCTION get\_statements(INTEGER) TO abiturient;

GRANT EXECUTE ON FUNCTION get\_id\_enrolled\_abit(INTEGER) TO abiturient;

CREATE ROLE abit\_138 LOGIN PASSWORD 'abit138password';

GRANT abiturient TO abit\_138;

Перевірка створеної ролі:

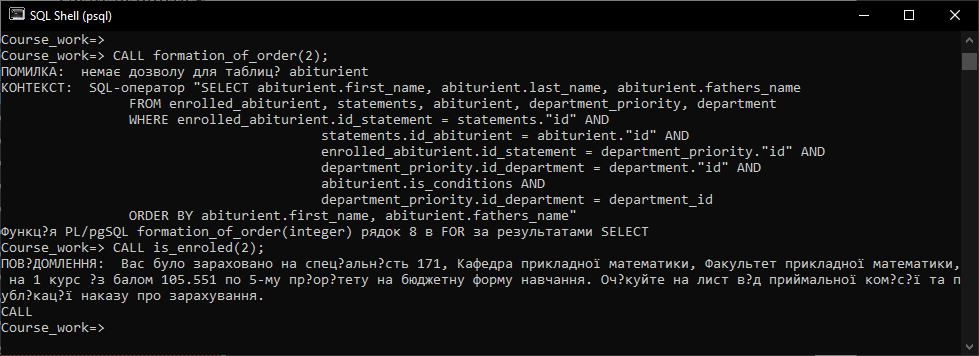


Рисунок 5.1 – Результат виконання скрипта

* + 1. Член приймальної комісії

Текст скрипта:

CREATE ROLE selection\_committee NOLOGIN;

GRANT INSERT ON enrolled\_abiturient TO selection\_committee;

GRANT SELECT ON enrolled\_abiturient, statements TO selection\_committee;

GRANT EXECUTE ON FUNCTION get\_id\_enrolled\_abit(INTEGER) TO selection\_committee;

GRANT EXECUTE ON PROCEDURE enroll\_abit(INTEGER, BOOLEAN) TO selection\_committee;

CREATE ROLE selection\_committee\_26 LOGIN PASSWORD 'SC26pass';

GRANT selection\_committee TO selection\_committee\_26;

Перевірка створеної ролі:

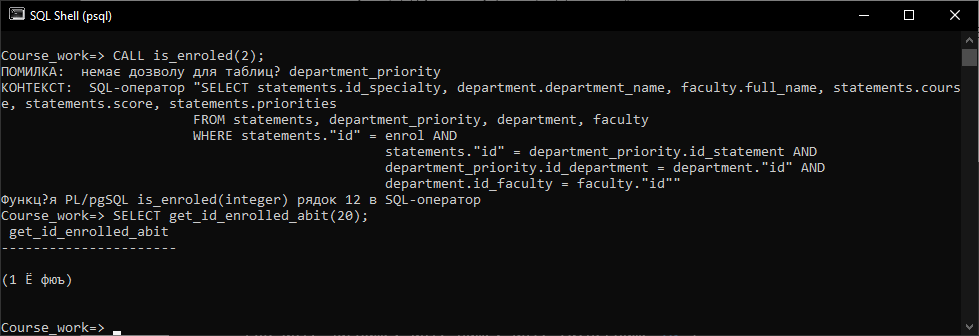


Рисунок 5.2 – Результат виконання скрипта

* + 1. Декан

Текст скрипта:

CREATE ROLE dean NOLOGIN;

GRANT SELECT ON enrolled\_abiturient, statements, abiturient, department\_priority, department TO dean;

GRANT EXECUTE ON PROCEDURE formation\_of\_order(INTEGER) TO dean;

CREATE ROLE dean\_4 LOGIN PASSWORD 'DeanJoe112233';

GRANT dean TO dean\_4;

Перевірка створеної ролі:

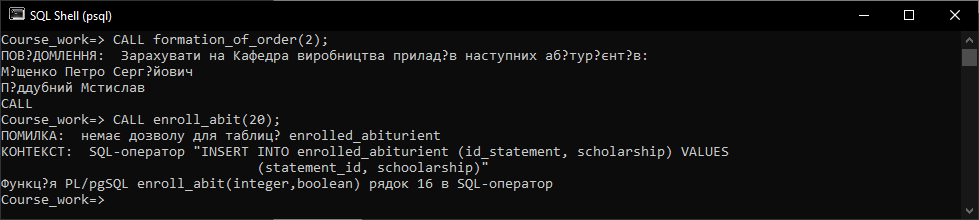


Рисунок 5.3 – Результат виконання скрипта

* 1. SQL запити
     1. Тригери, що будуть змінюватись користувачами
* Тригер, що буде генерувати повідомлення із інформацією для про зарахованого абітурієнта

Текст скрипта:

CREATE OR REPLACE FUNCTION add\_enroll\_abit() RETURNS TRIGGER

AS $$

    DECLARE

        abit\_name VARCHAR;

        abit\_lastname VARCHAR;

        abit\_fathername VARCHAR;

        abit\_email VARCHAR;

    BEGIN

        SELECT abiturient.first\_name, abiturient.last\_name, abiturient.fathers\_name, abiturient.email

        FROM statements, abiturient

        WHERE statements.id\_abiturient = abiturient."id" AND

              statements."id" = NEW.id\_statement

        INTO abit\_name, abit\_lastname, abit\_fathername, abit\_email;

        RAISE NOTICE '

            Новий абітурієнт

            ПІБ: %

            Пошта: %

            ', concat(abit\_name, ' ', abit\_lastname, ' ', abit\_fathername), abit\_email;

        RETURN NEW;

    END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER add\_enroll\_abit\_trigger

    AFTER INSERT

    ON enrolled\_abiturient

    FOR EACH ROW

EXECUTE FUNCTION add\_enroll\_abit();

Результат:

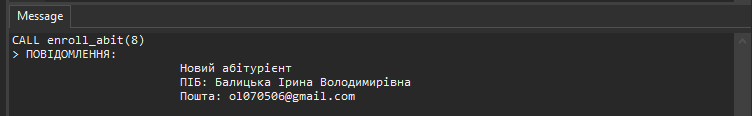


Рисунок 5.4 – Результат виконання скрипта

* Тригер, що буде генерувати текст для відправки абітурієнту листа про його зарахування

Текст скрипта:

CREATE OR REPLACE FUNCTION generate\_letter\_enroll\_abit() RETURNS TRIGGER

AS $$

    DECLARE

        abit\_name VARCHAR;

        abit\_lastname VARCHAR;

        abit\_fathername VARCHAR;

        abit\_email VARCHAR;

        abit\_course INTEGER;

        depart VARCHAR;

        facult VARCHAR;

        abit\_score "numeric"(6, 3);

    BEGIN

        SELECT abiturient.first\_name, abiturient.last\_name, abiturient.fathers\_name, abiturient.email

        FROM statements, abiturient

        WHERE statements.id\_abiturient = abiturient."id" AND

              statements."id" = NEW.id\_statement

        INTO abit\_name, abit\_lastname, abit\_fathername, abit\_email;

        SELECT statements.course, department.department\_name, faculty.short\_name, statements.score

        FROM statements, department\_priority, faculty, department

        WHERE department\_priority.id\_department = department."id" AND

              department\_priority.id\_statement = statements."id" AND

              department.id\_faculty = faculty."id" AND

              statements."id" = NEW.id\_statement

        INTO abit\_course, depart, facult, abit\_score;

        RAISE NOTICE E'SEND LETTER;\nsend\_to=%;\ntext="Шановний(на) %\\nВас зараховано на %, %, % із вступним балом %. Вам потрібно буде з\*явитися в деканаті протігом тижня з дня надходження цього листа для підписання документів.\\n\\nЗ повагою, Деканат";

            ', abit\_email, concat(abit\_name, ' ', abit\_lastname, ' ', abit\_fathername), abit\_course, depart, facult, abit\_score;

        RETURN NEW;

    END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER generate\_letter\_enroll\_abit\_trigger

    AFTER INSERT

    ON enrolled\_abiturient

    FOR EACH ROW

EXECUTE FUNCTION generate\_letter\_enroll\_abit();

Результат:

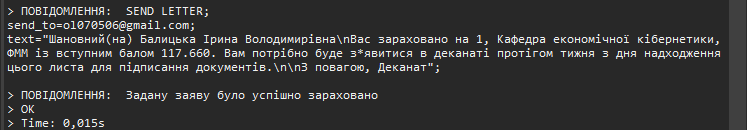


Рисунок 5.5 – Результат виконання скрипта

* + 1. Процедури та функцій для взаємодії користувачів з базою даних
       1. Процедури та функції абітурієнта
* Процедура, що повідомляє абітурієнта чи був він зарахований, та при ствердній відповіді надає більш детальну інформацію

Текст скрипта:

CREATE OR REPLACE PROCEDURE is\_enroled(IN abit\_id INTEGER)

AS $$

    DECLARE

        enrol INTEGER := get\_id\_enrolled\_abit(abit\_id);

        specialty\_id INTEGER;

        department\_name VARCHAR;

        faculty\_name VARCHAR;

        course INTEGER;

        score "numeric"(6, 3);

        priorities INTEGER;

    BEGIN

        IF (enrol IS NOT NULL) THEN

            SELECT statements.id\_specialty, department.department\_name, faculty.full\_name, statements.course, statements.score, statements.priorities

            FROM statements, department\_priority, department, faculty

            WHERE statements."id" = enrol AND

                  statements."id" = department\_priority.id\_statement AND

                  department\_priority.id\_department = department."id" AND

                  department.id\_faculty = faculty."id"

            INTO specialty\_id, department\_name, faculty\_name, course, score, priorities;

            IF (priorities IS NULL) THEN

                RAISE NOTICE 'Вас було зараховано на спеціальність %, %, %, на % курс із балом % на контрактну форму навчання. Очікуйте на лист від приймальної комісії та публікації наказу про зарахування.', specialty\_id, department\_name, faculty\_name, course, score;

            ELSE

                RAISE NOTICE 'Вас було зараховано на спеціальність %, %, %, на % курс із балом % по %-му пріорbтету на бюджетну форму навчання. Очікуйте на лист від приймальної комісії та публікації наказу про зарахування.', specialty\_id, department\_name, faculty\_name, course, score, priorities;

            END IF;

        ELSE

            RAISE NOTICE 'Вас ще не було зараховано';

        END IF;

    END;

$$ LANGUAGE plpgsql;

Результат:

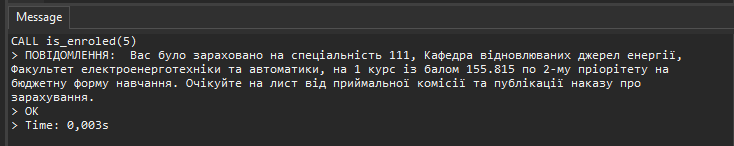


Рисунок 5.6 – Результат виконання скрипта

* Процедура, що повідомляє користувача чи отримуватиме він стипендію при умові, що він був зарахований

Текст скрипта:

CREATE OR REPLACE PROCEDURE is\_scholarship(IN abit\_id INTEGER)

AS $$

    DECLARE

        enrol INTEGER := get\_id\_enrolled\_abit(abit\_id);

        schollarship BOOLEAN;

    BEGIN

        IF (enrol IS NOT NULL) THEN

            SELECT enrolled\_abiturient.scholarship

            FROM enrolled\_abiturient

            WHERE enrolled\_abiturient.id\_statement = enrol

            INTO schollarship;

            IF (schollarship) THEN

                RAISE NOTICE 'Ви отримуватимете стипендію';

            ELSE

                RAISE NOTICE 'Ви НЕ отримуватимете стипендію';

            END IF;

        ELSE

            RAISE NOTICE 'Вас ще не було зараховано';

        END IF;

    END;

$$ LANGUAGE plpgsql;

Результат:

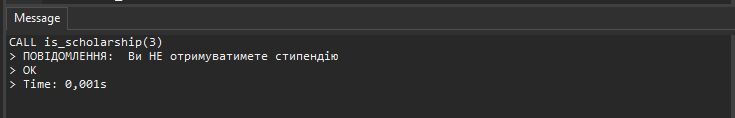


Рисунок 5.7 – Результат виконання скрипта

* Функція, що повертає інформацію про всі подані заяви абітурієнтом

Текст скрипта:

CREATE OR REPLACE FUNCTION get\_statements(abit\_id INTEGER)

RETURNS TABLE(

    specialty INTEGER,

    prioritet SMALLINT,

    score "numeric"(6, 3),

    course SMALLINT)

AS $$

    BEGIN

        RETURN query

            SELECT statements.id\_specialty, statements.priorities, statements.score, statements.course

            FROM statements

            WHERE statements.id\_abiturient = abit\_id

            ORDER BY statements.priorities;

    END;

$$ LANGUAGE plpgsql;

Результат:

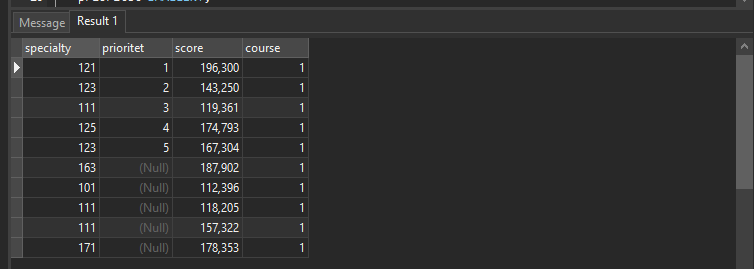


Рисунок 5.8 – Результат виконання скрипта

* + - 1. Процедури та функції члена приймальної комісії
* Процедура яка зараховує абітурієнта по заяві, якщо його не було зараховано до цього

Текст скрипта:

CREATE OR REPLACE PROCEDURE enroll\_abit(IN statement\_id INTEGER, IN schoolarship BOOLEAN DEFAULT(FALSE)) AS $$

BEGIN

        IF EXISTS(SELECT \*

                  FROM enrolled\_abiturient

                  WHERE enrolled\_abiturient.id\_statement = statement\_id)

        THEN

            RAISE NOTICE 'Абітурієнта за даною заявою вже було зараховано';

        ELSE

            IF NOT EXISTS(SELECT \*

                          FROM enrolled\_abiturient, statements

                          WHERE enrolled\_abiturient.id\_statement = statements."id" AND

                                statements.id\_abiturient = (SELECT statements.id\_abiturient

                                                            FROM statements

                                                            WHERE statements."id" = statement\_id))

            THEN

                INSERT INTO enrolled\_abiturient (id\_statement, scholarship) VALUES

                (statement\_id, schoolarship);

                RAISE NOTICE 'Задану заяву було успішно зараховано';

            ELSE

                RAISE NOTICE 'Заданого абітурієнта вже було зараховано за іншою заявою';

            END IF;

        END IF;

    END;

$$ LANGUAGE plpgsql;

Результат:

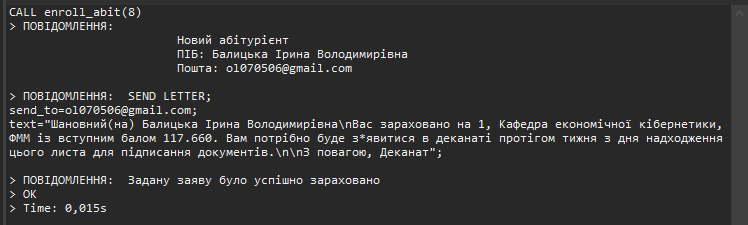


Рисунок 5.9 – Результат виконання скрипта

* + - 1. Процедури та функції декана
* Процедура, що дає можливість сформувати наказ про зарахування абітурієнтів на певний факультет

Текст скрипта:

CREATE OR REPLACE PROCEDURE formation\_of\_order(department\_id INTEGER) AS $$

    DECLARE

        abit\_name VARCHAR;

        abit\_surname VARCHAR;

        abit\_fathername VARCHAR;

        res TEXT := E'\n';

    BEGIN

        FOR abit\_surname, abit\_name, abit\_fathername IN

        SELECT abiturient.first\_name, abiturient.last\_name, abiturient.fathers\_name

        FROM enrolled\_abiturient, statements, abiturient, department\_priority, department

        WHERE enrolled\_abiturient.id\_statement = statements."id" AND

              statements.id\_abiturient = abiturient."id" AND

              enrolled\_abiturient.id\_statement = department\_priority."id" AND

              department\_priority.id\_department = department."id" AND

              abiturient.is\_conditions AND

              department\_priority.id\_department = department\_id

        ORDER BY abiturient.first\_name, abiturient.fathers\_name

        LOOP

            res := concat(res, abit\_surname, ' ', abit\_name, ' ', abit\_fathername, E'\n');

        END LOOP;

        RAISE NOTICE 'Зарахувати на % наступних абітурієнтів:%', department.department\_name FROM department WHERE department."id" = department\_id, left(res, -1);

    END;

$$ LANGUAGE plpgsql;

Результат:

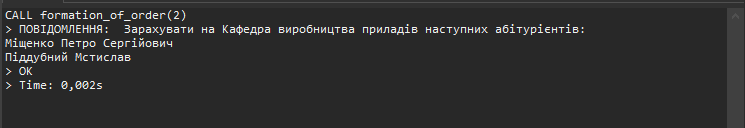


Рисунок 5.10 – Результат виконання скрипта

* + - 1. Інші процедури та функції
* Функція, що повертає ІД заяви, по якій було зараховано абітурієнта або NULL

Текст скрипта:

CREATE OR REPLACE FUNCTION get\_id\_enrolled\_abit(abit\_id INTEGER)

RETURNS INTEGER

AS $$

    DECLARE

        enrol INTEGER;

    BEGIN

        SELECT enrolled\_abiturient.id\_statement

        FROM enrolled\_abiturient

        WHERE enrolled\_abiturient.id\_statement IN (

            SELECT statements."id"

            FROM statements

            WHERE statements.id\_abiturient = abit\_id)

        INTO enrol;

        RETURN enrol;

    END;

$$ LANGUAGE plpgsql;

Результат:

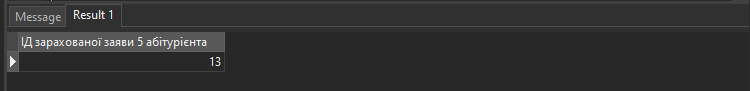


Рисунок 5.11 – Результат виконання скрипта

* + 1. Представлення
* Представлення короткої інформації про подані заяви

Текст скрипта:

CREATE OR REPLACE VIEW statements\_info AS

    SELECT statements."id", abiturient.first\_name, abiturient.last\_name, abiturient.certificate\_score, specialty.specialty\_name, statements.priorities, statements.score, statements.course

    FROM statements, abiturient, specialty

    WHERE statements.id\_abiturient = abiturient."id" AND

          statements.id\_specialty = specialty."id"

    ORDER BY statements.course, statements.priorities, statements.score, abiturient.certificate\_score;

Результат:

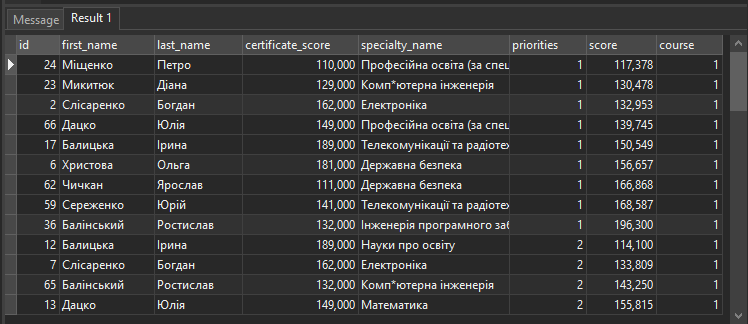


Рисунок 5.12 – Результат виконання скрипта

* Представлення повної інформації про всі подані заяви

Текст скрипта:

CREATE OR REPLACE VIEW statements\_full\_info AS

    SELECT department\_priority."id", statements\_info.first\_name, statements\_info.last\_name, faculty.short\_name, department.department\_name, statements\_info.specialty\_name, statements\_info.certificate\_score, statements\_info.priorities, statements\_info.score, statements\_info.course, department\_priority.priorities AS department\_priority

    FROM department\_priority, department, faculty, statements\_info

    WHERE department\_priority.id\_department = department."id" AND

          department.id\_faculty = faculty."id" AND

          department\_priority.id\_statement = statements\_info."id"

    ORDER BY statements\_info.first\_name, statements\_info.last\_name, statements\_info.priorities, department\_priority, department\_priority."id";

Результат:

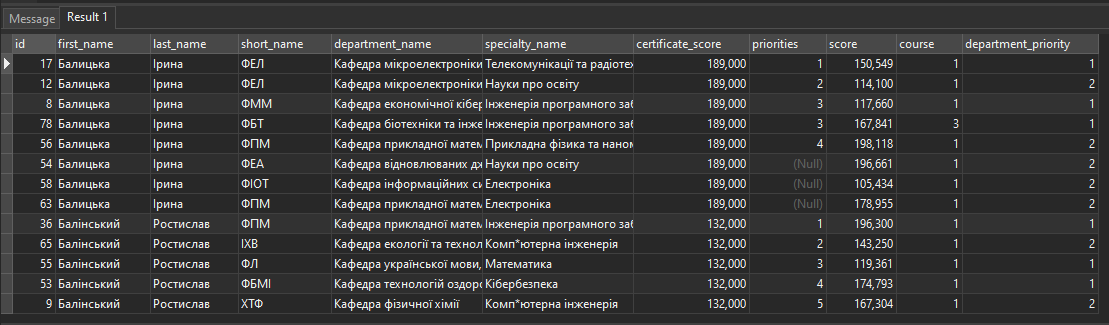


Рисунок 5.13 – Результат виконання скрипта

* + 1. DML-запити
* Мінімальний бал по предметам

Текст скрипта:

SELECT zno\_result.id\_subject, MIN(zno\_result."result")

FROM zno\_result

GROUP BY zno\_result.id\_subject

ORDER BY zno\_result.id\_subject;

Результат:

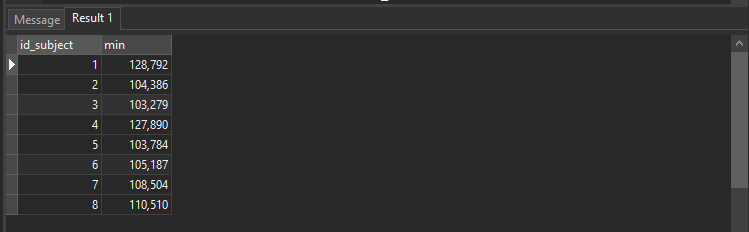


Рисунок 5.14 – Результат виконання скрипта

* Максимальний бал по предметам

Текст скрипта:

SELECT zno\_result.id\_subject, MAX(zno\_result."result")

FROM zno\_result

GROUP BY zno\_result.id\_subject

ORDER BY zno\_result.id\_subject;

Результат:

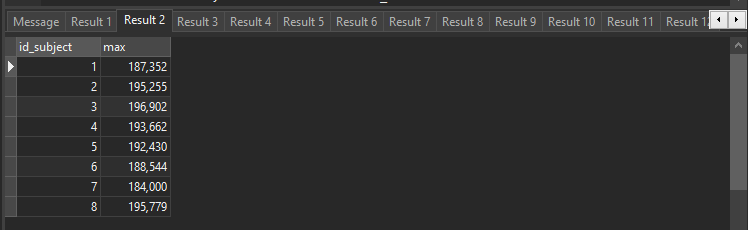


Рисунок 5.15 – Результат виконання скрипта

* Середній бал по предметам

Текст скрипта:

SELECT zno\_result.id\_subject, AVG(zno\_result."result")

FROM zno\_result

GROUP BY zno\_result.id\_subject

ORDER BY zno\_result.id\_subject;

Результат:

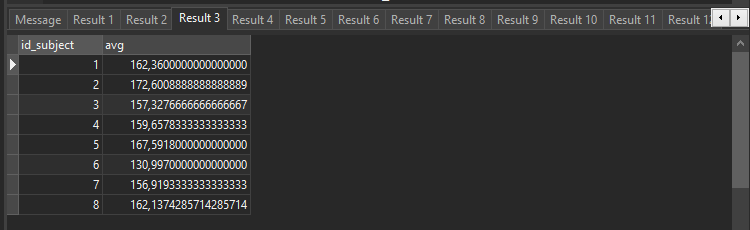


Рисунок 5.16 – Результат виконання скрипта

* Витягнути всі паспорта

Текст скрипта:

SELECT \*

FROM "document"

WHERE "document".short\_description = 'Паспорт';

Результат:

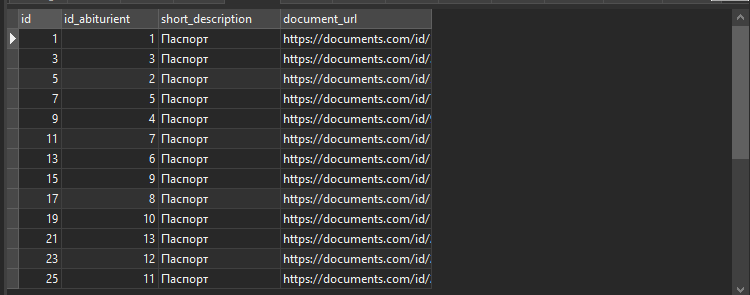


Рисунок 5.17 – Результат виконання скрипта

* Всі кафедри та відповідні їм факультети

Текст скрипта:

SELECT faculty.full\_name, faculty.short\_name, department.department\_name

FROM department, faculty

WHERE department.id\_faculty = faculty."id"

ORDER BY faculty.full\_name;

Результат:

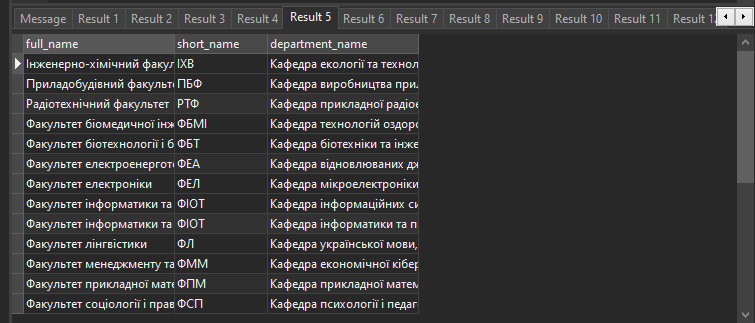


Рисунок 5.18 – Результат виконання скрипта

* Всі результати атестатів по абітурієнтам

Текст скрипта:

SELECT abiturient."id", abiturient.certificate\_score

FROM abiturient

ORDER BY abiturient."id";

Результат:

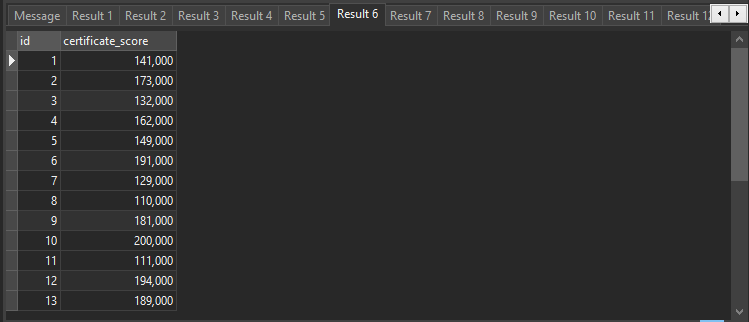


Рисунок 5.19 – Результат виконання скрипта

* Кількість поданих заяв на різні курси

Текст скрипта:

SELECT statements.course, COUNT(statements.course)

FROM statements

GROUP BY statements.course;

Результат:

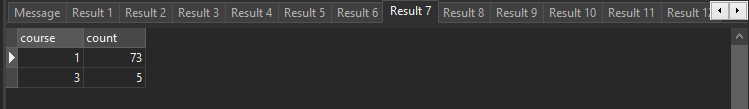


Рисунок 5.20 – Результат виконання скрипта

* "Популярність" спеціальностей по поданим заявам

Текст скрипта:

SELECT statements.id\_specialty, COUNT(statements.id\_specialty)

FROM statements

GROUP BY statements.id\_specialty

ORDER BY count DESC, statements.id\_specialty;

Результат:

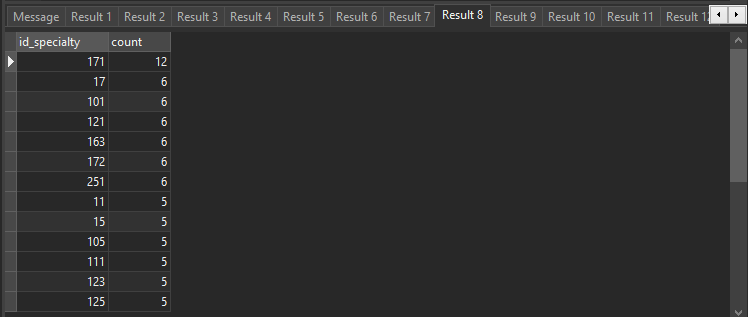


Рисунок 5.21 – Результат виконання скрипта

* ПІБ абітурієнта, предмет та результат ЗНО по цьому предмету

Текст скрипта:

SELECT ROW\_NUMBER() over(ORDER BY(abiturient."id")), concat(abiturient.first\_name, ' ', abiturient.last\_name, ' ', abiturient.fathers\_name) AS "ПІБ", zno\_list.subject\_name, zno\_result."result"

FROM zno\_result, abiturient, zno\_list

WHERE zno\_result.id\_abiturient = abiturient."id" AND

      zno\_result.id\_subject = zno\_list."id";

Результат:

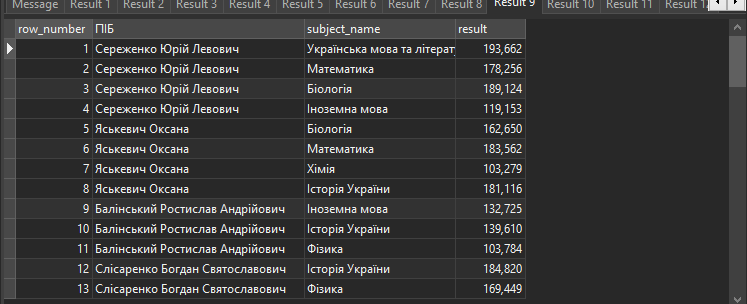


Рисунок 5.22 – Результат виконання скрипта

* Вибірка всіх документів по студентам

Текст скрипта:

SELECT "document"."id", concat(abiturient.first\_name, ' ', abiturient.last\_name, ' ', abiturient.fathers\_name), "document".short\_description, "document".document\_url

FROM "document", abiturient

WHERE "document".id\_abiturient = abiturient."id";

Результат:

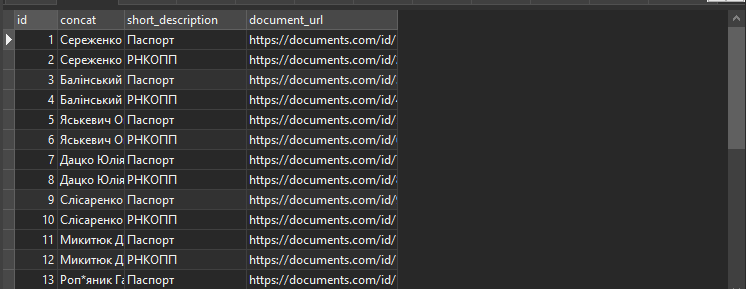


Рисунок 5.23 – Результат виконання скрипта

* Вибірка всіх даних для зворотнього зв'язку із студентами

Текст скрипта:

SELECT abiturient.first\_name, abiturient.last\_name, abiturient.fathers\_name, abiturient.email, '+' || abiturient.phone\_number AS "phone\_number"

FROM statements, abiturient

WHERE statements.id\_abiturient = abiturient."id";

Результат:

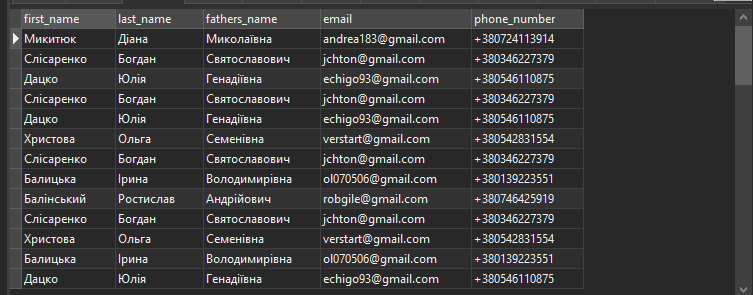


Рисунок 5.24 – Результат виконання скрипта

* "Популярність" кафедр серед поданих заяв

Текст скрипта:

FROM department\_priority, department, statements

WHERE department\_priority.id\_statement = statements."id" AND

      department."id" = department\_priority.id\_department

GROUP BY department.department\_name

ORDER BY count DESC;

Результат:

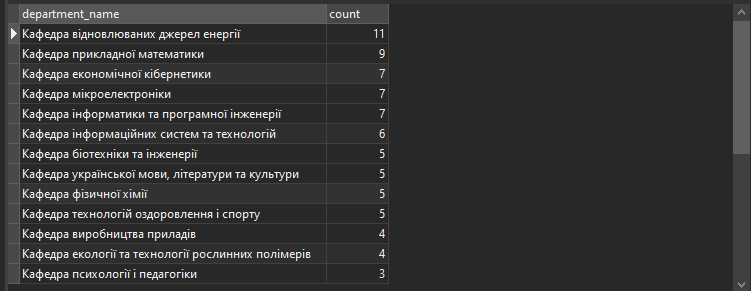


Рисунок 5.25 – Результат виконання скрипта

* "Популярність" факультетів

Текст скрипта:

SELECT faculty.full\_name, COUNT(faculty.full\_name)

FROM department\_priority, department, statements, faculty

WHERE department\_priority.id\_statement = statements."id" AND

      department."id" = department\_priority.id\_department AND

      faculty."id" = department.id\_faculty

GROUP BY faculty.full\_name

ORDER BY count DESC;

Результат:

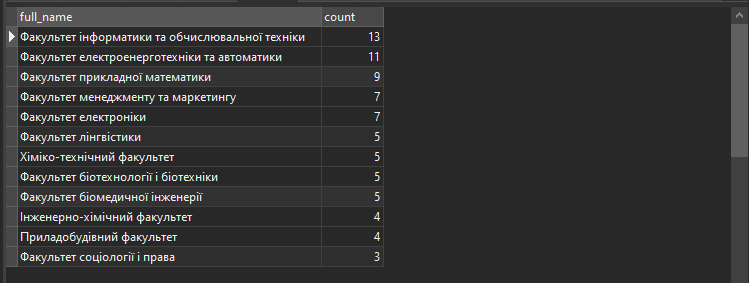


Рисунок 5.26 – Результат виконання скрипта

* Кількість людей, які склали даний предмет на прохідний бал

Текст скрипта:

SELECT zno\_list.subject\_name, COUNT(zno\_list.subject\_name)

FROM zno\_result, zno\_list

WHERE zno\_list."id" = zno\_result.id\_subject AND

      zno\_result."result" > zno\_list.min\_score

GROUP BY zno\_list.subject\_name;

Результат:

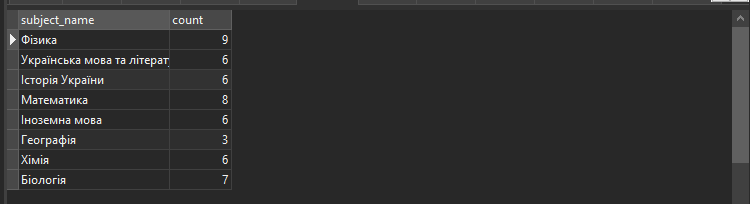


Рисунок 5.27 – Результат виконання скрипта

* "Популярність" предметів ЗНО

Текст скрипта:

SELECT zno\_list.subject\_name, COUNT(zno\_list.subject\_name)

FROM zno\_result, zno\_list

WHERE zno\_list."id" = zno\_result.id\_subject

GROUP BY zno\_list.subject\_name

ORDER BY count DESC;

Результат:

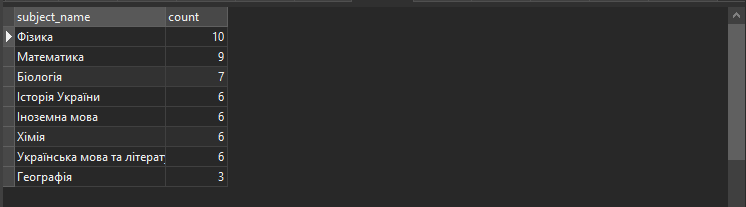


Рисунок 5.28 – Результат виконання скрипта

* Рейтинг студентів які отримують стипендію

Текст скрипта:

SELECT concat(abiturient.first\_name, ' ', abiturient.last\_name, ' ', abiturient.fathers\_name), statements.score

FROM enrolled\_abiturient, statements, abiturient

WHERE enrolled\_abiturient.id\_statement = statements."id" AND

      abiturient."id" = statements.id\_abiturient AND

      enrolled\_abiturient.scholarship

ORDER BY statements.score DESC;

Результат:

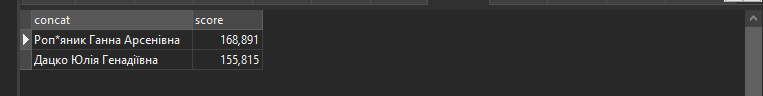


Рисунок 5.29 – Результат виконання скрипта

* Предмет і кількість людей, які "завалили" його

Текст скрипта:

SELECT zno\_list.subject\_name, COUNT(zno\_list.subject\_name)

FROM zno\_result, zno\_list

WHERE zno\_list."id" = zno\_result.id\_subject AND

      NOT zno\_result."result" > zno\_list.min\_score

GROUP BY zno\_list.subject\_name;

Результат:

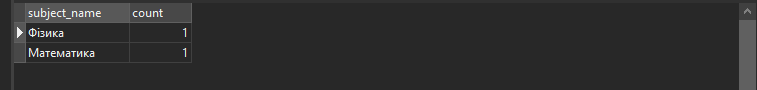


Рисунок 5.30 – Результат виконання скрипта

* Інформація про зарахованих абітурієнтів

Текст скрипта:

SELECT statements."id", concat(abiturient.first\_name, ' ', abiturient.last\_name, ' ', abiturient.fathers\_name) AS "ПІБ", abiturient.certificate\_score, specialty.specialty\_name, statements.priorities, statements.score, statements.course

FROM statements, abiturient, specialty, enrolled\_abiturient

WHERE statements.id\_abiturient = abiturient."id" AND

      statements.id\_specialty = specialty."id" AND

      enrolled\_abiturient.id\_statement = statements."id"

ORDER BY statements.course, statements.priorities, statements.score, abiturient.certificate\_score;

Результат:

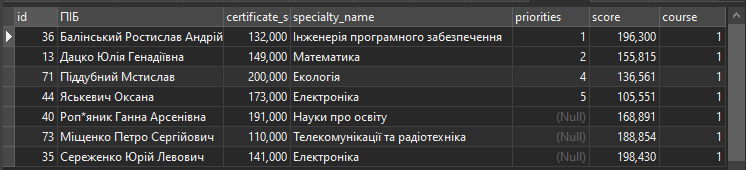


Рисунок 5.31 – Результат виконання скрипта

* "Популярність" кафедр серед зарахованих абітурієнтів

Текст скрипта:

SELECT specialty."id", specialty.specialty\_name, COUNT(specialty.specialty\_name)

FROM enrolled\_abiturient, statements, abiturient, specialty

WHERE enrolled\_abiturient.id\_statement = statements."id" AND

            statements.id\_abiturient = abiturient."id" AND

            statements.id\_specialty = specialty."id"

GROUP BY specialty."id", specialty.specialty\_name

ORDER BY count DESC;

Результат:

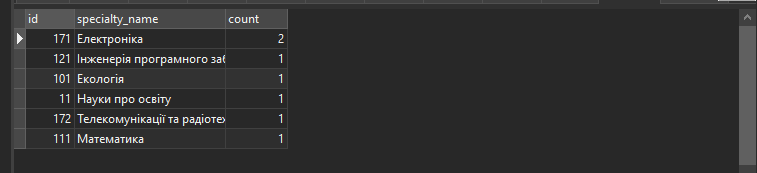


Рисунок 5.32 – Результат виконання скрипта

* Абітурієнт та відповідний йому середній бал ЗНО та середній бал атестату

Текст скрипта:

SELECT concat(abiturient.first\_name, ' ', abiturient.last\_name, ' ', abiturient.fathers\_name) AS "ПІБ", abit.certificate\_score, abit.zno\_res

FROM (SELECT abiturient."id", abiturient.certificate\_score, AVG(zno\_result."result") AS zno\_res

      FROM abiturient, zno\_list, zno\_result

      WHERE zno\_list."id" = zno\_result.id\_subject AND

            zno\_result.id\_abiturient = abiturient."id"

      GROUP BY abiturient."id", abiturient.certificate\_score) AS abit, abiturient

WHERE abit."id" = abiturient."id";

Результат:

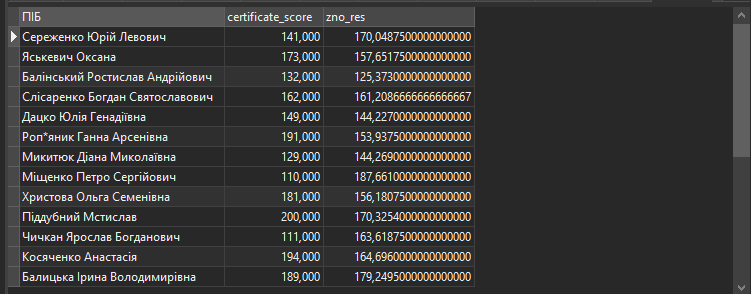


Рисунок 5.33 – Результат виконання скрипта

* + 1. Генератори
* Генератор реалізовано через автоінкремент в сутності «Зараховані абітурієнти»

Тип змінної SERIAL в СУБД PostgreSQL – є типом INTEGER з вбудованим генератором у вигляді автоінкременту

Результат:

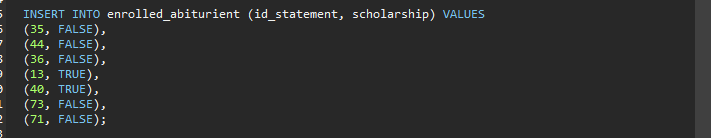


Рисунок 5.30 – Приклад вхідних даних для випробування генератора

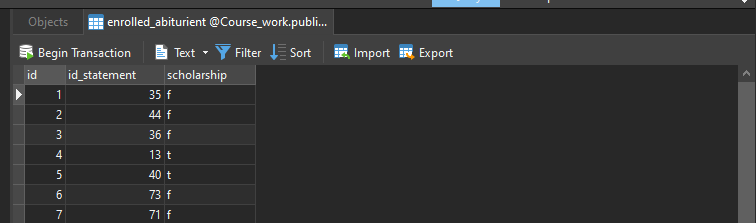


Рисунок 5.34 – Результат виконання скрипта

* + 1. Індекси

Спочатку зафіксуємо дані до створення індексу та оптимізації за допомогою запиту

Текст запиту:

SELECT \* FROM statements WHERE statements.score > 150;

explain (costs off) SELECT \* FROM statements WHERE statements.score > 150;

Результат:

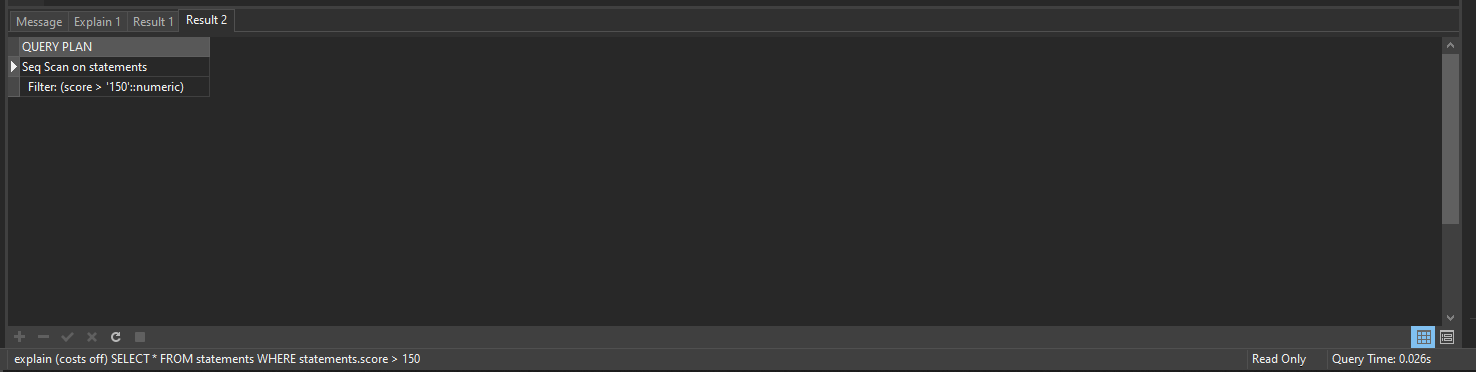


Рисунок 5.35 – Результат виконання скрипта до оптимізації

За допомогою вбудованого методу індексу – hash оптимізуємо даний запит

Текст скрипта:

CREATE INDEX ON zno\_result USING hash (id\_subject);

SELECT \* FROM statements WHERE statements.score > 150;

explain (costs off) SELECT \* FROM statements WHERE statements.score > 150;

Результат:

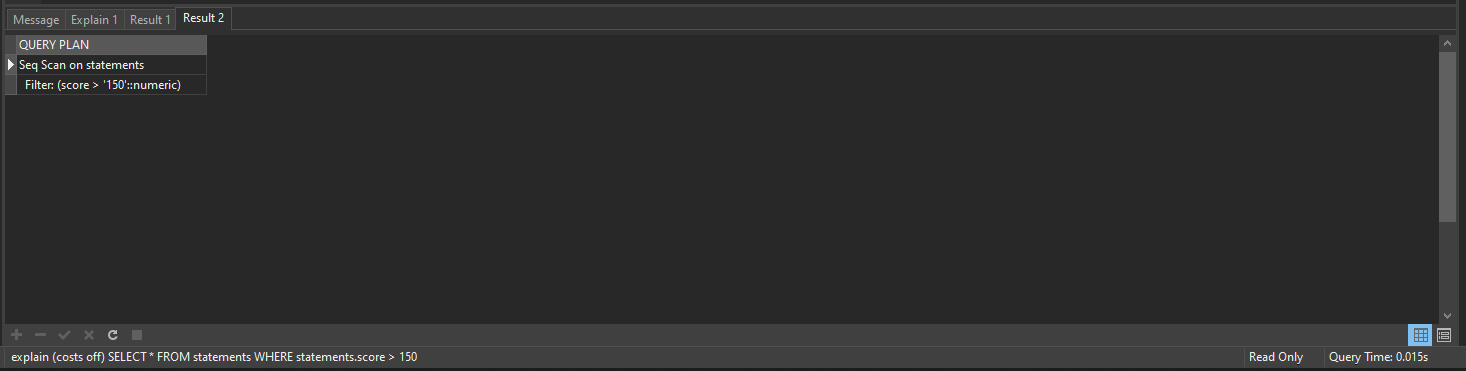


Рисунок 5.36 – Результат виконання скрипта після оптимізації

На Рисунку 3.6 ми можемо побачити, що час виконання запиту зменшився з 0.026 секунди до 0.015 секунди, що є досить суттєвим результатом.

# ВИСНОВКИ

В ході даної курсової роботи було розроблено та реалізовано базу даних для приймальної комісії Університету. Було проаналізовано предметну область і на її основі створено опис предметного середовища. Також було визначено технічне завдання з огляду на опис предметного середовища. Було спроектовано та створено ER-діаграму для БД, де було чітко визначено перелік сутностей та їх атрибутів. Наступним кроком було реалізовано даталогічну модель для системи управління базами даних – PosgreSQL. Після цього було створено всі таблиці, обмеження та зв’язки між ними. Далі всі таблиці були заповнені даними. Після цього було створено та представлено всі SQL-запити відібрані в попередніх кроках та користуючись методичними вказівками до курсової роботи.

Загалом було отримати базу даних для приймальної комісії Університету та протестовано її на реальних даних з перевіркою на виконання усіх бізнес-правил, створених у ході написання БД.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Chapter 11. Indexes. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/current/indexes.html> (date of access: 08.01.2023).
2. CREATE FUNCTION. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/current/sql-createfunction.html> (date of access: 08.01.2023).
3. CREATE PROCEDURE. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/current/sql-createprocedure.html> (date of access: 08.01.2023).
4. CREATE TRIGGER. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/current/sql-createtrigger.html> (date of access: 08.01.2023).
5. Database Users and Privileges. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/8.0/user-manag.html> (date of access: 08.01.2023).
6. PostgreSQL: Downloads. *PostgreSQL: The world's most advanced open source database*. URL: <https://www.postgresql.org/download/> (date of access: 08.01.2023).

# ДОДАТОК А. ТЕКСТИ INSERT ЗАПИТІВ



*студента групи ІП-13 ІІ курсу*

*Замкового Дмитра Володимировича*

(Обсяг програми (документа), арк., Кб)

*12 Кб, 8 аркушів*

(Вид носія даних)

*Онлайн-репозиторій*

(Найменування програми (документа))

*Тексти INSERT запитів для заповнення таблиць бази даних приймальної комісії*

INSERT INTO zno\_list VALUES

(1, 'Історія України', 100),

(2, 'Математика', 125),

(3, 'Хімія', 100),

(4, 'Українська мова та література', 125),

(5, 'Фізика', 125),

(6, 'Іноземна мова', 100),

(7, 'Географія', 100),

(8, 'Біологія', 100);

INSERT INTO abiturient VALUES

(1, 'Сереженко', 'Юрій', 'Левович', '2005-03-07', 380454854924, 'jaderfreitag@gmail.com', 141, NULL, TRUE, FALSE),

(2, 'Яськевич', 'Оксана', NULL, '2006-08-21', 380145181256, 'onegoodlovingman63@gmail.com', 173, 2, TRUE, TRUE),

(3, 'Балінський', 'Ростислав', 'Андрійович', '2005-11-22', 380746425919, 'robgile@gmail.com', 132, NULL, TRUE, TRUE),

(4, 'Слісаренко', 'Богдан', 'Святославович', '2005-07-19', 380346227379, 'jchton@gmail.com', 162, NULL, TRUE, TRUE),

(5, 'Дацко', 'Юлія', 'Генадіївна', '2006-12-13', 380546110875, 'echigo93@gmail.com', 149, NULL, TRUE, FALSE),

(6, 'Роп\*яник', 'Ганна', 'Арсенівна', '2005-03-19', 380342266864, 'jlangman@gmail.com', 191, NULL, TRUE, FALSE),

(7, 'Микитюк', 'Діана', 'Миколаївна', '2005-07-13', 380724113914, 'andrea183@gmail.com', 129, 1, TRUE, TRUE),

(8, 'Міщенко', 'Петро', 'Сергійович', '2005-09-25', 380758423657, 'andreabicchi@gmail.com', 110, NULL, TRUE, TRUE),

(9, 'Христова', 'Ольга', 'Семенівна', '2006-04-24', 380542831554, 'verstart@gmail.com', 181, 3, TRUE, FALSE),

(10, 'Піддубний', 'Мстислав', NULL, '2005-10-31', 380656678433, 'geforty@gmail.com', 200, NULL, TRUE, FALSE),

(11, 'Чичкан', 'Ярослав', 'Богданович', '2005-11-29', 380753132838, 'dtmorpheus@gmail.com', 111, 1, TRUE, TRUE),

(12, 'Косяченко', 'Анастасія', NULL, '2006-06-25', 380115370787, 'lektta@gmail.com', 194, NULL, TRUE, TRUE),

(13, 'Балицька', 'Ірина', 'Володимирівна', '2006-11-25', 380139223551, 'ol070506@gmail.com', 189, NULL, TRUE, FALSE);

INSERT INTO specialty VALUES

(11, 'Науки про освіту'),

(15, 'Професійна освіта (за спеціалізаціями)'),

(17, 'Фізична культура і спорт'),

(101, 'Екологія'),

(105, 'Прикладна фізика та наноматеріали'),

(111, 'Математика'),

(121, 'Інженерія програмного забезпечення'),

(123, 'Комп\*ютерна інженерія'),

(125, 'Кібербезпека'),

(163, 'Біомедична інженерія'),

(171, 'Електроніка'),

(172, 'Телекомунікації та радіотехніка'),

(251, 'Державна безпека');

INSERT INTO faculty VALUES

(1, 'Інженерно-хімічний факультет', 'ІХВ', 380625231972),

(2, 'Приладобудівний факультет', 'ПБФ', 380650176399),

(3, 'Радіотехнічний факультет', 'РТФ', 380435377110),

(5, 'Факультет біомедичної інженерії', 'ФБМІ', 380509303750),

(7, 'Факультет біотехнології і біотехніки', 'ФБТ', 380371095274),

(9, 'Факультет електроенерготехніки та автоматики', 'ФЕА', 380271350556),

(12, 'Факультет електроніки', 'ФЕЛ', 380472094905),

(17, 'Факультет інформатики та обчислювальної техніки', 'ФІОТ', 380176940857),

(20, 'Факультет лінгвістики', 'ФЛ', 380120440525),

(21, 'Факультет менеджменту та маркетингу', 'ФММ', 380174521408),

(22, 'Факультет соціології і права', 'ФСП', 380573684871),

(23, 'Факультет прикладної математики', 'ФПМ', 380747108181),

(25, 'Хіміко-технічний факультет', 'ХТФ', 380821985285);

INSERT INTO zno\_result VALUES

(1, 1, 6, 119.153),

(2, 9, 2, 104.386),

(3, 3, 6, 132.725),

(4, 10, 2, 189.392),

(5, 1, 4, 193.662),

(6, 13, 6, 188.544),

(7, 1, 8, 189.124),

(8, 3, 5, 103.784),

(9, 6, 1, 152.470),

(10, 10, 5, 179.970),

(11, 11, 5, 182.650),

(12, 12, 5, 178.598),

(13, 2, 2, 183.562),

(14, 10, 4, 127.890),

(15, 6, 8, 141.520),

(16, 13, 5, 131.987),

(17, 11, 3, 184.569),

(18, 6, 3, 125.981),

(19, 10, 4, 178.256),

(20, 13, 7, 184.000),

(21, 7, 8, 148.120),

(22, 9, 1, 187.352),

(23, 1, 2, 178.256),

(24, 5, 3, 142.351),

(25, 13, 5, 187.652),

(26, 12, 7, 178.254),

(27, 4, 8, 187.259),

(28, 4, 1, 184.820),

(29, 8, 2, 187.661),

(30, 4, 6, 128.807),

(31, 4, 4, 188.413),

(32, 6, 8, 195.779),

(33, 9, 2, 159.277),

(34, 13, 3, 190.884),

(35, 9, 5, 173.708),

(36, 3, 1, 139.610),

(37, 4, 7, 108.504),

(38, 2, 8, 162.650),

(39, 7, 2, 179.500),

(40, 5, 1, 128.792),

(41, 10, 2, 176.119),

(42, 12, 3, 196.902),

(43, 7, 6, 105.187),

(44, 5, 8, 110.510),

(45, 11, 5, 175.690),

(46, 12, 4, 129.156),

(47, 4, 5, 169.449),

(48, 11, 6, 111.566),

(49, 5, 2, 195.255),

(50, 2, 3, 103.279),

(51, 2, 1, 181.116),

(52, 12, 4, 140.570),

(53, 13, 5, 192.430);

INSERT INTO document VALUES

(1, 1, 'Паспорт', 'https://documents.com/id/1'),

(2, 1, 'РНКОПП', 'https://documents.com/id/2'),

(3, 3, 'Паспорт', 'https://documents.com/id/3'),

(4, 3, 'РНКОПП', 'https://documents.com/id/4'),

(5, 2, 'Паспорт', 'https://documents.com/id/5'),

(6, 2, 'РНКОПП', 'https://documents.com/id/6'),

(7, 5, 'Паспорт', 'https://documents.com/id/7'),

(8, 5, 'РНКОПП', 'https://documents.com/id/8'),

(9, 4, 'Паспорт', 'https://documents.com/id/9'),

(10, 4, 'РНКОПП', 'https://documents.com/id/10'),

(11, 7, 'Паспорт', 'https://documents.com/id/11'),

(12, 7, 'РНКОПП', 'https://documents.com/id/12'),

(13, 6, 'Паспорт', 'https://documents.com/id/13'),

(14, 6, 'РНКОПП', 'https://documents.com/id/14'),

(15, 9, 'Паспорт', 'https://documents.com/id/15'),

(16, 9, 'РНКОПП', 'https://documents.com/id/16'),

(17, 8, 'Паспорт', 'https://documents.com/id/17'),

(18, 8, 'РНКОПП', 'https://documents.com/id/18'),

(19, 10, 'Паспорт', 'https://documents.com/id/19'),

(20, 10, 'РНКОПП', 'https://documents.com/id/20'),

(21, 13, 'Паспорт', 'https://documents.com/id/21'),

(22, 13, 'РНКОПП', 'https://documents.com/id/22'),

(23, 12, 'Паспорт', 'https://documents.com/id/23'),

(24, 12, 'РНКОПП', 'https://documents.com/id/24'),

(25, 11, 'Паспорт', 'https://documents.com/id/25'),

(26, 11, 'РНКОПП', 'https://documents.com/id/26');

INSERT INTO statements VALUES

(1, 7, 101, 4, 176.792, 1),

(2, 4, 171, 1, 132.953, 1),

(3, 5, 172, 4, 126.381, 3),

(4, 4, 17, NULL, 162.410, 1),

(5, 5, 163, NULL, 120.910, 1),

(6, 9, 251, 1, 156.657, 1),

(7, 4, 171, 2, 133.809, 1),

(8, 13, 121, 3, 117.660, 1),

(9, 3, 123, 5, 167.304, 1),

(10, 4, 15, 4, 194.332, 1),

(11, 9, 125, NULL, 177.350, 1),

(12, 13, 11, 2, 114.100, 1),

(13, 5, 111, 2, 155.815, 1),

(14, 6, 105, 5, 189.945, 1),

(15, 5, 101, 5, 154.585, 1),

(16, 9, 171, 2, 163.981, 1),

(17, 13, 172, 1, 150.549, 1),

(18, 1, 17, 5, 120.872, 1),

(19, 4, 163, 5, 162.246, 1),

(20, 12, 251, NULL, 129.716, 1),

(21, 9, 171, 3, 124.850, 1),

(22, 8, 121, NULL, 166.220, 3),

(23, 7, 123, 1, 130.478, 1),

(24, 8, 15, 1, 117.378, 1),

(25, 5, 125, NULL, 195.232, 1),

(26, 7, 11, NULL, 113.596, 1),

(27, 5, 111, NULL, 122.334, 1),

(28, 7, 105, 5, 110.403, 1),

(29, 9, 101, NULL, 133.579, 1),

(30, 1, 171, 2, 199.513, 1),

(31, 6, 172, 2, 180.766, 1),

(32, 9, 17, NULL, 151.925, 1),

(33, 11, 163, 2, 165.340, 1),

(34, 1, 251, 3, 184.816, 1),

(35, 1, 171, NULL, 198.430, 1),

(36, 3, 121, 1, 196.300, 1),

(37, 2, 123, 3, 124.204, 1),

(38, 7, 15, NULL, 121.597, 1),

(39, 6, 125, NULL, 141.341, 1),

(40, 6, 11, NULL, 168.891, 1),

(41, 3, 111, NULL, 118.205, 1),

(42, 9, 105, NULL, 119.255, 1),

(43, 10, 101, NULL, 149.800, 1),

(44, 2, 171, 5, 105.551, 1),

(45, 7, 172, NULL, 134.833, 1),

(46, 12, 17, 5, 160.128, 1),

(47, 3, 163, NULL, 187.902, 1),

(48, 2, 251, NULL, 174.485, 3),

(49, 5, 171, NULL, 187.886, 1),

(50, 12, 121, 3, 108.804, 1),

(51, 5, 123, NULL, 133.369, 1),

(52, 2, 15, 2, 194.396, 1),

(53, 3, 125, 4, 174.793, 1),

(54, 13, 11, NULL, 196.661, 1),

(55, 3, 111, 3, 119.361, 1),

(56, 13, 105, 4, 198.118, 1),

(57, 3, 101, NULL, 112.396, 1),

(58, 13, 171, NULL, 105.434, 1),

(59, 1, 172, 1, 168.587, 1),

(60, 10, 17, NULL, 108.134, 1),

(61, 11, 163, NULL, 159.441, 1),

(62, 11, 251, 1, 166.868, 1),

(63, 13, 171, NULL, 178.955, 1),

(64, 8, 121, NULL, 126.682, 1),

(65, 3, 123, 2, 143.250, 1),

(66, 5, 15, 1, 139.745, 1),

(67, 9, 125, NULL, 161.413, 1),

(68, 10, 11, 2, 151.350, 3),

(69, 3, 111, NULL, 157.322, 1),

(70, 9, 105, 4, 161.849, 1),

(71, 10, 101, 4, 136.561, 1),

(72, 3, 171, NULL, 178.353, 1),

(73, 8, 172, NULL, 188.854, 1),

(74, 6, 17, NULL, 158.662, 1),

(75, 9, 163, NULL, 123.631, 1),

(76, 1, 251, NULL, 141.913, 1),

(77, 9, 171, NULL, 195.407, 1),

(78, 13, 121, 3, 167.841, 3);

INSERT INTO department VALUES

(1, 1, 'Кафедра екології та технології рослинних полімерів', 380953256787),

(2, 2, 'Кафедра виробництва приладів', 380548523698),

(3, 3, 'Кафедра прикладної радіоелектроніки', 380495324862),

(4, 5, 'Кафедра технологій оздоровлення і спорту', 380874523791),

(5, 7, 'Кафедра біотехніки та інженерії', 380596542579),

(6, 9, 'Кафедра відновлюваних джерел енергії', 380445954843),

(7, 12, 'Кафедра мікроелектроніки', 380529674532),

(8, 17, 'Кафедра інформатики та програмної інженерії', 380126744598),

(9, 17, 'Кафедра інформаційних систем та технологій', 380286845215),

(10, 20, 'Кафедра української мови, літератури та культури', 380859476358),

(11, 21, 'Кафедра економічної кібернетики', 380558468265),

(12, 22, 'Кафедра психології і педагогіки', 380462155915),

(13, 23, 'Кафедра прикладної математики', 380123485923),

(14, 25, 'Кафедра фізичної хімії', 380589462185);

INSERT INTO enrolled\_abiturient (id\_statement, scholarship) VALUES

(35, FALSE),

(44, FALSE),

(36, FALSE),

(13, TRUE),

(40, TRUE),

(73, FALSE),

(71, FALSE);

INSERT INTO department\_priority VALUES

(1, 10, 1, 1),

(2, 11, 2, 2),

(3, 2, 3, 3),

(4, 6, 4, 1),

(5, 8, 5, 2),

(6, 9, 6, 3),

(7, 5, 7, 2),

(8, 11, 8, 1),

(9, 14, 9, 2),

(10, 1, 10, 3),

(11, 4, 11, 1),

(12, 7, 12, 2),

(13, 6, 13, 3),

(14, 4, 14, 1),

(15, 6, 15, 2),

(16, 6, 16, 3),

(17, 7, 17, 1),

(18, 5, 18, 2),

(19, 13, 19, 3),

(20, 12, 20, 1),

(21, 10, 21, 2),

(22, 6, 22, 3),

(23, 4, 23, 2),

(24, 11, 24, 1),

(25, 6, 25, 2),

(26, 14, 26, 3),

(27, 8, 27, 2),

(28, 13, 28, 1),

(29, 13, 29, 2),

(30, 6, 30, 3),

(31, 7, 31, 2),

(32, 1, 32, 1),

(33, 7, 33, 2),

(34, 7, 34, 3),

(35, 8, 35, 2),

(36, 13, 36, 1),

(37, 9, 37, 2),

(38, 5, 38, 3),

(39, 13, 39, 2),

(40, 9, 40, 1),

(41, 6, 41, 2),

(42, 13, 42, 3),

(43, 10, 43, 2),

(44, 13, 44, 1),

(45, 9, 45, 2),

(46, 8, 46, 1),

(47, 11, 47, 3),

(48, 14, 48, 2),

(49, 12, 49, 1),

(50, 1, 50, 2),

(51, 8, 51, 3),

(52, 14, 52, 2),

(53, 4, 53, 1),

(54, 6, 54, 2),

(55, 10, 55, 1),

(56, 13, 56, 2),

(57, 5, 57, 3),

(58, 9, 58, 2),

(59, 10, 59, 3),

(60, 2, 60, 2),

(61, 14, 61, 1),

(62, 11, 62, 1),

(63, 13, 63, 2),

(64, 6, 64, 3),

(65, 1, 65, 2),

(66, 7, 66, 3),

(67, 11, 67, 2),

(68, 9, 68, 1),

(69, 8, 69, 2),

(70, 11, 70, 1),

(71, 2, 71, 2),

(72, 8, 72, 3),

(73, 2, 73, 2),

(74, 6, 74, 1),

(75, 7, 75, 2),

(76, 4, 76, 3),

(77, 12, 77, 2),

(78, 5, 78, 1);